Escuela Superior Politécnica del Litoral

Facultad de Ingeniería en Electricidad y Computación

Mejora de Diseño para Horno Industrial con Control por PLC para Secado de Motores

INGE-2834

Proyecto Integrador

Previo la obtención del Título de:

Ingeniería en electricidad

Presentado por:

Villamar Calderón Diego Antonio
Córdova Sánchez Delia Fiorella

Guayaquil - Ecuador
Año: 2025

Dedicatoria

Este trabajo está dedicado a mi familia, que siempre me han apoyado y motivado a ser mejor persona.

Agradecimientos

Agradezco a mi familia, a mis amigos y a Dios por ser los pilares en mi vida y fuente de mi mayor motivación e inspiración.

Agradezco a mi abuelita Mercy, por todo su amor y apoyo que me ha brindado, por ser una madre para mí.

Agradezco a mi tío Andrés por haberme enseñado tanto no solo en conocimiento académico sino también a crecer como persona.

Declaración Expresa

Nosotros Delia Fiorella Córdova Sánchez y Diego Antonio Villamar Calderón acordamos y reconocemos que:

La titularidad de los derechos patrimoniales de autor (derechos de autor) del proyecto de graduación corresponderá al autor o autores, sin perjuicio de lo cual la ESPOL recibe en este acto una licencia gratuita de plazo indefinido para el uso no comercial y comercial de la obra con facultad de sublicenciar, incluyendo la autorización para su divulgación, así como para la creación y uso de obras derivadas. En el caso de usos comerciales se respetará el porcentaje de participación en beneficios que corresponda a favor del autor o autores.

La titularidad total y exclusiva sobre los derechos patrimoniales de patente de invención, modelo de utilidad, diseño industrial, secreto industrial, software o información no divulgada que corresponda o pueda corresponder respecto de cualquier investigación, desarrollo tecnológico o invención realizada por mí/nosotros durante el desarrollo del proyecto de graduación, pertenecerán de forma total, exclusiva e indivisible a la ESPOL, sin perjuicio del porcentaje que me/nos corresponda de los beneficios económicos que la ESPOL reciba por la explotación de mi/nuestra innovación, de ser el caso.

En los casos donde la Oficina de Transferencia de Resultados de Investigación (OTRI) de la ESPOL comunique los autores que existe una innovación potencialmente patentable sobre los resultados del proyecto de graduación, no se realizará publicación o divulgación alguna, sin la autorización expresa y previa de la ESPOL.

Guayaquil, 12 de septiembre de 2025.

Nombre: CORDOVA SANCHEZ DELIA FIORELLA
Emitido por: UANATACA CA2 2016

Delia Córdova

Firmado electrónicamente por:
DIEGO ANTONIO VILLAMAR CALDERON
Validar únicamente con FirmaEC

Diego Villamar

Evaluadores

Firmado electrónicamente por:
SIXIFO DANIEL
FALCONES ZAMBRANO
Validar únicamente con FirmaEC

Ing. Sixifo Falcones

Profesor de Materia

Firmado electrónicamente por:
DOUGLAS MAURICIO
AGUIRRE HERNANDEZ
Validar únicamente con FirmaEC

Ing. Douglas Aguirre

Tutor de proyecto

Resumen

El presente proyecto tiene como objetivo diseñar y validar mediante simulación un sistema de automatización para optimizar el control operativo de un horno industrial de convección, utilizado en el secado de motores en la empresa Electro Industrial Micabal S.A.. La solución busca mejorar la eficiencia energética, la seguridad y la supervisión remota del proceso, integrando sensores, electroválvulas, mecanismos de seguridad y una interfaz Hombre-Máquina (HMI). La arquitectura se diseñó de manera modular, garantizando su escalabilidad hacia otros hornos de la planta.

Para la implementación, se utilizó TIA Portal con un PLC Siemens S7-1200 simulado en PLCSIM, organizando las variables mediante UDT y bloques de datos. La comunicación externa se estableció a través de NetToPLCSIM, permitiendo la integración con Node-RED, donde se desarrollaron flujos y dashboards en modos automático y manual. Con el fin de asegurar conectividad remota segura, se implementó Tailscale como red privada virtual. Finalmente, se desarrolló un gemelo digital en Minecraft mediante el mod CC:Tweaked y scripts en Lua, que gestionan la comunicación por WebSocket con Node-RED.

Los resultados muestran mejoras en el control de parámetros críticos, validando escenarios de histéresis, activación de actuadores, paro de emergencia y detección de fallas, con potencial de replicabilidad en entornos industriales similares.

***Palabras clave**: Automatización industrial, PLC simulado, Node-RED, Gemelo digital, WebSocket*

Abstract

This project aims to design and validate, through simulation, an automation system to optimize the operational control of an industrial convection oven used for motor drying at Electro Industrial Micabal S.A. The solution seeks to improve energy efficiency, safety, and remote supervision, integrating sensors, solenoid valves, safety mechanisms, and a Human-Machine Interface (HMI). The architecture was designed in a modular way, ensuring scalability to other ovens in the plant.

The implementation was carried out in TIA Portal with a Siemens S7-1200 PLC simulated in PLCSIM, structuring variables through UDTs and data blocks. External communication was enabled via NetToPLCSIM, allowing integration with Node-RED, where flows and dashboards were developed for automatic and manual operation. To ensure secure remote connectivity, Tailscale was used as a private virtual network.

Finally, a digital twin in Minecraft was developed using the CC:Tweaked mod and Lua scripts, which handle communication with Node-RED via WebSocket. The results demonstrate improvements in the control of critical parameters, validating hysteresis operation, actuator activation, emergency stop, and fault detection, with potential replicability in similar industrial environments.

Keywords: *Industrial automation, Simulated PLC, Node-RED, Digital twin, WebSocket*

Índice general

Capítulo 1

1.1 Introducción

La automatización industrial constituye en la actualidad un eje estratégico para mejorar la competitividad, la seguridad y la eficiencia energética en los procesos productivos. En particular, empresas dedicadas al mantenimiento y reacondicionamiento de equipos eléctricos, como Electro Industrial Micabal S.A., enfrentan desafíos significativos en la etapa de secado de motores eléctricos. Esta fase, realizada en hornos industriales de convección, resulta crítica para garantizar la calidad del aislamiento y el correcto funcionamiento posterior de los equipos. Sin embargo, al ser ejecutada de manera manual y sin un sistema centralizado de supervisión, se generan limitaciones en la trazabilidad, la ergonomía del trabajo y la capacidad de respuesta ante fallos.

En este contexto, se plantea el diseño de un sistema de automatización que integre tecnologías de control basadas en PLC Siemens S7-1200, interfaces de supervisión (HMI) y mecanismos de seguridad, con el fin de optimizar la operación del horno, reducir riesgos y permitir un monitoreo en tiempo real. No obstante, debido a restricciones de recursos y a la naturaleza académica del presente trabajo, la solución se desarrolla como un prototipo simulado, validado en un entorno virtual que permite reproducir fielmente las condiciones de operación del proceso sin necesidad de hardware físico.

Para ello, se utiliza TIA Portal como plataforma de programación, junto con PLCSIM para emular el controlador, y NetToPLCSIM para asignar una dirección IP accesible al PLC virtual. La comunicación con Node-RED posibilita la creación de dashboards en modos automático y manual, mientras que el uso de Tailscale garantiza una conectividad remota segura entre dispositivos. Finalmente, se implementa un gemelo digital en Minecraft,

mediante el mod CC:Tweaked y scripts en Lua, que permite visualizar en un entorno tridimensional el comportamiento de actuadores y sensores, logrando una representación didáctica e interactiva del sistema.

De esta manera, el proyecto se enmarca como un prototipo académico de validación virtual, que no pretende reemplazar directamente los sistemas físicos de la empresa, pero que aporta una base sólida para futuras implementaciones. Asimismo, constituye una contribución al campo de la Industria 4.0, al integrar controladores virtuales, conectividad en red y gemelos digitales como herramientas de formación y experimentación académica, con potencial de replicabilidad en entornos industriales reales.

1.2 Descripción del Problema

Electro Industrial Micabal S.A. es una empresa ecuatoriana dedicada al mantenimiento, reparación y reacondicionamiento de motores eléctricos industriales. Uno de sus procesos clave es el secado de motores en hornos de convección a gas, etapa crítica posterior al bobinado que garantiza la eliminación de humedad y preserva el aislamiento del equipo. En la actualidad, este proceso se realiza de forma manual y sin un sistema centralizado de supervisión, lo cual genera limitaciones técnicas y riesgos operativos.

El problema principal radica en la ausencia de un sistema automatizado que permita controlar y monitorear en tiempo real las variables físicas del proceso (temperatura interna, tiempo de operación, presión en tuberías) y los estados de operación (apertura de puerta, modo de trabajo manual o automático, estado del proceso de histéresis). Esta carencia impide realizar ajustes precisos, aumenta la dependencia del operador y expone al personal a riesgos en pruebas reales, además de dificultar la supervisión remota.

La situación se enmarca en los desafíos comunes de la automatización industrial, donde destacan:

- Costos elevados en la adquisición de hardware y simuladores industriales.

- Riesgos en pruebas en planta real, que comprometen la seguridad operativa.
- Limitaciones en capacitación, al no contar con herramientas virtuales accesibles para entrenar al personal.

Frente a estos retos, los métodos tradicionales de prueba suelen ser extensos, costosos y rígidos, mientras que un prototipo simulado ofrece rapidez, menor costo y mayor flexibilidad. Bajo estas condiciones, se plantea el desarrollo de un sistema de automatización validado en un entorno simulado académico, que permita evaluar la factibilidad técnica de la propuesta, optimizar el control del proceso y servir como base para futuras implementaciones reales en la empresa o en entornos industriales similares.

Entre las restricciones más relevantes se encuentran:

- La disponibilidad limitada de presupuesto para la adquisición de hardware.
- La necesidad de cumplir con normativas de seguridad industrial vigentes.
- La compatibilidad con las plataformas ya utilizadas en la empresa, como el PLC Siemens S7-1200 y el entorno de desarrollo TIA Portal.

Por otra parte, los requerimientos técnicos y académicos del proyecto contemplan:

- Diseñar un sistema de control que pueda operar en modo manual y automático, aplicando un control por histéresis.
- Simular la comunicación entre el PLC virtual y plataformas externas mediante NetToPLCSIM, Node-RED y WebSocket.
- Desarrollar una interfaz de usuario (HMI) accesible local y remotamente, con visualización de las principales variables de proceso.
- Representar el comportamiento del horno en un entorno tridimensional interactivo, a través de un gemelo digital en Minecraft.

La problemática descrita es observable, medible y susceptible de análisis técnico, lo que la convierte en un escenario adecuado para el desarrollo de un prototipo simulado que

integre automatización industrial, conectividad en red y gemelos digitales, como aporte académico y con potencial de aplicación futura en la industria.

1.3 Justificación del Problema

La automatización del proceso de secado de motores en hornos industriales de convección resulta fundamental para mejorar la eficiencia, la seguridad y la calidad del servicio que ofrece Electro Industrial Micabal S.A. En su estado actual, la operación manual del horno limita la precisión en el control de variables críticas como la temperatura, el tiempo y la presión, lo que aumenta la probabilidad de errores humanos y reduce la confiabilidad del proceso. Resolver esta problemática es importante porque incide directamente en la vida útil de los motores eléctricos, en la seguridad del personal que opera los equipos y en la capacidad de la empresa para ofrecer un servicio competitivo en el mercado.

Desde la perspectiva académica, el proyecto aborda los desafíos típicos de la automatización industrial, tales como el costo elevado de hardware especializado, los riesgos de realizar pruebas en planta real y las limitaciones en la capacitación del personal. Ante estas barreras, la construcción de un prototipo simulado se convierte en una alternativa viable y económica para validar la arquitectura de control, capacitar operadores y probar diferentes escenarios sin exponer recursos físicos.

La solución propuesta integra herramientas modernas como PLC simulado, Node-RED, Tailscale y un gemelo digital en Minecraft, demostrando que es posible combinar plataformas de software, comunicación segura y entornos 3D interactivos en proyectos de Industria 4.0. Su importancia radica no solo en solventar las necesidades técnicas de la empresa de referencia, sino también en generar un modelo replicable en otros procesos industriales que enfrenten retos similares.

1.4 Objetivos

1.4.1 Objetivo general

- Diseñar, implementar y validar mediante simulación un sistema de automatización para un horno industrial, integrando un PLC virtual, una interfaz de control en Node-RED y un gemelo digital en Minecraft, empleando WebSocket como protocolo de comunicación, con el fin del análisis del desempeño y la optimización de la interacción hombre-máquina en un entorno de prueba.

1.4.2 Objetivos específicos

- Diseñar la lógica de control del horno en TIA Portal, estructurando las variables mediante tipos de datos definidos por el usuario (UDT) y bloques de datos (DB).
- Simular el comportamiento del PLC Siemens S7-1200 mediante PLCSIM, asegurando la ejecución de la programación y la validación de las rutinas de control.
- Configurar la comunicación entre el PLC virtual y plataformas externas a través de NetToPLCSIM, garantizando la asignación de una dirección IP accesible bajo el protocolo ISO-on-TCP.
- Implementar en Node-RED flujos de datos y dashboards de operación en modos automático y manual, que permitan la interacción con las principales variables del proceso.
- Desarrollar un gemelo digital en Minecraft con el mod CC:Tweaked y scripts en Lua, para representar gráficamente actuadores, sensores y estados de operación mediante comunicación por WebSocket.
- Validar el sistema a través de escenarios de prueba que incluyan control por histéresis, activación de actuadores, cambio de modos de operación, paro de emergencia y detección de fallas.

1.5 Marco teórico

1.5.1 Controladores Lógicos Programables (PLC)

1.5.1.1 Tipos de PLC y características del Siemens S7-1200

El Controlador Lógico Programable (PLC) es un dispositivo electrónico diseñado para ejecutar operaciones de control secuencial en tiempo real dentro de sistemas industriales. Su arquitectura incluye una Unidad Central de Procesamiento (CPU), módulos de memoria de trabajo y programa (RAM, EEPROM), interfaces de entradas/salidas (E/S), y un bus de datos interno para la comunicación entre componentes. Las señales de entrada, provenientes de sensores digitales o analógicos, son procesadas por la CPU de acuerdo con la lógica almacenada en la memoria, generando respuestas que activan actuadores como electroválvulas, motores o indicadores [1].

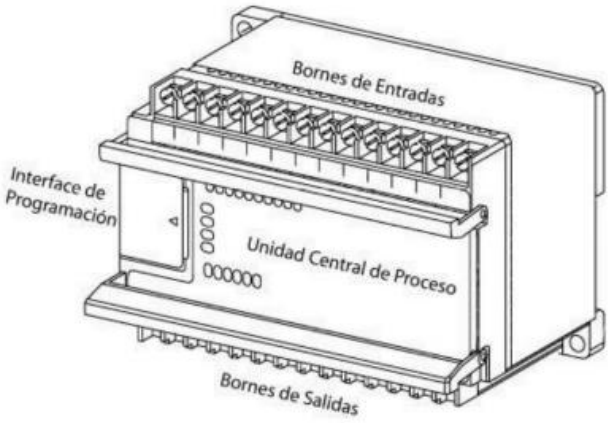


Ilustración 1. Arquitectura general de los PLCs.

Los PLC pueden clasificarse por su tamaño (nano, compacto, modular). El S7-1200 de Siemens es un PLC modular compacto que destaca por su escalabilidad, capacidad de comunicación (PROFINET, OPC UA, TCP/IP), y compatibilidad con TIA Portal [2].

1.5.1.2 PLC Siemens S7-1200

Como parte del sistema de automatización del horno de secado, se ha seleccionado el PLC Siemens S7-1200 por su compatibilidad con arquitecturas modulares, su capacidad para

manejar señales analógicas y digitales, y su facilidad de integración con sistemas HMI. Ofrece un entorno robusto para aplicaciones industriales críticas, especialmente aquellas que requieren control preciso de variables como temperatura, tiempo y estado del proceso. Además, permite la programación y monitoreo a través del entorno TIA Portal, y su arquitectura soporta comunicación Ethernet, facilitando la expansión futura del sistema.

Ilustración 2. PLC Siemens S7-1200 modelo
CPU 1214C AC/DC relé

En el presente proyecto se emplea el PLC Siemens S7-1200, modelo CPU 1214C AC/DC/Relé. Esta unidad central es ideal para tareas de control secuencial y de procesos en aplicaciones de tamaño pequeño a mediano, como el horno industrial automatizado del presente estudio.

Especificaciones técnicas clave que se indican en el manual de usuario del equipo:

- Alimentación: 120/230 V AC.
- Entradas digitales (DI): 14 canales (24 V DC).
- Salidas digitales (DO): 10 canales de tipo relé, lo que permite manejar cargas AC y DC.
- Entradas analógicas: 2 canales integrados (0–10 V).

- Memoria de programa: 100 KB.
- Memoria de carga: 4 MB.
- Interfaz de comunicación: 1 puerto Ethernet integrado (soporta PROFINET y OPC UA).
- Capacidad de expansión: hasta 8 módulos de señal adicionales.
- Soporte para HMI y comunicación externa a través del entorno TIA Portal y protocolos como Modbus TCP/IP u OPC UA.

1.5.1.3 Arquitectura modular del S7-1200

La familia S7-1200 está diseñada con una arquitectura modular que permite la expansión del sistema mediante la adición de diferentes tipos de módulos:

- Módulos de señal (entradas/salidas digitales o analógicas).
- Módulos de comunicación (por ejemplo, RS485, RS232, Ethernet).
- Módulos de tecnología (como contadores rápidos o módulos de medición de temperatura).

El controlador CPU puede soportar hasta:

- 8 módulos de señal (SM).
- 3 módulos de comunicación (CM).
- 1 módulo de tecnología (TM).

1.5.2 Softwares de Simulación y Supervisión

1.5.2.1 TIA Portal (Totally Integrated Automation)

El Totally Integrated Automation Portal (TIA Portal) es el entorno de desarrollo integrado de Siemens que permite la programación, configuración, monitoreo y diagnóstico de sistemas de automatización industrial. Este software centraliza el diseño de proyectos que involucren controladores lógicos programables (PLC), interfaces Hombre-Máquina (HMI),

redes industriales y dispositivos de entrada/salida distribuidos. Gracias a su interfaz unificada y herramientas de diagnóstico avanzadas, TIA Portal facilita la gestión del ciclo de vida completo de una instalación automatizada, desde la planificación hasta la puesta en marcha y mantenimiento (Siemens, 2024).

En el presente proyecto, TIA Portal ha sido esencial para la programación del PLC Siemens S7-1200. La estructura de programación del software está basada en la organización por bloques, donde los Organizational Blocks (OB) definen el ciclo principal del programa, los Function Blocks (FB) encapsulan funciones reutilizables con memoria propia, y los Function Calls (FC) permiten operaciones sin retención de datos. Además, el entorno permite la gestión de variables globales y locales, promoviendo una programación modular y estructurada (Siemens, 2024).

Una funcionalidad clave utilizada en este proyecto es la facilidad de poder trabajar con PLCSIM. TIA Portal permite habilitar el S7/ISO-on-TCP vía NetToPLCSIM “integrado “en el PLC, lo que posibilita la exposición de variables definidas como nodos accesibles para aplicaciones externas. Esta configuración resulta fundamental para establecer la comunicación entre el entorno virtual 3D desarrollado y el sistema automatizado, mediante el intercambio de datos en tiempo real (Siemens, 2024).

