UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE CUENCA

CARRERA DE ELECTRÓNICA Y AUTOMATIZACIÓN

SISTEMA DE COMUNICACIÓN INDUSTRIAL BASADO EN OPC UA PARA LA OPERACIÓN Y MONITOREO MULTI ESTACIÓN DE PROCESOS INDUSTRIALES

Trabajo de titulación previo a la obtención del título de Ingeniero en Electrónica

AUTORES: JOSÉ DAVID ABRIL VERA
RENATA FRANCISCA GUILLEN SARMIENTO

TUTOR: ING. JULIO CESAR ZAMBRANO ABAD, PhD.

Cuenca - Ecuador
2023

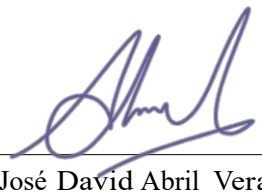
CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, José David Abril Vera con documento de identificación N° 0106132574 y Renata Francisca Guillen Sarmiento con documento de identificación N° 0150206118; manifestamos que:

Somos los autores y responsables del presente trabajo; y, autorizamos a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Cuenca, 28 de julio del 2023

Atentamente,



José David Abril Vera

0106132574

Renata Francisca Guillen Sarmiento

0150206118

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Nosotros, José David Abril Vera con documento de identificación N° 0106132574 y Renata Francisca Guillen Sarmiento con documento de identificación N° 0150206118, expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del Proyecto técnico: "Sistema de comunicación industrial basado en OPC UA para la operación y monitoreo multi estación de procesos industriales" el cual ha sido desarrollado para optar por el título de: Ingeniero en Electrónica, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribimos este documento en el momento que hacemos la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Cuenca, 28 de julio del 2023

Atentamente,

José David Abril Vera

01061132574

Renata Francisca Guillen Sarmiento

0150206118

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Julio Cesar Zambrano Abad con documento de identificación N° 0301489696, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: SISTEMA DE COMUNICACIÓN INDUSTRIAL BASADO EN OPC UA PARA LA OPERACIÓN Y MONITOREO MULTI ESTACIÓN DE PROCESOS INDUSTRIALES, realizado por José David Abril Vera con documento de identificación N° 0106132574 y Renata Francisca Guillen Sarmiento con documento de identificación N° 0150206118, obteniendo como resultado final el trabajo de titulación bajo la opción Proyecto técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Cuenca, 28 de julio del 2023

Atentamente,

Julio Cesar Zambrano Abad
0301489696

AGRADECIMIENTOS

Dedicatoria de José David Abril Vera

En este pequeño espacio, quiero expresar mi sincero agradecimiento por cada uno de ustedes. Mi familia, por ser mi sostén incondicional, por su amor eterno que me da fuerzas para enfrentar cualquier desafío. Mis amigos, por llenar mi vida de risas, aventuras y complicidad, por estar siempre ahí, sin importar la distancia. A mis queridos docentes, por su dedicación y sabiduría, por guiarme en mi aprendizaje y por inspirarme a superarme cada día. Cada uno de ustedes ha dejado una huella imborrable en mi corazón. Gracias por ser parte de mi vida.

Dedicatoria de Renata Francisca Guillen Sarmiento

Agradezco a mi madre y a mi padre. Este logro es gracias a su amor y apoyo constante. Vuestra fe en mí me impulsó a seguir adelante cuando perdí la esperanza. Dedico este éxito a ustedes, mis pilares inquebrantables. Los amo con todo mi corazón. Agradezco a mi director de tesis, Ing. Julio Zambrano, gracias por ser mi guía y mentor en esta tesis. Su apoyo y sabiduría fueron fundamentales para alcanzar este logro. Aprecio enormemente su dedicación y confianza en mí.

DEDICATORIAS

DEDICATORIA

Dedicatoria de José David Abril Vera

En cada latido de mi corazón, resuenan los lazos profundos que nos unen. Son mi refugio, mi apoyo inquebrantable y la fuente inagotable de amor y cariño. Cada día a su lado es un regalo que atesoro con gratitud. Juntos hemos compartido risas, lágrimas y sueños, forjando recuerdos imborrables. Su presencia ha llenado mi vida de significado y alegría. En los momentos de dificultad, su fuerza y aliento me han impulsado a seguir adelante. Gracias por ser mi familia, mi roca en la tormenta y mi hogar en este vasto mundo.

Dedicatoria de Renata Francisca Guillen Sarmiento

Esta tesis va dedicada a mis queridos amigos de la universidad. Gracias por ser mi apoyo incondicional y enseñarme lo que el colegio no pudo. Vuestra amistad y conocimientos compartidos han sido invaluables en este camino académico. Cada risa, consejo y momento juntos ha sido un regalo en mi vida.

Índice general

Índice de figuras

Índice de tablas

Índice de Algoritmos

Resumen

El presente estudio tiene como objetivo principal desarrollar e implementar un sistema de comunicación industrial altamente confiable y eficiente, basado en la tecnología OPC UA. El enfoque central de esta investigación recae en la operación y monitoreo de procesos industriales que involucran múltiples estaciones.

Para alcanzar este propósito, se llevó a cabo un análisis detallado de la tecnología OPC UA, haciendo hincapié en los autómatas S7-1500 de SIEMENS. Además, se examinaron las estaciones de monitoreo basadas en WINCC RT. La investigación se enfocó en explorar cómo esta tecnología se puede integrar con el OPC UA para lograr una operación y monitoreo eficiente en múltiples estaciones.

Para validar la viabilidad y efectividad del sistema propuesto, se realizó una implementación práctica al controlar y monitorear un motor industrial. Durante esta fase, se generaron y recopilaron las variables de interés que posteriormente serán monitorizadas por las estaciones HMI. La implementación del sistema HMI utilizando WINCC RT permitió la visualización y control de las estaciones involucradas en el proceso industrial, garantizando una interfaz amigable y accesible para los operadores y personal de monitoreo.

La culminación del trabajo consistió en la integración de las estaciones HMI con el controlador industrial mediante la tecnología de comunicación OPC UA. Este paso fue crucial para asegurar una interacción fluida y segura entre los diferentes componentes del sistema, habilitando la transferencia de datos y comandos en tiempo real.

Palabras clave: OPC UA; WINCC RT; sistema de comunicación; S7-1500 de SIEMENS; HMI; monitoreo multi estación; proceso industrial.

Abstract

The main objective of this study is to develop and implement a highly reliable and efficient industrial communication system based on OPC UA technology. The main focus of this research is on the operation and monitoring of industrial processes involving multiple stations.

To achieve this purpose, a detailed analysis of OPC UA technology was carried out, with emphasis on SIEMENS S7-1500 PLCs. In addition, WINCC RT-based monitoring stations were examined. The research focused on exploring how this technology can be integrated with OPC UA to achieve efficient multi-station operation and monitoring.

To validate the feasibility and effectiveness of the proposed system, practical implementation was carried out by controlling and monitoring an industrial engine. During this phase, the variables of interest that would later be monitored by the HMI stations were generated and collected. The implementation of the HMI system using WINCC RT allowed the visualization and control of the stations involved in the industrial process, ensuring a friendly and accessible interface for operators and monitoring personnel.

The culmination of the work consisted of the integration of the HMI stations with the industrial controller using OPC UA communication technology. This step was crucial to ensure a smooth and secure interaction between the different components of the system, enabling the transfer of data and commands in real-time.

Keywords: OPC UA; WINCC RT; communication system; SIEMENS S7-1500; HMI; multi-station monitoring; industrial process.

Antecedentes

La importancia de la supervisión en los sistemas industriales con PLC (Programmable Logic Controller) radica en la capacidad de monitorizar y controlar con eficacia y precisión. A nivel industrial, esto se traduce en un aumento de la productividad, una mejora del control de calidad y una reducción del tiempo de inactividad por paras debido a mantenimientos preventivos o correctivos. Mediante el uso del PLC, los operarios pueden recopilar datos en tiempo real de varias estaciones o procesos y utilizarlos para tomar decisiones fundamentadas sobre la optimización y el control. Además, con la ayuda de protocolos de comunicación industrial como OPC UA (Open Platform Communications Unified Architecture), los PLC pueden intercambiar datos sin problemas con otros dispositivos y sistemas, lo que facilita la mejora de la productividad y la facilidad de uso [1].

En países desarrollados, OPC UA se utiliza ampliamente en varios sectores y para diferentes aplicaciones, como el intercambio de datos y la interoperabilidad entre varios dispositivos, máquinas y subsistemas dentro de un proceso industrial. Su aplicabilidad abarca desde la comunicación a nivel de dispositivo hasta la conectividad a nivel de TI empresarial, lo que la hace extremadamente escalable y versátil. El marco OPC UA también ofrece una serie de funciones, como el cifrado de datos y la autenticación segura, que son esenciales para garantizar un intercambio de datos seguro y fiable a través de las redes industriales. Como tal, OPC UA se ha convertido en un componente clave de la Industria 4.0, donde la interoperabilidad y el intercambio de datos entre diferentes sistemas industriales son cruciales.[2]

OPC UA es la evolución del OPC (Open Protocol Communication) clásico. Esta nueva tecnología de comunicación industrial se caracteriza por ser abierta y orientada

a servicios sin dependencia al hardware y el sistema operativo. De esta manera, resulta práctico conectar varias estaciones industriales sin tener que preocuparse de la tecnología subyacente. Bajo una estructura cliente/servidor, los dispositivos industriales pueden integrarse e intercambiar información.

Actualmente, existe en el mercado diversas marcas de PLCs que brindan opciones de comunicación a través de OPC UA. Sin embargo, se debe tener en cuenta, que este sistema de comunicación industrial es relativamente nuevo y que únicamente se puede encontrar estas opciones de comunicación en autómatas nuevos. En el caso de la marca SIEMENS, esta tecnología de comunicación está disponible para algunas versiones de PLCs SIMATIC S7-1200 y S7-1500. Todas las funcionalidades de comunicación pueden ser configuradas a través de la plataforma informática de TIA PORTAL. Gracias a esto, hoy en día se pueden realizar redes de comunicación entre PLCs utilizando esta tecnología. Sin embargo, este no es el nicho principal de OPC UA. La idea principal de OPC UA es interconectar la planta con los departamentos de TI. En otras palabras, OPC UA surge como una herramienta para solventar los problemas de conectividad entre PLCs y estaciones de monitoreo y operación basadas en PC. [3]

Dentro de la marca SIEMENS, las estaciones de monitoreo y operación pueden ser insertadas en las redes industriales gracias a WinCC, la cual es una herramienta informática industrial de nueva generación que puede facilitar la integración inteligente de estaciones y dispositivos industriales. Sin embargo, como se mencionó anteriormente, no es necesario que los equipos de red sean del mismo proveedor. En este sentido, se pueden utilizar insertar estaciones de monitoreo de otros fabricantes como, por ejemplo, Intouch. Incluso, hoy en día se pueden encontrar herramientas de desarrollo de código abierto para crear sistemas de operación y monitoreo sin depender de un fabricante.

El presente proyecto se cimienta en la necesidad de innovar los procesos industriales de nuestro medio, utilizando tecnologías de punta y que brinden soluciones a los actuales problemas de integración que atraviesa la industria. Bajo este sentido, se propone la integración de varias estaciones de monitoreo y operación de un proceso industrial a través de OPC UA. En este contexto, se debe manifestar que

un proceso industrial que es controlado y monitorizado desde varios puntos distintos puede brindar diversos beneficios. En primer lugar, permite una mayor flexibilidad y redundancia en la supervisión y control del proceso. Si uno de los puntos de control falla, el otro puede continuar operando para mantener la continuidad del proceso. Además, controlar y monitorear desde varios puntos puede mejorar la eficiencia del proceso al permitir la optimización en tiempo real desde ambas ubicaciones. Esto también puede permitir la detección temprana de cualquier problema o falla en el proceso, lo que puede prevenir posibles interrupciones o pérdidas económicas.

Justificación

Actualmente, en nuestro país, la tecnología vinculada a la Industria 4.0 resulta relativamente nueva, por lo tanto, existe una escasa explotación de estos recursos en pro de la innovación. Todo este desconocimiento se puede ir mitigando desde la academia generando proyectos que aporten a la innovación de la industria local. Sobre esta base se plantea el presente proyecto, mediante el cual se implementará un sistema de comunicación basado en OPC UA para el monitoreo multi estación de un proceso industrial.

El enfoque de este proyecto es mejorar las capacidades del laboratorio de redes industriales de la Universidad Politécnica Salesiana. El laboratorio ya está equipado con lo necesario para desarrollar un sistema de comunicaciones. A través de la investigación que se está realizando en el marco de este proyecto, se busca establecer las bases para la aplicación de la tecnología OPC UA en el ámbito académico y compartir conocimientos que puedan beneficiar al sector industrial.

Objetivos

Objetivo General

- Implementar un sistema de comunicación industrial basado en OPC UA para la operación y monitoreo multi estación de procesos industriales.

Objetivos específicos:

- Realizar un análisis de la tecnología OPC UA haciendo énfasis en los modos de configuración en los autómatas S7-1500 de SIEMENS y en estaciones de monitoreo basadas en WINCC RT.
- Realizar el control de un proceso industrial de laboratorio para genera las variables de interés que serán monitorizadas por las estaciones HMI.
- Implementar un sistema HMI utilizando WINCC RT para la operación y monitoreo multi estación del proceso industrial.
- Realizar la integración de las estaciones HMI y el controlador industrial mediante la tecnología de comunicación OPC UA.

Introducción

Las redes de comunicación eficientes son fundamentales en el panorama industrial actual para el funcionamiento y la supervisión sin fisuras de los procesos industriales. El objetivo general de este proyecto es construir un sistema de comunicación industrial basado en OPC UA para el funcionamiento de varias estaciones y la supervisión de procesos. El estándar de intercambio de datos OPC UA se utiliza ampliamente en la automatización y las comunicaciones industriales [4].

OPC UA es un protocolo de comunicación que desempeña un papel crucial en el sector industrial, especialmente en el contexto de la Industria 4.0 y el IoT (Internet de las cosas). Su relevancia radica en diversos puntos clave dentro de la industria: primero, proporciona un marco estandarizado que permite definir e intercambiar datos entre diferentes plataformas y proveedores, garantizando la interoperabilidad. Esta característica permite una integración fluida y sin la necesidad de un extenso desarrollo de software, lo que resulta fundamental para asegurar el funcionamiento eficaz de los procesos industriales. [5]

En este proyecto se plante la supervisión de un proceso industrial que consiste de un motor trifásico de inducción comandado por un accionamiento SINAMICS G120. El accionamiento opera con una unidad de control que posee comunicaciones Profinet. De esta manera, un PLC S7-1500 intercambiará información con el accionamiento a través de un sistema de comunicación basado en Ethernet. Siguiendo con la estructura de un sistema de comunicación basado en OPC UA, se plantea una configuración cliente/servidor. Donde el servidor será el controlador lógico programable, mientras que los clientes serán estaciones HMI desarrolladas en WinCC. Se plantea utilizar tres estaciones con las mismas prestaciones para que las

tres puedan acceder a la información del proceso.

Capítulo 1

OPC UA

En este capítulo, se presenta una introducción al protocolo de comunicación OPC UA. Como parte del estudio, también se examina el acceso de los autómatas SIMATIC S7-1500 a OPC UA, considerando la versión requerida de firmware para habilitar servidores y clientes. Finalmente, se presenta una configuración general para implementar servidores OPC UA en el controlador SIMATIC S7-1500, junto con la comunicación básica con WINCC RT. Este análisis busca sentar las bases para una interacción eficiente entre dispositivos industriales y sistemas de supervisión, fomentando la optimización de soluciones en automatización.

1.1. ¿Qué es OPC UA?

(OPC) es un estándar de comunicación abierto que proporciona una solución confiable para la integración de sistemas de automatización y control en entornos industriales. (OPC) proviene de las siglas OLE para Control de Procesos. El acrónimo ”OLE” se refiere a ”Object Linking and Embedding”, una tecnología desarrollada por Microsoft para facilitar la comunicación y el intercambio de datos entre aplicaciones en el entorno Windows. En otras palabras, se trata de un estándar para transferir datos entre equipos de proceso y aplicaciones de PC. Estas interfaces se basan en el Modelo de Objetos Componentes (COM) de Microsoft, lo que permite su integración en los sistemas operativos de la compañía. Utilizando COM o DCOM (Modelo de Objetos Componentes Distribuidos), OPC facilita la comunicación entre

procesos y el intercambio de información entre aplicaciones, incluso más allá de los límites de los ordenadores. Los clientes OPC (basados en COM) pueden intercambiar información con los servidores OPC (basados en COM) utilizando componentes del sistema operativo de Microsoft. Sin embargo, es importante destacar que tanto los servidores como los clientes OPC solo son compatibles con entornos Windows o redes locales debido al uso de COM o DCOM. También se requieren técnicas adicionales, como túneles, para superar cortafuegos o evitar la complejidad de la configuración de DCOM en la conexión de red. Además, solo las aplicaciones C++ pueden acceder directamente a la interfaz OPC, mientras que las aplicaciones .NET o Java requieren una capa adicional. Estas restricciones aumentan la complejidad y el esfuerzo necesario para configurar y utilizar OPC [6].

La Fundación OPC ha establecido una nueva plataforma bajo el nombre de Arquitectura Unificada OPC durante los últimos siete años, con el fin de abordar estas restricciones prácticas y las nuevas normas. Como resultado, los componentes y sistemas pueden compartir información de forma coherente. OPC UA es una norma accesible como IEC 62541 y, como tal, sirve de base para otras normas internacionales. Estas son las funcionalidades que proporciona OPC UA:

- Combina toda la información y características de OPC, en una interfaz universal que no es dependiente del sistema operativo.
- Utiliza protocolos abiertos y de plataforma neutra para la comunicación en red y entre procesos.
- Permite una comunicación y acceso a Internet a través de cortafuegos.
- Brinda métodos de control de acceso y seguridad integrados a nivel de protocolo y aplicación.
- Posee una amplia gama de posibilidades de asignación de modelos orientados a objetos. Los objetos pueden incluir etiquetas, métodos y propiedades de activación de eventos.
- Ofrece un sistema de tipos extensible para tipos de datos complejos y tipos de objetos.

- Brinda capacidad para escalar desde minúsculos sistemas integrados a aplicaciones empresariales y desde sencillos espacios de direcciones a intrincados modelos orientados a objetos.

1.2. OPC UA dentro de los controladores SIMATIC S7-1500

El estándar OPC UA es ampliamente utilizado en la industria para facilitar la comunicación y el intercambio de datos entre sistemas de automatización. Los controladores SIMATIC S7-1500 de Siemens cuentan con un servidor OPC UA y, a partir de la versión de firmware 2.6, también incorporan un cliente OPC UA [7]. El servidor OPC UA de la CPU S7-1500 permite que otros sistemas, denominados clientes, accedan a la información disponible. Estos clientes pueden leer y escribir datos en el servidor, así como llamar a métodos implementados en el mismo. Esto brinda la posibilidad de acceder en línea a los datos del servidor, así como a información de rendimiento y diagnóstico a través de un cliente OPC UA.

Una de las funciones clave del cliente OPC UA es la capacidad de navegación, conocida como ”Browsen”. Esto permite explorar la estructura de los datos disponibles en el servidor y obtener información sobre las variables y métodos disponibles para su acceso. En la Figura 1.1 se puede observar el uso de la función ”Browsen”. Es importante destacar que un sistema basado en los controladores SIMATIC S7-1500 puede actuar simultáneamente como cliente y servidor OPC UA. Esto significa que puede proporcionar datos y métodos a otros sistemas como servidor, al mismo tiempo que actúa como cliente para acceder a datos de otros servidores OPC UA.

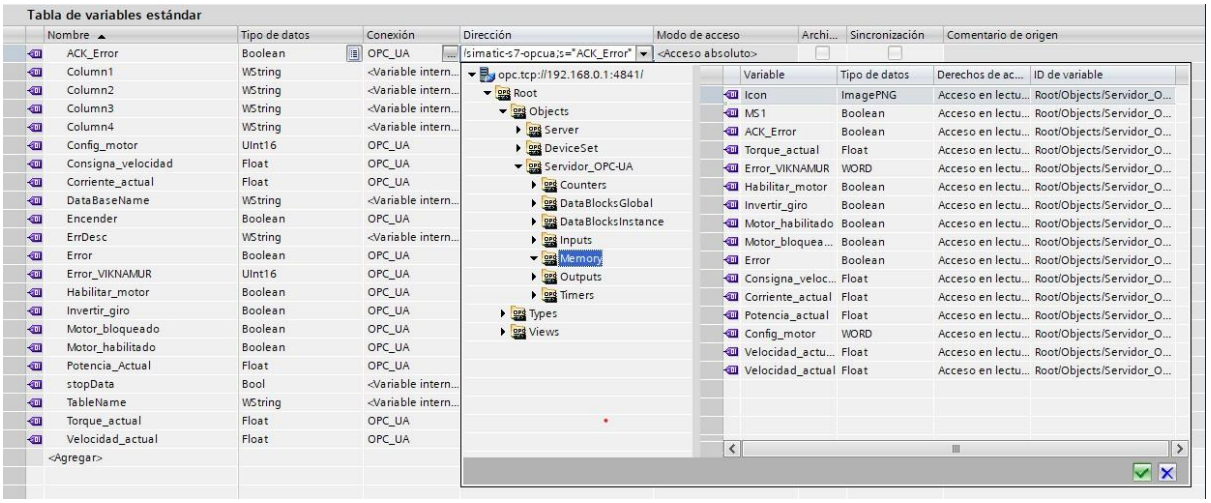


Figura 1.1: Función de Navegación Browsen en un Cliente OPC UA.

La configuración del servidor OPC UA en la CPU S7-1500 implica establecer los datos y métodos que estarán disponibles para los clientes. Por otro lado, la CPU S7-1500 también puede desempeñar el rol de cliente OPC UA, lo que permite leer datos de otros servidores OPC UA, escribir datos en ellos y llamar a sus métodos. Siemens proporciona herramientas como el STEP 7 (TIA Portal) para facilitar la creación de programas de usuario que actúen como clientes OPC UA. Estas herramientas incluyen un editor para interfaces de cliente y una parametrización para establecer las conexiones OPC UA necesarias.

1.3. SIMATIC S7-1500 como servidor OPC UA

El servidor OPC UA en un controlador SIMATIC S7-1500 requiere el uso de la herramienta TiaPortal 15.1, la cual proporciona una interfaz intuitiva y funcional. Durante el proceso de configuración, se debe seguir algunos pasos para garantizar el correcto funcionamiento del servidor. Entre otras cosas se debe establecer la configuración de parámetros de comunicación y la asignación de variables.

Antes de iniciar la configuración del servidor en el controlador S7-1500, es necesario llevar a cabo dos pasos fundamentales. En primer lugar, se debe crear un nuevo proyecto en TIA Portal, utilizando la versión 15.1 o una versión más reciente. En segundo lugar, se debe configurar un controlador SIMATIC S7-1500 que cuente con un firmware de versión 2.6 o superior. Estos requisitos son fundamentales para garantizar una configuración adecuada y compatible con el servidor en cuestión.

Por defecto, el servidor OPC UA del controlador S7-1500 está desactivado. A continuación se indican las instrucciones para activar el servidor:

- Se accede a las **Propiedades** de la CPU S7-1500 configurada dentro de TIA Portal. Luego, en la ventana del inspector se navega hasta **Licencias runtime** >**OPC UA**, donde se selecciona la licencia necesaria. Este procedimiento se puede observar en la Figura 1.2.

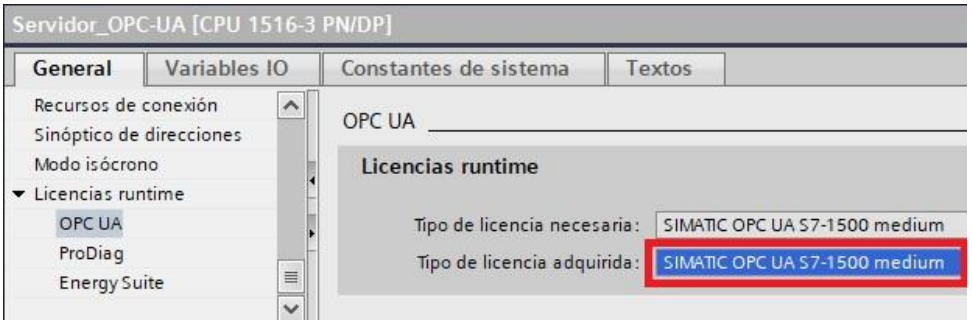


Figura 1.2: Selección de licencia para OPC UA.

- Luego, en la sección **OPC UA** >**Servidor** >**General**, en la ventana de **Accesibilidad del servidor** se debe marcar la opción **Activar servidor OPC UA** para que el servidor quede activado. Este procedimiento se puede observar en la Figura 1.3. Además, se debe tener en cuenta las **Direcciones del servidor**. Estas direcciones pueden verificarse en el HMI de la CPU. Para este caso, la interfaz X1 tiene la dirección 192.168.0.1, mientras que, la interfaz X2 tiene la dirección 192.168.1.1. En el caso de que se utilice la interfaz de comunicación X1 del PLC, la dirección de servidor sería **opc.tcp://192.168.0.1:4841**. En cambio, si se utiliza la interfaz X2 la dirección del servidor sería **opc.tcp://192.168.1.1:4841**.

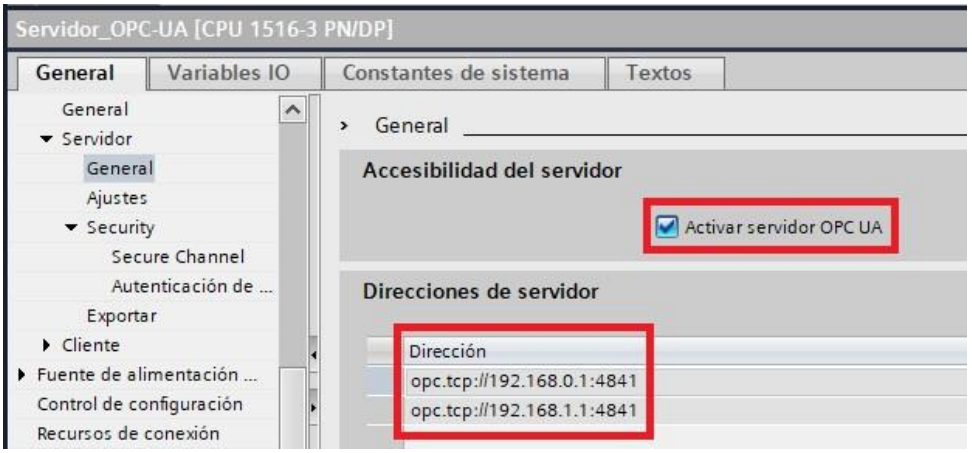


Figura 1.3: Activación del servidor OPC UA.

- En la ventana de inspección, se selecciona **OPC UA >Servidor >Ajustes** y se introduce el número de puerto que se desea utilizar para el servidor OPC UA de la CPU. Se asigna también al servidor OPC UA un **Mínimo intervalo de muestreo** y un **Mínimo intervalo de envió**. Este procedimiento se puede observar en la Figura 1.4.

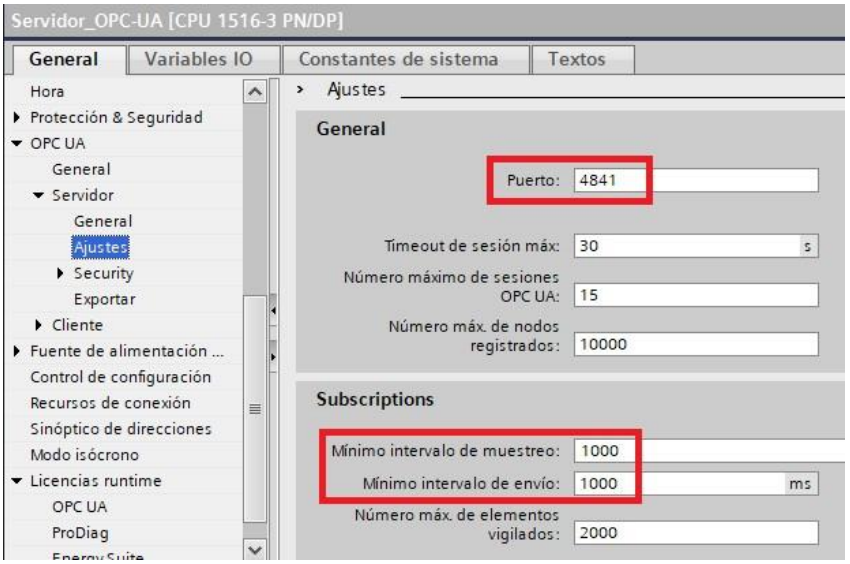


Figura 1.4: Ajustes del Servidor OPC UA.

Por cuestiones de seguridad y restricciones para los usuarios en el laboratorio que se realizó la implementación, se realiza la siguiente configuración.

- En la ventana de inspección, se selecciona **OPC UA >Scuruty >Secure Channel** y en **Security Policys disponibles en el servidor** se asigna **Ninguna seguridad**.

En la Figura se indica este paso 1.5.

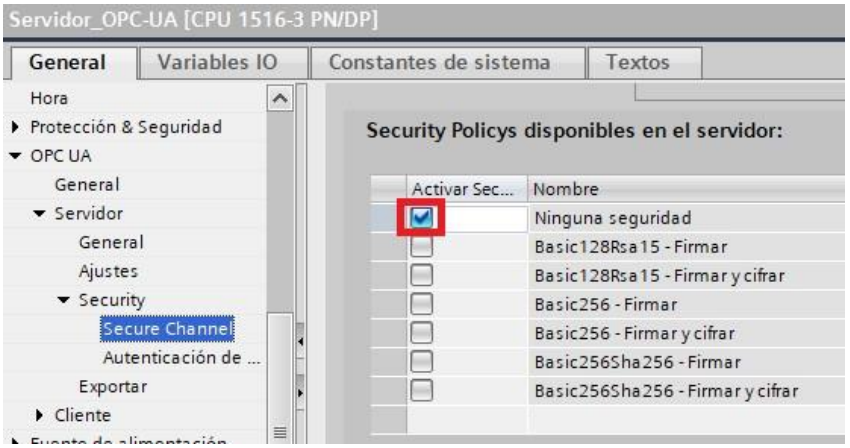


Figura 1.5: Función de Navegación Browsen en el Cliente.

Realizados los ajustes que se indica anteriormente, se realiza la carga del programa en el autómata, para que los servicios de OPC UA sean levantados.

- En la ventana de **Árbol del proyecto**, se selecciona el autómata cuyo nombre dado es **Servidor_OPC-UA** y se compila **Software** y **Hadware**, tal como se indica en la Figura 1.6.

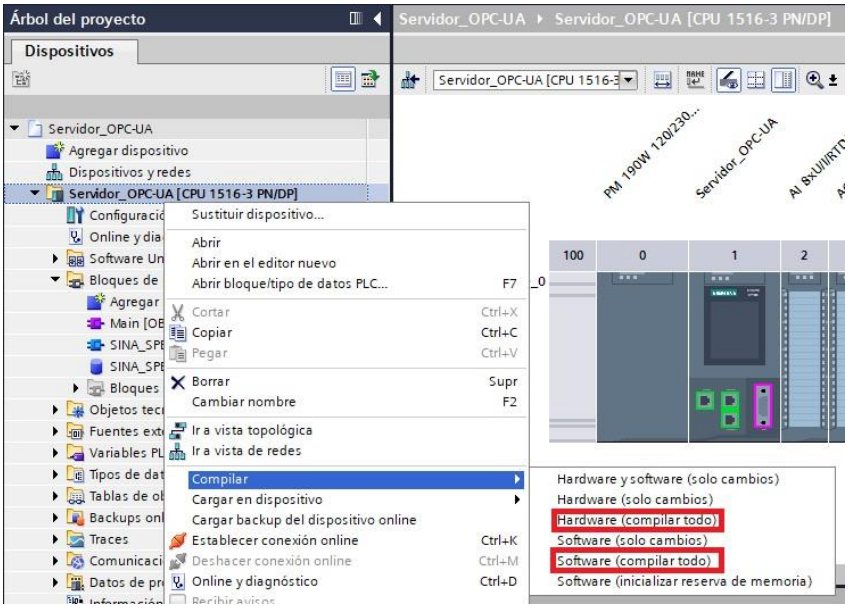


Figura 1.6: Compilación del **Software** y **Hadware** para el levantamiento del Servidor OPC UA en el autómata.

