UNIVERSIDAD DE CUENCA

FACULTAD DE INGENIERÍA

Escuela de Ingeniería Electrónica y Telecomunicaciones

"Diseño e implementación de un prototipo de supervisión de un sistema de control industrial utilizando plataformas empotradas de bajo costo y controladores lógicos programables PLCs."

Tesis previa a la obtención del
título de Ingeniero en Electrónica y Telecomunicaciones

Autores:
Edwin Patricio Patiño Ramón
Juan Carlos Solano Minchalo

Director:
Dr. Luis Ismael Minchala Ávila

Cuenca - Ecuador
2016

Resumen

Palabras clave: OPC, OPC Foundation, Raspberry Pi, Python.

El uso de una arquitectura de comunicación OPC en un proceso industrial, permite una mejor gestión de éste, incluyendo monitoreo y control. Sin embargo, muchas industrias en la actualidad ven limitado su acceso a esta tecnología por el costoso esquema de licenciamiento. Otra de las barreras es la imposición de protocolos definidos por OPC Clásico. Debido a esto la *OPC Foundation* ha desarrollado la nueva versión del protocolo OPC, denominado OPC UA. El presente trabajo presenta un prototipo de servidor OPC UA, implementado en un dispositivo embebido de bajo costo. Se ha escogido a un Raspeberry Pi debido a sus características de procesamiento y facilidad de conexión con otros dispositivos. El servidor se encuentra desarrollado en Python, permitiendo su ejecución en múltiples plataformas. En conclusión, el prototipo planteado ofrece interesantes ventajas en la industria, permitiendo una sencilla implementación de esta tecnología, con un costo muy bajo.

Abstract

Keywords: OPC servers, OPC Foundation, Raspberry Pi

Using OPC servers allow a better industrial process control. However, nowadays their high cost is a limitation for many industries. Another barrier is that the classic OPC protocols are still limited in communication capabilities. Because of this, OPC Foundation has developed the new OPC Protocol version named OPC UA. This research work presents the development of an OPC UA server implemented in a low cost embedded platform. The Raspberry Pi has been chosen because of its processing characteristics, and its capabilities of communication with other devices. The OPC server is implemented in Python, so it is multiplatform. The prototype developed in this work offers interesting advantages for industries, such as easiness in development in OPC standard communication with a low investment.

Índice general

Índice de figuras

Índice de tablas

Edwin Patricio Patiño Ramón, autor de la tesis "Diseño e implementación de un prototipo de supervisión de un sistema de control industrial utilizando plataformas empotradas de bajo costo y controladores lógicos programables PLCs", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor.

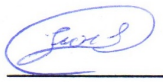
Cuenca, Abril 2016

Edwin Patricio Patiño Ramón

C.I: 0104953344

Juan Carlos Solano Minchalo, autor de la tesis "Diseño e implementación de un prototipo de supervisión de un sistema de control industrial utilizando plataformas empotradas de bajo costo y controladores lógicos programables PLCs", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor.

Cuenca, Abril 2016



Juan Carlos Solano Minchalo

C.I: 0106625262

Edwin Patricio Patiño Ramón, autor de la tesis "Diseño e implementación de un prototipo de supervisión de un sistema de control industrial utilizando plataformas empotradas de bajo costo y controladores lógicos programables PLCs.", reconozco y acepto el derecho de la Universidad de Cuenca, en base al Art. 5 literal c) de su Reglamento de Propiedad Intelectual, de publicar este trabajo por cualquier medio conocido o por conocer, al ser este requisito para la obtención de mi título de *ingeniero Electrónico y en Telecomunicaciones*. El uso que la Universidad de Cuenca hiciere de este trabajo, no implicará afección alguna de mis derechos morales o patrimoniales como autor.

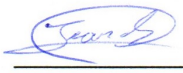
Cuenca, Abril 2016

Edwin Patricio Patiño Ramón

C.I: 0104953344

Juan Carlos Solano Minchalo, autor de la tesis "Diseño e implementación de un prototipo de supervisión de un sistema de control industrial utilizando plataformas empotradas de bajo costo y controladores lógicos programables PLCs.", reconozco y acepto el derecho de la Universidad de Cuenca, en base al Art. 5 literal c) de su Reglamento de Propiedad Intelectual, de publicar este trabajo por cualquier medio conocido o por conocer, al ser este requisito para la obtención de mi título de *ingeniero Electrónico y en Telecomunicaciones*. El uso que la Universidad de Cuenca hiciere de este trabajo, no implicará afección alguna de mis derechos morales o patrimoniales como autor.

Cuenca, Abril 2016



Juan Carlos Solano Minchalo

C.I: 0106625262

CERTIFICO

Que el presente proyecto de tesis: "Diseño e implementación de un prototipo de supervisión de un sistema de control industrial utilizando plataformas empotradas de bajo costo y controladores lógicos programables PLCs." fue dirigido por mi persona.

Dr. Luis Ismael Minchala Ávila.
Director de Tesis

Agradecimientos

En primer lugar agradecemos a Dios por habernos guiado en nuestras vidas.
En segundo lugar agradecemos a nuestra institución por brindado la oportunidad de prepararnos profesionalmente.
Agradecemos a nuestros padres y familiares por brindarnos su apoyo incondicional.
Agradecemos a nuestro director de tesis, Dr. Ismael Minchala, quien con su esfuerzo y dedicación nos supo guiar en el desarrollo de nuestra tesis.
Agradecemos al Dr. Fabián Astudillo, por brindar su colaboración y ayuda en el desarrollo de la tesis.
Al Sr. Francisco Sánchez por permitirnos usar las instalaciones y los equipos necesarios para el desarrollo de la tesis.
Agradecemos a los profesores de la facultad, quienes colaboraron en el desarrollo de la tesis.
Finalmente, agradecemos a todos nuestros compañeros y amigos de clase, en especial a Saúl Ochoa y Esteban Velecela, quienes con su apoyo nos impulsaron a la terminación de nuestra tesis.

Dedicatoria

Dedico este trabajo a mi familia, en especial a mis padres Carlos y Ana que con sus sabios consejos y apoyo me ayudaron a superar las dificultades y desarrollo de la tesis. A mis hermanos Carlos, Nanncy y mis sobrinas Michelle y Gaby que fueron y son la fuerza necesaria para afrontar las barreras que se me presentan en la vida.

Edwin

Dedico este trabajo principalmente a Dios, por haberme dado la vida y permitirme haber llegado hasta este momento tan importante de mi formación profesional. A mi madre, por ser el pilar más importante en mi vida y demostrarme siempre su amor incondicional. A mis hermanas, quienes con sus consejos, me supieron brindar la fuerza necesaria para nunca rendirme. Finalmente a mis amigos, por siempre brindarme su apoyo en el trascurso de mi vida.

Juan.

Abreviaciones

PLC	Programmable Logic Controller
SOA	Service Oriented Architecture
IEC	International Electrotechnical Commission
POO	Programación Orientada a Objetos
IT	Information Technology
FDT	Field Device Tool
OLE	Object Linking and Embedding
MES	Manufacturing Execution Systems
PCS	Process Control Systems
VPP	Virtual Power Plant
DER	Distributed Energy Resources
CPU	Central Processing Unit
HDMI	High-Definition Multimedia Interface
USB	Universal Serial Bus
CIM	Computer Integrated Manufacturing
PID	Control Proporcional Integral Derivativo
ERP	Enterprise Resource Planning
CAD/CAM/CAE	Computer Aided Design/ Manufacturing/ Engineering
LAN	Local Area Network
MAP	Manufacturing Automation Protocol
TOP	Technical and Office Protocol
MMS	Manufacturing Message Specification
TCP/IP	Transmission Control Protocol/ Internet Protocol
CSMA/CD	Carrier Sense Multiple Access with Collision Detection

OPC	Object Linking and Embedding for Process Control
COM	Component Object Model
DCOM	Distributed COM
DDE	Dynamic Data Exchange
OPC DA	OPC Data Access
OPC UA	OPC Unified Architecture
HTTP	Hypertext Transfer Protocol
SOAP	Simple Object Access Protocol
SDK	software development kit
EMS	Energy Management Systems
DMS	Distribution Management Systems
FSK	Frequency Shift Keying
UTP	Unshielded Twisted Pair
XML	Extensible Markup Language
SQL	Structured Query Language
UFW	Uncomplicated Firewall

Capítulo 1

Introducción

1.1. Antecedentes

En la actualidad, las redes de comunicación tienen gran protagonismo en la mayor cantidad de actividades que desarrollamos diariamente. Sin embargo, las redes de comunicación en el ámbito industrial no son tan amplias. Sin embargo, en las últimas dos décadas han ganado reputación en la implementación de sistemas escalables dentro de la industria.

La nueva sistemática de automatizar procesos se basa en la obtención de una mayor eficiencia y optimización de recursos. Esto se logra al implantar sistemas distribuidos que van desde procesos de fabricación y manufactura hasta una integración de diferentes áreas dentro de la empresa (fabricación, gestión de producción, control de calidad, distribución, etc.).

En la actualidad la industria ecuatoriana experimenta cambios en los procesos de automatización, impulsado por el mejoramiento constante de diferentes técnicas de producción, optimización de uso de materias primas, ahorro energético y mejoramiento de la calidad de los productos.

La mayoría de empresas con sistemas productivos, tienen la necesidad de llevar control de la información de estos procedimientos. La información de los métodos existentes en las plantas industriales, proviene de sensores instalados en planta, alarmas y eventos generados por dispositivos de automatización, estados de actuadores, entre otros [6] [7].

En muchas ocasiones, la información proveniente de estos procesos es muy po-

bre o hasta nula, debido a la poca inversión en nuevas tecnologías de automatización, control, comunicaciones industriales y servicios de consultoría especializada. Sin embargo, la migración a nuevas tecnologías como el cambio o renovación de los controladores lógicos programables (PLC, por sus siglas en inglés) tiene un costo elevado.

El crecimiento de las industrias, en muchos casos se ha dado de una forma poco ordenada. Es decir, los nuevos procesos se adaptan a los anteriores, pero no significa que se integraron de forma óptima, por lo que no se aprovecha de una manera eficiente los recursos. Esto conlleva a integrar PLCs de diferentes marcas, una baja integración de redes de sensores en una misma industria. Adicionalmente, la mayor parte de las veces no existe un control central de todos los procesos y menos aún de un seguimiento de la calidad de éstos, mediante almacenamiento y análisis de datos.

Es preciso considerar el análisis de la instalación de nuevos elementos en una fábrica. Con este análisis se verifica que todos y cada uno de los aspectos de la fábrica puedan integrarse a una red de comunicaciones. De este modo, al implementar una red industrial se adquiere la posibilidad de conectar elementos de automatización (autómatas programables, sensores, actuadores, paneles de visualización, etc.) con elementos de gestión y supervisión de fábrica entre sí, sin dejar atrás los dispositivos de control de procesos.

Esta disposición se realiza por etapas, sin efectuar un cambio radical en los sistemas industriales previamente implantados. Debido a que, no siempre se tiene la posibilidad de una intercomunicación global en las instalaciones viejas.

El concepto de sistemas industriales distribuidos comprende un campo muy amplio en la industria, ejecutando aplicaciones que que despliegan desde una automatización simple de un proceso aislado hasta la gestión completa de una fábrica. Por tanto, los sistemas de control distribuidos a nivel industrial, proveen de herramientas que permiten interconexión entre dispositivos y permiten mejorar la productividad, confiabilidad y la operabilidad de los procesos.

1.2. Objetivos

1.2.1. Objetivo general

Diseñar un prototipo de sistema de control distribuido con su respectivo software y expandir las funcionalidades de automatización para dispositivos de automatización clásicos, que permitan incluir funciones de comunicación, preprocesamiento y procesamiento de variables de proceso, almacenamiento y control basado en eventos.

1.2.2. Objetivos específicos

a) Generar una arquitectura orientada a servicios (SOA, por sus siglas en inglés) para control industrial

b) Observar los PLCs que existen en el laboratorio y sus funcionalidades

c) Explorar la plataforma embebida de bajo costo (Raspberry Pi)

d) Generar una base de datos con la información proporcionada por los procesos industriales

1.3. Revisión bibliográfica

La Comisión Electrotécnica Internacional (IEC, por sus siglas en inglés), es una organización de normalización en los campos eléctrico, electrónico y tecnologías relacionadas. Esta comisión en su estándar 61131 publica un conjunto de normas e informes técnicos con el objetivo de normalizar los autómatas programables. Uno de ellos son los PLCs, que se utilizan con bastante frecuencia en la industria con el fin de llevar a cabo diferentes procesos de automatización, producción, etc.

El estándar 61131 tiene la finalidad de:

- Definir y establecer las principales características para seleccionar y aplicar un PLC con sus diferentes periféricos
- Determinar los requerimientos mínimos que se necesita en condiciones de servicio, aspectos constructivos, características funcionales, ensayos aplicables y seguridad general de los PLCs y sus periféricos

- Especificar los lenguajes de programación de mayor uso, reglas sintácticas y semánticas, juego de instrucciones fundamental, ensayos y los medios de ampliación y adaptación de los equipos
- Proporcionar información general y pautas aplicativas a los usuarios
- Establecer la intercomunicación que se desarrolla entre los PLCs y otros sistemas [8]

El estándar IEC 61131-3 [9] es ampliamente utilizado por los ingenieros de control en la especificación de la parte de software de sistemas, en el dominio de la automatización industrial. No obstante, impone varias restricciones para el desarrollo de sistemas complejos.

La programación de sistemas de automatización industrial en la actualidad se basa en el estándar IEC 61131. Sin embargo, no cumplen los requerimientos de la programación orientada a objetos (POO, por sus siglas en inglés) de sistemas distribuidos grandes. Las herramientas de POO deben tener requerimientos adicionales como configuración directa de señales E/S, programación multiparadigma, considerando que, los sistemas de control industrial no son sistemas de cómputo genéricos [10] [11].

La POO se fundamenta en la norma IEC 61499 [10] que es considerada como una extensión de la norma IEC 61131. Esta norma se diseñó con el fin de brindar una referencia, facilitando el desarrollo de software y aplicaciones de control con lógica descentralizada. Para este propósito, propone una arquitectura fundamentada en bloques funcionales.

Estos bloques se caracterizan por las entradas, salidas y sus funciones internas, al igual que los diagramas de bloques clásicos. A partir de las entradas y la definición del bloque, es decir su comportamiento interno, se obtiene una salida deseada. Al momento de realizar la interconexión de los bloques se consigue modelar aplicaciones simples y complejas.

Otro aspecto de esta norma, trata sobre la ejecución de los bloques funcionales. Este define que, la ejecución de los bloques no se realiza hasta no recibir una señal de entrada, por lo que los bloques permanecen en reposo hasta que sean solicitados. Sin embargo, en la norma IEC 61131 los componentes son verificados

periódicamente por subrutinas. Esta ejecución tiene como finalidad la portabilidad y reutilización de los bloques funcionales, impulsado por que los bloques tienen su propio contexto y ejecución. Finalmente, si se produce una petición se considera independiente del resto del sistema.

La capacidad para responder rápidamente a causa de cambios, y optimizar los procesos, es un factor importante para el buen funcionamiento y control de un proceso industrial. La agilidad de estos factores, puede verse cuestionado si se encuentra en entornos de la tecnología de la información (IT, por sus siglas en inglés), si no se entrega una respuesta rápida y de forma flexible a los cambios que afectan a la actividad de la industria. En consecuencia, se debe liberar la capacidad que poseen las aplicaciones y recursos de IT, con el fin de aumentar su disponibilidad en forma general a la industria, para facilitar la optimización y agilidad de los procesos industriales.

La arquitectura orientada a servicios, es un concepto de arquitectura de software. Este concepto define el uso de servicios para dar soporte a ciertos requisitos de un conjunto de procesos industriales [12]. Esta arquitectura permite crear sistemas autónomos e interoperables altamente escalables, que mejoran el rendimiento de los procesos industriales [13].

1.4. Estado del arte

La industria hoy en día requiere una reacción eficaz a los eventos críticos que ocurren en la planta de producción. Por lo tanto, los datos a nivel de dispositivo necesitan ser integrados en los procesos de negocio de una manera estandarizada y flexible. Esto se realiza, con el fin de evitar el consumo excesivo de tiempo, que como consecuencia ralentiza los medios de comunicación.

La incorporación de sistemas de software en código abierto para controlar una red distribuida de PLCs se ha incrementado en los últimos años. Este tipo de soluciones permiten a las empresas implementar su propia plataforma de software, con el fin de disminuir costos de desarrollo de soluciones de conectividad industrial con esquemas de licenciamiento pagados [14].

En [15] se presenta un desarrollo de un concepto que integra las ventajas de

las tecnologías de lenguaje descriptivo de dispositivos electrónicos (EDDL, por sus siglas en inglés), y herramienta de equipo de campo (FDT, por sus siglas en inglés) en uno coherente. Este nuevo concepto define un modelo para incorporar dispositivos y una arquitectura cliente-servidor, que hace uso de las diversas ventajas de la vinculación e incrustación de objetos (OLE, por sus siglas en inglés) para procesos de control con arquitectura unificada (OPC UA, por sus siglas en inglés). Este concepto se evalúa mediante la creación de un prototipo integral que incluye diferentes dispositivos de campo de proceso y automatización de fábricas.

En [16] se muestra un enfoque para la utilización de OPC UA en conjunción con el *middleware*. Esto se basa en SOA y los estándares ISA-88/95, con el fin de integrar los sistemas de ejecución de fabricación (MES, por sus siglas en inglés) y los sistemas de control de procesos (PCS, por sus siglas en inglés) en el contexto de la gestión de procesos por lotes. La integración de estos conceptos se realiza identificando los requisitos de comunicación. Seguido de la propuesta de un diseño de software, que combina las tecnologías mencionadas. El diseño se evalúa con la asistencia de un escenario experimental de implementación y prueba.

En [17] se expone una integración de sistemas empresariales con actividades de fabricación, fundamentada en servicios web. Para ello, se usa dispositivos integrados en red con SOA listo para funcionar. Para incorporar estos conceptos se debe examinar los requisitos, con el fin de derivar una arquitectura apropiada que soporte este nuevo sistema incorporado. Este nuevo sistema ofrece un suministro oportuno de datos, por lo que da lugar a costes de producción reducidos. Evita, el impacto de la información a nivel de dispositivos, en procesos de negocio, en la comunicación bidireccional directa, con los servicios a nivel de dispositivo.

En [18] muestra cómo IEC 61850 puede beneficiarse de la tecnología OPC UA y acelerar el desarrollo de redes inteligentes. OPC UA se basa en la arquitectura orientada a servicios (SOA). Se expone cómo el paradigma SOA, proporciona una característica de valor añadido para el control de planta de potencia virtual (VPP, por sus siglas en inglés). Como resultado, se plantea una arquitectura de automatización de VPP.

La arquitectura de automatización de VPP es el resultado del análisis de la metodología de integración entre IEC 61850 y OPC UA, mediante el uso de servidores de recursos energéticos distribuidos (DER, por sus siglas en inglés) como

dispositivos SOA listos para funcionar. Se llegó a esta solución, al observar la evolución de los sistemas de energía existentes, hacia redes inteligentes.

Las redes inteligentes tienen un gran número de peligros potenciales y problemas no resueltos que están posponiendo su avance aerodinámico. El control colectivo de los DERs a través del modelo de planta de potencia virtual, es uno de los conceptos clave de red inteligente, que carecen de la capacidad de integración sin fisuras.

Una de las principales contribuciones del estándar IEC 61850, son los modelos semánticos de datos. Estos describen diversos subsistemas de servicios públicos, incluyendo DER. La posibilidad de utilizar la semántica DERs, con las tecnologías de automatización industrial como OPC UA, podría reducir significativamente el periodo de tiempo en la integración de VPP.

1.5. Contribuciones de las tesis

Las contribuciones de la tesis son las siguientes:

- Desarrollo de un prototipo de expansión de funcionalidades de PLCs utilizando una plataforma embebida de bajo costo, Raspberry Pi
- Metodología de integración de un servidor OPC UA abierto (software libre) en la Raspberry Pi
- Configuración del servidor OPC UA en el dispositivo embebido en una red de prueba de la Universidad de Cuenca e integración con un cliente OPC UA comercial (UA expert de Unified Automation)
- Integración del servidor OPC con la librería Snap7 para establecer comunicaciones confiables con PLCs de marca Siemens
- Reducción importante de costos en la implementación de un servidor OPC UA
- Reducción de problemas de comunicaciones entre los PLCs de la red industrial y el servidor OPC UA de código abierto

En la figura 1.1 se muestra la arquitectura y funcionalidades de la alternativa de servidor OPC UA de código abierto para comunicación industrial.

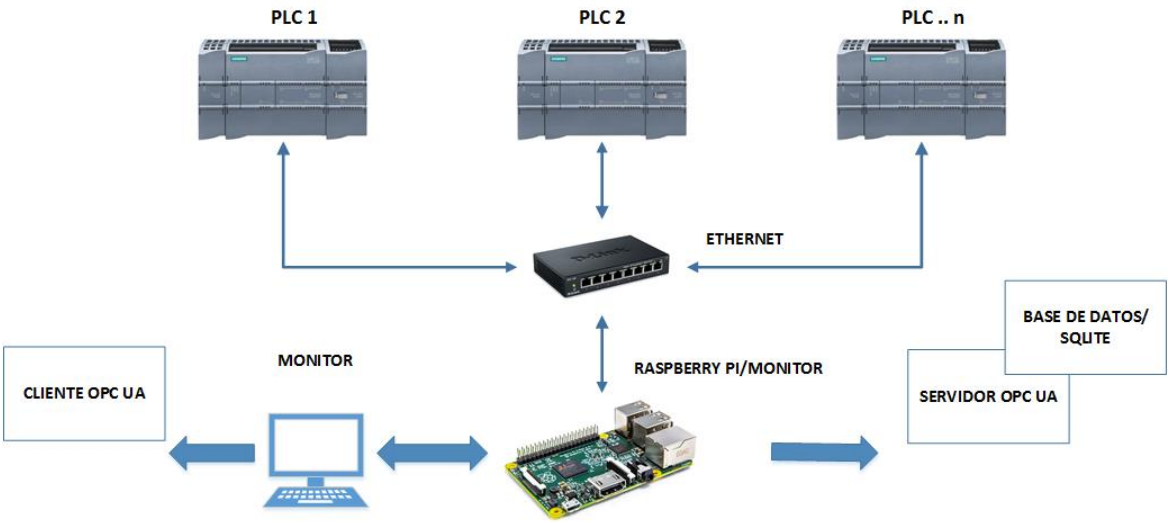


Figura 1.1: Concepto de integración del prototipo DEFA

Capítulo 2

Marco teórico

2.1. PLC, funcionalidad y características

2.1.1. Controladores lógicos programables

En la actualidad, un controlador forma parte indispensable dentro de cualquier sistema. Básicamente, el control se lleva a cabo mediante la lectura y escritura de variables. Cada variable dentro de un controlador está relacionada con el estado actual de un proceso.

En lugar de disponer de hardware específico para cada situación de control, es posible utilizar un solo sistema para todas las situaciones. Los microprocesadores son una alternativa viable hoy en día. Cada instrucción en un microcontrolador culmina en una acción, que puede ser tan simple como la de un interruptor o tan compleja como controlar la velocidad de giro de un motor.

Un controlador lógico programable (PLC, por sus siglas en inglés), es una forma especial de un microprocesador, como se muestra en la Figura 2.1.

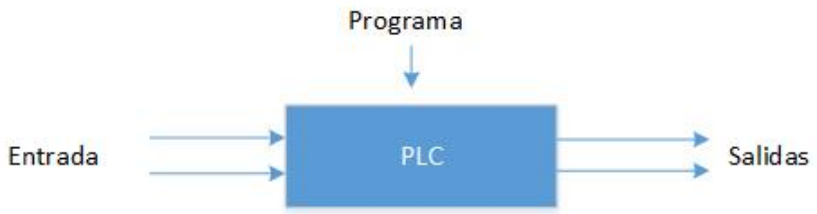


Figura 2.1: Diagrama E/S de un PLC

Un PLC dispone de una memoria programable para almacenar instrucciones e

implementar funciones lógicas, secuenciales, y aritméticas para controlar máquinas y procesos industriales [19]. Están diseñados para ser operados por ingenieros con un conocimiento limitado sobre ordenadores y lenguajes de computación. El término lógico se utiliza debido a la programación que ocupa principalmente la aplicación de lógica booleana y la conmutación de las operaciones.

El PLC monitorea constantemente sus periféricos de entrada y de acuerdo a su programación genera la salida deseada. Los PLC son similares a las computadoras, pero mientras que los ordenadores están optimizados para tareas de cálculo y procesamiento de datos, los PLC están optimizados para las tareas de control en un entorno industrial.

Todo PLC está compuesto por: procesador, unidad de memoria, unidad de fuente de alimentación, unidad de entrada/salida, interfaz de comunicaciones y la unidad de programación como se muestra en la Figura 2.2.

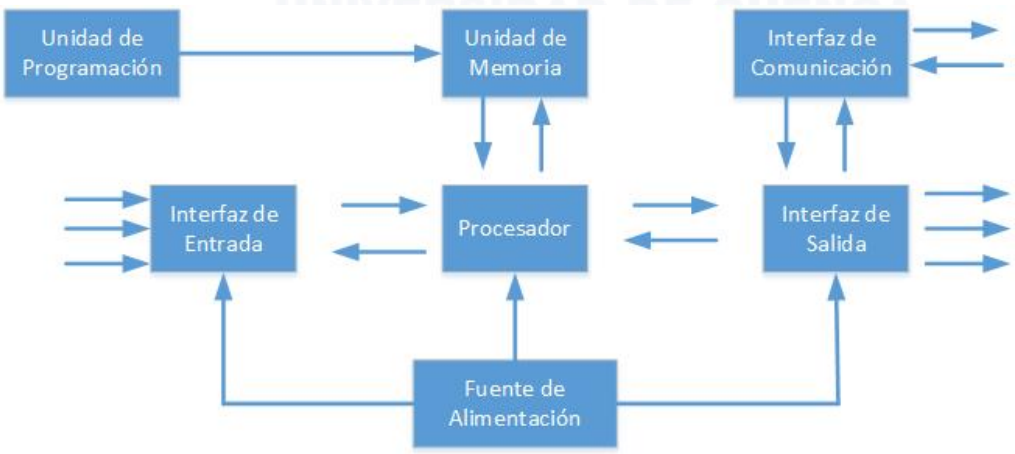


Figura 2.2: Diagrama de bloques del funcionamiento interno de un PLC

Los bloques de la Figura 2.2 se describen a continuación.

- La unidad central de procesamiento (CPU, por sus siglas en inglés) del microprocesador es la encargada de interpretar las señales de entrada y llevar a cabo las acciones de control
- La unidad de suministro de energía, convierte el voltaje alterno a continuo, por lo general de 5V. Sin embargo, depende de los módulos de entrada y salida a las que esté conectado

- Unidad de programación es usado para colocar el programa o instrucciones dentro de la memoria del procesador
- La unidad de memoria, lugar donde el programa es almacenado para ser usado por las acciones de control
- Las interfaces de entrada y salida del procesador reciben información de dispositivos externos. La señal de entrada puede provenir de conmutadores o de sensores eléctricos, así como de mecanismo de control, entre otros

2.1.2. Programación

La programación se puede realizar mediante:

- Dispositivos portátiles: que posean la suficiente memoria para permitir a la unidad de programación del PLC retener el programa, mientras la tarea se lleva a cabo
- Consolas de escritorio: una unidad con un teclado y un visor de pantalla
- Computador personal: mayormente usado, debido a que poseen tarjetas de comunicación especiales. Una ventaja importante de utilizar un ordenador, es que el programa puede ser almacenado en el disco duro y realizar copias fácilmente

Los fabricantes de PLCs tienen softwares de programación para sus equipos, por ejemplo, Mitsubishi tiene MELSOFT.

2.2. Raspberry Pi

Es un dispositivo de hardware, empleado en aplicaciones de desarrollo tanto de software como de hardware. Capaz de recibir, procesar y transmitir datos. Cuenta con periféricos para conectarse a dispositivos externos. Una de sus características más importantes, es la conexión a monitores o pantallas mediante su conexión HDMI, además de su conexión USB, que le permite la conexión de teclado, ratón y otros dispositivos.

A pesar de su tamaño reducido es capaz de procesar gran cantidad de datos. En algunos casos se lo compara con una computadora básica de escritorio. Su sistema operativo por defecto, permite la creación de aplicaciones en Scratch como en Python. Es uno de los dispositivos más utilizados en los últimos años debido a su gran prestación en múltiples proyectos.

Características físicas

El análisis se basa en el modelo B de segunda generación, aunque en la actualidad existe la versión B de tercera generación. Pese a ello, se siguen manteniendo muchas de las características de sus antecesores.