1.5.2.2 PLCSIM

La simulación virtual de controladores lógicos programables (PLC) representa una fase crítica en el diseño y validación de sistemas automatizados. En este proyecto, se hace uso de PLCSIM, como plataforma para ejecutar el programa del PLC Siemens S7-1200 sin requerir hardware físico. Esta herramienta permite reproducir fielmente el comportamiento lógico del sistema de control, ofreciendo un entorno seguro para verificar la funcionalidad del programa antes de su implementación real.

PLCSIM permite la ejecución paso a paso del código, así como la supervisión en tiempo real de las variables internas, entradas y salidas digitales, lo que facilita el ajuste fino de la lógica de control. Su integración con TIA Portal proporciona un flujo de trabajo continuo entre el desarrollo y la simulación, permitiendo probar reacciones ante distintos escenarios operativos, como fallos de sensores, condiciones límite o secuencias automatizadas complejas [3].

1.5.2.3 NetToPLCSIM

NetToPLCsim es una extensión de red diseñada para el software PLCSIM, permitiendo que un PLC simulado en TIA Portal (como el S7-1200 o S7-1500) sea accesible desde aplicaciones externas a través de una red TCP/IP local. Esto es posible gracias a que NetToPLCsim actúa como un puente entre la simulación y la capa de transporte de datos, representando un servidor virtual que redirige la comunicación hacia el simulador del PLC mediante la interfaz S7online de Siemens [4].

A diferencia de los PLC físicos, los simuladores PLCSIM no exponen directamente una dirección IP accesible por red. NetToPLCsim resuelve este inconveniente al generar una IP virtual que otras herramientas como Node-RED u otros sistemas SCADA pueden utilizar para intercambiar datos con el PLC simulado. Esta solución resulta especialmente útil para entornos de pruebas, desarrollo e implementación de sistemas de automatización sin requerir hardware real [4].

1.5.2.4 Node-RED

Node-RED es una plataforma de desarrollo basada en flujos, diseñada para facilitar la conexión de dispositivos físicos, APIs y servicios en línea. Esta herramienta se basa en un entorno gráfico donde los flujos de datos se construyen mediante nodos que representan funciones, entradas/salidas o transformaciones lógicas.

En el trabajo de referencia de Herrera Flores & Valdiviezo Vilema, 2022, Node-RED fue implementado como interfaz de comunicación con un PLC simulado, utilizando la red TCP/IP para capturar, procesar y representar gráficamente el estado de variables como entradas digitales, señales de activación o estados del sistema. Además, se configuró como un panel HMI básico accesible desde navegadores web, lo que facilitó la visualización del proceso desde distintos dispositivos.

Esta lógica es replicada en el presente proyecto, donde Node-RED actúa como intermediario entre el entorno de simulación del PLC (PLCSIM + NetToPLCSIM) y el entorno virtual Minecraft. Su utilización permite no solo visualizar el estado del horno industrial automatizado en tiempo real, sino también interactuar con variables de proceso desde una interfaz web, simular condiciones de operación y validar el sistema sin la necesidad de hardware físico.

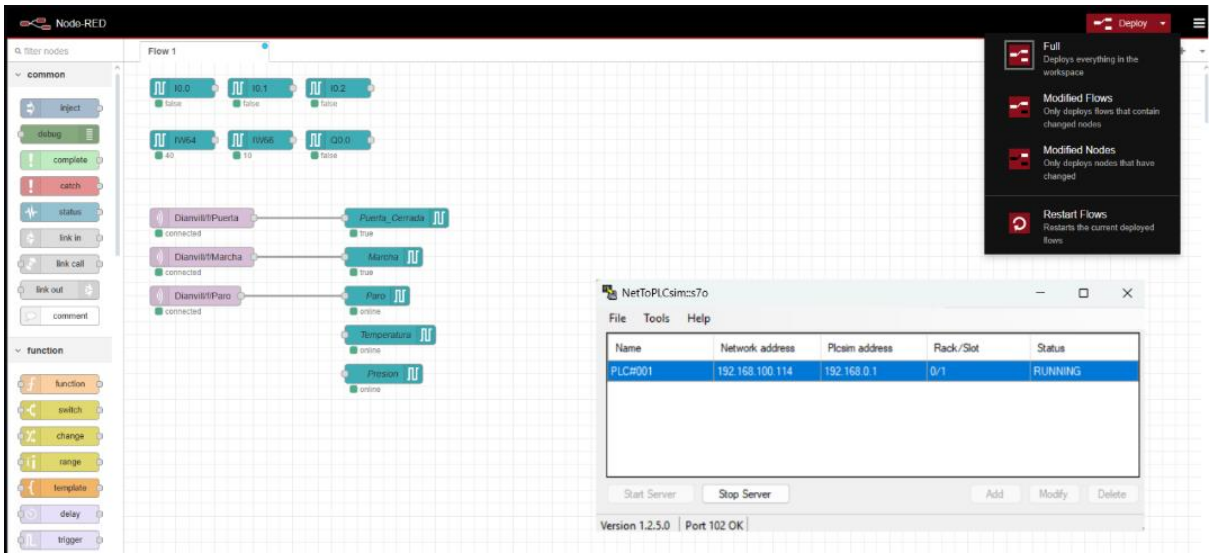


Ilustración 3. Comunicación entre NetToPLCsim con Node Red a partir de la dirección IP.

Fuente: Elaboración propia.

1.5.2.5 Minecraft

En este proyecto, Minecraft se emplea como un entorno representativo virtual que permite crear un gemelo digital interactivo del horno industrial. Una de sus ventajas es la

posibilidad de instalar mods especializados, como CC:Tweaked, que extiende las funcionalidades del juego para programar computadoras internas con el lenguaje Lua. A través de este mod es posible editar archivos internos como startup.lua (documento de texto ejecutado al inicio de la computadora virtual) y network.toml (archivo de configuración de red), lo cual permite definir la lógica de comunicación y la asociación de variables entre el entorno gráfico y el sistema de control.

La interacción entre Minecraft y Node-RED se realiza mediante WebSocket, un protocolo de comunicación en tiempo real basado en TCP, que mantiene un canal de conexión persistente y bidireccional entre cliente y servidor. Gracias a esta integración, el dashboard de Node-RED puede enviar comandos y recibir retroalimentación de los elementos representados en el mundo virtual, logrando un control remoto y seguro del prototipo simulado.

Diversos estudios han explorado el uso de Minecraft como plataforma de aprendizaje y de investigación. Por ejemplo, Perkins et al. (2015) lo plantean como una herramienta de investigación en entornos educativos, mientras que Marek et al. (2022) lo aplican para la enseñanza de mecánica de materiales mediante simulaciones lúdicas. Estos trabajos evidencian que Minecraft puede trascender su función de videojuego para convertirse en una plataforma académica y experimental.

1.5.3 Protocolos de comunicación

1.5.3.1 TCP/IP

El TCP/IP es un conjunto de protocolos que permite la comunicación entre dispositivos en red. TCP garantiza la entrega íntegra y ordenada de los datos dividiéndolos en paquetes, mientras que IP se encarga de direccionarlos correctamente hasta su destino (Fortinet, s.f.).

En este proyecto, TCP/IP se utiliza para que el PLC simulado en PLCSIM, a través de NetToPLCSIM, disponga de una dirección IP accesible. Esto permite su integración con Node-RED y posteriormente con el gemelo digital en Minecraft, asegurando el intercambio confiable de información entre todos los componentes.

1.5.3.2 WebSocket

WebSocket es un protocolo de comunicación basado en TCP que establece un canal bidireccional y persistente entre cliente y servidor. A diferencia del modelo tradicional de HTTP, permite un intercambio de mensajes en tiempo real con baja latencia y menor sobrecarga. Fue estandarizado por la IETF en la RFC 6455, lo que asegura su interoperabilidad en aplicaciones modernas [5].

En el ámbito académico, estudios como el de Wang et al. (2013) destacan que WebSocket ofrece una solución eficiente para aplicaciones que requieren transmisión continua de datos en tiempo real.

En este proyecto, WebSocket se emplea como protocolo de comunicación entre Node-RED y el entorno virtual en Minecraft (mod CC:Tweaked), permitiendo enviar comandos y recibir estados del horno simulado de forma bidireccional. Gracias a esta integración, el dashboard de Node-RED puede controlar y visualizar en tiempo real el comportamiento del gemelo digital desarrollado en Minecraft.

1.5.3.3 Diagrama de arquitectura de comunicación

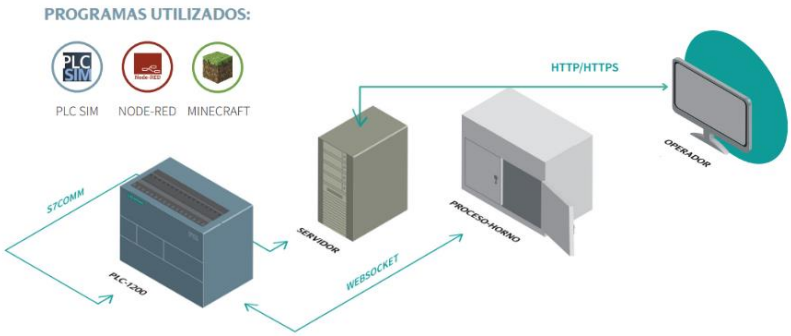


Ilustración 4. Flujo de conexión entre programas, servidores y protocolos de comunicación.

Fuente: Elaboración propia.

1.5.4 Gemelos digitales (Digital Twins)

1.5.4.1 Definición y relación con la Industria 4.0

Un gemelo digital es una representación virtual de un sistema físico, que replica su comportamiento en tiempo real. Es un componente clave en la digitalización industrial y la Industria 4.0. Permiten validar procesos, predecir fallas, optimizar rendimientos y capacitar operarios sin necesidad de interacción directa con equipos reales.

1.5.5 Normativas técnicas relacionadas

Normativas de seguridad de equipos eléctricos:

IEC 60204-1 establece requisitos para la seguridad de los equipos eléctricos de máquinas.

IEC 61439 regula los tableros de baja tensión.

El diseño de la arquitectura adoptada está alineado con:

IEC 61131-3, que promueve la abstracción del hardware mediante estructuras lógicas como bloques de datos (DB), evitando el acceso directo a entradas físicas desde sistemas externos.

IEEE 1451, que establece las directrices para sensores y transductores inteligentes, permitiendo digitalizar variables físicas simuladas mediante entornos virtuales.

ISA-95, que sugiere la estructuración de sistemas industriales en niveles funcionales; en este caso, Node-RED opera como la capa de conectividad (Nivel 3) entre el PLC y los sistemas externos.

Recomendaciones de Siemens, que sugieren usar marcas (%M) o bloques de datos (DB) en lugar de forzar entradas físicas, especialmente en escenarios de simulación o virtualización.

GAMP 5, que promueve el uso de entornos simulados trazables y validados antes del despliegue real de sistemas automatizados, garantizando la calidad y confiabilidad del diseño.

Capítulo 2

2. Metodología

2.1 Análisis del sistema y definición de requerimientos

Para el desarrollo del sistema de automatización y control de un horno industrial por histéresis en Electro Industrial Micabal S.A., se planteó un prototipo simulado con el fin de validar el comportamiento del proceso en un entorno seguro y de bajo costo. El objetivo fue reproducir de manera virtual la lógica de control y la supervisión remota, utilizando herramientas de programación, simulación y conectividad industrial, sin necesidad de hardware físico.

2.1.1 Requerimientos técnicos y empresariales

La empresa colaboradora ha manifestado la necesidad de contar con un sistema automatizado que sea capaz de operar un horno de convección utilizado para el secado de motores, el cual presenta limitaciones físicas que deben respetarse: temperatura máxima de operación de 150 °C y presión límite de 2 psi. Asimismo, se requiere que el horno funcione en dos modos de operación (manual y automático) seleccionables desde una interfaz HMI accesible desde la planta o remotamente desde la oficina del operador.

La empresa ha facilitado datos históricos de temperatura, tiempo y normativas técnicas, que han sido esenciales para modelar el comportamiento térmico del horno, definir tiempos de operación óptimos y establecer protocolos de seguridad. Esta cooperación ha permitido construir un modelo de control confiable y alineado a las condiciones reales del proceso.

Los requerimientos definidos para el proyecto son:

- Diseñar un sistema de control con modo manual y automático, aplicando un control por histéresis.

- Simular la comunicación entre el PLC virtual y plataformas externas mediante NetToPLCSIM, Node-RED y WebSocket.
- Desarrollar una interfaz HMI en Node-RED, accesible local y remotamente, con visualización de las principales variables del proceso.
- Representar el comportamiento del horno en un entorno 3D interactivo, mediante un gemelo digital en Minecraft.

2.1.2 Restricciones

- Presupuesto limitado para la adquisición de hardware físico.
- Cumplimiento de normativas de seguridad industrial vigentes.
- Compatibilidad con plataformas utilizadas en la empresa, como el PLC Siemens S7-1200 y el entorno TIA Portal.

2.1.3 Modos de operación

Modo Manual: permite al operador activar actuadores directamente desde el HMI o el entorno 3D. Incluye pruebas de encendido de resistencias, válvulas y ventiladores con medidas de seguridad básicas.

Modo Automático: el usuario ingresa una temperatura de consigna y un tiempo de operación. El sistema regula el proceso mediante un control por histéresis y bloquea reinicios prematuros hasta que la temperatura descienda a 40 °C, garantizando seguridad en la operación.

2.1.4 Razonamiento de diseño

Se optó por una solución completamente virtual que permitiera validar la lógica de control, la conectividad y la visualización de datos. La dinámica lenta del horno hace innecesario un control PID, siendo suficiente un control por histéresis con un margen de ±10 °C.

2.2 Justificación del uso de NETtoPLCSIM sobre OPC UA en entornos Siemens S7-1200

Si bien el protocolo OPC UA es estándar en la industria, el S7-1200 no soporta de forma nativa esta funcionalidad sin licencias adicionales. En cambio, NetToPLCSIM permite exponer una IP virtual del PLC simulado, habilitando la comunicación con Node-RED bajo protocolo S7, sin costos adicionales. Esto lo convierte en la mejor opción académica y de prototipado.

2.3 Etapas de desarrollo del proyecto

Con el fin de organizar el proceso metodológico, el proyecto se desarrolló en seis etapas consecutivas que integran la programación del controlador, la simulación, la comunicación y la validación final del sistema.

2.3.1 Etapa 1: Programación del control

El desarrollo del sistema inició con la programación en TIA Portal, donde se definieron estructuras de datos que permitieran organizar de manera clara y escalable la información del proceso. Para ello, se crearon UDT (User Defined Types) destinados a las entradas, salidas y variables generales, como por ejemplo la temperatura que es un valor por definir por el usuario, lo tenemos como variable “i” de int (entero) o las variables que tienen “b” que corresponden a valores booleanos a usar dentro de la programación. El uso de UDT responde a la necesidad de establecer un orden lógico y facilitar futuras modificaciones en la programación, ya que agrupar variables en estructuras permite mantener consistencia y simplificar la lectura del código.

UDT_Salidas

	Name	Data type	Default value	Accessible f...	Writa...	Visible in ...	Setpoint	Comment
1	bGenerador_Chispa	Bool	false	☑	☑	☑	☐	Q0.0 - Indica cuando es SEGU.
2	bEV_Superior	Bool	false	☑	☑	☑	☐	Q0.1 - ABRE ELECTROVALVUL...
3	bEV_Inferior	Bool	false	☑	☑	☑	☐	Q0.2 - ABRE ELECTROVALVUL...

Ilustración 5. UDT que definen los tipos de datos para las variables de salida.

UDT_General

	Name	Data type	Default value	Accessible f...	Writa...	Visible in ...	Setpoint	Comment
1	Startup	Struct		☑	☑	☑	☐	Variables de inicio/reset
2	Control	Struct		☑	☑	☑	☐	Estados de modo y proceso
3	Seguridad	Struct		☑	☑	☑	☐	Estado de sensores críticos
4	Temperatura	Struct		☑	☑	☑	☐	Control térmico del horno
5	Presion	Struct		☑	☑	☑	☐	Valores de presión
6	Estado	Struct		☑	☑	☑	☐	Estado general del horno
7	Manual	Struct		☑	☑	☑	☐	Ordenes del operador en ma...
8	Automatico	Struct		☑	☑	☑	☐	

Ilustración 7. UDT que definen variables generales que se usarán para la programación PLC.

UDT_Entradas

	Name	Data type	Default value	Accessible f...	Writa...	Visible in ...	Setpoint	Comment
1	bBoton_Marcha	Bool	false	☑	☑	☑	☐	I0.0 - BOOLEANO 1=INICIA PR...
2	bSensor_Puerta	Bool	false	☑	☑	☑	☐	I0.1 - BOOLEANO 1=CERRAD...
3	bSensor_Llama	Bool	false	☑	☑	☑	☐	I0.2 - BOOLEANO 1=HAY LLA...
4	bBoton_Paro	Bool	false	☑	☑	☑	☐	I0.3 - BOOLEANO N/C 1=SE A...
5	iTemp_Actual	Int	0	☑	☑	☑	☐	IW64 - SENSOR INTERIOR DEL ...
6	iPresion_Actual	Int	0	☑	☑	☑	☐	IW66 - SENSOR VALVULA PRI...

Ilustración 6. UDT que definen los tipos de dato para las variables de entrada.

A partir de estas definiciones, se construyeron bloques de datos (DB): los DB de entradas agrupan las señales provenientes del operador (botón de marcha, paro, consignas), los DB de salidas representan los actuadores simulados (válvulas, resistencias, ventiladores) y el DB general integra las variables de proceso (temperatura, estados de histéresis, modos de operación y alarmas).

DB_General

	Name	Data type	Offset	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Comment
1	Static				☐	☐	☐	☐	☐	
2	Startup	Struct	0.0		☐	☑	☑	☑	☐	Variables
3	Control	Struct	2.0		☐	☑	☑	☑	☐	Estados d
4	Seguridad	Struct	4.0		☐	☑	☑	☑	☐	Estado de
5	Temperatura	Struct	6.0		☐	☑	☑	☑	☐	Control té
6	Presion	Struct	14.0		☐	☑	☑	☑	☐	Valores d
7	Estado	Struct	18.0		☐	☑	☑	☑	☐	Estado ge
8	Manual	Struct	278.0		☐	☑	☑	☑	☐	Ordenes d
9	Automatico	Struct	540.0		☐	☑	☑	☑	☐	

Ilustración 9. Bloques de datos (DB) para las variables generales, datos que vamos a usar durante la programación.

DB_Salidas

	Name	Data type	Offset	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Comment
1	Static				☐	☐	☐	☐	☐	
2	bGenerador_Chispa	Bool	0.0	false	☐	☑	☑	☑	☐	Q0.0 - Indi
3	bEV_Superior	Bool	0.1	false	☐	☑	☑	☑	☐	Q0.1 - ABR
4	bEV_Inferior	Bool	0.2	false	☐	☑	☑	☑	☐	Q0.2 - ABR

Ilustración 8. Bloques de datos (DB) para las variables de salida.

DB_Entradas

	Name	Data type	Offset	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Comment
1	Static									
2	bBoton_Marcha	Bool	0.0	false						I0.0 - BOO
3	bSensor_Puerta	Bool	0.1	false						I0.1 - BOO
4	bSensor_Llama	Bool	0.2	false						I0.2 - BOO
5	bBoton_Paro	Bool	0.3	false						I0.3 - BOO
6	iTemp_Actual	Int	2.0	0						IW64 - SEN
7	iPresion_Actual	Int	4.0	0						IW66 - SEN

Ilustración 11. Bloques de datos (DB) para las variables de entrada.

