UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TRABAJO DE FIN DE GRADO

Grado en Ingeniería Electrónica Industrial y Automática

MONITORIZACIÓN Y CONTROL REMOTO DE UNA PLANTA DE PRODUCCIÓN Y ENVASADO DE LÍQUIDOS

Memoria – Presupuesto

Autor: José Pablo Siles Gómez
Director: Javier Francisco Gámiz Caro

Convocatoria: Junio 2017

Resumen

A día de hoy es casi impensable concebir un sistema automático sin un medio de control y supervisión. Es por ello que los sistemas SCADA (Supervisión, Control y Adquisición de Datos) están a la orden del día en el sector de la automática, permitiendo así aumentar la producción, minimizar los tiempos de producción, ahorrar costes y aumentar la seguridad.

Este proyecto es un claro ejemplo de lo citado en el párrafo anterior, pues consiste en la implementación de un sistema SCADA en una planta de producción y envasado de líquidos.

Para ello, se emplea íntegramente tecnología del fabricante Siemens. Empleando un controlador lógico programable (PLC) S7-1500 por lo que respecta al hardware y en lo referente al software, se utiliza TIA Portal v14 para programar dicho autómata y SIMATIC WinCC v7.4 para el desarrollo de la interfaz gráfica del sistema SCADA.

Por lo que respecta al comportamiento del proceso de la planta, éste es completamente simulado mediante código embebido en el autómata, comunicándose éste último con el servidor de SCADA mediante el protocolo de comunicaciones PROFINET.

Para concluir, el sistema SCADA desarrollado cumple con los principales objetivos de un sistema de esta índole, siendo éstos en la gran mayoría de los casos, monitorizar y controlar los procesos y elementos que incluye la planta, permitir visualizar y exportar datos de interés tales como estados de alarma, gráficas de tendencia y registros de operaciones, y por último, implementar una gestión de usuarios que regule los diferentes permisos que se deseen implementar.

Resum

A dia d'avui és gairebé impensable concebre un sistema automàtic sense un mitjà de control i supervisió.

És per això que els sistemes SCADA (Supervisió, Control i Adquisició de Dades) són a l'ordre del dia en el sector de l'automàtica, permetent així augmentar la producció, minimitzar els temps de producció, estalviar costos i augmentar la seguretat.

Aquest projecte és un clar exemple del que s'ha esmentat en el paràgraf anterior, ja que consisteix en la implementació d'un sistema SCADA en una planta de producció i envasat de líquids.

Per a això, s'empra íntegrament tecnologia del fabricant Siemens. Emprant un controlador lògic programable (PLC) S7-1500 pel que fa al hardware i en quant al software, s'empra TIA Portal v14 per programar el ja esmentat autòmat i SIMATIC WinCC v7.4 pel desenvolupament de la interfície gràfica del sistema SCADA.

Pel que fa al comportament del procés de la planta, aquest és completament simulat mitjançant codi embegut en l'autòmat, comunicant aquest últim amb el servidor de SCADA mitjançant el protocol de comunicacions PROFINET.

Per concloure, el sistema SCADA desenvolupat compleix amb els principals objectius d'un sistema del mateix caire, sent aquests en la gran majoria dels casos, monitoritzar i controlar els processos i elements que inclou la planta, permetre visualitzar i exportar dades d'interès com ara estats d'alarma, gràfiques de tendència i registres d'operacions, i per últim, implementar una gestió d'usuaris que reguli els diferents permisos que es vulguin implementar.

Abstract

Nowadays, it is almost impossible to imagine an automatic system without any means of control and supervision. That is the reason why SCADA (Supervision, Control and Data Acquisition) systems are very usual in the automation sector, allowing to increase production, minimize production times, save costs and increase security.

This project is a clear example of what was mentioned in the previous paragraph, since it consists of the implementation of a SCADA system in a liquid production and packaging plant.

In order to achieve that purpose, Siemens technology is used for entire. Using a programmable logic controller (PLC) S7-1500, hardware wise. And concerning to software, TIA Portal v14 is used to program the PLC, and SIMATIC WinCC v7.4 is used for the development of the SCADA graphical user interface.

Regarding the behaviour of the process, it is completely simulated with embedded code in the PLC, being the latter communicated with the SCADA server through PROFINET communications protocol.

To conclude, the developed SCADA system fulfils the main objectives of these kind of systems, such as monitoring and controlling the process and the elements included in the plant, allowing the visualization and the exportation of interesting data such as alarm states, trend graphs and operations logs, and finally, implementing a user management function to regulate the different permissions that are desired to be implemented.

Agradecimientos

Quisiera dar mi especial agradecimiento a mis compañeros de Elecnor, por su gran ayuda brindada, tanto en lo referente al suministro del material necesario para la realización del proyecto como en lo referente al tiempo que me han dedicado para resolverme algunas dudas.

Glosario

- **SCADA**: Supervisión Control y Adquisición de Datos.
- **HMI**: *Human-Machine Interface* (Interfaz Hombre-Máquina).
- **PLC**: Controlador Lógico Programable.
- **DB:** Bloque de datos, permite almacenar datos con una dirección determinada.
- **FC**: Bloque de función sin DB asociado, no permite almacenar datos.
- **FB**: Bloque de función asociado a uno o varios DB's, permitiendo así almacenar datos.
- **UDT**: *User-Defined Type* (tipos definidos por el usuario), permite definir en TIA Portal una estructura/patrón de datos compuesta por distintos tipos de datos, pudiéndose crear múltiples instancias de dicho tipo.
- **TAG**: Etiqueta con la que se identifica a una variable.
- **SV**: Variable de estructura, permite crear en WinCC un patrón de diferentes tipos de datos, al cual se le pueden asignar múltiples instancias.
- **P&ID**: *Piping and instrumentation diagram* (Diagrama de tuberías e instrumentación).
- **TE**: Elemento de temperatura.
- **PE**: Elemento de presión.
- **FE**: Elemento de flujo.
- **LE**: Elemento de nivel.
- **BMB**: Bomba de impulsión.
- **VLV**: Válvula de paso.
- **TC**: Transmisor de caudal.
- **TP**: Transmisor de presión.
- **CLF**: Calefactor (Intercambiador de calor).
- **MC**: Motor de la cinta transportadora.
- **EXT**: Ventilador extractor.
- **MEZC**: Puede referirse a la etapa de mezclado o al equipo mezclador.
- **EMB**: Etapa de embotellado.
- **EVAP**: Etapa de evaporación.
- **DEP:** Depósito.
- **GRAFCET:** *Graphe Fonctionnel de Commande Etape Transition* (Grafo funcional de control etapa-transición).

Índice

1. Prefacio

1.1. Objeto del trabajo

El objeto del Trabajo de Fin de Grado (TFG) es acreditar de forma global la formación adquirida durante los estudios, que justifique la obtención del título de Ingeniero Técnico Industrial en la especialidad de Electrónica Industrial y Automática.

Se trata del desarrollo de un trabajo personal, de profundización y de síntesis dentro del ámbito de conocimiento de los estudios realizados.

A fin de demostrar la capacitación para ejercer como ingeniero en el sector de la automatización, se desarrolla un sistema SCADA realista, con tecnologías punteras en la industria y teniendo en consideración todos los aspectos funcionales que un cliente podría exigir, ofreciendo así, un producto sólido y fiable.

1.2. Motivación

La principal motivación para la realización de este proyecto han sido las ganas de profundizar un poco más en el mundo de la automática, programación de PLC's, desarrollo de sistemas SCADA y comunicaciones industriales.

Debido a la falta de una asignatura (en el plan de estudios viejo) que integrase estos tres últimos conceptos, decidí que no quería acabar el grado sin verlos de una manera global y conjunta, y debido a la imposibilidad de cursar la nueva asignatura de Integración de Sistemas Automáticos (ISA), tomé la decisión de embarcarme en este proyecto, con el fin de asimilar los conocimientos necesarios para enfocar mi perfil al del ingeniero en la especialidad de la automática.

2. Introducción

2.1. Objetivos del trabajo

El objetivo principal de este proyecto es dar respuesta a diversas necesidades que se plantean en la industria a la hora de automatizar un proceso, tanto a nivel de campo, control y supervisión. Siendo los dos últimos puntos los más resaltados en cuanto a desarrollo.

Es por ello que los objetivos para la implementación del sistema SCADA de la planta de producción y envasado de líquidos se agrupan según estos niveles:

- Nivel de campo: Compuesto por la instrumentación de campo y los elementos de proceso
    - Definición de los dispositivos necesarios para el proceso a automatizar.
    - Simulación de la lógica del comportamiento de los dispositivos de campo mediante código embebido en el PLC.
- Nivel de control: Compuesto por el PLC Siemens SIMATIC S7-1500
    - Implementación de un proyecto con TIA Portal v14 para la configurar el hardware y las conexiones del PLC.
    - Programación de la lógica de control de los elementos.
    - Programación de la lógica de control de las secuencias de proceso de la planta.
- Nivel de supervisión: Implementada mediante el software WinCC v7.4 de Siemens
    - Implementación de un proyecto mono-puesto para la aplicación SCADA.
    - Configuración de los drivers de comunicación con el PLC.
    - Creación gráfica y funcional del sistema SCADA.

2.2. Alcance del trabajo

Para concebir este proyecto, se hace un estudio de los diversos procesos con los que se va a dotar a la planta de producción y envasado de líquidos, siendo éstos, proceso de evaporación, proceso de mezclado y proceso de embotellado.

Asimismo, se escogen los dispositivos necesarios para poder llevar a cabo dichos procesos de la manera más optimizada.

Con los dispositivos y elementos ya definidos, se procede a programar su lógica con el fin de simular un comportamiento de los mismos lo más cercano posible a la realidad.

Para cumplir con los objetivos de producción de la planta será necesario implementar acciones de control sobre los distintos procesos de la misma, por lo que se empleará un autómata (PLC) encargado de controlar las diversas etapas.

También se determinarán y configurarán los protocolos de comunicaciones para establecer la red mediante la cual se comunicarán el autómata y el servidor del sistema SCADA.

Por lo que respecta a la aplicación de SCADA en sí, el presente proyecto propone crear una aplicación 100% personalizada, por lo que, aun siguiendo el estándar de la normativa ISA 101, el diseño gráfico de la aplicación se realizará teniendo en mente cumplir dicho propósito. Aplicándose esta filosofía al diseño de iconos, mandos de control, pantallas de proceso, pantallas de gestión, etc.

En cuanto al diseño funcional del SCADA, mediante un software robusto como es SIMATIC WinCC, se desarrolla una navegación, control y supervisión ágiles, con el fin de ejemplificar en un entorno gráfico el potencial que ofrece un sistema SCADA.

3. Análisis del problema

3.1. Descripción del proceso a automatizar

La planta a automatizar en este proyecto está diseñada para producir y envasar un determinado líquido constituido por una mezcla de productos, para ello se distinguen tres etapas:

1. Etapa de evaporación.
2. Etapa de mezclado.
3. Etapa de embotellado.

A continuación se adjunta el diagrama de tuberías e instrumentación (P&ID) de la planta, en el cual se pueden observar los diferentes elementos que constituyen la totalidad de la planta, así como la distribución de éstos para conformar las diferentes etapas.

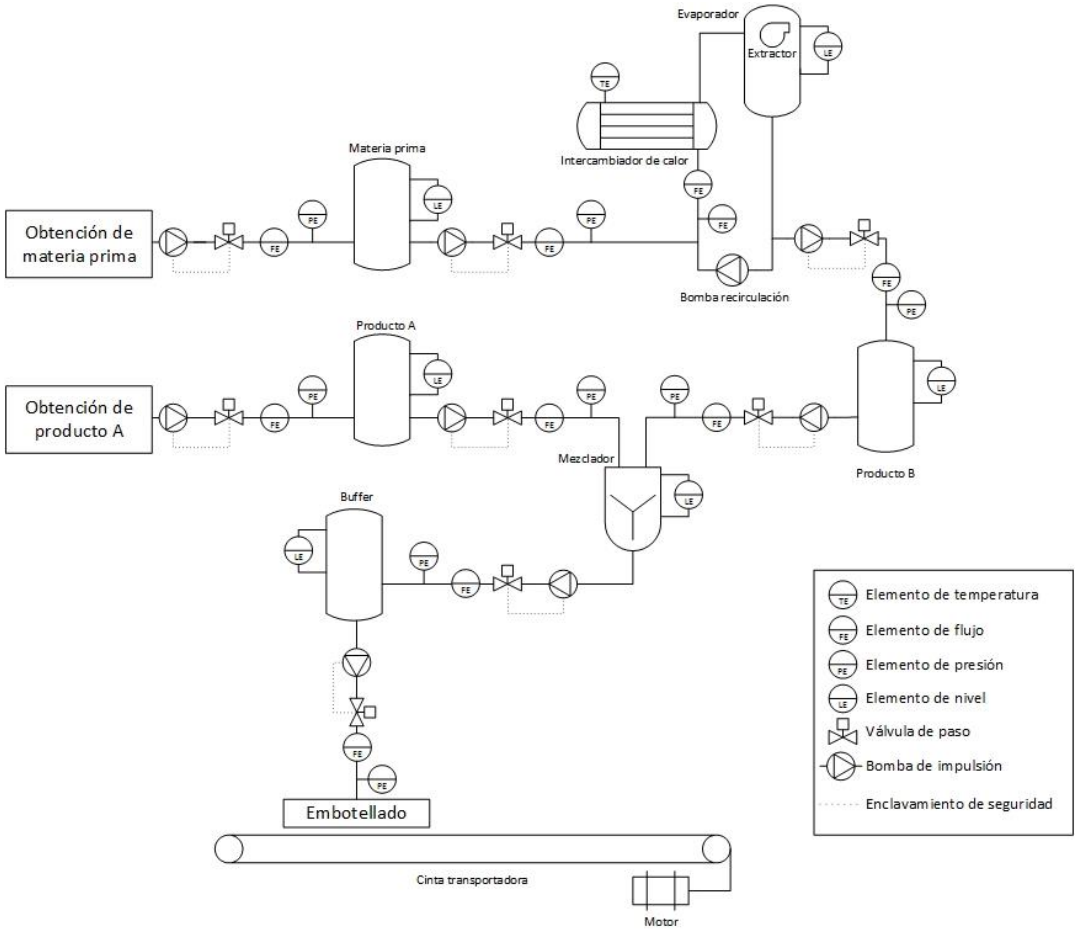


Figura 1. Diagrama de tuberías e instrumentación (P&ID) de la planta.

Cabe mencionar que las tres etapas son independientes, por lo que cada una de ellas puede estar produciendo simultáneamente siempre que se disponga de la materia necesaria para ello.

3.1.1. Etapa de evaporación

Esta etapa consiste en un proceso *batch* o por lotes, esta clase de procesos se caracterizan por realizar la carga de la materia al comienzo de la operación y pasado cierto tiempo, pasar a realizar la descarga de la misma. Ciñéndose a la definición estrictamente académica, en un proceso por lotes no hay transferencia de materia a través de las fronteras del sistema.

La función de esta etapa consiste en tratar una materia prima líquida de tal manera que se eliminen los residuos de otras materias no deseadas en el producto.

Para ello se recurre a calentar la mezcla de materia prima y materia no deseada hasta la temperatura de ebullición de la materia no deseada (siendo ésta la temperatura de ebullición más baja), de tal modo que la materia no deseada entra en estado de evaporación, pasando de estado líquido a gas, dejando así un líquido puro de la materia prima para la elaboración del producto.

Dicha evaporación requiere de un sistema de circulación forzada. Este sistema consta de:

- Intercambiador de calor: Calienta la mezcla hasta el punto de ebullición de la materia indeseada.
- Bomba de recirculación: Mantiene la mezcla fluyendo por un circuito cerrado para darle tiempo a alcanzar la temperatura de ebullición.
- Cámara de ebullición: Depósito en el que se irá almacenando el líquido a medida que se vaya completando la evaporación.
- Ventilador extractor: Forzará la expulsión del gas para evitar su condensación (retorno al estado líquido).

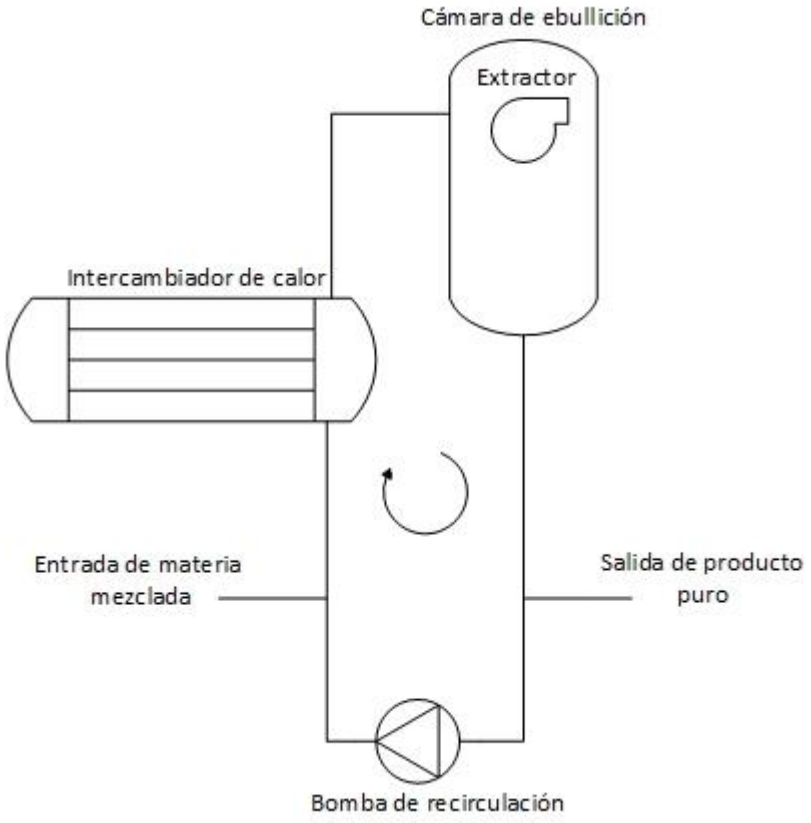


Figura 2. Diagrama de evaporador de circulación forzada.

Una vez explicados los principios teóricos de esta etapa, se puede comprender más fácilmente el proceso *batch* que lleva a cabo la misma:

1. Carga del evaporador de circulación forzada con determinado volumen de materia prima mediante una bomba de impulsión.
2. Proceso de evaporación por circulación forzada durante el tiempo que se determine.
3. Descarga del líquido puro de materia prima en el tanque de producto B mediante una bomba de impulsión.

3.1.2. Etapa de mezclado

Esta segunda etapa es del tipo *batch* de igual forma que la primera, pues consiste en:

1. Cargar cierto volumen de producto B (resultado de la etapa de evaporación) en el tanque de mezclado mediante una bomba de impulsión.
2. Cargar cierto volumen del producto A en el tanque de mezclado mediante una bomba de impulsión.

3. Homogeneizar mediante un mezclador la mezcla formada por los productos A y B durante el tiempo que se determine.
4. Descargar el producto resultante de la mezcla mediante una bomba de impulsión en un depósito *buffer* (intermediario) previo a la etapa de embotellado.

La razón por la que esta etapa se realiza como un proceso por lotes es debido a que se desea respetar una cierta proporción en la mezcla, de tal modo que cada lote constituya una mezcla con una determinada proporción de cantidad de productos A y B.

3.1.3. Etapa de embotellado

Cabe mencionar que el proceso de llenado del depósito buffer constituye una etapa intermedia entre el mezclado y el embotellado, pues se distingue por responder a un proceso continuo, esto es, consta de un flujo permanente de entrada y de salida de materia durante el tiempo que dura la etapa.

Esto es debido a que, gracias a la implementación del depósito *buffer*, la etapa de embotellado se independiza del tanque de mezclado, pudiéndose así estar llenando botellas a la vez que se está produciendo la descarga del tanque de mezclado al depósito *buffer*, siempre y cuando la proporción de la nueva mezcla coincida con la proporción de la mezcla contenida en el depósito *buffer*, impidiendo así mezclar lotes de distintas proporciones de mezclado.

Por lo que respecta a la etapa de embotellado en sí, ésta se trata nuevamente de un proceso *batch*, implicando la siguiente secuencia de acciones:

1. Transportar botellas vacías por la cinta transportadora al punto de llenado.
2. Llenar la botella con el volumen de líquido que se determine.
3. Etiquetar y empaquetar la botella llena (realizado por operarios).

3.2. Sistema de control

3.2.1. Equipos

En un sistema de control es imprescindible realizar una codificación de los elementos que lo integran, con ello se permite realizar de forma clara y sencilla un inventariado físico de los mismos, así como recopilar de forma ordenada información útil como su localización y facilitar la realización de las diversas tareas de mantenimiento y control de vida de las existencias.

En el presente proyecto se implementa una codificación significativa con la que se determina a qué etapa pertenece el equipo, qué tipo de elemento es y su numeración de inventariado.

Para ello se emplea la siguiente fórmula:

$$Etapa\_Elemento\_XX \qquad \textbf{(Expresión.1)}$$

Junto con los siguientes indicadores:

Tabla 1. Leyenda de los distintos indicadores empleados.

ETAPAS	
EVAP	Etapa de evaporación
MEZC	Etapa de mezclado
EMB	Etapa de embotellado
EQUIPOS	
BMB	Bomba de impulsión
VLV	Válvula de paso
TC	Transmisor de caudal
TP	Transmisor de presión
DEP	Depósito
MC	Motor
CLF	Calefactor
EXT	Extractor
MEZC	Mezclado

A continuación se expone un listado de los diversos equipos integrados en la planta y su respectiva codificación:

Tabla 2. Listado de codificación de equipos.

CODIFICACIÓN	DESCRIPCIÓN EQUIPO
MEZC_MEZC_01	Mezclador de producto
EVAP_CLF_01	Intercambiador de calor/Calefactor
EMB_MC_01	Motor cinta transportadora
EVAP_EXT_01	Ventilador extractor
EVAP_VLV_00	Válvula de carga materia prima 1
EVAP_VLV_01	Válvula de carga evaporador
EVAP_VLV_03	Válvula de carga depósito producto post-evaporación
MEZC_VLV_04	Válvula de carga producto post-evaporador al tanque de mezclado
MEZC_VLV_05	Válvula de carga materia prima 2 al tanque de mezclado
MEZC_VLV_06	Válvula de carga depósito *buffer*

CODIFICACIÓN	DESCRIPCIÓN EQUIPO
MEZC_VLV_07	Válvula de carga depósito materia prima 2
EMB_VLV_08	Válvula de embotellado
EVAP_TP_00	Sonda de presión carga materia prima 1
EVAP_TP_01	Sonda de presión carga evaporador
EVAP_TP_02	Sonda de presión circuito de recirculación del evaporador
EVAP_TP_03	Sonda de presión carga depósito post-evaporación
MEZC_TP_04	Sonda de presión carga post-evaporación al tanque de mezclado
MEZC_TP_05	Sonda de presión carga materia prima 2 al tanque de mezclado
MEZC_TP_06	Sonda de presión carga depósito buffer
MEZC_TP_07	Sonda de presión carga depósito materia prima 2
EMB_TP_08	Sonda de presión línea de embotellado
EVAP_TC_00	Caudalímetro carga materia prima 1
EVAP_TC_01	Caudalímetro carga evaporador
EVAP_TC_02	Caudalímetro circuito de recirculación del evaporador
EVAP_TC_03	Caudalímetro carga depósito post-evaporación
MEZC_TC_04	Caudalímetro carga post-evaporación al tanque de mezclado
MEZC_TC_05	Caudalímetro carga materia prima 2 al tanque de mezclado
MEZC_TC_06	Caudalímetro carga depósito buffer
MEZC_TC_07	Caudalímetro carga depósito materia prima 2
EMB_TC_08	Caudalímetro línea de embotellado
EVAP_BMB_00	Bomba de impulsión carga materia prima 1
EVAP_BMB_01	Bomba de impulsión carga evaporador
EVAP_BMB_02	Bomba de impulsión circuito de recirculación del evaporador
EVAP_BMB_03	Bomba de impulsión carga depósito post-evaporación
MEZC_BMB_04	Bomba de impulsión carga post-evaporación al tanque de mezclado
MEZC_BMB_05	Bomba de impulsión carga materia prima 2 al tanque de mezclado
MEZC_BMB_06	Bomba de impulsión carga depósito buffer
MEZC_BMB_07	Bomba de impulsión carga depósito materia prima 2
EMB_BMB_08	Bomba de impulsión línea de embotellado
EVAP_DEP_00	Depósito de materia prima 1
EVAP_DEP_01	Evaporador (cámara de ebullición)
EvapMezc_DEP_02	Depósito producto post-evaporado (Producto A)
MEZC_DEP_03	Depósito materia prima 2 (Producto B)
MEZC_DEP_04	Tanque de mezclado (Producto final)
MezcEmb_DEP_05	Depósito buffer

3.2.2. Fases del proceso

Para comprender la naturaleza del funcionamiento de las diversas fases, así como su interacción con su entorno y cuáles son sus estados, se recurre a la elaboración de diagramas GRAFCET de primer nivel, esto es, empleando un lenguaje natural que permita una descripción funcional del proceso.

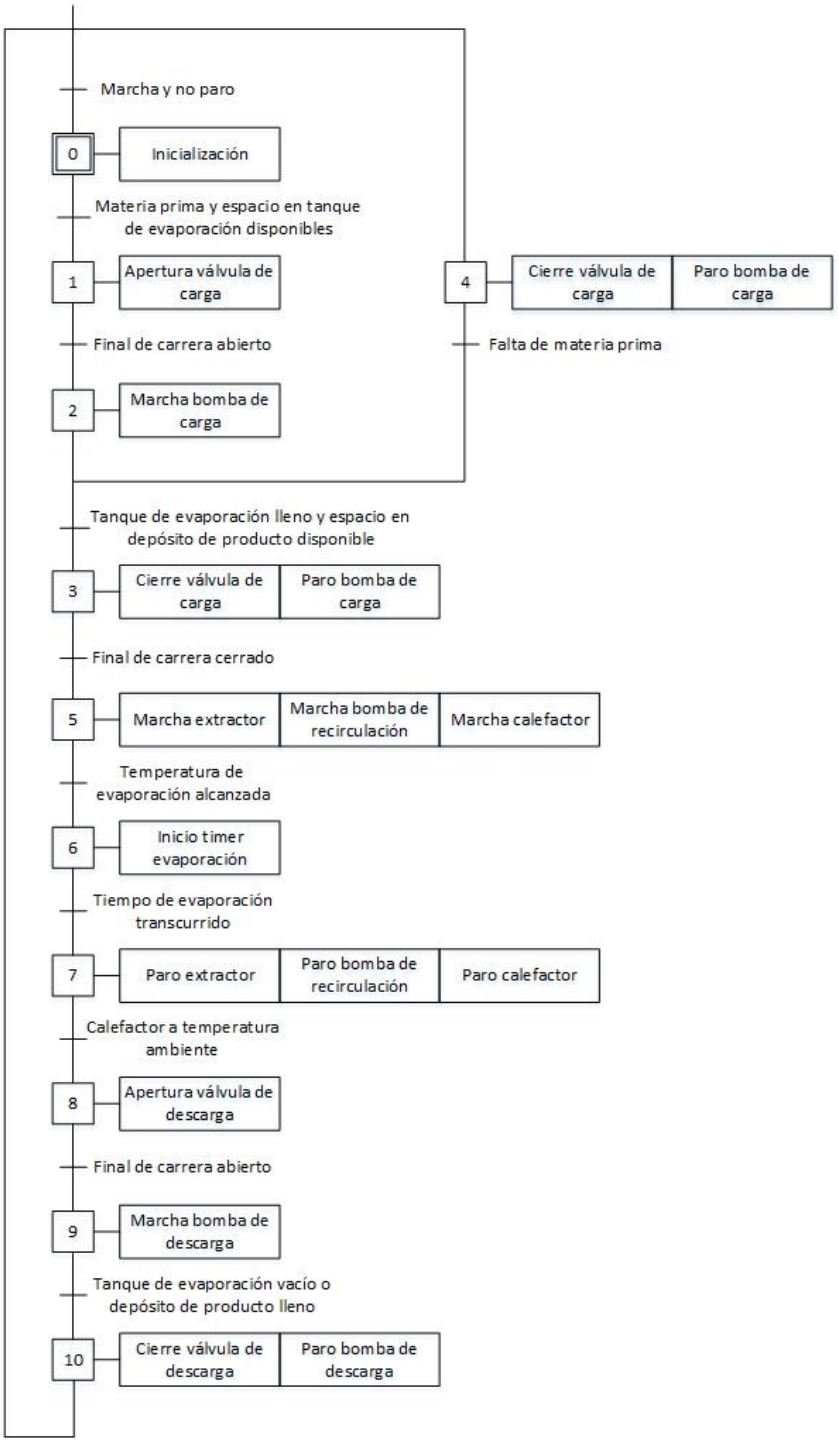


Figura 3. Diagrama GRAFCET del proceso automático de evaporación.

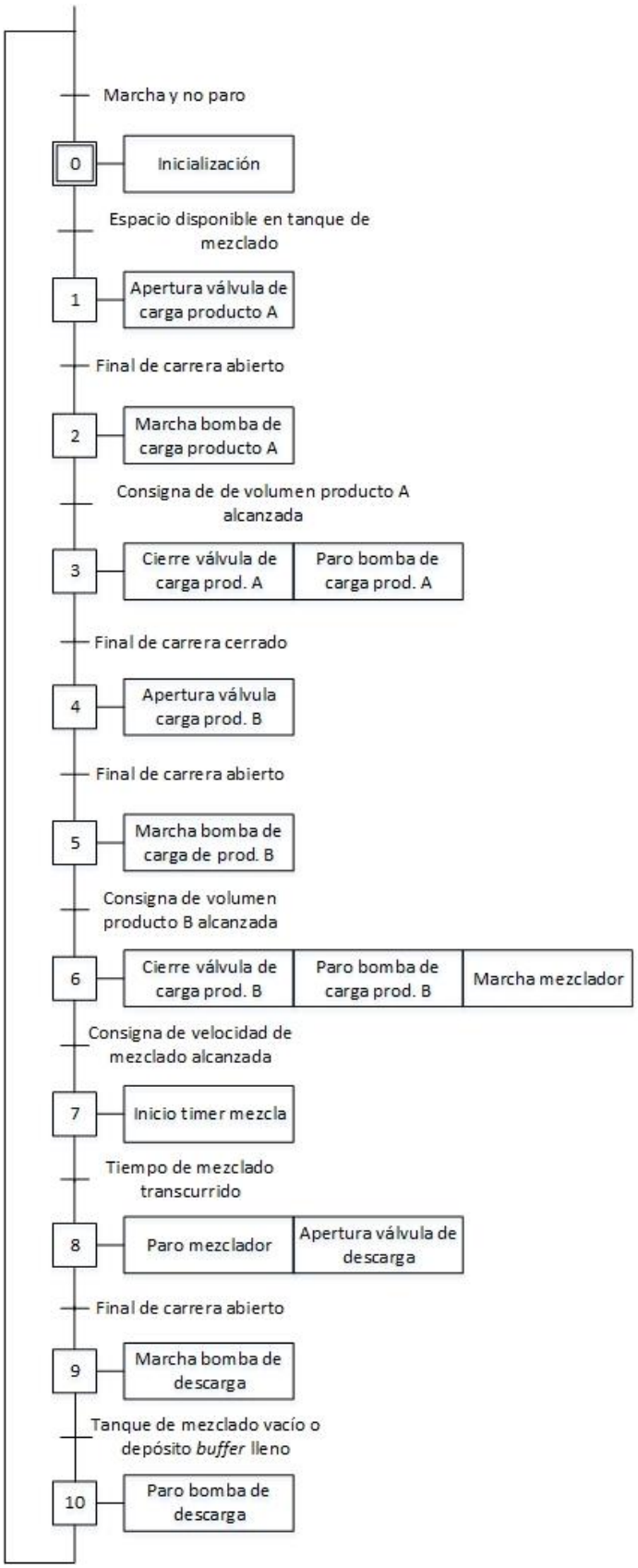


Figura 4. Diagrama GRAFCET del proceso automático de mezclado.

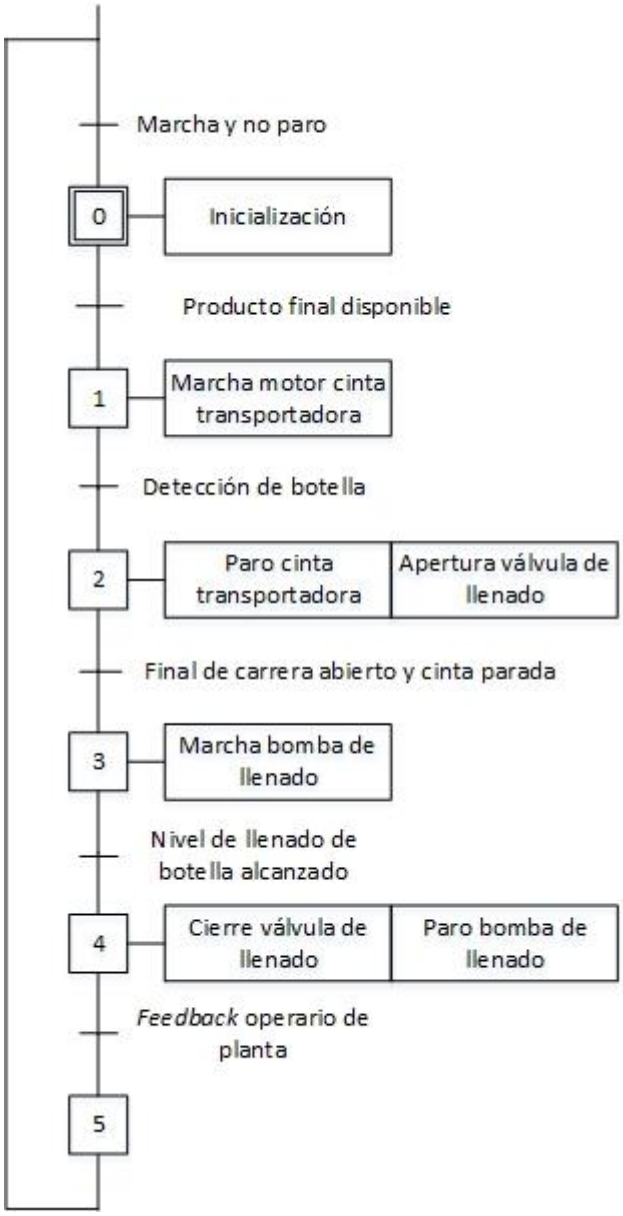


Figura 5. Diagrama GRAFCET del proceso automático de embotellado.

3.3. Requisitos funcionales

1. El sistema registrará cualquier cambio de un valor de consigna de proceso así como el usuario que lo realice y la fecha y hora del cambio.
2. El sistema registrará todos los valores de consigna así como los valores de proceso.
3. El sistema permitirá a cualquier usuario graficar el valor registrado de interés.
4. El sistema registrará todos los avisos de alarma, indicando su procedencia, su estado, la fecha y hora de aparición y su prioridad.
5. El sistema permitirá exportar a formato CSV los datos registrados.

6. El sistema permitirá filtrar los datos registrados según fecha, hora, prioridad, tipo, etc.
7. El sistema contará con tres niveles de prioridad para los avisos de alarma, siendo el nivel 0 para averías graves y fallos de comunicación, el nivel 1 para alarmas de valores críticos y el nivel 2 para advertencias de valores de riesgo y estados de discrepancia.
8. El sistema contará con una advertencia de alarma general de cada etapa que indicará si alguno de los elementos que formen parte de dicha etapa está en estado de fallo.
9. El sistema distinguirá las alarmas reconocidas de las no reconocidas dotando a estas últimas de un parpadeo, en el momento de su reconocimiento, quedarán fijas mientras siga la alarma activa.
10. El sistema controlará el acceso y lo permitirá solamente a usuarios autorizados.
11. El sistema contará con tres tipos de usuarios: administradores, supervisores y operadores.
12. Los usuarios del grupo de administradores podrán navegar, cerrar la aplicación, realizar cualquier acción de control, modificar consignas de proceso, reconocer alarmas, así como hacer cualquier gestión en la configuración de usuarios.
13. Los usuarios del grupo de supervisores podrán navegar, reconocer alarmas y realizar acciones de control.
14. Los usuarios del grupo de operadores tan solo podrán navegar y reconocer alarmas.
15. El sistema desconectará a los usuarios tras un tiempo determinado de inactividad: uno, dos y tres minutos para administradores, supervisores y operadores, respectivamente.
16. El sistema mostrará en todo momento la fecha y hora actuales así como el usuario que esté conectado.
17. El sistema permitirá cambiar de usuario sin necesidad de desconectar.
18. El sistema contará con una barra de navegación disponible en todo momento.
19. El sistema constará de una pantalla principal a modo de resumen del estado actual de los diversos procesos, las alarmas activas y las consignas marcadas.
20. El sistema permitirá cambiar a modo manual cualquier elemento sin necesidad de detener la secuencia.
21. En caso de que un elemento que forme parte de una secuencia entre en fallo, se indicará la imposibilidad de continuar con el subproceso del que forme parte.
22. En el caso de dar la orden de parar la secuencia, todos los elementos deberán detenerse.
23. Cualquier elemento en estado de fallo, pasará a su estado de seguridad.
24. El sistema representará en la etapa y sub-etapa en la que se encuentre el proceso.
25. En la secuencia automática de la etapa de evaporación, no se procederá a evaporar si el depósito de producto post-evaporación ya está lleno.
26. En la secuencia automática de la etapa de mezcla, no se podrá cambiar el valor de consigna de mezclado si hay un lote elaborándose.