- Posee un chip integrado Broadcom BCM2835, que contiene un procesador ARM11 con varias frecuencias de funcionamiento y la posibilidad de subirla (overclocking) hasta 1 GHz
- Un procesador gráfico VideoCore IV, y memoria RAM de 1 gigahertz
- Puertos GPIO, hasta 40 pines, que permiten hacer todo tipo de comunicación de hardware, así como el control de otros dispositivos
- Micro SD. El antiguo zócalo de la tarjeta SD ha sido reemplazado por una versión push-push de micro SD, lo que permite instalar un sistema operativo en una tarjeta de memoria de 2 GB o más (clase 4 o mejor)
- Menor consumo de energía. Mediante la sustitución de reguladores lineales, se ha reducido el consumo de energía entre 0.5W y 1W
- Mejor sonido. El circuito de audio incorpora una fuente de alimentación de bajo ruido dedicado
- Más pequeño, factor de forma más ordenada. Alineado el conector USB con el borde de la placa. Cuatro orificios de montaje colocados en ángulo recto
- Conexión ethernet, con posibilidad de añadir una conexión Wi-Fi, gracias a los dos puertos USB incluidos. Los puertos tienen una limitación de corriente, por lo que, si se desea conectar discos duros u otros dispositivos, se lo debe realizar a través de un hub USB con alimentación externa

2.3. Redes industriales

2.3.1. Definición

Una red industrial, es un conjunto de elementos electrónicos y de control. Generalmente la integran dispositivos como PLCs y sus respectivos sistemas de control. Se utilizan para gestionar y monitorear ciclos de vida de los procesos de manufactura.

Si bien las redes industriales, no son algo nuevo, en la última década, se ha dado paso a la evolución y transformación de las mismas. Se ha pasado de conexiones punto a punto entre equipos a un verdadero sistema de control mediante conexiones multipunto [3].

En un entorno industrial, se pueden encontrar equipos de diferentes fabricantes, y en algunas ocasiones, estos dispositivos se encuentran en diferentes localidades. La integración de los mismos, permite que el proceso de manufactura, pueda expandirse. Dicha expansión se logra gracias a la flexibilidad de software estandarizados.

La integración de ordenadores, controladores, sensores y actuadores, proporciona un mayor rendimiento, así como deja las puertas abiertas a nuevas posibilidades dentro de las redes industriales. Entre las características más notables al optar por este sistema están:

- Visualización y supervisión de todo el proceso de manufactura
- Toma de datos más rápido que otras redes
- Mejora del rendimiento de todo el sistema
- Posibilidad de intercambio de información, entre los distintos equipos conectados
- Integración y manejo de un sistema distribuido

2.3.2. PIRAMIDE CIM (Computer Integrated Manufacturing)

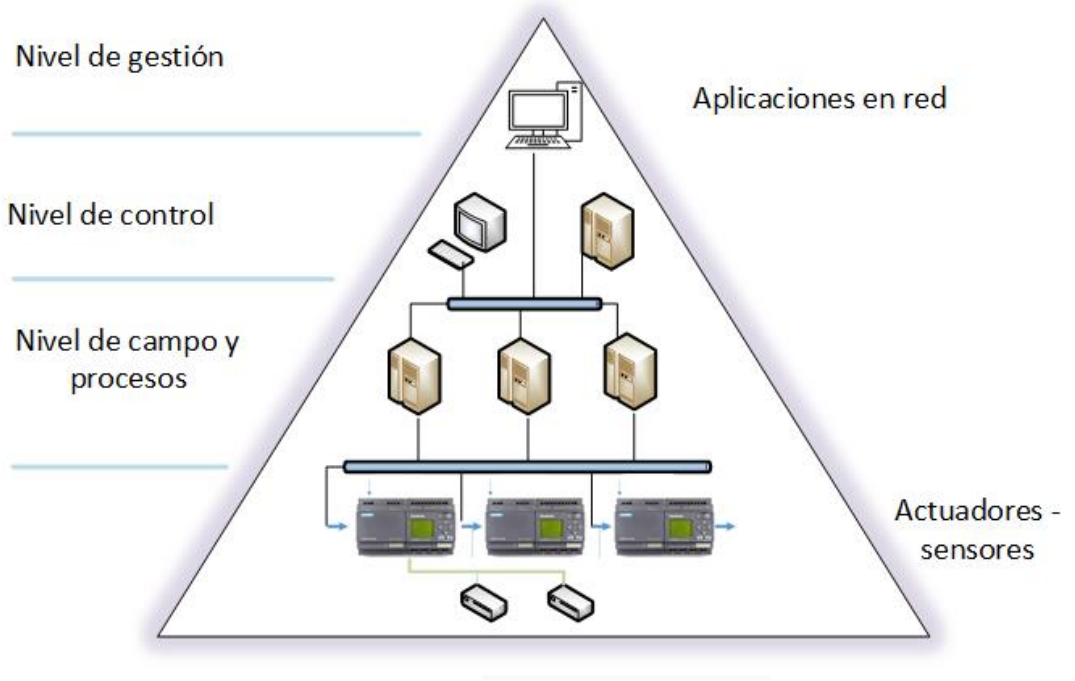


Figura 2.3: Pirámide CIM

La Figura 2.3 muestra la el modelo jerárquico de la pirámide CIM. Cada dispositivo conectado se encuentra agrupado de acuerdo al área y función que desempeña dentro de la industria. Actualmente se definen cuatro niveles [20], estos niveles son:

- Nivel de gestión: es el nivel más elevado, se encarga de integrar los siguientes niveles en una estructura de fábrica, además debe cumplir con los requerimientos de seguridad informática para el intercambio de datos, por lo general de 100 Mbits
- Nivel de control: enlaza las zonas de trabajo, cliente/servidor, para ello se emplean computadores dedicados al diseño y programación. La comunicación se la realiza por medio de la red LAN, ya sea en el medio eléctrico, óptico o radio. Este nivel establece la interfaz hombre-máquina
- Nivel de campo y procesos: en este nivel, se encuentran las subredes que interconectan a los diferentes dispositivos, tales como multiplexores de E/S,

controladores PID, etc.

- Nivel de E/S: es el nivel más bajo de la pirámide, se encuentran los sensores y actuadores que se encargan propiamente del proceso productivo. Toma en cuenta que los costes de cableado sean óptimos para la empresa

Dentro de un proceso de manufactura, la automatización integrada es un concepto importante, puesto que, al tener un sistema de control y gestión debidamente adecuado se puede verificar periódicamente los procesos involucrados en cada nivel del sistema de producción

Tabla 2.1: Parámetros de la pirámide CIM de acuerdo a los niveles de trabajo [3]

Nivel	Tipo de sistema electrónico de control	Parámetro		
		Tiempo de respuesta	Relación (%) de tareas gestion/control	Operatividad exigible (%)
4	Computador de Planta	De días a segundos	95 -100/0-5	>10
3	Controlador de área	De minutos a segundos	90 - 95/5-10	<10
2	Controlador de célula	De segundos a milisegundos	80 -90/10-20	80 - 90
1	Controlador de proceso	De milisegundos a microsegundos	5 -10/90-95	90 - 95

La Tabla 2.1 muestra el tiempo de respuesta de los procesos del nivel 3 y 4, son grandes en comparación con los niveles 1 y 2, esto se debe a que los procesos superiores no son críticos. Sin embargo, se puede notar que en los niveles más altos se manejan gran cantidad de datos. En cuanto a los niveles inferiores, al estar directamente involucrados con el proceso de manufactura, deben trabajar en tiempo real, por lo que el tiempo de respuesta debe ser lo más rápido posible.

2.3.3. Tipos de redes industriales

Debido a los requerimientos definidos con anterioridad, las redes industriales se las clasifica en dos tipos, las de datos y las de control [3].

Red de datos

Una red de datos establece los protocolos necesarios para una comunicación entre dispositivos, por lo que a su vez se subdividen en redes de empresa y célula.

La red de empresa ejecuta las siguientes aplicaciones:

- Programas de planificación de recursos empresariales (ERP, por sus siglas en inglés)
- Programas de ejecución de sistemas de manufactura (MES, por sus siglas en inglés)
- Programas de diseño asistido por computador, de fabricación e ingeniería (CAD/CAM/CAE, por sus siglas en inglés)
- Herramientas de aplicación general

Este tipo de red se caracteriza por utilizar casi en su totalidad Ethernet para redes de área local (LAN, por sus siglas en inglés) y el internet para comunicarse con sedes de la misma empresa.

Cualquier empresa, no cumple con los requerimientos de una red industrial. No están diseñadas para trabajar en un sistema hostil, con grandes perturbaciones y temperaturas extremas. En las últimas décadas, se han diseñado nuevas redes de comunicación, capaces de dar soporte a las operaciones de la empresa, con lo que se dio origen a las redes célula.

Red de control

Una red de control, resuelve problemas de comunicación de los niveles inferiores de la pirámide CIM. Esta red establece el camino por el que una señal de control llega hasta los sensores. Estos sensores son los encargados de supervisar y manejar el proceso de manufactura de la empresa.

Antes de este tipo de red, los procesos de automatización, eran procesos independientes. El problema surge al tener un elevado número de hilos trabajando. La Figura 2.4 muestra el desmesurado volumen de cableado que requieren estas redes. Para solucionar este inconveniente, surgen las redes de sensores – actuadores. Esta red dispone de un procesador de comunicaciones y un sistema de control.

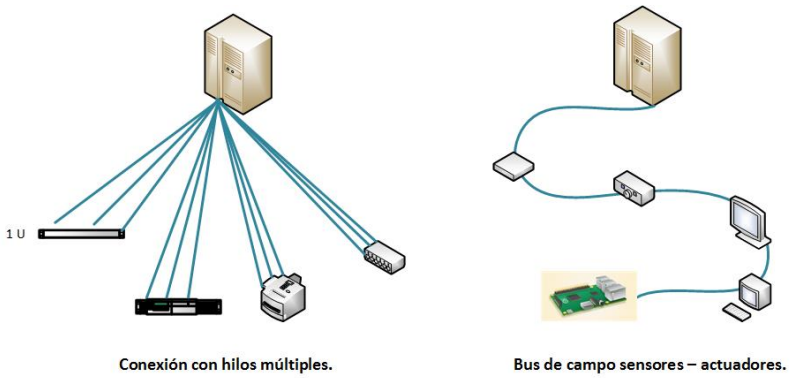


Figura 2.4: Conexión punto a punto vs multipunto de los diferentes sensores/actuadores

La red de sensores – actuadores está destinada a realizar la comunicación de varios sistemas electrónicos de control entre sí. Los servicios de comunicación no solo permiten intercambios de información, sino también llevar a cabo tareas de diagnóstico, carga, descarga, ejecución y depuración de los programas ejecutados.

2.3.4. Topologías de red de área local

La topología de red define la forma de interconexión de nodos. Esto determina las rutas entre pares de nodos. las topologías mas utilizadas son:

- Topología en estrella
- Topología en anillo
- Topología en bus

Topología en estrella

En esta topología cada nodo está conectado con un enlace de punto a punto al nodo central, como se muestra en la Figura 2.5. En algunas redes, toma todas las decisiones de ruteo [21].

las ventajas de una topología en estrella son:

- Posee una interfaz simple y de bajo costo
- Al poseer pocos nodos, no existe sobrecarga de red
- La falla de un nodo solo le afecta al mismo y no a toda la red

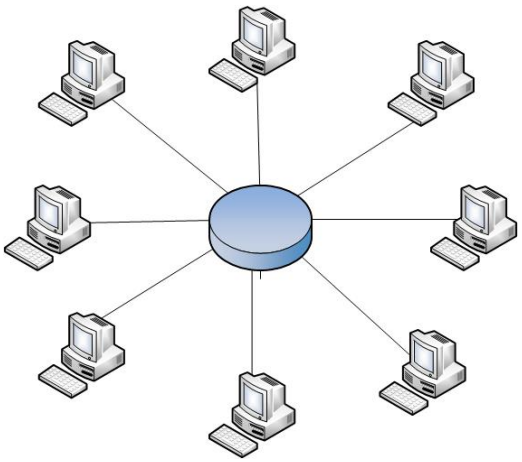


Figura 2.5: Topología en estrella

Topología en anillo

Esta topología posee un solo canal de comunicación. Al tener un solo canal, todos los mensajes pasan de nodo en nodo por todo el anillo, como se muestra en la Figura 2.6. Para aceptar un mensaje, el nodo correspondiente debe estar habilitado para reconocer su propia dirección en el mensaje.

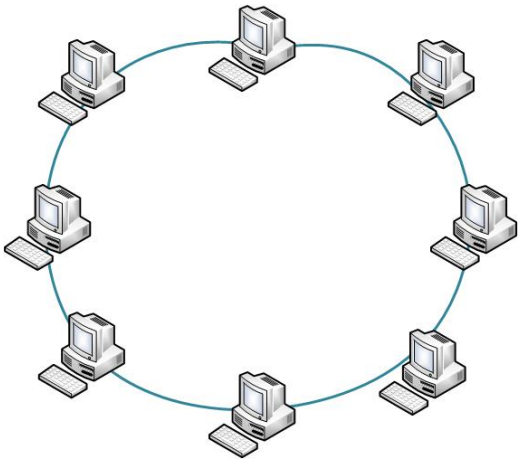


Figura 2.6: Topología en anillo

las ventajas de una topoogía en anillo son:

- Ausencia de un nodo central
- Debido a que la señal es regenerada en cada nodo, la tasa de error es menor

Topología en bus

En esta topología cada mensaje es enviado y recibido por todos los nodos, como se muestra en la Figura 2.7. Debido a que todos los nodos están comparten el mismo medio, solo uno puede transmitir un mensaje a la vez.

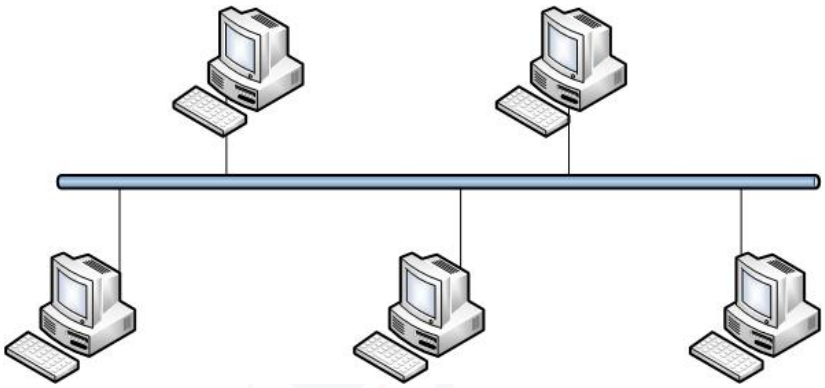


Figura 2.7: Topología en bus

Entre sus ventajas están:

- Al no poseer un nodo central, la falla de un nodo no afecta a toda la red
- Provee un alto nivel de flexibilidad y escalabilidad

2.3.5. Protocolos de comunicación

A lo largo del tiempo surgido diferentes protocolos de comunicación, los cuales se muestran a continuación:

A) **Modelo OSI**

OSI como tal no es un protocolo. Es el modelo para el intercambio de información. El modelo es una herramienta útil, para el diseño, descripción, implementación, estandarización, y uso de redes de comunicación.

La Figura 2.8 muestra el diagrama de bloques del modelo OSI creado por la ISO. Al igual que las redes tradicionales, una red industrial también utiliza este este modelo. Permite interconectar todos los equipos en red, además de que no impone un protocolo definido para cada nivel. Los fabricantes pueden elegir que software desean utilizar para la comunicación.

Figura 2.8: Modelo OSI, para la conexión de sistemas informáticos abiertos

Mediante la capa física se puede conectar dispositivos con interfaz RS232, RS422, RS423, entre otros.

La capa de enlace de datos, especifica los protocolos de control y de acceso al medio. Su principal es establecer y mantener los enlaces lógicos de entre un nodo y red local.

La capa de red provee ruteo y conmutación para cada servicio de la red. Estos servicios permiten la interconexión con otras redes de área local.

La capa de transporte, permite la transferencia de datos de nodo a nodo dentro de la red. Esta establece y mantiene una conexión confiable entre dos terminales.

La función de la capa de sesión es manejar y sincronizar el intercambio de información entre aplicaciones que están corriendo en la red.

La capa de presentación, recibe la información y la transforma a un formato específico. El formato depende de la maquina a la que le llegue el mensaje. Esto permite que dos terminales se comuniquen entre si, incluso si manejan diferentes formatos de información.

La función de la capa de aplicación es proveer todos los servicios a las aplicaciones, tales como: mensaje de transferencia, acceso a archivos, manejo de red, entre otros.

B) **Protocolo de automatización y manufactura (MAP)**

En la década de los 80, uno de los protocolos más empleados fue el protocolo de automatización y manufactura (MAP, por sus siglas en inglés). Apareció como una red de entorno industrial, proporcionando un medio determinista y normalizado por la IEEE.

Los objetivos del protocolo son:

- Conseguir compatibilidad en la comunicación, para integrar los diferentes dispositivos de planta
- Proveer un ambiente de participación entre diferentes fabricantes en una red de comunicación estándar

El protocolo MAP, se basa en su totalidad en el modelo OSI. Sin embargo, el protocolo fue rápidamente desplazada por el protocolo técnico y de oficina (TOP, por sus siglas en inglés). Con la unión de los dos surgió el MAP/TOP, que contempla la capacidad de intercomunicación de los sistemas de control con los de oficina.

C) **Protocolo de especificación de mensaje de manufactura (MMS)**

El protocolo de especificación de mensaje de manufactura (MMS, por sus siglas en inglés), es el protocolo más utilizado por las redes de célula. Este último permite la monitorización y gestión de los sistemas de control, como por ejemplos robots, PLCs, etc. [20].

D) **Modbus**

Es un protocolo de transmisión desarrollado por *Gould Modicon*. A diferencia de otros protocolos, la interfaz de la capa física no esta especificada.

Las limitaciones de Modbus, se deben a los enlaces seriales de EIA-232/485. También conocidos como RS232/RS485. la Tabla 2.2 muestra algunas de las características de este tipo de interfaces.

Tabla 2.2: Caracteristicas de las interfaces RS232/RS422/RS485

	RS-232	**RS-422**	**RS-485**
Cableado	Simple	Simple Multipunto	Multipunto
Número de dispositivos	1 transmisor 1 receptor	5 transmisores 10 receptores	32 transmisores 32 receptores
Modo de comunicación	full duplex	fullduplex half duplex	half duplex
Distancia máxima	50 pies a 19.2 Kbps	4000 pies a 100 Kbps	4000 pies a 100 Kbps
Tasa máxima de datos	19.2 Kbps	10 Mpbs	10 Mpbs
Señalización	no	si	si
Señal (dato 1)	-5 V min. -15 V max	2 V min. 6 V max.	1.5 V min. 5 V max.
Nivel de entrada mínimo	+/- 3 V	+/-0.2 V	+/-0.2 V
Salida de Corriente	500 mA	150 mA	250 mA

A pesar de las limitaciones, Modbus ofrece otras ventajas como, una amplia aceptación de los fabricantes y usuarios con múltiples sistemas de operación [22].

Los modos de transmisión de datos son:

- **ASCII:** debido a que es un formato legible, se lo utiliza para pruebas. Este modo hace referencia al Modbus-A. Cada mensaje tiene el doble de longitud de RTU
- **RTU:** es un formato hexadecimal, usado para operaciones normales. Hace referencia al Modbus-B

E) **Protocolo *Allen Brandley***

Existen tres estándares usados en comunicaciones de datos *Allen Brandley*

- Protocolo de datos *Highway*: es usado en redes de área local, que permiten comunicaciones *peer-to-peer* con mas de 64 nodos. Opera a una tasa de 57.6 Kbaud

- Protocolo de datos *Highway Plus*: similar al protocolo anterior, pero esta optimizado para trabajar con pocas computadoras

  Este protocolo usa tres capas del modelo OSI, la capa física, la capa de enlace de datos y la capa de aplicación.

Los problemas más comunes a este protocolo, están relacionados a problemas del cable y de diagnósticos de la red.

Para solucionar los problemas del cable, se presentan las siguientes recomendaciones:

- Realizar una conexión a tierra
- Retirar los cables de zonas cercanas a interferencias
- Tener precaución en la soldadura de los cables, entre otros

F) **Protocolo *highway addressable remote transducer* (HART)**

Este protocolo está diseñado para aplicaciones, en donde la información es obtenida de sensores, actuadores y sensores mediante técnicas digitales. Estos componentes son directamente enlazados a PLCs y computadores [22].

La ventaja de este protocolo es que permite transmisión de datos digitales sobre una red existente analógica. Este modelo híbrido, usa la modulación por desplazamiento de frecuencia (FSK, por sus siglas en inglés). Utiliza la frecuencia de 1200 y 2200 Hz para señales digitales de 1 y 0 respectivamente, como se muestra en la Figura 2.9

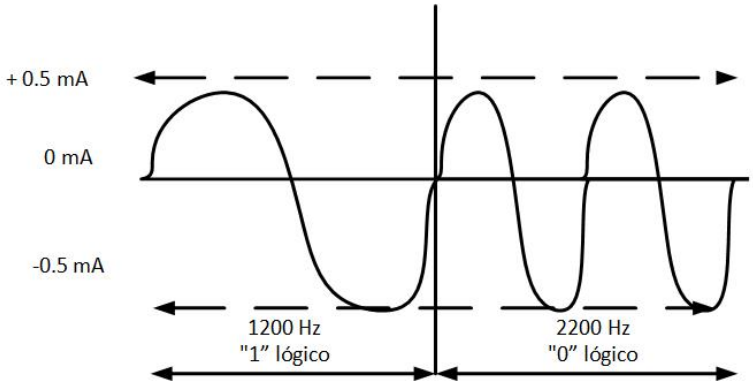


Figura 2.9: Señal generada en el protocolo HART

G) **Ethernet (IEEE802.3)**

Hoy en día, se habla de redes industriales mediante Ethernet. Esta red se encuentra ya desarrollada y que permite utilizar protocolos de comunicación TCP/IP. Utiliza la topología de bus en serie, para lo cual emplea el mecanismo CSMA/CD para el control del acceso al medio, como se muestra en la Figura 2.10. Se diferencian de las redes utilizadas en empresas e instituciones por su capacidad de adaptación en ambientes hostiles en los niveles de planta. Además, emplean redundancia de hardware y software y redefinición de protocolos.

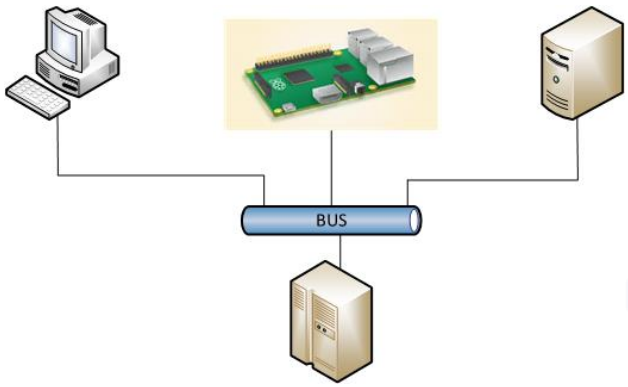


Figura 2.10: Conexión Ethernet

Para implementar una red Ethernet, se puede utilizar par trenzado o fibra óptica. Alcanza una velocidad de transmisión de 100 a 1000 Mb/s. Si bien el mecanismo de control y acceso al medio CSMA/CD funciona bien, debido a la demanda de velocidad del tráfico, se ha ido reemplazando por técnicas de conmutación.

La Figura 2.11 muestra una red de conmutación. Esta red aumenta no solo el ancho de banda, sino que también aumenta el número de nodos, capaces de conectarse a la misma red local [23]. La ventaja de esta red se refleja en la reducción de retardos por propagación de los paquetes emitidos.

Conforme avanzan las redes industriales, el grado de servicio (QoS, por sus siglas en inglés) ofrecido por la red debe ser capaz de cumplir con los siguientes objetivos:

- Mejorar el soporte de trafico crítico

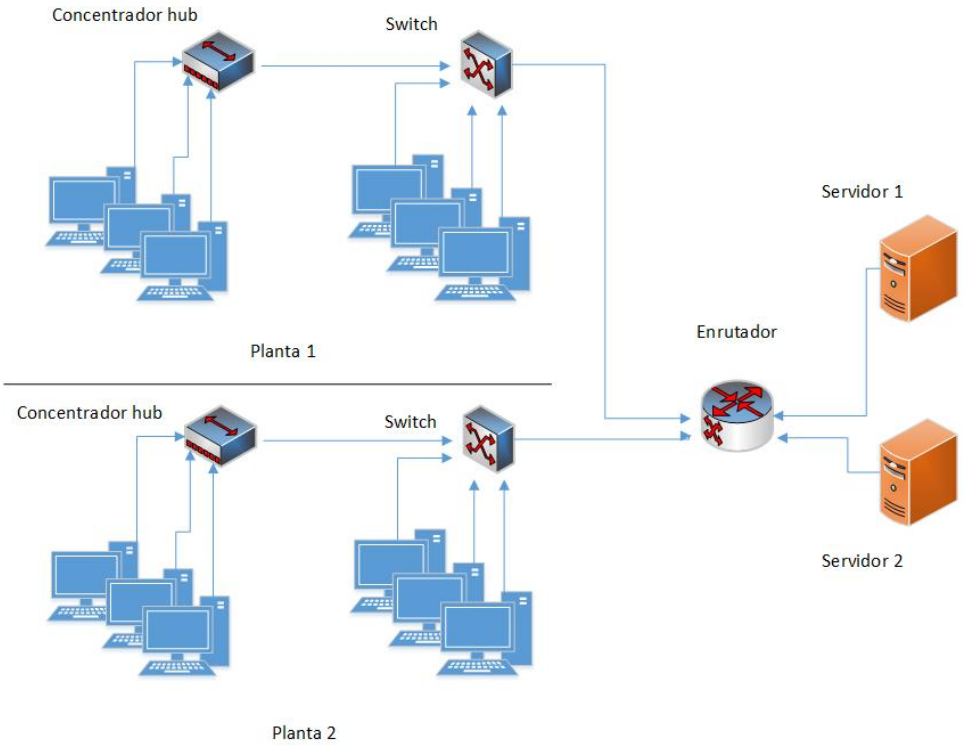


Figura 2.11: Conexión de equipos por medio de la red conmutada

- Limitar la propagación del tráfico multicast, mediante el establecimiento de prioridades de tráfico

La ventaja de una red industrial Ethernet, es que permite utilizar un gran número de protocolos tales como:

- Modbus TCP
- EtherNet/IP
- PROFINet
- EtherCat
- Powerlink
- Fieldbus high-speed Ethernet (HSE), entre otros.

EtherNet/IP y Profinet, son dos protocolos que trabajan a tiempo real. Mientras que EtherCat trabaja en tiempo real y síncrono. El resto de protocolos, no necesariamente trabajan a tiempo real, pero son eficaces al transferir información.

Los sistemas modernos industriales basados en Ethernet usan conectores RJ-45. Sin embargo un cable de par trenzado no blindado (UTP, por sus siglas en inglés), no puede utilizarse en este ambiente, debido a interferencia electromagnética.

Inicialmente se usaban los cables tipo D. Pero con el tiempo fue reemplazado por el IEC61076 [22], el cual describe el funcionamientos de conectores circulares M12. Estos conectores constan de conectores fijos y libres, con fijación por tornillo. Típicamente estos conectores dentro de una red industrial usan cables de 8 pines con conectores rosca. Estos conectores ofrecen mecanismos de protección contra corrosión, radiación y temperaturas extremas (-40 - 75 $°C$)

2.4. Protocolo *object linking and embedding* (OLE) *for process control* (OPC)

La tendencia actual, es integrar y facilitar la operatividad de los sistemas de producción y de sus componentes de manufactura. Con la finalidad de conseguir este objetivo, Microsoft creó la plataforma estándar de comunicación denominado OPC.

Antiguamente, para acceder a los datos de un equipo industrial, se debía adquirir el software adecuado para ese equipo. La comunicación y el traspaso de información entre dispositivos era totalmente nula debido a problemas de software propietario [24], esto llevó a otros conflictos como:

- Conflictos entre drivers de distintos fabricantes
- Cambios de hardware ocasionaban conflictos en software
- Conflictos al acceder al hardware de un dispositivo, cuando dos paquetes de software se usan al mismo tiempo

La Figura 2.12 muestra los problemas de una red industrial antes de la llegada de la comunicación OPC.

