ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

IMPLEMENTACIÓN DE UN SISTEMA DE ADQUISICIÓN DE DATOS Y SU REGISTRO EN LA NUBE BASADO EN ARQUITECTURA UNIFICADA DE COMUNICACIONES DE PLATAFORMA ABIERTA (OPC UA)

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE MAGISTER EN ELECTRÓNICA Y AUTOMATIZACIÓN, MENCIÓN REDES INDUSTRIALES

ANTHONY DAVID MOLINA ANCHATUÑA

DIRECTOR: ING. DIEGO DANILO VARGAS CULQUI MSC

CODIRECTOR: ING. ANA VERÓNICA RODAS BENALCÁZAR MBA

Quito, agosto 2023

AVAL

Certifico que el presente trabajo fue desarrollado por Anthony David Molina Anchatuña bajo nuestra supervisión.

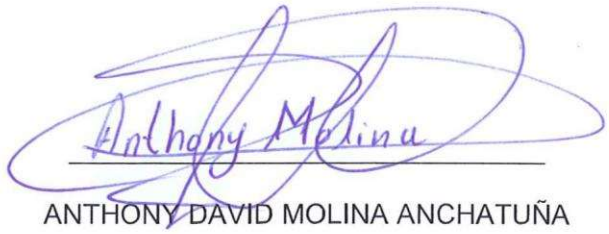
DIEGO DANILO VARGAS CULQUI
DIRECTOR DEL TRABAJO DE TITULACIÓN

ANA VERÓNICA RODAS BENALCÁZAR
CODIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, Anthony David Molina Anchatuña, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejó constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.



ANTHONY DAVID MOLINA ANCHATUÑA

DEDICATORIA

A Orlando mi padre:

Por la vida, las alas y raíces.

A Eugenia mi madre:

Por la vida, amor y cariño.

A Gabriela mi hermana:

Por su bondad.

A mis familiares

Por lo que tan generosamente me han apoyado.

A mis amigos

Por la fraternidad

AGRADECIMIENTO

Expreso mi gratitud a Dios por brindarme la oportunidad de la vida y por su ayuda en la finalización de otro capítulo más en mi vida. Quiero agradecer a mis padres y hermana por su aliento y respaldo, que me han permitido alcanzar este objetivo.

A mis familiares que siempre he encontrado en ellos la unidad y apoyo.

Afortunadamente este camino no se ha recorrido en solitario, he tenido la suerte de recorrerlo con amigos.

A mis tutores, el Ingeniero Diego Vargas y la Ingeniera Ana Rodas, por brindarme su apoyo y orientación durante el proceso de desarrollo de este proyecto. Su contribución fue fundamental para llevar a cabo exitosamente este trabajo.

A todos ustedes, desde mi corazón, mi gratitud y mi inmenso cariño.

ÍNDICE DE CONTENIDO

RESUMEN

La estandarización e integración colaborativa son fundamentales para la implementación de la Industria 4.0. En este contexto, el estándar de Plataforma de Comunicación Abierta de Arquitectura Unificada (OPC UA) juega un papel crucial en la comunicación al permitir el desarrollo de sistemas heterogéneos y facilitar el intercambio fluido de datos entre dispositivos. Para aprovechar al máximo las capacidades de OPC UA es necesario desbloquear otros servicios de aplicación, como la computación en la nube. Al disponer de más herramientas, las industrias pueden aumentar su eficiencia y optimizar la toma de decisiones a partir de datos.

En este estudio, se propone utilizar el estándar OPC UA en el modelo Cliente/Servidor para aprovechar los paradigmas de OPC UA junto con los Kits de Desarrollo de Software (SDK). El desarrollo se llevó a cabo utilizando Python como lenguaje de programación, tanto para alojar el servidor en una Raspberry Pi 4B como para establecer el cliente en un Computador Personal (PC). Este cliente centralizará los datos para su posterior envío a las plataformas de registro en la nube de Clever Cloud y para su visualización en la Web a través de Tago.IO.

Además, se ha abordado el problema de interoperabilidad al comunicar dos plataformas con distintos Sistemas Operativos (OS), mediante la integración del estándar OPC UA. El intervalo configurado de 10 segundos registró un promedio de 11,11 segundos, con un tiempo de espera máximo de 59 segundos para el registro de datos: temperatura, presión, humedad y altitud. El cliente demostró ser compatible con el Protocolo de Transferencia de Hipertexto Seguro (HTTPS), lo que permitió la conexión a las plataformas Web.

PALABRAS CLAVE: OPC UA, Industria 4.0, Monitoreo, Nube, Python.

ABSTRACT

Standardization and collaborative integration are fundamental for the implementation of Industry 4.0. The Open Communication Platform Unified Architecture (OPC UA) standard plays a crucial role in communication by enabling the development of heterogeneous systems and facilitating the seamless exchange of data between devices. To take full advantage of OPC UA's capabilities, it is necessary to unlock other application services, such as cloud computing. By having more tools at their disposal, industries can increase their efficiency and optimize data-driven decision-making.

In this study, it was proposed to use OPC UA standard in Client/Server model to leverage OPC UA paradigms along with Software Development Kits (SDKs). The development was carried out using Python as the programming language, both to host the server on a Raspberry Pi 4B and to set up the client on a Personal Computer (PC). This client will centralize the data for subsequent sending to the Clever Cloud registration platforms and for visualization on the Web through Tago.IO.

Furthermore, the interoperability issue has been addressed when communicating two platforms with different Operating Systems (OS), by integrating the OPC UA standard. The configured interval of 10 seconds recorded an average of 11.11 seconds, with a maximum waiting time of 59 seconds for data logging: temperature, pressure, humidity, and altitude. The client proved to be compatible with the Secure Hypertext Transfer Protocol (HTTPS), allowing connection to Web platforms.

KEYWORDS: OPC UA, Industry 4.0, Monitoring, Cloud, Python.

1. INTRODUCCIÓN

En la Industria 4.0, los sistemas de control y monitorización industrial necesitan adaptarse constantemente a las últimas tendencias, lo cual implica una continua investigación e innovación, la cual ofrece herramientas como minería de datos, internet de las cosas, automatización, fabricación aditiva, computación en la nube y energías renovables [1], que generan ventajas para las compañías para la creación de valor, servicios auxiliares, organización del trabajo y modelos de negocio [2]. El desafío es unificar y compartir información, por lo tanto, la independencia de la tecnología de comunicación será clave ya que en una red empresarial se presentan diferentes aplicaciones, protocolos industriales y sistemas operativos, la integración de estas aplicaciones puede llevar mucho tiempo y ser técnicamente desafiante.

En [3], se afirma que en la actualidad existen veinte protocolos de comunicación basados en Internet Industrial (EI), necesitando una extensa variedad de Tecnologías de la Información (TI) con enfoques en los niveles de aplicación, lo que conlleva a obtener parcialmente la conectividad estructural e interoperabilidad dentro de la propia industria [4].

EtherNet/IP, PROFINET o EtherCAT son ampliamente utilizados y corresponden a protocolos de comunicación abiertos, sin embargo, son incompatibles, lo que provoca una fragmentación de la red y frente a un crecimiento del sistema de automatización se dificultará aún más el compartir información recibida con las TI [4], lo que exige el desarrollo de una arquitectura de comunicación que estandarice la interconexión entre diferentes fabricantes.

Se plantea utilizar un protocolo de comunicación que permita la interoperabilidad y conectividad estructural total en una empresa, el protocolo propuesto deberá ampliar las capacidades de comunicación, además de ser flexible y escalable, lo que podrá impulsar a la Industria 4.0, ya que requiere un intercambio estandarizado y seguro de información.

Los protocolos de comunicación actuales proponen diversas soluciones para aportar a la interoperabilidad entre los niveles de la Manufactura Integrada por Computadora (CIM). El estándar que está siendo ampliamente utilizado es OPC UA [5], el cual permite la armonización de los diferentes protocolos existentes, ya que cumple con las siguientes características: equivalencia funcional, independencia de la plataforma, comunicación segura, diseño extensible y modelado integral de la información [6]. El estándar definido para OPC UA es IEC 62541. Desde una perspectiva económica, contar con OPC UA en dispositivos de hardware libre resultaría ventajoso, ya que para industrias que no

dispongan del hardware que permita utilizar esta arquitectura, podrán optar por tarjetas de desarrollo. El artículo publicado por [7] define a Arduino como un desarrollador de proyectos de baja precisión, pero su alta eficiencia y bajo costo lo hacen ideal para el desarrollo de propuestas.

Con base en lo expuesto, se propone utilizar el estándar OPC UA en tarjetas de desarrollo configuradas por software libre para la realización de un sistema de monitoreo para el registro de datos en la nube y la visualización Web utilizando plataformas gratuitas, permitiendo la estandarización de la comunicación a nivel de campo y gestión, lo cual favorecerá a su vez a la interoperabilidad con las TI y las herramientas de la Industria 4.0.

1.1. Objetivo General

Implementar un sistema de adquisición de datos y su registro en la nube basado en la Plataforma de Comunicación Abierta de Arquitectura Unificada (OPC UA).

1.2. Objetivos Específicos

- Comprender la Plataforma de Comunicación Abierta de Arquitectura Unificada, seleccionar los dispositivos en los cuales se establecerá comunicación (dispositivos de campo - computador) y estudiar las alternativas para el registro de datos en la nube.
- Diseñar e Implementar una solución tecnológica que permita monitorear dispositivos de campo desde un equipo y su registro en una base de datos gratuita, basada en OPC UA en el modelo Cliente/Servidor.
- Diseñar e Implementar una maqueta que simulara un ambiente de pruebas que permita medir las variables físicas de temperatura, presión y humedad.
- Realizar pruebas para validar el funcionamiento del sistema a través de la emulación de un proceso que involucre los dispositivos conectados.

1.3. Alcance

Para el proyecto se monitorizará tres variables temperatura, presión y humedad las cuales serán adquiridas por un transductor, en la implementación se utilizará dispositivos no industriales de hardware libre, siendo la Raspberry Pi la plataforma a utilizar, el modelo de esta tarjeta de desarrollo se seleccionará a partir del análisis bibliográfico correspondiente. La tarjeta de desarrollo recibirá las señales eléctricas de los transductores.

El protocolo de comunicación a implementar será OPC UA en el modelo Cliente/Servidor, en la tarjeta de desarrollo se instalará un servidor OPC UA para centralizar la información obtenida desde campo esta información se enviará a un cliente OPC UA, representado por un computador con el fin de leer los datos proporcionados por el servidor. La programación de la tarjeta de desarrollo (servidor) y el computador (cliente) se llevará a cabo con la herramienta de desarrollo Python.

La programación del cliente incluirá el código para la visualización Web y el registro de datos en la nube. La aplicación se conectará a plataformas Web de acceso libre, con el propósito de visualizar los datos obtenidos en campo junto con un registro. Los datos podrán visualizarse o descargarse por cualquier PC o dispositivo inteligente conectado a internet.

Se construirá una maqueta para realizar las pruebas del proyecto propuesto. Durante este proceso, se registrará la temperatura, presión y humedad mediante un registrador de datos junto al proyecto, con el propósito de contrastar los resultados adquiridos durante los experimentos. Estas pruebas se harán durante ocho horas para verificar la disponibilidad y estabilidad del proyecto.

Por último, se realizará un análisis del trabajo realizado respecto al estándar de comunicación OPC UA, su implementación y desempeño en un ambiente de pruebas, además se elaborarán las conclusiones, recomendaciones y trabajos futuros.

1.4. Marco Teórico

En la industria, los protocolos de comunicación desempeñan un papel clave para los diferentes dispositivos electrónicos, ya que facilitan la comunicación entre la parte de hardware y software, permitiendo un intercambio fluido de información. Cada protocolo de comunicación tiene sus propias reglas, sintaxis, semántica y sincronización. Los sistemas de comunicación utilizan protocolos predefinidos para intercambiar mensajes con un significado específico, dirigidos a un destinatario, con el fin de obtener una respuesta en situaciones determinadas.

En particular, la fundación OPC ha desarrollado dos estándares clave para la comunicación en entornos industriales. El primero es el estándar de Comunicaciones de Plataforma Abierta (OPC) Clásica, que fue creado inicialmente para permitir la interoperabilidad entre sistemas de monitoreo. Este estándar se ha utilizado ampliamente en la industria durante muchos años y ha demostrado su eficacia en la comunicación entre dispositivos y sistemas. Sin embargo, con la evolución de la Industria 4.0, surgieron nuevos estándares de

comunicación que se enfocan en la seguridad, escalabilidad y eficiencia energética como es el caso del estándar OPC UA.

El estándar OPC UA se presenta como una solución para la comunicación entre sistemas de monitoreo en entornos industriales. Este estándar proporciona interoperabilidad entre diferentes dispositivos y sistemas sin importar el fabricante o el sistema operativo utilizado. También, integra los distintos niveles jerárquicos de la automatización, lo que lo hace especialmente relevante para la Industria 4.0.

Se ha desarrollado una guía documentada que facilite la implementación del estándar OPC UA en conjunto con Python. En adición, se proporcionarán herramientas para registrar los datos en la nube y visualizarlos en una plataforma de monitoreo Web.

Para iniciar se presenta un análisis bibliográfico de la Industria 4.0, revisando su arquitectura de referencia, los requisitos específicos de esta industria emergente y cómo OPC UA puede solucionarlos; a continuación, se llevará a cabo un estudio detallado de OPC UA para comprender sus especificaciones y características; después, se examinarán los requisitos para la creación de un proyecto de monitorización y por último, se revisarán los dispositivos de hardware libre disponibles en el mercado que puedan ser utilizados en el proyecto propuesto.

1.4.1. Industria 4.0

La Industria 4.0, constituye la más reciente transformación industrial, la cual busca la convergencia de sistemas industriales con tecnologías avanzadas como la Internet Industrial de las Cosas (IIoT), Internet de las Cosas (IoT), Internet de Servicios (IoS), Sistemas Ciber Físicos (CPS), Inteligencia Artificial (IA) entre otras tecnologías [8]. La meta primordial de la Industria 4.0 es llegar a sistemas autónomos y eficientes a través del intercambio de datos y la comunicación en red entre diferentes dispositivos y áreas.

Uno de los retos más significativos de la Industria 4.0 es la interoperabilidad, es decir, la capacidad de diferentes dispositivos y sistemas para intercambiar información y trabajar juntos de manera efectiva [9]. Dado que existen muchos protocolos de comunicación diferentes, la falta de compatibilidad entre ellos puede dificultar la interoperabilidad y puede provocar la falta de comunicación entre nodos o dispositivos, lo que dificulta el intercambio de información. Por lo tanto, es fundamental que los sistemas y dispositivos estén diseñados para trabajar con una variedad de protocolos de comunicación, asegurándose así una comunicación fluida entre ellos.

Otro desafío importante en la Industria 4.0 es la protección y confidencialidad de los datos producidos por diversos dispositivos y sistemas [8]. La extensa cantidad de datos recabados y compartidos puede incluir información confidencial, como datos de producción y de clientes y cualquier brecha de seguridad podría ser perjudicial para la empresa. Por lo tanto, es importante implementar medidas de seguridad y privacidad adecuadas para proteger los datos.

1.4.1.1. Arquitectura de referencia de la Industria 4.0

La Asociación Alemana de Fabricantes y Equipos Electrónicos tuvo a su cargo el desarrollar el Modelo de la Arquitectura de Referencia de la Industria 4.0 (RAMI 4.0). Esta nueva industria es una iniciativa mundial que busca modernizar la fabricación a través de la integración de tecnologías avanzadas y la automatización.

La RAMI 4.0 es una representación espacial en tres dimensiones que muestra los aspectos claves de la Industria 4.0, proporcionando un enfoque estandarizado para impulsar la creación de próximos productos y esquemas de negocio. Esto se fundamenta en la Arquitectura Orientada a Servicios (SOAP) dividiéndose en tres ejes principales: niveles jerárquicos, flujo de valor del ciclo de vida y capas [10].

Los niveles jerárquicos indican las distintas funcionalidades dentro de las compañías y se han ampliado para la conexión a Internet de las cosas y servicios. El flujo de valor del ciclo de vida se fundamenta en la gestión de las fases de desarrollo para los sistemas y productos. Las capas describen la conectividad completa desde el nivel de producción hasta las aplicaciones comerciales. En la Figura 1.1 se exhibe el Modelo Arquitectónico de Referencia de la Industria 4.0, donde se observa un modelo tridimensional el cual proporciona una visión espacial y clara de los elementos y capas involucradas.

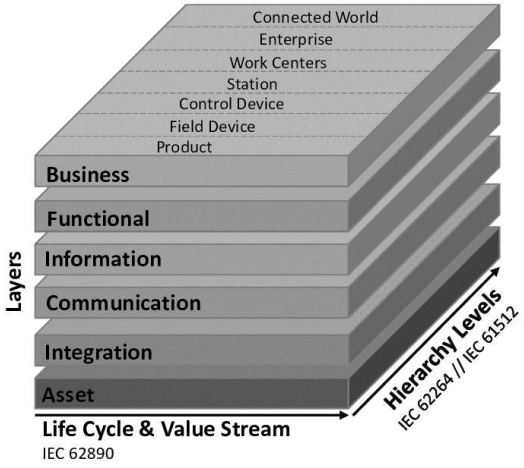


Figura 1.1. Modelo Arquitectónico de Referencia de la Industria 4.0 [10].

Es necesario integrar las distintas áreas presentes en una fábrica con tecnologías innovadoras de la Industria 4.0. Según el documento de [11], la conexión del área de producción a la nube representa uno de los fundamentos esenciales de esta industria. Sin embargo, esta conexión plantea un desafío de compatibilidad. En este sentido, OPC UA puede desempeñar un papel importante al proporcionar una integración vertical que permita transmitir los datos desde los dispositivos hasta las aplicaciones comerciales.

1.4.1.2. Industria 4.0 con OPC UA

Una solución para los desafíos de interoperabilidad y protección en el ámbito de la industria es el uso de OPC UA. No obstante, es importante destacar que OPC UA no se considera superior a protocolos como EtherNet/IP, ProfiNet o DeviceNet [12]. Estos protocolos han demostrado proporcionar una correcta composición de transporte, funcionalidad y simplicidad en el intercambio de información, ya que son tecnologías maduras y altamente eficientes para el control de sistemas.

Por otro lado, OPC UA se enfoca en proveer de información a áreas que lo requieran. Integrar este nuevo estándar demandará grandes esfuerzos para lograr la interoperabilidad de manera exitosa y cumplir con los requisitos de la Industria 4.0.

A continuación, se exhibe en la Tabla 1.1 los requisitos de la Industria 4.0 y como OPC UA pretende aportar y cubrir las exigencias de la actual industria.

Tabla 1.1. Exigencias de la Industria 4.0 cubiertos con OPC UA [12].