	Name	Address	Display format	Monitor value	Modify value	Comment
1	"DB_Salidas".bEV_Inferior	%DB2.DBX0.2	Bool		FALSE	
2	"DB_Salidas".bEV_Superior	%DB2.DBX0.1	Bool		FALSE	
3	"DB_Salidas".bGenerador_Chispa	%DB2.DBX0.0	Bool		FALSE	
4	"DB_Entradas".bBoton_Marcha	%DB1.DBX0.0	Bool		FALSE	
5	"DB_Entradas".bBoton_Paro	%DB1.DBX0.3	Bool		FALSE	
6	"DB_Entradas".bSensor_Llama	%DB1.DBX0.2	Bool		FALSE	
7	"DB_Entradas".bSensor_Puerta	%DB1.DBX0.1	Bool		FALSE	

Ilustración 10. Tablas de forzado para observar las variables de interés (datos de diferentes bloques la que sea de interés en el momento).

La lógica principal del sistema se implementó en el OB1 (Main), donde se programaron los modos de operación manual y automático, así como el control por histéresis. Este bloque constituye el núcleo del proceso, ya que gestiona el encendido de actuadores según consignas y condiciones de seguridad. Además, se utilizó el OB100 (Startup) para garantizar una inicialización segura del PLC virtual cada vez que la CPU se pone en marcha. En este bloque se incluyeron rutinas de reinicio de remanencias y la verificación de fallos de sensores; por ejemplo, en el caso de que la temperatura se registre en cero, el sistema interpreta este valor como indicio de un error de medición y activa las banderas de alarma correspondientes.

La programación general (Main) para el proceso de control de histéresis, automático y manual se muestra en el Apéndice A, Listado A.1. Luego, la programación Startup para “reiniciar” el PLC cuyo detalle completo puede consultarse en el Apéndice A, Listado A.2.

2.3.2 Etapa 2: Simulación del PLC

Una vez construida la lógica, se procedió a la simulación del PLC mediante PLCSIM. Sin embargo, se debe destacar que la dirección IP que PLCSIM genera no es una dirección real de red, por lo que no es posible establecer comunicación directa con aplicaciones externas. Para solventar esta limitación se empleó la herramienta NetToPLCSIM, que actúa como puente y expone una dirección IP válida y accesible en la red local.

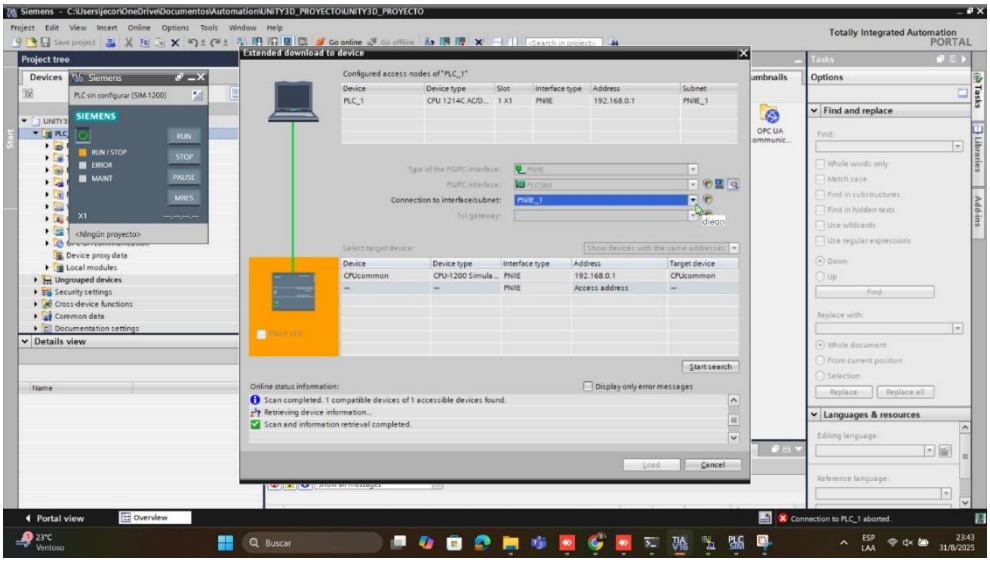


Ilustración 12. La dirección IP que proporciona el programa no es una dirección IP real como tal, que se pueda usar para la conexión entre TIA PORTAL y Node-RED.

Fuente: Elaboración propia.

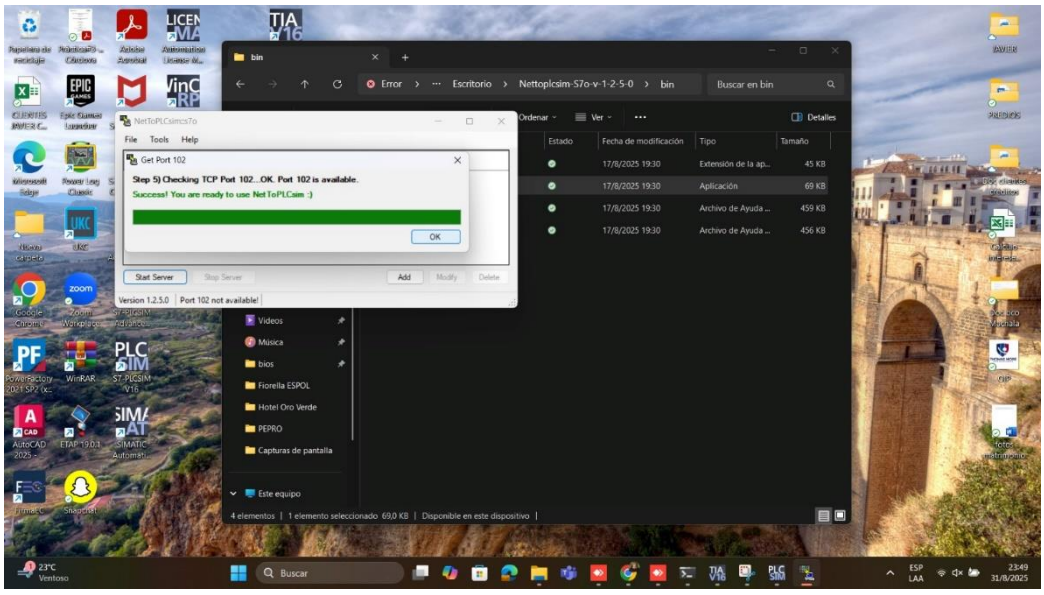


Ilustración 13. Se ejecuta el programa (como administrador) para que nos proporcione una IP real para la comunicación.

Fuente: Elaboración propia.

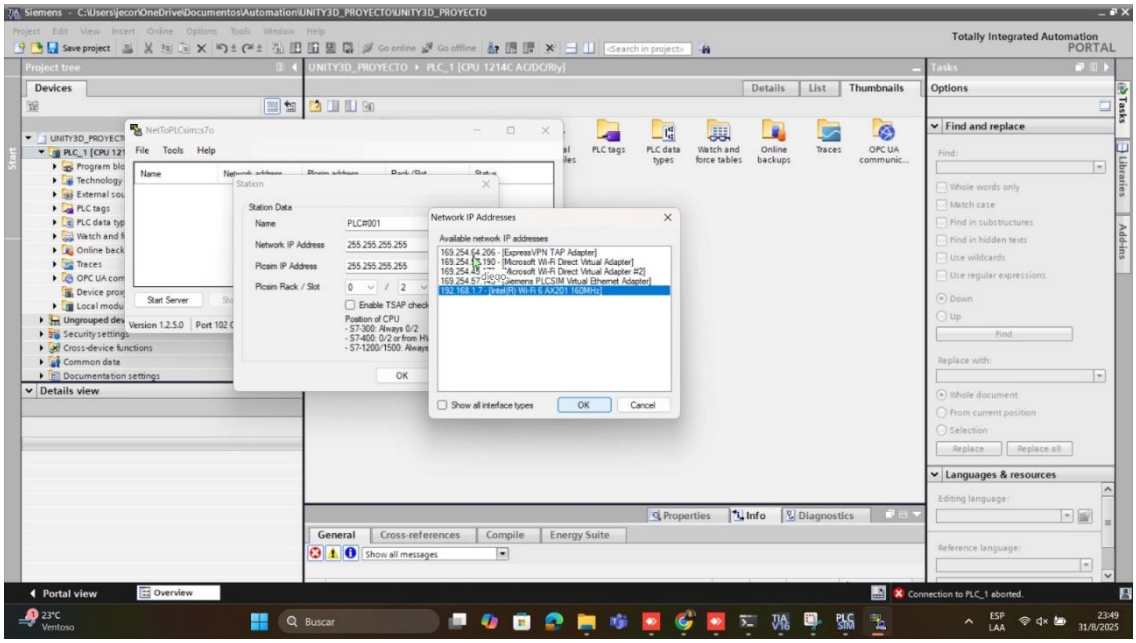


Ilustración 14. Selección de la dirección IP para comunicar el PLC con otros dispositivos mediante internet.

Fuente: Elaboración propia.

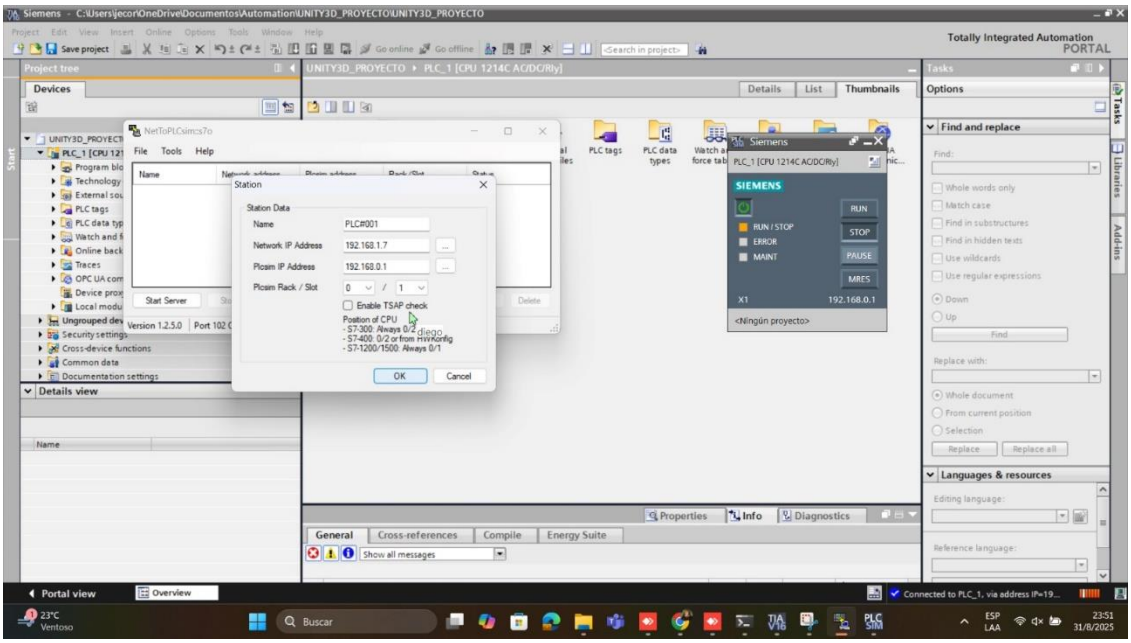


Ilustración 15. Finalmente, para la parte de PLCSIM Rack/Slot seleccionamos 0/1 que corresponde para el modelo que tenemos S7-1200 como se indica en la ventana emergente Station.

Fuente: Autoría propia

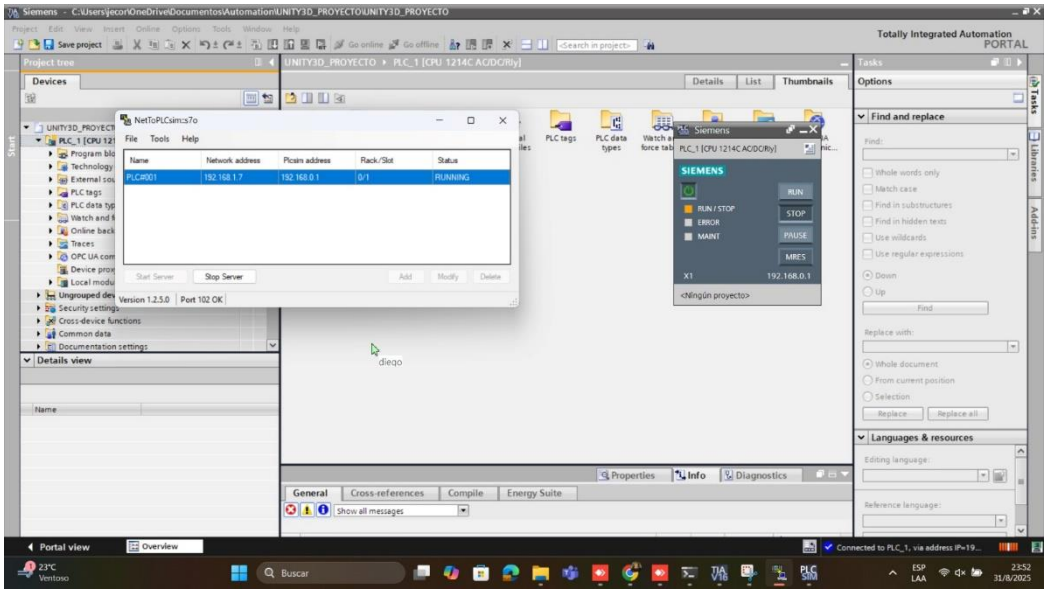


Ilustración 16. Para la selección de la dirección IP de PLCSIM seleccionamos el botón de opciones (…) como indica en la imagen para poder asignar la IP real.

Fuente: Autoría propia

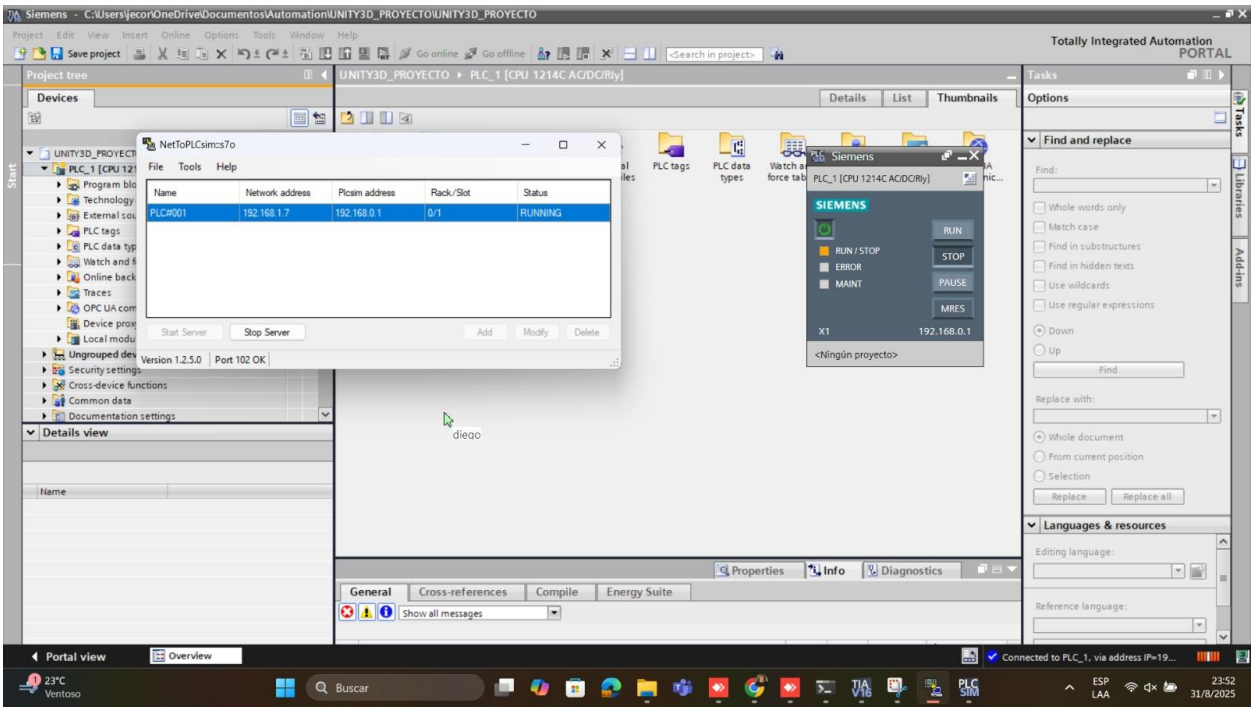


Ilustración 17. Tenemos la dirección IP para la red y para el PLC (mediante PLCSIM), en donde se puede observar que el PLC simulado se encuentra en estado “Running”.

Fuente: Autoría propia

Este procedimiento permitió asignar una dirección IP real al PLC virtual, configurando la comunicación a través del protocolo ISO-on-TCP en el puerto estándar 102. De esta manera, Node-RED y otros clientes pudieron acceder a las variables internas del controlador como si se tratara de un PLC físico. Finalmente, el sistema fue puesto en estado RUNNING, condición necesaria para ejecutar y validar la lógica implementada.

2.3.3 Etapa 3. Integración con Node-RED

Con la IP obtenida mediante NetToPLCSIM, se configuró la conexión entre el PLC simulado y Node-RED. En esta etapa se emplearon nodos específicos de comunicación S7 que permitieron la lectura y escritura de los bloques de datos previamente definidos en TIA Portal. Esta integración garantizó la transferencia bidireccional de información: Node-RED podía leer los estados de entradas y salidas, y al mismo tiempo enviar órdenes que modificaran el comportamiento del PLC virtual.

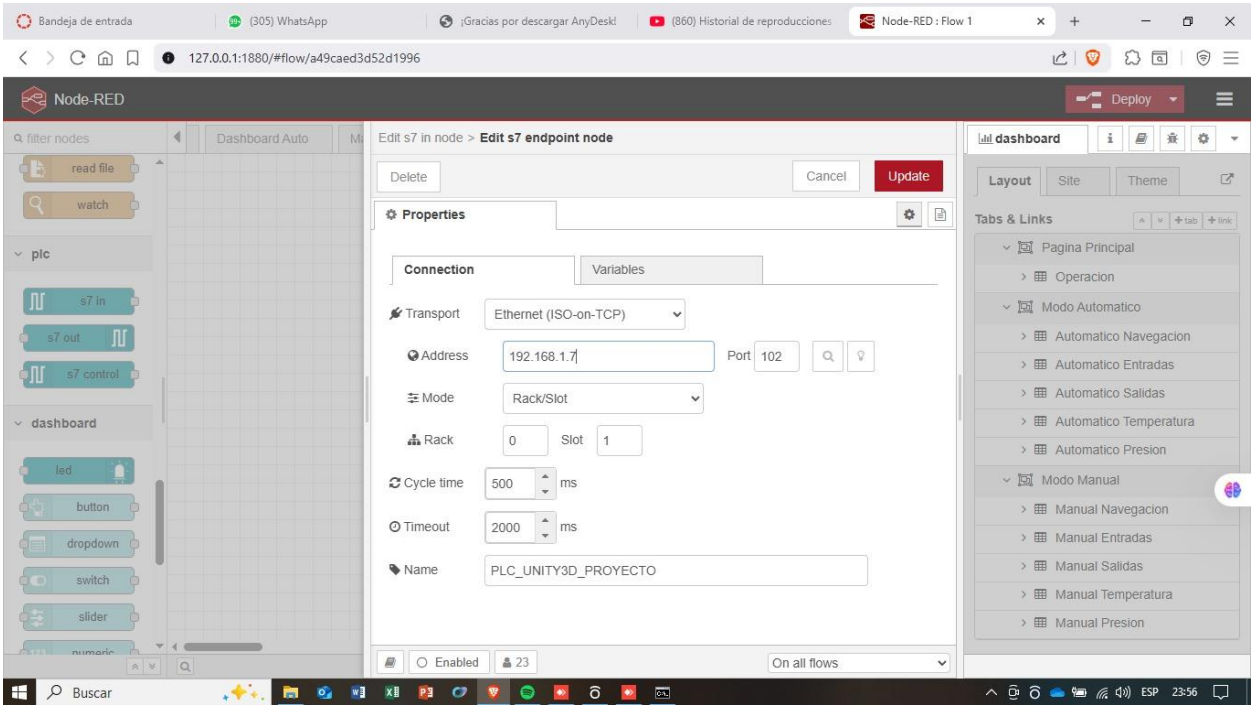


Ilustración 18. Agregamos un bloque S7 (en este caso de entrada) y vamos al apartado “Edit s7 endpoint node” en donde ingresamos la IP obtenida anteriormente mediante NetToPLCSIM en el apartado Address.

