Universidad Zaragoza

1542

Trabajo de fin de grado

CONTROL DE SISTEMAS DE EVENTOS DISCRETOS CON MATLAB/SIMULINK

CONTROL OF DISCRETE EVENT SYSTEMS WITH MATLAB/SIMULINK

Autor:

Iván Osta Lasheras

Director:

Ramón Piedrafita Moreno

ESCUELA DE INGENIERÍA Y ARQUITECTURA

SEPTIEMBRE 2017

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Iván Osta Lasheras,

con nº de DNI 78757615A en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado en Ingeniería Electrónica y Automática, (Título del Trabajo) CONTROL DE SISTEMAS DE EVENTOS DISCRETOS CON MATLAB/SIMULINK, CONTROL OF DISCRETE EVENT SYSTEMS WITH MATLAB/SIMULINK,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 21 de Septiembre de 2017

IVAN

Fdo: Iván Osta Lasheras

Control de Sistemas de Eventos Discretos con Matlab/Simulink.
Control of Discrete Event Systems with Matlab/Simulink.

RESUMEN

El objetivo principal de este proyecto es analizar las posibilidades que ofrece Simulink y su librería Stateflow para el control de sistemas de eventos discretos y la generación de código para autómatas programables, para ello se utilizará como plataforma de pruebas la célula de Fabricación que se encuentra en unos de los laboratorios del departamento de Informática e Ingeniería de Sistemas de la Universidad de Zaragoza.

La célula de fabricación se compone de 4 estaciones, y una cinta de transporte por la cual circularán los palets. El objetivo global de esta célula, consiste en la producción de cilindros en serie, donde cada una de las estaciones tendrá su función.

La primera estación tiene la labor de colocar el cilindro que se desea fabricar en el palet. La estación 2 se encarga de colocar en la culata un embolo y un muelle. La estación 3 tiene como cometido poner culatas a las piezas. Y, por último, la estación 4 se encarga de la verificación de las piezas y de su colocación en otra cinta de transporte para el posterior almacenaje de estas.

Además de la programación de estas estaciones, se ha creado un programa que engloba a todas ellas, consiguiendo mediante la cinta de transporte una producción de cilindros en serie.

La programación del control de la célula se ha realizado mediante 3 controles diferentes, aunque todos parten de Simulink Stateflow. Los controles son los siguientes

- Control centralizado desde Simulink

Este control se realizará completamente desde Simulink, en él se controlarán cada una de las estaciones y el proceso global. La comunicación con las estaciones se realizará con un servidor OPC, el cual, podrá leer los datos de los sensores y modificar los valores de los actuadores de todas las partes de la célula de fabricación.

Además, para facilitar el control, se dispondrá de la interfaz gráfica de usuaria de Matlab y también se realizará una supervisión por medio de un modelo esquemático en realidad virtual que nos permitirá ver los movimientos de las estaciones a la hora de realizar el proceso de fabricación de las estaciones.

- Control centralizado desde autómata Rockwell

Este control se realizará desde un autómata Rockwell, en el cual se controlará todo el proceso. La Programación se realizará en Simulink y posteriormente será exportada a lenguaje de PLC Rockwell con la librería PLC Coder.

También se incorporó un identificador de productos. Mediante una lectura de la memoria que incorporan los palets, se debe saber la operación que hay que realizar en cada una de las estaciones. Tras la realización de la operación, mediante este identificador de productos, se escribe la operación realizada en la memoria del palet, para que la siguiente estación tenga constancia de ello.

También he incorporado unas pantallas de explotación Magelis. En ellas, se puede seleccionar la pieza que se desea fabricar, además de controlar el funcionamiento de las estaciones y de la cinta de transporte y es posible ver el funcionamiento de las estaciones mediante los indicadores visuales de los sensores y actuadores.

Además, esta terminal servirá de nexo de comunicación entre el autómata Rockwell encargado del control general y el Autómata Schneider la estación 2, ya que dicha estación no puede ser controlada mediante el autómata Rockwell

- Control desde autómatas Schneider

Partiendo de la programación en Simulink, se ha generado código del control para los autómatas Schneider instalados en cada una de las estaciones.

Además de la programación, se ha creado un servidor OPC para la comunicación entre las estaciones para poder realizar el proceso completo dela fabricación con el transporte.

También se creó una interfaz gráfica de usuario de Matlab, con la cual, se podrá dar marcha a cada una de las estaciones individualmente o al proceso global, al igual que indicar que tipo de pieza se quiere fabricar.

Por último, se realizarán comparaciones entre los controles realizados en Simulink y los controles realizados directamente para los autómatas. También se analizarán los tiempos de ejecución de cada uno de los controles en Simulink.

Además, se realizará un análisis del código generado mediante la herramienta PLC Coder. Este código será el utilizado para el control de las estaciones.

INDICE

Índice de Figuras

Índice de Tablas

Tabla de Videos

VIDEO 1: CONTROL DESDE SIMULINK VÍA OPC

HTTPS://1DRV.MS/V/S!AQX4_6XP88_RIDBBWXALN_CUE4DJKQ

VIDEO 2: REALIDAD VIRTUAL ESTACIÓN 3

HTTPS://1DRV.MS/V/S!AQX4_6XP88_RIDBAGK3CQGBWILSDSG

VIDEO 3: REALIDAD VIRTUAL ESTACIÓN 1

HTTPS://1DRV.MS/V/S!AQX4_6XP88_RIDA_LNCA_Q7ILWTXQW

VIDEO 4: CONTROL DESDE PLCS SCHNEIDER DESCENTRALIZADO

HTTPS://1DRV.MS/V/S!AQX4_6XP88_RIDA-FQBUCZWEA0OMRA

VIDEO 5: CONTROL DESDE PLC ROCKWELL CENTRALIZADO

HTTPS://1DRV.MS/V/S!AQX4_6XP88_RIDA9N48GXULTRCE8HG

VIDEO 6: TIEMPOS DE EJECUCIÓN EN PLC SCHNEIDER

HTTPS://1DRV.MS/V/S!AQX4_6XP88_RIDA870WD_-KFJFYG-W

VIDEO 7: TIEMPOS DE EJECUCIÓN EN PLC ROCKWELL

HTTPS://1DRV.MS/V/S!AQX4_6XP88_RIDA7CE7L9UIYYO7RHQ

1 Introducción.

En este capítulo se van a encuadrar los aspectos principales de esta memoria, como puedan ser sus objetivos, alcance, contenido y siglas empleadas en la misma.

1.1 Objetivos

El objetivo principal de este proyecto es comprobar las posibilidades y funcionalidades que tiene Matlab y Simulink a la hora de implementar controles en el ámbito industrial, aprovechando las librerías que incorpora Matlab.

Los objetivos específicos que se plantean alcanzar en este proyecto son los siguientes:

- Desarrollo de máquinas de estados para poder controlar cada una de las estaciones en Simulink a través de un Stateflow (Entorno para modelar y simular máquinas de estado y diagrama de flujos).
- Realizar el control centralizado la célula de fabricación desde Simulink, mediante un sistema de comunicación mediante un servidor OPC el cual comunicará Simulink con el autómata Schneider.
- Realizar el control centralizo desde Rockwell, manejando toda la célula de fabricación junto con el identificador de producto y un terminal magelis desde un único autómata programable.
- Realizar el control descentralizado desde los autómatas Schneider instalados en las estaciones realizando la comunicación entre ellas mediante un servidor OPC
- Generación de código mediante la herramienta PLC Coder de Simulink para el PLC. adaptando las máquinas de estados para posibilitar la generación de código para PLC de Schneider y de Rockwell, analizando su código y exportándolo a Unity Pro y RSLogix, realizando los ajustes necesarios para conseguir su total funcionalidad.
- Creación de pantallas de explotación mediante la Toolbox “GUIDE”, haciendo más fácil e intuitivo el manejo de la célula de fabricación, pudiendo poner en marcha el proceso y la elección de la pieza a fabricar.
- Implementación de un modelo de realidad virtual en 3D en Matlab cuyo objetivo será supervisar el proceso de Fabricación en el control desde Simulink.
- Comparación de los controles realizados mediante Stateflow y PLC Coder con los controles implementados directamente en los PLCs en lenguaje SFC.

