UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE CUENCA

CARRERA DE ELECTRÓNICA Y AUTOMATIZACIÓN

"MONITOREO Y OPERACIÓN DE UN SISTEMA INDUSTRIAL DE SERVO POSICIONAMIENTO: HMI BASADA EN PYTHON Y OPC UA"

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

AUTOR: JOSÉ ANDRÉS MATUTE CAGUANA
MANUEL GUSTAVO ZHINDON SUMBA

TUTOR: JULIO CESAR ZAMBRANO ABAD, PhD.

Cuenca – Ecuador
2023

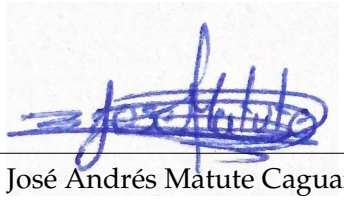
CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, José Andrés Matute Caguana con documento de identificación N° 1400716849 y Manuel Gustavo Zhindon Sumba con documento de identificación N° 0302455704; 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.

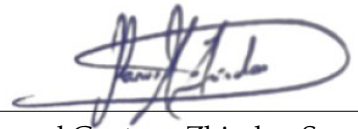
Cuenca, 28 de julio del 2023

Atentamente,



José Andrés Matute Caguana

1400716849



Manuel Gustavo Zhindon Sumba

0302455704

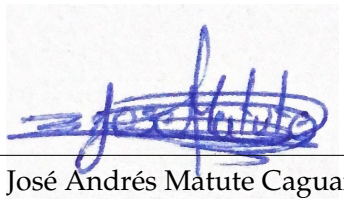
CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Nosotros, José Andrés Matute Caguana con documento de identificación N° 1400716849 y Manuel Gustavo Zhindon Sumba con documento de identificación N° 0302455704, 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: "Monitoreo y Operación de un Sistema Industrial de Servo Posicionamiento: HMI Basada en Python y OPC UA" 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 Andrés Matute Caguana

1400716849

Manuel Gustavo Zhindon Sumba

0302455704

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Julio César 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: MONITOREO Y OPERACIÓN DE UN SISTEMA INDUSTRIAL DE SERVO POSICIONAMIENTO: HMI BASADA EN PYTHON Y OPC UA, realizado por José Andrés Matute Caguana con documento de identificación N° 1400716849 y Manuel Gustavo Zhindon Sumba con documento de identificación N° 0302455704, 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 César Zambrano Abad
0301489696

AGRADECIMIENTOS

Agradecimiento de José Andrés Matute Caguana

En primer lugar, quiero expresar mi sincero agradecimiento a todas las personas que me han brindado su apoyo incondicional a lo largo de esta travesía académica. Estimado tutor Julio Zambrano, con gran emoción y gratitud, dedico esta tesis a ti, quien ha sido mi guía y mentor a lo largo de este arduo proceso académico. Tu apoyo inquebrantable, tu conocimiento y tus enseñanzas han sido fundamentales para mi desarrollo como estudiante y como persona. Gracias por desafiarme a alcanzar mi máximo potencial y por estar siempre disponible para responder mis preguntas y brindarme orientación. Esta tesis es un testimonio de tu dedicación y compromiso, y te agradezco de corazón por ser parte fundamental de este logro.

También quiero expresar mi agradecimiento a cada uno de los docentes por su invaluable contribución a mi formación académica. Su conocimiento compartido y su constante estímulo han sido cruciales para mi aprendizaje. Cada clase, cada lección y cada consejo han dejado una marca significativa en mi desarrollo como estudiante. Agradezco sinceramente su paciencia y dedicación para ayudarme a crecer intelectualmente y alcanzar mis metas educativas.

Queridas madres, Narcisa y Beatriz, elevó mi gratitud hacia ustedes, quienes han sido mis pilares fundamentales en esta travesía académica y en la vida en general. Su inquebrantable fe en mí, su amor incondicional y su apoyo constante han sido el motor que me ha impulsado en cada paso del camino. Agradezco por su sabiduría, su consejos y por inspirarme a ser una persona mejor. Esta tesis es un homenaje a su incansable amor, su intermidable paciencia y a su legado en mi vida.

Con mi más sincera gratitud y aprecio,

José.

Agradecimiento de Manuel Gustavo Zhindon Sumba

Quiero expresar mi sincero agradecimiento a todas las personas que contribuyeron de manera significativa a la realización de esta tesis. Su apoyo y colaboración han sido fundamentales para culminar este proyecto con éxito.

En primer lugar, quiero agradecer a mi tutor, Julio Zambrano, por su orientación, paciencia y valiosos consejos a lo largo de todo el proceso. Su guía experta y motivación fueron clave para superar los desafíos que surgieron durante la investigación.

Agradezco profundamente a mis padres, Manuel Zhindon y Maria Sumba, por su amor incondicional y constante apoyo. Gracias por ser mis pilares y brindarme el aliento necesario para seguir adelante en cada paso de esta trayectoria académica.

También quiero agradecer a mis hermanos y amigos, quienes me acompañaron en este camino y compartieron risas y momentos memorables. Su compañía y respaldo emocional fueron una fuente de fortaleza en todo momento.

Por último, agradezco a todos aquellos que, de una u otra manera, contribuyeron en el desarrollo de este proyecto, aunque no estén mencionados en estas líneas. Vuestras aportaciones han sido valiosas y significativas.

Este logro no habría sido posible sin el apoyo y comprensión de cada una de estas personas especiales en mi vida. Gracias por ser parte de este importante capítulo académico y por hacer posible este sueño.

Con gratitud sincera

Manuel

DEDICATORIAS

Dedicatoria de José Andrés Matute Caguana

Querida madre, tía, familia, amigos y tutor,

Con gran emoción y gratitud, dedico esta tesis a cada uno de ustedes, quienes han sido pilares fundamentales en mi vida y en este arduo proceso académico. Vuestra inquebrantable fe en mí, vuestro amor incondicional y vuestro apoyo constante han sido la fuerza impulsora detrás de cada paso que he dado hasta llegar aquí.

A mi madre Narcisa, eres mi ejemplo de fortaleza y dedicación. Tu amor inmenso y tus sacrificios han sido la luz que me ha guiado durante esta travesía académica. Gracias por tus palabras de aliento y por creer en mis capacidades incluso cuando yo dudaba de mí mismo. Esta tesis es un tributo a tu incansable amor y a tu infinita sabiduría.

A mi querida tía Beatriz, has sido mi mentora, confidente y segunda madre a lo largo de mi vida. Tus consejos sabios y tu apoyo constante han sido fundamentales para mi crecimiento personal y académico. Agradezco por tus palabras inspiradoras y por enseñarme la importancia de perseguir mis sueños con determinación. Esta tesis es también un reconocimiento a tu guía y amor incondicional.

A mi amada familia, ustedes han sido mi roca en los momentos más desafiantes. Vuestra paciencia, comprensión y aliento han sido el sostén que me ha permitido superar obstáculos y seguir adelante. Agradezco por vuestra confianza en mí y por ser mi red de seguridad cuando necesitaba un refugio. Esta tesis es un testimonio de nuestra unidad y amor familiar.

A mis queridos amigos, a través de risas y lágrimas, ustedes han estado a mi lado en

cada etapa de mi vida. Vuestra amistad incondicional y vuestro constante estímulo han sido esenciales para mantenerme enfocada en mis metas. Agradezco por los momentos compartidos, por las conversaciones estimulantes y por ser mi apoyo inquebrantable. Esta tesis celebra nuestra amistad duradera.

Y por supuesto a mi querido tutor Julio Zambrano, tu guía y conocimiento han sido fundamentales en mi desarrollo académico. Gracias por tu paciencia, por tus enseñanzas y por desafiarme a ir más allá de mis límites. Tu dedicación y compromiso han sido invaluables para mi crecimiento como estudiante y como persona. Esta tesis es un reconocimiento a tu apoyo constante y a tu valioso aporte a mi formación.

A todos ustedes, mi madre, tía, familia, amigos, abuela y mi querido tutor, les dedico esta tesis. Vuestro amor, apoyo y aliento han sido mi mayor motivación y han hecho posible este logro. Espero que este trabajo sea un testimonio de mi gratitud y unaforma de honrar la profunda influencia que cada uno de ustedes ha tenido en mi vida.

Con todo mi cariño y aprecio,

José.

Dedicatoria de Manuel Gustavo Zhindon Sumba

A mis amados padres, hermanos, amigos y profesor tutor,

Con profundo cariño y gratitud, dedico este logro a cada uno de ustedes, quienes han sido mi apoyo inquebrantable a lo largo de esta travesía académica. Vuestra presencia constante, aliento y confianza han sido el motor que me ha impulsado a alcanzar este sueño.

A mis padres, por ser mis guías y apoyarme en cada momento de mi vida. Vuestra dedicación, sacrificio y amor incondicional ha sido mi inspiración en cada desafío y ha hecho que los momentos de felicidad sean aún más especiales. Sin su apoyo, este logro no habría sido posible.

A mis hermanos, por compartir conmigo las alegrías y desafíos de la vida. Nuestro vínculo indestructible me ha dado fuerza y determinación para seguir adelante, incluso en los momentos más difíciles.

A mis amigos, por su lealtad y por alegrar mis días con risas y buenos momentos. Gracias por estar ahí en los altibajos de esta travesía y por brindarme vuestro respaldo sincero.

A mi profesor tutor, quien con su sabiduría, paciencia y orientación, ha sido mi guía académica y mentor en este proyecto. Sus conocimientos y confianza en mi capacidad han sido fundamentales para alcanzar este logro académico.

Este logro no solo me pertenece, sino que también es fruto del amor, el apoyo y la comprensión que he recibido de cada uno de ustedes. Sin vuestra presencia en mi vida, este camino habría sido mucho más arduo y menos significativo.

Gracias por ser mi sostén en cada desafío y celebrar conmigo cada triunfo. Esta tesis es el resultado de vuestro amor y apoyo incondicional.

Con cariño y agradecimiento eterno,

Manuel

Índice general

Índice de figuras

Índice de tablas

Resumen

La finalidad del proyecto es la implementación de una interfaz basada en OPC UA y Python para la operación y monitoreo de un sistema industrial de servo posicionamiento.

El sistema está conformado de un servomotor industrial, una estación de cómputo personal y un controlador lógico programable (SINAMICS S7-1500). El controlador lógico programable será el encargado de generar señales para controlar el sistema de servo posicionamiento del servomotor basado en la programación efectuada por el usuario, en la cual se implemento el uso del Telegrama 111 que nos permite el control del servomotor, junto con la definición de las variables correspondientes utilizadas en la lógica de control. Estas variables son enviadas a la interfaz HMI ubicada en la estación de cómputo personal, la misma que se desarrollara empleando el lenguaje de programación Python. De modo que mediante la estación de cómputo personal se pueda enviar y recibir datos al controlador para la operación y el monitoreo del sistema de servo posicionamiento logrando una comunicación bidireccional entre estos equipos.

EL controlador lógico programable y el equipo de cómputo personal están vinculados mediante el sistema estándar de comunicación OPC UA el mismo que posibilita el intercambio de datos entre distintos sistemas y equipos. En cambio, entre el controlador lógico programable y el sistema de servo posicionamiento se utiliza el protocolo de comunicación PROFINET el cual está desarrollado para la automatización de procesos.

Palabras clave: Controlador; HMI; OPC; Python; Servomotor; PROFINET

Abstract

The objective of the project is the implementation of an interface based on OPC UA and Python for the operation and monitoring of an industrial servo positioning system.

The system consists of an industrial servo motor, a personal computer station and a programmable logic controller (SINAMICS S7-1500). The programmable logic controller will be in charge of generating signals to control the servo positioning system of the servomotor based on the programming done by the user, in which the use of Telegram 111 was implemented, which allows us to control the servomotor, along with the definition of the corresponding variables used in the control logic. These variables will be sent to the HMI interface located in the personal computer station, which will be developed using the Python programming language. So that through the personal computer station can send and receive data to the controller for the operation and monitoring of the servo positioning system achieving a bidirectional communication between these teams.

The programmable logic controller and the PC are linked by means of the OPC UA standard communication system, which enables data exchange between different systems and devices. The PROFINET communication protocol developed for process automation is used between the programmable logic controller and the servo positioning system.

Keywords: Controller; HMI; OPC; Python; Servomotor; PROFINET

Antecedentes

En los últimos años, la industria ha experimentado un gran aumento en la demanda de alta precisión en la producción de bienes y servicios. Con este aumento en la demanda, ha habido un aumento en la necesidad de sistemas de servo posicionamiento industrial para controlar y posicionar diferentes tipos de maquinarias y equipos con alta precisión.

Los sistemas de servo posicionamiento industrial son utilizados para controlar y posicionar con alta precisión diferentes tipos de maquinarias y equipos en la industria. La necesidad de estos sistemas de servo posicionamiento industrial surge de una exigencia mayor en precisión y velocidad en la producción industrial, así como de la necesidad de reducir los errores y los tiempos de inactividad.

Sin embargo, uno de los principales problemas que enfrentan los sistemas de servo posicionamiento industrial es la complejidad de la programación y la calibración de los mismos que puede afectar la precisión del sistema. Además, la interferencia electromagnética (EMI) y el ruido eléctrico también pueden presentar un desafío para los operadores.

Por lo tanto, el estudio de los sistemas de servo posicionamiento industrial se centra en el desarrollo de tecnologías y técnicas para mejorar la precisión y la eficiencia de estos sistemas, así como en la identificación y solución de los problemas técnicos que puedan surgir. Esto incluye la investigación de nuevas tecnologías de motores y controladores, así como la mejora de la programación y la calibración de estos sistemas para una mayor facilidad de uso, generando las interfaces de control y monitoreo HMI, donde el operador puede controlar la velocidad y la posición de la maquinaria en tiempo real utilizando la interfaz de HMI.[1]

Estas interfaces HMI se realizan en diversos tipos de programación, sin

embargo, al realizar un HMI basado en Python puede proporcionar una interfaz de usuario personalizable que ofrece una amplia gama de bibliotecas y herramientas que pueden ayudar en la implementación de un sistema de control como el de un servoposicionamiento industrial. Esta interfaz puede ser rentable en comparación con otras soluciones de automatización industrial, ya que Python es un lenguaje de código abierto y es gratuito para su uso y distribución. Además, Python se puede ejecutar en hardware de bajo costo, lo que puede reducir el costo total en sistemas de monitoreo y control.[2]

La implementación de un HMI basado en Python para el control y monitoreo de un sistema de servoposicionamiento industrial puede proporcionar una solución rentable y altamente personalizable para la automatización de la línea de producción. Si se abordan adecuadamente los desafíos relacionados con la seguridad y la capacitación del personal, esta solución puede mejorar la eficiencia, la reducción de costos y la calidad del producto en la línea de producción.

Todos estos equipos utilizados en la industria se encuentran montados en el laboratorio de Redes Industriales, por lo que se enfocaría en realizar una interfaz HMI capaz de monitorear y controlar el sistema de servoposicionamiento industrial.

Justificación

Para abordar estos problemas, se ha desarrollado una amplia gama de tecnologías y técnicas para mejorar la eficiencia y la precisión de los sistemas de servo posicionamiento industrial. Estas tecnologías incluyen el uso de motores más avanzados y controladores de mayor capacidad de procesamiento, así como el desarrollo de técnicas de programación y calibración más intuitivas y fáciles de usar.

Además de la mejora técnica de los sistemas de servo posicionamiento industrial, también existe una necesidad creciente de justificación económica para el uso de estos sistemas. A medida que la competencia en la industria aumenta, las empresas buscan formas de mejorar la productividad y la eficiencia de sus operaciones.

En general, los costos de los sistemas de servoposicionamiento industrial pueden oscilar entre unos pocos cientos de dólares para sistemas de baja carga y precisión, hasta varios miles o decenas de miles de dólares para sistemas de alta capacidad y precisión.

Los sistemas de servo posicionamiento industrial permiten a las empresas aumentar la eficiencia y la precisión de sus operaciones, lo que a su vez puede conducir a un aumento en la productividad y una reducción en los costos operativos.

Al diseñar una interfaz en un lenguaje de código abierto y gratuito reducirá los costos en los sistemas de monitoreo de los sistemas de servoposicionamiento, a más de permitir realizar cambios para mejorar la velocidad, precisión y calidad del producto final al momento de llevar a cabo algún cambio de producción.

Objetivos

Objetivo General

- Implementar una interfaz basada en OPC UA y Python para el monitoreo y operación de un sistema industrial de servo posicionamiento.

Objetivos Específicos

- Analizar el modo de operación y las configuraciones del sistema de servo accionamiento para sentar las bases del diseño del sistema de automatización y monitoreo.
- Implementar el sistema de control y operación del servo motor sobre un controlador lógico programable SIMATIC S7-1500 utilizando los telegramas de control para comunicaciones PROFINET.
- Implementar el sistema de comunicación entre la estación de cómputo personal y el PLC utilizando el estándar de comunicación OPC UA.
- Diseñar e implementar el sistema HMI en Python para operar y monitorizar el sistema industrial de servo posicionamiento.

Introducción

En el ámbito industrial, los sistemas de servo posicionamiento desempeñan un papel fundamental en la automatización de procesos de fabricación. Estos sistemas permiten controlar con precisión la posición, velocidad y aceleración de componentes mecánicos, lo que resulta crucial para garantizar la calidad y eficiencia en la producción. En este contexto, la integración de módulos de Siemens, la comunicación a través del estándar OPC UA (Unified Architecture) y el uso de la interfaz en Python, han demostrado ser soluciones eficaces y versátiles para el control y monitoreo de sistemas de servo posicionamiento industrial.

El presente trabajo de tesis se centra en el diseño y desarrollo de un sistema de control y monitoreo de un sistema de servo posicionamiento industrial, utilizando módulos de Siemens como base para la implementación. La elección de los módulos de Siemens se basa en su reconocida fiabilidad, precisión y compatibilidad con una amplia gama de aplicaciones industriales.

Además, se empleará la comunicación OPC UA, un estándar ampliamente adoptado en la industria, que permite la interoperabilidad y el intercambio seguro de datos entre diferentes dispositivos y sistemas. La comunicación OPC UA garantiza la flexibilidad y escalabilidad del sistema, lo que facilita su integración con otros componentes y sistemas existentes en el entorno industrial.

En cuanto a la interfaz de usuario, se utilizará Python, un lenguaje de programación ampliamente utilizado en aplicaciones de control industrial y automatización. Python ofrece una amplia gama de bibliotecas y herramientas que facilitan el desarrollo de interfaces intuitivas y robustas para el control y monitoreo de sistemas de servo posicionamiento.

Capítulo 1

Sistema de servo posicionamiento industrial

En el presente capítulo se aborda las características y funciones de cada componente del proyecto de automatización, junto con un breve procedimiento de puesta en marcha del sistema de servo posicionamiento utilizando la plataforma STARTER de SIEMENS.

Para una mejor referencia, en la figura 1.1 se presenta la arquitectura del sistema de servo posicionamiento a implementar.

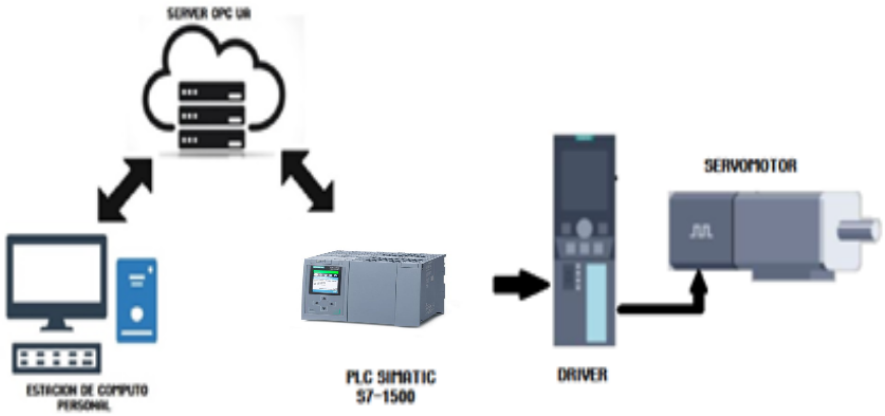


Figura 1.1: Arquitectura del sistema a implementar.

1.1. Introducción

El sistema de servo posicionamiento industrial es una tecnología altamente avanzada que permite un control preciso y eficiente del movimiento y posición de motores en aplicaciones industriales. Este sistema combina componentes clave, como motores sincrónicos, accionamientos y controladores, para lograr un rendimiento óptimo en términos de velocidad, precisión y estabilidad [3]. En este capítulo se proporcionará una descripción detallada de los componentes principales utilizados en el sistema de servo posicionamiento industrial, los cuales pertenecen a la marca SIEMENS.

1.2. Motor síncrono 1FT7/1FK7

Uno de los componentes fundamentales en el sistema de servo posicionamiento industrial es el motor sincrónico 1FK7 de SIEMENS (véase la figura 1.2). Este motor

es ampliamente reconocido por su excelente rendimiento y capacidad de respuesta. Diseñado para aplicaciones de alta dinámica, el motor 1FK7 ofrece una alta precisión en el posicionamiento y una amplia gama de velocidades.

Sus características clave incluyen una construcción compacta, una alta densidad de potencia y una baja inercia rotacional. Estas cualidades lo convierten en una opción ideal para aplicaciones industriales que demandan un control preciso del movimiento.

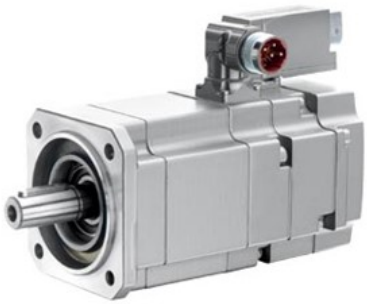


Figura 1.2: Motor 1FK7 [4].

El motor síncrono 1FK7 utiliza imanes permanentes para generar el campo magnético necesario para la rotación del rotor [5]. Esto permite un rendimiento óptimo en términos de eficiencia energética y potencia de salida. Además, estos motores contienen encoders incrementales y se benefician de tecnologías avanzadas de enfriamiento, lo que contribuye a su capacidad de manejar altas cargas de trabajo sin sobre calentarse. En la tabla 1.1 se muestra los valores óptimos de operación del motor.

Tabla 1.1: Valores óptimos para el funcionamiento del servo motor.

Variables	Valores óptimos del servo motor
Velocidad	6000 rpm
Potencia	0.4 kW
Corriente	1.4 A
Número de polos	6
Eficiencia	86.0 %
Torque	0.6 Nm
Constante del torque	0.46 Nm/A

Figura 1.3: Conector EPIC [6].

Para la conexión de potencia se utiliza un conector EPIC de 6 pines (véase la figura 1.3. La distribución de los pines se configura como se indica en la figura 1.4. Por otra parte, el detalle de los pines se explica en la tabla 1.2.

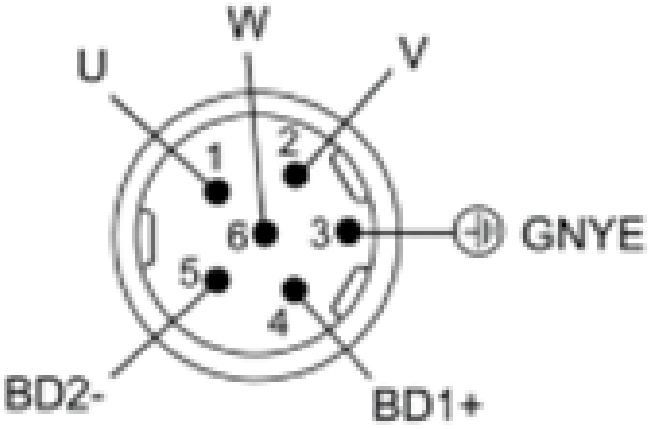


Figura 1.4: Distribución de pines del conector EPIC [7].

Tabla 1.2: Pines del Conector EPIC.

Pines	Identificador	Descripción
1	U	Fase U de Alimentación del Motor
2	V	Fase V de Alimentación del Motor
3	GNYE	Tierra Común
4	BD1+	Señal de Freno
5	BD2-	Señal de Freno
6	W	Fase W de Alimentación del Motor

La conexión para las señales de control se realiza mediante una interfaz

DRIVE-CLiQ, la cual se da mediante un cable RJ45 de 10 polos, permitiendo la conexión entre el motor y el Power Module como se evidencia en la Figura 1.5.

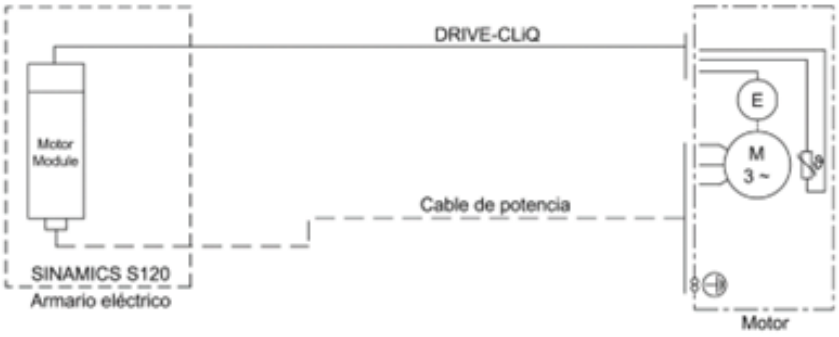


Figura 1.5: Conexión DRIVE-CLiQ [7].

1.3. Accionamiento SINAMICS S120

En esta sección se introduce el accionamiento SINAMICS S120 como un componente clave del sistema de servo posicionamiento industrial. Se resalta su papel en el control y suministro de energía al motor síncrono 1FT7/1FK7. Se destaca la modularidad y flexibilidad de este accionamiento, lo que le permite adaptarse a diversas aplicaciones industriales.

Se describen las características avanzadas, como perfiles de velocidad y aceleración programables. Además, se mencionan sus capacidades de diagnóstico y supervisión, junto con sus interfaces de comunicación para una integración fluida con sistemas de control superiores [8].

1.3.1. Descripción y Características

El accionamiento SINAMICS S120 es un sistema altamente flexible y modular que brinda un control preciso y eficiente a motores eléctricos en aplicaciones industriales. El accionamiento es una combinación de una etapa de potencia (módulo de potencia PM340) y una etapa de control (unidad de control CU310-2PN).

El módulo de potencia PM340 (6SL3210-1SB12-3AA0) es un accionamiento servo de la gama Sinamics S120. Estos accionamientos servo se utilizan principalmente

en aplicaciones de alto rendimiento en ingeniería mecánica y de sistemas. Este módulo de potencia tiene un formato Blocksize, el cual se muestra en la Figura 1.6. Por otra parte, las características principales del módulo se resumen en la tabla 1.3.