Fuente: Elaboración propia.

Posteriormente, se diseñaron dashboards interactivos que permiten al usuario operar el horno simulado en dos modalidades:

Modo Automático, en el cual el sistema regula la temperatura de acuerdo con consignas predefinidas aplicando control por histéresis.

Modo Manual, que otorga al operador control directo sobre cada actuador, con funciones de encendido y apagado supervisadas.

Adicionalmente, se desarrolló un flujo que expone las variables hacia un servidor WebSocket, lo que posibilita la conexión con el entorno tridimensional de Minecraft. Este paso resultó fundamental para habilitar la interacción entre la simulación lógica del PLC y su representación gráfica en el gemelo digital.

2.3.4 Etapa 4: Comunicación segura con Tailscale

Para habilitar la comunicación entre dispositivos ubicados en distintas redes físicas se implementó Tailscale, una solución basada en el protocolo WireGuard que permite establecer redes privadas virtuales. Cada dispositivo que instaló Tailscale recibió una dirección IP privada en el rango 100.x.x.x, independiente de la red local en la que estuviera conectado.

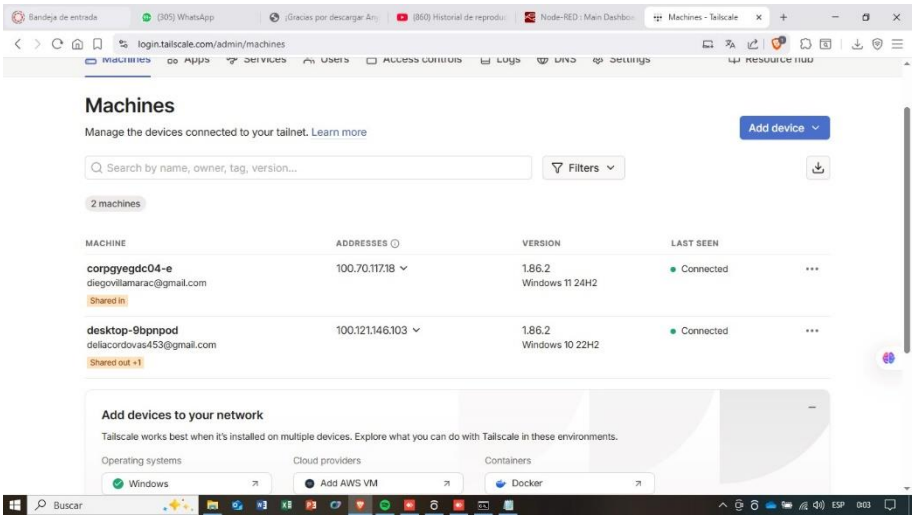


Ilustración 19. Uso de Tailscale para convertir los dispositivos en una red privada.

Fuente: Elaboración propia.

Gracias a esta capa de seguridad, la comunicación entre Node-RED, el PLC simulado y el entorno 3D en Minecraft pudo realizarse de manera segura y cifrada, sin necesidad de abrir puertos en Internet público. De esta forma, se garantizó que la arquitectura propuesta no solo fuese funcional, sino también robusta desde el punto de vista de la ciberseguridad.

2.3.5 Etapa 5. Gemelo digital en Minecraft

La última fase consistió en la creación del gemelo digital en el entorno tridimensional de Minecraft. Para ello se utilizó la plataforma Forge, sobre la cual se instalaron los mods CC:Tweaked y dependencias adicionales que permiten programar computadoras virtuales dentro del juego.

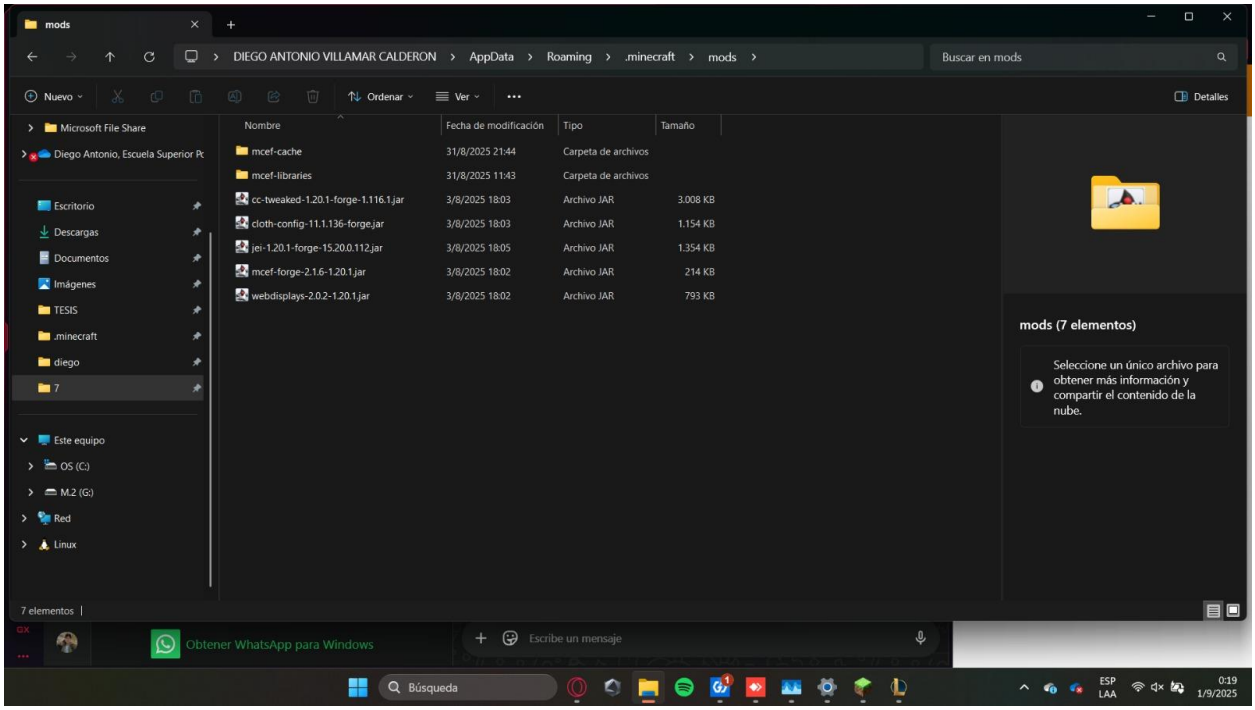


Ilustración 20. Verificación de instalación del mod CC:Tweaked.

Fuente: Elaboración propia.

En estas computadoras se configuraron scripts en Lua, principalmente startup.lua y main.lua, que establecen la conexión con Node-RED a través de WebSocket. La programación permitió que los actuadores representados en Minecraft (palancas, luces, mecanismos redstone) se activaran o desactivaran en función de las órdenes recibidas desde Node-RED, mientras que las entradas físicas del entorno virtual eran leídas y enviadas nuevamente al PLC simulado.

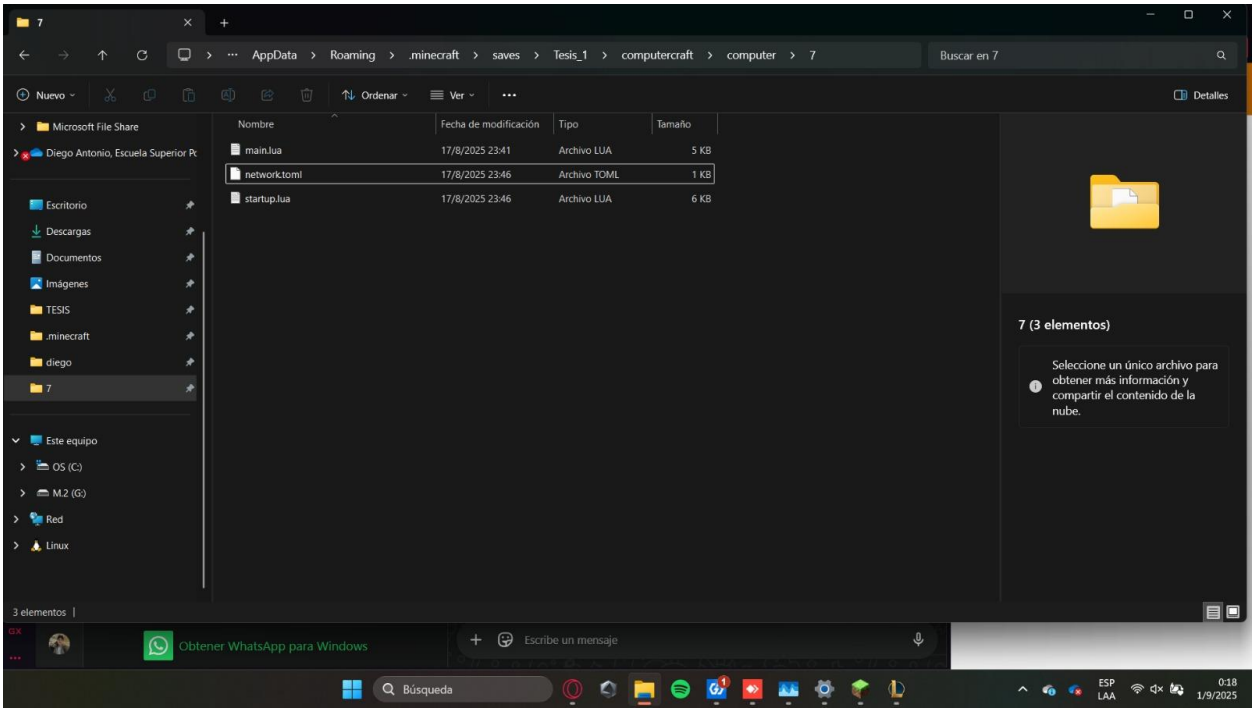


Ilustración 21. Ubicación de los scripts main.lua y startup.lua en nuestro proyecto, cuyos archivos de texto serán modificados para la conexión entre Node-RED y el entorno Minecraft a través de Websocket.

Fuente: Elaboración propia.

La programación completa para la parte de startup.lua y main.lua se encuentra en el apartado de Apéndice B, Listado B.1 y Listado B.2, la programación es la misma, ya que startup.lua funciona únicamente para inicializar el programa y ejecutarlo automáticamente.

El archivo network.toml cumplió la función de mapa de configuración, asociando las variables internas del PLC (por ejemplo, DB2_UDT_SALIDAS_bEV_Superior) con los tópicos manejados en WebSocket (ej. bEV_Superior). De esta manera, se estableció una

correspondencia directa entre las variables del controlador y los objetos interactivos en Minecraft.

El fragmento de código completo para network.toml puede verificarse en el Apéndice C, Listado C.1.

Capítulo 3

3. Resultados y Análisis

3.1 Simulación del PLC

La primera fase de validación consistió en ejecutar la lógica desarrollada en TIA Portal mediante el simulador PLCSIM. Como se explicó en la metodología, la dirección IP interna generada por PLCSIM no es accesible desde aplicaciones externas. Para solventar esta limitación se empleó NetToPLCSIM, que expone una dirección IP válida en la red local, permitiendo así la comunicación con Node-RED y otras plataformas.

La Ilustración 22 muestra la configuración de NetToPLCSIM con el PLC en estado RUNNING, lo que confirma que el simulador está operativo y listo para el intercambio de datos.

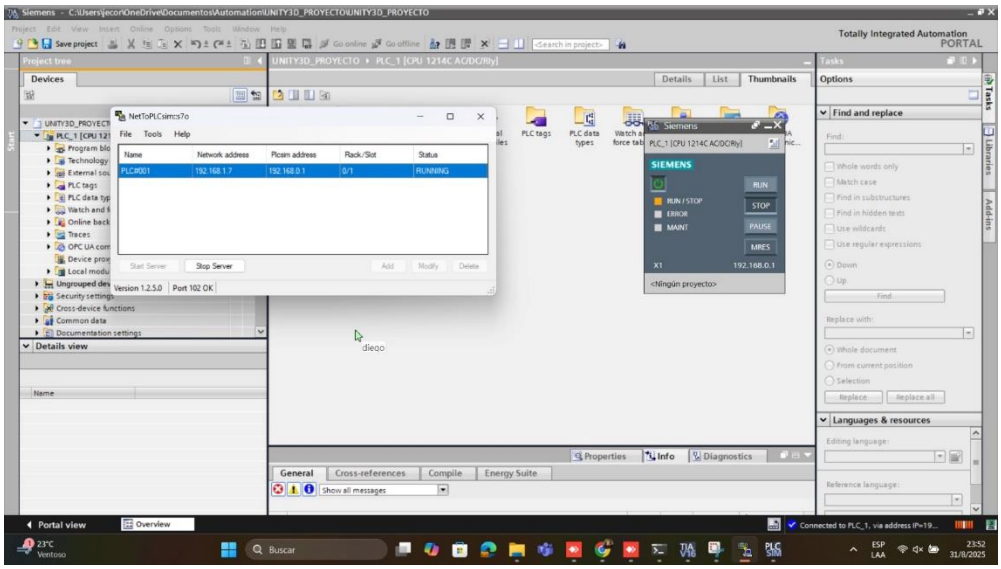


Ilustración 22. Configuración de NetToPLCSIM con el PLC simulado en ejecución.

Fuente: Elaboración propia

3.2 Integración con Node-RED

Una vez obtenida la IP accesible, se configuraron los nodos s7 in/out en Node-RED para establecer comunicación con los bloques de datos creados en TIA Portal. En la Ilustración 23 se observa el flujo denominado Escritura_Datos, que envía señales hacia el PLC simulado. Este flujo asegura que las entradas provenientes del dashboard y las salidas hacia los actuadores se encuentren correctamente enlazadas con las variables internas del PLC.

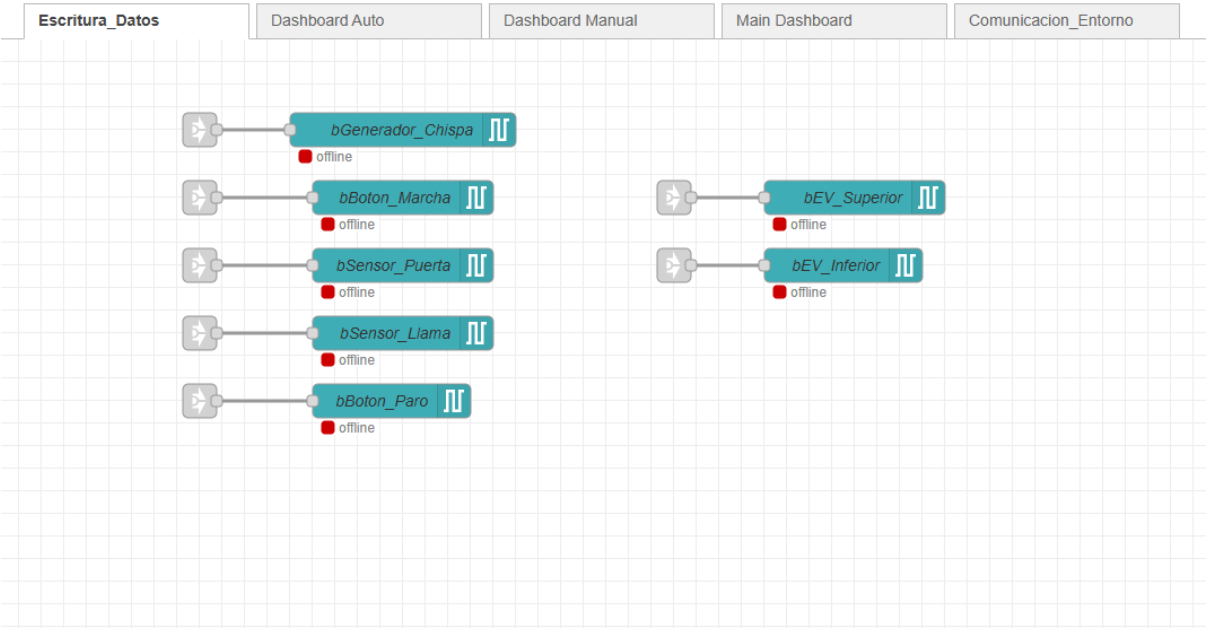


Ilustración 23. Flujo *Escritura_Datos* en Node-RED para el intercambio de variables con el PLC. *Fuente: Elaboración propia.*

3.3 Resultados del modo automático

En el modo automático, el operador únicamente debe seleccionar el comando de marcha, definir una temperatura objetivo y establecer un tiempo de operación. El sistema ejecuta la lógica de control por histéresis programada en el OB1, activando o desactivando las salidas según la condición de proceso.

La Ilustración 25 muestra el dashboard en modo automático, en el que se representan las entradas (botones de marcha y paro, sensores de puerta y llama) y las salidas (generador de chispa, electroválvula superior e inferior).

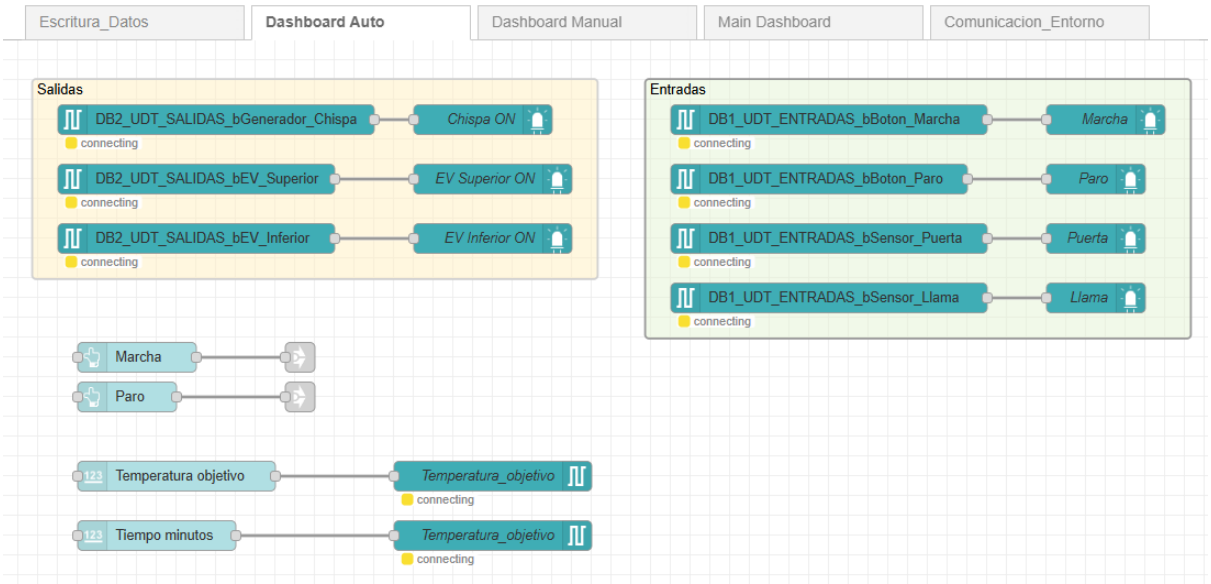


Ilustración 24. Flujo para el dashboard en Node-RED para el modo automático.