1.2 Alcance

A continuación, se muestran los métodos de control utilizados en este proyecto. En primer lugar, se muestra el control desde Simulink, encargado de ejecutar el control realizando los cálculos necesarios para obtener la respuesta deseada.

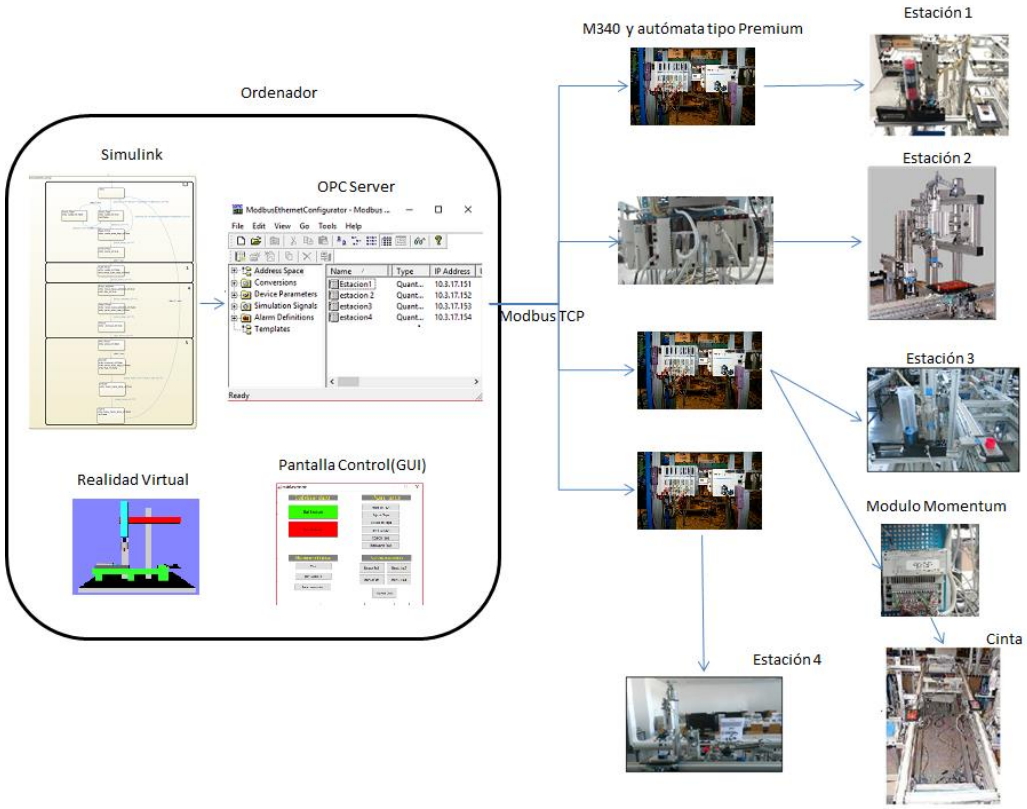


Figura 1.Esquema Control desde Simulink

En segundo lugar, se muestra el esquema cuando todo el control se realiza desde el PLC de Rockwell. Simulink solo se utilizará para implementar las máquinas de estados para los controles de las estaciones, los cuales serán importados al PLC.

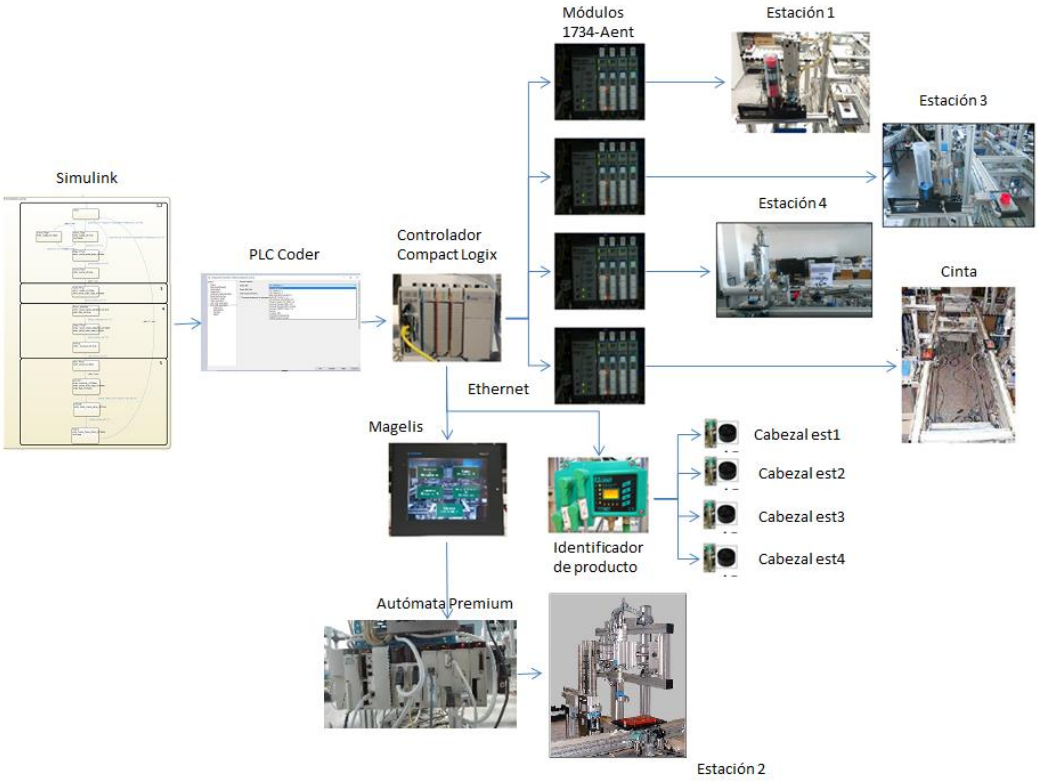


Figura 2.Esquema Control desde PLC de Rockwell

Por último, se muestra el esquema cuando el control se realiza desde el PLC de Schneider instalados en cada una de las estaciones, en este caso, desde Simulink solo se han tratado las operaciones realizadas desde la pantalla, los datos de comunicación entre estaciones y los datos para la supervisión con el modelo en 3D. Es en el PLC donde se encuentran los controles, de la célula, los cuales se han implementado también en Simulink.