27. El sistema permitirá modificar la consigna de tiempo de evaporación y mezclado en cualquier momento.
28. El sistema tendrá un control de producción al contabilizar las botellas producidas.
29. El sistema contará con indicadores de falta de materia.
30. El sistema contará con una doble confirmación para la activación y desactivación de las secuencias automáticas.
31. El sistema tendrá un contador reseteable de horas de funcionamiento para cada equipo, con el fin de controlar los tiempos de revisión y mantenimiento.
32. Se implementará un sistema de enclavamiento bomba-válvula para evitar que las primeras operen en vacío.
33. Se implementará un sistema de bloqueo (posición de cierre) en las válvulas para evitar que los depósitos rebosen.
34. Se implementará un sistema de bloqueo en el mezclador con el fin de impedir que se ponga en marcha si el tanque de mezcla está vacío.
35. El sistema contará con iconos según los estándares de la normativa ISA 101, indicando de forma clara la información principal que deban transmitir.
36. Cada icono mostrará una versión acortada del nombre del elemento.
37. Cada elemento contará con un mando de operaciones que ampliará la información dada por el icono.
38. El mando de operaciones mostrará el nombre completo del elemento así como una breve descripción.
39. El mando de operaciones permitirá deshabilitar, habilitar y reconocer las alarmas de su respectivo elemento, mostrar su tiempo de funcionamiento y permitir resetearlo, dar las respectivas órdenes de marcha, apertura, paro o cierre, así como mostrar los valores de proceso y de consignas de velocidad, temperatura, nivel de apertura, niveles y tiempos de tolerancia.
40. Cada elemento dispondrá de un mando de simulación con el que se podrán introducir valores de forzado y parámetros para su modelización.

3.4. Requisitos de diseño

En este apartado se procede a exponer los requisitos no funcionales, es decir, requisitos que puedan acarrear alguna restricción sobre el espacio de posibles soluciones.

Hardware:

- El control debe ser realizado por un autómata S7-1500 del fabricante Siemens.
- Se requiere de un ordenador de alto rendimiento para ejecutar la aplicación.

- El sistema SCADA debe ser del tipo estación monopuesto.

Software:

- La programación del autómata se debe realizar mediante TIA Portal v14.
- La aplicación SCADA se debe desarrollar mediante SIMATIC WinCC v7.4
- Tanto TIA Portal como WinCC se deben ejecutar en máquinas virtuales de VMWare, debido a la imposibilidad de adquirir licencias.

Rendimiento:

- El sistema debe soportar el manejo de gran cantidad de información durante el proceso.
- El tiempo de respuesta del sistema está sujeto a la velocidad de comunicación PLC-SCADA así como a la velocidad de comunicación PC-Máquina virtual.

Seguridad:

- El ingreso a la aplicación estará restringido a usuarios dados de alta.

Usabilidad:

- La aplicación debe ofrecer una alta calidad en la experiencia a los usuarios que interactúen con ella.

Interoperabilidad:

- El sistema debe intercambiar información entre el servidor de la aplicación SCADA y el autómata.

3.5. Metodología de ejecución

En procesos de la índole de la automática es muy importante estipular de forma clara la metodología que se deberá seguir para la realización del proyecto, pues la mayoría de sus partes están relacionadas entre sí de forma directa.

Es por ello que la elaboración del proyecto se divide, según modelo en cascada, en las siguientes fases:

1. Análisis previo: En esta fase se define el objeto del proyecto. Se describe el proceso y se esquematizan sus partes. Todo ello sirve para definir unos baremos del alcance del proyecto y determinar su viabilidad.

2. Ingeniería de detalle: En esta fase se escogen los dispositivos que será necesarios para cumplir con los requisitos operacionales del proceso, así como su instalación.
3. Programación del PLC: En esta fase se selecciona el lenguaje más adecuado para cada fase del proceso, se describe de forma detallada las funciones que se automatizan y se realiza un listado de las variables necesarias. En función de todo ello se escoge el autómata a emplear.
4. Comunicación con SCADA: En esta fase se definen las necesidades de comunicación y su interfaz requerida, así como las necesidades de supervisión para proceder a escoger el software a emplear.

3.6. Recursos empleados

Para la realización del proyecto se han necesitado diversos recursos tecnológicos tanto a nivel de hardware como de software.

Los diferentes recursos empleados se listan a continuación y se profundiza más en sus características y funcionalidades en los apartados *4.1.1. Hardware del sistema* y *4.1.2. Software del sistema*.

Recursos del hardware:

- Ordenador de alto rendimiento.
- Autómata SIMATIC S7-1500.
- Fuente de alimentación SIMATIC PM 1507.

Recursos del software:

- Software de virtualización: VMWare Workstation 12.
- Software de programación y configuración PLC: TIA Portal v14.
- Software de desarrollo de la aplicación SCADA: SIMATIC WinCC v7.4.

3.7. Planificación de las tareas

Para ilustrar la planificación de las tareas a realizar, se adjunta un diagrama de Gantt, con el cual se enmarcan las diversas tareas que componen el proyecto en su intervalo de tiempo de realización, determinando la duración en días dedicadas a su realización.

TAREA	INICIO	FIN	DÍAS
Esquematización de la planta (P&ID)	2-feb.	3-feb.	2
Descripción de características de la planta	4-feb.	4-feb.	1
Definición de requisitos funcionales de la planta	5-feb.	5-feb.	1
Listado de elementos y equipos necesarios	6-feb.	6-feb.	1
Codificación de equipos	7-feb.	7-feb.	1
Diagramas GRAFCET de los procesos a automatizar	8-feb.	10-feb.	3
Definición de las variables de control	11-feb.	15-feb.	5
Instalación del software necesario	16-feb.	16-feb.	1
Configuración del hardware del programa	17-feb.	17-feb.	1
Configuración comunicaciones TIA Portal - PLC	18-feb.	18-feb.	1
Configuración comunicaciones PLC - WinCC	19-feb.	19-feb.	1
Programación de bloques de herramienta	20-feb.	24-feb.	5
Programación bloques simulación/control elementos	25-feb.	21-mar.	25
Definición de los tags necesarios	22-mar.	23-mar.	2
Asignación de dirección a los tags	24-mar.	24-mar.	1
Diseño de los iconos de equipos en WinCC	25-mar.	3-abr.	10
Diseño mandos de control y simulación equipos	4-abr.	13-abr.	10
Programación de las secuencias automáticas	14-abr.	3-may.	20
Diseño de las pantallas de proceso	4-may.	8-may.	5
Programación de scripts de navegación	9-may.	13-may.	5
Diseño funcional y gráfico del sistema de navegación	14-may.	15-may.	2
Definición de avisos de alarma	16-may.	17-may.	2
Definición de variables a historizar	18-may.	18-may.	1
Definición de grupos de ususarios y niveles de permisos	19-may.	19-may.	1
Diseño de las pantallas de gestión	20-may.	22-may.	3

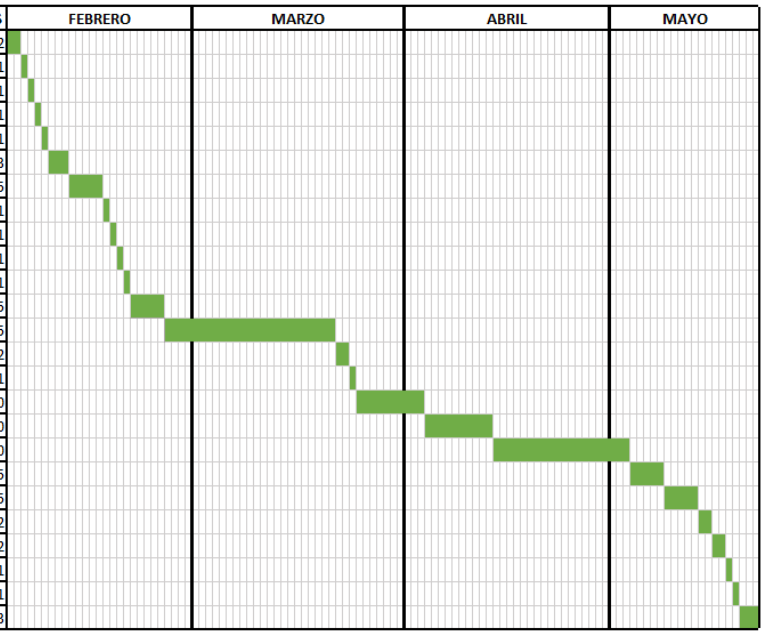


Figura 6. Diagrama de Gantt de la planificación de tareas del proyecto.

4. Diseño e implementación de la solución

4.1. Arquitectura del sistema de control

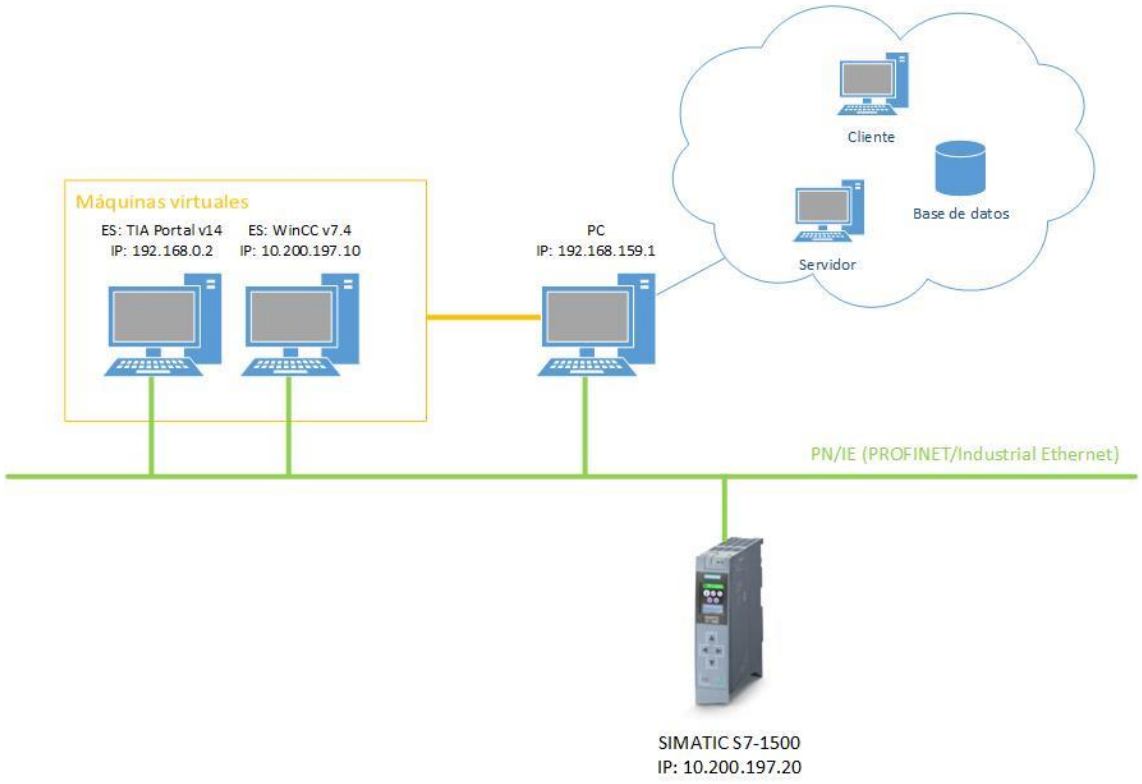


Figura 7. Arquitectura de red del sistema de control

Como se puede apreciar en la anterior vista global de las comunicaciones, el sistema carece de conexiones a campo y consta solamente del autómata, que se comunica mediante un bus de datos PROFINET Industrial Ethernet al PC y a dos estaciones de ingeniería (ES) en máquinas virtuales cuyo host es el mismo PC.

Las estaciones de ingeniería son los equipos del sistema desde los cuales se pueden realizar acciones de desarrollo, en el caso de este proyecto, se dispone de una estación de ingeniería para TIA Portal y otra para WinCC.

A su vez, el PC asume el rol de lo que, en un sistema de control real, serían diversos equipos:

- Servidor: Proveedor de los recursos y los servicios del sistema.
- Cliente: Demandante de dichos recursos y servicios.
- Base de datos: Equipo en el que se almacenan sistemáticamente datos para su posterior uso.

4.1.1. Hardware del sistema

4.1.1.1. PLC SIMATIC S7-1500

Referencia: 6ES7 511-1AK01-0AB0

Es el autómata escogido para llevar a cabo el control del proyecto, los controladores SIMATIC S7-1500 se caracterizan por integrar un *display* acoplable y desacoplable para puesta en marcha y realizar funciones de diagnóstico. Además, cuentan con una interfaz PROFINET, lo que asegura buenos tiempos de respuesta y alta precisión en su comportamiento.

Por lo que respecta a la CPU, el autómata cuenta con una CPU estándar (no dispone de puertos de entrada/salida integradas) 1511-1 PN cuyas características destacables son:

- Memoria de trabajo: 1,23 Mbytes.
- Tiempo de ejecución de operaciones con bits: 60 ns.
- Interfaces: dos puertos PROFINET, utilizables como puertos en anillo para topologías en anillo redundantes en Ethernet.
- OPC UA: La CPU actúa como servidor OPC UA y puede comunicarse con clientes OPC UA.
- Servidor web integrado.
- Lenguajes de programación: KOP (diagrama de contactos), FUP (diagrama de funciones), AWL (lista de instrucciones, lenguaje máquina), S7-SCL (lenguaje estructurado), S7-GRAPH (lenguaje en formato GRAFCET).

Figura 8. PLC SIMATIC S7-1500 1511-1PN.

4.1.1.2. Fuente de alimentación SIMATIC PM 1507

Referencia: 6EP1332-4BA00

Es la fuente de tensión encargada de alimentar al autómata, es exclusiva para S7-1500 y ET200MP.

Es una fuente de alimentación estabilizada con una entrada de corriente alterna de 120/130 V y proporciona una tensión 24 V DC a 3 A.

Asimismo, está diseñada para operar a una temperatura de entre -40º C y 70º C.

Figura 9. Fuente de alimentación SIMATIC PM 1507.

4.1.2. Software del sistema

4.1.2.1. VMWare Workstation Player

Para poder tener acceso a los diversos softwares de Siemens, se requiere o bien tener acceso a dichos productos o bien tener acceso a una máquina que cuente con ellos. Como se carece de las licencias para instalar los programas necesarios, se opta por la segunda opción. Es en este punto donde entra en juego la virtualización.

La virtualización es el proceso de crear una representación de una máquina o aplicación basada en software, en vez de en hardware.

En este proyecto, se utiliza la aplicación de virtualización VMWare Workstation Player para acceder a una máquina virtual con TIA Portal v14 y WinCC v7.4.

4.1.2.2. TIA Portal v14

TIA Portal (Totally Integrated Automation Portal), es un software de entorno ingenieril del fabricante Siemens que unifica diversos softwares para abarcar todas las tareas de control, visualización y accionamiento de las soluciones de automatización de Siemens.

TIA Portal cuenta con:

- STEP 7 Professional: Software de programación PLC sucesor de SIMATIC S5, permite la programación en FBS, KOP, AWL, S7 SCL y S7-Graph.
- WinCC Basic: Variante de SIMATIC WinCC que solo permite configurar paneles HMI (Interfaz Hombre-Máquina).
- S7-PLCSIM: Simulador integrado de PLC.

Del variado paquete de productos que ofrece TIA Portal, en el proyecto solamente se emplea STEP 7 puesto que WinCC Basic no cumple con los requisitos para realizar un sistema de SCADA más allá de paneles HMI y S7-PLCSIM se vuelve del todo innecesario al disponer de un autómata físico con el cual poder realizar las pruebas.

A la hora de programar y configurar el PLC, se ha empleado STEP 7, software orientado a la programación de bloques de funciones y de datos.

La filosofía de este software consiste en tener diversos bloques de organización interactuando entre sí.

Por un lado se tiene el bloque de organización principal (OB1), dicho bloque se ejecuta de forma cíclica cada cierto tiempo.

Además del bloque de organización cíclica, STEP 7 también dispone de otros tipos de bloques de organización que se procesan en función de ciertos eventos como, por ejemplo, al arrancar, por interrupciones cíclicas o de hardware, a cierta hora determinada, etc.

La función del bloque de organización es la de pautar la secuenciación de llamadas de las diferentes partes del programa, es por ello que desde él se instancian y llaman a otros bloques, pudiendo ser éstos de diferentes tipos:

- Bloques de función (FB): Los bloques de función son bloques lógicos que depositan sus valores permanentemente en bloques de datos de instancia, de modo que los datos siguen estando disponibles después de procesar el bloque.
- Funciones (FC): Las funciones son bloques que no disponen de memoria, por lo que no se almacenan los datos una vez el bloque ha sido procesado.
- Bloques de datos (DB): Los bloques de datos sirven para almacenar datos del programa, estos bloques pueden ser de tipo global (accesibles desde cualquier bloque) o asociados a un FB, de tal manera que solo este último bloque puede acceder a los datos.

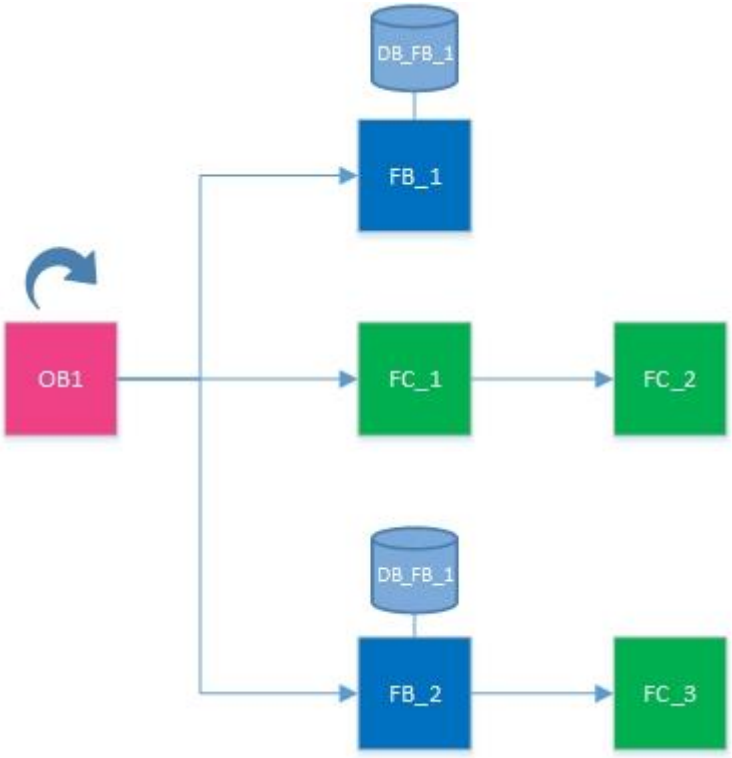


Figura 10. Ejemplo de la arquitectura de ejecución por bloques.

Por lo que respecta a los distintos lenguajes de programación que incluye STEP 7 se tiene:

Lenguaje KOP (esquema de contactos):

KOP es un lenguaje de programación gráfico, cuya representación es similar a los esquemas de circuitos.

El programa se mapea en uno o varios segmentos, conteniendo en el margen izquierdo una barra de alimentación de la que parten los circuitos.

Las consultas de las señales binarias se disponen en los circuitos en forma de contactos, de tal manera que se pueden crear distribuciones en serie o paralelo.

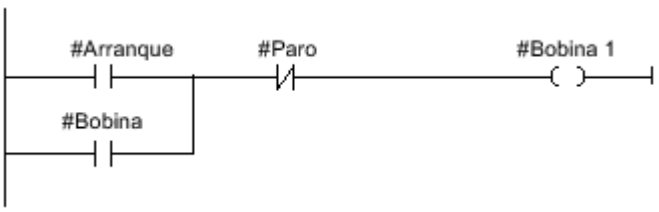


Figura 11. Ejemplo de la interfaz gráfica del lenguaje KOP (Fuente: Sistema de información TIA Portal).

Lenguaje FUP (diagrama de funciones):

FUP es un lenguaje de programación gráfico cuya representación es similar a los diagramas de bloques de circuitos electrónicos.

El programa se mapea en uno o varios segmentos, conteniendo uno o varios circuitos lógicos. Las consultas de señales binarias se combinan lógicamente mediante cuadros y para representar la lógica se emplean símbolos lógicos gráficos del álgebra booleana.

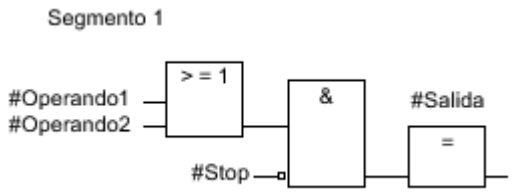


Figura 12. Ejemplo de la interfaz gráfica del lenguaje FUP (Fuente: Sistema de información TIA Portal).

Lenguaje AWL (Lista de instrucciones):

AWL es un lenguaje de programación basado en un texto con el cual programar bloques de código.

El programa se divide en segmentos, con una o varias líneas cada uno. La numeración de las líneas se ordenará en función de su orden de ejecución, empezando en cada segmento por uno.

En las líneas de los segmentos se programa las instrucciones a ejecutar, con un máximo de una instrucción por línea, representando cada instrucción una prescripción de trabajo para la CPU.

AWL	Explicación
O #S1	// Consultar si el pulsador de marcha S1 es "1"
O #S3	// Consultar si el pulsador de marcha S3 es "1"
S #MOTOR_ON	// Si uno de los pulsadores de marcha (S1 o S3) devuelve el estado lógico "1", se pone en marcha el motor de la cinta transportadora.
O #S2	// Consultar si el pulsador de paro S2 es "1"
O #S4	// Consultar si el pulsador de paro S4 es "1"
R #MOTOR_ON	// Si uno de los pulsadores de paro (S2 o S4) devuelve el estado lógico "1", se para el motor de la cinta transportadora.

Figura 13. Ejemplo de código en lenguaje AWL (Fuente: Sistema de información TIA Portal).

Lenguaje SCL (lenguaje de control estructurado):

SCL es un lenguaje de programación de alto nivel que se orienta a PASCAL, se basa en la norma DIN EN-61131-3, la cual estandariza los lenguajes de programación para PLC. Además, también cumple con el *PLCopen Basis Level* del lenguaje estructurado definido en dicha norma.

SCL cuenta con los elementos del lenguaje típicos del PLC (entradas, salidas, temporizadores, marcas) además de elementos de alto nivel como expresiones, asignaciones de valor y operadores.

En cuanto al control del programa, SCL ofrece instrucciones prácticas para el control del programa que permiten, entre otras cosas, realizar ramas, bucles o saltos del programa.

```
IF "Marcha_izquierda_S1" OR "Marcha_derecha_S3" THEN
"MOTOR_ON" := 1;
"MOTOR_OFF" := 0;
END_IF;

IF "Paro_izquierda_S2" OR "Paro_derecha_S4" THEN
"MOTOR_ON" := 0;
"MOTOR_OFF" := 1;
END_IF;
```

Figura 14. Ejemplo de código en lenguaje SCL (Fuente: Sistema de información TIA Portal).

Lenguaje de programación GRAPH:

GRAPH es un lenguaje de programación gráfico para crear controles secuenciales. Permite programar controles secuenciales de manera clara y rápida utilizando cadenas secuenciales.

El proceso se descompone en etapas individuales con un alcance funcional delimitable y se organiza en las cadenas secuenciales. Las acciones a ejecutar se determinan en las etapas individuales, identificando el paso de una etapa a la otra mediante condiciones de transición.

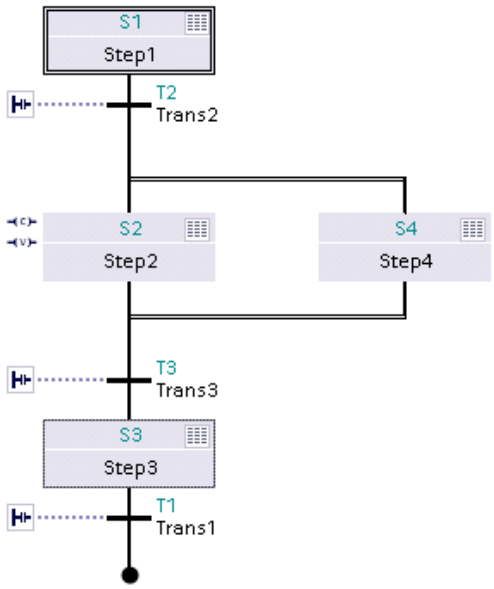


Figura 15. Ejemplo de la interfaz gráfica del lenguaje GRAPH (Fuente: Sistema de información TIA Portal).

Para finalizar, cabe mencionar que los editores más destacables en este software, coincidiendo con las que más se han necesitado explotar para la realización del proyecto son:

- Configuración de dispositivos: Gestiona la configuración del hardware del sistema así como su topología de red.
- Bloques de programa: Gestiona los bloques del programa de los que constará el proyecto
- Variables PLC: Gestiona las variables de entrada y salida del PLC así como las diversas marcas.
- Tipos de datos PLC: Permite la creación de patrones de estructuras de datos definidos por el usuario (UDT).
- Tablas de observación y forzado permanente: Permite definir variables a observar, forzar sus valores y poder hacer pruebas.

4.1.2.3. SIMATIC WinCC

SIMATIC WinCC, es un software de desarrollo de sistemas SCADA del fabricante Siemens y puede ser utilizado junto con cualquier controlador del mismo fabricante.

A diferencia de su versión reducida, WinCC Basic, WinCC puede ser empleado en sistemas SCADA generales y no tan solo en paneles HMI, así como ejecutar Runtime (modo de ejecución de la aplicación) avanzado e importar o elaborar textos del sistema.

Por lo que respecta a sus principales características operativas, WinCC permite crear proyectos de diferente tipo:

- Estación monopuesto: Es el que se implementa en el proyecto. Se caracteriza por registrar en el mismo servidor todos los eventos del registro de tags, alarmas y gestión de usuarios. Además de realizar la función de servidor web en el caso de haber sido configurado.
- Estación multipuesto: Se caracteriza por repartir las funciones descritas en el tipo de estación monopuesto en diversos servidores.
- Cliente: Demandante de los recursos y servicios al servidor al que esté asociado.

Está dotado de una interfaz de programación con los lenguajes Visual Basic Script Edition (VBS Script) y ANSI C y emplea Microsoft SQL Server para realizar las tareas de inicio de sesión y gestión de base de datos.

En cuanto a drivers de comunicación, WinCC dispone de drivers para comunicar con múltiples protocolos. A modo de resumen WinCC consta de los drivers siguientes:

- SIMATIC S7
- SIMATIC S5

- SIMATIC TI
- Allen Bradley
- Mitsubishi Ethernet
- Modbus TCPIP
- OPC
- Profibus

Para finalizar, cabe mencionar que los editores más destacables en este software, coincidiendo con las que más se han necesitado explotar para la realización del proyecto son:

- Equipo: Gestiona las características de arranque y runtime del equipo.
- Administración de variables: Crea y edita variables y estructuras de variables así como los drivers de comunicación.
- *Graphics designer:* Crea y edita sinópticos del proceso (pantallas, mandos de operación, iconos, etc).
- Menús y barras de herramientas: Configura las barras de herramientas y menús que se desean implementar.
- *Alarm logging:* Configura los avisos, tipos de aviso, grupos de aviso y archiva eventos.
- *Global Script:* Librería donde se pueden crear acciones en C o VBS para dinamizar el proyecto.
- *Text Library:* Permite crear o editar textos de usuario en función del idioma.
- *User administrator:* Gestiona los grupos de usuarios, usuarios y sus derechos y permisos.

4.1.3. Comunicaciones PLC – SCADA

Como ya se expuso en el apartado *4.1. Arquitectura del sistema de control*, a nivel de red, el PLC emplea una interfaz PROFINET/Industrial Ethernet (PN/IE) mediante la cual establece conexión con el SCADA.

Para implementar dicha conexión a nivel de software se debe configurar el hardware del sistema en TIA Portal y los drivers de comunicación en WinCC, así como asegurar que todos los elementos que participen en la comunicación formen parte de un rango de IP's compatible.

En TIA Portal se deberá asignar la subred PN/IE al puerto del PLC, para ello se deberá:

1. Abrir el editor de dispositivos y redes situado en el menú de la izquierda y acceder a la pestaña superior derecha de vista de redes.

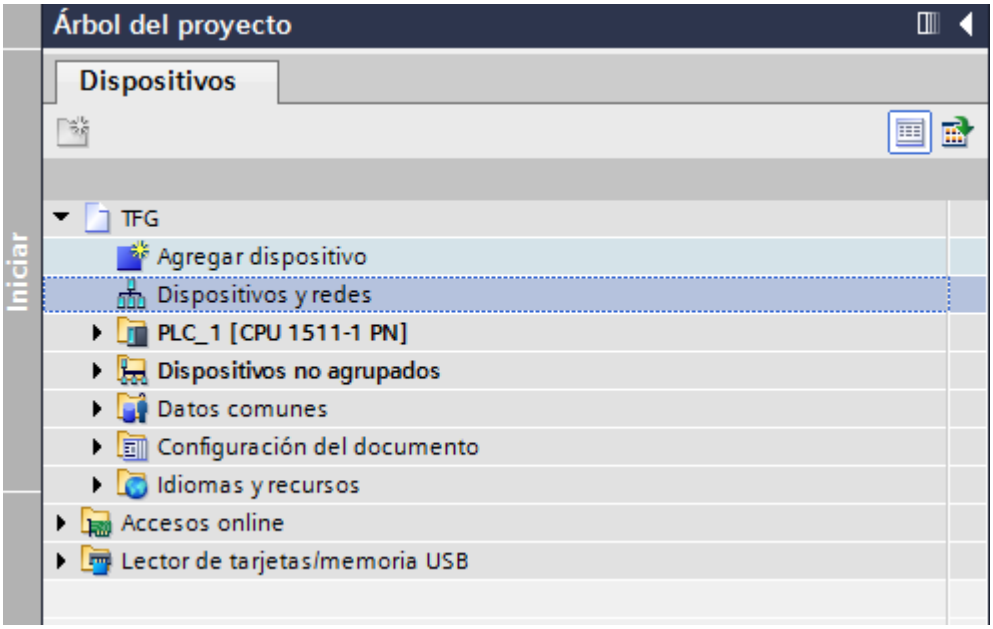


Figura 16. Acceso a la visualización de dispositivos y redes en TIA Portal.

Aparecerá una pantalla donde solamente se verá el hardware que se haya configurado.

2. Hacer clic derecho con el ratón sobre el PLC y seleccionar propiedades.
3. Acceder a Interfaz PROFINET [X1] → Direcciones Ethernet

Se verá que en el apartado de interfaz conectada en red no aparece ninguna subred conectada.

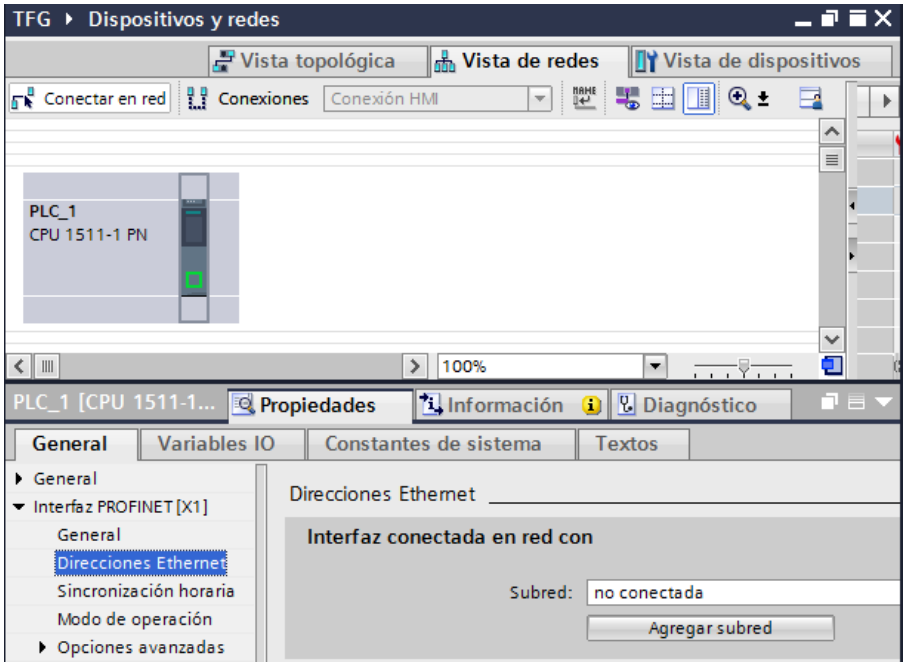


Figura 17. Configuración la interfaz conectada en red.

4. Seleccionar y agregar subred PN/IE_1.

Se observará que en la pantalla que antes solamente aparecía el hardware, ahora aparece indicada la interfaz de red.

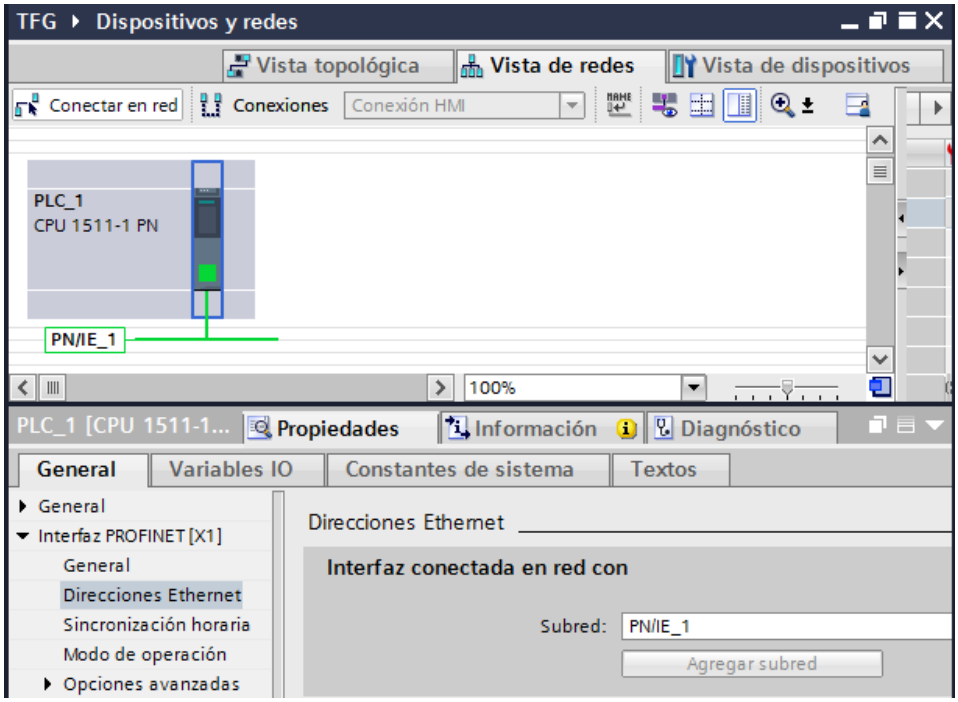


Figura 18. Configuración la interfaz conectada en red.

Habiendo realizado estos pasos, se habrá configurado el puerto del PLC para acceder a la red, el siguiente requisito necesario será configurar WinCC para que pueda comunicar con el PLC.

En primero lugar se deberá cerciorar que el punto de acceso de la aplicación esté asociado a la tarjeta de red del equipo servidor.

Para ello se deberá:

1. Buscar en el buscador de programas de Windows “Set PG-PC Interface” y ejecutarlo.
2. En la pestaña de vía de acceso habrá que asignar el punto de acceso S7ONLINE (STEP 7) a la tarjeta de red del equipo.

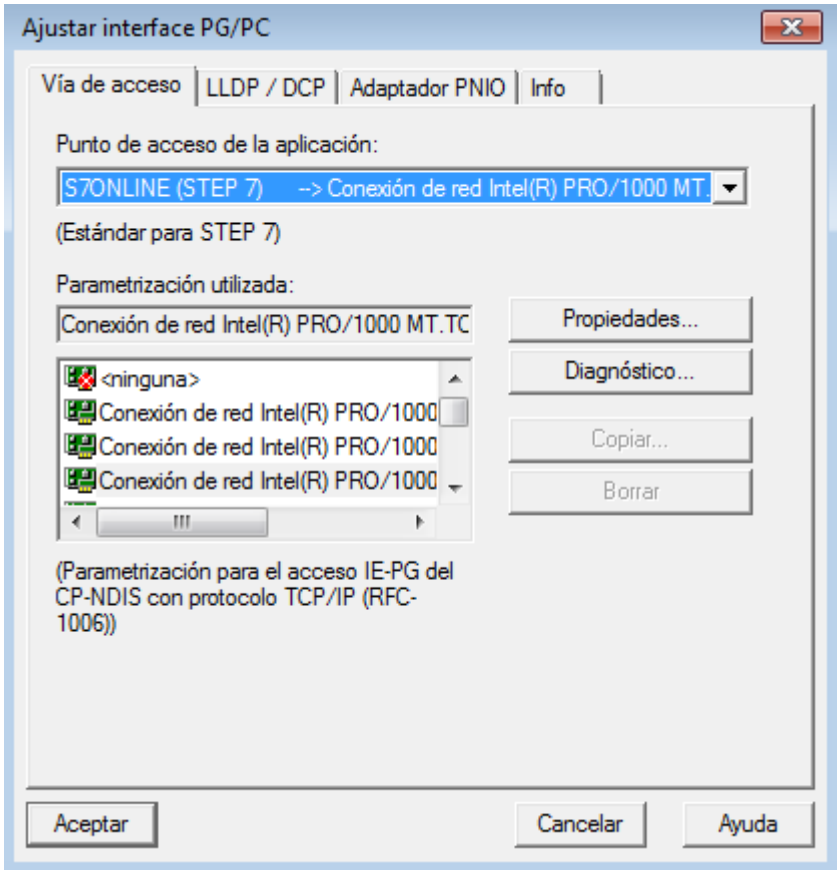


Figura 19. Asignación de punto del punto de acceso S7ONLINE de STEP 7.

Este punto de acceso será el que hará de nodo de enlace con los drivers de comunicación que se deben configurar para comunicar con el PLC, para ello se procederá a:

1. Desde el WinCCExplorer, abrir el editor de administración de variables.

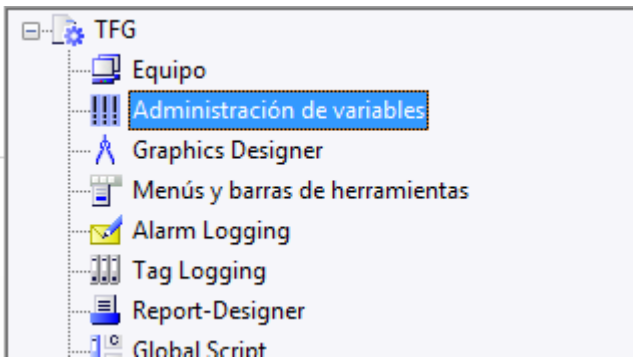


Figura 20. Vista del listado de editores en WinCCExplorer.

