ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

Co-simulación de una estación de paletizado entre un PLC virtual SIEMENS y el modelo de la planta en C#.

PROYECTO INTEGRADOR

Previo la obtención del Título de:

Ingeniero en Electrónica y Automatización

Presentado por:

Sally Krystle Aguilar García

GUAYAQUIL - ECUADOR

Año: 2020

DEDICATORIA

El presente proyecto se lo dedico a Dios por darme la fortaleza para nunca rendirme, a mis hijos Gia, Matheo, Amy, Sergio y Sebastián por ser la razón por la que me levanto todos los días, a mi familia, mi hermano Luis y mis padres Sally y Luis por ser un apoyo fundamental en mi vida y a mi compañero de vida Leonardo, quien me enseño que nunca es tarde para volver a empezar, y quien ha sido mi apoyo para culminar esta etapa muy importante de mi vida profesional.

Sally Aguilar García

AGRADECIMIENTOS

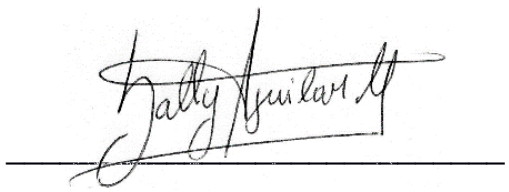
Mi más sincero agradecimiento a los profesores con los que trabajé este proyecto que fueron guías fundamentales para su desarrollo y se portaron como excelentes amigos por el apoyo incondicional brindado; al Ing. Livington Miranda por su ayuda en el desarrollo del proyecto, al tutor de mi proyecto el PhD. Douglas Plaza por su confianza y conocimientos impartidos, y al PhD. Wilton Agila por su ayuda y paciencia.

Y también a todos los excelentes maestros, que a lo largo de mi carrera estudiantil me impartieron sus conocimientos.

Sally Aguilar García

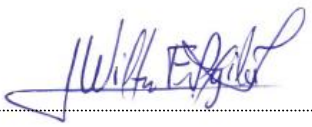
DECLARACIÓN EXPRESA

“Los derechos de titularidad y explotación, me corresponden conforme al reglamento de propiedad intelectual de la institución; Sally Krystle Aguilar García doy mi consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"



Sally Krystle Aguilar García

EVALUADORES



PhD. Wilton Agila Gálvez

PROFESOR DE LA MATERIA

PhD. Douglas Plaza

PROFESOR TUTOR

RESUMEN

Actualmente, para acceder a recursos prácticos en el campo de la automatización, es necesario hacer uso de elementos donde se desarrollen diferentes escenarios, más parecidos al entorno de acción que encontraremos en el ejercicio profesional, estos son: un PLC y una planta de procesos industriales, que permita la ejecución de instrucciones, así como fallas y demás pruebas, existiendo el riesgo intrínseco de daños en recursos de uso común. Por lo expuesto, en este trabajo se desarrolló una solución práctica para estudiantes, basada en co-simulación, que les permitirá desarrollar una extensa variedad de escenarios de prueba, sin comprometer la integridad de los dispositivos con los que cuenta el laboratorio, y sin la necesidad de que se encuentren físicamente en él.

Para el desarrollo de este proyecto nos basamos en la planta de paletizado IMS10, del fabricante Lucas-Nülle, sus componentes son visibles en HMI virtual (TP900). Para simular su comportamiento, se creó un entorno en Visual Studio 2013. Finalmente, para el control, se simuló un CPU SIEMENS S7 1500, programado en TIA-PORTAL V. 15.1.

Como resultado, se completó el diseño de la planta de paletizado en lenguaje C#, ejecutada en Visual Studio. Con esto se comprobó que, tanto la programación del PLC, y su simulación gráfica, ambas desarrolladas en TIA-PORTAL, fueron completadas exitosamente.

Con la ejecución de este proyecto, se cuenta con un sistema que permite a los estudiantes hacer simulaciones en un ambiente de pruebas, accesible, inclusive remotamente, con la posibilidad de plantear escenarios sin riesgo de provocar daños en los equipos del Laboratorio de Control de Procesos Industriales.

Palabras Clave: Co-simulación, PLC S7 1500 SIEMENS, planta de paletización.

ABSTRACT

Nowadays, to access practical resources in the field of automation, it is necessary to make use of elements where different scenarios are developed, more similar to the action environment that we will find in professional practice, these are: a PLC and a Plant that allows the execution of instructions, as well as failures and other tests, with the intrinsic risk of damage to commonly used resources. Therefore, in this work a practical solution was developed for students, based on Co-simulation, which will allow them to develop a wide variety of test scenarios, without compromising the integrity of the devices in the laboratory, and without the need that they are physically in it.

For the development of this project we are based on the L-N IMS10 palletizing plant, its components are visible in Virtual HMI (TP900). To simulate its behavior, an environment was created in Visual Studio 2013. Finally, for the control, a SIEMENS S71500 CPU was simulated, programmed in TIA-PORTAL V. 15.1.

As a result, the design of the palletizing plant in C # language, executed in Visual Studio, was completed. With this it was verified that both the programming of the PLC, and its graphical simulation, both developed in TIA Portal, were completed successfully.

With the execution of this project, there is a system that allows students to do simulations in a test environment, accessible, even remotely, with the possibility of posing scenarios without the risk of causing damage to the automation laboratory equipment.

Keywords: Co-simulation, PLC S7 1500 SIEMENS, Pallet Stacking plant.

ÍNDICE GENERAL

ABREVIATURAS

ESPOL	Escuela Superior Politécnica del Litoral
FIEC	Facultad de Ingeniería en Electricidad y Computación
PLC	Programmable Logic Controller
IP	Internet Protocol
SCADA	Supervisory Control and Data Acquisition
FMI	Functional Mock-up Interface
CPU	Central processing Unit
IDE	Entorno de desarrollo Integrado
FC	Funciones
FB	Bloques de Función
DB	Bloques de datos
OB	Bloque de objetos
PC	Personal Computer
TCP/IP	Transmission Control Protocol / Internet Protocol
FTP	File Transfer Protocol
STEP7	Controlador Simple de Programación
API	Application Programming Interface
DNS	Domain Name Service

FAT	Prueba de aceptación de fábrica
LED	Diodo emisor de luz
FAT	Prueba de aceptación de fábrica
GUI	Graphic User Interface
HMI	Human Machine Interface

SIMBOLOGÍA

Ms	Milisegundo
VDC	Voltaje de corriente directa
Mm	milímetro
Gr	Gramos
%T	Temporizador (PLC)
%I	Entradas (PLC)
%M	Entradas de memoria (PLC)
%Q	Direcciones de salida (PLC)
%MW	Palabra Word (PLC)

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

1 INTRODUCCIÓN.

En la actualidad, el recurso físico (hardware) es limitado, para el desarrollo de pruebas en un entorno de simulación, donde, mediante la configuración precisa del PLC de SIEMENS, de igual manera, se pueden desarrollar una cantidad reducida de escenarios de prueba y error, y siempre bajo la supervisión del profesor del laboratorio.

La posibilidad de contar con un entorno de co-simulación que permita, entre otras cosas, provocar fallos que no comprometan la integridad de los equipos, a lo que le agregamos la factibilidad de ser implementado, al tratarse de una biblioteca que puede ser importada desde un sistema “real”, pudiéndose realizar estas pruebas de manera remota, accediendo a un servidor donde se pueda alojar el ambiente de co-simulación, convierten a este proyecto en una herramienta fundamental, ante la posibilidad de ser utilizada en el Laboratorio de Control de Procesos Industriales

Basándonos en las exigencias del Laboratorio de Control de Procesos Industriales, se va a diseñar un proyecto acorde a las exigencias de este frente académico, permitiendo inclusive, que la herramienta sea utilizada mediante escritorio remoto, alojado en la computadora del laboratorio donde sería instalada.

1.1 Descripción del problema.

El perfil del Ingeniero Electrónico, especializado en Automatización Industrial, exige contar con bases sólidas en el diseño e implementación de sistemas automatizados, de preferencia en tecnologías homologadas, y actualmente utilizadas por la industria.

En la Facultad de Ingeniería en Electricidad y Computación de la Escuela Superior Politécnica del Litoral (FIEC), se cuenta con el Laboratorio de Control de Procesos Industriales, mismo que, a su vez cuenta con la estación de almacenamiento intermedio IMS-10 de la planta Lucas-Nülle, configurable y administrable desde el terminal implementado en el aula.

La infraestructura mencionada, ha sido de inmensurable ayuda para la formación de los estudiantes en el campo del control de procesos, sin embargo, y debido al flujo de usuarios que la operan, se encuentra expuesta a situaciones que ponen en riesgo la integridad de este sistema, por diversos factores como, por ejemplo:

- Error humano en las secuencias ingresadas en el PLC.
- Deterioro físico de los componentes de la planta, atribuible al desgaste mecánico natural durante el tiempo de vida de la planta.
- Fallas en la operación del equipo, que puedan representar períodos fuera de servicio de la planta.

Por lo tanto, resulta un problema no contar con un PLC de forma física y una planta real para que los alumnos del Laboratorio de Control de Procesos Industriales, puedan tener un entorno de pruebas y fallas, donde se puedan realizar diagnósticos de errores y simulaciones de eventos adversos.

1.2 Justificación del problema.

En el Laboratorio de Control de Procesos Industriales, se cuenta con un CPU configurable desde el aplicativo STEP 7, incluido en el S7 PLCSIM 14, desde donde se ingresa la configuración que se ejecutará en el equipo físico, limitando las pruebas a los recursos específicos disponibles.

Por esto, se vuelve de gran ayuda la implementación de un entorno de co-simulación, que permita, entre otras cosas, llevar a cabo la ejecución de las secuencias ingresadas en el STEP 7, permitiendo, además, la creación de múltiples escenarios con los diferentes dispositivos incluidos en la biblioteca de este software, en tiempo real, desde cualquier computador.

Desarrollar un entorno de co-simulación, permitirá al estudiante, realizar una extensa variedad de escenarios de prueba, así como va a permitir comprobar en tiempo real el correcto funcionamiento, y la simulación de errores, sin comprometer la integridad de los dispositivos reales con los que se cuenta en el laboratorio. Así mismo, dicha solución, podrá implementarse en un sistema real (hardware), permitiendo desarrollar proyectos más eficientes.

1.3 Objetivos.

1.3.1 Objetivo General.

Diseñar el sistema de automatización de la planta de paletizado, IMS-10, del Laboratorio de Control de Procesos Industriales; que incluye la validación de su funcionamiento, mediante la implementación de una co-simulación entre, el PLC virtual SIEMENS® y el modelo de la estación de paletizado, en C#; para el análisis de diferentes pruebas, fallas y diagnóstico sobre el sistema automatizado, de forma virtual.

1.3.2 Objetivos Específicos.

- Implementar el programa de automatización en PLC SIEMENS S7-1500, con el programa SIMATIC S7 PLCSIM ADVANCED, para la simulación de funciones previamente programadas, utilizando la herramienta de configuración STEP7, ambas desarrolladas por SIEMENS®.
- Desarrollar el modelo de comportamiento de la planta de paletizado en lenguaje de programación C#, y ejecutarla en el programa Microsoft Visual Studio 2013®, a fin de simular las respuestas de los sensores de la planta Lucas-Nülle.
- Implementar la co-simulación entre PLC virtual y el modelo de la planta, usando una interfaz (API) para conectar el controlador virtual con la planta simulada, con lo que se realiza el testeo y validación de la misma.
- Diseñar el “HMI” (interfaz de usuario), de la serie Comfort TP900, para mostrar en tiempo real, las secuencias de funcionamiento de la Planta Lucas-Nülle, además de errores generados desde el panel de control del ambiente de co-simulación.

1.4 Marco teórico.

1.4.1 Co-simulación.

La base de nuestro proyecto es la co-simulación, por lo que nos hemos provisto de las herramientas necesarias para lograrlo, sin embargo, es necesario mencionar definiciones y reseñas sobre este tema.

La co-simulación también es conocida como simulación cooperativa, es una metodología de simulación que permite a componentes individuales ser simulados en ambientes

virtuales y ejecutándose simultáneamente, por lo que se tiene intercambio de información de manera colaborativa. [1]

Puede ser definido también como una simulación conjunta de sub-simuladores independientes acoplados levemente, con varios criterios a considerar, como lo son:

Existirá un algoritmo de co-simulación, que se encargará de la sincronización e interacción entre estos sub-simuladores. Esta sincronización se realizaría únicamente en los puntos de comunicación previamente definidos. Fuera de los puntos de comunicación, los sub-simuladores operan de manera independiente.

Se puede interpretar el comportamiento de estos subcomponentes, como el de cajas negras, cada uno maneja sus entradas y salidas (por medio de los puertos de comunicaciones), además de tomar un tiempo determinado en completarse y cuyas salidas se pueden utilizar como entradas de otras cajas negras.

En modelos estandarizados, como por ejemplo, el FMI (Functional Mock-up Interface), se indica que:

> “La co-simulación aprovecha la estructura modular de problemas acoplados en todas las etapas del proceso de simulación, comenzando con la configuración del modelo por separado y el pre-procesamiento para los subsistemas individuales, en diferentes herramientas de simulación” [2]

En esta definición podemos definir nuestro proyecto, partiendo del hecho que existen módulos acoplados, representados por los diferentes aplicativos utilizados, resaltando que cada uno de ellos maneja un componente de la co-simulación, por una parte, actuando bajo instrucciones, y por otra, simulando las señales de los sensores.

Famic Technologies Inc., en su sitio web, representa de forma gráfica (Figura 1.1), la arquitectura de una co-simulación mediante API.

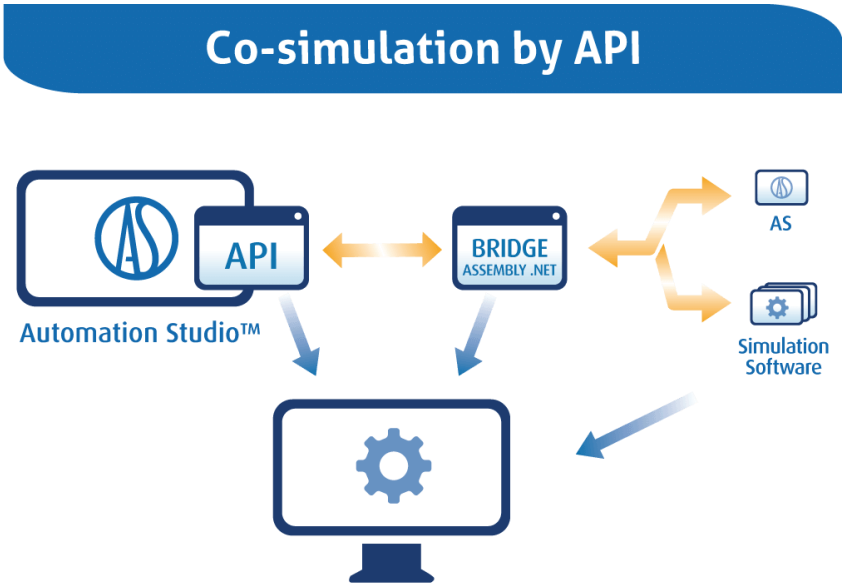


Figura 1.1 Co-simulación mediante API [6]

Automation Studio™ ejecuta su co-simulación con otras plataformas. Los usuarios pueden crear un entorno de prueba en lazo, utilizando API para simular sistemas multi-tecnología modelados, con otro software de simulación multi-físico complementario. También se puede comunicar con cualquier dispositivo mediante un bus, ya sea para probar el algoritmo de un controlador con una máquina virtual completa o para crear un entorno de formación realista. [3]

La co-simulación resulta convirtiéndose en una herramienta básica para el diseño de prototipos, basándose en soluciones reales, que permitirán elegir las mejores opciones entre modelos de dispositivos, dimensiones, velocidades, aceleraciones, y demás condiciones físicas o variables.

1.4.2 S7 PLCSIM ADVANCED

1.4.2.1 Visión general

Utilizando el software SIMATIC S7 PLCSIM ADVANCED, se pueden crear controladores virtuales para la simulación completa de funciones de una CPU S7-1500 o ET 200SP. Por lo tanto, no es necesario usar controladores reales para probar un programa STEP 7 (Figura 1.2). [4]

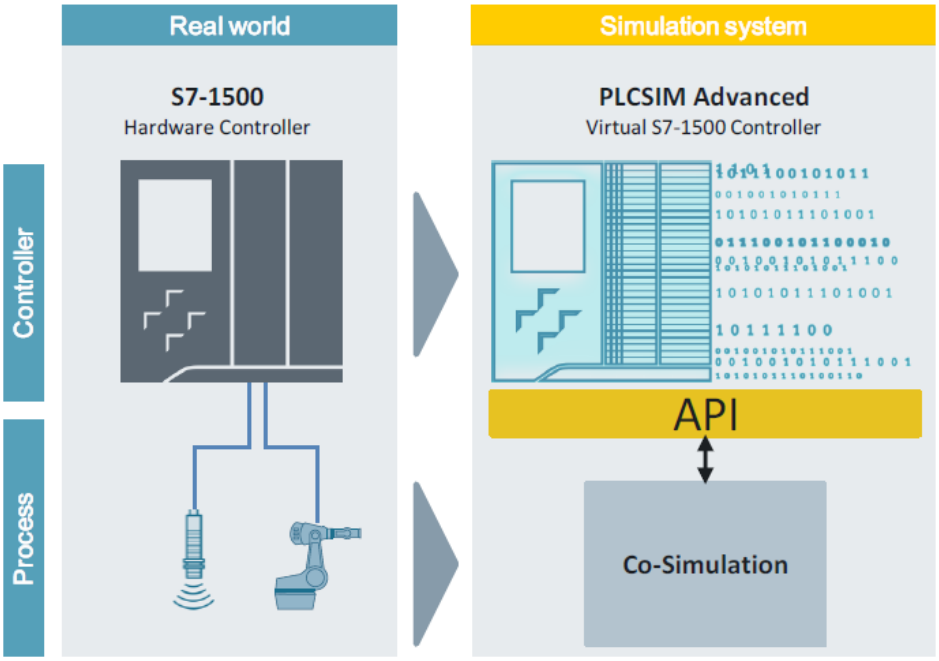


Figura 1.2 Visión General [4]

Por lo tanto, con S7 PLCSIM ADVANCED, es posible simular los programas de su CPU en un controlador virtual.

Utilizando esta modalidad, la de controlador virtual, se configura el CPU de la misma forma, mediante el programa STEP 7, siguiendo la secuencia habitual que es: Inicialmente, programar la lógica de su aplicación, para luego cargar la configuración de hardware y el programa en el controlador virtual. Desde este entorno se puede ejecutar la lógica del programa, observar los efectos de entradas y salidas simuladas y luego adaptar los cambios necesarios de acuerdo a la aplicación de los programas.

También es posible comunicarse a través de SOFTBUS, y vía Ethernet, ya que S7 PLCSIM ADVANCED proporciona una conexión Ethernet completa, pudiéndose realizar una comunicación distribuida.

Usando sistemas de simulación es posible desarrollar mayor diversidad de programas y la implementación de los mismos en el área de la producción. En el mundo de la automatización, tener la posibilidad de contar con un entorno de prueba simulado acorta los tiempos de puesta en marcha y mejora eficiencia de los procesos, pudiendo detectar con anticipación posibles errores. Al contar con SIMATIC S7 PLCSIM ADVANCED se tiene la posibilidad de probar el programa después de cambios de programación en el

controlador virtual, antes de que se cargue en el controlador real (PLC) que se usa físicamente en la planta, y finalmente, que la misma se ponga en funcionamiento.

S7 PLCSIM ADVANCED permite la interacción con programas o software nativos C++ o C# a través del uso de interfaz de usuario (API). [5]

1.4.2.2 Modo de operación – Visión General

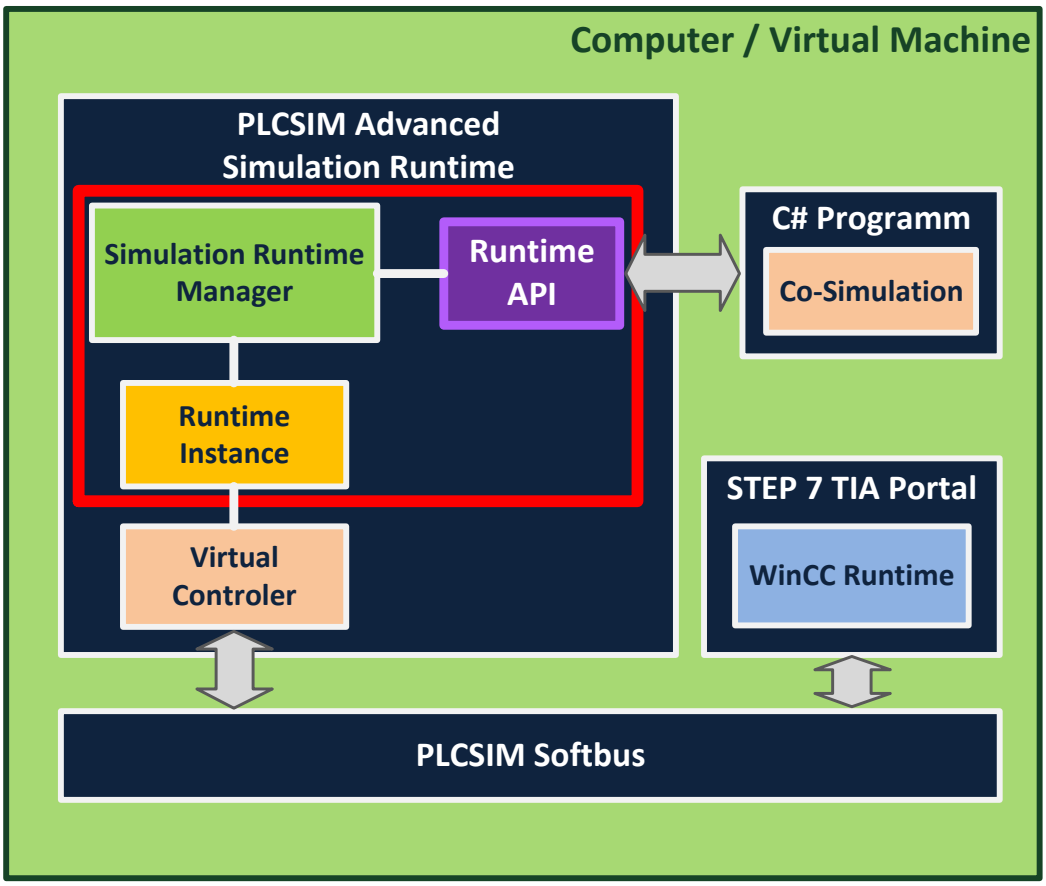


Figura 1.3 Visión general del principio de aplicación del proyecto [5]

En la Figura 1-3 se entiende que, con el S7 PLCSIM ADVANCED, como sistema de simulación independiente, la programación creada puede ser simulada y comprobada en el controlador virtual.

El *runtime* del API de S7 PLCSIM ADVANCED, provee de interfaces de usuario para acceder a la simulación, se encuentran las siguientes opciones por medio de programación:

- Creación de una instancia de tiempo de ejecución del controlador virtual.

- Cambio de modo del controlador virtual.
- Intercambio de datos de entrada y salida con una co-simulación.

1.4.2.3 Propiedades Generales

Comunicación local y distribuida. Aparte de la comunicación mediante SOFTBUS, el S7-PLCSIM AVANCED, ofrece conexión Ethernet completa, además de desarrollar comunicación distribuida. Para la comunicación entre el STEP 7 y las instancias del S7 PLCSIM ADVANCED, En las tablas 1-1 y 1-2, se detallan las opciones de comunicación disponibles [5]:

Tabla 1-1 Rutas de Comunicación [5]

Ruta de comunicación	Local		Distribuida
Protocolo	Softbus	TCP/IP	TCP/IP
Interfaz de comunicación en S7 PLCSIM ADVANCED	PLCSIM	Adaptador Ethernet Virtual PLCSIM	Adaptador Ethernet Virtual PLCSIM
STEP7 e instancias	En PC o máquina virtual	En PC o máquina virtual	Distribuidas

Tabla 1-2 Opciones de Comunicación [5]

Opciones de comunicación	Local		Distribuida
Protocolo	Softbus	TCP/IP	TCP/IP
Entre STEP 7 e instancias	Si	Si	Si
Entre instancias	Si	Si	Si
Vía OPC UA y web server	No	Si	Si
Entre una instancia y un CPU real	No	No	Si
Entre una instancia y una HMI real	No	No	Si
Entre una instancia y HM simulada	Si	Si	No

Comportamiento de tiempo virtual, EL controlador virtual utiliza internamente 2 tipos de relojes para la simulación:

- Reloj Virtual. - Forma las bases del programa y es utilizado por los componentes que dependen de control de procesos (por ejemplo, OB's cíclicos, monitoreo cíclico, tiempo de ciclo mínimo, tiempo del sistema y cálculos de tiempo). El reloj virtual puede acelerarse o desacelerarse.
- Reloj Real. - No puede acelerarse o desacelerarse, y es utilizado por los componentes que no son dependientes de control de proceso, como, por ejemplo, la comunicación con el STEP7.

Con el S7 PLCSIM ADVANCED se puede acelerar y desacelerar el reloj virtual para propósito de testeos.

1.4.2.4 Campos de aplicación

Podemos utilizar el S7 PLCSIM ADVANCED, en las siguientes aplicaciones, ya sea para el campo industrial, como con fines educativos.

- Programa de simulaciones en lazo para herramientas de puesta en marcha virtuales
- En combinación con software de terceros:
    - Simulación de máquinas y plantas de producción
    - Simulación combinada de automatización y mecánica [4]
- Testeo de funcionamiento del programa STEP 7 – también en el contexto de máquina / sistema
- Validación de funciones virtuales hasta la puesta en servicio virtual
- Entrenamiento para la automatización con el PLC S7-1500, sin necesidad de hardware
- Prueba de aceptación de fábrica (FAT) [5]

1.4.3 Ventajas del uso de S7 PLCSIM ADVANCED

Con S7 PLCSIM ADVANCED como sistema de simulación independiente, el usuario puede simular y realizar testeos de la programación con el controlador virtual. El *runtime*

API del S7 PLCSIM ADVANCED, provee de interfaces de usuario para acceder a la simulación en *runtime*. Esas interfaces de usuario ofrecen las siguientes ventajas por medio de la programación en C# [4] [5]:

- Crea instancias para corridas de un controlador o PLC virtual.
- Cambio de modo en el controlador virtual.
- Intercambio de datos de entrada/salida entre el controlador virtual y la planta.
- Se pueden realizar interacciones similares con distintos usuarios.
- Mejora en la calidad de los proyectos de automatización.
- Acelera el tiempo de comercialización.
- Reduce los tiempos de producción.
- Reduce el riesgo en la puesta en servicio.
- Evita los costos de hardware en entornos de simulación.
- Incrementa la eficiencia en el mantenimiento.
- Detección temprana de errores validación de funcionalidad, lo que representa una alta calidad en el código de programación del STEP 7
- No es necesario contar con un CPU físico, ahorrando costos de compra
- Mejora de la eficiencia mediante la optimización de las partes de la programación
- Tiempos cortos en el desarrollo, puestas en marcha, y comercialización.
- Posibilidad de entrenamiento académico, previo a la actividad profesional.

1.4.3.1 Diferencias y ventajas entre S7 PLCSIM ADVANCED frente a S7-PLCSIM

Respecto a su predecesor, el S7 PLCSIM V.14, EL S7 PLCSIM ADVANCED, incluye nuevas características, entre otras cosas, el uso de API para la co-simulación.

Se detallan estas diferencias en la tabla 1-3:

Tabla 1-3 Diferencias entre S7 PLCSIM ADVANCED y PLCSIM V14 [4]

Función	S7 PLCSIM ADVANCED V1.0	PLCSIM V. 14
Runtime **(Tiempo de ejecución)**	Independiente	Junto con STEP 7
Interfaz de usuario	Panel de control	"Look&Feel" de TIA Portal
Series de CPU soportados	S7-1500(C,T,F), ET 200SP y ET 200SPF	S7-1200(F), S7-1500(C,T,F), ET 200SP y ET 200SPF
API para co-simulación	✓	×
Web server	✓	×
OPC UA	✓	×
Diagnóstico de procesos	✓	×
Comunicación S7	✓	Via SOFTBUS
Comunicación abierta de usuario	✓	Via SOFTBUS
Trazados	✓	✓
Animación	✓	✓
Bloques protegidos (KHP)	✓	×
Instancias múltiples	Hasta 16	Hasta 2
Instancias Distribuidas	✓	×
Tiempo virtual	✓	×
Conexión de CPUs/HMIs reales	✓	×
Uso de DNS	✓	×
Tarjeta de memoria virtual	✓	×

1.4.4 Seguridades de S7 PLCSIM ADVANCED (restricciones para seguridad)

Es importante tener en cuenta las siguientes restricciones durante el uso del S7 PLCSIM ADVANCED:

1.4.4.1 Autenticación:

- Las interfaces de usuario (API), no tienen opciones para autenticación y autorización, por ende, no existe protección usando cuentas de usuario o contraseña.
- El *Runtime Manager (*administrador del tiempo de ejecución), no está protegido por autenticación.

1.4.4.2 Comunicación:

- La comunicación durante simulación multi-computadores, no es encriptada.
- Un puerto TCP/IP se abre para la comunicación de red cruzada en el computador.
- El componente WinPCap (requisito de instalación), provee acceso a la red de comunicaciones TCP/IP

1.4.5 C# - Visual Studio

1.4.5.1 Lenguaje C#

El lenguaje de programación C# (leído “C Sharp”), es un lenguaje creado por Microsoft para su plataforma .NET, tiene la característica de ser un programa seguro y orientado a objetos. Este lenguaje fue creado posteriormente a C++ y Java, por tal motivo combina y mejora gran parte de las características más interesantes y conocidas de ambos lenguajes, es decir, que el programador que conozca a fondo el lenguaje C#, no va a tener restricciones en poder programar tanto en a C++ como en Java. [6]

Actualmente se usa para desarrollar aplicaciones web basadas en ASP.NET, formularios de Windows y aplicaciones de escritorio basadas en WPF (Windows Presentation Foundation), actualmente también se utiliza este recurso para la creación de aplicaciones Windows en la telefonía y a su vez para las aplicaciones de Android.

Sin duda alguna, el software más recomendado para la creación de aplicaciones en el lenguaje de programación C#, es el ambiente integrado de Microsoft Visual Studio, adecuado para el desarrollo de aplicaciones profesionales. [7]

Hay que recordar también, que aparte de tener un alcance completo sobre la Web y ser uno de los más populares en el ambiente de internet, el mismo es también disponible para la creación y desarrollo de programas de propósito general, ya que en los últimos tiempos este lenguaje ha sido uno de los más utilizados para el desarrollo de aplicaciones. Sin importar la complejidad que tenga cualquier aplicación, C#, le proporciona al programador el nivel de abstracción preciso para poder abordar cualquier tipo de desarrollo, y a su vez permite desarrollar aplicaciones complejas con facilidad y rapidez sin sacrificar la potencia y el control que ofrecen otros lenguajes como C, C++ y Java.

Otra ventaja de este lenguaje es que elimina errores de programación comunes como, por ejemplo:

- “El “garbage collector” libera al programador del peso que conlleva el manejo manual de la memoria”
- “Todos los objetos creados dinámicamente, así como las matrices son iniciados a cero, y aunque C# no inicia automáticamente las variables locales, el compilador avisará cuando se intente utilizar una antes de iniciarla”
- “C# unifica el sistema de tipos permitiendo ver a cada uno de ellos en el lenguaje como un objeto”

En resumen, concluimos que con C# es posible trabajar con todo tipo de datos, crear estructuras de forma dinámica, trabajar con ficheros, acceder a bases de datos, diseñar interfaces gráficas de usuario, esto permite al programador generar programas modulares y fácilmente adaptables, ideales para la aplicación industrial, dónde podremos ir escalando a otras etapas del proceso. [8]

1.4.5.2 Microsoft Visual Studio

Este programa es un entorno de desarrollo de aplicaciones eficaces y de alto rendimiento, que cuenta con un conjunto de herramientas y varias tecnologías, creado

por la compañía Microsoft y disponible para sistemas operativos Windows, Linux y MacOS, y a su vez es posible el desarrollo de aplicaciones en múltiples lenguajes de programación, como lo son, C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP, y también a entornos de desarrollo web, como ASP.NET. [9]

Permite al programador o desarrollador crear sitios y aplicaciones web, así como aplicaciones de escritorio y aplicaciones móviles, en cualquier entorno que soporte la plataforma.

Los lenguajes de programación antes descritos utilizan el mismo entorno de desarrollo integrado (IDE), y Visual Studio permite habilitar el uso compartido de herramientas y por ende facilitar la creación de soluciones en varios lenguajes.

Al utilizar este programa contamos con las siguientes ventajas:

- Mayor productividad: Correcciones y mejoras de código, navegación y depurado; ya que existe un ahorro significativo de tiempo y recursos, al realizar en las tareas cotidianas en la programación sin importar el lenguaje o la plataforma que se utilice. Visual Studio agiliza y acelera el flujo de código con nuevas características en tiempo real.
- Azure: Facilita la configuración, compilación, depurado y “package”, al contar de forma integrada en la suite de las herramientas de Azure, ya que permite al programador crear de forma fácil aplicaciones “cloud first” bajo Microsoft Azure.
- Desarrollo móvil: En conjunto con Xamarin hace más rápido y fácil para el programador la compilación, conexión y ajuste de aplicaciones móviles para Android, iOS y Windows. [10]

1.4.6 Entorno de Co-simulación: API

La nomenclatura API, proviene de sus siglas en inglés “Application Programing Interface”, que en español significa Interfaz de Programación de Aplicaciones.

Hemos recopilado el siguiente concepto de API:

API es un equipo electrónico, programable en lenguaje no informático, diseñado para controlar en tiempo real y en ambiente de tipo industrial procesos secuenciales. [11]

En conclusión, es un conjunto de funciones o procedimientos que utiliza la programación orientada a objetos, con diversos lenguajes de programación, para tener comunicación con otras aplicaciones, mediante el llamado de distintas bibliotecas.

1.4.6.1 Interfaces de usuario (API)

A través de las interfaces de usuario S7 PLCSIM ADVANCED, se habilita la interacción con aplicaciones externas, por ejemplo, el propio C++/C#, así como softwares para la simulación de maquinaria y sistemas de producción. Con estas funciones es posible crear instancias, cambiar el modelo de los controladores virtuales y el intercambio de datos de entrada/salida (I/O Data).

Es posible acceder a la simulación del *Runtime* (Tiempo de ejecución), por medio del API. Existe un administrador de este segmento de la co-simulación. Finalmente, las instancias se cargan en las bibliotecas del controlador virtual.

La figura 1.4 presenta esquemáticamente el acceso de aplicaciones externas a la simulación del *Runtime*, mediante el API. Una aplicación externa puede ser otro software de simulación o un GUI, por ejemplo.

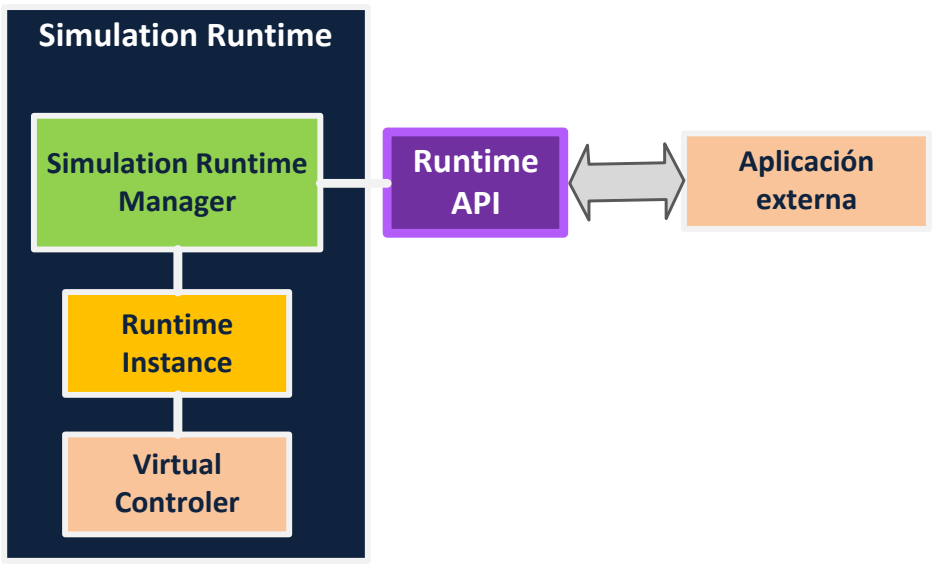


Figura 1.4 Aplicación externa y simulación del *Runtime* (tiempo de ejecución) [5]

1.4.6.2 Componentes de la simulación del *Runtime*

La tabla 1-4 muestra los componentes relevantes para el manejo de la simulación del *Runtime* del S7 PLCSIM ADVANCED [5]:

Tabla 1-4 Descripción de los componentes [5]

Componente	Descripción
SIEMENS.Simatic.Simulation. Runtime.Manager.exe	Un proceso de Windows que se ejecuta en Segundo plano. Componente principal del *Runtime*, que administra todos los demás componentes de este servicio. El proceso es iniciado automáticamente tan pronto como se intenta inicializar el *Runtime* API. El proceso finaliza automáticamente apenas no haya aplicaciones en ejecución que inicialicen el servicio.
SIEMENS.Simatic.Simulation. Runtime.Instance.exe	El proceso de la instancia que carga un DLL de un controlador virtual. Cada uno de los controladores virtuales crea su propio proceso.
SIEMENS.Simatic.Simulation. Runtime.Api.x64.dll	Bibliotecas API que tiene que cargar una aplicación a fin de ser utilizada por la simulación del *Runtime*. Las bibliotecas incluyen interfaces para códigos nativos y administrados. Los DLL's son basados en .NET Framework 4.0
SimulationRuntimeApi.h	Encabezado del archivo que describe todos los tipos de datos que requiere una aplicación nativa de C++ a fin que sea utilizada por la biblioteca API.

1.4.7 Descripción de la planta Paletizadora

A continuación, se describen cada uno de los elementos de la planta paletizadora física, que se encuentra en el Laboratorio de Control de Procesos Industriales, de la FIEC, mismos que serán emulados en la ejecución de este proyecto, con las opciones disponibles en los simuladores.

1.4.7.1 Interfaz gráfica: SIEMENS TP900, SIMATIC HMI Comfort Panels [12]

Figura 1.5 SIMATIC HMI TP900 Comfort, panel táctil

La interfaz gráfica que se emula en este proyecto, es la SIEMENS, de la serie SIMATIC HMI, modelo TP900 (Figura 1.5), de la que hemos reunido sus características principales en el siguiente listado:

- Pantalla TFT *widescreen* 19" de diagonal (con 16M de colores); funcionalidad homogénea de gama alta con archivos, scripts, visor PDF/Word/Excel, Internet Explorer, Media Player y servidor web.
- Iluminación variable de 0 a 100 % vía PROFIenergy, desde el proyector HMI o desde un controlador.
- Máximo rendimiento para actualizar los sinópticos en un tiempo mínimo.
- Idoneidad para los entornos industriales más severos con homologaciones especiales como ATEX 2/22 y homologaciones para la industria naval.
- Variantes de teclas con LED en cada tecla de función y nuevo mecanismo de introducción de textos, similar al de los teclados de los teléfonos móviles.
- Teclas con vida útil de 2 millones de pulsaciones.
- Configuración con el software de ingeniería WinCC, del entorno de ingeniería TIA Portal.

1.4.7.2 Procesador: CPU SIEMENS 1516-3 PN/DP [12]

Figura 1.6 CPU SIEMENS 1516-3 PN/DP

Para la ejecución de las instrucciones, se emulará el procesador de Marca SIEMENS, modelo de la serie SIMATIC S7-1500 (1516-3) (Figura 1.6).

- Solución de sistema para una variedad de aplicaciones de automatización discretas.
- Configurable exclusivamente en el portal Totally Integrated Automation (TIA), con STEP 7 Professional V12 o superior.
- CPU con gran memoria de programa y de datos en la gama de productos de los controladores S7-1500, para aplicaciones con requisitos elevados en cuanto a volumen de programas y conectividad.
- Interfaz PROFINET IO IRT con switch de 2 puertos.
- Funcionalidad I-Device de PROFINET para conectar la CPU a modo de dispositivo inteligente PROFINET con un controlador SIMATIC o PROFINET I/O no SIEMENS.
- Otra interfaz PROFINET integrada con dirección IP independiente para aislar la red, para conectar otros dispositivos PROFINET IO RT o para comunicación rápida en calidad de I-Device.
- **Interfaz maestro PROFIBUS DP.**
- Servidor y cliente UA como opción *runtime*, para integrar con facilidad SIMATIC S7-1500 en sistemas y equipos no SIEMENS.

- Modo isócrono a nivel centralizado y descentralizado en PROFIBUS y PROFINET
- Funciones Motion Control integradas para controlar ejes de velocidad y ejes de posicionamiento; compatibilidad con encoders externos, levas/perfiles de levas y detectores.
- Servidor web integrado para el diagnóstico y con la posibilidad de crear páginas web definidas por el usuario.

1.4.7.3 Segmento doble de banda transportadora con motor de 24V [13]

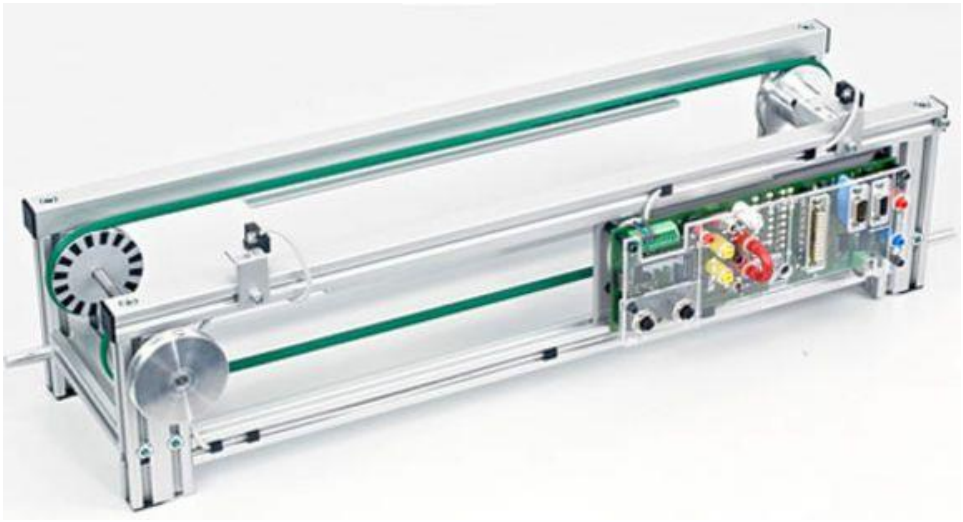


Figura 1.7 Segmento de cinta transportadora doble, motor de 24 V

La “banda transportadora” (Figura 1.7), desde donde además de comandarse el avance del pallet hasta su almacenamiento, se conocerá la ubicación del pallet, sea este al inicio, o al final, gracias a los sensores con los que cuenta. A continuación, listamos sus características principales, obtenidas desde el sitio web del fabricante alemán Lucas-Nülle:

Módulo mecatrónico básico, controlado por un motor de velocidad variable de 24-V, con sensores de fin de carrera y Módulo integrado de PROFIBUS. Diseñado para experimentos básicos o para sistemas mecatrónicos que controlan flujo de materiales, es decir, puede ser usado de forma individual, o más compleja, como subsistema de un sistema más complejo. Plantas de Almacenamiento Intermedio pueden ser conectadas directamente a la banda y controladas de manera conjunta por medio de PROFIBUS.

- Longitud = 600 mm, ancho = 160.

- Ancho de la cinta = 120 mm.
- Motor Reductor, 24 V DC.
- Control de velocidad por modulación de ancho de pulso.
- Ajuste de velocidad continuo por medio de un potenciómetro o entrada análoga de 0-10 V.
- Cambio de sentido por medio de interruptor manual.
- 2 Sensores de fin de carrera.
- 2 x M12 interfaces para sensores /actuadores adicionales.
- Sockets para circuitos de corte de emergencia (desconexión de todo el voltaje hacia los módulos de salida).
- Fuente de alimentación externa, mediante conectores de seguridad de 4mm o conectores de poder coaxiales.
- Conector de 9 pines para contactores LOGO o PLC.
- Disco con *encoder* incremental para detección de posición y velocidad por medio de sensores ópticos.
- Visualización como modelo interactivo en 3D en bases de dato de IMS-virtual
- Requisitos de control: entradas digitales 4, salidas digitales 3.

1.4.7.4 Estación de almacenamiento intermedio (IMS) [14]

Figura 1.8 Estación de almacenamiento de materiales

Se trata de una estación para almacenamiento de piezas transportadas dentro de un Sistema mecatrónico complejo, y para flujo controlado de material. Cuando se integra

con una banda transportadora, puede almacenar hasta 4 pallets con carga, o 10 pallets descargados, dentro de su sección de almacenamiento temporal (Figura 1-8).

- Recámara para almacenamiento, hasta 4 pallets cargados o 10 descargados.
- Micro-switch para el monitoreo del nivel.
- Cilindro de parada de doble acción.
- Sistema de elevación telescópico; elevación paralela, a través de 2 cilindros de elevación, incrementada por un cilindro a prueba de giro.
- Dos cilindros de acción simple, operados paralelamente.
- 6 Sensores de fin de carrera.
- 4 Válvulas neumáticas.
- 1 Válvula de 3/2.
- 3 Válvulas de 4/2.
- Interfaz PLC con conector SUB-D de 25-pines.
- Requisitos PLC: 6 Salidas digitales, 4 Entradas digitales.