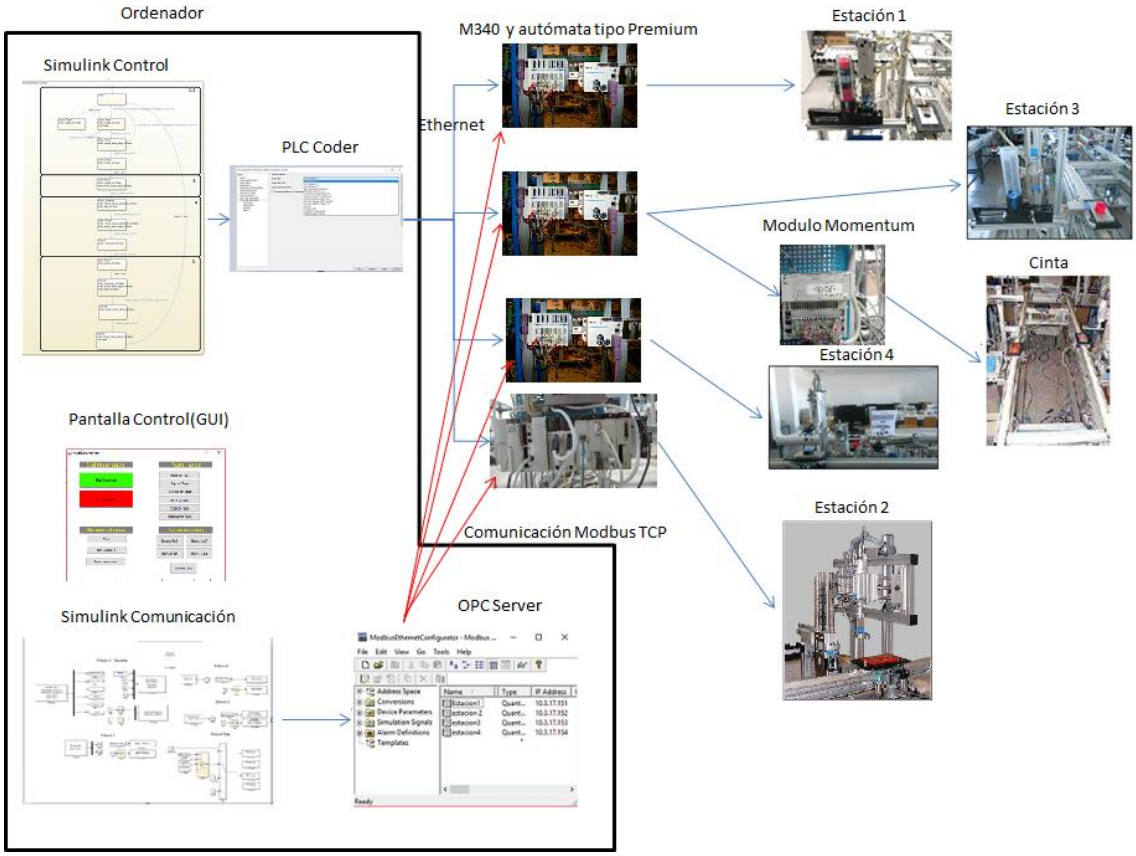


Figura 3.Esquema Control desde PLC de Schneider

1.3 Contenido de la memoria

La memoria está estructurada en 10 capítulos:

Capítulo 1- Introducción. Se indican los objetivos principales del proyecto, su alcance, su estructura y las siglas empleadas en el mismo.

Capítulo 2- Descripción General de la célula de fabricación. Se describen todas las estaciones, se explica el funcionamiento de cada una de ellas en el proceso de fabricación. Además, también se describirán los diferentes modos de funcionamiento para las estaciones

Capítulo 3- Hardware y Software. Se explicarán los diferentes programas y PLCs que se utilizarán a lo largo del proyecto.

Capítulo 4-. Programación en Stateflow y Generación de Código. Se describirán los elementos disponibles en la herramienta Stateflow para programar. Además de cómo,

realizar la importación de un modelo realizado en Simulink mediante la herramienta PLC Coder

Capítulo 5- Control desde Simulink mediante OPC. Se realizará el control de la célula de fabricación con la programación en Stateflow de Simulink.

Capítulo 6- Control desde Rockwell. Se realizará el control de la célula de fabricación con la programación importada desde Simulink para un autómata de Rockwell.

Capítulo 7- Control desde Schneider Se realizará el control de la célula de fabricación con la programación importada desde Simulink para autómatas de Schneider.

Capítulo 8- Comparación controles y análisis del código generado. Se comparará la ejecución de los controles realizados a lo largo del trabajo mediante Stateflow y PLC Coder con otros controles programadas en SFC en los PLCs.

Capítulo 9- Conclusiones finales. La memoria finaliza con un capítulo en el que se presentan las conclusiones finales del proyecto, realizando un análisis de los objetivos establecidos junto con su cumplimento o sus defectos y proponiéndose posibles mejoras a futuro o líneas de trabajo

1.4 Lista de acrónimos

PLC: Programmable Logic Controller (Controlador Lógico Programable o también Autómata programable).

GUI: Grafic User Interface (interfaz gráfica de usuario)

SFC: Secuential Function Chart

IdP: Identificador de Producto

2 Descripción General de la célula de fabricación

2.1 Introducción

Antes de explicar todo el proceso que se ha realizado para la implementación de los diferentes modos de control de la célula de fabricación, se va a explicar la función de cada una de las estaciones en el proceso de fabricación.

La célula de fabricación está compuesta por dos módulos principales y uno de unión. El primer módulo es el encargado de la fabricación y verificación, el módulo de unión se encarga de la identificación y clasificación y, por último, el módulo II, es el encargado del montaje de los pedidos y de su almacenamiento final.

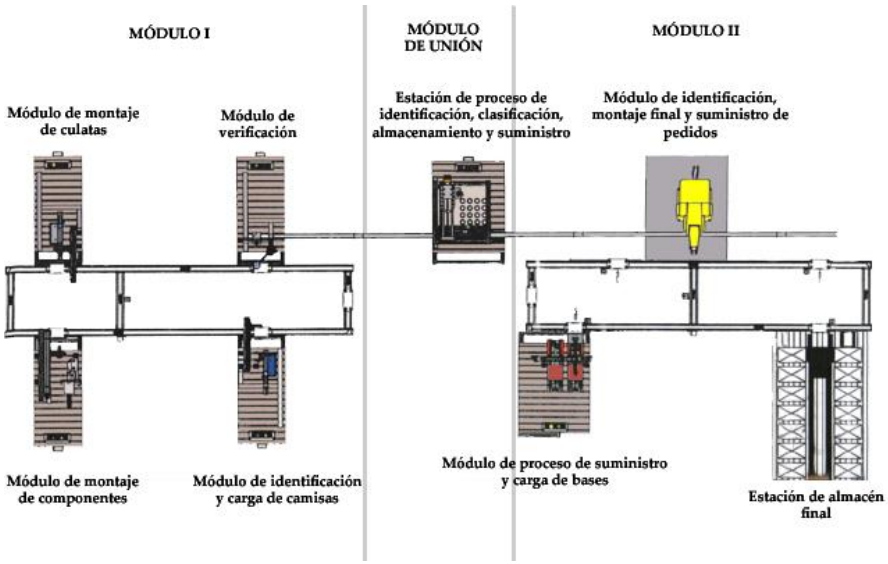


Figura 4.Esquema general de la célula de fabricación

2.2 Partes de la célula de fabricación.

El montaje de las piezas se realizará por medio de la acción conjunta de todas las estaciones que componen la célula de fabricación. Cada una de las estaciones posee una función determinada en el proceso para conseguir montar la pieza con las características adecuadas.

Las estaciones programadas llevan a cabo el ensamblado de los diferentes elementos que componen la pieza a fabricar en cada momento:

- **Transporte:** Encargado del traslado de las piezas entre las estaciones por medio de una cinta transportadora.
- **Estación 1:** Encargada de la colocación de la camisa deseada en el palet de la cinta transportadora.
- **Estación 2:** Encargada de colocar el émbolo y el muelle en las piezas que así lo requieran. La colocación del émbolo se realiza de forma acorde al tipo de pieza tratada en cada momento.
- **Estación 3:** Esta estación coloca y rosca la tapa a las piezas que así lo requieran.
- **Estación 4:** Encargada de recoger la pieza del palet y dirigirla a almacenaje.

2.3 Piezas a fabricar

Los diferentes tipos de piezas que es posible fabricar se pueden dividir en dos grupos diferentes. Por un lado, los cilindros neumáticos (sin Tapa) y por otro los cilindros cerrados (con tapa).

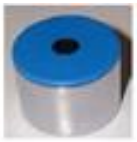
Piezas	Pieza con Tapa	Pieza sin Tapa
Negra		
Roja		
Metálica		

Tabla 1.Diferentes tipos de pieza que existen en la célula.