Fuente: Elaboración propia.

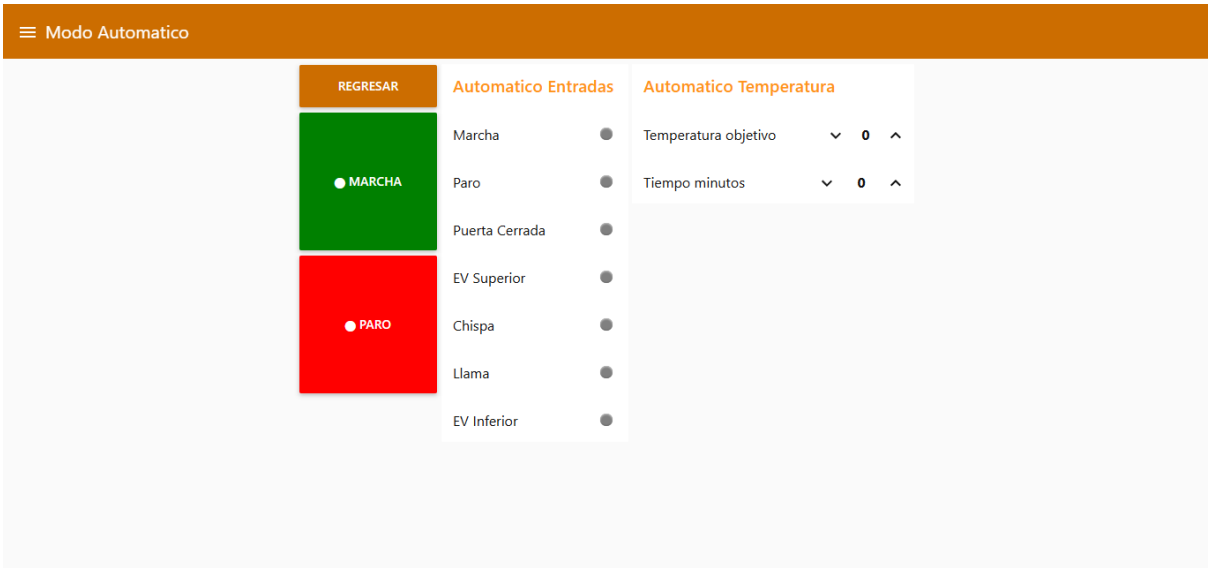


Ilustración 25. Interfaz HMI en Node-RED para el modo automático, con control por histéresis y consignas de temperatura y tiempo.

Fuente: Elaboración propia.

3.4 Resultados del modo manual

En el modo manual, el operador tiene control directo de los actuadores y variables de proceso, sin intervención del algoritmo de histéresis. Esto permite ajustar en tiempo real la presión y la temperatura mediante controles deslizantes (sliders), así como operar los botones de marcha y paro.

En la Ilustración 27 se observa el dashboard diseñado para este modo, donde las variables controladas se representan con indicadores tipo gauge. Este esquema resulta especialmente útil para pruebas y ajustes, aunque carece de la optimización automática que ofrece el modo anterior.

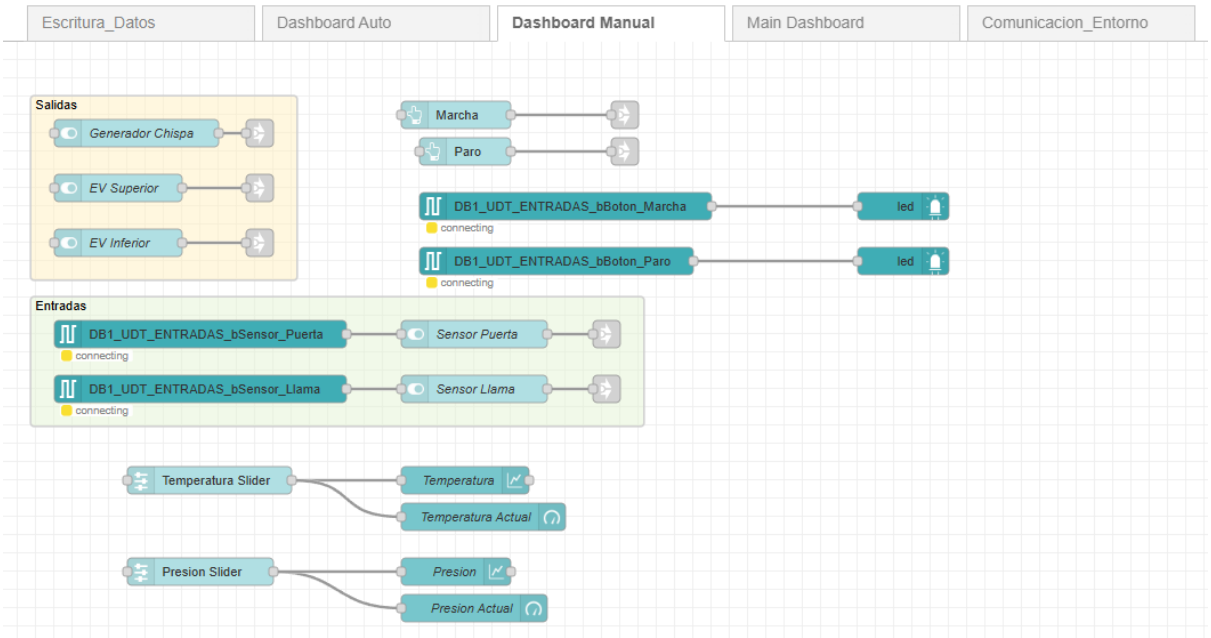


Ilustración 26. Flujo para el dashboard en Node-RED para el modo manual.

Fuente: Elaboración propia.

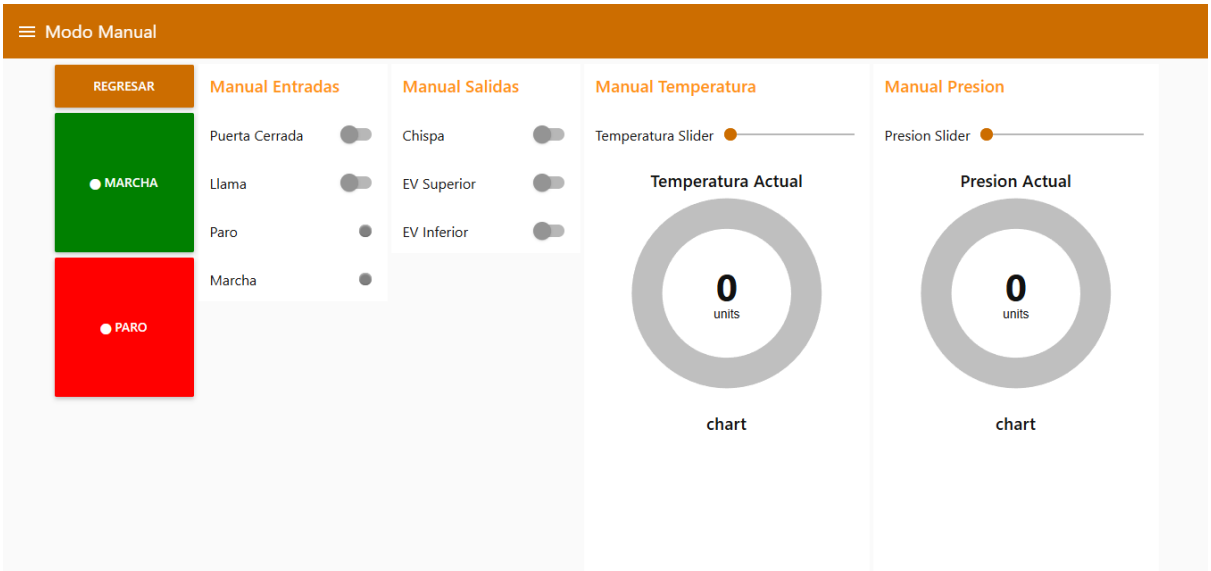


Ilustración 27. Interfaz HMI en Node-RED para el modo manual, con control directo de actuadores, presión y temperatura.

Fuente: Elaboración propia.

3.5 Selección de modos y navegación

El sistema de control contempla un dashboard principal, desde el cual el operador puede seleccionar el modo manual o el modo automático. Este flujo se presenta en la Ilustración 29, donde además se incluyen botones de retorno que facilitan la navegación entre pantallas.

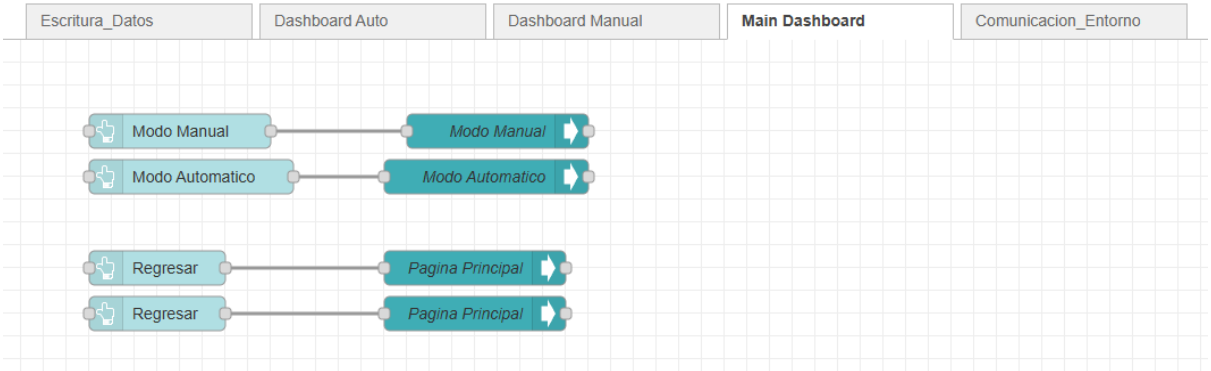


Ilustración 28. Flujo para el dashboard principal de selección de modo de operación.

Fuente: Elaboración propia.

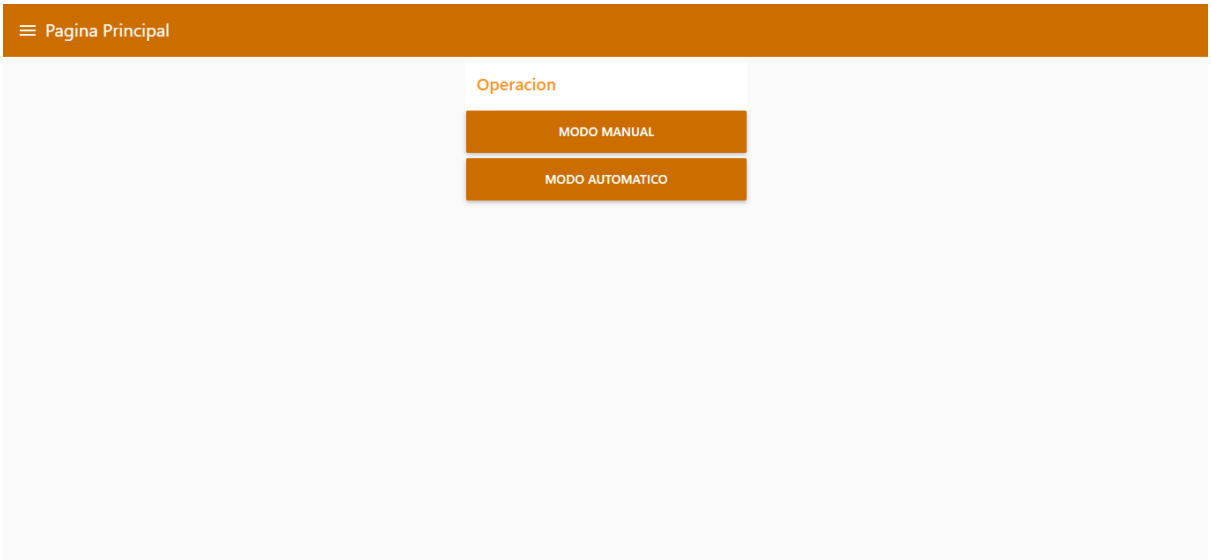


Ilustración 29. Dashboard principal en Node-RED para la selección de modo de operación (manual o automático).

Fuente: Elaboración propia.

3.6 Comunicación con Minecraft

Para validar la interacción con el gemelo digital, se desarrolló un flujo en Node-RED que expone las variables hacia un servidor WebSocket, configurado para ser consumido por Minecraft mediante el mod CC:Tweaked.

La Figura 27 muestra el flujo Comunicación_Entorno, encargado de enviar las salidas del PLC (EV superior, EV inferior y generador de chispa) hacia Minecraft, y de recibir las señales de entrada desde el entorno virtual.

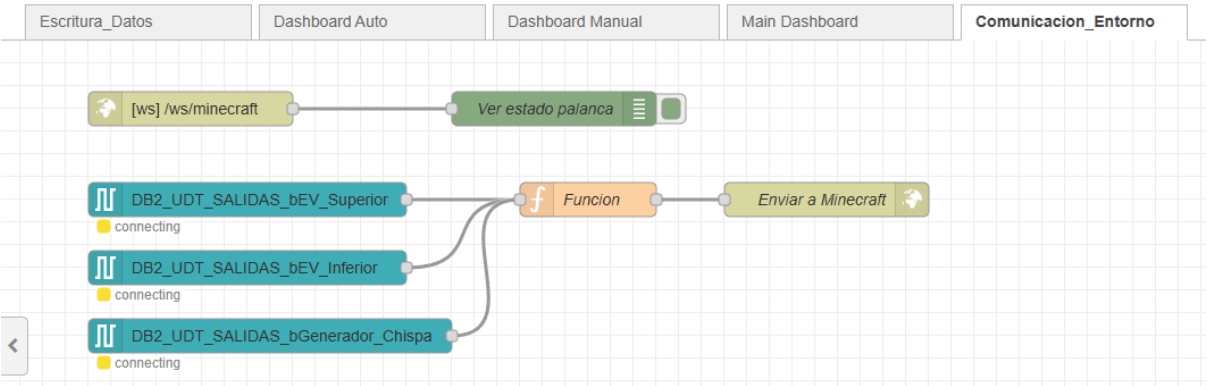


Ilustración 30. Flujo *Comunicación_Entorno* para la integración Node-RED–Minecraft mediante WebSocket.

Fuente: Elaboración propia.

Finalmente, la Figura 28 evidencia la operación conjunta del sistema: en el computador se observa el entorno virtual en Minecraft con los elementos programados, mientras que en la tableta se visualiza el dashboard en Node-RED con las variables en tiempo real. Esta validación confirma la sincronización entre el PLC simulado, Node-RED y el gemelo digital.

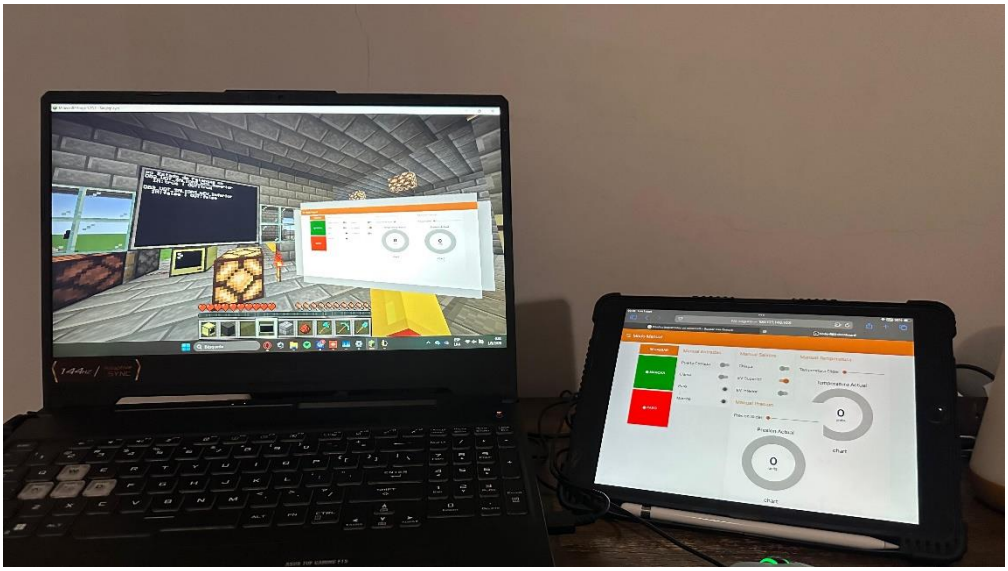


Ilustración 31. Integración del dashboard Node-RED con el gemelo digital en Minecraft.

Fuente: Elaboración propia.

3.7 Diseño eléctrico del panel

El conexionado de la CPU Siemens S7-1200 1214C AC/DC/Relé se realizó considerando la separación entre señales de control a 24 VDC y cargas de potencia a 220 VAC, con el fin de garantizar la seguridad y la correcta operación del sistema.

Las entradas digitales (DI) de la CPU se alimentaron con 24 VDC provenientes de la fuente auxiliar. Se cablearon de la siguiente manera:

- I0.0: pulsador de Marcha (contacto NO).
- I0.1: pulsador de Paro (contacto NC, fail-safe).
- I0.2: sensor de puerta cerrada (contacto de seguridad).

- I0.3: señal de “llama OK” proveniente del controlador de llama.

Todas las entradas comparten como referencia el borne 1M, conectado al negativo de la fuente de 24 VDC.

Para reforzar la seguridad del sistema, se incorporó un Paro de emergencia físico (E-Stop) cableado en serie con la salida de la fuente de 24 VDC. De esta forma, al presionar el pulsador de emergencia se interrumpe de inmediato la alimentación de todos los dispositivos de entrada, evitando que el PLC reciba cualquier señal desde los sensores o pulsadores. Este esquema asegura que, en condiciones críticas, el sistema quede inhabilitado en su totalidad sin depender de la lógica programada.

Las entradas analógicas (AI) se conectaron al módulo de expansión SM1231 AI (4x13-bit). Se integraron dos transmisores 4–20 mA:

- AI0: transmisor de temperatura del horno.
- AI1: transmisor de presión.

Los lazos se alimentaron con la fuente de 24 VDC, asegurando la puesta a tierra de blindajes para evitar interferencias electromagnéticas.

Las salidas digitales (DO) de la CPU son contactos de relé que no suministran tensión; por lo tanto, se emplearon para energizar las bobinas de relés intermedios Finder de 24 VDC. Estas bobinas fueron alimentadas con la fuente auxiliar, de modo que cada salida Q controla un relé:

- Q0.0 → Relé K1 → Transformador de ignición (AC).
- Q0.1 → Relé K2 → Electroválvula superior (AC).
- Q0.2 → Relé K3 → Electroválvula inferior (AC).

Los contactos de potencia de los relés intermedios conmutan la fase AC hacia las cargas, mientras que el neutro se cablea en forma directa. Se incluyó un disyuntor de control

bipolar para la CPU y otro para la fuente 24 VDC, además de fusibles de protección en cada rama de potencia.

El esquema completo de conexiones, con detalle de bornes X10, X11, X12 y módulo SM1231, se presenta en el Anexo 1.