2. Click derecho en administración de variables → Agregar nuevo driver → SIMATIC S7-1200, S7-1500 Channel

Figura 21. Agregación del driver necesario para comunicar con PLC's de la familia ST-1200 o S7-1500.

Este driver es característico para las comunicaciones con PLC's de las familias S7-1200 y S7-1500, una vez agregado se crea y se configura una nueva conexión:

3. Clic derecho en la unidad de canal OMS+ → Nueva conexión. La nombramos como se desee (S7-1500).
4. Clic derecho en la nueva conexión → Parámetros de enlace
5. En la pantalla emergente, se designará la IP del PLC, el punto de acceso previamente asignado a la tarjeta de red y la familia del PLC.

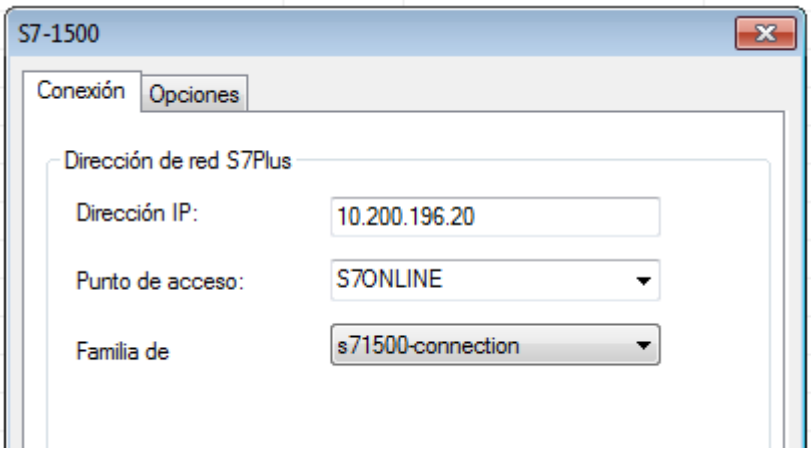


Figura 22. Configuración de la conexión S7-1500 creada.

Llegados a este punto, el PLC y el SCADA estarán configurados para comunicar datos entre ellos. Para ello, cuando se defina una variable externa, se deberá asignar al campo de conexión la conexión creada S7-1500 y el campo de dirección deberá apuntar al bloque de datos (DB) donde se aloje la variable en el PLC, así como la posición que ocupe dentro de dicho DB.

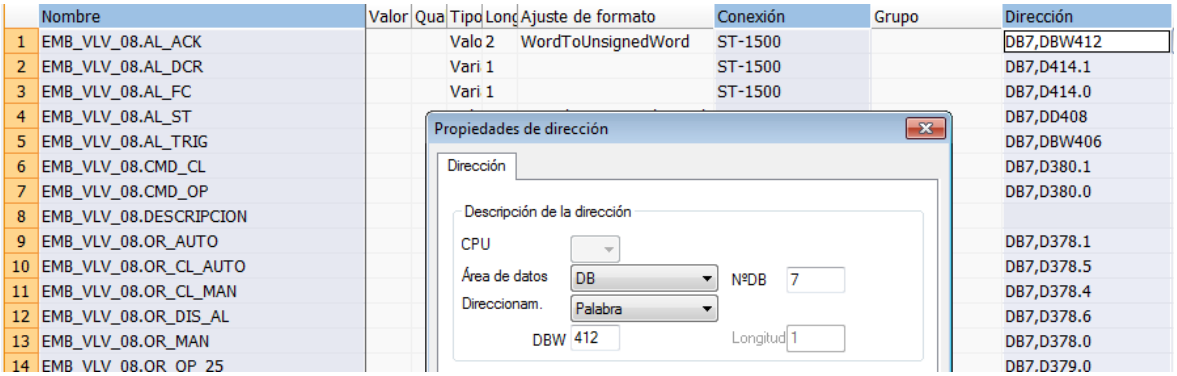


Figura 23. Asignación de dirección de comunicación de una variable del SCADA con el PLC.

Para finalizar este apartado, cabe matizar que para facilitar el direccionamiento de los diversos tags que constituyen el proyecto, se emplean estructuras de variable.

Estas estructuras permiten crear un patrón de tags (estructura del tipo válvula, por ejemplo), para ello se asigna a los diferentes elementos que forman la estructura el offset que tienen en el PLC, es decir, la distancia que hay de uno al otro (variando según el tipo de dato).

Elementos de estructura [SV_VALVULA]

	Nombre	Longitud AS	Offset AS	Offset de bit AS
1	OR_MAN	2	0	0
2	OR_AUTO	2	0	1
3	OR_OP_MAN	2	0	2
4	OR_OP_AUTO	2	0	3
5	OR_CL_MAN	2	0	4
6	OR_CL_AUTO	2	0	5
7	OR_DIS_AL	2	0	6

Figura 24. Asignación del offset en el PLC a los elementos de la estructura de variables.

A posterior, se crean instancias de estas estructuras (EVAP_VLV_01, por ejemplo), asignando a esta instancia en particular el DB del PLC al que se debe dirigir y su punto inicial dentro de este (DBB).

Variables de estructura [SV_VALVULA]

	Nombre	Conexión	Dirección
1	EMB_VLV_08	ST-1500	DB7,DBB378
2	EVAP_VLV_00	ST-1500	DB7,DBB0
3	EVAP_VLV_01	ST-1500	DB7,DBB54
4	EVAP_VLV_03	ST-1500	DB7,DBB162
5	MEZC_VLV_04	ST-1500	DB7,DBB216

Figura 25. Asignación de la dirección de DB y DBB del tag de la instancia de la estructura de variables.

Por último, WinCC relaciona la dirección de la instancia con los offset previamente definidos, creando así la dirección definitiva del tag.

Elementos de variables de estructura [SV_VALVULA]

	Nombre	Conexión	Dirección
1	EMB_VLV_08.AL_ACK	ST-1500	DB7,DBW412
2	EMB_VLV_08.AL_DCR	ST-1500	DB7,D414.1
3	EMB_VLV_08.AL_FC	ST-1500	DB7,D414.0
4	EMB_VLV_08.AL_ST	ST-1500	DB7,DD408

Figura 26. Dirección final del tag creado en la estructura de variables.

Como se puede apreciar en las figuras previas, la fórmula que emplea WinCC para crear los tags consiste en concatenar las instancias definidas con los elementos de la estructura siguiendo la siguiente fórmula:

$$Tag = Instancia.Elemento \qquad \textbf{(Expresión.2)}$$

4.2. Descomposición del problema de control

La problemática subyacente al control del proyecto radica en su estructuración en diversas partes que dependen unas de otras.

En este proyecto, antes de poder realizar la programación de una secuencia de proceso, se deberá haber realizado la programación de los elementos individuales que participan activamente en dicha secuencia.

A su vez, dichos elementos deberán interactuar unos con otros, de tal modo que, previamente, se deberá aplicar una lógica para simular su comportamiento.

Para finalizar, el control del proceso se divide en dos modos de funcionamientos claramente distinguidos: el modo manual y el modo automático.

La aplicación del control deberá mantener ambos modos de funcionamiento aislados, es decir, se deberá gestionar por un lado el modo manual y por otro el automático.

En el modo de operación manual, todo el control de los elementos y la producción se realizará desde la aplicación de SCADA, es por ello que también podría considerarse un control remoto, pues no se realiza desde la planta.

Por otro lado, en el modo automático será el PLC quien determine las acciones que deberán realizar los distintos elementos, quedando la aplicación SCADA relegada a determinar ciertos valores de consigna y a cambiar el modo de operación.

4.3. Codificación de los elementos y sistemas

La codificación de los elementos empleada tanto en la programación del PLC como en la programación de la aplicación SCADA, coincide con la codificación ya expuesta y detallada en el punto *3.2.1. Equipos*.

Por otro lado, los sistemas del proceso, entendidos como las diversas etapas que lo componen se codifican según la tabla siguiente:

Tabla 3. Codificación de los sistemas.

CODIFICACIÓN	DESCRIPCIÓN DEL SISTEMA
SEC_01	Sistema de evaporación
SEC_02	Sistema de mezclado
SEC_03	Sistema de embotellado

4.4. Definición del fichero de intercambio PLC – SCADA

Para determinar la relación entre los tags de la aplicación SCADA con las señales del PLC, se debe realizar un documento que recoja dichos tags y deje clara constancia del modo en que interaccionan con el controlador, indicando si son entradas (E), salidas (S) o entrada-salida (E/S).

A continuación se adjuntan en formato tabla los ficheros de intercambio para las diversas estructuras de variables empleadas:

Tabla 4. Fichero de intercambio PLC –SCADA del objeto válvula.

Estructura de variable: Válvula			PLC			SCADA		
Descripción	**TAG**	**Tipo**	**E**	**E/S**	**S**	**E**	**E/S**	**S**
Orden de modo manual	OR_MAN	Bool	1					1
Orden modo automático	OR_AUTO	Bool	1					1
Orden de apertura en modo manual	OR_OP_MAN	Bool	1					1
Orden de apertura en modo automático	OR_OP_AUTO	Bool	1					1
Orden de cierre en modo manual	OR_CL_MAN	Bool	1					1
Orden de cierre en modo automático	OR_CL_AUTO	Bool	1					1
Orden de deshabilitar alarmas	OR_DIS_AL	Bool	1					1
Orden de reset de horas de funcionamiento	OR_RST_FUNC	Bool	1					1
Orden de apertura al 25%	OR_OP_25	Bool	1					1
Orden de apertura al 50%	OR_OP_50	Bool	1					1
Orden de apertura al 75%	OR_OP_75	Bool	1					1
Orden de apertura al 100%	OR_OP_100	Bool	1					1
Orden de refresco del valor de consigna	OR_REFRESH_SP_OP	Bool	1					1
Comando de abrir	CMD_OP	Bool			1	1		
Comando de cerrar	CMD_CL	Bool			1	1		
Estado modo manual	ST_MAN	Bool			1	1		
Estado modo automático	ST_AUTO	Bool			1	1		
Estado abierto	ST_OP	Bool			1	1		
Estado cerrado	ST_CL	Bool			1	1		
Estado deshabilitación de alarmas	ST_DIS_AL	Bool			1	1		
Estado de bloqueo (posición de seguridad)	ST_LCK	Bool			1	1		
Consigna de nivel de apertura	SP_OP	Real			1	1		
Valor de las horas de funcionamiento	PV_HORAS_FUNC	Real			1	1		
Valor de los minutos de funcionamiento	PV_MIN_FUNC	Real			1	1		
Valor de los segundos de funcionamiento	PV_SEC_FUNC	Real			1	1		
Valor de proceso de nivel de apertura	PV_OP	Real			1	1		
Palabra de generación de alarmas	AL_TRIG	Word			1	1		
Palabra del estado general de alarmas	AL_ST	Dword	1					1
Palabra de reconocimiento de alarmas	AL_ACK	Word		1			1	
Alarma de finales de carrera	AL_FC	Bool			1	1		
Alarma de discrepancia	AL_DCR	Bool			1	1		

Tabla 5. Fichero de intercambio PLC –SCADA del objeto sonda.

Estructura de variable: Sonda			PLC			SCADA		
Descripción	TAG	Tipo	E	E/S	S	E	E/S	S
Orden de deshabilitación de alarmas	OR_DIS_AL	Bool	1					1
Orden para leer nivel de apertura válvula	OR_REFRESH_SP_OP	Bool	1					1
Estado de deshabilitación de alarmas	ST_DIS_AL	Bool			1	1		
Valor de lectura	PV_VALUE	Real		1			1	
Consigna de tolerancia para la alarma muy alta	SP_V_AL_HIHI	Real	1					1
Consigna de tolerancia para la alarma alta	SP_V_AL_HI	Real	1					1
Consigna de tolerancia para la alarma baja	SP_V_AL_LO	Real	1					1
Consigna de tolerancia para la alarma muy baja	SP_V_AL_LOLO	Real	1					1
Consigna de tiempo para la alarma muy alta	SP_T_AL_HIHI	Real	1					1
Consigna de tiempo para la alarma alta	SP_T_AL_HI	Real	1					1
Consigna de tiempo para la alarma baja	SP_T_AL_LOLO	Real	1					1
Consigna de tiempo para la alarma muy baja	SP_T_AL_LO	Real	1					1
Consigna valor máximo de lectura	SP_VALUE_MAX	Real	1					1
Consigna valor mínimo de lectura	SP_VALUE_MIN	Real	1					1
Palabra de generación de alarmas	AL_TRIG	Word			1	1		
Palabra del estado general de alarmas	AL_ST	Dword	1					1
Palabra de reconocimiento de alarmas	AL_ACK	Word		1			1	
Alarma por fallo de comunicaciones	AL_NR	Bool			1	1		
Alarma por valor muy elevado	AL_HIHI	Bool			1	1		
Alarma por valor elevado	AL_HI	Bool			1	1		
Alarma por valor bajo	AL_LO	Bool			1	1		
Alarma por valor muy bajo	AL_LOLO	Bool			1	1		

Tabla 6. Fichero de intercambio de archivos PLC –SCADA de la secuencia de mezclado.

Estructura de variable: Secuencia de mezclado			PLC			SCADA		
Descripción	TAG	Tipo	E	E/S	S	E	E/S	S
Orden marcha	OR_MARCHA	Bool	1					1
Orden paro	OR_PARO	Bool	1					1
Orden dispositivos de la etapa en modo auto.	OR_DISP_AUTO	Bool	1					1
Estado fase de carga producto A	ST_FASE_MAT_1	Bool			1	1		
Estado fase de carga producto B	ST_FASE_MAT_2	Bool			1	1		
Estado fase de mezclado	ST_FASE_MEZCLA	Bool			1	1		
Consigna de tiempo de mezclado	SP_T_MEZCLA	Real	1					1
Estado fase de descarga	ST_DECARGA	Bool			1	1		
Estado mezcla finalizada	ST_MEZCLA_RDY	Bool			1	1		
Estado de marcha	ST_RUN	Bool			1	1		
Estado de paro	ST_STOP	Bool			1	1		
Tiempo de mezclado transcurrido	T_ACTUAL	Bool			1	1		
Consigna de carga del producto A	SP_MAT_1	Real	1					1
Consigna de carga del producto B	SP_MAT_2	Real	1					1
Habilitación del cambio de consigna de mezcla	DIS_SP	Bool			1	1		
Alarma en fase de carga de producto A	AL_PROD_1	Bool			1	1		
Alarma en fase de carga de producto B	AL_PROD_2	Bool			1	1		
Alarma en fase de mezclado	AL_MEZCLA	Bool			1	1		
Alarma en fase de descarga	AL_DESCARGA	Bool			1	1		

Tabla 7. Fichero de intercambio PLC –SCADA de la secuencia de evaporación.

Estructura de variable: Secuencia de evaporación			PLC			SCADA		
Descripción	**TAG**	**Tipo**	**E**	**E/S**	**S**	**E**	**E/S**	**S**
Orden marcha	OR_MARCHA	Bool	1					1
Orden paro	OR_PARO	Bool	1					1
Orden dispositivos de la etapa en modo auto.	OR_DISP_AUTO	Bool	1					1
Estado fase evaporación	ST_FASE_EVAP	Bool			1	1		
Estado fase de carga	ST_FASE_CARGA	Bool			1	1		
Consigna tiempo de evaporación	SP_T_EVAP	Real			1	1		
Estado evaporación lista	ST_EVAP_RDY	Bool			1	1		
Estado de marcha	ST_RUN	Bool			1	1		
Estado de paro	ST_STOP	Bool			1	1		
Tiempo de evaporado transcurrido	T_ACTUAL	Real			1	1		
Alarma en la fase de carga	AL_CARGA	Bool			1	1		
Alarma en la fase de evaporación	AL_EVAP	Bool			1	1		
Alarma en la fase de descarga	AL_DESCARGA	Bool			1	1		

Tabla 8. Fichero de intercambio de archivos PLC –SCADA de la secuencia de embotellado.

Estructura de variable: Secuencia de embotellado			PLC			SCADA		
Descripción	**TAG**	**Tipo**	**E**	**E/S**	**S**	**E**	**E/S**	**S**
Orden dispositivos de la etapa en modo auto.	OR_DISP_AUTO	Bool	1					1
Orden marcha	OR_MARCHA	Bool	1					1
Orden paro	OR_PARO	Bool	1					1
Estado de colocación botella	STEP_ACTUAL	Bool			1	1		
Estado botella detectada	ST_DET	Bool			1	1		
Estado fase de translación	FASE_TRANS	Bool			1	1		
Estado fase de llenado	FASE_LLENANDO	Bool			1	1		
Estado de marcha	ST_RUN	Bool			1	1		
Estado de paro	ST_STOP	Bool			1	1		
Estado de falta de producto	ST_STNDBY	Bool			1	1		
Variable de contaje de botellas producidas	CONT_BOTTLE	Real			1	1		
Orden para resetear contador	OR_RST_CNT	Bool	1					1
Alarma en fase de translación	AL_TRANS	Bool			1	1		
Alarma en fase de llenado	AL_LLENADO	Bool			1	1		

Tabla 9. Fichero de intercambio PLC –SCADA del objeto motor (aplicado en los elementos con motor).

Estructura de variable: Motor			PLC			SCADA		
Descripción	**TAG**	**Tipo**	**E**	**E/S**	**S**	**E**	**E/S**	**S**
Orden de modo manual	OR_MAN	Bool	1					1
Orden modo automático	OR_AUTO	Bool	1					1
Orden de marcha en modo manual	OR_MARCHA_MAN	Bool	1					1
Orden de marcha en modo automático	OR_MARCHA_AUTO	Bool	1					1
Orden de paro en modo manial	OR_PARO_MAN	Bool	1					1
Orden de paro en modo automático	OR_PARO_AUTO	Bool	1					1
Orden de deshabilitar alarmas	OR_DIS_AL	Bool	1					1
Orden para resetear horas funcionamiento	OR_RST_FUNC	Bool	1					1
Orden para refrescar el SP	OR_REFRESH_SP	Bool	1					1
Orden nivel 1 de velocidad	OR_200rpm	Bool	1					1
Orden nivel 2 de velocidad	OR_400rpm	Bool	1					1
Orden nivel 3 de velocidad	OR_600rpm	Bool	1					1
Orden nivel 4 de velocidad	OR_800rpm	Bool	1					1
Orden nivel 5 de velocidad	OR_1000rpm	Bool	1					1
Comando de marcha	CMD_MARCHA	Bool			1	1		
Comando de paro	CMD_PARO	Bool			1	1		
Comando de aceleración	CMD_AC	Bool			1	1		
Comando deceleración	CMD_DEC	Bool			1	1		
Estado modo manual	ST_MAN	Bool			1	1		
Estado modo automático	ST_AUTO	Bool			1	1		
Estado marcha	ST_MARCHA	Bool			1	1		
Estado paro	ST_PARO	Bool			1	1		
Estado bloqueado (posición de seguridad)	ST_LCK	Bool			1	1		
Estado de deshabilitación de alarmas	ST_DIS_AL	Bool			1	1		
Valor horas de funcionamiento	PV_HOUR_FUNC	Real			1	1		
Valor minutos de funcionamiento	PV_MIN_FUNC	Real			1	1		
Valor segundos de funcionamiento	PV_SEC_FUNC	Real			1	1		
Valor de proceso velocidad de giro	PV_VALUE	Real			1	1		
Consigna de velocidad de giro	SP_VALUE	Real			1	1		
Valor de tolerancia alarma altas revoluciones	SP_TOL_HI	Real	1					1
Valor de tolerancia alarma bajas revoluciones	SP_TOL_LO	Real	1					1
Valor de tiempo alarma altas revoluciones	SP_T_HI	Real	1					1
Valor de tiempo alarma bajas revoluciones	SP_T_LO	Real	1					1
Palabra de generación de alarmas	AL_TRIG	Word			1	1		
Palabra del estado general de alarmas	AL_ST	Dword	1					1
Palabra de reconocimiento de alarmas	AL_ACK	Word		1			1	
Alarma del térmico	AL_TR	Bool		1			1	
Alarma de discrepancia	AL_DCR	Bool		1			1	
Alarma revoluciones altas	AL_HI	Bool			1	1		
Alarma revoluciones bajas	AL_LO	Bool			1	1		
Alarma por avería	AL_AV	Bool		1			1	

Tabla 10. Fichero de intercambio PLC –SCADA del objeto depósito.

Estructura de variable: Depósito			PLC			SCADA		
Descripción	**TAG**	**Tipo**	**E**	**E/S**	**S**	**E**	**E/S**	**S**
Orden lectura estado elementos de carga	OR_REFRESH_SP_CARGA	Bool	1					1
Orden lectura estado elementos de descarga	OR_REFRESH_SP_DESCARGA	Bool	1					1
Estado lleno	ST_LLENO	Bool			1	1		
Estado vacío	ST_VACIO	Bool			1	1		
Estado nivel alto	ST_NIVEL_HI	Bool			1	1		
Estado nivel bajo	ST_NIVEL_LO	Bool			1	1		
Consigna volumen del depósito	SP_NIVEL_LLENO	Bool	1					1
Consigna nivel alto	SP_NIVEL_HI	Real	1					1
Consigna nivel bajo	SP_NIVEL_LO	Real	1					1
Valor de llenado	PV_NIVEL	Real		1			1	
Valor de llenado en porcentaje	PV_PORCENTAJE	Real			1	1		

Tabla 11. Fichero de intercambio PLC –SCADA del objeto calefactor.

Estructura de variable: Calefactor			PLC			SCADA		
Descripción	**TAG**	**Tipo**	**E**	**E/S**	**S**	**E**	**E/S**	**S**
Orden modo manual	OR_MAN	Bool	1					1
Orden modo automático	OR_AUTO	Bool	1					1
Orden marcha en manual	OR_MARCHA_MAN	Bool	1					1
Orden marcha en automático	OR_MARCHA_AUTO	Bool	1					1
Orden paro en manual	OR_PARO_MAN	Bool	1					1
Orden paro en automático	OR_PARO_AUTO	Bool	1					1
Orden de deshabilitación de alarmas	OR_DIS_AL	Bool	1					1
Orden reset horas de funcionamiento	OR_RST_FUNC	Bool	1					1
Orden de refresco del valor de consigna	OR_REFRESH_SP	Bool	1					1
Comando de marcha	CMD_MARCHA	Bool			1	1		
Comando de paro	CMD_PARO	Bool			1	1		
Comando de calentamiento	CMD_CALOR	Bool			1	1		
Comando de enfriamiento	CMD_FRIO	Bool			1	1		
Estado manual	ST_MAN	Bool			1	1		
Estado automático	ST_AUTO	Bool			1	1		
Estado de marcha	ST_MARCHA	Bool			1	1		
Estado de paro	ST_PARO	Bool			1	1		
Estado de alarmas deshabilitadas	ST_DIS_AL	Bool			1	1		
Valor horas de funcionamiento	PV_HOUR_FUNC	Real			1	1		
Valor minutos de funcionamiento	PV_MIN_FUNC	Real			1	1		
Valor segundos de funcionamiento	PV_SEC_FUNC	Real			1	1		
Valor de proceso temperatura	PV_VALUE	Real			1	1		
Valor de consigna temperatura	SP_VALUE	Real	1					1
Valor de tolerancia alarma alta temperatura	SP_TOL_HI	Real	1					1
Valor de tolerancia alarma baja temperatura	SP_TOL_LO	Real	1					1
Valor de tiempo alarma alta temperatura	SP_T_HI	Real	1					1
Valor de tiempo alarma baja temperatura	SP_T_LO	Real	1					1
Palabra de generación de alarmas	AL_TRIG	Word			1	1		
Palabra de estado general de alarmas	AL_ST	Dword	1					1
Palabra de reconocimiento de alarmas	AL_ACK	Word		1			1	
Alarma del térmico	AL_TR	Bool		1			1	
Alarma de discrepancia	AL_DCR	Bool		1			1	
Alarma temperatura alta	AL_HI	Bool			1	1		
Alarma temperatura baja	AL_LO	Bool			1	1		
Alarma por avería	AL_AV	Bool		1			1	

4.5. Simulación del proceso

4.5.1. Bloques del sistema y estructura de la simulación

Como ya se expuso previamente, la simulación del comportamiento de los diversos dispositivos y procesos se realiza mediante código embebido en el autómata.

Siendo el caso de código embebido, se podría definir como un único bloque que realiza la simulación.

Sin embargo, cada dispositivo o secuencia, cuenta con un bloque propio que contiene tanto su lógica de control como su lógica de simulación.

Es por ello que en este apartado se determinan los diversos sub-bloques que integran la simulación y se explica la manera en la que está estructurada.

En primer lugar, se tienen los diversos bloques de dispositivos, del tipo función (FC), en ellos se programa en lenguaje KOP el comportamiento de los mismos. A modo de ejemplo, en estos bloques se define de qué manera deberá una sonda de presión mostrar sus lecturas, esto es, sus límites inferior y superior, cada cuanto tiempo actualizará su valor, sus pasos de incremento o decremento, etc.

Los bloques empleados en el proyecto se ilustran en la siguiente figura, siendo, válvulas, sondas de presión y caudal, motores, depósitos y el intercambiador de calor (calefactor).

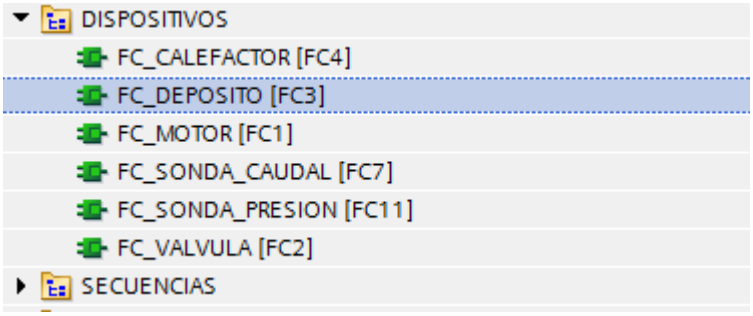


Figura 27. Bloques (FC) de los dispositivos.

El proceso de llamada a los bloques de función y su instanciamiento en un dispositivo concreto, se detalla más en profundidad en los apartados relativos a la programación PLC, así que de forma resumida e introductoria, cabe destacar que estos bloques ejecutan la lógica de control y simulación cada vez que sean llamados desde el programa principal, asignándoseles una instancia propia en un bloque de datos (DB) en el cual depositan sus datos (estados, valores, alarmas, etc).

Una vez determinados los bloques que se encargan de la simulación individual del comportamiento de los dispositivos, queda definir de qué manera se simula la interacción entre los mismos.

Por ejemplificar de igual manera que se hizo con los bloques de dispositivos, estos bloques de secuencias contendrán el código que controlará el valor de lectura de una sonda de caudal en función del nivel de apertura de la válvula asociada y la velocidad de bombeo de la bomba de impulsión.

La programación de estos bloques se realiza en lenguaje SCL y queda integrada junto con los bloques de función (FB) de las diferentes etapas del proceso, siendo éstas evaporación, mezcla y embotellado.

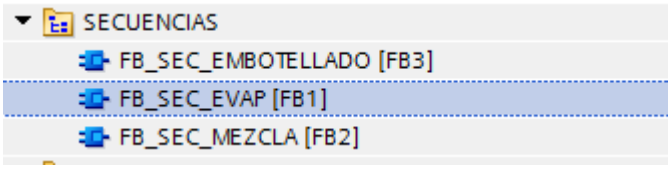


Figura 28. Bloques (FB) de los secuencias.

El motivo por el cual se integra el código de simulación de los dispositivos con su control y el código de simulación de interacción de una secuencia junto con el control mismo de dicha secuencia, es para minimizar el número de llamadas a bloques que deberá realizar el programa, así como hacerlo lo más compacto posible, con el fin de hacer una agrupación funcional que evite mezclar dispositivos de una etapa con dispositivos de otra.

4.5.2. Diseño del código embebido

En este apartado se explicará, aportando imágenes que lo ilustren, la parte de código más relevante para la simulación del proceso. El código completo se adjuntará en los anexos que acompañan a este documento.

4.5.2.1. Alarmas

Por lo que respecta a la gestión de alarmas, se tratan como bits que se pueden forzar desde la aplicación SCADA y se guardan en el PLC en una palabra de 16 bits (AL_TRIG), con la cual WinCC opera para tratar las alarmas como se explicará más adelante.

Todos los dispositivos que presenten alarmas, cuentan con el siguiente segmento, el cual mediante la función “AL_TRIG_GENERATOR” (explicada en el apartado de programación PLC) depositan el bit de alarma forzado en la palabra AL_TRIG, siempre y cuando esté inhabilitado el bit de deshabilitación de alarmas (ST_DIS_AL).

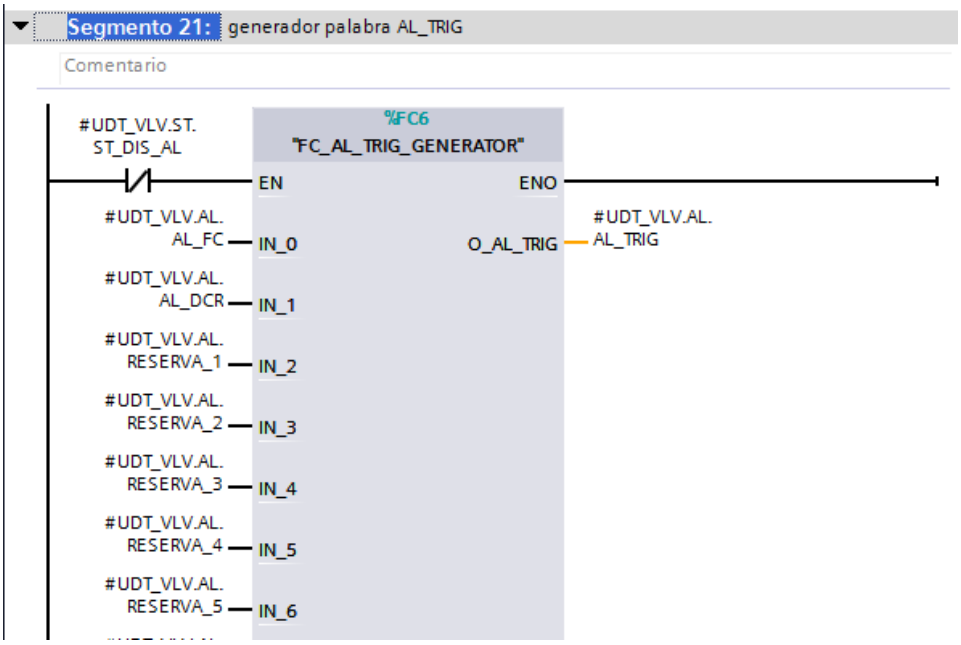


Figura 29. Segmento KOP de simulación de alarmas.

4.5.2.2. Válvulas

Las válvulas implementadas, permiten regular su nivel de apertura para controlar el flujo de caudal que debe circular, por ello, se debe asignar el nivel de consigna de apertura (SP_OP) al valor de proceso que realiza la confirmación (PV_OP) y asegurarse de que dicho valor esté a 0% en caso de que la válvula esté cerrada (ST_CL).

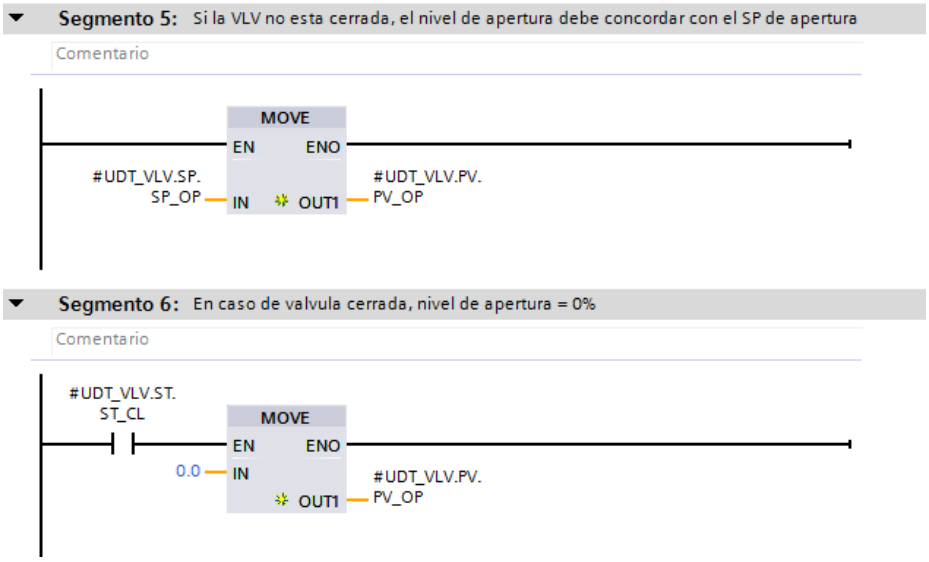


Figura 30. Segmentos KOP para asignar el valor de proceso del nivel de apertura de la válvula.

La otra parte de la simulación de las válvulas consiste en demorar la detección de estado de los finales de carrera una vez se realice la orden de abrir o cerrar. Con esto se pretende simular el tiempo de apertura o cierre de una válvula real, para ello se emplean las órdenes de comando (CMD_CL para el cierre y CMD_OP para la apertura), las cuales permanecen activas desde que son activadas mediante la orden de apertura o cierre hasta que se resetean una vez transcurridos los segundos de temporización, activando así el estado de final de carrera pertinente.

Con esta espera de 3 segundos, desde la aplicación SCADA podremos saber si la válvula está en fase de transición entre el estado de apertura y el estado de cierre.

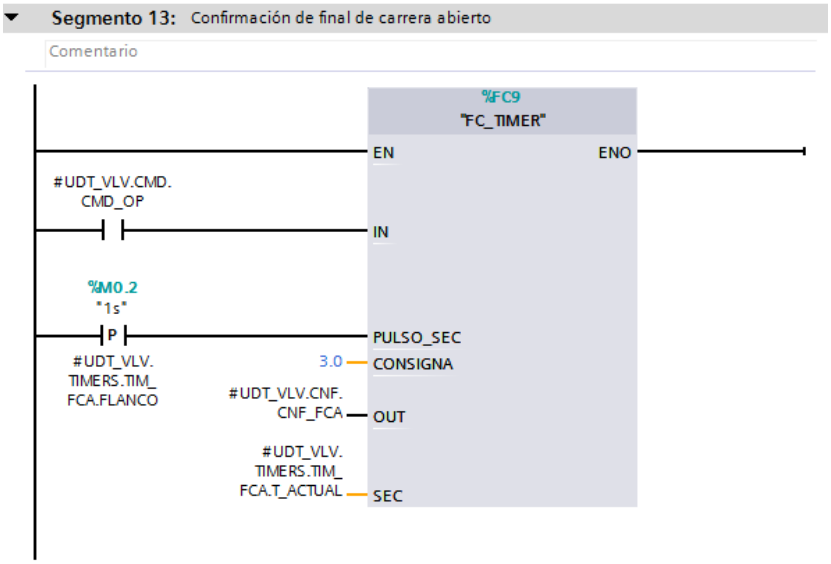


Figura 31. Segmento KOP para la transición de estado cerrado – abierto.

Por último, como ya se expuso, no disponemos de una planta real, por lo que los finales de carrera son simulados. Es por ello que se asignará el estado de apertura o cierre (ST_OP y ST_CL, respectivamente) a si final de carrera asociado.

Con el siguiente segmento, se gestiona el estado de apertura, activándolo cuando se detecta el final de carrera abierto y desactivándolo cuando se detecta el estado de carrera cerrado.

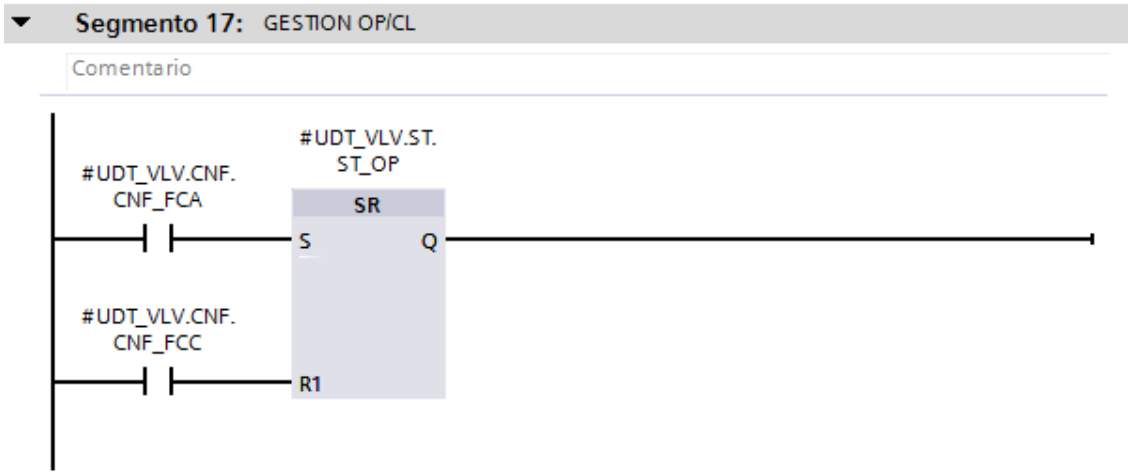


Figura 32. Segmento KOP de gestión del estado de apertura de la válvula.

A su vez, para gestionar el estado de cierre (ST_CL), se emplean los siguientes segmentos que realizan la acción opuesta que el segmento ilustrado en la figura anterior.