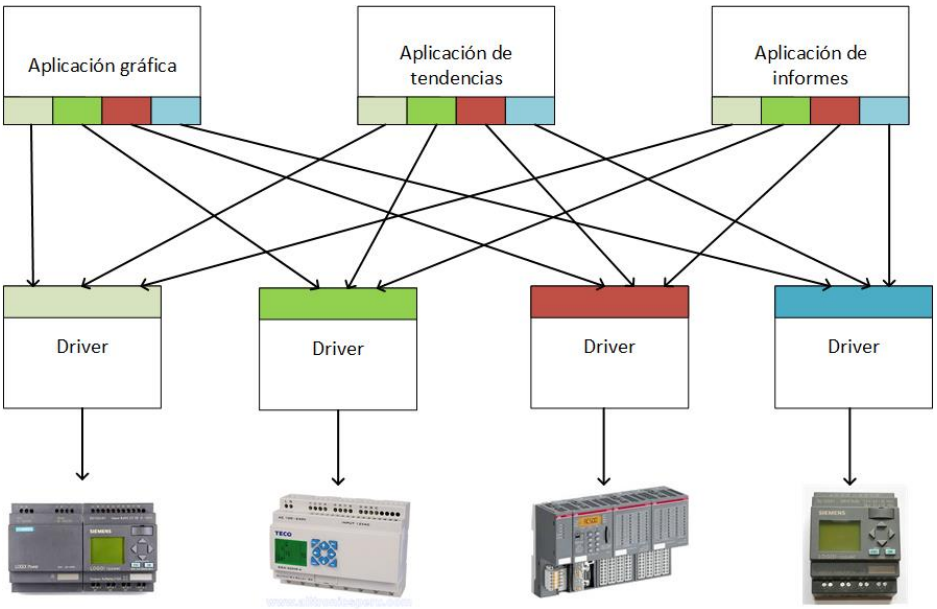


Figura 2.12: Red industrial antes de OPC

OPC clásico se basa en las tecnologías COM y DCOM de Microsoft. Con la estandarización de los protocolos, algunos de los desarrolladores más importantes, contribuyeron para la creación de un solo software denominado OPC Server. El nuevo servidor, es capaz de obtener el valor de las variables de los PLCs y almacenarlo. Esto posibilita a que cualquier cliente basado en este protocolo pueda hacer uso de dichos datos, tanto en lectura como escritura.

2.4.1. Desventajas de la tecnología COM/DCOM

Algunas de las desventajas de usar la tecnología COM/DCOM son:

- No existía independencia de la plataforma de Windows, lo que impedía a los creadores de software emplear otra plataforma en sus diseños
- El manejo de los paquetes generados por DCOM, presentan un grado de complejidad en su manejo. A pesar de que está diseñado para aplicaciones sobre internet, son difícilmente enviados debido a problemas de Firewall [4]
- La coordinación y manejo de los diferentes servidores OPC a menudo se tornaba compleja

2.4.2. Ventajas de OPC

Entre las ventajas mas importantes en OPC se destacan:

- Antes de OPC el protocolo empleado para comunicaciones industriales fue el protocolo dinámico de intercambio de datos (DDE, por sus siglas en inglés). Pese a ello, este no tenía características de seguridad robustas, no era confiable, ni flexible
- OPC ofrece alta flexibilidad y funcionabilidad a bajo costo
- El número de servidores, así como de clientes es extenso, lo que permite al usuario escoger el que mejor se ajuste a sus necesidades
- Cuenta con el soporte de la OPC Foundation. Dispone de soporte suficiente para ayudar al usuario a encontrar especificaciones referentes al protocolo OPC
- El cliente no necesariamente adquiere datos de un mismo servidor, sino que lo puede hacer de varios e incluso de diferentes fabricantes y al mismo tiempo

2.4.3. Variaciones del protocolo OPC

Dependiendo del tipo de aplicación y de los procesos manejados industrialmente, se han creado diferentes variaciones del protocolo OPC. Se destacan: OPC de acceso de datos (DA, por sus siglas en inglés), OPC de Gestión de Alarmas, OPC de Acceso a Datos Históricos y el más reciente OPC de arquitectura unificada (UA, por sus siglas en inglés). Básicamente un servidor tiene la capacidad de tomar datos a nivel de procesos de planta, proporcionando una interfaz al objeto conectado, mas no lo controla.

Servidor OPC de acceso de datos (OPC DA)

Este tipo de servidor define un conjunto de estándares, tales como el protocolo COM, Objetos, métodos y propiedades. Los datos obtenidos procedentes de diferentes fuentes, están disponibles mediante la estructura de espacio de nombres del servidor [25].

Este servidor presenta la siguiente arquitectura: un servidor propiamente dicho, el cual mantiene información relevante sobre el mismo, grupos, dentro de los cuales almacena las variables o ítems de cada equipo conectado y finalmente las etiquetas, que son colocadas como referencias a las variables almacenadas en

equipos como los PLCs.

Para observar los datos que contiene un PLC en la industria, el servidor crea en primera instancia un objeto, el cual contiene toda la información requerida. El mecanismo para el intercambio de datos es mediante llamadas síncronas, llamadas asíncronas, mensajes de actualización y suscripciones. Cuando el cliente realiza una llamada síncrona, espera a que el servidor envíe los datos, a diferencia de una llamada asíncrona. Los mensajes de actualización y suscripción llaman a métodos establecidos en el servidor, donde especifican el nodo del cual desean obtener los datos.

Servidor OPC de alarmas y eventos

A diferencia del servidor anterior, este no hace visible las variables o ítems almacenados, sino que manifiesta su control mediante alarmas. Estas se activan al ocurrir un proceso fuera del rango de operación normal. Para que se genere una alarma, el servidor dispone de eventos o condiciones que se asocian a cada variable, las mismas que son clasificadas por rango de importancia (HighAlarm, Normal, LowAlarm).

Para llevar a cabo la transferencia de datos, el servidor crea un Objeto del tipo OPCEventServer. La calidad y el tiempo de procesamiento de un evento o alarma depende de los criterios de filtrado que disponga el cliente. El filtro se lo realiza mediante el tipo de evento, por prioridad o la fuente del evento.

Servidor de datos históricos

La mayoría de las acciones tomadas en un proceso industrial se basan en predicciones. Son el resultado del análisis de datos históricos procedentes de sus dispositivos conectados. Su base de datos es accesible mediante protocolos OPC, a diferencia de bases de datos tradicionales. Al utilizar la interfaz DCOM desarrollada por Microsoft, aplicaciones como MS office también puede acceder a este tipo de servidores.

El servidor permite al cliente obtener dos tipos de datos: datos sin procesar y datos ya almacenados. De igual forma dependiendo de las necesidades de la

industria se pueden implementar dos tipos de servidores:

- Un servidor simple, el cual solo implementa una interfaz para el acceso de datos históricos
- Un servidor de análisis y comprensión de datos, el cual procesa los datos y analiza valores como promedios, valores mínimos, máximos, adición de anotaciones, entre otros

A diferencia de un servidor OPC DA, aquí no se maneja el concepto de objeto. El cliente puede acceder a cualquier variable mediante la dirección del ítem en el servidor.

Servidor OPC de arquitectura unificada (OPC UA)

Es la versión más reciente de los servidores OPC. Integra características destacables de sus antecesores, además de un gran cambio en su arquitectura. No utiliza la interfaz DCOM para su comunicación con otros sistemas.

Antes de este cambio, los servidores solo podían ejecutarse en equipos con sistema operativo de Microsoft Windows. Se abre un abanico de posibilidades para nuevos dispositivos y nuevas formas de interacción de datos. La nueva forma de comunicación se realiza mediante protocolos HTTP y SOAP. Al ser un servidor nuevo, existen temas relacionados a seguridad que todavía faltan por ser mejorados, sin embargo, sus bondades desvelan desarrollos importantes para el mismo.

Las nuevas características integradas en OPC UA son:

- OPC UA provee un espacio de dirección y modelo de servicio consistente e integrado. Permite a los servidores reunir capacidades de acceso a datos, alarmas, datos históricos en su propio espacio de dirección. Para acceder a estos datos, el cliente usa el mismo conjunto de servicios [1]
- OPC UA permite al servidor proveer al cliente objetos definidos por su tipo
- Añade una nueva jerarquía entre nodos. Brinda nuevas relaciones entre los mismos nodos en lugar de limitarse a la jerarquía simple de sus antecesores

- Soporta una nueva estructura de datos. No utiliza la simple organización jerárquica de ítems, sino que ofrece meta-información de sus nodos. El cliente no necesita entender el tipo de datos que maneja, ya que le basta con analizar la descripción del nodo seleccionado
- La OPC Foundation anteriormente ofrecía soporte SDK en C/C++, C# y Java. Sin embargo, este soporte se ha extendido a otros lenguajes de programación, por lo que el desarrollo de OPC UA se lo puede realizar desde Windows, Linux, Mac OS, o sus derivados
- OPC UA está diseñado para soportar una gran gama de servidores. OPC define un conjunto de características, pudiendo los servidores implementar solo un subconjunto de las mismas
- Presenta un mecanismo de seguridad mejorado. OPC clásico no tenía claramente definido su mecanismo de seguridad. OPC UA define sus propios mecanismos de seguridad en el canal de comunicación. Utiliza la autentificación de certificados, además de la utilización de una clave pública
- Al unificar características importantes de sus antecesores, los datos actuales y los datos históricos pueden ser almacenados en variables, en una misma forma. Asimismo, los comandos de control para los dispositivos conectados pueden llevarse a cabo mediante métodos generados en el mismo servidor

Espacio de direcciones

Es un concepto nuevo e importante dentro de OPC UA. Todos los bloques o dispositivos conectados se deben almacenar en un solo modelo jerárquico. Dicho modelo de almacenamiento de datos puede ser establecido por el propietario del servidor de acuerdo a sus necesidades. En este sentido, cada objeto conectado es tratado como un nodo dentro del espacio de dirección como se muestra en la Figura 2.13.

En el espacio de direcciones, a cada nodo se le asigna atributos y referencias. Los atributos describen el nodo y las referencias hace mención a las relaciones existentes de dicho nodo con otros.

La Tabla 2.3 describe los atributos de un nodo. Estos atributos son parte de la meta-data incorporada en esta especificación del protocolo OPC. Todos los nodos

Figura 2.13: Espacio de direcciones de un servidor OPC UA [1]

Tabla 2.3: Atributos públicos de una nodo [4]

Atributo	Descripción
NodeId	Es el identificador del nodo dentro del espacio de dirección del servidor
NodeClass	Define la categoría a la que pertenece el nodo.
BrowseName	Identifica al nodo por su nombre en lugar de su NodeId dentro del espacio de dirección
DisplayName	Contiene el nombre del nodo, el mismo que es usado por el cliente para identificarlo
Description	En algunos servidores, se permite la asignación de una pequeña descripción para el nodo
WriteMask	Especifica que atributos son de escritura
UserWriteMask	Especifica que nodos pueden ser modificados por el usuario o cliente.

generados a partir del nodo base, pueden ser creados y eliminados a excepción del nodo padre sobre el cual se monta el espacio de dirección y hereda determinados atributos a los nodos generados.

A excepción de los atributos, las referencias no hacen mención a la relación entre el nodo padre y el nodo seleccionado. Una referencia hace mención a las relaciones existentes entre nodos de un mismo nivel, es decir, de los nodos hermanos. Estas son solo accesibles desde el nodo seleccionado y no desde el espacio de dirección del servidor.

Las referencias empleadas en cada nodo se basan en ReferenceType, las cua-

les son utilizadas a manera de atributos para describir un nodo. Dentro de las especificaciones OPC UA, se describen tres tipos importantes de ReferenceType: HasComponent, Organizes y HasSubtype. HasComponent es utilizada para definir los componentes contenidos por objetos. Organizes es usado para definir la jerarquía del directorio. Finalmente, HasSubtype es usado para definir el tipo de herencia de un nodo, es decir, se lo utiliza para crear nuevos tipos de ReferenceType [4].

Dentro del espacio de direcciones, se encuentra el objeto, el cual tiene una referencia del tipo HasComponent. Un nodo objeto almacena las variables creadas por el servidor y las almacena como atributos del mismo. Cada variable tiene atributos DataType, los cuales indican la naturaleza de los datos almacenados. Entre los DataType, están: variables del tipo Int32, *boolean, double* entre otros. Además de las variables, un objeto puede también contener métodos, los cuales pueden ser utilizados desde el cliente en cualquier momento.

2.4.4. Formas de migración de OPC clásico a OPC UA

Con la finalidad de permitir una fácil migración del OPC clásico a la nueva infraestructura de OPC UA, se han desarrollado estrategias para migrar de un protocolo de comunicación a otro, como:

- ***Wrappers***

  Son utilizados a manera de puente para conectar un sistema a otro. Permiten a un cliente OPC UA acceder a un servidor OPC clásico. Un componente *wrapper* consta de un cliente OPC clásico y de un servidor OPC UA, como se muestra en la Figura 2.14. Estos dos sistemas funcionan al mismo tiempo, permitiendo al cliente OPC UA acceder a variables contenidas en otro sistema de manera casi inmediata.

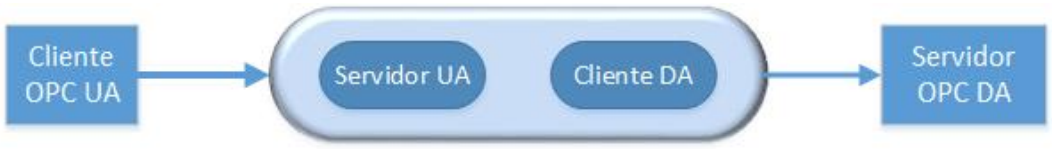


Figura 2.14: UA Wrapper para un Servidor DA

- ***Proxies***

  Un *proxy* permite que un cliente OPC clásico pueda acceder a los datos de un servidor OPC UA. Está compuesto por un cliente UA en una de sus interfaces y un servidor OPC clásico en la otra, como se muestra en la Figura 2.15.

Figura 2.15: DA COM Proxy para un Cliente DA

Estos tipos de componentes han sido integrados en uno solo. Sin embargo, su uso está restringido a determinados productos, ya que el procesamiento de datos debido al software adicional crea inconvenientes en los sistemas. La sobrecarga se debe a configuraciones adicionales, además del mantenimiento necesario.

Limitaciones de la migración de OPC clásico a UA

Dependiendo del tipo de comunicación que se desea establecer se puede usar *wrappers o proxies*, los cuales presentan sus ventajas e inconvenientes al mismo tiempo. Cabe destacar que en ninguno de los casos se integra totalmente las características definidas por el estándar OPC UA.

La utilización de un *UA wrapper* es una buena alternativa si se desea acceder de manera rápida a un servidor OPC clásico. Pese a ello, presenta inconvenientes. El primero es que para que el cliente pueda acceder a los datos del servidor clásico necesita estar siempre conectado al *UA wrapper*. El segundo inconveniente es que se necesita la implementación de un *UA wrapper* por cada servidor DA. El problema es aún mayor si se considera que un cliente OPC UA no puede almacenar en su mismo espacio de dirección dos dispositivos diferentes, ya que cada *UA wrapper* genera su propio espacio de dirección [1].

Como alternativa a este inconveniente, se opta por la utilización de un segundo servidor UA, el cual accede a los datos de los *UA wrapper* y vuelve a almacenarlos en su propio espacio de dirección para que estos puedan ser accedidos por

cualquier cliente, como se muestra en la Figura 2.16.

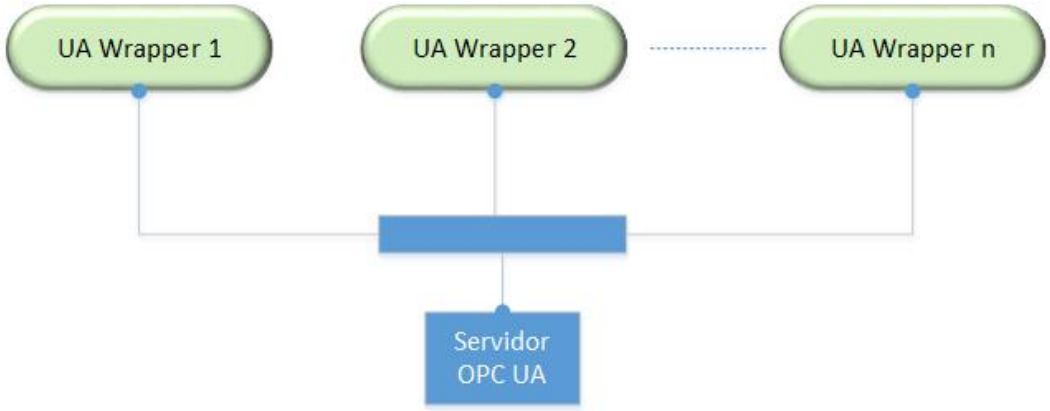


Figura 2.16: Conexión de un Servidor OPC UA con UA Wrappers

2.4.5. Posibles aplicaciones de un servidor OPC UA

A pesar de que OPC UA, nació como una solución al eminente crecimiento de las redes industriales, gracias a la estandarización del protocolo, se ha abierto una puerta para posibles fututas aplicaciones, como es el caso de las redes inteligentes (Smartgrids).

Al hablar de una red inteligente, se habla de tecnologías de comunicación. Incorpora sistemas de administración de energía (EMS, por sus siglas en inglés) y sistemas de distribución de energía (DMS, por sus siglas en inglés). El constantemente intercambio información, hace que esta tarea pueda tornarse compleja. Para manejar su infraestructura, un sistema de control es indispensable en esta área.

La comunicación OPC UA está definida como un servicio abstracto. Facilita a nuevas tecnologías hacer uso del mismo para intercambiar información. Dicho intercambio se lleva a cabo mediante el mapeo de una tecnología a otra [26]. El mapeo no solo permite escoger la codificación de la información, si no también escoger el protocolo de transporte. Si se desea establecer una comunicación sobre internet, se puede usar protocolo de acceso simple de objetos (SOAP, por sus siglas en inglés), basado en servicios web existentes. La información también puede codificarse en lenguaje de etiquetado extensible (XML, por sus siglas en

inglés) o simplemente usar el formato binario establecido por defecto en OPC UA.

2.4.6. Seguridad en OPC UA

En cualquier sistema de comunicación, el tema de la seguridad es un aspecto importante a considerar. Consta de parámetros claves como la autentificación, autorización, confidencialidad, integridad y disponibilidad.

La Figura 2.17 muestra el modelo de seguridad para los objetos creados, definido por OPC UA. El servidor y el cliente negocian funcionabilidades de seguridad y crean un canal seguro. El canal establecido debe ofrecer encriptación para mantener la confidencialidad, firmas, para mantener la integridad y certificados para asegurar la autentificación entre las aplicaciones [2].

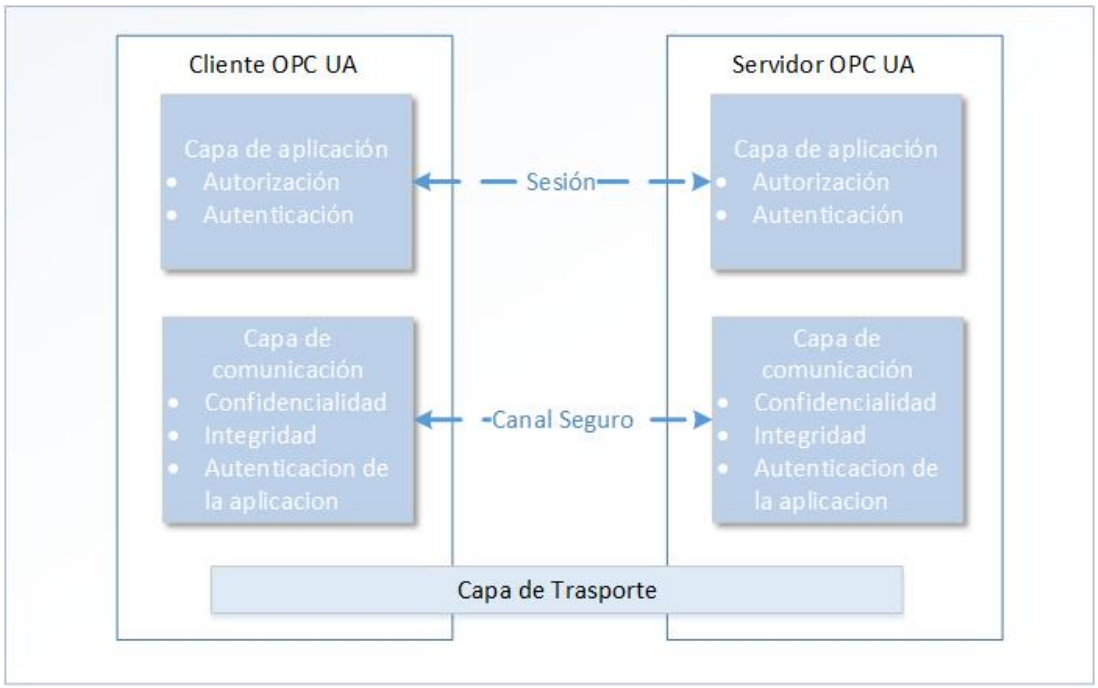


Figura 2.17: Modelo de seguridad definido por OPC UA [2]

Las funciones de seguridad del usuario son manejadas por la capa de aplicación. Esta a su vez esta manejada por los servicios de sesión. Los parámetros clave dentro de la seguridad se los puede conseguir si y solo si se sigue el esquema planteado por el protocolo OPC UA.

Dependiendo del tipo de aplicación empleada, los requerimientos de seguridad también son distintos, a esto se le suma el tipo de red sobre la cual se está trabajando. Por tal motivo la OPC Foundation ha establecido deferentes niveles de seguridad para cada tipo de comunicación, como se muestra en la Tabla 2.4.

Tabla 2.4: Clasificación de la seguridad en OPC UA [2]

Clase	Condiciones			Medida de seguridad
	Red	Información	Eficiencia de Comunicación	
0	L	R	N	N o AA
1	L	R, W	N	AA, UA y A
2	I	R	H	AA, UA y S
3	VI	SR, W	N	AA, UA, S y A
4	I o VI	SR, W	L	AA, UA, S, A y E

Donde:

- La red: tiene diferentes clases, ya sean, L para redes de área local, I para internet, VI para redes privadas virtuales
- La información: R para lectura, W para escritura, SR para lectura susceptiva, SW para escritura susceptiva
- Eficiencia de comunicación: N para no requerida, L para una baja tasa de comunicación requerida, H para una tasa de comunicación alta requerida
- Medida de Seguridad: N para una medida de seguridad nula, AA para Autenticación de la Aplicación, UA para Autenticación de Usuario, A para revisión (Audit), S para firma (Signature) y E para encriptación

La Tabla 2.4 es relevante para los desarrolladores, ya que dependiendo del tipo de aplicación que se desea implementar, se puede identificar qué puntos de seguridad deben cumplir.

Capa de seguridad del canal

La capa de comunicación, que maneja la seguridad del canal es la parte más importante al momento de implementar una comunicación. Cuando un mensaje es recibido, este es decodificado y lo analizado para determinar a qué tipo pertenece. Si el mensaje es una petición de servicio, el módulo del manejo de canales,

configura un canal seguro. El canal generado es almacenado con las medidas de seguridad correspondientes, además de los mecanismos de encriptación. Luego de crear la conexión, si los siguientes mensajes hacen referencia a la conexión establecida, se procede a validar la firma del mismo. Si la firma es válida, entonces se pasa el mensaje a la capa de sesión.

Capa de sesión

El mensaje recibido es direccionado a la aplicación correspondiente, en donde nuevamente se analiza y se establece la sesión correspondiente. Al crearse una nueva sesión, se le asigna a la misma un identificador y se lo almacena. Si la información del mensaje es una petición de servicio de OPC UA, entonces la información requerida es obtenida del espacio de direcciones del servidor y se responde a la petición.

Capítulo 3

Diseño y construcción del prototipo

3.1. Requisitos del modelo

OPC presenta la mejor alternativa de conectividad en aplicaciones a nivel industrial. Por tal motivo, se propone la implementación de un servidor OPC UA básico. Este modelo de comunicación consta de tres etapas (dispositivos en red, servidor OPC UA, cliente OPC UA) para tener un correcto funcionamiento.

La Figura 3.1 muestra las tres etapas que posee el modelo de comunicación OPC.

Figura 3.1: Diagrama del modelo de comunicación OPC

- **Dispositivo en red**

  El objetivo de una red industrial es monitorear constantemente los procesos que se cumplen en planta. Por ello, se debe tener conectado los dispositivos (PLCs) a la red, con el fin de ofrecer al usuario la información generada.

- **Servidor OPC UA**

  El servidor OPC UA es el dispositivo encargado en acceder a los valores de las variables disponibles en cada PLC dentro de la red. Estas variables dependen del procesamiento y función que se asigne por parte de dicho servidor. Adicionalmente, una de las ventajas del servidor OPC UA, es la

facilidad de implementarse en diversas plataformas. Razón por la cual, se ha decidido implementar el servidor OPC UA en la plataforma Raspberry Pi.

- **Cliente OPC UA**

  El cliente OPC UA es el objeto que se encargará de recibir la información del servidor. Para ello, es necesario que el cliente que funcione mediante el mismo protocolo de comunicación. La ventaja de llevar a cabo la propuesta, es que no es necesario la utilización de un cliente determinado. El usuario puede utilizar cualquier cliente OPC, licencia comercial o libre.

3.2. Requerimientos del prototipo

El desarrollo de esta investigación propone la implementación de un servidor OPC UA en una plataforma embebida de bajo costo para expandir las funcionalidades de dispositivos de automatización (DEFA). La funcionalidad de DEFA es interconectar la capa 1 con la capa 2 de la pirámide CIM, permitiendo desarrollar aplicaciones de supervisión y control.

El servidor OPC UA trabaja en redes de área local, redes corporativas e incluso mediante internet, sin embargo, el proyecto será implementado dentro de una red de área local. Las siguientes consideraciones se llevan a cabo para implementar el servidor OPC:

- **Comunicación PLC – Servidor**

  El prototipo propuesto tiene como objetivo el manejo de las variables internas que de un PLC. Para lograr este cometido se debe crear la interfaz de comunicación del PLC con la plataforma Raspberry Pi.

  Al trabajar con dispositivos como PLCs, es necesario tomar en cuenta que los PLCs se configuran con una IP específica. Esta IP tiene la finalidad de identificar cada PLC en la red que se encuentran conectados.

  La IP por defecto de un PLC está configurado en el rango de 192.168.0.1/255. Esta configuración depende del fabricante, sin embargo, se puede realizar el

cambio de IP manualmente. Al realizar cualquier cambio se debe considerar que no afecte a otros dispositivos de la red. Adicionalmente, en este rango se encuentran configurados IPs de otros dispositivos de interfaz con el usuario (HMI, por sus siglas en inglés).

La Figura 3.2 muestra el bosquejo general de la comunicación entre PLC y Raspberry Pi. Siendo Snap7 el driver de interconexión de estos dispositivos.

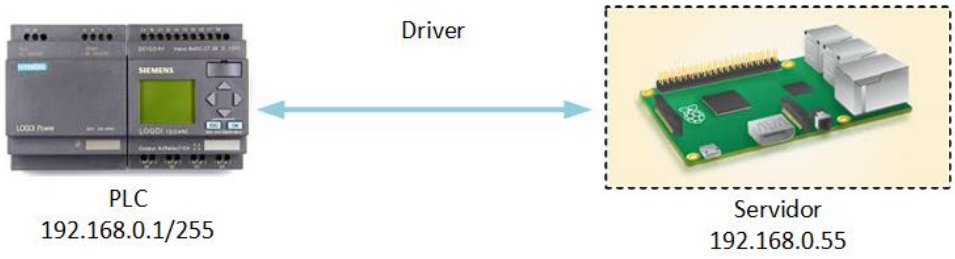


Figura 3.2: Diagrama del modelo de comunicación OPC

- **Comunicación Servidor – Cliente**

  La comunicación entre cliente y servidor OPC UA ofrece al operario toda la información acerca de los actuadores, sensores, etc., conectados a los PLCs.

  Uno de los requisitos para que exista una correcta comunicación entre cliente y servidor, es la habilitación de los puertos 4840 o 4841 por parte del cliente. Esto es necesario, ya que son definidos por el protocolo OPC UA. En algunos casos, cuando el cliente está trabajando en un entorno de Windows, es recomendable añadir excepciones y habilitar los puertos. Caso contrario el Firewall bloquea las peticiones del cliente.

  Una de las ventajas de este tipo de conexión, es el enlace con múltiples usuarios a la vez. El servidor debe aceptar diferentes peticiones de manera simultánea.

  Los usuarios conectados pueden estar trabajando con un cliente OPC UA instalado en diferentes sistemas operativos. El único requisito del usuario para conectarse al servidor, es tener una IP definida dentro del dominio de la red local.

La Figura 3.3 muestra la conexión de un servidor OPC UA con varios clientes OPC UA al mismo tiempo.