Exigencias de la Industria 4.0	Solución con OPC UA
Protocolos de comunicación independientes del sistema operativo y fabricante.	OPC UA estandariza la información en todos los OS.
Escalabilidad en hardware (sensores, controladores, ordenadores) y en aplicaciones en la nube.	OPC UA se ejecuta en 15 KB y diversas arquitecturas, incluyendo sistemas embebidos, gateways, Supervisión Control y Adquisición de Datos (SCADA)/Interfaz Hombre Máquina (HMI) y Sistema de Ejecución de Manufactura (MES)/ Planificación de Recursos Empresariales (ERP).
Garantía de protección en el intercambio de datos, asegurando una protección confiable tanto para los usuarios como para la aplicación.	OPC UA usa varias opciones para la autenticación de aplicaciones en el intercambio de información.
Comunicación estandarizada vía Internet y cortafuegos.	OPC UA ofrece dos protocolos: Protocolo de Control de Transmisión (TCP) y SOAP/ Protocolo de Transferencia de Hipertexto (HTTP) para asegurar la conectividad, ambos con codificación binaria y estructura de mensaje simple para alto rendimiento.
Simplificación en la implementación.	OPC UA usa un modelo de objetos en red con metadatos y descripción de objetos. Las estructuras se crean por instancias y tipos referenciados con herencia.
Integración y extensión semántica en ingeniería para conectar aplicaciones empresariales con fuentes de datos.	Fundación OPC colabora con ML Automatización para mejorar la conexión entre herramientas de ingeniería y empresas.
Comprobación si se cumple con el estándar definido.	OPC UA es un estándar IEC 62541. Hay herramientas para probar y certificar la conformidad, mejorando la calidad y compatibilidad para empresas e ingenieros.

1.4.2. OPC Arquitectura Unificada

La finalidad de OPC Clásica es establecer un estándar de interoperabilidad independiente de la plataforma, para facilitar la lectura y escritura estandarizada en los sistemas de automatización. Sin embargo, esta tecnología depende del sistema operativo Windows ya que utiliza el Modelo de Objetos y Componentes Distribuidos (DCOM) para la comunicación remota, lo que conlleva a dificultar la configuración, tiempos de espera largos y limitaciones en la comunicación a través de internet [13].

OPC Clásica tiene tres aplicaciones específicas: OPC DA (Acceso de Datos), que permite acceder a los datos de forma periódica y garantiza el intercambio fiable de información; OPC A&E (Alarmas y Eventos), que facilita el acceso a alarmas y eventos del proceso y OPC HDA (Acceso a Datos Históricos), que permite acceder a datos históricos del proceso.

Para ampliar el espectro de comunicación, se ha desarrollado un nuevo estándar denominado Arquitectura Unificada OPC. En este nuevo estándar, se lleva a cabo una migración tecnológica en la capa de transporte utilizando protocolos como SOAP, HTTP o TCP, lo que proporciona una comunicación de alto rendimiento [12]. Asimismo, OPC UA integra todas las funciones de OPC Clásica como OPC DA, OPC A&E y OPC HDA.

Las capas de alto rendimiento de OPC UA permiten la creación de aplicaciones en dispositivos con recursos limitados [12], lo cual incrementa considerablemente la utilidad en aplicaciones basadas en OPC UA en comparación con las aplicaciones OPC Clásica. Esto se traduce en una mayor eficiencia en la comunicación durante la trasmisión de datos, lo que a su vez se ve reflejado en una mayor utilidad y eficacia para sistemas de monitorización y sistemas empresariales como el MES o REP.

OPC UA dispone de un modelo de información que se fundamenta en el concepto de programación orientada a objetos. Esto implica definir un modelo de información que puede ser utilizada por otras organizaciones o fabricantes, logrando la estandarización de la información. OPC UA se fundamenta en diversas capas como se exhibe en la Tabla 1.2.

Tabla 1.2. Capas del estándar OPC UA.

Modelos de información implementado OPC UA	
Modelo de la información básico de OPC UA	
Servicios de OPC UA	
Transporte: SOAP/HTTP TCP	Concepto: Espacios de direcciones Modelado de la información

1.4.2.1. Características de OPC UA

A continuación, se mencionan las características y ventajas que ofrece OPC UA [14].

- Múltiples lenguajes de programación: OPC UA es compatible con diversos idiomas de codificación como Python, Java, C entre otros. Esto permite a los desarrolladores seleccionar el idioma de codificación que más se adecue a su aplicación, facilitando la integración con otros sistemas.
- Seguridad: OPC UA garantiza un alto nivel de seguridad en la comunicación. Incluye características como la autentificación, autorización, encriptación de datos y firmas digitales, esto asegura la privacidad y protección de los datos compartidos.
- Escalabilidad: OPC UA permite la conexión con un gran número de dispositivos a un único servidor, simplificando la administración y el sostenimiento de los sistemas, lo que lo conlleva a ser altamente escalable.
- Interoperabilidad: OPC UA es compatible con diversas plataformas y OS, pudiendo integrarse a diferentes sistemas y tecnologías, lo que promueve la interoperabilidad.

1.4.2.2. Modelo Cliente/Servidor OPC UA

El modelo Cliente/Servidor puede ser utilizado por OPC UA para establecer una comunicación. En cada sistema de automatización, puede existir varios clientes y servidores, donde cada cliente tiene la posibilidad de vincularse con uno o varios servidores lo cual también aplicaría para el lado del servidor [12], en la Figura 1.2 se ilustra la interacción entre un cliente y un servidor a través del estándar OPC UA, destacando las capas y componentes clave involucrados en esta comunicación, como la sesión de aplicación, el canal seguro de transporte y las stack OPC UA tanto en el cliente como en el servidor.

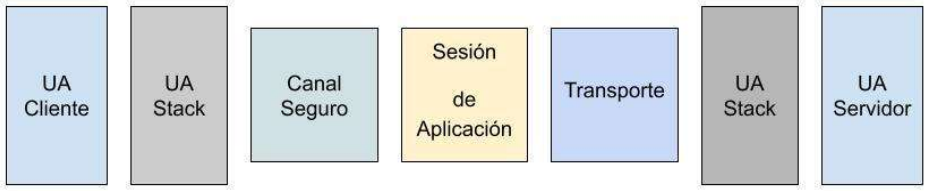


Figura 1.2. Arquitectura para la comunicación OPC UA.

El cliente OPC UA es el punto inicial en el que comienza la sesión de comunicación, obteniendo como resultado un vínculo entre el cliente y el servidor. El cliente podrá utilizar una Interfaz de Programación de Aplicaciones (API), para establecer los atributos de conexión a fin de enviar y recibir los servicios de solicitud y respuesta por parte del servidor [12]. Por otra parte, el servidor OPC UA cumple la función de punto de comunicación final, suministrando los datos al cliente OPC UA. El rol del servidor consistirá en recibir datos provenientes de los clientes y posteriormente enviarles información en respuesta [12].

La idea detrás de OPC UA es que el fabricante del hardware proporcione un servidor OPC UA para sus sistemas, lo que permitirá un acceso normalizado para la comunicación. Este servidor OPC UA puede suministrarse como un software independiente o como un servicio integrado del controlador de la máquina [12].

Una aplicación común es la recopilación de datos en un Sistema de Control Distribuido (DCS) donde la Unidad Terminal Maestra (MTU) debería actuar como un cliente OPC UA para recopilar datos de las Unidades Terminales Remotas (RTUs), adicionalmente, la MTU tendrá la capacidad de funcionar como un servidor OPC UA, permitiendo la exposición de los datos a otros clientes OPC UA sin perder ninguna de sus funcionalidades, tal y como se exhibe en la Figura 1.3.

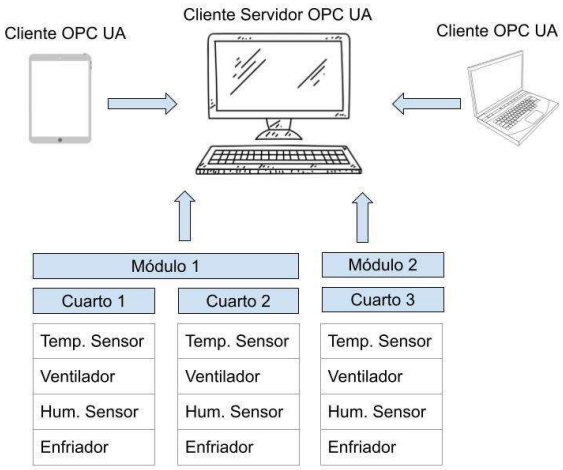


Figura 1.3. Escenario de aplicación.

1.4.2.3. Direcciones de OPC UA

OPC UA no solo aborda el tópico de comunicación, sino que también provee el modelado de información. El servidor OPC UA estará compuesto por un conjunto de espacios de direcciones a los que el cliente podrá acceder [12]. Los espacios de direcciones OPC UA estarán conformados por nodos y atributos, cada nodo tendrá un conjunto de atributos, se podrá interconectar entre nodos mediante el uso de referencias, tal y como se exhibe en la Figura 1.4.

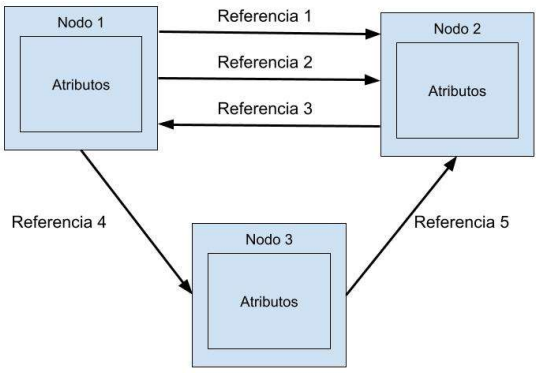


Figura 1.4. Modelo del estándar OPC UA.

Los nodos podrán disponer de un conjunto fijo de atributos, los atributos se precisan como datos para describir al nodo. En el servidor OPC UA, cada nodo será identificado de manera única, por lo que los clientes OPC UA necesitarán identificar el nodo en sus solicitudes de servicio para poder acceder a los valores de los atributos.

1.4.2.4. Seguridad de OPC UA

Una parte esencial de la comunicación es la seguridad, la cual tendrá el deber de firmar, codificar y decodificar la información intercambiada entre el servidor y cliente. La Fundación OPC propone perfiles de seguridad que incluyen parámetros y mecanismos de seguridad empleados para la sesión de comunicación. Según [12] actualmente OPC UA dispone de cuatro configuraciones de seguridad las cuales son:

- Ninguno: Sin ninguna configuración de seguridad.
- Basic128Rsa15: Configuración de seguridad media.
- Basic256: Configuración de seguridad media alta.
- Basic256Sha256: Configuración alta de seguridad.

La Fundación OPC ofrece opciones para intercambiar tokens de identificación para iniciar sesión. Estas opciones son las siguientes [12]:

- Anónimo: No requiere información del usuario.
- Usuario: Para acceder, es necesario proporcionar un nombre de usuario junto con una contraseña. Debiendo suministrar esta información para iniciar sesión y obtener acceso a los recursos.
- X509v3: Un certificado X509v3 se refiere a un tipo de certificado digital que almacena datos de identidad del usuario y lo genera una entidad certificadora de confianza. El usuario debe presentar este certificado para iniciar sesión y acceder a los servicios.
- Certificado: En esta opción, el usuario se identifica mediante un WS-SecurityToken, que puede ser un token Lenguaje de Marcado de Aserción de Seguridad (SAML) o un Kerberos-Ticket. Estos tokens de seguridad contienen información de identidad y son utilizados para autenticar al usuario y permitir el acceso a los servicios.

1.4.2.5. Transporte de datos de OPC UA

La Fundación OPC acepta múltiples protocolos de comunicación que se encargan de facilitar el intercambio de datos entre el cliente y el servidor dentro del marco de OPC UA. Algunos de los protocolos que son soportados por OPC UA son los siguientes:

- Protocolo de Control de Transmisión (TCP): El protocolo de comunicación se sustenta en TCP para el transporte de información, lo que otorga un canal full-dúplex, también proporciona una comunicación confiable y orientada a la conexión entre los clientes y servidores OPC UA. Es ampliamente utilizado ya que ofrece garantías de entrega y control de flujo [12].
- Protocolo Simple de Acceso a Objetos/Protocolo de Transferencia de Hipertexto (SOAP/HTTP): Este protocolo de comunicación opera con mensajes estructurados en el SOAP, que se enviarán a través de HTTP. Estos mensajes serán una unión de datos que se requieren a nivel de información entre diversos equipos, pudiendo trabajar de modo seguro por lo que será práctico de implementar y compatible con contrafuegos [12].

1.4.3. Bibliotecas para OPC UA

Considerando el proyecto y la propuesta de emplear código abierto para la implementación de OPC UA en el modelo Cliente/Servidor, se presentan diversas bibliotecas desarrolladas en varios lenguajes de programación. La Tabla 1.3 expone el estado actual y las características clave de cada una de estas bibliotecas, ofreciendo así una visión general de sus atributos principales.

Tabla 1.3. Bibliotecas de OPC UA de código abierto [15].

Nombre	Lenguaje	Estado	Documentación	Modelo Cliente/Servidor	Política de seguridad
Open6251	C	Activado	Amplia	Si	Completa
OpenScada	C++	Activado	Amplia	Si	Completa
ASNeG	C++	Activado	Parcial	Si	Completa
asyncua	Python	Activado	Amplia	Si	Completa

Uno de los lenguajes de programación más populares es Python, gracias a su simplicidad, versatilidad y la gran comunidad de desarrolladores que la respalda. Además, cuenta con extensas bibliotecas que lo hacen adecuado para una amplia gama de aplicaciones.

La librería asyncua es una implementación de código abierto del protocolo OPC UA para Python, según su documentación [16], se compone de un conjunto de bibliotecas como asyncio, que es un módulo de la biblioteca estándar de Python para la programación asíncrona. La biblioteca asyncua es una opción sólida para la implementación ya que puede adaptarse a una amplia gama de aplicaciones desde la monitorización de procesos hasta la gestión de activos.

1.4.3.1. Herramientas para asyncua en el modelo Cliente/Servidor

Basados en la premisa del proyecto, se puede proporcionar una descripción general para un servidor OPC UA usando el módulo asyncua:

- Inicialización del Servidor OPC UA: Se crea una instancia del servidor OPC UA utilizando la clase `Server` de la biblioteca asyncua. Luego, se establece la URL del servidor y su nombre.

- Configuración del espacio de nombres: Se registra un espacio de nombres personalizado para el servidor OPC UA utilizando el método “register_namespace”. Esto permite organizar los nodos y objetos en el servidor bajo un URI específico.
- Creación de nodos y variables: Se crean varios nodos y variables que representan diferentes parámetros y valores en el servidor OPC UA. Por ejemplo, se crean nodos para la temperatura, presión, humedad y altitud, para ser asociados con variables para almacenar sus valores.
- Inicio del servidor: Se inicia el servidor OPC UA con el método “start()”, lo que lo pone en funcionamiento y permite que los clientes se conecten y accedan a los datos.
- Actualización de valores en el servidor: Los valores capturados se actualizan en las variables asociadas en el servidor OPC UA utilizando el método “set_value()”.

De igual modo se puede establecer una descripción general para un cliente OPC UA usando el módulo asyncua:

- Dirección del servidor: Se define la dirección del servidor OPC UA al que el cliente se conectará.
- Creación del cliente: Se instancia un cliente OPC UA utilizando la dirección del servidor especificado.
- Conexión al servidor: El cliente se conecta al servidor OPC UA utilizando el método “connect()”. Una vez que se establece la conexión, el cliente está listo para interactuar con el servidor.
- Obtención de valores de nodos del servidor: Se obtienen los valores de los nodos en el servidor OPC UA, utilizando el método “get_node()”, mediante el espacio de nombre e identificadores de nodo.
- Obtención de los valores de los nodos: Se recuperan los valores asociados a los nodos utilizando el método “get_value()”.

1.4.4. Sistemas de monitorización

Actualmente, se ha optado por implementar sistemas flexibles y adaptables que respondan a las necesidades del mercado de forma incesante. Según [17] la dificultad de los procesos industriales, las interdependencias en las diversas áreas de producción y la falta de flexibilidad en los procesos productivos complican la gestión.

El seguimiento del proceso contribuye a la retroalimentación, permitiendo establecer criterios para realizar el ajuste en la cadena de producción. De acuerdo a [17], la monitorización se define como la automatización de estos procesos de vigilancia proporcionando al operador la capacidad de detectar situaciones irregulares acompañadas de un monitoreo constante de las distintas variables del proceso. Esto se logra mediante una interfaz de fácil y rápido entendimiento.

Los sistemas de control actuales deben ser flexibles y adaptables, de acuerdo a su producto, por lo que el disponer de los datos resultará un factor crucial para tomar decisiones adecuadas y para interactuar efectivamente con el proceso. Igualmente, se requieren mecanismos para obtener un registro de la evolución de las variables, por lo que, es necesario almacenar los datos para respaldar la toma de decisiones. Los sistemas de monitorización actuales deben satisfacer todas estas necesidades a través de las mediciones periódicas y de mecanismos de alertas.

1.4.4.1. Adquisición y registro

La exploración de sistemas que puedan trabajar juntos de manera eficiente y adaptable es fundamental para la expansión de la comunicación el uso de tarjetas funcionales, posibilitan el acceso a la representación eléctrica de diversas dimensiones del proceso. Debido a esto, se ha promovido la conexión entre el procedimiento y los sistemas de supervisión.

Los Controladores Lógicos Programables (PLC), DCS y los Computadores Personales Industriales (IPC) han ampliado el espectro, ya que su aplicación se ha expandido como interfaces de acceso al proceso. Igualmente, los protocolos de comunicación industrial se desarrollaron de forma paralela, lo cual ayuda a la interconexión entre dispositivos, actuadores y transductores.

Según [17] el objetivo de los sistemas de monitorización es la centralización de la información en un ordenador. Por lo que se deberán seleccionar dispositivos y plataformas de operación que sean compatibles entre sí. En consecuencia, OPC UA reducirá esta brecha al habilitar la interoperabilidad e independencia de la plataforma para trasladar la información entre las secciones de producción y comercial [12].

1.4.4.1.1. Dispositivos para la adquisición de datos

A continuación, se presentan las soluciones tecnológicas más comunes utilizadas para la adquisición de datos en tareas de monitorización. Estas opciones no son alternativas equivalentes, sino que pueden coexistir o excluirse mutuamente según las especificaciones del proyecto.

Una opción económica consiste en utilizar Tarjetas de Adquisición de Datos (TAD), las cuales establecen una conexión directamente con el bus del ordenador y no requieren alimentación ni otros requisitos adicionales. Estas tarjetas necesitan un software específico para su configuración y control [18]. Como es el caso de la instrumentación de laboratorio especializada para ciertas mediciones, se pueden utilizar buses de instrumentación como GPIB y VXI, que permiten la interconexión de instrumentos con el controlador.

En instalaciones más grandes los PLC actúan como sistemas de adquisición y control, donde estos dispositivos establecen comunicación con el ordenador a través de interfaces serie [18]. Se utilizan buses de campo que permiten la interconexión entre dispositivos, por ejemplo, la comunicación SPI, I2C, UART entre otros. Estos buses facilitan la distribución física de dispositivos y tareas de control en un DCS.

Conforme aumenta el volumen de datos y la necesidad de integración, se requieren redes de comunicación más avanzadas, que garanticen la interconectividad entre las distintas soluciones y dispositivos.

1.4.4.1.2. Registro de datos

Según [18], en todos los sistemas de monitorización, independientemente del método de adquisición empleado, se lleva a cabo una conversión digital de la señal. Esto implica convertir la señal analógica en una serie de muestras numéricas que representan la señal original.