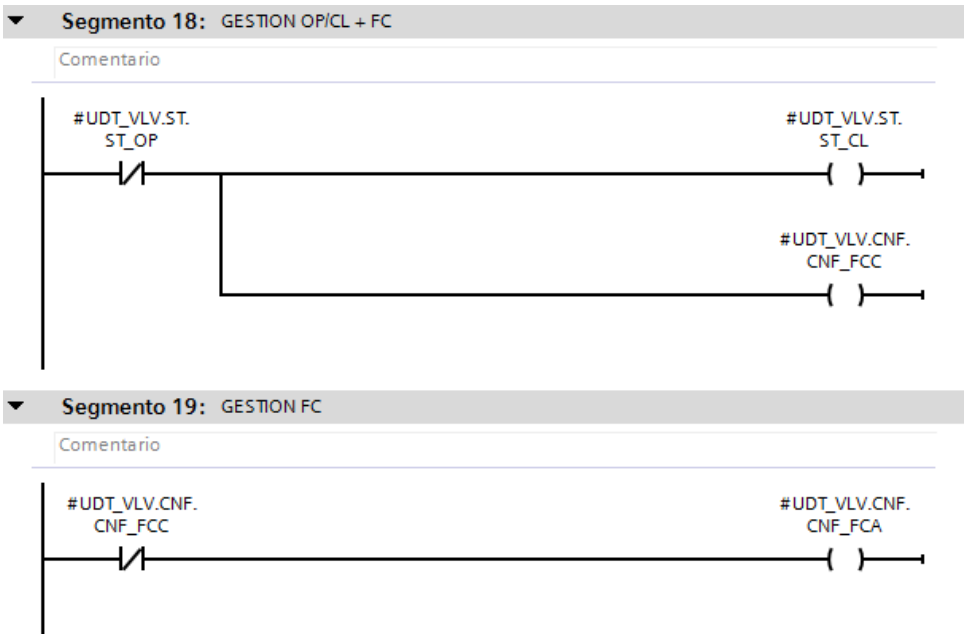


Figura 33. Segmentos KOP de gestión del estado de cierre de la válvula.

4.5.2.3. Motores

El bloque motor se emplea para el control y la simulación de diversos elementos: las bombas, el mezclador, el ventilador extractor y el motor de la cinta transportadora.

La simulación consistirá en mostrar un valor de proceso en función del valor de consigna seleccionado, distinguiendo 5 niveles de revoluciones por minuto: 200, 400, 600, 800 y 1000.

Con los siguientes segmentos se activará el comando de aceleración (CMD_AC) o el comando de deceleración (CMD_DEC), siempre que se dé la orden de actualizar la consigna (OR_REFRESH_SP) con el motor en marcha (ST_MARCHA) y dependiendo de si el valor de proceso está por encima o por debajo del valor de consigna.

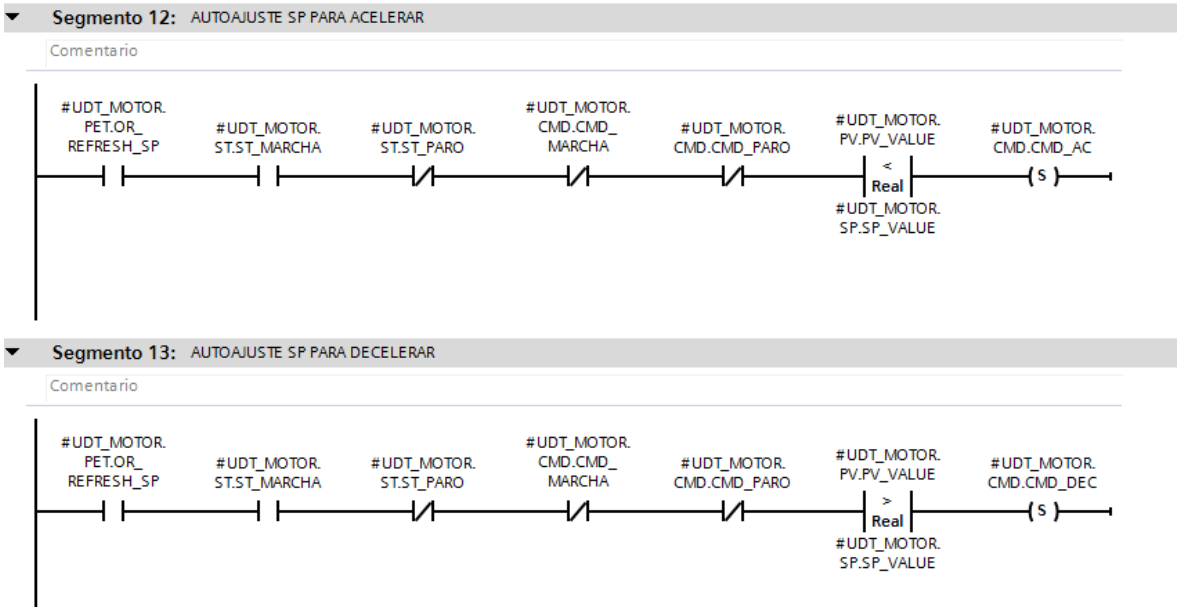


Figura 34. Segmentos KOP para la activación de los comandos de aceleración y deceleración.

A continuación, se procede a incrementar o decrementar el valor de proceso (PV_VALUE) con un paso de 200 en función del comando activo:

- Comando de marcha (CMD_MARCHA): Se da cuando se activa la orden de marcha con el motor en estado de paro.
- Comando de paro (CMD_PARO): Se da cuando se activa la orden de paro con el motor en estado de marcha.
- Comando de aceleración (CMD_AC): Se da cuando estando en estado de marcha, se recibe un cambio de nivel de consigna a una velocidad más alta.
- Comando de deceleración (CMD_DEC): Se da cuando estando en estado de marcha, se recibe un cambio de nivel de consigna a una velocidad más baja.

Como se ve en la siguiente figura, mediante un pulso de 1 segundo, se incrementa o decrementa un valor de 200 al valor de proceso en función del comando activo hasta que se llegue al nivel deseado.

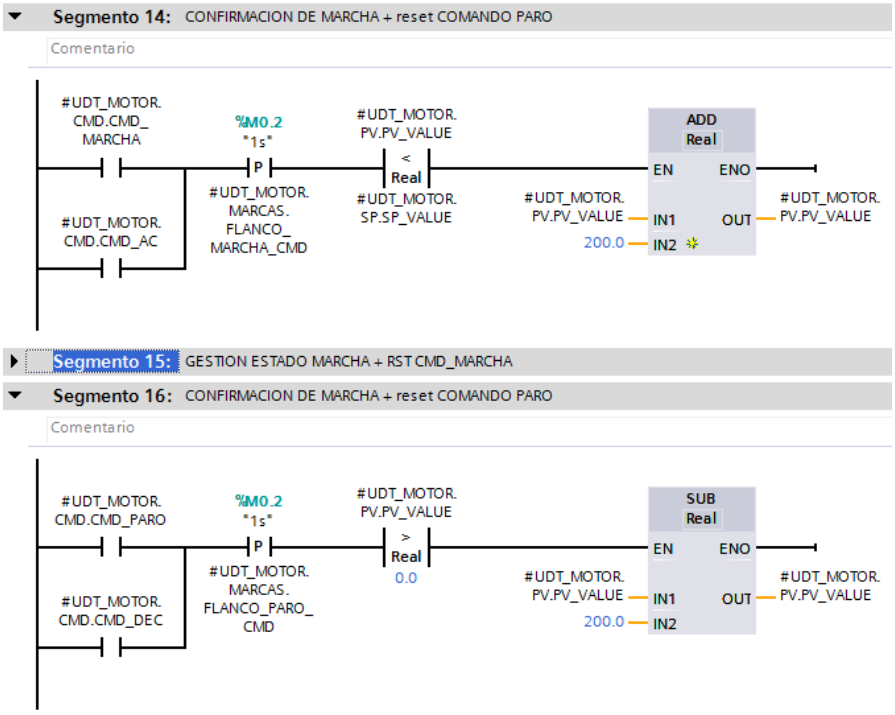


Figura 35. Segmentos KOP para asignar el valor de proceso de velocidad giro del motor.

Una vez el valor de proceso ha alcanzado el valor de consigna, se procede a desactivar el comando y a asignar el estado correspondiente, como se muestra en el siguiente ejemplo:

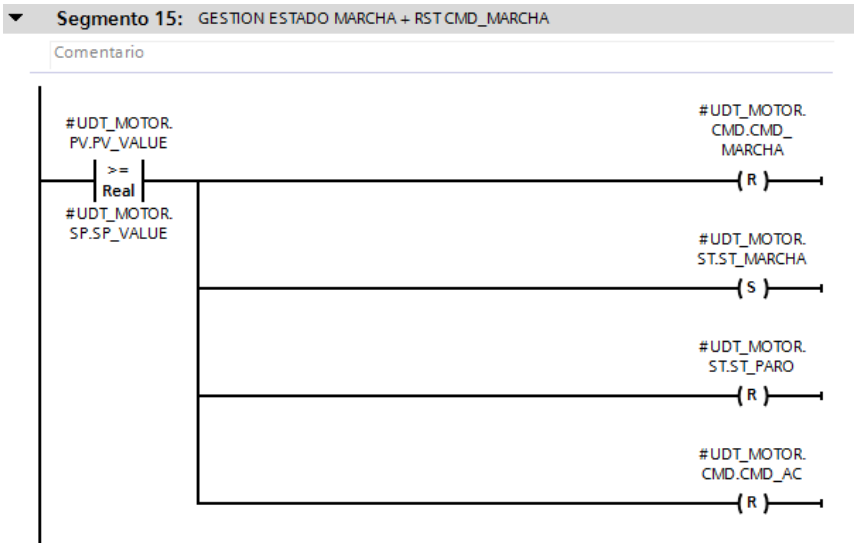


Figura 36. Segmento KOP para la desactivación de comandos de marcha y aceleración y la activación del estado de marcha.

4.5.2.4. Calefactor / Intercambiador de calor

La simulación del intercambiador de calor, coincide con el código KOP del motor en muchos aspectos.

Distinguiéndose por tener un paso de 0,75 en vez de 200, y sustituyendo los comandos de aceleración y deceleración por comandos de calor (CMD_CALOR) y frío (CMD_FRIO), respectivamente.

De igual manera que con el motor, cada vez que se realice un refresco del valor de consigna, el valor de proceso aumentará o disminuirá según el paso definido hasta alcanzar el nuevo valor de consigna, reseteando el comando una vez se alcance dicho nivel.

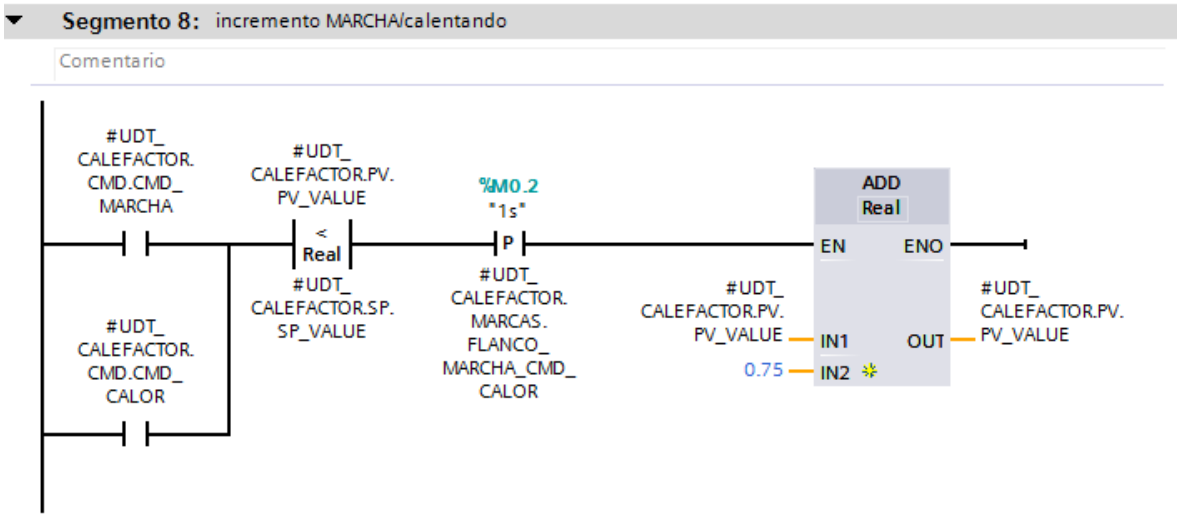


Figura 37. Segmento KOP para asignar el valor de proceso de temperatura al intercambiador de calor.

Por otro lado, la principal diferencia de este bloque de simulación, reside en la implementación de una oscilación en el valor de lectura para recrear un comportamiento realista de una sonda de temperatura.

Para ello se implementa una báscula (*flip-flop*), que cada 3 segundos incrementa 0,1 el valor de lectura para pasados otros 3 segundos, decrementar el mismo valor.

Esta báscula se asocia a la marca M03, la cual alterna su estado lógico cada 3 segundos como se detallará en el apartado de programación PLC.

Su utilidad en este caso es incrementar el valor cuando presenta un cambio de flanco ascendente y decrementarlo cuando se dé un cambio de flanco descendente.

En total se integran dos básculas, una para el estado de marcha y otra para el estado de paro, ya que se simula que el estado de paro se distingue por una lectura de una temperatura ambiente de 25 ºC.

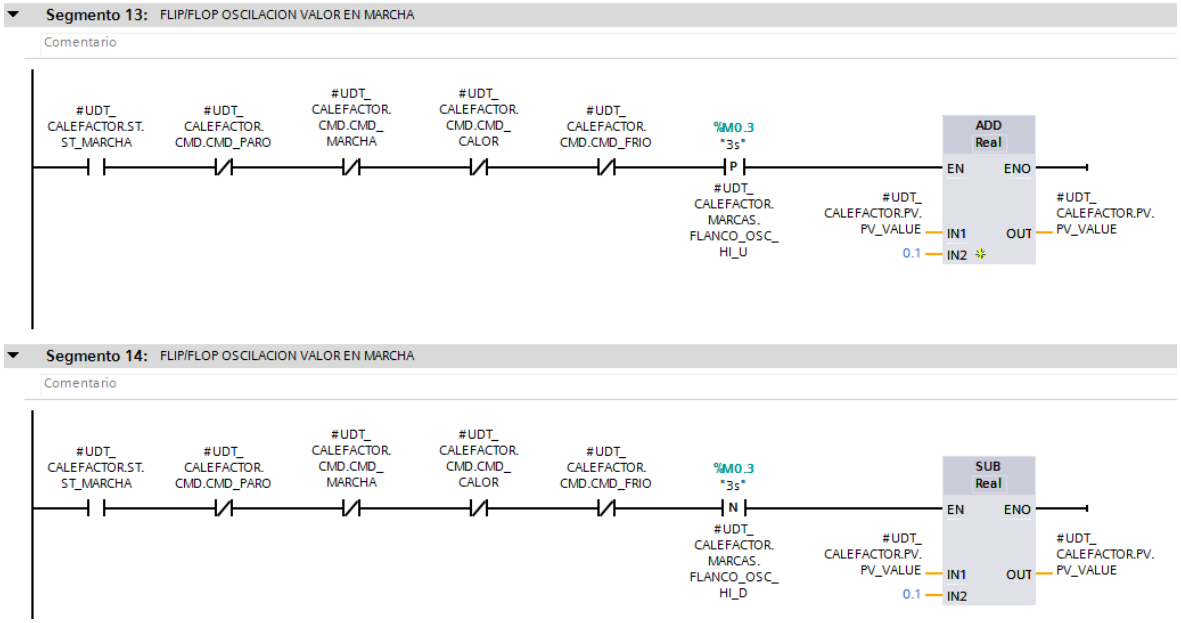


Figura 38. Segmentos KOP para el *flip-flop* de oscilación de lectura en el estado de marcha.

4.5.2.5. Sondas

Las sondas son los dispositivos del proyecto que se caracterizan por no tener ningún control más allá de su gestión de alarmas, son equipos mayoritariamente simulados.

Como se puede ver en el apartado del fichero de intercambio de las sondas, desde la aplicación SCADA se asignarán los límites de su lectura, esto es, el valor máximo y mínimo con el que deben operar.

En función de estos límites, se asignará el valor de consigna de lectura al que deba llegar la sonda en función de dos parámetros: el nivel de apertura de la válvula (SP_OP_VLV) y la velocidad de giro de la bomba (SP_RPM_BMB).

En el caso de las sondas de caudal (caracterizadas por tener un valor mínimo de lectura de valor cero), el flujo presenta una relación directamente proporcional a ambos parámetros. Sin embargo en las sondas de presión (caracterizadas por tener un valor mínimo de lectura diferente de cero), la presión presenta una relación directamente proporcional al nivel de apertura de la válvula pero inversamente proporcional a la velocidad de giro del motor de la bomba.

En las siguientes figuras se podrá observar cómo dependiendo de los valores de estos parámetros el nivel de lectura máximo se modifica.

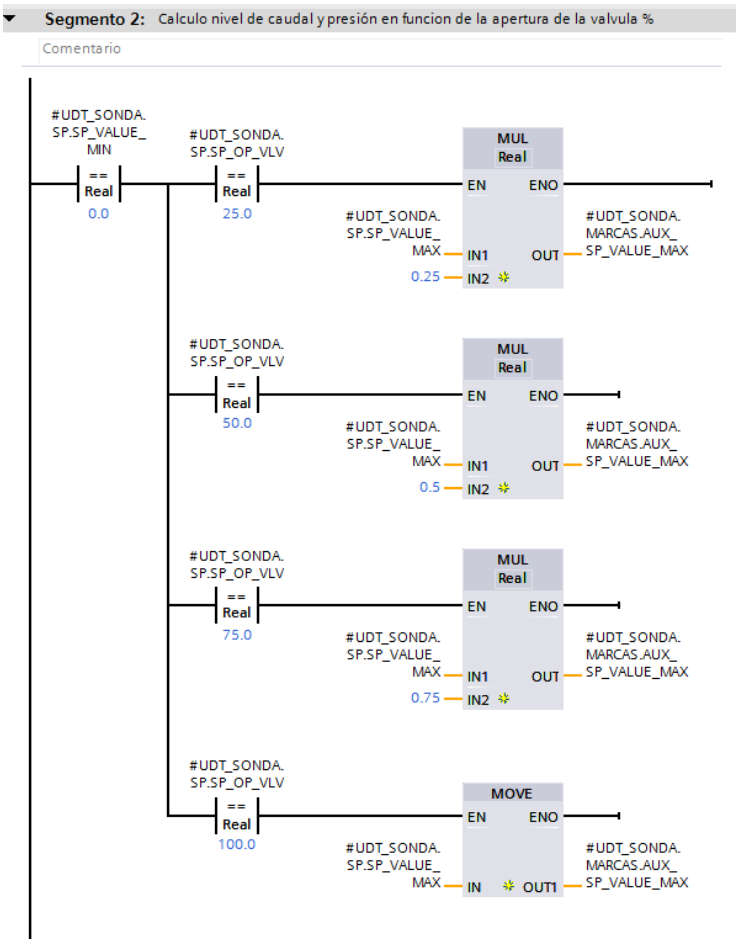


Figura 39. Segmento KOP para el cálculo del valor de consigna de una sonda de presión, en función del nivel de apertura de la válvula.

Una vez se obtiene el valor de consigna de la lectura en función del nivel de apertura de la válvula, se multiplica por un factor para reducirlo o amplificarlo en función de la velocidad de giro de impulsión de la bomba.

Como se dijo previamente, este factor es directamente proporcional para las sondas de caudal (a más velocidad mayor lectura de caudal) pero inversamente proporcional en las de presión (a mayor velocidad menor lectura de presión).

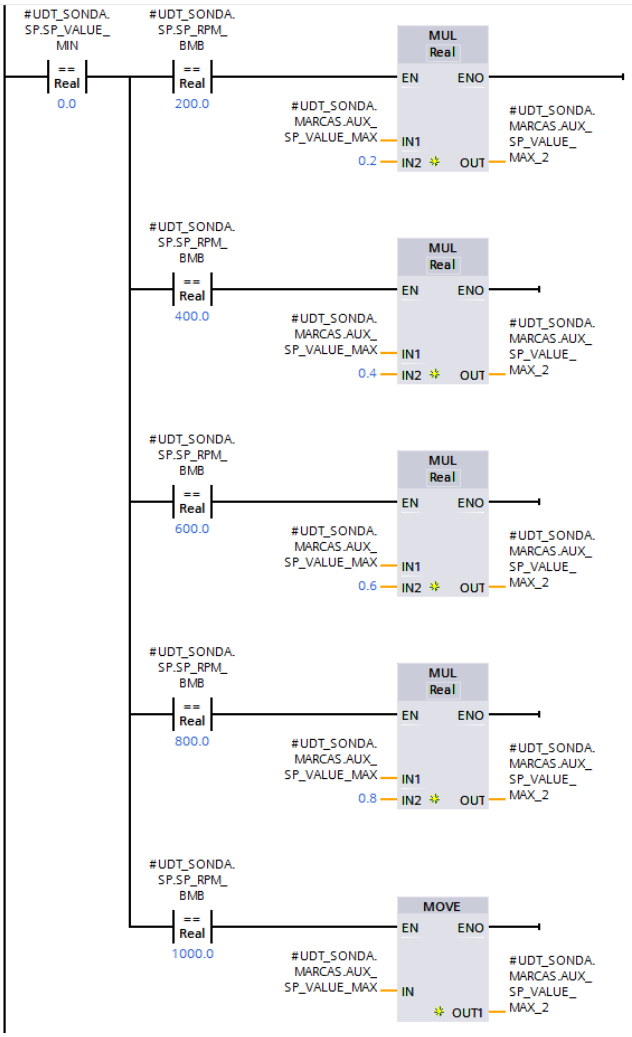


Figura 40. Segmento KOP para el cálculo del valor de consigna de una sonda de presión, en función de la velocidad de giro de la bomba.

Una vez se determina el rango de valores de lectura con la que la sonda debe operar, se calcula el paso de incremento (AUX_INCR) para que la sonda el nivel máximo pertinente.

Para ello se calcula la diferencia entre máximo y mínimo y se divide entre el tiempo deseado.

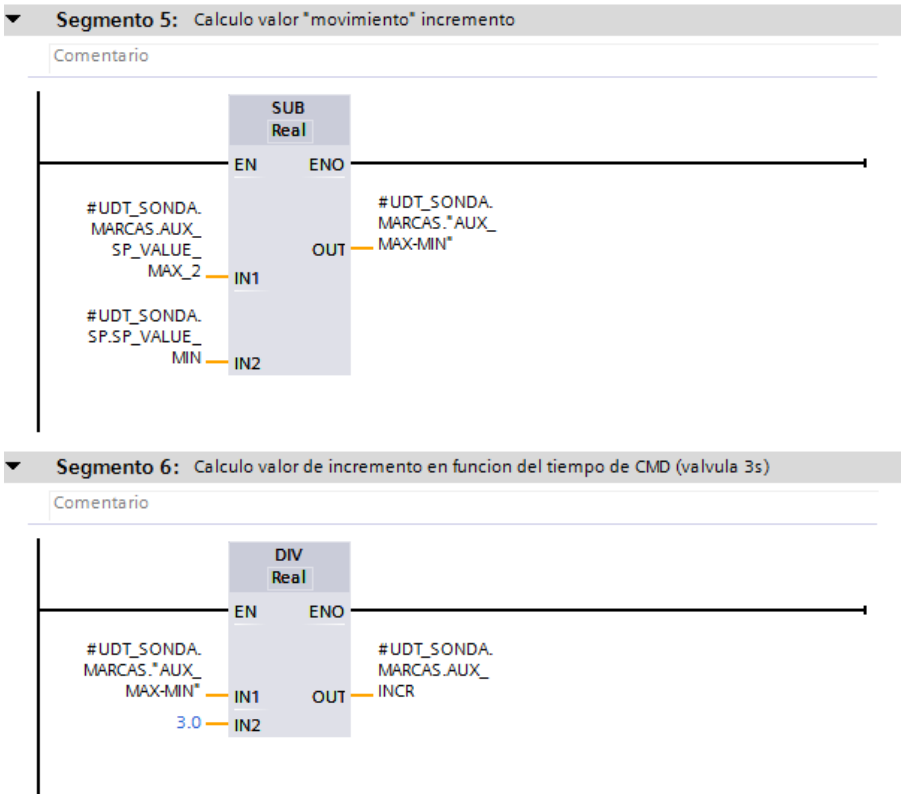


Figura 41. Segmentos KOP para el cálculo del valor de paso de incremento para la lectura de una sonda.

Por otro lado, en caso de tener que disminuir el valor de lectura, se debe calcular la "distancia" desde el valor de consigna actual al nuevo valor de consigna y dividirlo entre el tiempo deseado para calcular el paso de decremento (AUX_DECR) con el que deberá trabajar la sonda.

Para ello se guarda el valor de lectura actual cuando se detecta la orden de paro de la bomba (el cual se asocia al estado de decremento ST_DEC) y se calcula el paso de decremento dividiendo por el tiempo deseado la diferencia entre el valor actual y el mínimo.

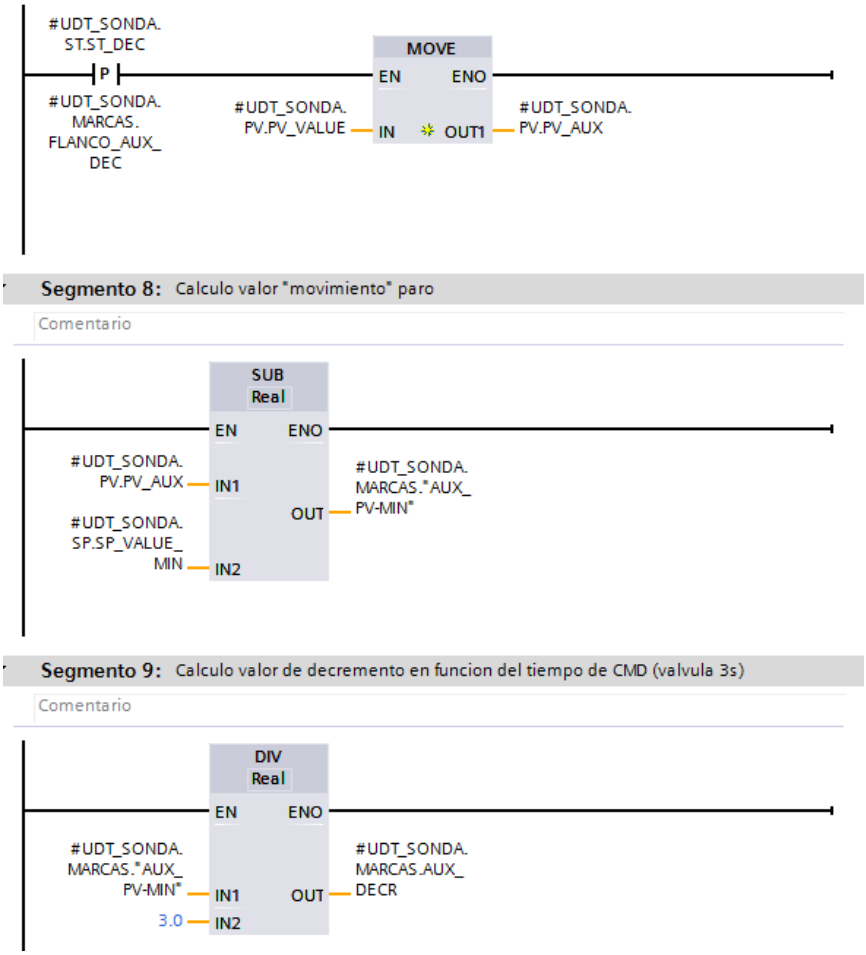


Figura 42. Segmentos KOP para el cálculo del valor de paso de decremento para la lectura de una sonda.

Por último, una vez se han calculado los pasos de incremento o decremento, se procede a incrementar, decrementar o hacer oscilar el valor de igual manera que se hacía con los motores, en función de los diferentes estados que puede haber:

- ST_INC: Asociado al comando de marcha de una bomba, pasa del valor mínimo al máximo.
- ST_DEC: Asociado al comando de paro de una bomba, pasa del valor máximo al mínimo.
- ST_OSC_MARCHA: Asociado al estado de marcha de una bomba, oscila entre el valor máximo y valores cercanos al mismo.
- ST_OSC_PARO: Asociado al estado de paro de una bomba, oscila entre el valor mínimo y valores cercanos al mismo.

4.5.2.6. Depósitos

Para la simulación del comportamiento de los depósitos, cabe distinguir dos partes: la carga y la descarga.

La carga consistirá en sumar un determinado valor (INCR_CARGA) al valor de proceso a mostrar en función del nivel de apertura de la válvula de carga y su bomba de impulsión. Mientras que la descarga restará un determinado valor (INCR_DESCARGA) en vez de sumarlo.

De esta manera, en caso de que los dispositivos de carga y descarga operen con los mismos parámetros, el nivel del depósito quedará estable sin mostrar ninguna variación en su valor de proceso.

A continuación se adjuntan figuras que ilustran el código para realizar la carga, en el caso de la descarga sería simplemente emplear las variables de descarga en vez de las de carga y restar en vez de sumar.

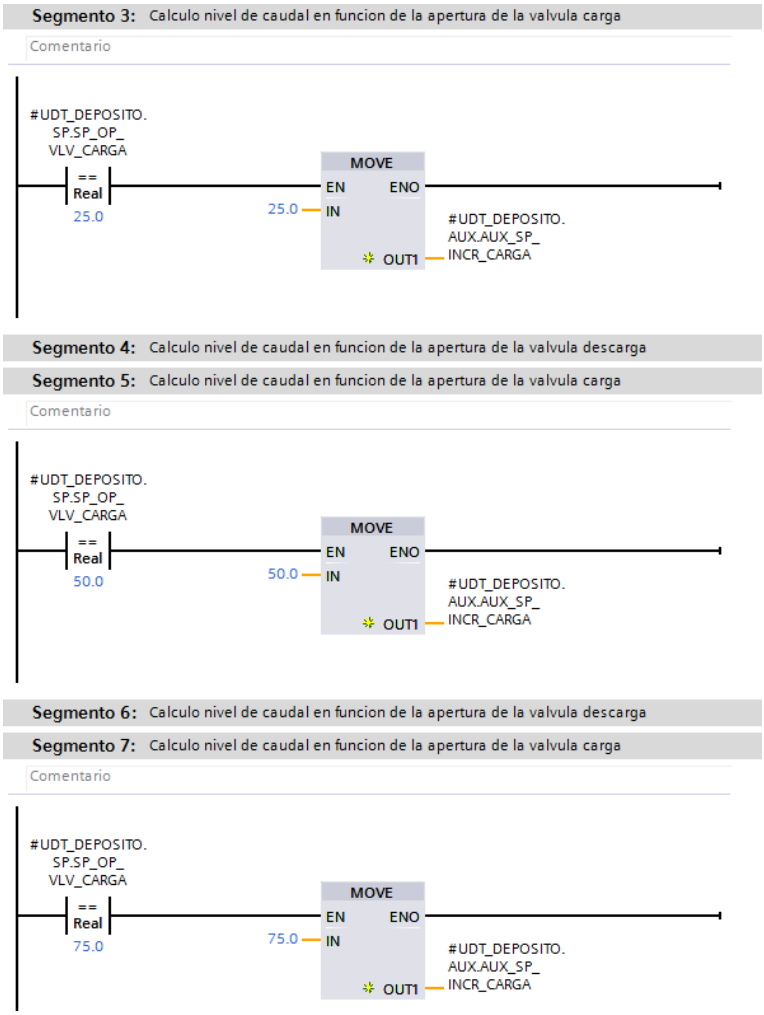


Figura 43. Segmentos KOP para el cálculo del valor de incremento de carga del depósito según el nivel de apertura de la válvula de carga.

Por otro lado, de igual forma que con las sondas, el valor de incremento se amplifica por un factor de 1 a 5 en función de la velocidad de impulsión de la bomba (de 200 rpm a 1000 rpm).

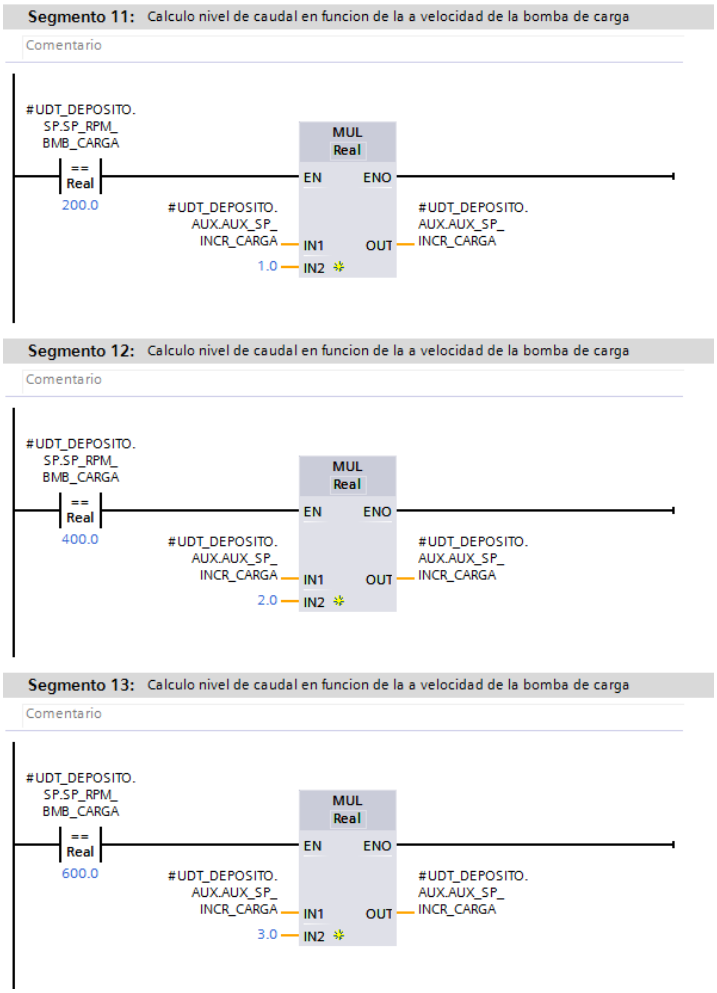


Figura 44. Segmentos KOP para el cálculo del valor de incremento de carga del depósito según la velocidad de impulsión de la bomba.

Una vez calculados los valores de incremento y decremento, estos se van sumando o restando del valor de proceso (PV_VALUE) de igual manera que en los motores en función de los siguientes comandos:

- Comando de carga (CMD_CARGA): Asociado al estado de marcha de la bomba de carga.
- Comando de descarga (CMD_DESCARGA): Asociado al estado de marcha de la bomba de descarga.

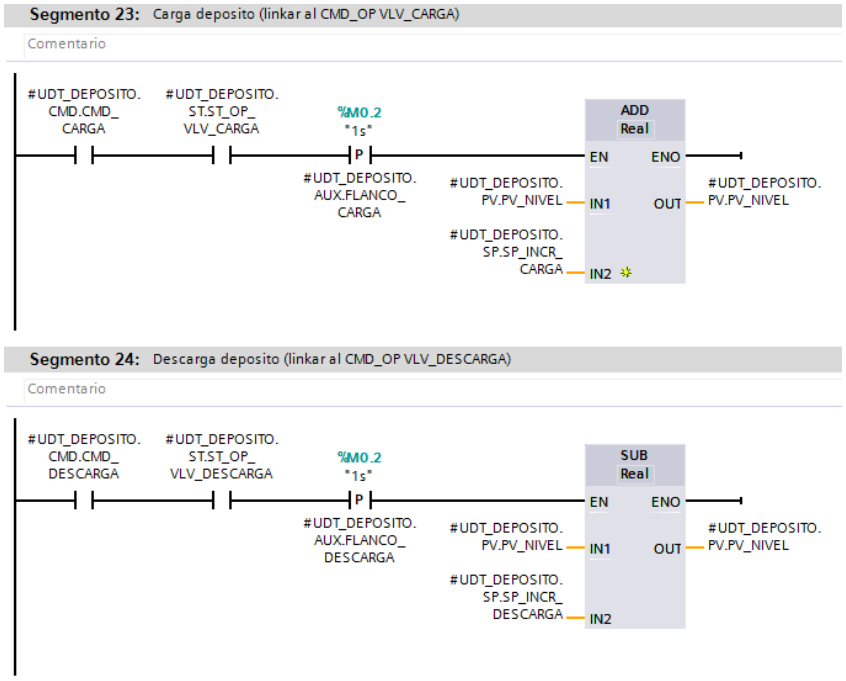


Figura 45. Segmentos KOP para el incremento/decremento del valor de proceso del nivel de llenado del depósito.

Para finalizar con el apartado de los depósitos, se añaden unos segmentos para pasar el valor de proceso del nivel de llenado a porcentaje. Para ello se elabora una simple relación con el nivel de capacidad máxima del depósito (SP_NIVEL_LLENO).

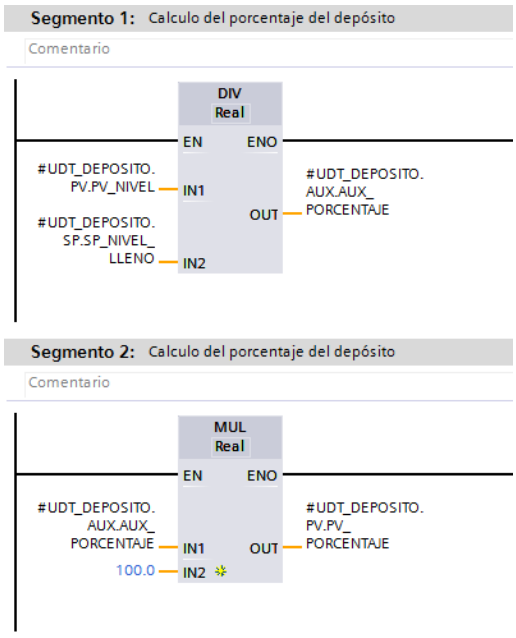


Figura 46. Segmentos KOP para el cálculo del porcentaje de llenado del depósito

4.5.2.7. Interacciones entre elementos

Finalmente, una vez explicada la manera en que se realiza la simulación interna de los dispositivos, queda explicar la forma con la cual interactúan entre ellos.

Para ello, como se dijo previamente, se recurre a los bloques de función (FB) de las secuencias donde se hacen las asociaciones necesarias.

Dichas asociaciones son las citadas en los puntos previos sobre la simulación de los dispositivos, es decir: sondas con válvulas y bombas; depósitos con válvulas y bombas de carga y descarga.