Como se puede ver en la tabla, dentro de un mismo tipo de piezas hay 3 posibilidades de las mismas según el color que posea cada una de ellas.

Además, cuando se vaya a producir una pieza sin tapa, habrá que distinguir 2 procesos diferentes en la estación 2 en función del color de la culata.

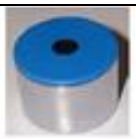
Tipo de Pieza			
Camisa	**Negra**	Roja	Metálica
Embolo	Metálico  (longitud Corta)	Negro  (longitud larga)	
Muelle	**Estándar**		

Tabla 2.Piezas que formas las piezas sin tapa.

Una vez descritas cada una de las estaciones se debe indicar que según la pieza que se está fabricando, la pieza deberá pasar por un recorrido u otro de la zona de fabricación.

Recorrido	Pieza con Tapa	Pieza sin Tapa
Indiferencia de color	Estaciones 1 y 4	Estaciones 1 ,2, 3 y 4

Tabla 3.Plan del recorrido en la célula dependiendo del tipo de pieza

2.4 Modos de Funcionamiento de las estaciones

2.4.1 Modo Manual

En este modo, cada una de las estaciones es controlada de forma manual. Ya que las botoneras no están preparadas para asumir el control de todas las funciones de que disponen las estaciones, este modo se realizará desde una terminal Magelis.

2.4.2 Modo Test

Este modo realiza una secuencia para que se pueda detectar cualquier tipo de fallo de ejecución. Además, también permite ver los movimientos que se realizan en cada estación, así como el correcto funcionamiento de todos los sensores y actuadores.

2.4.3 Modo Automático Local

En este modo de funcionamiento las estaciones pueden realizar su producción sin tener en cuenta al resto de estaciones, ni al transporte. Los mandos que indican a la estación que tipo de producción debe realizar y cuando la debe realizar se encuentran en la magelis.

2.4.4 Modo Automático Integrado

En modo automático integrado, la célula realiza la producción en conjunto y obtendrán productos terminados. Para que las estaciones trabajen de forma autónoma, entran en juego las cintas transportadoras y las memorias de los palets.

2.5 Descripción de las Estaciones y Transporte

2.5.1 Estación 1

El objetivo de la estación 1 es la colocación de la camisa deseada en el palet de la cinta transportadora. Como se puede observar, la estación consta de dos partes principales:

- Almacén de cilindros: tiene forma cilíndrica y en él se situarán las culatas.
- Brazo: puede desplazarse tanto horizontalmente (hacia adelante o hacia atrás) como verticalmente (arriba o abajo) como lateralmente (izquierda o derecha). En su parte inferior se localiza una pinza encargada de coger o soltar los cilindros.

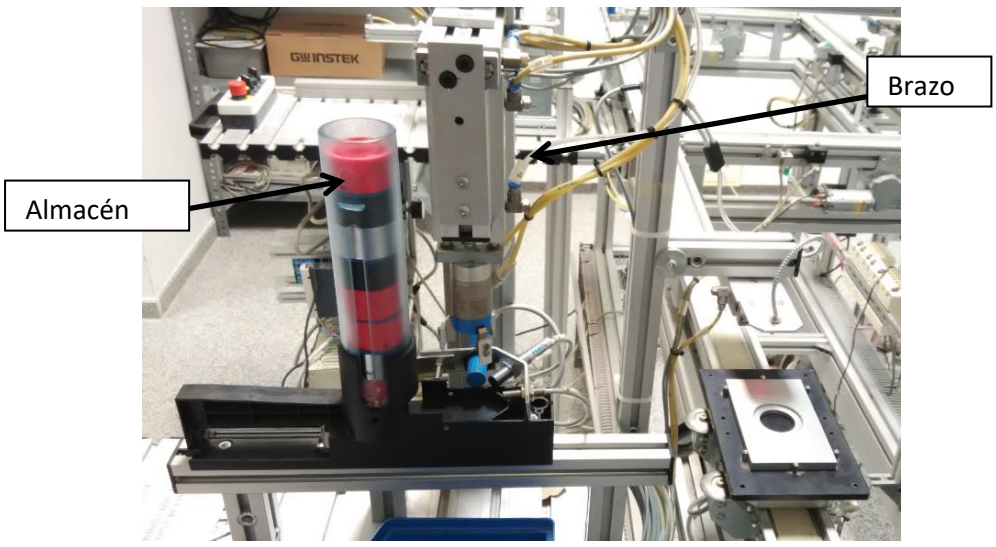


Figura 5.Vista general estación 1

Se dispone de 4 sensores para reconocer la pieza. Los 3 primeros nos las separaran según sean negras, rosas o metálicas. El último sensor nos indicará si lleva tapa o si no lleva.

Tipo Pieza	S. Inductivo	S. Óptico	S Capacitivo	S. Óptico (tapa)
Negro sin tapa	0	0	1	0
Roja sin tapa	0	1	1	0
Metálica sin tapa	1	1	1	0
Negra con tapa	0	0	1	1
Roja con tapa	0	1	1	1
Metálica con tapa	1	1	1	1

Tabla 4.Estado de los sensores identificadores de camisas y tapa

Si el color de la pieza corresponde con la pieza requerida, se depositará encima del palet, en caso contrario, el cilindro será llevado a un contenedor azul que está colocado a la derecha de la estación. Después se comprueba si la pieza se corresponde a la requerida en cuanto a tener o no tapa. Si no corresponde, el cilindro se llevaría de vuelta al contenedor azul.

2.5.2 Estación 2

Esta estación es la encargada de realizar el montaje de los elementos internos que componen los cilindros neumáticos fabricados. Por tanto, se encargará de colocar el émbolo adecuado para el tipo de pieza fabricada en cada momento, así como el muelle que llevan dichos elementos. En la siguiente figura, se muestra una imagen de la estación 2.

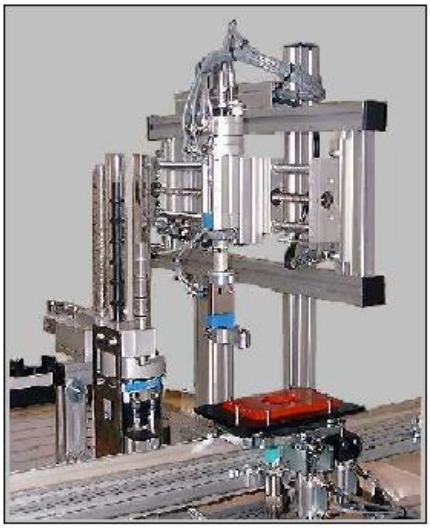


Figura 6 Vista general estación 2

En esta estación tiene tres almacenes de elementos. Uno para los émbolos negros, otro para los émbolos metálicos y otro para los muelles. Así la estación recogerá las piezas necesarias de cada uno de los almacenes en función de la pieza que se esté fabricando en cada momento. Como es lógico, la estación colocará sobre la camisa ya colocada en el trasbordador, en primer lugar, el émbolo apropiado para, posteriormente colocar el muelle.

La principal característica de esta estación, es que a diferencia de las otras donde todos los actuadores eran neumáticos, ahora se tiene como actuador un motor paso a paso. Para su control, se dispone de un módulo que conectado en el autómata controla el motor.

2.5.3 Estación 3

La misión principal de la estación 3 es la colocación de las culatas (tapas) en los cilindros correspondientes. Dichas culatas tienen forma circular e incluyen un orificio como salida del émbolo. Esta misión será realizada en el modo automático de funcionamiento, aunque como ya se explicó también están implementados el modo manual, test y posicionamiento inicial.