- Nuevamente en la ventana de **Árbol del proyecto**, se selecciona el autómata cuyo nombre dado es **Servidor_OPC-UA** y se realiza la carga en dispositivo **Software(cargar todo)** y **Hadware(cargar todo)** tal como se indica en la Figura 1.7.

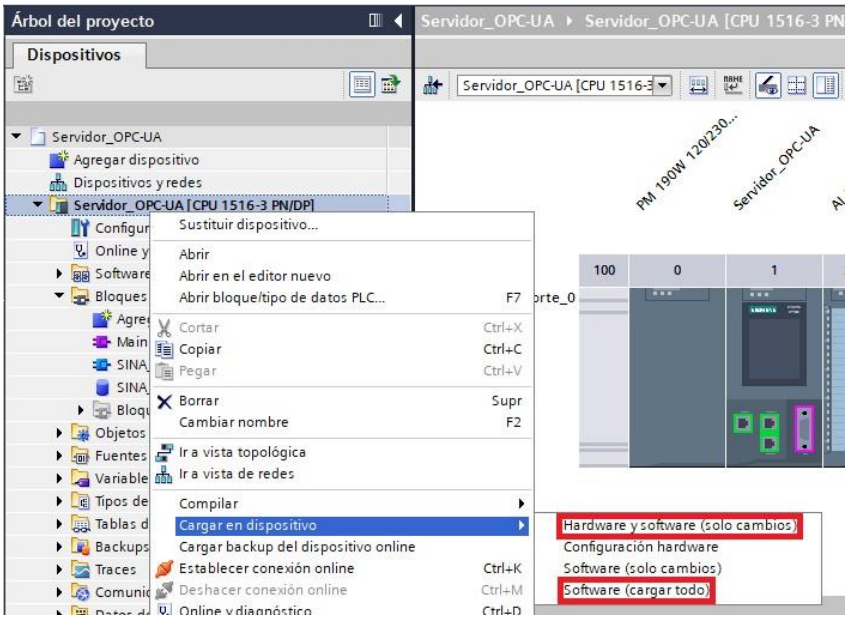


Figura 1.7: Carga en el dispositivo la configuración realizada en **Software** y **Hadware**.

Finalmente, el autómata SIMATIC S7-1500 está configurado y en operación como **Servidor_OPC-UA**. Con esto realizado, cualquier variable que se genere en el servidor puede ser consumida o controlada a través de un cliente.

1.4. WinCC RT Professional como cliente OPC UA

En la sección 1.3 se presentó la implementación de un servidor OPC UA en el autómata SIMATIC S7-1500. Ahora, en este apartado se presenta la configuración de un cliente, que se trata de un sistema PC a través de una aplicación HMI desarrollado con WinCC RT Professional. WinCC RT Professional es un sistema de software utilizado para la supervisión y el control de procesos industriales. Combina componentes de software y hardware, incluidas unidades terminales remotas y PLC para supervisar y controlar dispositivos [8].

Para establecer una comunicación cliente-servidor del tipo OPC UA se debe crear un nuevo proyecto en TIA Portal, utilizando la versión 15.1 o una versión más reciente. Luego se debe seguir los siguientes pasos para lograr establecer una correcta comunicación cliente-servidor.

- En la ventana de **Árbol del proyecto**, se selecciona la opción de **Agregar dispositivo** tal como se indica en la Figura 1.8.

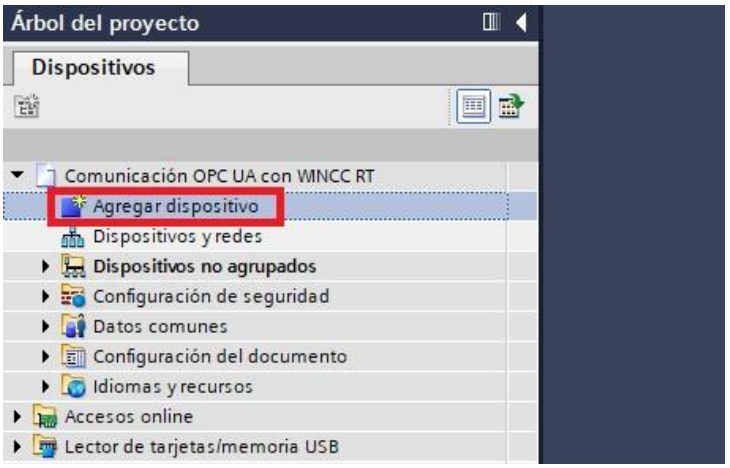


Figura 1.8: Cuadro de diálogo en TIA Portal para agregar un dispositivo

- En la nueva ventana de **Agregar dispositivo** se selecciona **Sistemas PC >SIMATIC HMI Application >WinCC RT Professional**, tal como se observar en la Figura 1.9.

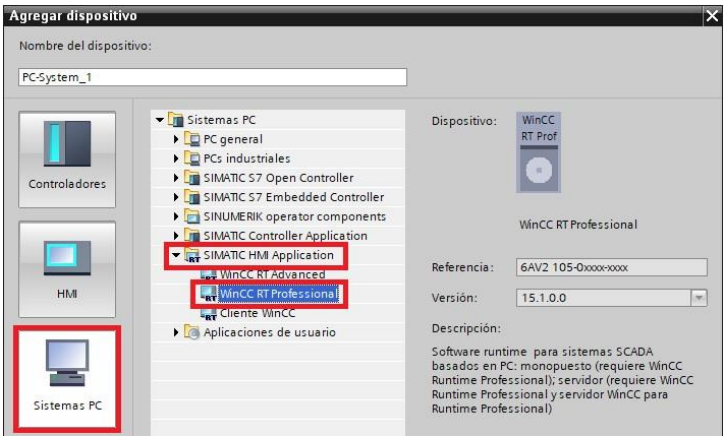


Figura 1.9: Agregar nuevo dispositivo **WinCC RT Professional**.

- En la ventana de **Catálogo de hardware** se selecciona **Módulos de comunicación** >**PROFINET/Ethernet** >**IE general**. Este procedimiento se puede observar en la Figura 1.10.

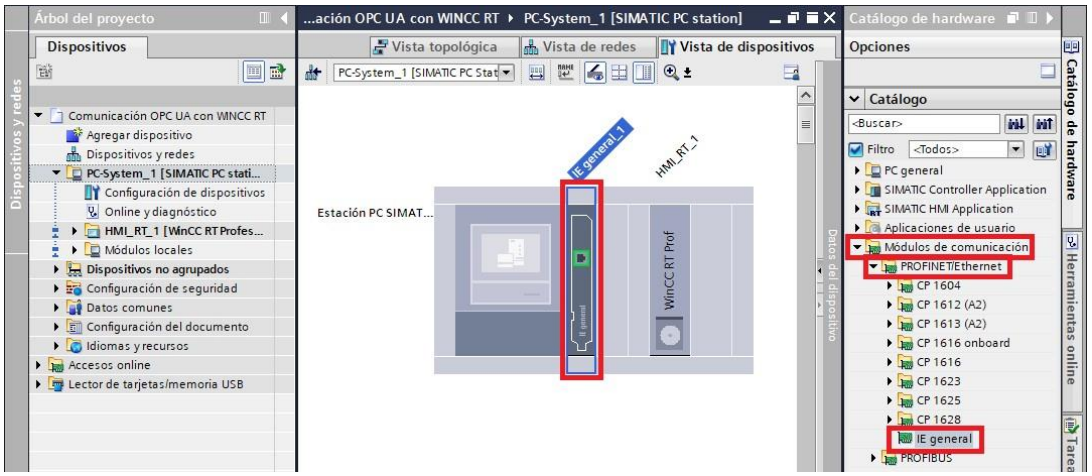


Figura 1.10: Módulo de comunicaciones PROFINET/Ethernet.

- Para configurar el módulo de comunicaciones **IE general_1 [IE General]**, se accede a sus propiedades y, en la ventana del inspector, se navega hasta la sección **Direcciones Ethernet**. Allí se introduce la **Dirección IP:** 192.168.0.200 y la **Máscara de subred:** 255.255.255.0. La Figura 1.11 proporciona una representación visual de este procedimiento. Es importante destacar que la dirección IP puede variar según el cliente y se debe asegurar que la dirección a utilizar no esté ya en uso. Evite relacionar con la frase proporcionada anteriormente.

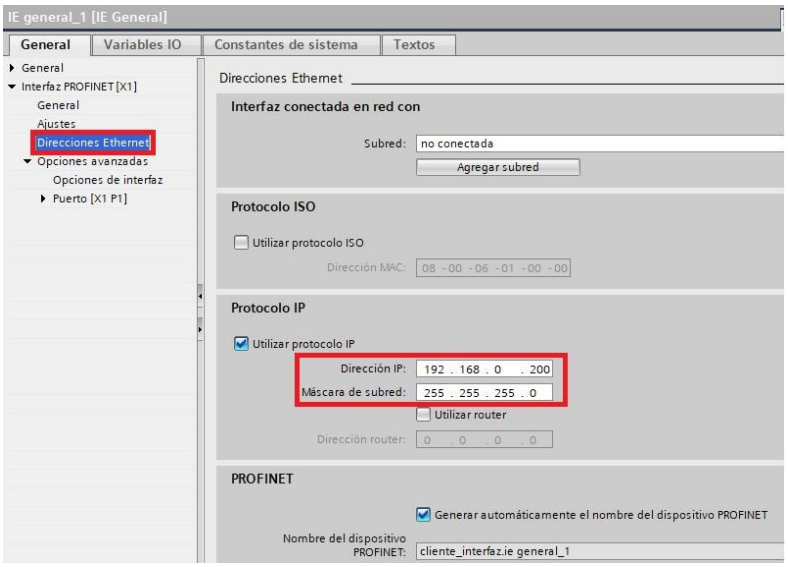


Figura 1.11: Configuración dirección IP.

- En la ventana de **Árbol del proyecto**, se selecciona **PC-System_1 [SIMATIC PC station] >HMI_RT_1 [WinCC RT Professional] >Conexiones**, donde se puede agregar diferentes conexiones a través la opción de **Agregar**, tal como se indica en la Figura 1.12.

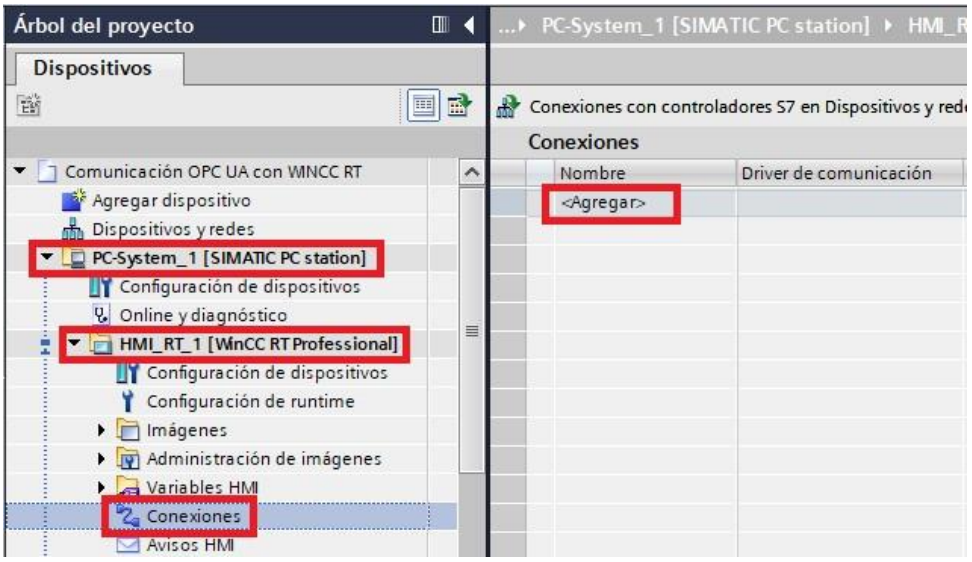


Figura 1.12: Conexiones WinCC RT Professional.

- Se agrega una conexión con el nombre de **OPC_UA** y se selecciona el **Driver de comunicación** con la opción de **OPC UA** como se indica en la Figura 1.13.
- Posteriormente, se despliega la ventana de **Parámetro**. Está ventana permite

seleccionar una **Interfaz** de tipo **OPC** y como un parámetro clave permite ingresar el **Servicio de búsqueda de servidor UA URL** el cual consiste en las **Direcciones de servidor**. En el caso que se use la interfaz X1 la dirección de servidor sería **opc.tcp://192.168.0.1:4841**, en cambio sí se usa la interfaz X2 la dirección de servidor sería **opc.tcp://192.168.1.1:4841** como se indicó en la sección 1.3, en la Figura 1.3.

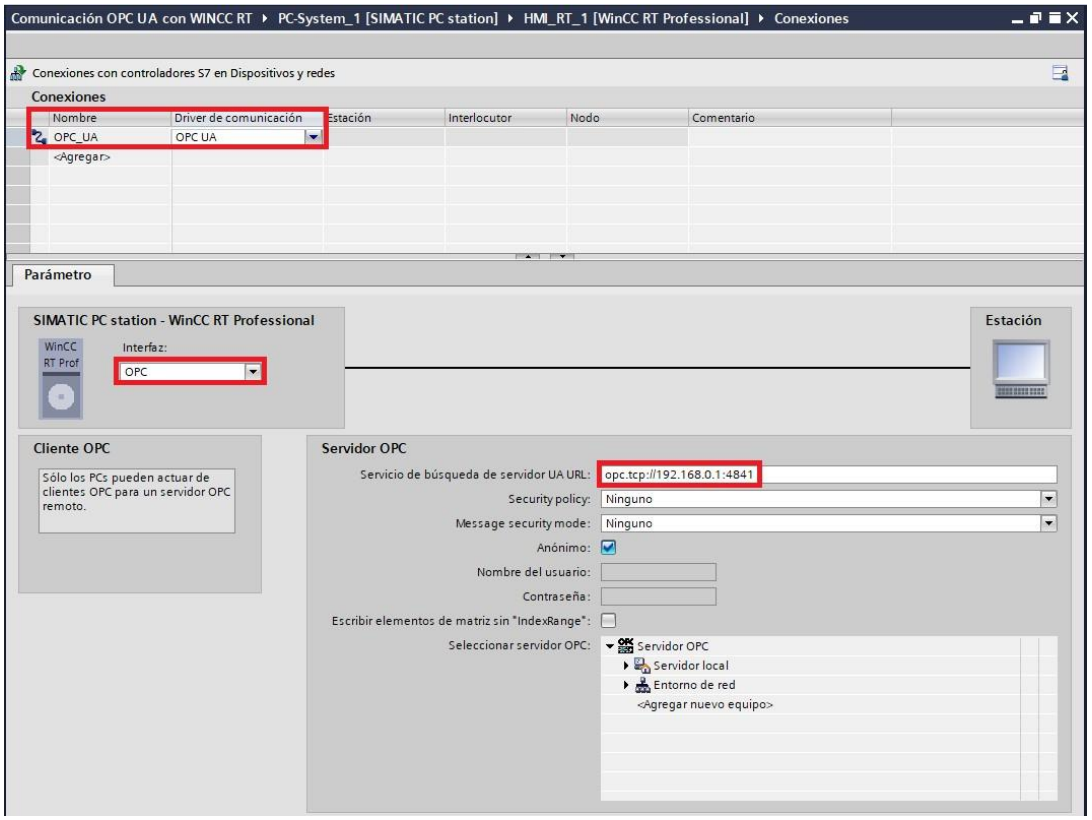


Figura 1.13: Configuración de conexión OPC UA en WinCC RT Professional.

Finalmente, el cliente en WinCC RT Professional esta configurado como cliente OPC UA, de esta manera se puede consumir o controlar las variables generadas con el servidor OPC UA implementado en el autómata SIMATIC S7-1500.

1.5. Comunicación cliente-servidor OPC UA

Esta sección tiene la finalidad de comprobar la comunicación del servidor en el autómata SIMATIC S7-1500 con el cliente HMI desarrollado con WinCC RT

Professional y a su vez implementado en un sistema PC. La topología para este ejemplo se puede observar en la Figura 1.14.

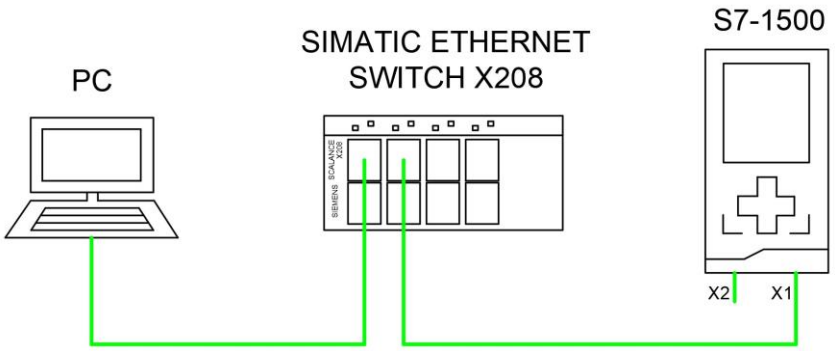


Figura 1.14: Topología de ejemplo de comunicación cliente-servidor OPC UA.

1.5.1. Ejemplo de comunicación cliente-servidor OPC UA

Realizado el conexionado de la Figura 1.14, se procede a configurar el servidor y el cliente como se indicó en las secciones 1.3 y 1.4, respectivamente. A continuación se realiza la programación. Dado que se trata de un pequeña aplicación para demostrar la conectividad, para este caso se programa el accionamiento **ON/OFF** de una salida digital del autómata SIMATIC S7-1500.

Programación del servidor OPC-UA en el autómata SIMATIC S7-1500

Para realizar esta tarea, debemos acceder al **Árbol del proyecto** y seleccionar **Servidor_OPC-UA [CPU 1516-3 PN/DPI] >Variables PLC >Tabla de variables estándar [65]**. Luego, procedemos a agregar las dos variables que se desean controlar desde el cliente. La Figura 1.15 visualiza claramente el proceso a seguir.

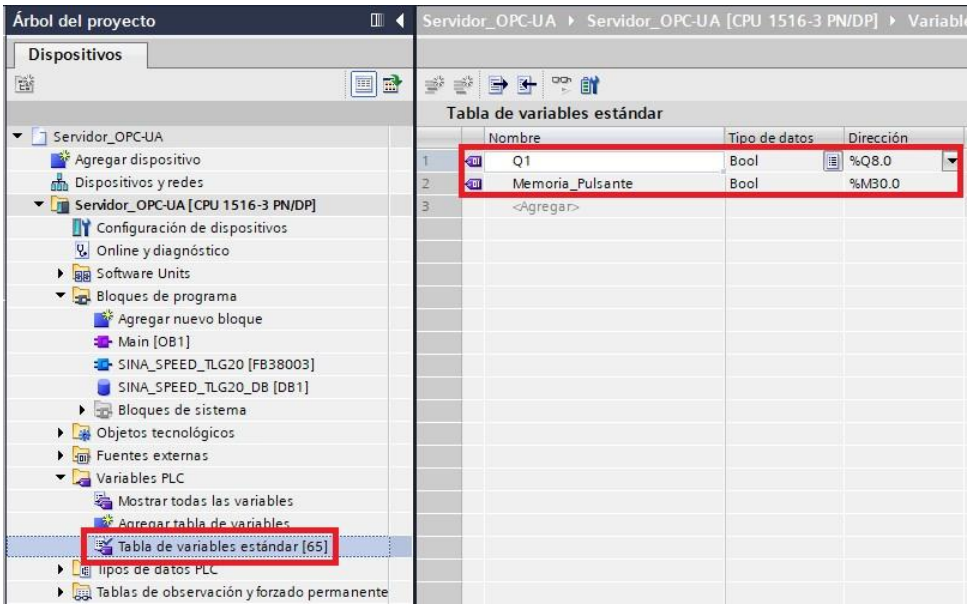


Figura 1.15: Tabla de variables para el programa del servidor OPC UA.

Luego, se debe dar clic en **Servidor_OPC-UA [CPU 1516-3 PN/DPI] >Bloques de programa >Main [OB1]** (véase la Figura 1.16). Dentro del bloque de programación se realiza el programa para el control **ON/OFF** de la salida digital **Q8.0**. Con esto el **Servidor_OPC-UA** esta completo y se procede cargar el programa en el autómata.

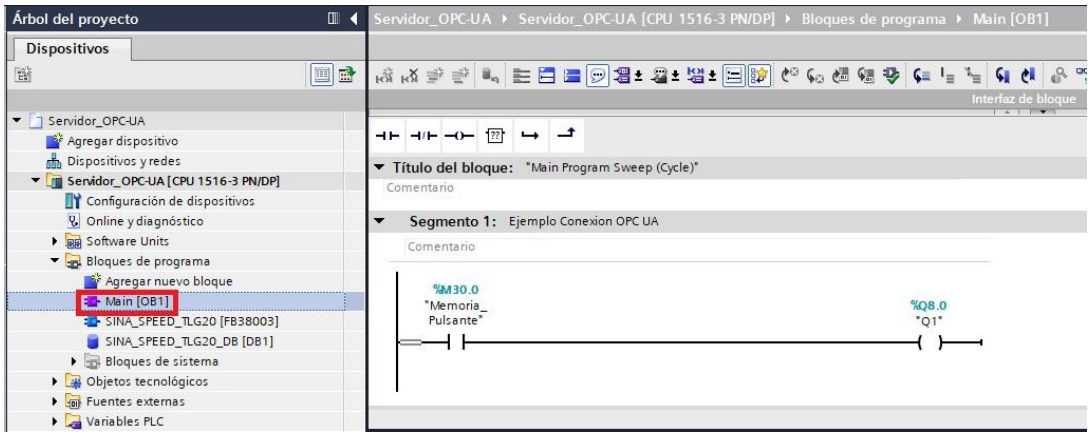


Figura 1.16: Programa de control de la salida digital **Q8.0**.

Programación del cliente OPC-UA en WinCC RT Professional

Para ello en el proyecto del cliente, en el **Árbol del proyecto** se debe dar clic en **PC-System [SIMATIC PC station] >HMI_RT_1[WinCC RT Professional] >Variables HMI >Tabla de variables estándar [48]** y se debe agregar la variable **Encender** de tipo

Boolean. Posteriormente, se despliega las propiedades de **Encender[Variable_HMI]** loc cuál permite escoger el tipo de conexión que va a tener la variable. Este proceso se puede observar en la Figura 1.17.

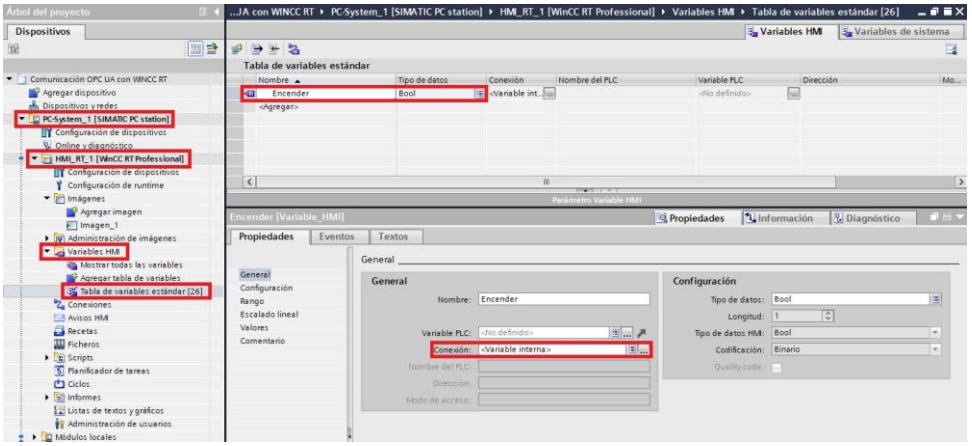


Figura 1.17: Tabla de variables para el programa del cliente OPC UA.

Ahora se establece la conexión de la variable, para ello se sigue el siguiente procedimiento. En las propiedades de **Encender[Variable_HMI]** se selecciona la opción de **Conexión** desplegando una nueva ventana que permite escoger el tipo de conexión para la variable, y se selecciona la conexión **OPC_UA** que se creó al momento de configurar el cliente OPC UA como se indica en la sección 1.4. Este proceso se puede observar en la Figura 1.18.

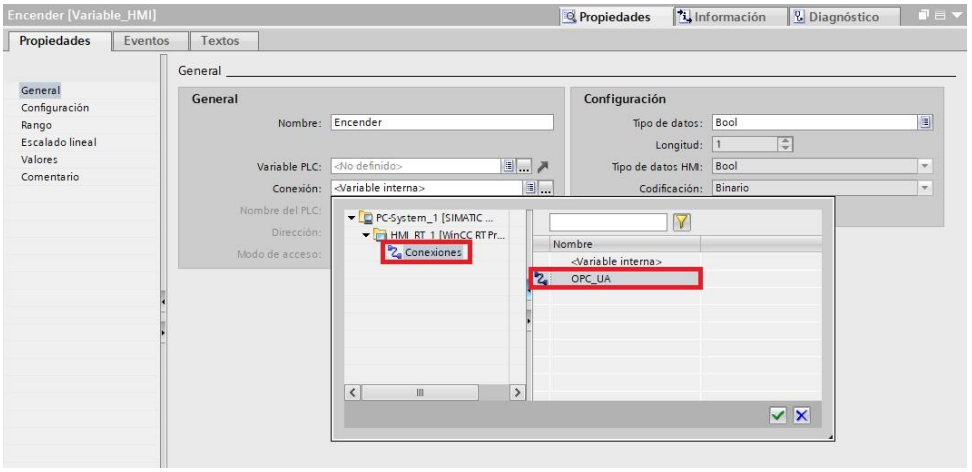


Figura 1.18: Selección de conexión para la variable a trabajar.

Luego se escoge la dirección que tiene la variable en el servidor OPC UA. De esta manera queda referenciada y conectada la variable del cliente con la variable del

servidor. Para ello se escoge la opción de **Dirección**, luego en la nueva ventana se navega hasta **Memory** y se selecciona la variable a utilizar como se indica en la Figura 1.19.

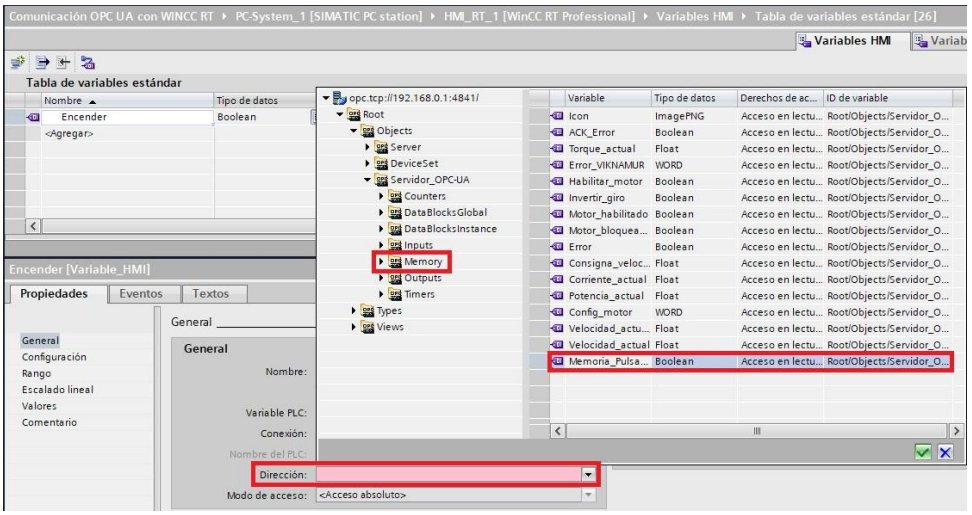


Figura 1.19: Enlace de variable cliente-servidor.

Por ultimo, se realiza la Aplicación HMI implementado con WinCC RT Profesional, para ello en el **Árbol del proyecto** se debe dar clic en **PC-System [SIMATIC PC station] >HMI_RT_1[WinCC RT Professional] >Imágenes** y se agrega una nueva imagen, en esta nueva imagen se agrega un boton **ON/OFF**, cuya función es activar y desactivar la salida digital **Q8.0** del servidor. Para realizar la interfaz gráfica se utiliza las herramientas de **Objetos básicos** y **Elementos**. Estas herramientas permiten agregar figuras, formas, textos, botones y distintas herramientas para visualización y control en la aplicación HMI. Esto se puede observar en la Figura 1.20.

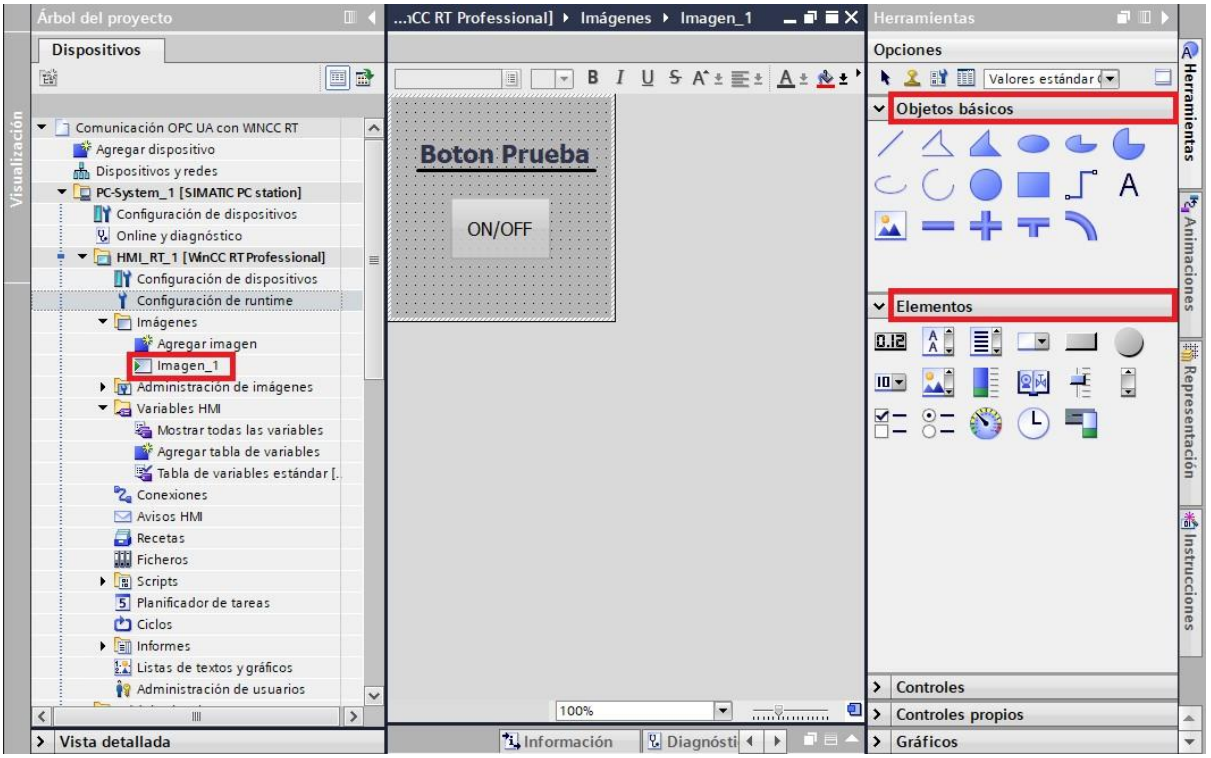


Figura 1.20: Conexiones WinCC RT Professional.