Si se trabaja con señales discretas o binarias, basta con utilizar un único dígito binario para representarlas. No obstante, en situaciones que involucren señales analógicas o continuas, se emplea una combinación de bits que determina su valor dentro de un rango específico. Este rango está vinculado al número de bits mediante la relación 2N, donde N representa la cantidad de bits utilizados. El convertidor y el microprocesador integrados en el sistema de adquisición imponen un límite en la cantidad de bits que pueden ser empleados en la representación.

Por lo general, el proceso de transformación de señales analógicas a digitales es transparente para el usuario del sistema de monitorización, al igual que ocurre con la comunicación entre los dispositivos de recopilación de datos y la aplicación de monitoreo.

1.4.4.2. Representación del proceso

La monitorización requiere una representación visual del proceso como elemento fundamental y se recomienda aplicar una distribución adecuada de los elementos y una

selección correcta de la iconografía y colores para facilitar la integración entre los operadores y el proceso.

A continuación, se presentan las consideraciones sugeridas por [18] para la creación de interfaces de operación:

- Mantener la consistencia en las ventanas, menús, botones e iconografía en todo el proyecto, siempre que sea posible.
- La representación gráfica del proceso debe ser acorde a la disposición espacial de las células de producción.
- Los datos numéricos deben ubicarse sobre la iconografía que los genera.
- Utilizar colores de manera lógica, como rojo, amarillo, verde, azul, purpura, blanco, gris y negro, para facilitar la compresión rápida de la situación.

No hay criterio para la estandarización del HMI porque la mayoría de software de SCADA usan sus propias librerías. Sin embargo, muchos de ellos conservan similitudes en su iconografía, la Sociedad de Instrumentos de América (ISA) es utilizada ampliamente para la documentación de procesos y puede ser una alternativa para la representación gráfica de los sistemas físicos.

La integración de una plataforma de monitoreo en línea es una solución poderosa, ya que proporciona una interfaz accesible que facilita la adopción de decisiones informadas y la detección temprana de problemas. Proporcionando una visión general y contextualizada de los datos del proceso lo que ayuda a los operadores a mantener un control efectivo y eficiente de los procesos industriales.

1.4.4.2.1. Plataformas de monitoreo en línea

Las plataformas de monitorización Web o paneles IoT están diseñadas para conectar dispositivos con usuarios, proporcionando una interfaz sencilla. Esto permite que los usuarios visualicen la información e interactúen a través de un PC o un dispositivo móvil.

Los paneles IoT son utilizados para tareas de control y monitoreo de dispositivos, integran funciones como el cambio de parámetros, registro de datos con el uso de gráficas y tendencias, etc. Esta interfaz podrá centralizar los datos de diferentes fuentes de información, pudiendo agregar nuevos dispositivos al panel IoT, ampliando la flexibilidad de esta tecnología para el usuario.

Los desarrolladores de plataformas IoT serán los encargados de recopilar los datos del hardware desde una estación al panel IoT [19], los cuales serán procesados por los operadores lo que aportará en la toma de decisiones ya que se visualiza las tendencias y el estado actual de los dispositivos.

Esta plataforma dispondrá de widgets para facilitar la compresión de los sensores conectados a la plataforma IoT. Una herramienta fundamental será el historial, que se emplea para analizar los datos actuales de los sensores conectados a los dispositivos asociados a la plataforma. La lectura de los sensores se obtiene periódicamente por la plataforma y se almacena en el historial junto con la fecha y hora de cada registro previo.

Empresas industriales como Duracell, DHL, Wago [20], entre otras utilizan paneles IoT para ser utilizadas en interfaces Web y móviles para sistemas de monitoreo.

1.4.4.3. Alarmas y eventos

Las alarmas se fundamentan en la supervisión de las magnitudes de las variables del sistema [21]. Se trata de situaciones no deseables, puesto que su existencia puede acarrear dificultades en el funcionamiento, requiriendo la vigilancia del operador para dar solución antes de llegar a un escenario crítico que pare el proceso.

En los escenarios normales de operación en las cuales exista un cambio en las constantes de funcionamiento se nombraran como eventos [21], los cuales no requerirán la vigilancia del operador sin embargo deberán ser registradas de forma automática.

1.4.4.4. Gráficas y tendencias

La representación gráfica en el tiempo de una magnitud es una herramienta ampliamente utilizada en la industria. Esta representación debe estar correctamente acompañada de la fecha y hora para evitar confusiones al comparar los registros. Esta herramienta es de suma importancia, ya que permite al operador anticipar situaciones de alarma y realizar comparaciones con otras variables [18].

1.4.4.5. Histórico y base de datos

Los sistemas de monitorización logran leer los datos de los instrumentos y almacenarlos según su configuración, considerando las limitaciones específicas de cada variable de proceso.

La ventaja de disponer de una centralización radica en su capacidad de almacenamiento. El registro continuo de datos permite la posterior recuperación para la creación de análisis,

estadísticas y retroalimentación [18]. Por lo tanto, contar con un historial es crucial dentro de un entorno de monitorización.

Una base de datos permite el acceso tanto desde el entorno de monitorización como desde otras aplicaciones diseñadas para trabajar con grandes volúmenes de datos. Esto requerirá el uso de lenguajes de programación especializados [18].

1.4.4.5.1. Base de datos en la nube

La nube computacional, como técnica que aprovecha la virtualización de recursos informáticos físicos, ha demostrado ser una herramienta altamente beneficiosa para las empresas al minimizar costos y convertir las TI en una ventaja competitiva. Gracias a la madurez de Internet y la disponibilidad de conexiones de alta velocidad, las aplicaciones complejas que operan de forma remota son cada vez más comunes, lo que abre nuevas posibilidades para las TI en tres áreas clave: reducción de costos, planificación de capacidad y agilidad comercial.

Una de las formas de implementar la computación en la nube es mediante la utilización de bases de datos alojadas en plataformas Web, ya sea en entornos privados o públicos. Estas bases de datos, combinados con los servicios de Software como Servicio (SaaS), Plataforma como Servicio (PaaS) e Infraestructura como Servicio (IaaS), permiten a los usuarios compartir recursos generales y costos con otros usuarios. En el caso de IaaS, se provee y gestiona el hardware externamente, mientras que en PaaS se incluye la gestión del OS y en SaaS se añade la gestión de las aplicaciones.

Al hacer uso de una base de datos en línea, es posible sacar provecho de los beneficios que proporciona la nube computacional en conjunto con las TI. Entre estos beneficios se encuentran la implementación rápida, ya que los recursos informáticos están disponibles de inmediato; el acceso desde cualquier dispositivo conectado a Internet; permitiendo ajustar las capacidades según las exigencias fluctuantes de la compañía sin complicaciones al no requerir personal intermedio.

Sin embargo, hay que considerar que la dependencia de la conectividad de red es una posible desventaja de la nube computacional. En caso de una interrupción en la red, el servicio puede sufrir fallas. Aunque es así, las ganancias de la computación en la nube hacen de esta solución una opción atractiva para maximizar la eficiencia y la competitividad de las compañías en el contexto actual.

1.4.5. Plataformas de hardware libre

Las tarjetas de desarrollo o hardware libre son dispositivos de un bajo costo, pero con altas prestaciones, diseñadas para que los usuarios puedan crear y experimentar con diferentes proyectos, Las tres tarjetas de desarrollo más populares y comerciales son la Raspberry Pi, Arduino y ESP.

Las tarjetas de desarrollo son dispositivos ideales para el impulso de proyectos debido a su bajo costo y alta eficiencia, aunque su precisión puede ser limitada. A continuación, se revisará cada una de estas tarjetas de desarrollo:

- Raspberry Pi: Es un dispositivo informático compacto de placa única propuesto por la Fundación Raspberry Pi. Se destaca por su facilidad de uso y capacidad de conexión, lo que la convierte en un recurso de gran utilidad para proyectos tecnológicos. Puede conectarse a Internet a través de Ethernet o WiFi, esto lo hace valioso para proyectos de IoT. Asimismo, cuenta con una amplia gama de complementos, lo que amplía las posibilidades de proyectos que se pueden desarrollar.

- Arduino: Es un entorno de desarrollo destinado a la creación de proyectos electrónicos que contiene un microcontrolador y un Entorno de Desarrollo Integrado (IDE) lo que facilita a los desarrolladores el programar la placa de forma sencilla para controlar los dispositivos conectados. Las placas Arduino admiten lenguajes como C y C++, contando con una amplia diversidad de bibliotecas y herramientas disponibles para permitir la creación de proyectos.

- ESP: Es una familia de microcontroladores desarrollados por Espressif Systems que se utilizan comúnmente en proyectos de IoT debido a su capacidad para conectarse a WiFi y Bluetooth. Las placas de desarrollo más populares de ESP son el ESP8266 y el ESP32, que cuentan con procesadores de bajo consumo de energía y una amplia gama de periféricos integrados. Al igual que las otras tarjetas de desarrollo, el ESP se puede programar en varios lenguajes de programación, incluyendo C y C++. Espressif Systems ofrece un IDE fundamentado en Eclipse para facilitar la creación de proyectos.

Tanto las tarjetas de desarrollo Raspberry Pi cómo ESP ofrecen diversas ventajas para la ejecución de un servidor OPC UA. Por ejemplo, tanto Raspberry Pi cómo ESP son opciones versátiles que cuentan con capacidades de conectividad, cómo Ethernet y WiFi, lo que permite la comunicación con otros dispositivos en una red. Además, su potencia de

procesamiento y capacidad de almacenamiento las convierten en opciones viables para ejecutar un servidor OPC UA.

Otro aspecto a considerar es la flexibilidad de estas tarjetas y su capacidad de programación, respaldada por bibliotecas y herramientas disponibles que facilitan el desarrollo del proyecto y la creación de un servidor OPC UA.

2. METODOLOGÍA

El proyecto tiene como objetivo implementar un sistema para monitorear, registrar y visualizar variables como temperatura, presión, humedad y altitud. Estas mediciones se envían a una tarjeta de desarrollo (servidor OPC UA) utilizando el estándar OPC UA, que se encargará de transmitir las lecturas al nivel de supervisión. El nivel de supervisión está equipado con un computador (cliente OPC UA) conectado a la red, cuya función es leer los datos del servidor y posteriormente registrar los datos en la nube. Esta información puede ser accesible y visualizada a través de una interfaz, permitiendo el acceso desde cualquier dispositivo que esté conectado a internet.

El servidor y el cliente OPC UA son programados en un lenguaje de desarrollo como Python, Java o C++. La elección del lenguaje dependerá de los objetivos específicos de la aplicación del proyecto.

La metodología empieza con la ingeniería de detalle, donde se elige el hardware y el software requerido para el desarrollo del proyecto; se define el protocolo de conexión entre la tarjeta de desarrollo y el transductor; se revisa el desarrollo del software, detallando cómo se configuró la conexión entre el transductor y la tarjeta de desarrollo, así como la programación del servidor OPC UA; también se aborda el desarrollo del cliente OPC UA, incluyendo sus plataformas complementarias y la Interfaz Gráfica de Usuario (GUI); se efectúa la configuración de las plataformas con el propósito de establecer la vinculación con el cliente OPC UA; finalmente, se lleva a cabo una revisión de los dispositivos utilizados en las pruebas y su disposición en la maqueta.

2.1. Ingeniería de detalle

2.1.1. Selección del lenguaje de programación

Para implementar una comunicación OPC UA junto con otros servicios, existen varias soluciones disponibles y la elección de la solución depende de los requisitos y necesidades específicas del proyecto. OPC UA ofrece diversas API desarrolladas en varios lenguajes de programación, como Python, Java, C++ y .NET. Además, están disponibles SDK comerciales y algunas SDK de código abierto, lo que posibilita a elegir el lenguaje de desarrollo de preferencia.

Uno de los lenguajes de desarrollo disponibles es Python, el cual se destaca como una excelente opción debido a su versatilidad, permitiendo su ejecución en diferentes OS con soporte en diferentes enfoques de desarrollo, como el imperativo, orientado a objetos y

funcional. Adicionalmente, Python cuenta con una amplia variedad de extensiones que pueden ser utilizadas en diferentes tareas de comunicación, como la administración de bases de datos y la visualización de datos en plataformas Web [22]. Los SDK y bibliotecas se presentan como un conjunto de código predefinido y reutilizable que contiene funciones y rutinas que se pueden utilizar para el desarrollo de proyectos.

En el estudio realizado por [23], se utilizó Python en una aplicación de comunicación industrial para recopilar información en tiempo real. Los resultados demostraron que Python era capaz de manejar grandes cantidades de datos y procesamiento de manera eficiente para aplicaciones IIoT. También, las aplicaciones desarrolladas en Python son fáciles de mantener y ampliar, lo que permite una mayor flexibilidad para futuras actualizaciones.

Un aspecto importante es la selección de una librería que satisfaga las metas establecidas para el proyecto. Por lo tanto, se propone utilizar FreeOPCUa actual asyncua, la cual se refiere a una solución de código abierto que ha sido desarrollada en Python que sigue el estándar PEP8, esta librería adopta el modelo de programación orientada a objetos lo que facilita su implementación. Se considera que esta librería es la mejor opción, ya que en el documento expuesto por [15] demuestra que es una solución efectiva para el concepto de monitorización para la Industria 4.0. Esta biblioteca es de gran ayuda, ya que simplifica la creación del servidor y cliente, también cuenta con las políticas de seguridad altas como Basic128Rsa15. Una de las fortalezas al elegir Python como lenguaje de programación es su compatibilidad con la filosofía de OPC UA, ya que ambos son multiplataformas. Asimismo, Python cuenta con una amplia gama de SDK y librerías que permiten expandir las funcionalidades de la aplicación. En este caso, podremos utilizar estas bibliotecas para enviar la información obtenida a través de la comunicación OPC UA a una base de datos en línea y a una plataforma de monitorización Web. Esto hace de Python la mejor opción para nuestro proyecto.

2.1.2. Selección de la tarjeta de desarrollo

Se propone utilizar la Raspberry Pi 4B como microcontrolador para el proyecto debido a varias ventajas destacadas. En primer lugar, su rápida puesta en marcha y amplia disponibilidad comercial hacen que sea una opción conveniente y accesible para el desarrollo del proyecto tecnológico. La Raspberry Pi 4B es una tarjeta de desarrollo altamente reconocida y utilizada en diversos proyectos. En términos de rendimiento, la Raspberry Pi 4B está equipada con un procesador Quad Core Cortex-A72, lo que permite ejecutar tareas y aplicaciones exigentes de manera eficiente.

Una de las ventajas clave de la Raspberry Pi 4B es su conectividad WiFi incorporada, esto permite una fácil conexión a redes inalámbricas, lo que resulta conveniente para el proyecto, ya que se puede acceder al dispositivo de manera remota. En cuanto a los protocolos soportados, la Raspberry Pi 4B es compatible con SPI, I2C y UART. Estos protocolos son ampliamente utilizados en la industria, facilitando la integración y conexión con el transductor necesario para el proyecto.

Otra ventaja significativa de elegir la Raspberry Pi 4B es que su OS es Raspbian Pi OS, que es una distribución ligera de Linux; especialmente diseñada para esta placa. Este sistema operativo es conocido por su estabilidad y facilidad de uso, lo que favorece el desarrollo y ejecución de proyectos. Al utilizar la Raspberry Pi 4B como servidor, se puede comprobar la interoperabilidad entre diferentes sistemas operativos. En este caso, se utiliza Windows 10 como OS del cliente, lo que permite verificar la compatibilidad y el funcionamiento fluido del sistema en un entorno Cliente/Servidor.

En el estudio presentado por [24] se ha utilizado la tarjeta Raspberry Pi 4B la cual se presentó como una plataforma viable en general, concluyendo que el modelo Raspberry Pi 4B es una plataforma factible para iniciar la incorporación de OPC UA en aplicaciones IIoT.

En el estudio [14], los autores probaron la ejecución de OPC UA en la Raspberry Pi 4B para la comunicación entre dispositivos de la Industria 4.0 y encontraron que la plataforma funciona correctamente y cumple con los requisitos de rendimiento necesarios. Los autores concluyeron que Raspberry Pi 4B es una plataforma rentable y efectiva por su costo beneficio para la implementación de OPC UA.

En ambos estudios, no se presentaron cuellos de botella significativos o problemas importantes con el uso del modelo Raspberry Pi 4B al realizar la incorporación de OPC UA en aplicaciones IIoT y automatización industrial. Sin embargo, se destacó la importancia de optimizar el rendimiento y la eficiencia de la plataforma según los requerimientos particulares de la aplicación en cuestión.

2.1.3. Selección del transductor

La optimización de recursos de la tarjeta Raspberry Pi 4B es importante. En este sentido, se propone utilizar el transductor multivariable BME280, que se muestra como una opción sobresaliente para medir la temperatura, presión, humedad y altitud (cabe destacar que la medición de la altitud es una variable calculada por el transductor, no medida directamente). Los transductores multivariables en forma general presentan las siguientes ventajas [25].

- Versatilidad: El sensor multivariable permiten medir múltiples variables en un solo dispositivo, lo que brinda versatilidad en la adquisición de datos en diferentes aplicaciones.
- Reducción de costos: Al utilizar un solo sensor multivariable en lugar de múltiples sensores monovariables, se reduce el costo de los componentes necesarios para la medición de diferentes variables.
- Facilidad de integración: Los sensores multivariables están diseñados para ser compatibles con diferentes plataformas. Su fácil integración reduce la complejidad de la configuración del sistema, lo que resulta en una puesta en marcha más rápida y eficiente.
- Mayor precisión: Los sensores multivariables están diseñados para proporcionar mediciones precisas de múltiples variables. Al medir varias variables simultáneamente, el sensor puede compensar errores en una variable con mediciones precisas de otras variables, lo que aumenta la precisión general de las mediciones.

A continuación, se presentan las características del transductor BME280, siendo un transductor de temperatura, presión y humedad fabricado por Bosch Sensortec. Las especificaciones del transductor BME280 de Adafruit se exhiben en la Tabla 2.1.

Tabla 2.1. Especificaciones del transductor BME280 de Adafruit.

Intervalo de temperatura operativa	- 40 a 85 °C
Intervalo de humedad operativa	0 a 100 %
Intervalo de presión operativa	300 a 1.100 hPa
Resolución de temperatura	0,01 °C
Resolución de humedad	0,008 %
Resolución de presión	0,18 hPa
Exactitud de temperatura	±1 °C
Exactitud de humedad	±3 %
Exactitud de presión	±1 hPa
Interfaz de comunicación	I2C y SPI
Voltaje de funcionamiento	3,3 V - 5 V.

El modelo BME280 de Adafruit cuenta con conectores STEMMA QT, que están conmutados con la interfaz de comunicación I2C. Esto simplifica la conexión con la Raspberry Pi 4B, lo cual es especialmente importante en sistemas donde el espacio físico es limitado.

2.2. Desarrollo del proyecto

El sistema está formado por una combinación de hardware y software que permiten su operatividad, los fundamentos conceptuales de la estructura del proyecto se presentan visualmente en la Figura 2.1. Se mide y registra con herramientas de la computación en la nube las variables de temperatura, presión, humedad y altitud. Se utiliza la solución de comunicación I2C para establecer la vinculación entre la tarjeta Raspberry Pi 4B y el transductor. Adicionalmente, en la tarjeta se implementa un servidor OPC UA, este servidor permite establecer una conexión con un cliente OPC UA específicamente diseñado para realizar funciones de lectura y centralización de los datos del servidor, estos datos son registrados y visualizados en las plataformas Web, permitiendo el acceso a los datos de temperatura, presión, humedad y altitud desde cualquier dispositivo.