A continuación se aportan ejemplos del lenguaje SCL de dichas asociaciones.

```
//Links para las sondas
//Orden refresco SP
"DB_SONDA_CAUDAL".EMB_TC_08.PET.OR_REFRESH_SP := "DB_VLV".EMB_VLV_08.PET.OR_REFRESH_SP_OP OR "DB_MOTOR".EMB_BMB_08.PET.OR_REFRESH_SP;
"DB_SONDA_PRESION".EMB_TP_08.PET.OR_REFRESH_SP := "DB_VLV".EMB_VLV_08.PET.OR_REFRESH_SP_OP OR "DB_MOTOR".EMB_BMB_08.PET.OR_REFRESH_SP;
//SP apertura VLV
"DB_SONDA_CAUDAL".EMB_TC_08.SP.SP_OP_VLV := "DB_VLV".EMB_VLV_08.SP.SP_OP;
"DB_SONDA_PRESION".EMB_TP_08.SP.SP_OP_VLV := "DB_VLV".EMB_VLV_08.SP.SP_OP;
//SP rpm BMB
"DB_SONDA_CAUDAL".EMB_TC_08.SP.SP_RPM_BMB := "DB_MOTOR".EMB_BMB_08.SP.SP_VALUE;
"DB_SONDA_PRESION".EMB_TP_08.SP.SP_RPM_BMB := "DB_MOTOR".EMB_BMB_08.SP.SP_VALUE;
//Estado aperttura VLV (true/false) SOLO NECESARIO EN SONDAS DE PRESION!
"DB_SONDA_PRESION".EMB_TP_08.ST.ST_OP_VLV := "DB_VLV".EMB_VLV_08.ST.ST_OP;
//Estados incrementar/decrementar/oscilar marcha/ ((oscilar PARO => SOLO en SONDAS de PRESION!))
"DB_SONDA_CAUDAL".EMB_TC_08.ST.ST_INC := "DB_MOTOR".EMB_BMB_08.CMD.CMD_MARCHA;
"DB_SONDA_CAUDAL".EMB_TC_08.ST.ST_OSC_MARCHA := "DB_MOTOR".EMB_BMB_08.ST.ST_MARCHA;
"DB_SONDA_CAUDAL".EMB_TC_08.ST.ST_DEC := "DB_MOTOR".EMB_BMB_08.CMD.CMD_PARO;
"DB_SONDA_CAUDAL".EMB_TC_08.ST.ST_PARO := "DB_MOTOR".EMB_BMB_08.ST.ST_PARO;
"DB_SONDA_PRESION".EMB_TP_08.ST.ST_INC := "DB_MOTOR".EMB_BMB_08.CMD.CMD_MARCHA;
"DB_SONDA_PRESION".EMB_TP_08.ST.ST_OSC_MARCHA := "DB_MOTOR".EMB_BMB_08.ST.ST_MARCHA;
"DB_SONDA_PRESION".EMB_TP_08.ST.ST_OSC_PARO := "DB_MOTOR".EMB_BMB_08.ST.ST_PARO;
"DB_SONDA_PRESION".EMB_TP_08.ST.ST_DEC := "DB_MOTOR".EMB_BMB_08.CMD.CMD_PARO;
```

Figura 47. Código SCL para las distintas asociaciones para la simulación de las sondas.

```
//Links para depósitos
//Orden refresco SP apertura VLV carga
"DB_DEPOSITO".EMB_DEP_06."OR".OR_REFRESH_SP_CARGA := "DB_VLV".EMB_VLV_08.PET.OR_REFRESH_SP_OP OR "DB_MOTOR".EMB_BMB_08.PET.OR_REFRESH_SP;
//Estado apertura VLV carga/descarga
"DB_DEPOSITO".EMB_DEP_06.ST.ST_OP_VLV_CARGA := "DB_VLV".EMB_VLV_08.ST.ST_OP;
//SP apertura valvula carga/descarga
"DB_DEPOSITO".EMB_DEP_06.SP.SP_OP_VLV_CARGA := "DB_VLV".EMB_VLV_08.SP.SP_OP;
//SP velocidad bomba carga/descarga
"DB_DEPOSITO".EMB_DEP_06.SP.SP_RPM_BMB_CARGA := "DB_MOTOR".EMB_BMB_08.SP.SP_VALUE;
//Estado carga
"DB_DEPOSITO".EMB_DEP_06.CMD.CMD_CARGA := "DB_MOTOR".EMB_BMB_08.ST.ST_MARCHA OR "DB_MOTOR".EMB_BMB_08.CMD.CMD_MARCHA;
```

Figura 48. Código SCL para las distintas asociaciones para la simulación de los depósitos.

4.6. Programa del controlador

4.6.1. Estructura del programa

El programa implementado en el controlador se caracteriza por presentar cuatro niveles de llamadas.

En primer lugar y ejecutándose de forma cíclica se tiene el bloque de organización *main* (OB1), desde el cual se realizan las llamadas de manera individual a los tres bloques de función que se encargan de controlar las etapas de evaporación, mezcla y embotellado respectivamente.

Estos bloques de función, ejecutan la lógica necesaria para realizar el control automático de las etapas y realizan la llamada a las funciones de los dispositivos que participen en dicha etapa.

Como último nivel, las funciones de dispositivos llaman a su vez a funciones auxiliares (tool) que se encargan de realizar tareas varias que se detallarán en los próximos apartados, siendo las más destacables, realizar temporizaciones, cuentas o asignaciones de bits en palabras.

Para ilustrar la estructura del programa se adjunta una figura que muestra la jerarquía de llamadas que realiza el programa del autómata.

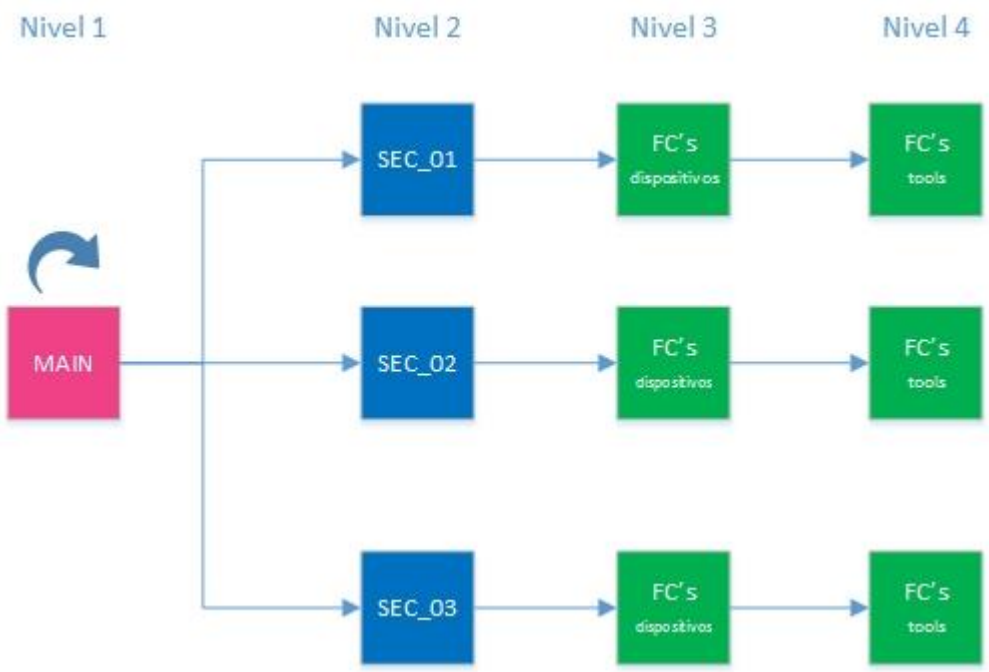


Figura 49. Esquema de la jerarquía de llamadas del programa del controlador.

4.6.2. Definición del tipo de datos

Los tipos de datos para las diversas variables son los que se muestran en el apartado *4.4. Definición del archivo de intercambio PLC – SCADA.*

Sin embargo, cabe destacar el empleo de tipos de variable definidas por el usuario (UDT's) que permiten crear patrones mediante la indexación de variables con el fin de poder emplear este patrón varias veces para elementos del mismo tipo.

En el proyecto, lo que se hace es crear una función para cada tipo de dispositivo. Como se explicó previamente, las funciones carecen de memoria interna, por ello se requerirá crear una instancia en un bloque de datos (DB) por cada elemento de la función.

Como se ve en la siguiente figura, en la función de la válvula solamente hay una variable del tipo entrada-salida creada (UDT_VLV) y es del tipo "UDT_VALVULA".

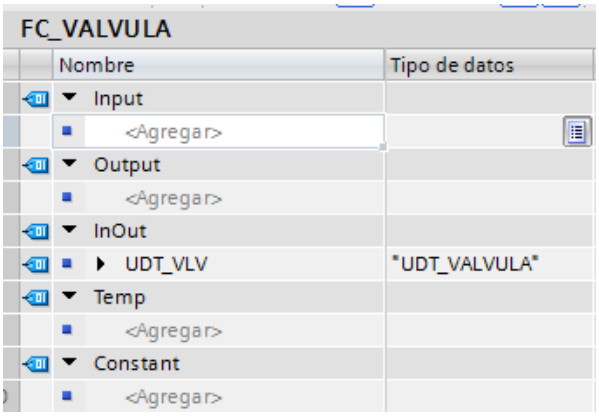


Figura 50. Variables definidas para la función (FC) del objeto válvula.

Este tipo de datos se define en el apartado de "Tipos de datos PLC", pudiendo asignar las diferentes variables que formaran parte del tipo creado.

Figura 51. Definición de las variables que componen la UDT del objeto válvula.

Para cada tipo de dispositivo se crea una UDT propia.

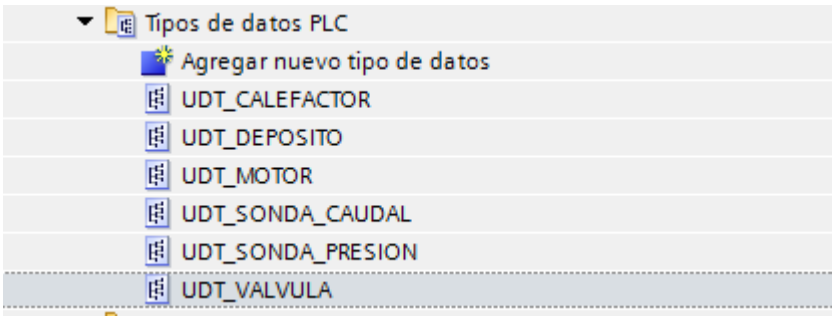


Figura 52. UDT's definidas en el proyecto.

Llegados a este punto, tenemos una función que define una serie de variables en función de la UDT que esté definida en la función, sin embargo, cabe explicar cómo y dónde almacena sus datos la función.

Para ello, se debe crear una instancia de datos del tipo de UDT pertinente para cada elemento que se vaya a instanciar, es decir, en el bloque de datos de las válvulas, se tendrá una instancia del tipo "UDT_VALVULA" por cada objeto válvula que se vaya a instanciar.

DB_VLV

	Nombre	Tipo de datos	Offset
	Static		
	EVAP_VLV_00	"UDT_VALVULA"	0.0
	EVAP_VLV_01	"UDT_VALVULA"	54.0
	MEZC_VLV_07	"UDT_VALVULA"	108.0
	EVAP_VLV_03	"UDT_VALVULA"	162.0
	MEZC_VLV_04	"UDT_VALVULA"	216.0
	MEZC_VLV_05	"UDT_VALVULA"	270.0
	MEZC_VLV_06	"UDT_VALVULA"	324.0
	EMB_VLV_08	"UDT_VALVULA"	378.0

Figura 53. Bloque de datos del objeto válvula en el que se ve una instancia de datos del tipo UDT_VALVULA por cada válvula a instanciar.

Una vez se tiene la función asociada a la UDT de igual forma que el bloque de datos, se podrá llamar a dicha función asignándole la instancia de datos del DB en el que se deseen guardar los datos.

```
"FC_VALVULA"("DB_VLV".EVAP_VLV_00);
"FC_VALVULA"("DB_VLV".EVAP_VLV_01);
"FC_VALVULA"("DB_VLV".EVAP_VLV_03);
```

Figura 54. Instancias de la función válvula asignadas a diferentes instancias del bloque de datos válvula.

Para finalizar, la utilización de tipos de dato UDT puede parecer algo enrevesada en un principio, sin embargo, una vez comprendida la filosofía de utilización facilita muchísimo la programación al permitir realizar llamadas a funciones de una forma más ordenada e indexar objetos para instanciarlos sin tener que definir variables que se repitan más de una vez.

4.6.3. Lógica de control de los elementos

Antes de empezar a comentar el código empleado en los dispositivos del proyecto, cabe mencionar que la lógica y la finalidad de las funciones auxiliares (FC_TIMER, FC_PULSE, FC_AL_TRIG_GENERATOR, FC_CONTADOR_HORAS y FC_STEP) se podrán ver de forma detallada en el documento que incluye los anexos.

4.6.3.1. Válvulas

Para el objeto válvula, desde SCADA se podrá asignar mediante órdenes la consigna de nivel en porcentaje de apertura con la que la válvula de be operar.

Para ello, en función de la orden (OR_OP_25, por ejemplo) que el SCADA active se asigna un valor de 25 a 100 a la variable del valor de consigna de apertura (SP_OP).

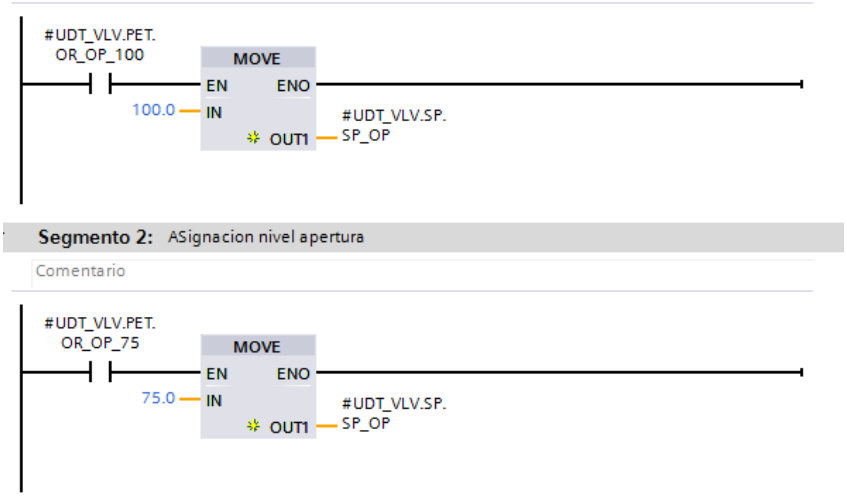


Figura 55. Asignación de la consigna de nivel de apertura en función de la orden del SCADA.

A su vez, para asegurarse que siempre haya un modo de funcionamiento activo (manual o automático) se emplean los siguientes dos segmentos:

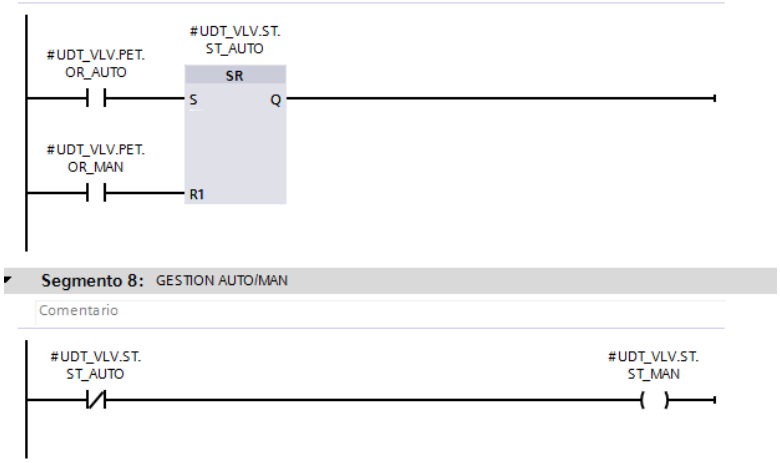


Figura 56. Gestión del modo de operación (automático o manual).

Para controlar el comando de apertura, se deben considerar tres aspectos:

- La válvula no se debe abrir si está en su posición de seguridad (AL_LCK).
- La válvula solo se abrirá si recibe orden de apertura manual estando en modo manual.
- La válvula solo se abrirá si recibe orden de apertura automática estando en modo automático.

Para ello se emplea el siguiente segmento, que activa el comando de apertura según los criterios previos y desactiva el comando de cierre por si estuviera la válvula realizando una transición al estado de cierre.

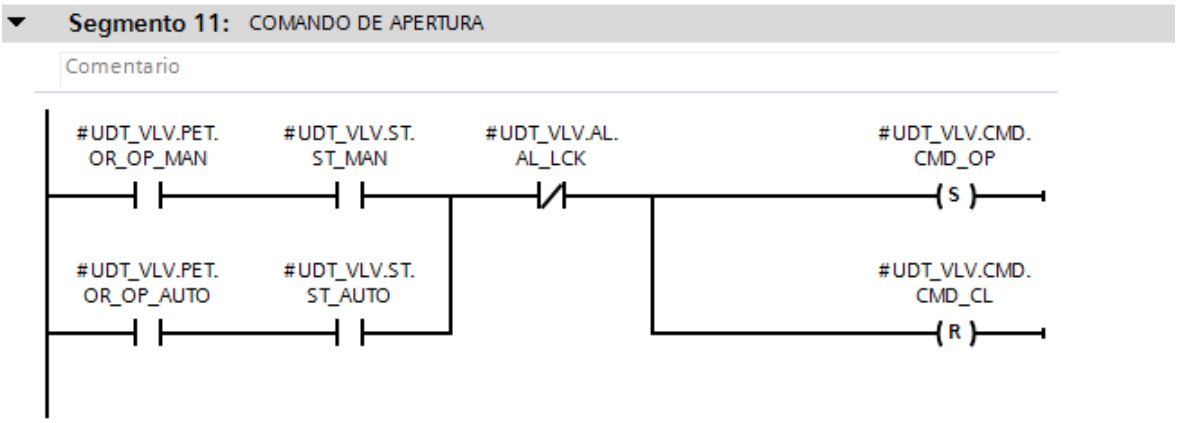


Figura 57. Gestión del comando de apertura de la válvula.

Respecto al comando de cierre, además de discernir entre órdenes en manual y automático, debe ser activado siempre que la válvula se bloquee (ST_LCK) o detecte una alarma (AL_LCK) que requiera posicionar la válvula en su posición de seguridad (cerrada).

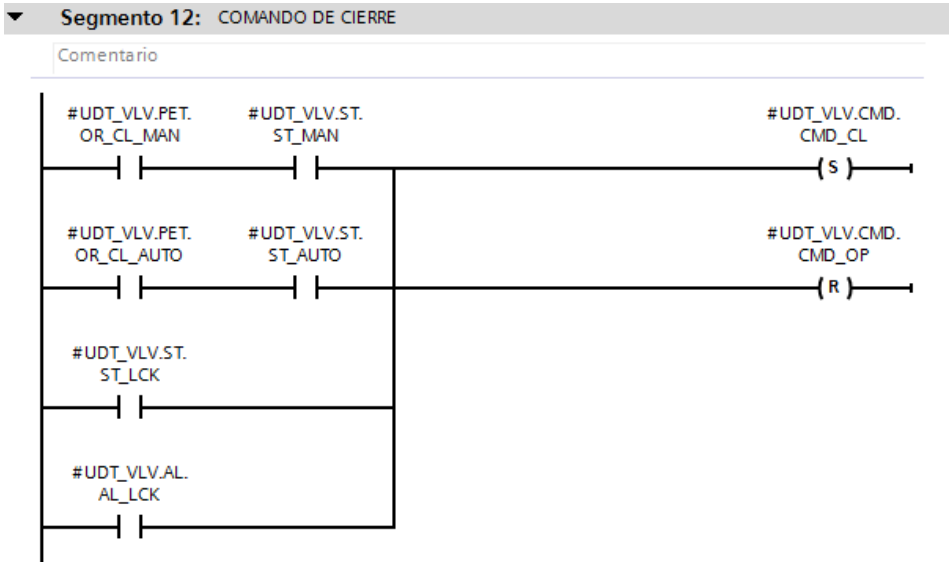


Figura 58. Gestión del comando de cierre de la válvula.

La variable AL_LCK se emplea para poner la válvula en posición de seguridad en caso de que se diera una alarma que lo requiriese. En el caso de la válvula solo se debe cerrar la válvula en caso de presentar una avería en los finales de carrera (AL_FC).

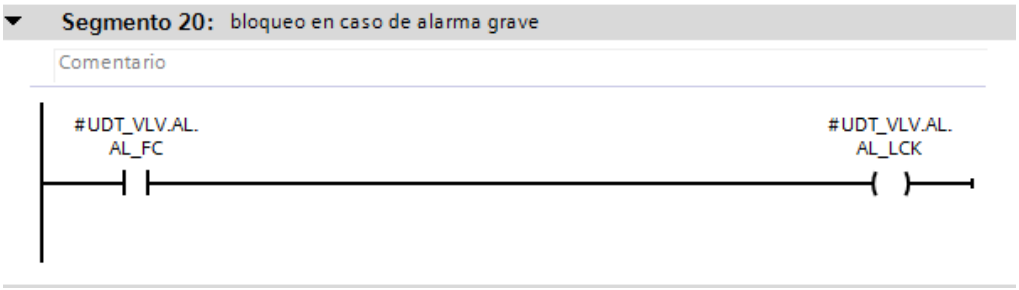


Figura 59. Gestión del comando de bloqueo por alarma grave.

El objeto válvula cuenta con un contador de horas de funcionamiento que almacena las horas, minutos y segundos en sus respectivas variables PV_HORAS_FUNC, PV_MIN_FUNC y PV_SEC_FUNC, siempre que la válvula esté abierta (ST_OP) o en transición de apertura (CMD_OP).

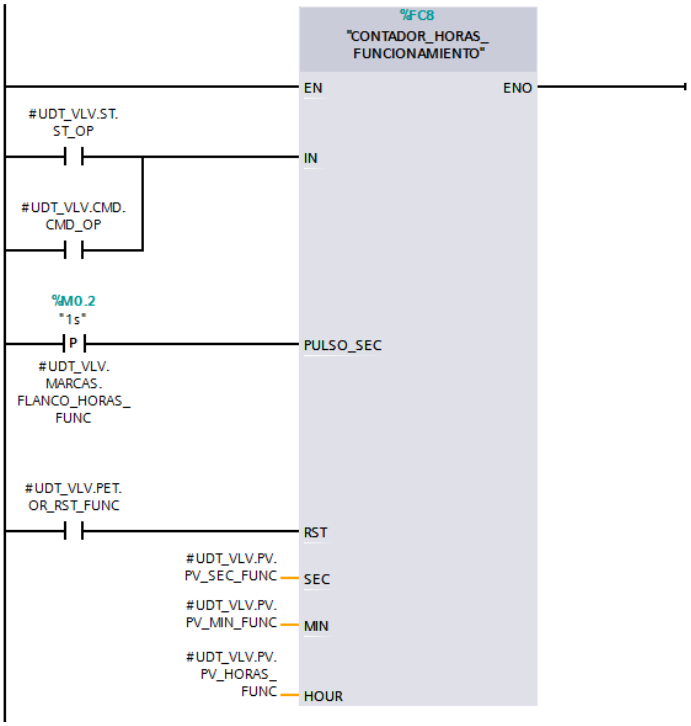


Figura 60. Contador de horas de funcionamiento de la válvula.

En los últimos segmentos de la función se resetean todas las órdenes provenientes del SCADA.

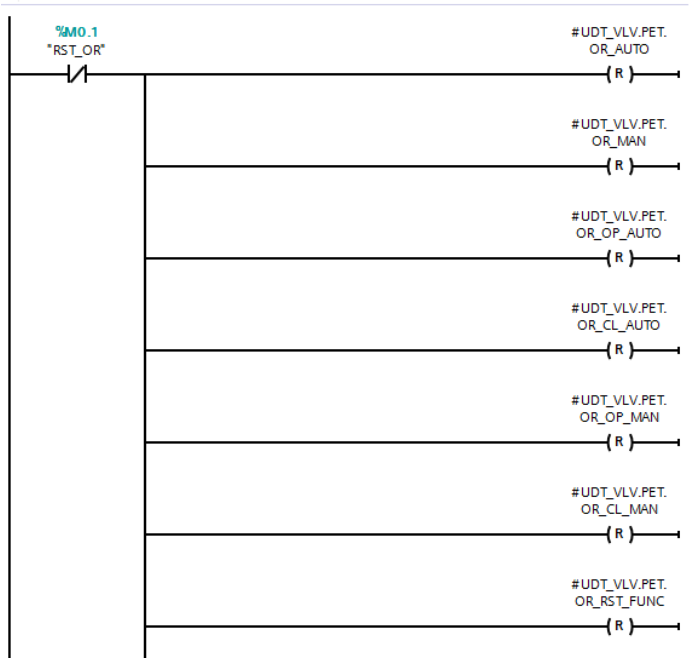


Figura 61. Reseteo de las órdenes provenientes del SCADA.

Por último, realizando una operación booleana AND con dos bytes de ceros, se resetea, tras un retraso de un segundo, la palabra que contiene el estado de reconocimiento de las alarmas (AL_ACK).

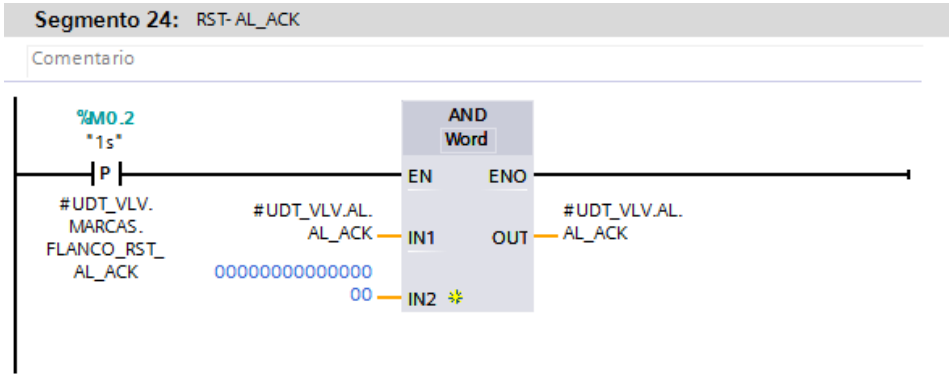


Figura 62. Reseteo de la palabra AL_ACK para el reconocimiento de alarmas.

Cabe mencionar que, el comportamiento de las palabras de gestión de alarmas AL_TRIG, AL_ST y AL_ACK, se detalla en el apartado *Diseño de la interfaz de alarmas del sistema* del punto *3.8.Programa del software SCADA.*

4.6.3.2. Sondas

Como ya se comentó en el apartado sobre el código embebido de la simulación de las sondas, la mayoría de su código tiene como fin realizar funciones de simulación.

Por lo que respecta al código de control de las sondas, se tiene de igual manera que en las válvulas, el reseteo de la palabra de reconocimiento de alarmas (AL_ACK) y la gestión de las alarmas en sí.

Desde SCADA, se puede determinar los niveles de tolerancia para las alarmas de valor muy alto (HIHI), alto (HI), bajo (LO) y muy bajo (LOLO), además del tiempo que deben darse estos valores para activar la alarma.

De esta manera si se da el caso en que el valor de proceso excede uno de los valores de tolerancia, se acciona una variable auxiliar (AUX_AL_HIHI) que inicia un temporizador. Si transcurre el tiempo determinado con el valor todavía excediendo la tolerancia, se activa el bit de alarma correspondiente (AL_HIHI).

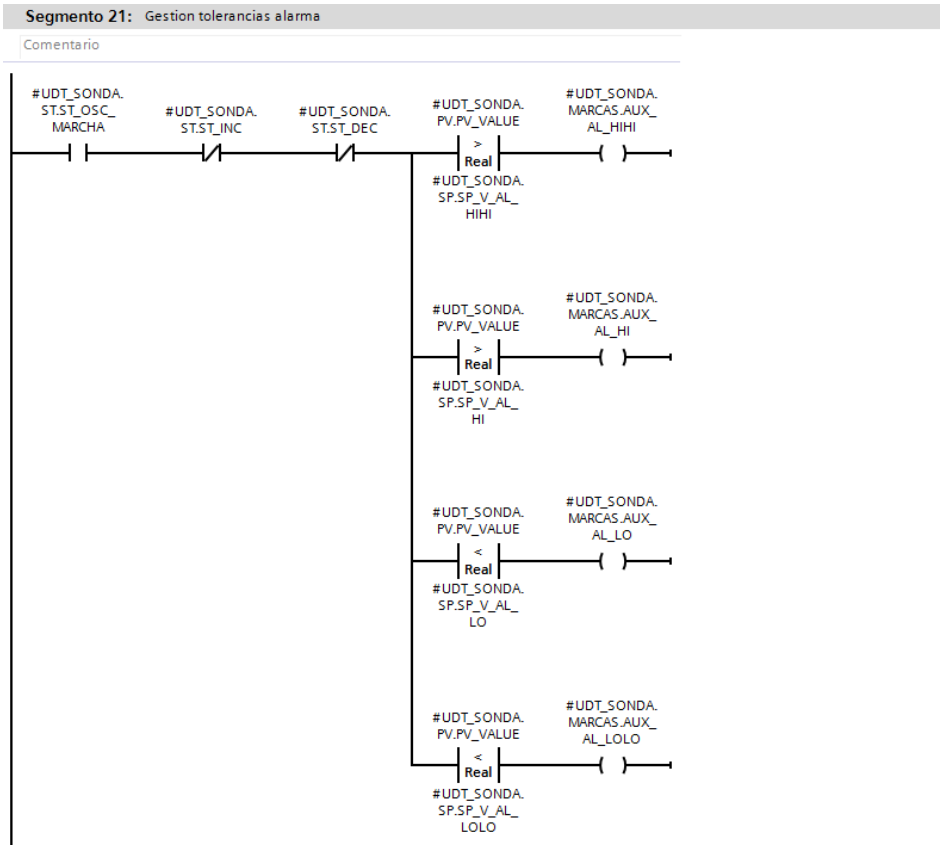


Figura 63. Gestión del estado de alarmas en función de los valores de tolerancia.

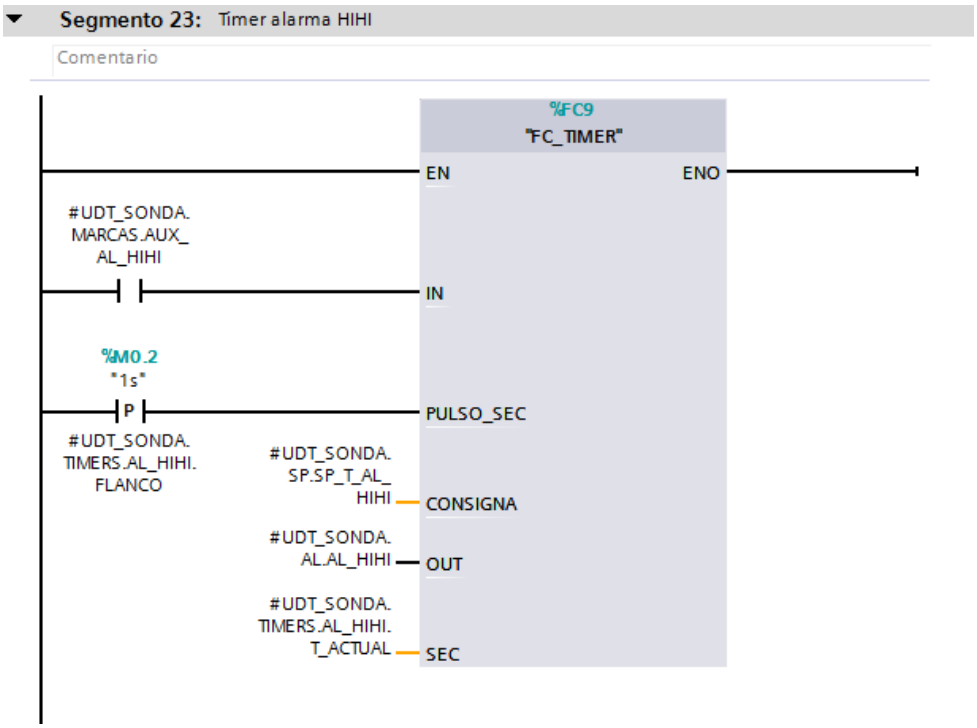


Figura 64. Temporizador de activación del bit de alarma por valor muy elevado.

4.6.3.3. Motor/Calefactor

Con el fin de no abarrotar el documento con imágenes que aporten la misma información, los segmentos para realizar el control de los objetos motor y calefactor que se asimilen a los del control de la válvula serán explicados sin adjuntar una imagen.

Sin embargo, en los anexos se podrá ver la totalidad del código empleado para realizar el control de los motores y el calefactor.

Primeramente, las funciones motor y calefactor se distinguen por el código empleado para su simulación, siendo su código de control casi idéntico a excepción de ciertos aspectos que se matizan a continuación.

De igual forma que las válvulas, estos bloques constan de una gestión del estado de operación para que siempre haya un modo de funcionamiento activo (manual o automático).

A la hora de activar el comando de marcha y paro, se tienen en cuenta los mismos criterios que se definieron a la hora de activar el comando de apertura o cierre de la válvula, dependiendo esta vez el bloqueo por alarma grave (AL_LCK) de diversas alarmas (salto del térmico, avería, revoluciones altas o revoluciones bajas para el motor y salto del térmico, temperaturas altas y temperaturas bajas para el calefactor) tal y como se muestra en la siguiente figura.

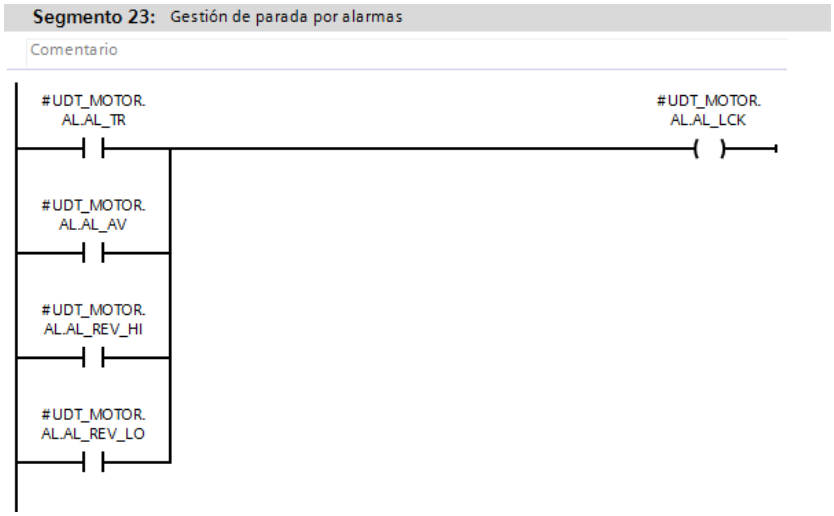


Figura 65. Gestión del comando de bloqueo por alarmas graves de la función del motor.

Tanto la función motor como calefactor, cuentan con un contador de horas de funcionamiento y desde SCADA se pueden definir diversas consignas de velocidad del motor en función de la orden que se active:

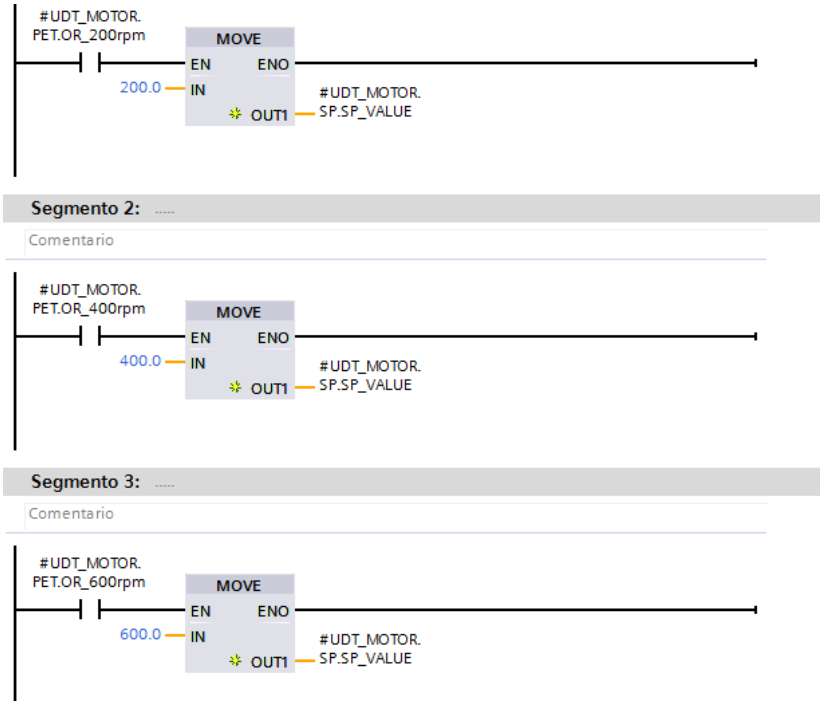


Figura 66. Asignación de la consigna de velocidad de giro en función de la orden del SCADA.

Mientras que para asignar una nueva consigna de temperatura al calefactor basta con escribir la temperatura deseada en el campo pertinente de la aplicación SCADA y dar la orden de refrescar consigna.

Todas las órdenes se resetean al final del bloque, así como la palabra de gestión de reconocimiento de alarmas (AL_ACK).

Desde SCADA también se pueden asignar los niveles de tolerancia de revoluciones altas y bajas, así como sus tiempos para activar los respectivos bits de alarma.

Para ello, de igual forma que en el caso de las sondas, se recurre a una variable auxiliar que se activa si el valor de proceso supera el valor de tolerancia, activando así el temporizador del estado de alarma.