Finalmente, se realiza las pruebas respectivas y a través de la herramienta de **observación** en el servidor, se puede visualizar el funcionamiento que existe con la comunicación OPC UA del cliente con el servidor, activando y desactivando la salida digital **Q8.0**. En la Figura 1.21 se puede observar el estado de **OFF**, y al momento de pulsar el botón de prueba se observar su estado de **ON** como se visualiza en la Figura 1.22

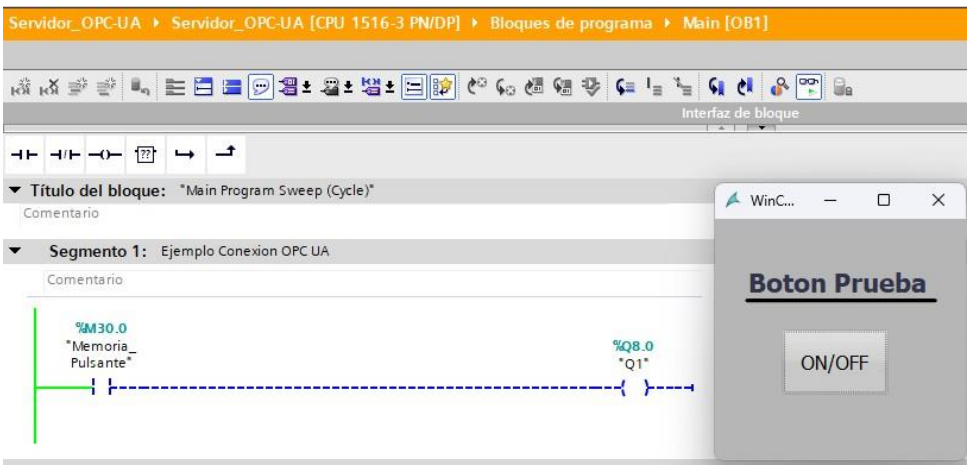


Figura 1.21: Salia digital en estado de **OFF**.

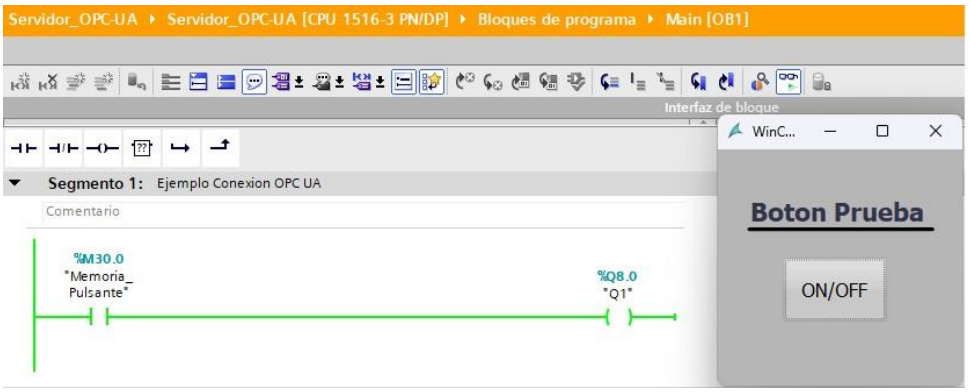


Figura 1.22: Salia digital en estado de **ON**.

Capítulo 2

SINAMICS STARTER

En este capítulo se llevará a cabo una descripción de la herramienta informática de puesta en marcha SINAMICS STARTER y como se puede usar esta por medio de comunicación PROFINET, para lo cual se dará una breve introducción al software, se detallará la interfaz de usuario con la que cuenta y se explicará cómo poner en marcha un motor usando esta herramienta.

2.1. Introducción al Software SINAMICS STARTER

SINAMICS STARTER, desarrollado por SIEMENS, es una solución integral que facilita la optimización, configuración y diagnóstico de los accionamientos pertenecientes a la familia SINAMICS. Su notable usabilidad se destaca por las guías de diálogo que orientan al usuario durante el proceso de puesta en marcha, así como las funciones de auto optimización que reducen la necesidad de ajustes manuales. Además, cuenta con una rutina de primera puesta en marcha que asegura un arranque exitoso con solo unos pocos ajustes para poner el motor en funcionamiento. SINAMICS STARTER es compatible con PC y ofrece diversas opciones de comunicación, como PROFIBUS, PROFINET/Ethernet o una interfaz serie, permitiendo una amplia conectividad [9].

SINAMICS STARTER ofrece una variedad de tareas que pueden llevarse a cabo [10]:

- Iniciar el accionamiento para ponerlo en funcionamiento.
- Controlar el accionamiento mediante el panel de control.
- Realizar optimizaciones y diagnósticos en el accionamiento.
- Configurar y activar funciones de seguridad (Safety).

La herramienta SINAMICS STARTER brinda una gama de opciones para llevar a cabo estas tareas, permitiendo una gestión efectiva y segura del accionamiento.

2.2. Configuración y parametrización del motor asíncrono

Lo primero que se debe realizar es crear un nuevo proyecto, para esto en el menú Proyecto, se selecciona la opción **Nuevo...** como se ve el la figura 2.1.

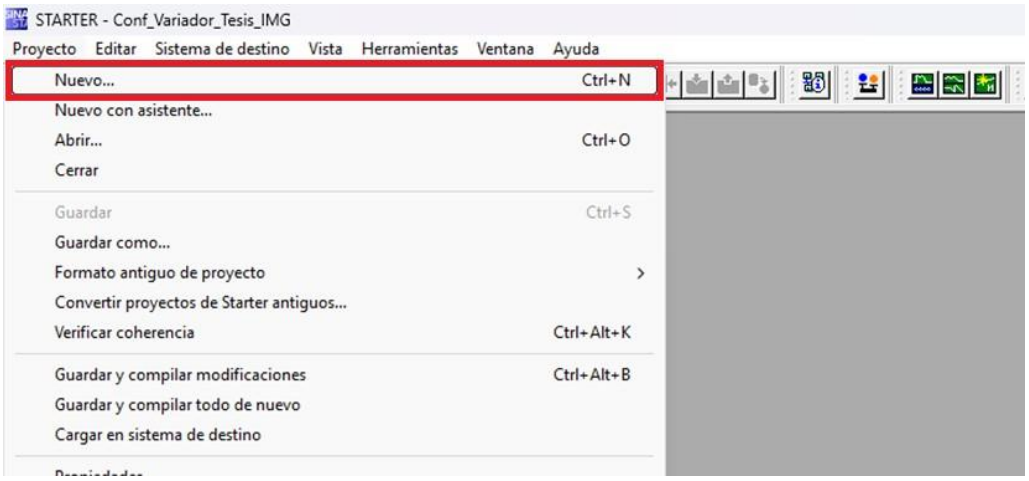


Figura 2.1: Creación de un nuevo proyecto.

Al seleccionar la opción **Nuevo...** se abre la venta **Nuevo proyecto**. En esta ventana se debe colocar el nombre que se va a dar al proyecto y se selecciona la ubicación donde se va a guardar. Después de llenar estos campos, se da clic en **Aceptar** para que se cree el nuevo proyecto, de manera visual, se ilustra en la figura 2.2.

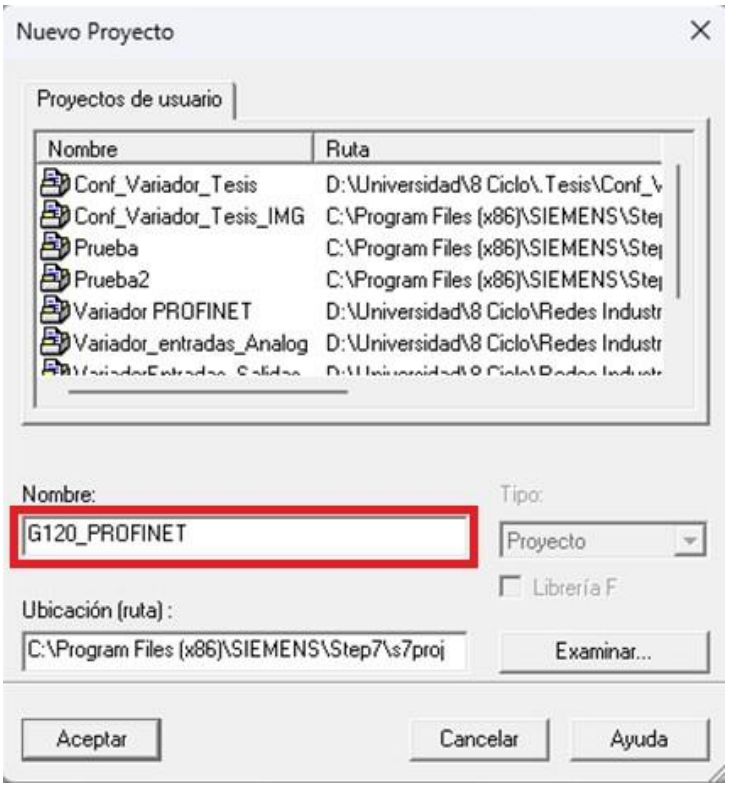


Figura 2.2: Ventana nuevo proyecto.

Una vez creado el proyecto se debe dar clic en el botón **Nodos accesibles** para poder conectarse al accionamiento. En la figura 2.3 se visualiza la ubicación del botón.

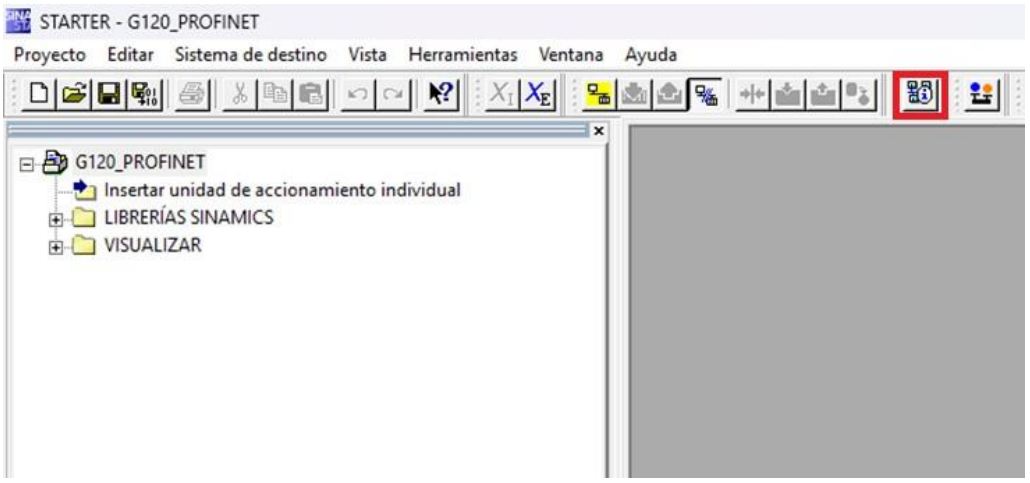


Figura 2.3: Nodos accesibles.

Al dar clic en **Nodos accesibles**, aparecen los dispositivo conectados a la computadora. Para este caso se debe seleccionar el variador **G120_CU250S-2_PN_VECTOR**. Acto seguido se da clic en el botón **Conectar**

con dispositivos de destino seleccionado, como se observa en la figura 2.4.

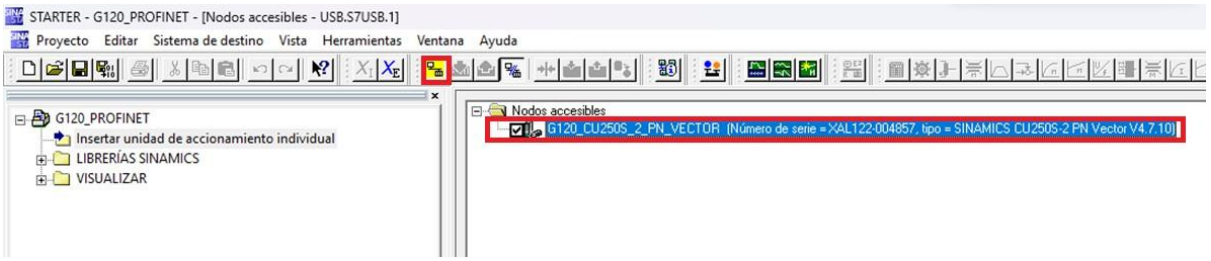


Figura 2.4: Conectarse al accionamiento.

Después de establecer la conexión, es común que aparezca una ventana que indica una discrepancia entre los datos almacenados en el variador y los nuevos datos del proyecto. Es crucial que los datos **en línea** (online) y **fuera de línea** (offline) sean idénticos, lo cual implica unificarlos. Resolver este problema es sencillo, solo se debe hacer clic en **Cargar configuración de HW en la PG**, como se visualiza en la Figura 2.5. Con esta acción, los datos se sincronizarán correctamente, asegurando una configuración coherente entre el variador y el proyecto.

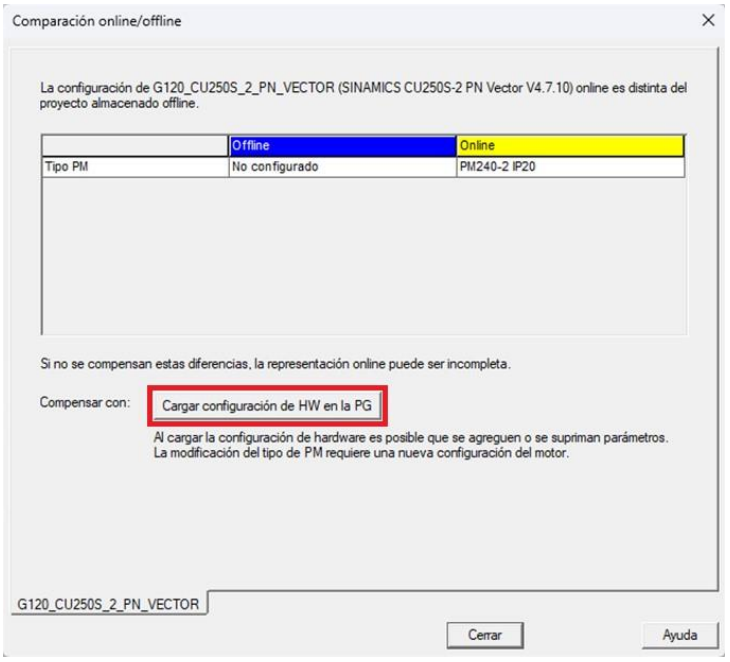


Figura 2.5: Conexión en modo online.

Para realizar la configuración y parametrización de un motor asíncrono de forma sencilla, se puede ocupar un **Asistente de configuración**. Para acceder a este

se debe seleccionar la opción **configuración** en el menú **control unit**, después en el área de trabajo se selecciona **Asistente**, como se visualiza en la figura 2.6.

Figura 2.6: Asistente de configuración.

Al ingresar al **Asistente**, se abrirá una nueva ventana que contiene varios apartados para facilitar la configuración del motor. En el primer apartado, denominado **Clase de aplicación**, se tiene la opción de seleccionar el tipo de aplicación con la que trabajará el accionamiento. Para este caso particular, se sugiere seleccionar la opción **(1) Standard Drive Control (SDS)**, la cual es especialmente apropiada para aplicaciones que involucran bombas, ventiladores, compresores, entre otros. Para una referencia visual, se puede observar la Figura 2.7.

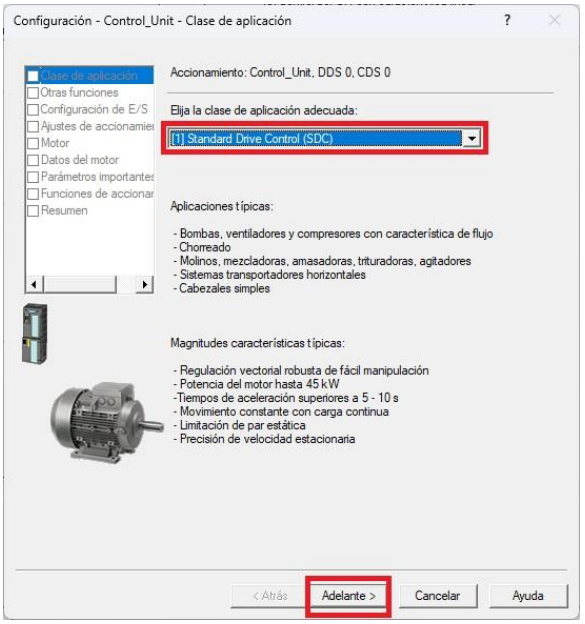


Figura 2.7: Selección de la clase de aplicación.

En el segundo apartado **Otras funciones**, se puede escoger otras funciones para aplicaciones, sin embargo, en este caso no se agregará ninguna función como se visualiza en la figura 2.8.

Figura 2.8: Otras funciones.

En el tercer apartado **Configuración E/S** se elige el tipo de ajuste predeterminado. Los ajustes establecen una configuración en los bornes de entrada y

salida del accionamiento. Para este caso se debe seleccionar el ajuste **7.)Bus de campo con conmutación de juego de datos** como se indica en la figura 2.9.

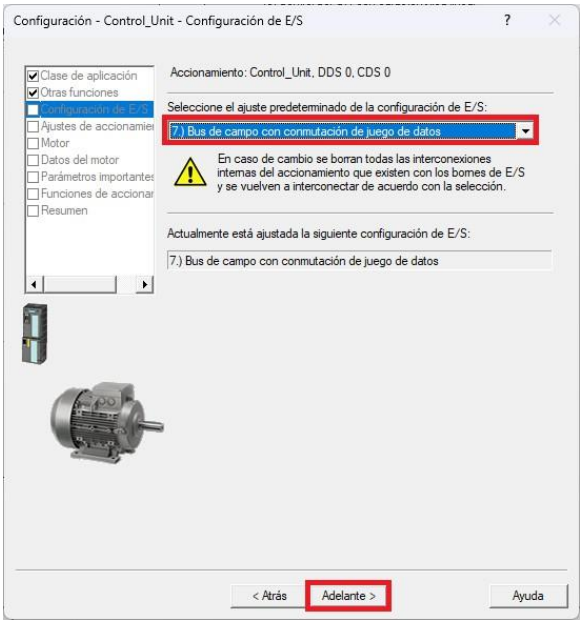


Figura 2.9: Configuración E/S.

En el cuarto apartado, titulado **Ajustes de accionamiento**, es necesario elegir la norma del motor a utilizar y la tensión a la que se encuentran conectados los dispositivos. En este caso, la norma seleccionada es **Motor IEC (50Hz, unidades SI)** y la tensión es de 220 voltios, como se visualiza en la Figura 2.10. Es importante realizar estas configuraciones de manera precisa para asegurar el correcto funcionamiento y rendimiento del accionamiento en el sistema. Una elección adecuada de la norma y la tensión garantiza una operación eficiente y segura del motor en las aplicaciones deseadas.

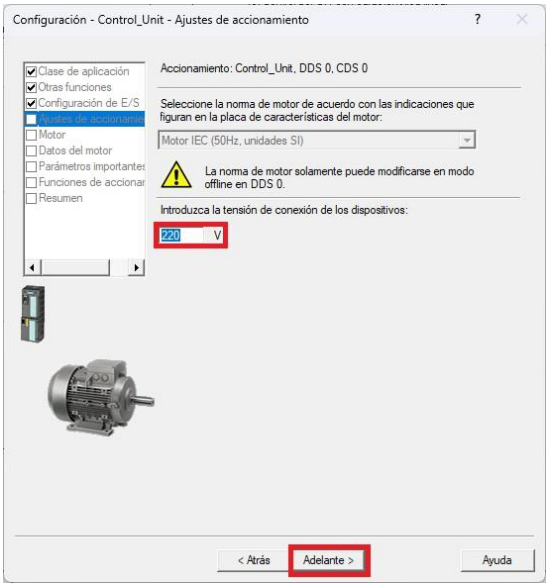


Figura 2.10: Ajustes de accionamiento.

En el quinto apartado, denominado **Motor**, es necesario elegir el tipo de motor que se empleará. En esta configuración, se ha optado por un motor **Asíncrono** y se ha seleccionado la opción de **Introducir datos de motor**, tal como se visualiza en la Figura 2.11. Esta elección del motor es esencial para garantizar una adecuada correspondencia entre el accionamiento y el tipo de motor utilizado en el sistema. Al introducir los datos del motor, se asegura una configuración precisa y óptima que permitirá el funcionamiento eficiente y seguro del conjunto motor-accionamiento en la aplicación deseada.

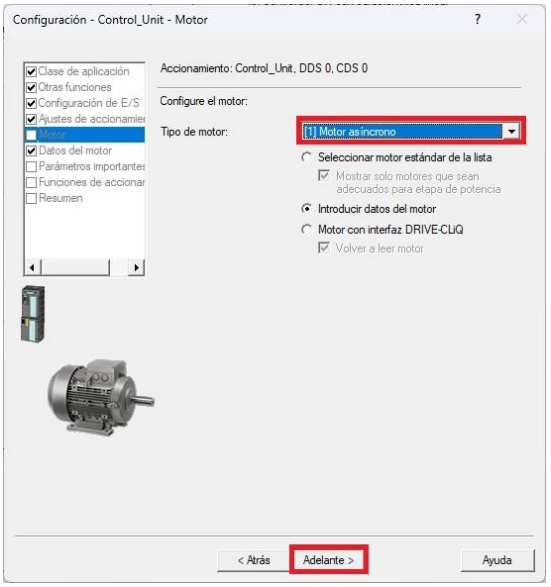


Figura 2.11: Motor.

En el sexto apartado, **Datos de motor**, es necesario ingresar los detalles específicos del motor que se utilizará, los cuales se encuentran en la placa del motor. Para el motor en uso, es esencial proporcionar con precisión los datos que se indican en la tabla 2.12. La Figura 2.12 proporciona una guía visual sobre cómo deben ingresarse dichos datos de manera adecuada. Es crucial garantizar que se capturen correctamente los detalles del motor, ya que esto asegurará una configuración óptima y un rendimiento eficiente en el funcionamiento del motor y el accionamiento. La información precisa del motor es vital para lograr una sincronización adecuada entre ambos componentes y asegurar un funcionamiento seguro y confiable en la aplicación deseada.

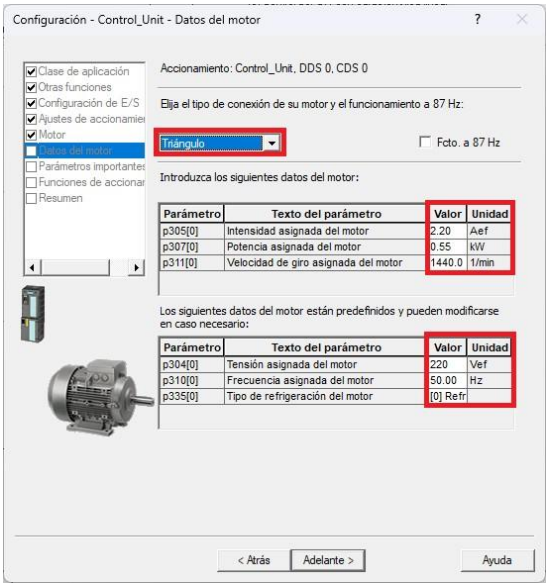


Figura 2.12: Datos de motor.

En el séptimo apartado **Parámetros importantes** se definen los valores límites a los que el motor deberá funcionar. Este apartado evitara que se dañe el motor al notificar de un error por el variador en caso de que unos de estos parámetros se sobrepasen. En la Figura 2.13 se puede observar la configuración de estos parámetros que se debe usar en este caso.

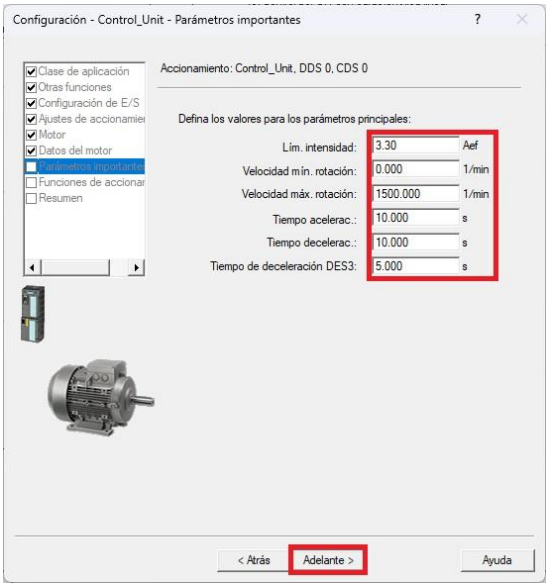


Figura 2.13: Parámetros importantes.

En el octavo apartado **Funciones de accionamiento** se escoge la utilización tecnológica adecuada y la forma en que se realizara la identificación del motor. Para

este caso se selecciona la opciones **[0] Carga constante (característica lineal)** y **[2] Identificar datos de motor (en parada)** respectivamente, como se visualiza en la Figura 2.14.

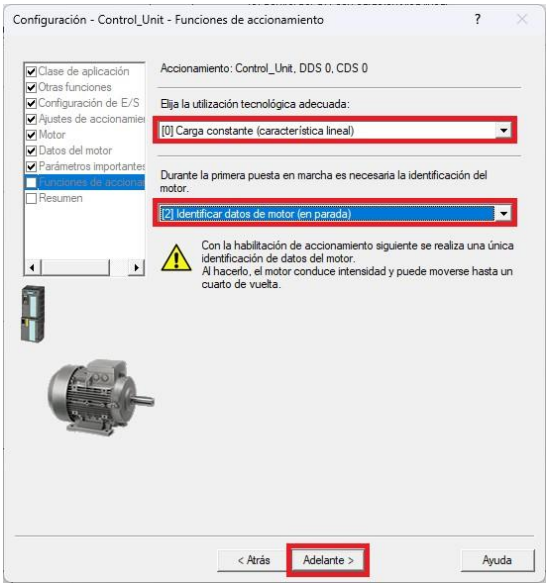


Figura 2.14: Funciones de accionamiento.

En la sección final **Resumen**, se encuentra un desglose detallado de todos los parámetros configurados a través del asistente. Una vez completada la configuración, para asegurar que los datos sean almacenados permanentemente en el accionamiento y que el motor funcione de acuerdo con los parámetros establecidos, es necesario seleccionar la opción **RAM a ROM (guardar datos en accto.)**, tal como se muestra en la Figura 2.15. Este paso es fundamental para garantizar un rendimiento óptimo en la aplicación deseada.

Figura 2.15: Resumen.

En la Tabla 2.1 se puede observar el resumen de los parámetros que se ingresa en el motor asíncrono.

Tabla 2.1: Parámetros del motor asíncrono.

Parámetro	Configuración
Norma del motor	Nema(60 Hz, SI)
Sel. Tipo de motor	Asíncrono
Tensión alimentación	220 v
Corriente del motor	2.20 Aef
Potencia del motor	0.55 kW
Velocidad del motor	1440.00 rpm
Tensión del motor	220 v
Frecuencia del Motor	60 Hz
Velocidad Mínima	0.00 rpm
Velocidad Máxima	1440.0 rpm
Tiempo de aceleración	5 s
Tiempo de deceleración	5 s
Configuración de E/S	(7) Bus de campo con conmutación de juego de datos.

2.3. Configuración del Telegrama para la comunicación Profinet

En la sección 2.2, se ha presentado la configuración y parametrización del motor asíncrono utilizado. Ahora, se procederá a realizar los ajustes necesarios en el variador SINAMCIS G120 para habilitar la comunicación Profinet con el PLC. Profinet es un protocolo de comunicación ampliamente utilizado en automatización industrial para interconectar dispositivos y sistemas en una red [11]. Siendo líder en Europa, este sistema de comunicación es ideal para entornos industriales exigentes y proporciona la precisión y velocidad necesarias para las plantas de fabricación [12]. A continuación, se detallarán los pasos que deben llevarse a cabo para llevar a cabo esta configuración.

En el árbol del proyecto se selecciona **G120_CU250S_2_PN_VECTOR >Control_Unit >Lista de experto**. La lista de experto permite acceder a todos los parámetros que tiene el variador, en la Figura 2.16 se puede visualizar este paso.

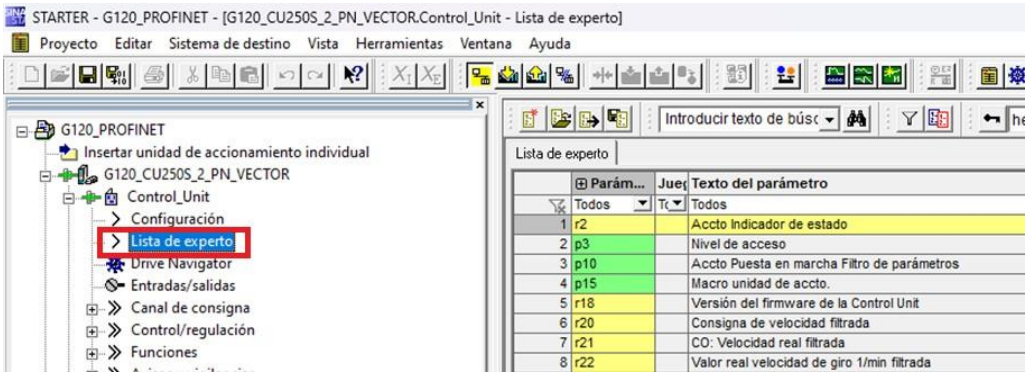


Figura 2.16: Lista de experto

En la ventana de **Lista de experto**, se busca el parámetro **p922**. El texto del parámetro debe indicar: **PROFIdrive PZD Selección de telegrama**. Como su nombre lo indica, el parámetro permite seleccionar el telegrama. En la Figura 2.17 se puede visualizar este paso.

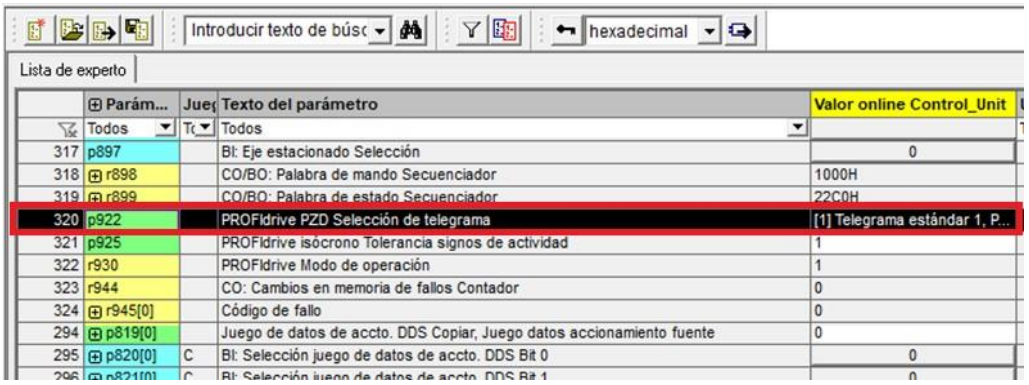


Figura 2.17: Selección del telegrama

Luego al desplazar la lista de opciones del parámetro **p922**, se selecciona **[20] Telegrama estándar 20, PZD-2/6** tal como se visualiza en la Figura 2.18

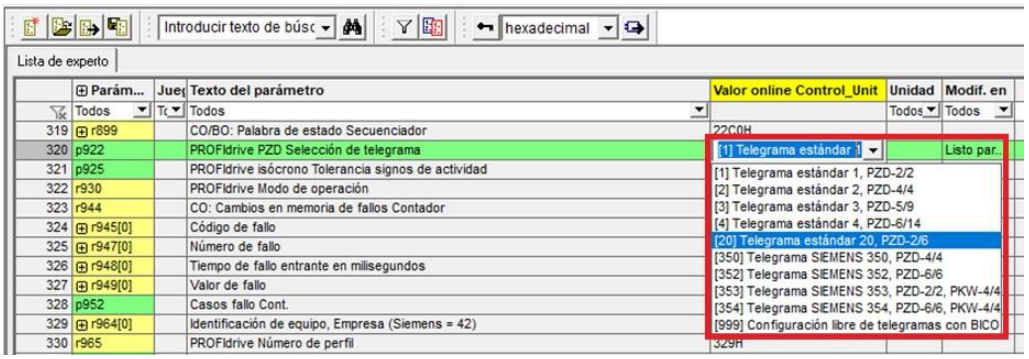


Figura 2.18: Selección del telegrama 20

En la Tabla 2.2, se presenta la estructura de los datos mediante las palabras de mando **STW1** y **ZSW1**. El **Telegrama estándar 20, PZD-2/6** consta de dos palabras para la recepción de datos y seis palabras para la transmisión de datos.