Tabla 1.3: Características del módulo de potencia PM340.

Características del módulo de potencia PM340	
Número de serie	6SL3210-1SB12-3AA0
Potencia de salida	0.37kW
Voltaje de salida	3AC 200-240V
Corriente de salida	2,3 A
Voltaje de entrada	1AC 200-240V
Frecuencia	50/60 Hz
Tamaño	Blocksize size FSA

Figura 1.6: Vista panorámica del módulo de potencia PM340 [9]

La unidad de control CU310-2PN (véase la figura 1.7) sirve para controlar la unidad de potencia PM340. Esta unidad cuenta con un interfaz de comunicación Profinet para comunicarse con un PLC o con otros dispositivos industriales [7]. La

CU310-2PN está equipada con interfaces específicas que incluyen:

- 11 entradas digitales con aislamiento galvánico,
- 8 entradas/salidas digitales sin aislamiento galvánico,
- una salida digital con aislamiento galvánico,
- una entrada analógica sin aislamiento galvánico,
- Interfaz Drive-CLiQ,
- 2 interfaces Profinet. Uno de los interfaces cuenta con un puerto, mientras que el otro cuenta con dos.
- un interfaz RS232,
- un LAN (ethernet),
- un interfaz de encóder (TTL/HTL/SSI),
- un borne EP,
- 3 hembrillas de medida,
- una entrada para sensor de temperatura.

Esta unidad puede ser pilotada desde el interfaz STARTER de Siemens. Además, para su funcionamiento se requiere una tarjeta de memoria que debe ser insertada y retirada únicamente cuando no haya tensión en la unidad de control. Es importante destacar que este componente es altamente sensible a descargas electroestáticas [7].

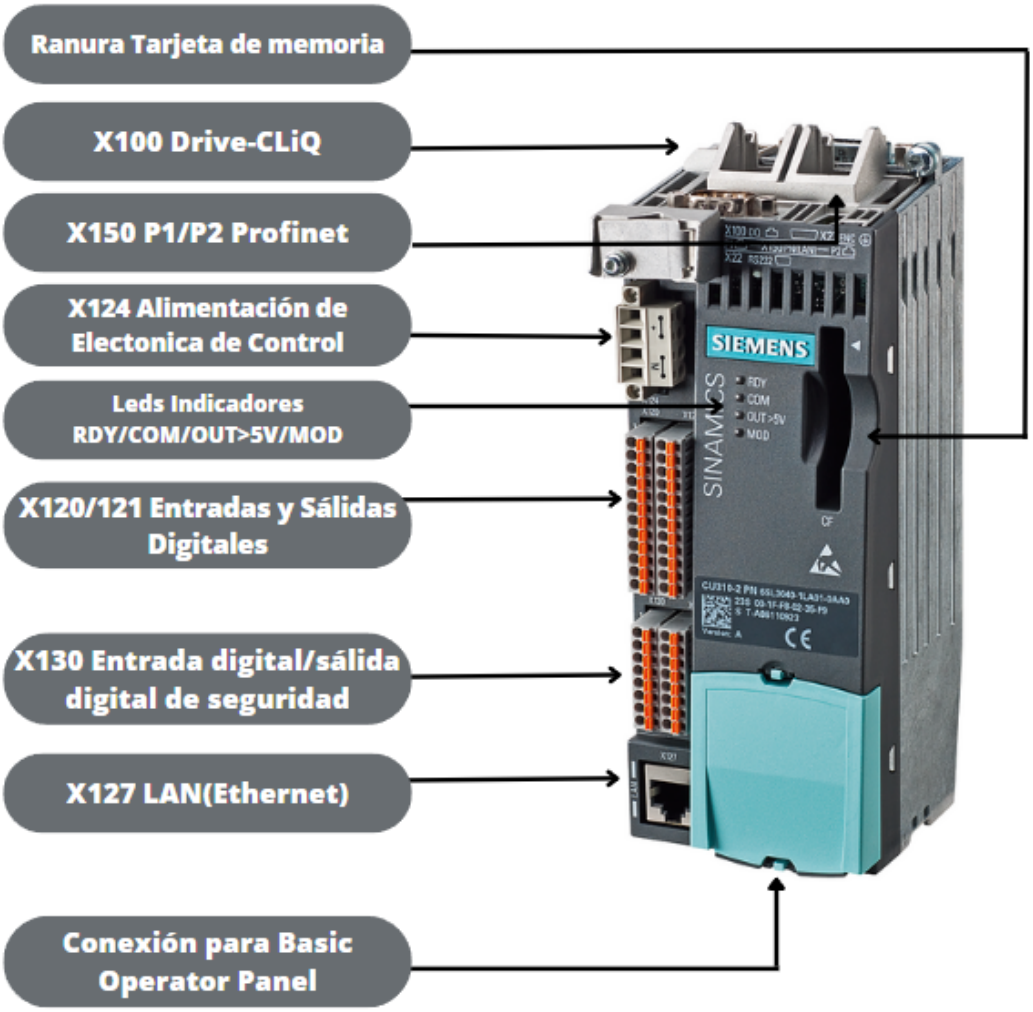


Figura 1.7: Control Unit S120-CU310-2PN [10].

1.4. Puesta en marcha básica con STARTER

Para llevar a cabo una puesta en marcha básica y rápida del servo accionamiento, se utiliza el programa STARTER de Siemens. A continuación, se detallan los requisitos necesarios para configurar adecuadamente el servo accionamiento:

- STARTER V5.5
- Conexión de la CU310-2 PN en cualquier puerto PROFINET

Una vez completados los requisitos, se procede a crear un nuevo proyecto en STARTER utilizando el asistente de proyectos. Luego, se selecciona la función de <Buscar unidades accionamiento online>. Todo este proceso se ilustra detalladamente en la Figura 1.8.

Figura 1.8: Asistente de proyectos (STARTER).

Una vez seleccionada la función mencionada, como se presenta en la Figura 1.9, se procede a modificar el nombre del proyecto para su identificación y se avanza a la siguiente pestaña, denominada <Ajustar interfaz de PG/PC>, como se muestra en la Figura 1.10.

Figura 1.9: Crear Proyecto Nuevo (STARTER).

Figura 1.10: <Ajustar interfaz de PG/PC>(STARTER).

En este punto, como se observa en la Figura 1.11, se procede a configurar la red a través de la cual se conectará el accionamiento. En este caso, se utiliza un punto de acceso DEVICE con la interfaz de la tarjeta de red del ordenador. Una vez que se ha establecido la interfaz, se avanza a la siguiente pestaña, presentada en la Figura 1.12, donde se visualiza la unidad de accionamiento que se desea configurar.

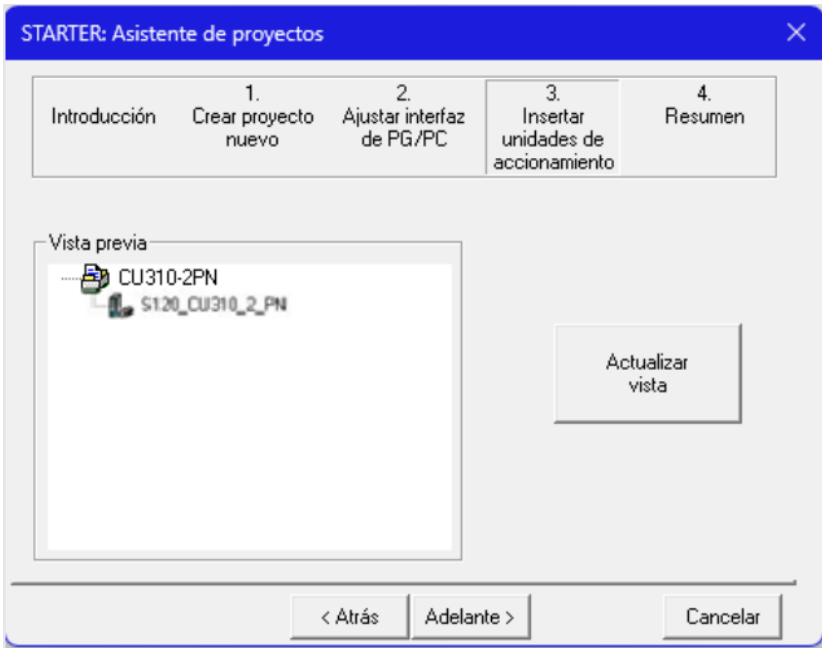


Figura 1.11: Insertar unidades de accionamiento (STARTER).

Figura 1.12: Resumen (STARTER).

La unidad de accionamiento se desplegará únicamente si la interfaz configurada se encuentra en red con el accionamiento deseado. Una vez que se ha seleccionado la unidad de accionamiento, se procede a finalizar la creación del proyecto y se mostrará una ventana con las propiedades y el nombre del proyecto guardado.

Para iniciar la configuración, es necesario establecer una conexión en línea con el accionamiento (Establecer ONLINE). Esto permite que la interfaz de STARTER se sincronice con el programa cargado en la unidad de control, asegurando la correspondencia y permitiendo la configuración y monitoreo adecuados.

Una vez en línea, se procede a agregar el servo motor al proyecto. Para ello, se despliega el proyecto y se pulsa sobre <Insertar accionamiento>, como se evidencia en la Figura 1.13. Al hacerlo, se abrirá una pequeña ventana donde se modificará el nombre que se desea asignar al accionamiento y se seleccionará el tipo de objeto, como se observa en la Figura 1.14.

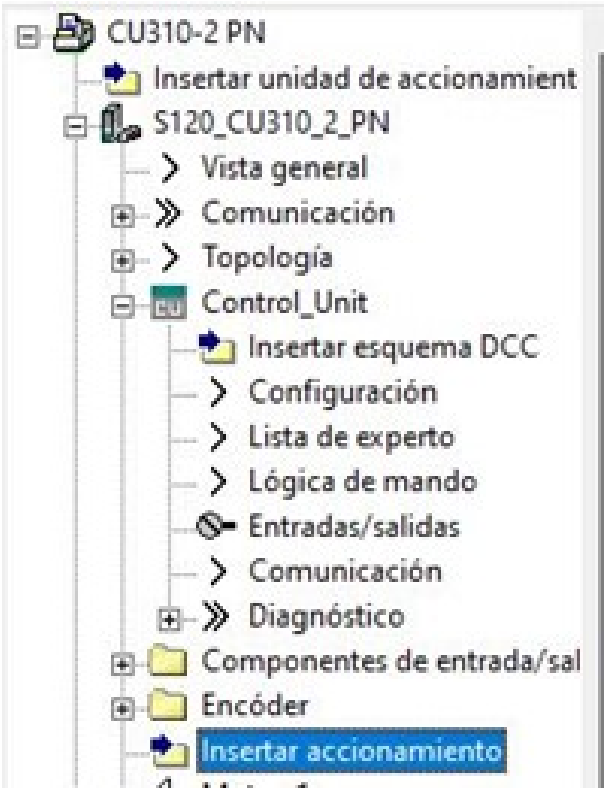


Figura 1.13: Lista del proyecto (STARTER).

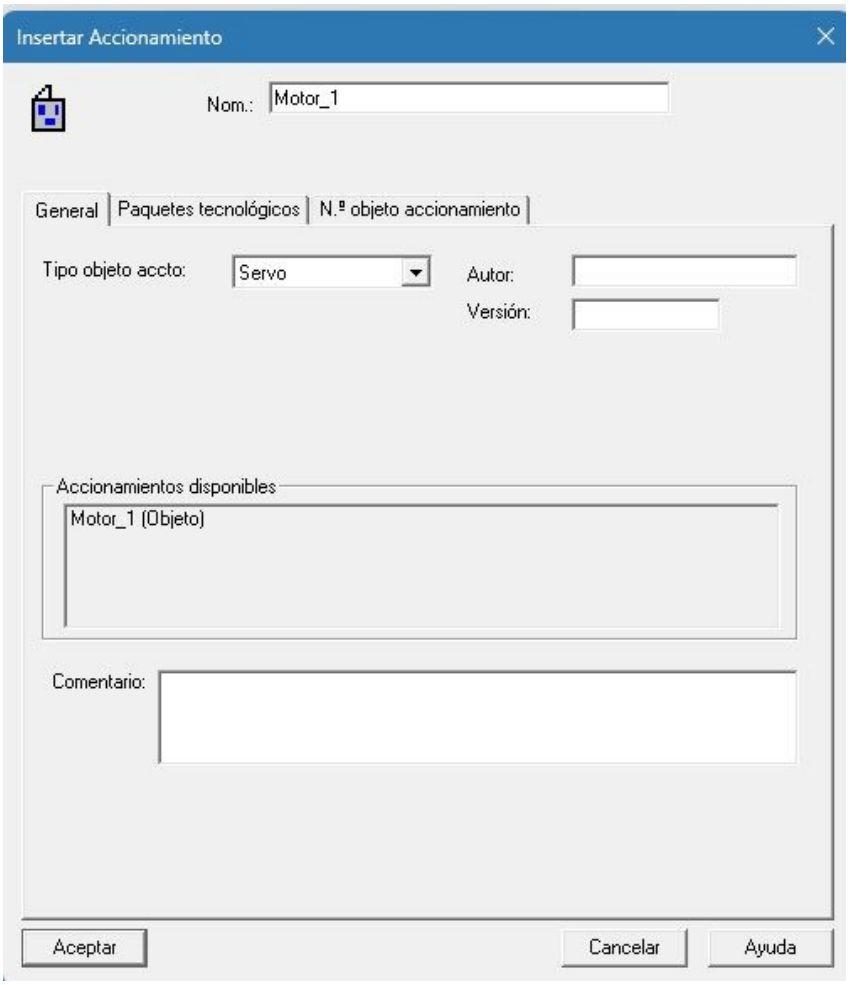


Figura 1.14: Ventana - insertar accionamiento (STARTER).

A continuación, se procede a configurar el accionamiento. Para ello, es necesario acceder a la pestaña <Configuración>en la sección del accionamiento. Al hacerlo, aparecerá una opción resaltada en amarillo llamada ”Configurar DDS...”, como se muestra en la figura 1.15. Al seleccionar esta opción, se abrirá una ventana de configuración que permitirá ajustar los parámetros del servo motor y su accionamiento.

Figura 1.15: Configuración DDS (STARTER).

En la ventana de Estructura de regulación, como se muestra en la Figura 1.16, se selecciona el módulo de función <Posicionador simple>, y se verifica que el tipo de

regulación sea <Regulación de velocidad de giro con encoder>

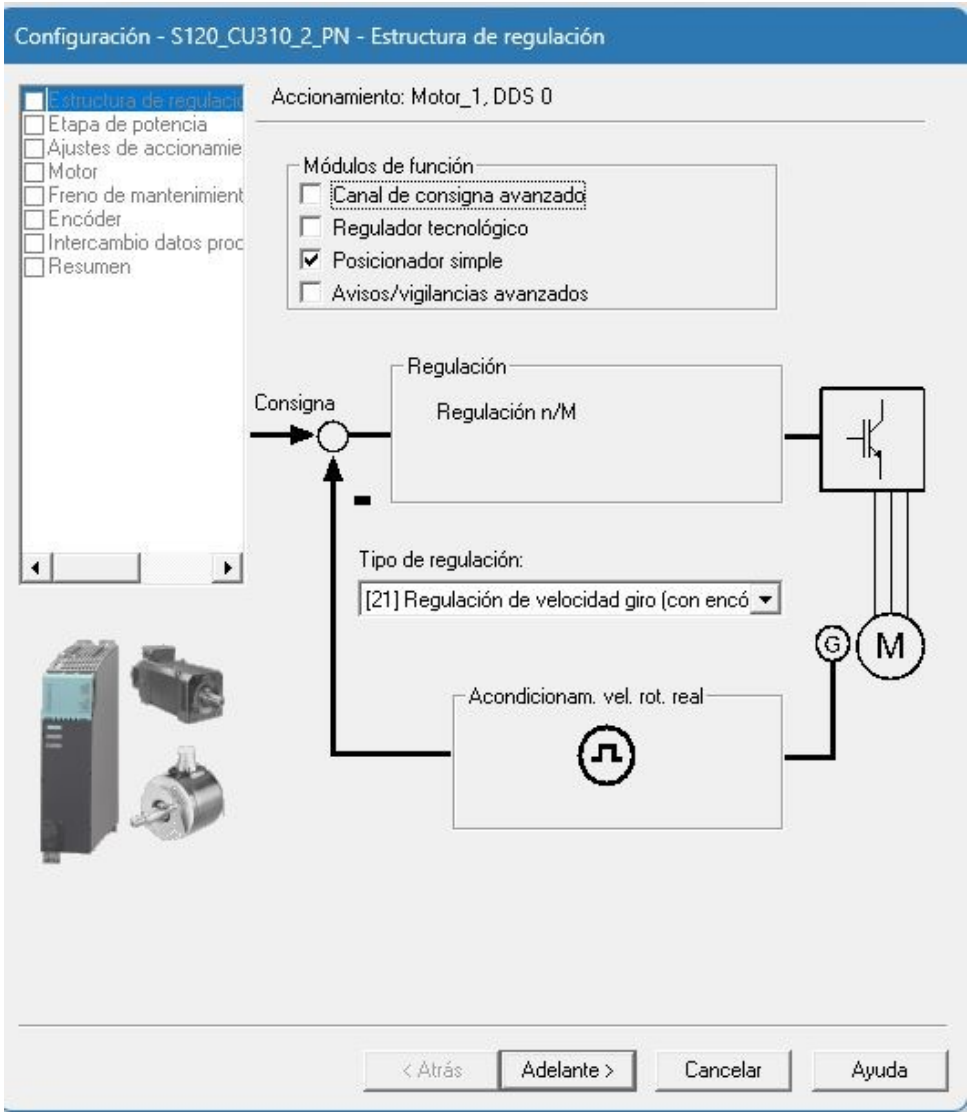


Figura 1.16: Estructura de regulación (STARTER).

En la siguiente pestaña, mostrada en la Figura 1.17, se encuentra la etapa de potencia, donde se buscan las características respectivas al Power Module previamente mencionado.

Para este caso, se utiliza la tensión de conexión de 1AC/3AC 200-240V, se selecciona un módulo con potencia de 0.37kW y una intensidad de 2.5A, ya que esas son las especificaciones del PM 340 que se encuentra en el proyecto.

Figura 1.17: Etapa de potencia (STARTER).

Siguiendo las opciones, se debe verificar que el producto coincida con el número de referencia del modelo usado y algunas de sus características, tal como se evidencia en la Figura 1.18.

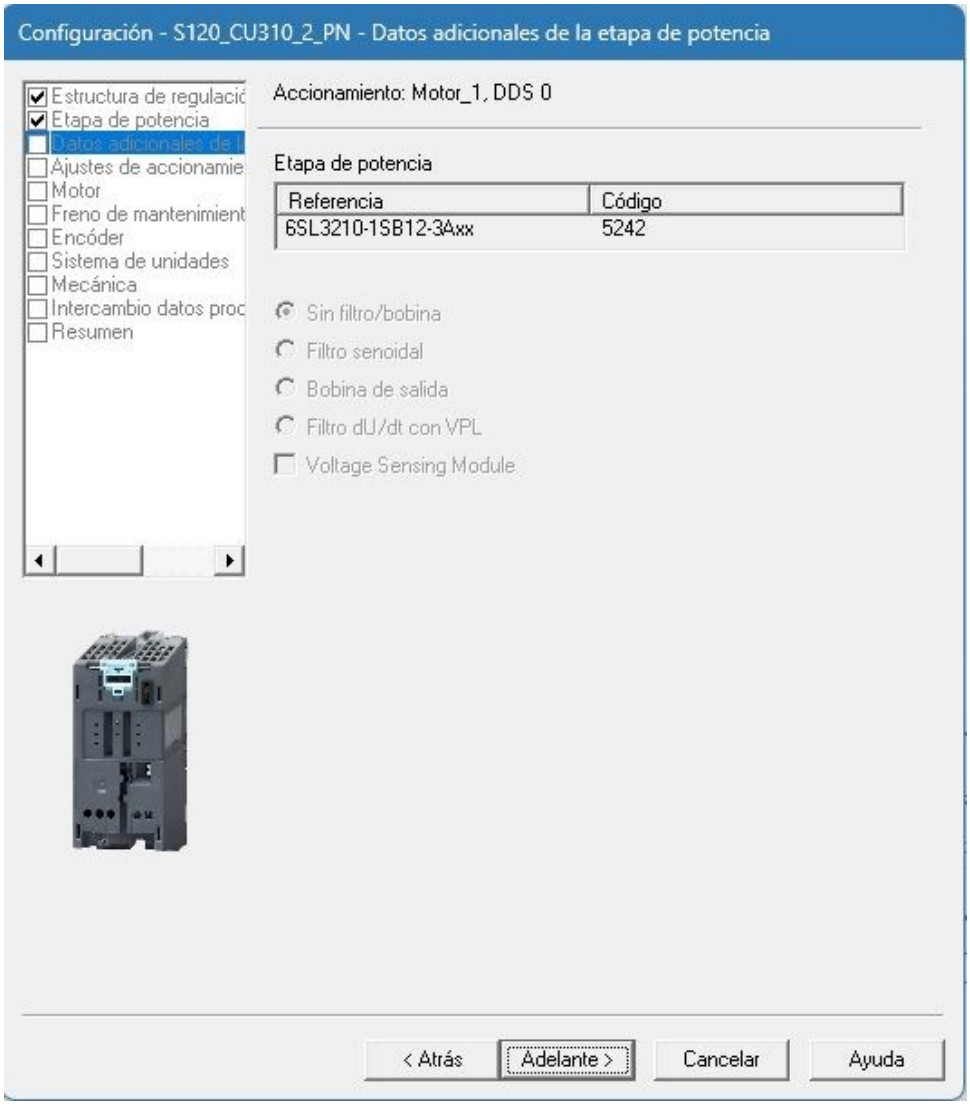


Figura 1.18: Datos adicionales de la etapa de potencia (STARTER).

En la siguiente ventana, es necesario confirmar la norma del motor que se empleará. En este caso, se selecciona la norma <Motor IEC a 50 HZ unidades del SI>, tal como se observa en la Figura 1.19.

Figura 1.19: Ajustes de acccionamiento (STARTER).

En la siguiente ventana mostrada en la Figura 1.20, se procede a ingresar los valores del motor. Para ello, se utiliza el interfaz DRIVE-CLiQ, el cual permite leer los datos del motor en el momento en que se encuentra en paro.

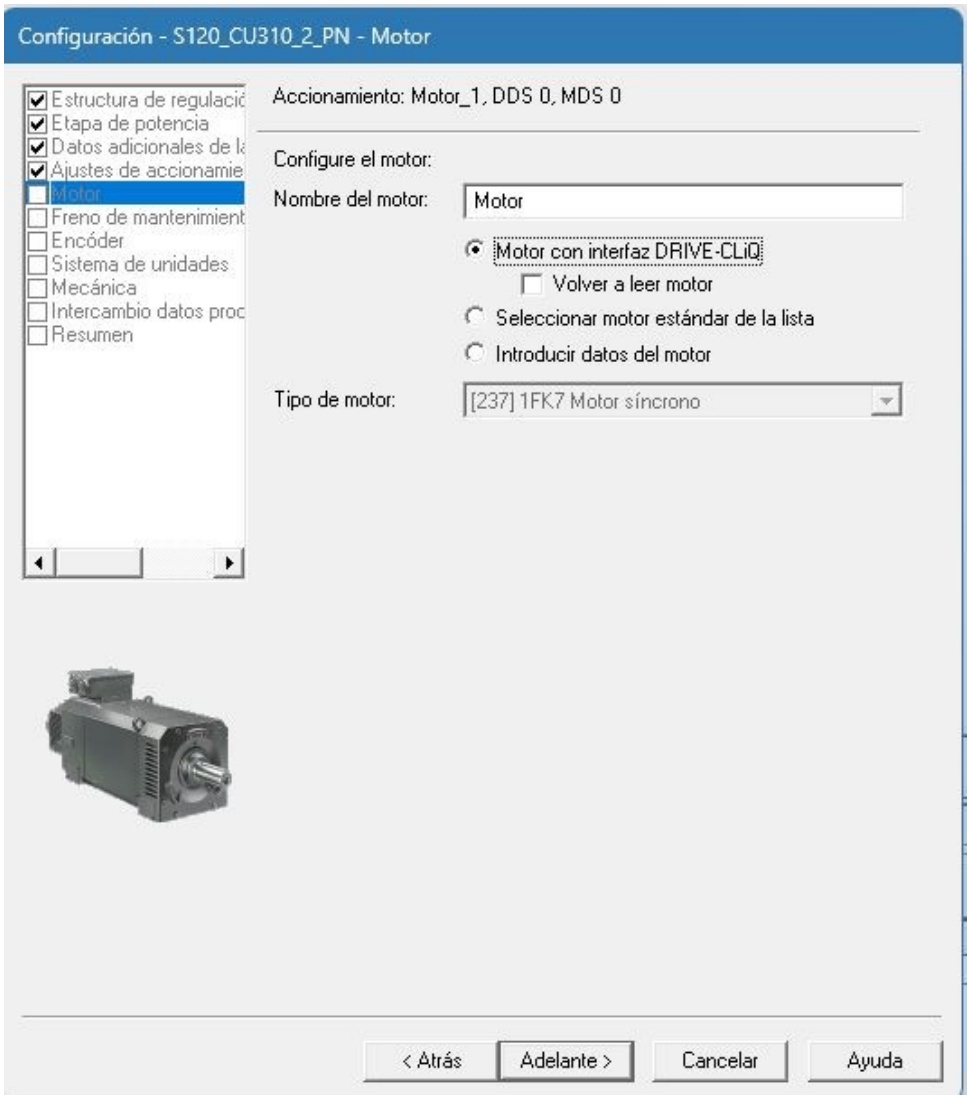


Figura 1.20: Valores del motor (STARTER).

En la ventana mostrada en la Figura 1.21, se pueden obtener las opciones para la serie de motores con freno de mantenimiento interno y motores sin freno de mantenimiento. En este caso, la activación del freno de mantenimiento del motor no será utilizada.