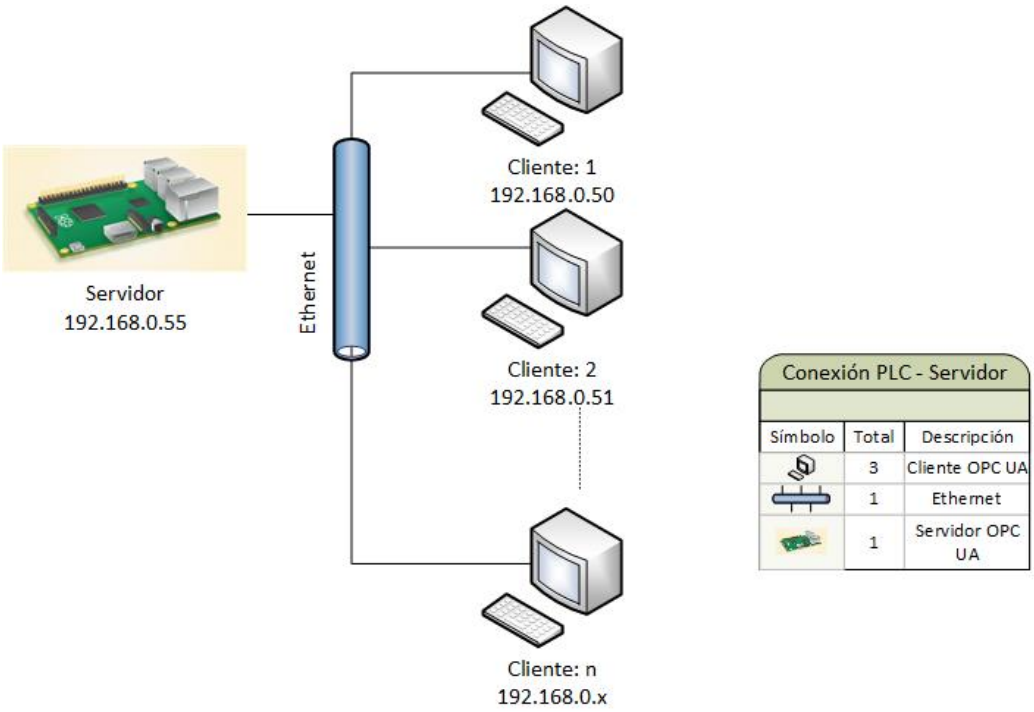


Figura 3.3: Comunicación Servidor – Cliente

- **Comunicación Servidor – Base de Datos**

  En el modelo propuesto se encuentra integrada una base de datos local. La base de datos guarda las variaciones que entregan los dispositivos conectados al PLC en forma de variables. Esta información asiste al operario encargado informando el comportamiento que tiene el PLC con los dispositivos conectados.

  La base de datos entrega un informe o historial sobre el comportamiento que tuvieron las variables en un tiempo establecido. Al evaluar este historial se puede prevenir futuros fallos o corregir errores que se producen en los procesos encargados al PLC.

  La comunicación que posee el servidor OPC UA y la base de datos es a través de librerías. Las librerías proporcionan los métodos necesarios para guardar, escribir, leer y editar la base de datos local implementada.

Las librerías implementadas en el servidor se basan en un lenguaje de consulta estructurado (SQL, por sus siglas en inglés). SQL es un lenguaje explicativo sobre el acceso a base de datos relacionales, permite declarar diferentes tipos de operaciones en la base de datos. La ventaja de utilizar SQL como lenguaje gestor, es el manejo interno de álgebra y cálculo relacional para realizar consultas y cambios en los datos requeridos.

La Figura 3.4 explica la conexión general entre el cliente OPC, servidor OPC, PLCs y la base de datos que gestiona el servidor OPC.

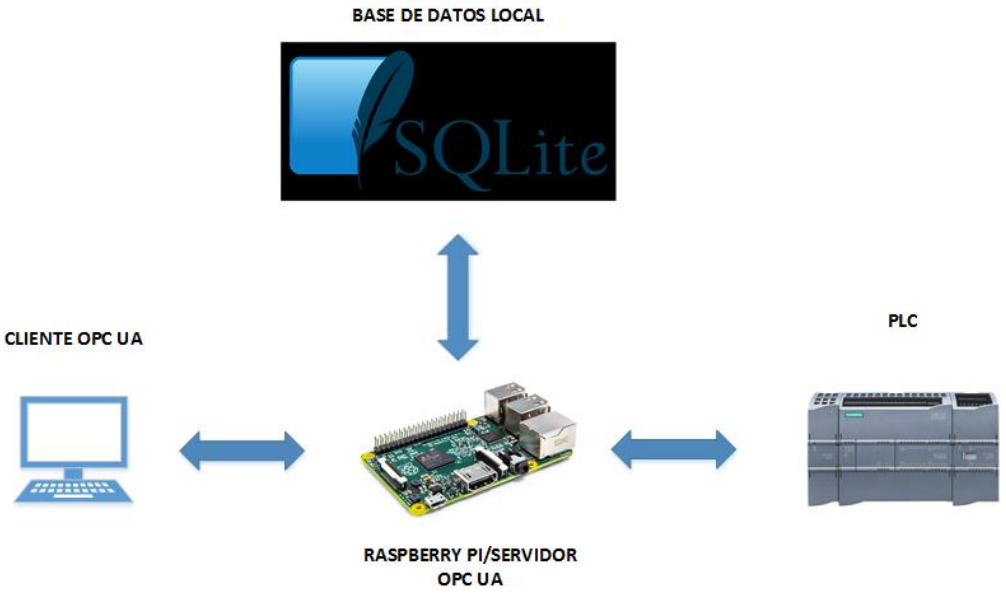


Figura 3.4: Diagrama de conexión con la base de datos

3.3. Diseño de la interfaz

Uno de los lenguajes con mayor acogida actualmente es Python. Permite crear y ejecutar aplicaciones en múltiples plataformas. Cuenta con la participación de numerosos desarrolladores de software, razón por la cual el número de módulos y librerías disponibles, son extensas. Para el desarrollo del servidor se ha visto conveniente utilizar la versión de Python 2.7. La ventaja de esta versión es su estabilidad en comparación con nuevas versiones.

3.3.1. Interfaz Raspberry Pi – PLC

Para poder comunicar un PLC con la Raspberry Pi, es necesario construir una interfaz de comunicación. Para este caso se ha visto conveniente la utilización de la librería Snap7, como se muestra en la Figura 3.2.

Snap7 es una librería de código abierto y multiplataforma. Distribuida bajo la licencia pública general reducida (LGPL, por sus siglas en inglés). Capaz de ejecutarse en plataformas de 32 y 64 bits, ideal para plataformas embebidas como Raspberry Pi. Permite la comunicación Ethernet con dispositivos nuevos de PLCs, así como versiones antiguas de Siemens.

Protocolo S7

El protocolo S7 es la estructura principal de las comunicaciones de dispositivos Siemens. Su implementación está normalizada por la ISO TCP (RFC1006).

Este protocolo está orientado a funciones o comandos. Cada transmisión o respuesta tiene un comando. Un comando consiste de los siguientes parámetros:

- Cabecera
- Un conjunto de parámetros
- Datos de parámetro
- Un bloque de datos

Establecimiento de la conexión (Raspeberry Pi - PLC)

Para acceder a un dato del PLC la librería implementa el método *connect*. Este método recibe como parámetro, la IP del PLC, el slot, como se muestra en la Figura 3.5.

La Tabla 3.1 muestra las características comunicación de PLCs mas utilizados.

Adquisición de las variables del PLC

El intercambio de información se la realiza por bloques. Para llevar a cabo esta tarea, la librería ha implementado el método *read_area*. Este método recibe

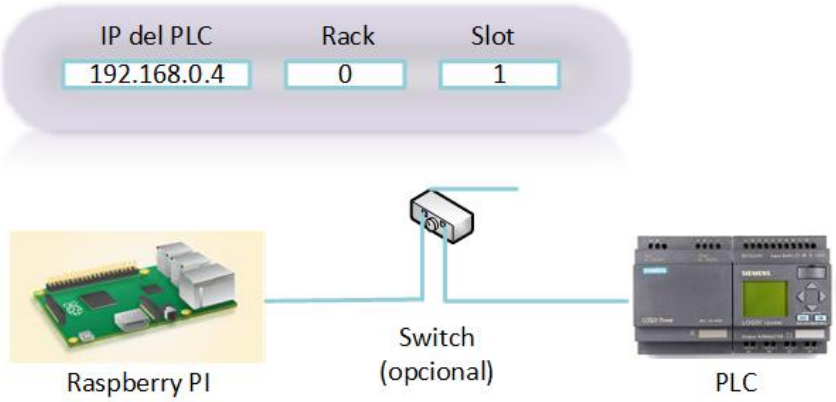


Figura 3.5: Conexion Raspberry - PLC

Tabla 3.1: Parámetros de comunicación de los PLCs s200, s300, s400, s1200 [5]

	Descripción	S7-200	S7-300/400/1200
Tipo de comunicación	Define la forma física de comunicación	CP243-Ethernet	Ethernet ISO TCP Standard
Rack (0...7)	Define el número de bastidor, donde esta alojada la CPU para la comunicación.	valor por defecto 0	valor por defecto 0
CPU Slot (0...31)	Define el número de ranura donde se encuentra la CPU para la comunicación	valor por defecto 2	valor por defecto 1
Tipo de conexión	Define el tipo de conexión	1.- Modo programación 2.- Modo de operación 3.- Otro tipo de conexión	1.- Modo programación 2.- Modo de operación 3.- Otro tipo de conexión

como parámetro el área, la ubicación de la variable, y la longitud del dato a leer.

La Tabla 3.2 muestra el valor hexadecimal del área de memoria de un PLC.

Al trabajar con PLCs de la marca Siemens, el tipo de datos que se pueden obtener se muestran en la Tabla 3.3.

Con el método anterior se puede obtener cualquier tipo de variable programada en el PLC. No obstante, esta tarea puede resultar mas fácil, mediante el uso de una segunda librería creada por otro desarrollador. La librería se llama S71200.

La ventaja de esta librería es que recibe un solo parámetro. Dicho parámetro es la concatenación de nomenclatura del área, la longitud del dato, ya la variable a leer. El dato concatenado es de la forma *Mx0.0*.

Escritura de las variables del PLC

Tabla 3.2: Area de memoria de un PLC [5]

Valor del área de un PLC (hexadecimal)	Descripción	Nomenclatura convencional
0x81	Datos digitales de entrada	I
0x82	Datos digitales de salida	Q
0x83	Locación en memoria o etiqueta, tanto para lectura como escritura	M
0x84	Bloque de datos	DB
0x1C	Temporizador	T
0x1D	Contador	C

Al igual que el caso anterior la librería de Snap7 implementa el método *write_area*. Este método recibe como parámetro el área, la ubicación de la variable y un puntero con el tamaño de la variable.

De igual forma, la librería S71200 incorpora un método que facilita la tarea de escritura. El método se llama *writeMem*. Recibe como parámetro el valor concatenado del área, la longitud de bits a escribir y la variable, ademas de una segundo parámetro que es su nuevo valor. Sus parámetros tienen la forma *Mx0.1, False*.

3.3.2. Interfaz Raspberry Pi – Computador (Servidor/Cliente)

La estandarización del protocolo OPC UA, ha permitido el desarrollo de librerías basadas en él. Este el caso se las librerias FreeOpcUA, las cuales son públicas a través de la plataforma GitHub. Su código fuente está escrito totalmente en Python. Distribuidas bajo la licencia LGPL.

Las librerías han sido probadas con clientes y servidores comerciales. Ofrece una interfaz de fácil intercambio de estructuras definidas UA. Cuenta con clases de alto nivel, permitiendo al desarrollador leer y escribir objetos en el servidor.

Clase Server

Tabla 3.3: Longitud del dato a leer/escribir de un PLC

Nomenclatura convencional	Tipo	Número de Bytes
X	Bit, Booleano	1 bit
B	Unsigned Byte	1-byte
W	Unsigned Byte	2-byte
INT	Signed Word	2-byte
D	Unsigned Double Word	4-byte
DINT	Signed Double Word	4-byte
Bbcd	Unsigned Byte en código BCD	1-byte
Wbcd	Unsigned Word en código BCD	2-byte
Dbcd	Unsigned Word en código BCD	4-byte
REAL	Real Number	4-byte
LREAL	Long Real Number	8-byte
CHAR	Signed Byte	1-byte
STRING	S7 string	1-byte
DTL	Date and Time	12-bytes

Con la finalidad de facilitar el uso de las librerías, los desarrolladores han implementado la clase *Server*. La Figura 3.6 muestra los atributos y métodos de la clase.

Los atributos de la clase Server son los siguientes:

- **endpoint:** Dentro de esta variable el servidor almacena el identificador de recursos uniforme (URL, por sus siglas en inglés), mediante la cual un cliente accederá al mismo, ejemplo "$opc.tcp : //192{,}168{,}0{,}55 : 4841/freeopcua/server/$"
- **aplication_uri:** Es el identificador de recursos uniforme (URI, por sus siglas en inglés)
- **product_uri:** Es la variable en donde guarda el nombre de las librerías empleadas en FreeOpcUA
- **name:** Es la variable en donde se guarda el nombre del servidor
- **default_timeout:** Es la variable que almacena el tiempo de inactividad del servidor
- **certificate:**: En esta variable se almacena el certificado de seguridad del servidor

Figura 3.6: Clase Server de la librería FreeOpcUA

- **private_key:** En esta variable, el servidor almacena su llave primaria como mecanismo de seguridad

- **_policies[]:** Al iniciar el servidor se guardan las de seguridad del servidor en un lista o *array*

- **iserver:** Es una variable del tipo IntenalServer. Esta variable hereda los atributos de la clase antes mencionada. Mediante esta variable se crea el servidor propiamente dicho. la clase *InternalServer* se encuentra implementada internamente en unos de los módulos de las librerías

Los métodos de la clase son:

- **load_certificate:** Mediante este método, el servidor adquiere el certificado y lo almacena
- **load_private_key:** A través de esta función, el servidor adquiere una llave primaria y la almacena
- **disable_clock:** Este método es usado para realizar pruebas del servidor. Es útil cuando se desea que el servidor nunca se suspenda en las pruebas por falta de actividad
- **set_aplication:** Este método registra el URI del servidor. Esto permite que en el espacio de direcciones de la clase *InternalServer* no existan dos instancias del mismo servidor
- **find_servers:** Cuando una petición de conexión de un cliente llega al servidor, este método compara los parámetros de llegada con el servidor registrado en la clase *InternalServer*
- **register_to_discovery:** Mediante este método, la conexión de un cliente es renovada cada 10 minutos. Permitiendo al servidor saber que clientes se encuentran conectados durante este tiempo
- **unregister_to_discovery:** Este método se utiliza cuando la conexión con un cliente es finalizada
- **renew_registration:** Esta método trabaja conjuntamente con el método register_to_discovery
- **get_client_to_discovery:** Este método crea un cliente básico. Este cliente sirve para realizar pruebas del funcionamiento del servidor
- **allow_remote_admin:** Permite crear o eliminar nodos remotamente. Es necesario obtener permisos. Mediante este método una conexión remota adquiere permisos de administrador
- **set_endpoint:** Este método registra el URL del servidor en el atributo *endpoint*
- **get_endpoint:** Este método devuelve la URL del servidor
- **_setup_server_nodes:** Al inicial el servidor, los nodos son configurados y ubicados en el espacio de direcciones del servidor

- **_set_endpoints:** Este método trabaja conjuntamente con el método set_endpoint
- **set_server_names:** Este método registra el nombre del servidor. Por defecto el servidor tiene el nombre "$FreeOpcUaPythonServer$"
- **start:** Una vez que todos los parámetros del servidor han sido establecidos, mediante este método, el servidor es puesto en marcha
- **stop:** Es método es usado para detener el servidor
- **get_root_node:** Cuando se desea registrar un nodo, es necesario seleccionar el nodo raíz. Este método devuelve el nodo principal, sobre el cual el resto de nodos serán registrados
- **get_object_node:** Este método devuelve el nodo objeto. En este nodo se encuentra información del servidor propiamente dicho
- **get_server_node:** Este método devuelve el nodo servidor. En este nodo se encuentra información sobre los métodos implementados en el servidor. Sin embargo estos métodos son solo de lectura
- **get_node:** este método es implementado por el servidor. Este método devuelve un nodo a partir de su identificación
- **create_suscription:** Este método permite a un usuario suscribirse a un nodo. Esta suscripción permite conocer los cambios de un nodo en cuanto a sus valor
- **get_namespace_array:**Este método devuelve un listado de todo el espacio de direcciones del servidor
- **register_namespace:** Al registrar un nodo, este debe se debe almacenar en el espacio de direcciones del servidor para que sea accesible desde cualquier cliente. Este método toma como parámetro el identificador de un nodo y lo registra
- **get_namespace_index:** A diferencia del método get_namespace_array, este solo muestra el espacio de dirección de un nodo seleccionado
- **delete_nodes:** Este método sirve borra nodos del servidor. Sin embargo, este método lo puede hacer uno por uno o recursivamente. Al seleccionar la opción de borrado recursivo, es necesario enviar como parámetro una lista de nodos

3.3.3. Interfaz con el usuario

Con la finalidad de conseguir una interfaz amigable con el usuario, la interfaz gráfica ha sido diseñada en PyQt. Esta es una adaptación de la biblioteca de Qt para Python. Se encuentra disponible en la version v4 y v5. Sin embargo, por temas de compatibilidad con la Raspberry se escogió la version v4 de PyQt.

La interfaz de usuario implementa las siguientes clases:

A) **Clase V_principal:**

la clase V_principal es una instancia de la interfaz gráfica V_principal.ui creada en PyQt. Al iniciar el servidor, el usuario visualiza la interfaz que se muestra en la Figura 3.7, en donde el usuario puede constar la IP y el puerto sobre el cual un cliente va a acceder al servidor.

Figura 3.7: Interfaz V_principal

Para poder iniciar el servidor la clase V_principal implementa los atributos y métodos que se muestran en la Figura 3.8.

Los atributos de la clase se describen a continuación:

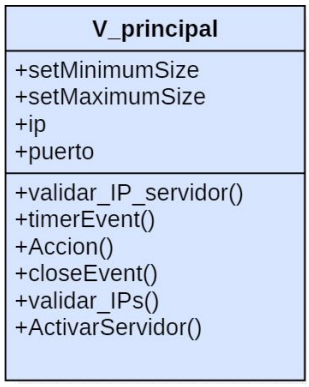


Figura 3.8: Clase V_principal

- **setMinimumSize:** Este atributo es heredado de la interfaz gráfica y permite establecer el tamaño mínimo de la ventana al iniciar el programa
- **setMaximumSize:** Permite establecer el límite máximo de la ventana principal.
- **ip:** Este atributo guarda la IP del servidor, para que cualquier método de la clase haga uso del mismo
- **puerto:** Los puertos definidos por OPC UA son el 4840 y 4841. El usuario puede escoger cualquiera de los dos, sin embargo, el valor por defecto es el 4841

Los métodos de la clase son los siguientes:

- **validar_IP_servidor:** Con la finalidad de evitar errores, esta clase valida el formato de ingreso de la IP del servidor
- **timerEvent:** Este método permite ejecutar una animación de estado de carga del servidor. La animación se encuentra en forma de barra de progreso de 0 a 100 % como se muestra en la parte inferior de la Figura 3.7. La barra de progreso llega a 100 % cuando el servidor se ha terminado de configurar y se encuentra en red
- **Accion:** Este método es implementado al seleccionar el botón *Conectar*. Configura parámetros correspondientes al método *timerEvent*. Al terminar su ejecución llama al método antes mencionado
- **closeEvent:** Su función es prevenir que el usuario cierre la conexión de servidor por accidente. Al generarse un evento *closeEvent*, este método genera una ventana informativa como se muestra en la Figura 3.9, de ser afirmativo la conexión del servidor es finalizada

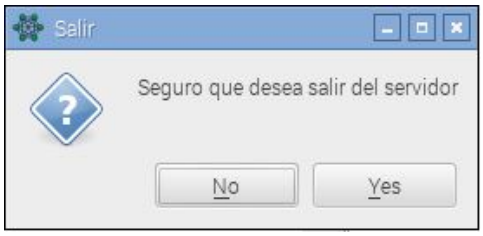


Figura 3.9: Ventana de cierre de sesión

- **validar_IPs:** Este método trabaja conjuntamente con el método *validar_IP_servidor*. Al verificar que la IP ha sido ingresada en el formato adecuado, se llama a la método *ActivarServidor*
- **ActivarServidor:** Una vez que todos los parámetros anteriores han sido configurados, este método finalmente llama a las librerías de FreeOpcUA y activa el servidor en la red

B) **detIp:**

Esta clase ayuda al usuario a determinar la IP del dispositivo sobre el cual se va a ejecutar el servidor.

C) **V_secundaria:**

De igual forma que la clase V_principal, esta clase también es una instancia de la interfaz V_secundaria.ui creada en PyQt. Esta clase permite visualizar la interfaz mostrada en la Figura 3.10. La misma que se muestra solo si el servidor ha sido iniciado correctamente

La Figura 3.11 muestra los atributos y métodos implementados en esta clase.

Los atributos de la clase V_secundaria se describen a continuación:

- **setMinimumSize:** Este atributo es heredado de la interfaz V_secundaria y permite configurar el tamaño mínimo de la ventana al iniciar
- **ipServer:** Este atributo guarda la IP del servidor dentro de esta clase
- **_IpPLC:** Este atributo guarda un listado de IPs de los PLCs que se encuentran en la red
- **tree_ui:** Mediante este atributo se instancia a la clase *TreeUI*
- **atts_ui:** Al igual que el método anterior se instancia a la clase *AttrsUI*
- **refs_ui:** Instancia a la clase *RefsUI*

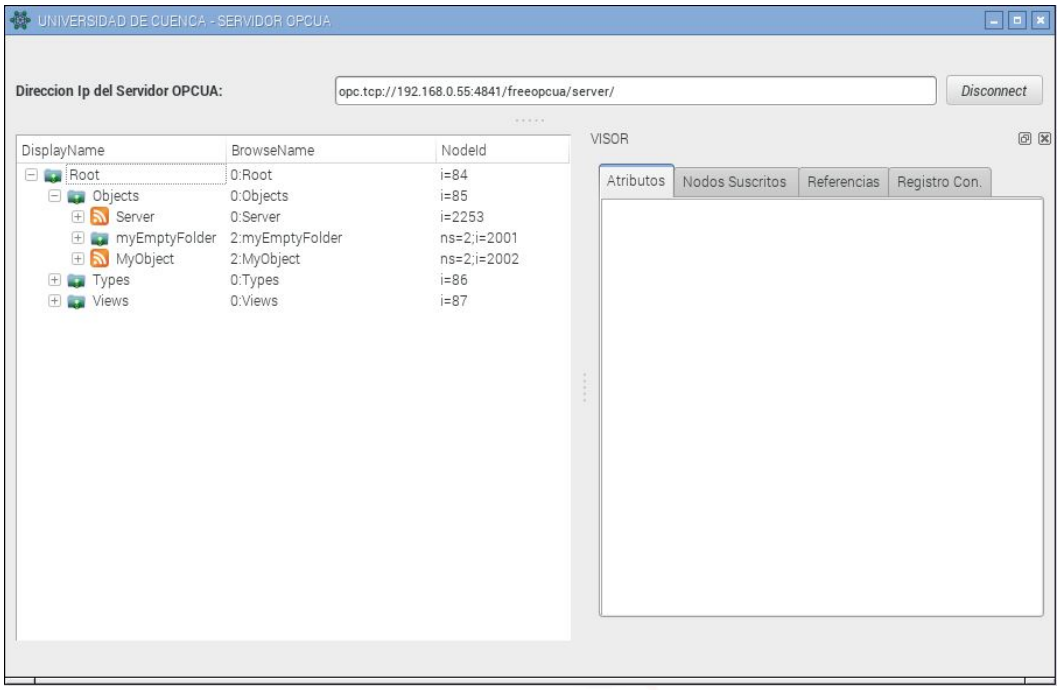


Figura 3.10: Interfaz V_secundaria

- **datachange_ui:** Instancia a la clase *DataChangeUI*

A través de los atributos anteriores, la clase V_secundaria puede usar los métodos de cada clase previamente instanciada y poder visualizar sus valores en la interfaz gráfica.

Los métodos de la esta clase se describen a continuación:

- **showContextMenu:** Este método permite crear un menú secundario. Al seleccionar sobre un nodo es posible ejecutar funciones propias de OPC UA, como crear y eliminar nodos, como se muestra en la Figura 3.12, además de suscribirse a cambios, crear y eliminar objetos y mostrar el historial de un nodo, dependiendo del nodo que se seleccione.
- **show_error:** Este método permite visualizar algún error interno del servidor. El error se puede visualizar en la parte inferior de la ventana como se muestra en la Figura 3.13
- **Actualizar:** Posibilita pasar atributos de la clase V_principal a la clase V_secundaria. Los valores que recibe de parámetro son la IP y el puerto del servidor. Además, este método actualiza el espacio de direcciones, al momento de abrir el la ventana por primera vez

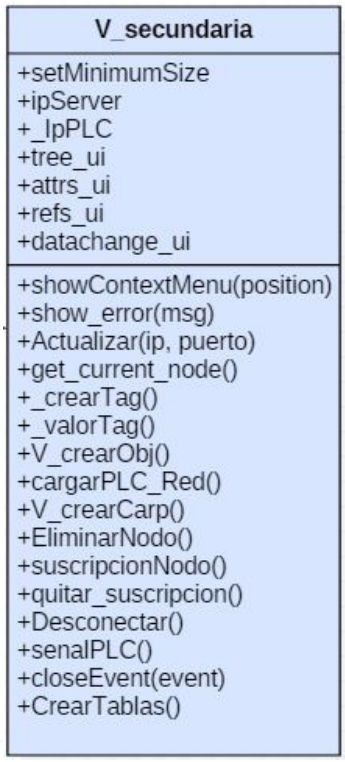


Figura 3.11: Clase V_secundaria

- **get_current_node:** Permite obtener el nodo seleccionado del espacio de direcciones del servidor
- **_crearTag:** Mediante este método se crea una instancia de la clase V_crearTag y se la ejecuta
- **_valorTag:** Este método es similar al método anterior. Sin embargo, desde este método no se permite crear una etiqueta, si no solo cambiar su valor
- **V_crearObj:** A través de este método se crea una instancia a la clase

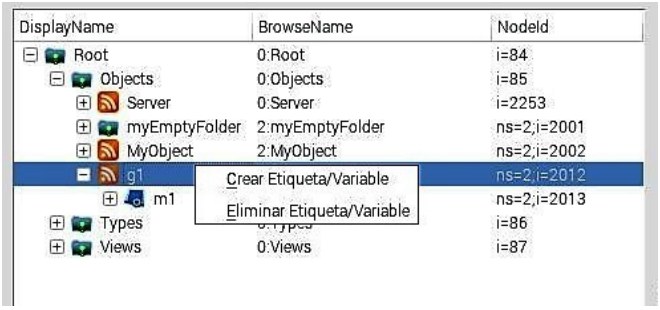


Figura 3.12: Menú contextual o secundario en el espacio de dirección del servidor

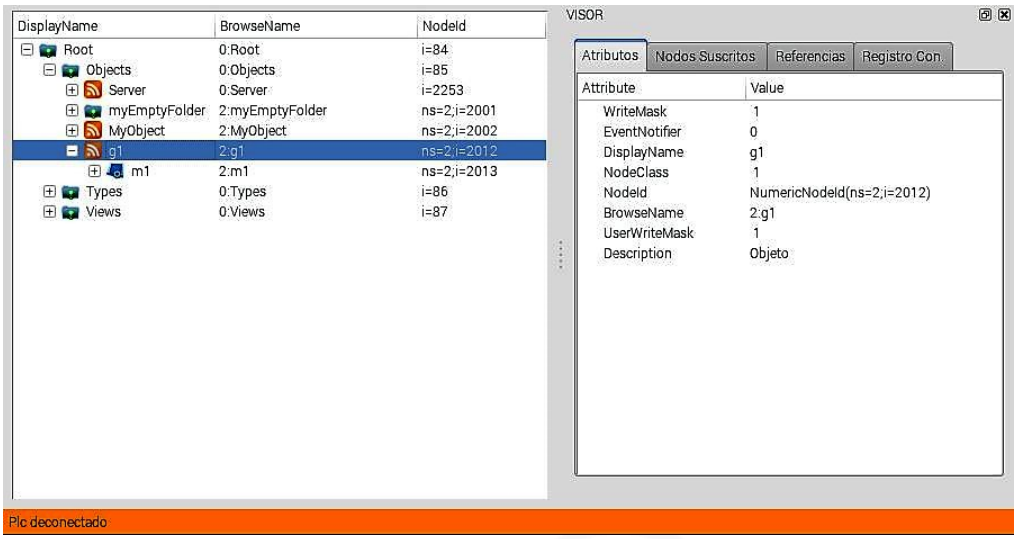


Figura 3.13: Visor de errores en la ventana secundaria

V_crearObj para ejecutar sus propios métodos

- **cargarPLC_Red:** Permite identificar que PLCs se encuentran conectados a la red y guarda un listado de IPs validas
- **V_crearCarp:** Una de las ventajas de OPC UA, es su capacidad crear carpetas ademas de objectos y etiquetas. Mediante este método un usuario puede crear carpetas y almacenar variables que no se encuentran en un PLC y hacer uso de ellas como cualquier otro nodo
- **EliminarNodo:** Crea una instancia de la clase V_eliminarTag. Actualiza una tabla con todos los nodos hijos del nodo seleccionado. Al seleccionar un nodo de la nueva tabla este inmediatamente es borrado del espacio de direcciones del servidor
- **suscripcionNodo:** A través de este método, todos los nodos suscritos pasan como parámetro a un hilo. Este hilo constantemente verifica el valor de un nodo. En caso de producirse un cambio se llama a la clase *DataChangeHandler*
- **quitar_suscripcion:** Cuando un nodo es eliminado desde el servidor antes de eliminar la suscripción se genera un error. Este método se ejecuta inmediatamente después de eliminar un nodo, con la finalidad de evitar posibles errores
- **Desconectar:** Este método borra todos los nodos creados en el servidor y regresa al usuario a la pantalla de inicio. La base de datos anterior es eliminada y crea una nueva

- **senalPLC:** Dentro de Python el concepto de señales y *slots* es muy importante. Permiten el intercambio de información entre objetos de la interfaz. la señal parte de un objeto emisor y llega a un objeto receptor. Este método decide si ejecuta un *slot* y finaliza el proceso

  La señal es generada desde el hilo que mantiene la conexión con el PLC. Al generarse un error se envía una señal a la interfaz gráfica del la clase V_secundaria y se muestra un mensaje de error.