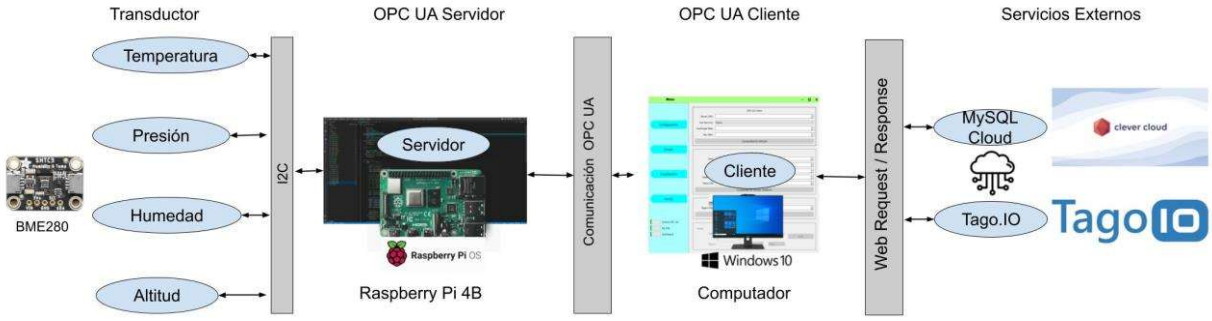


Figura 2.1. Arquitectura del sistema propuesto.

2.2.1. Conexión del transductor con la tarjeta de desarrollo

Con el propósito de establecer la vinculación entre la Raspberry Pi 4B y el transductor BME280, se utiliza la interfaz de comunicación I2C. Es factible emplear este protocolo de comunicación I2C para la interacción con el transductor BME280 y la Raspberry Pi 4B, sin descuidar el propósito principal de implementar una comunicación OPC UA. Como se mencionó anteriormente, el sensor BME280 es compatible con el protocolo I2C, lo que lo convierte en una opción simple y práctica para establecer la comunicación con la Raspberry Pi 4B.

Es importante destacar que OPC UA es un estándar de comunicación que no depende del hardware, por lo que puede ser utilizado en diferentes tipos de redes y dispositivos. En la

documentación del transductor [26] se muestran las conexiones que deben ser realizadas para establecer la comunicación a través del conector STEMMA QT y la interfaz de comunicación I2C con la Raspberry Pi 4B mostrada en la Figura 2.2.

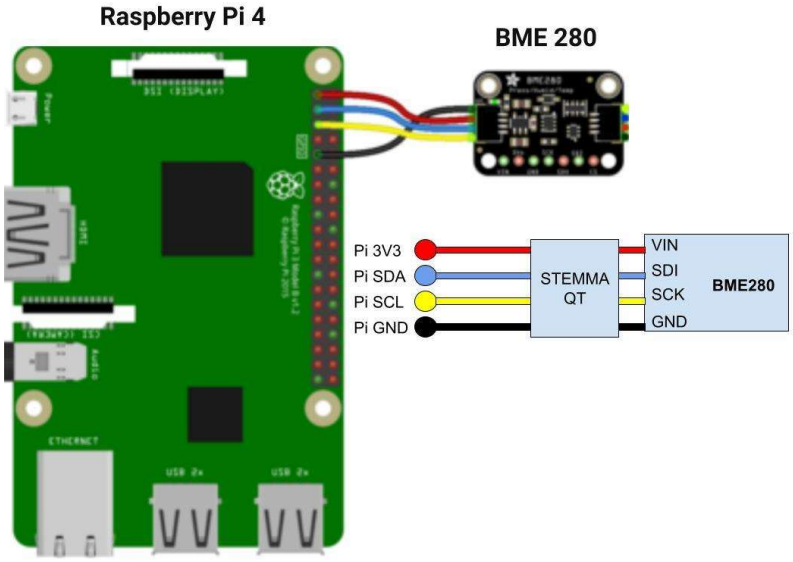


Figura 2.2. Diagrama de conexión para el transductor BME280 y la Raspberry Pi 4B.

Las Entrada/Salida de Uso General (GPIO) utilizadas en la Raspberry Pi son las siguientes:

- Pi 3V3 - Rojo: Esta conexión suministra la alimentación de 3.3 V desde la Raspberry Pi al sensor BME280. Asegura que el sensor reciba la energía necesaria para su funcionamiento adecuado.
- Pi GND - Negro: Esta conexión proporciona la conexión a tierra común entre la Raspberry Pi y el sensor BME280. Es importante tener una referencia de tierra compartida para la correcta transferencia de datos y evitar problemas de señal.
- Pi SCL – Amarillo: SCL (Serial Reloj) en la Raspberry Pi se conecta a SCK (Serial Reloj) en el sensor BME280. Esta conexión permite que ambos dispositivos sincronicen la transferencia de datos a través del bus I2C.
- Pi SDA - Azul: SDA (Serial Datos) en la Raspberry Pi se conecta a SDI (Serial Datos de Entrada) en el sensor BME280. Esta conexión posibilita la transferencia bidireccional de datos entre la Raspberry Pi y el transductor empleando el protocolo I2C.

Al seguir esta conexión específica, se establece una comunicación adecuada entre la Raspberry Pi y el sensor BME280.

2.2.2. Configuración del Servidor OPC UA

La propuesta del proyecto es implementar un servidor OPC UA en una Raspberry Pi 4B. El servidor está conectado al transductor BME280 para la medición de temperatura, presión humedad y altitud. El código sugerido consta de dos etapas o fases:

La etapa inicial, consiste en la configuración del vínculo entre la Raspberry Pi 4B y el transductor BME280. Esta comunicación se implementó a través del protocolo I2C. En el código se utilizó el objeto "board" para establecer la comunicación I2C entre la placa de desarrollo y el transductor BME280. Se creó un objeto "Adafruit_bme280.Adabruit_BME_280(i2c)" para interactuar con el transductor y configurar los parámetros. La configuración del transductor se efectuó mediante las propiedades del objeto "bme280", que incluyen:

- sea_level_pressure: Se establece la presión barométrica al nivel del mar para calcular la altitud.
- mode: Establece el modo de operación del transductor. El transductor tiene cuatro modos de operación, en este caso se establece MODE_NORMAL el cual es un modo de operación por defecto proporcionando lecturas continuas del sensor.
- sandby_period: Se establece el tiempo de espera del modo de operación seleccionado, para este caso se estableció en STANDBY_TC_500 esperando 500 ms antes de realizar una nueva lectura.
- iir_filter: Establece el factor de suavización de la lectura del transductor el cual dispone de cuatro modos de suavizado, se establece el modo IRR_FILTER_X16 el cual proporciona el máximo factor de suavizado para una mayor precisión en la lectura del transductor.
- overscan: Permite establecer la resolución tanto para las señales de temperatura, presión y humedad, se dispondrá de cuatro niveles de resolución para cada una de las señales.

Una vez configurado el transductor BME280, se dispone de los valores de temperatura, presión, humedad y altitud del sensor, utilizando los métodos “bme280.temperature”, “bme280.pressure”, “bme280.humidity” y “bme280.altitude”, respectivamente.

La segunda fase de este código continúa con la escritura de los valores del transductor en el servidor OPC UA. Para la creación del servidor OPC UA se utiliza la librería asyncua. La configuración del servidor inicia con la definición del objeto “Server()”, el cual representa el

servidor OPC UA, consecutivamente se asigna la dirección URL a través de una conexión TCP/IP, a esta se establece el puerto lógico 4840 siendo utilizado y estandarizado para servidores OPC UA, los métodos utilizados para establecer el servidor son el “set_endpoint()” y “set_server_name()” respectivamente. El fragmento de este código configurado seria:

- server = Server(): Se creo una instancia de un servidor OPC UA utilizando la clase Server() de la biblioteca asyncua. Esta instancia será el objeto a través del cual se gestionará la comunicación y la administración de los datos en el servidor.
- url = "opc.tcp://192.168.0.118:4840": Se establece la URL de conexión del servidor OPC UA. Esta URL indica cómo los clientes se conectarán al servidor, en este caso es "opc.tcp://192.168.0.118:4840".
- server.set_endpoint(url): Se define dónde los clientes deben conectarse para interactuar con el servidor OPC UA. En este caso, se utiliza la URL definida.
- server.set_server_name("OpcUa Server"): Se establece el nombre del servidor, esto es útil para identificar el servidor y proporcionar información descriptiva sobre él. En este caso, el nombre del servidor se establece como "OpcUa Server".

Este fragmento de código es esencial para la creación y configuración del servidor OPC UA que será utilizado en la aplicación, la URL dependerá de la IP del dispositivo.

En la siguiente sección del código se define el "namespace" utilizando el método "register_namespace()". Los "namespace" en asyncua se utilizan para separar y evitar la reutilización de nombres de variables, métodos y objetos. Luego, se procede a la creación de los nodos y variables que representan los parámetros de interés. Estos nodos se agregan al objeto raíz del espacio de direcciones del servidor OPC UA mediante los métodos "add_object()" y "add_variable()". La sección de este código seria:

- myobj = server.nodes.objects.add_object(idx, "Parameters"): Se creo un nuevo objeto dentro del espacio de direcciones del servidor. Este objeto se denomina "Parameters" y se añade al espacio de objetos (objects). El argumento idx indica el índice del espacio de nombres que se utilizó para registrar.
- Param = myobj.add_variable(idx, "MyVariable", 0): Se creo una nueva variable dentro del objeto "Parameters". Esta variable se llama "MyVariable". El tercer argumento 0 se refiere al valor inicial de la variable. Param es la referencia a esta variable.

- Temp = myobj.add_variable(idx, "Temperature", 0, varianttype=ua.VariantType.Double): Similar a la línea anterior, se crea una variable llamada "Temperature" dentro del objeto "Parameters". Esta variable es de tipo Double y se inicializa con el valor 0. Temp es la referencia a esta variable.
- Press = myobj.add_variable(idx, "Pressure", 0, varianttype=ua.VariantType.Double): Se crea una variable llamada "Pressure" con las mismas características que la anterior.
- Humi = myobj.add_variable(idx, "Humidity", 0, varianttype=ua.VariantType.Double): Se crea una variable llamada "Humidity" con las mismas características que las anteriores.
- Alti = myobj.add_variable(idx, "Altitude", 0, varianttype=ua.VariantType.Double): Se crea una variable llamada "Altitude" con las mismas características que las anteriores.

Finalmente, se inicializa el servidor utilizando el método "start()". En el bucle principal del código, se actualizan los valores de las constantes de temperatura, presión, humedad y altitud cada vez que se completa un ciclo de muestreo. Esto se logra utilizando el método "set_value()", que permite actualizar el valor de las constantes definidas en el servidor OPC UA. El código principal del bucle seria:

- Temperature = bme280.temperature, Humidity = bme280.humidity, Pressure = bme280.pressure, Altitude = bme280.altitude: Obtiene los valores de temperatura, humedad, presión y altitud del transductor BME280 y se almacenan en sus variables correspondientes.
- Temp.set_value(Temperature), Humi.set_value(Humidity), Press.set_value(Pressure), Alti.set_value(Altitude): Se actualizan las variables del servidor OPC UA con los valores del transductor, esto permite que los clientes OPC UA accedan a estos datos.

Es importante destacar que el estándar OPC UA cuenta con variantes para una comunicación segura y encriptada. Uno de los estándares de seguridad más altos es el Basic256Sha256, disponible en la librería usando el método "Basic256Sha256_SignAndEncrypt". El enfoque de seguridad adoptado se basa en la implementación de un cifrado AES de 256 bits y un algoritmo SHA-256, lo que asegura la protección y veracidad de la información transmitida. Para llevar a cabo esta estrategia de

seguridad, resulta fundamental establecer una clave y un certificado que permitan la comunicación segura entre el servidor y los clientes OPC UA. La clave y el certificado se pueden cargar utilizando los métodos "server.load_certificate()" y "server.load_private_key()", respectivamente.

A continuación, en la Figura 2.3 se muestra el esquema gráfico en el cual se detalla cómo se ha implementado el servidor OPC UA en la Raspberry PI 4B. En el Anexo A se encuentra detallado el código necesario para configurar el servidor OPC UA.

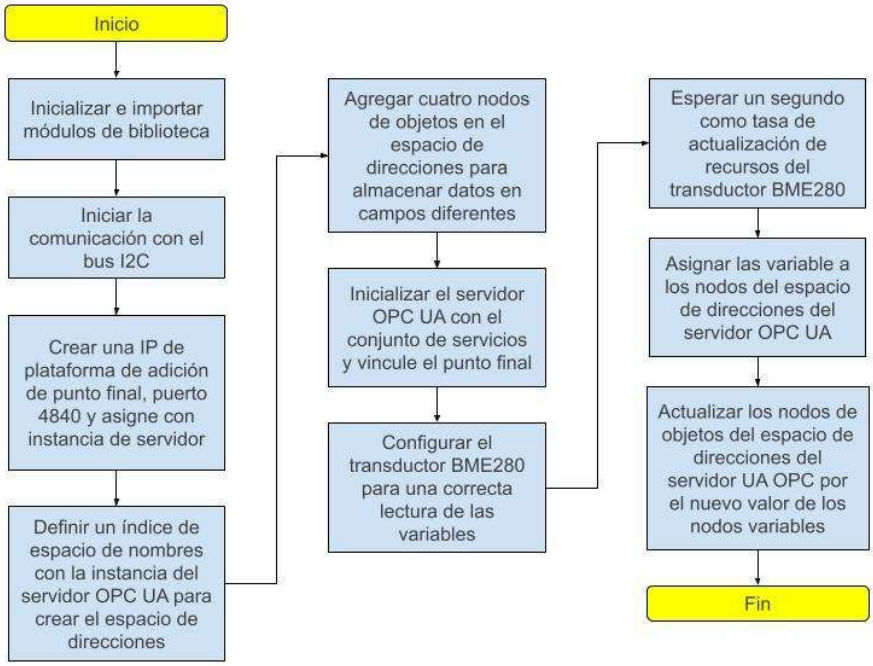


Figura 2.3. Esquema gráfico del servidor OPC UA propuesto.

2.2.3. Configuración del Cliente OPC UA

Es esencial desarrollar un cliente OPC UA para poder obtener acceso a los datos del servidor OPC UA. El proyecto contempla la integración del estándar OPC UA con servicios para el registro de datos en la nube y la visualización de esta información en una plataforma de monitoreo en red. Por lo que a continuación se muestra el desarrollo de la programación del cliente OPC UA junto con los demás dominios, el esquema gráfico de esta configuración se dispone en la Figura 2.5. Finalmente, se desarrollará un GUI que integrará todos estos servicios, permitiendo una rápida y efectiva puesta en marcha del cliente OPC UA.

2.2.3.1. Configuración de conexión a un Servidor OPC UA

Para establecer la conectividad OPC UA en el cliente, se utiliza la misma librería asyncua mencionada anteriormente. Se importa la clase "Client" de dicha biblioteca, que se utiliza

para conectarse al servidor OPC UA. Se ha creado una clase de conexión que contiene un constructor "__init__" y un método "is_connected".

El constructor "__init__" tomara los argumentos, los cuales deben ser definidos por el usuario. Entre ellos estará "server_url" para establecer la dirección del servidor OPC UA, "use_security" un atributo que indica si se requiere o no un nivel de seguridad pudiendo ser “None” o "Basic256Sha256", "certificate_path" es la ruta del archivo de certificado, en caso de usar seguridad y "private_key_path" es la ruta del archivo de clave privada, en caso de usar seguridad. El código del constructor seria:

- self.url = server_url: Asigna la dirección del servidor a la variable de instancia self.url.
- self.client = Client(self.url): Crea una instancia de un cliente OPC UA utilizando la dirección del servidor proporcionado.
- if use_security is not None: Comprueba si se especificó use_security. Si es diferente de None, indica que se desea usar seguridad.
- self.client.set_security_string(f"Basic256Sha256,SignAndEncrypt,{self.certificate},{ self.private_key}"): Si se requiere seguridad, configura el cliente con el protocolo "Basic256Sha256" además de las rutas del certificado y la clave privada proporcionadas. Si no se requiere seguridad, el atributo use_security será None y no se realizará ninguna configuración adicional de seguridad.

El método "is_connected" es utilizado para verificar si la conexión se ha realizado con éxito. Devuelve "True" si la variable "self.client" no es "None".

La configuración utilizada para conectar el cliente OPC UA al servidor OPC UA requiere especificar la URL exacta del servidor OPC UA, que incluye el nombre del host y el puerto, definiendo el punto final de conexión. También hay que considerar la configuración de seguridad del servidor OPC UA, si el servidor requiere niveles de seguridad, es necesario cargar las credenciales correspondientes para establecer la conexión. Todos estos atributos y los que se revisarán podrán ser definidos desde el GUI y no directamente desde el código. El código de la conexión se encuentra en el Anexo B.

Dado que esta aplicación está destinada a la monitorización, no es necesario especificar una acción de lectura o escritura, ya que se predefinió para realizar únicamente lectura. En relación al espacio de direcciones del servidor OPC UA, el cliente debe hacer referencia al

índice del espacio de nombre (ns) y al identificador (i) la configuración de estos atributos está en la sección 2.2.3.4.

2.2.3.2. Configuración de conexión con la base de datos en la nube

Con el propósito de conectar la aplicación y la base de datos de Clever Cloud, se utilizó la librería PyMySQL. En la clase de conexión, se utilizó el método "__init__" para inicializar la configuración de la comunicación con la base de datos. Los atributos contienen: "host" (dirección del host), "user" (nombre de usuario), "passwd" (contraseña), "db" (nombre de la base de datos) y "namedb" (nombre de la tabla en la base de datos).

En el método "__init__", se utiliza "pymysql.connect()" para la comunicación con la base de datos y se almacena el objeto de conexión devuelto en la variable "self.conn". Esta variable contiene los parámetros de conexión necesarios. Se creó un objeto cursor utilizando el método "conn.cursor()". Este cursor es utilizado para ejecutar consultas en la base de datos y se almacena en la variable "self.cur". El código de este constructor seria:

- self.conn = mysql.connector.connect(...): Crea una conexión a la base de datos MySQL utilizando los argumentos proporcionados. Esta conexión es almacenada en la variable de instancia self.conn. Las variables definidas son: user para el nombre de usuario para la conexión a la base de datos, passwd la contraseña para la conexión a la base de datos, db el nombre de la base de datos específica a la cual se va a conectar y namedb el nombre de la base de datos, que es almacenado como una variable de instancia.
- self.cur = self.conn.cursor(): Se obtiene un cursor para realizar operaciones en la base de datos. El cursor es almacenado en la variable de instancia self.cur.
- self.namedb = namedb: Almacena el nombre de la tabla de la base de datos en la variable.

El método "is_connected" se utiliza para verificar si la comunicación con la base de datos se ha establecido, se devuelve un "True" en la variable "self.conn" o un "None". Esto determina si la conexión existe o no. El código de la programación se encuentra en el Anexo B.