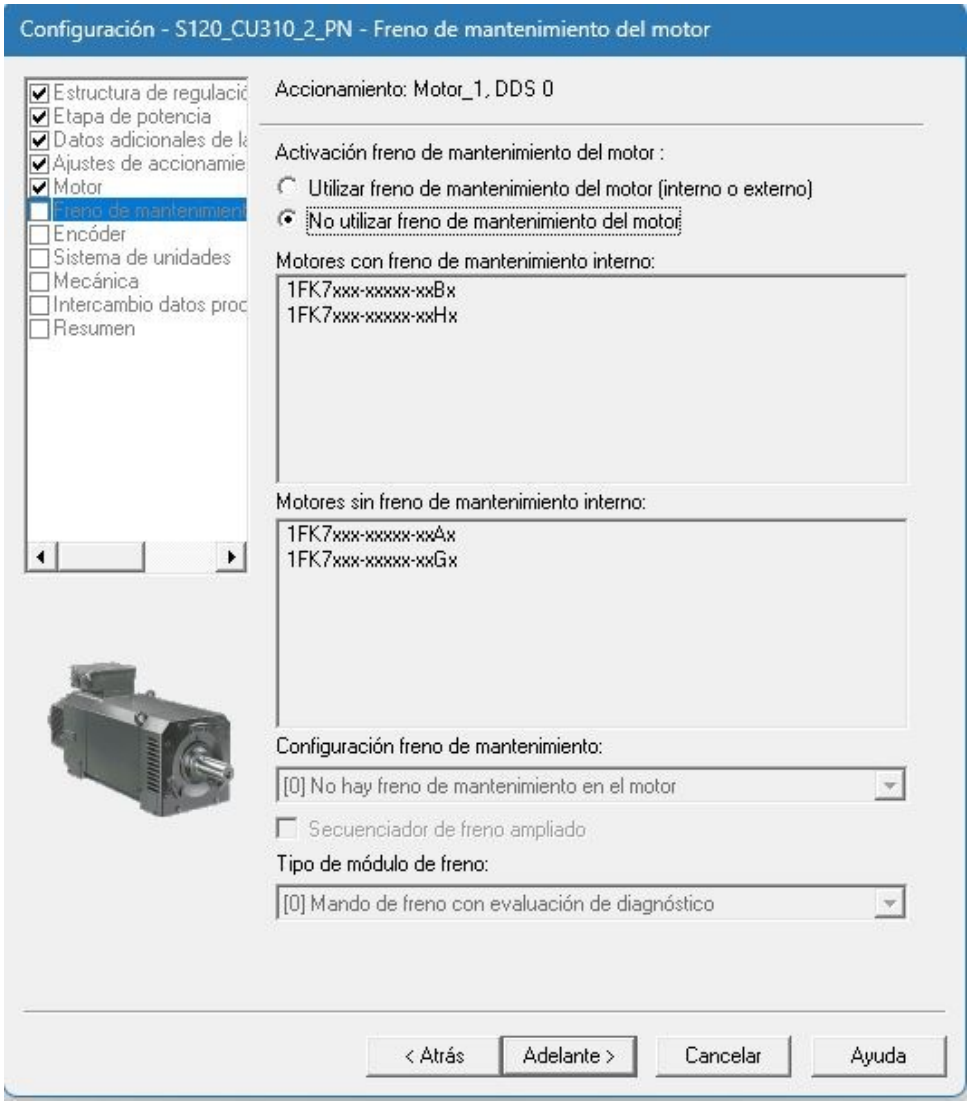


Figura 1.21: Freno de Mantenimiento (STARTER).

En la ventana de encoder, se selecciona el encoder que se desea utilizar. En este caso, mediante el interfaz DRIVE-CLiQ, se encuentra y selecciona automáticamente el Encoder_1, tal como se ilustra en la Figura 1.22.

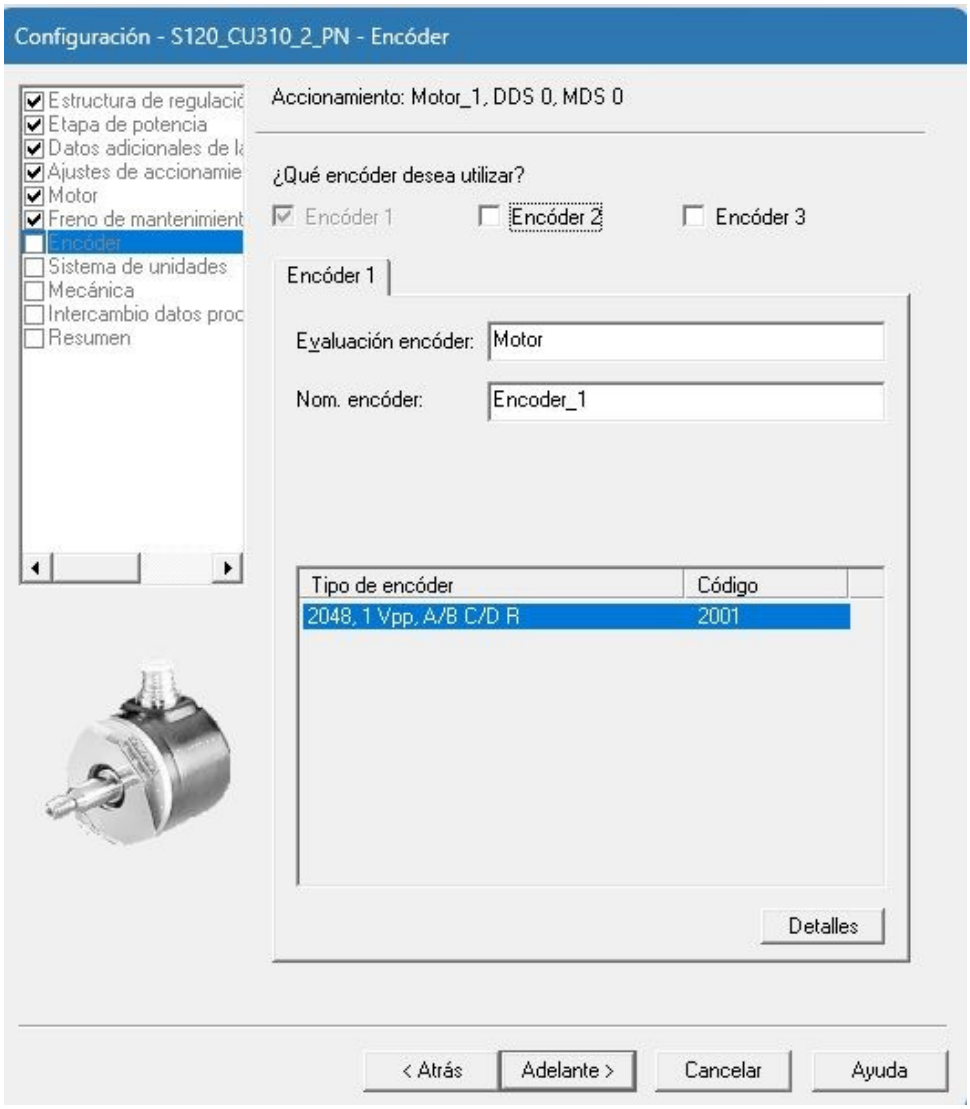


Figura 1.22: Selección del encoder (STARTER).

A continuación, se selecciona el sistema de encoder para la regulación de posición, el cual dependerá del juego de datos de accionamiento (DDS), como se evidencia en la Figura 1.23.

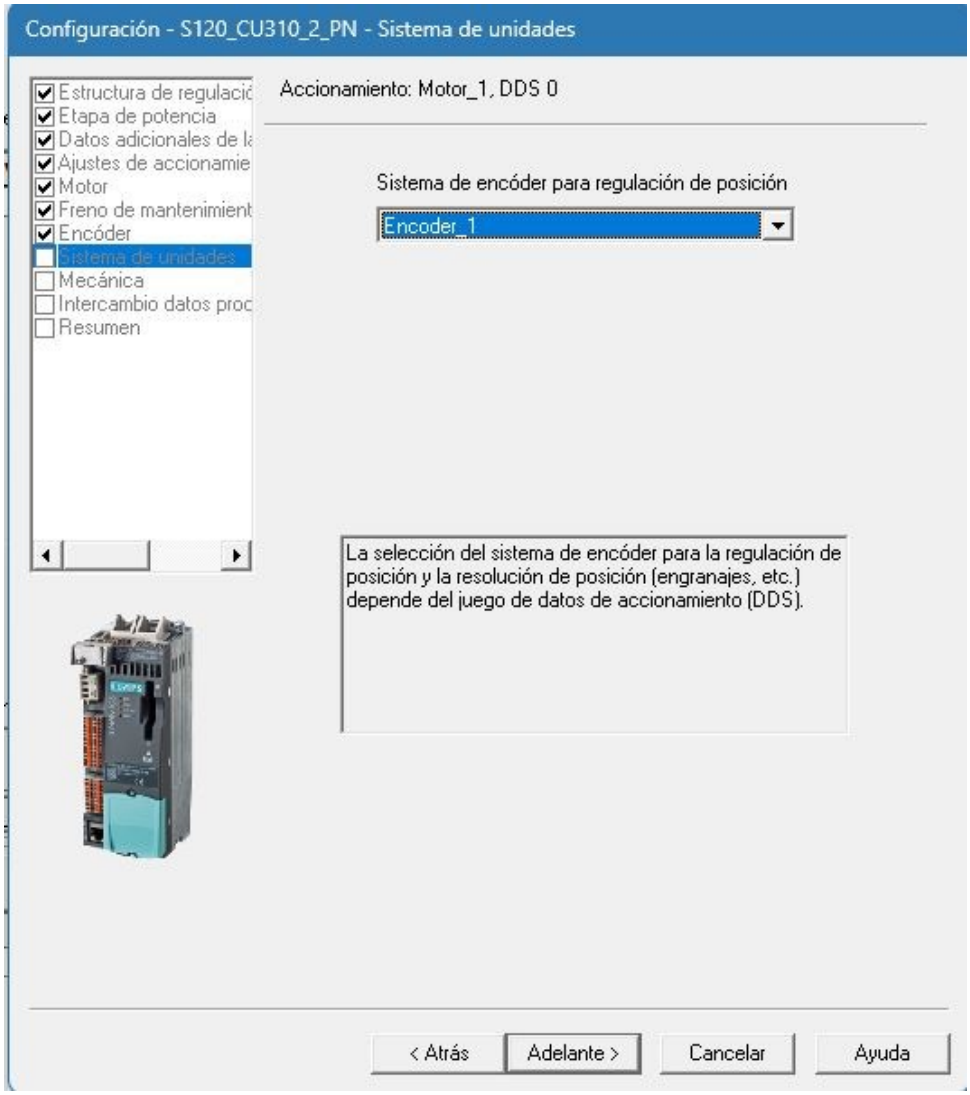


Figura 1.23: Sistema de unidades (STARTER).

En la ventana de mecánica, se visualizan los valores cargados del total de unidades de longitud (LU), la resolución del encoder y la resolución final, tal como se ilustra en la Figura 1.24.

Además de esto, se tiene la opción de modificar los valores de LU para cada revolución e incluso activar la corrección del módulo después de alcanzar un valor establecido en 360,000 LU.

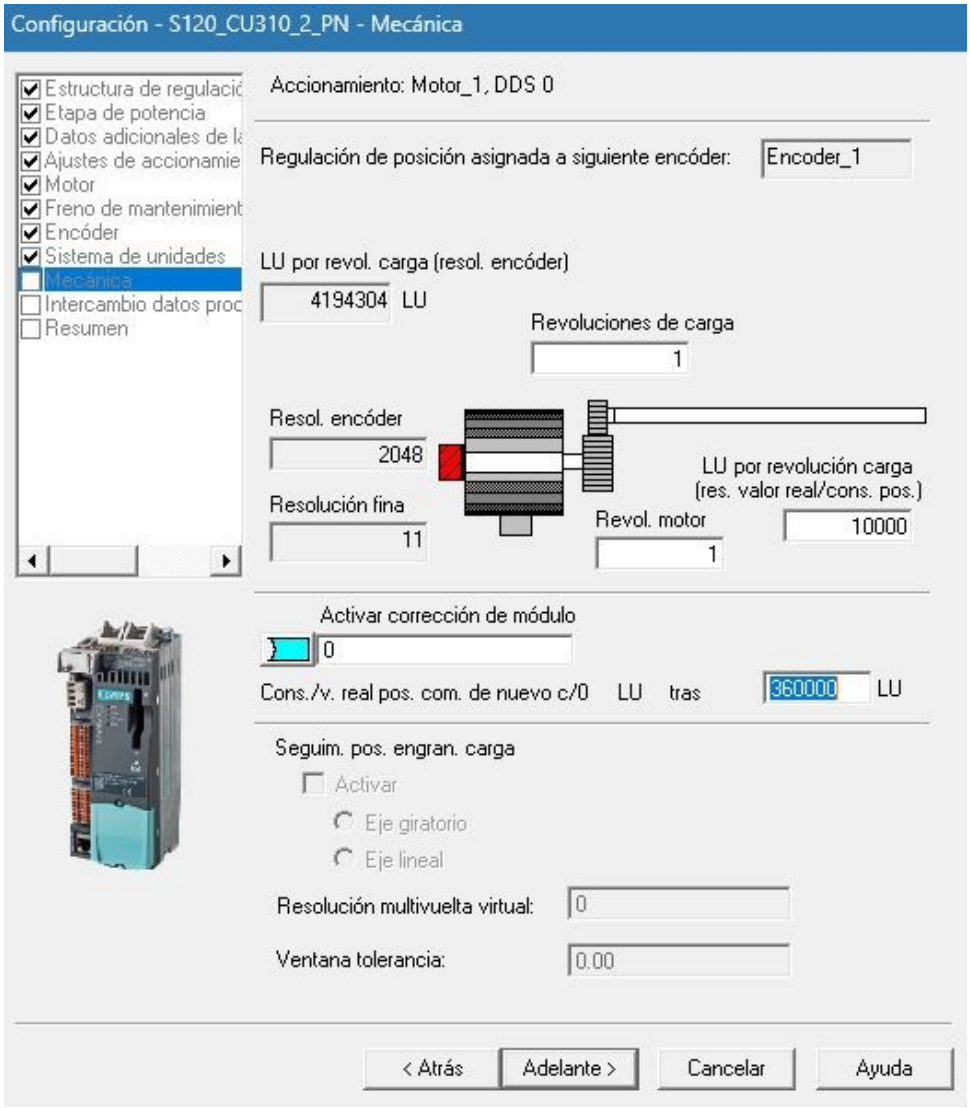


Figura 1.24: Mecánica (STARTER).

En la penúltima ventana, se selecciona el telegrama que se utilizará en el accionamiento. Dado que se emplea el bloque SINAPOS, se utiliza un Telegrama 111 proveniente de Siemens con una longitud de 12 palabras de entrada y 12 palabras de salida, como se muestra en la Figura 1.25.

Figura 1.25: Selección del telegrama de control (STARTER).

En la última ventana, se presenta un resumen de los cambios realizados en la configuración DDS. Esta ventana proporciona una visión general de todas las opciones seleccionadas y los ajustes realizados para asegurar que la configuración sea coherente y correcta antes de aplicar los cambios definitivos en el accionamiento. Esto se evidencia en la Figura 1.26.

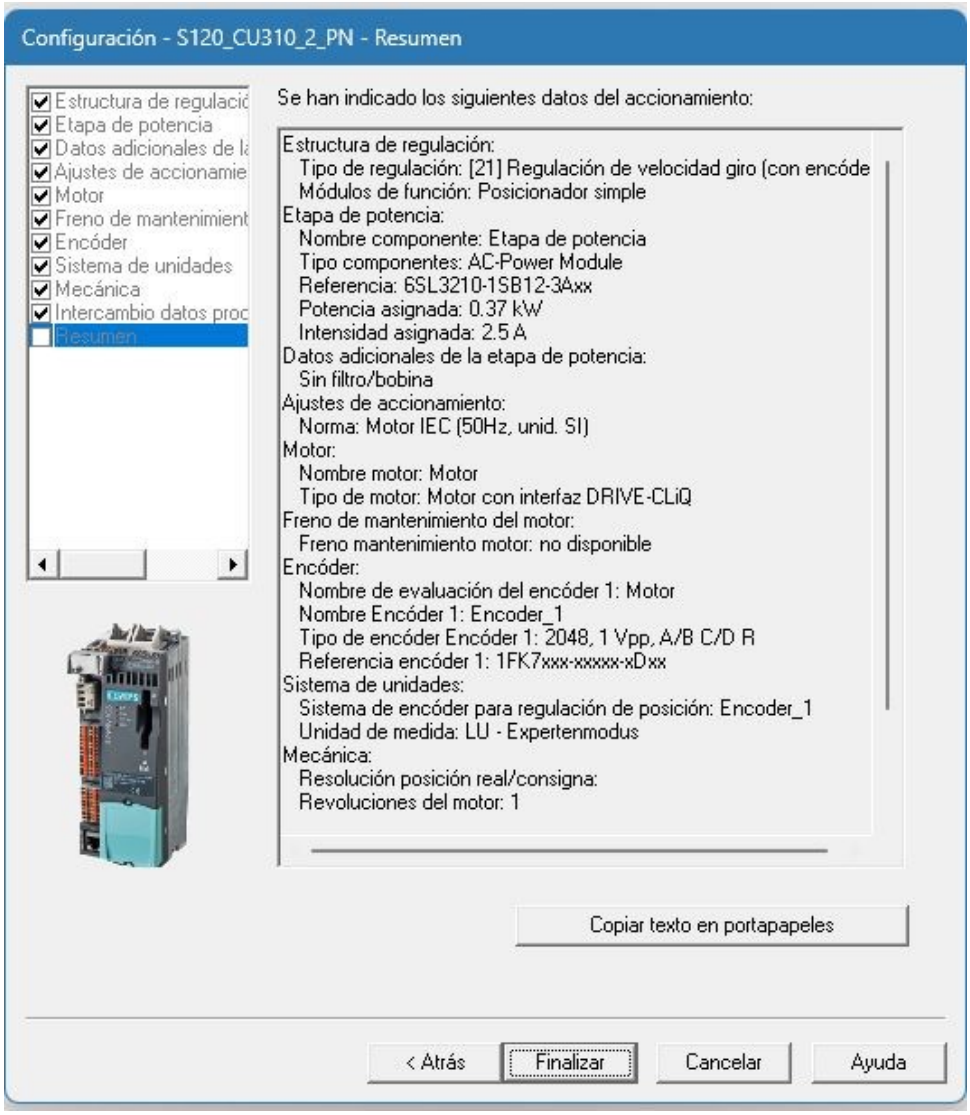


Figura 1.26: Cuadro de diálogo de resumen (STARTER).

Una vez configurado el DDS, se procede a ingresar a la configuración de Telegramas para confirmar la selección del Telegrama 111 en el accionamiento insertado en el proyecto, tal como se ilustra en la Figura 1.27.

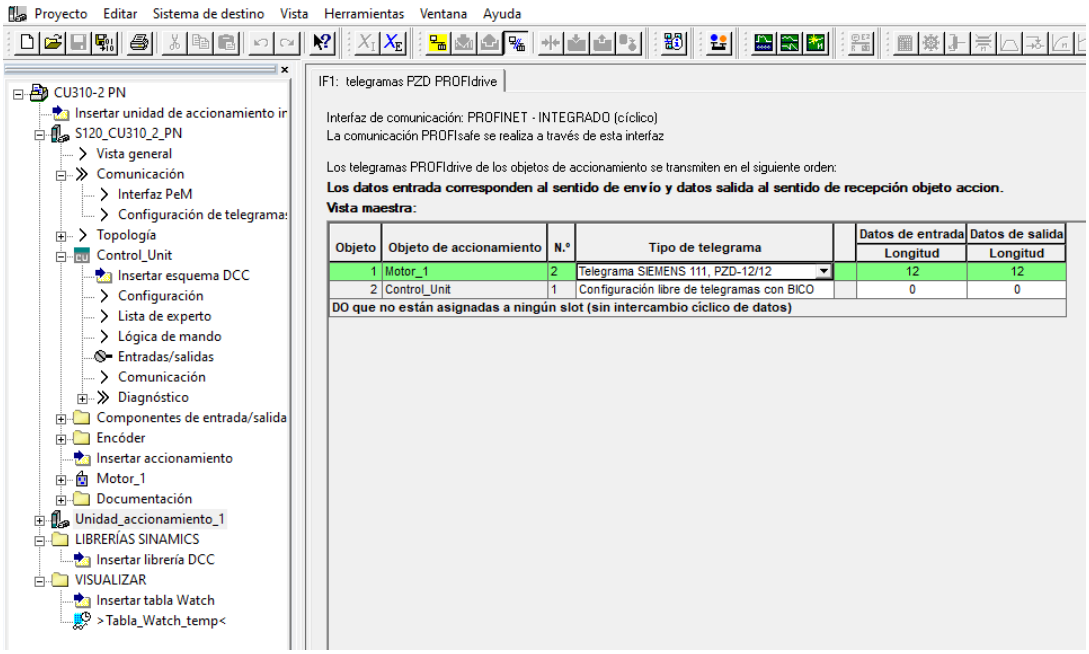


Figura 1.27: Configuración de telegramas (STARTER).

Se procede a modificar las salidas digitales, tomando en la DO8 la consigna de <r889.0: Palabra de estado secuenciador>, tal como se aprecia en la Figura 1.28. Esto permitirá que el relé de activación de la parte de potencia se active en el momento en que el accionamiento se comunique con el PLC.

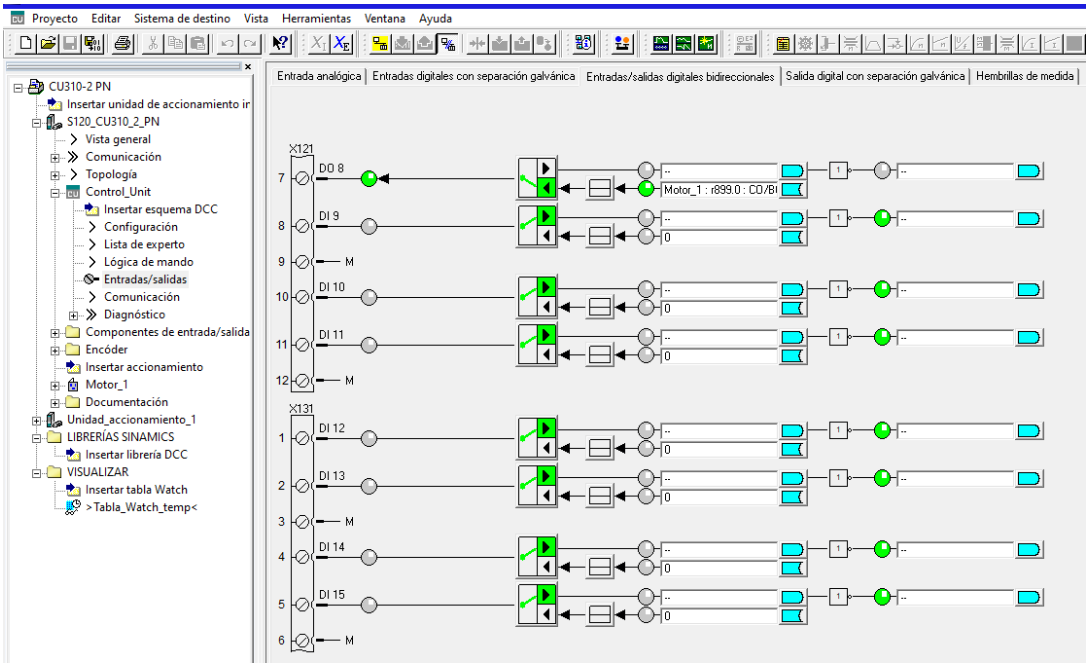


Figura 1.28: Configuración E/S (STARTER).

Luego se procede a configurar las limitaciones del motor, como se observa

en la Figura 1.29, donde se modifican la aceleración y desaceleración máximas, así como también la velocidad máxima del motor para asegurar condiciones óptimas de funcionamiento.

Además, se tiene la posibilidad de configurar el valor máximo de posición final, tanto positiva como negativa, en unidades de longitud (LU). Esta configuración permite establecer los límites del recorrido del motor, evitando que sobrepase ciertos valores y garantizando un control seguro y preciso del posicionamiento.

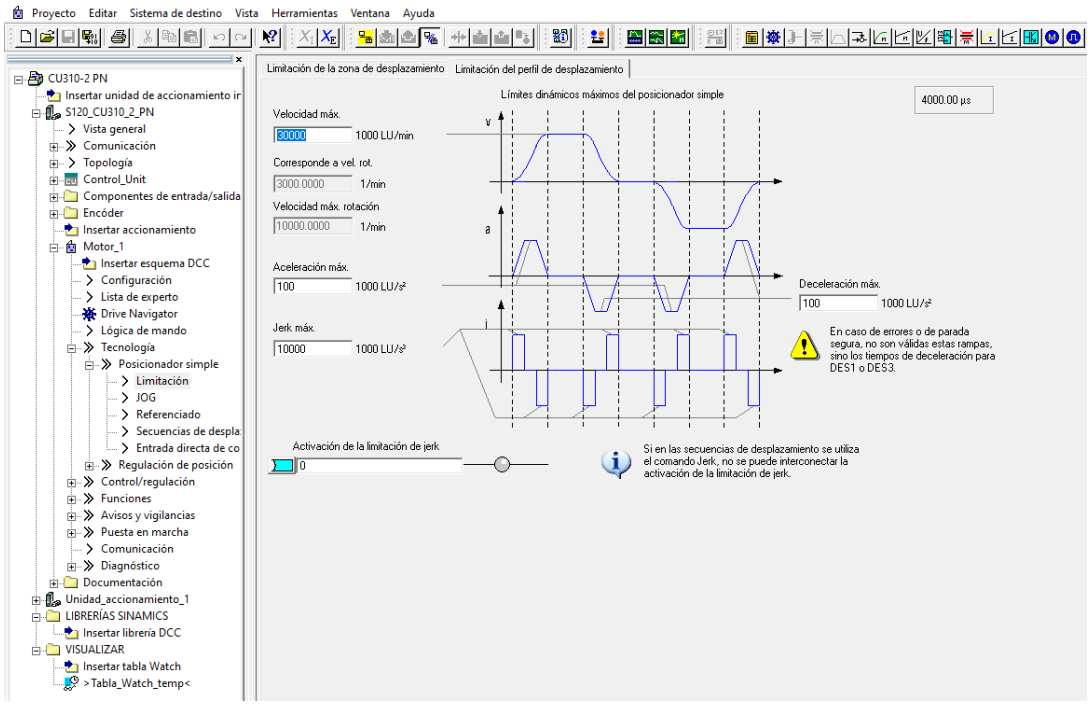


Figura 1.29: Configuración de limitaciones (STARTER).

En la Figura 1.30, se puede observar el Drive Navigator del accionamiento, que muestra las etapas de regulación, motor y encoder. Esta interfaz proporciona una vista detallada de la configuración y los ajustes realizados en cada una de estas etapas, lo que permite una supervisión y control precisos del accionamiento.