Tabla 2.2: Telegrama 20, regulación de velocidad VIK/NAMUR [13].

	PZD01	**PZD02**	**PZD03**	**PZD04**	**PZD05**	**PZD06**
Recibir dato	STW1	NSOLL_E				
Transmitir dato	ZSW1	NIST_A_GLATT	IAIST_GLATT	MIST_GLATT	PIST_GLATT	MELD_NAMUR

El variador de frecuencia recibe los datos de control y la consigna de velocidad a través del telegrama, mientras que transmite los datos de estado, como también

los valores reales de velocidad, corriente, par y potencia mediante dicho telegrama. Además, tiene la capacidad de enviar datos de error de acuerdo con la especificación **VIK_NAMUR**. Cada palabra de datos recibidos y transmitidos se encuentra definida en detalle en la Tabla 2.3.

Tabla 2.3: Descripción de palabras de recepción y transmisión de datos para el Telegrama 20 [13].

Abreviatura	Explicación
STW1	Palabra de mando 1
ZSW1	Palabra de estado 1
NSOLL_A	Consigna de velocidad 16 bits
NIST_A_GLATT	Velocidad real 16 bits
IAIST_GLATT	Intensidad real filtrada
MIST_GLATT	Par real filtrado
PIST_GLATT	Potencia activa real
MELD_NAMUR	Palabra de fallo según definición VIK-NAMUR

Seleccionado el tipo de Telegrama que se va utilizar, se procede a configurar la dirección IP del variador, para ello, en la ventana de **Lista de experto**, se busca tres parámetros claves para que el variador pueda comunicarse por Profinet. Estos parámetros son **p8920**, **p8921** y **p8923** como se indica en la Figura 2.19.

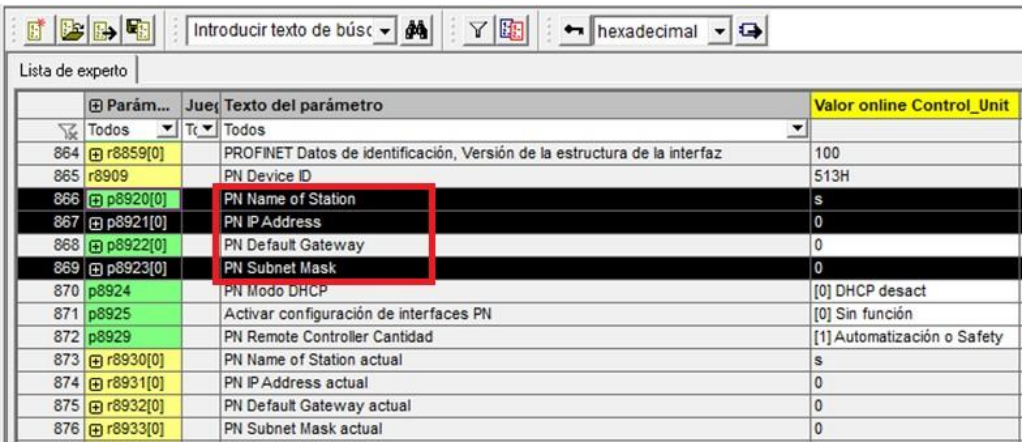


Figura 2.19: Datos de la red PROFINET

A través del parámetro **p8920 (PN Name of Station)**, se asigna el nombre de identificación. Par este caso, el nombre es **sinamics-g120sv-pn**. Esto se puede observar en la Figura 2.20.

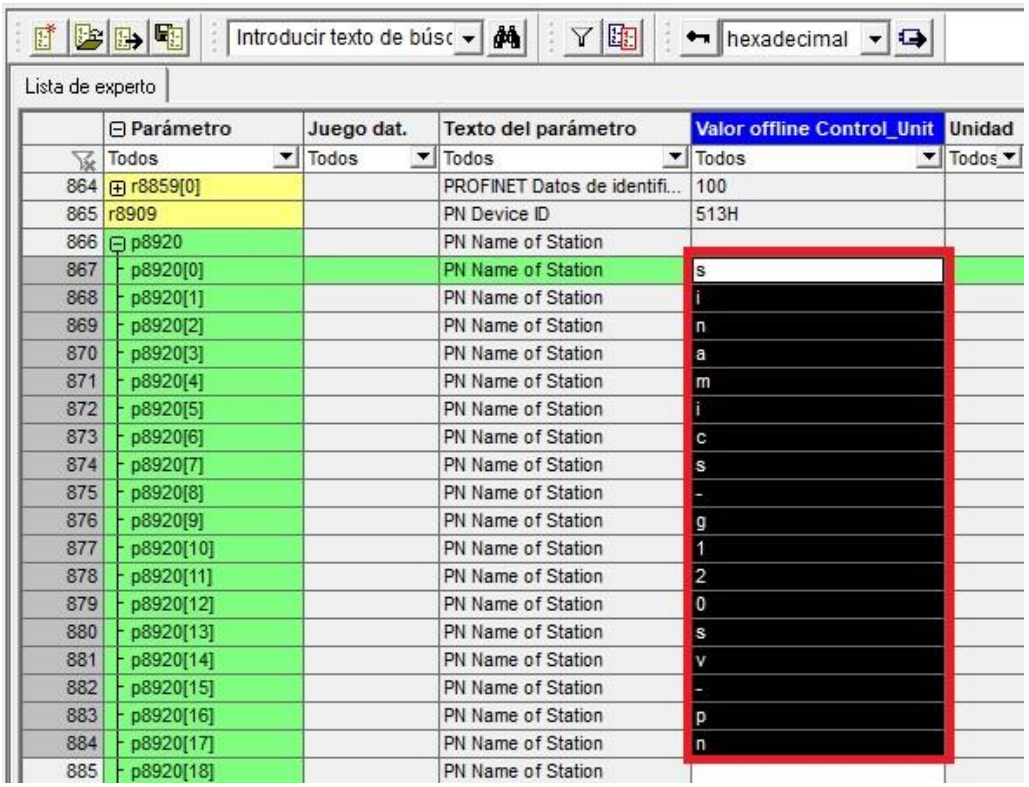


Figura 2.20: Asignación del nombre de la estación para el variador SINAMCIS G120 (CU250S-2 PN)

El parámetro **p8921 (PN IP Address)**, permite asignar la dirección IP para el variador. Para este caso se ha utilizado la dirección **192.168.0.3**. Esto se puede observar en la Figura 2.21.

Figura 2.21: Asignación de la dirección IP para el variador SINAMCIS G120 (CU250S-2 PN)

A continuación se asigna una máscara de subred al variador. En este caso, a través del parámetro **p8923 (PN Subnet Mask)**, se ha introducido la máscara **255.255.255.0**. Esta información se visualiza en la Figura 2.22.

Figura 2.22: Asignación de la máscara de red para el variador SINAMCIS G120 (CU250S-2 PN)

A continuación, se carga el proyecto en la PG como se visualiza en la Figura 2.23.

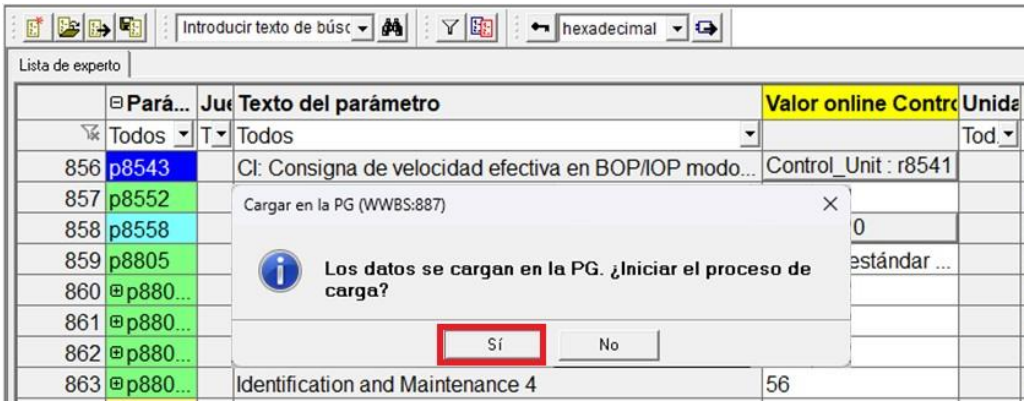


Figura 2.23: Carga de datos en la PG.

Concluida la carga el proyecto en la PG, se procede a realizar la carga de los datos en el sistema de destino como se visualiza la Figura 2.24.

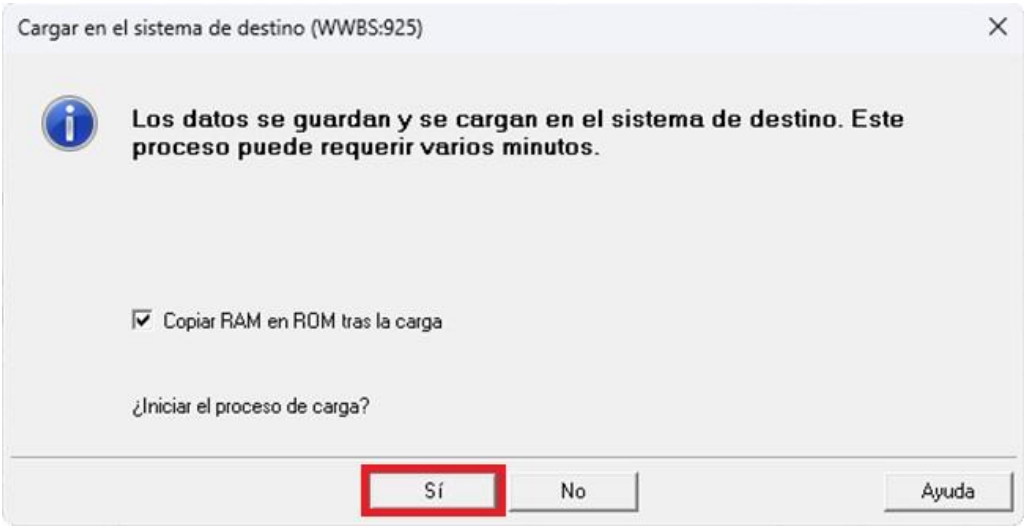


Figura 2.24: Carga de datos en el sistema de destino.

Realizada la carga de las configuraciones, se procede a conectar la fuente de alimentación trifásica 3 X 220 V CA a las entradas del convertidor de frecuencia, tal como se observa en la Figura 2.25.

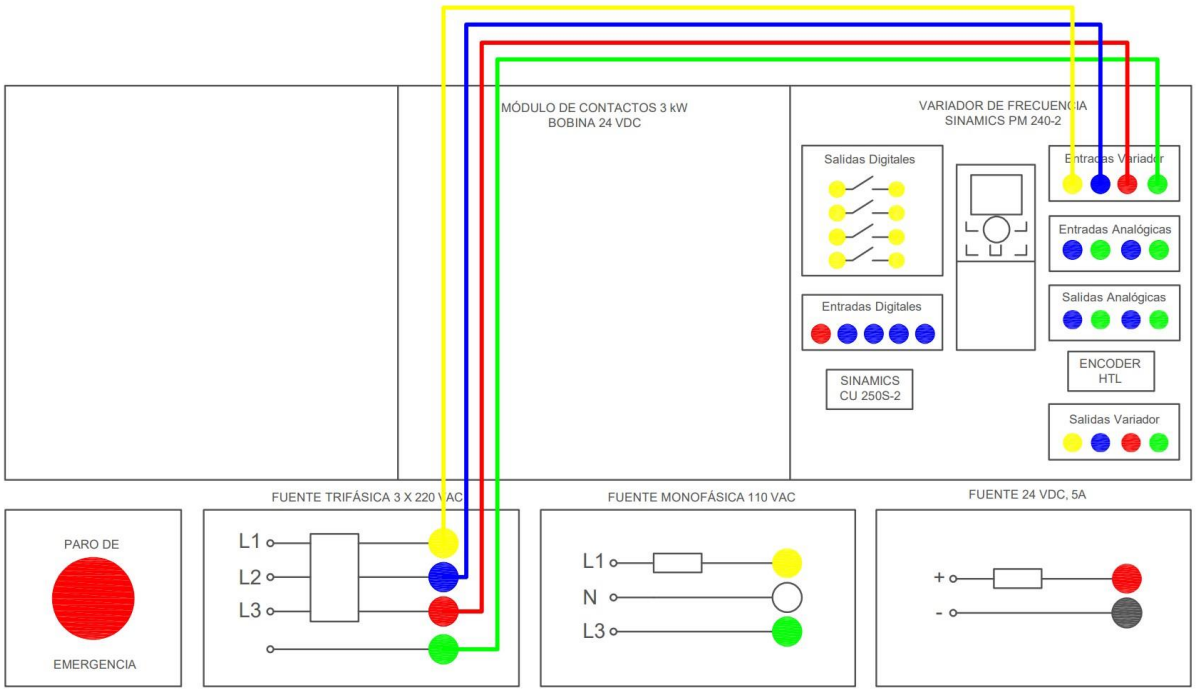


Figura 2.25: Conexionado en el banco de trabajo.

Luego se realiza la conexión del motor al convertidor de frecuencia G120, con los devanados del motor dispuestos en configuración delta, como se observa en la Figura 2.26. Las salidas **U2, V2, W2** del variador están conectadas a las entradas **V2,**

U2, W2 del motor.

Figura 2.26: Conexión delta del motor.

Una vez realizada la conexión del motor en el IOP-2 (Panel de Operador Inteligente), se generarán los siguientes avisos que indican que la **fuente trifásica 3 x 220 VAC** estuvo apagada y, además, que no se ha realizado la identificación del motor. Por consiguiente, se procederá a encender la fuente, y el IOP-2 confirmará los errores con la tecla **OK**, tal y como se visualiza en la Figura 2.27.

Figura 2.27: Confirmación de los fallos en el Variador.

A continuación se procede a realizar la identificación, para ello se accede al

modo manual a través de la tecla **Hand/Auto** como se indica en la Figura 2.28.

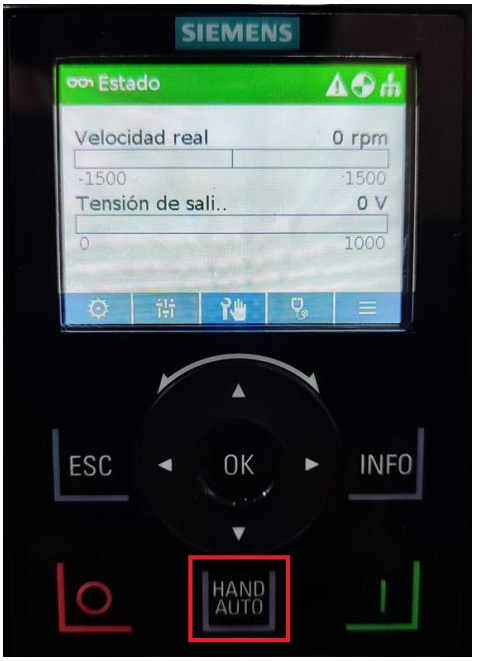


Figura 2.28: Identificación manual del motor.

Dentro del modo manual, se procede a encender el motor como se visualiza en la Figura 2.29.

Figura 2.29: Inicio la de identificación del motor.

Finalmente, el variador comenzará a realizar la identificación del motor, tal como se visualiza en la Figura 2.30. De esta manera se tiene el variador con acceso a

una comunicación Profinet y a su vez, el motor podrá ser controlado con el Telegrama de control número 20.

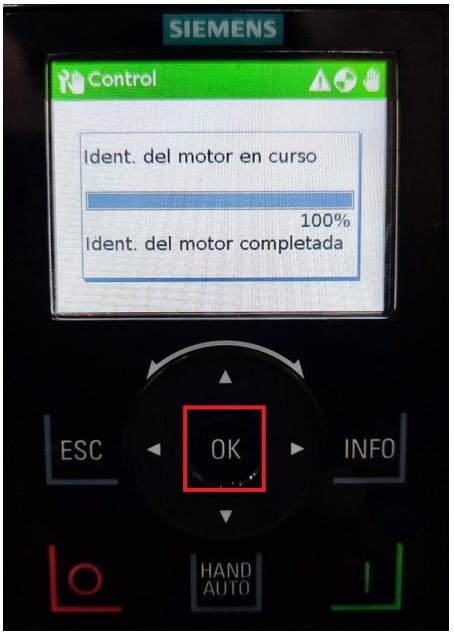


Figura 2.30: Identificación del motor completa.

Capítulo 3

Implementación del sistema de comunicación

En este capítulo se realiza la implementación del sistema de comunicación. Para la implantación se parte de las configuraciones establecidas en los capítulos 1 y 2, teniendo como finalidad de explicar como se crea y se configura una aplicación HMI para operar el proceso industrial. Cabe destacar que el estándar de comunicación utilizado es OPC UA.

3.1. Arquitectura del sistema de comunicación

El sistema de comunicación industrial en cuestión sigue la topología mostrada en la Figura 3.1 y se basa en el uso del estándar OPC UA para la operación y monitorización de un proceso industrial utilizando múltiples estaciones. El uso de esta arquitectura proporciona beneficios sustanciales en términos de eficiencia y flexibilidad al permitir una comunicación fluida entre las distintas estaciones, permitiendo la integración y la recopilación de datos en tiempo real. Además, el sistema brinda la capacidad de detectar fallos a tiempo y realizar ajustes remotos desde diferentes puestos de trabajo.

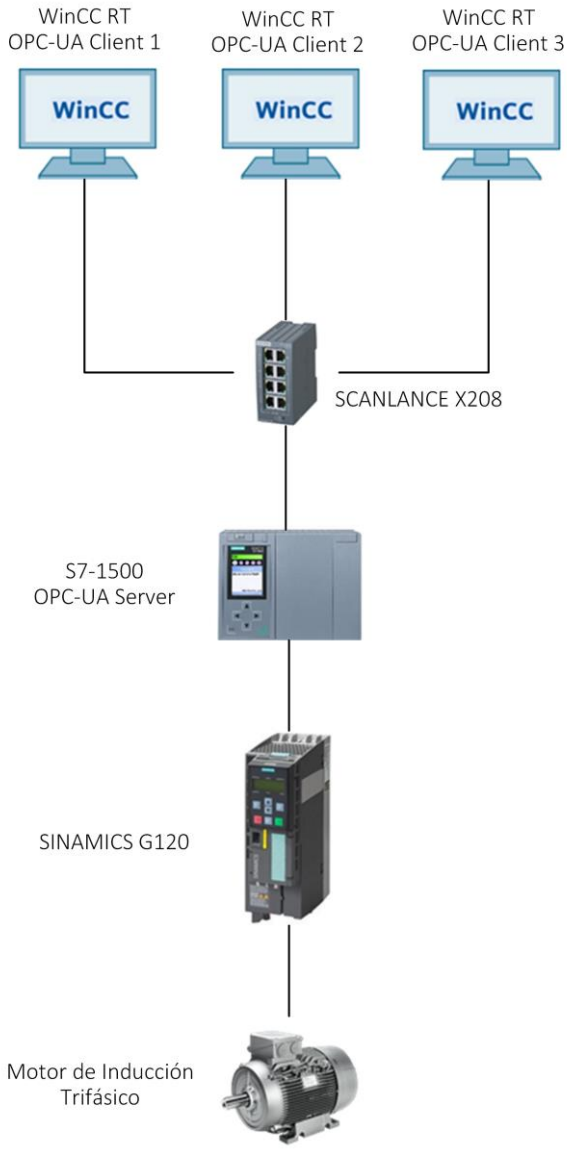


Figura 3.1: Topología del Sistema de comunicación industrial basado en OPC UA.

3.2. Configuración y programación del servidor

La implementación de la arquitectura parte de la configuración del servidor OPC UA en el PLC SIMATIC S7-1500, lo cual fue explicado con detalle en la sección 1.3. Culminada la configuración se procede a crear las variables necesarias para la programación que se va implementar en el servidor OPC UA. En la Figura 3.2 se observa la **Tabla de variables estándar** que se utilizan en esta programación.

Servidor_OPC-UA ▸ Servidor_OPC-UA [CPU 1516-3 PN/DP] ▸ Variables PLC

Tabla de variables estándar

	Nombre	Tipo de datos	Dirección	Rema...
1	Q1	Bool	%Q8.0	
2	Memoria_Pulsante	Bool	%M30.0	
3	ACK_Error	Bool	%M0.1	
4	Habilitar_motor	Bool	%M0.0	
5	Invertir_giro	Bool	%M0.2	
6	Motor_habilitado	Bool	%M0.3	
7	Motor_bloqueado	Bool	%M0.4	
8	Error	Bool	%M0.5	
9	Consigna_velocidad	Real	%MD4	
10	Lectura_velocidad_ABS	Real	%MD34	
11	Lectura_corriente	Real	%MD12	
12	Lectura_torque	Real	%MD16	
13	Lectura_potencia	Real	%MD20	
14	Config_motor	Word	%MW24	
15	Error_VIKNAMUR	Word	%MW26	
16	Lectura_velocidad	Real	%MD8	
17	<Agregar>			

Figura 3.2: Tabla de variables en el **Servidor_OPC-UA**.

La utilidad de las variables creadas se puede reconocer intuitivamente. Estas variables sirven para interactuar con la información de estado del motor, así como los parámetros de operación, como por ejemplo: corriente, torque, potencia y velocidad.

Una vez definidaslas variables, se realiza la programación. Para esto se comienza configurando el sentido de giro del motor en el bloque de programa **Main**, para lo que se ocupan bloques **MOVE** (obsérvese la figura 3.3). Es importante considerar que el sentido de giro se activa según la palabra de mando utilizada. La palabra **16#003F** activa el giro en sentido horario, mientras que la palabra **16#007F** activa el giro en sentido antihorario.

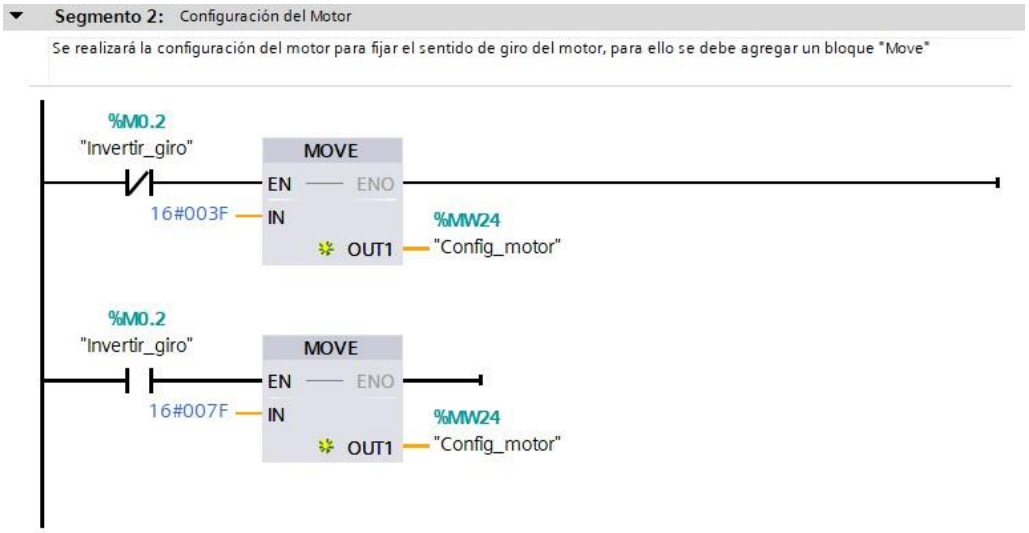


Figura 3.3: Inversión de giro con bloques MOVE.

Con el sentido de giro ya configurado, se procede a configurar los parámetros del telegrama 20. Para esto, lo primero que se debe realizar es añadir la librería **LSINAext**. Esta librería permite realizar una comunicación entre el cliente y el variador. Esto se logra con el bloque **SINA_SPEED_TLG20**. Esta librería se encuentra disponible para su descarga en la pagina de SIEMENS. Para agregar esta librería a la programadora TIA Portal, en el menú **Librerías** que se encuentra en el lado derecho de la pantalla, en **Librerías globales**, se da clic en el botón **Abrir librería global**, como se visualiza en la figura 3.4.

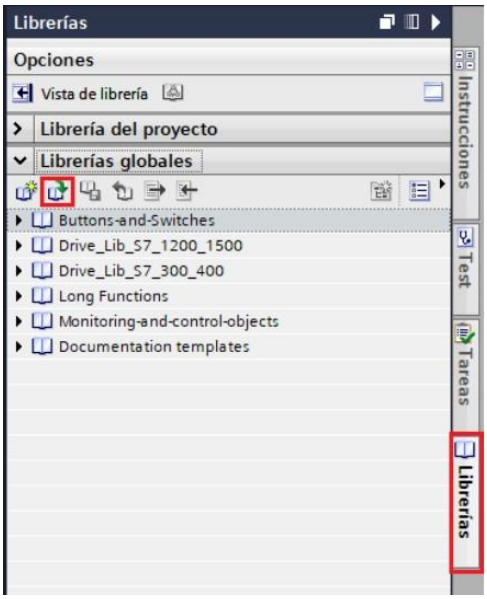


Figura 3.4: Librerías en TIA Portal.

Al dar clic en el botón **Abrir librería global**, se abrirá una ventana con el mismo nombre del botón. En esta ventana se debe buscar la carpeta **LSINAext**. En esta se selecciona el archivo **LSINAext_V15.1**. Cuando se tiene seleccionado el archivo se da clic en **Abrir**, como se observa en la figura 3.5.

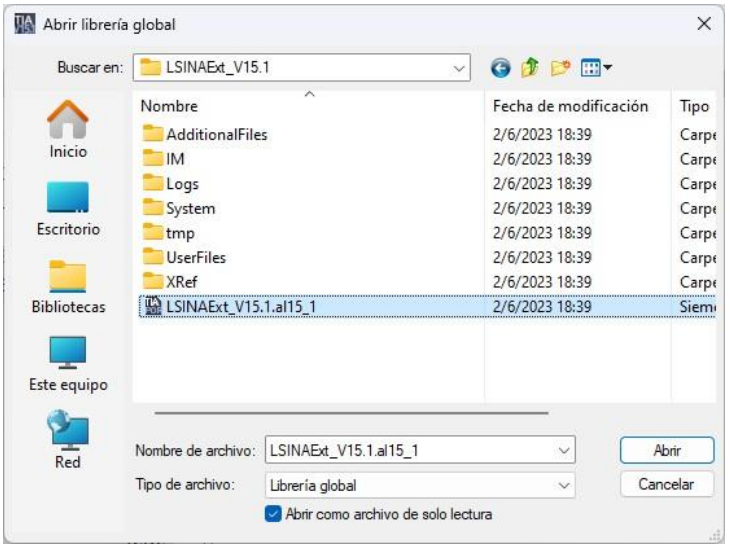


Figura 3.5: Abrir librería global.

Para colocar el bloque **SINA_SPEED_TLG20** hay que dirigirse a las librerías globales, donde ahora se encuentra la librería **LSINAext**. En esta se debe dirigir a **Plantillas maestras** y selccionar **SINA_SPEED_TLG20**, tal como se visualiza en la figura 3.6.

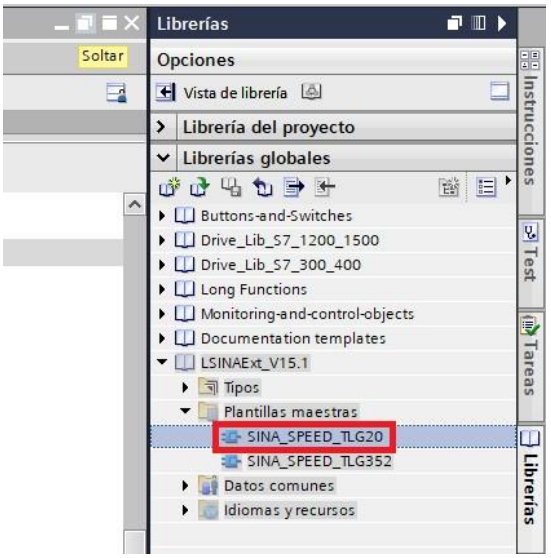


Figura 3.6: Bloque SINA_SPEED_TLG20.

El bloque **SINA_SPEED_TLG20** cuenta con parámetros de entrada y salida que posibilitan el control y monitoreo de un motor mediante el variador de frecuencia **SINAMICS G120 PN**. Además, proporciona estados de error y sus correspondientes avisos. Los parámetros de entrada son transmitidos al variador para su configuración y funcionamiento. Estos se especifican en la tabla 3.1.

Tabla 3.1: Descripción parametros de entrada del bloque **SINA_SPEED_TLG20**. [13].

Entrada	Tipo de Dato	Función
enableAxis	Bool	Activar el funcionamiento del motor
ackError	Bool	Realizar acuse de fallo
SpeedSP	Real	Asignar consigna de fallo
selectCDS	————-	————————————-
refSpeed	Real	Asignar referencia de velocidad
refCurrent	Real	Asignar referencia de corriente
refTorque	Real	Asignar referencia de torque
refPower	Real	Asignar referencia de potencia
configAxis	Word	Palabra de control: 16#003F: Giro sentido horario 16#007F: Giro sentido antihorario

Los parámetros de salida obtienen los datos de estado del variador, además de mostrar los parámetros actuales de velocidad, corriente, torque y potencia. Estos se especifican en la tabla 3.2 y en la figura 3.7.

Tabla 3.2: Descripción parametros de salida del bloque **SINA_SPEED_TLG20**. [13].

Salida	Tipo de Dato	Función
axisEnable	Bool	Indicar si el motor se encuentra activo
lockout	Bool	Indicar si el motor se está bloqueado
actCDS	————-	————————————-
actSpeed	Real	Indica la velocidad actual (rpm)
actCurrent	Real	Indica la corriente actual (A)
actTorque	Real	Indica la torque actual (Nm)
actPower	Real	Indica la potencia activa actual (kW)
faultVIKNAMUR	Word	Palabra de fallo según VIK-NAMUR
Error	Bool	Indica si existe un error

El bloque **SINA_SPEED_TLG20** se debe colocar en el segmento que le sigue al de la configuración de giro. Una vez colocado el bloque se procede a llenar los parámetros del telegrama 20 de la manera que se visualiza en la figura 3.8.