Como se puede observar, la estación consta de tres partes principales:

- Almacén de culatas: tiene forma cilíndrica y en él se situarán las culatas, dispuestas verticalmente una encima de otra.
- Brazo: puede desplazarse tanto horizontalmente (hacia adelante o hacia atrás) como verticalmente (arriba o abajo). En su parte inferior se localiza una pinza encargada de coger o soltar las culatas y gracias a la cual, mediante su giro, se roscarán las culatas
- Abrazadera o mordaza: se encarga de sujetar el cilindro para que no haya movimiento alguno cuando se rosque la culata.

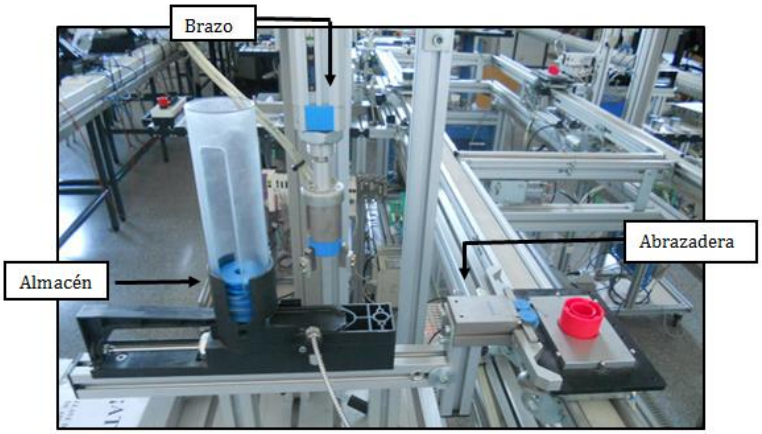


Figura 7. Vista general estación 3

2.5.4 Estación 4

La principal función de la estación 4 es la verificación de cilindros, para ello se debe provocar primero la salida del émbolo, en caso de que la pieza lo tenga.

Aquellas piezas que tengan tapa no deberán pasar la verificación puesto que no tienen émbolo ya que siempre darían como resultado que son pieza defectuosa, por tanto, se mandarían directamente sin realizar la verificación.

La estación consta de tres partes principalmente:

- Brazo: está equipado con dos ventosas y se encargará de recoger las piezas para llevarlas al verificador.
- -Verificador: mediante la inyección de aire se conseguirá medir el recorrido del émbolo para comprobar la validez de la pieza.
- -Brazo giratorio: recoge la pieza del verificador y la lleva hasta la cinta, en caso de ser válida pasará, de lo contrario se desechará.

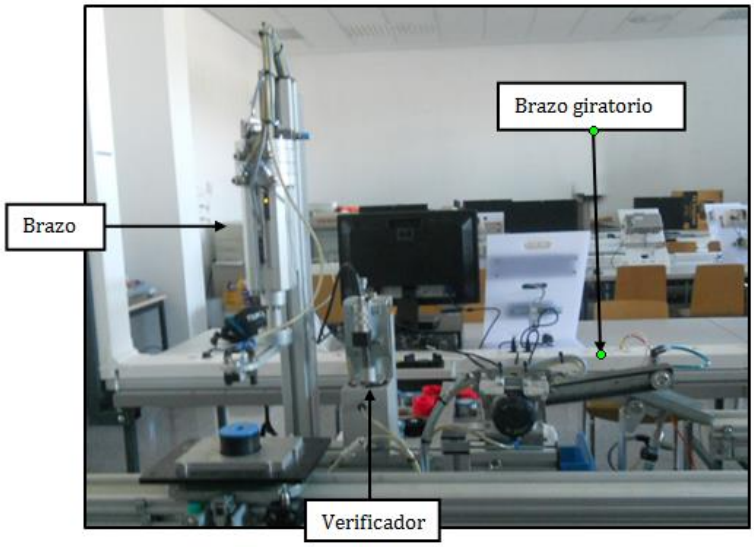


Figura 8. Vista general estación 4

2.5.5 Transporte

Para llevar las piezas desde una estación a otra y para que cada una de ellas realice su función se dispone de la cinta de transporte. Una serie de cintas transportadoras, enclavamientos y topes aseguran la circulación o detención de cada uno palets afectados.

Figura 9.Cinta de transporte célula de fabricación

Para el desplazamiento de las piezas a lo largo del recorrido se usan unos transbordadores sobre los cuales se coloca un palet de transporte metálico. Sobre dicho elemento se situará una base plástica dotada de una única hendidura en su parte central en la cual se adaptan perfectamente las piezas a fabricar. De este modo las piezas podrán ser trasladadas de estación a estación de forma segura.

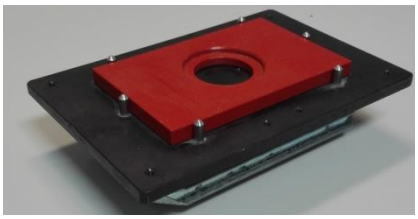


Figura 10.Palet

Cada uno de los transbordadores posee en su parte inferior una pequeña memoria en la cual se podrá almacenar la información de la mercancía que transporta. Por medio de una serie de cabezas lectoras del identificador de producto se podrán realizar lecturas y escrituras de información.

En cada una de las estaciones hay un tope que se encargará de detener el trasbordador frente a la entrada de cada una de ellas. Posteriormente, y por medio de un enclavamiento, se fijará el trasbordador para mantenerlo fijo en la posición correcta. Así las cabezas lectoras podrán realizar sus operaciones de lectura y escritura.

Figura 11.Enclavamiento, Tope, Sensor I.P

Como se puede observar en la figura, De izquierda a derecha aparece en primer lugar la cabeza lectora que leerá la información de la memoria del trasbordador. A continuación, el enclavamiento que fija el palet en la posición correcta, para que no se desplace. Por último, se encuentra el tope que se encarga de detener o liberar el palet según corresponda.

3 Hardware y Software

3.1 HARDWARE

3.1.1 Autómata programable Rockwell

Figura 12.Controlador Compact Logix 1768-L32E

El autómata programable Rockwell utilizado será Controlador Compact Logix 1769-L32E. Este controlador puede soportar 8 tareas ejecutadas simultáneamente con su memoria volátil 1784-CF64 Compact Flash y con su capacidad de 750 Kb. Además de esto, también puede soportar hasta 16 módulos de entradas y salidas. Tiene 2 puertos de comunicación un puerto Ethernet/IP y un puerto RS232 serie los cuales se utilizarán en el proyecto.

-Modos de funcionamiento del controlador

El controlador tiene un conmutador de llave de tres posiciones situado en la parte frontal que dirige los modos de funcionamiento del controlador. Se pueden seleccionar los modos siguientes:

En marcha (Run)

- Cargar proyectos
- Ejecutar el programa y habilitar las salidas
- En dicho modo no se podrán crear o borrar tareas, programas o rutinas. Del mismo modo no se podrán borrar o crear etiquetas o editar en línea.

Programar (Prog)

- Deshabilita las salidas
- Cargar o descargar proyectos
- Crear, modificar o borrar tareas, programas y rutinas
- El controlador no ejecutará tareas mientras se mantenga dicho modo de ejecución

Remoto (Rem): este modo es habilitado por software, y puede ser Program (Programa), Run (Marcha) o Test (Prueba).

- Cargar o descargar proyectos
- Cambiar entre programación remota, prueba remota y los modos de ejecución remotos a través del software de programación.

3.1.2 Identificador de Producto

Figura 13.Identificador de Producto IC-KP-B12-V45

El identificador de productos IC-KP-B12-V45 será el encargado de escribir mediante RFID en las memorias de los palets para saber que producto lleva el palet, y que operaciones se han realizado en él.

Para ello, los palets que recorren la cinta transportadora llevan en su parte inferior unos chips de memoria en el que está guardada la información de la pieza que transportan. Cada estación de trabajo dispone de una cabeza lectora que, mediante su correspondiente instrucción, puede leer o escribir sobre la memoria de los palets. Estas unidades lectoras están todas conectadas al identificador de productos, el cual se comunicará con el autómata.