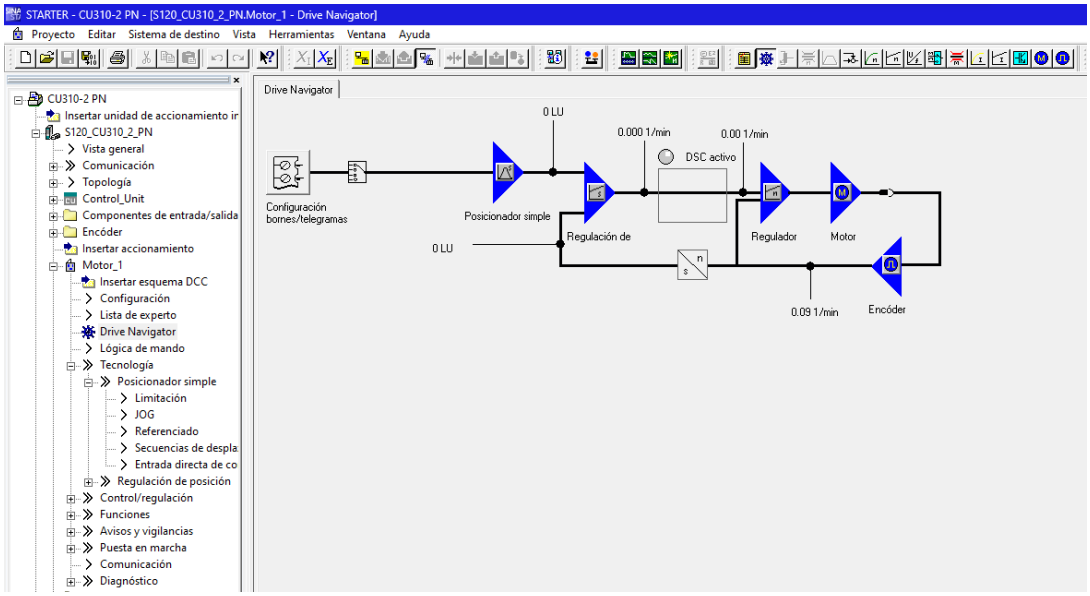


Figura 1.30: Drive Navigator (STARTER).

En la Figura 1.31, se muestra el panel de control desde el cual se puede interactuar y seleccionar opciones como el posicionador simple o la especificación de consigna N.

El botón verde permite arrancar el módulo, para ello se deberá tomar el mando y habilitar el accionamiento. Por otro lado, el botón rojo permite frenar el módulo en caso sea necesario.

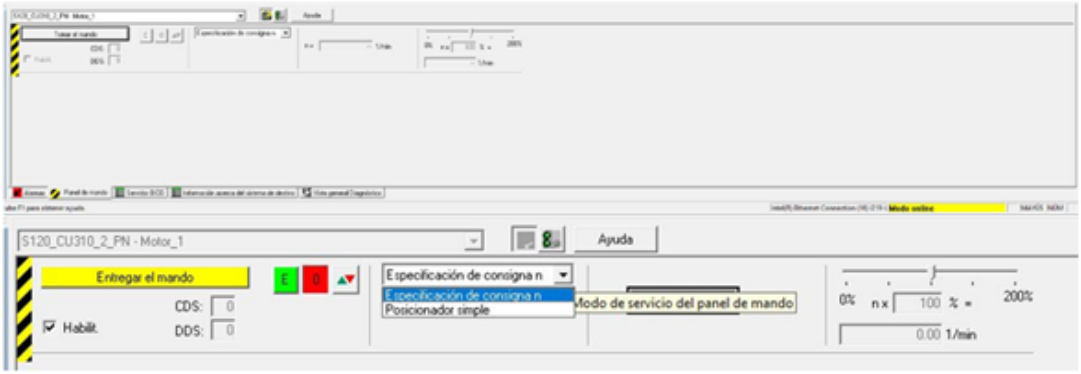


Figura 1.31: Panel de mando (STARTER).

Capítulo 2

Sistema de control e interfaz de comunicación

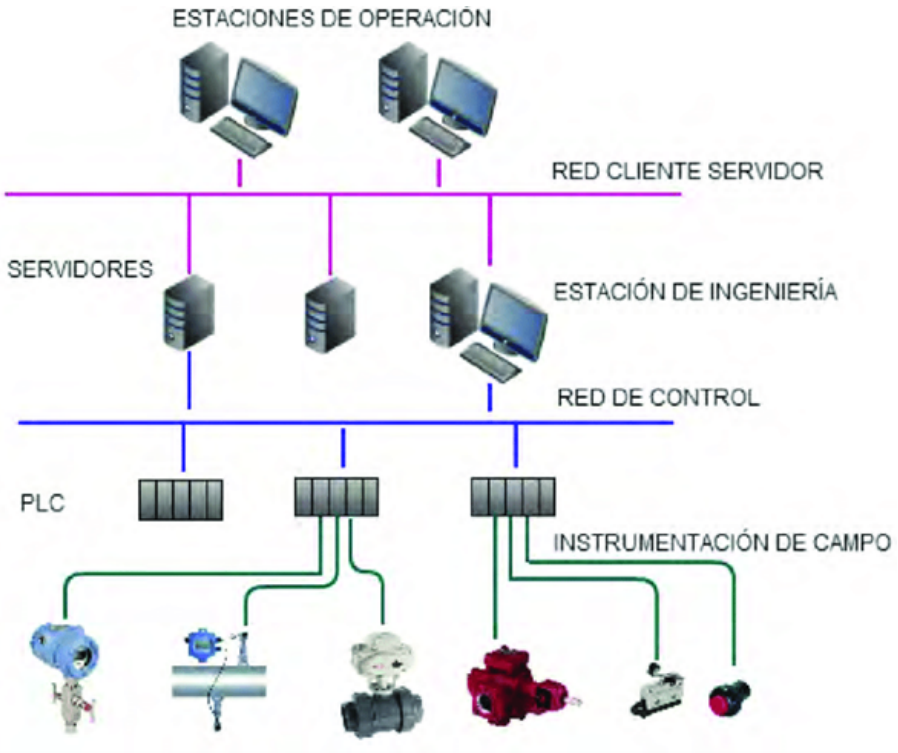


Figura 2.1: Topología de sistemas de comunicación industriales[11].

El sistema de control e interfaz de comunicación desempeña un papel fundamental en el funcionamiento eficiente y coordinado de los sistemas industriales modernos. Este sistema se encarga de la gestión y supervisión de diversas operaciones y procesos, permitiendo la interconexión y comunicación fluida entre los distintos

componentes y dispositivos involucrados [12].

En este capítulo se brinda una explicación detallada sobre el diseño y funcionamiento del controlador para el sistema de servo posicionamiento. Además se aborda la interfaz de comunicación, considerando sus características y las tecnologías que se emplean.

2.1. Control del servomotor con la CU310-2 PN (PROFINET)

La CU310-2 PN es un sistema compacto que se utiliza en aplicaciones de automatización industrial para el control preciso de motores. PROFINET, por otro lado, es un estándar de comunicación industrial ampliamente utilizado para la conexión de dispositivos en redes de automatización [12].

Para comenzar, es importante asegurarse de que tanto la CU como el servomotor estén conectados a una fuente de alimentación. La CU debe establecer una conexión con la red PROFINET para facilitar la comunicación con otros dispositivos y sistemas de control.

En primera instancia, se debe configurar la CU mediante el uso de un software de programación compatible, como TIA Portal. Este programa facilita la configuración y programación del controlador.

Dentro del software de programación, se crean bloques de función o programas específicos destinados a controlar el servomotor. Estos bloques de función pueden configurarse para establecer parámetros cruciales como la velocidad, posición, aceleración y frenado del motor.

La comunicación entre la CU y el servomotor se realiza a través de comandos y respuestas utilizando el protocolo PROFINET. Los comandos se envían desde el controlador al servomotor para establecer la acción deseada, como girar en una dirección específica o detenerse en una posición determinada. A su vez, el servomotor enviará respuestas al controlador, proporcionando información sobre el estado actual del motor, como la velocidad o la posición alcanzada [12].

Es importante destacar que el proceso de control del servomotor con la CU310-2

PN y PROFINET puede variar según las características específicas del servomotor y el entorno de aplicación. Por lo tanto, es recomendable consultar la documentación proporcionada por el fabricante del controlador y el servomotor, así como seguir las pautas y mejores prácticas recomendadas para una configuración y programación adecuadas.

2.1.1. Bloque de Función SINAMICS SINAPOS

El bloque de función SINAMICS SINAPOS desempeña un papel fundamental en el sistema de accionamiento. Dicho bloque de función ofrece un conjunto de características y funcionalidades avanzadas que elevan el rendimiento y la precisión de los sistemas de control de movimiento.

El bloque de función SINAMICS SINAPOS incluye algoritmos de control altamente eficientes que permiten un control de posición y velocidad preciso y dinámico [13]. Estos algoritmos están fundamentados en modelos matemáticos avanzados y sofisticadas técnicas de control, lo que garantiza un rendimiento óptimo en diversas aplicaciones industriales. Además, establecen distintas instrucciones que operan en su propia base de datos y en el programa general, tal como se muestra en la Figura 2.2.

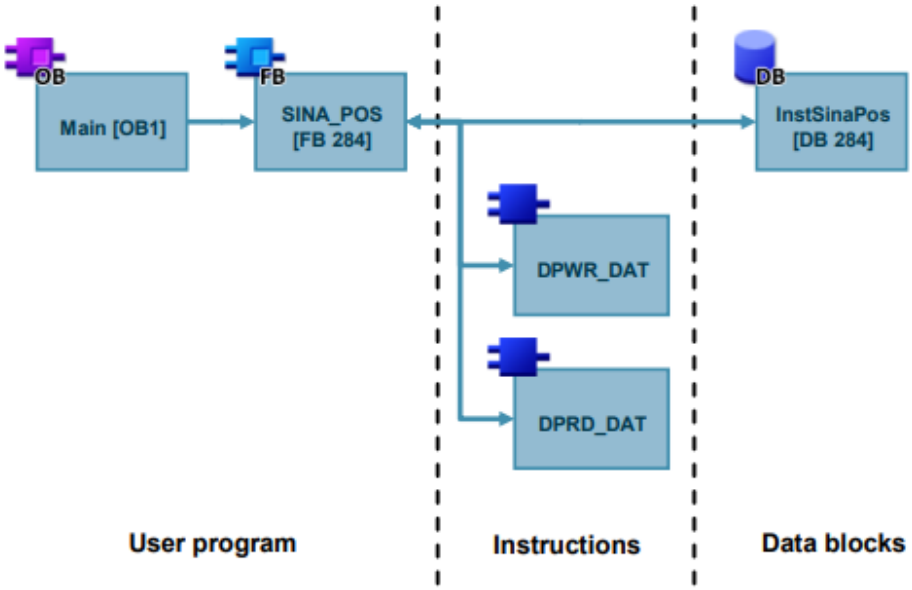


Figura 2.2: Vista general del bloque SINAPOS[13].

Tabla 2.1: Modos de operación SINAPOS.

ModePos (Bloque SINAPOS)	
Modo de Operación	Valor del Modo
Posicionamiento Relativo	1
Posicionamiento Relativo	2
Punto de Referencia Aproximación	4
Establecer Punto de Referencia	5
JOG	7
JOG Incremental	8

Además, el bloque de función SINAMICS SINAPOS presenta una amplia gama de funciones de seguridad integradas. Estas garantizan un funcionamiento seguro del sistema de accionamiento, protegiendo tanto a los operadores como a la maquinaria. Entre estas funciones se incluyen la supervisión del frenado, la detección de fallos y la protección contra sobrecargas, entre otras.

Otra característica destacada del bloque de función SINAMICS SINAPOS es su capacidad de adaptación a diferentes requisitos de aplicación [14]. Puede configurarse y ajustarse de manera sencilla para adaptarse a distintos tipos de maquinaria y procesos de producción. Esta versatilidad y flexibilidad permiten su implementación en una amplia gama de industrias y aplicaciones.

Para este bloque de función ilustrado en la Figura 2.3, se determina los parámetros de funcionamiento ya que contiene modos de posicionamiento a partir de un solo bloque.

Por otra parte, en la tabla 2.1 se muestran los distintos modos de operación del bloque SINAPOS.

2.1.2. El telegrama 111

EL telegrama 111 permite controlar la velocidad y el control de un posicionador simple. Este telegrama tiene una capacidad de 12 palabras para recepción y 12 palabras para la emisión.

Generalmente, el telegrama 111 se utiliza en conjunto con bloques EPOS para intercambiar datos hacia y desde la unidad. De esta manera, se logra el control del

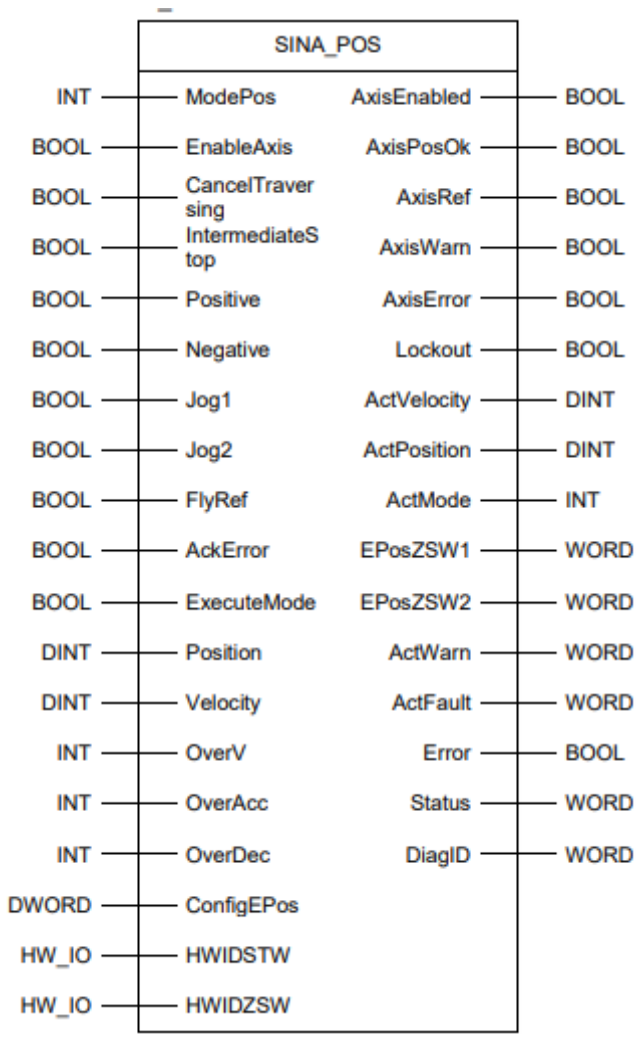


Figura 2.3: Bloque SINAPOS [13].

accionamiento mediante un PLC.

El mensaje del telegrama 111 ofrece la siguiente funcionalidad [15]:

- Posicionador básico.
- Valor de posición real de 32 bits.
- Valor de velocidad de 32 bits.
- Monitoreo de señales de vida.
- Códigos de error a través de una interfaz cíclica.
- Códigos de advertencia a través de una interfaz cíclica.

2.2. Programación del PLC SIMATIC S7-1500

La programación del PLC SIMATIC S7-1500 se lleva a cabo en la plataforma de TIA Portal V16. Esta plataforma cuenta con los archivos GSD del accionamiento S120 CU310-2 PN, lo que facilita la creación de la red industrial para el sistema de servo posicionamiento.

2.2.1. Configuración de dispositivos

En primer lugar, se genera un nuevo proyecto en TIA Portal V16, como se ilustra en la Figura 2.4. Luego, se procede a guardar el archivo con el nombre deseado y en una ubicación accesible en cualquier momento, como se aprecia en la Figura 2.5

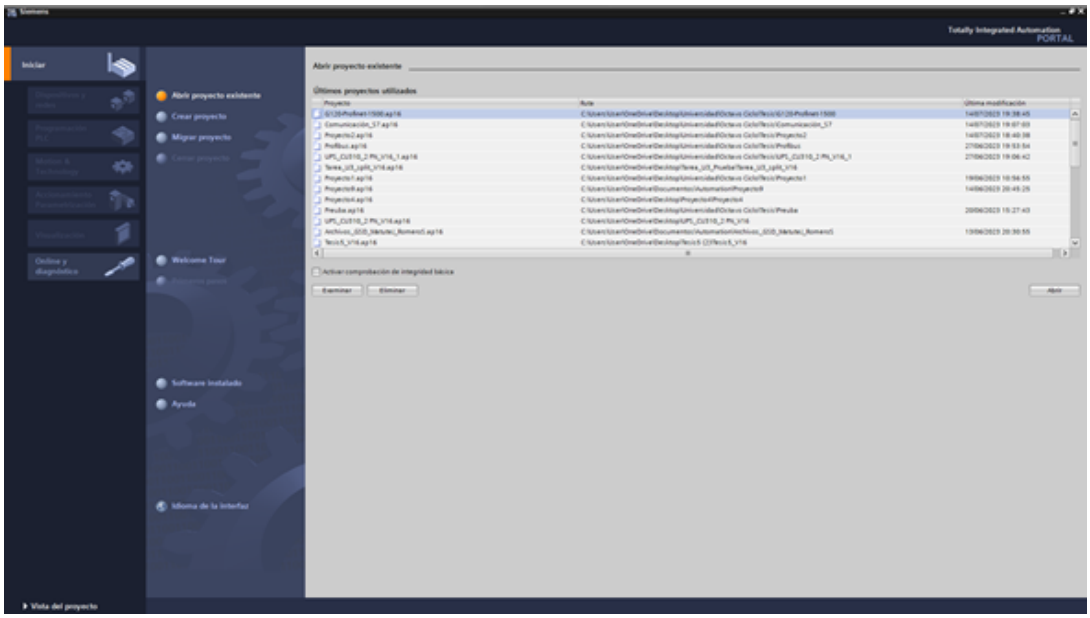


Figura 2.4: Creación de un proyecto en TIA Portal.

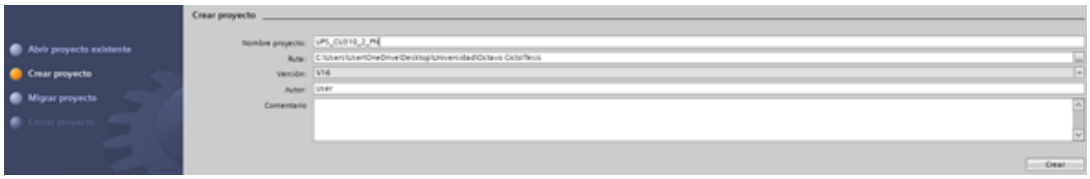


Figura 2.5: Detalles del archivo.

En el siguiente paso, se incorpora el PLC encargado del control del sistema de servo posicionamiento. Para ello, se selecciona la opción configurar un dispositivo,

como se muestra en la Figura 2.6. En este proyecto, se emplea la CPU 1516-3 PN/DP, y se verifica la versión de Firmware adecuada, como se puede apreciar en la Figura 2.7.

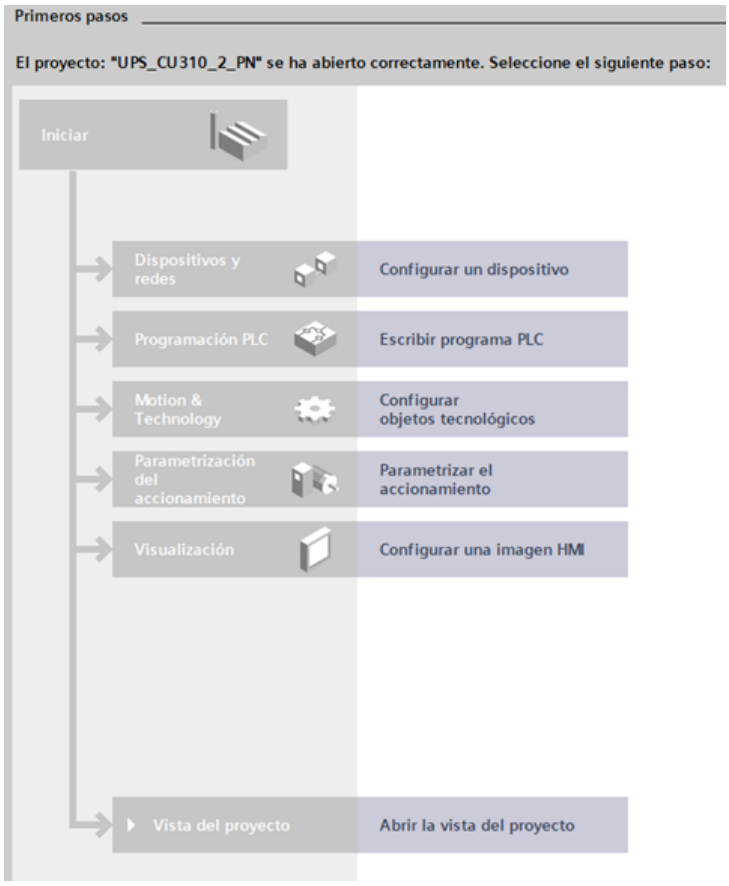


Figura 2.6: Cuadro de diálogo para configurar un dispositivo en TIA Portal.

Una vez que el programa se ha cargado y abierto, se procede a visualizar la CPU del PLC seleccionado, tal como se muestra en la Figura 2.8. A continuación, se agrega un módulo de potencia de 190W ubicado en la parte lateral derecha, en la sección PM del catálogo de hardware, como se puede observar en la Figura 2.9.

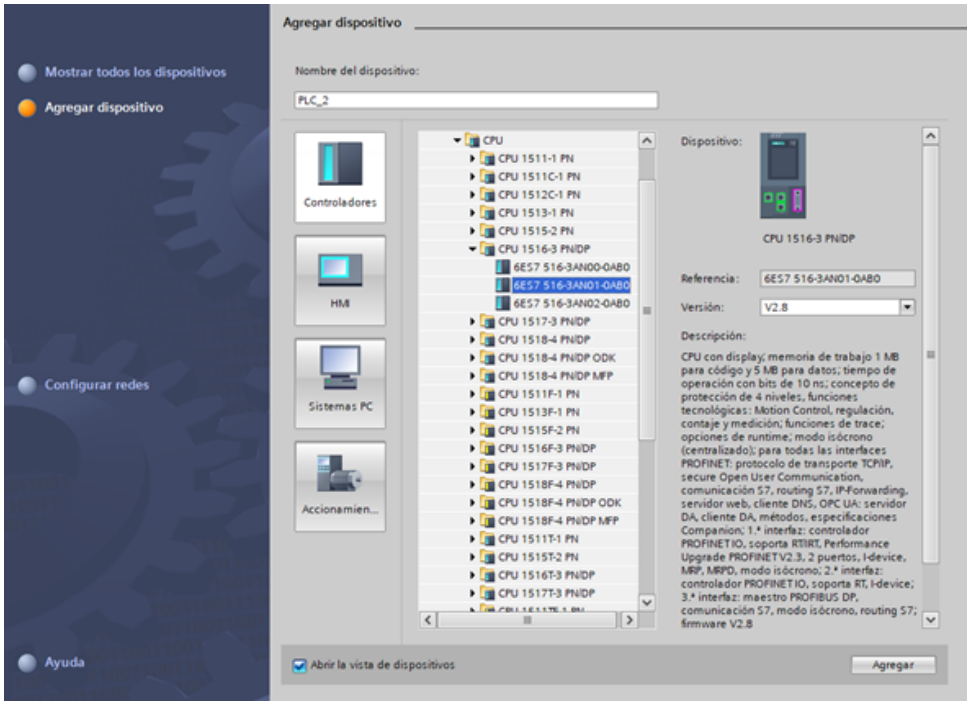


Figura 2.7: Cuadro de diálogo para agregar un dispositivo.

Al hacer clic en el módulo, este se añade automáticamente al PLC, como se ilustra en la Figura 2.10.

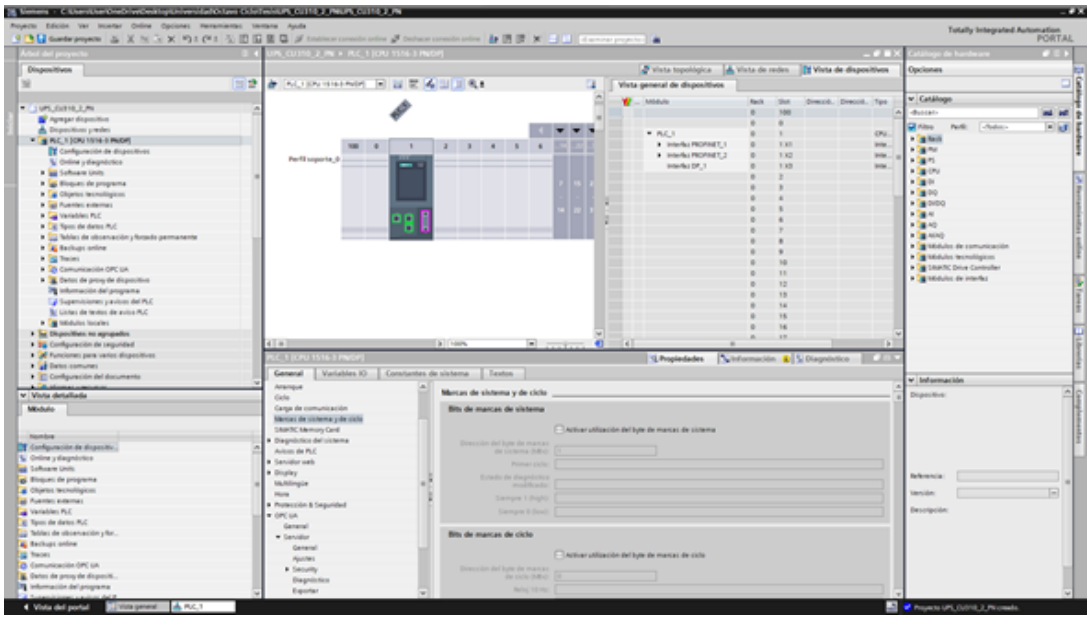


Figura 2.8: Vista general del proyecto con la CPU agregada.