Se requiere una unidad de mantenimiento, para asegurar la funcionalidad y mayor tiempo de vida de los componentes neumáticos. Esta consiste de filtros, válvulas de control de presión con manómetro, y lubricación.

CAPÍTULO 2

2 METODOLOGÍA

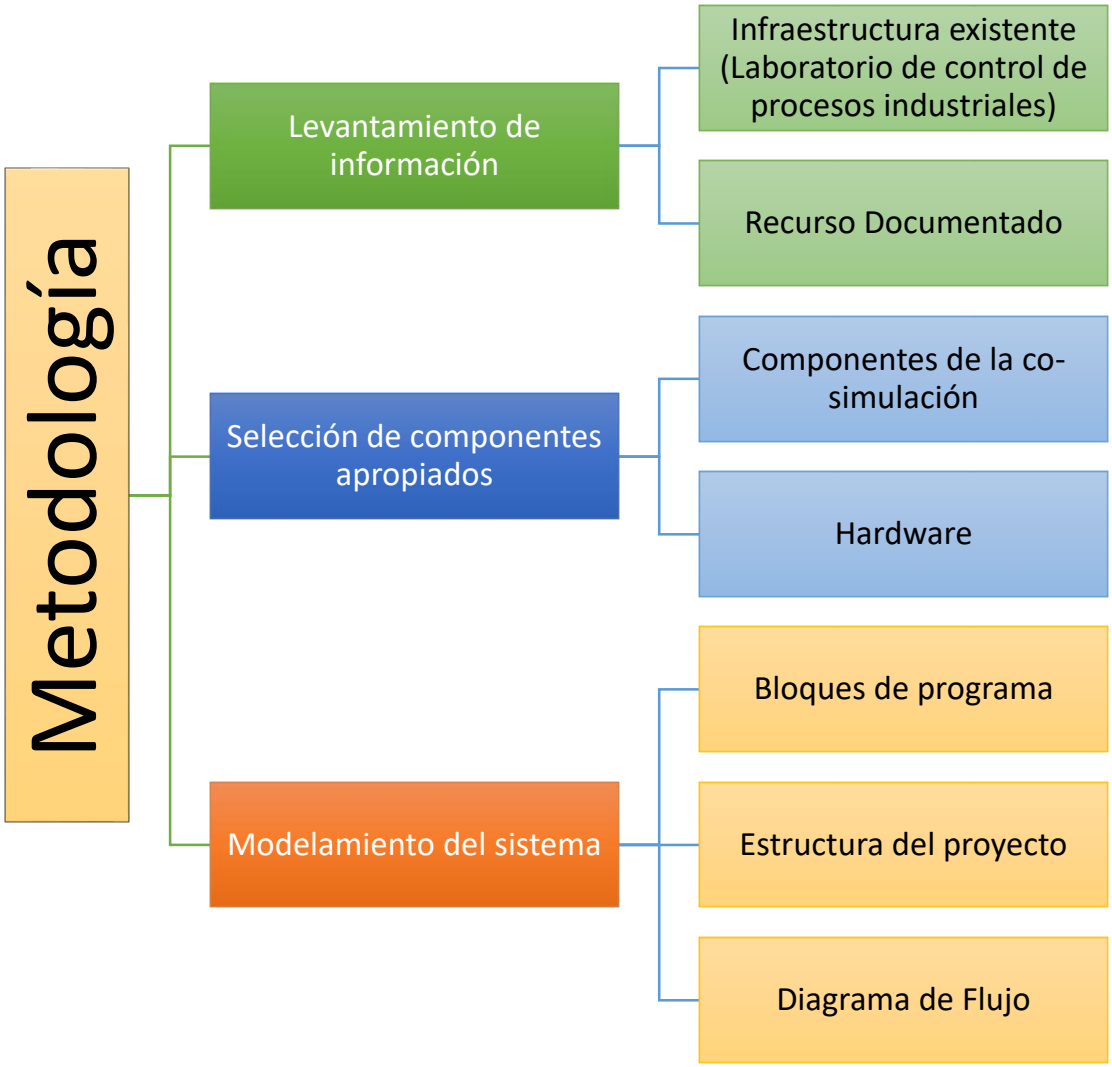


Figura 2.1 Esquema de la Metodología aplicada en este proyecto

Como parte de la ejecución del proyecto, fue necesario dirigirse al repositorio de ESPOL, DSPACE que se lo puede encontrar en la dirección https://www.dspace.espol.edu.ec/, así como recurrir a las prácticas del Laboratorio de Control de Procesos Industriales, y contactar al personal académico para obtener detalles de:

- Infraestructura existente,
- Situaciones adversas más comunes,
- Recursos lógicos existentes,

- Sugerencias de soluciones que sería conveniente implementar, etc.

De esta manera se logró desarrollar la solución entregada (Figura 2-1), cumpliendo con gran parte de las demandas del laboratorio, enfocándose principalmente en que la característica modular del proyecto, que permite entre otras cosas, darle el enfoque académico a este proyecto.

Se planteó un esquema conformado por 3 etapas principales, siendo estas:

1. Levantamiento de información
2. Selección de componentes apropiados
3. Modelamiento del sistema

2.1 Levantamiento de Información

Tomando en cuenta que el proyecto desarrollado comprendía principalmente la simulación de parte del Laboratorio de Control de Procesos Industriales de la FIEC, la información necesaria consistió en:

- Prácticas del Laboratorio de Control de Procesos Industriales
- Fichas técnicas de los componentes de la planta paletizadora, desde el CPU con el que cuenta (PLC S7-300), pasando por la HMI (Panel View TP 700), así como de los actuadores y sensores de la planta IMS10.

Cabe señalar que la información recopilada de estas fuentes ha sido debidamente citada durante el desarrollo de este proyecto, cumpliendo con los prescritos de propiedad intelectual que deben de acatarse.

2.1.1 Infraestructura existente, Laboratorio de Control de Procesos Industriales

Del sitio web oficial de la FIEC obtuvimos una breve reseña del Laboratorio:

“El Laboratorio de Control de Procesos Industriales cuenta con un sistema de producción de six-pack de botellas (IPA 26), compuesta de cinco estaciones de trabajo modulares de la marca Lucas-Nülle y un brazo robótico de la marca Kawasaki.” [15]

Este proyecto se desarrolló sobre dos de las cinco estaciones existentes, como son, el segmento de transporte y el segmento almacenamiento de pallets (Figura 2-2).

Los sensores y actuadores de esta planta se comunican a través de los esclavos PROFINET, al procesador PLC SIEMENS S7 300.

Finalmente, la Interfaz gráfica con el Usuario o Interfaz Hombre máquina, se da por medio de la pantalla SIEMENS TP700.

Las interconexiones entre estos componentes se realizan mediante los enlaces:

- PROFINET, entre El CPU y la HMI, y
- PROFIBUS, entre CPU y Planta (esclavos: actuadores y sensores)

En este proyecto, la comunicación entre componentes se realiza sobre el aplicativo que lo simula: PLCSIM SOFTBUS.

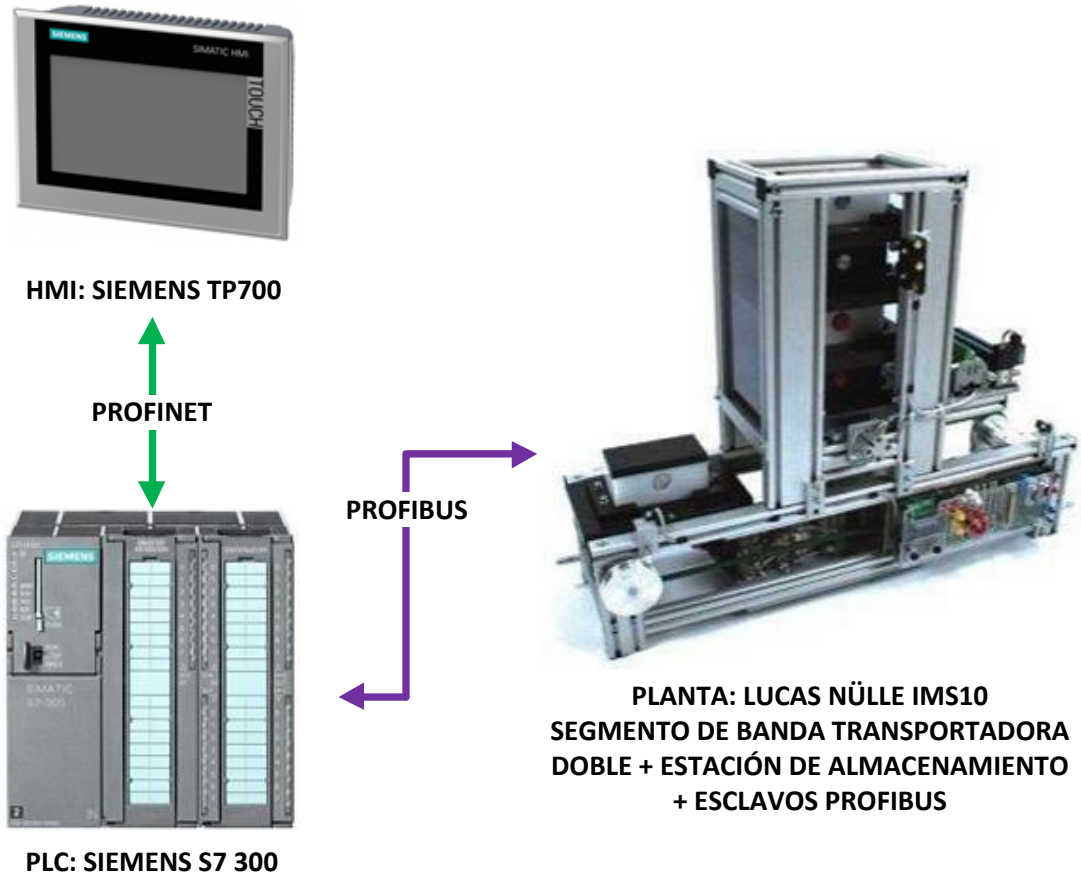


Figura 2.2 Componentes del Laboratorio de Control de Procesos Industriales de la FIEC

2.1.2 Recurso documentado

Es importante señalar que SIEMENS, proporciona programas ejemplo para su ejecución en los aplicativos correspondientes, donde se obtuvieron las instrucciones básicas, que permitieron la comprensión y posterior ejecución del proyecto, además del manual de funciones del software de co-simulación SIMATIC S7-1500 S7 PLCSIM ADVANCED

En cuanto a material académico utilizado, se recurrió a tesis de grado, así como prácticas del Laboratorio de Control de Procesos Industriales, de éstas se obtuvieron soluciones ya implementadas, utilizadas como plantillas. Una observación que es importante hacer al respecto, en ambos casos, las prácticas tuvieron un enfoque para una implementación física, y se desarrollaron con el hardware existente, además de aplicaciones, en una versión anterior a la utilizadas en este proyecto, por lo que fueron necesarios ciertos ajustes como, por ejemplo, migraciones de versión.

2.2 Selección de componentes apropiados

2.2.1 Componentes de la Co-simulación

Considerando que la co-simulación, en el entorno deseado, era posible únicamente a través del software **S7 PLCSIM ADVANCED**, se sobreentiende la elección de este programa, se encuentran detalles en el epígrafe del mismo nombre.

La co-simulación desarrollada conjuga, Hardware simulado, y Software que compiló y ejecutó los programadas desarrollados.

2.2.1.1 Hardware (Simulado)

Estuvo compuesto por:

- Procesador (PLC): CPU 1516 PN/DP
- Interfaz de usuario (HMI): TP900 Comfort

Ambos elementos ya fueron descritos en los respectivos epígrafes del mismo nombre.

2.2.1.2 Software

Los aplicativos utilizados para la programación del hardware simulado, visualizar el HMI, puesta en marcha del Hardware simulado, y finalmente para simular la planta, fueron:

- STEP 7 Professional V15.1, herramienta encontrada dentro del TIA Portal V15.1, utilizado para la programación y configuración del controlador virtual, que determina las condiciones, instrucciones y secuencias con las que trabajará el PLC.
- WinCC RT Start, entorno de desarrollo de SIEMENS en el marco de los SCADA, para visualización y control de procesos industriales.
- S7 PLCSIM ADVANCED V2.0 SP1, permite se conecten mediante SOFTBUS, o una interfaz Ethernet Virtual, los componentes del Sistema, y ajustar entre otras cosas, los *runtimes* (siendo posible hacer escala del tiempo de ejecución respecto al tiempo real), la comunicación, además de notificar el estado de las instancias.
- Microsoft Visual Studio Ultimate 2013, entorno de desarrollo integrado, donde se simulaban las interacciones de la planta.

Como componente opcional, y con fines de realizar una comprobación inicial de la programación y configuración del controlador virtual, se recomienda el uso del S7-PLCSIM V15.1, para visualizar paso a paso las secuencias del PLC, teniendo en cuenta que no puede correr de manera simultánea con el S7 PLCSIM ADVANCED.

2.2.2 Hardware

Respecto a los requerimientos del sistema donde se instale el proyecto, es válido señalar que el mismo debe ser de un relativo alto rendimiento, esto porque parte de la co-simulación consiste en la simulación de procesamientos, alojamientos de memoria, transferencia de datos en red, animación de la ejecución de la simulación; desde una vista en bloques, y desde el HMI simulado. Todo lo anterior representa una carga muy alta para un computador con prestaciones, más bien modestas. El equipo donde se implemente el proyecto, debe cumplir los requerimientos mínimos detallados en la tabla 2-1 a continuación:

Tabla 2-1 Requerimientos mínimos del sistema [5]

Hardware / Software	Requerimiento
Procesador	2.2 GHz Intel® Celeron® Dual Core
RAM	• 4 GB para una instancia • 8 GB para 4 instancias
Espacio libre en disco duro	5GB
Hardware / Software	Requerimiento
Sistema operativo de 64 bits	• Windows 7 Home Premium SP1 • Windows 7 Professional SP1 • Windows 7 Enterprise SP1 • Windows 7 Ultimate SP1 • Windows Server 2012 R2
Resolución de pantalla	1024 x 768

El computador utilizado contaba con los siguientes recursos:

- Procesador: Core I5 1035G1 (4 núcleos, 10ma generación)
- RAM: 12 GB
- HDD: 1TB
- S.O.: Windows 10 de 64 bits
- Resolución: 1366 x 768

2.2.3 Interfaz Gráfica de la Planta IMS10

Para la elaboración de la interfaz gráfica, se recurre a la Práctica 2 del Laboratorio de Control de Procesos Industriales, adjunta a este documento.

Adicional a lo detallado en la práctica, se procedió a crear secuencias adicionales para este proyecto como lo fueron: La alarma de pallet atorado, y el parpadeo intermitente del pallet en la banda.

Se ajustaron también los encabezados y logos.

2.3 Modelamiento del sistema

Se tomó como referencia para realizar este proyecto, el ejemplo de aplicación, disponible en la página de soporte a la industria del fabricante (figura 2-3), SIEMENS, disponible en el enlace: https://support.industry.SIEMENS.com/cs/document/109739660/simatic-s7%E2%80%91plcsim-advanced%3A-co%E2%80%91simulation-via-api?dti=0&lc=en-WW

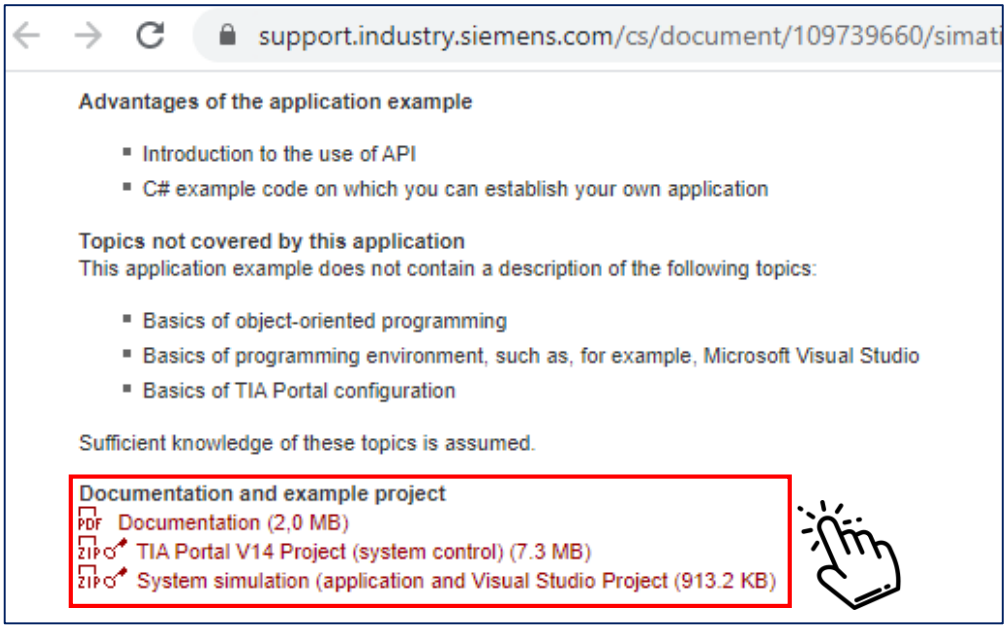


Figura 2.3 Enlaces de descarga de documentación y proyecto ejemplo

2.3.1 Bloques de Programa (estructura de programación en STEP7)

Es importante entender que un sistema diseñado mediante la suite de programas de SIEMENS, se organiza, para mayor eficiencia, como una programación estructurada. A continuación, detallaremos los bloques que conformaron la estructura de este proyecto:

- Bloques de Organización
- Bloques de Función
- Funciones
- Bloques de datos

En la tabla 2-2, a continuación, encontramos una breve descripción de los tipos de bloques que se instalaron en este proyecto:

Tabla 2-2 Tipos de Bloques en el programa de usuario [16]

Bloque	Descripción breve de la función
Bloques de Organización (OB)	Los OBs definen la estructura del programa de usuario.
Bloques de función (FB)	Los FBs son bloques con "memoria" que puede programar el mismo usuario.
Bloque	Descripción breve de la función
Funciones (FC)	Las FCs contienen rutinas de programa para funciones frecuentes.
Bloques de datos (DB)	Los DBs son áreas de datos que almacenan los datos de usuario. Adicionalmente a los datos asociados a un determinado bloque de función, se pueden definir también datos globales a los que pueden acceder todos los bloques.

Contienen partes del programa, por lo que se los conoce también como bloques lógicos. Su tamaño y número máximo dependerán del CPU (PLC Virtual) utilizado.

2.3.1.1 Bloque de Organización (OB) [17]

Los bloques de organización (OB) constituyen la interfaz entre el sistema operativo del controlador (CPU) y el programa de usuario. Estos bloques son llamados por el sistema operativo y controlan los procesos siguientes:

- Ejecución cíclica,
- Comportamiento en arranque del controlador,
- Ejecución del programa controlada por alarmas,
- Tratamiento de errores

En todo proyecto debe existir por lo menos un OB para la ejecución cíclica del programa. Se representa en el TIA PORTAL con el símbolo mostrado en la Figura 2-4:

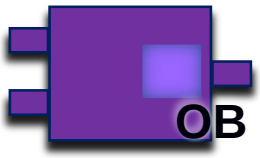


Figura 2.4 Símbolo de Bloque de organización en TIA PORTAL

2.3.1.2 Funciones (FC) [16]

Las funciones son bloques programables. Una función es un bloque lógico "sin memoria". Las variables temporales de las FCs se memorizan en la pila de datos locales. Estos datos se pierden tras el tratamiento de las FCs. Para fines de memorización de datos, las funciones pueden utilizar bloques de datos globales.

Como una FC no tiene asignada ninguna memoria, se han de indicar siempre parámetros actuales. A los datos locales de una FC no se pueden asignar valores iniciales. Se representa en el TIA PORTAL con el símbolo mostrado en la Figura 2-5:

Figura 2.5 Símbolo de Funciones en TIA Portal

2.3.1.3 Bloques de función (FB) [16]

Los bloques de función son bloques programables. Un FB es un bloque "con memoria". Dispone de un bloque de datos asignado como memoria (bloque de datos de instancia). Los parámetros que se transfieren al FB, así como las variables estáticas, se memorizan en el DB de instancia. Las variables temporales se memorizan en la pila de datos locales.

Los datos memorizados en el DB de instancia no se pierden al concluir el tratamiento del FB. Los datos memorizados en la pila de datos locales se pierden al concluir el tratamiento del FB. Se representa en el TIA PORTAL con el símbolo mostrado en la Figura 2-6:

Figura 2.6 Símbolo de Bloque de Funciones en TIA PORTAL

2.3.1.4 Bloques de datos globales (DB) [16]

Al contrario de los bloques lógicos, los bloques de datos no contienen instrucciones STEP 7. En cambio, sirven para depositar datos de usuario, es decir que los bloques de datos contienen datos variables con los que trabaja el programa de usuario. Los bloques de datos globales contienen datos de usuario utilizables desde otros bloques.

El tamaño de los DBs puede variar. El tamaño máximo admisible se indica en las descripciones del CPU. Se representa en TIA PORTAL con el símbolo de la figura 2-7:

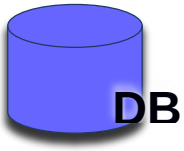


Figura 2.7 Símbolo de Bloque de datos en TIA PORTAL

2.3.2 Estructura del proyecto (programación STEP 7)

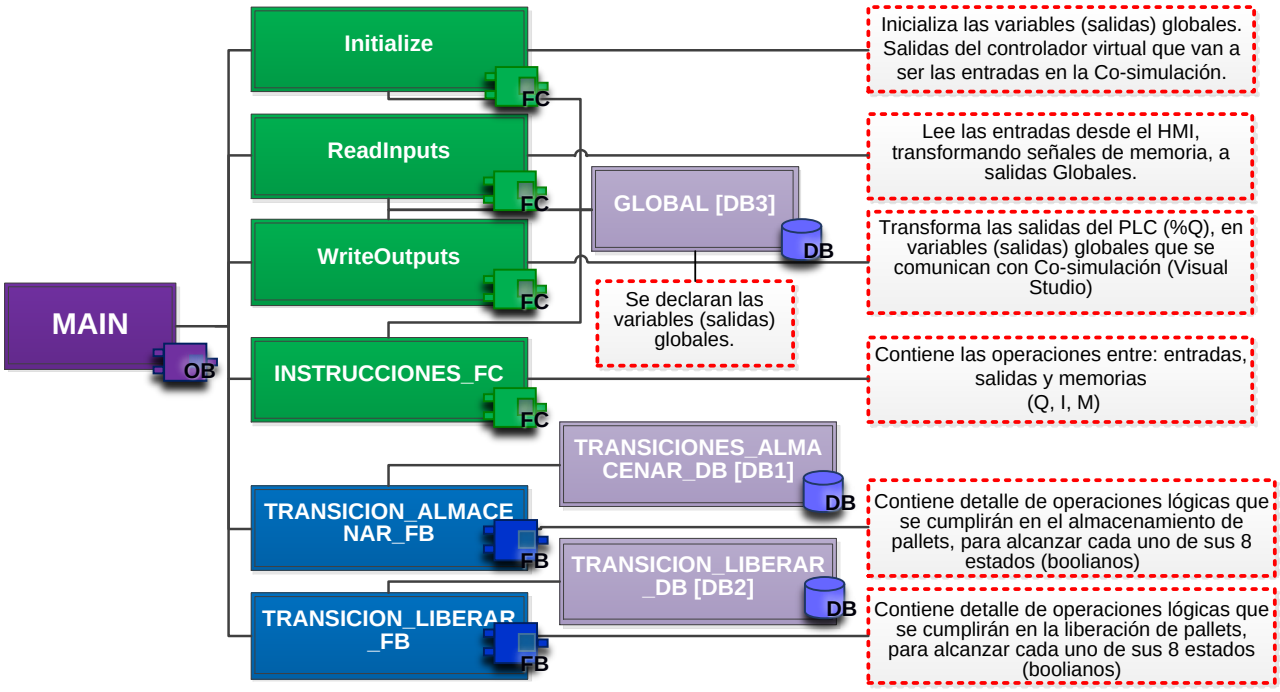


Figura 2.8 Estructura del proyecto (programación STEP 7)

La configuración del CPU mediante el STEP 7 de SIEMENS, incluye la creación de un Bloque de Datos Global (DB), que, en resumen, se trata de un recurso que podrá ser accedido por varias funciones, para nuestro caso particular, dicha base de datos nos permitió la declaración de instancias de co-simulación.

La configuración realizada (Figura 2-8), puede comprobarse mediante simulación, por medio del uso de la versión anterior del software, el S7 PLCSIM V. 15.1. La diferencia reside en que no se puede ejecutar un *runtime*, es decir una secuencia en tiempo real y de forma autónoma, sin embargo, es posible visualizar de forma gráfica, tanto en HMI como en las “Transiciones”, el avance entre los diferentes estados.

Tal como se describió en los conceptos anteriores, las instrucciones, secuencias, transiciones y estados, se almacenan en los Bloques de Función, Funciones, y Bloques de Datos; Es posible su configuración mediante lenguajes como GRAPHCET, o FUP, que es el caso de nuestro proyecto.

En cualquiera de los dos lenguajes mencionados, se aplica la lógica booleana en su estructura, por lo que el diseño previo mediante un diagrama de flujo, es posible; resaltando las ventajas en cuanto a simplicidad en el lenguaje y escritura que representa el uso de esta herramienta. Dicha lógica (la booleana), nos permitió elaborar la posterior tabla de Transiciones (sensores) vs. Salidas en PLC (actuadores), para la transcripción en C# de nuestra co-simulación.

2.3.3 Diagrama de flujo

En las siguientes páginas se muestra el diagrama de flujo desarrollado, así como su división entre las transiciones “ALMACENAR” y “LIBERAR”.

En esta primera parte (Figura 2-9), la etapa inicial se asume un estado de inactividad, hasta que se comande desde el teclado el inicio del programa (Q_INICIO_HMI), luego veremos que lo siguiente, será la posibilidad de liberar (Q_LIBERAR_HMI), o de almacenar pallets (Q_ALMACENAR_HMI), ambas comandadas desde el HMI, por ende, recibidas a memorias del PLC, corresponde co-simular las instrucciones hacia la planta simulada.

Nótese que luego de las órdenes de almacenar o liberar pallets, continúan el STEP 2 y el STEP 8 correspondientemente, esto debido a que la primera secuencia en completarse, será la de almacenar, pasando luego a la otra secuencia, es decir, la secuencia “Almacenar” comprende desde el STEP 2 hasta el 7 (incluida la secuencia creada de error); mientras que desde el STEP 8 hasta el 11 se contempla la secuencia “Liberar”.

Todos los sensores, y actuadores que intervienen en el siguiente diagrama, lo hacen también de forma activa en la co-simulación, es por ello que las instancias que los describen, se encuentran declaradas en el Bloque de Datos “Global”, desde donde las demás aplicaciones (Visual Studio y WinCC) pueden vincularse al sistema co-simulado.

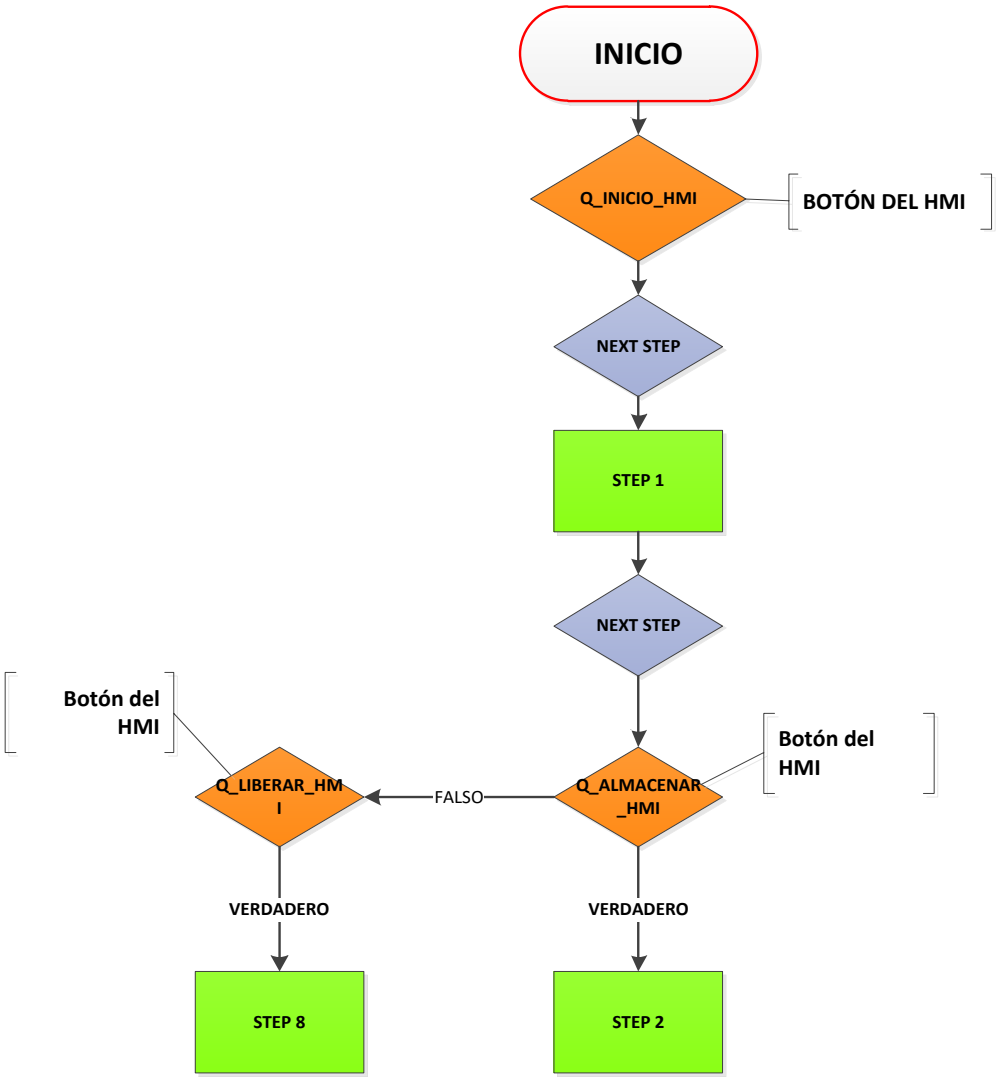


Figura 2.9 Diagrama de Flujo, Parte 1

2.3.3.1 Transición “Almacenar”

En la transición “Almacenar” Figura 2-10, se inicia el movimiento del motor, se detecta el paso del pallet por medio del sensor magnético a la izquierda de la banda transportadora I_IMS_10_B1 _SENSOR _IZQUIERDA.

En el paso siguiente, es posible simular un error, “PALLET ATORADO”, se crea una instrucción independiente en la función del PLC INTRUCCIONES_FC, se incluye además la opción de reconocer el error. Esto es posible ejecutarlo únicamente desde la Co-simulación (Visual Studio). Una vez validado, la secuencia, continúa.

Durante esta etapa se realiza el transporte y sensado del pallet hasta el punto de almacenamiento, activándose los cilindros paralelos.

Una vez que se complete la elevación, se recibe la señal de posición de “avance” de los cilindros paralelos. Esto se ve representado gráficamente en la herramienta WinCC, donde se reproduce la animación de la planta en tiempo real.

Figura 2.10 Diagrama de Flujo, transición Almacenar 1ra parte

Se realiza entonces las demás acciones relacionadas con la transición de almacenamiento, luego del STEP 4, como son: La activación del cilindro elevador Q_IMS_10_M3_CIL_NEU_ELEVADOR, mientras continúan encendidos los cilindros paralelos Q_IMS_10_M4_CIL_LAT_PARALELO, haciendo llegar el pallet a la posición de almacenamiento dentro de la cámara. De manera mecánica se abren las compuertas laterales que sostienen los pallets dentro de la torre, hablamos del sensor con TAG: Q_IMS_10_M5_CILINDRO_SEPARADOR.

En esta etapa es donde se registra en memoria, el conteo de pallets dentro del almacén, y por ende se muestra en la pantalla del HMI. La señal que activa el conteo es la recibida desde el sensor en posición de retroceso de los cilindros paralelos (STEP 5).

Si el almacenamiento de pallets ha llegado al máximo de su capacidad, se activará el sensor I_FIN_CARRERA, deteniendo el ciclo de almacenamiento. Si el usuario desea liberar pallets del almacén presionando el botón MARCHA/PARO del HMI, puede detener también el ciclo, dando nuevamente la opción de almacenar o liberar (STEP 1), este estado vendría a representar un STAND-BY del sistema. (Figura 2-11)

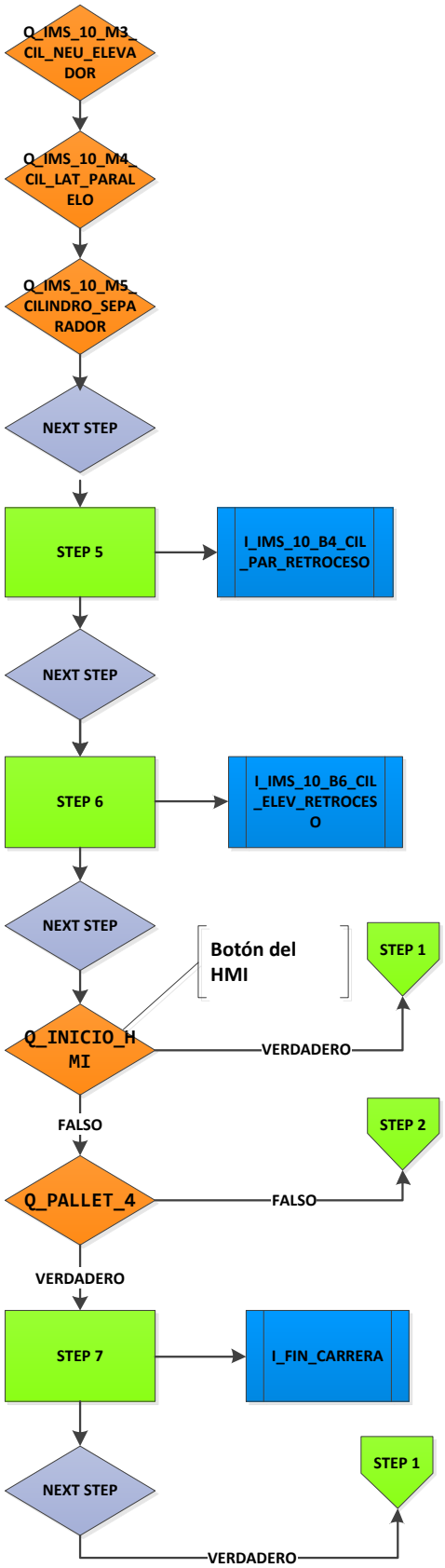


Figura 2.11 Diagrama de Flujo, transición Almacenar 2da parte

2.3.3.2 Transición “Liberar”

Es posible iniciar esta transición (Figura 2-12), en dos momentos; el primero de ellos se da al llenarse la capacidad del almacén (4 pallets), habiéndose activado el sensor de fin de carrera, quedando el sistema en STAND-BY, y presionando el botón “LIBERAR PALLETS” en el HMI; el segundo momento es cuando se cuente con al menos un pallet en el almacén, dado que, al encontrarse en el STEP 1, con un valor en memoria mayor (>) a 1 pallet, se habilita la opción de iniciar la transición.

Se inicia entonces desde el STEP 8, considerando que el STEP 7 se daba en la transición de almacenar, con la elevación de los cilindros paralelos, una vez que estos alcancen la posición de “avance” (totalmente extendidos), se dará por sentado el STEP 9.

La transición hacia el paso siguiente, se da con extensión completa del cilindro elevador, estando aun activos, los paralelos. Es en este paso que se reduce el número en memoria de pallets almacenados, esto debido a la señal recibida desde el sensor de retroceso del cilindro lateral de separación, que como vimos anteriormente, es quien se encarga de sostener los pallets dentro del almacén.

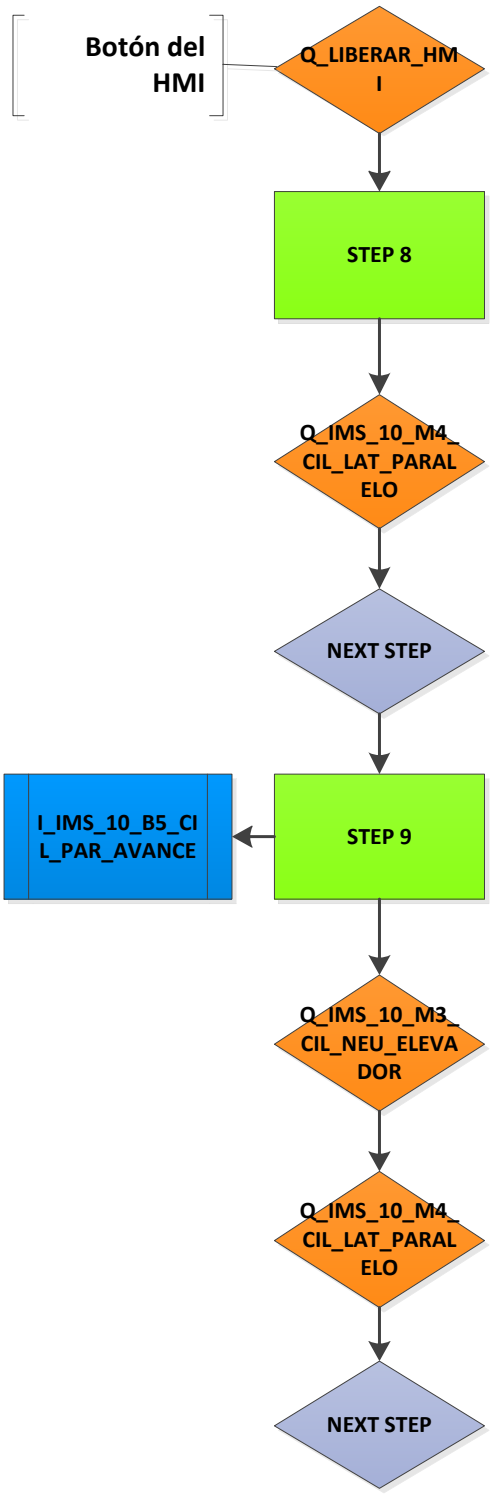


Figura 2.12 Diagrama de Flujo, Transición Liberar, 1ra parte

Habiendo culminado el descenso del pallet, recibiendo la señal desde el cilindro elevador, en posición de retroceso, I_IMS_10_B6_CIL_ELEV_RETROCESO, se alcanza el STEP 10, acto seguido inicia la salida del pallet del almacén, sobre la banda transportadora, en esta transición, activándose el motor 1_IMS_10_MOTOR_ADELANTE.

Una vez que se completa el recorrido del pallet, activándose el sensor al final de la banda I_IMS_10_B2_SENSOR_DERECHA, se alcanza el STEP 11, iniciándose nuevamente la transición LIBERAR, de manera cíclica (a partir del STEP 8), hasta que se confirme en memoria que no existan pallets, o que, en su defecto, se indique en HMI, mediante el botón MARCHA/PARO, la interrupción para luego retomar un posible nuevo ciclo de almacenamiento. (Figura 2-13)

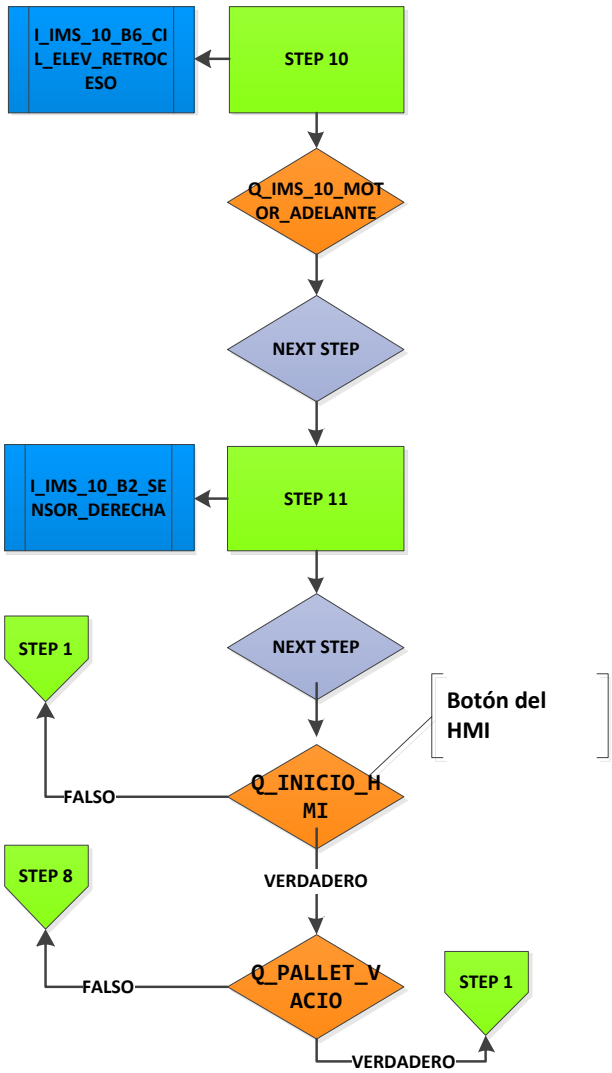


Figura 2.13 Diagrama de Flujo, Transición Liberar, 2da parte

CAPÍTULO 3

3 RESULTADOS Y ANÁLISIS

En el presente capítulo se van a presentar los resultados obtenidos durante el desarrollo del proyecto, presentando la solución práctica a la problemática planteada en el capítulo 1, basada en co-simulación, que les permitirá a los estudiantes diseñar una extensa variedad de escenarios de prueba, sin comprometer la integridad de dispositivos con los que cuenta el laboratorio, y sin la necesidad de que se encuentren físicamente en él.

Para lo cual, se va a detallar, de forma resumida, el proceso de desarrollo del sistema implementado, y se mostrarán los resultados obtenidos con su respectivo análisis.

3.1 Configuración e Ingeniería de Componentes (Software) a utilizar

Una vez, que se obtuvieron los recursos para realizar la respectiva programación de los componentes de la co-simulación, detallados en Metodología (epígrafe 2.2.1.2, Software), se procedió a realizar las configuraciones, modelamientos de sistemas y programaciones, de acuerdo a lo explicado en el epígrafe 2.3 (Modelamiento del sistema), de la siguiente manera:

3.1.1 Configuración y programación del controlador virtual PLC S7-1500 de SIEMENS

Para programar el controlador virtual PLC S7-1500 de SIEMENS, se utilizó el aplicativo TIA Portal V15.1 y su herramienta SIMATIC STEP 7 Professional V15.1.

Inicialmente, cuando se creó el proyecto nuevo, se realizaron los siguientes ajustes para poderlo simular con S7 PLCSIM ADVANCED (Figura 3-1).