2.2.3.3. Configuración de conexión para la monitorización Web

Se creó una clase para la conexión entre la aplicación y la plataforma Tago.IO a través de su biblioteca. Al igual que en las demás conexiones, se utiliza el método "__init__". El parámetro para la conexión es "ubi_key", donde se especifica la clave de acceso o token

de la plataforma Tago.IO. Se utiliza el objeto "Device()" de la librería de Tago.IO. Este objeto almacena la llave o token de acceso a la plataforma. La llave de acceso es una clave de autenticación que permite interactuar y acceder a los servicios de la plataforma. La información requerida se almacena en la variable "self.myDevice". El código del constructor seria.

- self.myDevice = Device({"token": ubi_key}): Este objeto representa un dispositivo en la plataforma y se inicializa con el token de autenticación proporcionado.

El método "is_connected" se utiliza para comprobar si la conexión con la plataforma Tago.IO está establecida correctamente. Esto se verifica mediante la variable "self.myDevice". El código de la programación se encuentra en el Anexo B.

2.2.3.4. Configuración y almacenamiento de los nodos

Se propone implementar un código para leer y almacenar las direcciones de los nodos de un servidor OPC UA. El código consta de dos métodos: uno para la inicialización y otro para la lectura de las variables.

En el método de inicialización, se utiliza tres diccionarios para almacenar los valores recibidos por los nodos del servidor OPC UA. Estos diccionarios son: "var_OPCUA_dict" para almacenar las variables recibidas por la comunicación OPC UA, "sql_OPCUA_dict" para las variables que se enviarán a la base de datos alojada en la nube y "ubi_OPCUA_dict" para las variables que se enviarán a la plataforma de visualización Web. Se crearon los atributos "node_address" y "node_name" que se inicializan como None para indicar que aún no se ha leído ningún nodo de OPC UA. Por último, se crea una lista vacía "self.accumulated_values" para almacenar los atributos de los nodos del servidor OPC UA a lo largo del tiempo.

El segundo método se utiliza para leer el valor de un nodo específico en el servidor OPC UA. Este método requiere cinco argumentos: "node_address", que representa la dirección del nodo en el servidor; "node_name", que es el nombre asignado a la dirección del nodo; "var_dict", un diccionario utilizado para almacenar el valor leído y su nombre correspondiente; y "sql_dict" y "ubi_dict", que son diccionarios opcionales utilizados para almacenar los valores a registrar en la nube o para visualizarlos en la plataforma Web.

Dentro de este método, se obtiene el nodo del servidor OPC UA utilizando el método "get_node()" junto con el atributo "node_address" y luego se utiliza el método "get_value()" para obtener el valor actual del nodo. Ambos métodos son proporcionados por la biblioteca asyncua. A continuación, se utiliza "var_dict[node_name]" para agregar la clave y valor del

nodo leído al diccionario "var_dict", el cual es responsable de almacenar los valores de los nodos OPC UA.

Si los diccionarios "sql_dict" y "ubi_dict" no son None, los valores de los nodos se agregan a sus respectivos diccionarios. Estos diccionarios se actualizan con el último valor leído del nodo especificado. Esta sección de código se ejecuta dentro de un ciclo repetitivo junto con la lista de atributos para leer varios valores de los nodos del servidor a la vez.

Antes de enviar los valores a registrar en la nube, se verifica que el diccionario para la base de datos contenga al menos un valor. Si es así, se construye una consulta SQL INSERT que envuelve el nombre de las columnas y los valores como parámetros. El nombre de las columnas es el atributo "node_name" y el valor es el último valor leído del nodo. Luego, se ejecuta la consulta utilizando "cur.execute()" y se confirma la transacción utilizando "conn.commit()". Finalmente, se genera una salida para visualizar los datos enviados.

De manera similar, para enviar los valores almacenados a la plataforma Tago.IO, se verifica que el diccionario contenga al menos un valor. En caso afirmativo, se genera una lista que incluye tanto los nombres de las variables como sus respectivos valores, para luego enviar los datos utilizando la función “senData()”. Se genera una salida para visualizar la información enviada a la plataforma Web.

La parte principal del código que se está ejecutando dentro del bucle seria:

- if not all(getattr(mi_app, attr) for attr in ["node_address", "node_name"]): Comprueba que existan los atributos necesarios como: node_address y node_name.
- self.accumulated_values.append((mi_app.node_address, mi_app.node_name, mi_app.sql_std, mi_app.ubi_std)): Añade una tupla con información sobre la dirección del nodo, nombre del nodo y si se va a enviar a MySQL Cloud o TagoIO, a la lista self.accumulated_values.
- for node_address, node_name, sql_std, ubi_std in self.accumulated_values: Itera sobre las tuplas en self.accumulated_values y desempaqueta la información en variables individuales como valor y nombre.
- self.read_variable(node_address, node_name, self.var_OPCUA_dict, sql_std, ubi_std): Llama a una función read_variable con la dirección y el nombre del nodo, y los diccionarios para MySQL Cloud y TagoIO. La salida de la función para cada interacción del proyecto seria: read_variable("ns=2;i=3", "Temperature",

var_OPCUA_dict, sql_OPCUA_dict, ubi_OPCUA_dict), read_variable("ns=2;i=4", "Pressure", var_OPCUA_dict, sql_OPCUA_dict, ubi_OPCUA_dict), read_variable("ns=2;i=5", "Humidity", var_OPCUA_dict, sql_OPCUA_dict, ubi_OPCUA_dict) y read_variable("ns=2;i=6", "Altitude", var_OPCUA_dict, sql_OPCUA_dict, ubi_OPCUA_dict)

- columns = ', '.join(self.sql_OPCUA_dict.keys()): Se obtienen los nombres de las columnas de `self.sql_OPCUA_dict` y se concatenan en una cadena separada por comas. Esta cadena representa los nombres de las columnas en la tabla de la base de datos donde se insertarán los valores.

- values = ', '.join(['%s']*len(self.sql_OPCUA_dict)): Se crea una cadena de marcadores de posición `%s` separados por comas. La cantidad de marcadores de posición es igual al número de columnas en self.sql_OPCUA_dict. Esto se hace para garantizar que haya el mismo número de valores que de columnas al hacer la inserción.

- sql_query = f"INSERT INTO {self.sql_OPCUA.namedb}({columns}) VALUES ({values})": Se construye una consulta SQL de inserción utilizando la cadena columns para especificar los nombres de las columnas y la cadena values para representar los valores a insertar. La variable self.sql_OPCUA.namedb contiene el nombre de la base de datos.

- self.sql_OPCUA.cur.execute(sql_query, list(self.sql_OPCUA_dict.values())): Se ejecuta la consulta SQL utilizando el cursor de la conexión a la base de datos. Los valores que se insertarán provienen de self.sql_OPCUA_dict, y se convierten en una lista usando list().

- self.sql_OPCUA.conn.commit(): Se confirma la transacción en la base de datos. Esto implica que los cambios realizados se guardan de manera permanente en la base de datos.

- for name, value in self.ubi_OPCUA_dict.items(): Se itera a través de los elementos del diccionario self.ubi_OPCUA_dict. Cada elemento contiene un nombre de variable y su respectivo valor.

- values.append({'variable': name, 'value': value}): Se agrega una lista que contiene dos atributos: variable y value, que representan el nombre de la variable y su valor respectivamente. Estos son los datos que se enviarán a TagoIO.

- self.ubi_OPCUA.myDevice.sendData(values): Se utiliza el método sendData del objeto myDevice de la instancia de la clase Device (self.ubi_OPCUA.myDevice). Este método envía los datos almacenados en la lista values a TagoIO para su visualización.

2.2.3.5. Interfaz de gráfica de usuario para la configuración

Se propone utilizar un GUI desarrollada en QT Designer para integrar el código descrito anteriormente y crear una aplicación que permita al usuario final configurar un cliente OPC UA, añadiendo los complementos propuestos, como el envío de datos para el registro en la nube y la visualización de dichos valores en una interfaz de monitoreo en la red.

La interfaz gráfica consta de tres áreas principales. La primera área contiene los botones de control de la ventana, la segunda área es la barra de opciones o menú y la última área es el espacio de trabajo, cuyo contenido dependerá de la acción seleccionada en la barra de opciones.

La barra de opciones consta de cuatro alternativas correspondientes a cuatro áreas de trabajo. La primera área es la de Configuración, que permite ingresar los datos de conexión al servidor OPC UA, así como la configuración de la comunicación a la base de datos de Clever Cloud y la plataforma Tago.IO. Una vez establecida la comunicación con el servidor OPC UA, se podrá agregar la dirección de los nodos del servidor. Del mismo modo, una vez establecida la conexión con la plataforma SQL o Tago.IO, se podrán utilizar sus funcionalidades.

La segunda área de trabajo es la de Estado, donde se muestra la dirección del nodo en el servidor OPC UA, junto con el nombre asignado a esa dirección de nodo.

La tercera área de trabajo es la de Visualización, donde se muestra de manera continua la información contenida en los diccionarios “var_dic”t, “sql_dict” y “ubi_dict”.

La cuarta área de trabajo es la de Ayuda, donde se proporciona una documentación de usuario de la aplicación del cliente OPC UA.

A continuación, se presenta la Figura 2.4 la cual representa la interfaz gráfica para el área de configuración, donde se indican los campos correspondientes a los atributos del código que el usuario deberá completar para poner en marcha el proyecto. El código de la programación del GUI se encuentra en el Anexo C.

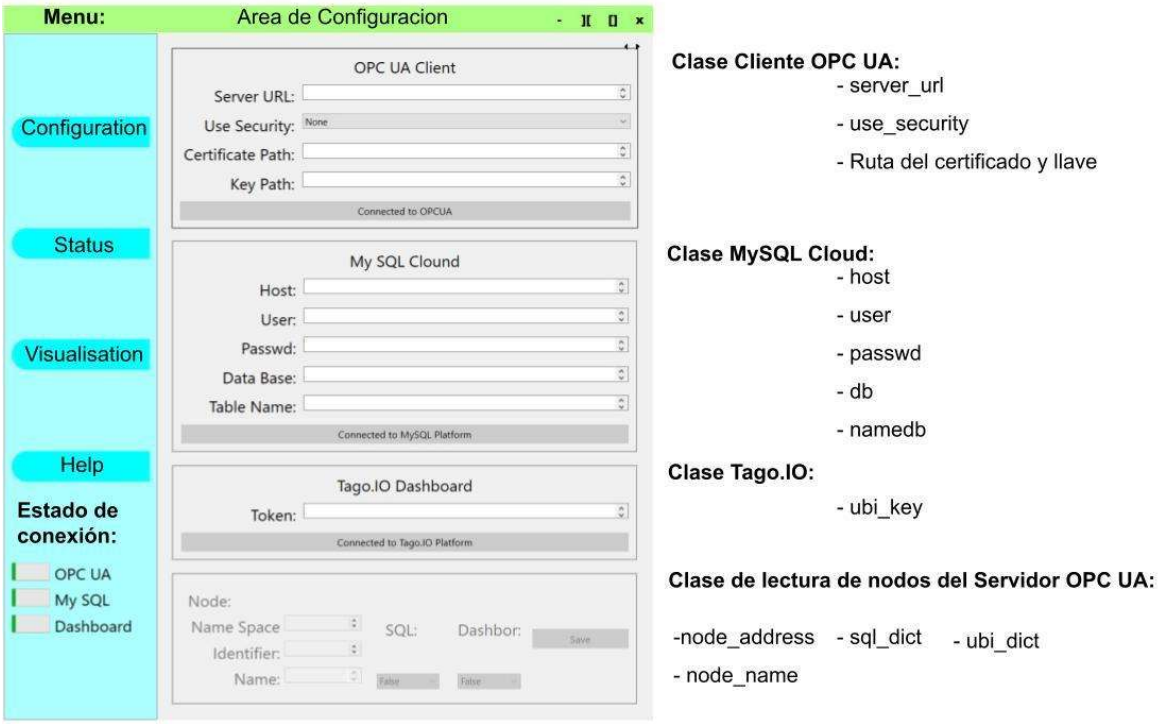


Figura 2.4. Interfaz gráfica de usuario para el cliente OPC UA.

En la Figura 2.5 se presenta el diagrama de flujo en el cual se integra el código del cliente OPC UA en conjunto con el GUI. El código del cliente OPC UA integrado al GUI se localiza en el Anexo D.

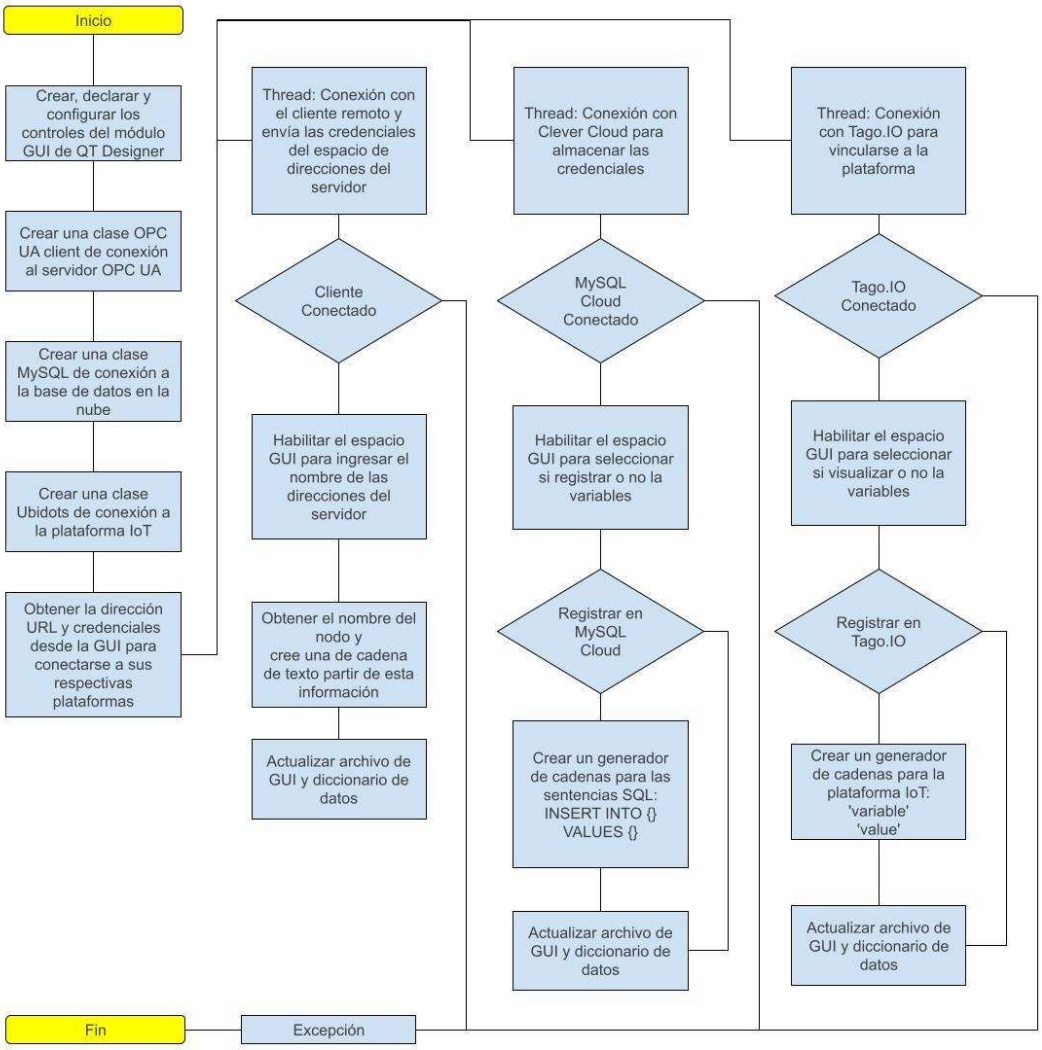


Figura 2.5. Esquema gráfico de la configuración para el cliente OPC UA.

2.3. Configuración de plataformas

2.3.1. Configuración de la base de datos en la nube

Antes de enviar los datos desde el cliente OPC UA a la base de datos en línea, es necesario realizar la parametrización de la plataforma SQL Cloud. En este proyecto, se propone utilizar la plataforma Clever Cloud, que ofrece una infraestructura para una base de datos SQL gratuita, integrándose adecuadamente con la propuesta del proyecto.

No hace falta realizar la instalación de algún software adicional para acceder a la plataforma, pero se requerirá una cuenta para su uso. El desarrollo del registro de datos en la nube implica la generación de un complemento SQL en la plataforma. Al crear el complemento, se obtiene información importante como el Host, Database Name, User y

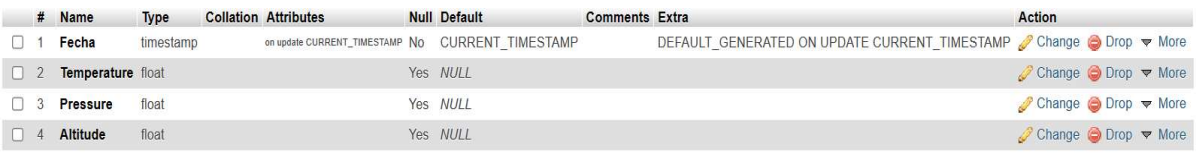
Password, que se emplean para establecer la vinculación entre la aplicación y la plataforma SQL Cloud.

La administración de la base de datos se efectúa a través de PHPMyAdmin, una herramienta frecuentemente empleada para la configuración gráfica de bases de datos. Será necesario crear una tabla que corresponda con la aplicación. En este caso, se almacenan las variables de temperatura, presión, humedad y altitud. Es importante definir previamente estas variables antes de la configuración en el cliente OPC UA para evitar cualquier inconsistencia que pueda surgir entre el registro de datos y el cliente OPC UA.

Una vez seleccionadas las variables a almacenar, se procede a la creación de la tabla llamada "opc_ua_sql". Esta tabla contiene cinco campos, uno para la fecha y cuatro para las variables previamente definidas. El primer campo corresponde a la Fecha y se define como tipo de dato "timestamp". Se asignará automáticamente al insertar una fila en la tabla, por lo que se deben agregar los atributos "CURRENT_TIMESTAMP" y "DEFAULT_GENERATED ON UPDATE CURRENT_TIMESTAMP" para especificar tanto el valor predeterminado como el comportamiento de actualización automática de la columna.

Las columnas para las variables son similares, ya que comparten las mismas características. A continuación, se describe el proceso para el campo de la variable de temperatura. El nombre de la variable es "Temperature", que debe coincidir con el asignado al atributo "node_name" para evitar conflictos entre la aplicación y la plataforma. El tipo de dato recibido es "float". Un atributo importante es permitir el valor NULL, ya que los nodos del cliente OPC UA se ingresan individualmente, lo que significa que se insertará un valor nulo en la columna si no se proporciona ningún valor durante la inserción de los atributos desde la ventana de Configuración. La configuración completa de la plataforma se encuentra en el Anexo E.

En la Figura 2.6, se presenta una captura de la tabla para el registro de datos en la nube desde la plataforma Clever Cloud.



	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	Fecha	timestamp		on update CURRENT_TIMESTAMP	No	CURRENT_TIMESTAMP		DEFAULT_GENERATED ON UPDATE CURRENT_TIMESTAMP	Change Drop More
☐	2	Temperature	float			Yes	NULL			Change Drop More
☐	3	Pressure	float			Yes	NULL			Change Drop More
☐	4	Altitude	float			Yes	NULL			Change Drop More

Figura 2.6. Esquema de la tabla opc_ua_sql.