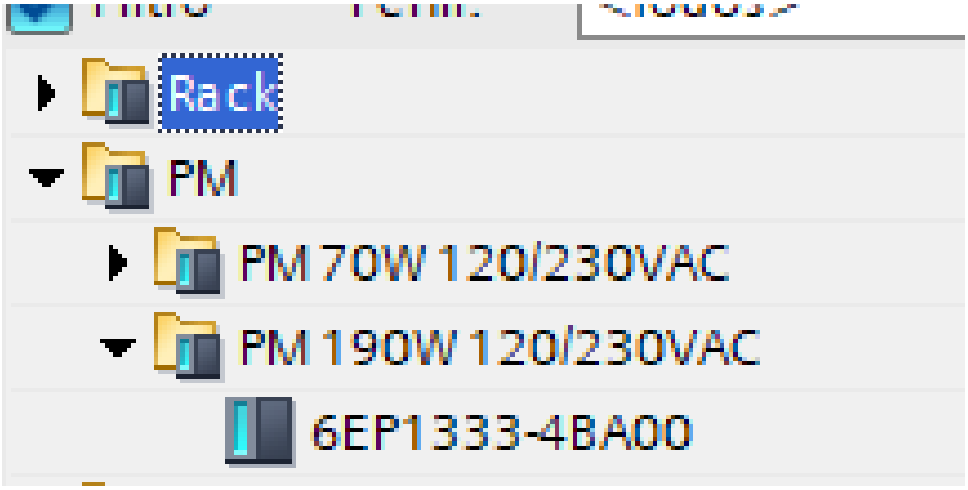


Figura 2.9: Catálogo de hardware.

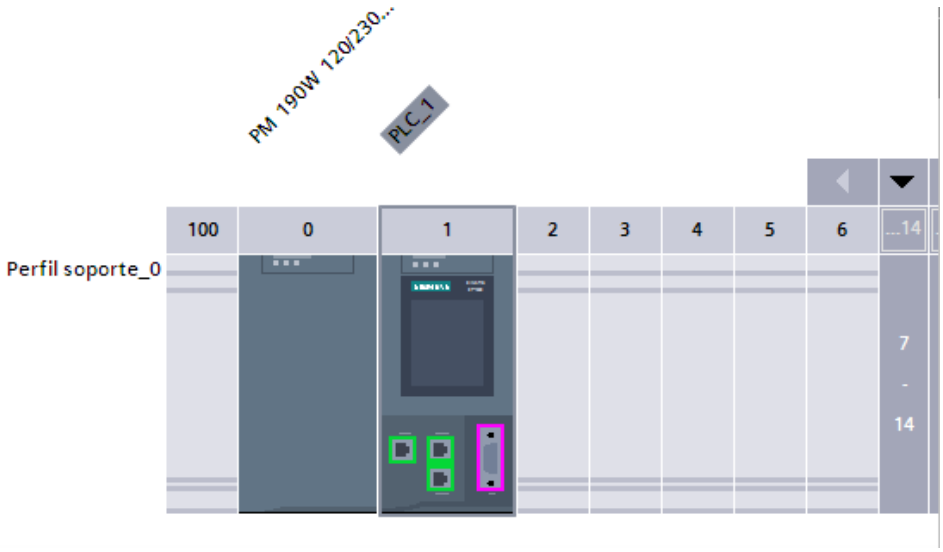


Figura 2.10: Vista general del proyecto con los componentes de hardware agregados

Después, se accede a la vista de Redes para agregar el driver correspondiente. En la sección del <Cátalogo de Hardware/Otros Dispositivos de Campo/PROFINET IO/Drivers/SIEMENS AG/SINAMICS/> se elige el GSD del Accionamiento S120-CU310-2PN con la versión de firmware de la unidad de control adecuada para el proyecto. En este caso se seleccionó la versión V5.2, como se observa en la Figura 2.11.

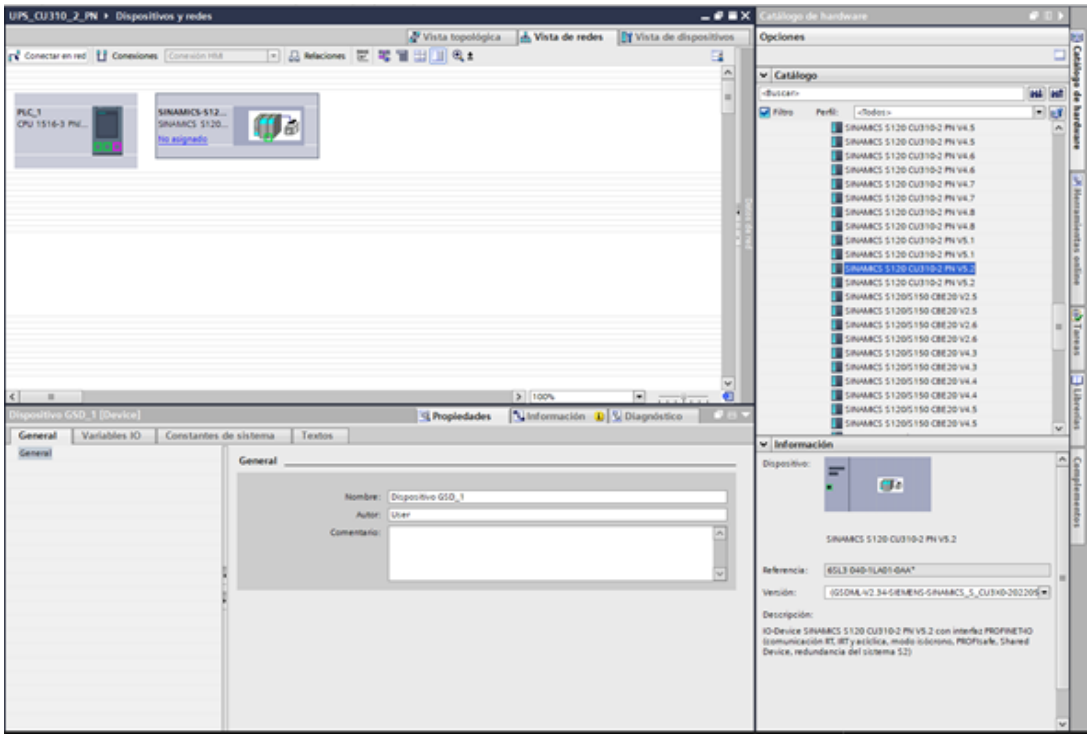


Figura 2.11: Vista de Redes y selección del accionamiento.

Al añadir el accionamiento, se despliega la vista del dispositivo, como se muestra en la Figura 2.12. En el catálogo de hardware, se visualizan los módulos de control que pueden ser usados, junto con el módulo de cabecera en caso de que se desee reemplazar el accionamiento.

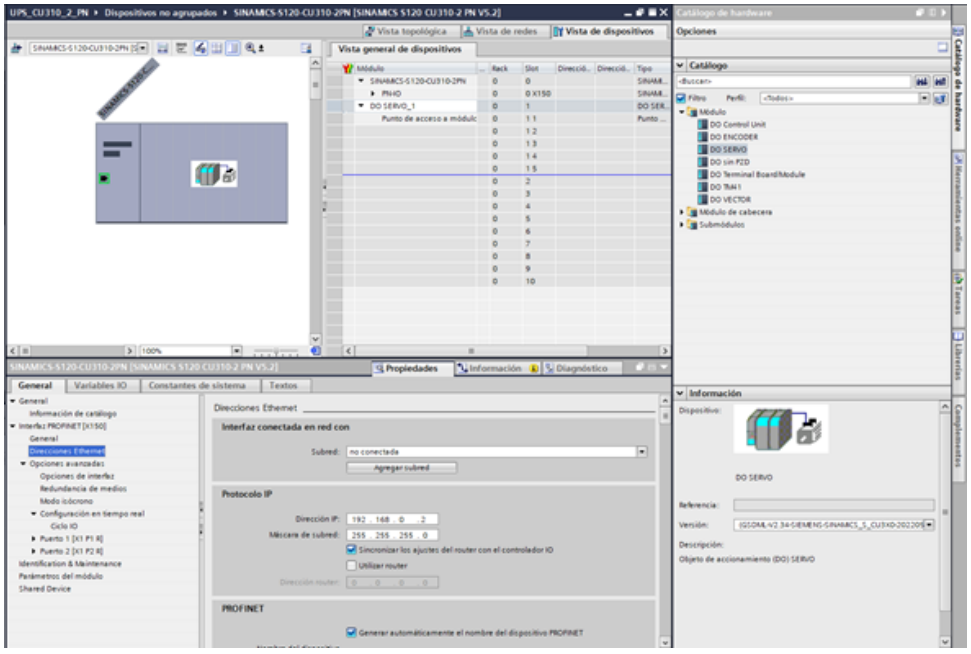


Figura 2.12: Catálogo de hardware - módulos.

Luego, se procede a seleccionar el módulo "DO SERVOz se carga una nueva carpeta llamada <Submódulos>. En esta carpeta se encuentran todos los Telegramas existentes en Tía Portal, como se observa en la Figura 2.13. Entre los telegramas disponibles, se elige el Telegrama SIEMENS 111 PZD 12/12, ya que es el adecuado para el control del posicionador simple en este proyecto.

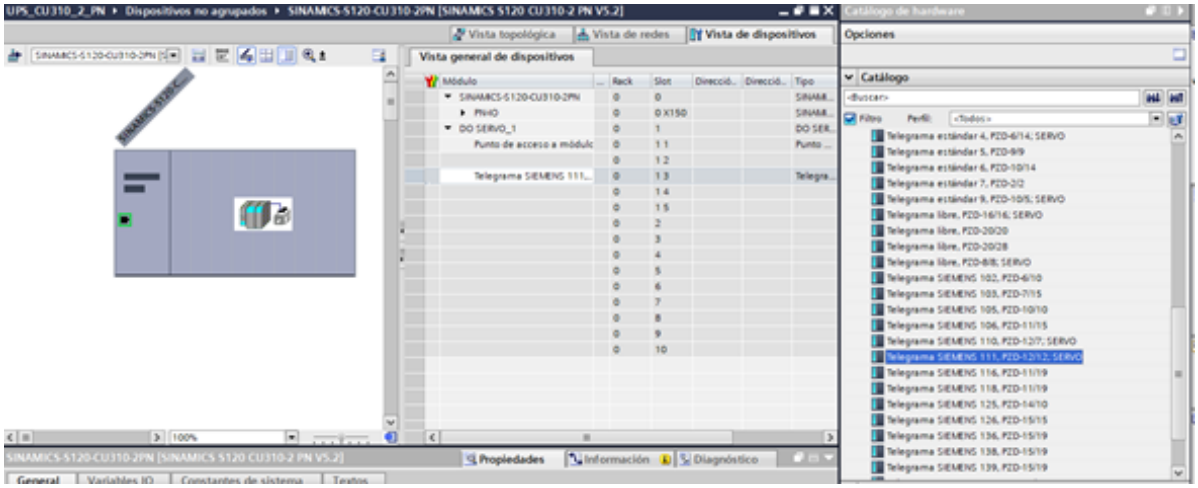


Figura 2.13: Cátalogo de hardware - telegramas.

Para finalizar la configuración del PLC y el accionamiento, se interconectan en una red, tal como se aprecia en la Figura 2.14. A continuación, se procede a modificar las direcciones IP de cada dispositivo. Para este proyecto se utilizaron las siguientes direcciones IP:

- Dirección IP del PLC = 192.168.0.1
- Dirección IP del accionamiento S120 = 192.168.0.2
- Dirección IP del computador = 192.168.0.8

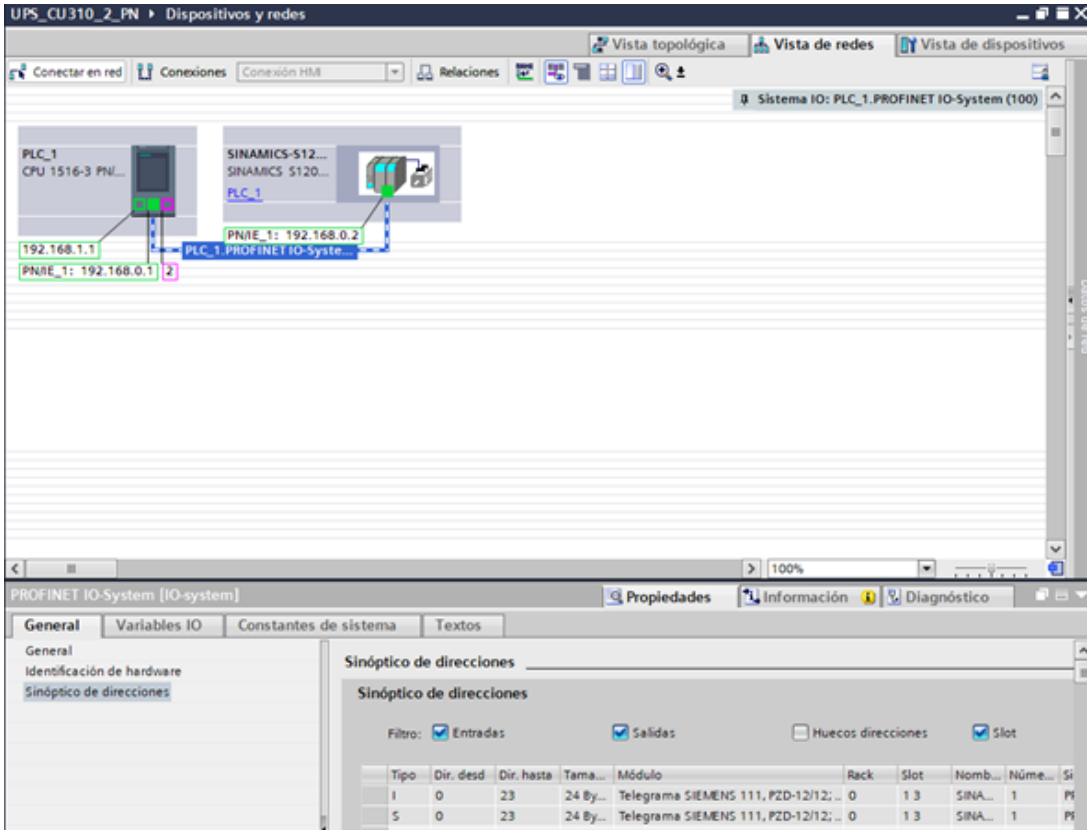


Figura 2.14: Vista de redes (Interconexión de dispositivos).

2.2.2. Bloques de programa

Una vez completada la configuración de los dispositivos, se procede a agregar los bloques de programa que serán utilizados en el proyecto. Para ello, en el árbol del proyecto, se pueden visualizar los bloques de programa del PLC seleccionado, como se ilustra en la Figura 2.15.

Para comenzar, se agrega una función con el nombre "POSITIONERING_M1". En esta función, es posible añadir diferentes bloques de funciones. En este caso, se incluye el bloque de función SINAPOS, como se muestra en la Figura 2.16.

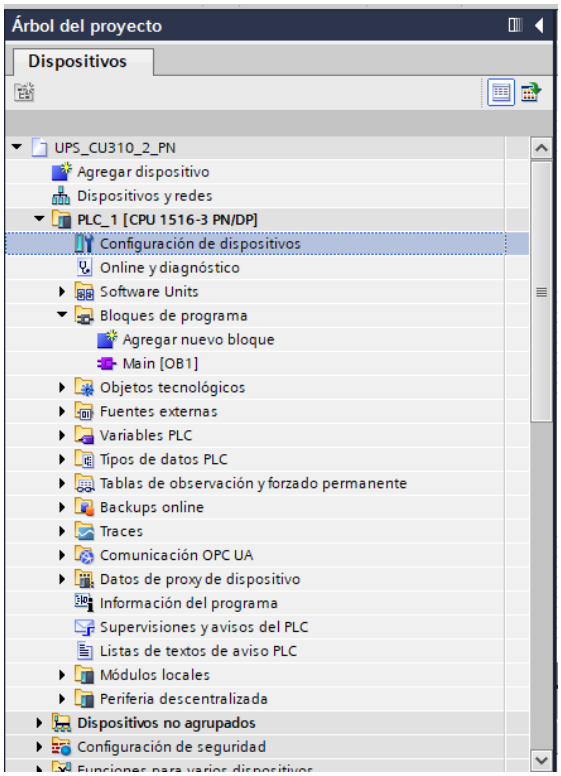


Figura 2.15: Vista del árbol del proyecto.

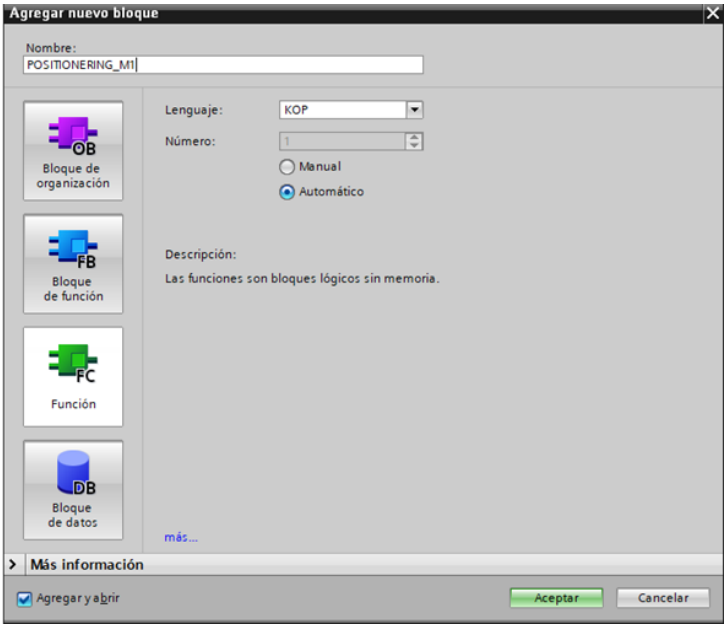


Figura 2.16: Cuadro de diálogo para agregar una función.

Para agregar el bloque SINAPOS, se debe dirigir a la sección de <Instrucciones/Paquetes Opcionales/SINAMICS>, tal como se muestra en la Figura 2.17. Al hacer esto, el bloque se cargará en la función y automáticamente se generarán

los bloques de función necesarios para el SINAPOS.

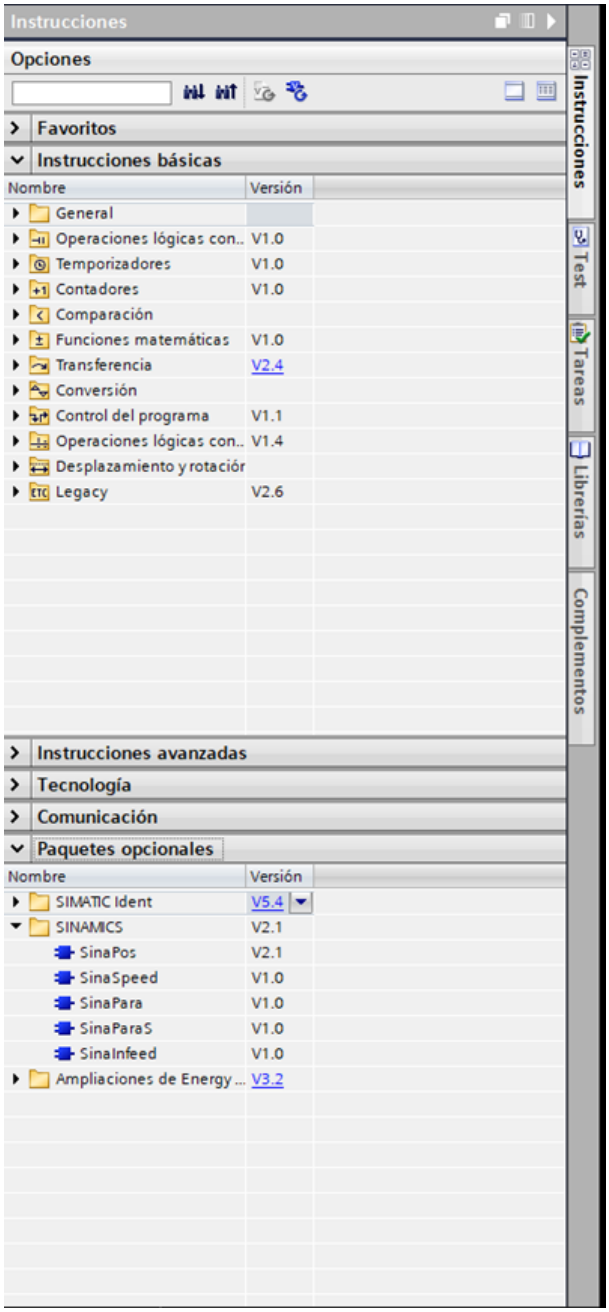


Figura 2.17: Ruta para agregar el bloque SINAPOS.

Una vez completada la carga del bloque SINAPOS, éste se puede agregar al programa de automatización (véase la figura 2.18). A partir de aquí, se identifican las entradas y salidas que serán enviadas y modificadas en la interfaz HMI de prueba.

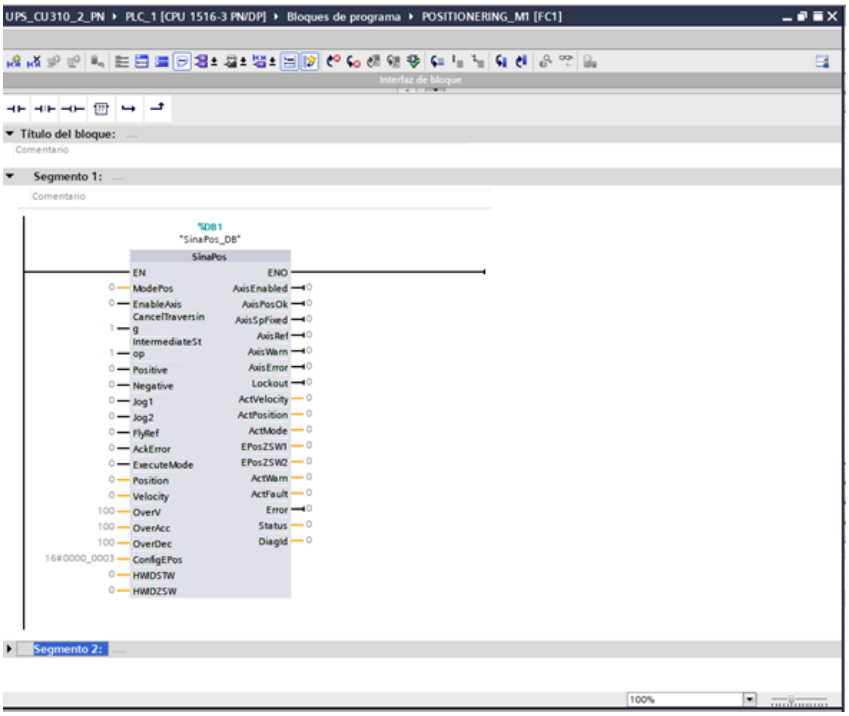


Figura 2.18: Bloque SINAPOS.

Para insertar las variables en el bloque SINAPOS de la función utilizada, se generan bloques de datos, como se muestra en la Figura 2.19. Estos bloques de datos son creados tanto para las variables de conversión, las que se utilizarán en la interfaz HMI, y las variables específicas del bloque SINAPOS.

Una vez obtenidos los bloques de datos, se añade un bloque de función que se encargue de realizar las conversiones correspondientes para cada indicador o señal de salida del SINAPOS, tal como se aprecia en la Figura 2.20. Es importante destacar que al ejecutar la función "POSITIONERING_M1", se generará automáticamente el bloque de función "SINA_POS".

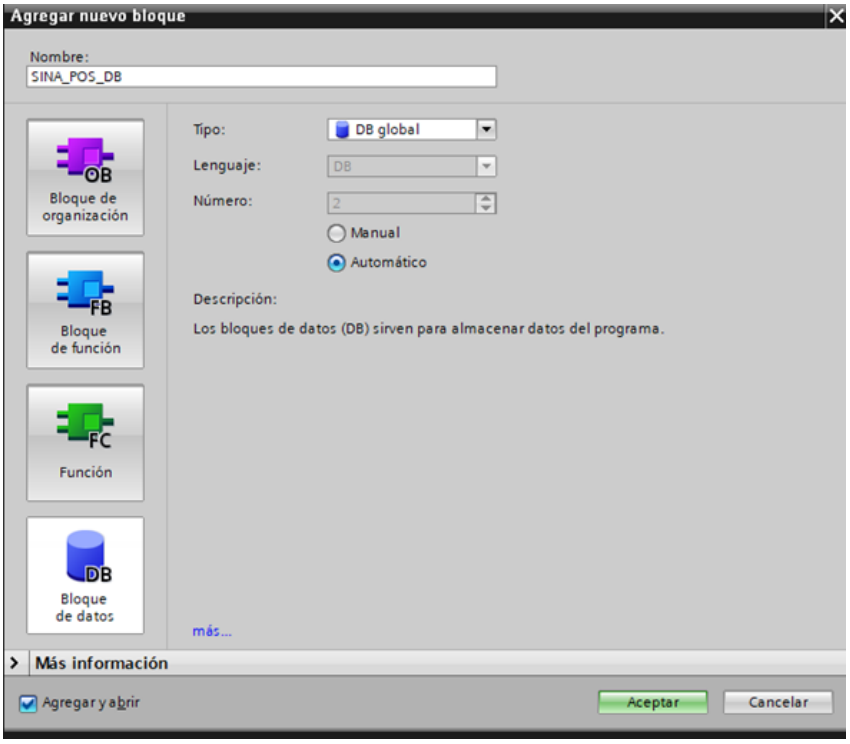


Figura 2.19: Cuadro de diálogo para agregar bloques de datos.

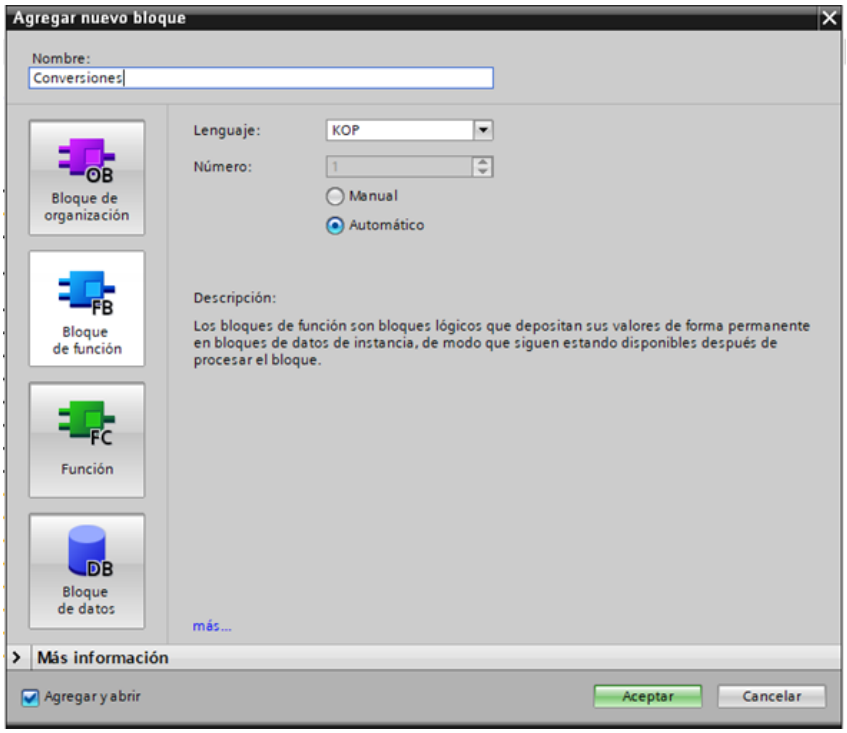


Figura 2.20: Cuadro de diálogo para agregar bloques de función (Conversiones).