El conexionado de fuerza del sistema se diseñó considerando la separación entre el circuito de control (24 VDC) y el circuito de potencia (220 VAC), con el objetivo de garantizar seguridad y confiabilidad en la operación.

Cada una de las cargas principales, el generador de chispa y las electroválvulas de gas (superiores e inferiores), es gobernada a través de relés intermedios (K1, K2 y K3), cuya bobina se alimenta con 24 VDC desde la fuente auxiliar, controlada por las salidas digitales del PLC Siemens S7-1200. De esta forma, las salidas del PLC (Q0.0, Q0.1 y Q0.2) no conmutan directamente cargas en AC, sino que activan las bobinas de los relés, aislando la electrónica de la CPU de los transitorios de potencia.

Los contactos de potencia de cada relé se conectan a través de un disyuntor bipolar de 10 A, el cual interrumpe simultáneamente las líneas de fase y neutro (L1 y N). Esto permite que cada carga esté protegida de manera independiente y evita retroalimentaciones peligrosas en caso de falla. El esquema eléctrico unifilar muestra claramente el recorrido de la alimentación desde la red AC hasta cada una de las cargas:

- K1: gobierna el transformador de ignición para el generador de chispa.
- K2: controla la electroválvula de gas superior.
- K3: controla la electroválvula de gas inferior.

El tablero se dimensionó con una cara útil aproximada de 400 × 400 mm (profundidad típica 180–220 mm), priorizando: i) separación entre potencia (230 VAC) y control (24 VDC), ii) espacios para canalizaciones y mantenimiento, iii) reserva para futuras expansiones

(≈ 20 %). La selección de este tamaño se justifica por el conjunto de equipos a montar en riel DIN y sus claros requerimientos de holgura.

Equipos y dimensiones (ancho × alto × fondo):

- CPU Siemens S7-1200 1214C AC/DC/Relé (6ES7 214-1BG40-0XB0): 110 × 100 × 75 mm.
- Módulo analógico SM1231 AI (p. ej., 4 canales): 45 × 100 × 75 mm.
- Fuente 24 VDC (tipo riel DIN, 60–100 W): ≈ 65–90 × 90–125 × 55–120 mm (varía por modelo; se consideró 80 mm de ancho como referencia).
- Relés/intermedios o mini-contactores para cargas AC (3 uds): ≈ 27–45 mm c/u de ancho según base/serie.
- Interruptores automáticos (MCB) bipolares 10 A (3 uds): ≈ 36 mm c/u.
- Borneras y seccionadores: ≈ 60–90 mm de ancho total (según polos).
- Canaletas porta-cables: 25–40 mm de ancho por lado (2 laterales).

Cálculo de ocupación horizontal (referencial):

- CPU (110) + SM1231 (45) + PSU (80) + 3 relés (3×30=90) + 3 MCB (3×36=108) + borneras (70) ≈ 503 mm de suma lineal.

Para una distribución ergonómica se dispusieron dos filas en riel DIN (potencia arriba, control abajo), reduciendo el ancho efectivo de cada fila a ≈ 250–300 mm. Con canaletas laterales (2×30=60 mm) y márgenes de borde (2×25=50 mm), un frente de ≈ 400 mm permite alojar una fila por nivel con holgura.

Cálculo de ocupación vertical (referencial):

- Altura de equipos (100–125 mm) + canaleta superior (30 mm) + canaleta inferior (30 mm) + separación entre filas (40–50 mm) + margen superior/inferior (2×25 mm) ≈ 320–360 mm, compatible con 400 mm de alto.

Criterios de disposición:

- Separación de dominios: fila superior para potencia AC (MCB, contactos de potencia de relés/contactor) y fila inferior para control 24 VDC (PSU, PLC, SM1231, bobinas de relés).
- Canalización y servicio: canaletas laterales y pasillos verticales de ≥ 25 mm para radios de curvatura y pruebas.
- Clareo térmico: ≥ 25 mm libres sobre/bajo equipos, y ≥ 10 mm entre laterales de módulos para convección natural.
- Reserva de expansión: ≥ 20 % del riel libre (espacio para un segundo módulo SM12xx o borneras adicionales).
- Seguridad: el E-Stop frontal actúa sobre la salida positiva de la PSU 24 V; la potencia AC dispone de MCB bipolares independientes por cada rama (ignitor, EV superior, EV inferior).
- Puesta a tierra: carril DIN y chasis a PE; blindajes de señales analógicas aterrizados en un solo punto.

Frente del tablero: piloto “Horno encendido” (verde) y Pulsador de Paro de Emergencia (rojo) accesibles; distribución conforme a criterios de visibilidad y alcance del operador.

Las ilustraciones de la disposición interna se presentan con representación esquemática y no a escala; las cotas anteriores se emplearon para el dimensionamiento, dejando holguras para canaletas, borneras, radios de cable y expansión.

Con este dimensionamiento, un gabinete 400×400×200 mm cubre los requerimientos de montaje, ventilación pasiva, seguridad y mantenibilidad del prototipo, manteniendo separación clara entre circuitos AC y DC y dejando reserva para crecimiento.

De igual manera, el diseño del tablero de control con la disposición de los componentes y el esquema de fuerza también se incluye en los Anexos, sirviendo de complemento gráfico a la descripción metodológica.

3.8 Diseño del sistema alternativo con lógica de relés

Como alternativa al control mediante PLC, se desarrolló un esquema de automatización basado en lógica de relés, en el que se implementó tanto el diagrama de control como el de fuerza. En esta opción se tomaron las electroválvulas como un conjunto único, gobernado a través de un contactor principal, y se utilizó un timer ON-delay para garantizar la seguridad en la secuencia de encendido. Este temporizador retrasa la activación del transformador de ignición, de manera que solo se permita la chispa si la electroválvula ya está habilitada y se interrumpe el ciclo en caso de no detectarse la señal de llama dentro del tiempo programado.

Se incorporaron elementos de seguridad adicionales como el presostato, encargado de fijar límites máximos y mínimos de presión en la línea de gas, y el sensor de puerta cerrada, que condiciona el arranque del sistema. Asimismo, se integraron alarmas visuales y acústicas, así como el paro de emergencia, que corta inmediatamente la alimentación del circuito de control en caso de falla.

El diagrama de fuerza se diseñó en corriente alterna (AC), dado que las cargas principales (electroválvulas y transformador de ignición) operan en 220 VAC, mientras que el diagrama de control se implementó en corriente continua (DC) debido a que los sensores y señalizaciones funcionan a 24 VDC.

El tablero de control para esta opción mantiene el mismo dimensionamiento de 40 × 40 cm, justificado en función de los componentes a instalar: relés, temporizadores, contactores, protecciones y canaletas. Al igual que en la solución con PLC, se consideró espacio para canalización, ventilación pasiva y reserva de expansión. Cabe destacar que el

dibujo presentado no está a escala, pero representa fielmente la disposición de los equipos y el conexionado interno.

Los diagramas completos, así como el diseño del tablero, se encuentran en los siguientes anexos:

- Anexo 5: Diagrama de control con lógica de relés.
- Anexo 6: Diagrama de fuerza con lógica de relés.
- Anexo 7: Diseño del tablero de control (vista interior).
- Anexo 8: Diseño del tablero de control (vista exterior).

3.9 Comparación de soluciones

El análisis de costos entre un sistema de control industrial basado en PLC y uno con lógica de relés constituye un aspecto clave para evaluar la factibilidad técnica y económica de la automatización de un horno. Aunque los relés presentan una inversión inicial más baja y son de uso extendido en sistemas tradicionales, los PLC ofrecen una mayor capacidad de integración, flexibilidad y reducción de tiempos de intervención en caso de fallas o modificaciones. A través de las tablas siguientes se detallan los costos estimados para cada alternativa, lo que permite realizar una comparación directa de los requerimientos de inversión en función de los equipos y accesorios considerados.

Cantidad	Descripción técnica	Precio unitario	Precio real	Marca
1	Módulo entradas analógicas 4AI 13-bit (0–10 V / 4–20 mA)	$ 586,00	$ 586,00	Siemens
1	Módulo RTD 4 canales (PT100/PT1000)	$ 600,00	$ 600,00	Siemens
1	Sonda temperatura PT100 con vaina + transmisor 4–20 mA	$ 22,81	$ 22,81	WIKA
1	Transmisor de presión 0–1 psi (0–70 mbar) salida 4–20 mA	$ 200,00	$ 200,00	Dwyer
1	Detector de llama UV para quemador industrial	$ 185,00	$ 185,00	Honeywell
1	Fin de carrera (microswitch)	$ 1,80	$ 1,80	Omron
1	Interruptor de seguridad de puerta (codificado)	$ 352,79	$ 352,79	Schmersal
1	Transformador de ignición (activador de chispa) para gas	$ 250,00	$ 250,00	Brahma
1	Electrodo de encendido cerámico con cable HV	$ 15,00	$ 15,00	Genérico industrial
5	Cable alta tensión silicona para ignición (15 kV)	$ 5,00	$ 25,00	Genérico industrial
3	Relé intermedio, bobina 24 V DC	$ 8,00	$ 24,00	Finder
2	Base para relé	$ 1,50	$ 3,00	Finder
2	Fuente conmutada 24 VDC, 60 W, carril DIN	$ 100,00	$ 200,00	Mean Well
2	Breaker MCB 2 A curva C (carril DIN)	$ 50,00	$ 100,00	Siemens/Schneider
1	Borneras/regletas de señal y potencia	$ 1,00	$ 1,00	Phoenix Contact / WAGO
30	Cable blindado 2 hilos 18–22 AWG	$ 2,05	$ 61,50	Condumex / General Cable
1	Gateway Node-RED (Raspberry Pi 4 + microSD + carcasa DIN + fuente 5	$ 170,00	$ 170,00	Raspberry/Genérico
1	Panel/HMI genérico con navegador o cliente VNC (7–10")	$ 230,00	$ 230,00	Genérico industrial
1	PLC Siemens S7-1200 (CPU)	$ 650,00	$ 650,00	Siemens
1	Botón paro de emergencia	$ 2,30	$ 2,30	
1	Boton marcha	$ 1,57	$ 1,57	
1	Luz piloto roja	$ 2,32	$ 2,32	
1	Luz piloto verde	$ 2,32	$ 2,32	
4	Cable #16 (Rollos de 100m)	$ 31,00	$ 124,00	
1	Gabinete doble fondo 40x30x15	$ 38,00	$ 38,00	
2	Canaletas ranuradas 2.5 x 2 cm	$ 4,50	$ 9,00	
1	Controlador/relé de llama (flame safeguard) + base + amplificador UV	$ 455,00	$ 455,00	Honeywell
			$ 4.312,41	

Tabla 1. Costos de implementación del sistema con PLC

Fuente: Autoría propia.

En esta tabla se detallan los equipos necesarios para la implementación del control con un PLC Siemens, incluyendo módulos de entradas analógicas, módulos RTD, transmisores, sondas de temperatura y detectores de seguridad. Se muestran los precios unitarios, la cantidad requerida y el costo total real por cada componente. Este desglose permite evidenciar la inversión inicial necesaria en equipos de automatización moderna.

Descripción técnica	Precio unitario	Precio real	Marca
Breaker 2P-10A	$ 6,85	$ 20,55	Siemens
Breaker 2P-6A	$ 4,32	$ 8,64	Siemens
Transformador de ignición (activador de chispa) para gas	$ 250,00	$ 250,00	WIKA
Transmisor de presión 0–1 psi (0–70 mbar) salida 4–20 mA	$ 200,00	$ 200,00	Dwyer
Detector de llama UV para quemador industrial	$ 185,00	$ 185,00	Honeywell
Fuente conmutada 24 VDC, 60 W, carril DIN	$ 1,80	$ 1,80	Omron
Contactores	$ 10,25	$ 20,50	Schmersal
Timer On.Delay	$ 9,84	$ 9,84	Brahma
Luz piloto amarilla	$ 2,32	$ 6,96	Genérico industrial
Luz piloto verde	$ 2,32	$ 2,32	Genérico industrial
Botonera paro rojo	$ 1,57	$ 1,57	Finder
Botonera paro emergencia rojo	$ 2,30	$ 4,60	Finder
Botonera marcha verde	$ 1,57	$ 1,57	Mean Well
cable apantallado 2 hilos 22	$ 2,05	$ 61,50	
Gabinete doble fondo 30x30x15	$ 31,00	$ 31,00	
Canaletas ranuradas 2.5 x 2 cm	$ 4,50	$ 9,00	
Cable #16 (Rollos de 100m)	$ 31,00	$ 124,00	
		$ 938,85	

Tabla 2. Costos de implementación del sistema con lógica de relés

Fuente: Autoría propia.

Aquí se listan los componentes necesarios para el esquema de control basado en relés y protecciones convencionales, como breakers, transformadores de ignición, transmisores y detectores de llama. Los costos están expresados de forma similar al caso anterior, reflejando la menor inversión inicial en comparación con el PLC, aunque con limitaciones en cuanto a flexibilidad y escalabilidad.

3.10 Análisis de resultados

El diseño de la lógica de control del horno, implementado en TIA Portal, aprovecha la capacidad de estructurar las variables de forma ordenada y modular mediante el uso de UDTs (User Defined Types) y DBs (Data Blocks). Esta estructura permite una programación escalable, reutilizable y clara, lo que facilita tanto el mantenimiento como la expansión futura del sistema. La modularidad de esta metodología garantiza que, en caso de futuras modificaciones del proceso, se puedan realizar ajustes de manera eficiente y rápida.
La simulación en PLCSIM es un paso crucial para validar la lógica de control antes de realizar una implementación en hardware real. Al probar rutinas como secuencias de arranque, condiciones de seguridad y estrategias de control, se minimizan los riesgos de errores y se optimiza el tiempo de puesta en marcha, lo que contribuye a la confiabilidad del sistema.

La comunicación mediante NetToPLCSIM, que habilita la conexión del PLC virtual con plataformas externas a través de ISO-on-TCP, agrega una capa fundamental de integración. Esto permite simular un entorno industrial real, proporcionando la posibilidad de supervisión y control remoto, lo cual es esencial para garantizar el monitoreo continuo del proceso sin intervención física directa.

Por otro lado, la implementación de Node-RED como plataforma para la interfaz hombre-máquina (HMI) introduce un enfoque moderno y flexible para la visualización de variables y estados del horno. A través de flujos de datos y dashboards interactivos, el operador puede

gestionar tanto el modo automático como el manual, monitorear el estado de los actuadores y sensores, y tomar decisiones informadas en tiempo real.

El uso de Minecraft como gemelo digital, integrado con el mod CC:Tweaked y programado mediante scripts en Lua, representa una innovación significativa en este proyecto. Este entorno 3D proporciona una representación visual interactiva de los actuadores, sensores y estados operativos del sistema, facilitando la validación de la lógica de control y ofreciendo una herramienta didáctica que permite una mejor comprensión del proceso y su interacción. Finalmente, la validación mediante escenarios de prueba permite comprobar la robustez del sistema bajo diferentes condiciones. La implementación de control por histéresis, la activación de actuadores, los cambios de modos de operación, el paro de emergencia y la detección de fallas aseguran que el sistema sea confiable y cumpla con los requisitos de seguridad del proceso.

La comparación de costos entre la solución con PLC Siemens S7-1200 y la solución basada en lógica de relés demuestra diferencias significativas en términos de inversión inicial y escalabilidad a largo plazo.

Sistema con PLC

El costo total de la implementación del sistema con PLC Siemens S7-1200 fue de $4,312.41, lo que cubre los costos de la CPU, los módulos de expansión, el HMI, los sensores, los actuadores y la infraestructura de comunicación (Node-RED, WebSocket, etc.). Este enfoque, aunque más costoso inicialmente, ofrece una mayor flexibilidad, capacidad de expansión y control remoto, lo que lo convierte en una solución ideal para procesos industriales que requieren una automatización avanzada y supervisión remota.

Sistema con Lógica de Relés

En contraste, el sistema basado en lógica de relés tiene un costo significativamente menor, alcanzando un total de $938.85. Este sistema utiliza relés, temporizadores y contactores, lo que implica una solución más económica, pero con limitaciones en términos de escalabilidad y flexibilidad. Además, carece de las capacidades de control remoto y supervisión avanzadas que ofrece el sistema basado en PLC. Sin embargo, es una opción válida para procesos más sencillos o cuando el presupuesto es un factor crítico.

Capítulo 4

4. Conclusiones y recomendaciones

- La implementación del sistema de automatización utilizando el PLC Siemens S7-1200 permitió lograr un control preciso y flexible del horno de secado, cumpliendo con los objetivos de optimizar la eficiencia energética y la seguridad del proceso. La integración del módulo SM1231 para entradas analógicas y la comunicación con Node-RED para la supervisión remota proporcionaron una solución escalable que puede adaptarse a futuras expansiones del proceso, garantizando tanto el control local como la posibilidad de supervisar y ajustar parámetros de manera remota. Además, la simulación previa en PLCSIM, combinada con NetToPLCSIM, aseguró que el sistema fuera probado de manera virtual antes de su implementación física, reduciendo el riesgo de errores operativos.
- El gemelo digital en Minecraft proporcionó una visualización 3D interactiva, que no solo fue útil para la validación del sistema en un entorno visual, sino que también sirvió como herramienta educativa para comprender el comportamiento del proceso y la interacción entre los sensores y actuadores del sistema.
- La implementación del paro de emergencia directamente sobre la fuente de alimentación de 24 VDC, además de las señalizaciones de alarma y presostato para controlar límites de presión, demostró ser esencial para garantizar la seguridad operacional. El control por histéresis implementado para el encendido y apagado de los actuadores de gas (ignitor y electroválvulas) permitió una gestión eficiente de las cargas inductivas, mientras que la detección de fallas como la falta de llama o la

variación de temperatura y presión contribuyó a mantener un entorno seguro tanto para el operador como para los equipos.

- La comparación de costos entre las dos alternativas de automatización (PLC vs. lógica de relés) mostró que, aunque el sistema con PLC representa una mayor inversión inicial (aproximadamente $4,312.41 frente a $938.85 para la lógica de relés), su capacidad de integración, flexibilidad y facilidad para escalar el sistema en el futuro son factores determinantes para su elección en procesos industriales donde se requiere monitoreo remoto y control avanzado. Por otro lado, el sistema con lógica de relés sigue siendo una opción viable para procesos más simples o cuando el presupuesto es limitado.
- El tablero de control, se dimensionó correctamente para albergar todos los componentes esenciales, incluyendo la CPU, fuente de 24 VDC, relés de control, y disyuntores de protección. El diseño modular y el uso eficiente del espacio aseguraron un montaje ordenado y seguro, permitiendo el fácil acceso a los componentes para su mantenimiento. Este diseño también proporciona flexibilidad para futuras expansiones del sistema, asegurando que se puedan añadir más componentes o módulos sin la necesidad de rediseñar el tablero completo.
- La fase de validación realizada con escenarios de prueba resultó ser fundamental para garantizar la robustez del sistema en condiciones tanto normales como anormales. Las pruebas incluyeron la activación de actuadores, cambio de modos de operación, paro de emergencia, y detección de fallas, lo que permitió verificar que el sistema respondiera correctamente a las diferentes situaciones planteadas. Además, la integración con Minecraft para la validación visual del sistema proporcionó una herramienta útil para la simulación de condiciones y la verificación de la interacción entre los componentes del proceso.

Bibliografía

[1] A. A. Ba Villarreal, «Desarrollo y diseño de una interfaz para una red de PLC's basada en internet de las cosas,» Chetumal, 2021.