  La Figura 3.14 muestra el intercambio de información entre dos clases que implementan interfaz gráfica mediante señales y *slots*.

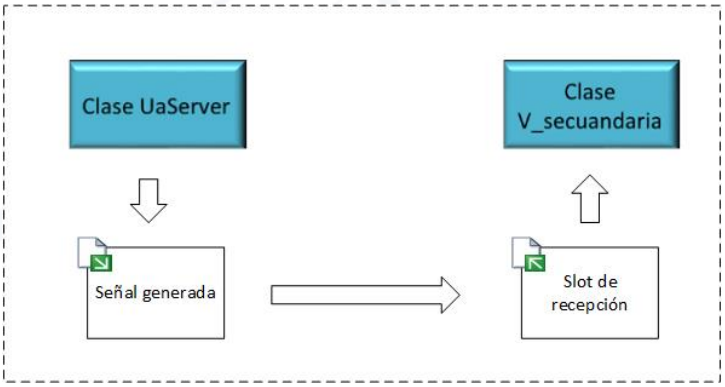


Figura 3.14: Intercambio de información entre dos clases que implementan una interfaz gráfica

- **closeEvent:** Este método permite al usuario no cerrar la sesión del servidor por equivocación. Mediante este método se crea una ventana informativa preguntando si la acción tomada es correcta

- **CrearTablas:** Permite crear las tablas para almacenar las variables en la base de datos. Este método se ejecuta una sola vez al iniciar el servidor. Al cerrar la sesión del servidor, los valores de la base se borran, sin embargo, las tablas permanecen intactas

D) **TreeUI:**

Esta clase permite visualizar los nodos creados en el servidor. Los nodos mostrados se organizan en forma de árbol jerárquico. Para obtener los valores

desde el servidor y visualizarlos en la interfaz gráfica. Ésta clase implementa los atributos y métodos mostrados en la Figura 3.15

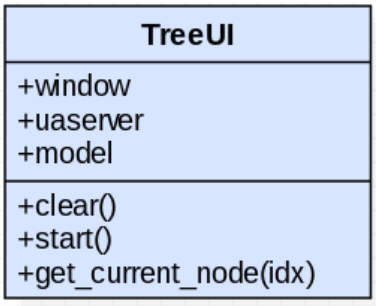


Figura 3.15: Clase TreeUI

Los atributos de la clase se describen a continuación:

- **window:** Mediante este atributo se crea una instancia de la clase *V_secundaria*. Esto permite a utilizar los métodos definidos por la clase instanciada
- **uaserver:** Crea una instancia a la clase *UaServer*. Esto permite utilizar los metodos definidos por las librerías de *FreeOpcUa*
- **model:** Crea una instancia a la clase *TreeViewModel*

Los métodos definidos por la clase son:

- **clear:** A través de esta método se puede borrar todo el directorio generado. Cada vez que el usuario crea un nodo, antes de actualizar el directorio, este es borrado para evitar problemas de solapamiento de información
- **start:** Cuando el programa inicia por primera vez, este método inicializa las variables de la clase
- **get_current_node:** Permite obtener el identificador del nodo seleccionado

E) **TreeViewModel:**

Para poder generar un árbol de directorio, es necesario crear un modelo o plantilla donde se almacenaran los datos. Esta clase implementa los atributos y métodos mostrados en la Figura 3.16.

Los atributos de la clase se describen a continuación:

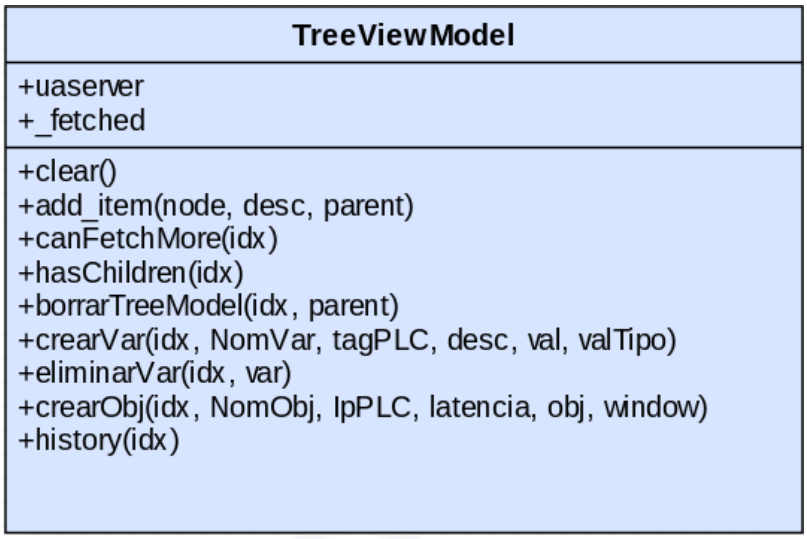


Figura 3.16: Clase TreeWiewModel

- **uaserver:** Permite a la clase heredar los métodos de la clase *UaServer*
- **_fetched:** Este atributo guarda todo los nodos que han sido analizados previamente, es decir guarda los nodos que abarcan a otros nodos

Los métodos de la clase se describen a continuación:

- **clear:** Facilita la tarea de eliminar todos los datos generado en el modelo de presentación
- **add_item:** Se utiliza este método para agregar nuevos nodos al modelo de directorio. Antes de realizar esta tarea, este método clasifica cada nodo. La clasificación se la realiza en base a la clase a la que pertenece el nodo. Cada nodo tiene un ícono que lo identifica de su clase como se muestra en la Figura 3.12
- **canFetchMore:** Al ser la clase del tipo *model*, ésta hereda métodos definidos en PyQt. Por ende, el método *canFetchMore* se encuentra definido. Sin embargo, debe ser implementado en el programa. A través del mismo se puede identificar si el nodo seleccionado puede o no contener nodos hijos. En caso de devolver un valor booleano verdadero se llama al método *hasChildren*
- **hasChildren:** A través de las librerías de FreeOpcUA, este método adquiere los nodos hijos del nodo seleccionado. Para agregar un nuevo nodo al directorio raíz, se llama al método *add_item*

- **borrarTreeModel:** Cuando un nodo es borrado del servidor, este también debe ser borrado del directorio raíz
- **crearVar:** Permite la adición de un nuevo nodo tanto en el servidor como en la interfaz gráfica
- **eliminarVar:** Se utiliza para eliminar un nodo seleccionado del servidor
- **crearObj:** Este método se lo emplea en la creación de un objeto como tal y para la creación de carpetas. Esto se puede lograr ya que en OPC UA, todo es identificado como nodos dentro del servidor
- **history:** Este método trabaja conjuntamente con la base de datos. Permite obtener la descripción del nodo seleccionado y así por saber que tipo de variable maneja el nodo

F) **AttrsUI:**

Esta clase permite visualizar todos los atributos de un nodo seleccionado, así como el valor de cada uno de ellos, esto se muestra en la Figura 3.17

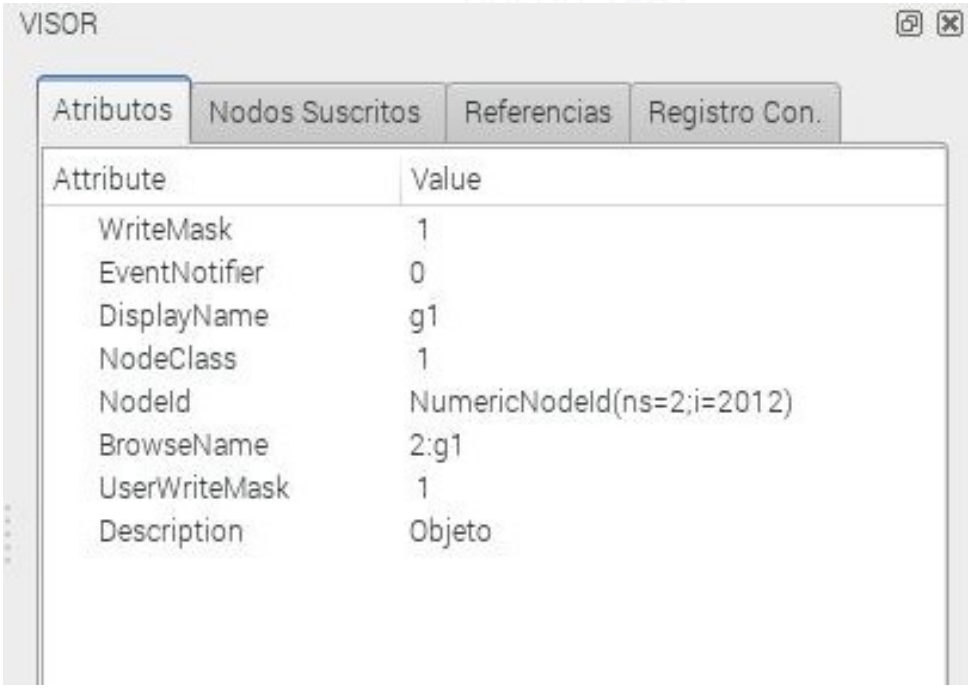


Figura 3.17: Visor de atributos de la clase AttrsUI

La clase implementa los atributos y métodos mostrados en la Figura 3.18.

Los atributos de la clase se describen a continuación:

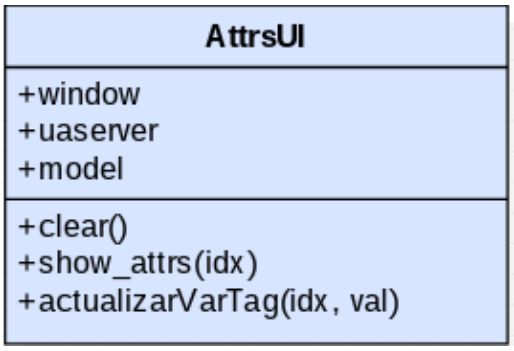


Figura 3.18: Clase AttrsUI

- **window:** Crea una instancia a la clase *V_secundaria*, para poder ejecutar sus métodos
- **uaserver:** Permite instanciar la clase *UaServer*
- **model:** Esta clase implementa el tipo *model* de la librería de PyQt. Permitiendo a la clase generar su propio modelo de organización de datos para visualizarlos en la interfaz gráfica de usuario

Los métodos de la clase son:

- **clear:** Permite borrar todo el contenido de la clase. Este método se ejecuta siempre que un nuevo valor o algún atributo a cambiado
- **show_attrs:** Al recibir un nodo como parámetro, lo pasa a la clase *UaServer*, en donde adquiere todos los atributos del mismo. Este método se ejecuta cada vez que el usuario selecciona un nodo del directorio raíz
- **actualizarVarTag:** Cuando un cliente necesita cambiar el valor de un nodo, la tarea es ejecutada mediante este método

G) **RefsUI:**

Cada nodo mantiene relación con otros nodos. A estas relaciones se las denomina referencias. Al seleccionar un nodo objeto, las referencias de este nodo son con la de sus nodos hijos. La Figura 3.19 muestra las referencias de un nodo objeto (PLC) y las de sus nodos hijos (etiquetas). Cada referencia esta compuesta por tipo e identificador del nodo

La clase esta compuesta por los atributos y métodos mostrados en la Figura 3.20

Los atributos de la clase son:

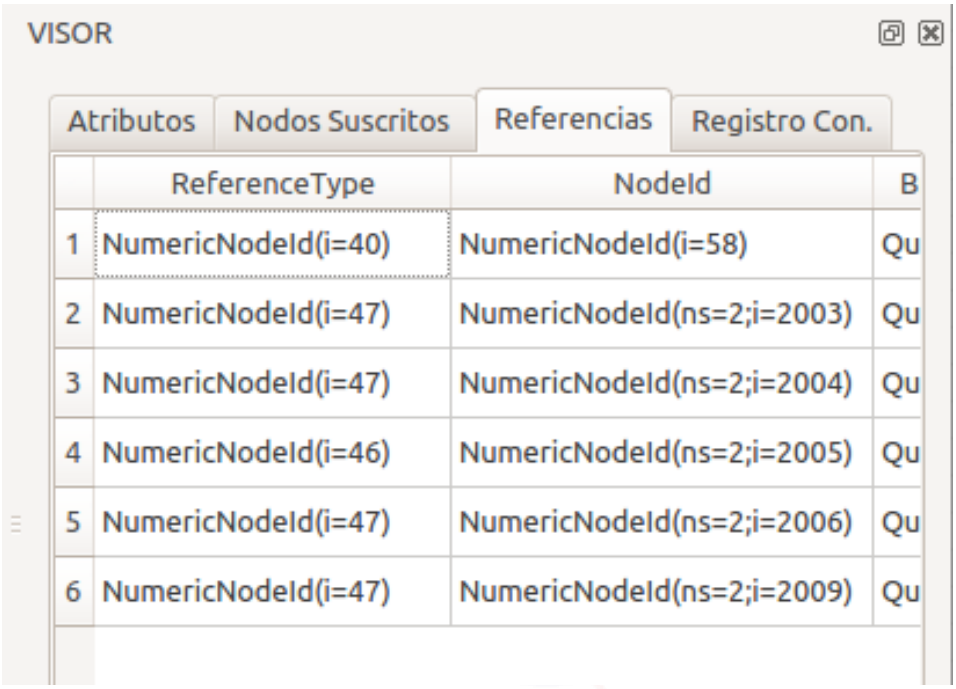


Figura 3.19: Visor de referencias de un nodo seleccionado

- **window:** Al igual que las clases anteriores este atributo hereda los metodos de la clase *V_secundaria*
- **uaserver:** Permite a la clase manejar las librerías de FreeOpcUa
- **model:** Para poder visualizar los datos de la clase en una interfaz gráfica es necesario implementar la clase *model*

Los métodos de la clase son:

- **clear:** Implementado para borrar todos los datos de la clase

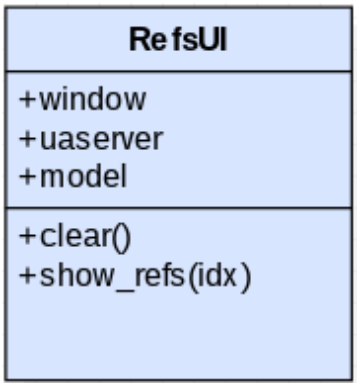


Figura 3.20: Clase RefsUI

- **show_refs:** Recibe como parámetro un nodo y devuelve todos los nodos a los que hace referencia el nodo seleccionado. Cada referencia se la realiza a un nodo, en donde se especifica el tipo al que pertenece y su identificador en el espacio de direcciones del servidor

H) **DataChangeHandler:**

Mediante esta clase se comunican los datos del servidor y la clase *DataChangeUI*. Para que un usuario pueda estar al tanto sobre la variación en cuanto al valor de un nodo es necesario la implementación de un hilo. Este hilo constantemente monitorea el estado de un nodo suscrito. Para comunicar el hilo con la interfaz gráfica, se ha implementado el mecanismo de señales y *slots*, los cuales se describieron anteriormente.

La Figura 3.21 muestra el atributo y el método definido en esta clase.

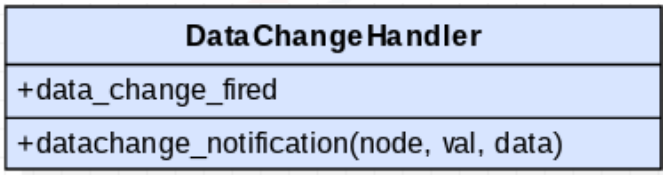


Figura 3.21: Clase DataChangeHandler

El Atributo implementado por esta clase es:

data_change_fired: Este atributo es el slot, en donde el hilo encargado envía el nuevo valor de un nodo al producirse un cambio.

Para poder trasferir los datos obtenidos a la interfaz gráfica se ha implementado el siguiente método:

datachange_notification: Este método envía como parámetro el atributo de la clase hacia la clase *DataChangeUI*.

I) **DataChangeUI:**

Cuando un nodo ha cambiado de valor, esta clase se encarga de notificarlo en la interfaz de usuario. Cada nodo suscrito es tratado independientemente por lo que el visor de suscripciones puede contener tantos nodos como el usuario necesite. La Figura 3.22 muestra los nodos suscritos, así como el valor del mismo en la hora que se produjo el cambio.

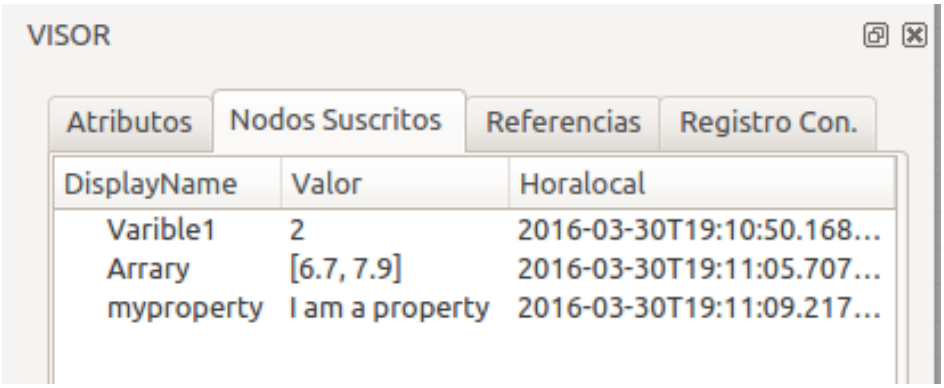


Figura 3.22: Visor de cambios de los nodos suscritos

Los atributos y métodos de la clase se muestran en la Figura 3.23.

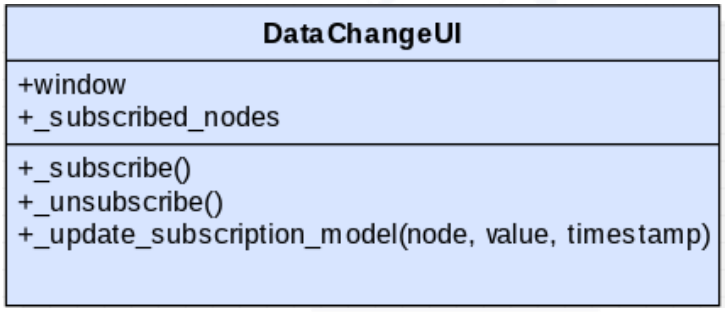


Figura 3.23: Clase DataChangeUI

Los atributos de la clase se describen a continuación:

- **window:** De la misma forma que en la clase anterior, este atributo permite a la clase poder usar los métodos de la interfaz gráfica
- **_subscribed_nodes:** Con el fin de llevar un control de los nodos suscritos, la clase almacena a cada uno de ellos en un listado. Este listado permite reconocer un nodo suscrito antes de eliminarlo, caso contrario generaría un error

Los métodos implementados son:

- **subscribe:** Este método recibe como parámetro un nodo y lo envía hacia el servidor. Este se encarga se crear un hilo para el nodo, de esta manera al generarse un cambio se genera una señal
- **_unsubscribe:** Permite al usuario detener las notificaciones sobre el nodo suscrito. Esta método es ejecutado siempre que un nodo va a ser eliminado

- **_update_subscription_model:** Este método actualiza el valor de los nodos suscritos en la interfaz de usuario

J) **V_crearTag:** Esta clase es una instancia de la interfaz V_AgregarTag.ui desarrollada en PyQt. Permite al usuario ingresar los datos de una nueva variable o etiqueta en el servidor

La Figura 3.24 muestra la ventana en donde se ingresan los parámetros necesarios para crear una etiqueta en el servidor. Los parámetros básicos de ingreso son el nombre, la dirección de la variable en el PLC y el tipo de dato. Los parámetros como el valor y la descripción son opcionales

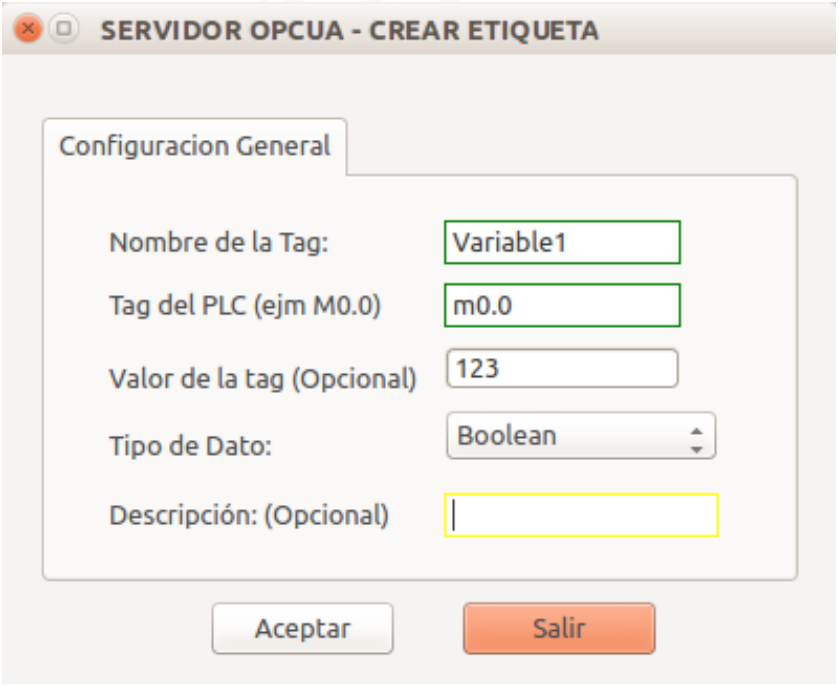


Figura 3.24: Interfaz V_AgregarTag

Los atributos y métodos de la clase se muestran en la Figura 3.25.

Los atributos de la clase se describen a continuación:

- **validarTag:** Este atributo es una variable del tipo booleano. Es utilizado por la clase para identificar si la *Tag del PLC* fue correctamente ingresada

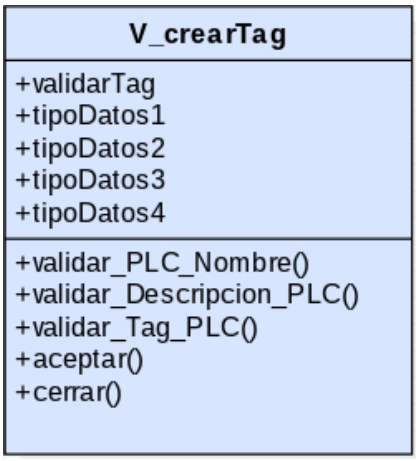


Figura 3.25: Clase V_crearTag

Dependiendo de la *Tag del PLC*, el servidor agrupa a las variables en cuatro tipo de datos. Esta agrupación se basa en el número de bits que se maneja en cada tipo. Cada tipo se mencionan a continuación:

- **tipoDatos1:** Las variables con este tipo de dato tienen una longitud de un bit, por ende, corresponden a variables de tipo booleano
- **tipoDatos2:** Las variables dentro de esta categoría tienen una longitud de 1 byte. Corresponden a variables del tipo *Byte, Char, UInt8 y SByte*
- **tipoDatos3:** Corresponden a variables de 2 bytes. En esta categoría están las variables del tipo *Int16, Int32, Int64, UInt16, UInt32 y UInt64*
- **tipoDatos4:** La longitud de las variables son de 4 bytes y corresponden a datos tipo *DateTime, Double, Float y String*

Los métodos de la clase se definen a continuación:

- **validar_PLC_Nombre:** Este método evita que el usuario ingrese un dato muy extenso en el nombre de la variable
- **validar_Descripcion_PLC:** De igual manera que el método anterior, esto evita que la descripción de la nueva variable creada sea demasiada extensa
- **validar_Tag_PLC:** Si la *Tag del PLC* fue ingresada correctamente, el atributo validarTag toma valor de verdadero, caso contrario, tomará un valor de falso
- **aceptar:** Este método se ejecuta al presionar el botón Aceptar de la interfaz V_AgregarTag. Antes de guardar un nuevo nodo, se ejecutan los

métodos descritos anteriormente. Luego de constatar de que los datos ingresados son correctos la ventana se cierra, dando a conocer que se ingreso un nuevo nodo, caso contrario se visualiza un mensaje para que el usuario revise los datos que ingresó

- **cerrar:** Permite cerrar la ventana sin ingresar ningún nodo

K) **V_eliminarTag:**

Para eliminar un nodo, es necesario seleccionar el nodo que lo contiene. En este caso para eliminar una etiqueta es necesario seleccionar el nodo objeto (PLC). Cuando el usuario ejecuta este método visualiza la interfaz mostrada en la Figura 3.26. Para eliminar un nodo, el usuario solo necesita seleccionar el nodo y presionar el botón eliminar. Luego de que un nodo es eliminado la ventana no se cierra, lo que permite borrar todos los nodos que se requieran de una forma fácil e intuitiva

Figura 3.26: Interfaz V_EliminarTag

La Figura 3.27 muestra los atributos y métodos implementados en la clase.

Esta clase implementa el siguiente atributo:

- **parent:** Cuando ésta es ejecutada, es necesario obtener el nodo padre (PLC) y el nodo hijo (etiqueta). Este atributo guarda el nodo padre,

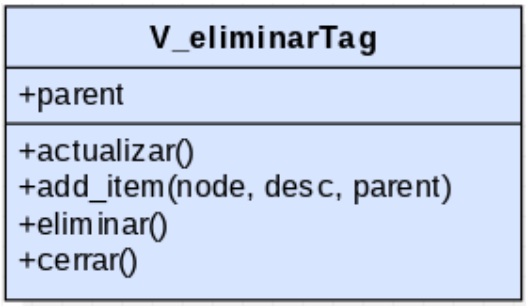


Figura 3.27: Clase V_eliminarTag

para que el método eliminar de la misma clase pueda distinguir entre estos dos nodos y pueda ejecutar esta tarea

Los métodos de la clase son:

- **actualizar:** Permite llenar con datos la interfaz gráfica. Los datos mostrados corresponden a los nodos hijos del nodo seleccionado. Este método se ejecuta antes de que la ventana se visualice
- **add_item:** Este método trabaja conjuntamente con el método anterior, permitiendo agregar nodo por nodo a la ventana gráfica
- **eliminar:** Este método se ejecuta al seleccionar el botón eliminar. En caso de generarse un error en el proceso de eliminación se visualiza un mensaje. Sin embargo, este mensaje es solo informativo pues puede ser generado por el mismo servidor, mas no por el usuario
- **cerrar:** Cierra la ventana sin eliminar ningún nodo

L) **V_crearObj:**

Esta clase facilita el proceso de creación de un nuevo objeto (PLC) en el servidor. Cuando se ejecuta se visualiza la interfaz mostrada en la Figura 3.28

Los parámetro de entrada mostrados en la Figura 3.28 son el nombre del objeto, Ip del PLC, latencia. Con el fin de evitar errores con ingreso de la IP del PLC, el servidor monitorea la red local y extrae solo las IP corresponden a un PLC. Esta verificación se basa en la capa de acceso al medio (MAC, por sus siglas en inglés) definida para los PLC de Siemens.

La Figura 3.29 muestra los atributos y métodos de la clase.