1. Abrimos el menú contextual del proyecto haciendo clic derecho en él.
2. Dando un clic en "Propiedades".

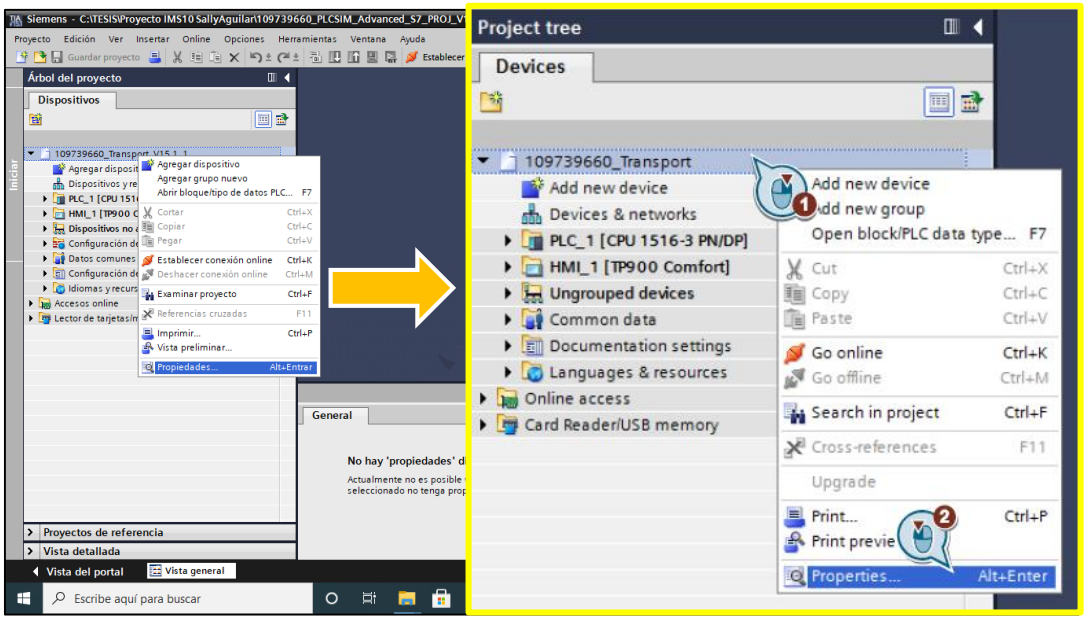


Figura 3.1 Configuración y programación del controlador virtual PLC S7-1500

1. Se seleccionó la pestaña "Protección".
2. Activando la casilla de opción "Permitir simulación al compilar bloques".
3. Se dio clic en “Aceptar”. (Figura 3-2)

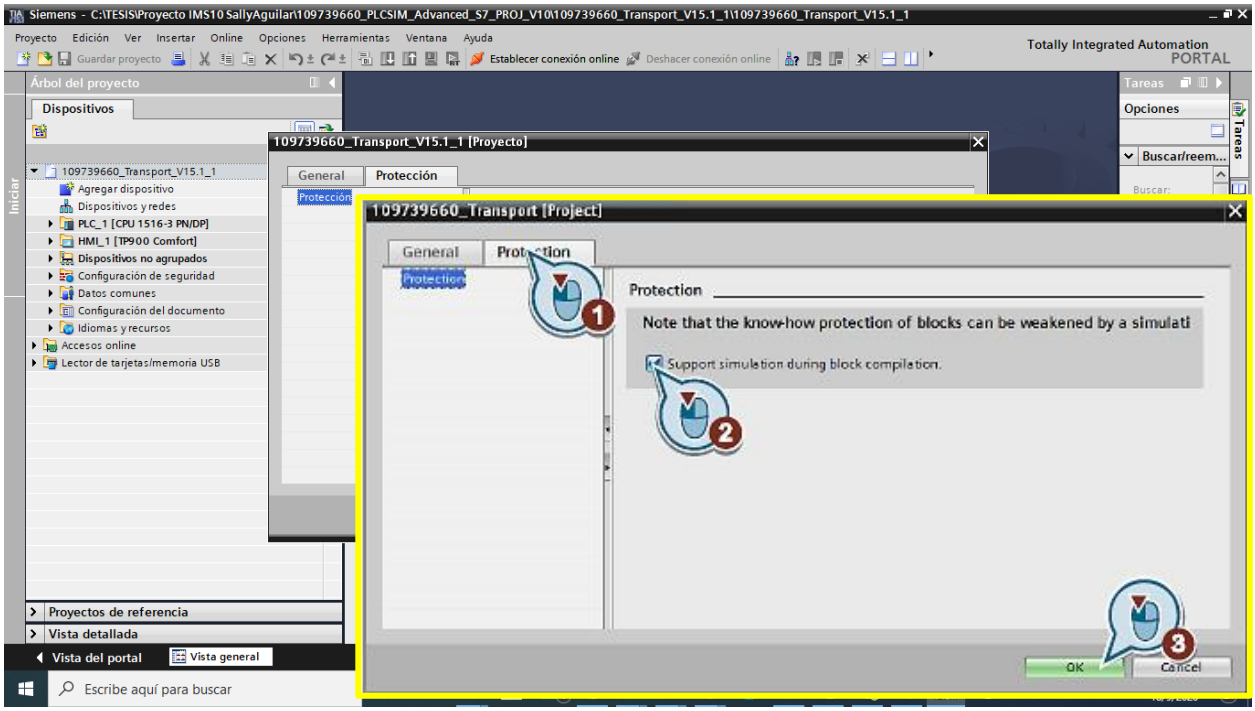


Figura 3.2 Permiso de simulación al compilar bloques

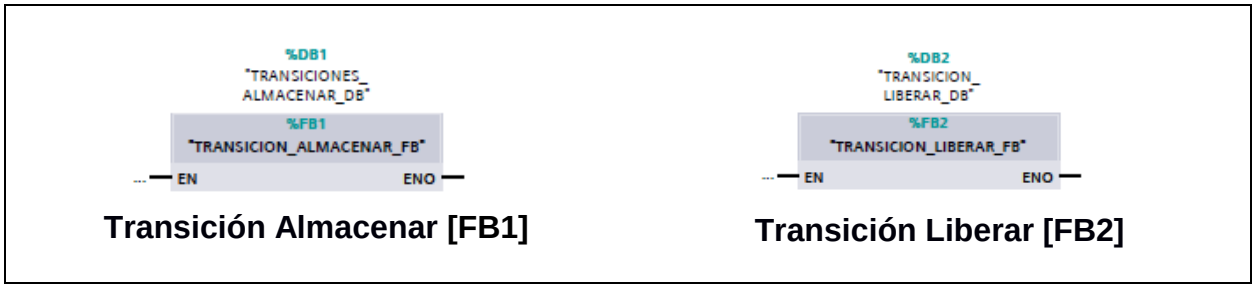
3.1.1.1 Programación de los bloques de organización, funciones y datos del controlador Virtual

Para programar los bloques de organización, funciones y datos del PLC, se siguió el esquema mostrado en la Figura 2-8 (Configuración TIA PORTAL del proyecto), descrita en el capítulo 2.3.1 (Bloques de Programa de metodología), de acuerdo a lo explicado a continuación:

3.1.1.1.1 Programación del bloque de organización Main [OB1]

Para realizar la programación de este bloque de organización Main [OB1], se usó el lenguaje de programación FUP (Diagrama de funciones), y se procedió a declarar los siguientes bloques de Función (Figura 3-3) y Funciones (Figura 3-4), para poder indicarle al programa la estructura que debe de seguir.

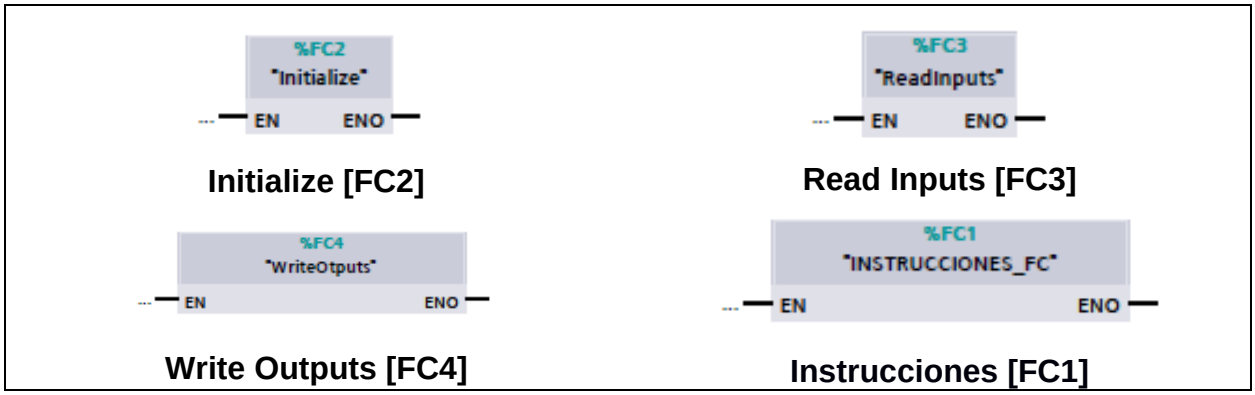
- Bloques de función (FB)



Transición Almacenar [FB1] **Transición Liberar [FB2]**

Figura 3.3 Bloques de Función

- Funciones (FC)



Initialize [FC2] **Read Inputs [FC3]**

Write Outputs [FC4] **Instrucciones [FC1]**

Figura 3.4 Funciones

- Bloques de Error. - En este bloque se ejecuta el ESTADO DE ERROR, se declararon 2 entradas: %I4.1 "simulate_error" y %I4.2 "ack_signal", variables que son controladas desde la co-simulación, y las que respectivamente activan o reinician el bloque Set/Reset, llamado %M4.0 “m_error”, que es una variable de memoria, encargada de presentar la ejecución de error “PALLET ATORADO”, en la interfaz gráfica HMI (Figura 3-5).

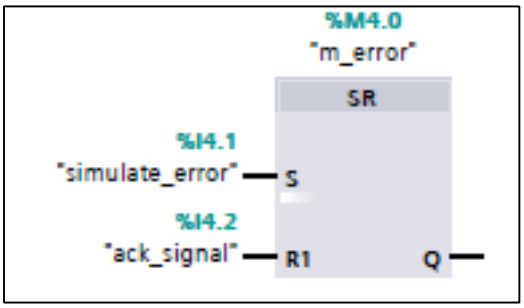
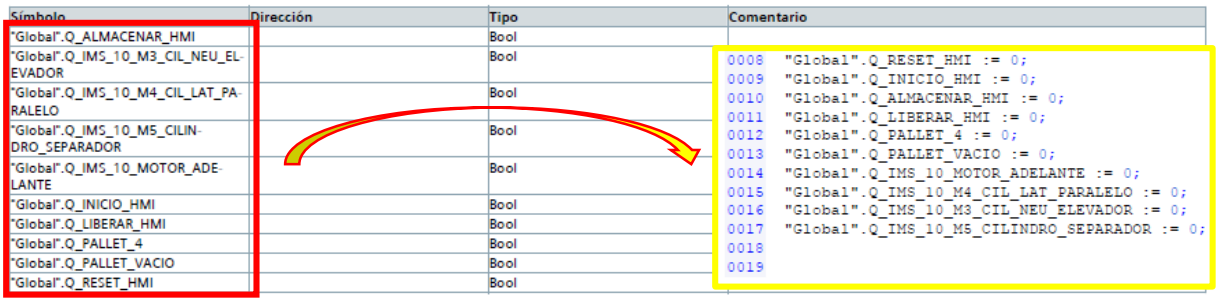


Figura 3.5 Bloque de Error

3.1.1.1.2 **Programación de Funciones (FC)**

En estos bloques se programaron funciones y operaciones lógicas que van a ser llamadas por las transiciones programadas como bloques de Función (FB), y van a tomar los datos obtenidos en estas instancias.

- **Initialize [FC2]. -** Para realizar la programación de este bloque de Función [FC], se usó el lenguaje de programación SCL (Structured Control Language), que es un lenguaje de programación de alto nivel orientado a PASCAL, siguiendo la referencia del ejemplo utilizado y citado en el capítulo de la sección Metodología, en este bloque se inicializa las variables (salidas) globales. Salidas del controlador virtual que van a ser las entradas en la Co-simulación (Figura 3-6).

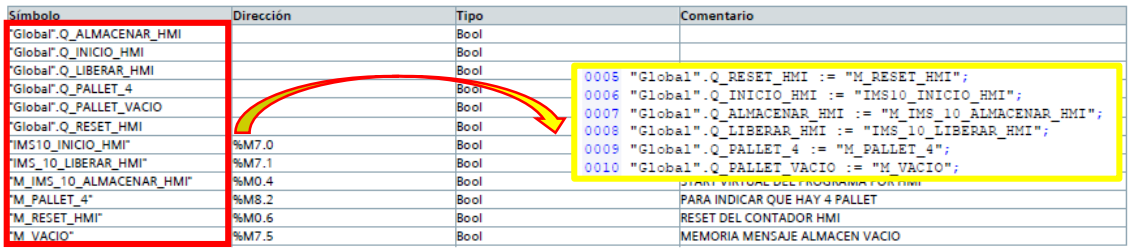


Símbolo	Dirección	Tipo	Comentario
"Global".Q_ALMACENAR_HMI		Bool	
"Global".Q_IMS_10_M3_CIL_NEU_ELEVADOR		Bool	
"Global".Q_IMS_10_M4_CIL_LAT_PARALELO		Bool	
"Global".Q_IMS_10_M5_CILINDRO_SEPARADOR		Bool	
"Global".Q_IMS_10_MOTOR_ADELANTE		Bool	
"Global".Q_INICIO_HMI		Bool	
"Global".Q_LIBERAR_HMI		Bool	
"Global".Q_PALLET_4		Bool	
"Global".Q_PALLET_VACIO		Bool	
"Global".Q_RESET_HMI		Bool	

```
0008  "Global".Q_RESET_HMI := 0;
0009  "Global".Q_INICIO_HMI := 0;
0010  "Global".Q_ALMACENAR_HMI := 0;
0011  "Global".Q_LIBERAR_HMI := 0;
0012  "Global".Q_PALLET_4 := 0;
0013  "Global".Q_PALLET_VACIO := 0;
0014  "Global".Q_IMS_10_MOTOR_ADELANTE := 0;
0015  "Global".Q_IMS_10_M4_CIL_LAT_PARALELO := 0;
0016  "Global".Q_IMS_10_M3_CIL_NEU_ELEVADOR := 0;
0017  "Global".Q_IMS_10_M5_CILINDRO_SEPARADOR := 0;
0018
0019
```

Figura 3.6 Variables / Salidas Globales en función INITIALIZE [FC2]

- **Read Inputs [FC3]. -** De igual manera que en el bloque anterior esta programación se la realizó con el lenguaje de programación SCL, aquí se lee las entradas desde el HMI, transformando las señales de memoria en salidas “globales” (Figura 3-7).

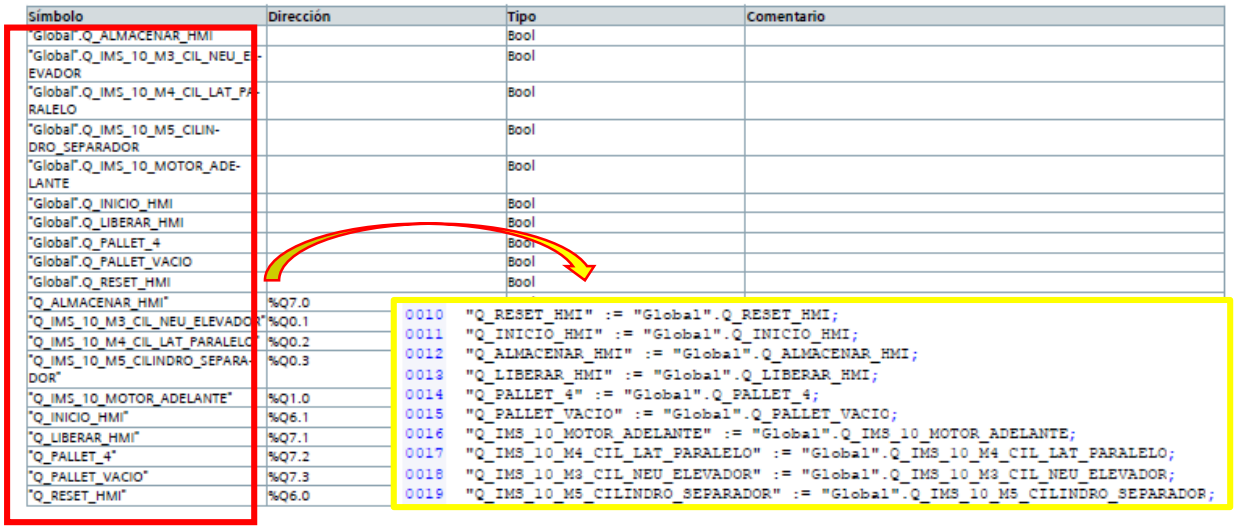


Símbolo	Dirección	Tipo	Comentario
"Global".Q_ALMACENAR_HMI		Bool	
"Global".Q_INICIO_HMI		Bool	
"Global".Q_LIBERAR_HMI		Bool	
"Global".Q_PALLET_4		Bool	
"Global".Q_PALLET_VACIO		Bool	
"Global".Q_RESET_HMI		Bool	
"IMS10_INICIO_HMI"	%M7.0	Bool	
"IMS_10_LIBERAR_HMI"	%M7.1	Bool	
"M_IMS_10_ALMACENAR_HMI"	%M0.4	Bool	
"M_PALLET_4"	%M8.2	Bool	PARA INDICAR QUE HAY 4 PALLET
"M_RESET_HMI"	%M0.6	Bool	RESET DEL CONTADOR HMI
"M_VACIO"	%M7.5	Bool	MEMORIA MENSAJE ALMACEN VACIO

```
0005 "Global".Q_RESET_HMI := "M_RESET_HMI";
0006 "Global".Q_INICIO_HMI := "IMS10_INICIO_HMI";
0007 "Global".Q_ALMACENAR_HMI := "M_IMS_10_ALMACENAR_HMI";
0008 "Global".Q_LIBERAR_HMI := "IMS_10_LIBERAR_HMI";
0009 "Global".Q_PALLET_4 := "M_PALLET_4";
0010 "Global".Q_PALLET_VACIO := "M_VACIO";
```

Figura 3.7 Variables / Salidas Globales en función READ INPUTS [FC3]

- **Write Outputs [FC4]. -** También programada con el lenguaje de programación SCL, aquí se transforman las salidas del PLC (%Q), en variables (salidas) globales que se comunican con Co-simulación (Visual Studio) (Figura 3-8).



Símbolo	Dirección	Tipo	Comentario
"Global".Q_ALMACENAR_HMI		Bool	
"Global".Q_IMS_10_M3_CIL_NEU_ELEVADOR		Bool	
"Global".Q_IMS_10_M4_CIL_LAT_PARALELO		Bool	
"Global".Q_IMS_10_M5_CILINDRO_SEPARADOR		Bool	
"Global".Q_IMS_10_MOTOR_ADELANTE		Bool	
"Global".Q_INICIO_HMI		Bool	
"Global".Q_LIBERAR_HMI		Bool	
"Global".Q_PALLET_4		Bool	
"Global".Q_PALLET_VACIO		Bool	
"Global".Q_RESET_HMI		Bool	
"Q_ALMACENAR_HMI"	%Q7.0		
"Q_IMS_10_M3_CIL_NEU_ELEVADOR"	%Q0.1		
"Q_IMS_10_M4_CIL_LAT_PARALELO"	%Q0.2		
"Q_IMS_10_M5_CILINDRO_SEPARADOR"	%Q0.3		
"Q_IMS_10_MOTOR_ADELANTE"	%Q1.0		
"Q_INICIO_HMI"	%Q6.1		
"Q_LIBERAR_HMI"	%Q7.1		
"Q_PALLET_4"	%Q7.2		
"Q_PALLET_VACIO"	%Q7.3		
"Q_RESET_HMI"	%Q6.0		

```
0010 "Q_RESET_HMI" := "Global".Q_RESET_HMI;
0011 "Q_INICIO_HMI" := "Global".Q_INICIO_HMI;
0012 "Q_ALMACENAR_HMI" := "Global".Q_ALMACENAR_HMI;
0013 "Q_LIBERAR_HMI" := "Global".Q_LIBERAR_HMI;
0014 "Q_PALLET_4" := "Global".Q_PALLET_4;
0015 "Q_PALLET_VACIO" := "Global".Q_PALLET_VACIO;
0016 "Q_IMS_10_MOTOR_ADELANTE" := "Global".Q_IMS_10_MOTOR_ADELANTE;
0017 "Q_IMS_10_M4_CIL_LAT_PARALELO" := "Global".Q_IMS_10_M4_CIL_LAT_PARALELO;
0018 "Q_IMS_10_M3_CIL_NEU_ELEVADOR" := "Global".Q_IMS_10_M3_CIL_NEU_ELEVADOR;
0019 "Q_IMS_10_M5_CILINDRO_SEPARADOR" := "Global".Q_IMS_10_M5_CILINDRO_SEPARADOR;
```

Figura 3.8 Variables / Salidas Globales en función WRITE INPUTS [FC3]

3.1.1.1.3 **Instrucciones [FC1]**

Este bloque de funciones [FC], fue programado con el lenguaje de programación FUP (Diagrama de funciones), aquí se realizan distintas operaciones entre: entradas (%I), salidas (%Q) y memorias (%M), con la finalidad de activar las variables de salidas y guardar datos en las variables de memoria, que son utilizados para activar la visualización en la interfaz gráfica.

Es posible programar funciones a las que se les pueda asignar parámetros. Como resultado, las funciones también se pueden utilizar para tareas repetitivas o funcionalidades complejas tales como cálculos.

Aquí se encuentra parte de la funcionalidad del programa; la secuencia de programación que se utilizó para programar las funciones dentro de esta instancia, fueron las instrucciones lógicas necesarias para activar salidas y guardar datos de memoria.

Esta programación se basa al estado en el que se encuentre la secuencia del proceso, que es el resultado de las transiciones Almacenar y Liberar descritas en los bloques de funciones (FB).

En la tabla 3-1, se detallan las funciones que se realizan en Instrucciones [FC1].

Se ahonda en detalles de los Segmentos que lo conforman en el ANEXO 1.

3.1.1.1.1 **Programación de Bloques de función (FB)**

Los bloques de funciones FB están concebidos para realizar tareas muy repetitivas o funcionalidades complejas, como tareas de control de lazo cerrado. El lenguaje de programación con el que se programaron estas funciones fue FUP (Diagrama de funciones), basándose en la tabla de transiciones, que indica las condiciones que deben de cumplirse para llegar a un estado determinado y que salida se activa en cada estado determinado.

Transición Almacenar [FB1]. - Reúne las condiciones que permitieron llegar a los 8 estados de esta transición, tal como se vio en el Diagrama de Flujo del Capítulo 2.

En la Figura 3-9, podemos comprobar, como se visualiza en el STEP7 del TIA PORTAL, este Bloque de Funciones, denominado TRANSICION_ALMACENAR_FB.

La información detallada de la imagen indicada, se expande en la tabla 3-2.

Se ahonda en detalles de los Segmentos que lo conforman en el ANEXO 2.

Nombre	Tipo de datos	Offset	Valor predet.	Accesible desde HMI/OPC UA	Es-cribi-ble desd e HMI/ OPC UA	Visible en HMI Engi-neering	Valor de ajuste	Supervi-sión	Comentario
Input									
Output									
InOut									
▼ Static									
ESTADO 1	Bool	0.0	false	True	True	True	False		
ESTADO 2	Bool	0.1	false	True	True	True	False		
ESTADO 3	Bool	0.2	false	True	True	True	False		
ESTADO 4	Bool	0.3	false	True	True	True	False		
ESTADO 5	Bool	0.4	false	True	True	True	False		
ESTADO 6	Bool	0.5	false	True	True	True	False		
ESTADO 7	Bool	0.6	false	True	True	True	False		
ESTADO 8	Bool	0.7	false	True	True	True	False		
Temp									
Constant									

Figura 3.9 Bloque de Funciones "TRANSICION_ALMACENAR[FB]", en STEP7

Transición Liberar [FB2]. - Reúne las condiciones que permitieron llegar a los 8 estados de esta transición, tal como se vio en el Diagrama de Flujo del Capítulo 2.

En la Figura 3-10, podemos comprobar, como se visualiza en el STEP7 del TIA PORTAL, este Bloque de Funciones, denominado TRANSICION_LIBERAR_FB.

La información detallada de la imagen indicada, se expande en la tabla 3-3.

Se ahonda en detalles de los Segmentos que lo conforman en el ANEXO 3.

Nombre	Tipo de datos	Offset	Valor predet.	Accesible desde HMI/OPC UA	Es-cribi-ble desd e HMI/ OPC UA	Visible en HMI Engi-neering	Valor de ajuste	Supervi-sión	Comentario
Input									
Output									
InOut									
▼ Static									
ESTADO 1.1	Bool	0.0	false	True	True	True	False		
ESTADO 2.1	Bool	0.1	false	True	True	True	False		
ESTADO 3.1	Bool	0.2	false	True	True	True	False		
ESTADO 4.1	Bool	0.3	false	True	True	True	False		
ESTADO 5.1	Bool	0.4	false	True	True	True	False		
ESTADO 6.1	Bool	0.5	false	True	True	True	False		
ESTADO 7.1	Bool	0.6	false	True	True	True	False		
ESTADO 8.1	Bool	0.7	false	True	True	True	False		
Temp									
Constant									

Figura 3.10 Bloque de Funciones "TRANSICION_LIBERAR[FB]", en STEP7

Tabla 3-1 Instrucciones [FC]

INSTRUCCIONES

SEGMENTO	CONDICION	DIRECCION	CONDICIONES	ESTADO	SEÑAL ACTIVADA		TIPO DE BLOQUE	TIPO DE DATO	ACCION	VARIABLE EN LA QUE GUARDA	
	NOMBRE				NOMBRE	DIRECCION				NOMBRE	DIRECCION
1	M_RESET_CONTADOR	%M0.2		5 4.1	CONTROL_PALLET	%DB4	CONTADOR	ENTERO	AUMENTA DISMINUYE RESETEA	MW_CANTIDAD_PALLET	%MW0
2				2 3 6.1 7.1	"Global".Q_IMS_10_MOTOR_ADELANTE Q_ESCLAVO8_MOTOR_ADELANTE Q_ESCLAVO9_MOTOR_ADELANTE	%Q3.0 %Q5.0					
3				3	QS_IMS_10_MOTOR_LENTO	%Q1.2					
4				5 6 3.1 4.1	"Global".Q_IMS_10_M3_CIL_NEU_ELEVADOR						
5				3	Q_IMS_10_M2_CIL_MED_PARADA	%Q0.0					
6				4 5 2.1 3.1	"Global".Q_IMS_10_M4_CIL_LAT_PARALELO						
7				2 3	QS_ESCLAVO8_MOTOR_LENTO	%Q3.2					
8	"Global".Q_RESET_HMI "I_RESET_FISICO"	%I200.2			M_RESET_CONTADOR	%M0.2					
9				5 4.1	Q_IMS_10_M5_CILINDRO_SEPARADOR	%Q0.3					
10	"CONTROL_PALLET".CV				MW_NUMERO_PALLETS_HMI	%MW2	TRANSFORMA	PALABRA			
11	"CONTROL_PALLET".CV				M_VACIO	%M7.5					
12	"CONTROL_PALLET".CV				M_PALLET_1	%M7.7					
13	"CONTROL_PALLET".CV				M_PALLET_2	%M8.0					
14	"CONTROL_PALLET".CV				M_PALLET_3	%M8.1					
15	"CONTROL_PALLET".CV				M_PALLET_4	%M8.2					
16	I_IMS_10_B1_SENSOR_IZQUIERDA	%I1.3			M_SENSOR_IZQ_HMI	%M8.4					
17	I_IMS_10_B2_SENSOR_DERECHA	%I1.4			M_SENSOR_DER_HMI	%M8.6					
18	Q_IMS_10_M5_CILINDRO_SEPARADOR	%Q0.3			M_SEP_CILIN	%M9.1					
19	IMS10_INICIO_HMI I_IMS_10_INICIO_FISICO	%M7.0 %I200.0			M_MOSTRAR_INICIAL	%M9.3					
20	I_FIN_CARRERA	%I0.6			M_FINCARRERA M_ALMACEN_LLENO	%M8.3 %M7.2					
23	M_MOSTRAR_INICIAL CONTROL_PALLET.CV	%M9.3 == 0	&		M_INICIADO_PROGRAMA	%M10.0					
25	I_IMS_10_B1_SENSOR_IZQUIERDA	%I1.3			M_CILINDRO DE PARADA	%M10.3	TEMPORIZADOR	4 SEG.	GUARDA VALOR DEL SENSOR		

Tabla 3-2 Transición Almacenar

TRANSICIÓN ALMACENAR

FUNCION	SEGMENTO	TRANSICIÓN		CONDICION		CONDICIONES		ESTADO	SEÑAL ACTIVADA		TIPO DE BLOQUE	TIPO DE DATO	ACCION	CONDICION	VARIABLE EN LA QUE GUARDA	
		INICIAL	FINAL	NOMBRE	DIRECCION				NOMBRE	DIRECCION					NOMBRE	DIRECCION
ALMACENAR	1	8	1	VENIR DEL ESTADO 8 IMS10_INICIO_HMI I_IMS_10_INICIO_FISICO m_error	%M7.0 %I200.0 %M4.0	OR							RESETEA	ESTADO 2		
								1	ESTADO DE ESPERA							
ALMACENAR	2	1	2	M_IMS_10_ALMACENAR_HMI I_IMS_10_ALMACENAR_FISICO VENIR DE ESTADO 1 O 1.1 CONTROL_PALLET.CV	%M0.4 %I200.1	OR < 4	&						RESETEA	OR	ESTADO 3 IMS10_INICIO_HMI I_IMS_10_INICIO_FISICO m_error	
INSTRUCCIONES INSTRUCCIONES INSTRUCCIONES INSTRUCCIONES	2 2 2 7							2	"Global".Q_IMS_10_MOTOR_ADELANTE Q_ESCLAVO8_MOTOR_ADELANTE Q_ESCLAVO9_MOTOR_ADELANTE QS_ESCLAVO8_MOTOR_LENTO	 %Q3.0 %Q5.0 %Q3.2						
ALMACENAR	3	2	3	VENIR DEL ESTADO 2 I_IMS_10_B1_SENSOR_IZQUIERDA	%I1.3	&							RESETEA	OR	ESTADO 4 IMS10_INICIO_HMI I_IMS_10_INICIO_FISICO m_error	
INSTRUCCIONES INSTRUCCIONES INSTRUCCIONES INSTRUCCIONES INSTRUCCIONES INSTRUCCIONES	2 2 2 3 5 7							3	"Global".Q_IMS_10_MOTOR_ADELANTE Q_ESCLAVO8_MOTOR_ADELANTE Q_ESCLAVO9_MOTOR_ADELANTE QS_IMS_10_MOTOR_LENTO Q_IMS_10_M2_CIL_MED_PARADA QS_ESCLAVO8_MOTOR_LENTO	 %Q3.0 %Q5.0 %Q1.2 %Q0.0 %Q3.2						
ALMACENAR	4	3	4	VENIR DEL ESTADO 3 T_DELAY__ESTADO4	%T2	6 SEG.	&						RESETEA	OR	ESTADO 5 IMS10_INICIO_HMI I_IMS_10_INICIO_FISICO m_error	
INSTRUCCIONES	6							4	"Global".Q_IMS_10_M4_CIL_LAT_PARALELO							
ALMACENAR	5	4	5	VENIR DEL ESTADO 4 I_IMS_10_B5_CIL_PAR_AVANCE	%I0.2	&							RESETEA	OR	ESTADO 6 IMS10_INICIO_HMI I_IMS_10_INICIO_FISICO m_error	
INSTRUCCIONES INSTRUCCIONES INSTRUCCIONES INSTRUCCIONES	1 4 6 9							5	CONTROL_PALLET "Global".Q_IMS_10_M3_CIL_NEU_ELEVADOR "Global".Q_IMS_10_M4_CIL_LAT_PARALELO Q_IMS_10_M5_CILINDRO_SEPARADOR	%DB4 %Q0.3	CONTADOR	ENTERO	AUMENTA		MW_CANTIDAD_PALLET	%MW0
ALMACENAR	6	5	6	VENIR DEL ESTADO 5 T_DELAY__ESTADO6	%T3	6 SEG.	&						RESETEA	OR	ESTADO 7 IMS10_INICIO_HMI I_IMS_10_INICIO_FISICO m_error	
INSTRUCCIONES	4							6	"Global".Q_IMS_10_M3_CIL_NEU_ELEVADOR							
ALMACENAR	7	6	7	VENIR DEL ESTADO 6 I_IMS_10_B4_CIL_PAR_RETROCESO	%I0.1	&							RESETEA	OR	ESTADO 8 IMS10_INICIO_HMI I_IMS_10_INICIO_FISICO m_error	
								7	NO HAY SALIDAS ACTIVAS							
ALMACENAR	8	7	8	VENIR DEL ESTADO 7 I_IMS_10_B6_CIL_ELEV_RETROCESO	%I0.3	&							RESETEA	OR	ESTADO 1 ESTADO 2 IMS10_INICIO_HMI I_IMS_10_INICIO_FISICO m_error	
								8	NO HAY SALIDAS ACTIVAS							

Tabla 3-3 Transición Liberar

TRANSICIÓN LIBERAR

FUNCION	SEGMENTO	TRANSICIÓN		CONDICION		CONDICIONES		ESTADO	SEÑAL ACTIVADA		TIPO DE BLOQUE	TIPO DE DATO	ACCION	CONDICION	VARIABLE EN LA QUE GUARDA	
		INICIAL	FINAL	NOMBRE	DIRECCION				NOMBRE	DIRECCION					NOMBRE	DIRECCION
LIBERAR	1	8.1	1.1	VENIR DEL ESTADO 8 .1 "CONTROL_PALLET".CV IMS10_INICIO_HMI I_IMS_10_INICIO_FISICO m_error	==0 %M7.0 %I200.0 %M4.0	&	OR						RESETEA	ESTADO 2.1		
								1.1	ESTADO DE ESPERA							
LIBERAR	2	1.1	2.1	IMS_10_LIBERAR_HMI I_IMS_10_LIBERAR_FISICO VENIR DE ESTADO 1 O 1.1 CONTROL_PALLET.CV	%M7.1 %I200.3	OR	&						RESETEA	OR	ESTADO 3.1 IMS10_INICIO_HMI I_IMS_10_INICIO_FISICO m_error	
INSTRUCCIONES	6							2.1	"Global".Q_IMS_10_M4_CIL_LAT_PARALELO							
LIBERAR	3	2.1	3.1	VENIR DEL ESTADO 2.1 I_IMS_10_B5_CIL_PAR_AVANCE	%I0.2	&							RESETEA	OR	ESTADO 4.1 IMS10_INICIO_HMI I_IMS_10_INICIO_FISICO m_error	
INSTRUCCIONES INSTRUCCIONES	4 6							3.1	"Global".Q_IMS_10_M3_CIL_NEU_ELEVADOR "Global".Q_IMS_10_M4_CIL_LAT_PARALELO							
LIBERAR	4	3.1	4.1	VENIR DEL ESTADO 3.1 T_DELAY_ESTADO4_LIBERAR	%T5		&						RESETEA	OR	ESTADO 5.1 IMS10_INICIO_HMI I_IMS_10_INICIO_FISICO m_error	
INSTRUCCIONES INSTRUCCIONES INSTRUCCIONES	1 4 9							4.1	CONTROL_PALLET "Global".Q_IMS_10_M3_CIL_NEU_ELEVADOR Q_IMS_10_M5_CILINDRO_SEPARADOR	%DB4 %Q0.3	CONTADOR	ENTERO	DISMINUYE		MW_CANTIDAD_PALLET	%MW0
LIBERAR	5	4.1	5.1	VENIR DEL ESTADO 4.1 T_DESACTIVAR_B8	%T15		&						RESETEA	OR	ESTADO 6.1 IMS10_INICIO_HMI I_IMS_10_INICIO_FISICO m_error	
								5.1	NO HAY SALIDAS ACTIVAS							
LIBERAR	6	5.1	6.1	VENIR DEL ESTADO 5.1 I_IMS_10_B6_CIL_ELEV_RETROCESO	%I0.3	&							RESETEA	OR	ESTADO 7.1 IMS10_INICIO_HMI I_IMS_10_INICIO_FISICO m_error	
INSTRUCCIONES INSTRUCCIONES INSTRUCCIONES	2 2 2							6.1	"Global".Q_IMS_10_MOTOR_ADELANTE Q_ESCLAVO8_MOTOR_ADELANTE Q_ESCLAVO9_MOTOR_ADELANTE	 %Q3.0 %Q5.0						
LIBERAR	7	6.1	7.1	VENIR DEL ESTADO 6.1 I_IMS_10_B2_SENSOR_DERECHA	%I1.4	&							RESETEA	OR	ESTADO 8.1 IMS10_INICIO_HMI I_IMS_10_INICIO_FISICO m_error	
INSTRUCCIONES INSTRUCCIONES INSTRUCCIONES	2 2 2							7.1	"Global".Q_IMS_10_MOTOR_ADELANTE Q_ESCLAVO8_MOTOR_ADELANTE Q_ESCLAVO9_MOTOR_ADELANTE	 %Q3.0 %Q5.0						
LIBERAR	8	7.1	8.1	VENIR DEL ESTADO 7.1 T_DELAY_ESTADO8_LIBERAR	%T6		&						RESETEA	OR	ESTADO 1.1 ESTADO 2.1 IMS10_INICIO_HMI I_IMS_10_INICIO_FISICO m_error	
								8.1	NO HAY SALIDAS ACTIVAS							

3.1.1.1.2 **Programación de Bloques de Datos Globales (DB)**

Dado que los Bloques de Datos Globales almacenan datos de usuario utilizables desde todos los demás bloques, se declararon en este grupo de bloques, las variables “globales” con las que se va a trabajar en la co-simulación, los estados de cada una de las transiciones de almacenar y liberar, y el contador donde se registra la cantidad de pallets que se encuentran dentro del almacén.

Para realizar la programación de este bloque de datos globales [DB], se usó el lenguaje de programación DB, típico de este tipo de bloques, que realiza una función equivalente a una base de datos organizada de acuerdo a las variables utilizadas en las funciones.

Global [DB3]. - En este bloque (Figura 3-11), se declararon todas las salidas del programa que se van a utilizar en la co-simulación:

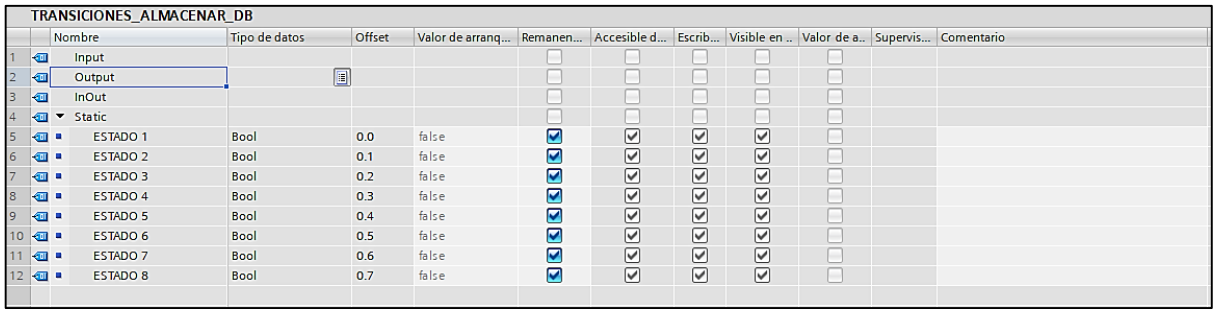
Global [DB3]

Global Propiedades							
General							
Nombre	Global	Número	3	Tipo	DB	Idioma	DB
Numeración	Automático						
Información							
Título		Autor		Comentario		Familia	
Versión	0.1	ID personalizado					

Nombre	Tipo de datos	Valor de arranque	Remanencia	Accesible desde HMI/OPC UA	Escribible desde HMI/OPC UA	Visible en HMI Engineering	Valor de ajuste	Supervisión	Comentario
Static									
Q_RESET_HMI	Bool	false	False	True	True	True	False		
Q_INICIO_HMI	Bool	false	False	True	True	True	False		
Q_ALMACENAR_HMI	Bool	false	False	True	True	True	False		
Q_LIBERAR_HMI	Bool	false	False	True	True	True	False		
Q_PALLET_4	Bool	false	False	True	True	True	False		
Q_IMS_10_MOTOR_ADELANTE	Bool	false	False	True	True	True	False		
Q_IMS_10_M4_CIL_LAT_PARALELO	Bool	false	False	True	True	True	False		
Q_IMS_10_M3_CIL_NEU_ELEVADOR	Bool	false	False	True	True	True	False		
Q_IMS_10_M5_CILINDRO_SEPARADOR	Bool	false	False	True	True	True	False		
Q_PALLET_VACIO	Bool	false	False	True	True	True	False		

Figura 3.11 Bloques de datos Global [DB3]

Transición Almacenar DB [DB1]. - En este bloque (Figura 3-12), están declarados los estados de la secuencia Almacenar a los que se llegan en cada segmento de la función, TRANSICION_ALMACENAR_FB [FB1].

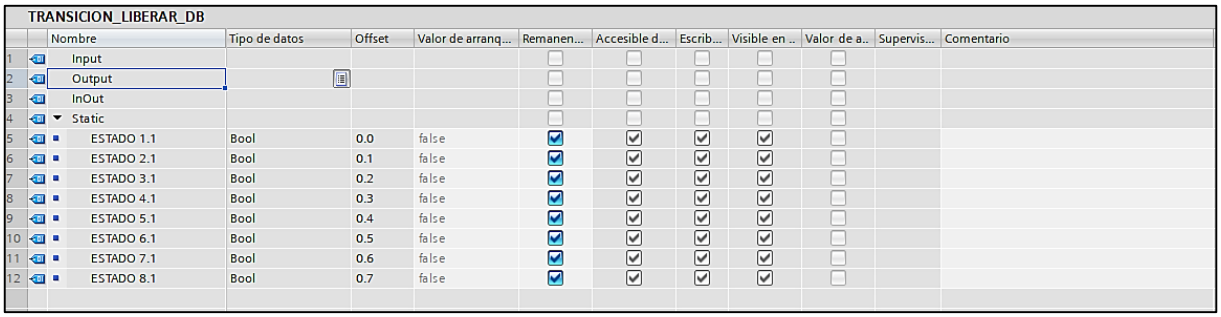


TRANSICIONES_ALMACENAR_DB

	Nombre	Tipo de datos	Offset	Valor de arranq...	Remanen...	Accesible d...	Escrib...	Visible en ...	Valor de a...	Supervis...	Comentario
1	Input				☐	☐	☐	☐	☐		
2	Output				☐	☐	☐	☐	☐		
3	InOut				☐	☐	☐	☐	☐		
4	Static				☐	☐	☐	☐	☐		
5	ESTADO 1	Bool	0.0	false	☑	☑	☑	☑	☐		
6	ESTADO 2	Bool	0.1	false	☑	☑	☑	☑	☐		
7	ESTADO 3	Bool	0.2	false	☑	☑	☑	☑	☐		
8	ESTADO 4	Bool	0.3	false	☑	☑	☑	☑	☐		
9	ESTADO 5	Bool	0.4	false	☑	☑	☑	☑	☐		
10	ESTADO 6	Bool	0.5	false	☑	☑	☑	☑	☐		
11	ESTADO 7	Bool	0.6	false	☑	☑	☑	☑	☐		
12	ESTADO 8	Bool	0.7	false	☑	☑	☑	☑	☐		

Figura 3.12 Bloques de datos, TRANSICIONES_ALMACENAR_DB

Transición Liberar DB [DB2]. - De igual manera que en la sección anterior, en este bloque (Figura 3-13), se detallaron los estados de la secuencia Liberar a los que se llegan en cada segmento de la función TRANSICION_LIBERAR_FB [FB2].

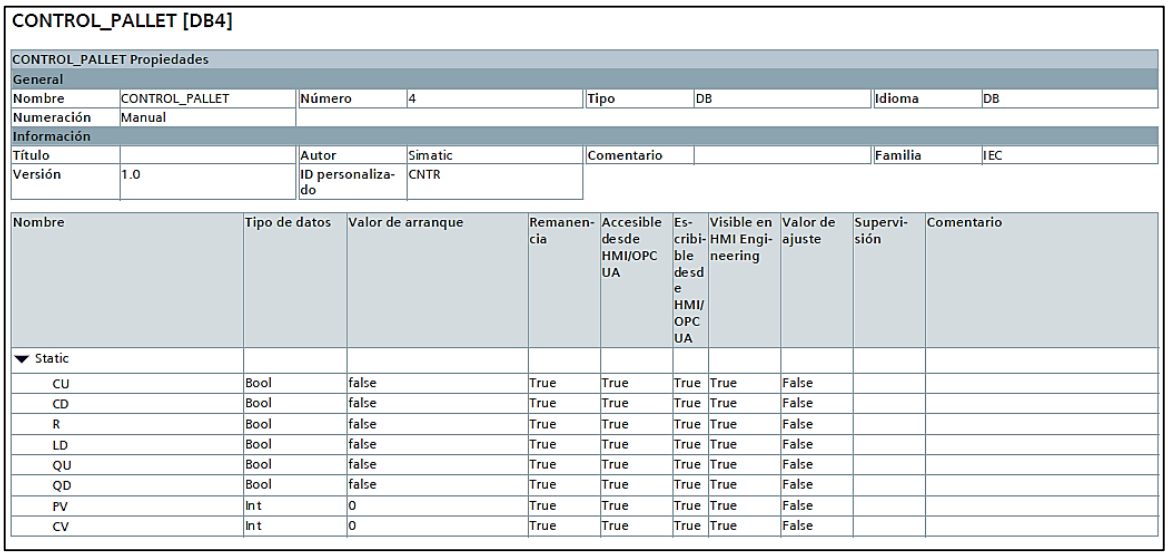


TRANSICION_LIBERAR_DB

	Nombre	Tipo de datos	Offset	Valor de arranq...	Remanen...	Accesible d...	Escrib...	Visible en ...	Valor de a...	Supervis...	Comentario
1	Input				☐	☐	☐	☐	☐		
2	Output				☐	☐	☐	☐	☐		
3	InOut				☐	☐	☐	☐	☐		
4	Static				☐	☐	☐	☐	☐		
5	ESTADO 1.1	Bool	0.0	false	☑	☑	☑	☑	☐		
6	ESTADO 2.1	Bool	0.1	false	☑	☑	☑	☑	☐		
7	ESTADO 3.1	Bool	0.2	false	☑	☑	☑	☑	☐		
8	ESTADO 4.1	Bool	0.3	false	☑	☑	☑	☑	☐		
9	ESTADO 5.1	Bool	0.4	false	☑	☑	☑	☑	☐		
10	ESTADO 6.1	Bool	0.5	false	☑	☑	☑	☑	☐		
11	ESTADO 7.1	Bool	0.6	false	☑	☑	☑	☑	☐		
12	ESTADO 8.1	Bool	0.7	false	☑	☑	☑	☑	☐		

Figura 3.13 Bloques de datos, TRANSICION_LIBERAR_FB

Control Pallet [DB4]. - El bloque de datos globales Control Pallet [DB4] (Figura 3-14), se lo encuentra en la sección “Recursos de programa”, dentro de “Bloques de sistema”, ya que esta instancia se la utiliza como un auxiliar, para poder realizar el conteo de pallets dentro del bloque de funciones Instrucciones [FC1].