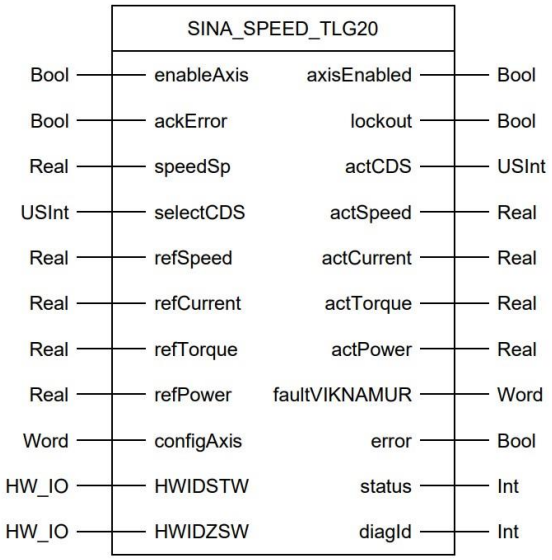


Figura 3.7: Bloque SINA_SPEED_TLG20 y su representación de tipo de variables.

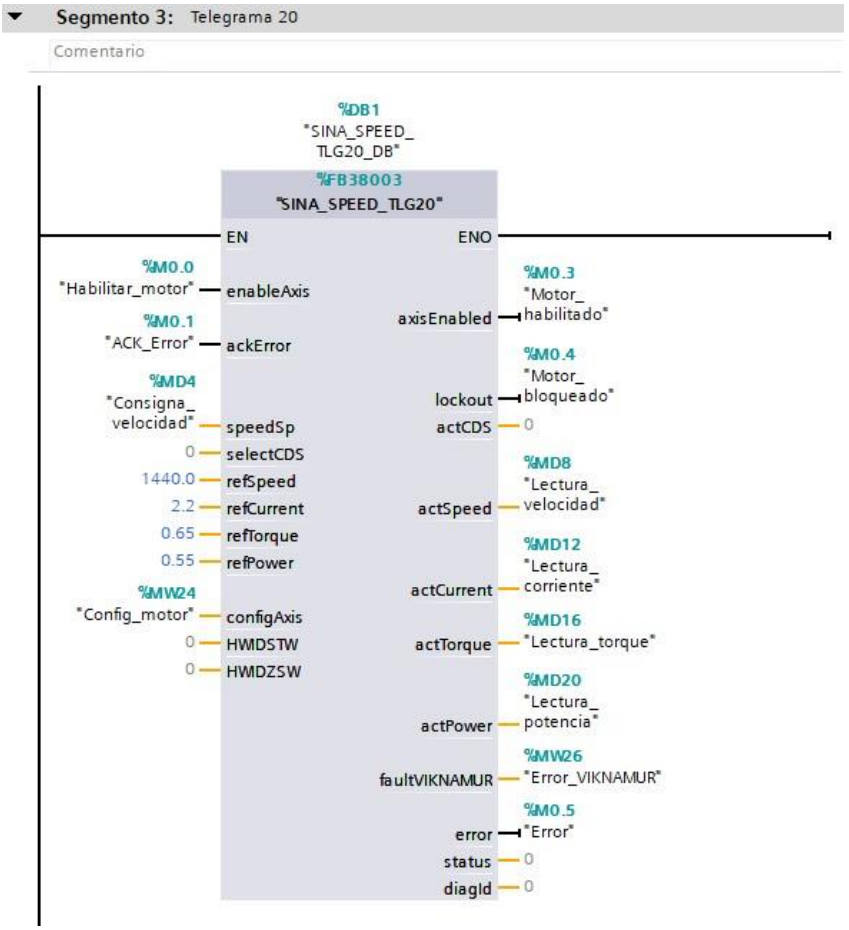


Figura 3.8: Configuración del bloque SINA_SPEED_TLG20.

Una vez que los parámetros del bloque están configurados, es necesario hacer una corrección en el código para evitar que el parámetro **error** inicie activado. Para realizar esta corrección, se debe hacer doble clic en la función **SINA_SPEED_TLG20** y dirigirse a la línea 196 del código para ajustarla según se visualiza en la Figura 3.9.

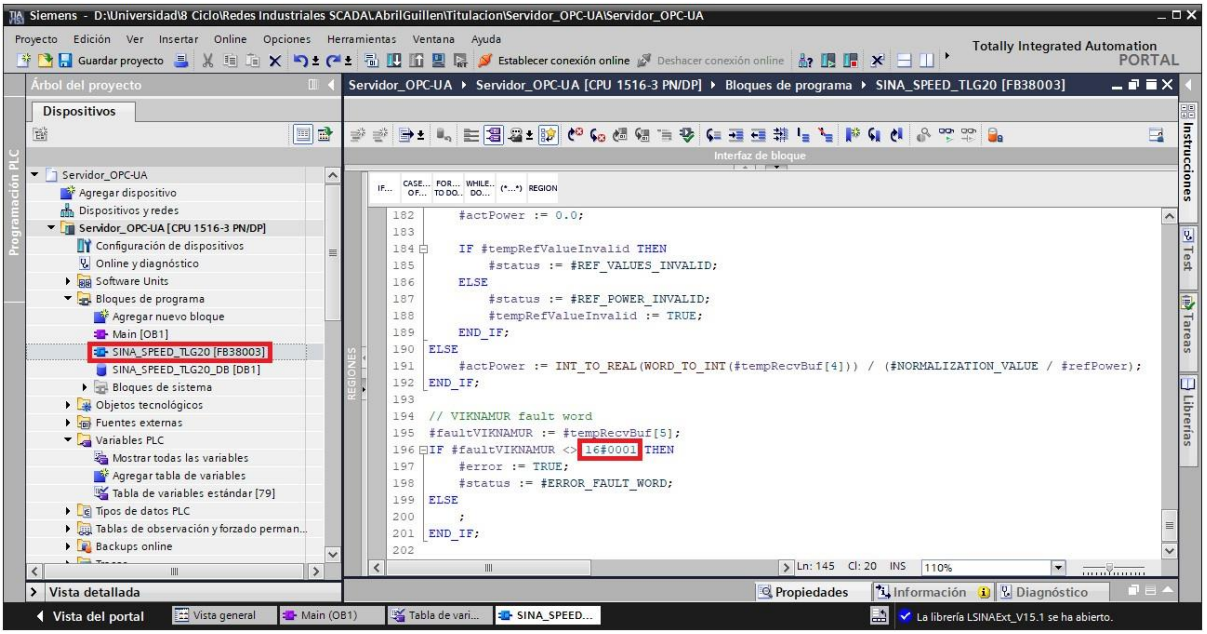


Figura 3.9: Corrección en la función SINA_SPEED_TLG20.

Tras solucionar el error, se incluyen las palabras de mando y estado que facilitan la comunicación entre el (PLCPLC) y el variador de frecuencia. Estas palabras son **HWDSTW** (envío) y **HWDZSW** (recepción). En primer lugar, se implementa la palabra de envío, como se visualiza en la Figura 3.10.

Figura 3.10: Configuración de la palabra de envío.

Después, se agrega la palabra de recepción de la forma que se visualiza en la figura 3.11.

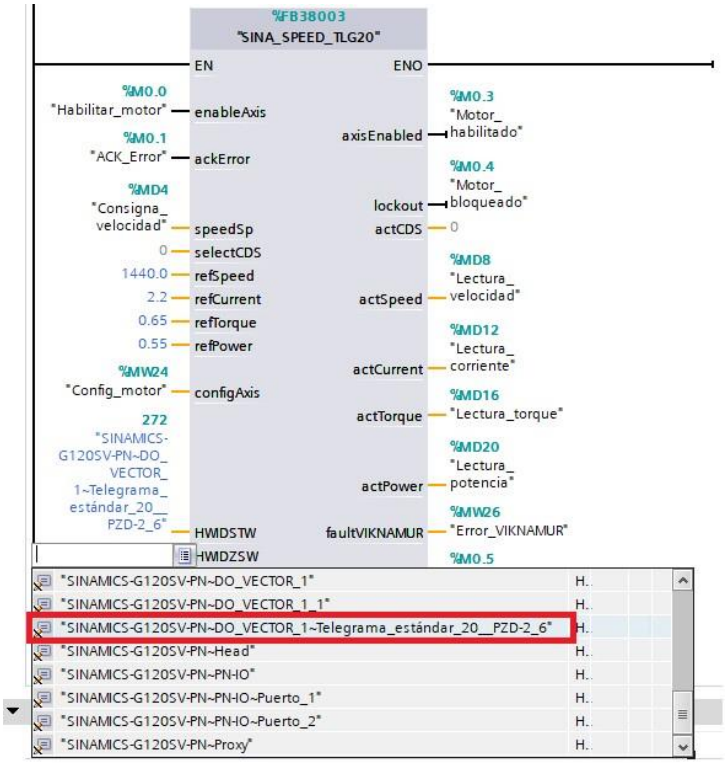


Figura 3.11: Configuración de la palabra de recepción.

Con los parámetros del telegrama 20 y las palabras de mando y estado ya agregadas, el bloque **SINA_SPEED_TLG20** se debe ver como se visualiza en la figura 3.12.

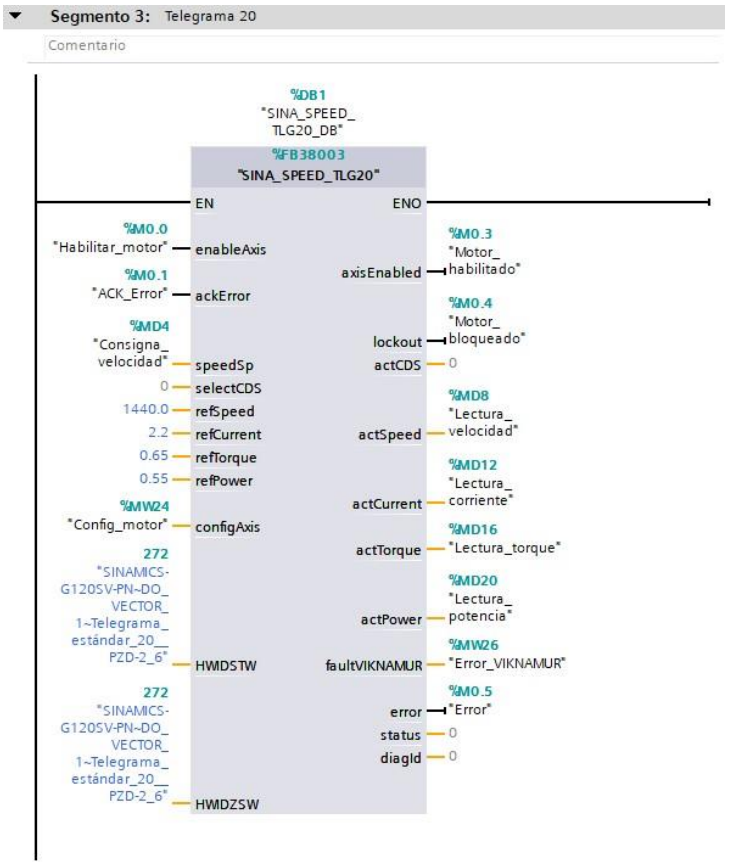


Figura 3.12: Configuración completa del bloque SINA_SPEED_TLG20.

Para finalizar con la programación, se agrega un bloque **ABS REAL**, para poder transformar el valor de velocidad que nos da el telegrama 20 en uno absoluto para evitar tener valores negativos. (Véase la figura 3.13).

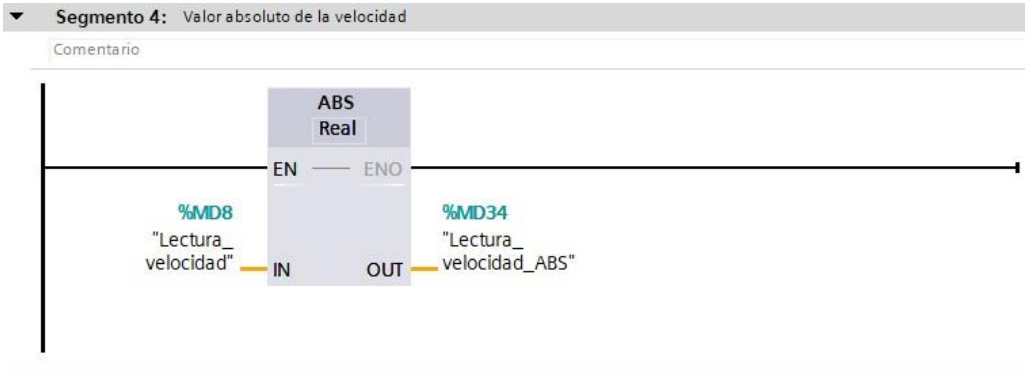
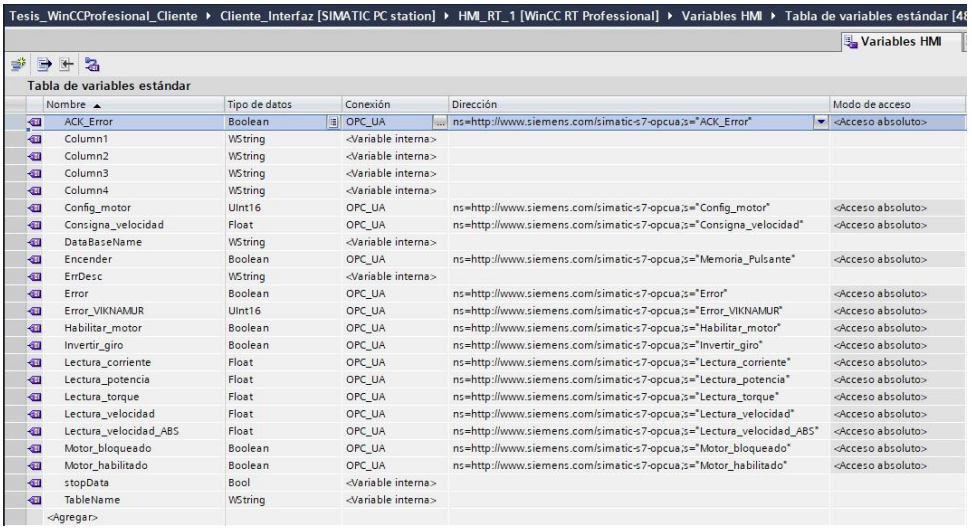


Figura 3.13: Valor absoluto de velocidad.

3.3. Configuración y programación del cliente

Una vez realizada la programación del servidor OPC UA en el SIMATIC S7-1500 se procede a realizar la aplicación HMI del cliente en WinCC RT Professional. Para realizar esta aplicación es necesario tener configurada la comunicación OPC UA como se indicó en la sección 1.4. Finalizado la configuración, se procede agregar las variables que se utilizan en la aplicación HMI en la **Tabla de variables estándar** como se visualiza en la Figura 3.14.



Nombre	Tipo de datos	Conexión	Dirección	Modo de acceso
ACK_Error	Boolean	OPC_UA	ns=http://www.siemens.com/simatic-s7-opcua;s="ACK_Error"	<Acceso absoluto>
Column1	WString	<Variable interna>		
Column2	WString	<Variable interna>		
Column3	WString	<Variable interna>		
Column4	WString	<Variable interna>		
Config_motor	UInt16	OPC_UA	ns=http://www.siemens.com/simatic-s7-opcua;s="Config_motor"	<Acceso absoluto>
Consigna_velocidad	Float	OPC_UA	ns=http://www.siemens.com/simatic-s7-opcua;s="Consigna_velocidad"	<Acceso absoluto>
DataBaseName	WString	<Variable interna>		
Encender	Boolean	OPC_UA	ns=http://www.siemens.com/simatic-s7-opcua;s="Memoria_Pulsante"	<Acceso absoluto>
ErrDesc	WString	<Variable interna>		
Error	Boolean	OPC_UA	ns=http://www.siemens.com/simatic-s7-opcua;s="Error"	<Acceso absoluto>
Error_VIKNAMUR	UInt16	OPC_UA	ns=http://www.siemens.com/simatic-s7-opcua;s="Error_VIKNAMUR"	<Acceso absoluto>
Habilitar_motor	Boolean	OPC_UA	ns=http://www.siemens.com/simatic-s7-opcua;s="Habilitar_motor"	<Acceso absoluto>
Invertir_giro	Boolean	OPC_UA	ns=http://www.siemens.com/simatic-s7-opcua;s="Invertir_giro"	<Acceso absoluto>
Lectura_corriente	Float	OPC_UA	ns=http://www.siemens.com/simatic-s7-opcua;s="Lectura_corriente"	<Acceso absoluto>
Lectura_potencia	Float	OPC_UA	ns=http://www.siemens.com/simatic-s7-opcua;s="Lectura_potencia"	<Acceso absoluto>
Lectura_torque	Float	OPC_UA	ns=http://www.siemens.com/simatic-s7-opcua;s="Lectura_torque"	<Acceso absoluto>
Lectura_velocidad	Float	OPC_UA	ns=http://www.siemens.com/simatic-s7-opcua;s="Lectura_velocidad"	<Acceso absoluto>
Lectura_velocidad_ABS	Float	OPC_UA	ns=http://www.siemens.com/simatic-s7-opcua;s="Lectura_velocidad_ABS"	<Acceso absoluto>
Motor_bloqueado	Boolean	OPC_UA	ns=http://www.siemens.com/simatic-s7-opcua;s="Motor_bloqueado"	<Acceso absoluto>
Motor_habilitado	Boolean	OPC_UA	ns=http://www.siemens.com/simatic-s7-opcua;s="Motor_habilitado"	<Acceso absoluto>
stopData	Bool	<Variable interna>		
TableName	WString	<Variable interna>		
<Agregar>				

Figura 3.14: Variables para la aplicación HMI del cliente.

Luego se procede a crear las imágenes necesarias para la aplicación HMI. En este caso se crea las imágenes (**Imagen_raiz**, **T_Avisos**, **G_Velocidad**, **G_Corriente**, **G_Potencia**, **G_Torque**, **DB**) como se visualiza en la Figura 3.15.

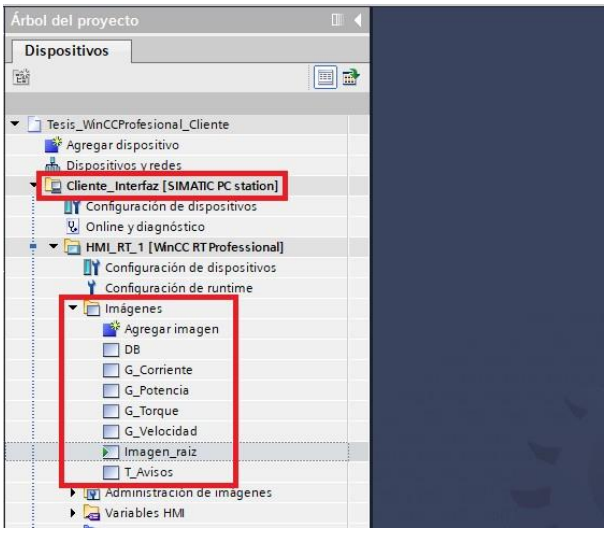


Figura 3.15: Imágenes para la aplicación HMI del cliente.

Creada las variables se procede a realizar la interfaz principal en la **Imagen_raíz** como observa en la Figura 3.16, para realizar la interfaz se usan herramientas que proporciona WinCC RT Professional. La interfaz que se diseñó consta de cinco partes.

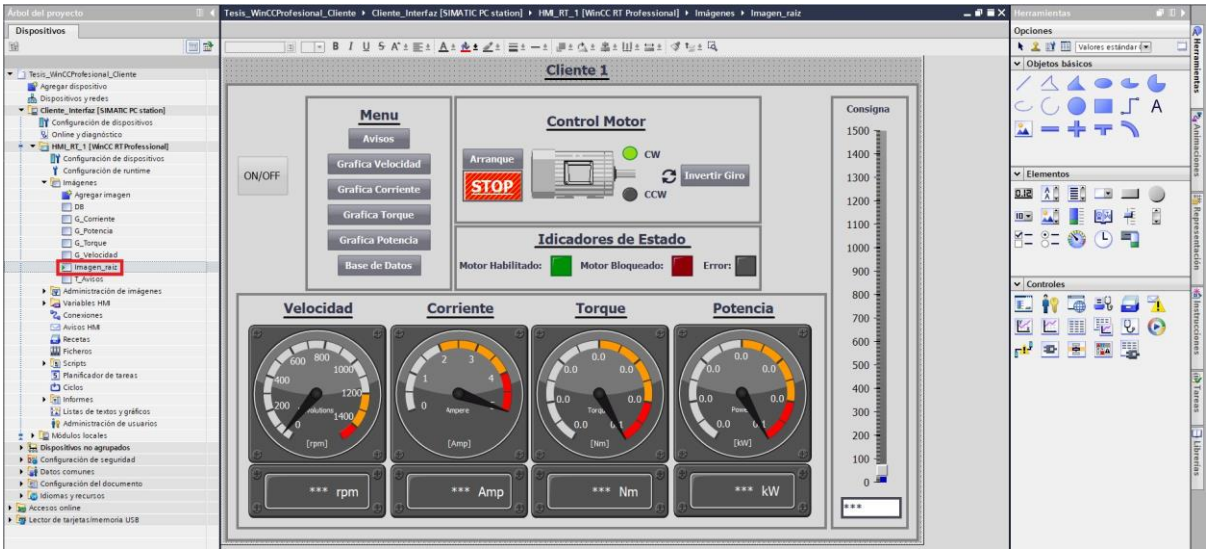


Figura 3.16: Interfaz principal de control y monitoreo de la aplicación HMI del cliente.

La primera parte, **Menu** (véase la figura 3.17), consta de cinco botones que permiten navegar entre las diferentes imágenes que se crearon para mostrar los avisos de errores, las curvas de los datos que da el motor y para llevar a cabo la creación de la base de datos.

Figura 3.17: Menu.

La segunda parte, **Control Motor** (véase la figura 3.18), está diseñada, como su nombre lo indica, para controlar el motor al poder encenderlo, apagarlo o cambiar el sentido de giro.

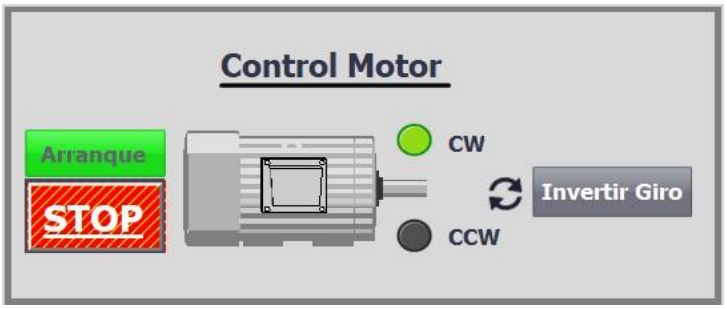


Figura 3.18: Control del Motor.

La tercera parte, **Identificadores de Estado** (véase la figura 3.19), muestra cuando el motor esta habilitado, bloqueado, o si se produjo un error.

Figura 3.19: Indicadores de Estado.

La cuarta parte, **Indicadores** (véase la figura 3.20) cuenta con cuatro indicadores en los que se puede observar la velocidad, la corriente, el torque y la potencia actual del motor.

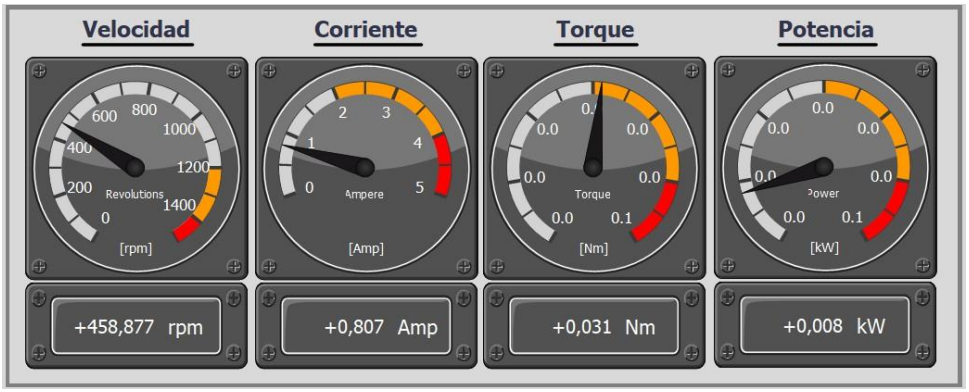


Figura 3.20: Indicadores de velocidad, corriente, torque y potencia.

La quinta y ultima parte **Consigna de velocidad** (véase la figura 3.21) consta de un deslizador y de un campo que permiten al usuario ingresar la velocidad a la que va a girar el motor.

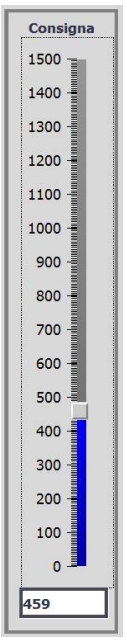


Figura 3.21: Consigna de velocidad.

Una vez creada la interfaz principal, el siguiente paso es asignar la lógica a cada uno de sus elementos. Comenzando con los botones, seleccionamos el botón **Avisos**, y en el menú **Propiedades >Eventos**, elegimos la opción **Pulsar el botón izquierdo del ratón**. A continuación, en la sección **<Agregar función>**, optamos por la función **ActivarImagen**. Para vincularlo adecuadamente, en la opción **Variable (Entrada/Salida)**, seleccionamos la variable **T_Avisos**. Todo este proceso se visualiza

en la Figura 3.22.

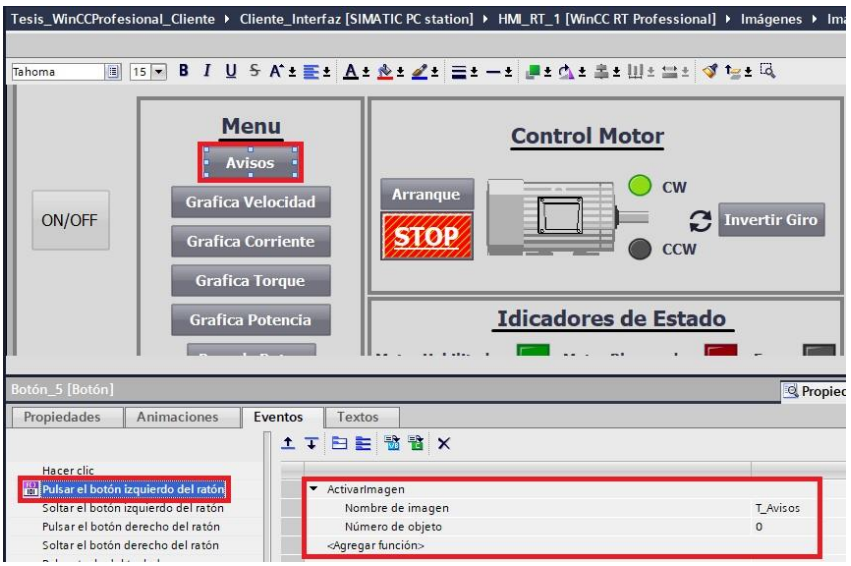


Figura 3.22: Configuración del botón **Avisos**.

De igual manera, todos los elementos de tipo **botón** siguen el mismo procedimiento para asignar su lógica individual. Los parámetros de configuración específicos para cada uno de estos elementos están detallados en la Tabla 3.3.

Tabla 3.3: Parámetros de configuración de los elementos de tipo botón.

Botón	Evento	Función	Parámetro de funcion
Avisos	Pulsar el botón izquierdo del ratón	ActivarImagen	T_Avisos
Grafica Velocidad	Pulsar el botón izquierdo del ratón	ActivarImagen	G_Velocidad
Grafica Corriente	Pulsar el botón izquierdo del ratón	ActivarImagen	G_Corriente
Grafica Torque	Pulsar el botón izquierdo del ratón	ActivarImagen	G_Torque
Grafica Potencia	Pulsar el botón izquierdo del ratón	ActivarImagen	G_Potencia
Base de Datos	Pulsar el botón izquierdo del ratón	ActivarImagen	DB
Arranque	Pulsar el botón izquierdo del ratón	ActivarBit	Habilitar_motor
STOP	Pulsar el botón izquierdo del ratón	DesactivarBit	Habilitar_motor
Invertir Giro	Pulsar el botón izquierdo del ratón	InvertirBit	Invertir_giro

Los indicadores siguen un proceso similar pero con una pequeña diferencia en los indicadores de inversión de giro, **CW** (giro horario) y **CCW** (giro antihorario). En el indicador **CW**, se accede a **Propiedades** >**General** y se selecciona **Proceso**. A continuación, se selecciona la variable **invertir_giro** y en la ventana de **Contenido**, se asigna **PilotLight_Round_GN_Off_256c** para el funcionamiento **On** y **PilotLight_Round_GN_On_256c** para el funcionamiento **Off**. Este proceso se puede visualizar en la Figura 3.23. Por otro lado, en el indicador **CCW**, el procedimiento

es similar, pero con los parámetros de **Contenido** invertidos; es decir, el parámetro **On** se le asigna **PilotLight_Round_GN_On_256c** y el parámetro **Off** se le asigna **PilotLight_Round_GN_Off_256c**.

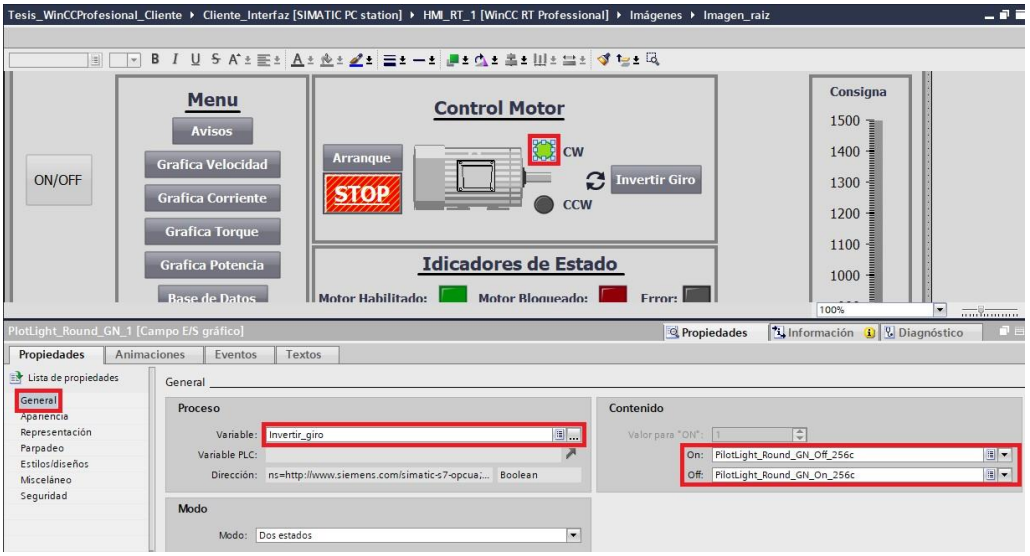


Figura 3.23: Configuración del indicador **CW**.

De igual forma, los elementos del tipo **indicadores de luz piloto** siguen un proceso similar al asignar su lógica individual. Los parámetros de configuración específicos para cada uno de estos elementos se encuentran detallados en la Tabla 3.4.

Tabla 3.4: Configuración de los indicadores para luces piloto.

Luz piloto	**Proceso**	**Contenido**	
	Variable	**On**	**Off**
CW	Invertir_giro	PilotLight_Round_GN_Off_256c	PilotLight_Round_GN_On_256c
CCW	Invertir_giro	PilotLight_Round_GN_On_256c	PilotLight_Round_GN_Off_256c
Motor Habilitado:	Motor_habilitado	PilotLight_Square_G_On_256c	PilotLight_Square_G_Off_256c
Motor Bloqueado:	Motor_bloqueado	PilotLight_Square_R_On_256c	PilotLight_Square_R_Off_256c
Error:	Error	PilotLight_Square_RN_On_256c	PilotLight_Square_RN_Off_256c

En el indicador dial de **Velocidad**, se ingresa a **Propiedades** >**D5_Gauge_1** >**General**, se ubica en **Proceso**, en **Valor máximo de escala** se escribe el valor máximo al que puede alcanzar la velocidad del motor, y en **Valor mínimo de escala** se escribe el valor mínimo. En este caso se coloca 1500 y 0 respectivamente. Luego en **Variable de proceso** se escoge la variable **Lectura_velocidad_ABS**. Después, se ubica en **Titulo** y se escribe **Revolutions**, en **Unidad** se coloca **(rpm)** y en **Graduación de escala** se coloca **200**. Todo este proceso se visualiza en la Figura 3.24.