Figura 14. Cabeza lectora y Chip de memoria

El identificador de producto se puede comunicar con un máximo de 4 cabezales, uno por estación, con unas velocidades de 10 MBits/s o 100 MBits/s dependiendo de los requisitos. Puede utilizar diferentes protocolos como SMTP, HTTP o MODBUS/TCP.

3.1.3 Módulo de entradas/salidas por Ethernet

Este módulo se encuentra tanto en las estaciones 1, 3, 4 y en el transporte, y nos permitirá transmitir la información necesaria para controlarlas estaciones mediante el protocolo de redes Ethernet/IP.

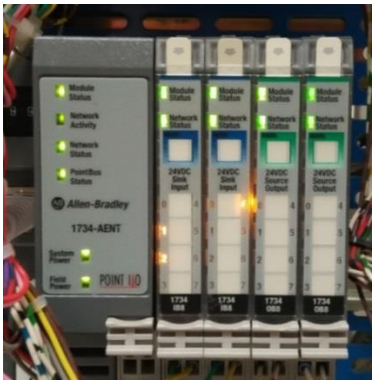


Figura 15.Adaptador Ethernet/IP 1734-Aent

El cual tiene varios módulos acoplados de entradas y salidas digitales

- Módulos de entradas digitales (1734-IB8/C y 1734-IB4/C)
- Módulos de salidas digitales (1734 OB8E)

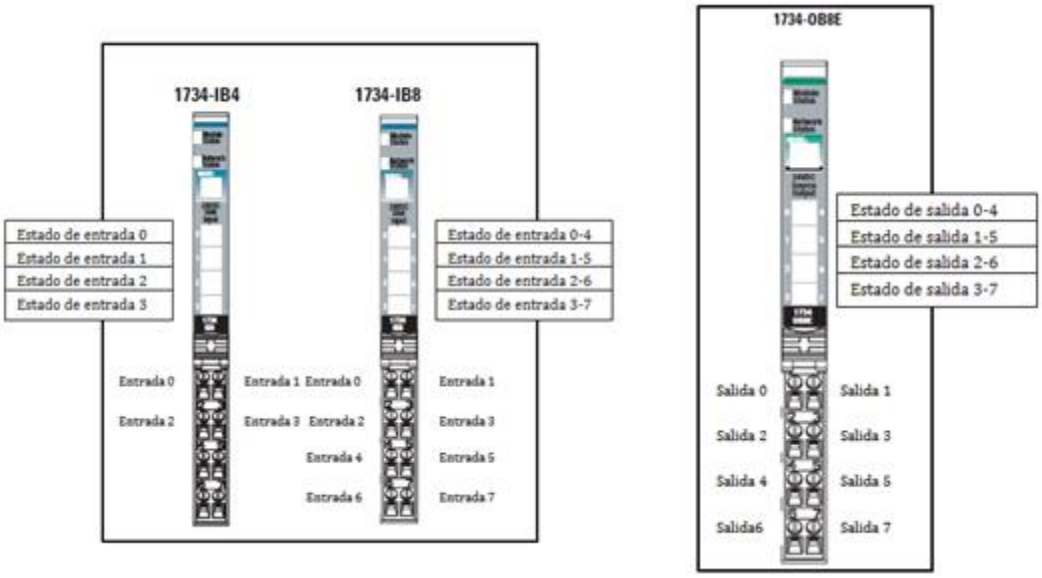


Figura 16.Módulos de entradas/salidas digitales

Estos módulos trabajan con una tensión de 24V y 4mA nominales. Con cada uno de ellos se podrá controlar de 4 o 8 variables en función del modelo del módulo.

3.1.4 Magelis

La familia de terminales de diálogo Magelis va asociada al software Vijeo Designer. En este proyecto se escogió la Magelis XBTGT4330, debido a que es la que se tiene disponible en el laboratorio.

Figura 17.Magelis XBTGT4330

Se caracteriza por su comodidad de uso y la facilidad de funcionamiento debido a su pantalla táctil de 7.5" y su fácil instalación. Además, tiene diferentes puertos de comunicación que le permiten conectarse con sus periféricos

-Ventajas de usar Magelis

- Visibilidad perfecta en sus aplicaciones
- Toda la familia de nuevos terminales gráficos Magelis XBT Cts. están ahora disponibles en muchos tamaños de pantalla desde 3.8'' a 15''
- Abierto, usted elige el tamaño que se adapte a sus necesidades y comunicarse libremente.
- Compactos, simples y robustas, etc. GTs Magelis XBT unen fuerzas para mejorar su productividad.

-Aplicaciones de la Magelis XBTGT4330

- Industria: máquinas compactas, manejo de sistemas, máquinas de alimentos y bebidas, etc.
- Infraestructura y automatización en edificios

3.1.5 Autómata Schneider

Autómata tipo Premium TSX P57 202

Este controlador nos permitirá el control de la estación 2 para cualquiera de los tres modos de control que se ha van a realizar en el proyecto.

EL autómata tiene la siguiente configuración hardware:

- Fuente de alimentación TSX PSY5500M
- Autómata tipo Premium TSX P57 5634M
- Módulo de entradas TSX DEY 32D2K
- Módulo de salidas TSX DSY 32T2K
- Módulo configurable de Ethernet TSX ETY 5103

- Módulo de control del motor paso a paso TSX CFY11
- Módulo de Interbus TSX IBY 100
- Módulo PCMCIA TSX SCY 21601

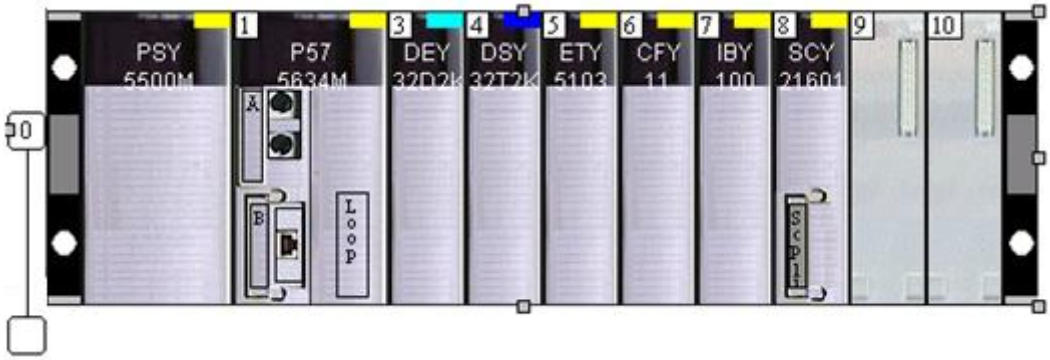


Figura 18.Configuración Hardware de la Estación 2

Modicon M340 BMX P34

El autómata Schneider utilizado en el resto de estaciones será Modicon M340 BMX P34. Este autómata está diseñado para controlar procesos secuenciales, recoger datos de los sensores y generar datos de salida que actuaran sobre dispositivos externos. También es capaz de almacenar datos en su memoria y leerlos para modificarlos si es preciso-

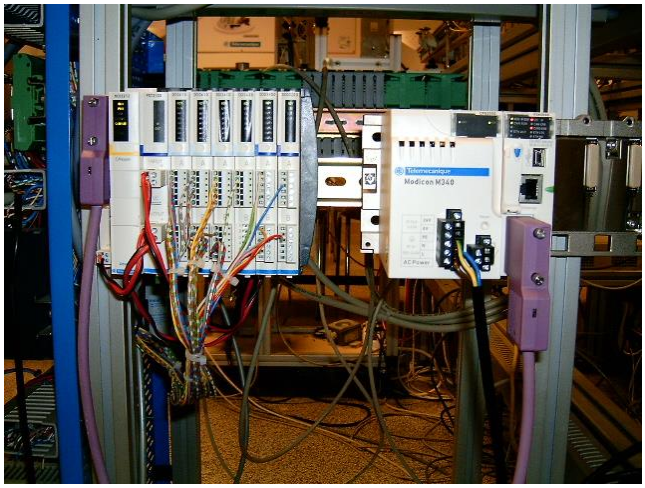


Figura 19.Vista frontal autómata Modicon M340 BMX P34 2030

A continuación, se explicará más detenidamente en las partes que acompañan al Modicon M340 BMX P34 2030 y en sus propiedades.