CONTROL_PALLET [DB4]

CONTROL_PALLET Propiedades

General							
Nombre	CONTROL_PALLET	Número	4	Tipo	DB	Idioma	DB
Numeración	Manual						
Información							
Título		Autor	Simatic	Comentario		Familia	IEC
Versión	1.0	ID personaliza-do	CNTR				

Nombre	Tipo de datos	Valor de arranque	Remanen-cia	Accesible desde HMI/OPC UA	Es-cribi-ble desd e HMI/ OPC UA	Visible en HMI Engi-neering	Valor de ajuste	Supervi-sión	Comentario
Static									
CU	Bool	false	True	True	True	True	False		
CD	Bool	false	True	True	True	True	False		
R	Bool	false	True	True	True	True	False		
LD	Bool	false	True	True	True	True	False		
QU	Bool	false	True	True	True	True	False		
QD	Bool	false	True	True	True	True	False		
PV	Int	0	True	True	True	True	False		
CV	Int	0	True	True	True	True	False		

Figura 3.14 Bloques de datos, CONTROL_PALLET

Como se ha indicado durante la redacción de este documento, el programa de co-simulación fue programado en Visual Studio.

Considerando que el control, se basa en la activación/desactivación de sensores, es decir valores de “unos” y “ceros”, se planteó la secuencia del programa en una tabla booleana, donde quedan indicados: Los pasos, y las condiciones que se ejecutan.

3.1.2 Configuración de la interfaz gráfica HMI

El sistema de paletizado, expuesto en este proyecto se procedió a modelar gráficamente mediante la interfaz HMI TP900 COMFORT, también virtual, diseñada igualmente en el software STEP 7 y ejecutada a través del software WINCC RT START (Figura 3-15), para simular su comportamiento, de acuerdo al ambiente creado en Visual Studio.

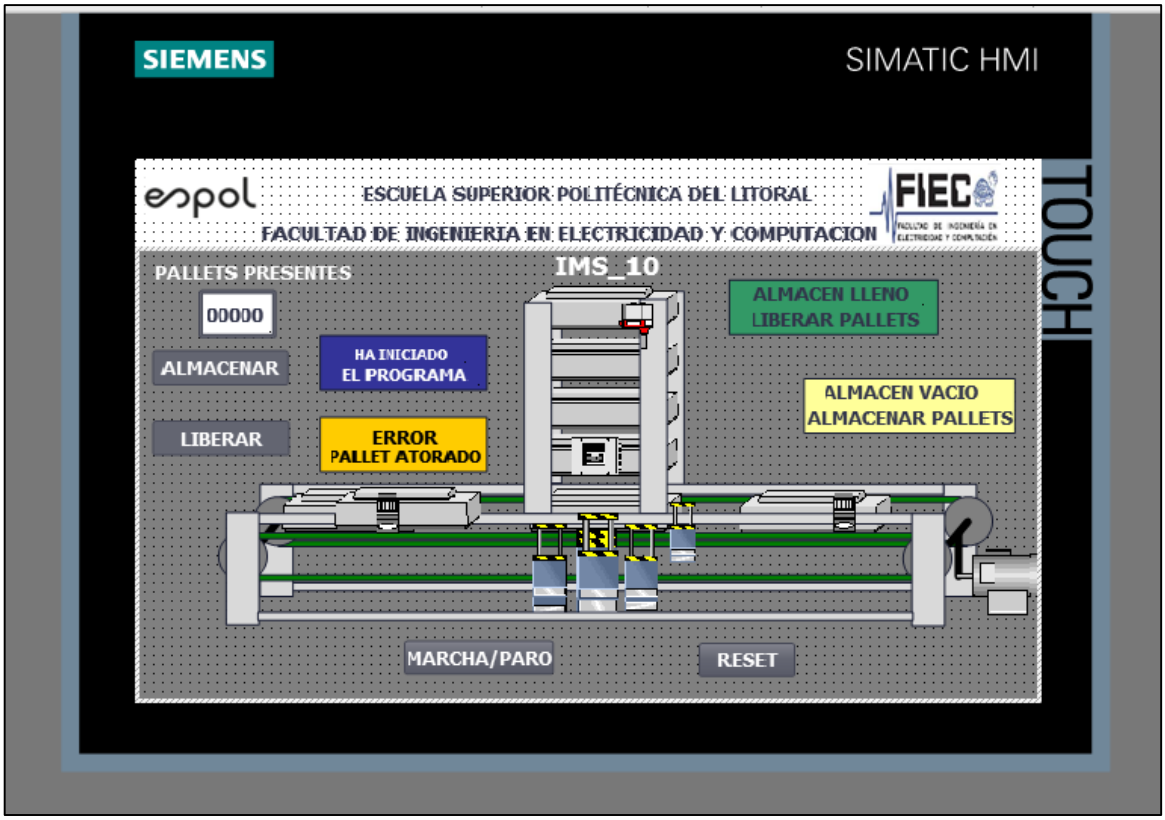


Figura 3.15 Sistema de paletizado, simulado, y presentado en WinCC RT Start

En esta imagen son visibles los componentes de la planta, de acuerdo a la tabla de etiquetas de variables que en ella se en las figuras 3-16 y 3-17, a continuación:

Variables HMI

Nombre	Tipo de datos	Conexión	Nombre del PLC	Variable PLC	Ciclo de adquisi..	Comentario de origen
I_FIN_CARRERA	Bool	HMI_Connectio...	PLC_1	I_FIN_CARRERA	1 s	INDICADOR DE LLENADO COMPLETO
I_IMS_10_B1_SENSOR_IZQUIERDA	Bool	HMI_Connectio...	PLC_1	I_IMS_10_B1_SENSOR_IZQUIERDA	1 s	SENSOR EN LA POSICION IZQUIERDA
I_IMS_10_B1_SENSOR_IZQUIERDA(1)	Bool	HMI_Connectio...	PLC_1	I_IMS_10_B1_SENSOR_IZQUIERDA	1 s	SENSOR EN LA POSICION IZQUIERDA
I_IMS_10_B2_SENSOR_DERECHA	Bool	HMI_Connectio...	PLC_1	I_IMS_10_B2_SENSOR_DERECHA	1 s	SENSOR EN LA POSICION DERECHA
I_IMS_10_B4_CIL_PAR_RETROCESO	Bool	HMI_Connectio...	PLC_1	I_IMS_10_B4_CIL_PAR_RETROCESO	1 s	CILINDRO PARALELO EN POSICION DE RETROCESO
I_IMS_10_B5_CIL_PAR_AVANCE	Bool	HMI_Connectio...	PLC_1	I_IMS_10_B5_CIL_PAR_AVANCE	1 s	CILINDRO PARALELO EN POSICION DE AVANCE
I_IMS_10_B6_CIL_ELEV_RETROCESO	Bool	HMI_Connectio...	PLC_1	I_IMS_10_B6_CIL_ELEV_RETROCESO	1 s	CILINDRO ELEVADOR EN POSICION DE RETROCESO
IMS_10_LIBERAR_HMI	Bool	HMI_Connectio...	PLC_1	IMS_10_LIBERAR_HMI	1 s	BOTON PARA LIBERAR PALLETS DESDE EL HMI
IMS10_INICIO_HMI	Bool	HMI_Connectio...	PLC_1	IMS10_INICIO_HMI	1 s	BOTON PARA DAR INICIO O PARADA DESDE EL HMI
M_ALMACEN_LLENO	Bool	HMI_Connectio...	PLC_1	M_ALMACEN_LLENO	1 s	MEMORIA PARA MOSTRAR MENSAJE LLENO
m_error	Bool	HMI_Connectio...	PLC_1	m_error	1 s	MEMORIA ERROR
M_FINCARRERA	Bool	HMI_Connectio...	PLC_1	M_FINCARRERA	1 s	MEMORIA PARA GUARADAR EL VALOR DE FIN DE CAR...
M_IMS_10_ALMACENAR_HMI	Bool	HMI_Connectio...	PLC_1	M_IMS_10_ALMACENAR_HMI	1 s	START VIRTUAL DEL PROGRAMA POR HMI
M_IMS10_START	Bool	HMI_Connectio...	PLC_1	M_IMS10_START	1 s	ARRANCAR EL PROCESO
M_INICIADO_PROGRAMA	Bool	HMI_Connectio...	PLC_1	M_INICIADO_PROGRAMA	1 s	MEMORIA PRESENTAR MENSAJE DE INICIO
M_PALLET_1	Bool	HMI_Connectio...	PLC_1	M_PALLET_1	1 s	PARA INDICAR QUE HAY 1 PALLET
M_PALLET_2	Bool	HMI_Connectio...	PLC_1	M_PALLET_2	1 s	PARA INDICAR QUE HAY 2 PALLET
M_PALLET_3	Bool	HMI_Connectio...	PLC_1	M_PALLET_3	1 s	PARA INDICAR QUE HAY 3 PALLET
M_PALLET_4	Bool	HMI_Connectio...	PLC_1	M_PALLET_4	1 s	PARA INDICAR QUE HAY 4 PALLET

Figura 3.16 Variables HMI

Variables HMI

Nombre	Tipo de datos	Conexión	Nombre del PLC	Variable PLC	Ciclo de adquisi..	Comentario de origen
M_RESET_CONTADOR	Bool	HMI_Connectio...	PLC_1	M_RESET_CONTADOR	1 s	RESETEA EL CONTADOR DEL CONTROL DE PALLET
M_RESET_HMI	Bool	HMI_Connectio...	PLC_1	M_RESET_HMI	1 s	RESET DEL CONTADOR HMI
M_SENSOR_DER_HMI	Bool	HMI_Connectio...	PLC_1	M_SENSOR_DER_HMI	1 s	MEMORIA PARA PRENSENOSDERECHA EN HMI
M_SENSOR_IZQ_HMI	Bool	HMI_Connectio...	PLC_1	M_SENSOR_IZQ_HMI	1 s	MEMORIA PARA PRENSENOSR IZQUIERDA EN HMI
M_VACIO	Bool	HMI_Connectio...	PLC_1	M_VACIO	1 s	MEMORIA MENSAJE ALMACEN VACIO
MW_NUMERO_PALLETS_HMI	Word	HMI_Connectio...	PLC_1	MW_NUMERO_PALLETS_HMI	1 s	MUESTRA LA CANTIDAD DE PALLETS EN EL HMI
Q_ESCLAVO8_MOTOR_ADELANTE	Bool	HMI_Connectio...	PLC_1	Q_ESCLAVO8_MOTOR_ADELANTE	1 s	MOTOR ENCENDIDO HACIA ADELANTE ESCLAVO 8
Q_IMS_10_M2_CIL_MED_PARADA	Bool	HMI_Connectio...	PLC_1	Q_IMS_10_M2_CIL_MED_PARADA	1 s	CILINDRO MEDIO NEUMATICO PARA EL CILINDRO MED...
Q_IMS_10_M3_CIL_NEU_ELEVADOR	Bool	HMI_Connectio...	PLC_1	Q_IMS_10_M3_CIL_NEU_ELEVADOR	1 s	CILINDRO NEUMATICO PARA EL CILINDRO ELEVADOR
Q_IMS_10_M4_CIL_LAT_PARALELO	Bool	HMI_Connectio...	PLC_1	Q_IMS_10_M4_CIL_LAT_PARALELO	1 s	CILINDRO NEUMATICO PARA EL CILINDRO LATERAL
Q_IMS_10_M5_CILINDRO_SEPARADOR	Bool	HMI_Connectio...	PLC_1	Q_IMS_10_M5_CILINDRO_SEPARADOR	1 s	CILINDRO NEUMATICO PARA EL CILINDRO SEPARADOR
QS_IMS_10_MOTOR_LENTO	Bool	HMI_Connectio...	PLC_1	QS_IMS_10_MOTOR_LENTO	1 s	MOTOR ENCENDIDO LENTAMENTE

Figura 3.17 Variables HMI (Continuación)

Las botoneras que se visualizan en el HMI, nos sirven para poder interactuar con la co-simulación, ya que, de acuerdo a lo visto en el **Diagrama de Flujo (2.3.3)**, se utilizan como condiciones, las instrucciones recibidas desde de los pulsadores “MARCHA / PARO”, “RESET”, “ALMACENAR” y “LIBERAR”, de acuerdo al estado en el que se encuentre el proceso descrito. Para poder realizar el procedimiento anteriormente explicado, se necesitó declarar como salidas del PLC, las memorias que almacenan las instrucciones de cada botonera, y a su vez, para que estas salidas puedan ser recibidas como entradas en la co-simulación, se tuvieron que leer de acuerdo a lo explicado en la **Función READ**, y declararlas como variables globales, que de acuerdo a lo ya explicado nos permite la comunicación entre distintas aplicaciones.

También se presenta el “Bloque de error”, PALLET ATORADO, el mismo que se lo expuso en el *main*, y que a su vez es comandado desde la simulación.

3.1.3 Configuración de la co-simulación y API

Se desarrolló un programa que controla un sistema de almacenado de pallets. Para poder realizar una prueba completa del funcionamiento, el programa de STEP 7 se carga a través de S7 PLCSIM ADVANCED, en un controlador S7-1500 virtual. Este controlador interactúa a través de la API con una co-simulación (simulación del sistema), para validar el programa STEP 7 en el contexto del sistema.

El sistema de almacenamiento de pallets que se controla, como ya lo vimos en el capítulo: Marco Teórico, consta de una banda transportadora y el almacén de pallets, con sus respectivos sensores y actuadores. Para fines prácticos, se realizó una lista de etiquetas de entrada y salida para que el controlador se comunique con el sistema de manejo de materiales, definida de la manera mostrada en las Figura 3-18, 3-19 y 3-20.

	Nombre	Tipo de datos	Dirección	Rema-nencia	Accesible desde HMI/OPC UA	Escribible desde HMI/OPC UA	Visible en HMI Engi-neering	Supervisión	Comentario
	I_IMS_10_INICIO_FISICO	Bool	%I200.0	False	True	True	True		INICIO FISICO DEL PROGRAMA
	M_IMS_10_INICIO_HMI	Bool	%M0.3	False	True	True	True		INICIO VIRTUAL DEL PROGRAMA POR HMI
	Q_IMS_10_M2_CIL_MED_PARADA	Bool	%Q0.0	False	True	True	True		CILINDRO MEDIO NEUMATICO PARA EL CILINDRO MEDIO DE PARADA
	M_MEM_BOTELLAS	Bool	%M9.6	False	True	True	True		MEMORIA ALMACENAR EL VALOR DE BOTELLAS
	I_IMS_10_ALMACENAR_FISICO	Bool	%I200.1	False	True	True	True		START FISICO PARA EMPEZAR A ALMA-CENAR
	Q_ESCLAVO8_MOTOR_ADELANTE	Bool	%Q3.0	False	True	True	True		MOTOR ENCENDIDO HACIA ADELANTE ESCLAVO 8
	QS_ESCLAVO8_MOTOR_LENTO	Bool	%Q3.2	False	True	True	True		MOTOR ENCENDIDO LENTAMENTE ESCLAVO 8
	M_RESET_HMI	Bool	%M0.6	False	True	True	True		RESET DEL CONTADOR HMI
	I_RESET_FISICO	Bool	%I200.2	False	True	True	True		RESET DEL CONTADOR FISICO
	M_IMS_10_LIBERAR	Bool	%M0.7	False	True	True	True		PARA LIBERAR PALLETS
	M_IMS_10_LIBERAR_HMI	Bool	%M1.1	False	True	True	True		LIBERAR VIRTUAL DEL PROGRAMA POR HMI
	M_VALOR_INICIAL	Bool	%M9.2	False	True	True	True		MEMORIA PARA GUARADR EL VALOR DEL DEL VALOR INICIAL
	T_DELAY_ESTADO4_LIBERAR	Timer	%T5	False	True	True	True		TIEMPO PARA LIBERAR DESDE EL ESTA-DO 3 AL ESTADO 4
	M_RESET_CONTADOR	Bool	%M0.2	False	True	True	True		RESETEA EL CONTADOR DEL CONTROL DE PALLET
	IMS10_INICIO_HMI	Bool	%M7.0	False	True	True	True		BOTON PARA DAR INICIO O PARADA DESDE EL HMI
	IMS10_INICIO_HMI	Bool	%M7.0	False	True	True	True		BOTON PARA DAR INICIO O PARADA DESDE EL HMI
	M_IMS_10_ALMACENAR_HMI	Bool	%M0.4	False	True	True	True		START VIRTUAL DEL PROGRAMA POR HMI
	IMS_10_LIBERAR_HMI	Bool	%M7.1	False	True	True	True		BOTON PARA LIBERAR PALLETS DESDE EL HMI
	M_PALLET_4	Bool	%M8.2	False	True	True	True		PARA INDICAR QUE HAY 4 PALLET

Figura 3.18 Variables internas PLC

109739660_Transport_V15.1_1 / PLC_1 [CPU 1516-3 PN/DP] / Variables PLC / InternTags [63]

Variables PLC

Variables PLC

	Nombre	Tipo de datos	Dirección	Rema-nencia	Accesible desde HMI/OPC UA	Escribible desde HMI/OPC UA	Visible en HMI Engi-neering	Supervisión	Comentario
	m_error	Bool	%M4.0	False	True	True	True		MEMORIA ERROR
	M_PARO_EMERGENCIA	Bool	%M10.1	False	True	True	True		MEMORIA ALMACENAR EL VALOR DE BOTELLAS
	Q_ESCLAVO9_MOTOR_ADELANTE	Bool	%Q5.0	False	True	True	True		MOTOR ENCENDIDO HACIA ADELANTE ESCLAVO 9
	QS_ESCLAVO9_MOTOR_LENTO(1)	Bool	%Q5.1	False	True	True	True		MOTOR ENCENDIDO LENTAMENTE ESCLAVO 9
	M_SENSOR_IZQ_HMI	Bool	%M8.4	False	True	True	True		MEMORIA PARA PRENSENOSR IZ-QUIERDA EN HMI
	T_TEMP_SENSOR_IZQ	Timer	%T9	False	True	True	True		TEMPORIZADOR PARA EL SENSOR IZ-QUIERDA
	M_IZQ	Bool	%M8.5	False	True	True	True		MEMORIA PARA GUARADAR EL VALOR DE LA ENTRADA DEL SENSOR IZQUIER-DA
	M_SENSOR_DER_HMI	Bool	%M8.6	False	True	True	True		MEMORIA PARA PRENSENOSDERECHA EN HMI
	M_DERECHA	Bool	%M8.7	False	True	True	True		MEMORIA PARA GUARADAR EL VALOR DE LA ENTRADA DEL SENSOR DERE-CHA
	T_TEMP_SENSOR_DERE	Timer	%T10	False	True	True	True		TEMPORIZADOR PARA EL SENSOR DER-ECHA
	M_SEP_CILIN	Bool	%M9.1	False	True	True	True		MEMORIA PARA ENCENDER CILINDRO EN EL HMI
	M_CILINDRO DE PARADA	Bool	%M10.3	False	True	True	True		MEMORIA PARA CILINDRO DE PARADA
	M_SEPARADOR	Bool	%M9.0	False	True	True	True		MEMORIA PARA GUARADR EL VALOR DEL CILINDRO SEPARADOR
	M_DESACTIVAR_B8	Bool	%M10.2	False	True	True	True		MEMORIA PARA DESCATIVAR B8
	T_TEMP_SEPARADOR	Timer	%T11	False	True	True	True		TEMPORIZADOR PARA PRENDER EL SENSOR SEPARADOR UN TIEMPO
	M_FINCARRERA	Bool	%M8.3	False	True	True	True		MEMORIA PARA GUARADAR EL VALOR DE FIN DE CARRERA
	M_MOSTRAR_INICIAL	Bool	%M9.3	False	True	True	True		MEMORIA PARA MOSTRAR EL VALOR INICIAL EN HMI
	M_SEP	Bool	%M9.4	False	True	True	True		MEMORIA PARA SENSOR DE SEPARA-CION EN HMI
	SENSOR_MAGNETICO	Bool	%I5.5	False	True	True	True		PARA EN SENSOR MAGENITO VERIFICA SI HAY BOTELLAS
	SENSOR_CAPACITIVO	Bool	%I5.6	False	True	True	True		PARA EN SENSOR CAPACITIVO VERIFI-CA SI HAY PALLETES
	T_TEMP_BOTELLAS	Timer	%T13	False	True	True	True		VERIFICA SI HAY BOTELLAS Y PALLETS
	MW_NUMERO_PALLETS_HMI	Word	%MW2	False	True	True	True		MUESTRA LA CANTIDAD DE PALLETS EN EL HMI
	M_ALMACEN_LLENO	Bool	%M7.2	False	True	True	True		MEMORIA PARA MOSTRAR MENSAJE LLENO
	T_TEMPOVACIO	Timer	%T8	False	True	True	True		TEMPORIZADOR VACIO
	M_PALLET_2	Bool	%M8.0	False	True	True	True		PARA INDICAR QUE HAY 2 PALLET
	M_PALLET_1	Bool	%M7.7	False	True	True	True		PARA INDICAR QUE HAY 1 PALLET
	M_PALLET_3	Bool	%M8.1	False	True	True	True		PARA INDICAR QUE HAY 3 PALLET
	T_TEMP_VALOR INICIAL	Timer	%T12	False	True	True	True		TEMPORIZADOR PARA PRENDER EL SENSOR SEPARADOR UN TIEMPO
	T_DELAY_ESTADO8_LIBERAR	Timer	%T6	False	True	True	True		TIEMPO PARA LIBERAR DESDE EL ESTA-DO 7 AL ESTADO 8
	M_VACIO	Bool	%M7.5	False	True	True	True		MEMORIA MENSAJE ALMACEN VACIO
	M_IMS10_INICIO	Bool	%M0.0	False	True	True	True		INICIO DEL PROGRAMA
	T_DELAY_ESTADO2_LIBERAR	Timer	%T4	False	True	True	True		TIEMPO PARA LIBERAR DESDE EL ESTA-DO 1 AL ESTADO 2
	I_IMS_10_LIBERAR_FISICO	Bool	%I200.3	False	True	True	True		LIBERAR FISICO DEL PROGRAMA
	T_DELAY_ESTADO1	Timer	%T0	False	True	True	True		TIEMPO DE REINICIO DEL PROGRAMA
	M_IMS10_START	Bool	%M0.1	False	True	True	True		ARRANCAR EL PROCESO
	I_IMS10_B8_SEPARACION	Bool	%I0.5	False	True	True	True		SENSOR DE SEPARACION
	MW_CANTIDAD_PALLET	Int	%MW0	False	True	True	True		GUARDA LA CANTIDAD DE PALLETS
	T_DELAY_ESTADO2	Timer	%T1	False	True	True	True		TIEMPO PARA IR AL ESTADO 2 DESDE 8
	T_DELAY__ESTADO4	Timer	%T2	False	True	True	True		TIEMPO PARA IR AL ESTADO 4 DESDE EL ESTADO 3
	T_DELAY_ESTADO6	Timer	%T3	False	True	True	True		TIEMPO PARA IR AL ESTADO 6 DESDE EL ESTADO 5
	M_INICIADO_PROGRAMA	Bool	%M10.0	False	True	True	True		MEMORIA PRESENTAR MENSAJE DE INICIO
	T_DESACTIVAR_B8	Timer	%T15	False	True	True	True		TEMPORIZADOR PARA DESCATIVAR B8
	M_SIN_BOTELLAS	Bool	%M9.7	False	True	True	True		SIN BOTELLAS
	QS_IMS_10_MOTOR_LENTO	Bool	%Q1.2	False	True	True	True		MOTOR ENCENDIDO LENTAMENTE
	T_TEMP_SIN_BOTELLAS	Timer	%T14	False	True	True	True		SIN BOTELLAS

Figura 3.19 Varibales Internas PLC (continuación)

109739660_Transport_V15.1_1 / PLC_1 [CPU 1516-3 PN/DP] / Variables PLC / CommunicationWithCoSim [18]

Variables PLC

Variables PLC

	Nombre	Tipo de datos	Dirección	Rema-nencia	Accesible desde HMI/OPC UA	Escribible desde HMI/OPC UA	Visible en HMI Engi-neering	Supervisión	Comentario
	simulate_error	Bool	%I4.1	False	True	True	True		ACTIVA SIMULAR ERROR
	ack_signal	Bool	%I4.2	False	True	True	True		
	Q_IMS_10_MOTOR_ADELANTE	Bool	%Q1.0	False	True	True	True		MOTOR ENCENDIDO HACIA ADELANTE
	Q_IMS_10_M4_CIL_LAT_PARALE-LO	Bool	%Q0.2	False	True	True	True		CILINDRO NEUMATICO PARA EL CILIN-DRO LATERAL
	Q_IMS_10_M3_CIL_NEU_ELEVA-DOR	Bool	%Q0.1	False	True	True	True		CILINDRO NEUMATICO PARA EL CILIN-DRO ELEVADOR
	Q_IMS_10_M5_CILINDRO_SEPA-RADOR	Bool	%Q0.3	False	True	True	True		CILINDRO NEUMATICO PARA EL CILIN-DRO SEPARADOR
	I_IMS_10_B1_SENSOR_IZQUIERDA	Bool	%I1.3	False	True	True	True		SENSOR EN LA POSICION IZQUIERDA
	I_IMS_10_B5_CIL_PAR_AVANCE	Bool	%I0.2	False	True	True	True		CILINDRO PARALELO EN POSICION DE AVANCE
	I_IMS_10_B4_CIL_PAR_RETROCE-SO	Bool	%I0.1	False	True	True	True		CILINDRO PARALELO EN POSICION DE RETROCESO
	I_IMS_10_B6_CIL_ELEV_RETROCE-SO	Bool	%I0.3	False	True	True	True		CILINDRO ELEVADOR EN POSICION DE RETROCESO
	I_IMS_10_B2_SENSOR_DERECHA	Bool	%I1.4	False	True	True	True		SENSOR EN LA POSICION DERECHA
	I_FIN_CARRERA	Bool	%I0.6	False	True	True	True		INDICADOR DE LLENADO COMPLETO
	Q_RESET_HMI	Bool	%Q6.0	False	True	True	True		BOTON RESET DEL HMI
	Q_INICIO_HMI	Bool	%Q6.1	False	True	True	True		BOTON MARCHA PARO DEL HMI
	Q_ALMACENAR_HMI	Bool	%Q7.0	False	True	True	True		BOTON ALMACENAR DEL HMI
	Q_LIBERAR_HMI	Bool	%Q7.1	False	True	True	True		BOTON LIBERAR DEL HMI
	Q_PALLET_4	Bool	%Q7.2	False	True	True	True		INDICA QUE HAY 4 PALLETS
	Q_PALLET_VACIO	Bool	%Q7.3	False	True	True	True		INDICA QUE NO HAY PALLETS

Figura 3.20 Variables del PLC: Comunicación con la co-simulación

En la Co-simulación desarrollada, tenemos el mismo PLC, pero de forma virtual, desarrollado en el software S7 PLCSIM Advanced, y a través de un API (Interfaz de Programación de Aplicaciones), nos comunicamos con la misma planta modelada en lenguaje de programación C# (ejecutada en Visual Studio), con esto llegarán al PLC las señales, que en un entorno real, se obtendrían por medio de las entradas físicas del hardware, y de esta manera, se ejecuten las acciones o comandos indicadas por el PLC virtual.

En resumen, las salidas del PLC (comandos hacia los actuadores), serán registradas como entradas en la co-simulación. Por otra parte, las señales provenientes de los sensores, entradas del PLC, serán las salidas en la co-simulación.

3.2 Programación de la Planta IMS-10 Simulada

Para modelar el comportamiento de la planta se realizó la programación en el lenguaje de programación orientado a objetos, C#, mismo que se ejecuta en el software Visual Studio de Microsoft.

El proyecto en Visual Studio del ejemplo utilizado, incluye facilidades en el acceso para la programación de la aplicación de S7 PLCSIM ADVANCED con la Co-simulación.

Algunas funciones básicas se incluyen en el ejemplo, como lo son: El encendido del controlador virtual, y el intercambio de datos de entrada y salida, entre otros, permitiendo profundizar en el desarrollo a criterio del programador.

En capítulos posteriores se agrega la visión general acerca del C# y funciones API que fueron programadas en este proyecto.

3.2.1 Programa en lenguaje de programación C#

Lo programado en la aplicación externa (C#), es lo siguiente:

- Funciones de interfaz de usuario (*Runtime* del API), a fin de interactuar con el controlador del S7 PLCSIM ADVANCED
- Simulación del sistema de almacenamiento (co-simulación)

La co-simulación reacciona a las señales de control (salidas), del controlador virtual y simula todas las señales de sensores requeridas (entradas), para el control secuencial de la simulación.

Para el intercambio de datos de entrada / salida, el *runtime* del API escribe y lee desde el área de memoria que está sincronizada en el punto de control de ciclo con la imagen del proceso interno, del controlador S7-1500.

3.2.1.1 Rango de funciones

En la aplicación escrita en C#, las funciones detalladas en la Tabla 3-4, son utilizadas y mostradas en el *runtime* del API:

Tabla 3-4 Funciones utilizadas y mostradas en el *runtime* del API

Función	Descripción
RegisterInstance()	Registra una nueva instancia de un controlador virtual en el Administrador de runtimes (Runtime Manager)
UnregisterInstance()	Saca del registro una instancia del administrador de Runtime
PowerOn()	Crea el proceso para la instancia del simuladore de Runtime, e inicia el firmware del controlador virtual.
PowerOff()	Desactiva la instancia de la simulación y cierra su proceso.
Run()	Solicita al controlador virtual que cambie al estado operativo RUN.
Stop()	Solicita al controlador virtual que cambie al estado operativo STOP.
SetIPSuite()	Establece la suite IP de la interfaz de red de un controlador virtual.
UpdateTagList()	Lee los tags desde el controlador virtual y los escribe en el almacenamiento compartido, ordenados por nombre.
ReadBool()	Lee el valor de un tag del PLC, simbólicamente.
WriteBool()	Escribe el valor de un tag del PLC, simbólicamente
IsAlwaysSendOnEndOfCycleEnabled{get; set;}	Provee o establece el modo AlwaysSendOnEndOfCycle. Cuando se establece este modo, el evento OnEndOfCycle es disparado para cada modo después de cada fin de ciclo.
CommunicationInterface{get; set;}	Establece la interfaz de comunicación del controlador virtual, o la retorna: Comunicación Local (softbus), o TCP/IP.
OnConfigurationChanged	Este evento es disparado cuando la configuración del controlador virtual ha cambiado.
OnEndOfCycle	Este evento es disparado cuando el controlador virtual ha alcanzado el fin del ciclo MAIN.

Programación de la interfaz gráfica en Visual Studio

A continuación, se detalla el procedimiento para la programación gráfica en Visual Studio.

1. Desde la aplicación, Se abre la vista de la ventana principal “MainWindow.xaml” (Carpeta del proyecto / en la carpeta “Views”) (Figura 3-21):

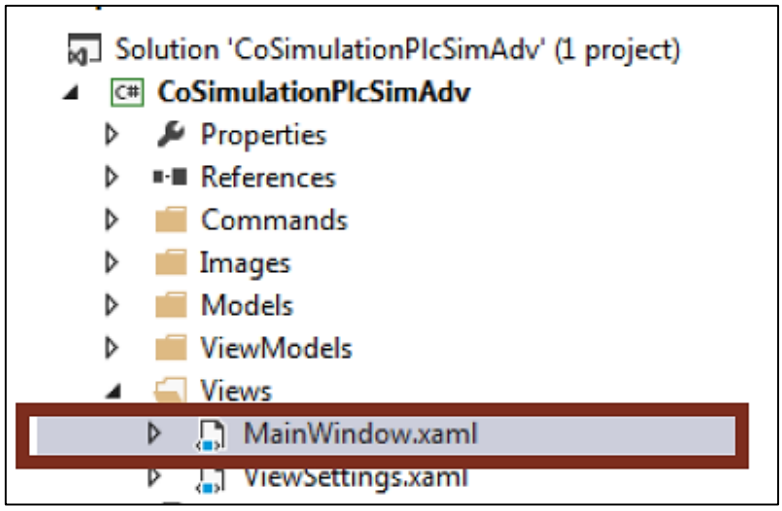


Figura 3.21 Programación de la interfaz Gráfica (Visual Studio)

2. Aquí, en el código XAML, entre otras cosas, se encuentran los comandos integrados para los botones o lo datos integrados para los campos del display (Figura 3-22).

```
    <Button Command="{Binding PowerOnInstanceCommand}" Grid.Column="0" Content="POWER ON" Horizontal.
    <Button Command="{Binding PowerOffInstanceCommand}" Grid.Column="0" Content="POWER OFF" Horizont
    <Button Command="{Binding RunInstanceCommand}" Grid.Column="0" Content="RUN" HorizontalAlignment
    <Button Command="{Binding StopInstanceCommand}" Grid.Column="0" Content="STOP" HorizontalAlignme

    <Button Command="{Binding CosimulationStartCommand}" Grid.Column="2" Content="START" HorizontalA
    <Button Command="{Binding CosimulationStopCommand}" Grid.Column="2" Content="STOP" HorizontalAli
    <Button Command="{Binding CosimulationErrorCommand}" Grid.Column="2" Content="SIMULATE ERROR" Ho
    <Button Command="{Binding CosimulationPackageOKCommand}" Grid.Column="2" Content="PACKAGE OK" Ho


    <TextBox Text="{Binding StatusPLCInstance}" Name="tBPLCInstance" Grid.Column="0" VerticalAlignme
    <TextBox Text="{Binding StatusCoSimulation}" Name="tBCoSimulation" Grid.Column="2" VerticalAlign

</Grid>
<ListBox Grid.Row="4" ItemsSource="{Binding StatusListView}" ScrollViewer.HorizontalScrollBarVisibil
    <ListBox.ItemTemplate>
```

Figura 3.22 Programación de la interfaz gráfica (Visual Studio), código XAML

3.2.2 Programación de la instancia del PLC

En este epígrafe, se mostrará el detalle de las *tags* principales que constan en el programa desarrollado en C#, se agregarán capturas de la presentación de cada una de ellas.

3.2.2.1 Creación de la instancia

1. En la carpeta “ViewModels”, se abre la clase “MainWindowViewModel.cs”
2. En el bloque de región # “Fields”, se define un tag “virtualController”, de tipo objeto “PLCInstance” (Figura 3.23)

```
/// <summary>
/// PLCSIM Adv. Instance of the virtual controller
/// </summary>
public PLCInstance virtualController = null;
```

Figura 3.23 Creación de tag de PLC “virtualController”

3. En el constructor de clases (bloque “C’Tor”), se crea un Nuevo objeto de clase “PLCInstance”, y se asigna al tag “VirtualController”. “TransportControl” es transferido como nombre para la nueva instancia. (Figura 3.24)

```
public MainWindowViewModel()
{
    StatusListView = new ObservableCollection<String>();

    try
    {
        // New PLC instance Name:"TransportControl"
        virtualController = new PLCInstance("TransportControl");
```

Figura 3.24 Creación de objeto " TransportControl"

1. En la carpeta “Models”, se abre la clase “PLCInstance.cs”.
2. En las Propiedades del bloque #region, se define un tag de instancia, la interfaz de usuario “IInstance”. (Figura 3.25)

```
/// <summary>
/// instance of PLCSIM Adv. virtual Controller
/// </summary>
public IInstance instance { get; set; }
```

Figura 3.25 Creación de interfaz de usuario “IInstance”

3. En el constructor de clases (bloque “C’Tor” de #region), se registra una nueva instancia de controlador virtual con la función RegisterInstance()”en el Runtime Manager del API, SimulationRuntimeManager. La función crea y regresa una interfaz desde esta instancia. La interfaz creada es asignada al tag de la instancia. (Figura 3.26)

```
public PLCInstance(string instanceName)
{
    instance = SimulationRuntimeManager.RegisterInstance(instanceName);
    instance.IsAlwaysSendOnEndOfCycleEnabled = true;
    instance.CommunicationInterface = ECommunicationInterface.Softbus;
    instance.OnConfigurationChanged += instance_OnConfigurationChanged;
    instance.OnEndOfCycle += instance_OnEndOfCycle;
}
```

Figura 3.26 Registro de instancia “SimulationRuntimeManager” (2)

4. Además de la instancia registrada, se realizan los siguientes ajustes (Figura 3.27):
   - “instance.CommunicationInterface = ECommunicationInterface.Softbus”, Ajusta la interfaz de comunicación del controlador virtual a la comunicación local (softbus).
   - “instance.IsAlwaysSendOnEndOfCycleEnabled = true”, provoca que el evento “OnEndOfCycle” se dispare al final de cada ciclo del CPU.
   - A cada uno delos eventos “OnConfigurationChanged” y “OnEndOfCycle” se les asigna un controlador de eventos.

```
public PLCInstance(string instanceName)
{
    instance = SimulationRuntimeManager.RegisterInstance(instanceName);
    instance.IsAlwaysSendOnEndOfCycleEnabled = true;
    instance.CommunicationInterface = ECommunicationInterface.Softbus;
    instance.OnConfigurationChanged += instance_OnConfigurationChanged;
    instance.OnEndOfCycle += instance_OnEndOfCycle;
}
```

Figura 3.27 Otros ajustes en "PLCInstance.cs" (3)

3.2.2.2 De-registro de instancias

1. En la carpeta "Views" Se abre la clase "MainWindow.xaml.cs".
2. El evento "Window_Closing" llama a la función del API IInstance: "UnregisterInstance()". Esto retira la instancia del registro del controlador virtual en el Runtime Manager. (Figura 3.28)

```
private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)
{
    ViewModels.MainWindowViewModel Application = this.DataContext as ViewModels.MainWindowViewModel;
    //Unregister Instance
    Application.virtualController.instance.UnregisterInstance();
}
```

Figura 3.28 Retirando instancias del registro

La aplicación se Cierra con el botón "EXIT" o con el ícono "close", en la interfaz de usuario. Esto desencadena el evento "Window_Closing"

3.2.2.3 Encendido de la instancia

1. En la carpeta "ViewModels", abrir la clase "MainWindowViewModel.cs". (Figura 3.29)
2. Dando clic en el botón "POWER ON" de la interfaz de usuario, se llama al método "PowerOnPLCInstance()", del controlador virtual, en el bloque de la #region "Public Methods".

```
/// <summary>
/// Power On registred Instance of virtual controller
/// </summary>
public void PowerOnController()
{
    try
    {
        WriteStatusEntry(String.Format("Power On Instance: {0}", virtualController.instance.Name));
        virtualController.PowerOnPLCInstance();
    }
    catch (SimulationRuntimeException simRtEx)
    {
        WriteStatusEntry(String.Format("PowerOn Instance failed: {0}", simRtEx.Message));
    }
}
```

Figura 3.29 Encendido de instancia, Método "PowerOnPLCInstance()"

1. En la carpeta "Models", se abre la clase "PLCInstance.cs".
2. El método "PowerOnPLCInstance()" en el bloque de #region "Public Methods", llama a las funciones "PowerOn()" y "SetIPSuite()", en la IInstance API. La función "PowerOn()" crea el proceso para la simulación de la instancia runtime e inicia el firmware del controlador virtual. Se configura un conteo de 60.000 ms. La función "SetIPSuite()" configura la interfaz de red del controlador virtual. (Figura 3.30)

```
/// <summary>
/// Power On PLCSIM Adv. Instanz, set IPSuite of instance
/// </summary>
/// <returns></returns>
public void PowerOnPLCInstance()
{
    instance.PowerOn(60000);
    instance.SetIPSuite(0, instanceIP, true);
}
```

Figura 3.30 Configuración de Interfaz de red y timer de 60 seg.

3. Los parámetros (dirección IP, máscara de subred y puerta de enlace estándar) de la interfaz de red se definen en las propiedades del bloque #region con la etiqueta "instanceIP" del tipo de estructura "SIPSuite4" (Figura 3.31)

```
private SIPSuite4 instanceIP = new SIPSuite4("192.168.0.101", "255.255.255.0", "0.0.0.0");
```

Figura 3.31 Definición de los parámetros de red

3.2.2.4 Apagado de la instancia

1. En la carpeta “ViewModels”, se abre la clase “MainWindowViewModel.cs”.
2. Dando clic en “POWER OFF” en la interfaz de usuario, se llama el método del controlador virtual “PowerOffPLCInstance()”, en el bloque de #region “Public Methods”. (Figura 3.32)

```
/// <summary>
/// Power Off registred Instance of virtual controller
/// </summary>
public void PowerOffController()
{
    try
    {
        WriteStatusEntry(String.Format("Power Off Instance: {0}", virtualController.instance.Name));
        virtualController.PowerOffPLCInstance();
    }
    catch (SimulationRuntimeException simRtEx)
    {
        WriteStatusEntry(String.Format("PowerOff Instance failed: {0}", simRtEx.Message));
    }
}
```

Figura 3.32 Apagado de la instancia en el bloque “Public Methods”

1. En la carpeta “Models”, se abre la clase “PLCInstance.cs”.
2. El método “PowerOffPLCInstance()” del bloque de #region “Public Methods”, llama la función “PowerOff()”, de la IInstance API. Apaga la simulación del runtime y cierra el proceso. Se configura un conteo de 6000 ms. (Figura 3.33)

```
/// <summary>
/// Power Off PLCSIM Adv. Instance
/// </summary>
public void PowerOffPLCInstance()
{
    instance.PowerOff(6000);
}
```

Figura 3.33 Apagado de la instancia PLC

3.2.2.5 Arranque de la Instancia

1. En la carpeta “ViewModels”, se abre la clase “MainWindowViewModel.cs”.
2. Dando clic en “RUN”, en la interfaz de usuario, se llama el método “RunPLCInstance()”, en el bloque #region “Public Methods” del controlador virtual”. (Figura 3.34)

```
/// <summary>
/// Run registred Instance of virtual controller
/// </summary>
public void RunController()
{
    try
    {
        WriteStatusEntry(String.Format("Run Instance: {0}", virtualController.instance.Name));
        virtualController.RunPLCInstance();
    }
    catch (SimulationRuntimeException simRtEx)
    {
        WriteStatusEntry(String.Format("Run Instance failed: {0}! Please load plc program before execute RUN.", simRtEx.Message));

    }
}
```

Figura 3.34 Arranque de la instancia (RUN en interfaz de usuario)

1. En la carpeta “Models”, se abre la clase “PLCInstance.cs”.
2. El método “RunPLCInstance()” del bloque de #region “Public Methods”, llama la función “Run()”, de la Instance API. Esta función solicita al controlador virtual pasar al modo RUN. Se configura un conteo de 6000 ms. (Figura 3.35)

```
/// <summary>
/// Run PLCSIM Adv. Instance
/// </summary>
public void RunPLCInstance()
{
    instance.Run(6000);

}
```

Figura 3.35 Arranque de la instancia en el Controlador Virtual

3.2.2.6 Deteniendo Instancias

1. En la carpeta “ViewModels”, se abre la clase “MainWindowViewModel.cs”.
2. Dando clic en “STOP”, en la interfaz de usuario, se llama el método “StopPLCInstance()”, del controlador virtual, en el bloque #region “Public Methods”. (Figura 3.36)

```
/// <summary>
/// Stop registred Instance of virtual controller
/// </summary>
public void StopController()
{
    try
    {
        WriteStatusEntry(String.Format("Stop Instance: {0}", virtualController.instance.Name));
        virtualController.StopPLCInstance();

    }
    catch (SimulationRuntimeException simRtEx)
    {
        WriteStatusEntry(String.Format("Stop Instance failed: {0}", simRtEx.Message));
    }
}
```

Figura 3.36 Parada de la instancia en la interfaz de usuario

1. En la carpeta “Models”, se abre la clase “PLCInstance.cs”.
2. El método “StopPLCInstance()” en el bloque de #region “Public Methods”, llama la función “Stop()”, de la Instancia API. Esta función solicita al controlador virtual pasar al modo STOP. Se configura un conteo de 6000 ms. (Figura 3.37)

```
/// <summary>
/// Stop PLCSIM Adv. Instance
/// </summary>
public void StopPLCInstance()
{
    instance.Stop(6000);
}

#endregion //Public Methods
```

Figura 3.37 Parada de instancias en el controlador virtual

3.2.2.7 Intercambio de datos de Entradas / Salidas (I/O Data)

1. En la carpeta “Models”, se abre la clase “PLCInstance.cs”.
2. El controlador de eventos “instance_OnConfigurationChanged” en el bloque de #region “Eventos”, llama la función “UpdateTagList()”, de la instancia API. Esta función lee los tags del controlador y escribe en ellos, ordenados por nombre, en la memoria conjunta. A fin de leer los tags de entrada y salida del controlador virtual, el parámetro “ETagListDetails.IO” es transferido a la función. (Figura 3.38)

```
/// <summary>
/// Event when Configuration changed of the PLC (during download)
/// </summary>
/// <param name="in_Sender"> PLC which fired this event</param>
/// <param name="in_ErrorCode"> ErrorCode of Runtime of the PLC</param>
/// <param name="in_DateTime"> DateTime when the configuration changed</param>
void instance_OnConfigurationChanged(IInstance in_Sender, ERuntimeErrorCode in_ErrorCode, DateTime in_DateTime,
    EInstanceConfigChanged in_InstanceConfigChanged, uint in_Param1, uint in_Param2, uint in_Param3, uint in_Param4)
{

    IsConfigured = false;

    try
    {
        instance.UpdateTagList(ETagListDetails.IO);
        IsConfigured = true;
    }
    catch (Exception ex)
    {
    }
}
```

Figura 3.38 Intercambio de datos de Entradas / Salidas (I/O Data)

1. El controlador de eventos “instance_OnEndOfCycle” en el bloque de #region “Event”, llama la función “ReadBool()”, de la instancia API. Así las tags Booleanas del controlador virtual son leídos y escritos mediante el nombre del tag. (Figura 3.39)

```
if (IsConfigured)
{
    try
    {
        // Read outputs of the virtual controller and assign to variables of the Co-Simulation
        coSimulation.Q_IMS_10_MOTOR_ADELANTE = instance.ReadBool("Q_IMS_10_MOTOR_ADELANTE");
        coSimulation.Q_IMS_10_M4_CIL_LAT_PARALELO = instance.ReadBool("Q_IMS_10_M4_CIL_LAT_PARALELO");
        coSimulation.Q_IMS_10_M3_CIL_NEU_ELEVADOR = instance.ReadBool("Q_IMS_10_M3_CIL_NEU_ELEVADOR");
        coSimulation.Q_IMS_10_M5_CILINDRO_SEPARADOR = instance.ReadBool("Q_IMS_10_M5_CILINDRO_SEPARADOR");
        coSimulation.Q_RESET_HMI = instance.ReadBool("Q_RESET_HMI"); // BOTON RESET DEL HMI IMS10_INICIO_HMI
        coSimulation.Q_INICIO_HMI = instance.ReadBool("Q_INICIO_HMI"); // BOTON MARCHA/PARO DEL HMI
        coSimulation.Q_ALMACENAR_HMI = instance.ReadBool("Q_ALMACENAR_HMI"); // BOTON ALMACENAR DEL HMI
        coSimulation.Q_LIBERAR_HMI = instance.ReadBool("Q_LIBERAR_HMI"); // BOTON LIBERAR DEL HMI
        coSimulation.Q_PALLET_4 = instance.ReadBool("Q_PALLET_4"); // MEMORIA DEL CONTROLADOR QUE INDICA QUE YA HAY 4 PALLETS EN EL ALMACEN
        coSimulation.Q_PALLET_VACIO = instance.ReadBool("Q_PALLET_VACIO"); // MEMORIA DEL CONTROLADOR QUE INDICA QUE NO HAY PALLETS EN EL ALMACEN

        // Call the Co-Simulation programm
        coSimulation.CoSimProgramm();

        // Write the Co-Simulation values to the inputs of the virtual controller
        instance.WriteBool("I_IMS_10_B1_SENSOR_IZQUIERDA", coSimulation.I_IMS_10_B1_SENSOR_IZQUIERDA);
        instance.WriteBool("I_IMS_10_B5_CIL_PAR_AVANCE", coSimulation.I_IMS_10_B5_CIL_PAR_AVANCE);
        instance.WriteBool("I_IMS_10_B4_CIL_PAR_RETROCESO", coSimulation.I_IMS_10_B4_CIL_PAR_RETROCESO);
        instance.WriteBool("I_IMS_10_B6_CIL_ELEV_RETROCESO", coSimulation.I_IMS_10_B6_CIL_ELEV_RETROCESO);
        instance.WriteBool("I_IMS_10_B2_SENSOR_DERECHA", coSimulation.I_IMS_10_B2_SENSOR_DERECHA);
        instance.WriteBool("I_FIN_CARRERA", coSimulation.I_FIN_CARRERA);
        // instance.WriteBool("ack_signal", coSimulation.ack_signal);
```

Figura 3.39 Intercambio de datos de Entradas / Salidas (I/O Data) Tags Booleanas

3.2.3 Programación de la co-simulación

El programa de co-simulación fue desarrollado en el proyecto de Visual Studio, en el método “CoSimProgramm()” de la clase “Cosimulación.cs”.