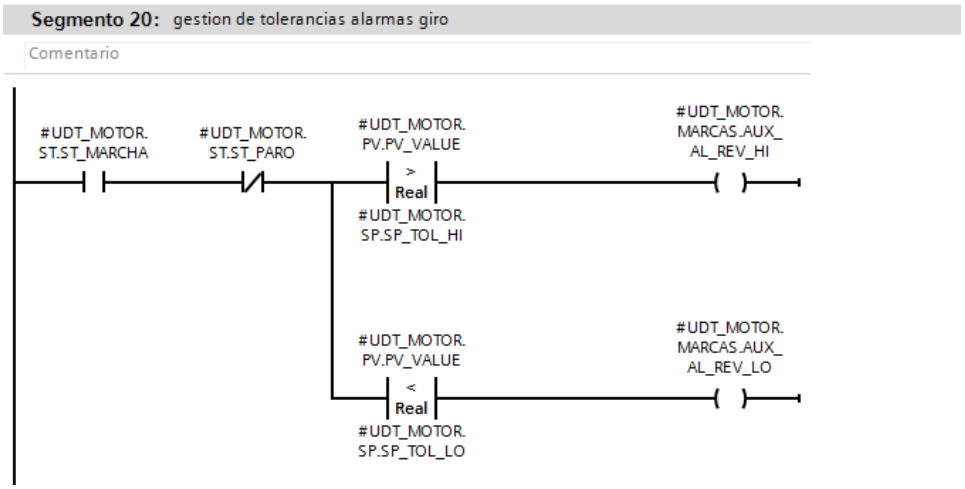


Figura 67. Gestión del estado de alarmas en función de los valores de tolerancia.

Cabe mencionar que, tal y como sucedía con las sondas, las alarmas solo pueden ser activadas estando el dispositivo en marcha y con el bit de deshabilitación de alarmas inhabilitado.

4.6.3.4. Depósitos

La función del depósito, al igual que la de las sondas, se caracteriza por primar el código de simulación sobre el código de control.

La única funcionalidad que presenta la función depósito en cuanto a control se refiere, consiste en indicar si el depósito está vacío (ST_VACIO), en nivel bajo (ST_NIVEL_LO), alto (ST_NIVEL_HI) o lleno (ST_LLENO).

Los estados de vacío y lleno se activan cuando el nivel vale cero (0%) o el máximo valor de capacidad del depósito (100%), respectivamente.

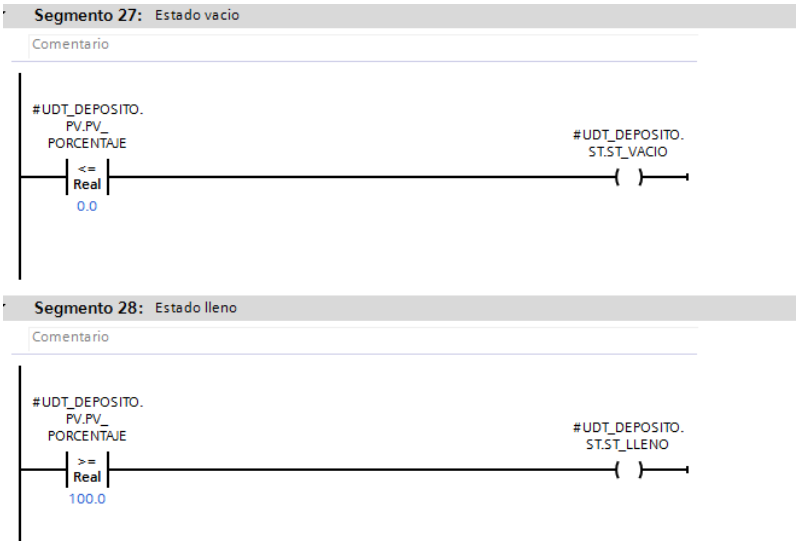


Figura 68. Gestión de los estados de vacío y lleno de la función depósito.

En cuanto a los estados intermedios, se realiza una comparación entre el valor de proceso del porcentaje de nivel del depósito (PV_PORCENTAJE) con la consigna (SP_NIVEL_HI y SP_NIVEL_LO) asignada a dichos niveles desde la aplicación SCADA, tal y como se representa en la siguiente figura:

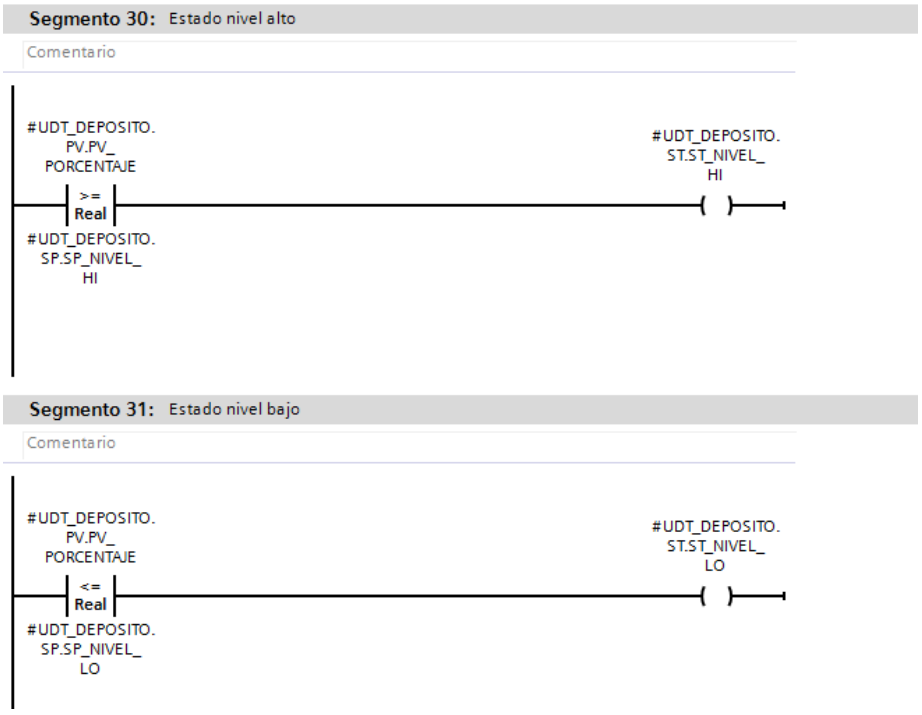


Figura 69. Gestión de los estados de nivel alto y nivel bajo de la función depósito.

4.6.4. Lógica de control de los sistemas

La lógica de control de los sistemas se ha realizado mediante bloques de función (FB) empleando código estructurado del lenguaje SCL.

Cada bloque de abarca a una etapa del proceso, teniendo así un bloque para la evaporación, un bloque para el mezclado y un bloque para el embotellado.

Indistintamente de la etapa, los tres bloques presentan una estructura interna común con el siguiente orden:

1. Definición de los sistemas de seguridad.
2. Asociaciones de elementos para la simulación.
3. Código para activar el modo automático.
4. Llamadas a funciones auxiliares.
5. Código de para activar alarmas en subprocesos.
6. Secuencia automática
7. Llamadas a funciones de dispositivos.

Lo primero que se programa en el bloque son los enclavamientos de seguridad válvula-bomba y depósito-válvula.

Básicamente se asocia el estado de cierre de las válvulas al comando de bloqueo de su bomba asociada, de tal manera que se evite que una bomba funcione en vacío.

De igual forma se asocia el estado de nivel lleno y vacío de los depósitos con el comando de bloqueo de las válvulas de carga y descarga, respectivamente. Con dicha asociación se evita hacer que un depósito rebose o que se intente extraer líquido de un depósito vacío, evitando de nuevo el funcionamiento en vacío de la bomba.

```
//Links seguridad
"DB_VLV".EVAP_VLV_03.CMD.CMD_LCK := "DB_DEPOSITO".EVAP_DEP_01.ST.ST_VACIO OR "DB_DEPOSITO".EvapMezc_DEP_02.ST.ST_LLENO;
"DB_VLV".EVAP_VLV_01.CMD.CMD_LCK := "DB_DEPOSITO".EVAP_DEP_00.ST.ST_VACIO OR "DB_DEPOSITO".EVAP_DEP_01.ST.ST_LLENO ;
"DB_VLV".EVAP_VLV_00.CMD.CMD_LCK := "DB_DEPOSITO".EVAP_DEP_00.ST.ST_LLENO;
"DB_MOTOR".EVAP_BMB_00.CMD.CMD_LCK := "DB_VLV".EVAP_VLV_00.ST.ST_CL;
"DB_MOTOR".EVAP_BMB_01.CMD.CMD_LCK := "DB_VLV".EVAP_VLV_01.ST.ST_CL;
```

Figura 70. Código para establecer las asociaciones de enclavamiento de seguridad.

En segundo lugar, se adjuntan las asociaciones necesarias para realizar la simulación, como ya se explica en el apartado *4.5.2.7. Interacciones entre elementos.* Estas asociaciones serían las necesarias para simular el comportamiento de las sondas y los depósitos en función del estado de las válvulas y bombas asociadas.

En tercer lugar, cada etapa dispone de una variable (OR_DISP_AUTO) activable desde el SCADA que permite poner todos los elementos que participen en el proceso en modo automático, para ello se dota al bloque con el siguiente código:

```
//Dispositivos en automatico
IF #OR_DISP_AUTO THEN
    "DB_CALEFACTOR".EVAP_CLF_01.PET.OR_AUTO := TRUE;
    "DB_MOTOR".EVAP_BMB_00.PET.OR_AUTO := TRUE;
    "DB_MOTOR".EVAP_BMB_01.PET.OR_AUTO := TRUE;
    "DB_MOTOR".EVAP_BMB_02.PET.OR_AUTO := TRUE;
    "DB_MOTOR".EVAP_BMB_03.PET.OR_AUTO := TRUE;
    "DB_MOTOR".EVAP_EXT_01.PET.OR_AUTO := TRUE;
    "DB_VLV".EVAP_VLV_00.PET.OR_AUTO := TRUE;
    "DB_VLV".EVAP_VLV_01.PET.OR_AUTO := TRUE;
    "DB_VLV".EVAP_VLV_03.PET.OR_AUTO := TRUE;
END_IF;
```

Figura 71. Código para activar el modo automático de los dispositivos del proceso de evaporación.

En cuarto lugar, se realizan las llamadas a funciones auxiliares como temporizadores o detectores de flanco y se asignan las variables pertinentes en cada caso.

A continuación se muestra como ejemplo la llamada al temporizador que se encarga de temporizar la duración de la fase de evaporación en función del tiempo determinado desde SCADA (SP_EVAP_TIME). Este temporizador se activa cuando se da inicio a la fase de evaporación (FASE_EVAP) y cuando concluye la temporización activa el bit que indica que la fase de evaporación ha concluido (EVAP_RDY).

Cabe destacar que el pulso se da con la variable FLANCO_TIM_EVAP, que determina, mediante la función del mismo nombre, cuándo la marca de ciclo “1s” presenta un flanco ascendente.

```
//Timer fase evaporación
"FLANCO_TIM_EVAP"(CLK:="1s",
                  Q=>#FLANCO_TIM_EVAP);

"FC_TIMER"(IN:=#FASE_EVAP,
           PULSO_SEC:=#FLANCO_TIM_EVAP,
           CONSIGNA:=#SP_EVAP_TIME,
           OUT:=#EVAP_RDY,
           SEC:=#T_ACTUAL);
```

Figura72. Llamada a la función del temporizador encargado de realizar la temporización del proceso de evaporación.

La siguiente parte del bloque tiene como objetivo, definir unas variables que indiquen que una fase de la etapa está en alarma en función de si alguno de sus dispositivos presenta una alarma grave de funcionamiento.

A modo de ejemplo, si alguno de los dispositivos de la fase de descarga de la etapa de mezclado fallase, se activaría la variable de alarma en la fase de descarga, permitiendo indicar en SCADA un fallo en dicha fase además de en el dispositivo en concreto.

```
//Subprocesos en alarma
#AL_CARGA := "DB_MOTOR".EVAP_BMB_01.AL.AL_LCK OR "DB_VLV".EVAP_VLV_01.AL.AL_LCK;
#AL_EVAP := "DB_MOTOR".EVAP_BMB_02.AL.AL_LCK OR "DB_MOTOR".EVAP_EXT_01.AL.AL_LCK OR "DB_CALEFACTOR".EVAP_CLF_01.AL.AL_LCK;
#AL_DESCARGA := "DB_MOTOR".EVAP_BMB_03.AL.AL_LCK OR "DB_VLV".EVAP_VLV_03.AL.AL_LCK;
```

Figura 73. Código para gestionar las alarmas de fases en función del estado de bloqueo por alarma grave de los dispositivos que la integran.

La sexta parte es la más importante de cada bloque, pues contiene el código de la secuencia automática de la etapa del bloque.

Para programar dichas secuencias se evita el empleo de bucles *for* o *while*, ya que en los autómatas programables estos bucles pueden hacer que el procesador del controlador se quede atascado en esas líneas de código y que una vez pasado el tiempo de ejecución del programa no se realizase bien la secuencia de: lectura de entradas, ejecución del código y escritura de salidas.

Una vez hecho este matiz, tal y como se puede ver en los anexos que acompañan al documento, todo el código se realiza mediante instrucciones *IF-THEN-ELSE* replicando así en lenguaje SCL los GRAFCET que se muestran en el punto *3.2.2. Fases del proceso*.

Por último, como ya se explicó en el apartado *4.6.2. Definición del tipo de datos* se instancian las diversas funciones de dispositivos que participen en la etapa, asociándoles una instancia propia en su respectivo bloque de datos.

```
"FC_SONDA_CAUDAL"("DB_SONDA_CAUDAL".EVAP_TC_00);
"FC_SONDA_CAUDAL"("DB_SONDA_CAUDAL".EVAP_TC_01);
"FC_SONDA_CAUDAL"("DB_SONDA_CAUDAL".EVAP_TC_02);
"FC_SONDA_CAUDAL"("DB_SONDA_CAUDAL".EVAP_TC_03);

"FC_SONDA_PRESION"("DB_SONDA_PRESION".EVAP_TP_00);
"FC_SONDA_PRESION"("DB_SONDA_PRESION".EVAP_TP_01);
"FC_SONDA_PRESION"("DB_SONDA_PRESION".EVAP_TP_02);
"FC_SONDA_PRESION"("DB_SONDA_PRESION".EVAP_TP_03);
```

Figura 74. Llamadas a las diversas instancias de las funciones de los dispositivos de la etapa.

4.7. Programa del software SCADA

4.7.1. Árbol de navegación

En el proyecto se emplea una jerarquía de navegación clara con el fin de hacer una navegación entre pantallas lo más amigable posible con el usuario que opere con la aplicación SCADA.

Indistintamente de la pantalla, en la parte superior de esta, habrá una barra de menú con la cual podremos navegar a cualquiera de las pantallas, cambiar de usuario o cerrar la aplicación (si el nivel de permisos lo permite).

Además de la navegación por el menú superior, en las pantallas de proceso, es decir, evaporación, mezclado y embotellado, se podrá navegar mediante una etiqueta adjunta al proceso anterior y al siguiente.

En la siguiente figura se muestra un diagrama que refleja la jerarquía de navegación implementada en el proyecto.

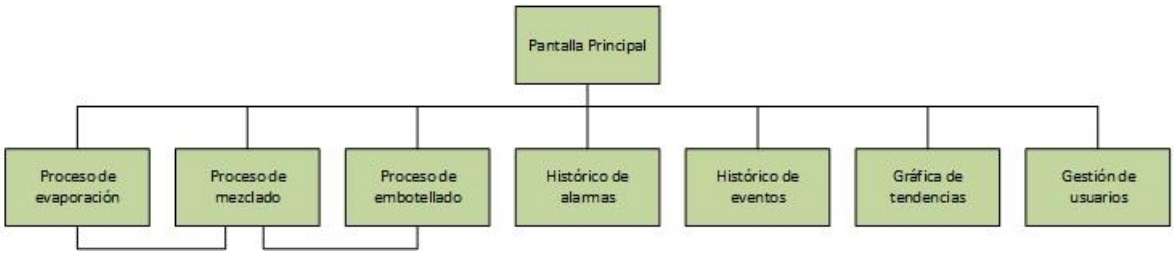


Figura 75. Jerarquía de navegación por pantallas de la aplicación SCADA.

Para diseñar la barra superior de navegación, se emplea la herramienta de WinCC "Menús y barras de herramienta", con esta herramienta se puede definir una barra de menú con sus diferentes botones, asignándoles a estos una imagen y un texto, así como el *script* que deben ejecutar al ser pulsados.

Este *script* será explicado en profundidad en el apartado dedicado a *sripts*, sin embargo, para entender lo que hace, simplemente comentar que coge el nombre de la pantalla del campo definido en "datos de usuario" y ejecuta el código con dicha pantalla, mostrándola por pantalla.

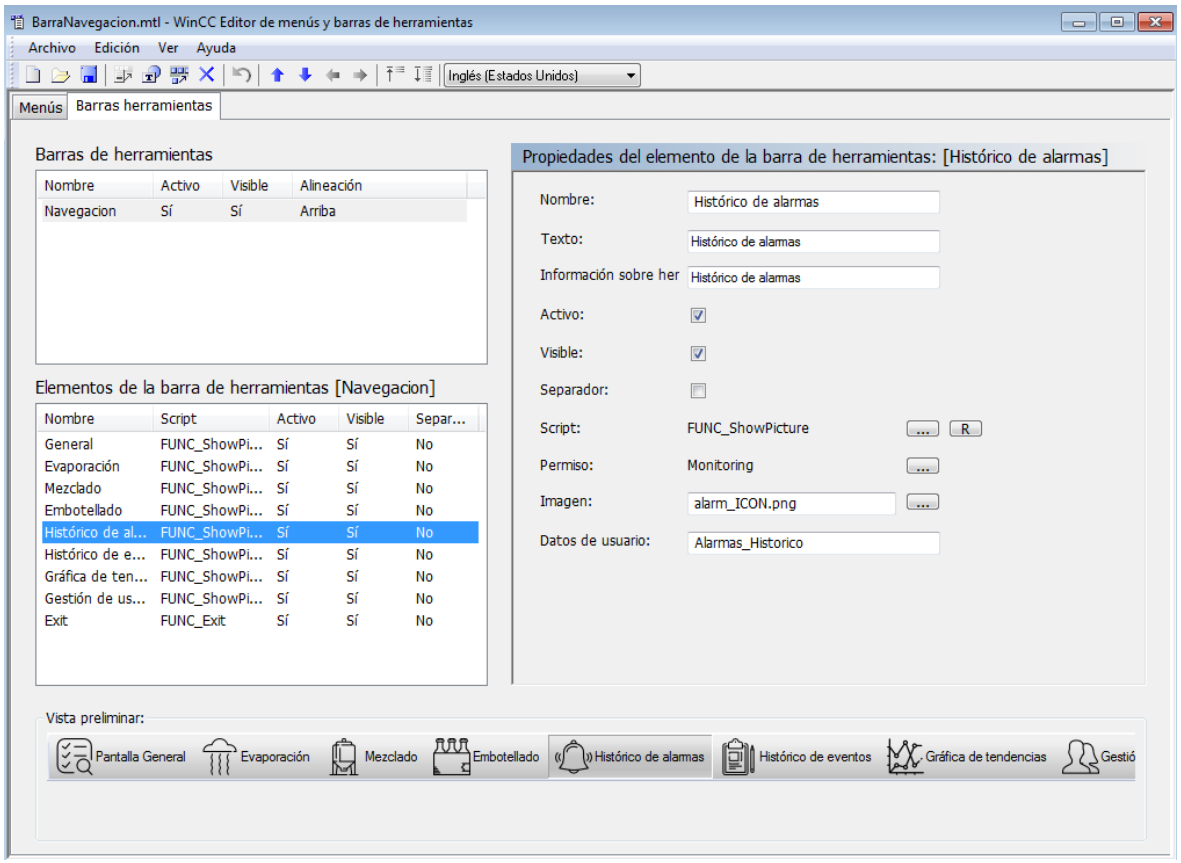


Figura 76. Editor de menús y barras de herramientas..

4.7.2. Definición de los tipos de datos

Como ya se explicó al final del apartado *4.1.3. Comunicaciones PLC* – SCADA, con el fin de simplificar al máximo posible la definición de los tipos de datos, se emplean estructuras de variables.

Como recordatorio, cabe mencionar que estas estructuras permiten crear patrones a la hora de definir tags que se van a emplear reiteradas veces para objetos del mismo tipo. Para ello, cuando se crea una nueva estructura, se determinan los elementos que formarán parte de la estructura y a continuación se definen sus variables. WinCC concatena por cada variable sus diversos elementos, definiendo así el tag correspondiente.

Además de los tags creados, se deben definir las variables de aviso, es decir las alarmas. Sin embargo, antes de explicar cómo se definen, se debe entender cómo trata WinCC a los avisos.

Como ya se vio en puntos anteriores, hay dos palabras y una palabra doble definidas, AL_TRIG, AL_ACK y AL_ST, respectivamente, que se encargan de realizar la gestión de las alarmas.

La palabra AL_TRIG (*alarm trigger*) se genera desde el PLC y aporta la información de si el suceso de alarma está activo o no, para ello, cada bit de la palabra está asociado a un suceso de alarma.

La palabra AL_ACK (*alarm acknowledge*) tiene como función realizar el reconocimiento de las alarmas, para ello desde SCADA se puede interaccionar con cada bit de la palabra para indicar que se ha reconocido la alarma asociada al bit en cuestión. Una vez se ha realizado el flanco ascendente del bit de reconocimiento desde SCADA, el PLC se encarga de volverlo a poner a 0 para así poder reconocer la alarma nuevamente en el futuro.

Por último, está la palabra doble AL_ST (*alarm state*), esta doble palabra es la que se asocia a los iconos de los diversos objetos y a los mandos de control. Aporta la información general del estado de una alarma, teniendo en cuenta sus diversas combinaciones:

- Activa y reconocida
- Activa y sin reconocer
- Inactiva y reconocida
- Inactiva y sin reconocer

Para ello, la doble palabra integra en sus bits menos significativos la palabra AL_TRIG y en sus bits más significativos la palabra AL_ACK. Sin embargo en el caso de los bits más significativos, no pone directamente la palabra AL_ACK, sino que cuando se activa una alarma, la parte asociada al reconocimiento de la palabra AL_ST se pone a nivel lógico 1 y cuando se detecta un cambio de flanco ascendente en el bit asociado de la palabra AL_ACK, el bit de la palabra AL_ST pasa a nivel lógico 0, indicando que la alarma ha sido reconocida.

En la siguiente imagen se ilustra la forma en la que las diferentes palabras para la gestión de avisos se relacionan:

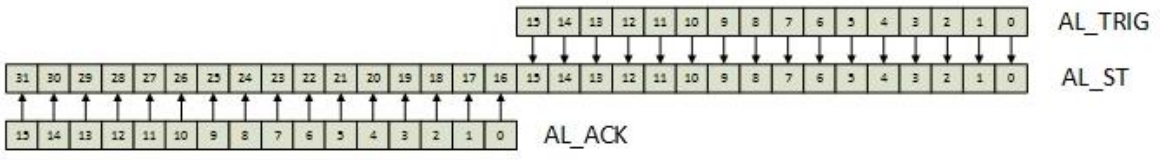


Figura 77. Asociación en WinCC entre las palabras de gestión de avisos.

Una vez explicado el modo en que WinCC gestiona los avisos, ya se puede entender el método mediante el cual se definen las variables de avisos.

En el editor *Alarm Logging*, se debe definir un aviso por cada alarma que se quiera registrar, para ello se deben definir una a una las variables de aviso, estado y acuse, estando cada una de ellas asociadas a un bit de las palabras AL_TRIG, AL_ST y AL_ACK, respectivamente.

	Número	Variable de aviso	Bit de aviso	Variable de estado	Bit de estado	Variable de acuse	Bit de acuse
15	62	EVAP_TP_00.AL_TRIG	0	EVAP_TP_00.AL_ST	0	EVAP_TP_00.AL_ACK	0
16	63	EVAP_TP_00.AL_TRIG	1	EVAP_TP_00.AL_ST	1	EVAP_TP_00.AL_ACK	1
17	64	EVAP_TP_00.AL_TRIG	2	EVAP_TP_00.AL_ST	2	EVAP_TP_00.AL_ACK	2
18	65	EVAP_TP_00.AL_TRIG	3	EVAP_TP_00.AL_ST	3	EVAP_TP_00.AL_ACK	3
19	66	EVAP_TP_00.AL_TRIG	4	EVAP_TP_00.AL_ST	4	EVAP_TP_00.AL_ACK	4

Figura 78. Definición de las variables de aviso de una sonda de presión.

4.7.3. Diseño de los iconos de los objetos

En este apartado se explicará el *modus operandi* seguido para el diseño e implementación de los iconos de los diversos objetos del proyecto, todos los diseños se podrán ver en las figuras de las pantallas de proceso.

Con el fin de obtener una aplicación SCADA totalmente personalizada, en vez de usar la galería de objetos, se diseñan (siguiendo el estándar ISA 5.5 explicado en el apartado sobre normativa) mediante la herramienta *graphics designer* de WinCC los diversos iconos, empleando el color verde para indicar activación, gris para desactivación y el color rojo para estado de fallo.

Desde dicho editor, se crea un nuevo tipo de *faceplate*, en el cual se realiza el diseño estético del icono.

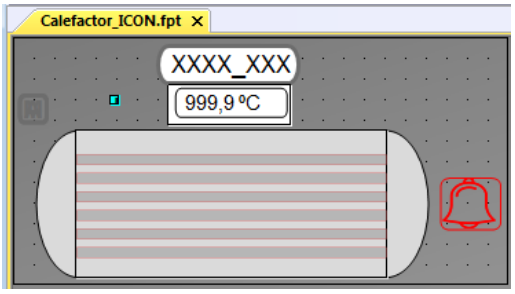


Figura 79. Diseño gráfico del icono del intercambiador de calor.

Además de la parte estética, se deben crear las propiedades de las que dispondrá el icono y las variables internas que puedan ser empleadas.

Para crear las propiedades, desde el menú de edición, se accede a "configurar tipo de *faceplate*". En la ventana que aparece se pueden crear propiedades y asociarlas a elementos gráficos del diseño,

por ejemplo, asociar la propiedad del comando de marcha (CMD_OP) con el parpadeo (FlashBackColor) del polígono que representa a la válvula en paro (GRUPO_CMD_ON).

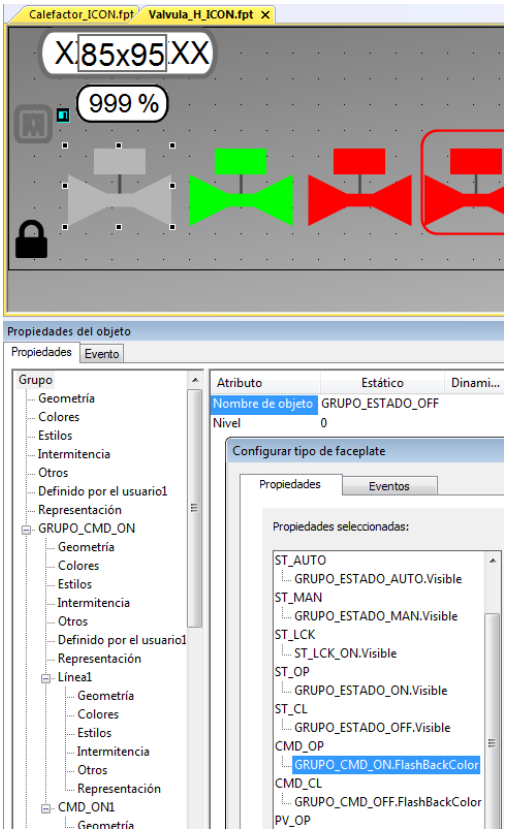


Figura 80. Asignación de las propiedades de la válvula a los elementos gráficos del icono.

Además de las asociaciones directas propiedad – elemento gráfico, los iconos que tienen la propiedad de alarmas AL_ST, requieren de un *sript* que extraiga de la palabra los bits de estado y acuse para emplearlos en diferentes elementos gráficos, es por ello que se requieren dos variables internas que se definen en el menú edición, “editar variables de faceplate”.

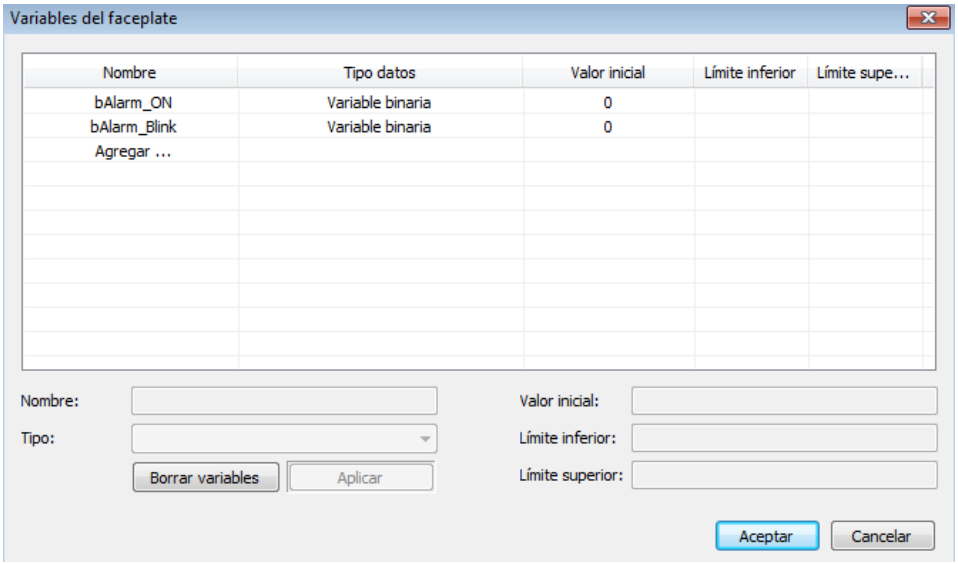


Figura 81. Herramienta de creación de variables internas de faceplate.

El *script* en cuestión se verá en detalle en el apartado de *srcipts*, sin embargo, para entender un poco la idea, lo que hace es coger los bits menos significativos de la palabra de estado (AL_ST) y mediante una función lógica OR asignarlos a la variable interna "bAlarm_On", de igual forma, los bits más significativos (asociados al acuse) se asignan a la variable interna "bAlarm_Blink".

Desde la configuración del tipo de faceplate, la propiedad AL_ST se asocia al elemento *trigger* que contiene el *script*, de tal manera que cada vez que haya un cambio de valor en la palabra de estado, se ejecutará el *script* y se actualizarán las variables internas definidas.

Por último, la asignación de las variables internas se realiza de forma directa sobre el elemento gráfico que deban animar, en el caso del agitador, por ejemplo, se asocia la variable "bAlarm_On" a la visibilidad del dibujo de la campana sin marco (ALRM_ON) y la variable "bAlarm_Blink" a la visualización intermitente del dibujo de la campana con su marco.

De esta forma, se crea el efecto de parpadeo si una alarma no está reconocida, mientras que en caso de estar solamente activa sin reconocer, no se aprecia ni el marco ni el parpadeo.

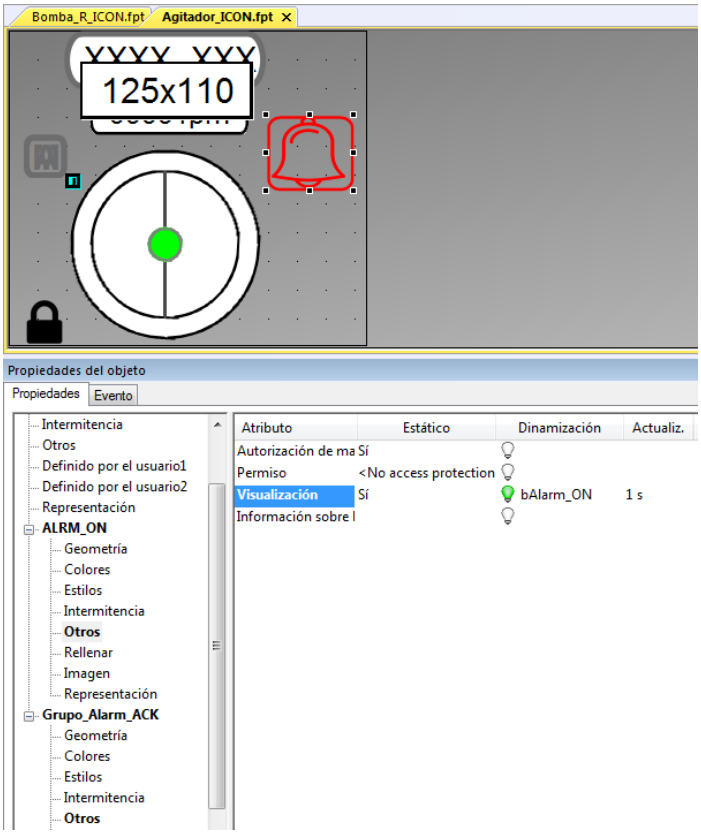


Figura 82. Asignación directa de las variables internas con los elementos gráficos a animar.

Finalmente, para instanciar los iconos en las diversas pantallas, en el menú lateral se selecciona "Instancia de faceplate", se selecciona el icono deseado, se ajusta su tamaño y se le asignan a las propiedades creadas los tags correspondientes.

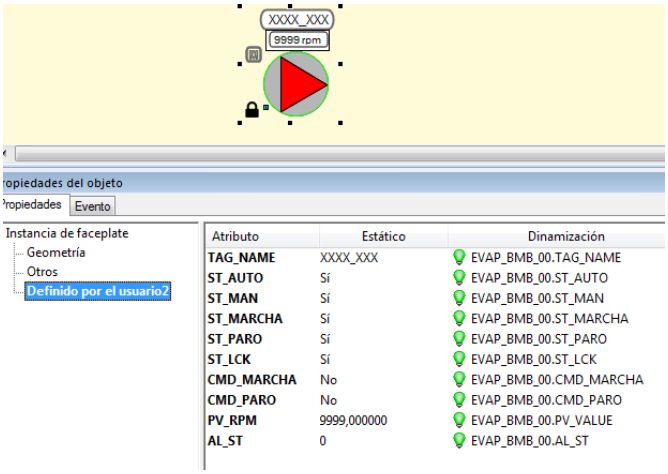


Figura 83. Asignación de los tags a las propiedades de la instancia del icono de una bomba.

4.7.4. Diseño de los mandos de control y simulación

De igual forma que con los iconos, en este apartado se desarrollará la problemática surgida a la hora de implementar los mandos de control y simulación, se especificará la solución adoptada y en los anexos que acompañan al documento se incluirá la galería de diseño de todos los mandos de control y simulación.

Para controlar y ver en detalle la información de un dispositivo, se emplean mandos de control desde los cuales se pueden dar las órdenes de funcionamiento, asignar valores de consigna, reconocer alarmas y ver información como el tiempo de funcionamiento, el estado de bloqueo, etc.

A parte del mando de control, debido al carácter simulado de la planta, cada dispositivo cuenta con un mando secundario de simulación que permite forzar valores y estados de alarma.

Cada dispositivo debe tener sus mandos de control y simulación asociados, para ello Siemens plantea el empleo de contenedores de imagen.

Estos contenedores se instancian en la pantalla en cuestión y se asocian al mando de cada dispositivo, por lo que requiere diseñar un mando por cada dispositivo y cargar las pantallas de contenedores de imagen. Esta solución resulta útil y fácil de implementar en proyectos con pocos dispositivos, sin embargo, en este proyecto resulta completamente inviable.

Para evitar esa solución tan poco optimizada, se recurre a otra un tanto más rebuscada pero que permite crear solo un mando de control por cada tipo de dispositivo.

Para ello, mediante la herramienta “editor de proyectos OS” de WinCC, se define la cantidad de pantallas físicas de las que dispondrá la aplicación SCADA, así como la resolución de las mismas.

Una vez hecho esto, en la herramienta de diseño gráfico aparecerán múltiples pantallas precedidas por el símbolo @, estas pantallas se contienen unas a otras y mediante un *script* se asocian las pantallas de proceso a la pantalla @Desk (contenida en la pantalla @1001), cuyo contenido varía conforme se ejecuta el *script*.

Es en esta pantalla, @1001, donde se disponen las ventanas de imagen que harán de contenedores para mostrar los diversos mandos de control.

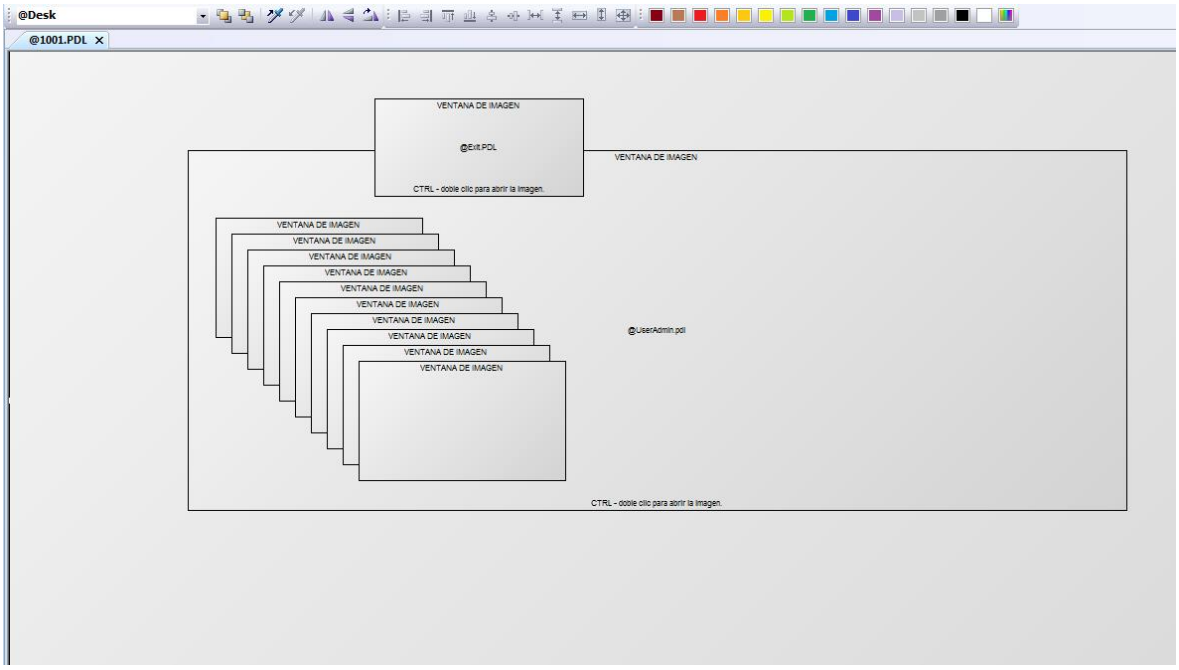


Figura 84. Ventanas de imagen contenidos en la pantalla @1001.