El bastidor utilizado es BMX XBP, el cual nos permite fijar todos los módulos de la estación del autómata y proporciona la alimentación a los módulos acoplados al procesador.

La fuente de alimentación utilizada en este autómata es una BMX CPS 2000, con alimentación mediante corriente alterna y con las siguientes características:

	Características
Tensión nominal	100 – 120 V/200 – 240 V
Tensión límite	85 – 264 V

Frecuencia límite/nominal		50-60 Hz/47-63 Hz
Potencia aparente		70 VA
Irms de corriente nominal absorbida		0,61 A a 115 V 0,31 A a 240 V
Conexión inicial a 25 ºC (1)	Corriente de señalización I	30 A a 120 V 60 A a 240 V
	I2t En el bloqueo	0,5 A2s a 120 V 2 A2s a 240 V
	It En el bloqueo	0,03 As a 120 V 0,06 As a 240 V
Duración aceptada de los microcortes		10 ms
Protección integrada en la fase	En la interna, fusible sin acceso	

Tabla 5.Caracteristicas Fuente alimentación BMX CPS 2000

3.1.6 Adaptador de comunicaciones Interbus

El adaptador de comunicación Interbus 170 INT 110 00 proporciona la comunicación entre el adaptador y la red Interbus. Este adaptador se puede conectar a cualquier base de E/S Momentum para crear una unidad de E/S funcional en la Red Interbus. Este adaptador nos permitirá controlar la cinta d transporte con un puerto Ethernet desde una estación.

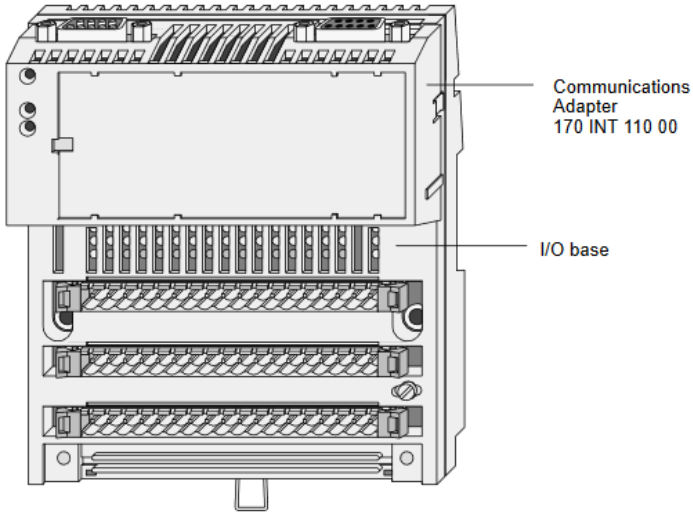


Figura 20.Momentum Interbus Communication Adapter

La red Interbus es una red maestro/esclavo diseñada para el intercambio eficiente de datos de E/S. Tiene la capacidad de comunicarse con hasta 256 dispositivos y leer y escribir 1024 variables en 4 ms sin comprometer el rendimiento del sistema ni la fiabilidad de los datos de E/S.

3.2 SOFTWARE:

En el proyecto se han utilizado distintos programas a la hora de realizar cada una de las tareas. A continuación, se explicará con más detalle los programas utilizados a lo largo del proyecto.

3.2.1 RSLOGIX 5000

Con el software de programación RSLogix 5000, sólo se necesita un paquete de software para controlar o programar aplicaciones discretas, de procesos por lotes, el movimiento de estas, la seguridad y la unidad base. El software RSLogix 5000 ofrece una solución fácil de usar, IEC61131-3 compatible con la interfaz, programación con estructuras y arreglos y un conjunto de instrucciones que es de gran utilidad para ejecutar aplicaciones variadas.

- ***Características de RSLogix 5000***

Dentro de los diferentes lenguajes de programación de los cuales nos permite hacer uso RSLogix están:

- Lenguaje estructurado (ST)
- Diagrama de bloques de función secuencial (FBD)
- Lenguaje Grafcet (SFC)
- Lenguaje de contactos (LD)

Además, el programa también ofrece el programa en lo referente a la programación:

- Configuración de los equipamientos necesarios para llevar a cabo el control de la maquinaria (módulos de entradas y salidas, adaptador de Ethernet, autómatas con los cuales se trabajan...)
- Monitorización y seguimiento de las variables (activación o no de las salidas, en caso de ser digitales "0" o "1", o su valor concreto si son analógicas) así como del estado en el cual se encuentra la programación (etapa).
- Configuración de todo lo referente a la programación (tanto la zona de código como la declaración de variables, definición de entradas y salidas...).

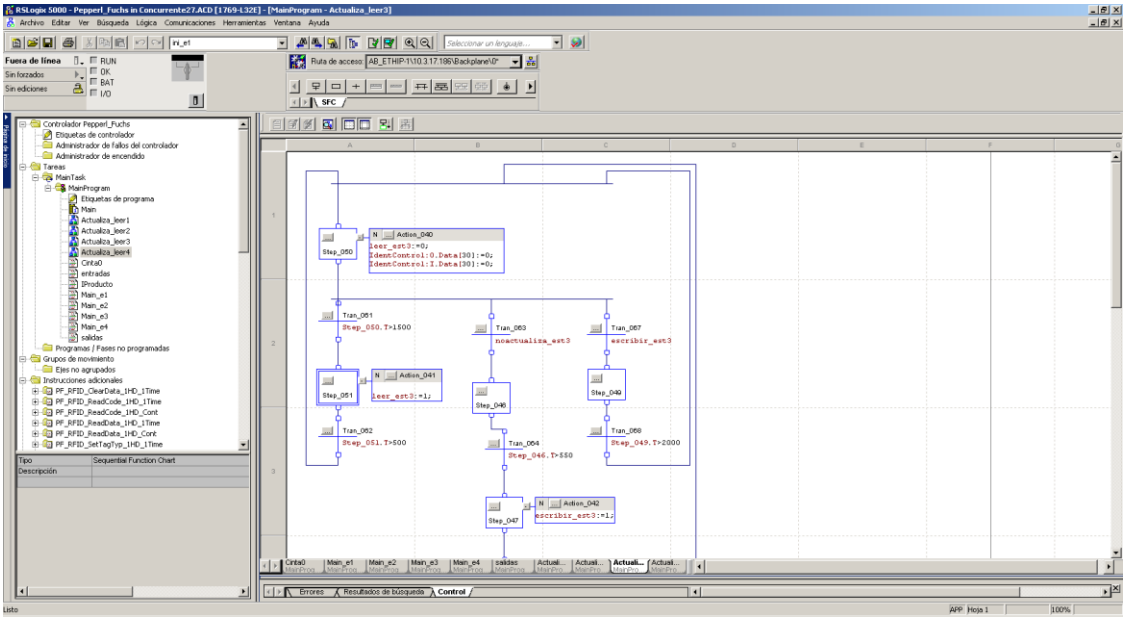


Figura 21.Programa RSLogix 5000

3.2.2 Unity pro XL

Unity Pro es un software común de programación, puesta a punto y explotación de los autómatas Modicon. Unity Pro tiene muchas posibilidades debido a que ofrece la utilización de las variables, direcciones y bits de palabras.