Se desarrolló en una secuencia de casos (casos – switch). Para la simulación del sensor y la operación de la máquina, un temporizador arranca en el respectivo paso, después de ese lapso, se establece la etiqueta de “nextStep”

Tal como se comentó en epígrafes anteriores, la representación de transiciones y estados en las secuencias “Almacenar” y “Liberar”, en forma de bloque binarios de función, permitió su representación total en tablas booleanas,

Los casos – switch, establecieron las condiciones necesarias para alcanzar los pasos. Realizando la equivalencia en la co-simulación, los pasos se alcanzaron una vez que se recibieron las salidas (activación de actuadores), desde el controlador virtual (PLC), lo que a su vez resultó en salidas desde la co-simulación (señales de sensores). Permitiendo que se complete el ciclo deseado.

Para el desarrollo de este proyecto se elaboraron dos tablas (3-5 y 3-6), una por cada ciclo o transición, “Almacenar” y “Liberar”, utilizando el esquema de la Figura 3-56.

			Salidas desde PLC Virtual, Condiciones para el cambio de paso. (ej. Actuadores) Entradas en Co-simulación.				
			x		y	z	
		Paso 0	→	Paso 1	→	Paso 2	→	Paso n
Salidas desde Programa de co-simulación (Visual Studio), entradas en el PLC Virtual (ej. sensores)	a	1		0		0		0
	b	0		1		0		0
	c	0		0		1		0
	d	0		0		0		1

Figura 3.40 Diagrama explicaivo de tablas Booleanas de este proyecto

En el epígrafe posterior, se mostrará el detalle de las *tags* principales que constan en los parámetros de co-simulación, desarrollados en C#, se agregarán capturas de la presentación de cada una de ellas.

Tabla 3-5 Booleana, Transición: Almacenar

TRANSICIÓN "ALMACENAR"

	SALIDA PLC / ENTRADA CO-SIMULACIÓN (CONDICIÓN) --------->		Q_INICIO_HMI	NEXT STEP	Q_ALMACENAR_HMI	Q_PALLET_4	NEXT STEP	Q_IMS_10_MOTOR_ADELANTE	NEXT STEP	Q_IMS_10_M4_CIL_LAT_PARALELO	NEXT STEP	Q_IMS_10_M3_CIL_NEU_ELEVADOR	Q_IMS_10_M4_CIL_LAT_PARALELO	Q_IMS_10_M5_CILINDRO_SEPARADOR	NEXT STEP	NEXT STEP	NEXT STEP	NEXT STEP	PACKAGE OK	
	PASO --------->		0		1			2		3		4				5		6		7		99		99
	NOTAS --------->		OTRO POSIBLE INICIO: - RESET Q_RESET_HMI		PROGRAMA INICIADO POR MEDIO DE BOTÓN MARCHA/PARO DE HMI, EN ESPERA DE INICIAR ALMACENAMIENTO			SE INICIA PROCESO DE ALMACENADO CON BOTÓN: ALMACENAR		EN ESTE ESTADO SE PUEDE PROVOCAR EL ERROR PRESIONANDO: SIMULATE ERROR EN EL MAIN WINDOW EN LA CO-SIMULACIÓN (VISUAL STUDIO) - SE ANULA ERROR CON BOTON: PACKAGE OK							EN ESTE ESTADO SE COMPLETA ALMACENAMIENTO DE UN PALLET; SI AUN SE VALIDA ESPACIO EN MEMORIA, CONTINÚA A PASO 2, REPITIÉNDOSE EL CICLO - ES POSIBLE DETENER ALMACENAMIENTO Y EMPEZAR LIBERACIÓN, PRESIONANDO MARCHA/PARO EN HMI, VOLVIENDO AL "ESTADO 1" DONDE SE PODRÁ DECIDIR ENTRE, LIBERAR O ALMACENAR - AL COMPLETAR LA CAPACIDAD DEL ALMACÉN, SE CONTINÚA AL PASO 7, AL HABERSE ACTIVADO EL SENSOR FIN DE CARRERA.		EN ESTE ESTADO SE LLENÓ EL ALMACEN DE PALLETS - SE ACTIVA SENSOR FIN DE CARRERA - SE DIRIGE HACIA EL PASO 1 ÚNICAMENTE CON LA OPCIÓN DE LIBERAR		ESTADO DE ERROR - EN STEP 3 PULSANDO: - "SIMULATE ERROR" EN LA MAIN WINDOW DE LA CO-SIMULACION, SE ACTIVA "simulate_error" EN CONTROLADOR - PARA REANUDAR, SE PULSA "PACKAGE OK" EN LA MAIN WINDOWS DE LA CO-SIMULACION, SE ACTIVA "ack_signal" EN CONTROLADOR		
	PLC tag	Address																					
TRANSICION DEL 2 AL 3	I_IMS_10_B1_SENSOR_IZQUIERDA	%I1.3	0		0			0		1		0				0		0		0		0		0
TRANSICION DEL 4 AL 5	I_IMS_10_B5_CIL_PAR_AVANCE	%I0.2	0		0			0		0		1				0		0		0		0		0
TRANSICION DEL 6 AL 7	I_IMS_10_B4_CIL_PAR_RETROCESO	%I0.1	0		0			0		0		0				1		0		0		0		0
TRANSICION DEL 7 AL 8	I_IMS_10_B6_CIL_ELEV_RETROCESO	%I0.3	0		0			0		0		0				0		1		0		0		0
ERROR	simulate_error	%I4.1	0		0			0		0		0				0		0		0		1		0
ERROR	ack_signal	%I4.2	0		0			0		0		0				0		0		0		0		1

Tabla 3-6 Booleana, Transición: Liberar

TRANSICIÓN "LIBERAR"

	SALIDA PLC / ENTRADA CO-SIMULACIÓN (CONDICIÓN) ---------->		Q_LIBERAR_HMI	NEXT STEP		Q_IMS_10_M4_CIL_LAT_PARALELO	NEXT STEP		Q_IMS_10_M3_CIL_NEU_ELEVADOR	Q_IMS_10_M4_CIL_LAT_PARALELO	NEXT STEP		Q_IMS_10_MOTOR_ADELANTE	NEXT STEP	
	PASO ---------->		0		8			9				10			11
	NOTAS ---------->		OTRO POSIBLE INICIO: - RESET Q_RESET_HMI		PROCESO "LIBERAR" INICIADO										EN ESTE ESTADO SE TERMINA DE COMPLETA LA LIBERACIÓN DEL PALLET: - SI AUN EXISTEN PALLETS (MEMORIA), REGRESA AL PASO 8, Y SE CONTINUA LIBERANDO. - SE PUEDE INTERRUMPIR EL LIBERADO Y VOLVER A ALMACENAR, PRESIONANDO **PARO/MARCHA** EN HMI, VOLVIENDO AL ESTADO 1 TENIENDO NUEVAMENTE LA OPCION ENTRE LIBERAR O ALMACENAR. - SI EL ALMACEN ESTA VACIO (CONTEO DE PALLETS= 0), PREMANECE EN ESTADO 1 A HASTA RECIBIR ORDEN **ALMACENAR** EXCLUSIVAMENTE
	PLC tag	Address													
TRANSICION DEL 2.1 AL 3.1	I_IMS_10_B5_CIL_PAR_AVANCE	%I0.2	0		0			1				0			0
TRANSICION DEL 5.1 AL 6.1	I_IMS_10_B6_CIL_ELEV_RETROCESO	%I0.3	0		0			0				1			0
TRANSICION DEL 6.1 AL 7.1	I_IMS_10_B2_SENSOR_DERECHA	%I1.4	0		0			0				0			1

3.2.3.1 Parámetros de simulación

1. En la carpeta “ViewModels”, se abre la clase “MainWindowViewModel.cs”.
2. En el bloque #region “Fields”, se define una tag “transportSystem” del objeto “Cosimulation” (Figura 3.41)

```
public Cosimulation transportSystem = null;
```

Figura 3.41 Parámetros de simulación, clase “MainWindowViewModel.cs”

3. Se crea un nuevo objeto “transportSystem” de clase “Cosimulation”, en el constructor de clases (bloque de la región “C’Tor”), con tiempos definidos para los actuadores y los sensores.

3.2.3.2 Intercambio de datos

1. En la carpeta “ViewModels”, se abre la clase “MainWindowViewModel.cs”.
2. En el objeto “transportSystem” se almacena una referencia del objeto “CoSimulation”, dentro del constructor de clases (bloque #region “C’tor”). (Figura 3.42)

```
virtualController.coSimulation = transportSystem;
```

Figura 3.42 Intercambio de datos, clase "MainWindowViewModel.cs"

1. En la carpeta “Models”, se abre la clase “PLCInstance.cs”.
2. El controlador de eventos “instance_OnEndOfCycle” en el bloque de #region “Event”, la lectura de los tags de salida del controlador virtual, son transferidos a la co-simulación (1). Acto seguido, hay un llamado del programa de co-simulación “CoSimProgramm()”(2). Finalmente, los valores simulados de la co-simulación se escriben en los tags de entrada del controlador virtual (3). (Figura 3.43)

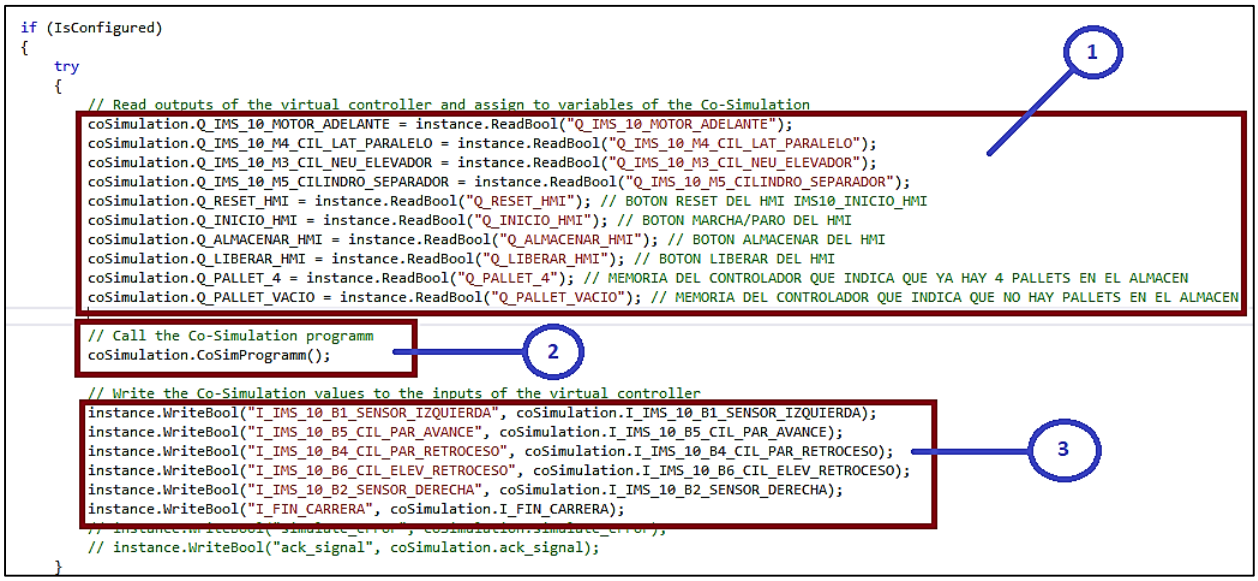


Figura 3.43 Intercambio de datos, clase "PLCInstance.cs"

3.2.3.3 Simulación del programa

1. En la carpeta "Models", se abre la clase "Cosimulation.cs"
2. Para el intercambio de datos con el controlador virtual, se definen las tags booleanas, en la declaración de clase. (Figura 3.44 y 3.45)

```
public class Cosimulation
{
    public event EventHandler<PropertyArgs> OnOperatingStateChanged;
    public event EventHandler<PropertyArgs> OnErrorSimulationStateChanged;

    //--// Inputs (Outputs of the virtual controller)
    public bool Q_IMS_10_MOTOR_ADELANTE; // ACTUADOR
    // public bool QS_IMS_10_MOTOR_LENTO;
    public bool Q_IMS_10_M4_CIL_LAT_PARALELO; // ACTUADOR
    public bool Q_IMS_10_M3_CIL_NEU_ELEVADOR; // ACTUADOR
    public bool Q_IMS_10_M5_CILINDRO_SEPARADOR; // ACTUADOR
    public bool Q_RESET_HMI; // BOTON RESET DEL HMI
    public bool Q_INICIO_HMI; // BOTON MARCHA/PARO DEL HMI
    public bool Q_ALMACENAR_HMI; // BOTON ALMACENAR DEL HMI
    public bool Q_LIBERAR_HMI; // BOTON LIBERAR DEL HMI
    public bool Q_PALLET_4; // MEMORIA DEL CONTROLADOR QUE INDICA QUE YA HAY 4 PALLETS EN EL ALMACEN
    public bool Q_PALLET_VACIO; // MEMORIA DEL CONTROLADOR QUE INDICA QUE NO HAY PALLETS EN EL ALMACEN
    // public bool acknowledgeActive;
    //--//

    //--// Outputs (Inputs of the virtual controller)
    // SENSORES DEL PROCESO
    public bool I_IMS_10_B1_SENSOR_IZQUIERDA = false;
    public bool I_IMS_10_B5_CIL_PAR_AVANCE = false;
    public bool I_IMS_10_B4_CIL_PAR_RETROCESO = false;
    public bool I_IMS_10_B6_CIL_ELEV_RETROCESO = false;
    public bool I_IMS_10_B2_SENSOR_DERECHA = false;
    public bool I_FIN_CARRERA = false;
    public bool simulate_error = false;
    public bool ack_signal = false;
```

Figura 3.44 Simulación del programa, tags booleanas

```
//--// Internal variables
private static bool run;
private static int step = 0;
private bool error;
private bool packageOk;
private static bool acknReady;
private bool nextStep;

private Timer movementTimer;
private Timer sensorTimer;

private static bool startedTimer;
```

Figura 3.45 Simulación del programa, tags booleanas de error

3. En el método “CoSimProgramm()” del bloque #region “Cosimulation”, los estados simulados del sistema, son programados en una instrucción switch-case. (Figura 3.46)

```
public void CoSimProgramm()
{
    if (Q_RESET_HMI) // Restart command from the virtual controller
    {
        step = 0; // Set start step
        startedTimer = false; // Reset indicator for started timer
        nextStep = false;
        OnErrorSimulationStateChanged(this, new PropertyArgs("Black"));  // Reset "SIMULATE ERROR" button collor
    }

    if (run) // Active when "START" button pressed for Co-Simulation
    {
        OnOperatingStateChanged(this, new PropertyArgs("ACTIVE")); // Shows "ACTIVE" state of the Co-Simulation

        // Reset all sensors at every call (sensors are set in the corresponding case)
       I_IMS_10_B1_SENSOR_IZQUIERDA = false;
        I_IMS_10_B5_CIL_PAR_AVANCE = false;
        I_IMS_10_B4_CIL_PAR_RETROCESO = false;
        I_IMS_10_B6_CIL_ELEV_RETROCESO = false;
        I_IMS_10_B2_SENSOR_DERECHA = false;
        I_FIN_CARRERA = false;
        simulate_error = false;
        ack_signal = false;

 switch (step)
 {
     case 0: // ES UN ESTADO DE ESPERA
     // sensorStartPos = true; // Sensor active

         if (!startedTimer) // Starts once the simulation time for the sensor
         {
             movementTimer.Start(); // Start timer for movement simulation
             startedTimer = true; // Set indicator for started timer
         }
         if (Q_INICIO_HMI & nextStep)// PROGRAMA INICIADO CON MARCHA/PARO DEL HMI
         {
             step = 1; // Set number of the next step
             nextStep = false; // Reset next step variable
             startedTimer = false; // Reset indicator for started timer
         }
         break;
```

Figura 3.46 Co-simulación del programa, estados simulados del sistema

Y de esta manera se continúa con los demás “Steps” detallados en las tablas booleanas (Figura 3.47)

```
case 1: // ESTADO DE PROGRAMA INICIADO

    if (!startedTimer) // Starts once the simulation time for the movement
    {
        movementTimer.Start(); // Start timer for movement simulation
        startedTimer = true; // Set indicator for started timer
    }
    if (Q_ALMACENAR_HMI & nextStep)// ESTADO DE PROCESO DE ALMACENADO INICIADO CON ALMACENAR DEL HMI, NO PREGUNTO POR LA CANTIADA DE PALLETS PORQUE
    {
        step = 2; // SE VA AL PROCESO DE ALMACENADO Set number of the next step
        nextStep = false; // Reset next step variable
        startedTimer = false; // Reset indicator for started timer
    }
    if (Q_LIBERAR_HMI & nextStep)// ESTADO DE PROCESO DE LIBERAR INICIADO CON LIBERAR DEL HMI, NO PREGUNTO POR LA CANTIADA DE PALLETS PORQUE ESO YA
    {
        step = 8; // SE VA AL PROCESO DE LIBERADO
        nextStep = false; // Reset next step variable
        startedTimer = false; // Reset indicator for started timer
    }
    break;
```

Figura 3.47 Co-simulación del programa, Caso 1

3.3 Comunicación y Carga del programa

Como este proyecto se basó en utilizar el programa S7-PLCSIM Advanced, se tuvo una ventaja para simular, debido a que el mismo se lo puede usar como sistema de simulación independiente, al mismo tiempo al usar el PLC S7-1500, dentro del sistema de control, se le da al usuario mayor confiabilidad y precisión, dado que hay la posibilidad de utilizar equipos y softwares actualizados y de uso industrial, para que tanto los estudiantes, como ingenieros en entrenamiento, puedan manipular recursos prácticos del entorno de acción que encontraremos en el ejercicio profesional.

Para esto, vamos a seguir el procedimiento abajo descrito

3.3.1 Integración de librería API

Para poder empezar la ejecución de simulación descrita en este proyecto, es necesario ejecutar inicialmente el programa de co-simulación realizado en Visual Studio y luego el programa del PLC, abriendo la aplicación TIA Portal.

La librería utilizada para la ejecución de este proyecto es la denominada: “SIEMENS.Simatic.Simulation.Runtime.Api.x64”, esto debido a que nuestro equipo maneja un sistema operativo con arquitectura de 64 bits. Se debe eliminar la referencia existente de esta biblioteca e integrarla nuevamente, para que en el proyecto se actualice la ruta.

Para esto, se creó una referencia en el proyecto para la biblioteca S7-PLCSIM Advanced (Figura 3-48), misma que se encuentra en el directorio "… \ Archivos de programa (x86) \ Archivos comunes \ SIEMENS \ PLCSIMADV \ API \ APIC".

Además, se debe establecer la propiedad " Copy Local " de la referencia en "True".

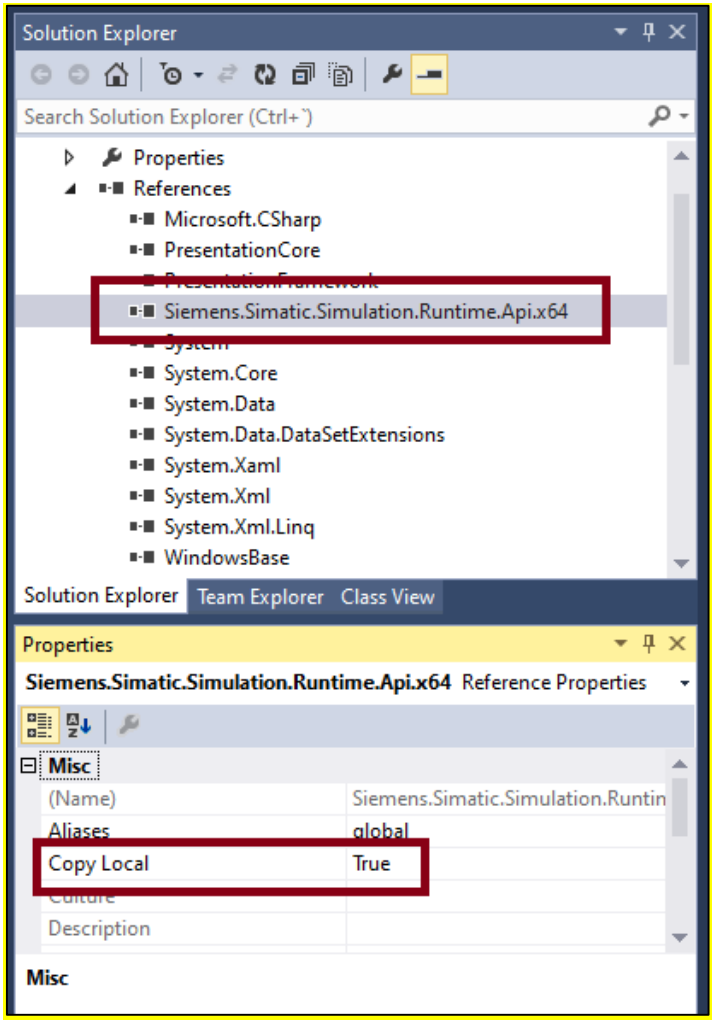


Figura 3.48 Integración de librería API

3.3.2 Interfaz con el usuario de Visual

Una vez, configurado lo descrito en el punto anterior se procede a dar inicia a la ejecución del proyecto, se obtiene la interfaz gráfica de usuario de la aplicación externa (Visual Studio) (Figura 3-49), con los siguientes comandos:

1. Muestra el estado de la comunicación con la Instancia PLC (Programa STEP7)
2. Muestra el estado de la interacción de la co-simulación (Visual Studio)

3. Botones para interactuar con el controlador virtual (Instancia PLC), utilizando la programación realizada en Visual Studio.
4. Botones para interactuar con la co-simulación, utilizando la programación realizada en Visual Studio.
5. Muestra la información del estado general de la co-simulación y mensajes de error.

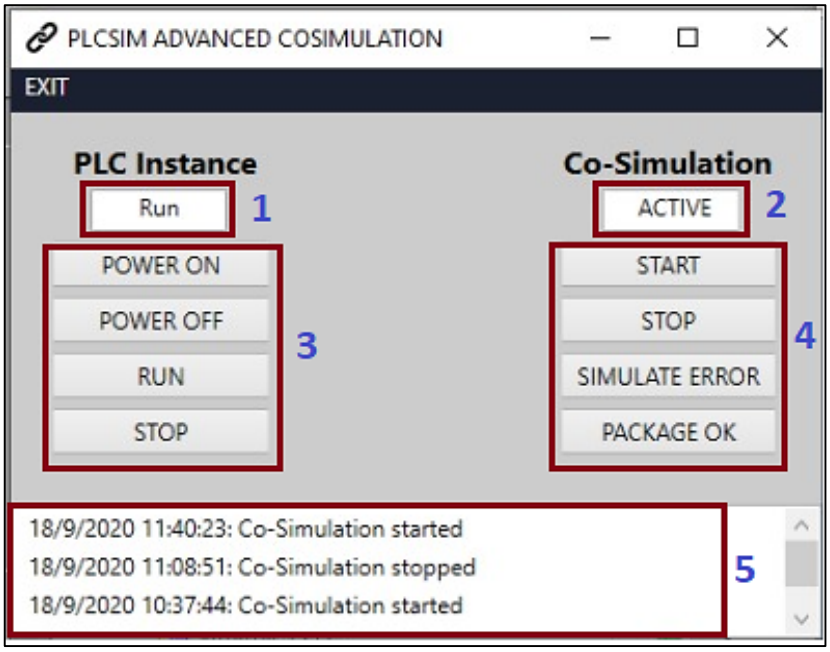


Figura 3.49 Interfaz gráfica de la Co-simulación (Visual Studio)

Cuando existe alguna falla debido a la comunicación con el controlador virtual ejecutado en el TIA Portal se visualizará en el cajón (5).

3.3.3 Ejecución de la aplicación S7-PLCSIM Advanced

Para ejecutar el controlador virtual (PLC simulado), se abre el programa S7-PLCSIM Advanced (Figura 3-50), y para que se pueda realizar la comunicación con la co-simulación, se da inicio a la instancia “TransportControl”, la misma que fue declarada en el programa de C#.

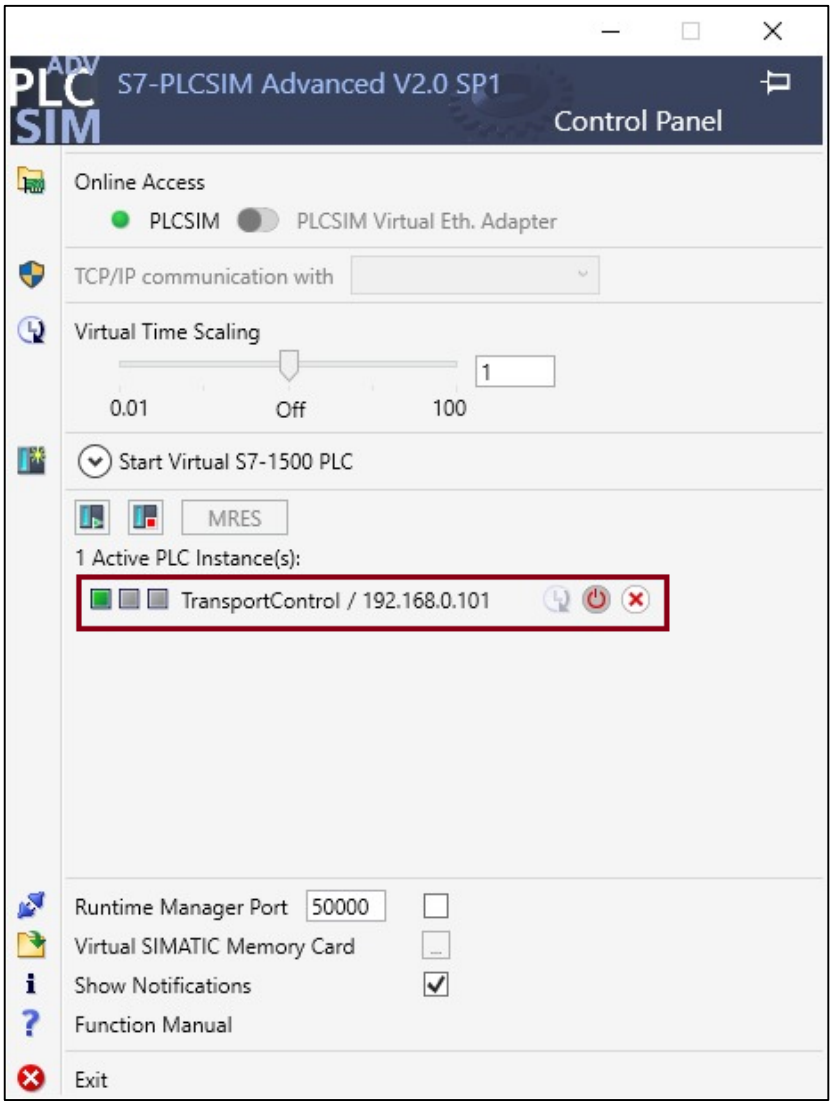


Figura 3.50 Panel de control del PLCSIM Advanced

Una vez ejecutada la instancia “TransportControl”; se presiona “POWER ON” en la interfaz gráfica del Visual Studio (Figura 3-51) que da como resultado que, en el cajón (1) se visualice la palabra “RUN”, y en el cajón (5) “Power On Instance: TransportControl”, que muestra la acción de haber cargado la instancia antes mencionada en el programa S7-PLCSIM Advanced. En el cajón (2) se visualizará el mensaje “STOP”, el mismo que cambiará a “ACTIVE” una vez ejecutado el programa del PLC, cuando se establece la “conexión online”, a través del programa TIA Portal.

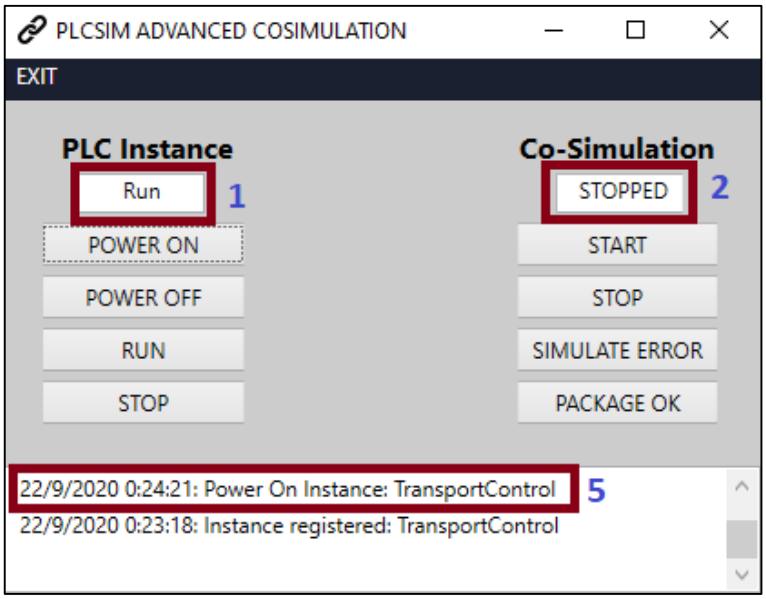


Figura 3.51 Interfaz de co-simulación, POWER ON

Cuando existe alguna falla debido a la comunicación con el controlador virtual ejecutado en el TIA PORTAL se visualizará en el cajón (5).

3.3.4 Carga del proyecto del TIA PORTAL en el controlador virtual

Una vez abierto el Proyecto en TIA PORTAL, se selecciona el PLC en la navegación del proyecto (Figura 3-52), para desplegar el menú "Online" y seleccionar "Carga avanzada en dispositivo...".

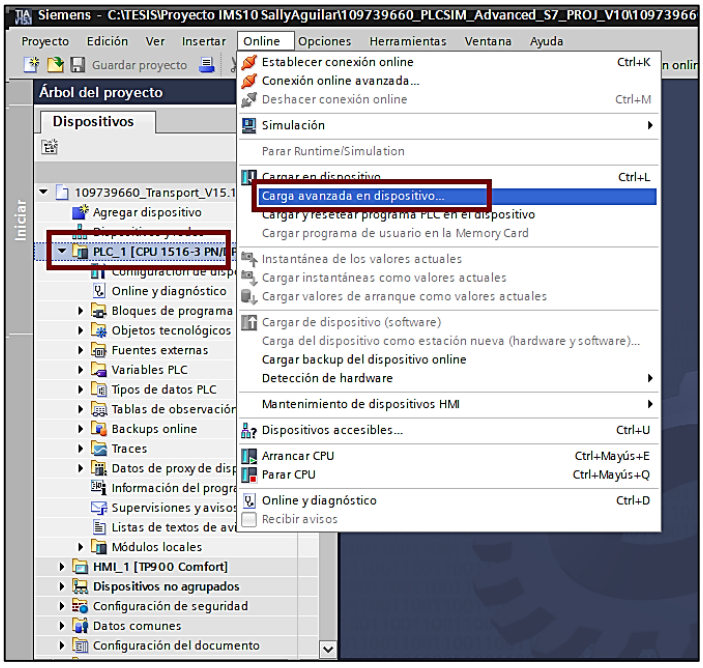


Figura 3.52 Carga del proyecto del TIA PORTAL en el controlador virtual

Luego, se configura “PLCSIM” como “Interfaz PG / PC”, para poder “Iniciar búsqueda”, se selecciona el simulador “CPU”, corroborando que dirección IP sea la misma que inicialmente se configuró en el CPU S7-1500 escogido, y se envía a “Cargar” (Figura 3-53).

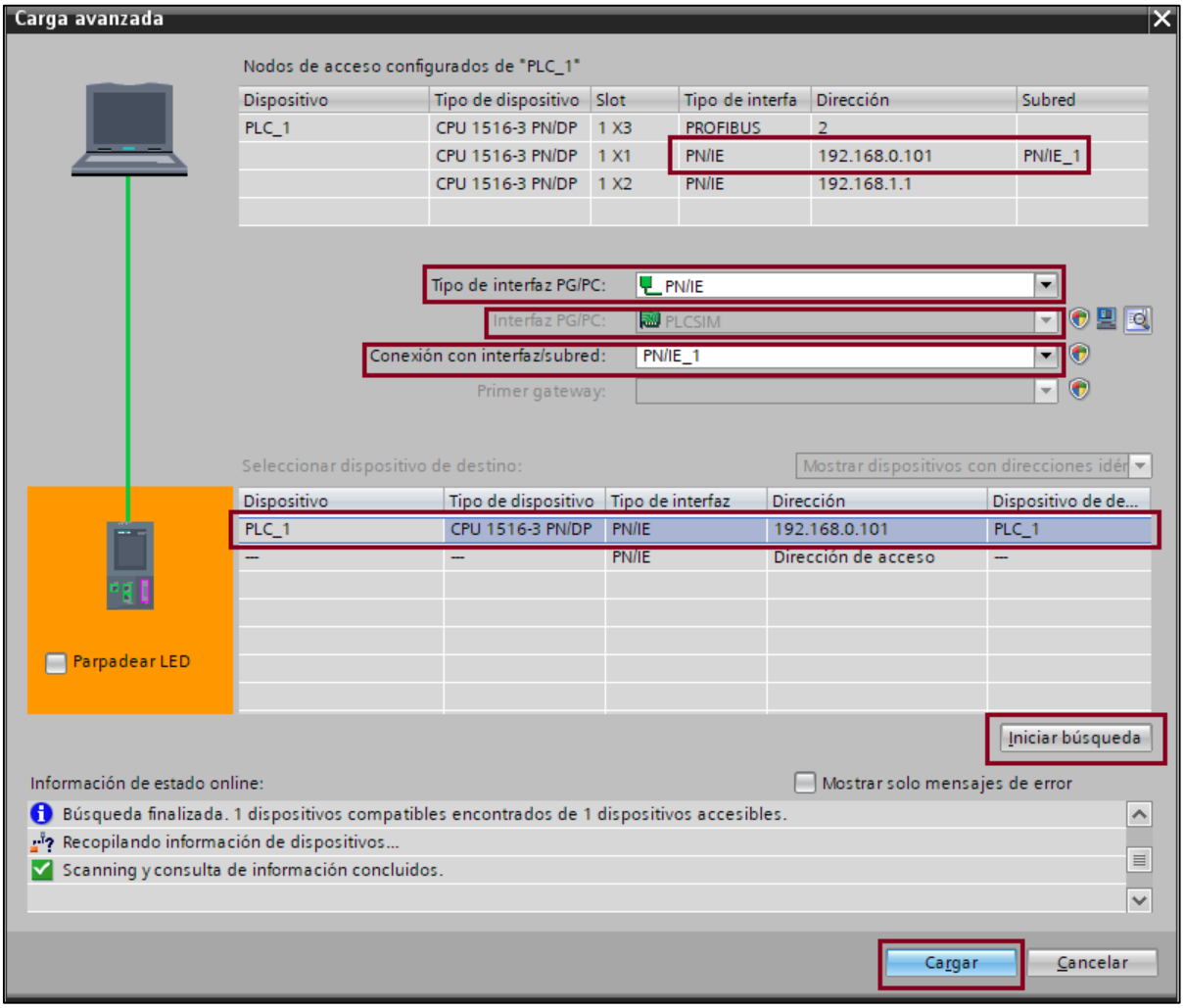


Figura 3.53 Conexión, Carga avanzada

Luego de realizar la carga respectiva de todos los módulos del programa, para comprobar la conexión, se activa la opción **“Establecer conexión online”** (Figura 3-54), y se confirma, mediante el indicador “●” la conectividad de todas las funciones de **“Bloques de programa”**.

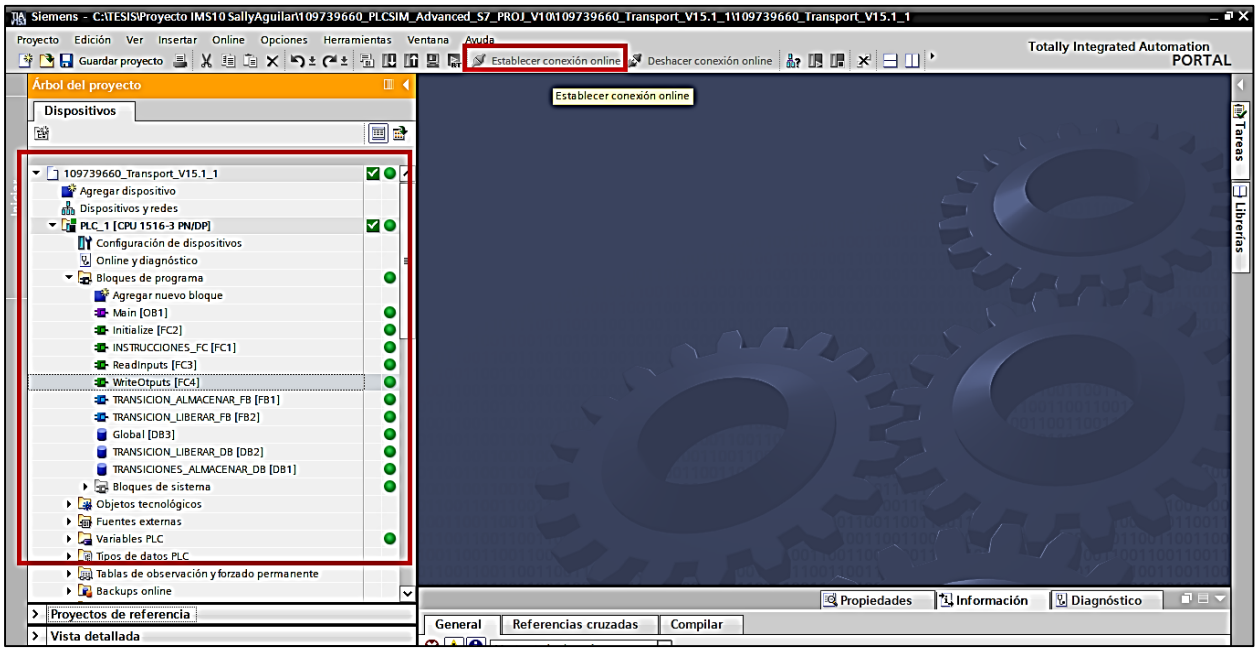


Figura 3.54 Verificación de conexión online

Y a la vez, se puede comprobar mediante el mensaje “ACTIVE” y “Co-Simulation started” (Figura 3-55), que se ha establecido la comunicación entre la programación del API, de la planta simulada en Visual Studio, y el controlador virtual.

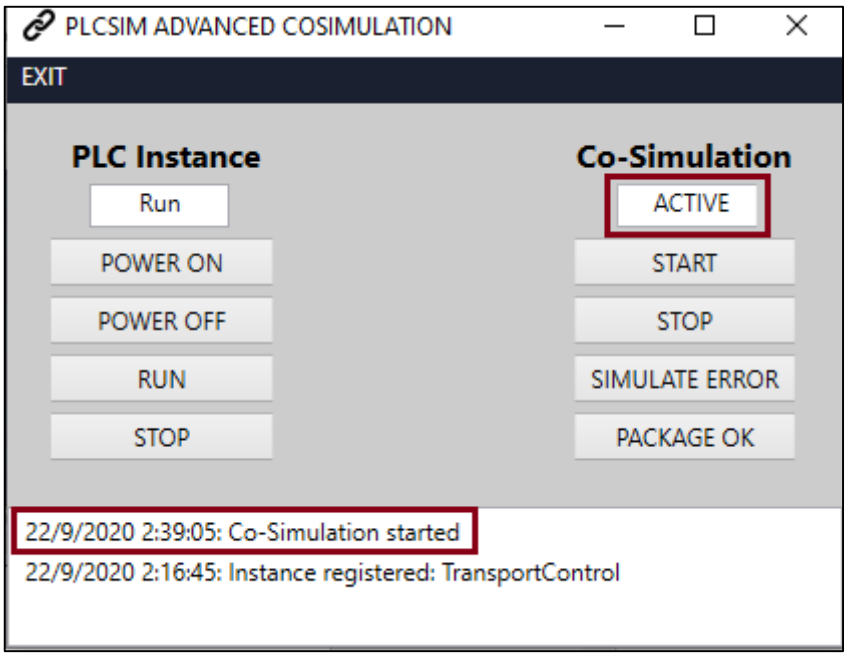


Figura 3.55 Verificación de conexión online, Co-simulación

3.4 Simulación del HMI en WinCC

Por último, para poder visualizar la animación del funcionamiento del proyecto descrito en el presente documento, se ejecuta la interfaz gráfica HMI, mediante la ejecución del software WinCC RT Start.

Para realizar esta ejecución, dentro del Proyecto en TIA PORTAL, se selecciona en la navegación del proyecto el HMI, previamente programado, para proceder a compilarlo y posteriormente a simularlo, tal como se muestra en la Figura 3-56.

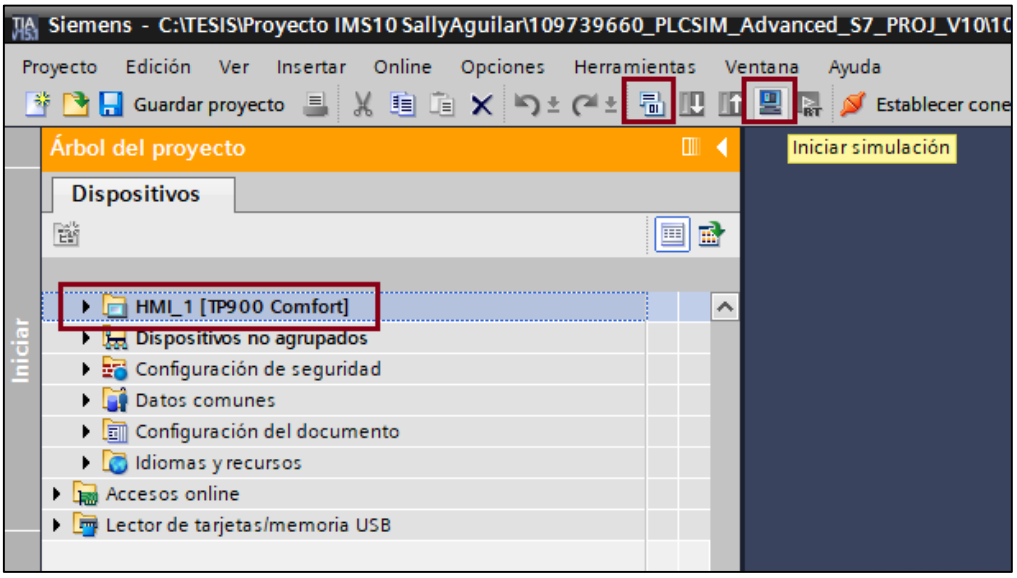


Figura 3.56 Puesta en marcha de la simulación desde S7 TIA PORTAL

Se puede apreciar que, una vez iniciada la simulación desde el HMI, emergerá la pantalla del HMI (Figura 3-57); con nuestra ventana inicial, donde podremos ver la planta simulada, así como las opciones iniciales, además del campo (en 0), de los pallets que se encuentran ya almacenados, en el campo “PALLETS PRESENTES”.