Una vez realizado esto, mediante un *script*, que será detallado en el apartado dedicado a los *scripts*, se asocia a cada dispositivo sus mandos, tanto el mando de control (accesible con el clic izquierdo del ratón) como el mando de simulación (accesible con el clic derecho).

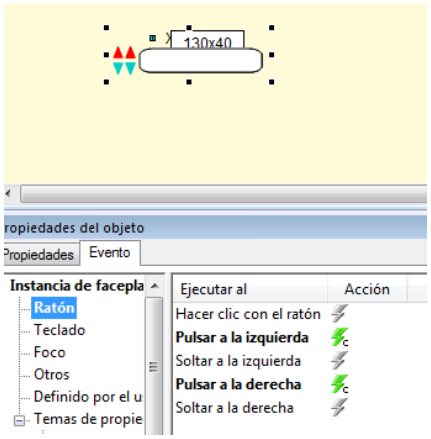


Figura 85. Asignación de los *scripts* de llamada a los mandos de control y simulación del icono de la sonda.

Este *script*, lo que hace es llamar a una de las ventanas de imagen contenidas en la pantalla @1001 y pasarle el nombre del mando que debe mostrar además de pasarle al mando en sí el nombre del icono que realiza la llamada.

Aplicando esta solución, se diseñan los mandos de control y simulación, pero en ellos no se asocian nombres de tags, sino de elementos de las estructuras de variable, ya que el *script* previamente citado,

se encarga de completar el nombre del tag concatenando al nombre del elemento el nombre del dispositivo que realiza la llamada. Es precisamente esto lo que permite diseñar un solo mando por cada tipo de elemento en vez de un mando por cada elemento en sí.

Figura 86. De izquierda a derecha, mando de control de una bomba y mando de simulación de una sonda.

Como se puede apreciar en la figura anterior, se emplea el borde punteado para los campos que contienen valores de escritura y el borde continuo para los campos que contienen valores de lectura.

4.7.5. Diseño de las pantallas de aplicación

El proyecto cuenta con una pantalla de aplicación para cada proceso y una pantalla general que aporta una visión global del estado de funcionamiento de la planta entera.

En la pantalla general, se disponen los depósitos de la planta permitiendo ver su nivel actual, también se añade un pequeño sumario de alarmas que mostrará solamente las alarmas activas.

Asimismo, la pantalla general mostrará el estado de las secuencias, pudiendo acceder a sus respectivas pantallas interactuando con el marco de la secuencia.

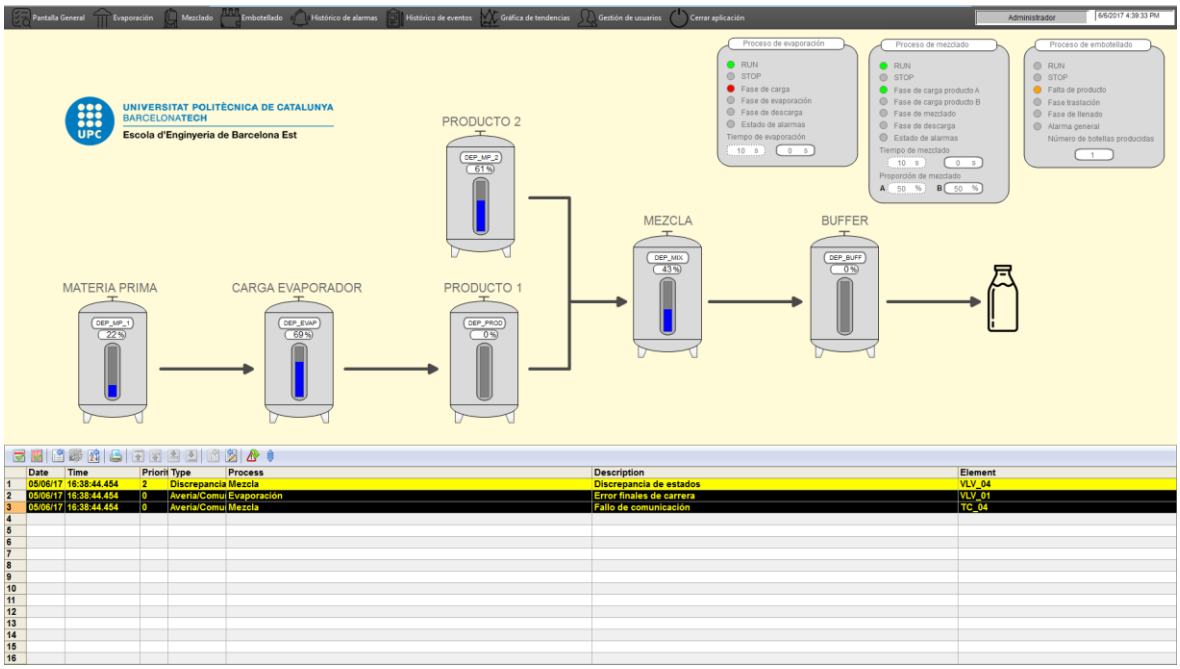


Figura 87. Pantalla general de la aplicación SCADA.

En cuanto a las pantallas de los procesos, cada una de ellas muestra el estado de la secuencia, permitiendo iniciarlas, detenerlas y cambiar sus valores de consigna. También se muestran si hay alguna alarma en alguna subfase del proceso.

Por otro lado, cuando se da una orden de marcha o paro de una secuencia, se muestra un *pop-up* para confirmar la operación.

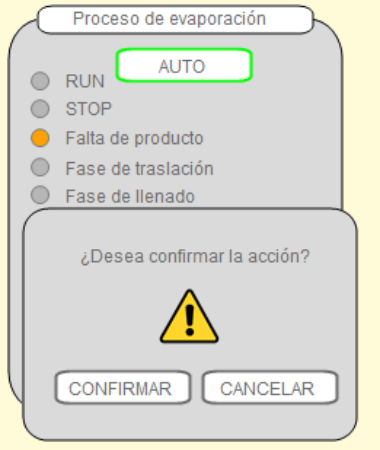


Figura 88. *Pop-Up* de doble confirmación para activar o detener los procesos automáticos..

En estas pantallas se instancian los diversos iconos de los dispositivos y se animan las tuberías para dejar claro por dónde circula líquido.

Finalmente, desde las etiquetas correspondientes, se puede navegar a las etapas del proceso anterior y siguiente.

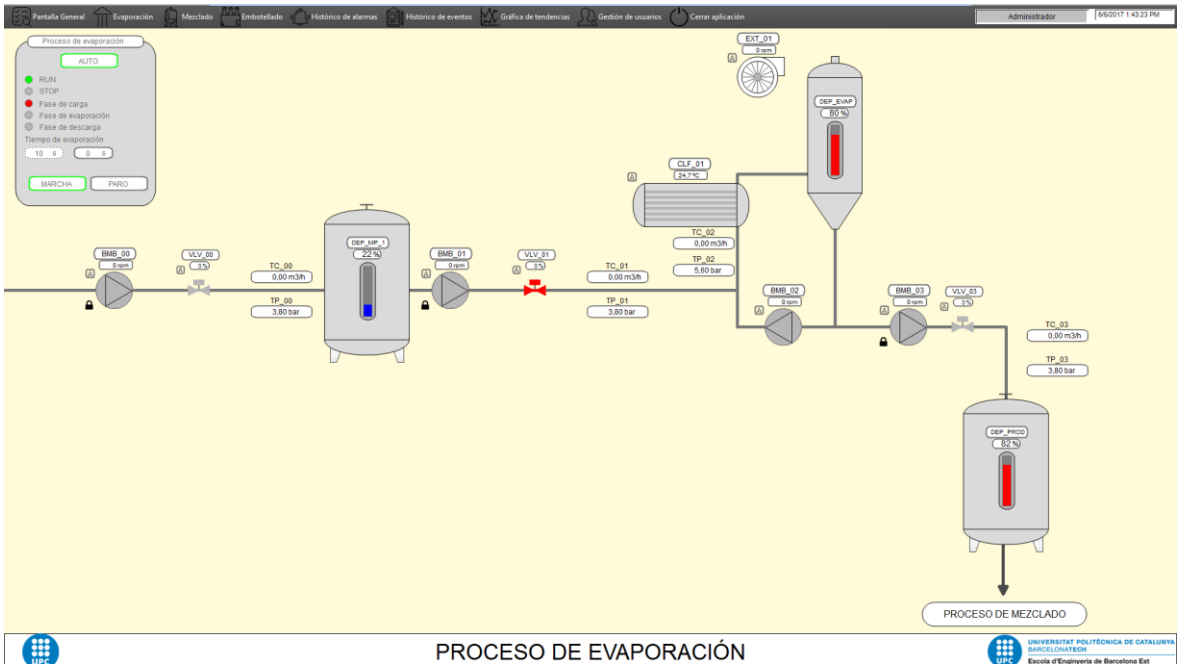


Figura 89. Pantalla del proceso de evaporación.

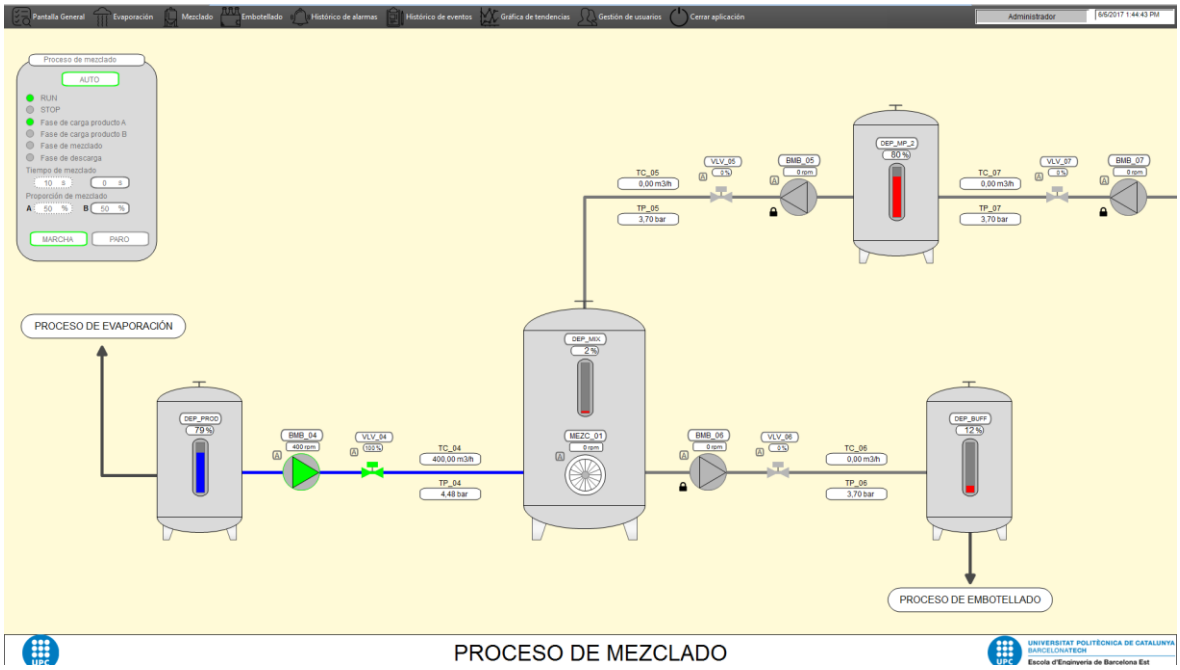


Figura 90. Pantalla del proceso de mezclado.

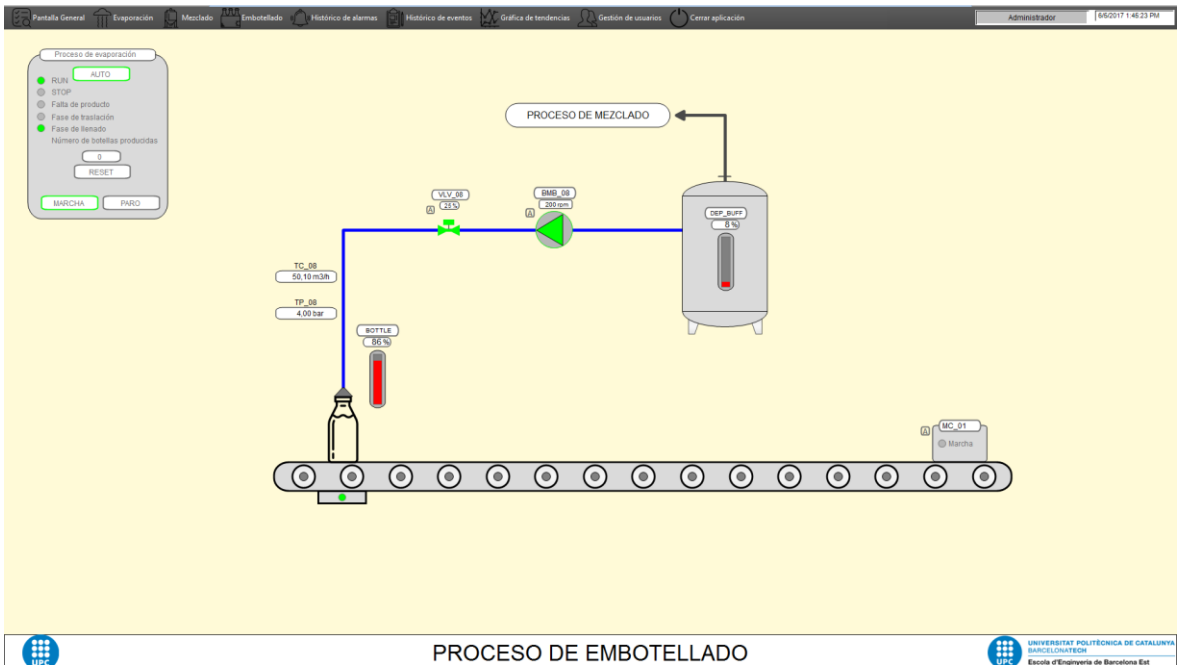


Figura 91. Pantalla del proceso de embotellado.

4.7.6. Scripts

En este apartado se explicarán los diferentes *scripts* empleados así como el motivo por el cual son necesarios.

4.7.6.1. ShowPicture

Este *script* en lenguaje VBS es el que se emplea para cargar en la ventana de imagen @Desk el contenido de la pantalla que se deba visualizar desde la barra de navegación superior.

El *script* toma como valor de entrada el nombre de la pantalla a llamar y lo guarda en la variable strPictureName. Posteriormente, coloca esta variable en el contenedor @Desk contenido en la pantalla @1001.

```
Sub FUNC_ShowPicture (Byval PictureName)

    Dim strPictureName

    strPictureName = PictureName.UserData

    HMIRuntime.Screens("@SCREEN.@WIN12:@1001").ScreenItems("@Desk").picturename = strPictureName

End Sub
```

Figura 92. Código de la función ShowPicture.

4.7.6.2. PictureOpen

Este *script* se emplea en los sumarios de alarma y permite navegar a la pantalla que contiene el dispositivo en alarma, a pesar de que su función es la misma que el *script* anterior, navegar a una pantalla, se debe programar en lenguaje C ya que es el único lenguaje que admite el gestor de alarmas.

Cuando se da clic sobre la alarma, se carga en la variable “lpszPictureName” el nombre de la pantalla definido en el gestor de avisos y con la función interna “SSMChangeWorkField” sustituye el sinóptico de proceso actual (Welcome.pdl) por el de la pantalla destino.

Loop In Alarm	Nombre de función	Parámetros de función
☑	FUNC_PictureOpen	Proceso_Evaporacion.pdl
☑	FUNC_PictureOpen	Proceso_Evaporacion.pdl
☑	FUNC_PictureOpen	Proceso_Evaporacion.pdl
☑	FUNC_PictureOpen	Proceso_Evaporacion.pdl
☑	FUNC_PictureOpen	Proceso_Mezclado.pdl
☑	FUNC_PictureOpen	Proceso_Mezclado.pdl
☑	FUNC_PictureOpen	Proceso_Evaporacion.pdl
☑	FUNC_PictureOpen	Proceso_Evaporacion.pdl
☑	FUNC_PictureOpen	Proceso_Mezclado.pdl
☑	FUNC_PictureOpen	Proceso_Mezclado.pdl
☑	FUNC_PictureOpen	Proceso_Mezclado.pdl

Figura 93. Asignación de la pantalla destino en el gestor de alarmas (*Alarm Logging*).

```
#include "apdefap.h"
void FUNC_PictureOpen(char *lpszPictureName)
{

#define PIC_1 "Welcome.Pdl"

SSMChangeWorkField(SSMGetScreen(PIC_1),lpszPictureName,TRUE);
}
```

Figura 94. Código de la función PictureOpen.

4.7.6.3. TopfieldOpen_FP

Este *script* en lenguaje C, tiene como objetivo llamar a una ventana de imagen disponible de la pantalla @1001 y cargar en ella el mando de control o simulación (PictureName) que determine el objeto que llame al *script*, de igual forma, dicho objeto le pasa la información (TagPrefix) con la que debe complementar los elementos de estructura del mando para formar el nombre de los tags con los que debe operar.

```
// Open the first TOPxx window, which is unvisible ============================================
for (i=1; i <= BST_FP_MAX; i++) {
      sprintf (szTopObjectName, "TOP%02d", i);
      // printf ("szTopObjectName: %s\r\n", szTopObjectName);

      if (GetPropBOOL(szParentPicture, szTopObjectName, "Visible" ) == FALSE) {
            SetPropChar(szParentPicture, szTopObjectName, "PictureName", lpszTopPictureName);
            SetPropChar(szParentPicture, szTopObjectName, "TagPrefix", szTagPrefix);
            SetPropBOOL(szParentPicture, szTopObjectName, "Visible", TRUE);
            break;
```

Figura 95. Código de la función TopfieldOpen_FP.

4.7.6.4. TRIGGER_QdwState

Este *script* en lenguaje VBS es el que se implementa en el elemento *trigger* de los iconos para coger los bits de alarma de la palabra doble AL_ST y asignar los relativos al estado de activación de la alarma a la variable interna "bAlarm_ON" y los relativos a la parte del acuse a la variable interna "bAlarm_Blink".

Para ello, se realiza una operación lógica AND de la palabra AL_ST ("value", ya que es el parámetro de entrada de la función) con una máscara de 32 bits puestos a 0 excepto el bit de la posición de la alarma.

Esto se realiza para todas las posiciones de la palabra que contengan información (bits 0, 1, 16 y 17 en la figura) y a posterior se asignan los resultados de las operaciones AND en la variable interna correspondiente realizando una función lógica OR.

```
Sub OutputValue_OnPropertyChanged(Byval Item, Byval value)

'Alarmas en bits 0 y 1
SmartTags("bAlarm_ON") = (value And &H00000001) Or (value And &H00000002)

'Reconocimiento en bits 16 y 17
SmartTags("bAlarm_Blink") = (value And &H00010000) Or (value And &H00020000)

End Sub
```

Figura 96. Código del *script* para gestionar las variables internas del icono de la válvula.

4.7.6.5. *Scripts* de asignación de valor

Finalmente, para dar órdenes desde la aplicación SCADA, se emplean *scripts* que se ejecutan al hacer clic que fuerzan un cierto valor en un tag.

A continuación se muestran los *scripts* empleados, tanto en lenguaje C como en VBS, para asignar valores a tags.

En el caso del lenguaje C, se emplea la función interna "SetTagWordWait":

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName)
{
SetTagWordWait("OR_200rpm",1);      //Return-Type: BOOL
SetTagWordWait("OR_REFRESH_SP",1);      //Return-Type: BOOL
}
```

Figura 97. Código del *script* en lenguaje C para activar la orden de velocidad de giro a 200 rpm y refrescar el valor de consigna.

Mientras que en el lenguaje VBS, se emplea la función interna “SmartTags”:

```
Sub OnClick(ByVal Item)
If SmartTags("SEC_01.OR_DISP_AUTO") = 1 Then

    SmartTags("SEC_01.OR_DISP_AUTO")= 0
Else
    SmartTags("SEC_01.OR_DISP_AUTO")= 1
End If
End Sub
```

Figura 98. Código del *script* en lenguaje VBS para realizar un *toggle* de la orden de activar el modo automático de los dispositivos de la secuencia de evaporación.

4.7.6.6. *Sript* de cierre de mando de control/simulación

Mediante este sencillo *script* se cierra el mando que se está visualizando en pantalla al hacer clic sobre el botón que contiene el *script*.

```
Sub OnClick(ByVal Item)

item.Parent.Parent.Visible = false

End Sub
```

Figura 99. Código del *script* en lenguaje VBS para dejar de visualizar un mando de control o simulación.

4.7.6.7. *Script* para cerrar la aplicación

Este *script* está asociado al botón de la barra de menú superior destinado a cerrar la aplicación SCADA.

Su funcionamiento es muy simple, pues simplemente llama a una función interna que realiza el cierre de la aplicación.

```
Sub FUNC_Exit (Byval Param)

    HMIRuntime.Stop

End Sub
```

Figura 100. Código del *script* en lenguaje VBS para cerrar la aplicación SCADA.

4.7.6.8. *Script* para la doble confirmación

Este *script* se emplea para agregar una doble confirmación a la hora de activar o detener un proceso automático.

Para ello, se dota a las secuencias de tres variables internas:

- CONF_SEC: Se activa al darle al botón de orden de marcha o paro. Cuando se activa muestra el *pop-up* de doble confirmación.
- CONF_MARCHA: Se activa al darle al botón de orden de marcha.
- CONF_PARO: Se activa al darle al botón de orden de paro.

Cuando se muestra el *pop-up* de la doble confirmación, este muestra dos botones, confirmar o cancelar, si se le da a cancelar simplemente se pone la variable CONF_SEC a cero y por tanto, el *pop-up* deja de ser visible.

Mientras que si se da al botón de confirmar, en función de qué orden de confirmación haya sido activada (CONF_MARCHA o CONF_PARO) se realizará un *toggle* de dicha orden (OR_MARCHA o OR_PARO), es decir, si estaba activada se desactiva y si estaba desactivada se activa.

Una vez se realiza el *toggle* se ponen a cero todas las variables de confirmación, con el fin de asegurar que cuando vuelva a ejecutarse el *script* no haya valores de la ejecución anterior.

```
Sub OnClick(Byval Item)

If SmartTags("SEC_03.CONF_MARCHA") = 1  Then
'Toggle para activar/desactivar la orden de marcha
    If SmartTags("SEC_03.OR_MARCHA") = 1 Then
        SmartTags("SEC_03.OR_MARCHA")= 0
        SmartTags("SEC_03.CONF_MARCHA")= 0
        SmartTags("SEC_03.CONF_SEC")= 0

    Else
        SmartTags("SEC_03.OR_MARCHA")= 1
        SmartTags("SEC_03.CONF_MARCHA")= 0
        SmartTags("SEC_03.CONF_SEC")= 0
    End If
End If

If SmartTags("SEC_03.CONF_PARO") = 1  Then
'Toggle para activar/desactivar la orden de paro
    If SmartTags("SEC_03.OR_PARO") = 1 Then
        SmartTags("SEC_03.OR_PARO")= 0
        SmartTags("SEC_03.CONF_PARO")= 0
        SmartTags("SEC_03.CONF_SEC")= 0

    Else
        SmartTags("SEC_03.OR_PARO")= 1
        SmartTags("SEC_03.CONF_PARO")= 0
        SmartTags("SEC_03.CONF_SEC")= 0
    End If
End If

End Sub
```

Figura 101. Código del *script* en lenguaje VBS para realizar la doble confirmación.

4.7.7. Diseño de la interfaz de alarmas del sistema

Además del pequeño sumario de alarmas en la pantalla general que mostraba solamente las alarmas activas, se incluye una pantalla para que muestra todas las alarmas historizadas con su estado (tanto activas como inactivas), su nivel de prioridad, la hora y fecha en la que se activaron, su tipo y descripción, así como el elemento averiado y el proceso al que pertenece, pudiendo navegar a la pantalla que contiene dicho elemento.

Los colores empleados para las alarmas son los siguientes:

- Amarillo sobre fondo negro: Averías y fallos de comunicaciones (Alarmas graves de nivel de prioridad 0).
- Blanco sobre fondo rojo: Alarmas por valores críticos (Alarmas de nivel de prioridad 1).

- Negro sobre fondo amarillo: Advertencias por valores de riesgo (Alarmas de nivel de prioridad 2).

Figura 102. Pantalla del histórico de alarmas.

4.7.8. Gestión de usuarios

La gestión de usuarios es esencial en un proyecto SCADA, pues se debe tener un cierto control y seguridad sobre quién puede hacer qué en la aplicación.

En este proyecto se tienen tres grupos de usuario, definidos mediante la herramienta *User Administrator* de WinCC.

El grupo de administradores puede realizar todo tipo de acciones, incluido gestionar los permisos de los demás grupos.

El grupo de supervisores, tiene restringido cambiar las consignas de las fases del proceso, pues están protegidas con el permiso de "Operaciones de alto nivel", permiso que solo tienen los administradores.

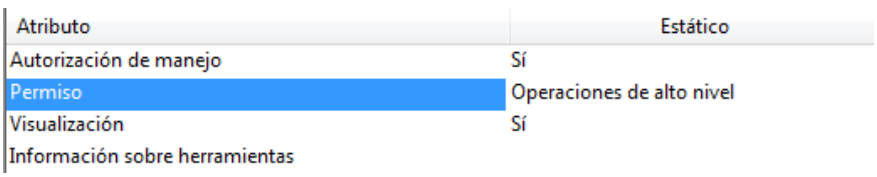


Figura 103. Atributo de seguridad de manejo de un objeto.

El último grupo, el de los operarios, tan solo puede navegar y reconocer alarmas, pues los botones para dar órdenes están protegidos con el permiso “Operaciones de bajo nivel”.

Además, se requiere de introducir un usuario para navegar, pues los botones de la barra de navegación y demás etiquetas que permitan navegar, cuentan de protección mediante el permiso “Observación”.

Para finalizar, con el fin de dotar de seguridad a la aplicación SCADA, se definen tiempos de desconexión por inactividad para los diversos usuarios, 1 minuto para administradores, 2 para supervisores y 3 para operarios.

Desde la pantalla de gestión de usuarios, siempre que se disponga de los permisos necesarios, se pueden crear nuevos usuarios, cambiar las contraseñas, modificar los permisos de los grupos, así como definir los tiempos de desconexión por inactividad.

Figura 104. Pantalla de gestión de usuarios.

4.7.9. Diseño de la interfaz del registro de operaciones y eventos

Al igual que las alarmas, en una aplicación SCADA es importante tener un registro de los cambios de consigna que afecten al proceso, así como qué usuario y cuándo realizó dicho cambio.

Todo eso se muestra en la pantalla de histórico de eventos, mostrando los cambios en todos los objetos que tengan activo su atributo “Aviso de operación”.

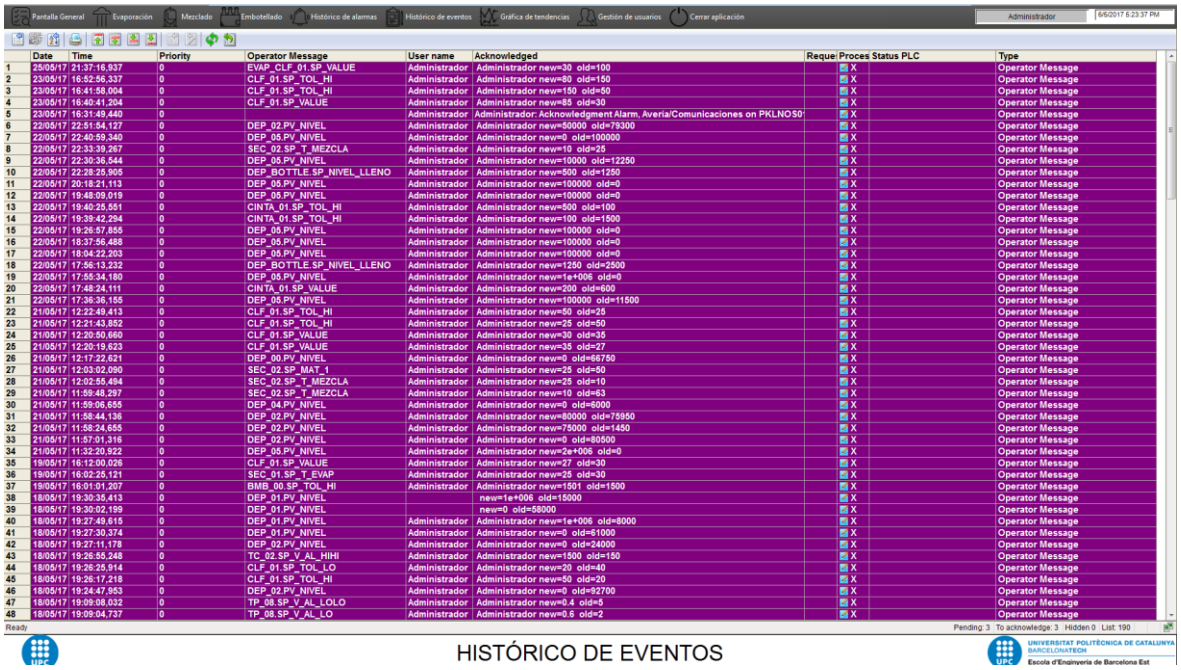


Figura105. Pantalla de histórico de eventos.

4.7.10. Gráficos de históricos y tendencias

Con la herramienta *Tag Logging* de WinCC, se pueden crear ficheros que contendrán todos aquellos tags que se deseen graficar.

En este proyecto, se han definido dos ficheros, uno para los valores de proceso y otro para los valores de consigna.

En la pantalla de tendencias, mediante el objeto *TrendControl* se pueden crear plantillas para graficar con ejes personalizados, acceder a los archivos definidos para seleccionar qué tags se quieren graficar y personalizar las curvas de tendencia como se desee.

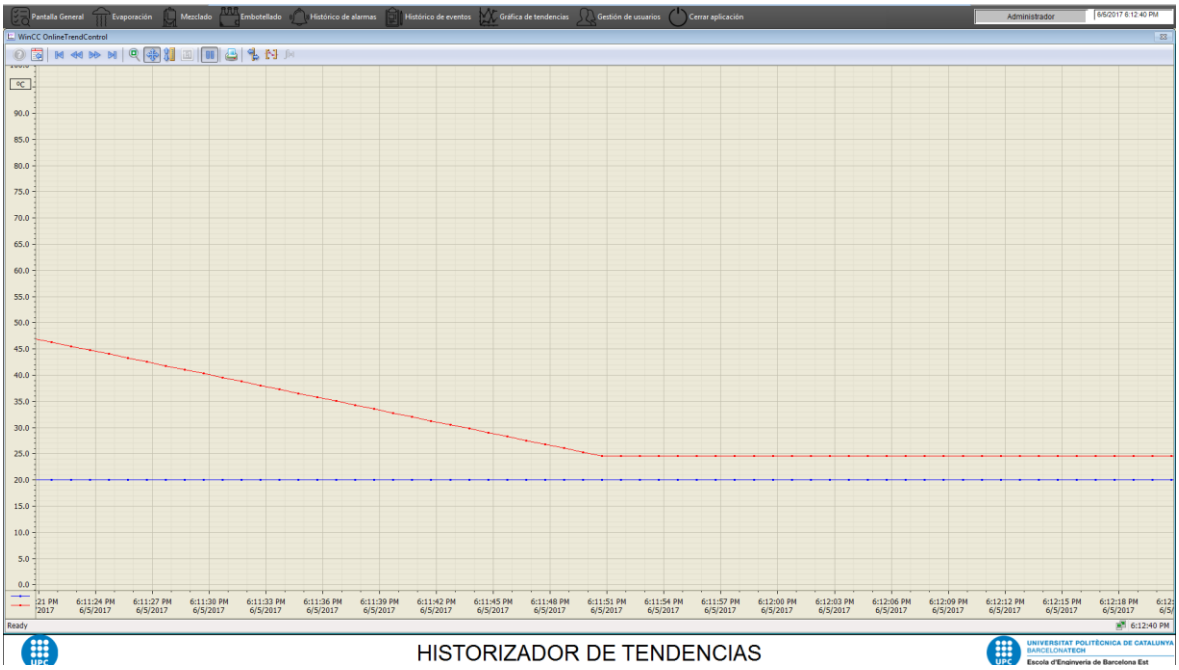


Figura 106. Pantalla de historizador de tendencias: valor de la temperatura (rojo) y nivel de consigna (azul).

5. Pruebas y resultados

5.1. Diseño de las pruebas de funcionalidad

En las fases más tempranas del desarrollo del proyecto, se empleaba la herramienta de *tablas de observación y forzado permanente* que ofrece TIA Portal.

Mediante esta herramienta, se confeccionaba una tabla para cada elemento a probar y se iban forzando variables para ver su correcto funcionamiento.

Por ejemplo, comprobar que se activa el final de carrera abierto pasados los 3 segundos tras haber dado la orden de apertura o que solo se activa el comando de cierre si se da la orden en el modo de funcionamiento seleccionado (manual o automático).

Nombre	Dirección	Format...	Valor de observac..
"DB_VLV".EVAP_VLV_00.PET.OR_MAN	%DB7.D...	BOOL	FALSE
"DB_VLV".EVAP_VLV_00.PET.OR_AUTO	%DB7.D...	BOOL	FALSE
"DB_VLV".EVAP_VLV_00.PET.OR_OP_MAN	%DB7.D...	BOOL	FALSE
"DB_VLV".EVAP_VLV_00.PET.OR_OP_AUTO	%DB7.D...	BOOL	FALSE
"DB_VLV".EVAP_VLV_00.PET.OR_CL_MAN	%DB7.D...	BOOL	FALSE
"DB_VLV".EVAP_VLV_00.PET.OR_CL_AUTO	%DB7.D...	BOOL	FALSE
"DB_VLV".EVAP_VLV_00.CMD.CMD_OP	%DB7.D...	BOOL	FALSE
"DB_VLV".EVAP_VLV_00.CMD.CMD_CL	%DB7.D...	BOOL	FALSE
"DB_VLV".EVAP_VLV_00.CNF.CNF_FCA	%DB7.D...	BOOL	TRUE
"DB_VLV".EVAP_VLV_00.CNF.CNF_FCC	%DB7.D...	BOOL	FALSE
"DB_VLV".EVAP_VLV_00.ST.ST_OP	%DB7.D...	BOOL	TRUE
"DB_VLV".EVAP_VLV_00.ST.ST_CL	%DB7.D...	BOOL	FALSE
"DB_VLV".EVAP_VLV_00.ST.ST_MAN	%DB7.D...	BOOL	FALSE
"DB_VLV".EVAP_VLV_00.ST.ST_AUTO	%DB7.D...	BOOL	TRUE

Figura 107. Tabla de observación y forzado para testear los modos de funcionamiento de una válvula.

A medida que los bloques del programa PLC se iban desarrollando y las pruebas de funcionalidad se realizaban satisfactoriamente, se realizaban las pruebas de funcionalidad para los iconos y los mandos de control de la aplicación SCADA.

Para ello, se replica la estructura de variables de cada objeto pero definiendo los tags como variables internas, con el fin de poder probar que el icono se comporte como se desea sin depender ni de la comunicación PLC-SCADA ni del programa del PLC.

Con las estructuras de variable internas de testeo, se realiza una pantalla de pruebas en la que además del icono, se añaden campos de entrada para introducir manualmente los datos que deberían provenir del PLC.

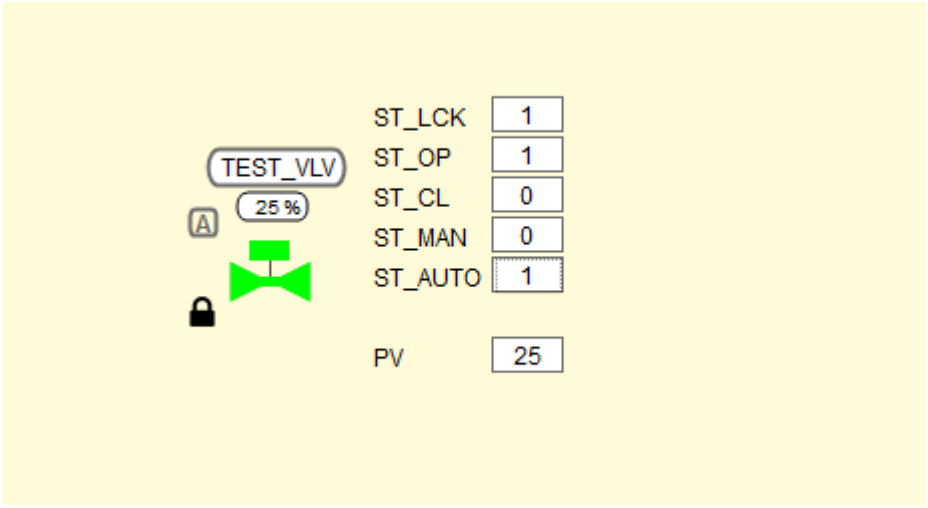


Figura 108. Pantalla de pruebas para testear las animaciones del icono válvula.

Para realizar las pruebas de funcionamiento de las secuencias automáticas, una vez probados los iconos y los mandos individualmente, se procede a seguir el código de secuencia en TIA Portal a la vez que se sigue en el sinóptico de proceso de la aplicación SCADA, determinando así si se realizan de forma correcta tanto el código como las animaciones.

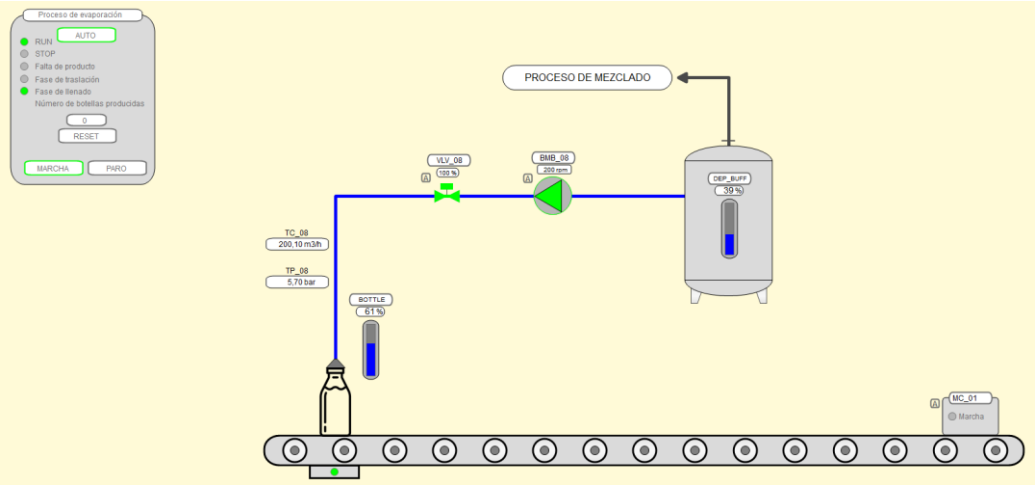
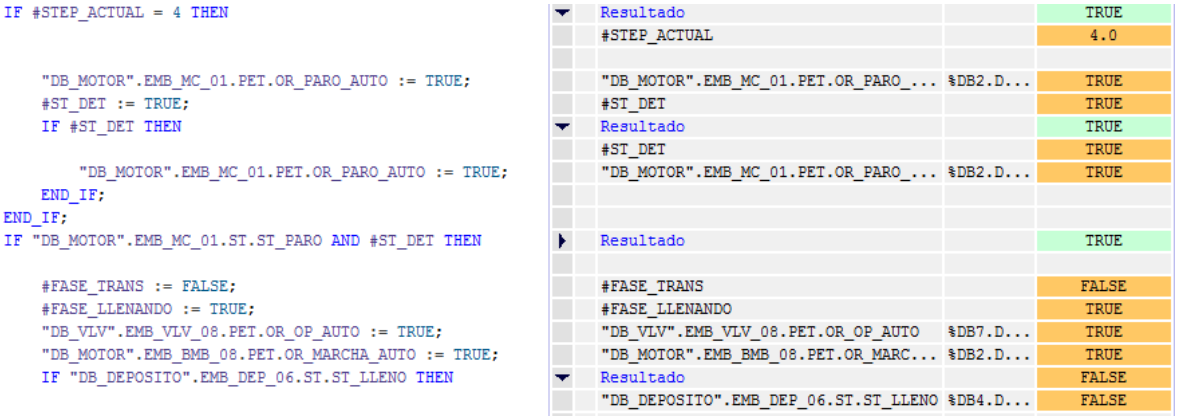


Figura 109. Sinóptico del proceso de embotellado.