2.3.2. Configuración de las plataformas de monitorización Web

La propuesta del proyecto es visualizar los datos del cliente OPC UA en una interfaz gráfica accesible desde la red. Para ello, se implementó la plataforma Tago.IO, que permite visualizar los datos de temperatura, presión, humedad y altitud desde cualquier dispositivo con conexión a Internet. Tago.IO se presenta como una plataforma de desarrollo de soluciones IoT para empresas y proyectos de investigación. Ofrece herramientas como la creación de alertas para los valores registrados, lo que facilita el desarrollo de soluciones IoT. La plataforma también es adecuada para la educación, ya que cuenta con planes gratuitos, solo requiriendo el registro en la plataforma.

Para utilizar Tago.IO, es necesario crear una cuenta. Una vez creada, no es necesario crear las variables manualmente, ya que una vez que la información se envía desde la aplicación del cliente OPC UA, se registra automáticamente en la plataforma y las variables pueden asignarse a widgets. Sin embargo, es importante obtener el token del dispositivo generado para crear un vínculo entre la plataforma y la aplicación cliente OPC UA.

La Figura 2.7 exhibe el dispositivo creado con el nombre "OPC_UA". Desde la pestaña "Información General" se generó el token necesario para la comunicación entre la plataforma y la aplicación del cliente OPC UA. La configuración completa de la plataforma se encuentra en el Anexo E.

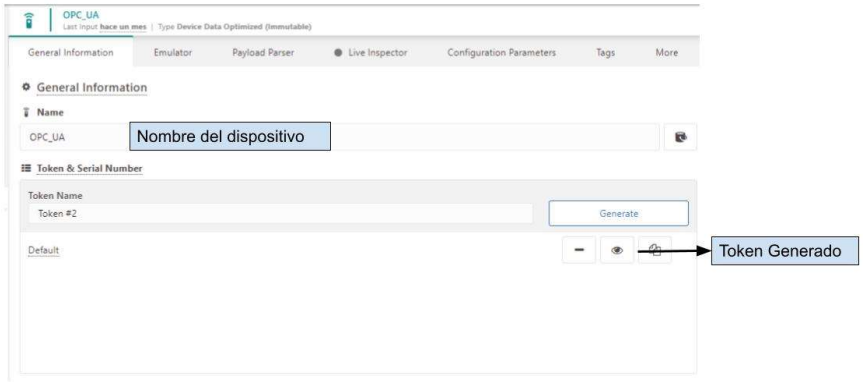


Figura 2.7. Token o clave de acceso remoto.

Establecida la conexión entre el cliente OPC UA, la plataforma y enviadas las variables a monitorear se pueden observar los datos a ser emparejados a los widgets de la plataforma para ofrecer una interfaz de monitorización. En la Figura 2.8, se exhiben las constantes de temperatura, presión, humedad y altitud disponibles en la plataforma, que se designó desde el cliente OPC UA.

Figura 2.8. Datos disponibles a ser utilizados en los widgets.

La plataforma Tago.IO ofrece la capacidad de gestionar alarmas para monitorear las variables enviadas por el cliente OPC UA, lo que simplifica la programación tanto en el servidor como en el cliente OPC UA. Tago.IO permite enviar notificaciones en la interfaz que pueden ser visualizadas por el operador, quien debe responder a la notificación desde la plataforma. Asimismo, dentro del widget "Tabla Dinámica" se pueden utilizar condiciones de color para resaltar las variables críticas durante el monitoreo.

En este caso, se crearon alertas tanto de notificación como de condiciones de color para la variable de temperatura. Las alertas se activarán cuando se supere un valor de consigna establecido.

A continuación, se muestran las Figuras 2.9 y 2.10 que ilustran la creación de las alertas de notificación y condiciones de color para la variable de temperatura.

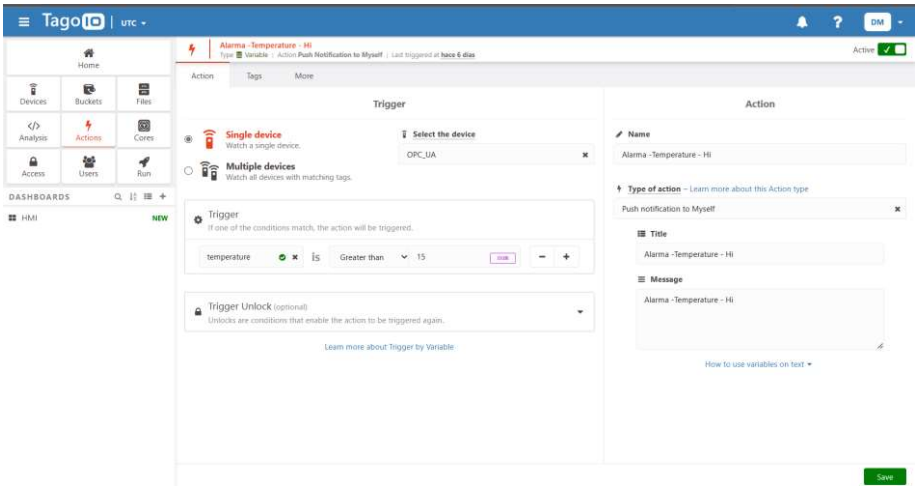


Figura 2.9. Configuración de las notificaciones para alertas de temperatura.

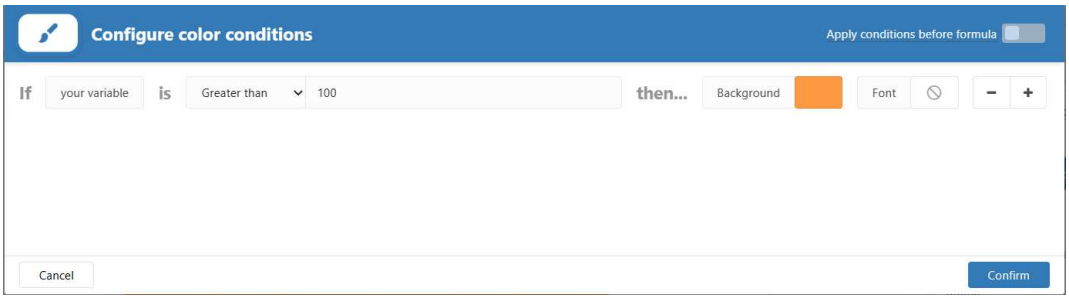


Figura 2.10. Configuración de las condiciones de color frente alertas de temperatura.

2.4. Entorno de pruebas del proyecto

En esta sección se describe el entorno de pruebas utilizado para verificar la estabilidad y funcionamiento del sistema propuesto de monitorización en un entorno cerrado. Es fundamental realizar pruebas exhaustivas para garantizar que el proyecto cumpla con su funcionamiento esperado y para identificar las posibles áreas de mejora.

2.4.1. Maqueta de pruebas del proyecto

En el entorno cerrado para las pruebas se implementó un sistema de control de temperatura y humedad, con el fin de poner a prueba el proyecto en un entorno real. Este sistema utiliza un controlador el cual procesa los datos provenientes de su propio sensor y controlan sus actuadores correspondientes. Los sensores tienen la función de tomar las mediciones de temperatura y humedad, mientras que los actuadores consisten en una niquelina (para controlar la temperatura) y un humidificador (para controlar la humedad).

Los dispositivos empleados en la maqueta para simular un ambiente cerrado se detallan en la Tabla 2.2.

Tabla 2.2. Componentes de la maqueta de pruebas.

Dispositivo	Descripción
Controlador STC-3028	Tarjeta electrónica encargada de controlar la temperatura y humedad. El control de la tarjeta es On/Off.
Sonda	Instrumento de medida de temperatura y humedad.
Interruptor	Interruptor de encendido y apagado del del controlador STC.
Luz piloto	Luz indicadora de encendido del controlador STC.
Niquelina	Fuente de calor.
Humidificador	Fuente de humedad.

El controlador STC-3028 utiliza su propia sonda para captar la temperatura y humedad. Su uso habitual es para procesos industriales y domésticos, donde se demanda un control preciso de temperatura y humedad. Los atributos específicos del dispositivo se detallan en la Tabla 2.3."

Tabla 2.3. Características de la tarjeta STC-3028.

Voltaje de entrada	110 VAC	
Rango de temperatura	-20 a + 80 °C	
Rango de humedad	0 a +100%	
Precisión	± 1 °C	0,1%
Capacidad de contacto de salida de relé	10 A / 110 VAC	

La Figura 2.11 exhibe el esquema gráfico que se emplea para el control de temperatura y humedad. Los parámetros utilizados para el control son utilizados a modo de ejemplo, estos parámetros son mayores a los presentes en el ambiente, con el fin de visualizar una variabilidad de temperatura y humedad durante las pruebas.

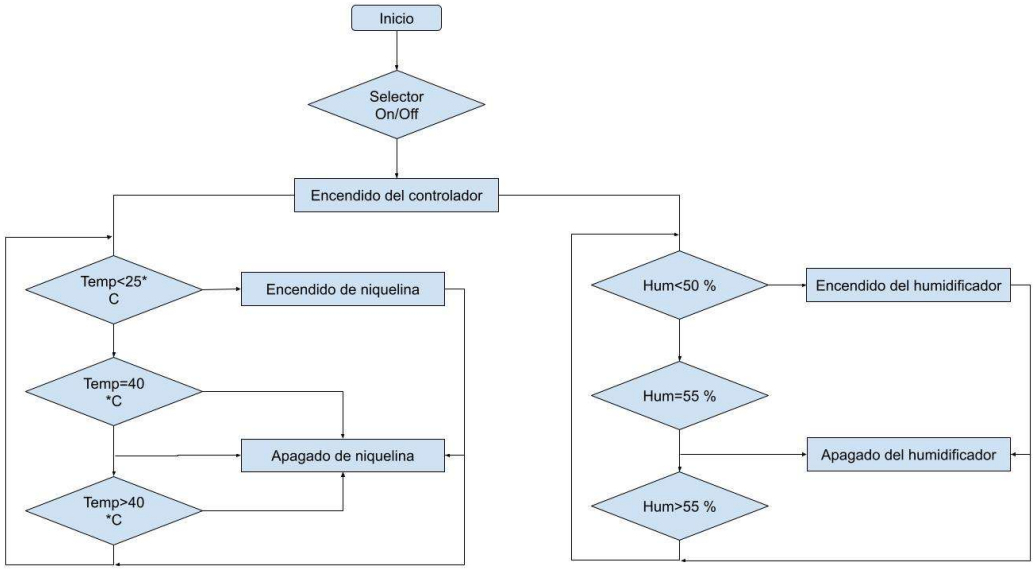


Figura 2.11. Configuración del control STC-3028.

La tarjeta de control proporciona una instalación sencilla. El diagrama de conexión del controlador se muestra en la Figura 2.12, lo cual facilita la interpretación y la conexión de los diferentes componentes propuestos para el entorno de pruebas del proyecto. También se incluye el diagrama de conexión para el servidor OPC UA.

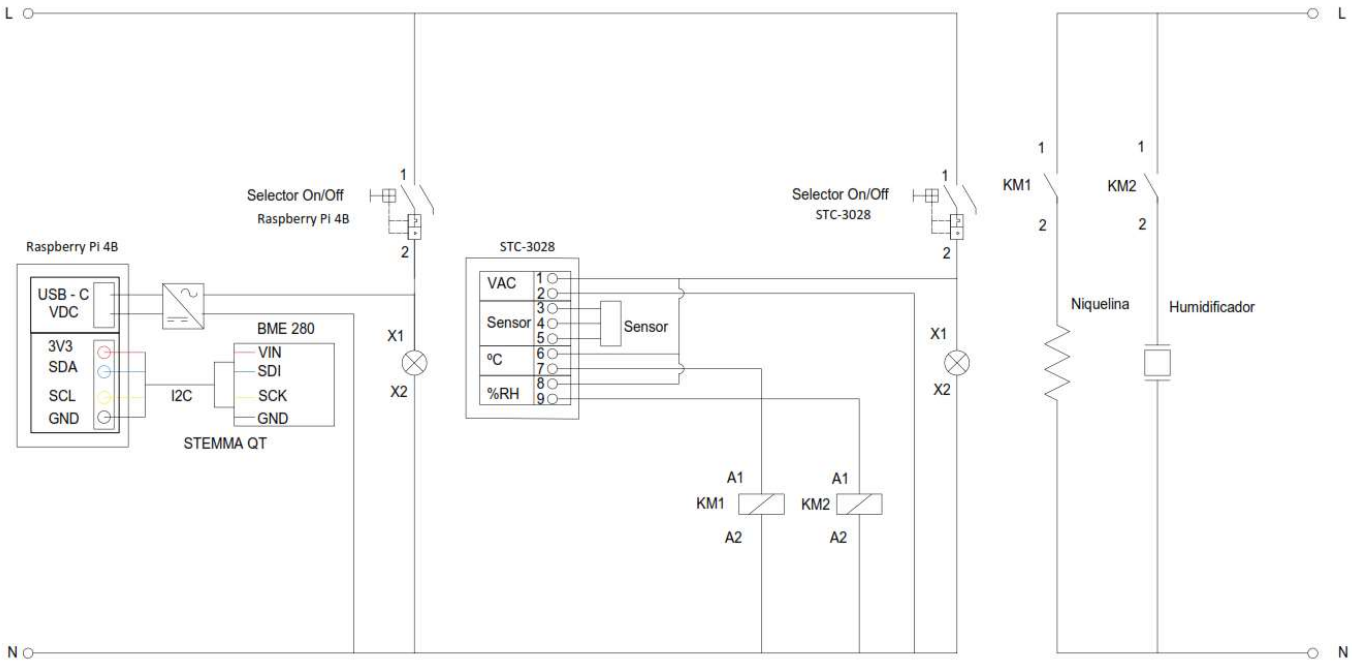


Figura 2.12. Diagrama de conexión.

2.4.2. Instrumento de medición patrón

Para verificar la precisión de los datos obtenidos durante el monitoreo y asegurarse de que no se hayan corrompido o transmitido incorrectamente, se utilizará un registrador de datos. En esta instancia, se ha optado por elegir el RHT35, un dispositivo capaz de registrar mediciones de temperatura, presión y humedad.

El registrador tiene una capacidad de almacenamiento de hasta 48.000 mediciones, con un período mínimo de 30 segundos entre cada registro. Los datos se transferirán a un computador a través de una conexión USB. Esta metodología garantiza la integridad y precisión de los datos recopilados durante el monitoreo.

En la Figura 2.13 se presenta una imagen del dispositivo registrador, que tiene un intervalo de medición de temperatura de -30 °C a +70 °C, presión de 300 hPa a 1.100 hPa y humedad de 1% a 99,9%.

Figura 2.13. Registrador de datos RHT35.

2.4.3. Disposición de los elementos del proyecto

A continuación, se muestra la disposición de los dispositivos electrónicos presentados en el capítulo, incluyendo los elementos del servidor OPC UA junto con su transductor, así como los elementos que constituyen el entorno de pruebas del proyecto. Se representa visualmente esta disposición en la Figura 2.14.

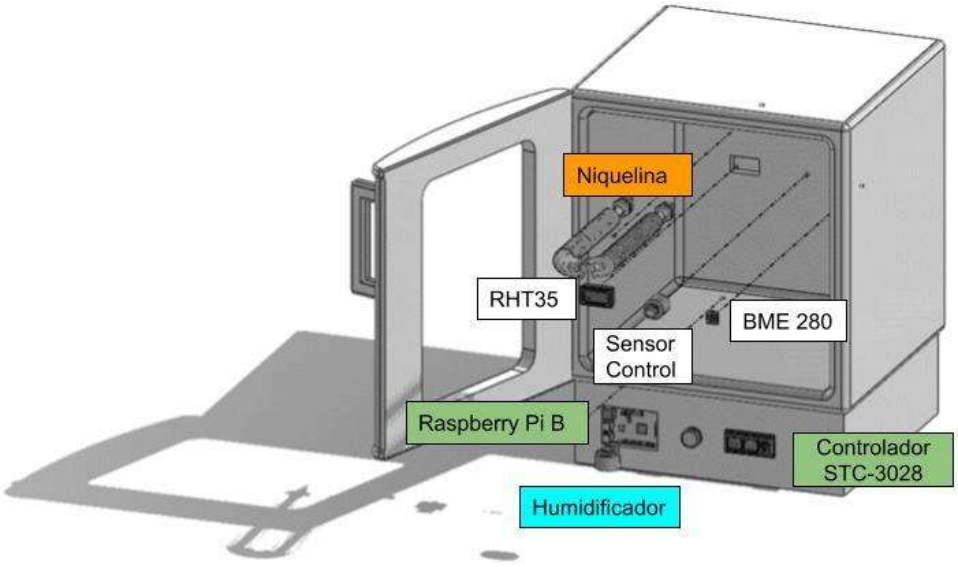


Figura 2.14. Disposición de los dispositivos del proyecto.

3. RESULTADOS Y DISCUSIÓN

En este capítulo se detallan los resultados alcanzados mediante los ensayos ejecutados en la puesta en marcha de un sistema de adquisición de datos y registro en la nube basado en OPC UA. Se muestran los datos enviados desde el servidor OPC UA hasta el cliente OPC UA, el cual se presentará en un GUI. Adicionalmente, el cliente OPC UA envía los datos a las plataformas de Clever Cloud y Tago.IO para obtener una base de datos en línea y una interfaz de visualización Web, respectivamente. Los valores medidos son: temperatura, presión, humedad y altitud. La configuración de las plataformas debe realizarse previamente, ya que el cliente OPC UA realiza el registro en cada una de las plataformas. Se exponen los datos registrados en la nube en conjunto con la interfaz de visualización Web, en base al estándar OPC UA para asegurar la calidad de los datos registrados durante un lapso de tiempo de ocho horas en diferentes intervalos de tiempo.

En la Figura 3.1, se presenta la maqueta construida que simula un entorno cerrado y controlado, conforme se detalló en el capítulo previo. Esta maqueta engloba un servidor OPC UA y un controlador encargado de regular tanto la temperatura como la humedad, el propósito de este controlador consiste en emular un entorno real para el proyecto tecnológico.

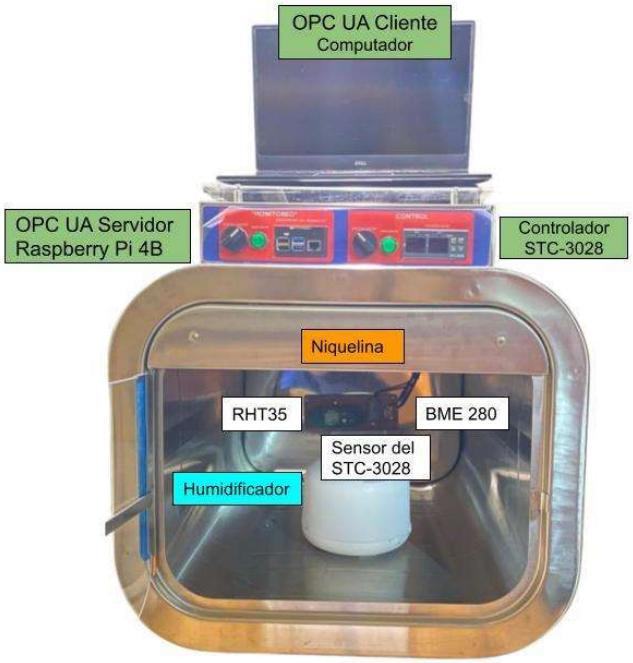


Figura 3.1. Maqueta de pruebas conjunto con el sistema de monitorización basado en el estándar OPC UA.