Los atributos de la clase se describen a continuación:

Figura 3.28: Interfaz V_AgregarObj

- **window:** Permite instanciar a la clase V_secundaria para ejecutar sus métodos y mostrar los resultados de un nuevo nodo creado
- **obj:** Este atributo es del tipo booleano. Cuando adquiere un valor de verdadero la clase crea un nodo objeto y configura los parámetros necesarios para esta tarea. Cuando su valor es falso, la clase crea un nodo del tipo carpeta

Los métodos implementados por la clase son:

- **validar_IntLatencia:** Con el fin de no saturar a la Raspberry PI, se

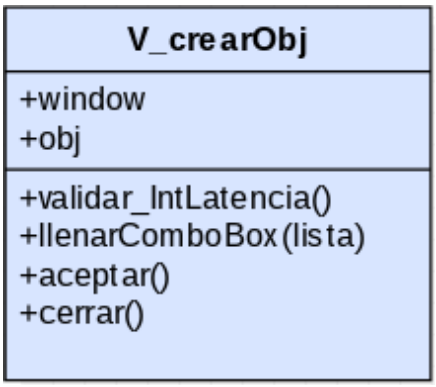


Figura 3.29: Clase V_crearObj

ha establecido el parámetro latencia. Este parámetro debe superior a 500 milisegundos, permitiendo al servidor procesar otros datos en este intervalo

- **llenarComboBox:** Ésta clase analiza la red local sobre la cual se encuentra el servidor. Al terminar la verificación obtiene las direcciones IP de todos los dispositivos conectados, para finalmente filtrar solo las que cumplen con los octetos 01:BD en dirección MAC
- **aceptar:** Este método verifica que todos los campos ingresados son correctos y procede a crear el nuevo objeto
- **cerrar:** Cierra la ventana sin crear ningún objeto

M) **UaServer:**

N) **Clase V_historial:**

la clase V_historial es una clase que tiene la finalidad de conectar la interfaz gráfica con la base de datos. Mostrar el historial de cambios que se han producido en las *tags*. Para ello, tiene los siguientes métodos:

- **Seleccionar:** Crea la interfaz necesaria para mostrar el historial de la base de datos
- **llenarComboBox:** Realiza la conexión entre la base de datos y la interfaz. Busca la *tag* seleccionada y muestra la tabla Registro con la fecha seleccionada.

Finalmente a través de esta clase las clases definidas anteriormente pueden hacer uso de las librerías FreeOpcUA. Esta clase importa todas las librerías antes mencionadas, permitiendo al programa llamar solo a la clase en lugar de ir importando cada librería una por una.

3.3.4. Implementación de una base de datos

La base de datos del servidor fue implementada con el uso de la librería SQLite3. Su código fuente es de dominio público y se encuentra integrada a Python, por lo que no es necesario la instalación de una librería adicional.

La librería SQLite3 se comunica directamente con el programa, es decir, pasa a ser parte integral del servidor. Ésta característica difiere de las bases de datos

que se basan en el sistema cliente-servidor. El servidor utiliza SQLite3 a través de llamadas simples a subrutinas y funciones. Esto provoca la reducción de la latencia en el acceso a la base de datos, debido a que, es más eficiente llamar a funciones, que la comunicación entre procesos.

Las características de la librería SQLite3, cumplen con los requerimientos de la base de datos local que se desea implementar. Para el uso de la librería establece la clase Database.

La Figura 3.30 expone la interfaz, que se realiza mediante la lectura de tablas de la base de datos.

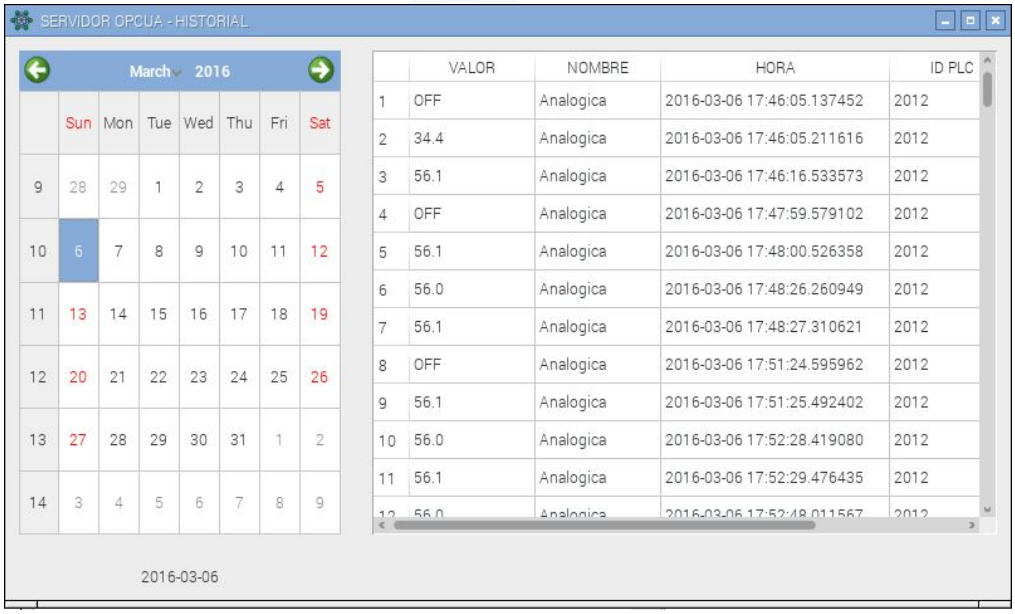


Figura 3.30: Interfaz historial

Clase Database

La clase Database maneja todas las funciones que utiliza el servidor para comunicar la librería SQLite3 con la clase Server.

La Figura 3.31 muestra los atributos y métodos implementados de la clase Database.

Los atributos de la clase Database son los siguientes:

- **conexion:** es el atributo encargado de realizar la conexión entre la base de datos y el programa del servidor

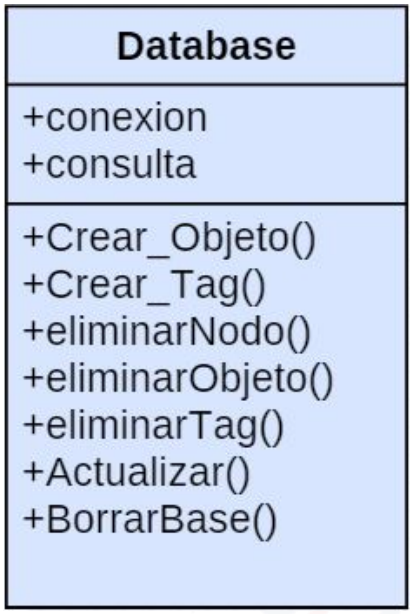


Figura 3.31: Clase Database

- **consulta:** este atributo es el cursor que administra las operaciones que se realizan en la base de datos

Los métodos de la clase son:

- **Crear_Objeto:** es el método encargado en receptar los datos provenientes al crear un nuevo objeto en el servidor. Ingresarlos, en cada campo que conforma la tabla PLC de la base de datos

- **Crear_Tag:** este método obtiene la información que proceden de crear una nueva *tag* en el servidor. Incorpora cada dato a los campos que tiene la tabla Tags en la base de datos

- **eliminarNodo:** esta función toma la información que proviene del nodo seleccionado. Reconoce el campo identificador de la información proporcionada y clasifica ya sea en objeto o en *tag*. Finalmente, realiza una llamada al método eliminarObjeto o al método eliminarTag, según la clasificación realizada

- **eliminarObjeto:** toma como información el campo identificador del objeto a eliminar, lo busca en campo identificador de objeto en la tabla PLC, Tags, Registro de la base de datos y lo elimina de cada tabla

- **eliminarTag:** obtiene el campo identificador de la *tag* a eliminar, la busca en campo identificador en las tablas Tags y Registros de la base de datos y procede a eliminar de cada tabla
- **Actualizar:** en este método ingresa información del campo valor y el identificador a la *tag* correspondiente en la base de datos. Compara el nuevo valor ingresado, con el último valor guardado en la tabla Registro de la base de datos. Como resultado, guarda el valor nuevo en la tabla Registro, si es diferente al último valor o, por el contrario, lo descarta
- **BorrarBase:** al realizar la desconexión o cierre del servidor OPC, ésta función, borra todos los campos que se ingresaron en las tablas de la base de datos

3.4. Funcionamiento

El funcionamiento de la interfaz del servidor, se basa en su mayor parte en dos ventanas principales. Estas ventanas, son asistidas por ventanas auxiliares para la exhibición de información temporal, toma de decisiones y mensajes al usuario.

3.4.1. Interfaz de inicio

En la interfaz de inicio del servidor se configuran los siguientes parámetros:

- **Puerto:** El puerto que habilita la comunicación. El valor por defecto es 4841, ya que el protocolo OPC UA tiene definido este puerto
- **Dirección IP del servidor:** Este valor se configura por defecto la ip fija de la Raspberry Pi, pero puede ser ingresado otro valor, en caso de que se haya reconfigurado la ip fija de la Raspberry Pi para otra red local

  El cuadro de ingreso de texto se encuentra programado para validar el texto ingresado, es decir, cambia de color según el texto ingresado en los siguientes colores:

  1. El borde se colorea de rojo al momento que se ingresa incorrectamente el texto, por lo que no valida el texto ingresado. La Figura 3.32 muestra el color rojo del borde del cuadro de ingreso de texto

Figura 3.32: Validación IP color rojo

2. El borde se colora de amarillo si no encuentra ningún texto escrito. La Figura 3.33 muestra el color amarillo del borde del cuadro de ingreso de texto

Figura 3.33: Validación IP color amarillo

3. El borde se tiñe de verde si el texto introducido tiene el formato correcto de una IPV4, en consecuencia, permite leer los datos introducidos. La Figura 3.34 muestra el color verde del borde del cuadro de ingreso de texto

Figura 3.34: Validación IP color verde

La interfaz cuenta con un indicador del estado en que se encuentra el servidor. Al momento que se abre el programa del servidor, el indicador imprime desconectado y la barra de progreso se encuentran en cero.

La Figura 3.35 muestra la interfaz de inicio del servidor al abrir el programa.

Al momento que se ingresó correctamente todos los parámetros pedidos en la interfaz, se procede a iniciar el servidor presionando el botón CONECTAR (Ver la Figura 3.36). La Figura 3.36 expone el botón a seleccionar para dar inicio al servidor.

Figura 3.35: Servidor desconectado

Cuando el servidor está iniciando todos sus módulos, cambia el mensaje del estado del servidor. A la vez que la interfaz posee una barra de estados que informa al usuario el avance del cargado de librerías por parte del servidor. La Figura 3.37 muestra el cambio de la barra y estado al cargar módulos de inicio.

Finalmente, cuando el servidor cargo en su totalidad los módulos, el indicador de estado de la interfaz cambia de mensaje a Conectado. Cierra esta ventana y procede a abrir la siguiente ventana.

3.4.2. Interfaz del servidor

La Figura 3.38 muestra la interfaz del servidor con todos los parámetros.
Las partes principales en las que se divide la interfaz son:

- En la parte superior de la interfaz se observa en un cuadro de texto la dirección IP del servidor al que se encuentra conectado
- En la esquina superior derecha se encuentra un botón que realiza la desconexión del servidor OPC UA
- En la parte izquierda se encuentra el espacio de dirección
- En la parte derecha se localiza el VISOR

Figura 3.36: Inicio de servidor mediante botón

El espacio de direcciones presenta los siguientes atributos:

- **DisplayName:** En este espacio de direcciones, se puede observar la estructura de carpetas, archivos, métodos, tipos y eventos que posee el servidor OPC UA. La Figura 3.39 muestra el espacio donde se encuentra el DisplayName

- **BrowseName:** Se encuentra en la parte central del espacio de direcciones. En ésta área, se encuentran los nombres de los nodos que se crean por defecto al iniciar el servidor. También, ingresa en este lugar los nombres de los nuevos nodos creados. En la Figura 3.40 se observa el espacio del BrowseName

- **NodeId:** En el último sitio, se expone los identificadores de cada nodo, carpeta, método que se encuentra en el servidor a disposición del cliente. La Figura 3.41 presenta el lugar donde se ubica el NodeId

El VISOR está compuesto por las siguientes pestañas:

- **Atributos:** En esta pestaña se expone todas las características que posee cada nodo en el servidor. La Figura 3.42 muestra los atributos del nodo

Figura 3.37: Servidor cargando módulos para iniciar

seleccionado

- **Nodos suscritos:** En la primera viñeta se encuentra los nodos que se suscribieron a cambios, es decir se observa el cambio a tiempo real del nodo suscrito. En la Figura 3.43 se observa un nodo suscrito a cambios

- **Referencias:** En esta pestaña se ubica las referencias de ubicación del nodo seleccionado. Este borde contiene los atributos del tipo de referencia al nodo padre, la identificación propia (nodo hijo), el nombre que tiene la *tag* y por último el tipo de datos que guarda. La Figura 3.44 expone la pestaña de referencia de los nodos

- **Registro de conexiones:** En el último sitio, se presenta un menú con todos los clientes OPC UA se conectaron al servidor. En ese menú se exhibe las IPs, nombres y los puertos de los clientes. La Figura 3.45 muestra el registro de conexiones de clientes OPC UA.

En el espacio de direcciones se encuentran los nodos y carpetas, que el servidor crea en forma de ejemplo, con el fin de exponer las características y métodos que

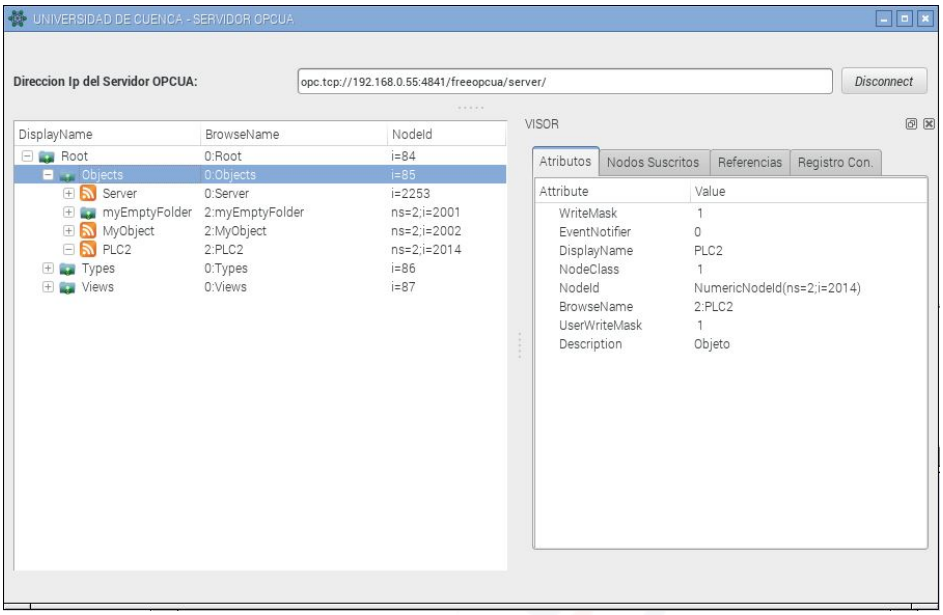


Figura 3.38: Interfaz del servidor

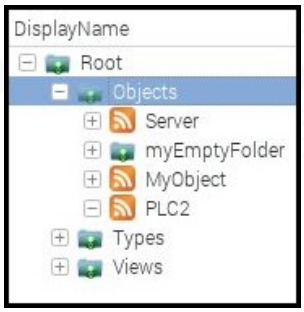


Figura 3.39: Espacio del DisplayName

se pueden crear en el servidor. La Figura 3.46 presenta las características iniciales

Adicionalmente, se localiza parte del servidor, visualizaciones y tipos (datos, eventos, referencias, variables y objetos) que ejecuta el servidor. Esto se crea al iniciar el servidor para ejemplificar el soporte que entrega. Las Figuras 3.47 y 3.48 muestran los atributos y tipos iniciales creados por el servidor respectivamente.

3.4.3. Proceso para crear un nuevo objeto

Una vez que el servidor ha cargado todos los módulos, el usuario puede crear nodos en el servidor, con el fin de conectar un controlador. Para ello, el usuario debe hacer clic derecho en la carpeta *Objects* y se desplegará un menú. El menú

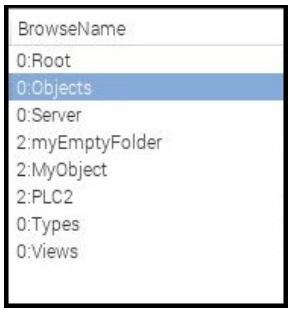


Figura 3.40: Espacio del BrowseName

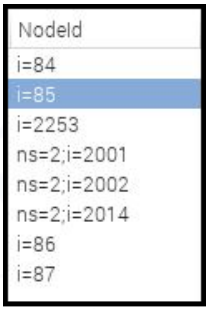


Figura 3.41: Espacio del NodeId

posee tres opciones.

La Figura 3.49 expone el menú de opciones al realizar clic derecho sobre *Objects*

- **Crear nuevo objeto:** El usuario debe ingresar los datos del nodo a crear (nombre, latencia). Estos cuadros de ingreso de datos están programados para validar datos, como la interfaz de inicio. La IP del PLC es identificada automáticamente por el servidor, por lo que se debe seleccionar de las opciones. La Figura 3.50 expone la ventana de creación de objetos

- **Crear nueva carpeta:** El usuario debe realizar el mismo procedimiento que al crear un objeto, pero con la diferencia que la IP no elige porque no es un requisito para crear la carpeta. La Figura 3.51 muestra la ventana de creación de carpetas

- **Eliminar objeto/carpeta:** Esta opción permite al usuario eliminar una carpeta o un objeto del servidor. El usuario debe seleccionar el objeto o

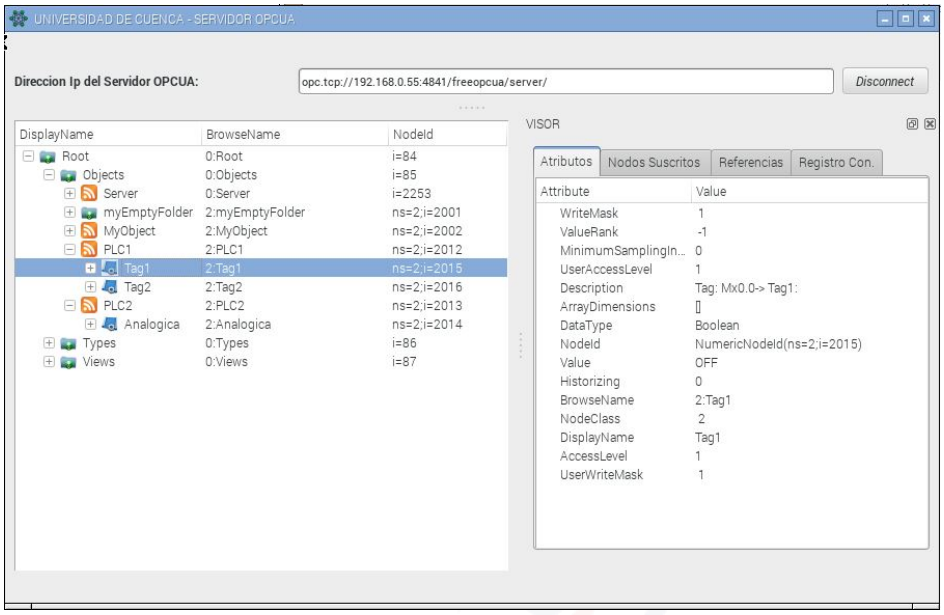


Figura 3.42: Atributos del nodo seleccionado

carpeta de la lista adjunta, y confirmar el cuadro de dialogo. La Figura 3.52 presenta la ventana de eliminación de objetos o carpetas

El servidor ofrece la posibilidad de crear *tags* en los objetos anteriormente creados. El usuario debe dar clic derecho en el ícono del objeto creado. Este desplegará un menú donde muestran las siguientes opciones (Figura 3.53):

- **Crear etiqueta/variable:** El usuario debe ingresar los datos de la variable o *tag* que desea crear (nombre, *tag*, tipo, valor, descripción). Estos cuadros de ingreso de datos están programados para validar datos, como en la creación de objetos. El tipo de variable es seleccionada automáticamente por el programa. La opciones de la descripción y del valor es opcional al ingreso. La Figura 3.54 expone la ventana de creación de *tag*

- **Eliminar etiqueta/variable:** El usuario con esta opción puede eliminar una variable del servidor. El usuario debe seleccionar la variable de la lista adjunta, y confirmar el cuadro de dialogo. La figura 3.55 presenta la ventana de eliminación de variables

El servidor brinda un último menú desplegable que obtiene al dar clic derecho en las variables creadas anteriormente. El menú presenta las siguientes opciones

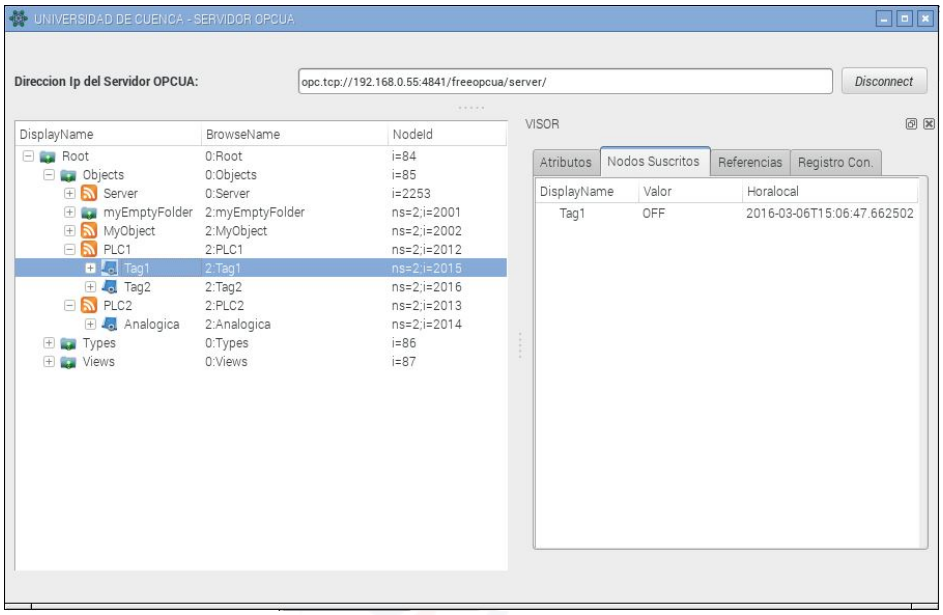


Figura 3.43: Nodo suscrito a cambios

(Figura 3.56):

- **Cambiar el valor a la variable:** El usuario tiene la posibilidad de cambiar el valor de la *tag*, en caso de que haya ingresado por error anteriormente. Para ello, debe seleccionar esta opción y llenar nuevamente el cuadro donde creó la *tag* en los procedimientos anteriores (Ver la Figura 3.54)

- **Suscribirse a cambios:** El usuario con esta opción puede suscribir a la variable, con el fin de observar cómo cambia la variable a tiempo real (Ver la Figura 3.43)

- **Quitar suscripción a cambios:** Una vez que usuario esté satisfecho al observar el cambio de la variable. Puede quitar de la suscripción a la variable

- **Mostrar historial:** El servidor proporciona la opción de tener un historial de los cambios que ha tenido la variable. Esta opción ofrece al usuario la posibilidad de seleccionar la fecha en que desea revisar los movimientos que ha realizado la *tag* seleccionada (Ver la Figura 3.30)