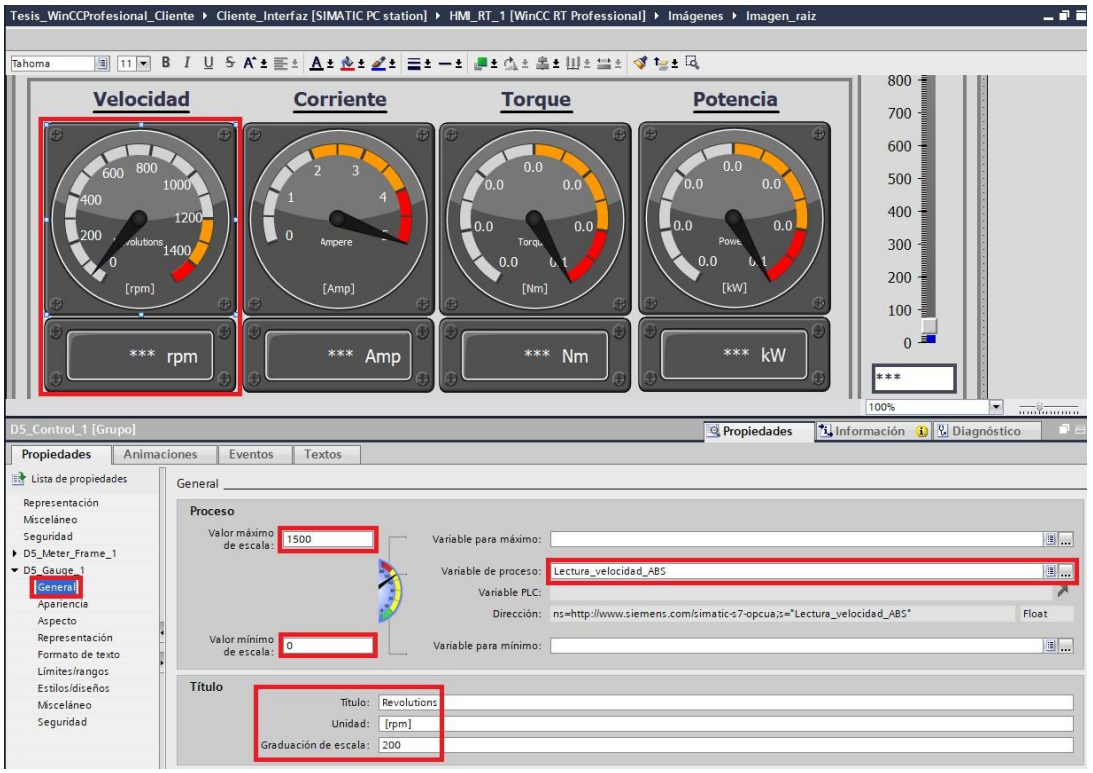


Figura 3.24: Configuración del indicador **Velocidad**.

Del mismo modo, los elementos clasificados como **indicadores de corriente, torque y potencia** siguen un proceso similar al asignar su lógica individual. Los parámetros de configuración específicos para cada uno de estos elementos se detallan en la Tabla 3.5.

Tabla 3.5: Configuración de los indicadores de velocidad, corriente, torque y potencia.

Dial	**Propiedades**					
	D5_Gauge					
	Proceso			**Titulo**		
	Valor maximo de escala	**Valor minimo de escala**	**Variable de proceso**	**Titulo**	**Unidad**	**Graduacion de la Escala**
Velocidad	1500	0	Lectura_velocidad_ABS	Revolutions	[rpm]	200
Corriente	5	0	Lectura_corriente	Ampere	[Amp]	1
Torque	0,06	0	Lectura_torque	Torque	[Nm]	0,01
Potencia	0,07	0	Lectura_potencia	Power	[kW]	0,02

En el deslizador **Consigna de Velocidad**, se ingresa a **Propiedades >General**, se ubica en **Proceso**. En **Valor máximo de escala** se escribe el valor máximo al que puede alcanzar la velocidad del motor, y en **Valor mínimo de escala** se escribe el valor mínimo, en este caso se coloca 1500 y 0 respectivamente. Luego en **Variable de proceso** se escoge la variable **Consigna_velocidad**, como se visualiza en la Figura 3.25.

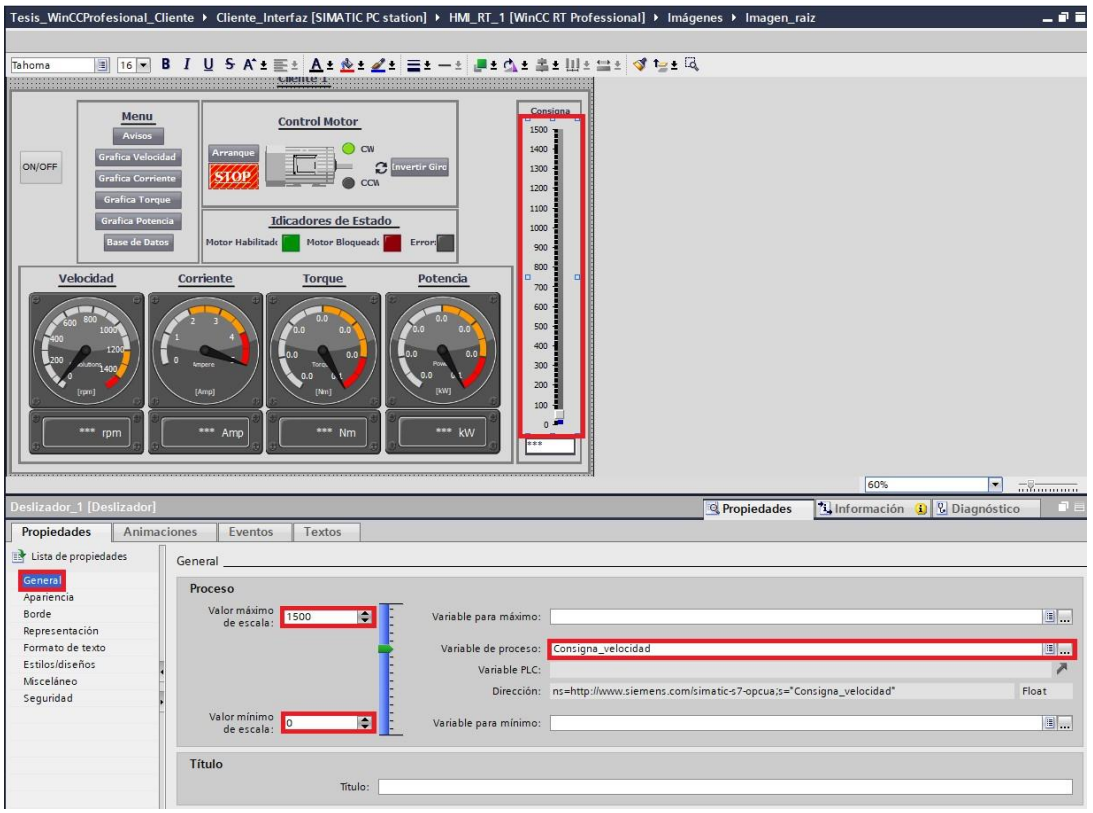


Figura 3.25: Configuración del deslizador **Consigna de Velocidad**.

Para configurar el campo **Consigna de Velocidad**, se accede a **Propiedades >General**. A contiuación se selecciona la opción **Proceso** y la variable **Consigna_velocidad**. Luego, en la sección **Tipo**, se selecciona el modo **Entrada/salida**, como se visualiza en la Figura 3.26.

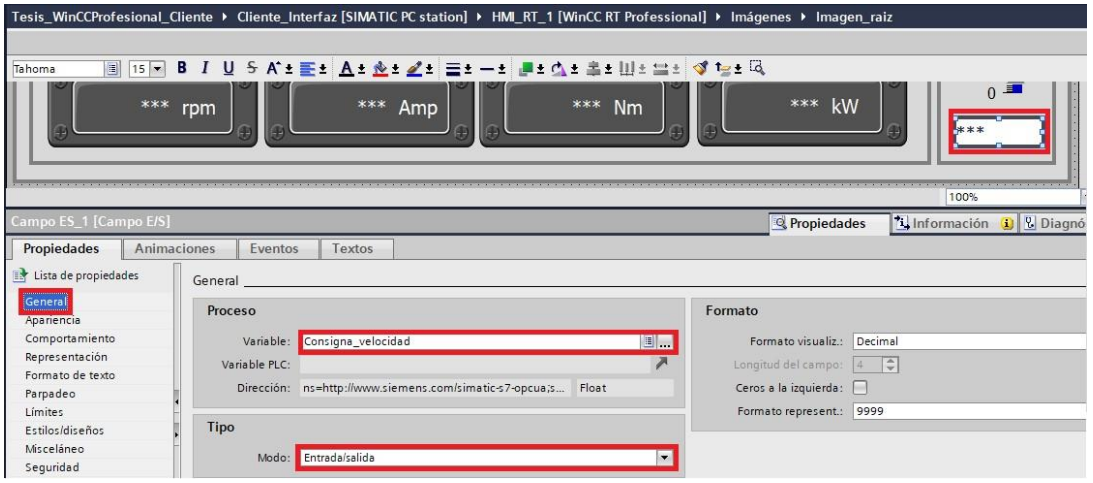


Figura 3.26: Configuración del campo **Consigna de Velocidad**.

Una vez realizadas las configuraciones para dar la lógica a todos los objetos de

la primera imagen del HMI (**Imagen_raiz**), se procede a realizar la configuración de los objetos de la imagen **T_Avisos**. Esta imagen esta compuesta por una tabla en la que se ve una lista con los errores y dos botones, el primer botón es para regresar a la **Imagen_raíz** y el segundo botón para acusar los fallos.

Primero se configura el **Visor de avisos**. Este se encuentra en:**Herramientas >Controles**. Para configurarlo, en el **Visor de avisos**, se ingresa a **Propiedades >General**, se ubica en **Visualización**, en **Tipo** se selecciona **Avisos pendientes**, en **Visualización** se selecciona **Avisos visibles**, y en **Doble clic** se selecciona **Acción depende de la columna** (véase la Figura 3.27).

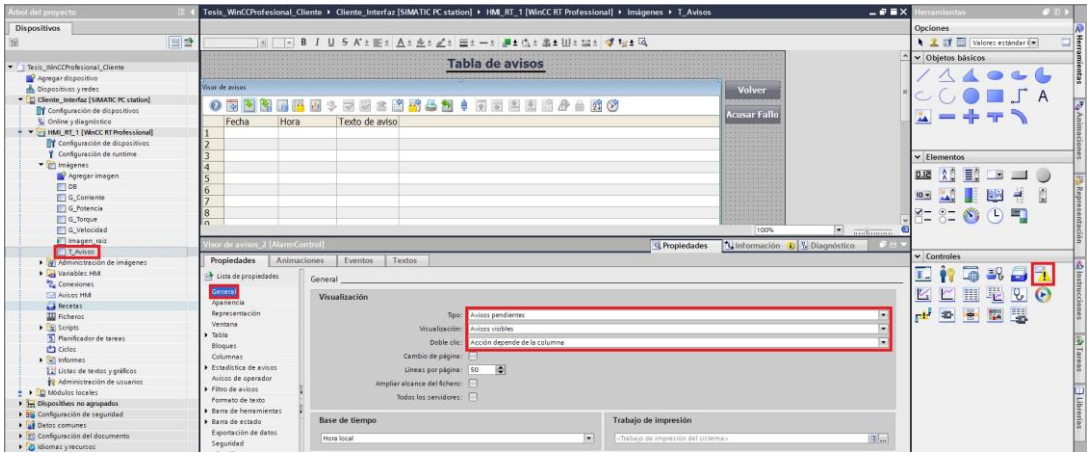


Figura 3.27: Configuración del **Visor de avisos**.

Después se configuran los botones. En el botón **Volver**, se ingresa a **Propiedades >Eventos**, se escoge la opción **Pulsar el botón izquierdo del ratón** y en **<Agregar función>** se selecciona **ActivarImagen**. Luego en **Nombre de imagen** se escoge la imagen **Imagen_raiz** y en **Numero de objeto** se coloca **0** (véase la Figura 3.28).

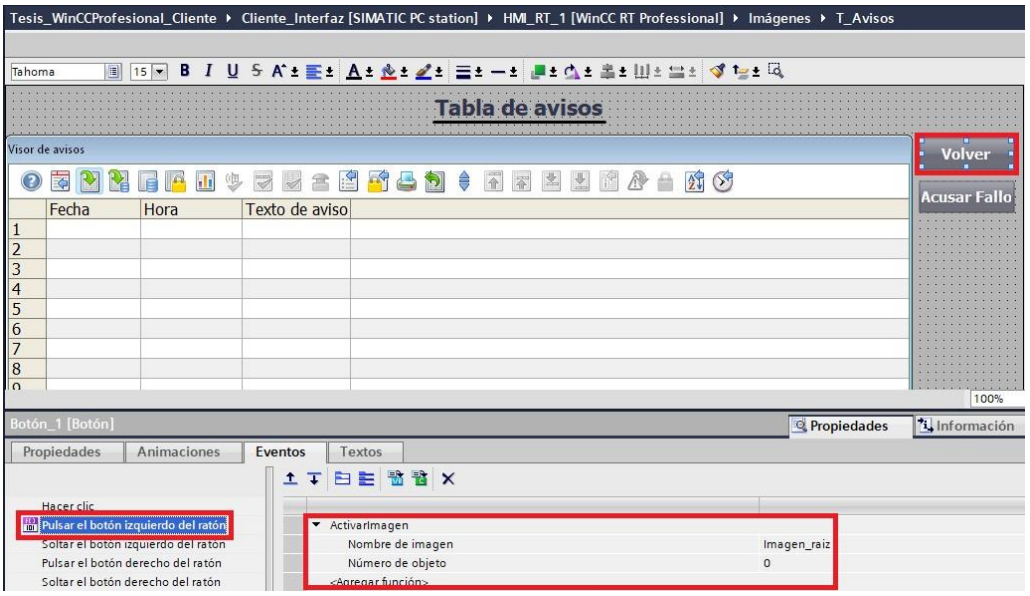


Figura 3.28: Configuración del botón **Volver** en la imagen **T_Avisos**.

En el botón **Acusar Fallo**, se ingresa a **Propiedades** >**Eventos**, se escoge la opción **Pulsar el botón izquierdo del ratón** y en **<Agregar función>** se selecciona **ActivarBit**. Luego en **Variable (Entrada/Salida)** se escoge la variable **ACK_Error**. Se repite lo mismo en la opción **Soltar el botón izquierdo del ratón** (véase la Figura 3.29).

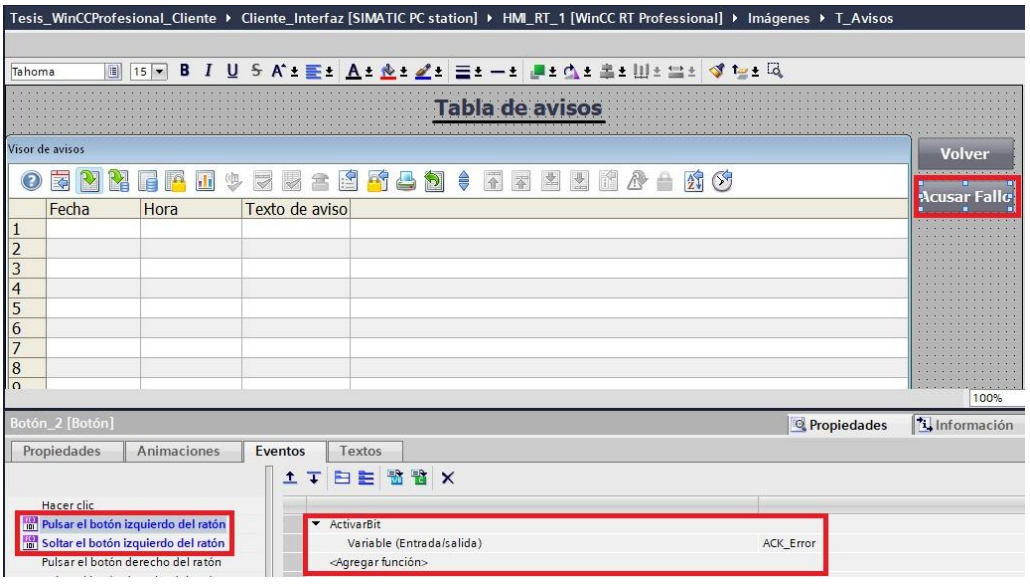


Figura 3.29: Configuración del botón **Acusar Fallo**.

En la Figura 3.32 se agregan los **Avisos HMI**. Es una funcionalidad de SIMATIC WinCC RT Professional, un sistema SCADA totalmente vinculado con el TIA Portal [14]. Los usuarios pueden utilizar la funcionalidad Avisos HMI para

mostrar mensajes y alarmas en la pantalla HMI, lo que les permite monitorizar el estado del proceso industrial en tiempo real. Esta función es extremadamente flexible y los usuarios pueden configurar mensajes y alertas para que se presenten en respuesta a determinadas condiciones o eventos [15].

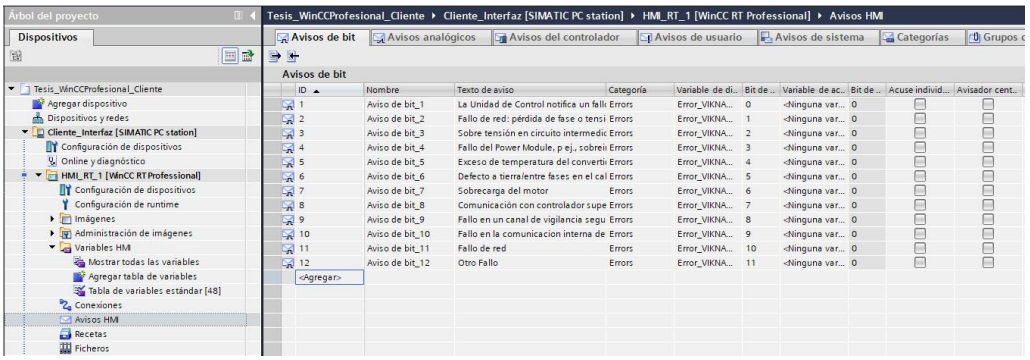


Figura 3.30: Creación **Avisos HMI**.

A continuación, se configura los objetos de las imágenes en las que se visualizan las gráficas de velocidad, corriente, torque y potencia. Cada una de estas imágenes cuentan con una gráfica, un botón para volver a la **Imagen_raíz**, un deslizador para cambiar la **consigna de velocidad** y un **indicador de inversión de giro**.

Para la imagen **G_Velocidad**, en la gráfica **Velocidad del Motor** se ingresa a **Propiedades >Curvas**, se ubica en la tabla **Curvas**, se agrega una curva y en **Nombre** se ingresa **Velocidad**, en **Origen de datos** se selecciona **[Lectura_velocidad(250 ms),(Variables)]** (véase la Figura 3.31). El botón **Volver** se configura de la misma forma en que se configuro este botón en la imagen **T_Avisos**.

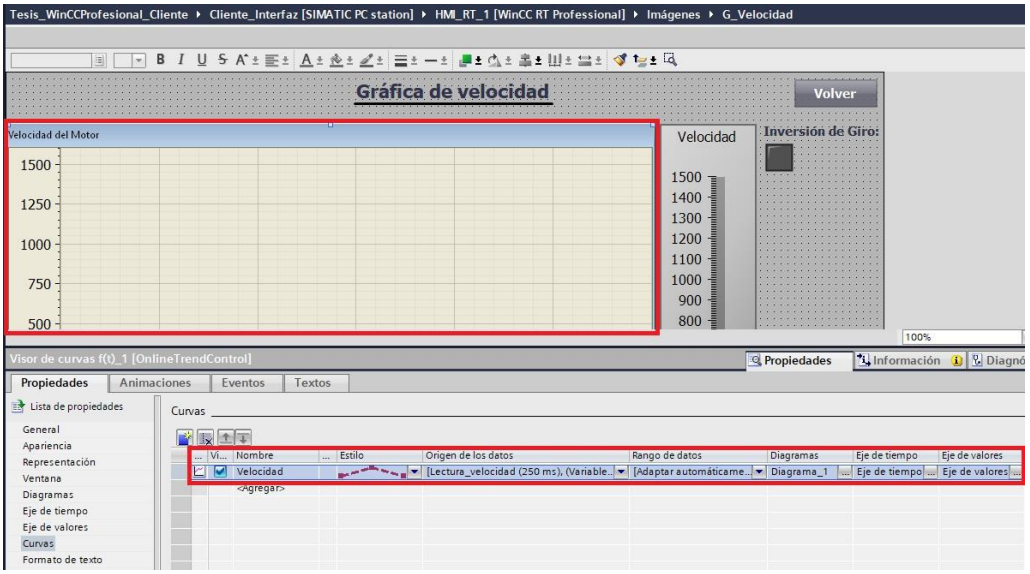


Figura 3.31: Configuración de la gráfica **Velocidad del Motor**.

Asimismo, los elementos del tipo **Visor de curvas** siguen un proceso similar al asignar su lógica individual. Los parámetros de configuración específicos para cada uno de estos elementos se detallan en la Tabla 3.6.

Tabla 3.6: Configuración de los visores de curvas de velocidad, corriente, torque, potencia.

Visor de curvas	**Propiedades**		
	Curvas		
	Nombre	**Estilo**	**Origen de los datos**
Gráfica de velocidad	Velocidad	Interpolada	[Lectura_velocidad (250 ms), (Variables)]
Gráfica de corriente	Corriente	Interpolada	[Lectura_corriente (250 ms), (Variables)]
Gráfica de torque	Torque	Interpolada	[Lectura_torque (250 ms), (Variables)]
Gráfica de potencia	Potencia	Interpolada	[Lectura_potencia (250 ms), (Variables)]

Por otro lado, en el contexto del proceso industrial, una herramienta clave es el **Planificador de Tareas**, una función disponible en SIMATIC WinCC RT Professional, el cual es un sistema SCADA completamente integrado en el TIA Portal. Dicha función, controlada por tiempo, permite a los usuarios programar una variedad de tareas, incluyendo el registro de datos, la gestión de alarmas y la generación de informes. Gracias a su versatilidad, los usuarios tienen la capacidad de personalizar las tareas para que se ejecuten en momentos o intervalos específicos, adaptándose a sus necesidades particulares [14]. Esto convierte al **Planificador de Tareas** en una poderosa herramienta para optimizar los procesos industriales y mejorar la eficiencia de los usuarios. Cabe destacar que esta función está disponible en las versiones de

WinCC Runtime Professional y WinCC Runtime Advanced del sistema.

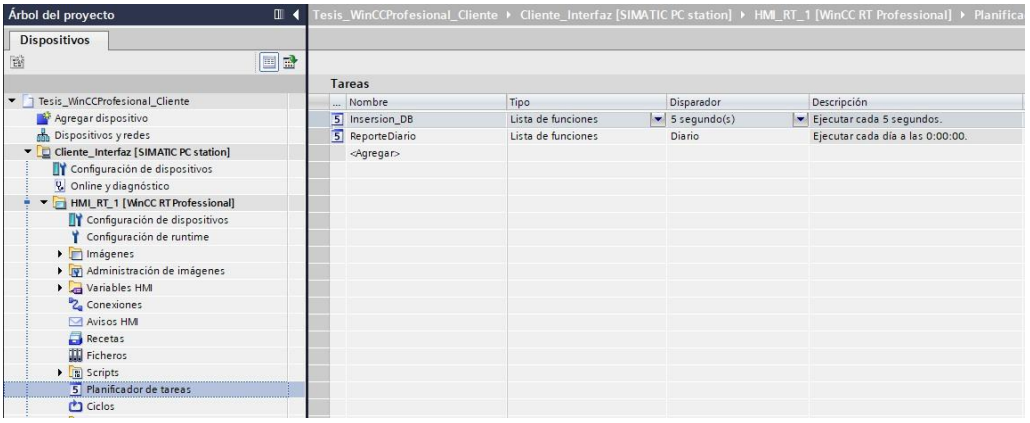


Figura 3.32: Configuración del **Planificador de tareas**.

3.4. Base de datos en un proceso industrial.

En un proceso industrial automatizado, una base de datos desempeña un papel crucial en el almacenamiento y gestión de los datos del proceso. Entre los aspectos más relevantes de una base de datos en la industria automatizada se encuentran el almacenamiento de datos en tiempo real e históricos relacionados con diversos aspectos del proceso, la eficiente recuperación y análisis de datos para la supervisión y toma de decisiones informadas, la visualización de datos en interfaces gráficas en tiempo real, el registro y auditoría de datos para cumplimiento y análisis posterior, y la integración con otros sistemas para lograr una visión holística del proceso y mejorar la coordinación y la toma de decisiones en diferentes departamentos [16].

Para crear una base de datos se requiere utilizar ODBC (Open Database Connectivity), que es una herramienta para configurar y gestionar conexiones a bases de datos ODBC. El Administrador de Origen de Datos ODBC se utiliza en SIMATIC WinCC RT Professional TIA Portal 15.v1 para establecer conexiones con bases de datos externas, como Microsoft SQL Server, que se emplean para almacenar y gestionar datos de procesos industriales [14].

Para configurar y gestionar las conexiones a bases de datos, WinCC RT Professional TIA Portal 15.v1 requiere la instalación de Microsoft SQL Server y del Administrador de fuentes de datos [14]. El Administrador de fuentes de datos ODBC

permite generar DNS (Data Source Names), que son identificadores utilizados para conectarse a bases de datos. Estos **DSN** incluyen la dirección del servidor, el nombre de la base de datos, el nombre de usuario y la contraseña necesaria para acceder a una base de datos [17]. En la figura 3.33 se puede observar como se agrega un nuevo usuario.

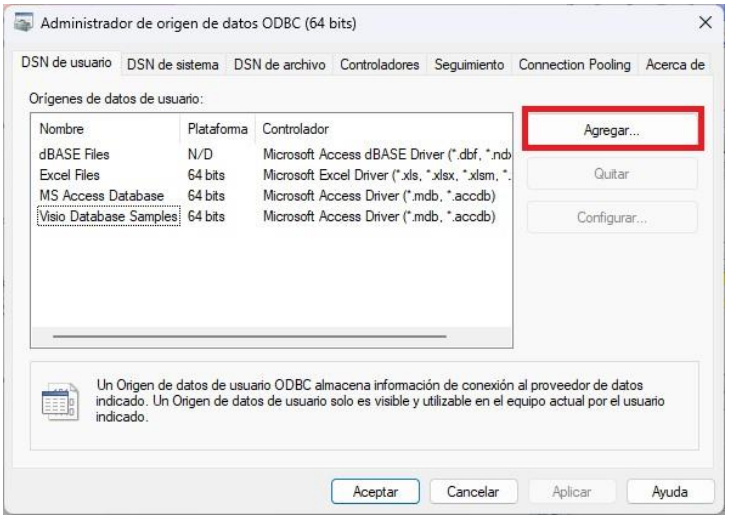


Figura 3.33: Nuevo **DNS** de usuario.

Luego, como se visualiza en la Figura 3.35 se selecciona el controlador para poder establecer un origen de datos. En este caso se selecciona **SQL Server**.

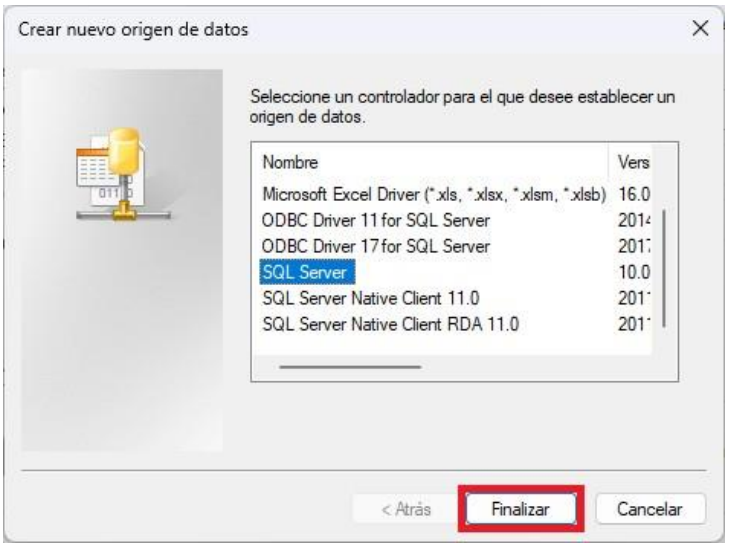


Figura 3.34: Selección del controlador para un nuevo de origen de datos.

En la Figura 3.35 se puede observar como se configura la **autenticidad de Id. de inicio de sesión**, para ello se selecciona la configuración predeterminada y se pulsa

en **Siguiente**.

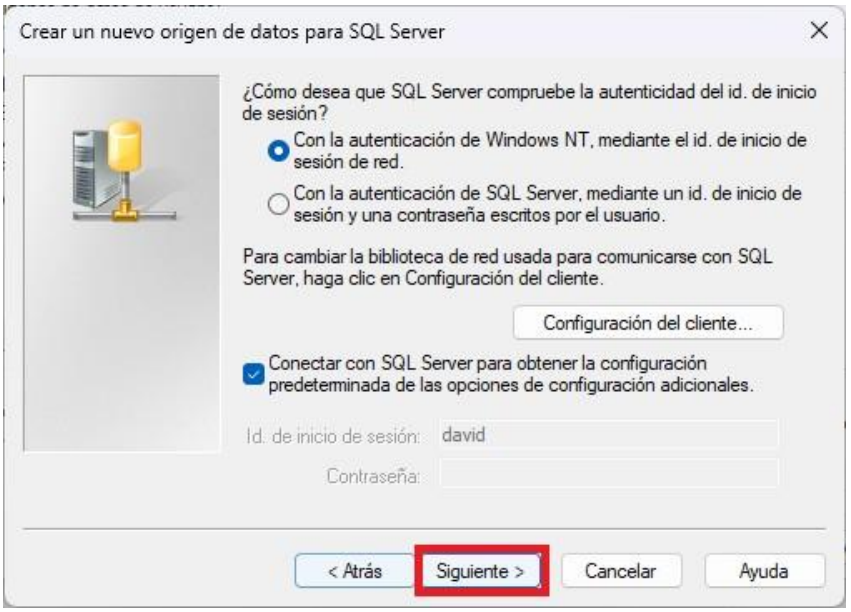


Figura 3.35: Configuración de seguridad para el servidor SQL.

Todas las otras configuraciones se deja en ajuste predeterminados y se pulsa en **Siguiente** como se indica en la Figura 3.36.

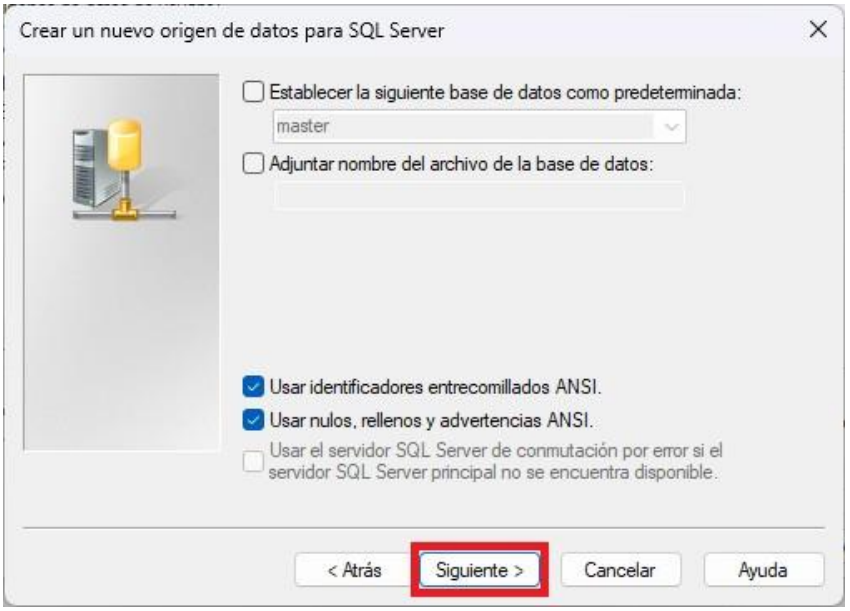


Figura 3.36: Selección de ubicación, acceso e identificadores de la base de datos.