A continuación, se muestra la Tabla 3.1 que contiene los atributos de comunicación del servidor OPC UA.

Tabla 3.1. Parámetros del servidor OPC UA.

Dirección URL	opc.tcp://192.168.0.118:4840			
Variables	Temperatura	Presión	Humedad	Altitud

Como se mencionó en el capítulo dos, antes de poder registrar los datos en la nube o visualizarlos en una interfaz, es necesario establecer la conexión con la aplicación del cliente OPC UA, para poder agregar las direcciones de las variables del servidor OPC UA. Los atributos de conexión dispuestos para la prueba de este proyecto se muestran en la Tabla 3.2.

Tabla 3.2. Atributos de conexión del cliente OPC UA.

OPC UA Cliente		
Dirección URL	opc.tcp://192.168.0.118:4840	
Use Security	None	Basic256Sha256_SignAndE
Certificate Path		G:\\Certificados\\cert_V4.der
Key Pacth		G:\\Certificados\\key_V4.pem
Clever Cloud		
Host	bqiv4fnapagfifbukruh- mysql.services.clever-cloud.com	
User	uxfsjsy2hwvivvvh	
Passwd	nOqDfP80tAi0a28l4EhJ	
Data Base	bqiv4fnapagfifbukruh	
Name DB (Tabla)	opc_ua_sql	
Tago.IO		
Token	73a46ace-7382-4d8a-b4b3-68d83898705c	

Un punto importante en la comunicación es la seguridad, por lo que se ofrece la opción de seleccionar un nivel de seguridad según las necesidades del usuario final. Se efectúan pruebas de conexión tanto en modo seguro como en modo sin seguridad.

Una vez establecida la conexión con el cliente OPC UA, así como con las plataformas Clever Cloud y Tago.IO, se podrán utilizar todas las funcionalidades propuestas en el proyecto. Los parámetros empleados para las pruebas están especificados en la Tabla 3.3.

Los nombres asignados desde el cliente OPC UA fueron redactados en inglés, siguiendo la disposición establecida en la tabla de la plataforma Clever Cloud (ver Figura 2.6). Es crucial que estos nombres sean idénticos, dado que la consulta SQL utilizará el atributo "node_name " para asignarlo al campo correspondiente de su tabla.

Tabla 3.3. Direcciones y configuraciones del cliente OPC UA.

Variables	Temperatura	Presión	Humedad	Altitud
Dirección del nodo	ns=2;i=3	ns=2;i=4	ns=2;i=5	ns=2;i=6
Name	Temperature	Pressure	Humidity	Altitude
SQL Cloud	True	True	True	True
Tago.IO	True	True	True	True

Como se muestra en la Tabla 3.3, se ha configurado el envío de todos los nodos a las plataformas seleccionadas. Esta configuración se realizó para verificar la fiabilidad y funcionalidad del proyecto propuesto. Sin embargo, es importante destacar que esta configuración puede adaptarse según las necesidades del usuario final. El usuario tiene la opción de utilizar o no las plataformas y puede seleccionar los nodos específicos que desea enviar a cada plataforma. De esta manera, se brinda flexibilidad para la configuración según los requisitos y preferencias del usuario final.

3.1. Pruebas de OPC UA con servicios propuestos

Las primeras pruebas son de conexión tanto del estándar OPC UA como de las plataformas Clever Cloud y Tago.IO. Además, para verificar la disponibilidad de los datos y servicios, se realizaron pruebas durante ocho horas continuas para el envío y recepción de las cuatro variables utilizando el estándar OPC UA, estas pruebas implican el registro en una base de datos en la nube y la visualización en una plataforma Web. Las variables de temperatura presión y humedad registradas son contrastadas entre el almacenamiento de Clever Cloud y el registrador RHT35. Estas pruebas fueron realizadas en diferentes intervalos de tiempo de diez, veinte y treinta segundos para conocer el tiempo en que tarda cada registro.

3.1.1. Prueba de conexión con OPC UA y servicios propuestos

Los hallazgos de comunicación logrados al utilizar el estándar OPC UA para los sistemas de monitorización en conjunto con la conexión a la base de datos alojada en Clever Cloud y la plataforma de visualización Tago.IO.

El proyecto se desarrolló con el propósito de establecer las conexiones entre los servicios propuestos. El servicio principal involucra la comunicación entre el servidor OPC UA alojado en la tarjeta Raspberry Pi 4B y el cliente OPC UA en una PC.

A continuación, se muestra la prueba de verificación de la comunicación entre el servidor OPC UA y el cliente OPC UA, así como las conexiones para el registro y la visualización. El servidor se configuró sin niveles de seguridad siendo configurado Use Security como "None", lo cual eliminó la necesidad de consideraciones adicionales por parte del cliente. Los parámetros de configuración se detallan en la Tabla 3.2, incluyendo los atributos de conexión del cliente OPC UA sin seguridad, así como los atributos de conexión para la base de datos alojada en Clever Cloud y la plataforma de visualización Tago.IO. En la Figura 3.2 se muestran los atributos de configuración y el estado de la conexión.

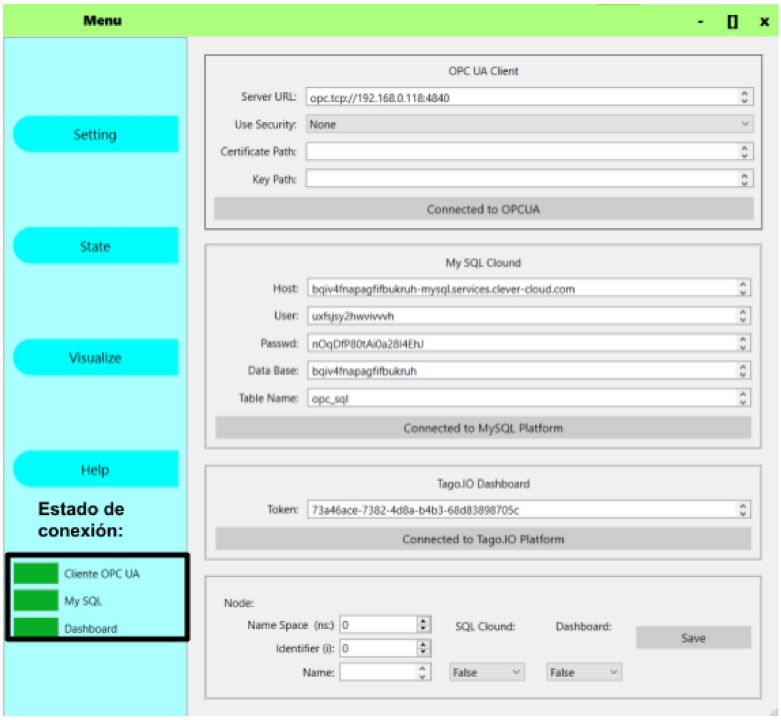


Figura 3.2. Funcionalidad del cliente OPC UA y demás servicios sin un nivel de seguridad.

En el proyecto se logró implementar el nivel más alto de seguridad propuesto por el estándar OPC UA. Para lograrlo, se incluyeron certificados y claves tanto en el servidor OPC UA como en el cliente OPC UA. Estos certificados y claves fueron auto firmados mediante OpenSSL para verificar el funcionamiento del programa. La decisión de utilizar el nivel de seguridad en la comunicación OPC UA dependerá exclusivamente del usuario final

y de los requisitos específicos de su proyecto. Los parámetros de configuración utilizados están disponibles en la Tabla 3.2.

La Figura 3.3 ilustra una exitosa conexión entre el estándar OPC UA y las plataformas propuestas, con la configuración del nivel de seguridad establecido a través del atributo Use Security como "Basic256Sha256" en conjunto con las rutas del certificado y clave.

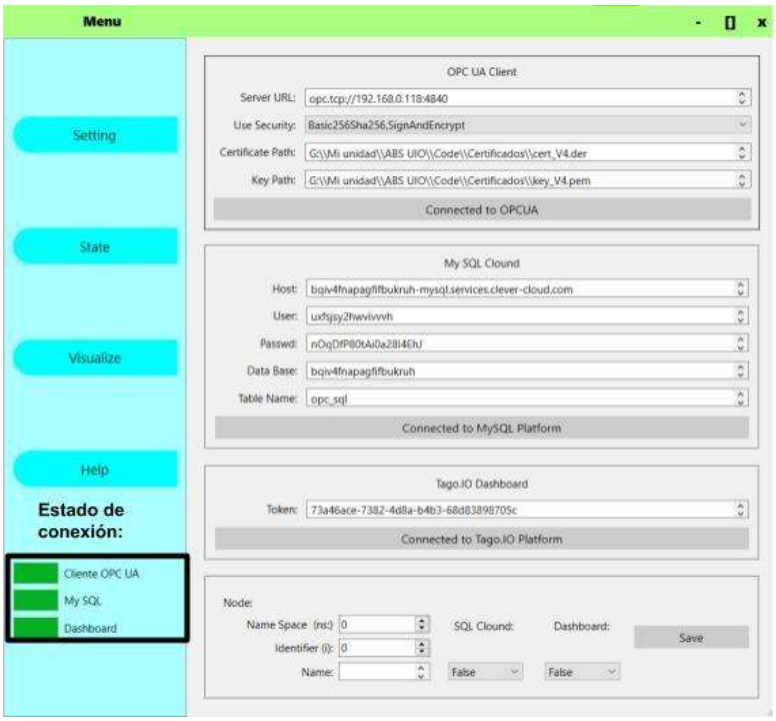


Figura 3.3. Funcionalidad del cliente OPC UA y demás servicios con un nivel de seguridad.

Las Figuras 3.2 y 3.3 evidencian el éxito en el establecimiento de una conexión con cada servicio propuesto tanto el estándar OPC UA como con Clever Cloud y Tago.IO, lo cual se puede apreciar en las barras de estado en color verde encerradas en el cuadrado en negro.

Hay que resaltar que, en las Figuras 3.2 y 3.3 se han habilitado los campos correspondientes a su servicio, lo que permite al usuario ingresar la dirección del nodo OPC UA y el utilizar tanto el registro en la nube como la visualización en una plataforma Web. Estos campos solo estarán disponibles si la conexión se ha establecido correctamente a su respectivo servició, lo cual juega un papel esencial en la prevención de errores en la configuración de la aplicación alojada en la PC.

3.1.2. Resultados de la implementación de OPC UA y servicios propuestos

Para conocer de manera integral el funcionamiento del proyecto, se realizaron tres pruebas controladas utilizando la maqueta diseñada. El objetivo principal de estas pruebas fue comparar los datos recolectados por el registrador de datos y la comunicación del estándar OPC UA al simular un entorno real de operación, los cuales fueron registrados en la base de datos de la plataforma Clever Cloud, analizando los tiempos entre cada registro para determinar el tiempo promedio de registro en diferentes intervalos. También se verificó el adecuado funcionamiento de la interfaz de visualización en la plataforma Web Tago.IO.

En las pruebas, se monitorean variables, como temperatura, presión, humedad y altitud (aunque esta última no es considerada en la comparación de resultados), para verificar la precisión y confiabilidad de los datos obtenidos. Los resultados de estas pruebas proporcionaron una visión integral del proyecto y su capacidad para recopilar y registrar datos de manera efectiva. A continuación, se muestran los resultados obtenidos en cada intervalo de tiempo dispuesto, visualizándose estas pruebas en las Figuras 3.4, 3.5 y 3.6, correspondiendo la línea roja a los registros de Clever Cloud y la línea azul representa los valores del registrador de datos RHT35.

La primera prueba se llevó a cabo el 2023-06-07 a las 09:00 de la mañana, con una duración cercana a las ocho horas y un intervalo de registro de 10 segundos. Durante esta prueba, se obtuvieron resultados satisfactorios, ya que no se detectaron errores durante la comunicación. La Figura 3.4 muestra de forma clara una comunicación continua y sin interrupciones. El intervalo más largo entre registros fue de 59 segundos y el promedio entre registros fue de 11,11 segundos.

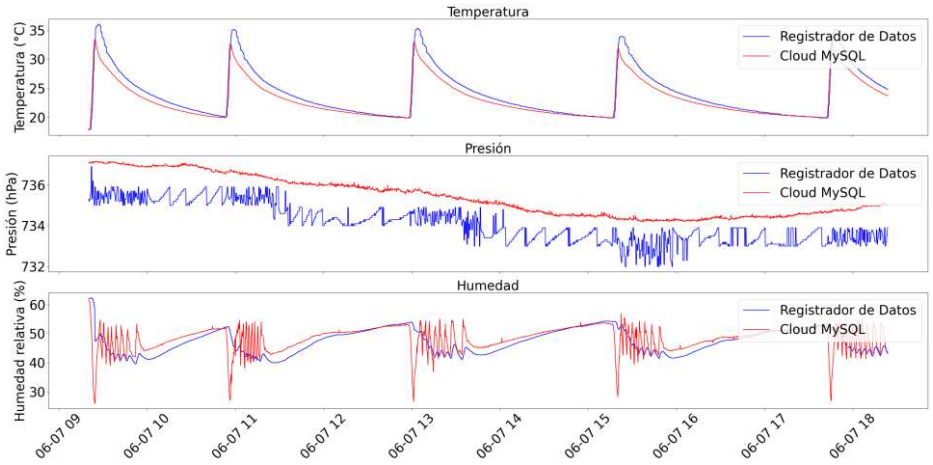


Figura 3.3. Línea de tiempo de la prueba por un intervalo de 10 s.

La segunda prueba, realizada el 2023-06-08 a las 09:00 de la mañana, se mantuvo la duración de ocho horas, pero se modificó el intervalo de registro a 20 segundos. Los resultados alcanzados fueron positivos, ya que no se registraron errores en la comunicación. La Figura 3.5 refleja una comunicación continua sin interrupciones durante toda la prueba. El intervalo más largo entre registros fue de 50 segundos y el promedio entre registros fue de 21,10 segundos.

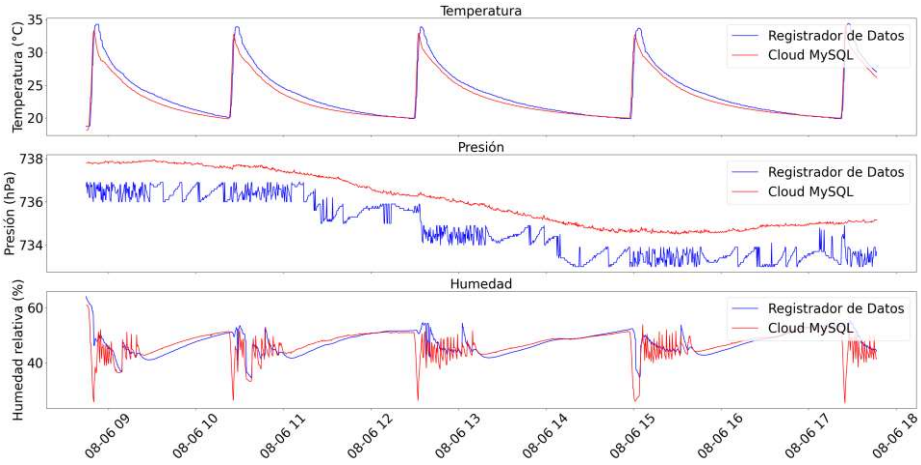


Figura 3.5. Línea de tiempo de la prueba por un intervalo de 20 s.

En cuanto a la tercera prueba, se llevó a cabo el 2023-06-09 a las 09:00 de la mañana, con una duración de ocho horas y un intervalo de registro de 30 segundos. Los hallazgos obtenidos fueron favorables, ya que no se registraron errores en la comunicación durante el desarrollo de la prueba. En la Figura 3.6 se exhibe una comunicación constante y sin interrupciones a lo largo de toda la prueba. El intervalo más largo entre registros fue de 51 segundos, mientras que el promedio entre registros fue de 31,14 segundos.

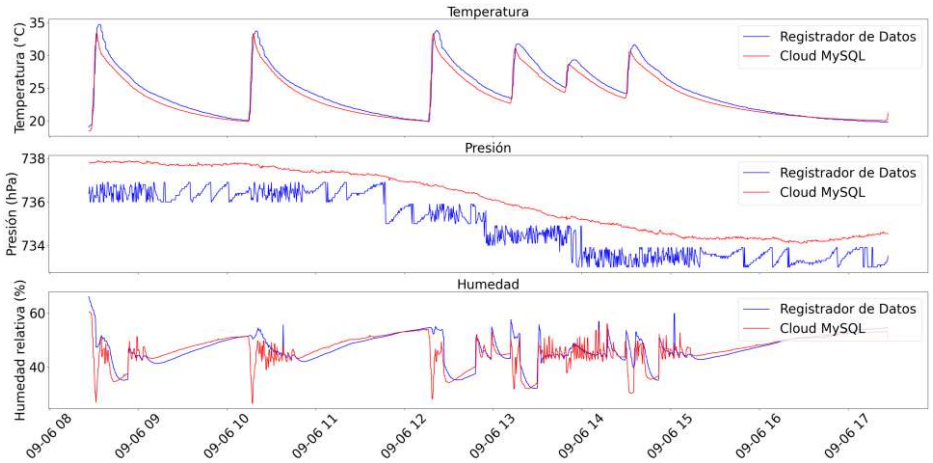


Figura 3.6. Línea de tiempo de la prueba por un intervalo de 30 s.

En las Figuras 3.4, 3.5 y 3.6 se evidencia que existen diferencias entre los registros en la base de datos y los almacenados por el registrador RHT35. Esta discrepancia puede atribuirse al factor mencionado anteriormente, que establece que las tarjetas de desarrollo, incluyendo la Raspberry Pi, tienen una precisión relativamente baja, pero son eficientes por su bajo costo.

A continuación, se expone una explicación detallada de las gráficas mostradas, con el fin de verificar la similitud entre los valores registrados por la base de datos y los del registrador de datos RHT35. Se monitorean cambios de temperatura, presión y humedad para garantizar la precisión de los datos obtenidos. El proyecto se enfoca únicamente en la monitorización, ya que las mediciones obtenidas con el estándar OPC UA se registran y se visualizan con herramientas de la computación en la nube.

Para conocer el nivel de discrepancia entre las dos colecciones de datos para cada una de las pruebas se empleó la Raíz del Error Cuadrático Medio (RMSE) [27]. Se utilizó un script en Python para calcular el RMSE que se obtiene al elevar al cuadrado la sustracción entre los valores estimados (Clever Cloud) y los datos reales (RHT35), luego se promedia y finalmente se toma la raíz cuadrada del resultado. Cuanto menor sea el valor calculado mejor será la precisión del proyecto implementado. Además, el script utilizado calcula el promedio total de cada una de las mediciones según el intervalo y dispositivo, la fórmula empleada suma todos los valores de la variable y luego divide esta suma por el número de registros obtenidos.

La Tabla 3.4 muestra el número de registros realizados en cada prueba, así como los valores promedio de temperatura, presión y humedad recolectados en cada una de ellas y el RMSE obtenido.

Tabla 3.4. Exposición de los resultados logrados durante las pruebas.