```
IF #STEP_ACTUAL = 4 THEN


    "DB_MOTOR".EMB_MC_01.PET.OR_PARO_AUTO := TRUE;
    #ST_DET := TRUE;
    IF #ST_DET THEN

        "DB_MOTOR".EMB_MC_01.PET.OR_PARO_AUTO := TRUE;
    END_IF;
END_IF;
IF "DB_MOTOR".EMB_MC_01.ST.ST_PARO AND #ST_DET THEN

    #FASE_TRANS := FALSE;
    #FASE_LLENANDO := TRUE;
    "DB_VLV".EMB_VLV_08.PET.OR_OP_AUTO := TRUE;
    "DB_MOTOR".EMB_BMB_08.PET.OR_MARCHA_AUTO := TRUE;
    IF "DB_DEPOSITO".EMB_DEP_06.ST.ST_LLENO THEN
```

Resultado			TRUE
#STEP_ACTUAL			4.0
"DB_MOTOR".EMB_MC_01.PET.OR_PARO_...	%DB2.D...		TRUE
#ST_DET			TRUE
Resultado			TRUE
#ST_DET			TRUE
"DB_MOTOR".EMB_MC_01.PET.OR_PARO_...	%DB2.D...		TRUE
Resultado			TRUE
#FASE_TRANS			FALSE
#FASE_LLENANDO			TRUE
"DB_VLV".EMB_VLV_08.PET.OR_OP_AUTO	%DB7.D...		TRUE
"DB_MOTOR".EMB_BMB_08.PET.OR_MARC...	%DB2.D...		TRUE
Resultado			FALSE
"DB_DEPOSITO".EMB_DEP_06.ST.ST_LLENO	%DB4.D...		FALSE

Figura 110. Seguimiento de la secuencia de embotellado en TIA Portal.

5.2. Diseño de las pruebas de comunicación PLC – SCADA

Para comprobar la correcta comunicación entre el autómata y el SCADA, se recurre a la herramienta de administración de variables de WinCC.

Dicha herramienta muestra todas las variables definidas para comunicar con el autómata ST-1500, así como su valor y su código de calidad.

Viendo el código de calidad se puede ver si el SCADA accede correctamente a la dirección del PLC definida y según el valor se puede ver si comunica con el dato adecuado.

Variables [ST-1500]

	Nombre	Valor	Quality Code
31	EMB_BMB_08.SP_T_HI	5	0x80 - good - ok
32	EMB_BMB_08.SP_T_LO	5	0x80 - good - ok
33	EMB_BMB_08.SP_TOL_HI	1500	0x80 - good - ok
34	EMB_BMB_08.SP_TOL_LO	100	0x80 - good - ok
35	EMB_BMB_08.SP_VALUE	200	0x80 - good - ok
36	EMB_BMB_08.ST_AUTO	1	0x80 - good - ok
37	EMB_BMB_08.ST_DIS_AL	0	0x80 - good - ok

Figura 111. Interfaz de la herramienta de administración de variables para comprobar el estado de las comunicaciones.

5.3. Resultados de las pruebas

A continuación se adjuntarán imágenes de las diferentes etapas del proceso verificando así el resultado satisfactorio de las pruebas de los diversos elementos y de los procesos automáticos en sí.

En dichas imágenes se puede apreciar el correcto funcionamiento de los dispositivos que forman parte de la fase en pruebas así como el correcto llenado y vaciado de los depósitos.

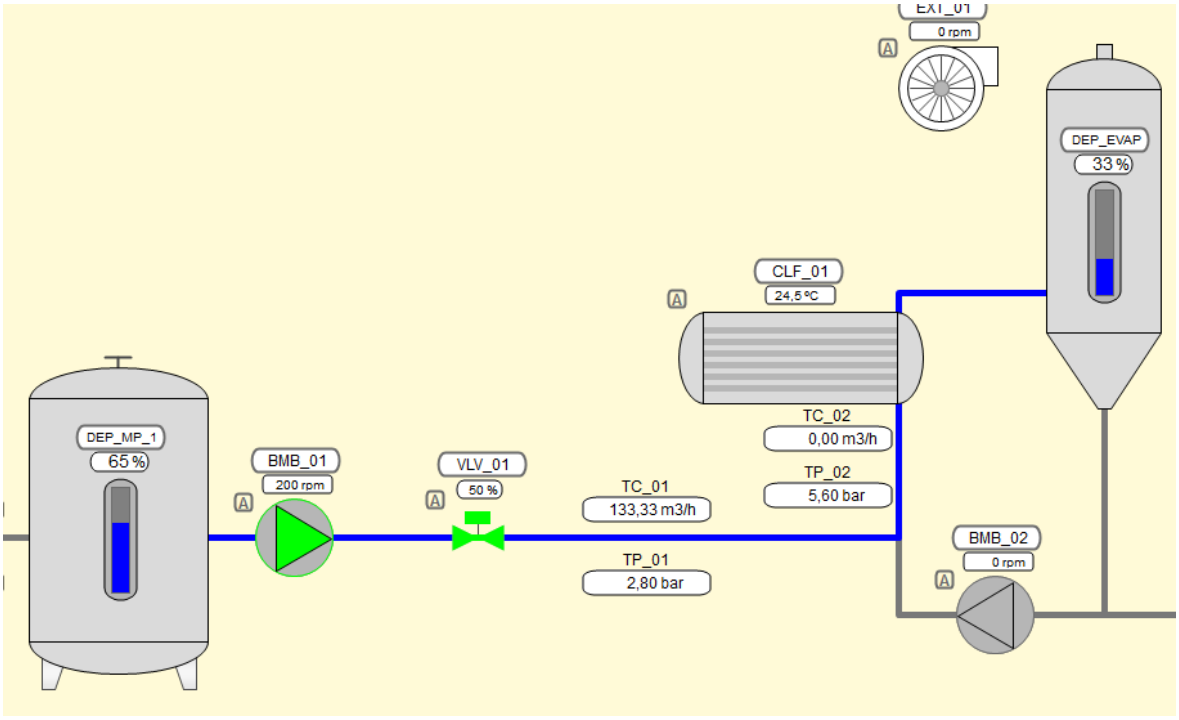


Figura 112. Pantalla del proceso de evaporación: prueba del funcionamiento de la fase de carga.

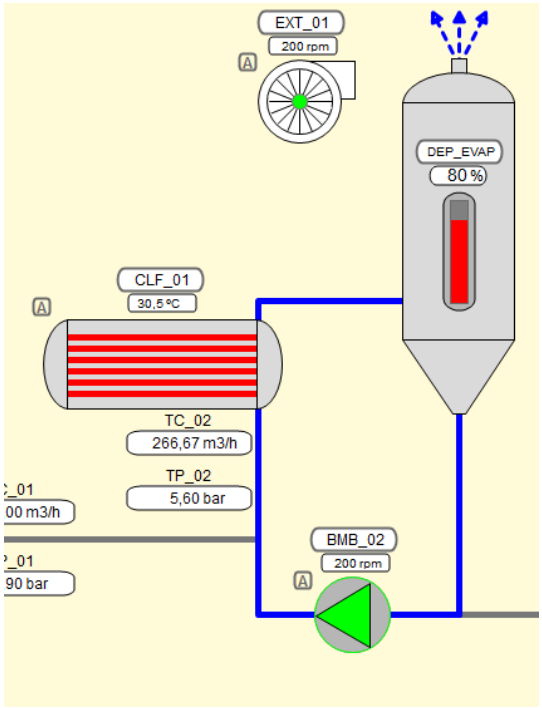


Figura 113. Pantalla del proceso de evaporación: prueba del funcionamiento de la fase de evaporación.

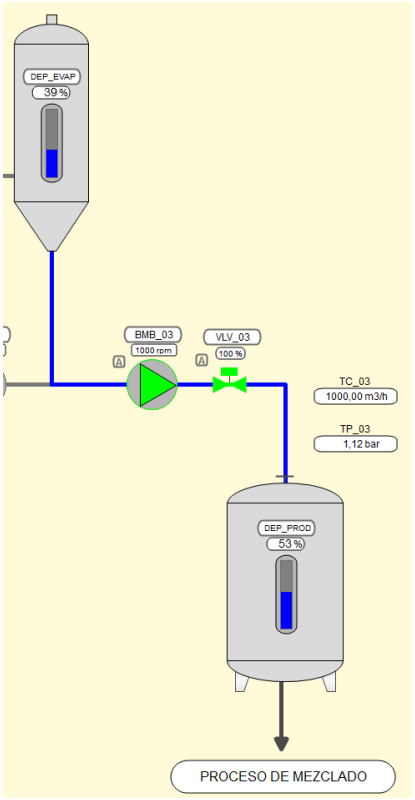


Figura 114. Pantalla del proceso de evaporación: prueba del funcionamiento de la fase de descarga.

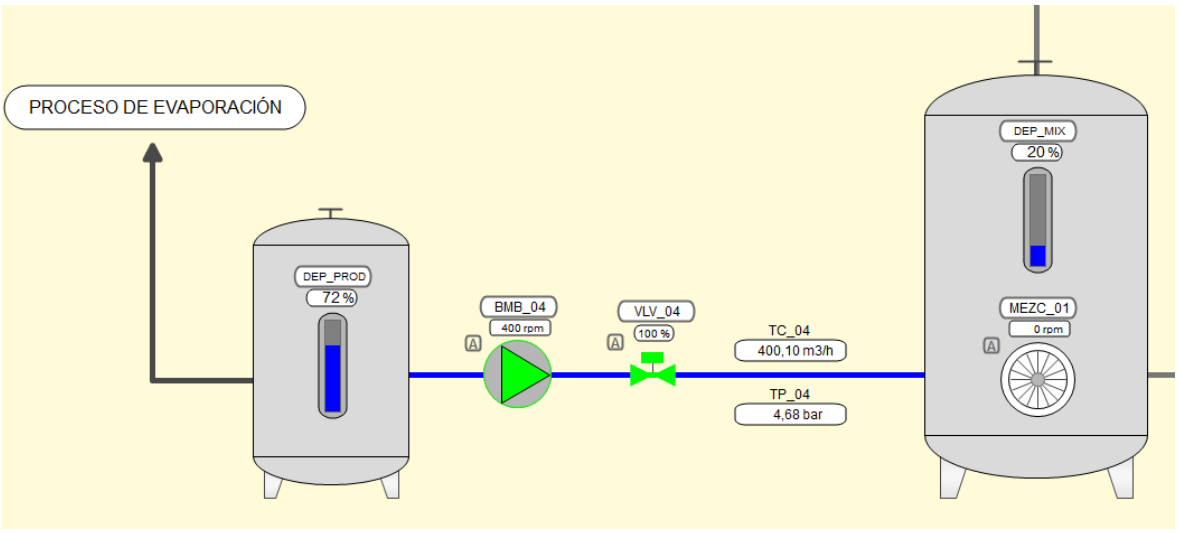


Figura 115. Pantalla del proceso de mezclado: prueba del funcionamiento de la fase de carga del producto A.

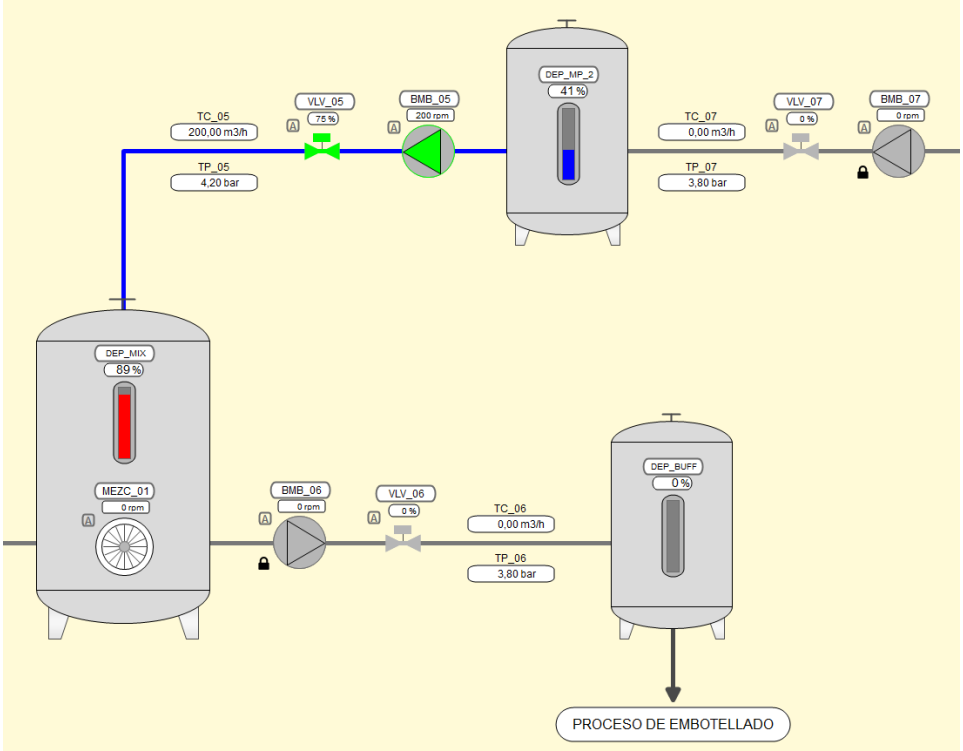


Figura 116. Pantalla del proceso de mezclado: prueba del funcionamiento de la fase de carga del producto B.

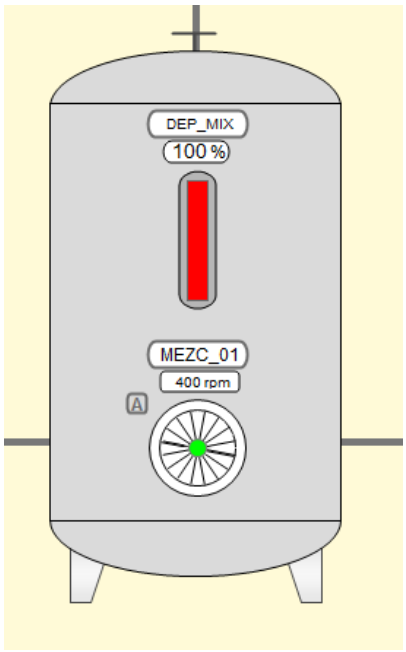


Figura 117. Pantalla del proceso de mezclado: prueba del funcionamiento de la fase de mezcla.

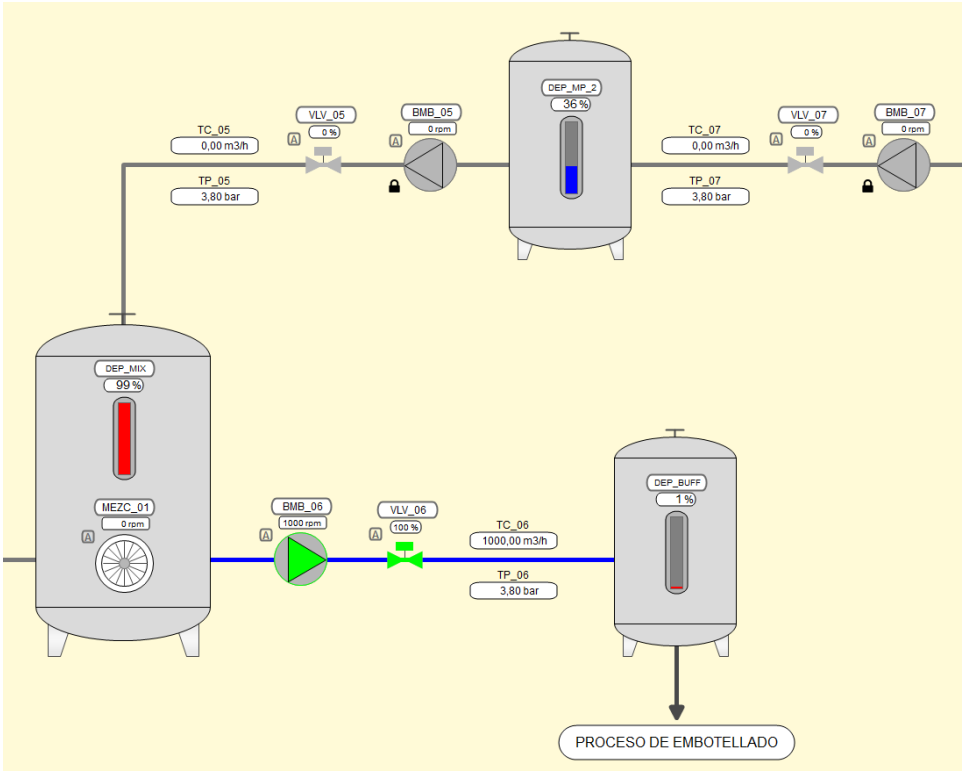


Figura 118. Pantalla del proceso de mezclado: prueba del funcionamiento de la fase de descarga.

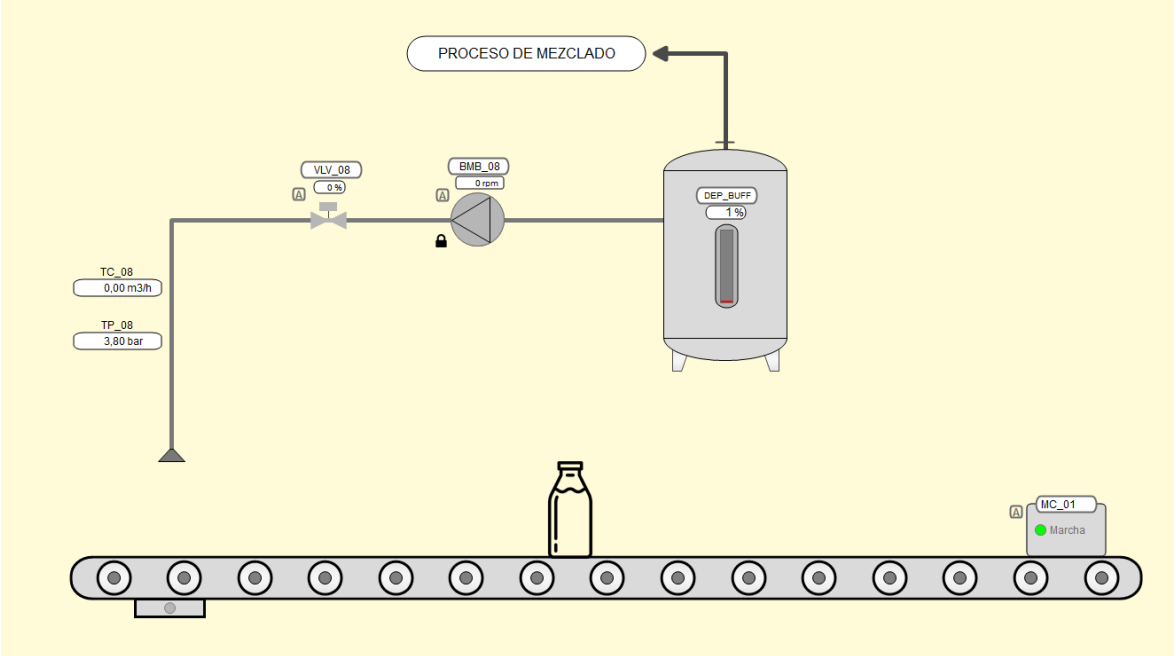


Figura 119. Pantalla del proceso de embotellado: prueba del funcionamiento de la fase de transporte.

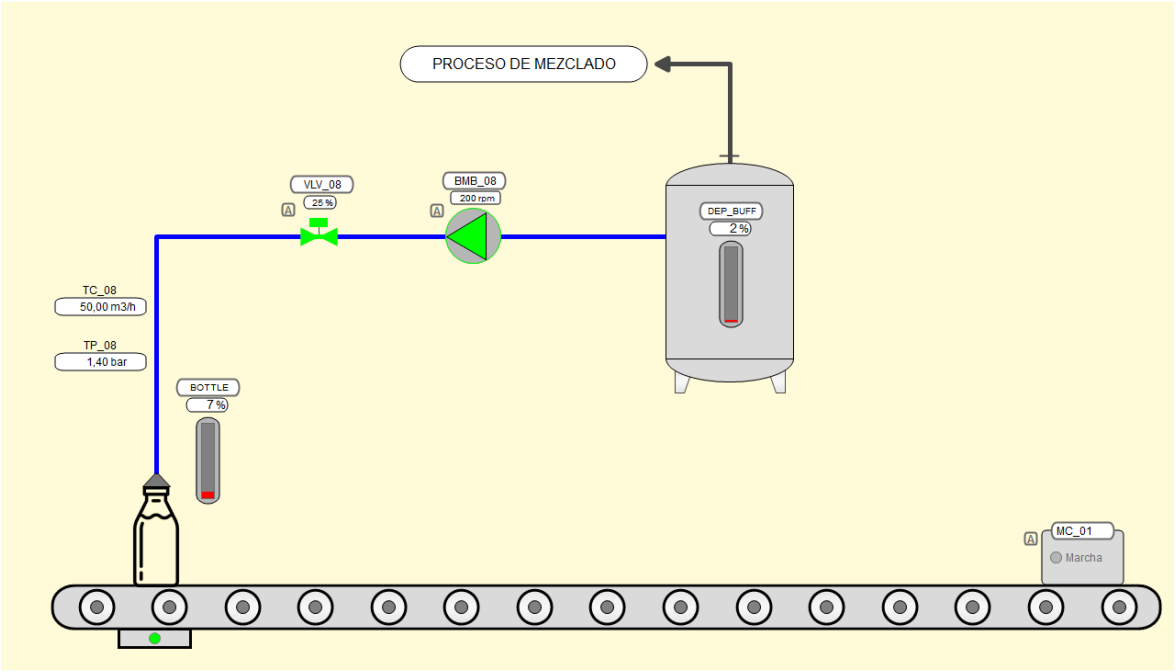


Figura 120. Pantalla del proceso de embotellado: prueba del funcionamiento de la fase de llenado.

6. Normativa

6.1. Implementación del programa del PLC

6.1.1. IEC 61131

Para la implementación del programa del PLC se ha seguido el estándar internacional IEC 61131. Dicho estándar es un conjunto de normas e informes técnicos que tiene como objetivo estandarizar todo lo referido a controladores lógicos programables y sus periféricos.

El estándar se compone por ocho normas:

- Parte 1: Información general.
- Parte 2: Especificaciones y ensayos de los equipos.
- Parte 3: Lenguajes de programación.
- Parte 4: Guías de usuarios.
- Parte 5: Comunicaciones.
- Parte 6: Seguridad funcional.
- Parte 7: Programación de control difuso.
- Parte 8: Directrices para la aplicación e implementación de lenguajes de programación.

6.1.2. IEC 61131-3

De las ocho normas del IEC 61131, cabe destacar la tercera, pues es la que se ha seguido a rajatabla a la hora de realizar el proyecto.

El estándar IEC 61131-3 trata los lenguajes de programación y define sus estándares:

- Lenguaje escalera (LD): Lenguaje de programación gráfico.
- Diagrama de bloques de funciones (FBD): Lenguaje de programación gráfico.
- Texto estructurado (ST): Lenguaje de programación textual.
- Lista de instrucciones (IL): Lenguaje de programación textual.
- Bloques de función secuenciales (SFC): Lenguaje que describe gráficamente el comportamiento secuencial de un programa.

En cuanto a su aplicación en el proyecto, como se explicó previamente en el documento, se emplea lenguaje escalera para las funciones de los dispositivos y lenguaje estructurado para los bloques de función de las secuencias de control.

Por otro lado, el estándar IEC 61131-9 también define los diferentes tipos de datos a emplear, con el propósito de prevenir errores en el desarrollo de programas.

Los tipos de datos que abarca el estándar son booleanos (bool), enteros (int), reales (float), bytes, word, date, time-of-day y strings.

El estándar también permite al usuario crear sus propios tipos de variable, conociéndose como Tipo de Datos Derivados.

Mediante los tipos de datos derivados, un programador es capaz de definir un canal de entrada como un tipo de dato y usarlo múltiple veces (instanciarlo).

La aplicación directa de esta parte de la norma es evidente en la definición de las variables del PLC, siendo todas de tipos de datos abarcados en la norma, además de emplearse las estructuras de datos definidas por el usuario (UDT's) que equivaldrían a los tipos de datos derivados.

Para finalizar, el estándar también define los programas, bloques de función y funciones como Unidades de Organización de Programa (POUs).

Además el IEC 61131-3 incluye instancias de funciones ya definidas (ADD, ABS, SQRT, SIN y COS) y permite al usuario crear sus propios bloques de funciones e instanciarlos múltiples veces, pudiendo contener tanto datos como algoritmos.

6.2. Programación del SCADA

6.2.1. ISA 101

Para el desarrollo de la aplicación SCADA, se ha seguido la norma ISA 101. Dicha norma establece estándares y prácticas recomendadas a la hora de realizar interfaces hombre-máquina para aplicaciones de fabricación o procesos.

La norma define los modelos de la terminología y el desarrollo de un HMI (Interfaz hombre-máquina), pretendiendo proporcionar orientación para diseñar, construir, operar y mantener HMI efectivas que resulten seguras, eficaces y hacer el control del proceso más eficiente.

Los puntos fuertes de la norma aplicados al proyecto han sido:

- Convención de navegación de pantallas.
- Convención de colores y gráficos.
- Elementos dinámicos.

- Convención de alarmas.
- Interfaces de bases de datos de históricos.

A fin de cuentas, el estándar ISA 101 se emplea para crear una interfaz gráfica fácil de entender y que dé opciones claras a los usuarios que operen con ella.

6.2.2. ISA 5.5

Otra norma a tener en cuenta a la hora de realizar un SCADA es el estándar ISA 5.5.

El propósito de esta norma es establecer un criterio de símbolos gráficos para las interfaces de las pantallas de procesos de supervisión y control.

Su principal objetivo es hacer lo más simple posible la comprensión de todo lo que se muestre en pantalla, agilizando así las gestiones que deba realizar el usuario.

La implementación relativa a esta norma en el proyecto consiste en la utilización de colores que la norma propone, mostrados en la siguiente figura:

Color	Generic meaning	Element association
Black	Background	
Red	Emergency	A) Stop B) Highest Priority Alarm C) Closed D) Off
Yellow	Caution	A) Abnormal Condition B) Second Priority Alarm
Green	Safe	A) Normal Operation B) Start C) Open D) On
Cyan (Light Blue)	Static & Significant	A) Process Equipment in Service B) Major Labels
Blue	Nonessential	A) Standby Process Equipment B) Labels, Tags, etc.
Magenta (Purple)	Radiation	A) Radiation Alarms B) Questionable Values
White	Dynamic Data	A) Measurements & State Information B) System Messages C) Trend D) Active Sequential Step

Figura 121. Tabla de empleo de colores según el estándar ISA 5.5.

6.2.3. Guía GEDIS

Por último pero no menos importante, siempre que se realice un proyecto de esta índole, se debe tener en cuenta la guía GEDIS (Guía Ergonómica de Diseño de Interfaz de Supervisión).

Esta guía sirve como referencia para evaluar un sistema SCADA según diversos parámetros:

- Consistencia
- Visibilidad
- Perceptibilidad
- Informatividad
- Interactividad
- Tiempo de respuesta

La guía GEDIS pretende cubrir todos los aspectos del diseño de la interfaz mediante los diez indicadores que se muestran en la siguiente figura:

Indicador	Definición	Entradas	Salidas
Arquitectura	Organización jerárquica de las pantallas	De la planta física a la monitorización gráfica	Mapa de relaciones entre pantallas y sus funciones
Distribución de pantallas	Plantillas de los diferentes tipos de pantalla	Diseño de los procesos físicos y subprocesos	Clasificación de tipos de pantallas y tipos de plantillas
Navegación	Modos de navegación entre pantallas	Controles de navegación entre subprocesos	Navegación equilibrada en anchura y profundidad
Uso del color	Asociación de funcionalidades en el ámbito del control de procesos	Requisitos sobre dispositivos de información visual	Uso del color adecuado en el contexto
Uso de fuentes e información textual	Abanico de fuentes y asociación de funcionalidades	Fuentes y tamaños legibles por el operario	Estándares de fuentes, acrónimos y abreviaturas
Estatus de los equipos y eventos de proceso	Símbolos e iconos gráficos para representar el estado de la planta y los cambios de estado	Estándares nacionales y/o internacionales en control supervisor	Uso de símbolos e iconos reconocibles por el operario experto
Información y valores de proceso	Presentación de los datos analógicos/digitales en los gráficos	Procesamiento de la información	Lista clasificada de las variables del proceso
Gráficos de tendencias y tablas	Presentación y agrupación de valores en gráficos de tendencias (históricos) y tablas	Procesamiento de la información	Lista de agrupaciones de datos en gráficos y tablas en los sinópticos de proceso
Comandos y entradas de datos	Modo de entrada de datos a la interfaz	Estándares de diseño de comandos y entrada de datos	Accesibilidad a la manipulación de parámetros y consignas
Alarmas	Características principales del subsistema de alarmas	Estimación del riesgo	Listado de alarmas, clasificación por prioridades

Figura 122. Ejemplo de indicadores típicos para el análisis según la guía GEDIS.

Cabe mencionar que estos indicadores pueden variar en función del desarrollador y se pueden desglosar en diversos subindicadores que los complementen, a modo de ejemplo el indicador de uso del color podría desglosarse en visibilidad, contraste, número de colores, diferenciabilidad, uso de colores típicos, consistencia, etc.

Para la evaluación de la interfaz, cada uno de los indicadores se mide en una escala de 1 a 5. De esta manera se pueden determinar los puntos a mejorar de la interfaz.

A continuación se adjunta el análisis según la guía GEDIS de la aplicación SCADA realizada en este proyecto.

Tabla 12. Evaluación de la aplicación SCADA según la guía GEDIS.

INDICADOR	NOTA	COMENTARIO
Arquitectura	5	
Distribución de pantallas	4	Las pantallas siguen un patrón invariable
Navegación	5	
Uso del color	5	
Uso de fuentes e información textual	3	Se podrían emplear distintas fuentes para diferenciar procesos y elementos
Estatus de los equipos y eventos	5	
Información de valores de proceso	5	
Gráficos de tendencias y tablas	3	Se podrían crear listas preestablecidas de valores a graficar
Comandos y entradas de datos	5	
Alarmas	5	
NOTA MEDIA	**4,5**	

El resultado obtenido indica que el sistema SCADA cumple con creces los requisitos mínimos que el desarrollador ha definido e indica los aspectos que han resultado ser menos claros a la hora de trabajar con la aplicación SCADA. Destacando entre ellos la gestión de tendencias y gráficas, pues la aplicación es muy poco intuitiva a la hora de realizar dichas gestiones.

Conclusiones

Una vez concluido el proyecto, es momento de echar la vista atrás y comparar los resultados obtenidos con los propósitos que se tenían al inicio del proyecto.

Cuando se empezó a plantear el proyecto, su principal propósito era desarrollar un sistema de control y supervisión profesionales, un producto realista que pudiera verse en el mercado. Dicho objetivo se ha cumplido, pues el producto resultante, la aplicación de SCADA, cumple con todas las funcionalidades que se determinaron en un principio.

Asimismo, el control implementado en el autómata, también realiza las funciones que se estipularon, y a pesar de que la simulación podría optimizarse más, ésta era un mero "trámite" para realizar el proyecto, pues en un principio se iba a realizar el proyecto en una planta piloto real.

Dejando de lado los objetivos cumplidos, a título personal, este proyecto me ha aportado la experiencia que buscaba: enlazar el mundo de los PLC con el mundo de los SCADA. Y por ende, he dado un paso más hacia la formación del perfil profesional que busco, quedando plenamente satisfecho del trabajo realizado y con el que culmina estos cuatro años de formación.

Presupuesto

Coste del material

Este apartado hace un análisis presupuestario de los costes referidos al material requerido para la realización del proyecto, tanto a nivel de hardware como a nivel de licencias para el software.

Tabla 13. Coste del material.

Concepto	Referencia	Unidades	Precio unitario (€)	Total
Ordenador		1	1177	1177
PLC SIMATIC S7-1500	6ES7 511-1AK01-0AB0	1	568,88	568,88
SIMATIC PM 1507	6EP1332-4BA00	1	95,21	95,21
Licencia TIA Portal v14	6ES7822-1AA04-0YA5	1	1741,8	1741,80
Licencia SIMATIC WinCC	6AV2103-0XA04-0AA5	1	3685,32	3685,32
COSTE BASE				7268,21
IVA 21%				1526,32
Total				**8794,53**

Coste de la mano de obra

Este apartado hace un análisis presupuestario de los costes referidos a la mano de obra y las horas de dedicación a las diversas tareas, determinando un precio por hora trabajada para un ingeniero novel y en función de si son horas de dedicación a tareas de desarrollo tecnológico o de documentación

Tabla 14. Coste de la mano de obra.

Concepto	Unidades (h)	Precio unitario(h)	Total
Búsqueda información/Planificación	60	10	600
Diseño del programa del PLC	200	20	4000
Diseño de la aplicación SCADA	240	20	4800
Documentación	105	10	1050
COSTE BASE			10450
IVA 21%			2194,50
Total	**605**		**12644,50**

Coste total

Para concluir, se adjunta el precio total de la realización del proyecto.

Tabla 15.. Coste total del proyecto.

Concepto	Unidades	Precio unitario (€)	Total
Coste del material	1	8794,53	8794,53
Coste mano de obra	1	12644,50	12644,50
Total			**21439,03**

Viendo el coste total del proyecto, se aprecia que no es un coste descabellado para un proyecto de tal magnitud, pues se debe tener en cuenta que la implementación del sistema SCADA proporcionaría una mejora de la producción y reduciría costes de mantenimiento, permitiendo así amortizar en poco tiempo la inversión.

Bibliografía

6.3. Bibliografía

- Diez Aznar, José Manuel, 2011. *Aprenda WinCC*. Servicio de publicación de la Universidad Politécnica de Valencia. ISBN 9788483637623.
- Rodríguez Penin, Aquilino, 2007. *Sistemas SCADA*. 3ª edición. Marcombo, S.A. ISBN 9788426717818.

6.4. Webgrafía

- Prezi. *Evaporador de circulación forzada*. [En línea]. [Consulta: 07 Febrero 2017]. Disponible en: < https://prezi.com/ffmovuiqx3qa/evaporador-de-circulacion-forzada/>
- Wikipedia. *Piping and instrumentation diagram.* [En línea]. [Consulta: 12 Febrero 2017]. Disponible en: < https://es.wikipedia.org/wiki/Piping_and_instrumentation_diagram>
- Siemens. *SIMATIC S7-1500.* [En línea]. [Consulta: 21 Marzo 2017]. Disponible en: <http://w5.siemens.com/spain/web/es/industry/automatizacion/simatic/controladores_modulares/controlador_avanzado/s71500/pages/default.aspx>.
- InfoPLC. *Tipos de datos (Estructuras)*. [En línea]. [Consulta: 02 Abril 2017]. Disponible en: < http://www.infoplc.net/descargas/106-siemens/software-step7-tiaportal/2287-tia-portal-estructura-datos>.
- InfoPLC. *Programación estructurada de autómatas Siemens*. [En línea]. [Consulta: 16 Abril 2017]. Disponible en: < http://www.infoplc.net/descargas/106-siemens/software-step7-tiaportal/2087-programacion-estructurada-automatas-siemens>.
- GrayMatterSystems. *4 things you have to know about ISA101.* [En línea]. [Consulta: 20 Mayo 2017]. Disponible en: < http://graymattersystems.com/the-4-things-you-need-to-know-about-isa-101-hmi-standard/>.
- InfoPLC. *ISA101 Norma para el diseño HMI*. [En línea]. [Consulta: 20 Mayo 2017]. Disponible en: < http://www.infoplc.net/actualidad-industrial/item/102902-isa101-hmi>
- SlideShare. *Guía GEDIS*. [En línea]. [Consulta: 21 Mayo 2017]. Disponible en: <https://es.slideshare.net/Buzz80/gua-gedis>.
- Avanceon. *ISA-5.5-1985*. [En línea]. [Consulta: 23 Mayo 2017]. Disponible en: <http://avanceon.com/wp-content/uploads/2017/01/ISA_5.5.pdf>