En la ventana que se visualiza en la Figura 3.37, se escoge la opción de **Realizar conversión de lo datos de caracteres**. En un futuro se puede explorar las demás

opciones que se visualizan en esta ventana, una vez seleccionado los ajustes deseados se pulsa el botón **siguiente**.

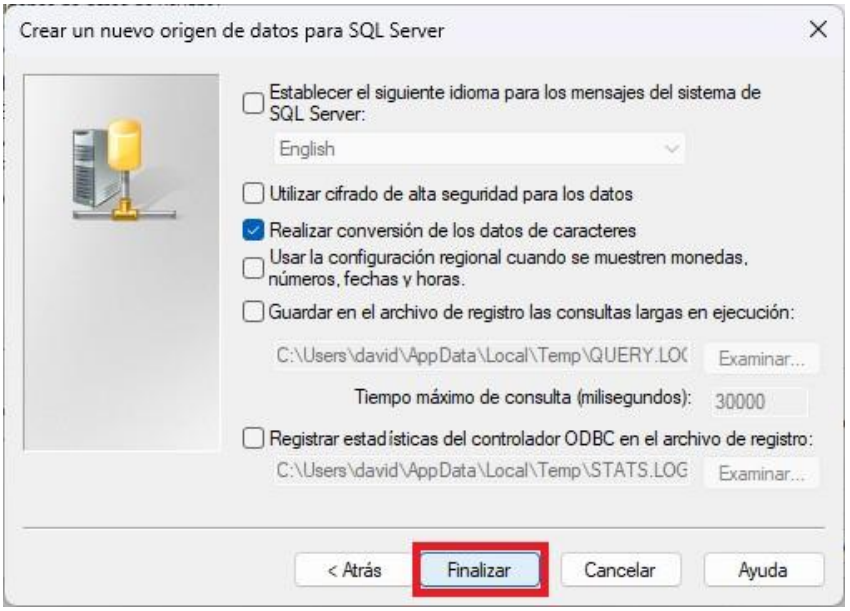


Figura 3.37: Establecer la conversión de los datos de caracteres.

Al finalizar la configuración, un nuevo origen de datos ODBC se presenta con un resumen de los ajustes seleccionados y además, permite probar si todo esta bien configurado y si se puede establecer un enlace. En la Figura 3.38 se visualiza la ventana de resumen y se selecciona la opción de **Probar origen de datos**.

Figura 3.38: Resumen de ajustes para el nuevo origen de datos de nombre **PLC**.

En la Figura 3.39 se observa que las pruebas realizadas han sido completadas correctamente. Por lo tanto se pulsa **Aceptar** y así se puede usar este DNS para conectarse a la base de datos a través de WinCC RT Professional.

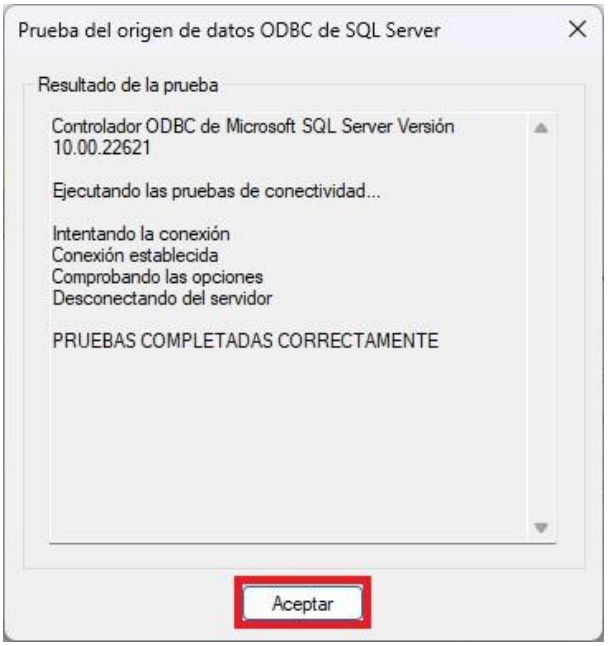


Figura 3.39: Resultado de prueba para el nuevo origen de datos de nombre **PLC**.

Para el desarrollo de la base de datos en WinCC RT Professional se llevó a cabo el desarrollo de scripts que cumplen dos objetivos esenciales: crear una base de datos y generar un informe para visualizar los datos del proceso industrial. Estas herramientas desempeñan un papel fundamental en la mejora de la gestión y comprensión de la producción industrial. La Figura 3.40 muestra las funciones a implementar. El lenguaje de programación **Script VBA** se emplea en SIMATIC WinCC RT Professional. Mediante esta función los usuarios pueden personalizar y automatizar las pantallas HMI, lo que les permite crear interfaces intuitivas y de fácil uso para la visualización y control de los procesos [18]. Dado que esta función es altamente personalizable, los usuarios tienen la capacidad de escribir scripts para realizar tareas específicas o automatizar procesos, adaptándolos según sus necesidades.

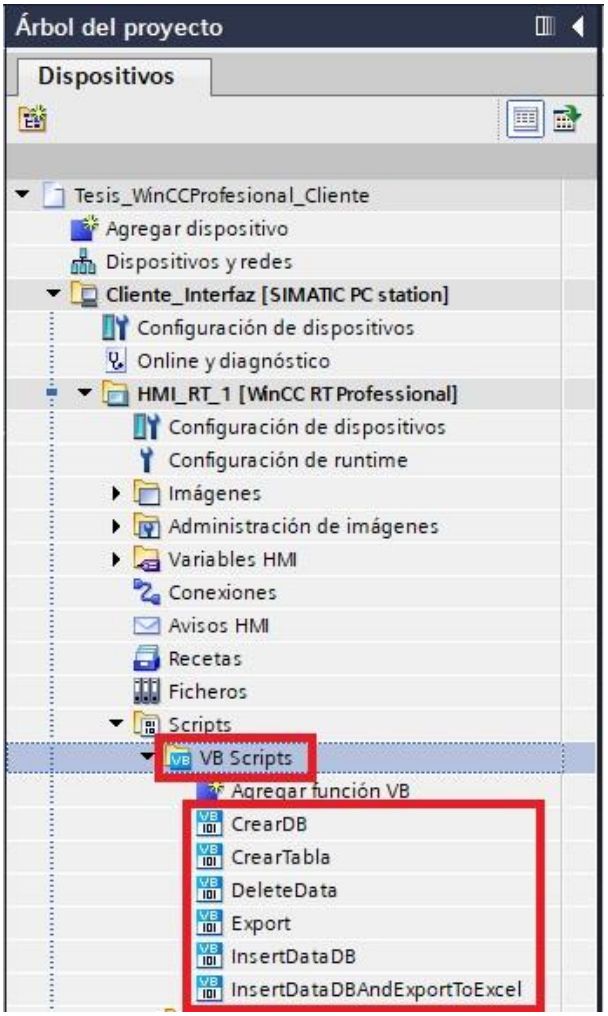


Figura 3.40: Creación de los Scripts necesarios para la base de datos.

El Algoritmo 1 define una función llamada **CrearDB** que tiene como objetivo crear una base de datos utilizando el objeto **ADODB.Connection** en un entorno de desarrollo o sistema de control industrial. El algoritmo intenta abrir una conexión a la base de datos especificada por el DSN **PLC**. Si la conexión se establece con éxito, ejecuta una consulta SQL para crear una nueva base de datos llamada **DB_1**. Si alguna de las operaciones falla, se captura el error y se registra en una etiqueta **ErrDesc** para su posterior manejo. Sin embargo, el código tiene algunos problemas, como el uso incorrecto del parámetro **Name_DSN**, la falta de uso del valor almacenado en **DBName**, y el manejo poco seguro de errores con **On Error Resume Next**. Estos problemas deben corregirse para asegurar un funcionamiento adecuado del algoritmo.

Algoritmo 1 Crear base de datos

```
Sub CrearDB(ByRef Name_DSN)
    ' Declaración de variables
    Dim connect, record, DBName, ErrorDesc, SQLTable

    ' Obtener el nombre de la base de datos desde los SmartTags
    DBName = SmartTags("DataBaseName")

    ' Permite continuar la ejecución en caso de errores
    On Error Resume Next

    ' Crear la conexión a la base de datos
    Set connect = CreateObject("ADODB.Connection")
    Set record = CreateObject("ADODB.Recordset")
    connect.Open "Provider=MSDASQL;DSN=PLC"

    ' Verificar si ocurrió algún error al ejecutar el comando SQL
    If Err.Number <> 0 Then
        ErrorDesc = Err.Description & " (" & Err.Number & ")"
        SmartTags("ErrDesc") = ErrorDesc
        Set connect = Nothing ' Liberar los recursos de la conexión
        Err.Clear
        Exit Sub
    End If

    ' Definir el comando SQL para crear la base de datos
    SQLTable = "CREATE DATABASE " & DBName

    ' Ejecutar el comando SQL para crear la base de datos
    Set record = connect.Execute(SQLTable)

    ' Verificar si ocurrió algún error al ejecutar el comando SQL
    If Err.Number <> 0 Then
        ErrorDesc = Err.Description & " (" & Err.Number & ")"
        SmartTags("ErrDesc") = ErrorDesc
        Set connect = Nothing ' Liberar los recursos de la conexión
        Err.Clear
        Exit Sub
    End If

    connect.Close ' Cerrar la conexión a la base de datos
    Set connect = Nothing ' Liberar los objetos
    Set record = Nothing

End Sub
```

Posteriormente, se debe definir el evento con el que se ejecutará la función relacionada para cada botón. Por lo tanto, para configurar el botón **Crear DB**, es necesario posicionar el cursor sobre él y luego seleccionar la pestaña **Eventos** en el menú inferior. En la opción **Pulsar el botón izquierdo del ratón**, se vincula la función **CrearDB**. La Figura 3.41 muestra el proceso descrito. De esta manera vinculamos una función que se hizo por Script a la aplicación HMI, el cual abre mas posibilidades para realizar una operación, control o consulta de datos en un proceso industrial.

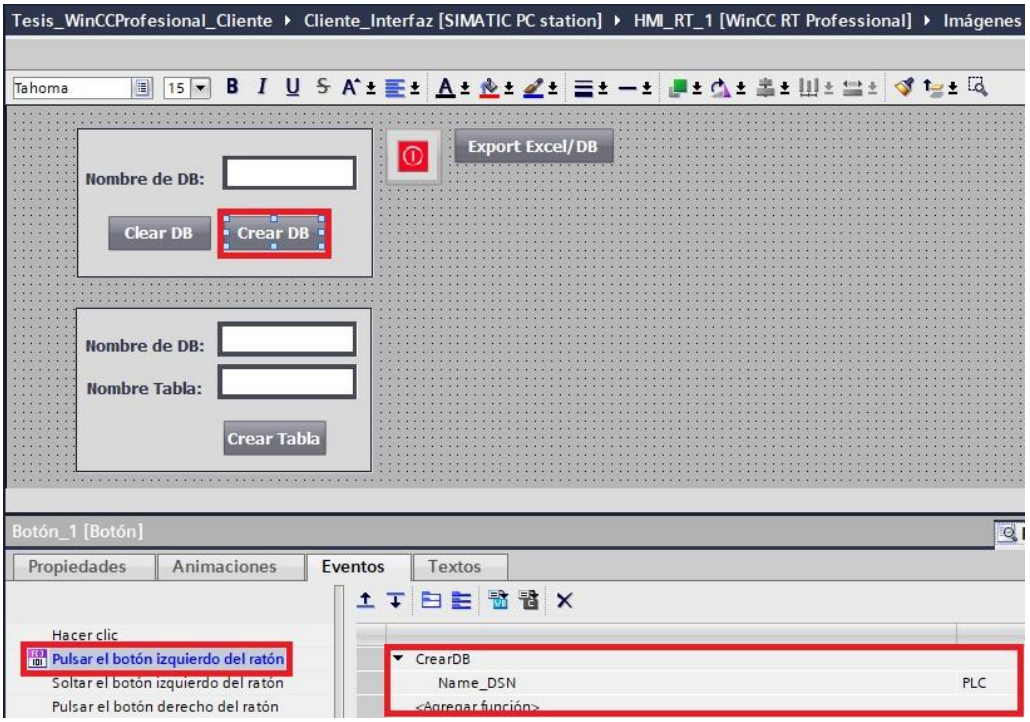


Figura 3.41: Configuración del botón **Crear DB**.

En el contexto de una base de datos, resulta imprescindible agregar tablas para almacenar datos. Los Algoritmos 2 y 3 representan una función conjunta que posibilita la creación de una tabla para alojar información y guardar los datos procedentes del motor, suministrados por el variador a través del Telegrama 20. Esta función, denominada **CrearTabla**, tiene como propósito la generación de una nueva tabla en la base de datos mediante el empleo del objeto **ADODB.Connection** en un entorno de desarrollo o sistema de control industrial.

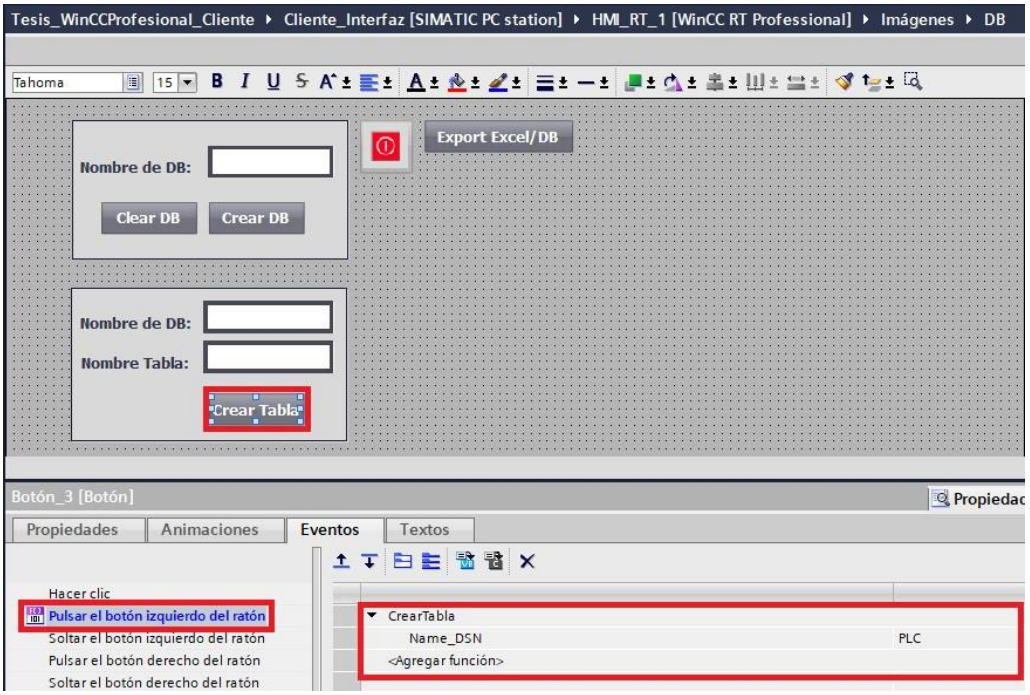


Figura 3.42: Configuración del botón **Crear Tabla**.

Algoritmo 2 Crear tabla: parte 1

```
Sub CrearTabla(ByRef Name_DSN)
    ' Declaración de variables
    Dim connect, record, DBName, TableName, SQLtable, ErrorDesc
    Dim Column1, Column2, Column3, Column4, stopData

    ' Obtener valores de las SmartTags
    DBName = SmartTags("DataBaseName")
    TableName = SmartTags("TableName")
    Column1 = SmartTags("Column1")
    Column2 = SmartTags("Column2")
    Column3 = SmartTags("Column3")
    Column4 = SmartTags("Column4")

    ' Ignorar errores y continuar
    On Error Resume Next

    ' Crear objetos de conexión y recordset
    Set connect = CreateObject("ADODB.Connection")
    Set record = CreateObject("ADODB.Recordset")

    ' Abrir la conexión a la base de datos
    connect.Open = "Provider=MSDASQL;Initial Catalog=DB_1;DSN=PLC"
```

El algoritmo recaba valores desde etiquetas inteligentes, denominadas **SmartTags**, las cuales contienen información relevante, tales como el nombre de la base de datos **DBName**, el nombre de la tabla **TableName** y los nombres de las columnas **Column1, Column2, Column3, Column4**. A continuación, se procede a intentar establecer una conexión con la base de datos, especificada por el DSN **PLC**, y se crea una tabla con una estructura predefinida que incluye columnas para la fecha y hora, velocidad, corriente, torque y potencia.

Algoritmo 3 Crear tabla: parte 2

```
    ' Verificar si ocurrió algún error al abrir la conexión
    If Err.Number <> 0 Then
        ErrorDesc = Err.Description & "(" & Err.Number & ")"
        SmartTags("ErrDesc") = ErrorDesc
        Set connect = Nothing
        Err.Clear
        Exit Sub
    End If

    ' Definir el comando SQL para crear la tabla
    SQLtable = "CREATE TABLE valores_PLC(fecha_hora datetime
                DEFAULT GETDATE(), Velocidad FLOAT, Corriente
                FLOAT, Torque FLOAT, Potencia FLOAT)"

    ' Ejecutar el comando SQL para crear la tabla
    Set record = connect.Execute(SQLtable)

    ' Verificar si ocurrió algún error al ejecutar el comando SQL
    If Err.Number <> 0 Then
        ErrorDesc = Err.Description & "(" & Err.Number & ")"
        SmartTags("ErrDesc") = ErrorDesc
        Set connect = Nothing
        Err.Clear
        Exit Sub
    End If

    ' Cerrar la conexión a la base de datos
    connect.Close

    ' Liberar los objetos
    Set connect = Nothing
    Set record = Nothing

End Sub
```

Cabe destacar que el algoritmo incluye un mecanismo de manejo de errores, para capturar cualquier eventualidad que pueda surgir, registrando dichos errores en la etiqueta **ErrDesc**, con el fin de abordarlos adecuadamente en un momento posterior. Finalmente, se concluye cerrando la conexión y liberando los objetos utilizados. No obstante, se recomienda una revisión y mejora del código para evitar posibles errores y garantizar un funcionamiento correcto y seguro.

Para la configuración del botón **Crear Tabla**. En esta ocasión, el evento **Pulsar el botón izquierdo del ratón** se enlaza con la función **CrearTabla** (consulte la Figura 3.42).

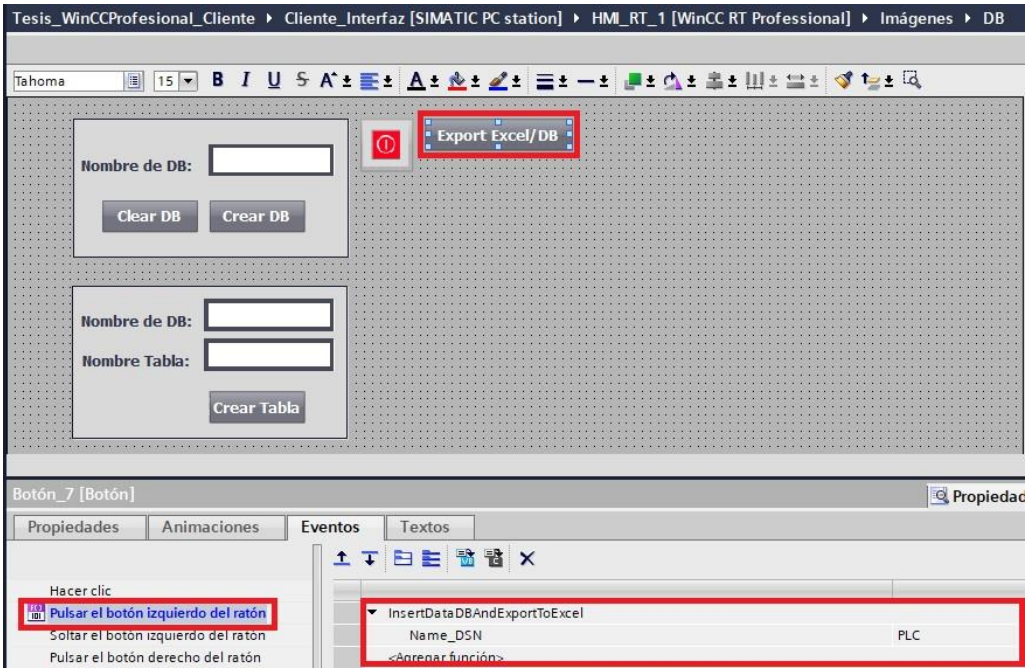


Figura 3.43: Configuración del botón **Export Excel/DB**.

Para generar los reportes se lo realiza a través de el botón **Exportar Excel/DB**. Este botón al igual que los anteriores que están ligados a un evento que accede a la función **InsertDataDBAndExportToExcel**, cuyo parámetro de ingreso es el nombre del DNS **PLC** (véase la Figura 3.44). Los algoritmos (8, 9 y 10) definen una subrutina llamada **InsertDataDBAndExportToExcel** que tiene como objetivo extraer datos de una tabla llamada **valores_PLC** en una base de datos **DB_1** utilizando una conexión **ADO** a través del DSN **PLC**. Luego, crea una instancia de Microsoft Excel, abre un archivo de plantilla, copia los datos extraídos de la tabla en la hoja de cálculo

y guarda el archivo con un nombre basado en la fecha y hora actual. El código también maneja errores y registra mensajes de error en caso de que ocurran problemas en la conexión o la consulta a la base de datos. Sin embargo, es importante tener en cuenta que hay algunos errores menores, como la declaración de variables no utilizadas (**excelWorkbook**, **yearStr**, **monthStr**, etc.) y una falta de declaración para la variable **minStr**. Estos detalles deben ser corregidos para asegurar un funcionamiento adecuado del algoritmo.

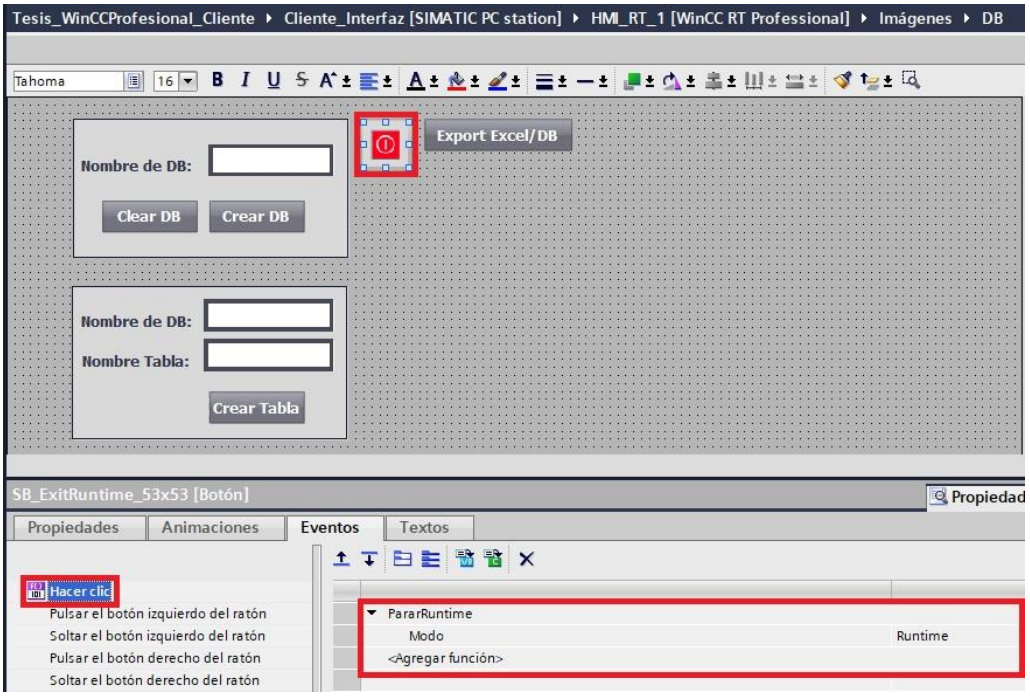


Figura 3.44: Configuración del botón .

3.5. Pruebas y resultados

Al realizar las pruebas del sistema de comunicación implementado en el proceso industrial de controlar y monitorear un motor, se pudo observar como este funcionó de la manera que se esperaba. Para probar la arquitectura se utilizaron tres clientes. Con el primer cliente, se probó el funcionamiento de las curvas y de los avisos cuando hay error. Primero se probó que se pueda cambiar la velocidad del motor con el HMI **Cliente 1**. Se comenzó con una velocidad baja que permite ver como los indicadores muestran los valores actuales de velocidad, corriente, torque y potencia. Además se comprobó que los indicadores de estado y de sentido de giro funcionen

correctamente al encenderse solo los indicadores de **Motor habilitado** y **CW**, lo que permiten al usuario saber que el motor se encuentra encendido y girando en sentido horario. (véase la figura 3.45).

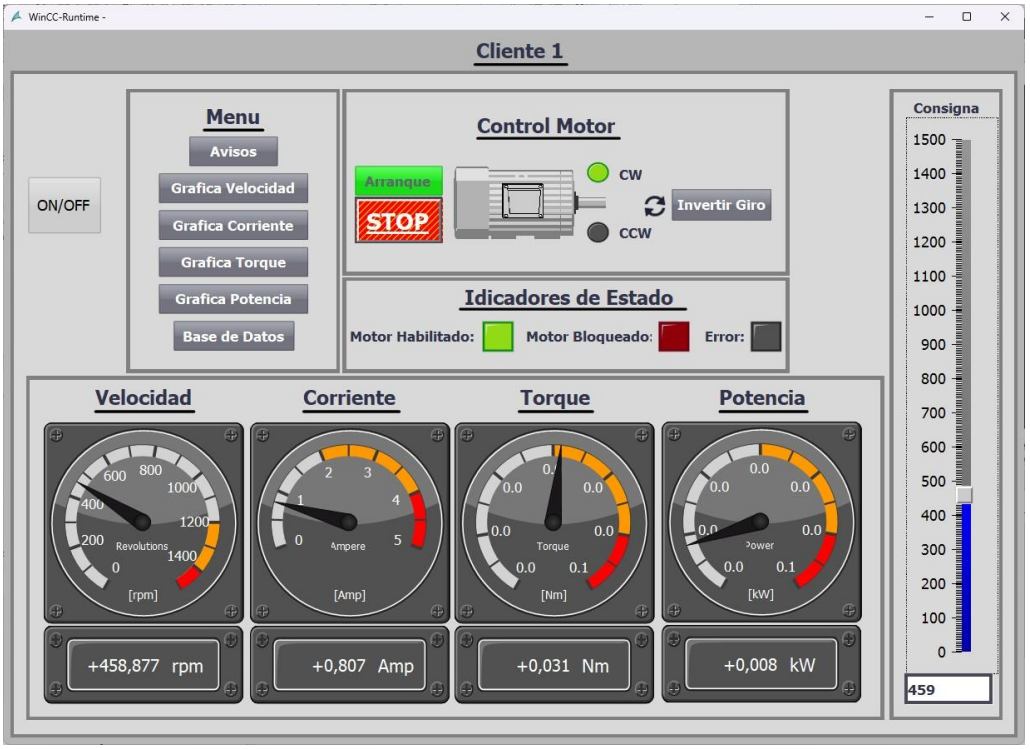


Figura 3.45: Prueba de funcionamiento HMI **Cliente 1** con una velocidad baja y el giro en sentido horario.

luego se incrementó la velocidad para asegurarse que los indicadores de velocidad, corriente, torque y potencia, sigan de forma correcta los cambios que se dan en el motor, además se cambio de sentido de giro para asegurarse que este botón funcione y que el indicador **CCW** se encienda cuando se da este cambio. Todo funcionó como se esperaba (véase la figura 3.46).

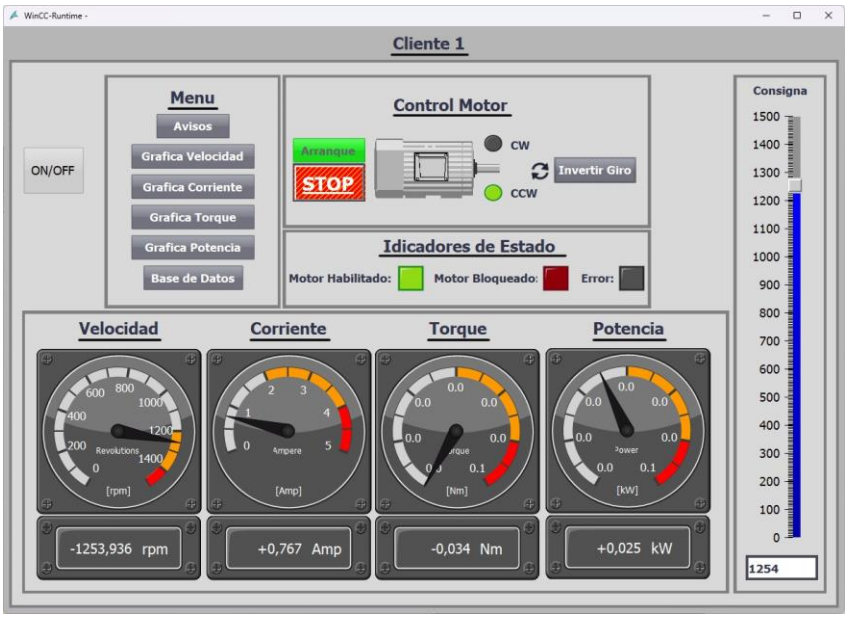


Figura 3.46: Prueba de funcionamiento HMI **Cliente 1** con una velocidad alta y el giro en sentido antihorario.

Lo siguiente fue comprobar el funcionamiento de las gráficas de las curvas de velocidad, corriente, torque y potencia. Para esto al estar en la imagen **G_Velocidad**, se movió el deslizador para cambiar la consiga de velocidad y generar cambios en la velocidad del motor, lo que ocasionó que se den cambios en la curva de la gráfica, mostrando que funciono correctamente (véase la figura 3.47).

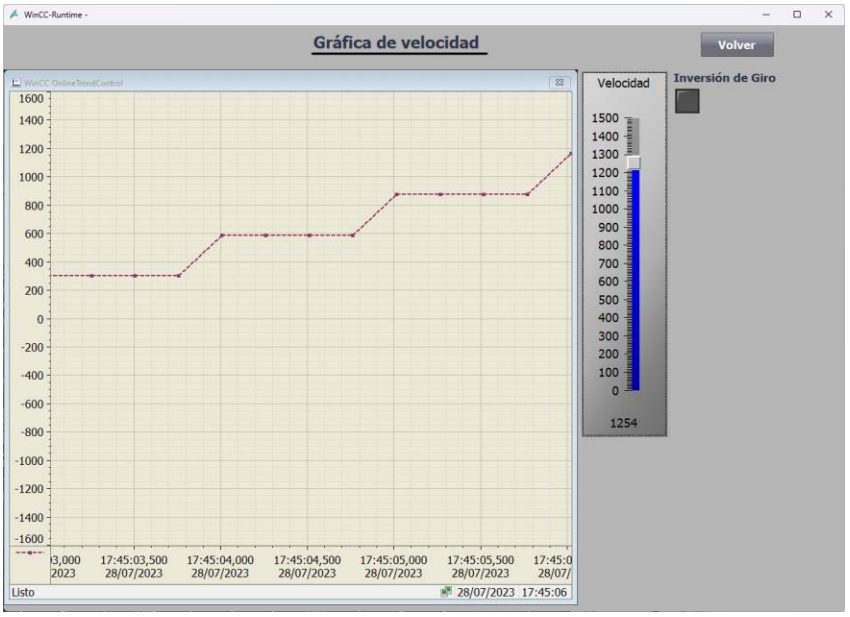


Figura 3.47: Prueba de funcionamiento del HMI - **Gráfica de velocidad**.