Intervalo	Variable	Dispositivos	Registros	Media	RMSE
10 (segundos)	Temperatura (°C)	RHT35	1128	23,85	1,44
		Cloud	2951	23,16	
	Presión (hPa)	RHT35	1128	734,15	1,36
		Cloud	2951	735,41	
	Humedad (%)	RHT35	1128	47,86	4,52
		Cloud	2951	48,25	
20 (segundos)	Temperatura (°C)	RHT35	1091	23,67	1,02
		Cloud	1543	23,13	
	Presión (hPa)	RHT35	1091	734,77	1,33
		Cloud	1543	736,04	
	Humedad (%)	RHT35	1091	47,49	4,21
		Cloud	1543	46,64	
30 (segundos)	Temperatura (°C)	RHT35	1085	24,29	0,94
		Cloud	1043	23,68	
	Presión (hPa)	RHT35	1085	734,91	1,21
		Cloud	1043	736,06	
	Humedad (%)	RHT35	1085	46,95	4,02
		Cloud	1043	46,16	

Para calcular el RMSE se tomó en cuenta la discrepancia entre el número de registros disponibles, por lo que se empleó el método de interpolación para estimar los datos faltantes del dispositivo RHT35, ya que como se mencionó su intervalo mínimo de registro es 30 segundos. Para cada intervalo en particular, se calcularon los valores de RMSE para las tres variables: temperatura, presión y humedad. Estos valores reflejan la diferencia promedio entre las mediciones del dispositivo RHT35 y los registros obtenidos en la base de datos. A continuación, se muestra la explicación adecuada para cada variable y momento particular:

- Temperatura: El RMSE de la temperatura muestra la discrepancia promedio entre las mediciones del dispositivo RHT35 y las mediciones del proyecto. A medida que aumenta el tiempo (de 10 a 30 segundos), el valor del RMSE disminuye. Esto

sugiere que la precisión de las mediciones del sensor en términos de temperatura mejora a medida que el intervalo de tiempo aumenta.

- Presión: El RMSE de la presión indica la discrepancia promedio entre las mediciones del dispositivo RHT35 y las mediciones del proyecto. En este caso, no hay una tendencia clara a medida que pasa el tiempo. Los valores de RMSE varían ligeramente, pero se mantienen en un rango similar.
- Humedad: El RMSE de la humedad muestra la diferencia promedio entre las mediciones del dispositivo RHT35 y las mediciones del proyecto. Similar a la temperatura, a medida que aumenta el tiempo, el valor del RMSE disminuye. Esto sugiere que la precisión de las mediciones del sensor en términos de humedad mejora a medida que pasa el tiempo.

Es crucial considerar el intervalo de registro, ya que un intervalo más corto, como 10 segundos, puede proporcionar una visión menos precisa de los cambios en el tiempo, mientras que un intervalo más largo, como 30 segundos, puede ofrecer una visión más general y suavizada de los datos. Lo cual demuestra una satisfactoria calidad de los valores registrados al compararlos con un instrumento de medición específico para el monitoreo frente a una plataforma de desarrollo.

Durante las pruebas, se visualizaron las mediciones de temperatura, presión, humedad y altitud en la plataforma Web Tago.IO, siguiendo el intervalo de tiempo establecido. La Figura 3.6 ilustra las gráficas de tendencias que reflejan el adecuado seguimiento de las variables, permitiendo comprender los cambios en el entorno físico a lo largo del tiempo y facilitando el análisis durante el monitoreo. Las notificaciones de alarmas resultaron muy útiles dentro de la plataforma de monitoreo, con respuestas rápidas y una destacada señal visual para transmitir mensajes urgentes al usuario.

En términos generales, la plataforma demostró ser adecuada para el propósito del proyecto. A pesar de su interfaz minimalista con la información necesaria, se destaca por su eficiencia en la detección de anomalías en el entorno físico. La simplicidad de la plataforma la hace intuitiva, lo que facilita la identificación de cambios en el entorno.

La plataforma Web actualiza los gráficos y textos de manera sincronizada con la base de datos alojada en la nube, asegurando procesos continuos. La recepción de información por parte del cliente OPC UA se traduce rápidamente en la generación de gráficos de tendencias y alertas.

A continuación, se presenta la Figura 3.7 que fue tomada desde la plataforma Web Tago.IO durante la prueba para el intervalo de 30 segundos. La plataforma está disponible desde cualquier dispositivo con acceso a internet y conexión a la plataforma Tago.IO.

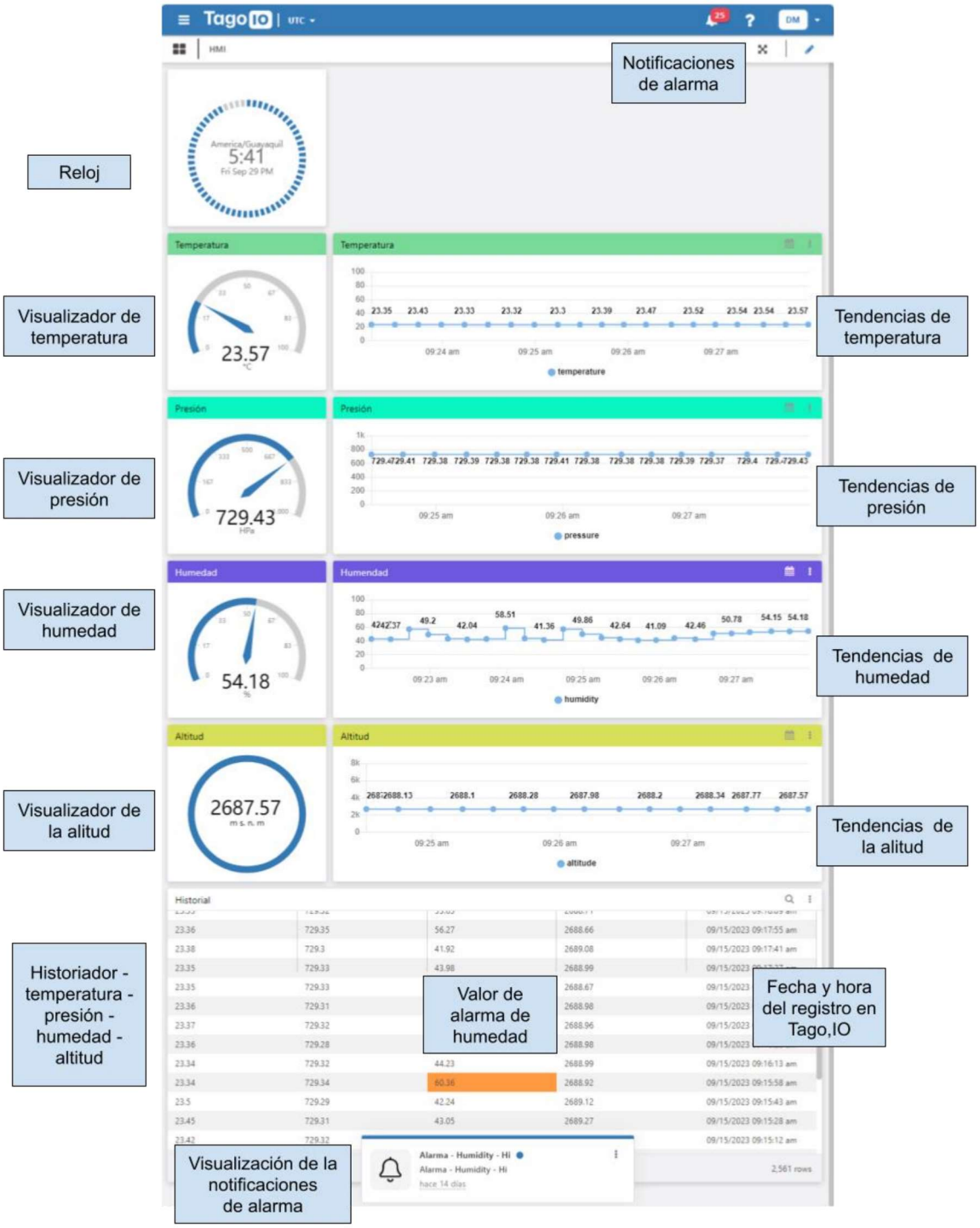


Figura 3.7. Interfaz de visualización de la plataforma Web Tago.IO.

4. CONCLUSIONES Y RECOMENDACIONES

La implementación de una solución tecnológica basada en OPC UA, siguiendo el modelo Cliente/Servidor, permitió un monitoreo continuo durante ocho horas, con diferentes intervalos de tiempo en cada prueba. No obstante, se identificaron retrasos en el registro y visualización de variables de temperatura, presión y humedad durante las pruebas. A pesar de estos inconvenientes, la solución tecnológica desarrollada en Python ofrece flexibilidad para sistemas de monitoreo. El proyecto resulta útil en entornos industriales, ya que permite visualizar y registrar los datos de los dispositivos de campo, posibilitando un seguimiento constante y facilitando decisiones ágiles y precisas basadas en la información recopilada.

Para implementar el estándar OPC UA, es necesario configurar los parámetros de conexión entre el servidor y un cliente OPC UA, como la dirección URL y de requerir la política de seguridad en conjunto con el certificado y la clave. Para acceder a los datos internos del servidor, el cliente debe especificar la dirección del nodo. En este proyecto, se utilizó el modelo Raspberry Pi 4B y el sensor BME280 de Adafruit, ambos compatibles con el protocolo de comunicación I2C. Al utilizar una tarjeta de desarrollo abre la posibilidad de que industrias que no posean el hardware, puedan aprovechar las herramientas de la computación en la nube basada en OPC UA. Durante la implementación, se pudo verificar que el estándar OPC UA permite la comunicación entre diferentes dispositivos con diferentes sistemas operativos, utilizando Raspberry Pi OS para el servidor y Windows 10 para el cliente.

Se implementó un proyecto tecnológico basado en OPC UA, siguiendo el modelo Cliente/Servidor. Para el registro de datos, se utilizó Clever Cloud y se empleó el complemento SQL, para la escritura en la base de datos, la plataforma ofrece un plan gratuito que tiene un límite de cinco dispositivos simultáneos y un tamaño de registro de 10 MB. Para la visualización de datos, se utilizó Tago.IO, también en su plan gratuito, que permite crear hasta cinco dispositivos y cinco tableros de visualización. Las limitaciones de estas plataformas no afectaron la disponibilidad de los datos; se pudo acceder tanto a la visualización durante las pruebas como a los registros una vez finalizadas las pruebas.

Una vez realizadas las pruebas en la maqueta, simulando un entorno controlado de temperatura y humedad, se comprobó la estabilidad y fiabilidad de la comunicación y mediciones para el sistema de monitoreo propuesto. Estas pruebas demostraron que el sistema es capaz de gestionar cargas de trabajo con cuatro variables. Para la prueba con un intervalo configurado de 10 segundos, con un promedio real medido de 11,11 segundos,

se registró que el mayor tiempo de espera es de 59 segundos. Los tiempos de espera que generan retrasos en el registro se deben a que el proyecto depende de la calidad de la red y la disponibilidad de servicio de las plataformas Clever Cloud y Tago.IO. Los tiempos de espera podrían mejorar si se utilizase plataformas de paga o el utilizar servicios locales.

La implementación del estándar OPC UA desarrollado en lado del Cliente es altamente compatible con las interfaces HTTPS, lo que permite su implementación en diversas plataformas armonizando diferentes dominios. Esto proporciona un excelente contexto para la Industria 4.0 debido a su flujo de datos simplificado y accesibilidad de contenido. Además, el proyecto incorpora una interfaz de aplicación que puede configurarse para conectar con un cliente OPC UA y registrar los datos en la nube o visualizarlos en una plataforma Web. Esta interfaz es fácilmente configurable por el usuario, no requiere conocimientos avanzados y facilita la conexión con clientes OPC UA, el registro en la nube y la visualización en la Web.

Al ofrecer un GUI para un cliente OPC UA junto con dominios desarrollados en Python, se abre la posibilidad de implementar actualizaciones a la aplicación del cliente. Por ejemplo, sería posible implementar una interfaz de visualización local dentro de la aplicación del cliente OPC UA. Si bien el servidor OPC UA se desarrolla en Python y podría ser modificable al utilizar la Raspberry Pi 4B, en dispositivos industriales el fabricante del hardware proporcionará el servidor OPC UA específico para sus dispositivos, lo que puede limitar su flexibilidad.

El proyecto actual está destinado únicamente para la monitorización, pero para trabajos futuros se podrían agregar dependencias para utilizar el estándar en sistemas de control, como el Protocolo de Tiempo de Red. OPC UA y esta dependencia pueden ser tecnologías complementarias en el entorno de automatización industrial. Mientras OPC UA proporciona la plataforma de comunicación y transferencia de datos, el Protocolo de Tiempo de Red asegura la sincronización precisa de tiempo entre los dispositivos conectados, lo que contribuye a la confiabilidad y precisión de las aplicaciones basadas en OPC UA.

Es factible el utilizar dos hilos, uno para el GUI y otro para la adquisición y envío de datos a las plataformas Web, ya que es una estrategia beneficiosa para aumentar la fluidez del programa y con esto mejorar la experiencia del usuario. Sin embargo, se debe prestar atención a los aspectos de sincronización y control para garantizar el funcionamiento sin problemas del programa.

Como última recomendación, podemos afirmar que la mejor solución es la más simple, como el uso de diccionarios para almacenar los atributos de configuración o el utilizar programas específicos para el desarrollo de interfaces gráficas de usuario.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] D. del Carmen, “Tecnolog\’\ias 4.0 empleadas en industrias del área metropolitana de Monterrey,” 2020.

[2] M. Sachon, “Los pilares de la industria 4.0,” *Revista de Antiguos Alumnos del IESE*, no. 148, 2018.

[3] A. Semle, “Protocolos IIoT para considerar,” *Aadeca Revista*, vol. 34, 2016.

[4] P. Drahos, E. Kucera, O. Haffner, and I. Klimo, “Trends in industrial communication and OPC UA,” in *2018 Cybernetics & Informatics (K&I)*, IEEE, Jan. 2018, pp. 1–5. doi: 10.1109/CYBERI.2018.8337560.

[5] M. Schleipen, S.-S. Gilani, T. Bischoff, and J. Pfrommer, “OPC UA & Industrie 4.0 - Enabling Technology with High Diversity and Variability,” *Procedia CIRP*, vol. 57, pp. 315–320, 2016, doi: https://doi.org/10.1016/j.procir.2016.11.055.

[6] M. H. Schwarz and J. Borcsok, “A survey on OPC and OPC-UA: About the standard, developments and investigations,” in *2013 XXIV International Conference on Information, Communication and Automation Technologies (ICAT)*, IEEE, Oct. 2013, pp. 1–6. doi: 10.1109/ICAT.2013.6684065.

[7] S.-M. Kim, Y. Choi, and J. Suh, “Applications of the Open-Source Hardware Arduino Platform in the Mining Industry: A Review,” *Applied Sciences*, vol. 10, no. 14, p. 5018, Jul. 2020, doi: 10.3390/app10145018.

[8] M. Hermann, T. Pentek, and B. Otto, “Design Principles for Industrie 4.0 Scenarios,” in *2016 49th Hawaii International Conference on System Sciences (HICSS)*, IEEE, Jan. 2016, pp. 3928–3937. doi: 10.1109/HICSS.2016.488.

[9] M. M. Rathore, A. Ahmad, A. Paul, and S. Rho, “Urban planning and building smart cities based on the Internet of Things using Big Data analytics,” *Computer Networks*, vol. 101, pp. 63–80, Jun. 2016, doi: 10.1016/j.comnet.2015.12.023.

[10] M. Hankel and B. Rexroth, “The reference architectural model industrie 4.0 (rami 4.0),” *Zvei*, vol. 2, no. 2, pp. 4–9, 2015.

[11] S. Vaidya, P. Ambad, and S. Bhosle, “Industry 4.0 – A Glimpse,” *Procedia Manuf*, vol. 20, pp. 233–238, 2018, doi: 10.1016/j.promfg.2018.02.034.

[12] W. Mahnke, S.-H. Leitner, and M. Damm, *OPC unified architecture*. Springer Science & Business Media, 2009.

[13] B. McIlvride, A. Thomas, and C. Real, “OPC Tunnelling–Know Your Options.” 2008.

[14] M. Ladegourdie and J. Kua, “Performance Analysis of OPC UA for Industrial Interoperability towards Industry 4.0,” *IoT*, vol. 3, no. 4, pp. 507–525, Dec. 2022, doi: 10.3390/iot3040027.

[15] H. Haskamp, M. Meyer, R. Mollmann, F. Orth, and A. W. Colombo, “Benchmarking of existing OPC UA implementations for Industrie 4.0-compliant digitalization solutions,” in *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, IEEE, Jul. 2017, pp. 589–594. doi: 10.1109/INDIN.2017.8104838.

[16] “GitHub - FreeOpcUa/opcua-asyncio: OPC UA library for python >= 3.7.” Accessed: Sep. 28, 2023. [Online]. Available: https://github.com/FreeOpcUa/opcua-asyncio

[17] D. Bailey and E. Wright, *Practical SCADA for industry*. Elsevier, 2003.

[18] J. Colomer, J. Melendez, and J. Ayza, “Introducción a la monitorización y supervisión experta de procesos. Métodos y herramientas.” España, 2000.

[19] R. Roshani, *Examining the Impact of Deep Learning and IoT on Multi-Industry Applications*. IGI Global, 2021. doi: 10.4018/978-1-7998-7511-6.

[20] “Ubidots,” Ubidots - Home. Accessed: Dec. 11, 2022. [Online]. Available: https://ubidots.com/

[21] A. R. Penin, *Sistemas Scada*. Marcombo, 2011.

[22] A. S. Saabith, M. M. M. Fareez, and T. Vinothraj, “Python current trend applications-an overview,” *International Journal of Advance Engineering and Research Development*, vol. 6, no. 10, 2019.

[23] S. An-dong and Z. Fang, “Research on Open Source Solutions of Data Collection for Industrial Internet of Things,” in *2021 7th International Symposium on Mechatronics and Industrial Informatics (ISMII)*, IEEE, Jan. 2021, pp. 180–183. doi: 10.1109/ISMII52409.2021.00045.

[24] S. J. Oks, S. Zöllner, M. Jalowski, J. Fuchs, and K. M. Möslein, “Embedded vision device integration via OPC UA: Design and evaluation of a neural network-based monitoring system for Industry 4.0,” *Procedia CIRP*, vol. 100, pp. 43–48, 2021, doi: 10.1016/j.procir.2021.05.007.

[25] “Emerson ES.” Accessed: Mar. 20, 2023. [Online]. Available: https://www.emerson.com/es-es/automation/measurement-instrumentation/pressure-measurement/about-multivariable-measurement

[26] “Adafruit BME280.” Accessed: Mar. 20, 2023. [Online]. Available: https://learn.adafruit.com/adafruit-bme280-humidity-barometric-pressure-temperature-sensor-breakout/downloads

[27] H. D. Kambezidis, “The Solar Resource,” in *Comprehensive Renewable Energy*, Elsevier, 2012, pp. 27–84. doi: 10.1016/B978-0-08-087872-0.00302-4.

6. ANEXOS

ANEXO A. Código de la programación del servidor OPC UA.

ANEXO B. Código de programación de conexión al servidor OPC UA, MySQL Cloud y Tago.IO.

ANEXO C. Código de programación del GUI.

ANEXO D. Código de programación del cliente OPC UA y plataformas.

ANEXO E. Manual de operación.