Al agregar el bloque de función de conversiones, se desarrolla la lógica necesaria para cada una de las conversiones dentro de dicho bloque y los resultados

obtenidos se almacenan en el bloque de datos de "Conversiones", tal como se observa en la Figura 2.21.

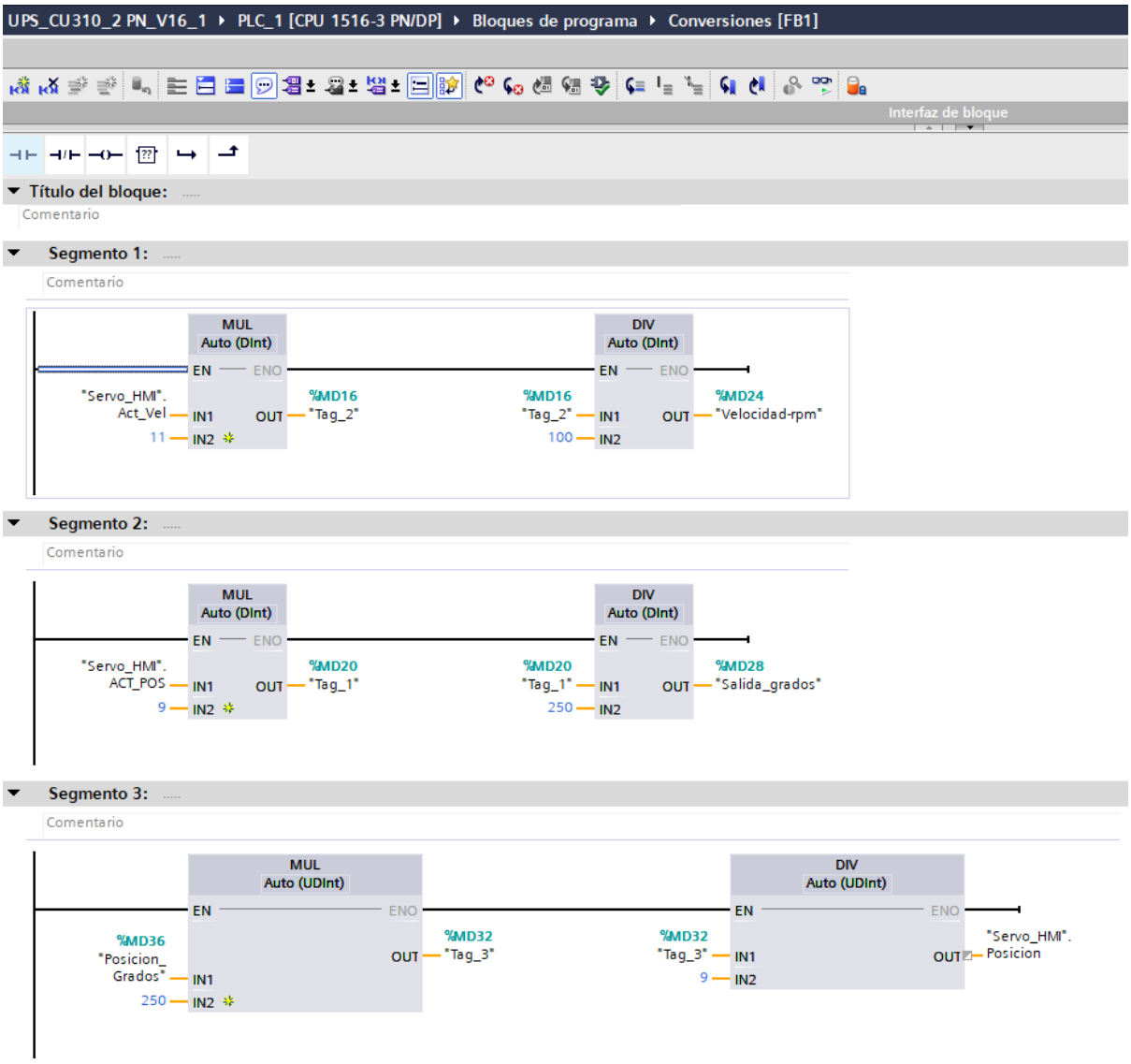


Figura 2.21: Operaciones de conversión.

Al concluir, se añaden los bloques de organización señalados en la Figura 2.22. Estos bloques de organización permiten una comunicación instantánea al momento de la conexión entre el PLC y el accionamiento, lo que corrige las alarmas presentes en el PLC.

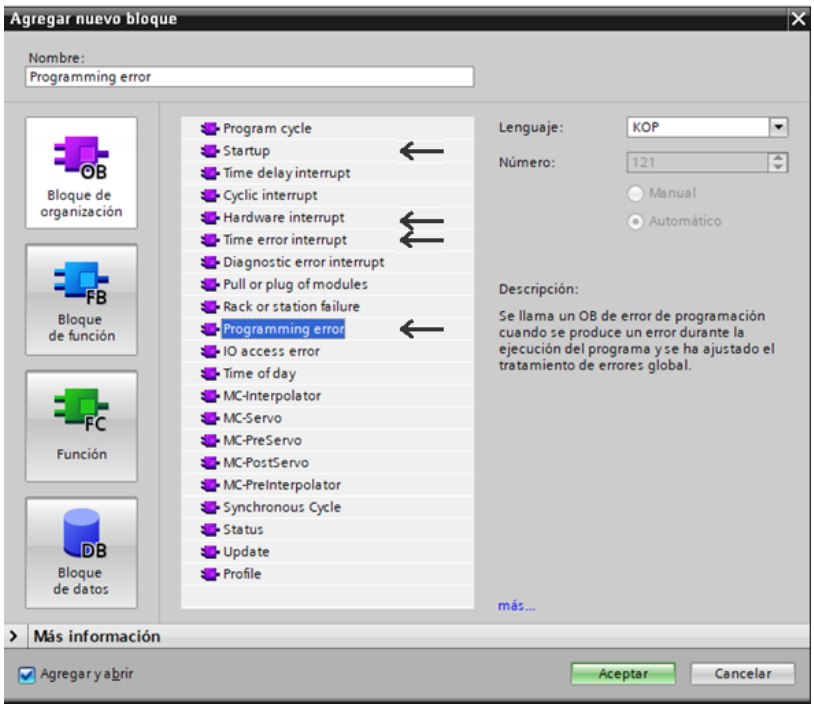


Figura 2.22: Cuadro de diálgo para agregar bloques de organización.

Para finalizar el proceso de configuración, se incorpora la función "POSITIONERING_M1" y el bloque de función "Conversiones" en el main (OB1), como se muestra en la Figura 2.23. Con esta acción, se habilitan las funciones y se ejecuta el programa con todas las configuraciones realizadas previamente.

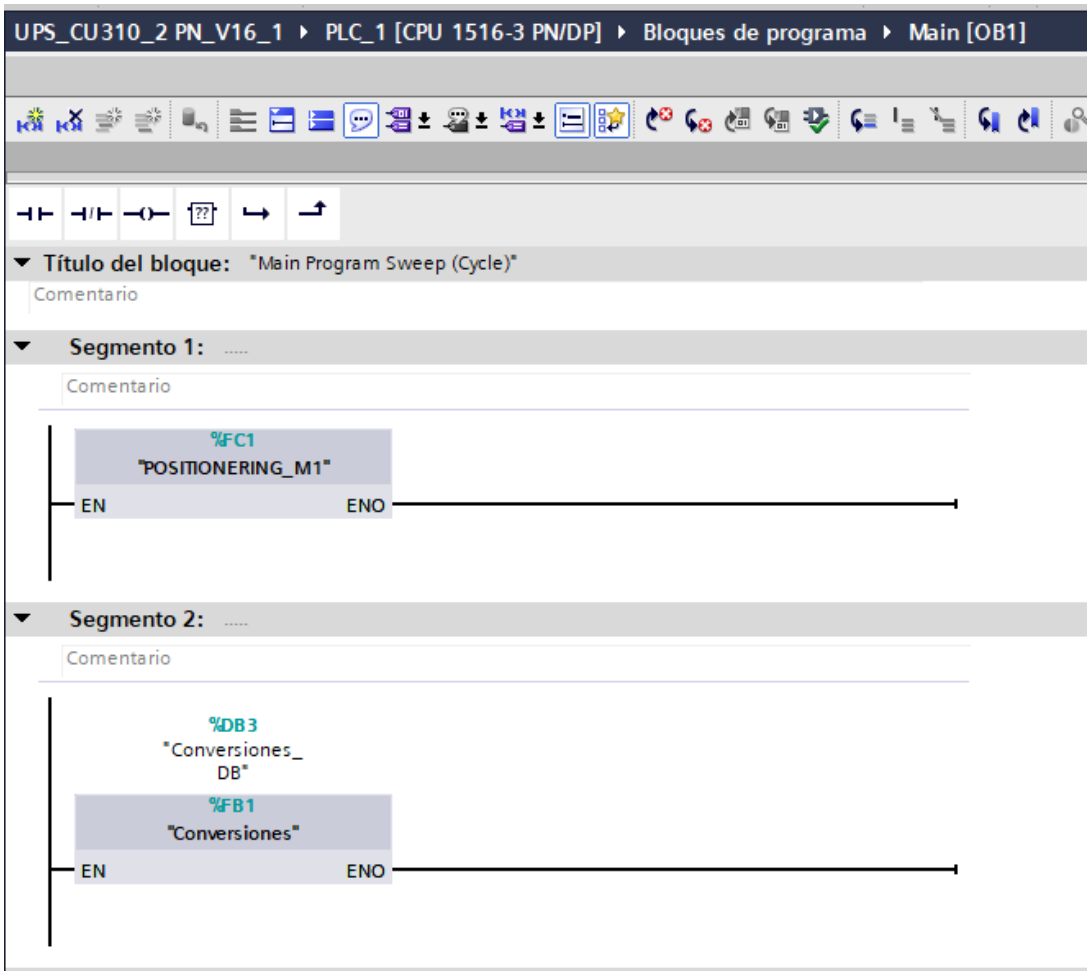


Figura 2.23: Vista del bloque principal - main(OB1).

En la Figura 2.24 se puede observar el árbol del proyecto, que incluye todos los bloques de programas generados para el adecuado funcionamiento del control y monitoreo del servo accionamiento con el PLC.

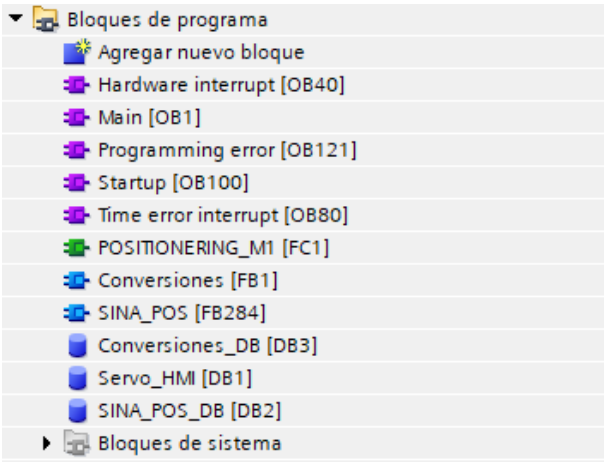


Figura 2.24: Árbol del proyecto con los bloques de programa creados.

2.3. Comunicación OPC UA

OPC-UA es un protocolo de comunicación entre diversos equipos comúnmente utilizada en aplicaciones de automatización industrial. Su estándar está basado mediante una arquitectura cliente-servidor en una plataforma independiente enfocada a servicios, y un mecanismo de publicación-suscripción. Esto permite a los clientes acceder a los recursos y a los datos de los servidores OPC-UA [16].

La capa de transporte define mecanismos optimizados para recibir y trasmitir datos entre sistemas OPC UA. La primera versión determina un mecanismo binario apoyado en el protocolo TCP, el cual consigue un alto rendimiento en las comunicaciones basadas en intranet [17].

2.3.1. Servidor OPC-UA en el controlador S7-1500

Para la creación del servidor OPC UA en el controlador lógico programable S7-1500, se realiza la siguiente configuración:

En la sección de Propiedades del PLC, se selecciona la opción OPC UA/Servidor/General. En la sección de Accesibilidad del servidor, se activa la opción 'Activar servidor OPC UA', como se muestra en la Figura 2.25. Al realizar esta configuración, se mostrarán en pantalla las direcciones del servidor, conocidas como 'EndPoint', las cuales permitirán acceder al servidor OPC UA.

Figura 2.25: Activación del Servidor OPC UA.

Luego, en la sección de OPC UA/Servidor/Ajustes, se pueden realizar modificaciones adicionales. Por ejemplo, se puede modificar el número del Puerto, siendo el valor utilizado en este caso el Puerto 4841. Además, se pueden ajustar

el Tiempo de sesión máxima, el número máximo de sesiones OPC UA y el número máximo de nodos registrados. En este caso, se dejaron las configuraciones predeterminadas. En la sección de 'Subscriptions', se puede visualizar el intervalo mínimo de muestreo y de envío de datos en milisegundos, como se muestra en la Figura 2.26.

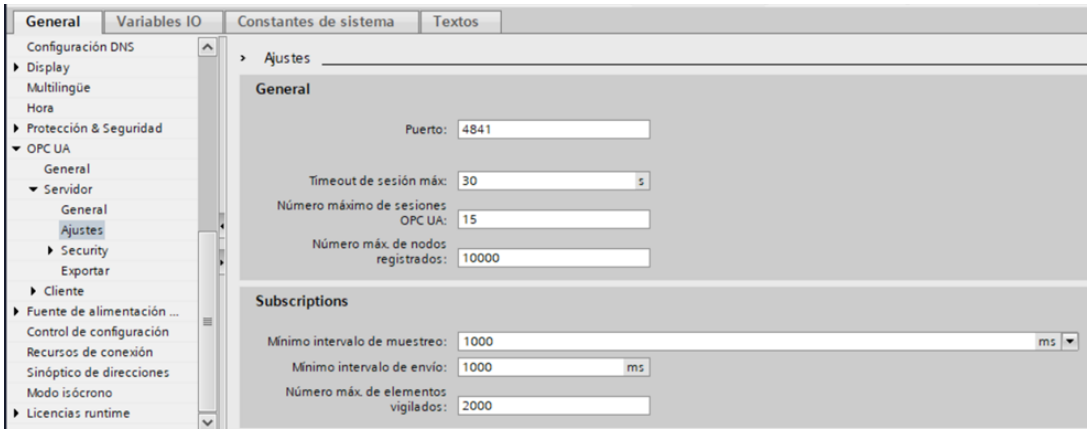


Figura 2.26: Ajustes del Servidor.

En la sección de OPC UA/Servidor/Security, se desmarcan todas las políticas de seguridad que vienen seleccionadas por defecto, excepto la primera opción 'Ninguna seguridad', que se mantiene seleccionada, como se muestra en la Figura 2.27.

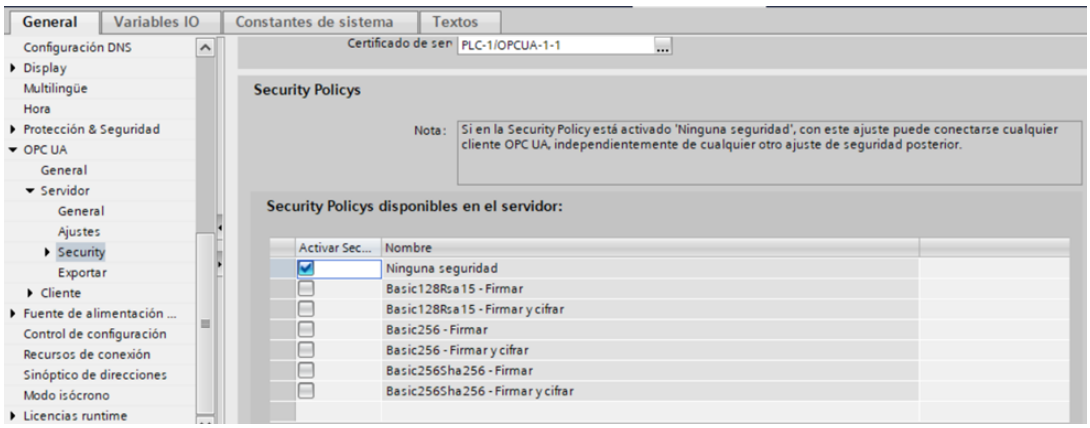


Figura 2.27: Políticas de seguridad del servidor.

En el mismo apartado de Security, como se ilustra en la Figura 2.28, en la sección de Autenticación de usuario, se puede activar la autenticación con nombre de usuario y contraseña. Para ello, en la sección de Administración de usuarios, se debe agregar el

nombre de usuario y su respectiva contraseña. En el proyecto en cuestión, se activará la autenticación de huésped, lo que permitirá ingresar al servidor sin la necesidad de un usuario.

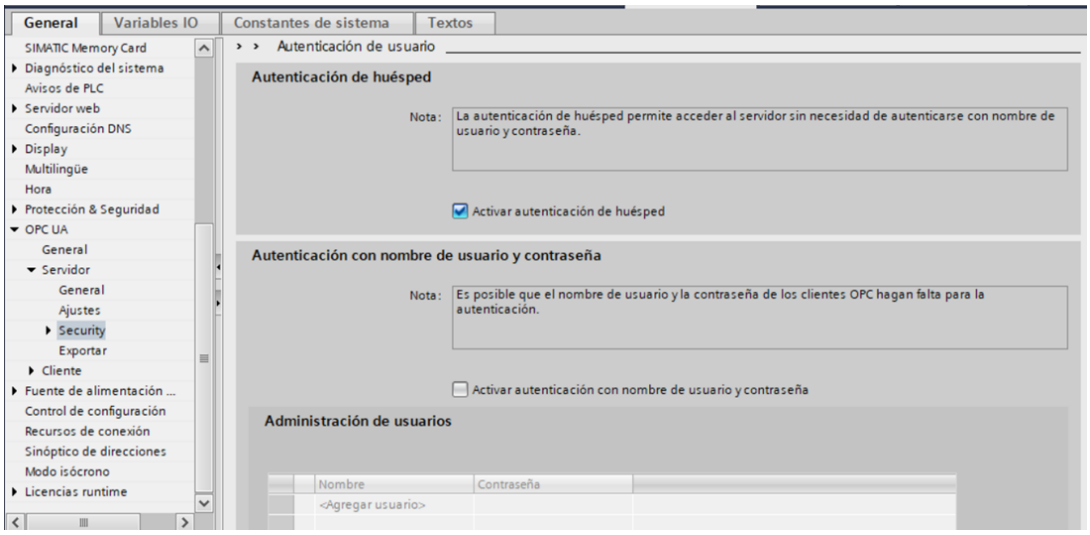


Figura 2.28: Autentificación del usuario de acceso al Servidor.

Finalmente, en la sección de General/Licencias runtime, en la parte del tipo de licencia adquirida, se seleccionará la licencia 'SIMATIC OPC UA S7-1500 medium', la cual debe ser la misma que está seleccionada en el tipo de licencia necesaria, como se ilustra en la Figura 2.29.

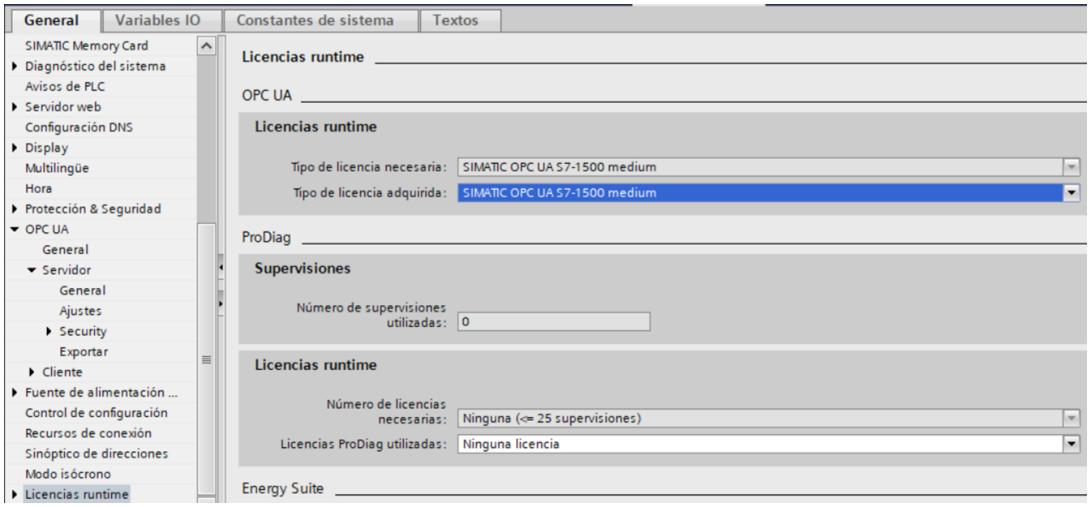


Figura 2.29: Licencia requerida para el Servidor.

2.3.2. Cliente OPC-UA

Para acceder al servidor OPC UA a través de Python, se utiliza la aplicación UA Expert en su versión 1.6.3, como se ilustra en la Figura 2.30. UA Expert es un cliente OPC UA que permite visualizar en tiempo real todas las variables que se encuentran en el controlador lógico programable (PLC) en el servidor.

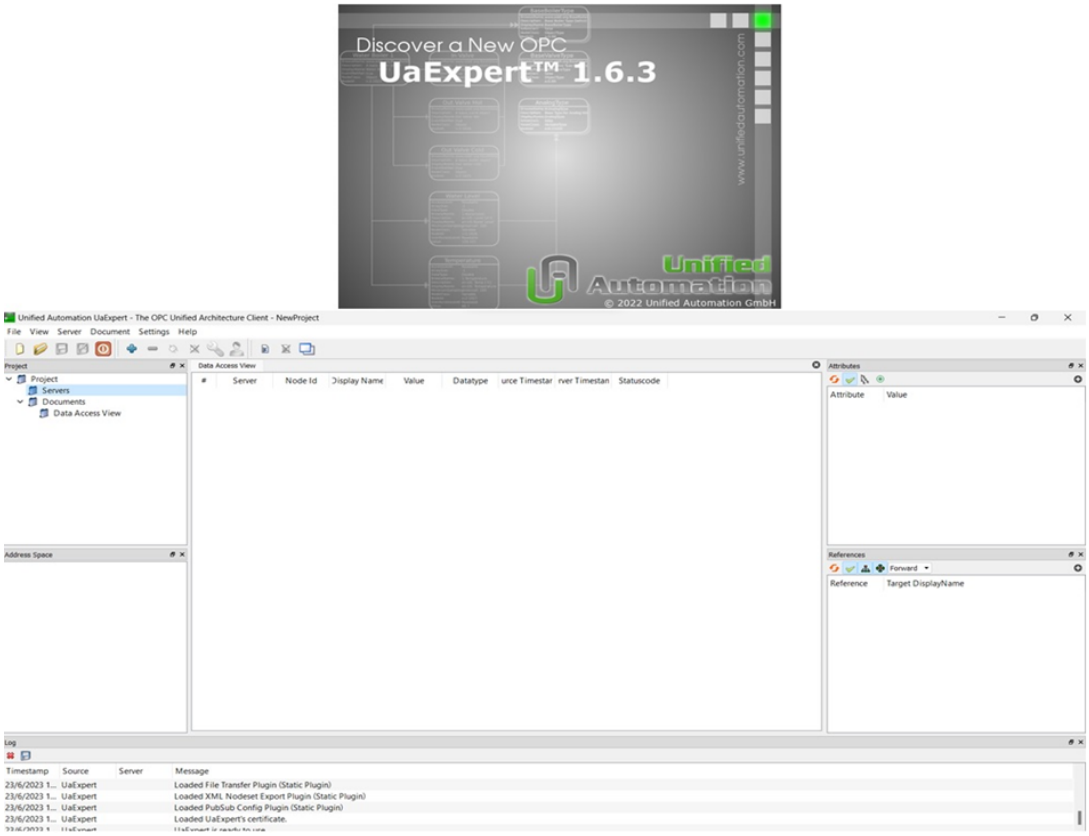


Figura 2.30: Unifed Automation UA Expert.

Para añadir el servidor en UA Expert, se accede a la sección de Project/Servers/Add. Al hacerlo, se abrirá una ventana llamada .Add Server", en la cual se puede añadir el servidor. En la parte de Çustom Discovery", se puede agregar el servidor, como se muestra en la Figura 2.31.

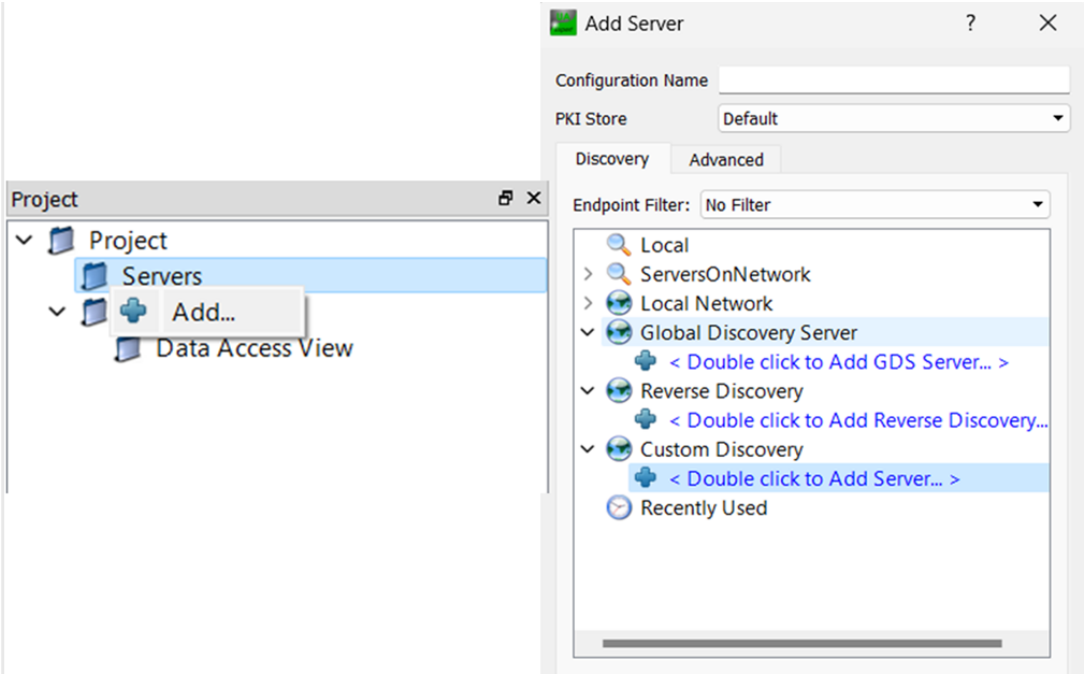


Figura 2.31: Vinculación del Servidor al proyecto.