Después se realizó lo mismo con la imagen **G_Corriente**, con lo que se logró ver que la corriente del motor se mantiene constante (véase la figura 3.48).

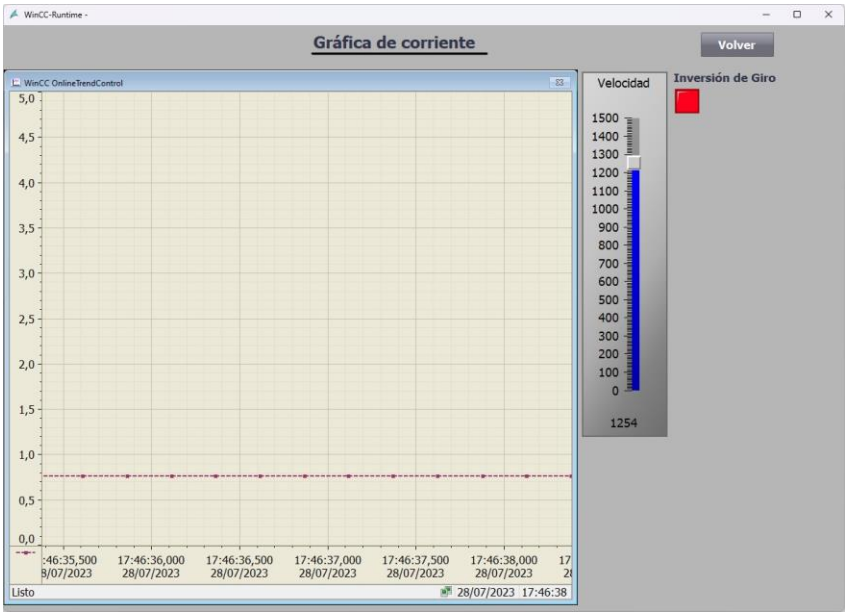


Figura 3.48: Prueba de funcionamiento HMI **Gráfica de corriente**.

Se realizó una comprobación adicional mediante la gráfica de torque. Durante la observación en la imagen **G_Torque**, se generaron cambios en la velocidad, lo que permitió confirmar su correcto funcionamiento y comportamiento esperado. Dado que no se aplicó una carga significativa al eje del motor, al cambiar las velocidades o invertir el sentido de giro, el torque mostró valores muy cercanos a cero (véase la Figura 3.49). Debido a esta característica, el uso de indicadores o gráficas para ilustrar este aspecto mediante un visor de curvas resulta poco conveniente.

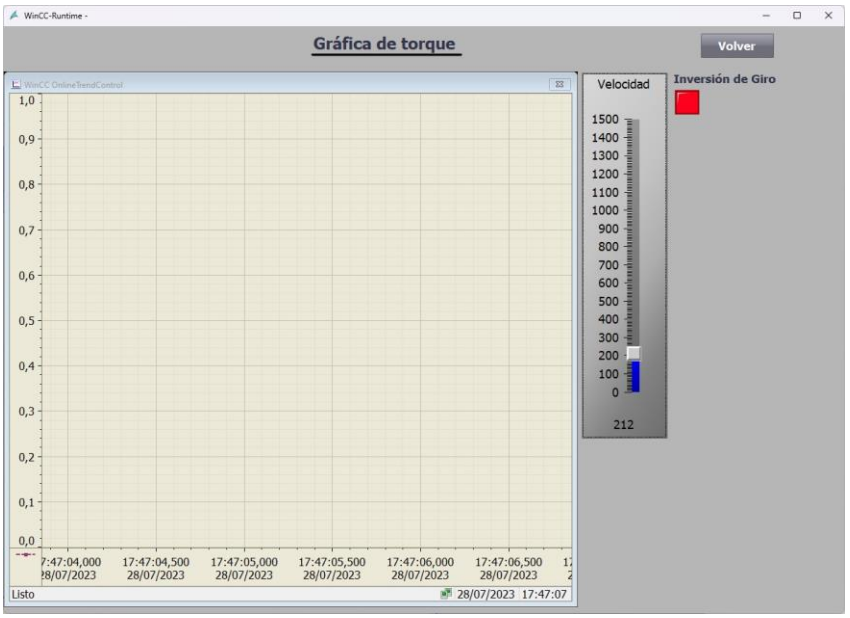


Figura 3.49: Prueba de funcionamiento del HMI - **Gráfica de torque**.

Finalizando la comprobación de las gráficas, al analizar la imagen **G_Potencia**, se llevó a cabo el mismo proceso que en los casos anteriores. Se obtuvo un resultado muy similar al caso del torque, y esto se debe a que el torque y la potencia de un motor de inducción están estrechamente relacionados [19]. Ambos son parámetros fundamentales que describen el rendimiento del motor y están influenciados por sus características físicas y eléctricas (véase la Figura 3.50).

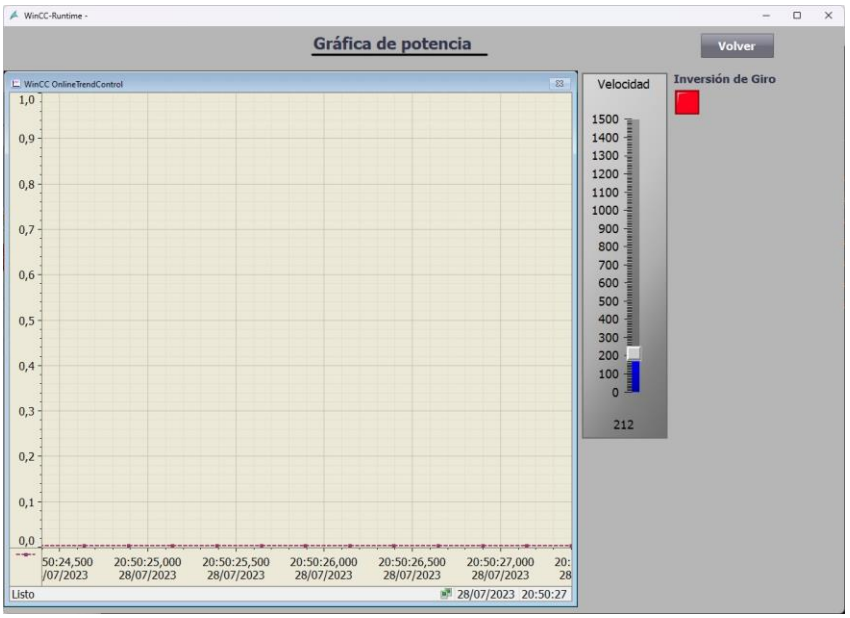


Figura 3.50: Prueba de funcionamiento del HMI - **Gráfica de Potencia**.

Tras verificar todas las gráficas, se procedió a evaluar la generación adecuada de advertencias de error en la tabla de avisos. Para ello, se desconectó la fuente de alimentación del motor de manera abrupta. Como resultado de esta acción, la tabla de avisos instantáneamente mostró un error relacionado con la alimentación del motor (ver Figura 3.51).

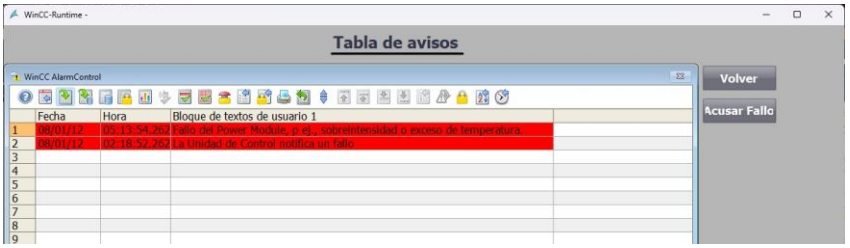


Figura 3.51: Prueba de funcionamiento del HMI - **Tabla de avisos** con errores.

Posteriormente, se hizo clic en el botón **Acusar Fallo** para confirmar que el error fuera notificado y que la tabla se limpiara, indicando que el fallo había sido reconocido (ver Figura 3.52). Además, se generaron otros errores para comprobar que la tabla de avisos los mostrara adecuadamente y que, de igual forma, los acusara. Todo el proceso funcionó correctamente.

Figura 3.52: Prueba de funcionamiento del HMI - **Tabla de avisos** >botón **Acusar Fallo**.

Luego de realizar todas las pruebas en el **Cliente 1**, se procedió a realizar las mismas pruebas con el HMI del **Cliente 2** (ver Figura 3.53). Todas las pruebas demostraron un correcto funcionamiento del sistema.

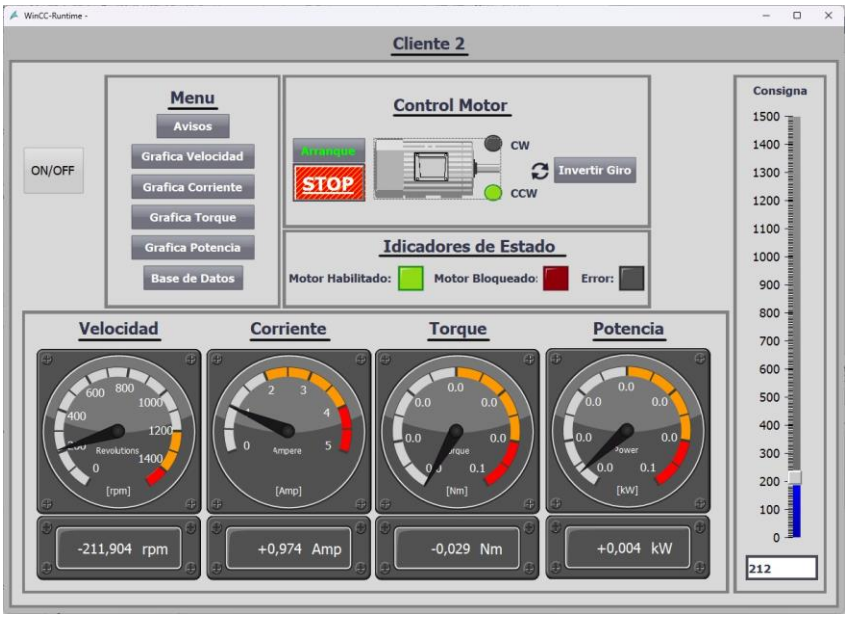


Figura 3.53: Prueba de funcionamiento del HMI - **Cliente 2**.

Una vez terminadas las pruebas en el **Cliente 2** se procedió a realizar las mismas con el HMI del **Cliente 3**. En todas las pruebas se tuvo el resultado esperado.

La Figura 3.54 representa un escenario en el cual la aplicación del HMI no se encuentra conectada al servidor de la arquitecturaOPC UA. Esta situación puede deberse a que la HMI se encuentre fuera de la red o a una configuración errónea tanto en el servidor como en el cliente. Para restaurar la comunicación y garantizar el correcto funcionamiento del sistema, es crucial identificar y corregir las posibles causas que han generado la falta de conexión. Un análisis exhaustivo de la infraestructura y ajustes relevantes permitirá solucionar este problema y asegurar la disponibilidad adecuada de la aplicación.

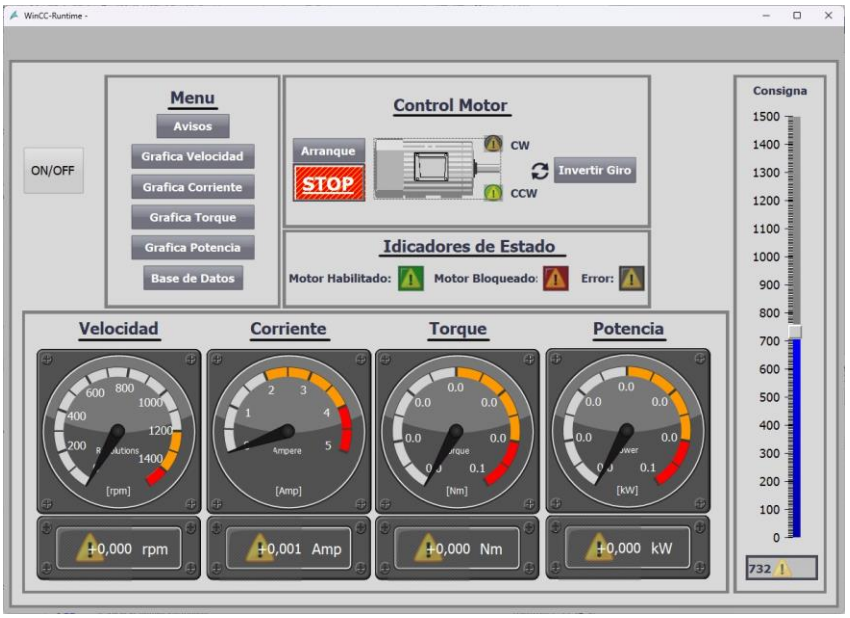


Figura 3.54: Prueba de funcionamiento HMI **Imagen_raíz** con errores.

La última prueba que se realizó fue la creación de una base datos. Para esto antes de prender el motor se creó una base de datos en la imagen **DB**, para que, mientras el motor esta prendido y se cambie las velocidades, se recopilen los datos del motor. Después de un tiempo de tener al motor funcionado, se procedió a exportar la base de datos a Excel, obteniendo la tabla que se esperaba (véase la figura 3.55).

20/7/2023

REGISTRO DE DATOS DEL PROCESO PRODUCTIVO (FROM SQL SERVER)

UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR

BY: Jose David Abril Vera; Renata Fransisca Guillen Sarmiento

Fecha & Hora (dd:mm:yy / HH:MM)	Velocidad (RPM)	Corriente (A)	Torque (Nm)	Potencia (kW)
20/7/2023	0	0.001208496	0	0
20/7/2023	0	0.001208496	0	0
20/7/2023	0	0.001208496	0	0
20/7/2023	0	0.001208496	0	0
20/7/2023	0	0.001208496	0	0
20/7/2023	512.9297	0.8048584	0.03622131	0.01091003
20/7/2023	512.9297	0.8000244	0.03578491	0.01077576
20/7/2023	512.9297	0.8001587	0.03511047	0.01057434
20/7/2023	512.9297	0.8004273	0.03459473	0.01040649
20/7/2023	512.9297	0.7990845	0.03471375	0.01044006
20/7/2023	512.9297	0.7996216	0.03415832	0.01030579
20/7/2023	512.9297	0.8004273	0.03451538	0.01040649
20/7/2023	512.9297	0.7990845	0.03384094	0.01020508
20/7/2023	512.9297	0.8004273	0.03403931	0.01023865
20/7/2023	512.9297	0.800293	0.03415832	0.01027222
20/7/2023	512.9297	0.7989502	0.03380127	0.01017151

Figura 3.55: Prueba de funcionamiento de base de datos

Capítulo 4

Conclusiones y Trabajos Futuros

Conclusiones

OPC UA es un protocolo esencial en la automatización industrial. Se destaca por su capacidad de integrar autómatas S7-1500 de SIEMENS y estaciones de monitoreo basadas en WINCC RT mediante diversos modos de configuración. Un tutorial relevante es el OPC UA Server Communication en Siemens TIA Portal, que detalla cómo configurar un S7-1500 como servidor OPC UA, resaltando la seguridad y el completo modelo de información que ofrece en comparación con protocolos tradicionales. También, se encuentra el S7-1500 Communication Manual, que proporciona información sobre el servidor OPC UA de las CPUs S7-1500, abarcando la comunicación segura OUC hacia PLCs externos. Asimismo, el WinCC Engineering V15.1 - Communication explica cómo configurar un dispositivo HMI como servidor OPC UA en el software WinCC Engineering. En resumen, OPC UA brinda modos de configuración que permiten una integración segura y eficiente entre autómatas S7-1500 y estaciones de monitoreo WINCC RT, facilitando el intercambio de datos en un entorno industrial.

Para poder generar las variables que se van a monitorear en un proceso industrial, es importante estudiar los equipos que se están usando dentro de este, como en el caso de este trabajo, que se utilizó un variador de frecuencia SINAMICS G120. Para saber que variables se utilizarían en el control de un motor por medio de este variador, se estudió las características de este variador y se analizó los diferentes

telegramas que este tiene, con lo que se descubrió que se puede hacer uso del telegrama 20, el telegrama mas completo en lo que se refiere al control y monitoreo de un motor. Este telegrama fue el que permitió generar las variables al relacionarlas con los parámetros que este telegrama permite al usuario cambiar en un motor.

Al momento de crear un HMI con WINCC RT, si se requiere realizar un HMI más complejo y con más opciones es recomendable ocupar la versión Professional, además es de suma importancia aprender sobre las distintas herramientas que este contiene, puesto que, al tener una gran variedad, puede ser que unas herramientas funcionen mejor para lo que se desea desarrollar, además de que algunas herramientas vienen con la lógica ya programada, logrando que sea más fácil el diseñar el HMI.

Para tener el mejor resultado posible al realizar la integración de las estaciones HMI y el controlador industrial mediante la tecnología de comunicación OPC UA, es necesario establecer una buena configuración para las conexiones OPC UA, tanto para los clientes como para el servidor, puesto que, solo se debe realizar una configuración para crear multi estaciones. Esta es la ventaja que da el usar OPC UA. Al servidor solo se le configura ciertos ajustes, mientras que el cliente, logra su conexion por medio de WINCC RT, el cual usa PROFINET para conectarse con el servidor, esta configuración se puede hacer en varios clientes permitiendo crear multi estaciones. Es de suma importancia habilitar la comunicación PROFINET en el controlador industrial para que todo este sistema de comunicacion funcione correctamente.

Trabajos Futuros

En el ámbito de la integración de autómatas S7-1500 y estaciones de monitoreo WINCC RT mediante el protocolo OPC UA, los trabajos futuros se enfocarían en mejorar la seguridad de las comunicaciones mediante la implementación de técnicas avanzadas de encriptación y autenticación, así como en optimizar la eficiencia de la transmisión de datos entre el servidor OPC UA y los clientes WINCC RT mediante estrategias de compresión y caché. Además, se explorarían alternativas para la generación de variables de monitoreo, considerando telegramas distintos al telegrama 20 del variador SINAMICS G120 y el uso de algoritmos de análisis avanzado para

detección de patrones. Asimismo, se buscaría desarrollar herramientas más intuitivas para la configuración y programación de HMIs en WINCC RT, incluyendo interfaces gráficas mejoradas y bibliotecas predefinidas, y se estudiaría la integración de otras plataformas y dispositivos industriales con OPC UA para ampliar la interoperabilidad en sistemas de automatización.

Glosario

AGV Automated Guided Vehicle.

COM Model of Objects Component.

CPU Central Processing Unit.

DNS Data Source Names.

HMI Human-Machine Interface..

IOP Intelligent Operator Panel.

MES Manufacturing Execution System.

ODBC Open DataBase Connectivityl.

OPC Open Platform Communications.

OPC UA Open Platform Communications Unified Architecture..

PLC Programmable Logic Controller..

PROFINET Runtime..

SCADA Supervisory Control and Data Acquisition..

TCP Transmission Control Protocol.

Referencias

[1] A. Rahman, Basic Control System: Get Familiar with PLC, DCS, SCADA, es, ago. de 2019. dirección: https://www.linkedin.com/pulse/basic-control-system-get-familiar-plc-dcs-scada-abdelrahman-kamal (visitado 10-04-2023).

[2] D. Collins, What is OPC UA and how does it compare with Industrial Ethernet? Mayo de 2020. dirección: https://www.motioncontroltips.com/what-is-opc-ua-and-how-does-it-compare-with-industrial-ethernet/ (visitado 10-04-2023).

[3] Estandarización y seguridad en el protocolo OPC UA, es, nov. de 2018. dirección: https://www.incibe-cert.es/blog/estandarizacion-y-seguridad-el-protocolo-opc-ua (visitado 10-04-2023).

[4] A. King, Understanding the OPC Unified Architecture (OPC UA) Protocol - Technical Articles, en, abr. de 2023. dirección: https://control.com/technical-articles/understanding-the-opc-ua-protocol/ (visitado 19-07-2023).

[5] M. Ladegourdie y J. Kua, «Performance Analysis of OPC UA for Industrial Interoperability towards Industry 4.0,» en, IoT, vol. 3, n.º 4, págs. 507-525, dic. de 2022, Number: 4 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2624-831X. DOI: 10.3390/iot3040027. dirección: https://www.mdpi.com/2624-831X/3/4/27 (visitado 29-07-2023).

[6] Siemens, «Learn-/Training Document,» 2019. dirección: https://www.automation.siemens.com/sce-static/learning-training-documents/tia-portal/advanced-communication/sce-092-300-opc-ua-s7-1500-r1807-en.pdf.

[7] OPC UA for S7-1200/S7-1500 CPUs - SIMATIC S7-1500, ET 200MP, ET 200SP, ET 200AL, ET ... - ID: 59192925 - Industry Support Siemens. dirección: https://support.industry.siemens.com/cs/mdm/59192925?c=161299668235&lc=en-AT (visitado 20-07-2023).

[8] Reptil.mx, Wincc, es-MX, jun. de 2021. dirección: https://industriasgsl.com/blogs/automatizacion/wincc-siemens (visitado 19-07-2023).

[9] Herramienta de puesta en marcha STARTER, es, ISSN: 1001-8085. dirección: https://mall.industry.siemens.com/mall/es/WW/Catalog/Products/10018085?tree=CatalogTree (visitado 23-07-2023).

[10] J. C. Sánchez Tapia y K. I. Sichiqui Velecela, «Estudio del variador de frecuencia G-120 y de las interfaces de comunicación PROFINET y PROFIBUS CU250S-2 y desarrollo de un manual de prácticas orientadas al aprendizaje de los sistemas de movimiento para la industria bajo el contexto de las comunicaciones industriales,» spa, Accepted: 2022-11-15T15:16:49Z, bachelorThesis, oct. de 2022. dirección: http://dspace.ups.edu.ec/handle/123456789/23795 (visitado 14-07-2023).

[11] Protocolo PROFINET IO, es-MX, Section: Videos ⁄ Webinarios, jul. de 2019. dirección: https://www.logicbus.com.mx/blog/que-es-el-protocolo-profinet-io/ (visitado 30-07-2023).

[12] PROFINET, en, fw_Converting. dirección: https://www.siemens.com/global/en/products/automation/industrial-communication/profinet.html (visitado 30-07-2023).

[13] A. SIEMENS, «Converter with the CU250S-2 Control Units,» en, págs. 123-124,

[14] SIMATIC WinCC RT Professional, en. dirección: https://new.siemens.com/global/en/product-services/automation/industry-software/automation-software/scada-systems/simatic-wincc-professional-rt.html (visitado 29-07-2023).

[15] HMI Design Masterclass, en, fw_Inspiring. dirección: https://www.siemens.com/global/en/products/automation/simatic-hmi/design-masterclass.html (visitado 29-07-2023).

[16] Sistema SCADA - seidamatic.com. dirección: http://www.seidamatic.com/es/Sistema-SCADA (visitado 29-07-2023).

[17] rwestMSFT, Abrir el Administrador de orígenes de datos ODBC - SQL Server, es-es, mayo de 2023. dirección: https://learn.microsoft.com/es-es/sql/database-engine/configure-windows/open-the-odbc-data-source-administrator?view=sql-server-ver16 (visitado 29-07-2023).

[18] «WinCC Engineering V15.1 - Programming reference,» en,

[19] Henry, Power and Torque in Induction Motors - The Engineering Knowledge, en-US, Section: Induction Motors, sep. de 2019. dirección: https://www.theengineeringknowledge.com/power-and-torque-in-induction-motors/,%20https://www.theengineeringknowledge.com/power-and-torque-in-induction-motors/ (visitado 30-07-2023).

Anexos

Algoritmo 4 Borrar datos de la tabla: parte 1

```
Sub DeleteData()
' Declaración de variables
Dim connect, ErrorDesc, SQLdelete

' Ignorar errores y continuar
On Error Resume Next

' Crear objeto de conexión
Set connect = CreateObject("ADODB.Connection")

' Abrir la conexión a la base de datos
connect.Open = "Provider=MSDASQL;Initial Catalog=DB_1;DSN=PLC"

' Verificar si ocurrió algún error al abrir la conexión
If Err.Number <> 0 Then
    ErrorDesc = Err.Description & "(" & Err.Number & ")"
    'ShowSystemAlarm "Error #" & Err.Number & " " & Err.Description
    SmartTags("ErrDesc") = ErrorDesc
    Set connect = Nothing
    Err.Clear
    Exit Sub
End If

' Comando SQL para eliminar todos los datos de la tabla
SQLdelete = "DELETE FROM valores_PLC"
```

Algoritmo 5 Borrar datos de la tabla: parte 2

```
    ' Ejecutar el comando SQL para eliminar los datos
    connect.Execute(SQLdelete)

    ' Verificar si ocurrió algún error al ejecutar el comando SQL
    If Err.Number <> 0 Then
        ErrorDesc = Err.Description & "(" & Err.Number & ")"
        'ShowSystemAlarm "Error #" & Err.Number & " " & Err.Description
        SmartTags("ErrDesc") = ErrorDesc
        Set connect = Nothing
        Err.Clear
        Exit Sub
    End If

    ' Cerrar la conexión a la base de datos
    connect.Close

    ' Liberar el objeto de conexión
    Set connect = Nothing
End Sub
```

Algoritmo 6 Inserción de la data en la tabla: parte 1

```
Sub InsertDataDB(ByRef Name_DSN)
    ' Declaración de variables
    Dim connect, record, DBName, TableName, ErrorDesc
    Dim SQLinsert, value1, value2, value3, value4

    ' Obtener valores de las SmartTags
    DBName = SmartTags("DataBaseName")
    TableName = SmartTags("TableName")
    value1 = SmartTags("Lectura_velocidad")
    value2 = SmartTags("Lectura_corriente")
    value3 = SmartTags("Lectura_torque")
    value4 = SmartTags("Lectura_potencia")

    ' Reemplazar comas por puntos en los valores
    ' para asegurar formato adecuado
    value1 = Replace(value1, ",", ".")
    value2 = Replace(value2, ",", ".")
    value3 = Replace(value3, ",", ".")
    value4 = Replace(value4, ",", ".")

    ' Ignorar errores y continuar
    On Error Resume Next

    ' Crear objetos de conexión y recordset
    Set connect = CreateObject("ADODB.Connection")
    Set record = CreateObject("ADODB.Recordset")
```

Algoritmo 7 Inserción de la data en la tabla: parte 2

```
    ' Abrir la conexión a la base de datos
    connect.Open = "Provider=MSDASQL;Initial Catalog=DB_1;DSN=PLC"

    ' Verificar si ocurrió algún error al abrir la conexión
    If Err.Number <> 0 Then
        ErrorDesc = Err.Description & "(" & Err.Number & ")"
        SmartTags("ErrDesc") = ErrorDesc
        Set connect = Nothing
        Err.Clear
        Exit Sub
    End If

    ' Comando SQL para insertar los datos en la tabla
    SQLinsert = "INSERT INTO valores_PLC (Velocidad, Corriente, Torque,
                Potencia) VALUES (" & value1 & "," & value2 & ",
                " & value3 & "," & value4 & ")"

    ' Ejecutar el comando SQL para insertar los datos
    Set record = connect.Execute(SQLinsert)

    ' Verificar si ocurrió algún error al ejecutar el comando SQL
    If Err.Number <> 0 Then
        ErrorDesc = Err.Description & "(" & Err.Number & ")"
        SmartTags("ErrDesc") = ErrorDesc
        Set connect = Nothing
        Err.Clear
        Exit Sub
    End If

    ' Cerrar la conexión a la base de datos
    connect.Close

    ' Liberar los objetos
    Set connect = Nothing
    Set record = Nothing
End Sub
```

Algoritmo 8 Exportar data a una plantilla de excel: parte 1

```
Sub InsertDataDBAndExportToExcel(ByRef Name_DSN)
    ' Declaración de variables
    Dim connect, record, DBName, TableName, ErrorDesc, SQLQuery
    Dim excelApp, excelWorkbook, excelWorksheet, excelFilePath
    Dim yearStr, monthStr, dayStr, hourStr, minuteStr, secondStr

    ' Definir el nombre de la base de datos y la tabla
    DBName = "DB_1"
    TableName = "valores_PLC"

    ' Ignorar errores y continuar
    On Error Resume Next

    ' Crear objetos de conexión y recordset
    Set connect = CreateObject("ADODB.Connection")
    Set record = CreateObject("ADODB.Recordset")

    ' Abrir la conexión a la base de datos
    connect.Open = "Provider=MSDASQL;Initial Catalog=DB_1;DSN=PLC"

    ' Verificar si ocurrió algún error al abrir la conexión
    If Err.Number <> 0 Then
        ErrorDesc = Err.Description & "(" & Err.Number & ")"
        SmartTags("ErrDesc") = ErrorDesc
        Set connect = Nothing
        Err.Clear
        Exit Sub
    End If
```

Algoritmo 9 Exportar data a una plantilla de excel: parte 2

```
' Crear la consulta SQL para obtener los datos de la tabla
SQLQuery = "SELECT * FROM " & TableName

' Abrir el recordset con la consulta SQL y la conexión
record.Open SQLQuery, connect

' Verificar si ocurrió algún error al abrir el recordset
If Err.Number <> 0 Then
    ErrorDesc = Err.Description & "(" & Err.Number & ")"
    SmartTags("ErrDesc") = ErrorDesc
    Set connect = Nothing
    Err.Clear
    Exit Sub
End If

' Crear una instancia de Excel
Set excelApp = CreateObject("Excel.Application")

' Abrir la plantilla de Excel
Set excelWorkbook = excelApp.Workbooks.Open("D:\Universidad\8 Ciclo\
                    .Tesis\Reportes\Plantilla.xlsx")

' Hacer visible la aplicación de Excel (opcional)
excelApp.Visible = True

' Copiar los datos de la tabla al libro de Excel
excelWorkbook.Sheets(1).Range("A3").CopyFromRecordset record

' Cerrar el recordset
record.Close
```

Algoritmo 10 Exportar data a una plantilla de excel: parte 3

```
    ' Obtener la fecha y hora actual para generar el nombre del archivo
    yearStr = Year(Date)
    monthStr = Right("0" & Month(Date), 2)
    dayStr = Right("0" & Day(Date), 2)
    hourStr = Right("0" & Hour(Now), 2)
    minuteStr = Right("0" & Minute(Now), 2)
    secondStr = Right("0" & Second(Now), 2)

    ' Especificar ruta y nombre del archivo con la fecha y hora actual
    excelFilePath = "D:\Universidad\8 Ciclo\.Tesis\Reportes\
                    ReporteProduccion_" & yearStr & monthStr &
                    dayStr & "_" & hourStr & minuteStr & secondStr &
                    ".xlsx"

    ' Guardar el libro de Excel en el archivo especificado
    excelWorkbook.SaveAs excelFilePath

    ' Cerrar el libro de Excel
    excelWorkbook.Close

    ' Cerrar la aplicación de Excel
    excelApp.Quit

    ' Liberar la memoria utilizada por los objetos de Excel
    Set excelWorksheet = Nothing
    Set excelWorkbook = Nothing
    Set excelApp = Nothing

    ' Cerrar la conexión a la base de datos
    connect.Close

    ' Liberar el objeto de conexión
    Set connect = Nothing
End Sub
```