[2] Process Solutions Inc., «Process Solutions,» 19 Marzo 2025. [En línea]. Available: https://processsolutions.com/a-brief-history-of-programmable-logic-controllers-plcs/.

[3] C. A. Valverde Serrano y J. E. Endara López, «SIMULACIÓN DE UN PROCESO DE MOLIENDA DE BALANCEADO PARA CAMARÓN USANDO LAS HERRAMIENTAS PLCSIM Y TIA PORTAL,» Guayaquil, 2024.

[4] T. Wiens, *NetToPLCsim - Network extension for Plcsim*, 2016.

[5] A. Dianocu, The WebSocket Handbook, Ably, 2021.

[6] N. Bong, «Progressive Automations,» 26 Abril 2022. [En línea]. Available: https://www.progressiveautomations.com/blogs/news/the-evolution-of-automation.

[7] N. Agudelo, G. Tano y C. A. Vargas, «Historia de la Automatización,» Bogota, 2020.

[8] «Alciro,» [En línea]. Available: http://www.alciro.org/alciro/arduino_32/control-motor-marcha-paro-enclavamiento_553.htm.

[9] L. C. Espinoza Endara y K. A. Loja Rodas, «Análisis de la herramienta Unity 3D para la creación de procesos virtuales y su automatización con el PLC Simatic S7-1500,» Cuenca, 2024.

[10] J. J. Herrera Flores y M. Á. Valdiviezo Vilema, «Automatización de un Sistema de Riego para la Empresa Sisantu mediante el Internet de las Cosas (IoT),» Guayaquil, 2022.

[11] Z. Beck, B. Alpert, A. Bowman, W. R. Watson y A. Buganza Tepole, «Research Gate,» 2021. [En línea]. Available: https://www.researchgate.net/publication/366397811_Elasticity_Solver_in_Minecraft_for_Learning_Mechanics_of_Materials_by_Gaming.

[12] H. A. Engelbrecht, «Research Gate,» 2014. [En línea]. Available: https://www.researchgate.net/publication/271472055_Transforming_Minecraft_into_a_research_platform.

[13] Fortinet, «Fortinet,» [En línea]. Available: https://www.fortinet.com/lat/resources/cyberglossary/tcp-ip.

[14] Q. Liu y X. Sun, «Research of Web Real-Time Communication Based on Web Socket,» *Scientific Research*, 2013.

Apéndice A – Programación en TIA Portal

Listado A.1. Fragmento de la lógica en OB1 (Main) – control por histéresis.

Fuente: Elaboración propia.

```
// Flanco de subida para enclavamiento de Marcha
#bMarchaEdge := NOT #bMarchaPrev AND  "DB_Entradas".bBoton_Marcha;
#bMarchaPrev := "DB_Entradas".bBoton_Marcha;

// Enclavamiento del ciclo de marcha
IF #bMarchaEdge THEN
    #bProcesoActivo := TRUE;
END_IF;

// Botón de paro: abre EVs y apaga ciclo
IF "DB_Entradas".bBoton_Paro THEN
    #bProcesoActivo := FALSE;

    // Abrir EVs (energizar)
    "DB_Salidas".bEV_Superior := TRUE;
    "DB_Salidas".bEV_Inferior := TRUE;

    // Apagar chispa
    "DB_Salidas".bGenerador_Chispa := FALSE;

    // Reset de llama detectada
    "DB_Entradas".bSensor_Llama := FALSE;

ELSE

    // Lógica principal del proceso
    IF #bProcesoActivo THEN

        IF ("DB_Entradas".iPresion_Actual >= 6) AND "DB_Entradas".bSensor_Puerta
THEN
```

```
        // Activar electroválvulas directamente
        "DB_Salidas".bEV_Superior := TRUE;
        "DB_Salidas".bEV_Inferior := TRUE;

        // Activar generador de chispa
        "DB_Salidas".bGenerador_Chispa := TRUE;

        // Detectar llama
        IF "DB_Entradas".bSensor_Llama THEN
            #bLlamaDetectada := TRUE;
        END_IF;

    ELSE
        // Si condiciones no se cumplen, mantener todo apagado
        "DB_Salidas".bEV_Superior := FALSE;
        "DB_Salidas".bEV_Inferior := FALSE;
        "DB_Salidas".bGenerador_Chispa := FALSE;
        #bLlamaDetectada := FALSE;
    END_IF;

    // Control de histeresis si ya hay llama detectada
    IF #bLlamaDetectada THEN
        #TempAlta := "DB_General".Automatico.iTiempo_Secado + 10;
        #TempBaja := "DB_General".Automatico.iTiempo_Secado - 10;

        IF "DB_Entradas".iTemp_Actual < #TempBaja THEN
            "DB_Salidas".bEV_Superior := TRUE;
            "DB_Salidas".bEV_Inferior := TRUE;
        ELSIF "DB_Entradas".iTemp_Actual > #TempAlta THEN
            "DB_Salidas".bEV_Superior := FALSE;
            "DB_Salidas".bEV_Inferior := TRUE;
        END_IF;
    END_IF;

ELSE
    // Si no hay ciclo activo (ni paro), resetear salidas
    "DB_Salidas".bEV_Superior := FALSE;
    "DB_Salidas".bEV_Inferior := FALSE;
    "DB_Salidas".bGenerador_Chispa := FALSE;
```

```
        #bLlamaDetectada := FALSE;
    END_IF;
END_IF;
```

Listado A.2. Bloque OB100 (Startup) – inicialización segura.

Fuente: Elaboración propia.

```
// FB_Reset

    "DB_General".Startup.bReset_En_Progreso := TRUE;

    // Reset general de control
    "DB_General".Control.bProceso_Activo := FALSE;
    "DB_General".Control.bModo_Manual := FALSE;
    "DB_General".Control.bModo_Automatico := FALSE;

    // Reset seguridad
    "DB_General".Seguridad.bSeguridad_OK := FALSE;
    "DB_General".Seguridad.bPuerta_OK := FALSE;
    "DB_General".Seguridad.bLlama_OK := FALSE;
    "DB_General".Seguridad.bEmergencia_OK := FALSE;

    // Reset temperatura
    "DB_General".Temperatura.iTemp_Actual := 0;
    "DB_General".Temperatura.bQuemadores_Encendidos := FALSE;
    "DB_General".Temperatura.bTemp_OK := FALSE;

    // Reset presión
    "DB_General".Presion.iPresion_Actual := 0;
    "DB_General".Presion.bPresion_OK := FALSE;

    // Reset estado general
    "DB_General".Estado.iCodigo_Alarma := 0;
    "DB_General".Estado.bAlarma_Activa := FALSE;

    // Reset salidas físicas (por si están activas)
    "DB_Salidas".bGenerador_Chispa := FALSE;
    "DB_Salidas".bEV_Superior := FALSE;
```

```
    "DB_Salidas".bEV_Inferior := FALSE;

    // Finaliza reset
    "DB_General".Startup.bPLC_Startup := FALSE;
    "DB_General".Startup.bReset_En_Progreso := FALSE;
```

Apéndice B – Archivos de configuración Lua (Minecraft)

Listado B.1. `startup.lua` – script de arranque en CC:Tweaked.

Fuente: Elaboración propia.

```
-- ========= util: leer archivo por líneas =========
local function read_lines(path)
  if not fs.exists(path) then return {} end
  local f = fs.open(path, "r"); if not f then return {} end
  local s = f.readAll() or ""; f.close()
  local t = {}
  for line in (s.."\n"):gmatch("(.-)\n") do table.insert(t, line) end
  return t
end

-- ========= lector TOML sencillo =========
local function read_toml(path)
  local cfg = { palanca = {} }
  local current = nil
  for _, line in ipairs(read_lines(path)) do
    line = line:gsub("^%s+",""):gsub("%s+$","")
    if line == "" or line:match("^#") then
      -- skip
    else
      local sec = line:match("^%[([^%]]+)%]$")
      if sec then
        current = sec
      else
        local k,vq = line:match('^([%w_%.-]+)%s*=%s*"(.-)"$')
        local k2,vn = line:match("^([%w_%.-]+)%s*=%s*([%d%.]+)$")
        if current and current:match("^palanca%.") then
          local key = current:sub(("palanca."):len()+1)
          cfg.palanca[key] = cfg.palanca[key] or {}
```

```
          if k then cfg.palanca[key][k] = vq
          elseif k2 then cfg.palanca[key][k2] = tonumber(vn) end
        else
          if k then cfg[k] = vq
          elseif k2 then cfg[k2] = tonumber(vn) end
        end
      end
    end
  end
  return cfg
end

-- ========= cargar config =========
local cfg        = read_toml("network.toml")
local WS_URL     = cfg.ws_url or "ws://127.0.0.1:1880/ws/minecraft"
local AUTH_TOKEN = cfg.auth_token or ""
local HZ         = tonumber(cfg.telemetry_hz or 2)     -- 2 Hz = 500 ms
local MON_SIDE   = cfg.monitor_side or "top"

-- palancas (nombres locales ↔ tópico WS corto)
local palancas, ws_to_local = {}, {}
for local_name, t in pairs(cfg.palanca or {}) do
  palancas[local_name] = {
    lado = t.lado or "right",
    ws_topic = t.ws_topic or local_name,
    estado_in = false,
    estado_out = false,
    anterior = false
  }
  ws_to_local[palancas[local_name].ws_topic] = local_name
end
-- si no hay TOML, añadir dos por defecto (opc.)
if next(palancas) == nil then
  palancas["DB2_UDT_SALIDAS_bEV_Superior"]            =            {lado="right",
ws_topic="bEV_Superior", estado_in=false, estado_out=false, anterior=false}
  palancas["DB2_UDT_SALIDAS_bEV_Inferior"]            =             {lado="left",
ws_topic="bEV_Inferior", estado_in=false, estado_out=false, anterior=false}
  ws_to_local["bEV_Superior"] = "DB2_UDT_SALIDAS_bEV_Superior"
  ws_to_local["bEV_Inferior"] = "DB2_UDT_SALIDAS_bEV_Inferior"
end
```

```
-- ========= monitor opcional =========
local monitor = nil
if peripheral.isPresent(MON_SIDE) and peripheral.getType(MON_SIDE) == "monitor" then
  monitor = peripheral.wrap(MON_SIDE)
  monitor.setTextScale(1)
  monitor.setBackgroundColor(colors.black)
  monitor.setTextColor(colors.white)
end

local function pintar()
  term.clear(); term.setCursorPos(1,1)
  print("== Estado de Palancas ==")
  if monitor then monitor.clear(); monitor.setCursorPos(1,1); monitor.write("== Estado de Palancas ==") end
  local i = 2
  for nombre, d in pairs(palancas) do
    local linea = ("%s   IN:%s   OUT:%s"):format(nombre, tostring(d.estado_in), tostring(d.estado_out))
    print(linea)
    if monitor then
      monitor.setCursorPos(1,i);   monitor.write(nombre)
      monitor.setCursorPos(3,i+1);             monitor.write(("IN:%s             | OUT:%s"):format(tostring(d.estado_in), tostring(d.estado_out)))
      i = i + 3
    end
  end
end

local function log(s) print(("[%s] %s"):format(os.date("%H:%M:%S"), s)) end
local function send(ws, tbl)
  if AUTH_TOKEN ~= "" then tbl.auth = AUTH_TOKEN end
  ws.send(textutils.serializeJSON(tbl))
end

-- ========= bucle de conexión SIN goto =========
while true do
  log("Conectando a "..WS_URL)
  local ws, err = http.websocket(WS_URL)
```

```
  if not ws then
    log("No se pudo conectar: "..tostring(err))
    sleep(2)
  else
    print("Conectado a Node-RED"); pintar()
    send(ws, { dir="hello", who="cc_tweaked", ts=os.epoch("utc") })

    -- --- hilos ---
    local function rx()
      while true do
        local raw = ws.receive()
        if not raw then error("WS cerrado") end
        local ok, msg = pcall(textutils.unserializeJSON, raw)
        if ok and type(msg)=="table" and msg.dir=="to-mc" and
type(msg.topic)=="string" then
          local local_name = ws_to_local[msg.topic] or msg.topic
          local p = palancas[local_name]
          if p and msg.type=="bool" and type(msg.value)=="boolean" then
            p.estado_out = msg.value
            redstone.setOutput(p.lado, p.estado_out)
          end
        end
      end
    end

    local function tx_inputs()
      local dt = 1 / (HZ > 0 and HZ or 2)  -- 2 Hz = 0.5 s
      while true do
        for nombre, d in pairs(palancas) do
          d.estado_in = redstone.getInput(d.lado)
          if d.estado_in ~= d.anterior then
            send(ws, { dir="to-nr", topic=(d.ws_topic or nombre), type="bool",
value=d.estado_in })
            d.anterior = d.estado_in
          end
        end
        pintar()
        sleep(dt)
      end
    end
```

```
    local function heartbeats()
      while true do
        send(ws, { dir="heartbeat", ts=os.epoch("utc") })
        sleep(10)
      end
    end

    local ok, perr = pcall(function()
      parallel.waitForAny(rx, tx_inputs, heartbeats)
    end)

    pcall(function() ws.close() end)
    log("Desconectado: "..tostring(perr))
    sleep(2)
  end
end
```

Listado B.2. `main.lua` – funciones principales de comunicación y control.

Fuente: Elaboración propia.

```
-- ========= util: leer archivo por líneas =========
local function read_lines(path)
  if not fs.exists(path) then return {} end
  local f = fs.open(path, "r"); if not f then return {} end
  local s = f.readAll() or ""; f.close()
  local t = {}
  for line in (s.."\n"):gmatch("(.-)\n") do table.insert(t, line) end
  return t
end

-- ========= lector TOML sencillo =========
local function read_toml(path)
  local cfg = { palanca = {} }
  local current = nil
  for _, line in ipairs(read_lines(path)) do
    line = line:gsub("^%s+",""):gsub("%s+$","")
    if line == "" or line:match("^#") then
```

```
      -- skip
    else
      local sec = line:match("^%[([^%]]+)%]$")
      if sec then
        current = sec
      else
        local k,vq = line:match('^([%w_%.-]+)%s*=%s*"(.-)"$')
        local k2,vn = line:match("^([%w_%.-]+)%s*=%s*([%d%.]+)$")
        if current and current:match("^palanca%.") then
          local key = current:sub(("palanca."):len()+1)
          cfg.palanca[key] = cfg.palanca[key] or {}
          if k then cfg.palanca[key][k] = vq
          elseif k2 then cfg.palanca[key][k2] = tonumber(vn) end
        else
          if k then cfg[k] = vq
          elseif k2 then cfg[k2] = tonumber(vn) end
        end
      end
    end
  end
  return cfg
end

-- ========= cargar config =========
local cfg        = read_toml("network.toml")
local WS_URL     = cfg.ws_url or "ws://127.0.0.1:1880/ws/minecraft"
local AUTH_TOKEN = cfg.auth_token or ""
local HZ         = tonumber(cfg.telemetry_hz or 2)     -- 2 Hz = 500 ms
local MON_SIDE   = cfg.monitor_side or "top"

-- palancas (nombres locales ↔ tópico WS corto)
local palancas, ws_to_local = {}, {}
for local_name, t in pairs(cfg.palanca or {}) do
  palancas[local_name] = {
    lado = t.lado or "right",
    ws_topic = t.ws_topic or local_name,
    estado_in = false,
    estado_out = false,
    anterior = false
  }
```

```
  ws_to_local[palancas[local_name].ws_topic] = local_name
end
-- si no hay TOML, añadir dos por defecto (opc.)
if next(palancas) == nil then
  palancas["DB2_UDT_SALIDAS_bEV_Superior"]            =            {lado="right",
ws_topic="bEV_Superior", estado_in=false, estado_out=false, anterior=false}
  palancas["DB2_UDT_SALIDAS_bEV_Inferior"]            =             {lado="left",
ws_topic="bEV_Inferior", estado_in=false, estado_out=false, anterior=false}
  ws_to_local["bEV_Superior"] = "DB2_UDT_SALIDAS_bEV_Superior"
  ws_to_local["bEV_Inferior"] = "DB2_UDT_SALIDAS_bEV_Inferior"
end

-- ========= monitor opcional =========
local monitor = nil
if peripheral.isPresent(MON_SIDE) and peripheral.getType(MON_SIDE) == "monitor"
then
  monitor = peripheral.wrap(MON_SIDE)
  monitor.setTextScale(1)
  monitor.setBackgroundColor(colors.black)
  monitor.setTextColor(colors.white)
end

local function pintar()
  term.clear(); term.setCursorPos(1,1)
  print("== Estado de Palancas ==")
  if monitor then monitor.clear(); monitor.setCursorPos(1,1); monitor.write("==
Estado de Palancas ==") end
  local i = 2
  for nombre, d in pairs(palancas) do
    local linea = ("%s   IN:%s   OUT:%s"):format(nombre, tostring(d.estado_in),
tostring(d.estado_out))
    print(linea)
    if monitor then
      monitor.setCursorPos(1,i);   monitor.write(nombre)
      monitor.setCursorPos(3,i+1);             monitor.write(("IN:%s             |
OUT:%s"):format(tostring(d.estado_in), tostring(d.estado_out)))
      i = i + 3
    end
  end
end
```

```
local function log(s) print(("[%s] %s"):format(os.date("%H:%M:%S"), s)) end
local function send(ws, tbl)
  if AUTH_TOKEN ~= "" then tbl.auth = AUTH_TOKEN end
  ws.send(textutils.serializeJSON(tbl))
end

-- ========= bucle de conexión SIN goto =========
while true do
  log("Conectando a "..WS_URL)
  local ws, err = http.websocket(WS_URL)
  if not ws then
    log("No se pudo conectar: "..tostring(err))
    sleep(2)
  else
    print("Conectado a Node-RED"); pintar()
    send(ws, { dir="hello", who="cc_tweaked", ts=os.epoch("utc") })

    -- --- hilos ---
    local function rx()
      while true do
        local raw = ws.receive()
        if not raw then error("WS cerrado") end
        local ok, msg = pcall(textutils.unserializeJSON, raw)
        if    ok    and    type(msg)=="table"    and    msg.dir=="to-mc"    and
type(msg.topic)=="string" then
          local local_name = ws_to_local[msg.topic] or msg.topic
          local p = palancas[local_name]
          if p and msg.type=="bool" and type(msg.value)=="boolean" then
            p.estado_out = msg.value
            redstone.setOutput(p.lado, p.estado_out)
          end
        end
      end
    end

    local function tx_inputs()
      local dt = 1 / (HZ > 0 and HZ or 2)  -- 2 Hz = 0.5 s
      while true do
        for nombre, d in pairs(palancas) do
```

```
          d.estado_in = redstone.getInput(d.lado)
          if d.estado_in ~= d.anterior then
            send(ws, { dir="to-nr", topic=(d.ws_topic or nombre), type="bool",
value=d.estado_in })
            d.anterior = d.estado_in
          end
        end
        pintar()
        sleep(dt)
      end
    end

    local function heartbeats()
      while true do
        send(ws, { dir="heartbeat", ts=os.epoch("utc") })
        sleep(10)
      end
    end

    local ok, perr = pcall(function()
      parallel.waitForAny(rx, tx_inputs, heartbeats)
    end)

    pcall(function() ws.close() end)
    log("Desconectado: "..tostring(perr))
    sleep(2)
  end
end
```

Apéndice C – Archivo de configuración de network.toml

Listado C.1. `network.toml` – mapeo de variables del PLC a tópicos Websocket.