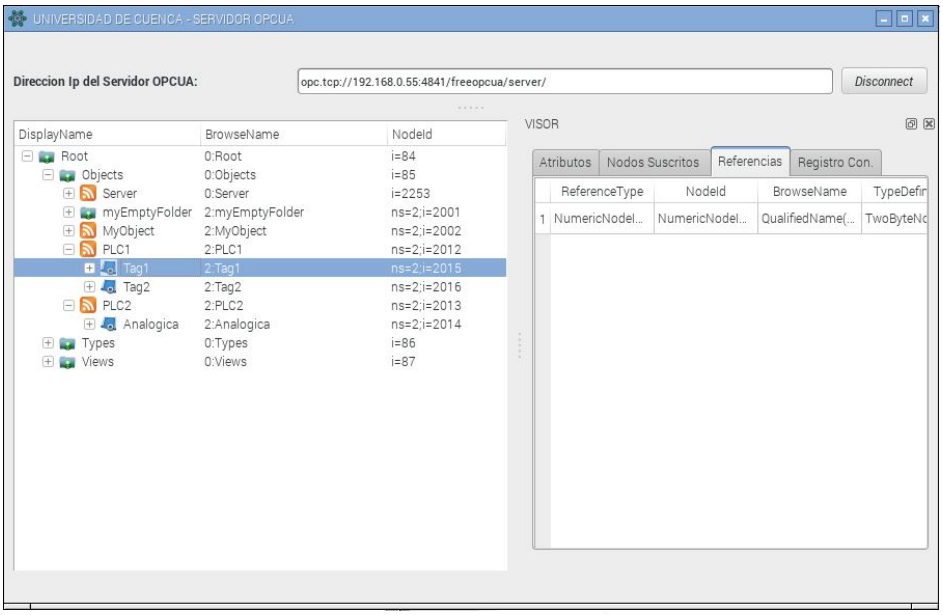


Figura 3.44: Referencias de los nodos

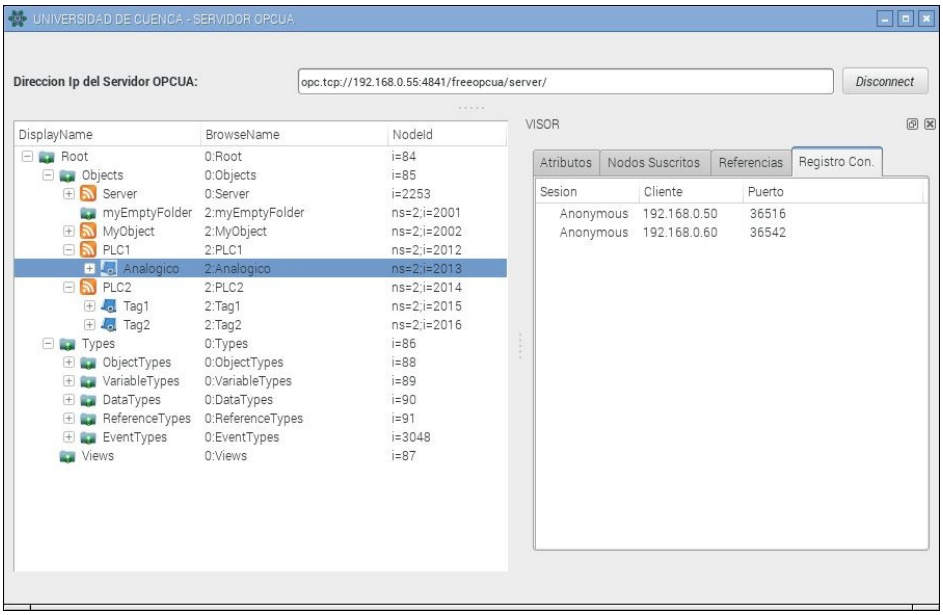


Figura 3.45: Registro de conexiones de clientes

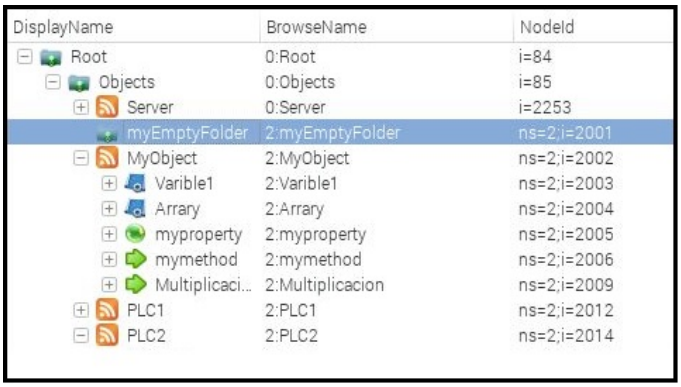


Figura 3.46: Características iniciales en el nodo creadas por el servidor

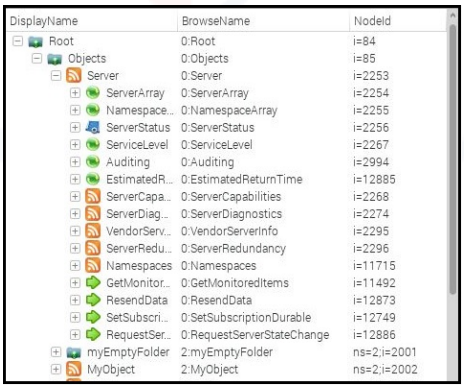


Figura 3.47: Atributos iniciales creados por el servidor

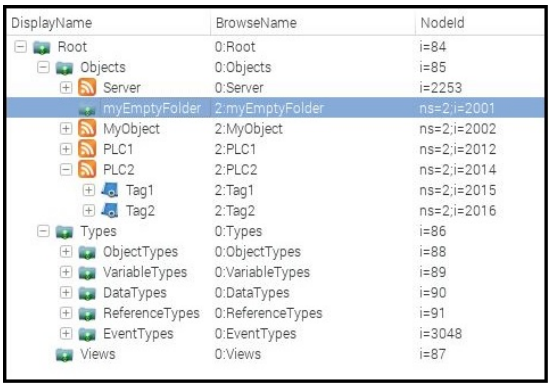


Figura 3.48: Tipos iniciales creados por el servidor

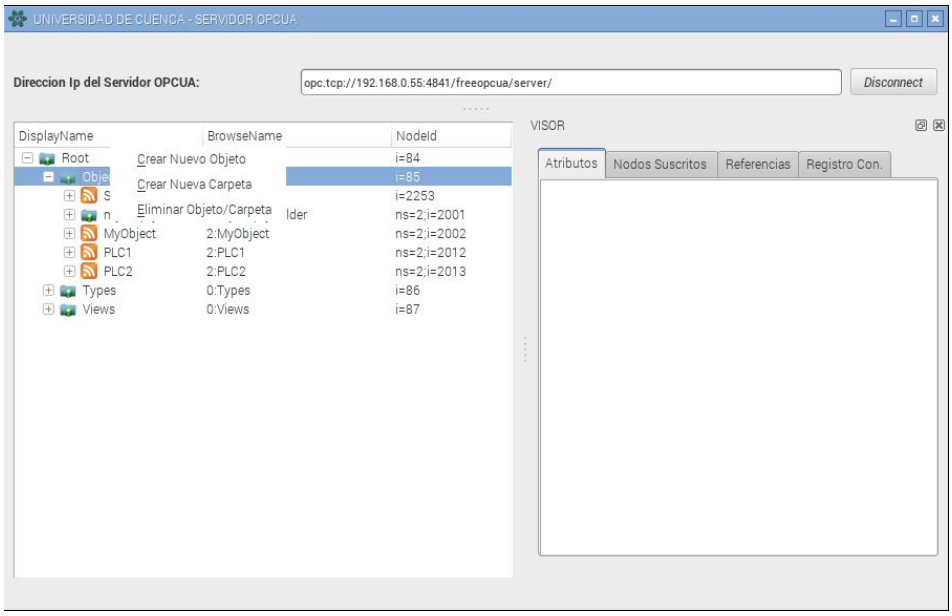


Figura 3.49: Menú de opciones de objeto

Figura 3.50: Ventana de creación de objetos

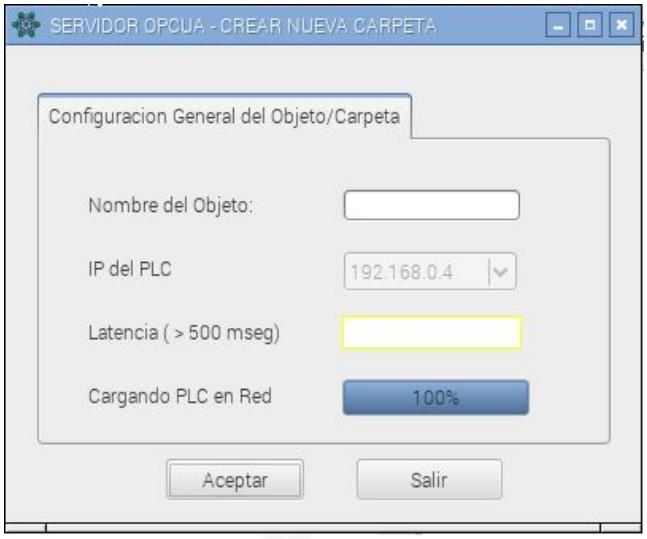


Figura 3.51: Ventana de creación de carpetas

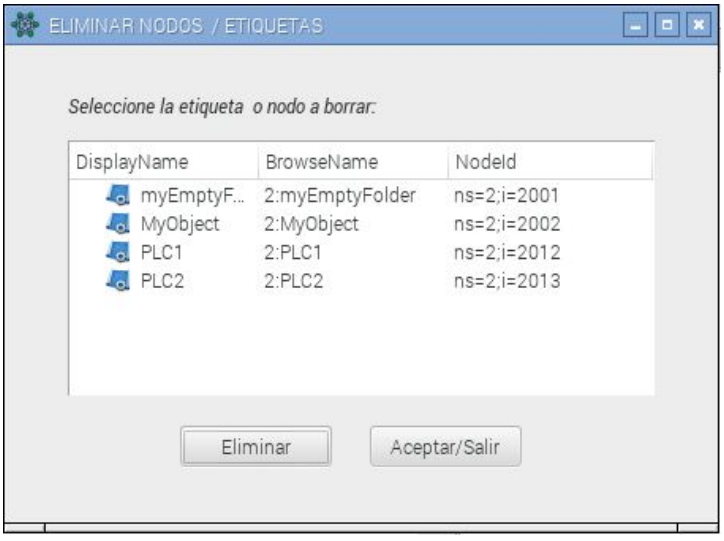


Figura 3.52: Ventana de eliminación de objetos

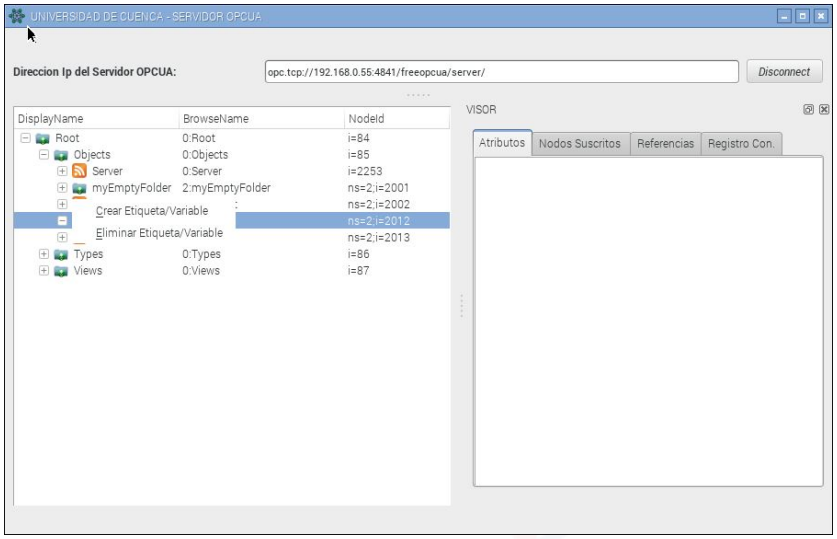


Figura 3.53: Menú que despliega el objeto

Figura 3.54: Ventana de creación de variables

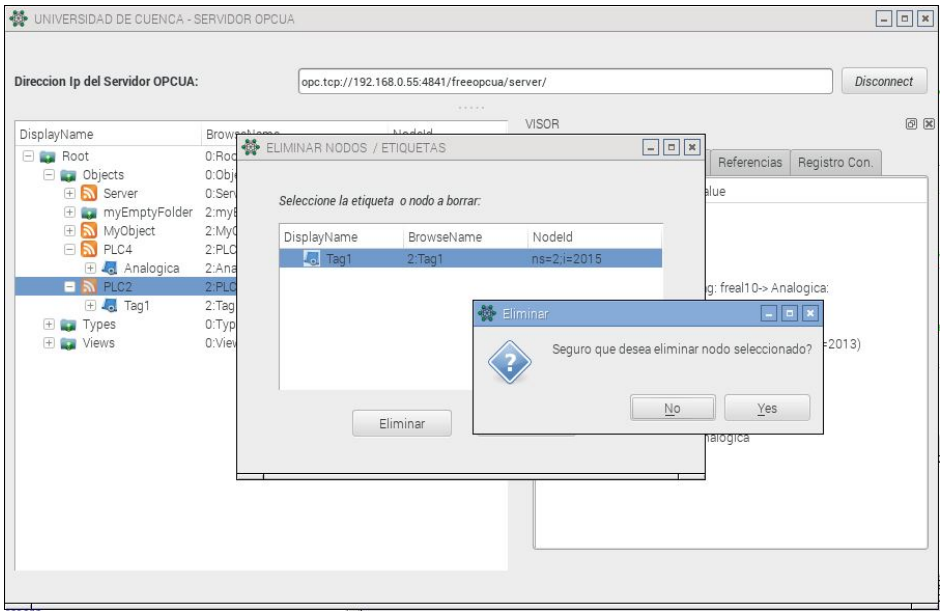


Figura 3.55: Ventana de eliminación de variables

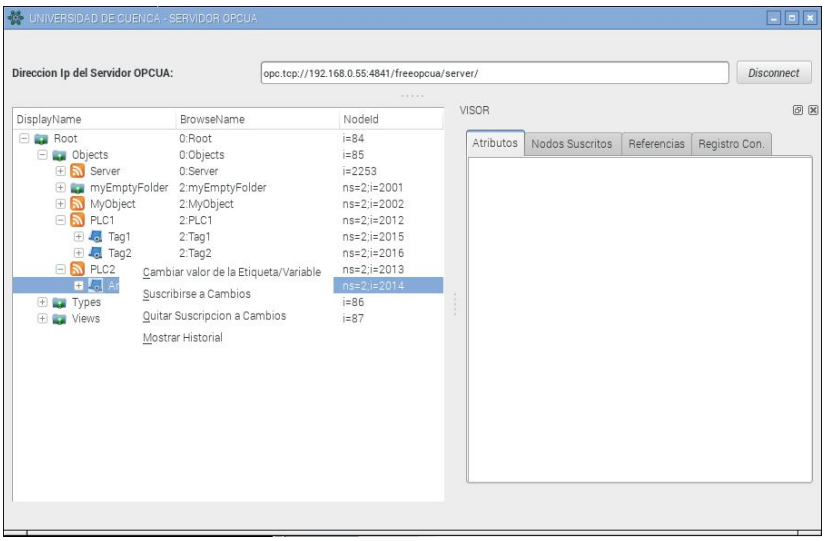


Figura 3.56: Menú que despliega la *tag*

Capítulo 4

Evaluación

4.1. Pruebas

Las pruebas en el servidor se han dividido en dos secciones. Las pruebas de conexión y las pruebas de funcionalidad. Estas pruebas se describen en las subsecciones siguientes:

4.1.1. Pruebas de conexión

La primera forma de verificar que el servidor está trabajando bajo el protocolo OPC UA es mediante la captura del tráfico generado. Debido a que este protocolo se basa en comunicaciones a través de TCP/IP, puede ser capturado mediante el software WireShark.

Wireshark es una herramienta capaz de capturar cualquier tráfico de red. Contiene varios filtros predefinidos para varios protocolos, entre ellos se encuentra OPC UA.

Para capturar el tráfico de entrada y salida del servidor se requieren las siguientes configuraciones:

- Seleccionar el puerto por el cual el servidor transmitirá los paquetes. La configuración se la realiza como se muestra en la Figura 4.1, en donde se elige el protocolo y un rango de puertos, en este caso es el protocolo OPC UA y los puerto 4840, 4841, 4842 definidos en el estándar.
- Seleccionar el tipo se captura. En este caso se seleccionó la interfaz eth0 que corresponde a la interfaz Ethernet del servidor. Además se debe seleccionar el filtro, en este caso *tcp port 4841* como se muestra en la Figura 4.2.

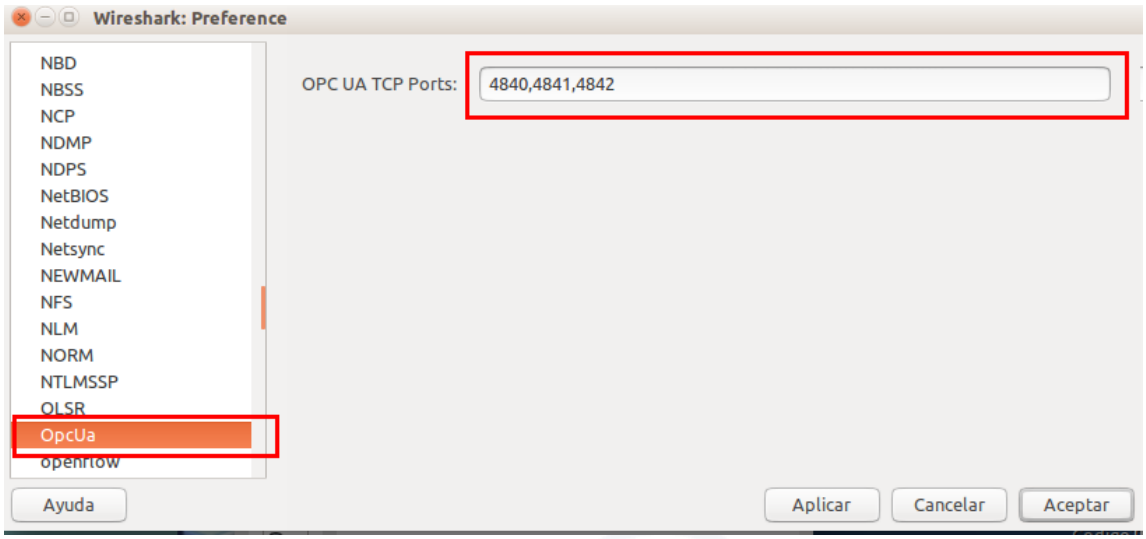


Figura 4.1: Selección del protocolo y puerto en WireShark

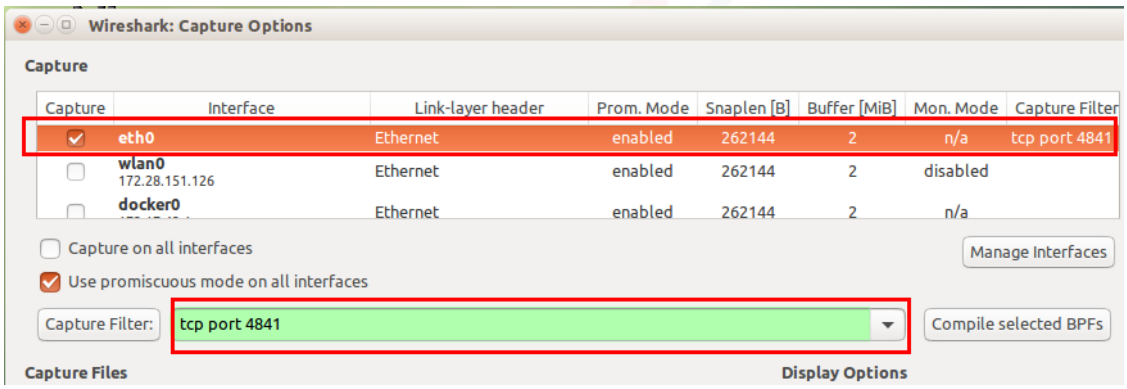


Figura 4.2: Selección del filtro para la captura de trafico OPC UA

Finalmente, cuando un cliente accede al servidor, el tráfico generado es capturado por WireShark como se muestra en la Figura 4.3.

La primera prueba en el servidor con respecto a conexión es satisfactoria. Permite conocer el proceso de transferencia de datos. Además de concluir que efectivamente el protocolo OPC UA se encuentra en marcha.

4.1.2. Pruebas de funcionalidad

Para probar la operabilidad del sistema se ejecutó el servidor con dos clientes.

La Figura 4.4 muestra el servidor ejecutado en la Raspberry PI. Al servidor se conectaron dos PLC a los cuales se los nombró PLC1 y PLC2. Cada uno tiene

No.	Time	Source	Destination	Protocol	Length	Info
15	12.8715560	192.168.0.	192.168.0.50	OpcUa	129	UA Secure Conversation Message
16	12.8722810	192.168.0.	192.168.0.50	OpcUa	122	UA Secure Conversation Message
18	12.8731320	192.168.0.	192.168.0.50	OpcUa	128	CloseSecureChannel message: Cl
27	13.6241520	192.168.0.	192.168.0.50	OpcUa	125	Hello message
29	13.6244230	192.168.0.	192.168.0.50	OpcUa	94	Acknowledge message
31	13.6260370	192.168.0.	192.168.0.50	OpcUa	198	OpenSecureChannel message: Oper
32	13.6267790	192.168.0.	192.168.0.50	OpcUa	205	OpenSecureChannel message: Oper
33	13.6280940	192.168.0.	192.168.0.50	OpcUa	338	UA Secure Conversation Message
34	13.6293350	192.168.0.	192.168.0.50	OpcUa	685	UA Secure Conversation Message
35	13.6331400	192.168.0.	192.168.0.50	OpcUa	222	UA Secure Conversation Message
36	13.6341090	192.168.0.	192.168.0.50	OpcUa	166	UA Secure Conversation Message
37	13.6367700	192.168.0.	192.168.0.50	OpcUa	208	UA Secure Conversation Message
38	13.6377520	192.168.0.	192.168.0.50	OpcUa	200	UA Secure Conversation Message

Figura 4.3: Trafico OPC UA capturado por WireShark

configurado una o dos variables para las primeras pruebas de concurrencia con los clientes.

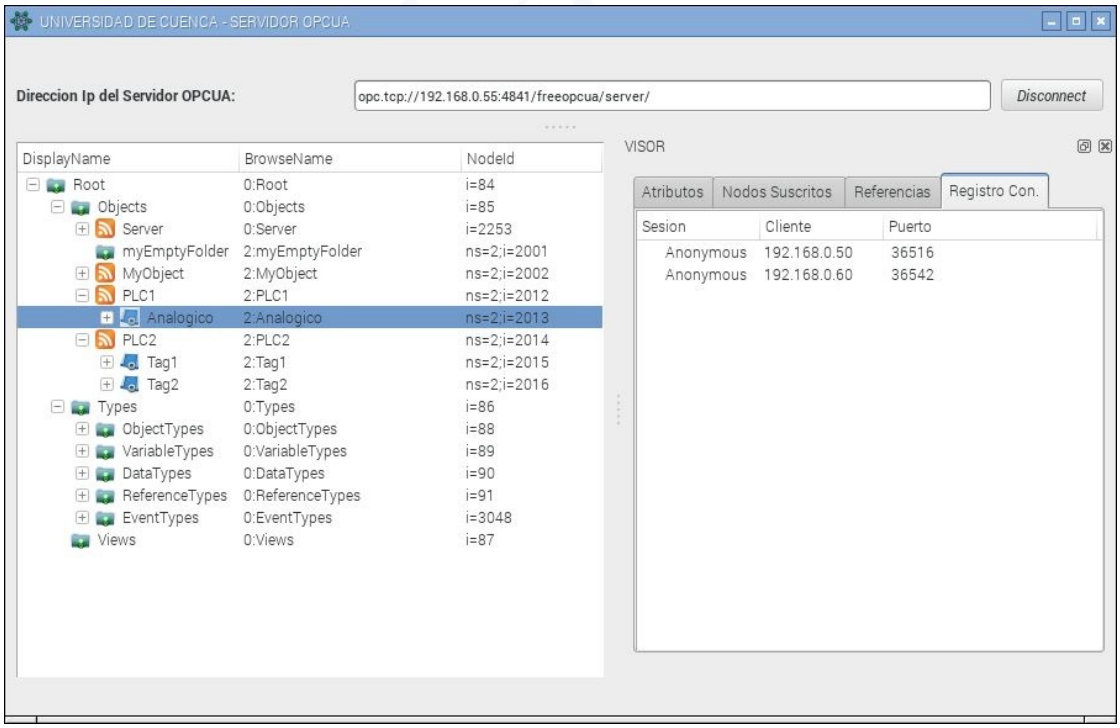


Figura 4.4: Servidor OPC UA ejecutado en la Raspberry PI

En la Figura 4.4, en la parte derecha, se puede observar un registro de conexiones. En esta sección se reconocen los clientes conectados, que para este caso en especial tienen las IPs 192.168.0.50 y 192.168.0.60. Estos clientes fueron ejecutados en deferentes plataformas, Ubuntu y Windows, respectivamente.

La Figura 4.5 muestra el cliente ejecutado en Ubuntu con un software desarrollado en Python. Sin embargo, aún se encuentra en desarrollo, pero permite

comprobar las funcionalidades básicas del servidor.

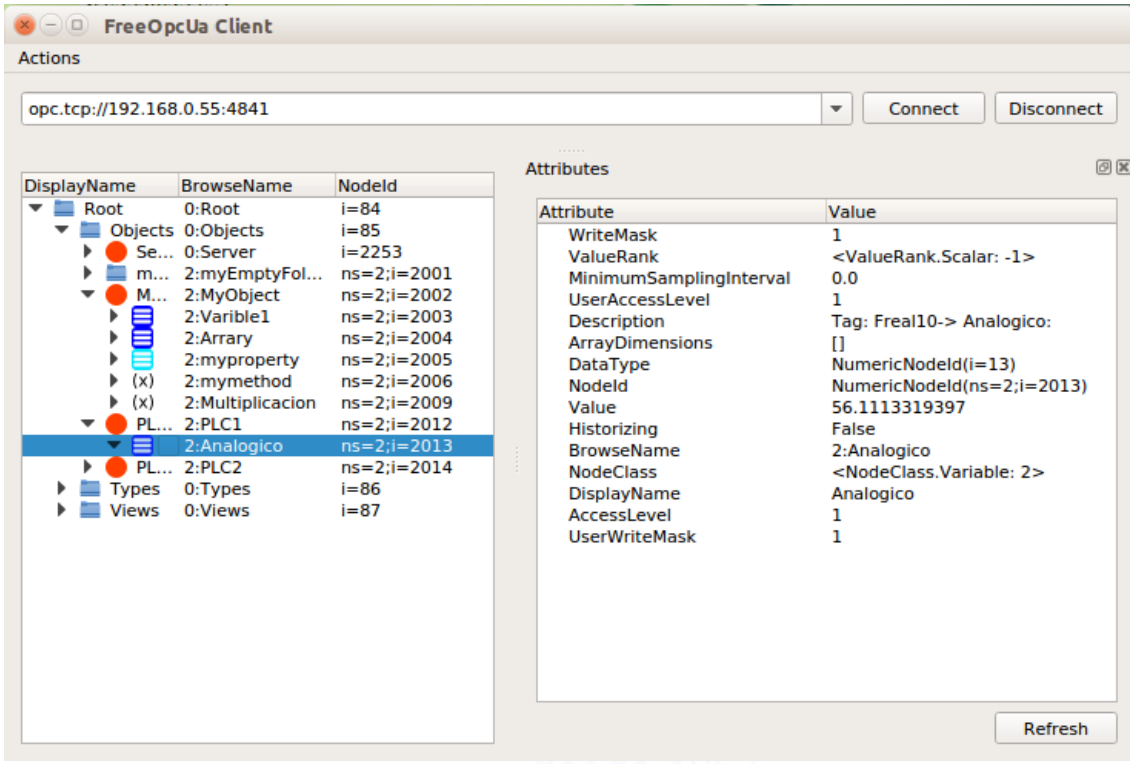


Figura 4.5: Cliente OPC UA ejecutado en Ubuntu

La Figura 4.6 muestra el segundo cliente ejecutado en la plataforma de Windows. Este cliente es desarrollado la empresa alemana *Unified Automation*.

Los cambios realizados en una variable del servidor se reflejan en los clientes. De igual forma un cambio desde un cliente también se refleja en el servidor y en los demás clientes, como se muestra en la Figura 4.7.

Al trabajar bajo las licencias de FreeOpcUa, el tema con respecto a concurrencia está cubierto. Estas librerías manejan métodos de sincronismo de escritura. Evita que dos clientes escriban sobre un mismo nodo al mismo tiempo.

Finalmente, la última prueba consistió en conectar el servidor a todos los PLC disponibles en el laboratorio. Es este caso se conectaron cinco PLC al servidor. A cada PLC se le creó una etiqueta como se muestra en la Figura 4.8.

Las pruebas realizadas muestran la eficacia del sistema. Por lo que, se concluye que el servidor trabaja satisfactoriamente con uno o varios PLCs a la vez. Al mismo tiempo, cumplió con los objetivos al conectar con tres clientes OPC UA

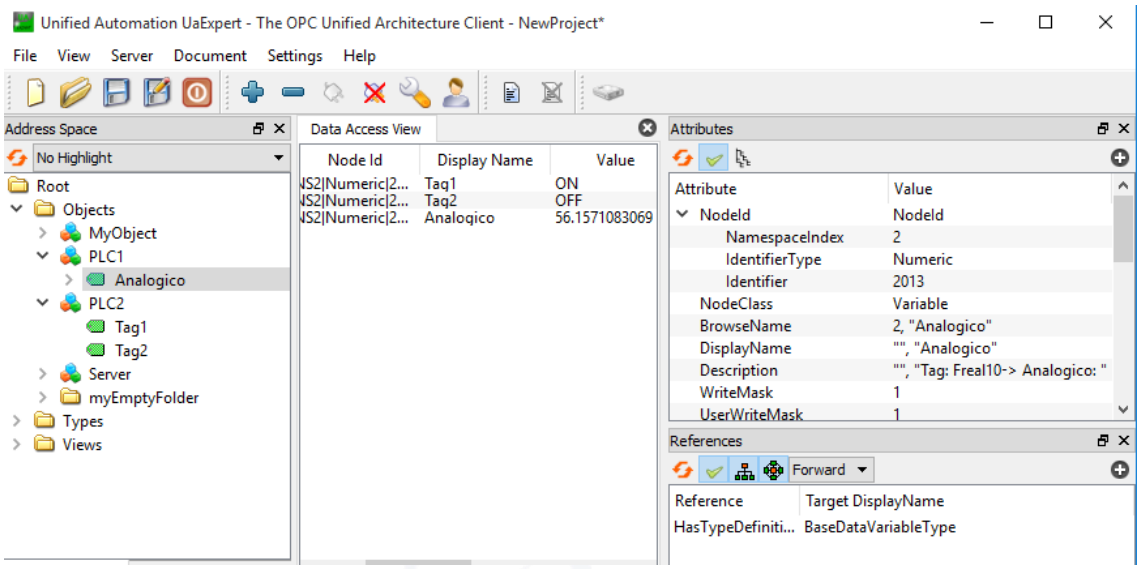


Figura 4.6: Cliente OPC UA ejecutado en Windows

de diferentes plataformas.

Los clientes OPC UA podían realizar todas las prestaciones que ofrece el servidor OPC UA. El cual puede llegar a compararse con un sevidor OPC UA comercial. La ventaja del prototipo está, en su fácil implementación, así como su libre uso y bajo costo en hardware.

4.2. Análisis de resultados

En cuanto a las pruebas de funcionalidad fue posible determinar lo siguiente:

- El tráfico generado, efectivamente, se produce bajo el protocolo OPC UA.
- El mayor tráfico se genera al iniciar la comunicación cliente-servidor, como se muestra en la Figura 4.9.

  En el eje de las abscisas se muestra el tiempo de actividad del servidor y en las ordenadas los bytes generados.
- Al tener baja generación de tráfico, permite al servidor soportar un mayor número de peticiones. Esto conlleva a un mayor número de clientes conectados.
- La plataforma bajo la cual se ejecuta el servidor (Raspberry PI) es capaz de soportar sin dificultad al servidor. Las pruebas se han realizado hasta

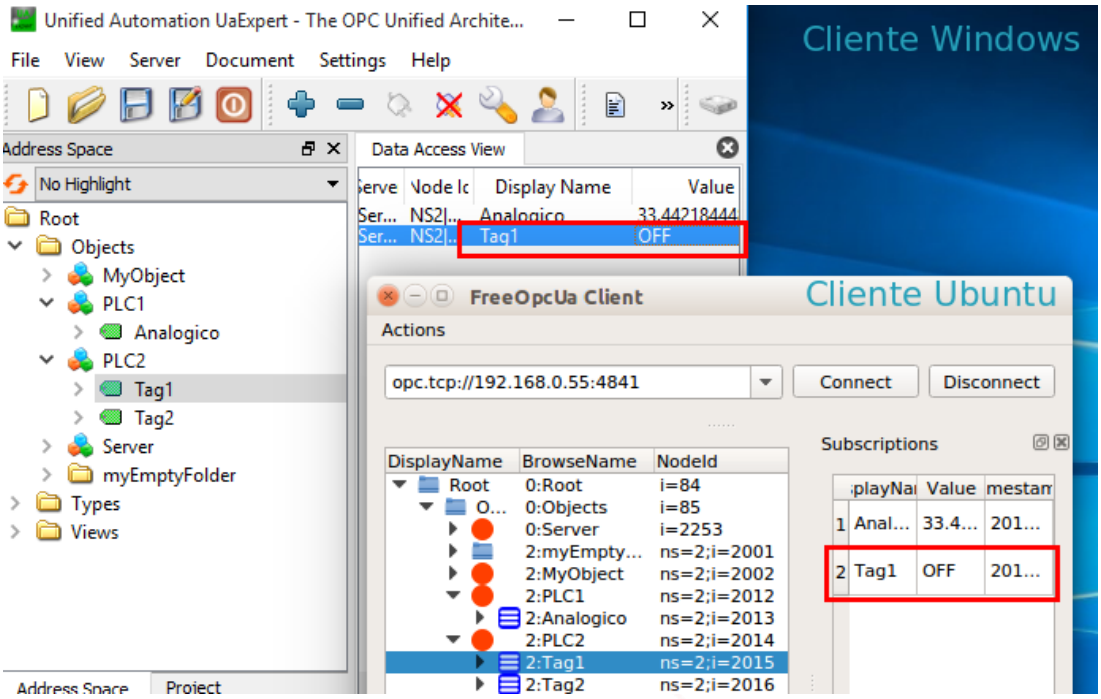


Figura 4.7: Escritura en las variables del servidor desde un cliente

con tres clientes. Sin embargo, se puede conectar un número mayor antes de empezar a saturar al sistema.