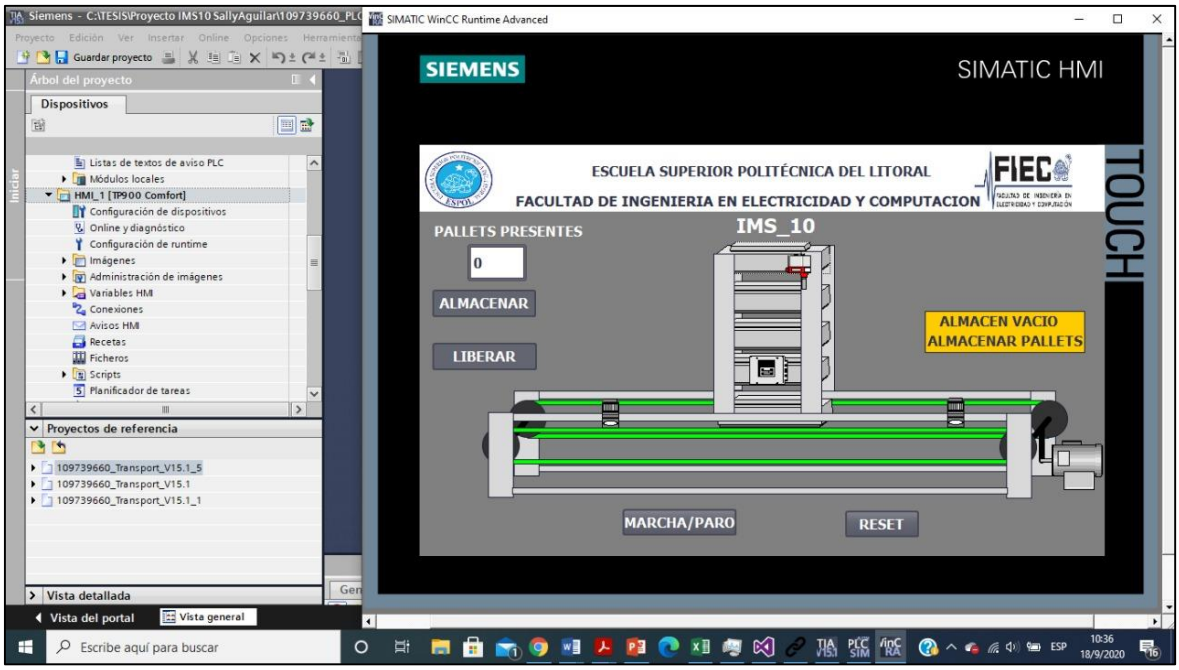


Figura 3.57 Pantalla inicial de simulación del HMI TP900

Tal como se indicó, en el diagrama de flujo, el primer paso luego del encendido del programa, será dar clic en el botón “MARCHA/PARO”, emergiendo la notificación, en celeste, “HA INICIADO EL PROGRAMA”, inicia la animación (Figura 3-58).

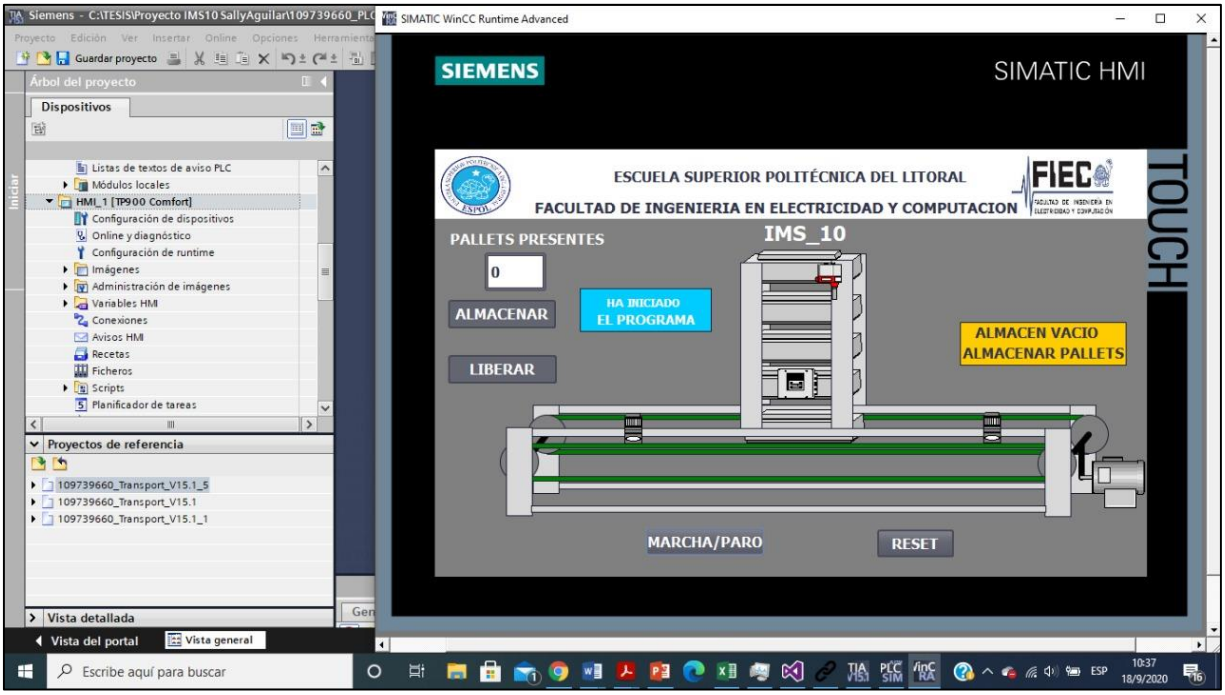


Figura 3.58 Simulación del HMI TP900

Lo siguiente será la secuencia de transiciones entre pasos y condiciones cumplidas, con su respectiva animación en la simulación, resaltándose los actuadores y sensores que intervienen (Figura 3-59.)

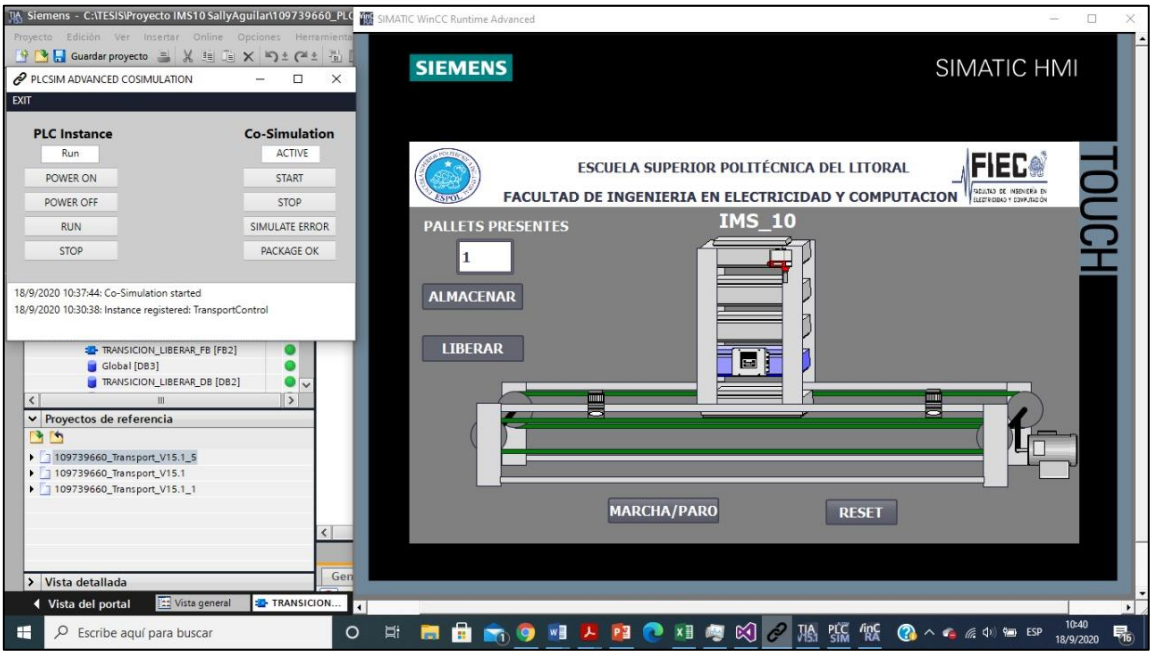


Figura 3.59 Simulación del HMI TP900 , primer pallet almacenado

En la secuencia almacenar, se contempla se alcance únicamente el almacenamiento de los 4 pallets, para llenar el almacén (Figura 3-60).

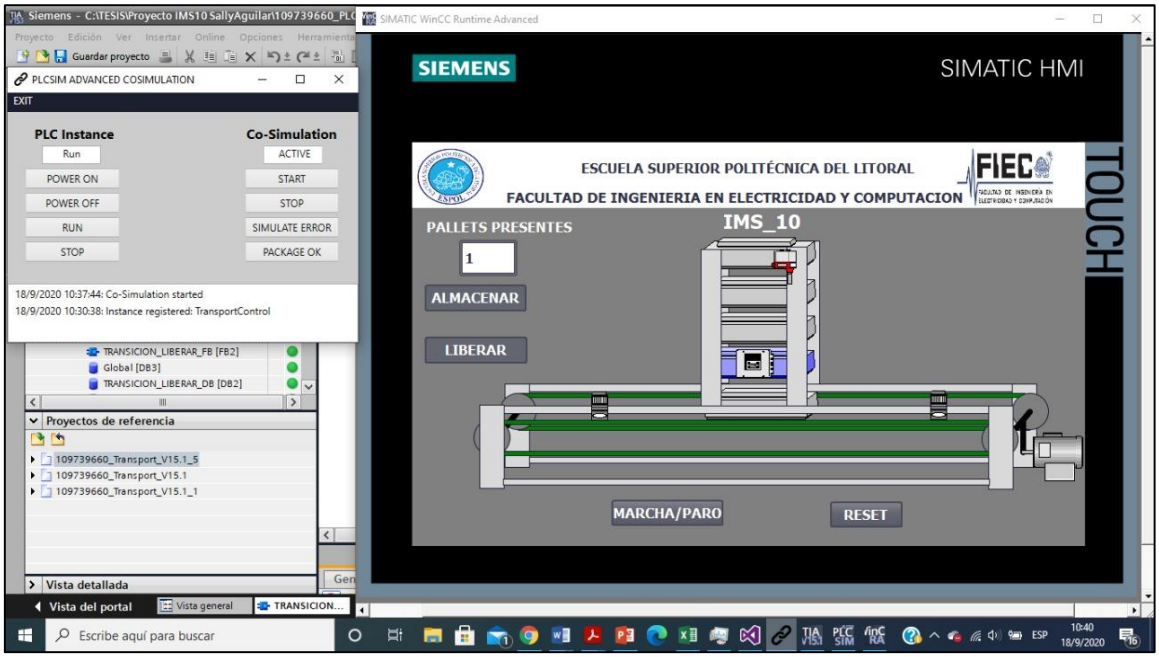


Figura 3.60 Simulación del HMI TP900, almacén completo

3.5 Análisis Económico del Sistema

Respecto al presupuesto requerido para la implementación del proyecto, es necesario, realizar la comparación, entre una implementación con elementos de hardware, frente a una solución, completamente virtual, como se propone en este documento. Véase detalles en la Tabla 3-7.

Tabla 3-7 Análisis económico, sistema co-simulado

Análisis Económico del Sistema Co-simulado					
ITEM	DESCRIPCION	Código	CANT	PVP UNIT	PVP TOTAL
1	STEP 7 Professional V14	6ES7822-1AE04-0YA5	1	$ 4.483,00	$ 4.483,00
2	S7-PLCSIM Advanced V1.0	6ES7823-1FE00-0YA5			
3	WinCC Advanced V14	6AV2102-0AA04-0AH5	1	$ 4.238,00	$ 4.238,00
4	Microsoft Visual Studio Professional (registro válido por 1 año)		1	$ 540,00	$ 540,00
				SUBTOTAL	$ 9.261,00
				IVA	$ 1.111,32
				TOTAL	$ 10.372,32

Tabla 3-8 Análisis económico, Presupuesto por implementación física del proyecto propuesto

Análisis Económico del Sistema					
ITEM	DESCRIPCION	Código	CANT	PVP UNIT	PVP TOTAL
1	CPU 1516-3 PN/DP V2.0 	6ES7516-3AN01-0AB0	1	$ 3.842,00	$ 3.842,00
2	TP900 Comfort V14.0.0.0 	6AV2124-0JC01-0AX0	1	$ 4.745,00	$ 4.745,00
3	Tablero de control (incluye componentes)	GLOBAL	1	$ 3.500,00	$ 3.500,00
4	Instalación, mano de obra, diseño y arquitectura. Administración y dirección técnica.	GLOBAL	1	$ 1.700,00	$ 1.700,00
5	STEP 7 Professional V14	6ES7822-1AE04-0YA5	1	$ 4.483,00	$ 4.483,00
6	S7-PLCSIM Advanced V1.0	6ES7823-1FE00-0YA5			
7	WinCC Advanced V14	6AV2102-0AA04-0AH5	1	$ 4.238,00	$ 4.238,00
				SUBTOTAL	$ 22.508,00
				IVA	$ 2.700,96
				TOTAL	$ 25.208,96

La tabla 3-8, muestra la misma solución, pero recurriendo al uso de hardware, lo que representa, además de los dispositivos, incurrir en gastos adicionales por paneles, diseño y mano de obra. Todo lo anterior eleva considerablemente el presupuesto final.

Es preciso tomar en cuenta que el software es prácticamente el mismo entre ambas opciones, con la diferencia que, en la solución con hardware, no se implementa un entorno de co-simulación, por lo que se obviaría el Microsoft Visual Studio.

Existe la posibilidad de optar por licencias académicas, en cuanto al software, esto representaría una reducción considerable en los costos, y ventajas en cuanto a la capacidad de computadores en los que se puede implementar

CAPÍTULO 4

4 CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

El poder realizar una co-simulación entre una planta de cualquier proceso industrial y un controlador virtual, sea el caso que se vaya a utilizar con fines académicos, o como entorno de pruebas de programación a nivel industrial, permite al usuario acceder a recursos prácticos en el campo de la automatización. Es aquí donde es indispensable hacer uso de elementos donde se desarrollen diferentes escenarios, lo más parecidos al entorno de acción que encontraremos en el ejercicio profesional, que son un PLC y una planta, para así poder ejecutar distintas instrucciones, como errores y demás pruebas de ejecución.

El fin de este estudio fue erradicar la problemática de la falta de recursos, y falencias en el entrenamiento previo del especialista en automatización y control en el diseño de sistemas automatizados. Para esto se propuso, realizar una analogía entre el mundo real de la automatización en un sistema co-simulado.

En el desarrollo de la co-simulación, contamos un PLC, igual al que se utiliza en un ambiente físico de automatización, pero de forma virtual, y a través de un API, que es una interfaz de programación de aplicaciones, existe la comunicación con la planta modelada en el lenguaje de programación C# y ejecutada en Visual Studio, con los actuadores y sensores de la existente en el Laboratorio de Control de Procesos Industriales. Se recibieron las señales equivalentes a las obtenidas desde una planta real, por lo consiguiente se pudieron ejecutar las acciones o comandos indicados por el controlador virtual.

El resultado fue un sistema de automatización de la estación de almacenamiento intermedio IMS-10 de la planta Lucas-Nülle, con las condiciones necesarias para poder realizar el intercambio de datos entre el PLC SIEMENS S7-1500, mediante la ejecución en el programa SIMATIC S7-PLCSIM ADVANCED, y el API ejecutado en Visual Studio.

Se recreó un error, la caída del pallet, quedando abierta la posibilidad de crear gran cantidad de otros eventos adversos, sin que esto recurra en el riesgo de daños en recursos de uso común del estudiantado.

También, al utilizar como software principal de esta aplicación al S7-PLCSIM ADVANCED, y habiendo entregado la programación completa, se pueden aprovechar todas sus demás prestaciones. Para futuras aplicaciones, se podría interactuar con distintos usuarios, de forma remota, incluso la posibilidad de un equipo remoto, y escalar, además, a otros niveles de comunicación, en combinación con software de terceros, quedando la posibilidad de integrarlo con hardware en una solución física.

En conclusión, en este trabajo se desarrolló una solución práctica para distintos usuarios, basándose en co-simulación. Permitirá diseñar una extensa variedad de escenarios de prueba, sin comprometer la integridad de dispositivos con los que se cuentan en un laboratorio o planta industrial, sin realizar paras en el proceso para las pruebas de funcionamiento y sin la necesidad de que se encuentren físicamente en el sitio, ya que todo este ambiente de pruebas se lo puede realizar dentro de una computadora personal.

4.2 Recomendaciones

Considerando que el enfoque de este proyecto fue resolver la problemática de un Laboratorio de Control de Procesos, se sugiere optar por la adquisición de la alternativa estudiantil del software necesario. En el sector industrial en cambio, el concepto de licencias múltiples, disponible en el catálogo de SIEMENS, reduciría de forma considerable la inversión en este rubro.

Por lo acontecido este año, la posibilidad de una educación (o trabajo), virtual, implementar esta solución con la modalidad de acceso remoto, permitiría, entre otras cosas, el desarrollo de prácticas de laboratorio. El acceso mediante credenciales de seguridad, sería recomendable, a fin de autenticar usuarios / estudiantes.

Finalmente, una recomendación tan válida como las anteriores, sería realizar siempre una programación estructurada organizada en el STEP 7, aparte de facilitar búsquedas entre los diferentes componentes del proyecto, permitirá llamar a las salidas, de forma secuencial en la co-simulación.

5 REFERENCIAS

[1] G.-L. Jorge, F. Chavez Montejano, A. Mendez y A. Hernández, «Sistema de CoSimulación de un Robot Industrial para Control,» *Revista de la Aplicación Científica y Técnica ,* 2016.

[2] Modelica Association Project “FMI”, Functional Mock-up Interface for Model Exchange and Co-Simulation, 2019.

[3] Famic Technologies Inc., «https://www.famictech.com/,» Famic Technologies, 2020. [En línea]. Available: https://www.famictech.com/en/famic-technologies-releases-version-63-of-its-machine-design-and-simulation-software-automation-studiosuptmsup4.

[4] Siemens, SIMATIC S7-PLCSIM Advanced: Co-Simulation via API, 2016.

[5] Siemens, SIMATIC, S7-1500, S7 PLCSIM Advanced Function Manual, NÜRNBERG: Siemens AG, 2016.

[6] Y. Cerezo López, O. Peñalba Rodríguez y R. Caballero Roldán, INICIACIÓN A LA PROGRAMACIÓN EN C# UN ENFOQUE PRÁCTICO, Madrid: Delta, Publicaciones Universitarias, 2007.

[7] T. Dimes, Programación en C# para Principiantes, Babelcube, 2016.

[8] F. J. C. Sierra, Microsoft C#™ Curso de programación, Madrid: RA-MA Editorial, 2011.

[9] B.-A. Guérin, ASP.NET con C# en Visual Studio 2017, Diseño y desarrollo de aplicaciones web, Barcelona: ENI, 2018.

[10] Microsoft, «https://www.msn.com/,» Microsoft News, 30 04 2020. [En línea]. Available: https://www.msn.com/es-cl/noticias/microsoftstore/%C2%BFqu%C3%A9-es-y-para-qu%C3%A9-sirve-visual-studio-2017/ar-AAnLZL9. [Último acceso: 08 2020].

[11] R. Sanchis Llopis , J. A. Romero Pérez y C. V. Ariño Latorre, Automatización Industrial, Castellón de la Plana: Publicacions de la Universitat Jaume I. Servei de Comunicació i Publicacions, 2010.

[12] SIEMENS, Productos para Totally Integrated Automation, Nürnberg: Siemens AG, 2019.

[13] Lucas-Nüelle, «http://www.lucas-nuelle.us/,» CMS, Webdesign and Realization cekom GmbH, 2020. [En línea]. Available: http://www.lucas-nuelle.us/2769/pid/21523/apg/10836/Double-conveyor-belt-segment,-24V-motor.htm.

[14] Lucas-Nülle, «https://www.lucas-nuelle.us/,» CMS, Webdesign and Realization cekom GmbH, 2020. [En línea]. Available: https://www.lucas-nuelle.us/2769/pid/24235/apg/12280/Material-buffering-station.htm.

[15] ESPOL & FIEC, «ESPOL - FACULTAD DE INGENIERIA EN ELECTRICIDAD Y COMPUTACION,» ESPOL, 2020. [En línea]. Available: https://www.fiec.espol.edu.ec/es/lab-control-procesos. [Último acceso: 2020].

[16] Siemens AG, SIEMENS SIMATIC, Programar con STEP7, NÜRNBERG: Siemens AG Division Digital Factory, 2017.

[17] Siemens, Módulo TIA Portal 031-200 Principios básicos de la programación de FB con SIMATIC S7-1200, Nuremberg: Siemens AG, 2018.

[18] C. P. Duran Salazar y V. A. Ortega Paz, «Repositorio Dspace,» 6 Febrero 2018. [En línea]. Available: http://www.dspace.espol.edu.ec/xmlui/handle/123456789/42553. [Último acceso: Agosto 2020].

ANEXO 1

Segmentos de Función: INSTRUCCIONES [FC1]

Segmento 1: Contador de Pallets. - En esta función se creó un contador CONTROL_PALLET (%DB4), para guardar el valor entero de la cantidad de pallets que se encuentran en el almacén, de acuerdo al almacenado o liberado.

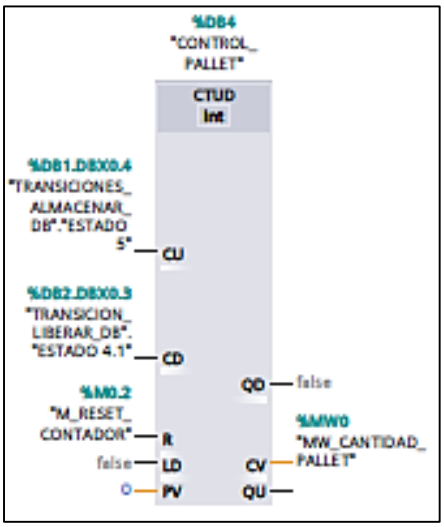


Fig. A1- 1 Segmento 1: Contador de Pallets

Segmento 2: Motor Encendido Hacia Adelante, Esclavo 8 Y 9.- en esta función solo avanza la banda hacia adelante.

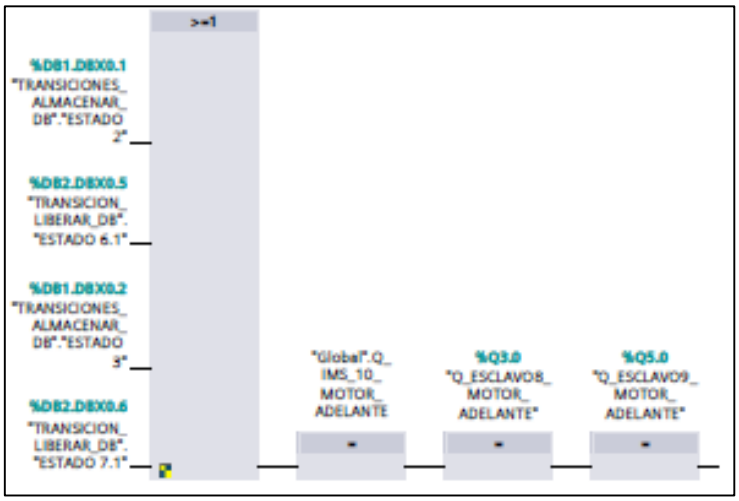


Fig. A1- 2 Segmento 2: Motor Encendido Hacia Adelante

Nota: Se procede a programar el Esclavo 8 y 9, ya que es parte del hardware original de la planta, pero estas salidas no se las va a tomar en cuenta en la programación de la co-simulación, dado que no se va a necesitar un medio adicional de comunicación al tener una planta virtual.

Segmento 3: Avance lento de la banda. - En esta función solo avanza la banda lentamente hacia adelante.

Fig. A1- 3 Segmento 3: Avance lento de la banda

Segmento 4: Cilindro Neumático para el Cilindro Elevador Central. - En esta función valida cuando sube o baja el cilindro elevador tanto para almacenamiento o liberación.

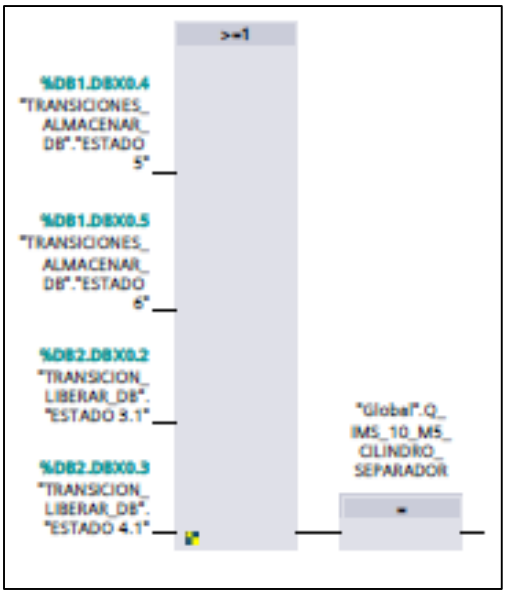


Fig. A1- 4 Segmento 4: Cilindro Neumático para el Cilindro Elevador Central

Segmento 5: M2_Cilindro Neumático para el Cilindro Medio de Parada. - En esta función se ordena se encienda el cilindro de parada en la transición almacenar.

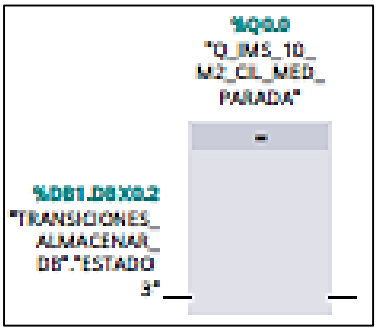


Fig. A1- 5 Segmento 5: M2_Cilindro Neumático para el Cilindro Medio de Parada

Segmento 6: Cilindro Neumático para el Cilindro Lateral (Paralelo). - En esta función se valida si los cilindros paralelos están activos o inactivos.

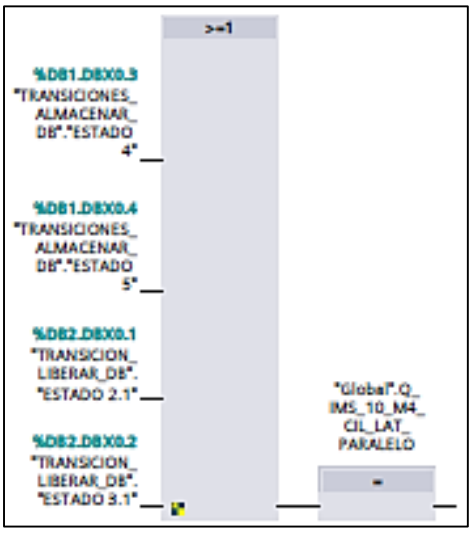


Fig. A1- 6 Segmento 6: Cilindro Neumático para el Cilindro Lateral (Paralelo)

Segmento 7: Motor Encendido Lentamente Esclavo 8.- Función para validar el motor en marcha lenta en la tarjeta con el esclavo 8.

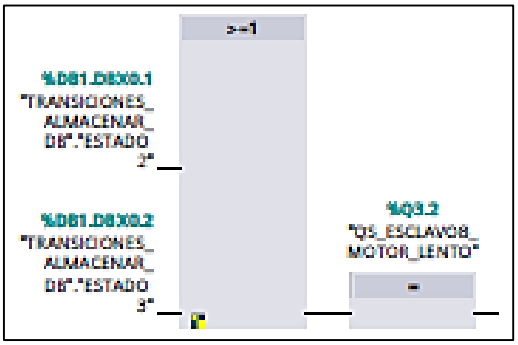


Fig. A1- 7 Segmento 7: Motor Encendido Lentamente Esclavo 8

Segmento 8: Resetea el Contador del Control de Pallets. - Validación de pulsar botón RESET para resetear el valor del contador.

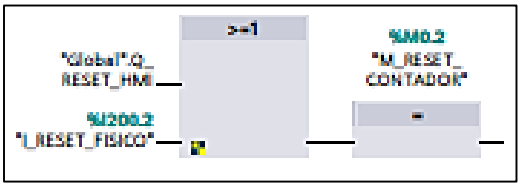


Fig. A1- 8 Segmento 8: Resetea el Contador del Control de Pallets

Segmento 9: Cilindro Neumático para el Cilindro Separador. - Validación para activar el cilindro de separación en el momento que almacena o libera los pallets.

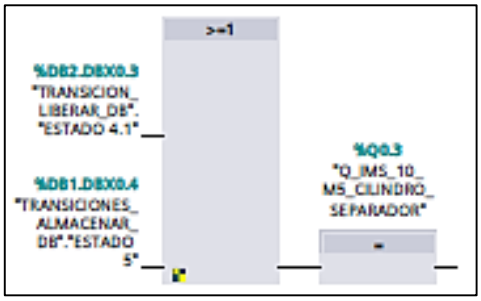


Fig. A1- 9 Segmento 9: Cilindro Neumático para el Cilindro Separador

Segmento 10: Transforma el Valor del Contador.-. Transforma el valor del controlador a un valor del tipo palabra (WORD), para mostrarlo en el HMI.

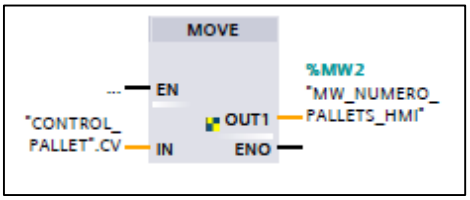


Fig. A1- 10 Segmento 10: Transforma el Valor del Contador

Segmento 11: Memoria mensaje Almacén Vacío en HMI.

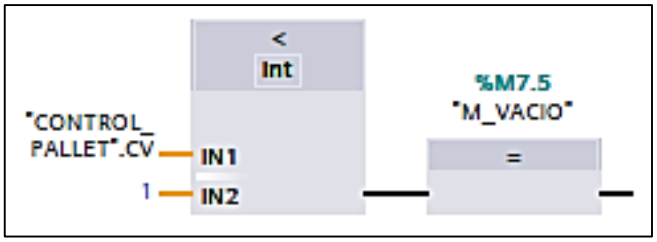
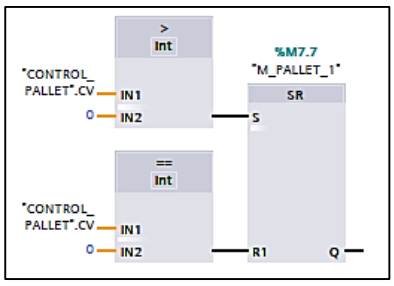
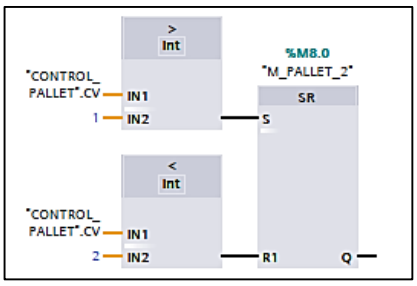


Fig. A1- 11 Segmento 11: Memoria mensaje Almacén Vacío en HMI

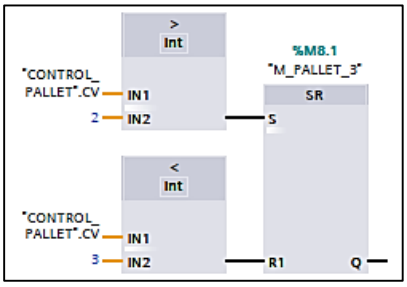
Segmentos del 12 al 15: Para indicar el número de pallets en el almacén.

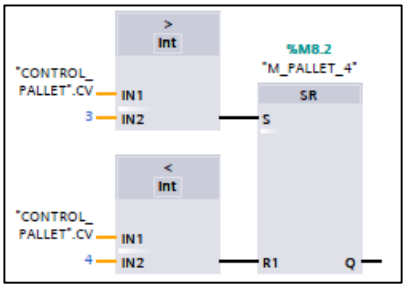


Segmento 12: Para indicar que hay 1 pallet



Segmento 13: Para indicar que hay 2 pallets





Segmento 14: Para indicar que hay 3 pallets	Segmento 15: Para indicar que hay 4 pallets

Fig. A1- 12 Segmentos 12, 13, 14, 15. Cantidades de pallets en el almacén

Segmento 16: Memoria para prender sensor izquierdo en HMI.

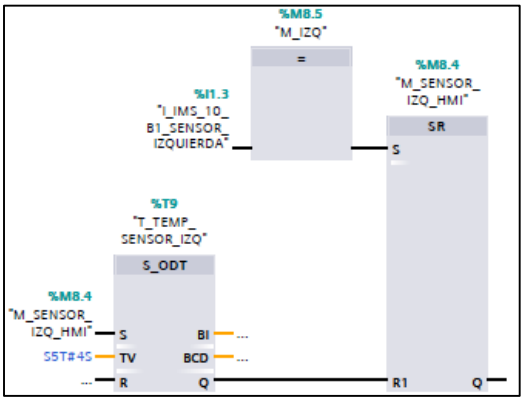


Fig. A1- 13 Segmento 16: Memoria para prender sensor izquierda en HMI.

Segmento 17: Memoria para prender sensor derecha en HMI.

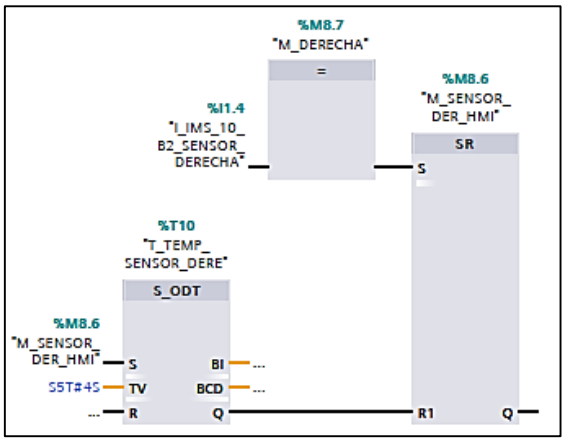


Fig. A1- 14 Segmento 17: Memoria para prender sensor derecha en HMI.

Segmento 18: Memoria para mostrar el cilindro de separación en el HMI.

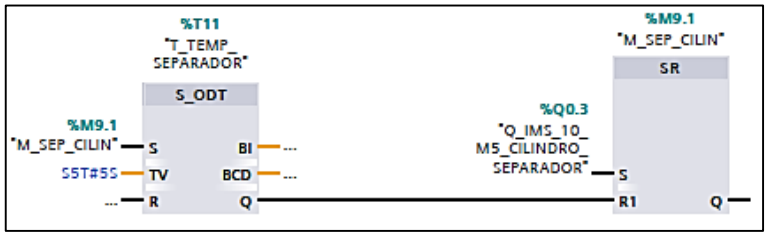


Fig. A1- 15 Segmento 18: Memoria para mostrar el cilindro de separación en el HMI

Segmento 19: Memoria para mostrar botón de marcha presionado.

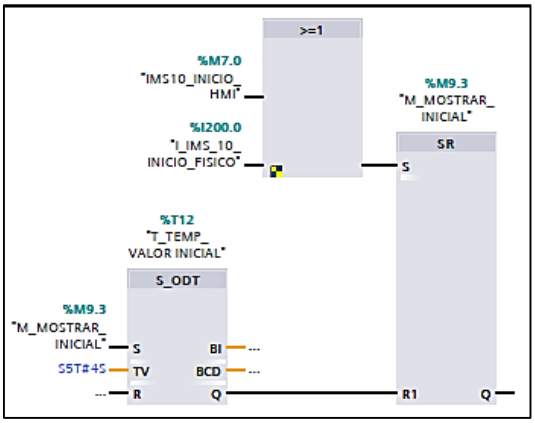


Fig. A1- 16 Segmento 19: Memoria para mostrar botón de marcha presionado.

Segmento 20: Memoria para guardar el valor de fin de carrera.

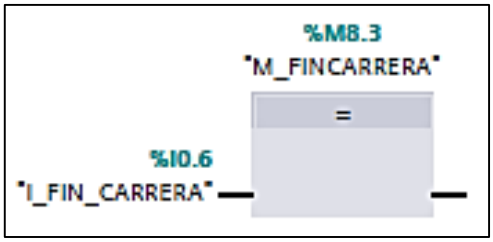


Fig. A1- 17 Segmento 20: Memoria para guardar el valor de fin de carrera.

Segmento 23: Memoria presentar mensaje de inicio.

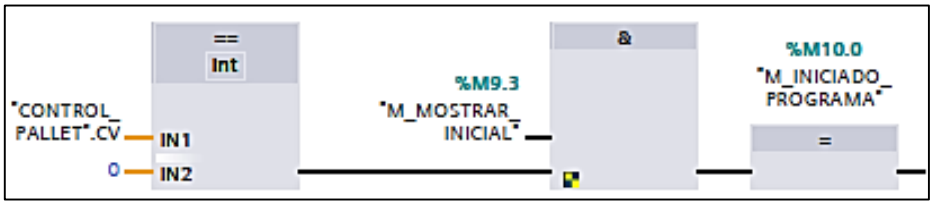


Fig. A1- 18 Segmento 23: Memoria presentar mensaje de inicio.

Segmento 25: Memoria para guardar el valor de la entrada del sensor izquierda.

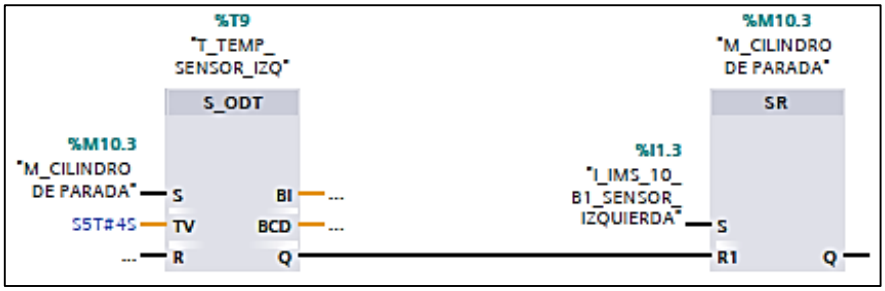


Fig. A1- 19 Segmento 25: Memoria para guardar el valor de entrada del sensor izquierda.

ANEXO 2

Segmentos del Bloque de Funciones:

TRANSICION_ALMACENAR [FB1]

Segmento 1: Transición del estado 8 al estado 1(Figura 3-10). - Se llega al estado 1 cumpliendo cualquiera de las siguientes condiciones:

1. Venir del estado 8
2. Presionar:
    - La botonera virtual en el HMI de Paro/Marcha: PULSAR IMS10_INICIO_HMI
    - La botonera física: I_IMS_10_INICIO_FISICO
    - La Botonera En La Co-simulación: m_error

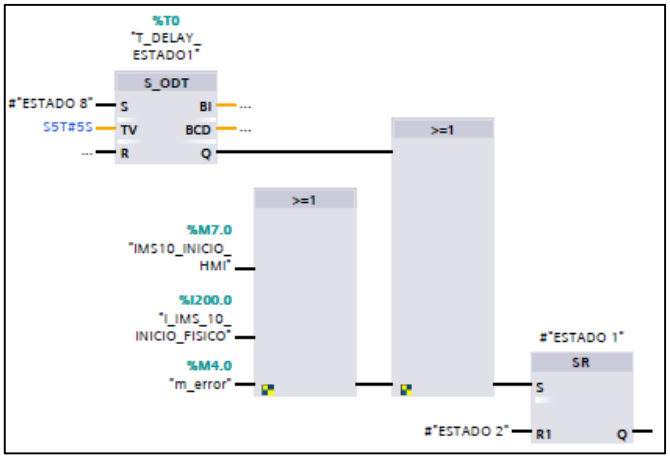


Fig. A2- 1 Transición del estado 8 al estado 1

ESTADO 1: Estado de espera de instrucción de almacenar o liberar.