Fuente: Elaboración propia.

```
# Conexión a Node-RED (Laptop Servidor)
ws_url = "ws://100.121.146.103:1880/ws/minecraft"
auth_token = ""          # deje vacío si no valida token en Node-RED
telemetry_hz = 2         # 2 Hz = revisa entradas cada 500 ms
monitor_side = "top"     # "top", "left", "right", etc.
```

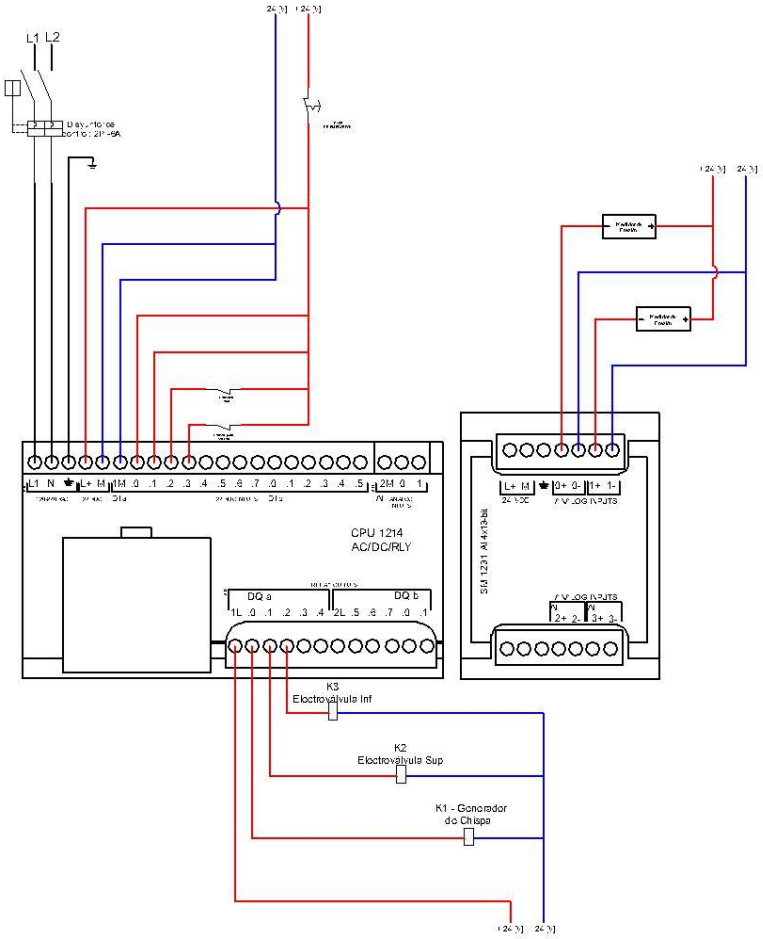
```
# Mapeo local ⇔ tópico WS (alias)
[palanca.DB2_UDT_SALIDAS_bEV_Superior]
lado = "right"
ws_topic = "bEV_Superior"

[palanca.DB2_UDT_SALIDAS_bEV_Inferior]
lado = "left"
ws_topic = "bEV_Inferior"
```

ANEXOS

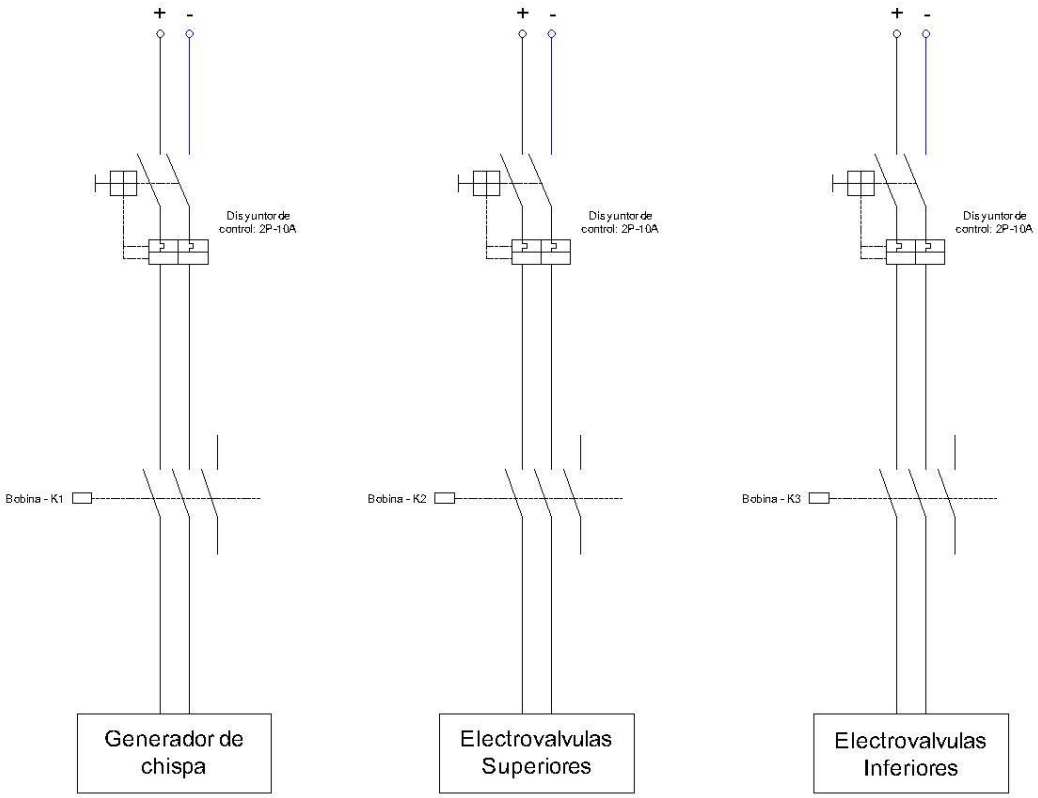
ANEXO 1:

Diagrama de conexión de PLC



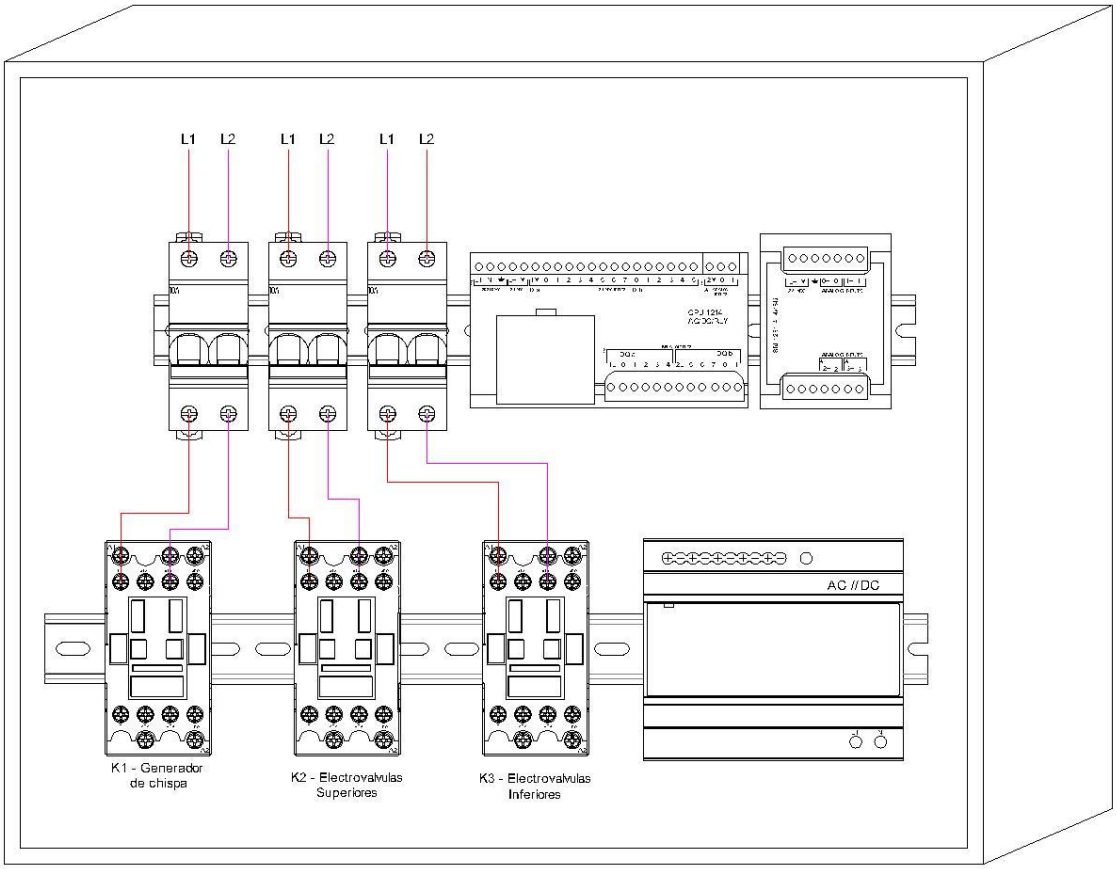
ANEXO 2

Diagrama de fuerza de PLC



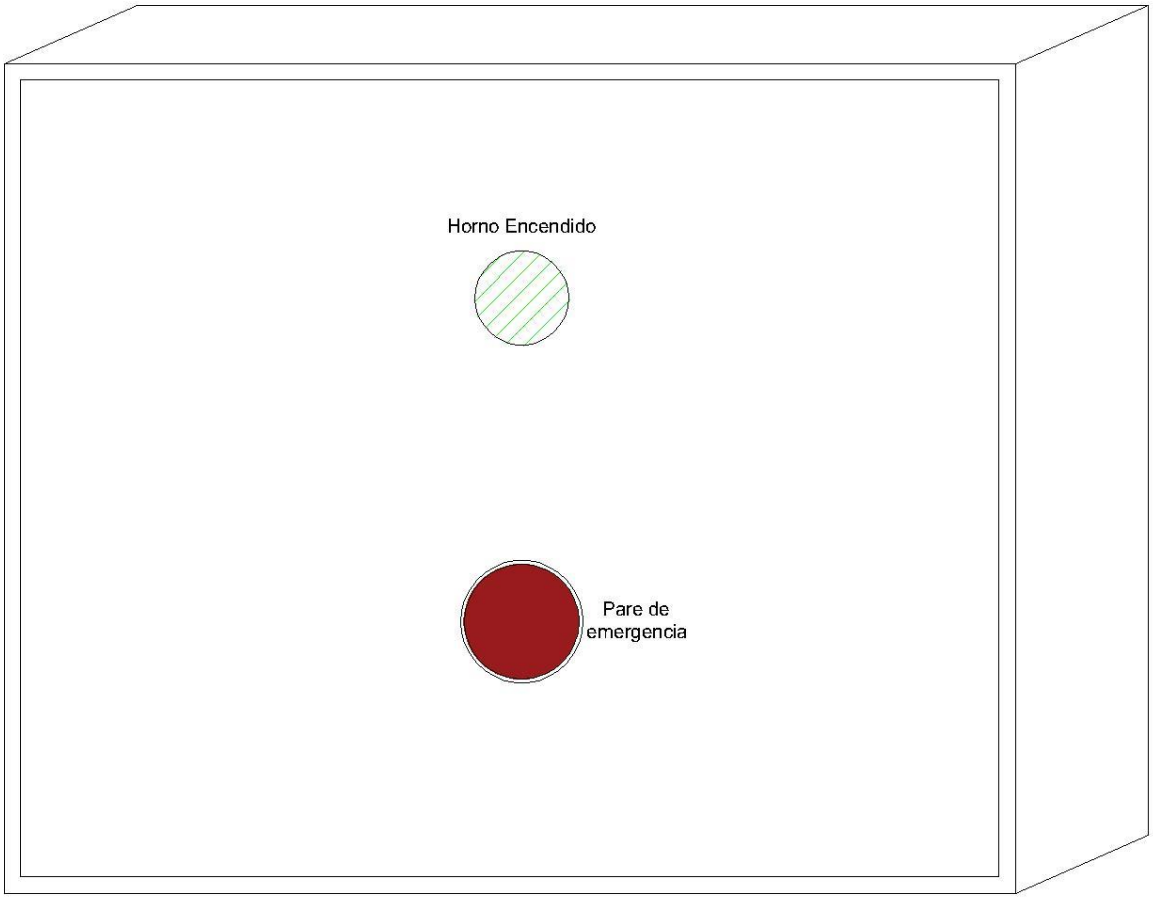
ANEXO 3

Representación en 2D Interior de Tablero de PLC



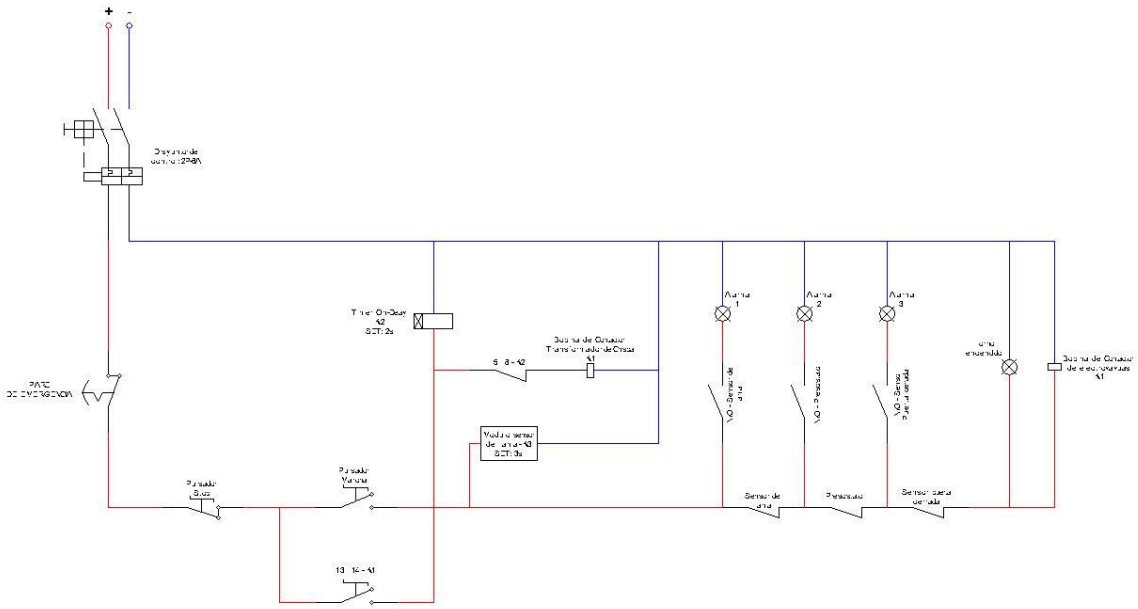
ANEXO 4

Representación en 2D Exterior de Tablero de PLC



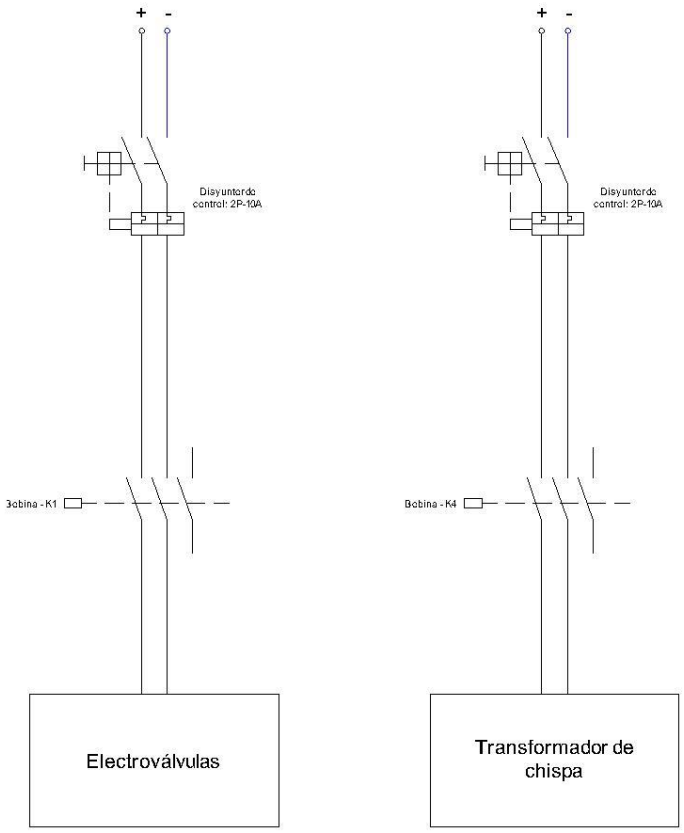
ANEXO 5

Diagrama de control de sistema mediante Relés



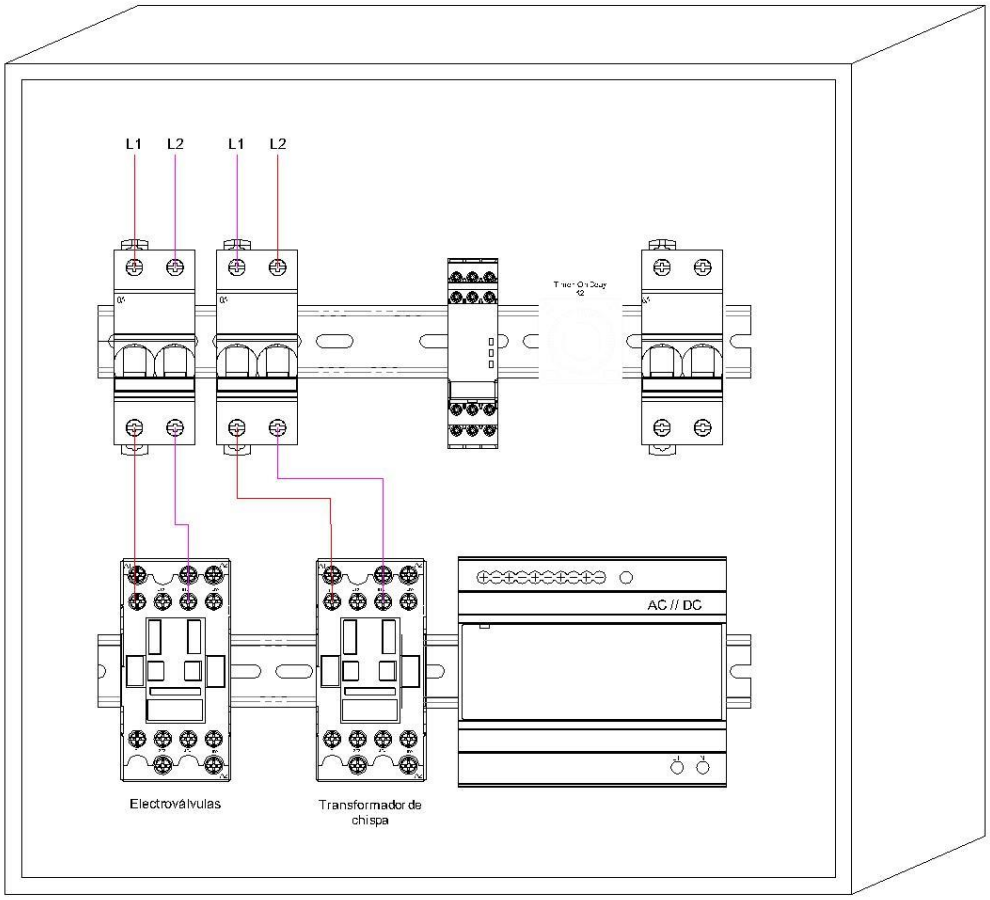
ANEXO 6

Diagrama de fuerza mediante uso de relés



ANEXO 7

Representación en 2D Interior Tablero de relés



ANEXO 8

Representación 2D de Exterior de Tablero de relés