Para las pruebas de funcionalidad se emplearon dos clientes en diferentes plataformas. Esto permite determinar la capacidad de la Raspberry PI para trabajar con otro tipo de plataformas. En base a estas pruebas se obtiene las siguientes conclusiones:

- La plataforma bajo la cual se ejecuta el cliente no es un inconveniente para el servidor
- El servidor es capaz de soportar concurrencia de peticiones
- La lectura y escritura de variables funciona de acuerdo a los requerimientos propuesto por el estándar OPC UA
- El servidor se ejecuto por casi tres horas sin presentar inconvenientes o fallas aparentes
- Un cliente puede eliminar nodos. No obstante, esta prueba solo se la pudo realizar en el cliente ejecutado desde Ubuntu. Esto se sebe a que este cliente está escrito en Python y por ende permite la modificación del código en la

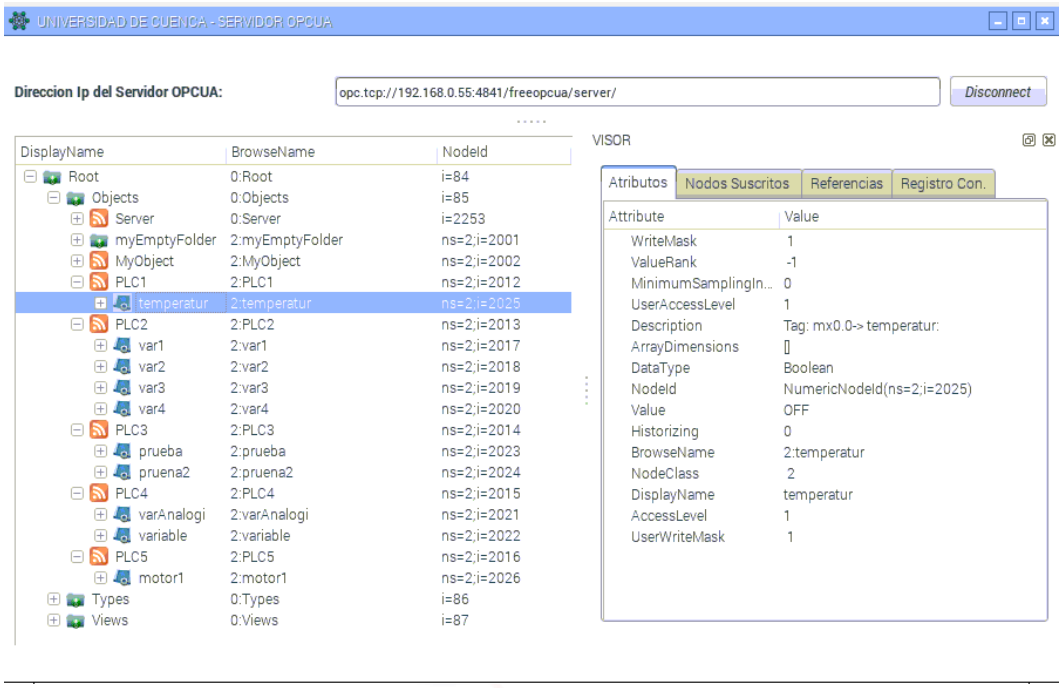


Figura 4.8: Servidor con múltiples PLC conectados

sección de conexión con el servidor. Para llevar a cabo la eliminación el cliente debe tener permisos de administrador

- Para determinar hasta cuántos PLCs se pueden conectar con el servidor, se ejecutó la prueba con cinco PLC Siemens. Pese a ello, el servidor obtuvo buenos resultados. Se estima que se pueden conectar más dispositivos, sin embargo, el procesador de la Raspberry Pi empieza a recalentarse

- Una de las últimas pruebas fue la desconexión repentina de uno de los PLCs, en donde se observó que el servidor es capaz de notificar este error, pese a

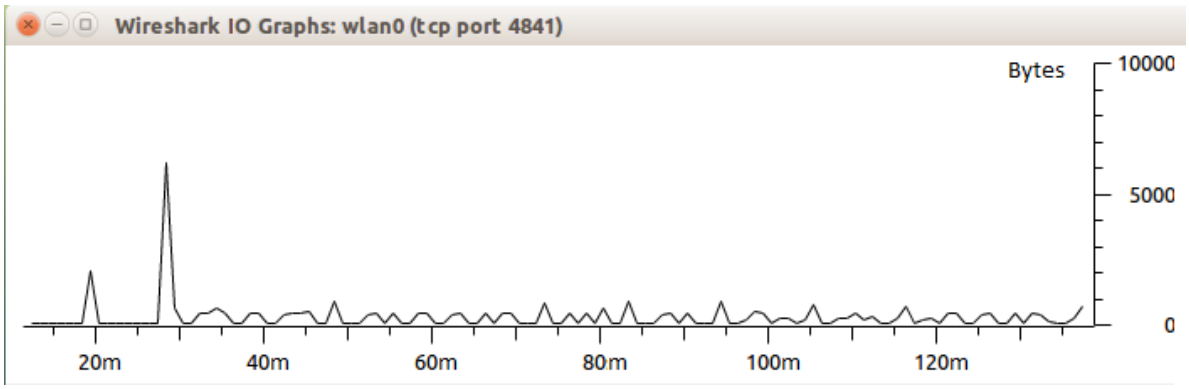


Figura 4.9: Tráfico generado por OPC UA

ello, eso no es visible en el cliente. Al generarse esta falla La Raspberry Pi intenta recuperar la conexión, la cual es posible si el PLC se reconecta a la red. No obstante la conexión es lenta. Cabe destacar que la conexión del resto de PLCs se mantienen funcionales

La Tabla 4.1 muestra un resumen de las pruebas realizadas en el servidor, así como el resultado obtenidos en ellas.

Tabla 4.1: Pruebas realizadas en el servidor

Pruebas	**Rendimiento**		
	Bueno	**Medianamente bueno**	**Insuficiente**
Comunicación bajo el protocolo OPC UA	✓		
Comunicación con el cliente	✓		
Comunicación con el PLC	✓		
Concurrencia por parte del cliente		✓	
Recuperación ante posibles fallos con el PLC			✓
Tiempo medio entre fallas		✓	

El valor medianamente bueno en el tiempo medio entre fallas se debe a que para poder comprobar la funcionalidad del servidor en su totalidad es necesario mas horas de prueba. Aunque, se puede vislumbrar un buen rendimiento con respecto a este punto.

Capítulo 5

Conclusiones

5.1. Conclusiones

Los servidores OPC UA son la parte fundamental para integrar nuevas tecnologías y conceptos en aplicaciones industriales. Este sólido modelo posee una arquitectura capaz de integrar y facilitar, la operatividad de sistemas de producción. La arquitectura de comunicación OPC es de gran importancia a nivel industrial debido a que simplifica la integración de otras aplicaciones como HMI, MES o sistemas SCADA.

El software está basado y desarrollado con librerías de código abierto. Integra varios componentes programados en su totalidad en Python, con el fin de que se ejecute en cualquier plataforma.

Para tener un sistema open-source en su totalidad, es decir tanto en la parte de hardware como en software, se ha visto conveniente montar el servidor OPC UA en una Raspberry Pi. El resultado de esta investigación es un sistema con una arquitectura orientada a servicios, para control industrial en una red local.

Estas características del servidor OPC UA, entregan como beneficio la reducción de costos en licencias de sistema operativo, software de servidores, base de datos y dispositivos electrónicos. A la vez, que es un sistema escalable, ya que por su bajo costo de materiales se puede integrar más servidores y operar de forma que cada servidor tenga un área específica.

El operario encargado de revisar los procedimientos que lleva a cabo el servidor, tiene dos posibilidades de realizarlo. La primera es desde la interfaz gráfica

del propio servidor, ya que el servidor con esta interfaz ofrece interactuar como cliente OPC UA. La segunda opción es desde un cliente OPC UA, ya sea comercial o de código libre, gracias a que el servidor posee una estructura unificada.

El servidor OPC UA es una herramienta muy importante en la industria. Provee un monitoreo y control de una red local de PLCs, de un modo transparente para el usuario. La plataforma Raspberry Pi la cumple satisfactoriamente la función de servidor OPC UA.

Con la finalidad de llevar un mayor control de los procesos a cargo del servidor, se implementó una base de datos. La base de datos guarda los cambios efectuados por las variables de los PLCs. El servidor en su interfaz gráfica ofrece una ventana con los reportes de la fecha seleccionada, donde el usuario se informa de los cambios sucedidos.

5.2. Recomendaciones

Con la finalidad de obtener resultados óptimos en la implementación de este prototipo, se recomienda lo siguiente:

- Para solventar los problemas de sobrecalentamiento y posible introducción de ruido a la Raspberry Pi, se recomienda la creación de un blindaje.

  Este blindaje debe ser capaz de filtrar que el ruido externo producto del propio ambiente industrial. Además, el blindaje debe ofrecer las condiciones necesarias de ventilación para el dispositivo

- De acuerdo al análisis de la sección anterior, se observó que el servidor es capaz de soportar la comunicación con diferentes PLC. Sin embargo, se encuentra limitado por sus características de procesamiento. Por este motivo se recomienda el uso de diferentes Raspberrys para cada proceso de planta. Esto permite que la carga de procesamiento sea distribuida garantizado un mejor resultado

- Si bien el servidor presentado ofrece ventajas a nivel de área local, sus funcionalidades podrían ser mejoradas si se expande su área de cobertura. Una de las tendencias actualidades es el Internet de las cosas. Si se consigue conectar los servicios de este servidor en línea, un operario podría tener

control de toda la planta en desde cualquier lugar. No obstante, para cumplir este objetivo, es importante desarrollar nuevos mecanismos de seguridad definidos en el servidor

- En el transcurso del desarrollado del proyecto, las primeras pruebas se realizaron en un cliente OPC UA básico en dispositivo inteligente (*Smartphone*). Por lo que se recomienda la implementación de una red Wi-Fi a nivel de gestión de acuerdo a la pirámide CIM. Esto permite que un operario pueda saber el estado de los procesos de planta y en cualquier instante

- El prototipo se basa en las librerías de FreeOpcUa, las cuales se encuentran en constante actualización. Por ello, se recomienda un continuo mantenimiento del sistema con el fin de obtener un resultado óptimo.

- Una forma de ampliar la funcionalidad del sistema es mediante la implementación de un sistema SCADA. Si bien el servidor es capaz de informar sobre los procesos de planta es necesario un sistema capaz de adquirir estos datos, procesar y remitir una nueva acción

- En el tiempo de desarrollo de este prototipo uno de los mejores sistemas embebidos es la Raspberry Pi. No obstante, con el desarrollo tecnológico, en la actualidad se encuentra disponible su version Raspberry Pi tercera generación. La ejecución del servidor bajo esta plataforma obtendría un mejor desempeño. Pues su capacidad de procesamiento es superior

5.3. Trabajos futuros

En base a los resultados obtenidos en la sección anterior se presentan las posibles mejoras que puede tener el sistema:

- En cuanto a posibilidad de eliminar y crear nodos remotamente, se puede implementar una mejor forma de registrarse en el servidor. Por ahora solo las conexiones con permiso de administrador pueden llevar a cabo esta tarea. Se podría mejor el prototipo implementando un listado de clientes con permisos especiales, permitiendo un manejo de datos remotos bajo la supervisión del operario desde el servidor

- Si bien el servidor implementa el protocolo OPC UA, su implementación es limitada en contraste con todas las funcionalidades que ofrece el estándar.

Una posible mejora es el desarrollo de la funcionalidad de acceso a datos históricos

El servidor implementa la funcionalidad de acceso de datos, sin embargo, en una industria muchas de las decisiones de control se basan en datos históricos. Si bien el servidor cuenta con un base de datos de todos los nodos, estos datos son sólo accesibles desde el servidor. Esta mejora permitiría al usuario llevar un mejor control y desde cualquier cliente sin la necesidad de emplear otro recurso de software.

- En relación a reconexión de un PLC, se puede implementar otro mecanismo de conexión en el servidor. La forma en la que actualmente las variables de un PLC son obtenidas es mediante hilos. Estos son los encargados de leer las variables y mantener la conexión activa. Sin embargo, podría optarse por un mecanismo de conexión y desconexión.

  Cuando la Raspberry Pi crea un nodo objeto, internamente crea un canal de comunicación con cada PLC en la red. Estos canales permanecen activos en todo momento. Esto permite que el servidor obtenga los valores de las etiquetas creadas en cualquier momento, ahorrando tiempo y recursos. Pero como se analizó en las pruebas, esto no es del todo eficiente. La posible mejora al problema es desarrollar otro algoritmo de conexión. El cual se basa en el mecanismo de conexión con el PLC, obtener sus datos y desconectar. Esto generaría un posible inconveniente, tales como recursos y tiempo empleado. Al existir muchas conexiones con el servidor, el procesador podría saturarse

- El software del servidor está desarrollado en su totalidad en Python, lo que le permite ser un programa multiplataforma. Su inconveniente es la instalación de complementos y librerías. Una posible mejora consiste en la compilación del mismo en un archivo ejecutable, permitiendo ejecutar el servidor en cualquier máquina, sin necesidad de instalar ningún complemento. Si se desea ejecutar el servidor en Windows, es indispensable crear el ejecutable desde esa plataforma, lo mismo aplica para sistemas basadas en Linux. Esto de debe a que cada sistema tiene sus propia extension para archivos ejecutables

Apéndices

Apéndice A

Requisitos previos para la instalación del servidor

A.1. Instalación del Sistema Operativo

Los pasos necesarios para la instalación del sistema operativo en el Raspberry Pi son:

1. Se recomienda el formateo de la tarjeta SD, antes de copiar los archivos en la misma
2. Se copia todos los archivos descargados de la pagina oficial *https://www.raspberrypi.org*
3. Al iniciar por primera vez la Raspberry Pi instalamos y configuramos el sistema operativo Raspbian
4. El nombre de usuario predeterminado para Raspbian es pi, con la contraseña de raspberry

A.2. Requisitos para instalar el servidor

Para la ejecución del servidor se necesita instalar los siguientes programas:

1. Python 2.7 para programar el software de la página oficial *https://www.python.org/download/releases/2.7/*. Se escogió la versión 2.7, ya que es la versión más estable
2. PyQtDesigner para crear la interfaz del servidor

3. FreeOpcUa para Python. Se instala las librerías necesarias para que se ejecute los métodos del servidor en Python

   Para ejecutar FreeOpcUa se necesita la instalación de las siguientes librerías:

   a) **Cryptography:** Es una librería de Python que sirve para utilizar métodos primitivos

   b) **Futures:** Este módulo proporciona una interfaz de alto nivel para ejecutar llamadas asíncronas

   c) **Trollius:** Proporciona la infraestructura para escribir código concurrente de un solo subproceso, usando corrutinas, acceso multiplexado de entrada y salida, a través de sockets y otros recursos, ejecutar clientes y servidores de red

   d) **Enum34:** Una enumeración es un conjunto de nombres simbólicos (miembros) unidos a valores únicos y constantes. Este módulo define dos clases de enumeración que se pueden utilizar para definir conjuntos únicos de nombres y valores

A.3. Instalación de librerías necesarias para el servidor

Para la ejecución del servidor se necesita instalar las siguientes librerías:

1. **Librería PyQt:** Es una biblioteca gráfica de Qt realizada para el lenguaje de programación Python. Brinda la opción de integrar métodos para la interfaz gráfica del servidor.

2. **Librería Snap7:** Es una librería destinada a la comunicación entre los PLCs y el servidor.

A.4. Configuraciones de IP y puertos de conexión

Para la ejecución del servidor se necesita configurar una IP fija y los puertos de conexión de la Raspberry Pi:

1. **Configuración de IP fija:** Para configurar una IP estática en la Raspberry Pi, es necesario editar el fichero que se encuentra en la dirección *sudo nano /etc/network/interfaces* como se indica en la figura A.1.

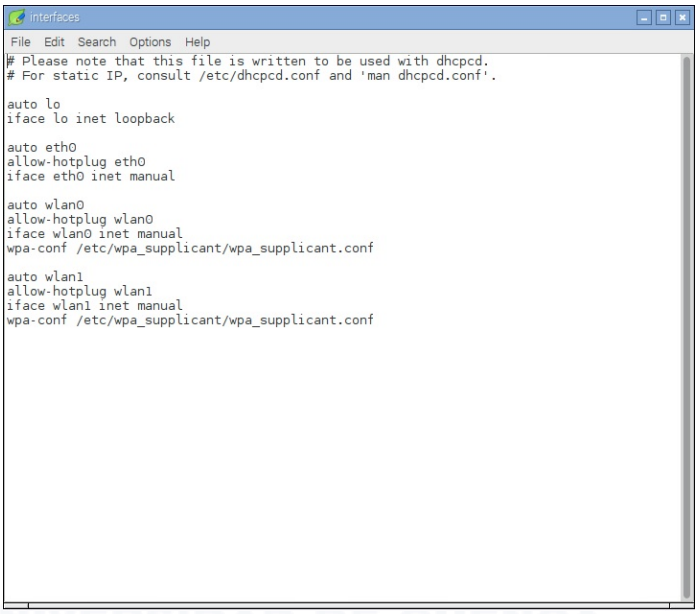


Figura A.1: Archivo interfaces

Ahora, se cambia los datos seleccionados en la figura A.2

Luego de cambiar los datos, se reinicia los servicios de red desde la terminal con el comando *sudo /etc/init.d/networking restart* ó a su vez se reinicia la Raspberry Pi manualmente.

2. **Configuración de puertos de comunicación:** Se debe instalar un *firewall* con el fin de proteger nuestra Raspberry Pi. Para ello se instala el cortafuegos sin complicaciones (ufw, por sus siglas en inglés) y se configura con los siguientes pasos:

    1. Se instala desde la terminal con el comando *sudo apt-get install ufw*
    2. Se escribe el comando *sudo ufw default deny incoming* y el comando *sudo ufw default allow outgoing* para configurar la entrada y salida de datos
    3. Para generar excepciones, es decir abrir puertos con nuestro consentimiento se debe configurar los siguientes puertos:
        a. Permitiremos la entrada por el puerto 22 tcp, que es el puerto estándar que corresponde al servidor SSH con el comando *sudo ufw allow 22/tcp*

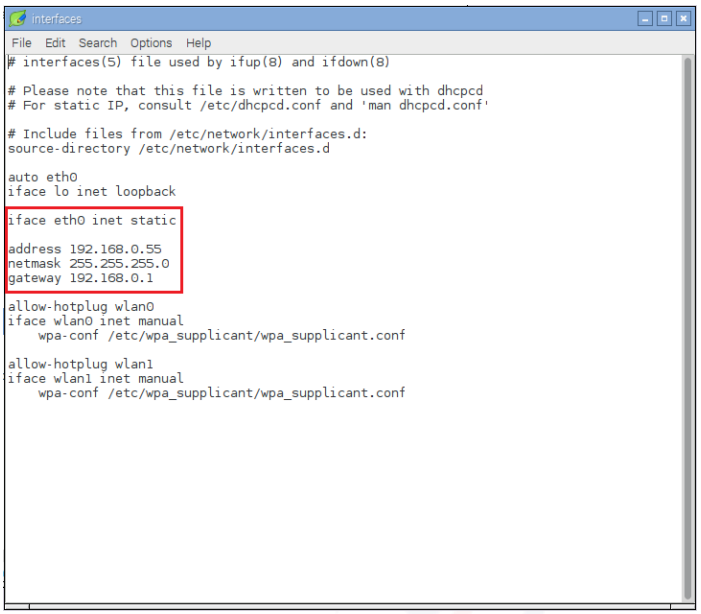


Figura A.2: Cambios al archivo interfaces

b. Permitimos la entrada y salida de datos para el servidor web que tengamos instalado, con el comando *sudo ufw allow 80/tcp*

c. Se debe habilitar los puertos de conexión del servidor OPC UA, para ello insertamos los comandos *sudo ufw allow 4840/tcp* y *sudo ufw allow 4841/tcp*

d. Habilitamos los puertos 5900 y 5901 de la Raspberry Pi, con el fin de que se pueda conectar la librería Snap7 con los PLCs. Insertamos los siguientes comandos *sudo ufw allow 5900/tcp* y *sudo ufw allow 5901/tcp*

Apéndice B

Diagrama de clases del sistema

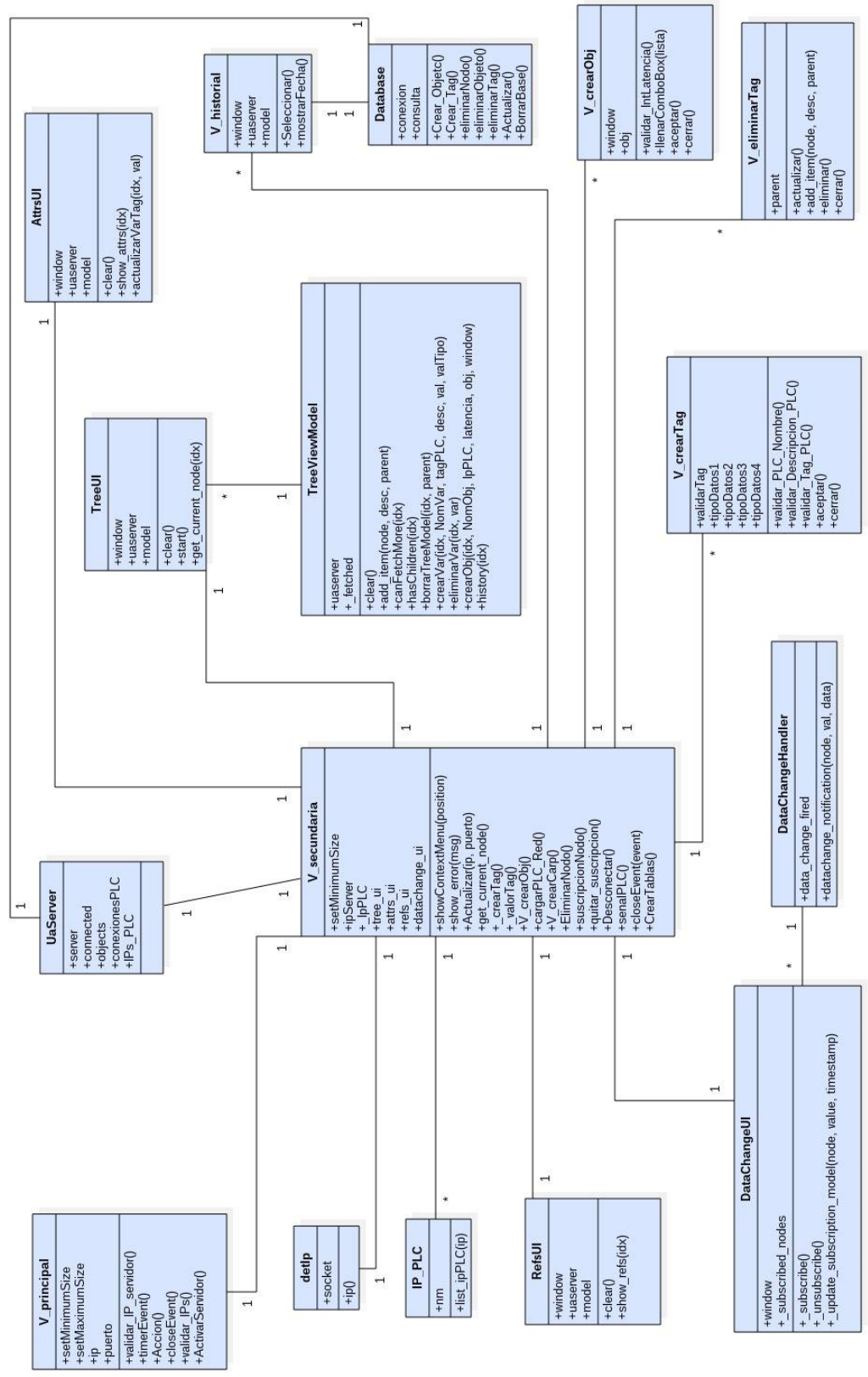


Figura B.1: Diagrama de clases del servidor OPC UA

Bibliografía

[1] Yang Chuanying, Li He, and Liu Zhihong. Implementation of migrations from class OPC to OPC UA for data acquisition system. In *System Science and Engineering (ICSSE), 2012 International Conference on*, pages 588–592. IEEE.

[2] R. Huang, F. Liu, and Pan Dongbo. Research on opc ua security. In *Industrial Electronics and Applications (ICIEA), 2010 the 5th IEEE Conference on*, pages 1439–1444, June 2010.

[3] José Armesto Q. Instalación de sistemas de automatización y datos. *Phys. Rev. Lett.*, June 2007.

[4] H. Lu and Yan Zhifeng. Research on key technology of the address space for opc ua server. In *Advanced Computer Control (ICACC), 2010 2nd International Conference on*, volume 3, pages 278–281, March 2010.

[5] Siemens. *How can you establish a connection between an S7-1200 PLC and SIMATIC NET OPC?* Elsevier Newnes, 1st ed edition.

[6] François Jammes and Harm Smit. Service-oriented paradigms in industrial automation. *Industrial informatics, IEEE transactions on*, 1(1):62–70, 2005.

[7] RW Lewis. Programming industrial control systems using iec 1131-iee 1998. Technical report, ISBN 0-85296-950-3 229 230 BIBLIOGRAFÍA.

[8] Karl-Heinz John and Michael Tiegelkamp. *IEC 61131-3: programming industrial automation systems: concepts and programming languages, requirements for programming systems, decision-making aids*. Springer Science & Business Media, 2010.

[9] IEC Iec. 61131-3: Programmable controllers–part 3: Programming languages. *International Standard, Second Edition, International Electrotechnical Commission, Geneva*, 1:2003, 2003.

[10] Kleanthis Thramboulidis, Georg Frey, et al. Towards a model-driven iec 61131-based development process in industrial automation. *Journal of Software Engineering and Applications*, 4(04):217, 2011.

[11] Kleanthis Thramboulidis. Service-oriented architecture in industrial automation systems-the case of iec 61499: A review. *arXiv preprint arXiv:1506.04615*, 2015.

[12] Tommaso Cucinotta, Antonio Mancina, Gaetano F Anastasi, Giuseppe Lipari, Leonardo Mangeruca, Roberto Checcozzo, and Fulvio Rusinà. A real-time service-oriented architecture for industrial automation. *Industrial Informatics, IEEE Transactions on*, 5(3):267–277, 2009.

[13] Wenbin Dai, Valeriy Vyatkin, James H Christensen, and Victor N Dubinin. Bridging service-oriented architecture and iec 61499 for flexibility and interoperability. *Industrial Informatics, IEEE Transactions on*, 11(3):771–781, 2015.

[14] Rubén Darío Sánchez Dams. Camino hacia la creación de clientes y servidores bajo el estándar ua de la fundación opc. *INGE CUC*, 6(1):157–166, 2010.

[15] Daniel Grossmann, Klaus Bender, and Benjamin Danzer. Opc ua based field device integration. In *SICE Annual Conference, 2008*, pages 933–938. IEEE, 2008.

[16] Jouko Virta, Ilkka Seilonen, Antti Tuomi, and Kari Koskinen. Soa-based integration for batch process management with opc ua and isa-88/95. In *Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on*, pages 1–8. IEEE, 2010.

[17] Stamatis Karnouskos, Oliver Baecker, Luciana Moreira Sá De Souza, and Patrik Spiess. Integration of soa-ready networked embedded devices in enterprise systems via a cross-layered web service infrastructure. In *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on*, pages 293–300. IEEE, 2007.

[18] Ante Martinić, Denis Francesconi, et al. Utilizing soa-ready devices for virtual power plant control in semantic-enabled smart grid analyzing iec 61850 and opc ua integration methodology. In *Smart Grid Communications (SmartGridComm), 2011 IEEE International Conference on*, pages 43–48. IEEE, 2011.

[19] W. Bolton. *Programmable logic controllers*. Elsevier Newnes, 4th ed edition.

[20] J. Blanco. *COMUNICACIONES INDUSTRIALES*. Elsevier Newnes, 1st ed edition.

[21] K. M. Middaugh. A comparison of industrial communications networks. *IEEE Transactions on Industry Applications*, 29(5):846–853, Sep 1993.

[22] Steve Mackay. *Practical industrial data networks: design, installation and troubleshooting*. Newnes, 2004.

[23] E. Schneider. *Guía soluciones de automatización y control industrial*. Schneider Electric, 1st ed edition.

[24] Li Zheng and Hiroyuki Nakagawa. OPC (OLE for process control) specification and its developments. In *SICE 2002. Proceedings of the 41st SICE Annual Conference*, volume 2, pages 917–920. IEEE.

[25] Mai Son and Myeong-Jae Yi. A study on OPC specifications: Perspective and challenges. In *Strategic Technology (IFOST), 2010 International Forum on*, pages 193–197. IEEE.

[26] Sebastian Lehnhoff, Sebastian Rohjans, Mathias Uslar, and Wolfgang Mahnke. OPC unified architecture: A service-oriented architecture for smart grids. In *Proceedings of the First International Workshop on Software Engineering Challenges for the Smart Grid*, pages 1–7. IEEE Press.