Segmento 2: Transición del Estado 1 a Estado 2 (Figura3-11). - Se llega al estado 2 con las siguientes condiciones:

1. Al presionar cualquiera de estas opciones:

- La botonera virtual en el HMI de almacenar: M_IMS_10_ALMACENAR_HMI, o
- La botonera física: I_IMS_10_ALMACENAR_FISICO

2. Y se cumpla con la condición que:
   - No esté lleno el almacén de pallets: CONTROL_PALLET.CV < 4
   - Y que venga del estado 1 o 1.1

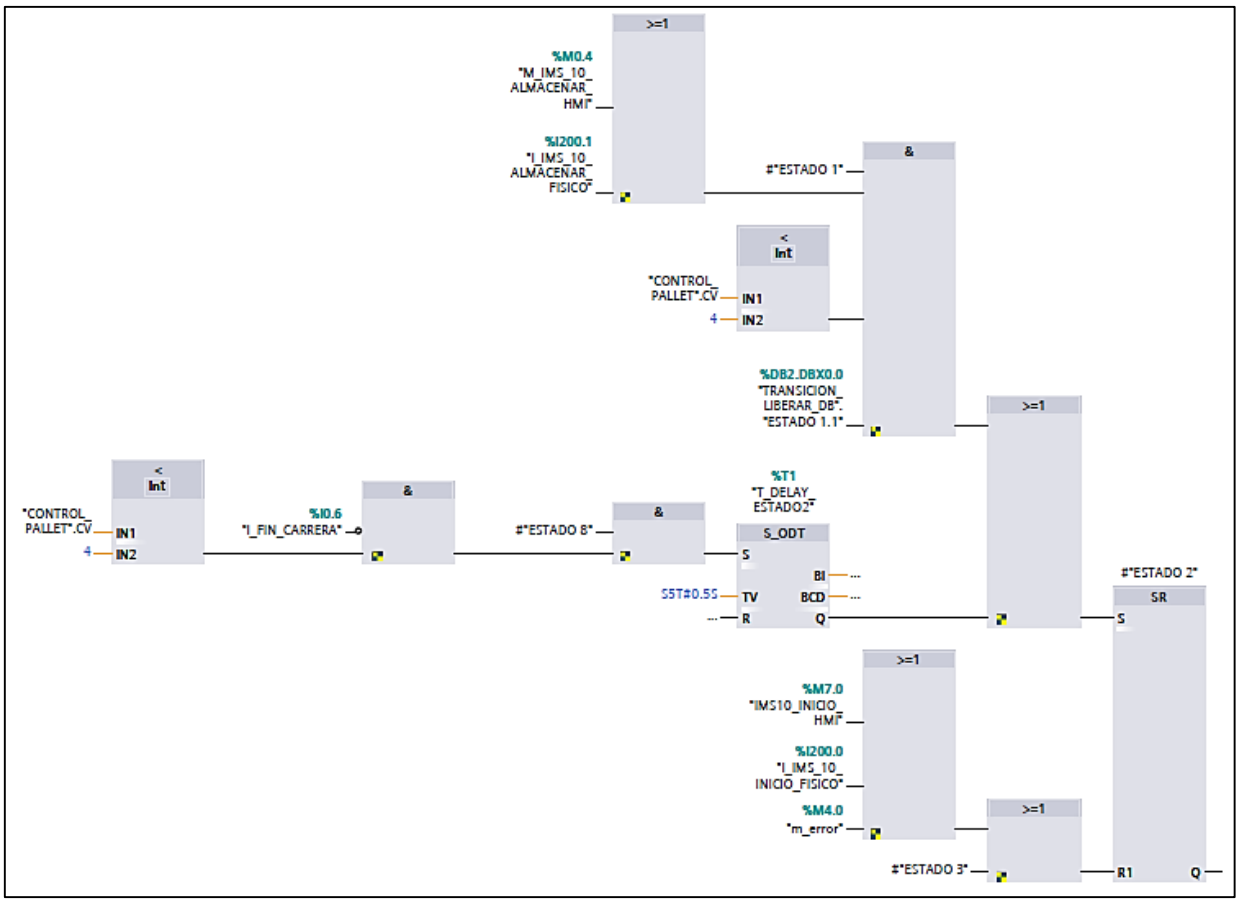


Fig. A2- 2 Transición del Estado 1 a Estado 2

ESTADO 2: Empieza el proceso de almacenar, realizando el transporte del pallet hacia la unidad de almacenamiento. Se activan las siguientes variables:

- "Global".Q_IMS_10_MOTOR_ADELANTE
- Q_ESCLAVO8_MOTOR_ADELANTE
- Q_ESCLAVO9_MOTOR_ADELANTE
- QS_ESCLAVO8_MOTOR_LENTO

Segmento 3: Transición del estado 2 al estado 3 (Figura 3-12). - Es el trayecto hacia la unidad de almacenamiento, en este estado verifica que haya pasado un pallet por el sensor de la izquierda. Se llega al estado 3, cumpliendo las condiciones:

1. Esté activo el sensor de posición final B1: I_IMS_10_B1_SENSOR_IZQUIERDA,
2. Que venga del estado 2

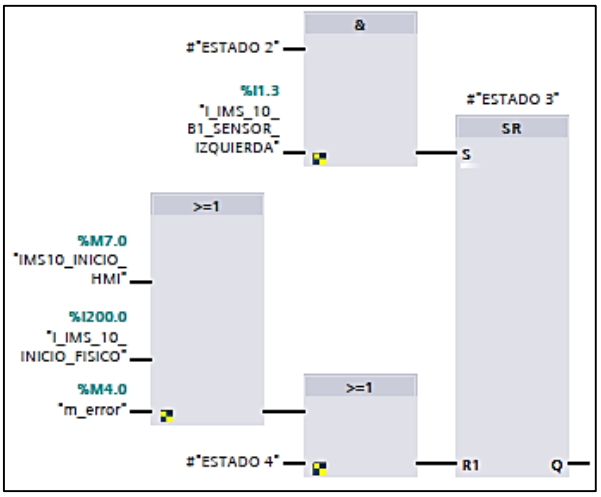


Fig. A2- 3 Transición del estado 2 al estado 3

ESTADO 3: El pallet alcanza la posición de almacenamiento, llega al almacén, se activan las siguientes variables:

- "Global". Q_IMS_10_MOTOR_ADELANTE
- Q_ESCLAVO8_MOTOR_ADELANTE
- Q_ESCLAVO9_MOTOR_ADELANTE
- QS_IMS_10_MOTOR_LENTO
- Q_IMS_10_M2_CIL_MED_PARADA
- QS_ESCLAVO8_MOTOR_LENTO

Segmento 4: Transición del estado 3 al estado 4 (SENSOR _IZQ) (Figura 3-13). - Se llega al estado 4 cumpliendo una pausa de seguridad de 6 segundos con las siguientes condiciones:

1. T_DELAY__ESTADO4, y
2. Que venga del estado 3

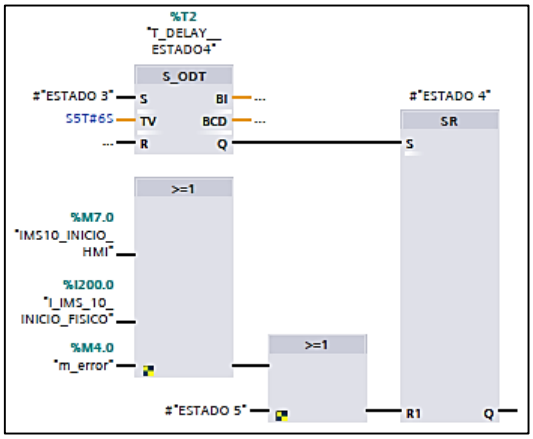


Fig. A2- 4 Transición del estado 3 al estado 4 (SENSOR _IZQ)

ESTADO 4: Para iniciar el proceso de elevación de pallets, se activan los cilindros paralelos:

- "Global".Q_IMS_10_M4_CIL_LAT_PARALELO

Segmento 5: Transición del estado 4 al estado 5 (Figura 3-14). - Se llega al estado 5 al completar el trayecto de los motores de los cilindros paralelos, activando el sensor en posición de avance con las siguientes condiciones:

1. I_IMS_10_B5_CIL_PAR_AVANCE, y
2. Que venga del estado 4

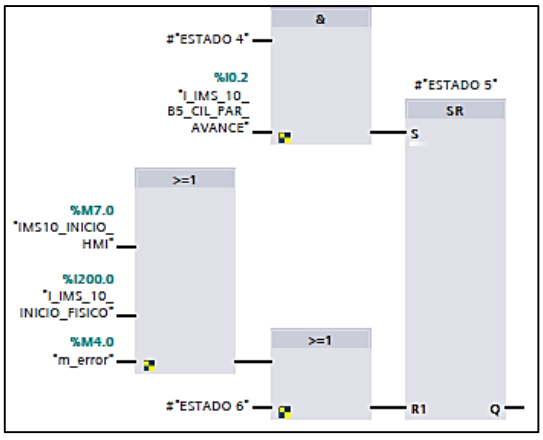


Fig. A2- 5 Transición del estado 4 al estado 5

ESTADO 5: Como parte del proceso de almacenamiento, se deben mantener activados los cilindros paralelos y cilindro de elevación, para permitir el ascenso del pallet, se activa el cilindro lateral, abriéndose las compuertas. Se activan las siguientes variables:

- "Global".Q_IMS_10_M3_CIL_NEU_ELEVADOR
- "Global".Q_IMS_10_M4_CIL_LAT_PARALELO
- Q_IMS_10_M5_CILINDRO_SEPARADOR
- Se realiza el conteo de pallets en el almacén con: CONTROL_PALLET

Segmento 6: Transición del estado 5 al estado 6 (Figura 3-15). - Se llega al estado 6 cumpliendo una pausa de seguridad de 6 segundos con las siguientes condiciones:

1. T_DELAY__ESTADO6
2. Y que venga del estado 5

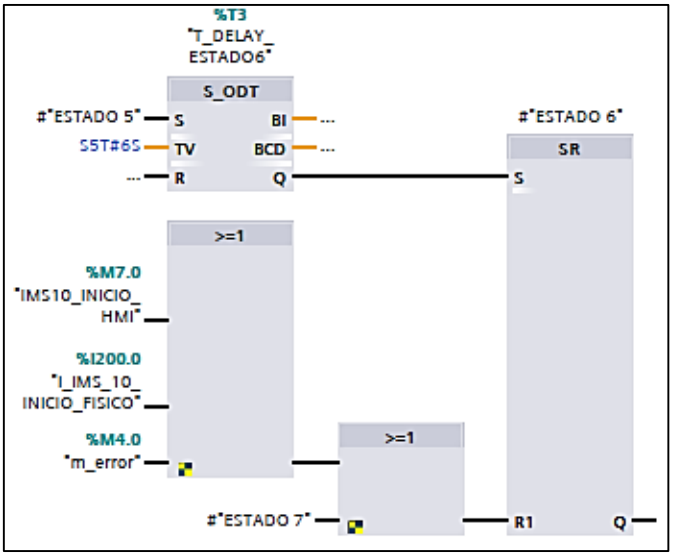


Fig. A2- 6 Transición del estado 5 al estado 6

ESTADO 6: Inicia el descenso de la plataforma, manteniendo activo únicamente el cilindro de elevación (cilindros paralelos inactivos):

- "Global".Q_IMS_10_M3_CIL_NEU_ELEVADOR

Segmento 7: Transición del estado 6 al estado 7 (Figura 3-16). - se llega al estado 7 al completar el descenso de cilindros paralelos, retrocediendo a su posición inicial, activándose el sensor de retroceso de los cilindros, con las siguientes condiciones:

1. I_IMS_10_B4_CIL_PAR_RETROCESO, y
2. Que venga del estado 6

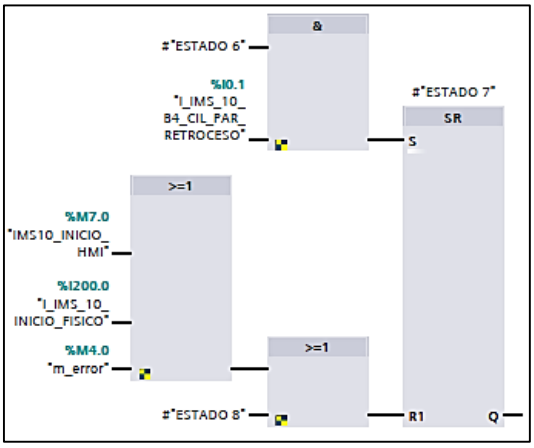


Fig. A2- 7 Transición del estado 6 al estado 7

ESTADO 7: No hay cilindros activos, se inicia descenso de cilindro elevador.

Segmento 8: Transición del estado 7 al estado 8 (Figura 3-17). - Se llega al estado 8, al completar el descenso de cilindro elevador, regresando a su posición inicial, activándose el sensor de retroceso, con las siguientes condiciones:

1. I_IMS_10_B6_CIL_ELEV_RETROCESO, y
2. Que venga del estado 7

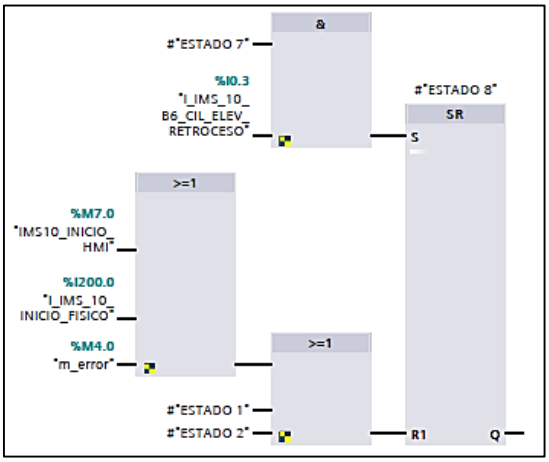


Fig. A2- 8 Transición del estado 7 al estado 8

ESTADO 8: No se ejecutan comandos, estado de espera de llenado del almacén de pallets.

ANEXO 3

Segmentos del Bloque de Funciones:

TRANSICION_LIBERAR [FB2]

Segmento 1: Estado 8.1 al Estado 1.1 (Figura 3-19). - Se llega al Estado 1.1, cumpliendo las siguientes condiciones:

1. Venir del estado 8.1,
2. Que el almacén de pallets esté vacío: CONTROL_PALLET.CV == 0, o
3. Al presionar:
   - La botonera virtual en el HMI de Paro/Marcha, pulsar IMS10_INICIO_HMI
   - La botonera física: I_IMS_10_INICIO_FISICO
   - La botonera en la co-simulación: m_error.

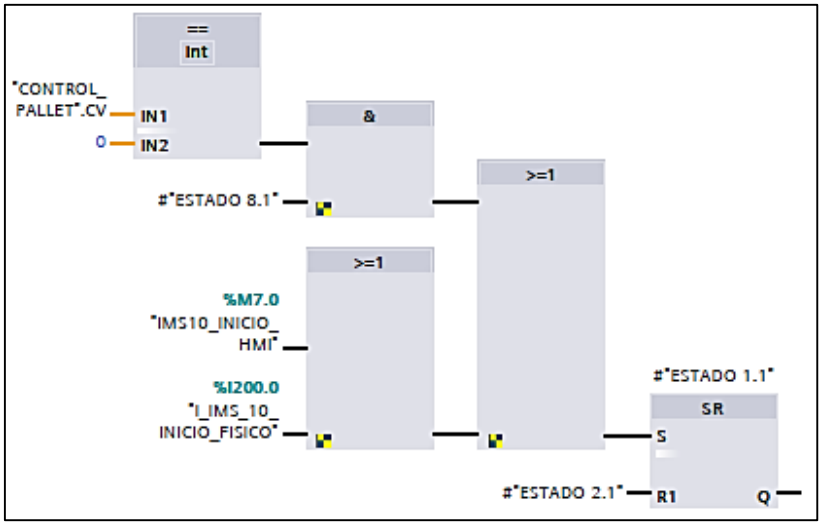


Fig. A3- 1 Estado 8.1 al Estado 1.1

ESTADO 1.1: Estado de espera de instrucción de almacenar o liberar.

Segmento 2: Estado 1.1 al Estado 2.1 (Figura 3-20). - Se llega al estado 2.1 con las siguientes condiciones:

1. Al presionar cualquiera de estas opciones:
   - La botonera virtual en el HMI de almacenar: IMS_10_LIBERAR_HMI
   - La botonera física: I_IMS_10_LIBERAR_FISICO
2. Que se cumpla con la condición que:
   - Que por lo menos haya un pallet en el almacén de pallets: CONTROL_PALLET.CV > 0
   - Y que venga del estado 1 o 1.1

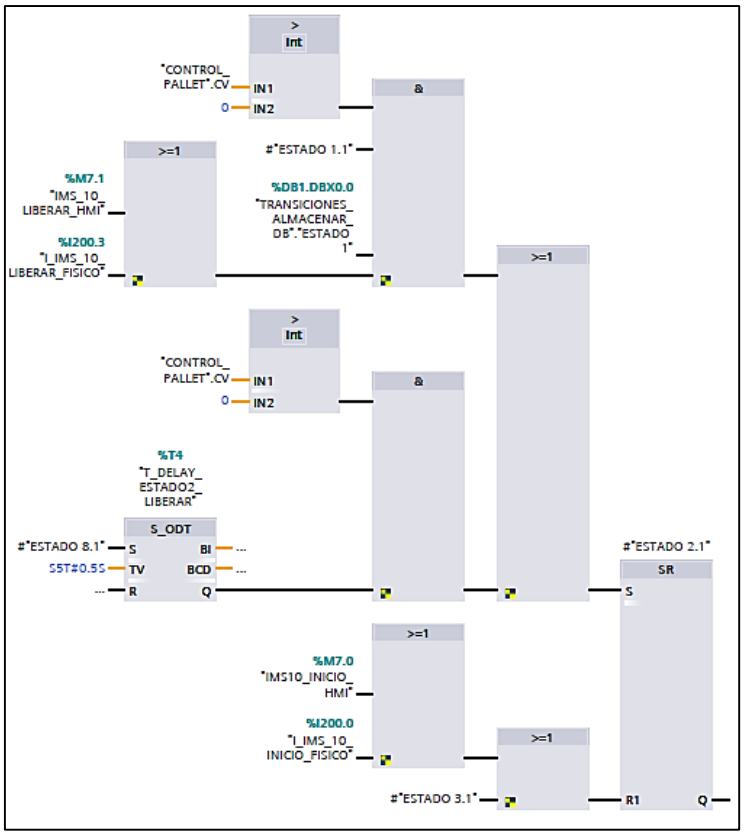


Fig. A3- 2 Estado 1.1 al Estado 2.1

ESTADO 2.1: Empieza el proceso de liberar, elevando los cilindros paralelos. Se activan las siguientes variables:

- "Global".Q_IMS_10_M4_CIL_LAT_PARALELO

Segmento 3: Estado 2.1 a Estado 3.1 (Figura 3-21). - Se llega al estado 3.1 cumpliendo las condiciones:

1. Esté activo el sensor de posición de avance del cilindro paralelo: I_IMS_10_B5_CIL_PAR_AVANCE, y
2. Que venga del estado 2.1

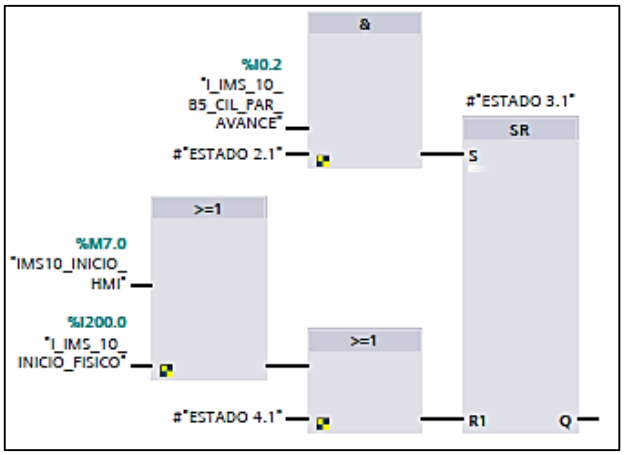


Fig. A3- 3 Estado 2.1 a Estado 3.1

ESTADO 3.1: Se inicia el descenso del pallet. Se activan las siguientes variables:

- "Global".Q_IMS_10_M3_CIL_NEU_ELEVADOR
- "Global".Q_IMS_10_M4_CIL_LAT_PARALELO

Segmento 4: Estado 3.1 a Estado 4.1 (distancia pallets) (Figura 3-22). - Se llega al estado 4.1, cumpliendo una pausa de seguridad de 3 segundos para distanciar los pallets dentro del almacén con las siguientes condiciones:

1. T_DELAY_ESTADO4_LIBERAR, y
2. Que venga del estado 3.1

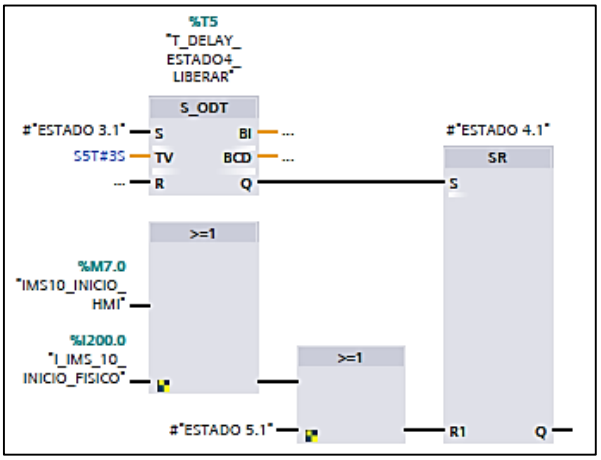


Fig. A3- 4 Segmento 4: Estado 3.1 a Estado 4.1 (distancia pallets)

ESTADO 4.1: Para iniciar el proceso salida de pallets, se activan los cilindros elevadores y separadores:

- Se realiza el conteo de pallets en el almacén con: CONTROL_PALLET
- "Global".Q_IMS_10_M3_CIL_NEU_ELEVADOR
- Q_IMS_10_M5_CILINDRO_SEPARADOR

Segmento 5: Estado 4.1 a Estado 5.1 (Figura 3-23). - Se llega al estado 5.1 cumpliendo una pausa de seguridad de 5 segundos, se desactivan los cilindros de parada y regresan a su estado inicial, con las siguientes condiciones:

1. T_DESACTIVAR_B8, y
2. Que venga del estado 4.1

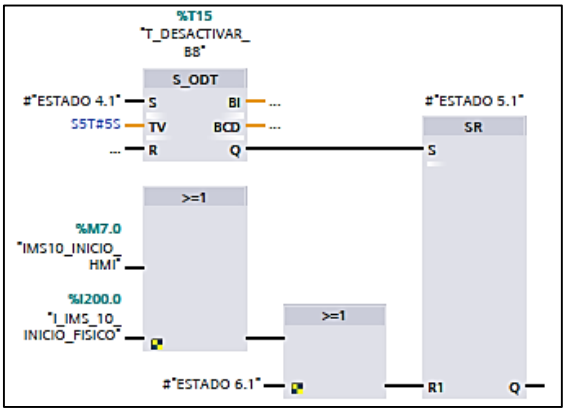


Fig. A3- 5 Segmento 5: Estado 4.1 a Estado 5.1

ESTADO 5.1: No se ejecutan comandos

Segmento 6: Estado 5.1 a Estado 6.1 (Figura 3-24). - Se llega al estado 6.1, desactivando el cilindro de elevación y regresando a su estado inicial, cumpliendo las condiciones:

1. Esté activo el sensor de posición de retroceso del cilindro elevador: I_IMS_10_B6_CIL_ELEV_RETROCESO,
2. Que venga del estado 5.1.

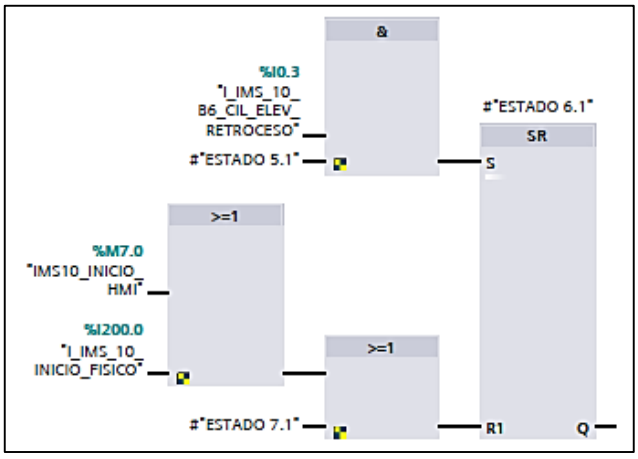


Fig. A3- 6 Segmento 6: Estado 5.1 a Estado 6.1

ESTADO 6.1: El pallet está listo para ser sacado de la posición de almacenamiento, y se activa la banda para traslado hacia la derecha. Se activan las siguientes variables:

- "Global".Q_IMS_10_MOTOR_ADELANTE
- Q_ESCLAVO8_MOTOR_ADELANTE
- Q_ESCLAVO9_MOTOR_ADELANTE

Segmento 7: Estado 6.1 a Estado 7.1 (Figura 3-25). - Se llega al estado 7.1, verificando si se ha realizado correctamente la liberación de un pallet, enviando una señal con el sensor del lado derecho, cumpliendo las condiciones:

1. Esté activo el sensor de posición del lado derecho de la banda: I_IMS_10_B2_SENSOR_DERECHA, y
2. Que venga del estado 6.1

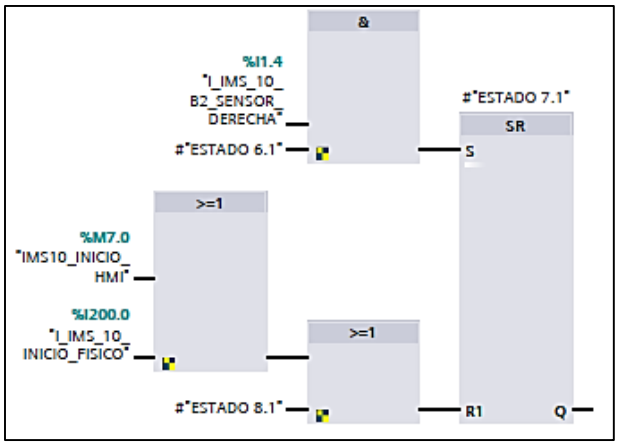


Fig. A3- 7 Segmento 7: Estado 6.1 a Estado 7.1

ESTADO 7.1: Se mantiene activa la banda para traslado hacia la derecha. Se activan las siguientes variables:

- "Global".Q_IMS_10_MOTOR_ADELANTE
- Q_ESCLAVO8_MOTOR_ADELANTE
- Q_ESCLAVO9_MOTOR_ADELANTE

Segmento 8: Estado 7.1 al Estado 8.1 (DESPUES DEL SENSOR DERECHA) (Figura 3-26). - Se llega al estado 8.1, cumpliendo una pausa de seguridad de 2 segundos y poder continuar liberando, cumpliendo las siguientes condiciones:

1. T_DELAY_ESTADO8_LIBERAR, y
2. Que venga del estado 7.1

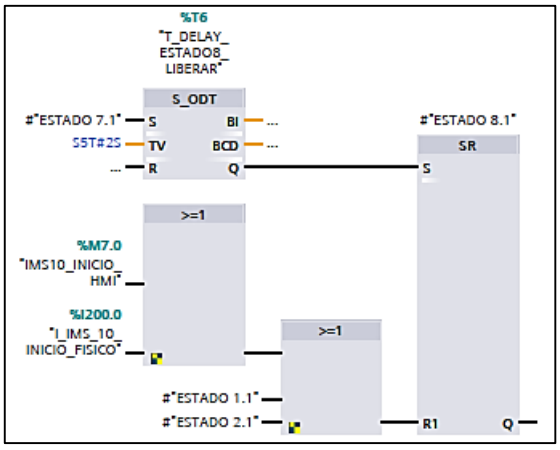
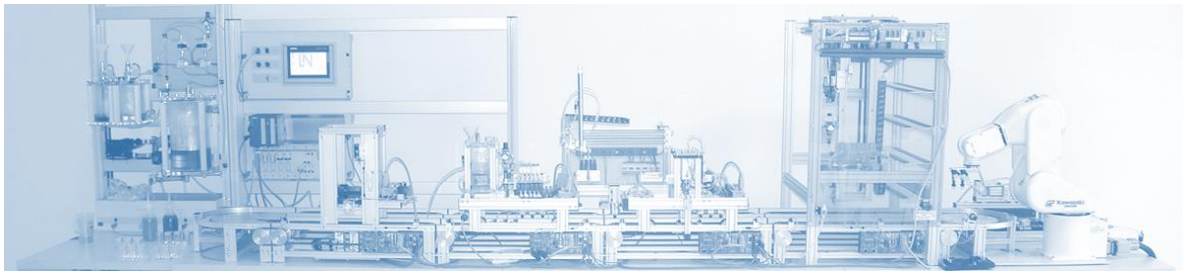


Fig. A3- 8 Segmento 8: Estado 7.1 al Estado 8.1

ESTADO 8.1: No se ejecutan comandos.

ANEXO 4

espol Facultad de Ingeniería en Electricidad y Computación



Laboratorio de Control de Procesos Industriales

Práctica Autónoma 2

AUTONOMA 2: Programación de una estación apiladora usando el grafico funcional de transición etapas GRAFCET

OBJETIVOS

GENERAL

Desarrollar el proceso de almacenamiento y despacho de pallets de acuerdo a la lista de instrucciones para la estación IMS 10

ESPECIFICOS

Diseñar el GRAFCET del ejercicio propuesto en esta práctica.

Desarrollar la programación en lenguaje de bloques haciendo uso de FBs y FCs.

INTRODUCCION

En la industria actual se pueden encontrar diferentes sistemas de apilado, con distintos diseños o distintos sistemas de funcionamiento dependiendo de la necesidad de apilar o dispensar de la empresa. Los apiladores/dispensadores se distinguen principalmente por las siguientes características:

- Por su sistema de elevación: hidráulico, neumático, eléctrico, etc.
- Por su disposición en funcionamiento fijo o móvil
- Por su capacidad de almacenaje: Carga máxima y numero de pallets.

Figure 1 Apilador/Dispensador de Pallets Fijo

Bloques de instrucciones básicas

A continuación, se muestra algunos bloques lógicos:

TON, Retardo a la conexión: Con esta instrucción se puede retardar la activación de la salida Q por el tiempo programado PT. La instrucción se inicia cuando el resultado lógico (RLO) de la entrada IN cambia de "0" a "1" (flanco de señal ascendente). Cuando el estado lógico de la entrada IN cambia de "1" a "0", se desactiva la salida Q. La función de temporización se reinicia al detectarse un nuevo flanco de señal ascendente en la entrada IN.

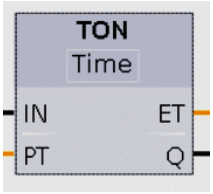


Figure 2 Bloque de retardo a la conexión TON

Parámetro	Declaración	Tipo de datos	Área de memoria	Descripción
IN	Input	BOOL	I, Q, M, D, L, T, C	Entrada de arranque
PT	Input	TIME	I, Q, M, D, L, P o constante	Tiempo del retardo al conectar El valor del parámetro PT debe ser positivo.
Q	Output	BOOL	I, Q, M, D, L	Salida que se activa una vez transcurrido el tiempo PT.
ET	Output	TIME	I, Q, M, D, L	Valor de tiempo actual

Figure 3 Parámetros de TON

CTUD, Contador Ascendente-Descendente: Esta instrucción incrementa y decrementa el valor de contaje en la salida CV. Cuando el estado lógico de la entrada CU cambia de "0" a "1" el valor de contaje de la salida CV se incrementa en uno. Cuando el estado lógico de la entrada CD cambia de "0" a "1" el valor de contaje de la salida CV se decrementa en uno. Si en un ciclo del programa se detecta un flanco de señal ascendente en las entradas CU y CD, el valor de contaje CV no se modifica.

Si el estado lógico de la entrada LD cambia a "1", el valor de contaje de la salida CV adopta el valor del parámetro PV. Mientras la entrada LD tenga el estado lógico "1", el estado lógico de las entradas CU y CD no tendrá efecto alguno en la instrucción.

El valor de contaje se pone a cero si el estado lógico de la entrada R cambia a "1". Mientras la entrada R tenga el estado lógico "1", un cambio del estado lógico de las entradas CU, CD y LD no tendrá efecto alguno en la instrucción.

El estado del contador ascendente se puede consultar en la salida QU. Si el valor de contaje CV es mayor o igual al valor del parámetro PV, la salida QU adopta el estado lógico "1".

El estado del contador descendente se puede consultar en la salida QD. Si el valor de contaje CV es menor o igual a cero, la salida QD adopta el estado lógico "1".

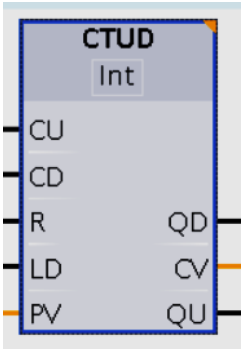


Figure 4 Registro de Contador ascendente/descendente

Parámetro	Declaración	Tipo de datos	Área de memoria	Descripción
CU	Input	BOOL	I, Q, M, D, L	Entrada de contaje ascendente
CD	Input	BOOL	I, Q, M, D, L	Entrada de contaje descendente
R	Input	BOOL	I, Q, M, D, L, T, C, P	Entrada de reset
LD	Input	BOOL	I, Q, M, D, L, T, C, P	Entrada de carga
PV	Input	INT	I, Q, M, D, L, P o constante	Valor con el que se activa la salida QU. / Valor al que se pone la salida CV cuando LD = 1.
QU	Output	BOOL	I, Q, M, D, L	Estado del contador ascendente
QD	Output	BOOL	I, Q, M, D, L	Estado del contador descendente
CV	Output	INT	I, Q, M, D, L, P	Valor de contaje actual

Figure 5 Parámetros de CTUD

Comparación:

==	La instrucción "Igual" consulta si el valor de la entrada IN1 es igual al valor de la entrada IN2.
<>	La instrucción "Diferente" consulta si el valor de la entrada IN1 es distinto al valor de la entrada IN2.

>=	La instrucción "Mayor o igual" consulta si el valor de la entrada IN1 es mayor o igual que el valor de la entrada IN2. Ambos valores de comparación deben ser del mismo tipo de datos.
<=	La instrucción "Menor o igual" consulta si el valor de la entrada IN1 es menor o igual que el valor de la entrada IN2. Ambos valores de comparación deben ser del mismo tipo de datos.
>	La instrucción "Mayor" consulta si el valor de la entrada IN1 es mayor que el valor de la entrada IN2. Ambos valores de comparación deben ser del mismo tipo de datos.
<	La instrucción "Menor" consulta si el valor de la entrada IN1 es menor que el valor de la entrada IN2. Ambos valores de comparación deben ser del mismo tipo de datos.

Si se cumple la condición de la comparación, la instrucción devuelve el resultado lógico (RLO) "1". Si la condición de la comparación no se cumple, la instrucción devuelve el RLO "0".

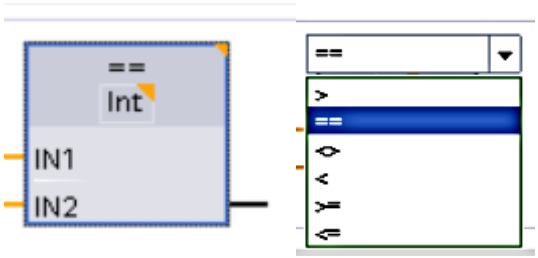


Figure 6 Bloque Comparador

Asignación Permanente

Esta instrucción es ejecutada mediante un bloque SR para una marca o salida del PLC (M, Q).

En la Figura 7 se puede observar el uso de la asignación permanente para la salida Q_IMS10_M5, la cual es activada permanentemente en el estado 2 del bloque de datos DB5 mediante el set de la marca M_IMS10_M5, esto hará que para el resto de los estados siguientes la salida Q_IMS10_M5 este activada. Se desactiva mediante el reset de M_IMS10_M5 cuando se activan los estados 1 y 4 de DB5.

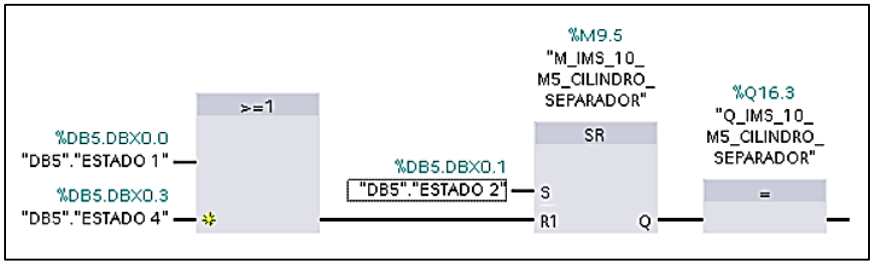


Figure 7 Uso de la asignación permanente

ACTIVIDADES

1) Implementar el GRAFCET del proceso de apilado o dispensado de pallets de acuerdo con la paridad de su paralelo tal y como indica en el presente documento.
2) Presentar la Tabla de Dirección de Entradas, Salidas, Marcas, Temporizadores y Registro contadores según sea necesario. (Revisar la sección de Anexos)
3) Desarrollar la programación mano o en TIA PORTAL del proceso que le corresponda haciendo uso de la metodología los bloques de programación FBs y FCs. De ser necesario puede utilizar las entradas, salidas, marcas, temporizados y registros contadores que sean necesarios.

APILADO DE PALLETS **(PARALELO PAR)**

TRANSICION E6 A E1	**Se espera 2 segundos**
TRANSICION E7 A E1	Se activa manualmente la marca de reconocimiento de **DEPOSITO LLENO**
INSTRUCCION E1	Nada
TRANSICION E1 A E7	Registro contador de pallets es igual a 4 (CTUD.CV==4)

INSTRUCCION E7	Se activa marca "**ALMACEN ESTA LLENO**"
TRANSICION E1 A E2	Se presiona el pulsador **START_ALM**, el pallet se encuentra en la posición inicial izquierda (**B1**), el cilindro paralelo está en posición de retroceso (**B4**), el cilindro elevador está en posición de retroceso (**B6**) y el almacén no está lleno.
INSTRUCCION E2	Se activa la marcha derecha de la banda transportadora (**QR**), el cilindro de parada (**M2**) y temporizador de retardo a la conexión T1 de 5S.
TRANSICION E2 A E3	El cilindro de parada llega a la posición de avance (**B3**) y se activa la salida Q de T1.
INSTRUCCION E3	Se activa **permanentemente** los cilindros paralelos (**M4:=1**)
TRANSICION E3 A E4	El cilindro paralelo llega a posición de avance (**B5**)
INSTRUCCION E4	Se activa **permanentemente** el cilindro elevador (**M3:=1**) y un temporizador de retardo a la conexión T2 de 4S
TRANSICION E4 A E5	Salida Q del temporizador T2 activada
INSTRUCCION E5	Se **desactiva** el cilindro paralelo (**M4=0**)
TRANSICION E5 A E6	Cilindro paralelo está en posición de retroceso (**B6**)
INSTRUCCION E6	Se **desactiva** el cilindro elevador (**M3=0**) y se aumenta en 1 el registro contador de pallets (CTUD.CV+1)

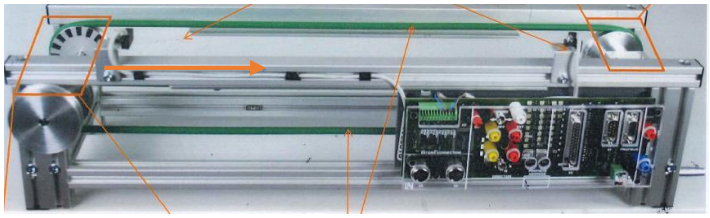
DISPENSADO DE PALLETS **(PARALELO IMPAR)**

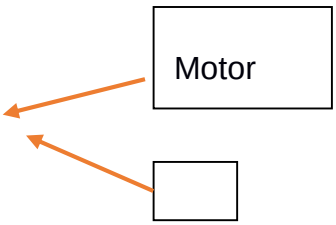
TRANSICION E6 A E1	El pallet se encuentra en la posición derecha de la banda transportadora **(B2)**
TRANSICION E7 A E1	Se activa manualmente la marca de reconocimiento de **DEPOSITO VACIO**
INSTRUCCION E1	Nada
TRANSICION E1 A E7	Registro contador de pallets es igual a cero (CTUD.CV==0)
INSTRUCCION E7	Se activa marca **ALMACEN ESTA VACIO**
TRANSICION E1 A E2	Se presiona el pulsador **START_RET**, el cilindro paralelo está en posición de retroceso (**B4**), el cilindro elevador está en posición de retroceso (**B6**) y el almacén contiene al menos un pallet (CTUD.CV > 0).
INSTRUCCION E2	Se activa **permanentemente** los cilindros paralelos (**M4:=1**)
TRANSICION E2 A E3	Cilindro paralelo en la posición de avance (**B5**)
INSTRUCCION E3	Se activa **permanentemente** el cilindro elevador (**M3:=1**) y un temporizador de retardo a la conexión T3 de 4S
TRANSICION E3 A E4	La salida Q de T3 esta activada.
INSTRUCCION E4	Se desactiva el cilindro elevador (**M3=0**) y se activa permanentemente el cilindro separador (**M5:=1**).
TRANSICION E4 A E5	Cilindro elevador en posición de retroceso (**B4**)
INSTRUCCION E5	Se desactiva el cilindro separador (**M5=0**) y los cilindros paralelos (**M4=0**)
TRANSICION E5 A E6	Cilindro paralelo en la posición de retroceso (**B4**)

INSTRUCCION E6	Se disminuye en 1 el registro de contador de pallet (CTUD.CV -1) y se activa marcha a la derecha de la banda transportadora (**QR**).

ANEXOS

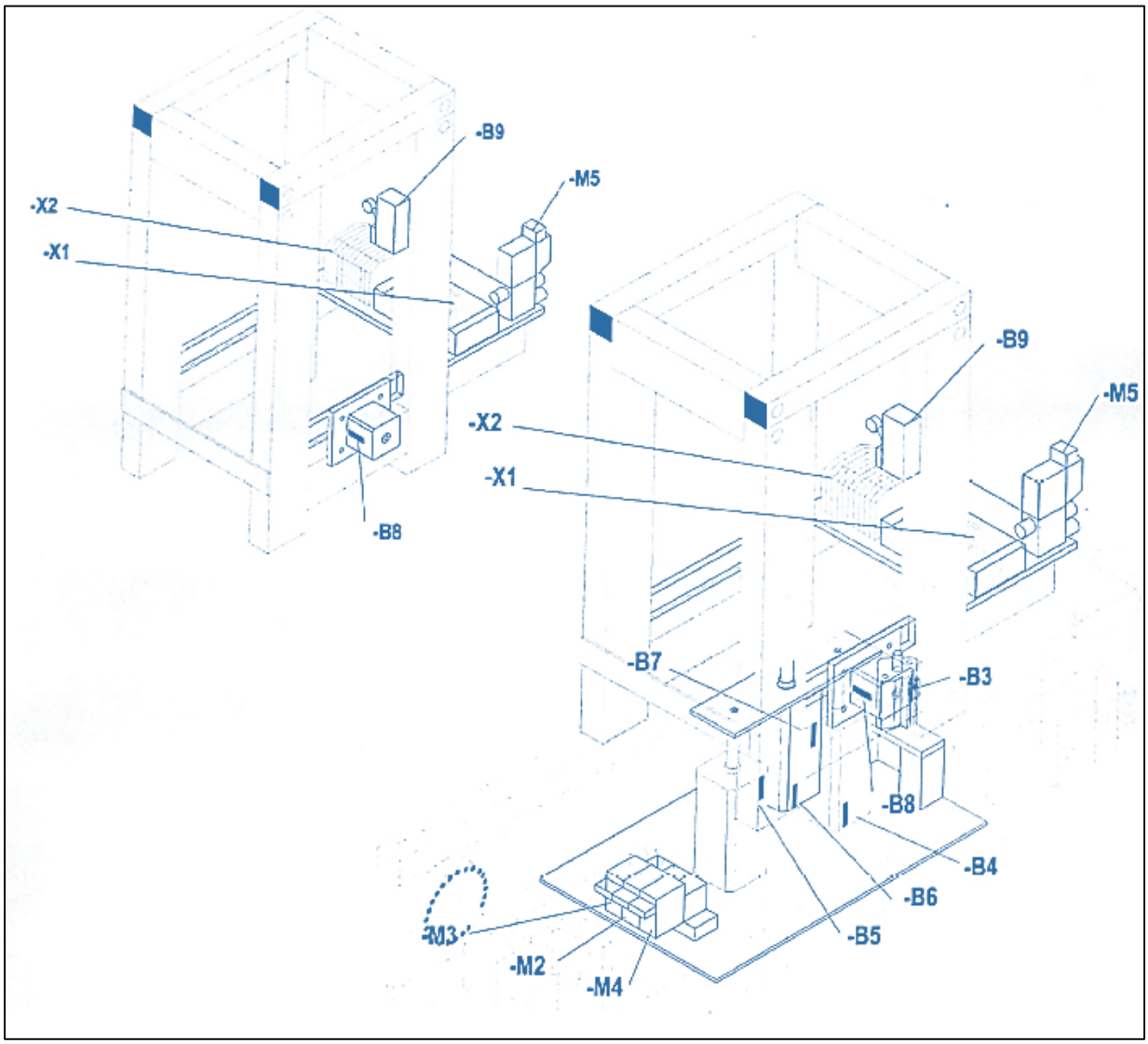
Banda Transportadora





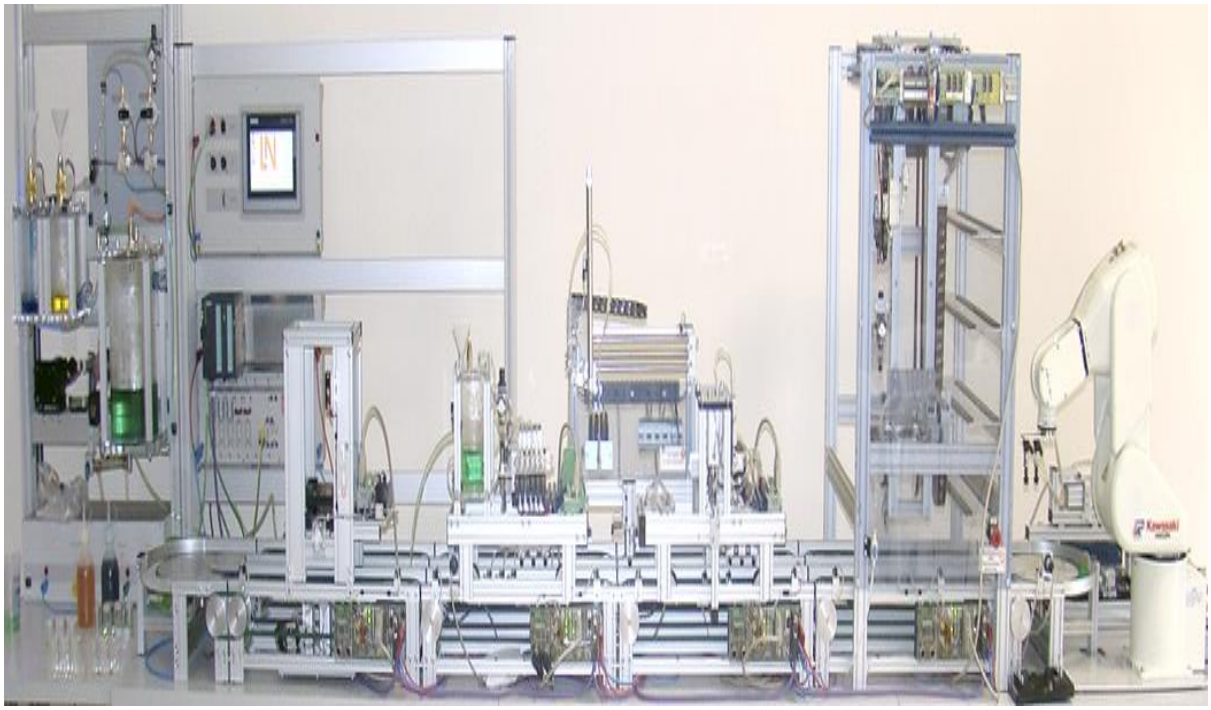
Descripción	E/S
Sensor de posición final -B1	E(X+1).3
Sensor de posición final -B2	E(X+1).4
Casquillo M12 (X3)	E(X+1).5
Casquillo M12 (X4)	E(X+1).6
Marcha a la derecha	A(X+1).0
Marcha a la izquierda	A(X+1).1
Marcha lenta	A(X+1).2

Apilador/Dispensador de Pallets



Descripción	ID	Pin	E/S
Sensor magnético: cilindro de parada en posición de avance	-B3	1	I(X+0).0
Sensor magnético: cilindro paralelo en posición de retroceso	-B4	2	I(X+0).1
Sensor magnético: cilindro paralelo en posición de avance	-B5	3	I(X+0).2
Sensor magnético: cilindro elevador en posición de retroceso	-B6	4	I(X+0).3
Sensor magnético: posición de separación del cilindro elevador	-B7	5	I(X+0).4
Sensor magnético: cilindro de separación en posición de retroceso	-B8	6	I(X+0).5
Sensor mecánico: almacén lleno	-B9	7	I(X+0).6
Válvula distribuidora de 4 a 2 vías: cilindro neumático para el cilindro de parada	-M2	14	Q(X+0).0
Válvula distribuidora de 4 a 2 vías: cilindro neumático para el cilindro elevador	-M3	15	Q(X+0).1
Válvula distribuidora de 4 a 2 vías: cilindro neumático para el cilindro paralelo	-M4	16	Q(X+0).2
Válvula distribuidora de 3 a 2 vías: cilindro neumático para el cilindro separador	-M5	17	Q(X+0).3

ANEXO 5



Laboratorio de Control de Procesos Industriales

Ing. Livington Miranda

Práctica 2

PRÁCTICA 2: Diseño de pantalla HMI para la apiladora de pallets.

OBJETIVOS

GENERAL

Diseñar una pantalla HMI mediante programación orientada objetos para la estación IMS 10.

ESPECIFICOS

Establecer una red virtual para la comunicación del PLC CPU 314C 2 PN/DP, pantalla TP 700 y esclavo IMS 10.

Constatar la red física profibus DP y Profinet.

Diseñar una pantalla Inicial y de proceso de apilado de pallets.