Cuando se realiza doble clic en 'Custom Discovery', se abre una ventana para ingresar el URL del servidor. En este caso, se introduce la dirección del servidor 'EndPoint' previamente creada. Una vez ingresada la dirección, esta aparece en 'Custom Discovery' y al hacer clic en la dirección, se muestra el nombre del PLC utilizado, como se observa en la Figura 2.32. Cabe destacar que al hacer clic en "None", se verifica la licencia del servidor, lo que posibilita el acceso al mismo.

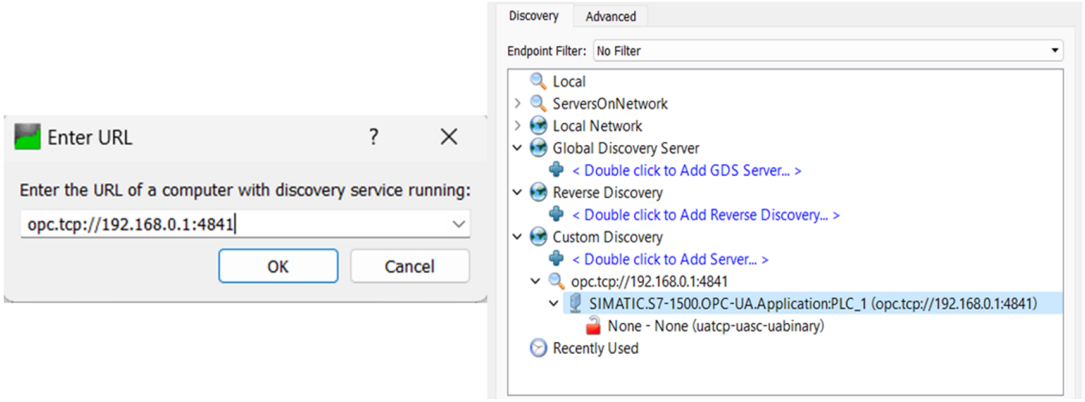


Figura 2.32: Ingreso del EndPoint y Visualización del PLC.

Finalmente, se puede establecer la conexión al servidor haciendo clic en él. Una vez hecho esto, se pueden visualizar las distintas variables de la programación

realizada en el PLC, tal como se ilustra en la Figura 2.33. El detalle de las variables se explica en la Tabla 2.2.

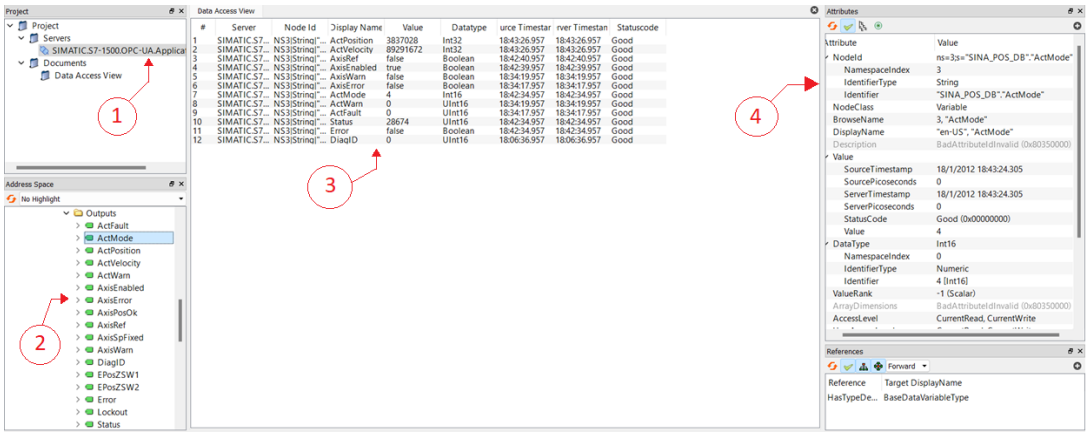


Figura 2.33: Visualización de las variables del Servidor.

Tabla 2.2: Descripción de las variables del servidor

1	Servidor SINAMIC S7-1500 OPC-UA conectado.
2	Espacio de direcciones de las variables del PLC.
3	Visualización de los valores de las variables seleccionadas en tiempo real.
4	Visualización de los atributos de la variable 'ActMode' seleccionada, en particular del nodo que nos permitirá acceder a la variable desde Python.

Capítulo 3

Sistema HMI con Python

En el siguiente capitulo se abordará la creación de la interfaz gráfica en Python utilizando las librerías PyQt5 y QtDesigner, las cuales servirán para el control y monitoreo del sistema de servo posicionamiento, incluyendo la estructura general del diseño, la disposición de los elementos de control, la visualización en tiempo real los parámetros del sistema, y la configuración de los diferentes modos de uso necesarias para lograr un control efectivo. También se incluirá la detección y notificación de alarmas, permitiendo al usuario identificar y resolver problemas potenciales de manera oportuna.

3.1. Estructura del sistema HMI

PyQt5 es una biblioteca de Python que permite crear interfaces gráficas de manera fácil y práctica utilizando solo unas pocas líneas de código. La claridad y legibilidad del código Python facilita enormemente la creación de interfaces gráficas. Además, PyQt5 permite a los desarrolladores utilizar la potencia y versatilidad de Qt en sus aplicaciones Python, brindando acceso a una amplia gama de funcionalidades para crear interfaces de usuario interactivas y atractivas [18].

Qt Designer es una herramienta que facilita la creación y diseño de interfaces gráficas de usuario (GUI) de forma intuitiva mediante el uso de widgets. Además, proporciona la flexibilidad de experimentar con diversos estilos y ofrece una vista previa en tiempo real de cómo se ve la interfaz [19].

PyQt5 y Qt Designer trabajan juntos para facilitar el desarrollo de interfaces gráficas de usuario en Python. Qt Designer proporciona una forma visual e intuitiva de diseñar interfaces, mientras que PyQt5 brinda los enlaces necesarios para utilizar ese diseño en una aplicación Python funcional.

3.1.1. Creación de la Interfaz Gráfica

Para la creación de la interfaz gráfica se hará uso de Qt Designer, la cual al iniciar presentará una ventana como se muestra en la Figura 3.1.

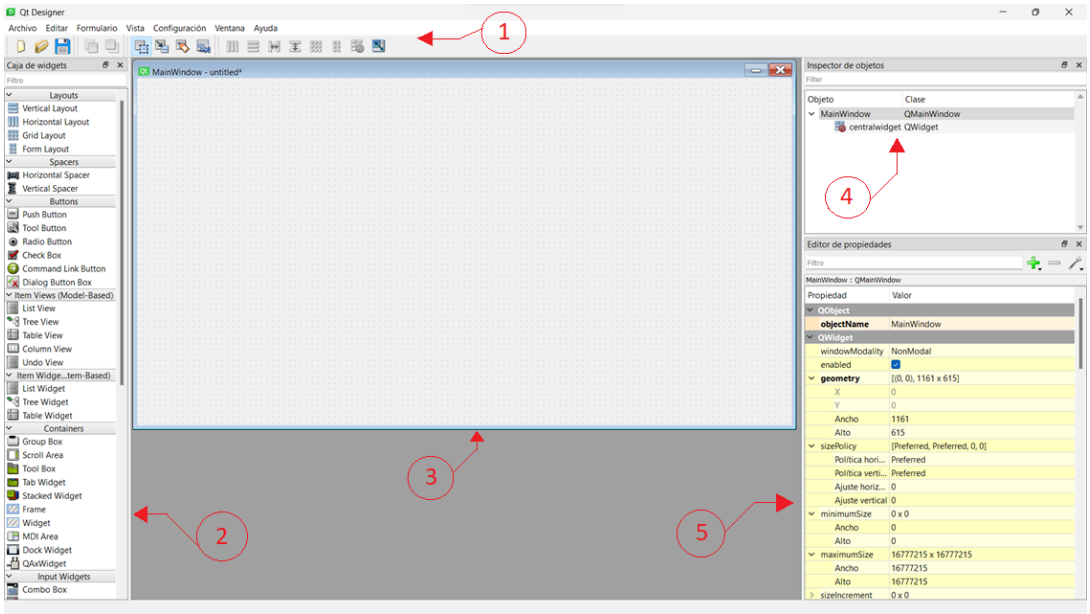


Figura 3.1: Ventana Principal de Qt Designer.

En la Tabla 3.1 se muestra una descripción de los elementos principales de la interfaz gráfica de Qt Designer.

Tabla 3.1: Elementos de la interfaz gráfica de Qt Designer

1	Barra de Herramientas.
2	Elementos gráficos comunes utilizados para el diseño.
3	Ventana Principal del Diseño de la Interfaz.
4	Lista de todos los objetos presentes en el diseño.
5	Propiedades del objeto utilizado como nombre, tamaño, etc.

En primer lugar, se procede a crear la barra de título de la interfaz, en la cual se ubican botones para cerrar, minimizar, expandir y restaurar la ventana. Además, se

incluirán dos botones adicionales que permiten cambiar el idioma de la interfaz entre español e inglés, como se muestra en la Figura 3.2.

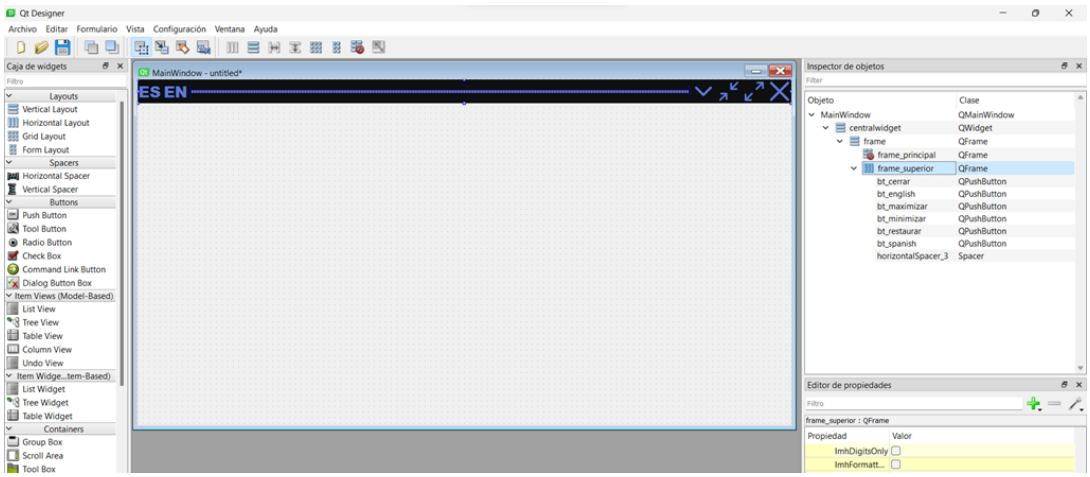


Figura 3.2: Barra de Titulo de la Interfaz.

A continuación, se procede a crear un frame que contendrá el menú de opciones de la interfaz. En dicho frame se ubican cuatro botones: uno para seleccionar el modo de operación del driver, otro para mostrar la velocidad del servo motor, un tercero para visualizar la posición actual del servo motor y un cuarto para mostrar las alarmas y fallos del driver. Además, se incluyen dos etiquetas para mostrar la hora y la fecha actual, así como dos indicadores para mostrar cuando el driver esté listo y cuando el punto de referencia se encuentre establecido, tal como se puede observar en la Figura 3.3.

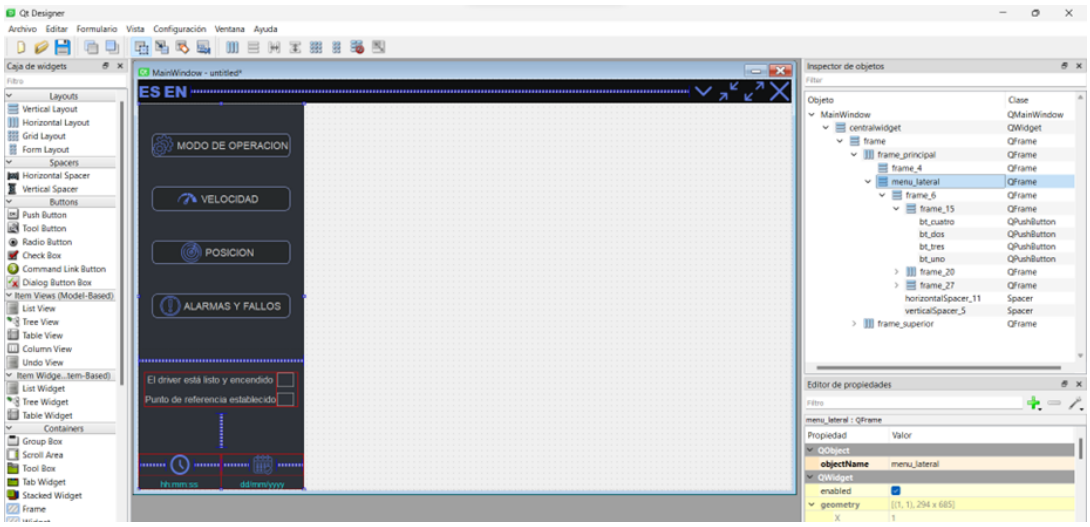


Figura 3.3: Menú lateral de opciones.

Posteriormente, se crea otro frame destinado al panel de control del driver. En este panel se incluyen los siguientes botones: uno para encender el driver, otro para ejecutarlo y ponerlo en marcha, y otro para restablecerlo. Además, se añaden dos botones para controlar la dirección de giro del servo motor. Asimismo, se agregan dos botones para incrementar y disminuir el valor del Jog. Por último, se incluye un botón de emergencia diseñado para corregir las posibles fallas del driver cuando se presenten. Todos estos elementos se pueden visualizar en la Figura 3.4.

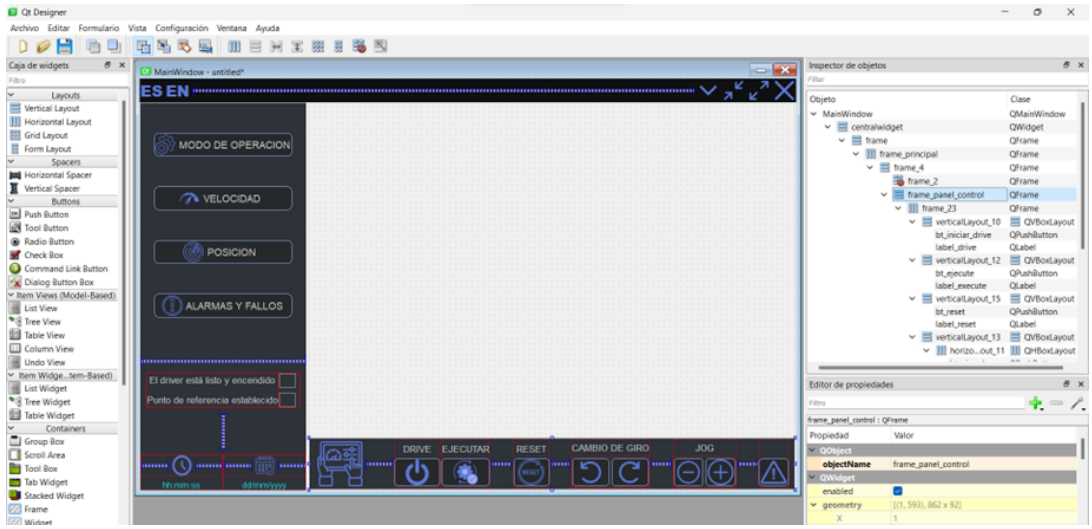


Figura 3.4: Elementos del Panel de Control.

Luego, se procede a crear un stackedWidget, el cual permite mostrar múltiples páginas apiladas una encima de la otra. Sin embargo, solo se mostrará una página a la vez. Con esta implementación, el objetivo es tener cuatro páginas disponibles que se mostrarán al presionar los botones correspondientes creados en el menú de opciones.

Para la visualización de la velocidad y la posición, se utilizan dos páginas idénticas. Cada página contendrá dos frames: uno para la representación gráfica de los valores actuales de velocidad y posición y otro para la representación gráfica de los puntos de los valores en función del tiempo. Esta estructura se ilustra en la Figura 3.5.

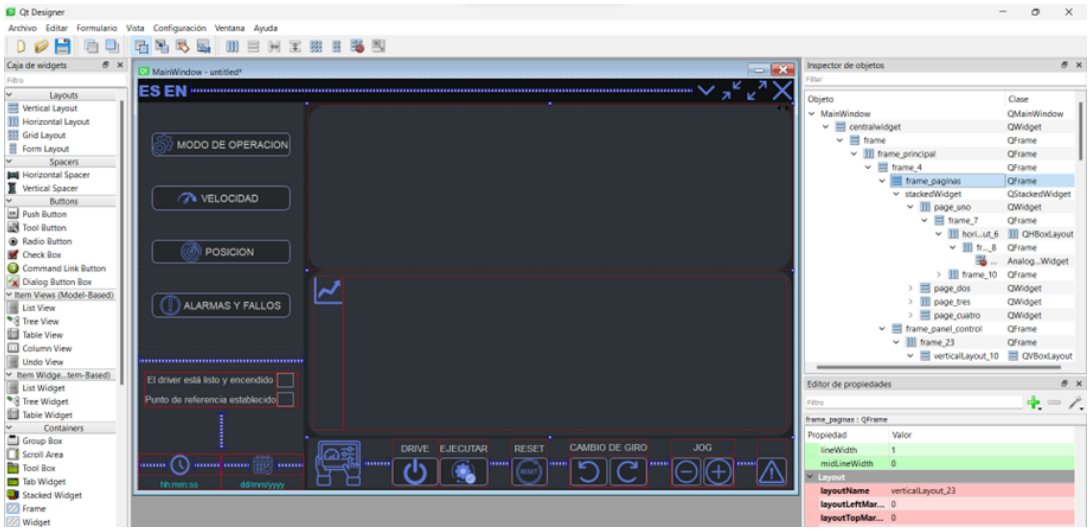


Figura 3.5: Páginas de visualización (velocidad y posición).

Otra página se utiliza para la visualización de los modos de operación del driver. Esta página consta de diferentes QRadioButtons que permiten seleccionar entre los siguientes modos de operación: Posicionamiento Relativo, Absoluto, Establecer el punto de referencia, Punto de referencia aproximado, modo Jog y jog incremental. Además, se incluye un botón de confirmación correspondiente a la selección del modo. Asimismo, se agregan dos QSpinBox para ingresar los valores de velocidad y posición respectivos. Para brindar información sobre el modo seleccionado, se incluirá una etiqueta en la página, tal como se muestra en la Figura 3.6.

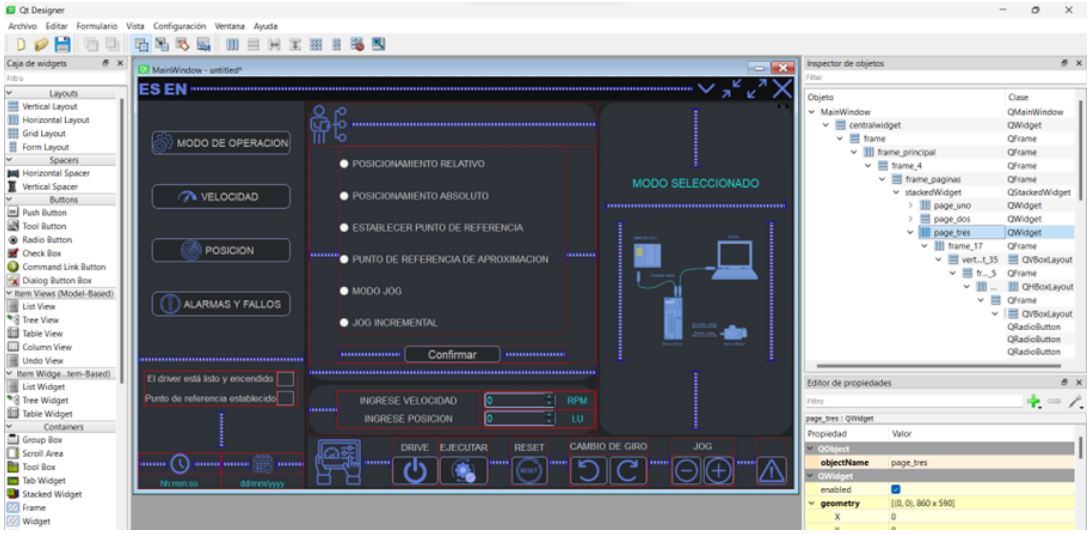


Figura 3.6: Página del modo de operación del drive.

La última página se utiliza para la visualización de las alarmas del driver. Esta

página consta de un QTextEdit que muestra las alarmas y fallos presentes en el driver en ese momento. Además, se incluyen dos indicadores que se encenderán cuando exista alguna alarma o fallo activo. Esta representación se muestra en la Figura. 3.7.

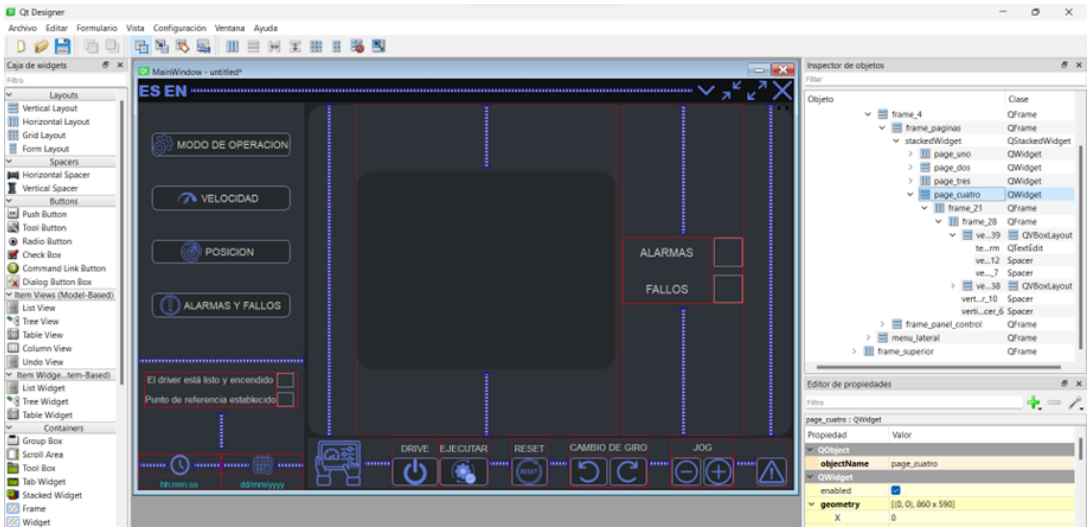


Figura 3.7: Página de alarmas y fallos.

Una vez finalizado el diseño de la interfaz, se procede a personalizarla según el estilo deseado y guardarla para poder utilizarla posteriormente en el código.

3.1.2. Programación y Vinculación del Servidor OPC-UA a Python

Una vez finalizado el diseño de la interfaz en Qt Designer, se procede a agregar las funcionalidades correspondientes utilizando un entorno de Python. Para ello, se importarán las librerías necesarias para el proyecto, como PyQt5 para la interfaz gráfica, pyqtgraph para las gráficas de velocidad y posición, y opcua para la vinculación del servidor a la interfaz, como se visualiza en el cuadro A0. Es importante asegurarse de tener instaladas todas las librerías en el entorno antes de comenzar.

A0: Importación de Librerías

```
import sys
import time
import collections
import pyqtgraph as pg

from PyQt5.QtWidgets import QApplication, QMainWindow
from PyQt5.QtCore import QPropertyAnimation, QEasingCurve, QTimer, QTime
from PyQt5.QtGui import QColor
from PyQt5 import QtCore, QtWidgets, QtGui
from PyQt5.uic import loadUi

from opcua import Client
from opcua import ua
```

Se crea la clase principal llamada 'VentanaPrincipal', la cual hereda de la clase 'QMainWindow', y se define el constructor 'init'. A continuación, se utiliza la función loadUi() para cargar el archivo realizado en Qt Designer llamado `'Interfaz_grafica.ui'` como se muestra en el cuadro A1

A1: Creación de la clase principal y carga del diseño de la interfaz

```
class VentanaPrincipal(QMainWindow):

def __init__(self):
    super(VentanaPrincipal,self).__init__()
    loadUi('Interfaz_grafica.ui', self)
```

A continuación, se presentan las funciones contenidas en el cuadro A2, las cuales permiten el control y la personalización de la barra de título de la ventana. En otras palabras, eliminan la barra de título predeterminada, habilitan el redimensionamiento de la ventana y permiten moverla al arrastrar un widget desde la barra de título personalizada. Además, con los botones correspondientes, es posible cerrar, minimizar, maximizar y restaurar la ventana.

A2: Control de la barra de titulo

```
    def control_bt_minimizar(self):
        self.showMinimized()

    def control_bt_restaurar(self):
        self.showNormal()
        self.bt_restaurar.hide()
        self.bt_maximizar.show()

    def control_bt_maximizar(self):
        self.showMaximized()
        self.bt_restaurar.show()
        self.bt_maximizar.hide()

    def resizeEvent(self, event):
        rect = self.rect()
        self.grip.move(rect.right() -
                       self.gripSize, rect.bottom() - self.gripSize)

    def mousePressEvent(self, event):
        self.clickPosition = event.globalPos()
```

Para ejecutar el programa y visualizar la interfaz gráfica, se utiliza el siguiente código, tal como se muestra en el cuadro A3. En estas líneas de código, además de iniciar el programa, se establece la conexión con el servidor OPC UA. Para ello, se crea una instancia del cliente OPC UA y se establece la conexión con la dirección del servidor. En este caso: `"opc.tcp://192.168.0.1:4841"`. Se utiliza 'client.connect()' para establecer la conexión y 'client.disconnect()' para desconectarse del servidor OPC UA al finalizar el programa. Luego, se ejecuta la aplicación con 'app.exec()'.