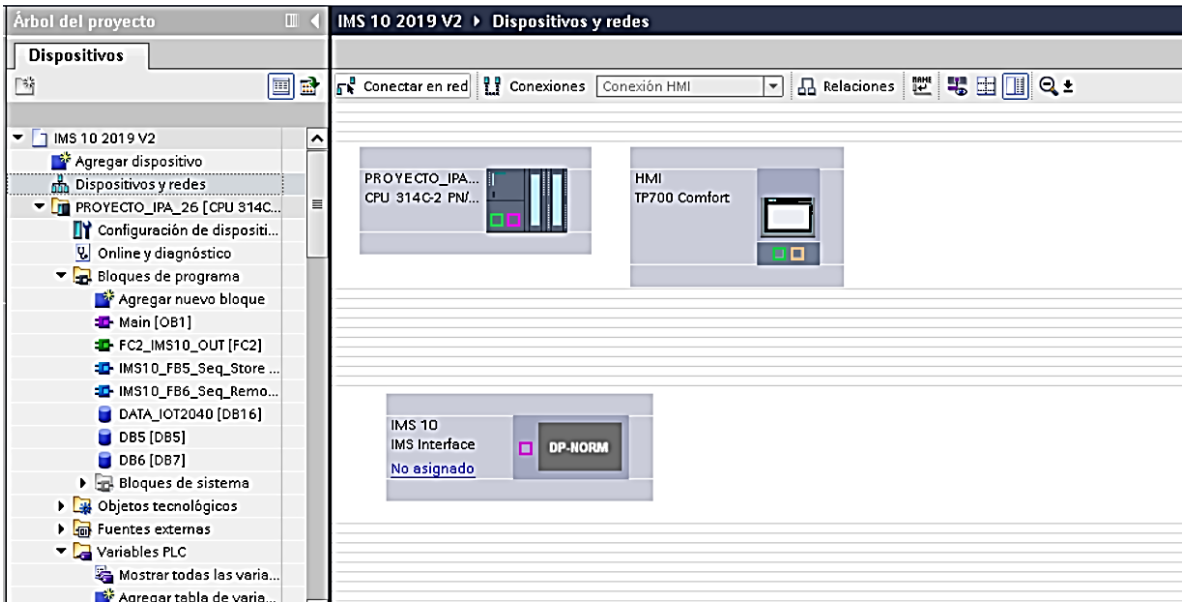
Supervisar el proceso de apilado de pallets mediante la pantalla HMI

PROCEDIMIENTO

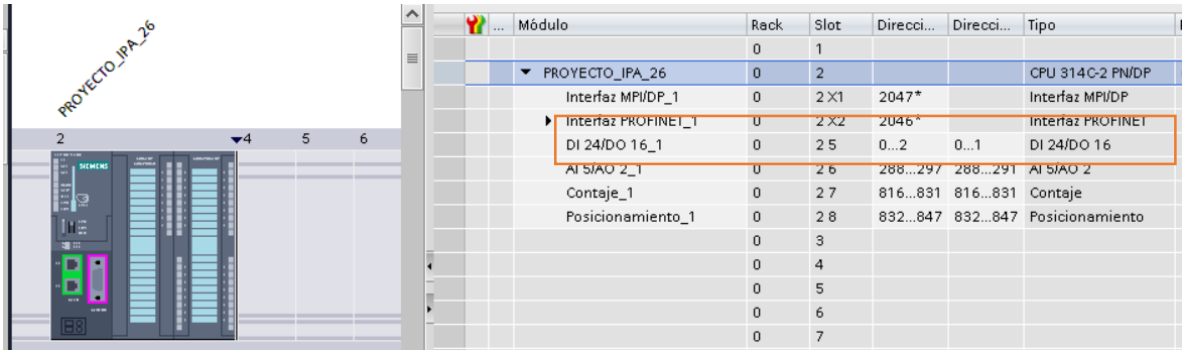
Comunicación PLC y esclavo Profibus DP IMS 10

1. Dirigirse a la carpeta CONTROL DE PROCESOS que está ubicada en el escritorio.
2. Abrir la carpeta correspondiente a su paralelo y editar con sus nombres y apellidos la carpeta de proyecto de la práctica guiada 2, (mantener el formato establecido).
3. Dirigirse a la sección de árbol de proyectos y dar doble clic a Dispositivos y redes.

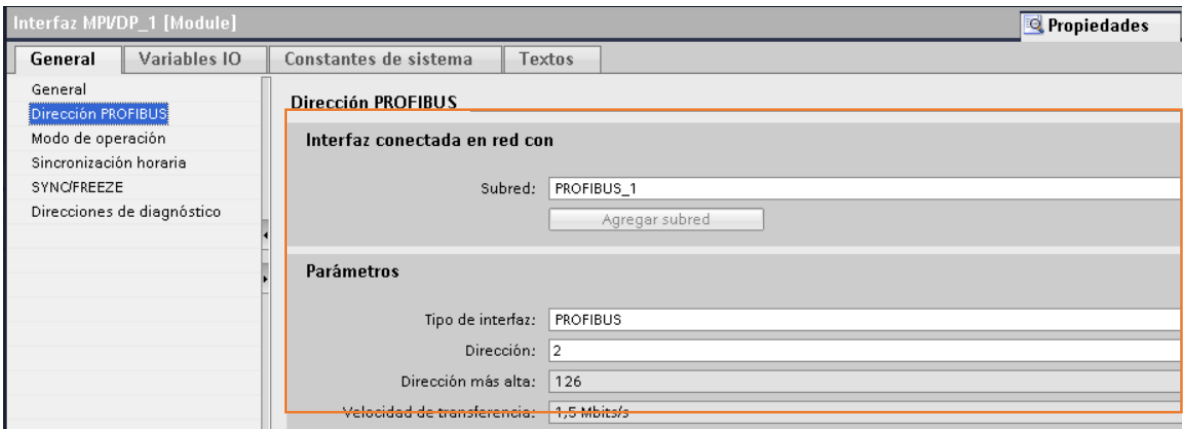
4. En la ventana de ***Vista de Redes*** observaremos tres dispositivos: PLC CPU 314C 2PN/DP, la pantalla Táctil TP700 y el esclavo profibus DP IMS 10.



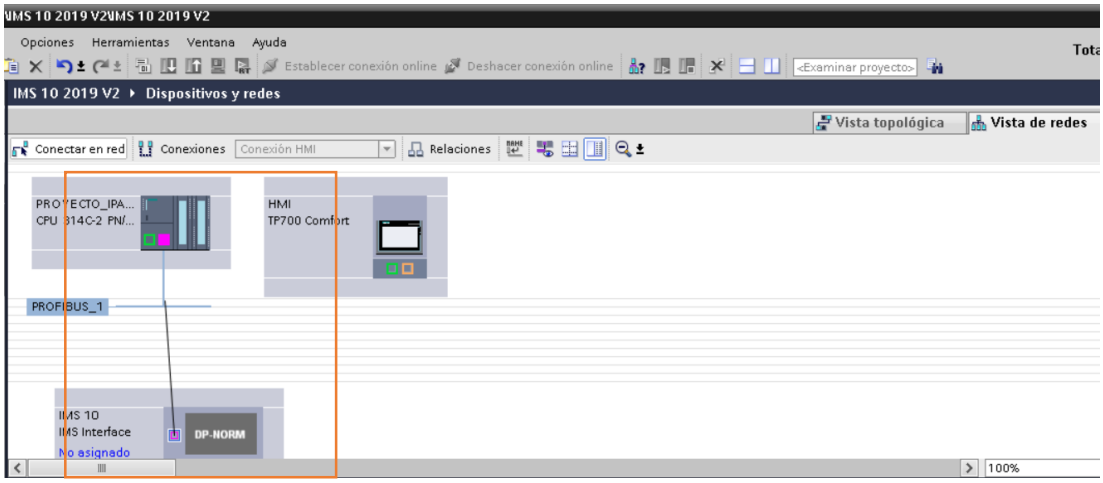
5. Vamos a configurar la dirección de E/S del PLC. Primero damos clic al PLC y verificamos las direcciones señaladas.



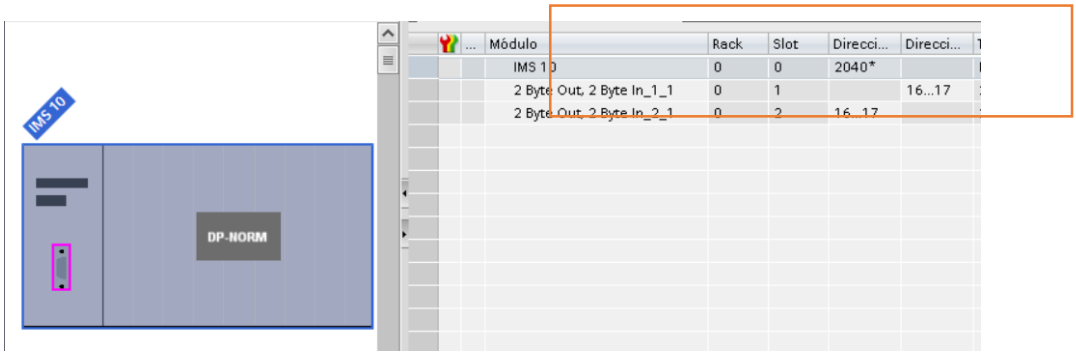
6. Siguiente configurar la dirección y subred profibus del PLC.



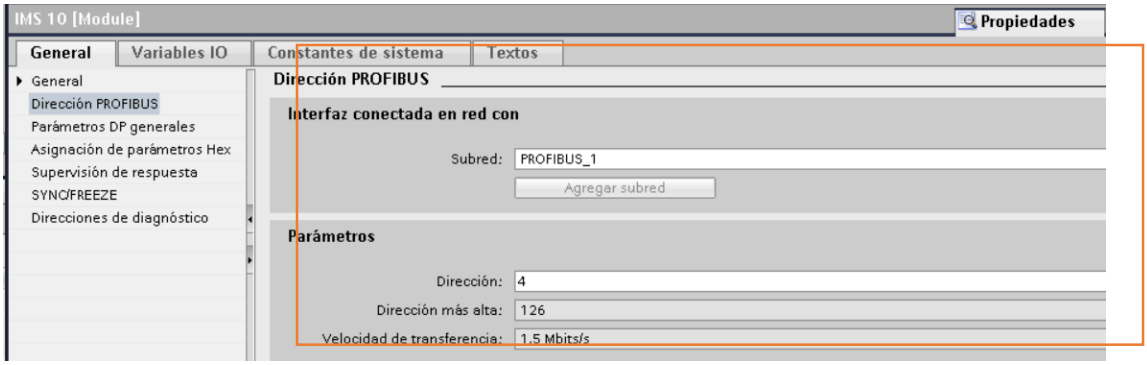
7. Nos dirigimos a la vista de redes y conectamos el esclavo IMS 10 a la red profibus DP.



8. Una vez conectado el IMS 10 a la red profibus DP, damos doble clic en este y configuramos las direcciones de E/S señaladas.



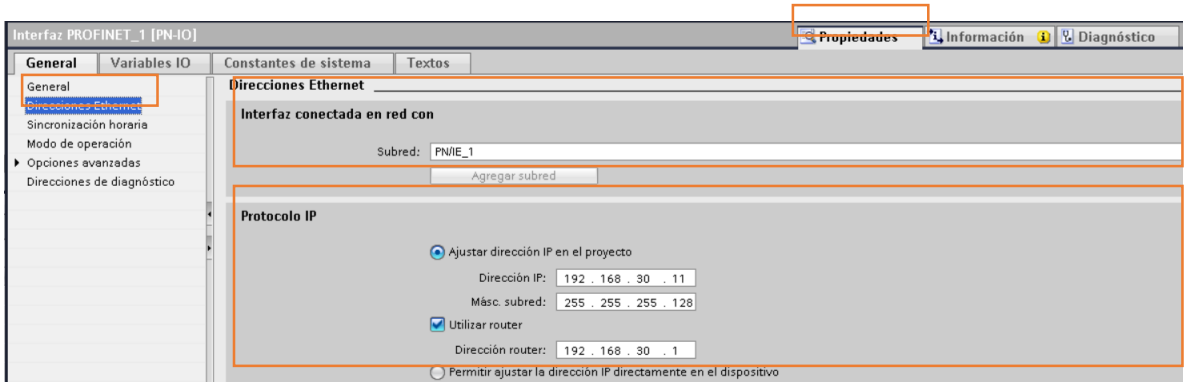
9. Luego se procede a configurar la dirección profibus DP del IMS 10. (Constatar que sea igual a la dirección física de la tarjeta esclavo profibus)



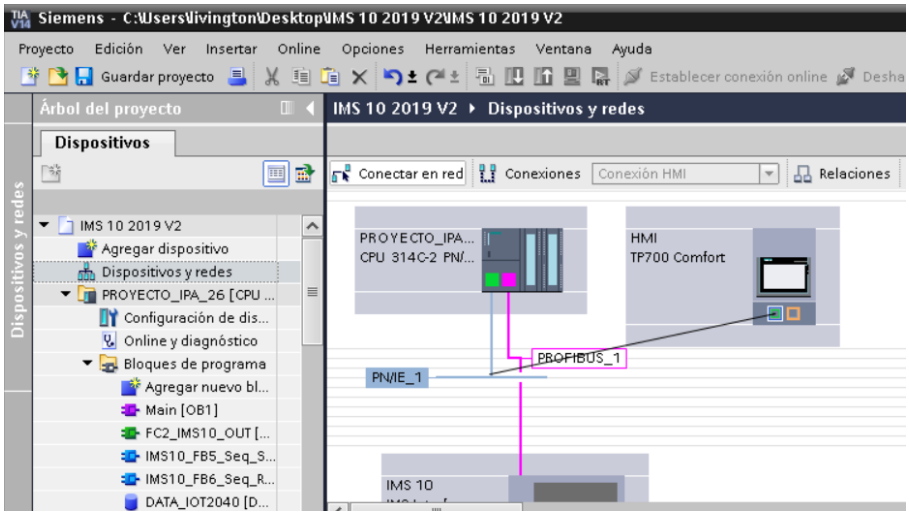
<u>Comunicación PLC y pantalla HMI TP 700</u>

1. Para comunicar el PLC y la pantalla HMI, primero verificamos las direcciones ethernet de ambos dispositivos. Nos ubicamos en la ***vista de redes*** y damos doble clic en el

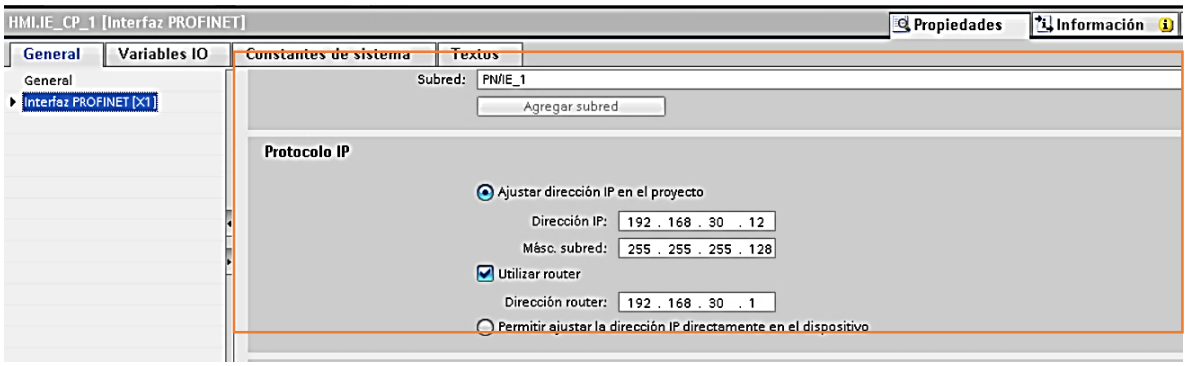
PLC. En la ventana de ***propiedades*** nos dirigimos a la sección de ***Direcciones Ethernet*** y verificamos la dirección IP y subred señalada.



2. Luego en ***vista de redes*** conectado la Pantalla HMI con la subred Profinet.



3. Dar doble clic sobre la pantalla HMI. En la ventana de propiedades dirigirse a la sección de Interfaz Profinet y verificar la dirección IP del dispositivo.



4. De esta manera hemos configurados comunicación entre los dispositivos mediante la red Profinet y profibus DP.

<u>Diseño de la pantalla HMI proyecto</u>

1. En la sección del árbol de buscamos la carpeta HMI [TP700 Comfort], Administrador de Imágenes, Plantillas. Damos doble clic a la plantilla con nombre BASE.

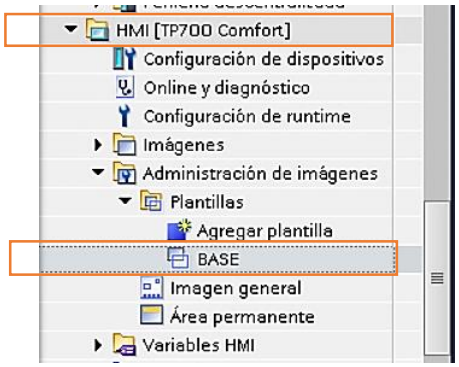


Ilustración 1 Plantilla BASE del diseño HMI

2. Se abrirá una ventana donde se presenta la plantilla base. Se debe editar la propiedad de Evento del Botón IMS 10.

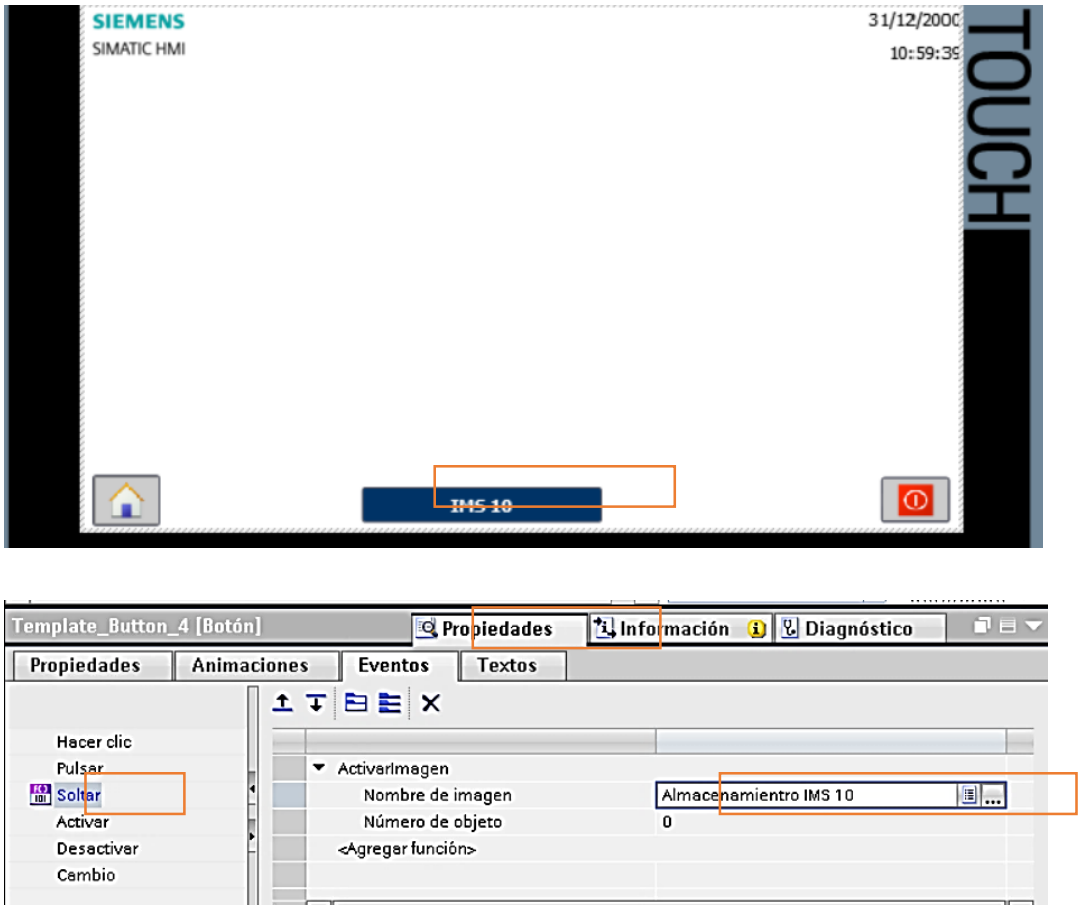
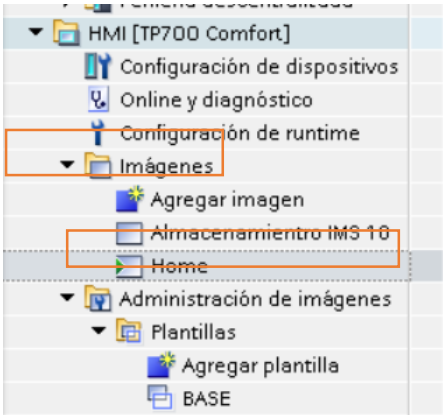


Ilustración 2 Activación de la Imagen IMS 10

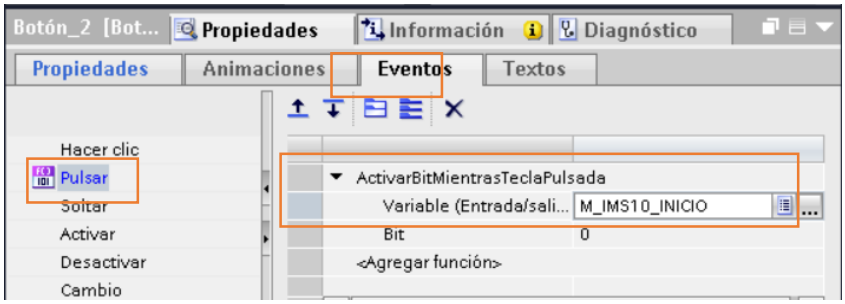
3. Verificar los eventos programados en los Iconos Home y Power Off y anotarlos en la siguiente tabla.

BOTÓN	EVENTO	FUNCION
HOME 		
POWER OFF 		

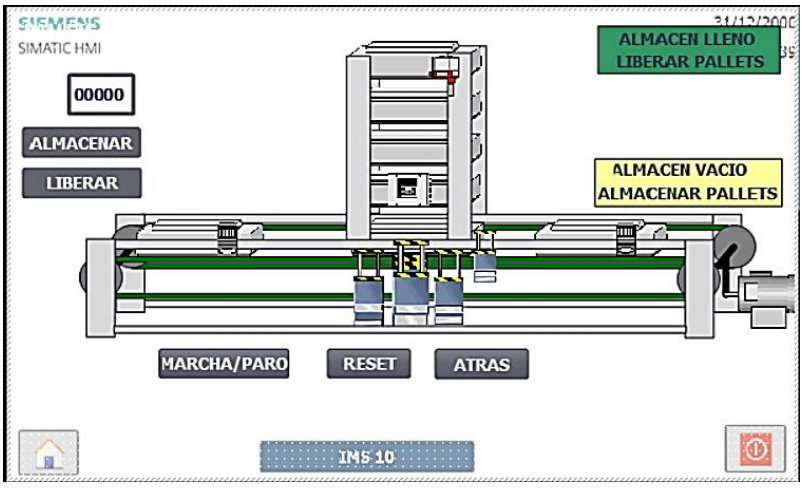
4. Ahora dirigirse a la sección de árbol de proyectos, Imágenes y dar doble clic sobre la pantalla HOME.



5. Se desplegará la pantalla HOME, en esta editar la caratula y los eventos del Botón Inicio



6. Luego proceder a la pantalla Almacenamiento IMS 10. Se observa un diseño de la estación IMS 10.

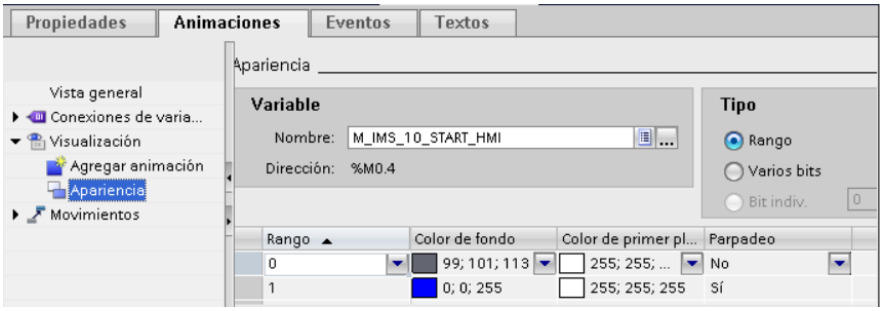
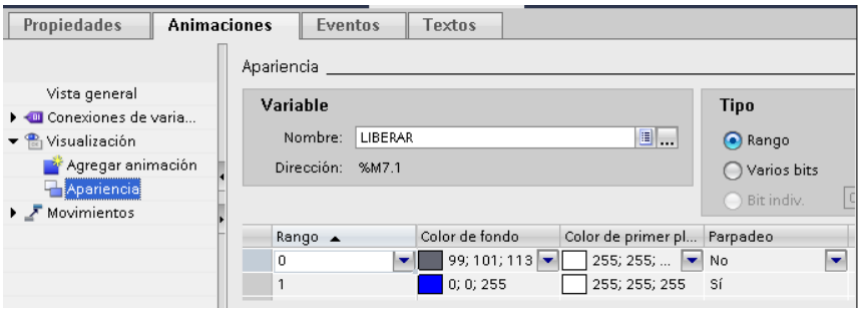


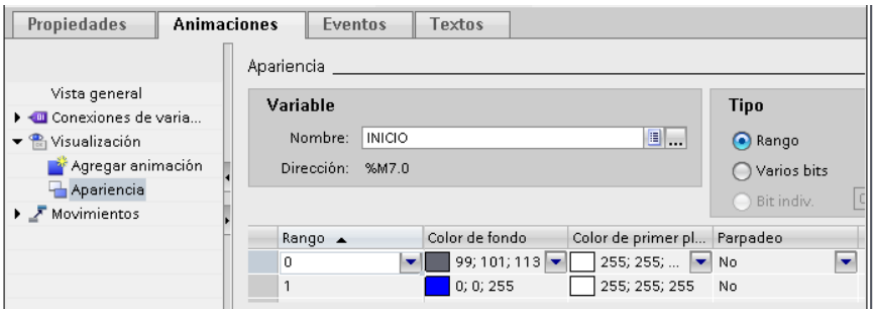
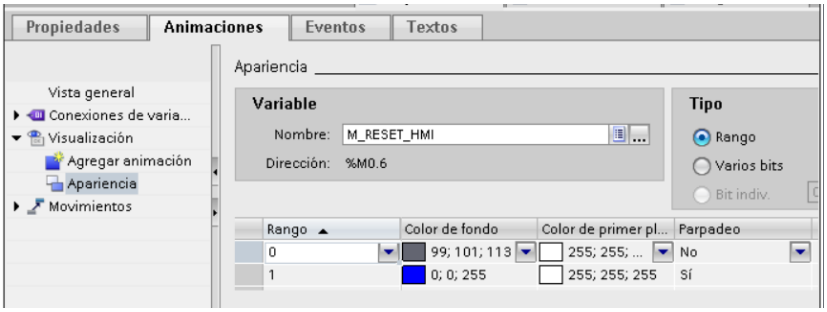
7. Programar los eventos de los botones ALMACENAR, LIBERAR, MARCHA/PARO, RESET, ATRÁS como se indica en la siguiente tabla.

BOTÓN	**EVENTO**	**FUNCION**
ALMACENAR	PULSAR	ActivarBitMientrasTeclaPusada Variable (Entrada/Salida) **M_IMS_10_START_HMI**
LIBERAR	PULSAR	ActivarBitMientrasTeclaPusada Variable (Entrada/Salida) **LIBERAR**

MARCHA/PARO	PULSAR	ActivarBitMientrasTeclaPusada Variable (Entrada/Salida) **M_IMS_10_INICIO_HMI**
RESET	PULSAR	ActivarBitMientrasTeclaPusada Variable (Entrada/Salida) **M_RESET_CONTADOR**
ATRAS	SOLTAR	ActivarImagen Variable (Entrada/Salida) **Home**

8. Para los mismos botones se programará la propiedad de animación como se indica en la siguiente tabla

BOTÓN	**ANIMACION**
ALMACENAR	
LIBERAR	

MARCHA/PARO	
RESET	

9. Luego se procede a programar la visibilidad de los cilindros neumáticos que se observan en la siguiente figura.

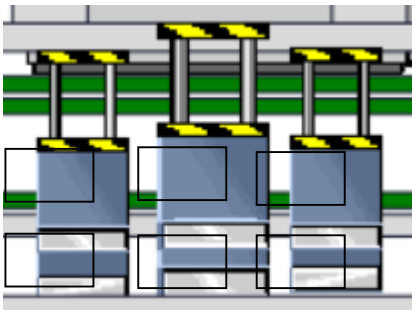
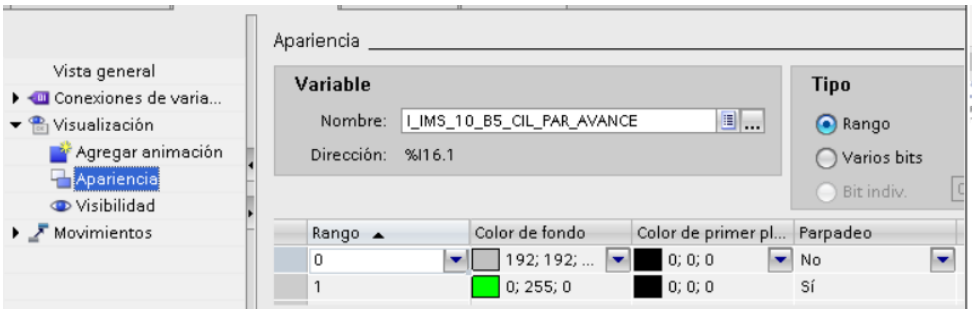
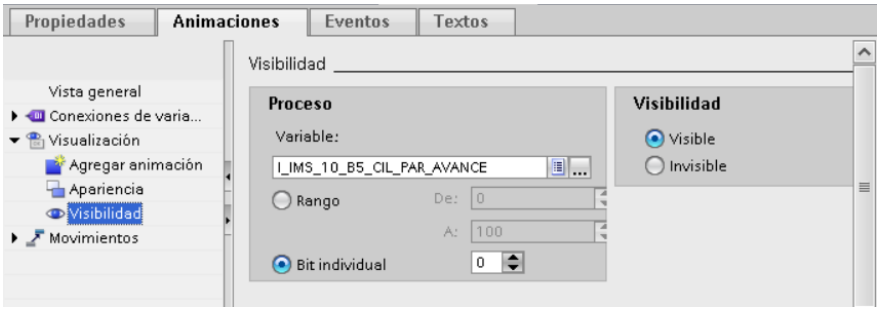
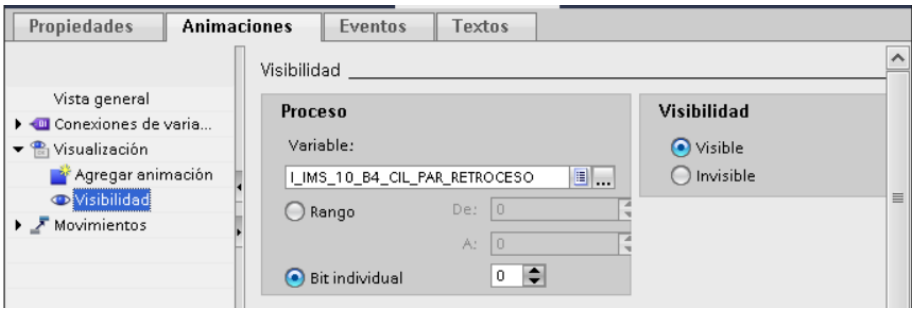
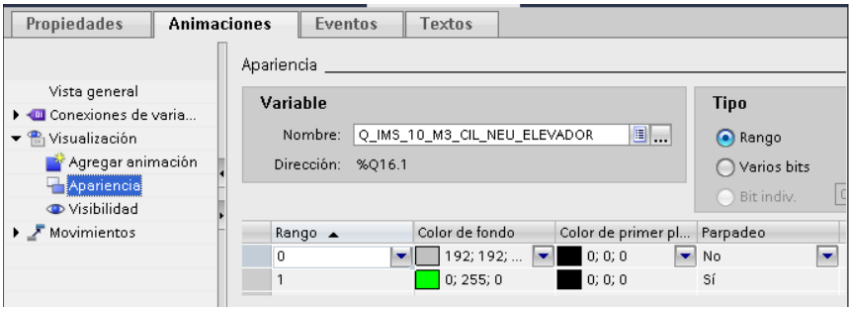
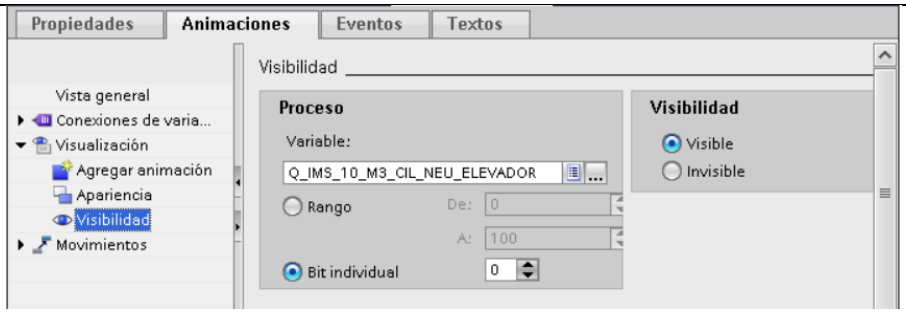
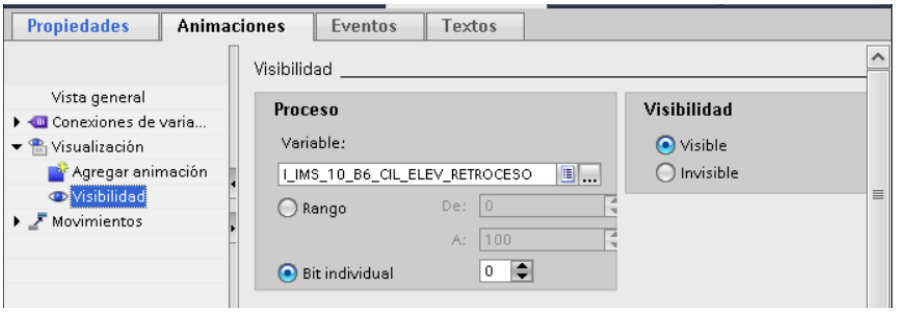
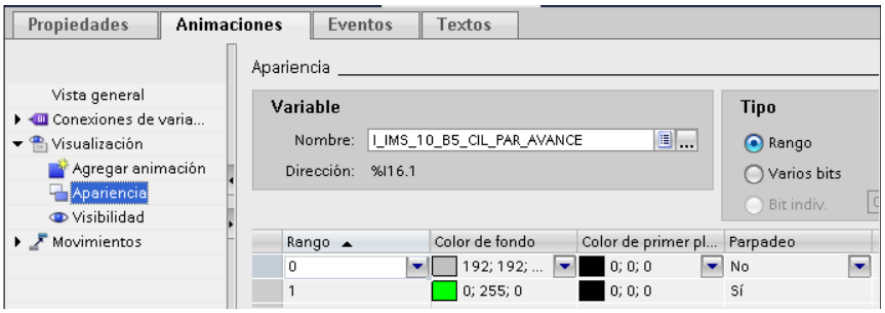
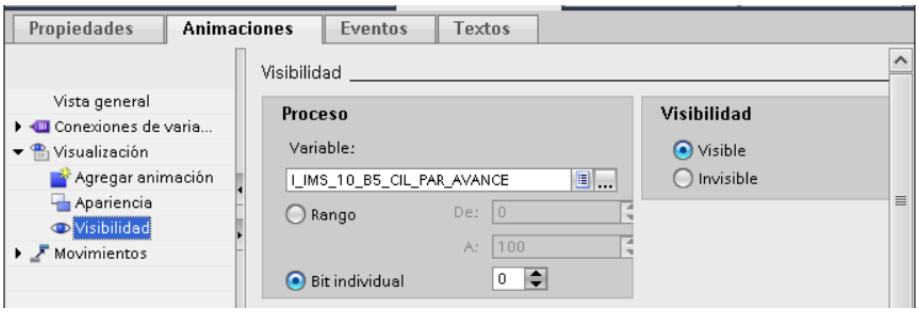
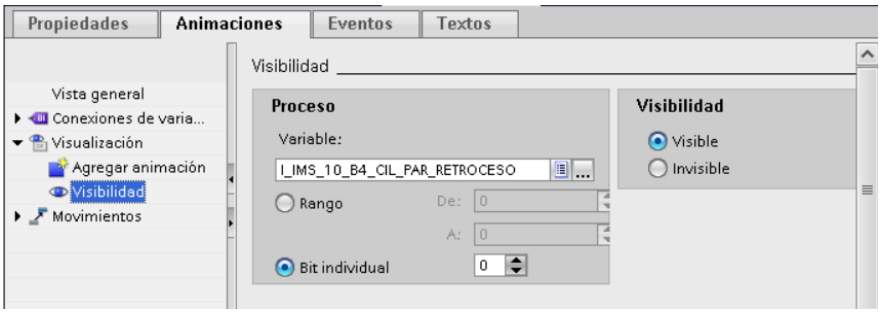


Ilustración 3 Librería de Símbolos pistones neumáticos

LIBRERÍA DE SIMBOLOS	**ANIMACION**

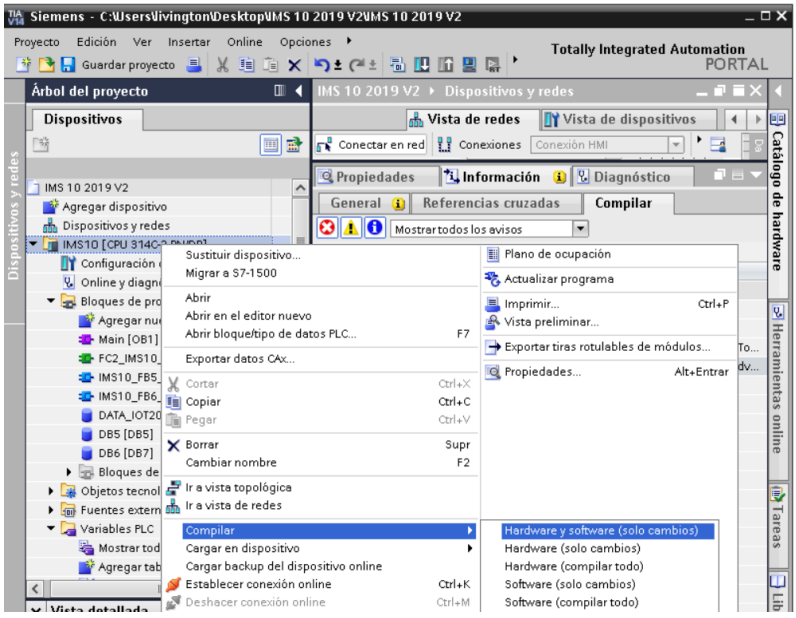
PISTON 1	 
PISTON 2	
PISTON 3	

	
PISTON 4	
PISTON 5	 

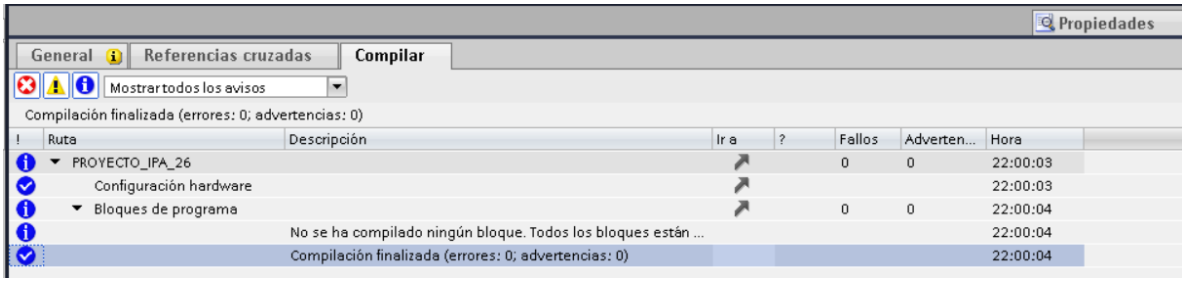
PISTON 6	

Carga del Proyecto al PLC y HMI

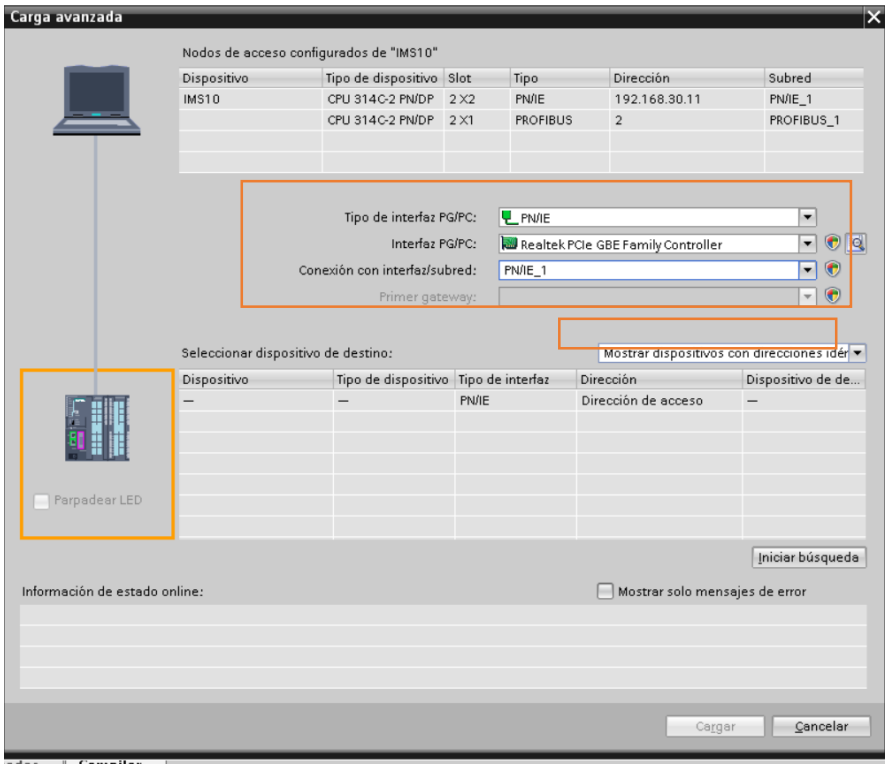
1. Para cargar el proyecto al sistema PLC y HMI, primero debemos compilar Hardware y Software del dispositivo PLC. De la sección de árbol de proyecto seleccionamos la carpeta IMS10 [CPU 314C-2 PN/DP], clic derecho, compilar, Hardware y software (solo cambios).



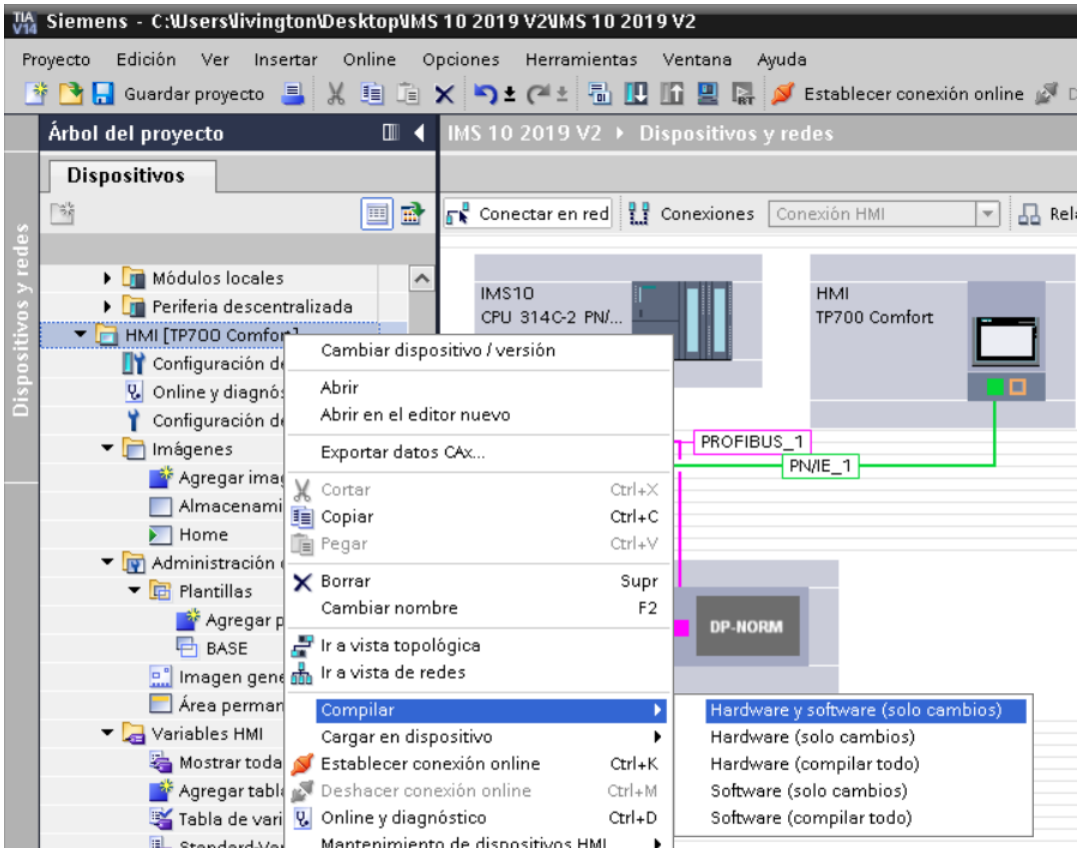
2. En la ventana de propiedades verificamos que no exista error de compilación.



3. Se desplegará la ventana de carga, la cual debe ser configurada como se señala a continuación.



4. Iniciamos la búsqueda del dispositivo y cargamos el proyecto.
5. Para la pantalla HMI haremos lo mismo, seleccionamos la carpeta, clic derecho, compilar, Hardware y Software (solo cambios).



6. Luego damos clic en cargar dispositivos, y aparecerá la ventana de carga, corroboramos lo señalado y cargamos el proyecto a la pantalla.