A3: Visualización de la interfaz y vinculación del servidor OPC al programa

```
if __name__ == "__main__":
client = Client("opc.tcp://192.168.0.1:4841")
try:
    client.connect()
    app = QApplication(sys.argv)
    mi_app = SplashScreen()
    mi_app.show()
    sys.exit(app.exec_())
finally:
    client.disconnect()
```

Una vez confirmada la conexión exitosa con el servidor OPC a través del programa, se procede a implementar las funcionalidades de los botones, tal como se muestra en A4. Esto implica que al hacer clic en un botón, se ejecuta el código contenido dentro del método asociado a dicho botón.

A4: Funcionalidades de los botones de la interfaz

```
self.bt_iniciar_drive.clicked.connect(self.control_bt_iniciar_drive)
self.bt_reset.clicked.connect(self.control_bt_reset)
self.bt_paro.clicked.connect(self.control_bt_paro)
self.bt_ejecute.clicked.connect(self.control_bt_ejecute)

self.bt_giro_der.clicked.connect(self.control_bt_giro_der)
self.bt_giro_izq.clicked.connect(self.control_bt_giro_izq)

self.bt_joc_1.pressed.connect(self.control_bt_joc_1_pressed)
self.bt_joc_1.released.connect(self.control_bt_joc_1_released)
self.bt_joc_2.pressed.connect(self.control_bt_joc_2_pressed)
self.bt_joc_2.released.connect(self.control_bt_joc_2_released)
```

El método asociado a cada botón tendrá una función distinta, dependiendo del botón que se accione. Sin embargo, todos los métodos seguirán la misma estructura,

como se visualiza en el cuadro A5. Dado que el botón estará vinculado a una función específica del driver, se necesita obtener el nodo correspondiente en el servidor, como se evidencia en la Figura 2.33, utilizando el método `'get_node'`.

Una vez obtenido el nodo, se puede obtener el valor actual de la variable utilizando `'get_value()'`. Para cambiar el valor de la variable, se puede utilizar una DataValue, especificando el nuevo valor y el tipo de variable deseado de la siguiente manera: `'ua.DataValue(ua.Variant(False, ua.VariantType.Boolean))'`. Luego, se utiliza `'set_data_value'` para actualizar el valor de la variable.

A5: Obtención del nodo para lectura y escritura de la variable asociada

```
def control_bt_iniciar_drive(self):
    nodo_iniciar= client.get_node('ns=3;s="Servo_HMI"."S120_ON"')
    valor_drive = nodo_iniciar.get_value()

    if valor_drive == True:
        opcion_iniciar = ua.DataValue(ua.Variant
                                      (False, ua.VariantType.Boolean))
        self.valor_drive1 = False
    else:
        opcion_iniciar = ua.DataValue(ua.Variant
                                      (True, ua.VariantType.Boolean))
        self.valor_drive1 = True

    nodo_iniciar.set_data_value(opcion_iniciar)

    self.animate_button_iniciar_drive()

def control_bt_ejecute(self): ...

def control_bt_paro(self):  ...
```

En este proceso de obtener las variables del servidor y modificarlas, se sigue el mismo enfoque para las demás funcionalidades del driver. La única diferencia radica

en la obtención del nodo correspondiente, ya que cada variable tiene su propio nodo asociado y su respectivo tipo de dato como se ilustra en el cuadro A6.

A6: Diferentes nodos de las variables del servidor

```
        def control_bt_reset(self):
        nodo_modo= client.get_node('ns=3;s="Servo_HMI"."MODE_SERVO"')
        opcion_modo = ua.DataValue(ua.Variant(0, ua.VariantType.Int16))
        nodo_modo.set_data_value(opcion_modo)

        nodo_vel = client.get_node('ns=3;s="Servo_HMI"."Velocidad"')
        w_vel = ua.DataValue(ua.Variant(0, ua.VariantType.Int32))
        nodo_vel.set_data_value(w_vel)

        nodo_pos = client.get_node('ns=3;s="Servo_HMI"."Posicion"')
        w_pos = ua.DataValue(ua.Variant(0, ua.VariantType.Int32))
        nodo_pos.set_data_value(w_pos)

        nodo_iniciar= client.get_node('ns=3;s="Servo_HMI"."S120_ON"')
        opcion_iniciar = ua.DataValue(ua.Variant
                                      (False, ua.VariantType.Boolean))
        nodo_iniciar.set_data_value(opcion_iniciar)

        nodo_ejecute= client.get_node('ns=3;s="Servo_HMI"."Execute"')
        nodo_ejecute.set_data_value(opcion_iniciar)

        nodo_giro_izq= client.get_node('ns=3;s="Servo_HMI"."Negativo"')
        nodo_giro_izq.set_data_value(opcion_iniciar)

        nodo_giro_der= client.get_node('ns=3;s="Servo_HMI"."Positivo"')
        nodo_giro_der.set_data_value(opcion_iniciar)
```

Por lo tanto, el procedimiento para obtener las variables del servidor y cambiarlas se realizará de manera similar para cada funcionalidad. Se obtendrá el

nodo específico de cada variable y se utilizará el tipo de dato correspondiente al realizar la lectura o escritura de valores.

De esta manera, se podrá acceder a las variables del servidor y actualizarlas según sea necesario, manteniendo la misma estructura básica, pero adaptándola a las particularidades de cada nodo y tipo de dato.

3.2. Presentación de datos

Una vez concluida la programación de las funcionalidades de la interfaz vinculadas al servidor OPC-UA, se procederá a poner a prueba el funcionamiento del sistema y la respectiva presentación de los resultados. Al iniciar el programa, se mostrará la siguiente ventana:

Figura 3.8: Ventana principal de la interfaz.

Se comienza con la explicación de los modos de operación JOG y JOG Incremental. En el modo JOG, al pulsar los botones JOG, el servo motor gira 30° y, si se mantiene el botón presionado, continúa girando en la misma dirección. En cambio, en el modo JOG Incremental, al pulsar cualquiera de los botones, se genera un pulso que hace girar el servo motor exactamente 30° en la dirección correspondiente. La selección del modo JOG se puede apreciar en la Figura.3.9.

Es importante destacar que, para utilizar cualquiera de los modos, es necesario que el botón 'drive' esté activado y que su indicador también esté encendido.

Figura 3.9: Modo de operación JOG.

El siguiente modo de operación es el de Establecer el punto de referencia. En este caso, es necesario activar el botón 'drive' y luego simplemente presionar el botón 'ejecutar'. Esto establecerá el punto de referencia en la posición actual del servomotor. Una vez realizado esto, el indicador correspondiente se encenderá para indicar que el punto de referencia se ha establecido, tal como se muestra en la Figura3.10.

Figura 3.10: Modo de operación: Establecer Punto de Referencia.

Pasamos ahora al siguiente modo de operación, el de Posicionamiento Relativo. Para este modo, es necesario ingresar el valor de velocidad del servomotor y el valor de posición al cual se desea llegar, tal como se observa en la Figura 3.11.

Figura 3.11: Modo de operación Posicionamiento Relativo.

Una vez que se presiona el botón 'ejecutar', el servomotor se moverá a la posición indicada. Luego, al presionar el botón 'posición', se cambiará a la página donde se puede visualizar tanto la posición actual del servomotor como su gráfica

respectiva. Esto se muestra en la Figura 3.12.

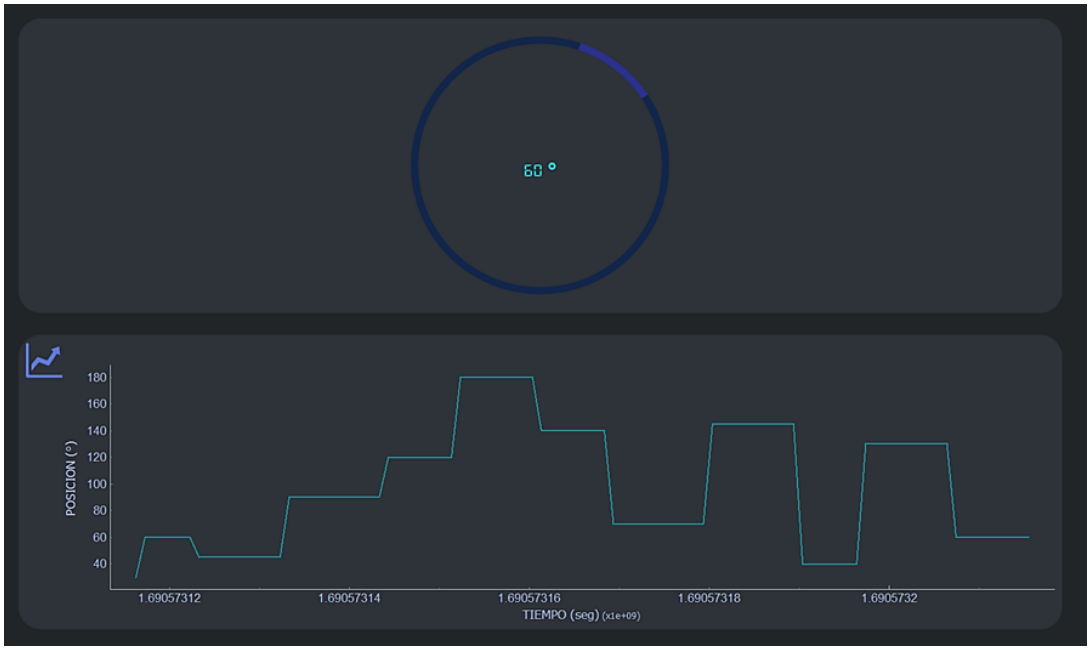


Figura 3.12: Visualización de la Posición del Servomotor.

En el modo de operación Posicionamiento Absoluto, el proceso se realiza de manera similar al modo de Posicionamiento Relativo. Esto se puede apreciar en las Figuras 3.13 y 3.14, donde se muestra el procedimiento correspondiente a ambos modos.

Figura 3.13: Modo de operación Posicionamiento Absoluto.

Figura 3.14: Visualización de la Posición del Servomotor.

En el modo de operación Punto de referencia aproximado, se debe activar el sentido de giro deseado, ya sea hacia la izquierda o hacia la derecha. Esto se muestra claramente en la Figura 3.15.

Figura 3.15: Modo de operación Punto de referencia Aproximado.

Al accionar el botón 'ejecutar', el servomotor comienza a girar a una velocidad preestablecida en 'STARTED'. Esto se muestra en la Figura 3.16. El servomotor seguirá

girando hasta alcanzar una posición predefinida, también en 'STARTED'.

Figura 3.16: Visualización de la Velocidad del Servomotor.

Para cambiar el idioma de la interfaz de español a inglés y viceversa, simplemente se debe accionar el botón 'EN', el cual cambiará el idioma a inglés, tal como se muestra en la Figura 3.17. Por otro lado, al accionar el botón 'ES', el idioma volverá a ser español. De esta manera, se puede alternar entre los idiomas de manera sencilla a través de los botones correspondientes.

Figura 3.17: Cambio de Idioma de la interfaz.

3.3. Alarmas y Reportes

El Driver Sinamics S120 CU310-2PN incluye múltiples alarmas y fallos que sirven como indicadores para ayudar a los usuarios a identificar y diagnosticar problemas en el sistema de accionamiento. Estas alarmas y fallos se generan cuando se presentan situaciones anormales o fallas en el funcionamiento del equipo. Estas notificaciones proporcionan información valiosa para el usuario, permitiéndoles tomar medidas correctivas adecuadas y mantener el sistema en óptimas condiciones de operación.

En la Tabla 3.2 se muestran algunas de las alarmas y fallos que posee el driver.

Dentro de la interfaz, se ha dedicado una sección exclusiva para mostrar las alarmas y fallos que podrían ocurrir durante el funcionamiento del driver. En caso de que se presente una alarma, el botón de 'Alarmas y Fallos' cambiará de color para indicar que existe una situación de alarma en el driver. Del mismo modo, si se produce un fallo en el driver, el botón de 'Emergencia' cambiará de color para indicar la presencia de un fallo. Esto se puede apreciar claramente en la Figura 3.18. Estos cambios de color proporcionan una retroalimentación visual inmediata al usuario, permitiéndole identificar rápidamente la presencia de alarmas y fallos en el sistema

Tabla 3.2: Tabla de Alarmas y Fallos del Driver. [20]

Número de Alarma	Descripción
A01500	Fallo de torque en el motor
A01600	Parámetros no válidos del controlador
A01700	Error de medición del par motor
A01800	Error de sobrecarga en el convertidor
A03100	Sobrecarga mecánica en el motor
A03200	Sobrecorriente en el motor
A04200	Sobrecarga del enlace interno del convertidor
A07200	Sobrecarga en la salida del convertidor
A07461	Fallo de habilitación del punto de referencia
A08200	Sobrecarga del módulo de frenado interno
A08300	Fallo en el módulo de frenado interno
A09100	Sobretensión del bus DC
A09101	Sobretensión del bus de CC
A09200	Subtensión del bus DC
A30016	Tensión de carga desconectada
F01000	Fallo en la comunicación con la tarjeta de control de potencia
F01001	Fallo en la comunicación con la tarjeta de control de motor
F01002	Fallo en la comunicación con la tarjeta de feedback
F01200	Sobrecalentamiento del módulo de potencia
F01300	Fallo en la tarjeta de entrada/salida
F01400	Sobrecorriente en el motor
F01600	Parametrización incorrecta del encoder
F02000	Sobrecalentamiento de la tarjeta de control de motor
F02100	Sobretensión de la fuente de alimentación DC intermedia
F02200	Subtensión de la fuente de alimentación DC intermedia
F02340	Error de comunicación PROFINET
F03002	Error de sobrevelocidad del controlador
F06100	Error en la comunicación del bus de controlador
F07490	Habilitacion anulada durante el desplazamiento
F07600	Error en la detección de la temperatura de refrigeración

de accionamiento.

Figura 3.18: Visualización de indicadores de alarma y fallo.

Una vez que se haya generado una alarma, un fallo, o ambos simultáneamente, se indicará en la sección de alarmas mediante texto que mostrará el número de alarma y el número de fallo, así como la causa o motivo de la misma. Además, los indicadores correspondientes a las alarmas se encenderán para informar visualmente si existe alguna alarma o fallo en el sistema. Esto se puede apreciar claramente en la Figura 3.19. De esta manera, la interfaz proporciona una forma clara y concisa de identificar y comprender las alarmas y fallos presentes en el driver.

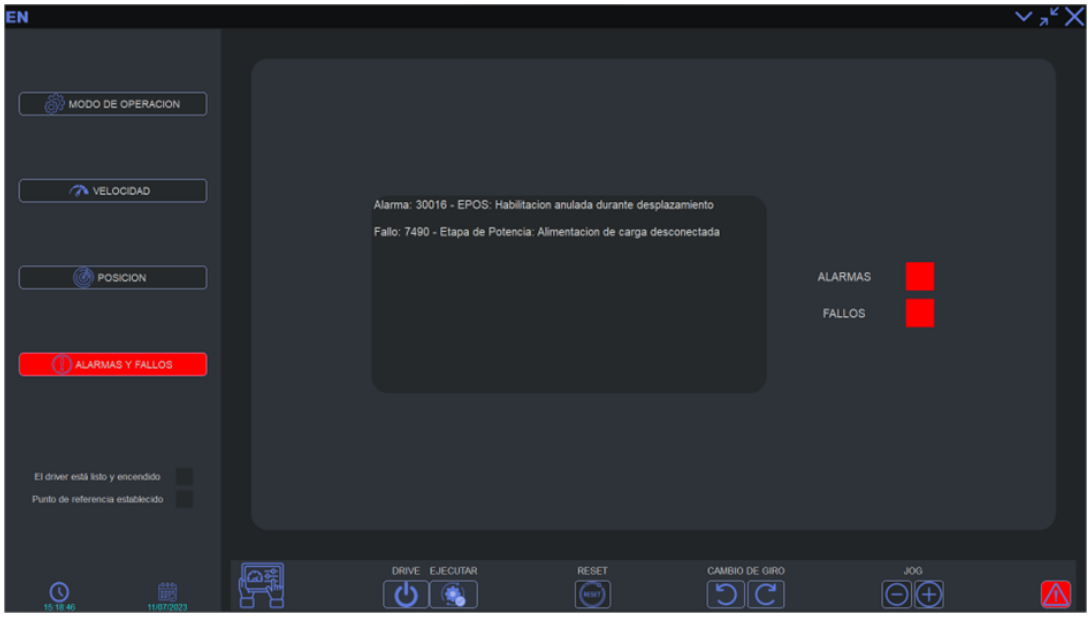


Figura 3.19: Visualización de las alarmas y fallos.

En caso de que se produzca un fallo en el sistema, la interfaz ofrece la posibilidad de corregirlo directamente desde ella al presionar el botón de emergencia. Una vez que se presiona el botón, transcurrirá algunos segundos y los fallos presentes en el driver se corregirán automáticamente. En la interfaz, los indicadores visuales volverán a su color original y los textos indicativos desaparecerán, lo que indicará que los fallos en el driver se han corregido satisfactoriamente. Esta funcionalidad permite una rápida respuesta y solución a los fallos, mejorando la eficiencia y el rendimiento del sistema de accionamiento.

Capítulo 4

Conclusiones, Recomendaciones y Trabajos Futuros

4.1. Conclusiones

El presente proyecto se enfocó en la implementación de una interfaz basada en OPC UA y Python para el monitoreo y operación de un sistema industrial de servo posicionamiento, logrando alcanzar un control preciso del sistema y una comunicación eficiente.

A través de un análisis detallado del sistema de servo posicionamiento, se adquirió un conocimiento profundo sobre su funcionamiento y configuraciones, en especial del accionamiento SINAMICS S120. Este análisis resultó esencial para sentar las bases del correcto control del servomotor.

La implementación del control del servomotor mediante el accionamiento CU310-2 pn, junto con la utilización del bloque de función SINA POS en la programación del PLC mediante el telegrama 111, demostraron ser enfoques efectivos para lograr un control preciso y una supervisión confiable del servomotor. Además, el uso de la interfaz de prueba en WinCC facilitó las primeras pruebas realizadas del sistema.

Así mismo, la configuración e implementación de la comunicación mediante el estándar OPC UA proporcionó una interacción efectiva y segura entre la estación de cómputo personal y el PLC, demostrando su eficiencia en el intercambio de datos

y permitiendo una integración fluida entre los componentes del sistema de servo posicionamiento.

El diseño y desarrollo del sistema HMI en Python resultó en una interfaz gráfica intuitiva y funcional. La vinculación del servidor OPC UA con Python posibilitó el ajuste sencillo de los parámetros del sistema y la presentación de datos en tiempo real de manera efectiva, lo que facilita la toma de decisiones oportuna por parte de los operadores.

En conclusión, la implementación de la interfaz basada en OPC UA y Python ha sido exitosa y ha cumplido con los objetivos propuestos. La solución desarrollada ha permitido un control preciso y una supervisión eficiente del sistema de servo posicionamiento industrial, mejorando significativamente su rendimiento y proporcionando a los operadores una herramienta efectiva para la toma de decisiones.

4.2. Recomendaciones

En base al desarrollo y resultados obtenidos en este proyecto, se formulan las siguientes recomendaciones para mejorar y optimizar el sistema de servo posicionamiento y su interfaz:

Se recomienda mantener actualizado el driver SINAMICS S120 con las últimas versiones de firmware proporcionadas por los fabricantes, garantizando así el correcto funcionamiento del driver a largo plazo.

Es aconsejable implementar redundancia en la comunicación para mejorar la fiabilidad del sistema y asegurar la continuidad de la operación en caso de fallos en la red.

Se sugiere incorporar medidas de seguridad en el servidor OPC UA, asignando un usuario para acceder al mismo, con el objetivo de proteger el sistema de posibles amenazas externas.

Se debe llevar a cabo evaluaciones periódicas del rendimiento del sistema, analizando indicadores clave como la precisión del control, la velocidad de respuesta y la estabilidad.

Se recomienda recopilar y analizar datos históricos del sistema que permitan

identificar patrones, tendencias y posibles problemas recurrentes.

Es importante mantenerse informado sobre los avances tecnológicos en el ámbito de los sistemas de servo posicionamiento para evaluar la posibilidad de implementar nuevas tecnologías que puedan mejorar la eficacia del sistema.

Siguiendo estas recomendaciones, se fortalecerá la eficiencia y la confiabilidad del sistema de servo posicionamiento, asegurando un óptimo desempeño y una mejor experiencia para los operadores.

Es importante destacar que, si bien esta tesis ha alcanzado los objetivos planteados, siempre hay espacio para la mejora continua y la exploración de nuevas funcionalidades. Se sugiere que futuras investigaciones se centren en la optimización del rendimiento del sistema y en la incorporación de funcionalidades avanzadas para maximizar su eficiencia y aplicabilidad en diversos entornos industriales.

4.3. Trabajos Futuros

Es importante destacar que, si bien esta tesis ha alcanzado los objetivos planteados, siempre hay espacio para la mejora continua y la exploración de nuevas funcionalidades. Se sugiere que futuras investigaciones se centren en:

1. La incorporación de funcionalidades avanzadas para maximizar la eficiencia y aplicabilidad del sistema en diversos entornos industriales.
2. La implementación de técnicas de seguridad funcional para garantizar la integridad y protección del sistema ante posibles riesgos.
3. El uso de tecnologías de comunicación inalámbrica, como Wi-Fi o Bluetooth, para permitir una mayor flexibilidad y movilidad en el control del sistema.
4. La expansión a sistemas multiestación que involucren más de un driver, lo que podría aumentar la capacidad y la versatilidad del sistema.

Al enfocarse en estas áreas de investigación, se podrá potenciar aún más el sistema de servo posicionamiento, permitiendo su adaptación a las demandas cambiantes de la industria y mejorando su desempeño en diferentes contextos industriales.

Glosario

GSD General Station Description.

HMI Human-Machine Interface.

OPC Open Platform Communications.

PLC Programmable Logic Controller.

PROFINET Process Field Network.

PyQt5 Python bindings for Qt 5.

SCADA Supervisory Control and Data Acquisition.

WinCC Windows Control Center.

Referencias

[1] L. I. Minchala, J. Peralta, P. Mata-Quevedo y J. Rojas, «An approach to industrial automation based on low-cost embedded platforms and open software,» *Applied Sciences (Switzerland)*, vol. 10, 14 jul. de 2020, ISSN: 20763417. DOI: 10.3390/APP10144696.

[2] J. S. Rocha-Doria, J. G. Fuentes-Velázquez y C. Angeles-Camacho, «Synchrophasor applications in distribution systems: real-life experience,» *Monitoring and Control of Electrical Power Systems Using Machine Learning Techniques*, págs. 107-136, ene. de 2023. DOI: 10.1016/B978-0-32-399904-5.00011-9.

[3] *1FK7 for SINAMICS S120. Siemens. Siemens, . . .* dirección: http://pzip.ru/catalog/siemens/?catid=22501.

[4] «1FK7 servomotors for SINAMICS S120,»

[5] «1FK7 servomotors for SINAMICS S120,»

[6] *25009692, CONECTOR CIRCULAR HEMBRA EPIC POWER LS1 KIT D6, 5 PINES +PE, P/ SERVOMOTORES – Redcoind | Tecnologia Y Control En Tus Manos*. dirección: https://redcoind.pe/producto/25009692-conector-circular-hembra-epic-power-ls1-kit-d6-5-pines-pe-p-servomotores/.

[7] C. E. N. Brito y D. por, «Implementación de un laboratorio de "Motion Control": motor síncrono 1FT7 con accionamiento SINAMICS S110,» 2017.

[8] *Sinamics S120 Motion Control Using S7–1500 and TIA Portal | by Karthik Muthineni | Medium*. dirección: https://karthikm618.medium.com/sinamics-s120-motion-control-using-s7-1500-and-tia-portal-734522f4b9f2.

[9] *SIEMENS SINAMICS S120 converter Power Module PM340 6SL3210-1SE11-3UA0 0.37kW 1.3A - AoteWell Automation | AoteWell Ltd*. dirección: https://www.industry-mall.net/siemens/siemens-sinamics-s120-converter-power-module-pm340-6sl3210-1se11-3ua0-0-37kw-1-3a.

[10] *Control Units CU310-2 para accionamientos monoeje - Rieder Tech Solutions*. dirección: https://techsolutions.rieder.com.py/product/control-units-cu310-2-para-accionamientos-monoeje-2/.

[11] *Arquitectura de un sistema de control típico | Download Scientific Diagram*. dirección: https://www.researchgate.net/figure/Figura-1-Arquitectura-de-un-sistema-de-control-tipico_fig1_317421942.

[12] $infoPLC_net_MSE6 - C2M - 5000 - FB33_FB34_FB35integrationwithS7 - 1500PLCusingTIAPortal.pdf - GoogleDrive$. dirección: https://drive.google.com/file/d/1LgsTgTC5bLIF7m9KyX-5c_TIBuB8aCCv/view.

[13] «SINAMICS G: Axis positioning with the $SINA_POSblock$,» dirección: http://www.siemens.com/industrialsecurity..

[14] «Posicionamento Básico (EPos) no SINAMICS V90 PN,» dirección: https://support.industry.siemens.com/cs/br/en/view/109779364.

[15] *Comunicación Profinet; Telegramas Admitidos - Siemens SINAMICS V90 Instrucciones De Servicio [Página 75] | ManualsLib*. dirección: https://www.manualslib.es/manual/145976/Siemens-Sinamics-V90.html?page=75#manual.

[16] O. Duymazlar y D. ENGİN, «Design, Application and Analysis of an OPC-based SCADA System,» *Journal of Polytechnic*, mar. de 2022, ISSN: 1302-0900. DOI: 10.2339/POLITEKNIK.1029629.

[17] J. M. Gutierrez-Guerrero y J. A. Holgado-Terriza, «Automatic configuration of OPC UA for industrial internet of things environments,» *Electronics (Switzerland)*, vol. 8, 6 jun. de 2019, ISSN: 20799292. DOI: 10.3390/ELECTRONICS8060600.

[18] *PyQT5 interfáces gráficas con Python - Cursos de Programación de 0 a Experto © Garantizados*. dirección: https://unipython.com/pyqt5-interfaces-graficas-con-python/.

[19] *Qt Designer Manual*. dirección: https://doc.qt.io/qt-5/qtdesigner-manual.html.

[20] L. Manual, «SINAMICS SINAMICS S120/S150,» 2014.