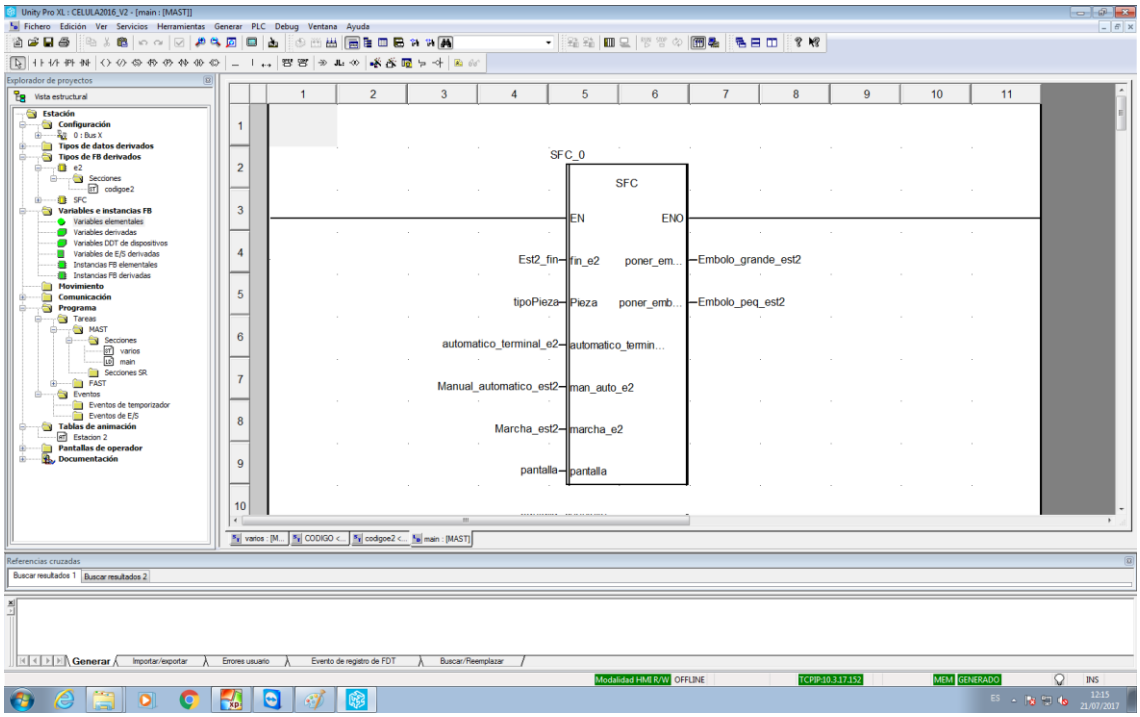


Figura 22.Unity Pro XL

-Descripción y características del software

Dentro del software Unity Pro XL se puede generar proyectos en varias familias de autómatas, como son Modicon M340, Premium, Quantum y Atrium. Todos ellos dentro de la marca Schneider Electric.

Las ventajas de la estandarización en Unity Pro ocasionan que un proyecto se pueda dividir en módulos funcionales que facilitan la implementación y la utilización en otros autómatas de la misma familia

3.2.3 Vijeo Designer

Vijeo Designer es un software de programación destinado para el diseño de aplicaciones para Magelis. De fácil uso, gracias a su interfaz intuitiva, ofrece además funciones avanzadas tales como aplicaciones multimedia y acceso remoto para más eficiencia.

-Funciones de Vijeo Designer

- Función de visualización y grabación de video
- Visualización de cualquier tipo de imagen y video en tiempo real desde una cámara
- Función de acceso remoto Web Gate: maneje sus aplicaciones HMI, a través de un simple navegador de internet usando arquitectura Ethernet.

-Aplicaciones

- Fabricantes de máquinas, sencillas o complejas (automoción, componentes electrónicos, productos farmacéuticos, industria química).
- Sector terciario e infraestructura: construcción, industria alimentaria, tratamiento de aguas y residuos.

3.2.4 ICONICS OPC Server

El Servidor OPC hace de interfaz entre el ordenador y los autómatas de las estaciones- En una arquitectura Cliente OPC/ Servidor OPC.

Las comunicaciones entre el Cliente OPC y el Servidor OPC son bidireccionales, lo que significa que los Clientes pueden leer y escribir en los dispositivos a través del Servidor OPC.

En el proyecto utiliza un OPC server de Iconics, el cual nos permitirá conectarnos con los autómatas Schneider de las estaciones.

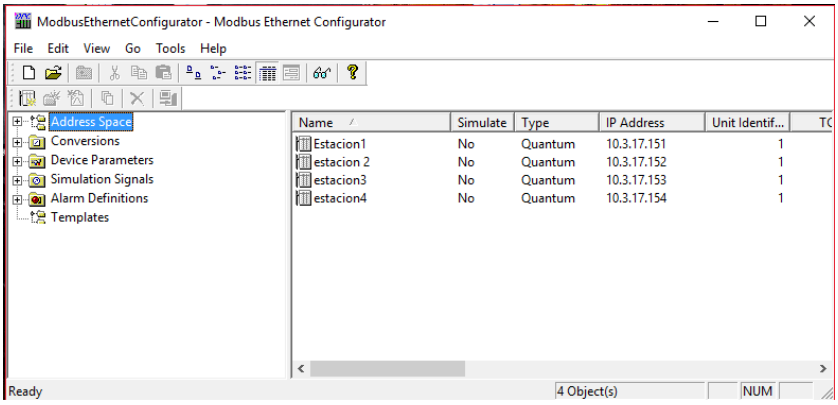


Figura 23.Pantalla Principal Programa OPC Server

3.2.5 OMRON Sysmac Studio

Sysmac Studio es un entorno de desarrollo de Omron en el cual se realizarán las funciones de configuración, programación, simulación y monitorización de un proceso. Esta herramienta de software tiene una programación sencilla ya que el propio programa es el que asigna automáticamente la memoria a la CPU para las variables, además, proporciona funciones de depuración con simulaciones, como la visualización de resultados y la visualización de dispositivos.

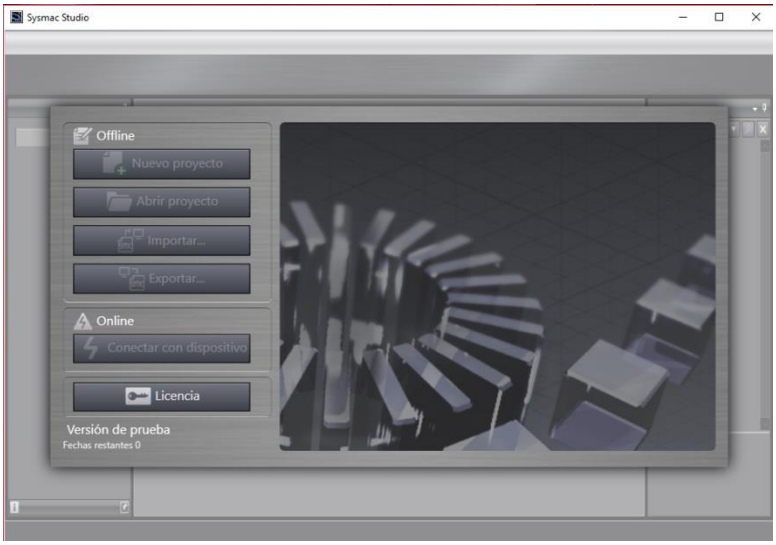


Figura 24.Sysmac Studio

3.2.6 Matlab/Simulink

Matlab es una potente herramienta matemática capaz de realizar cualquier tipo de operación si tienes las herramientas adecuadas.

Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware.

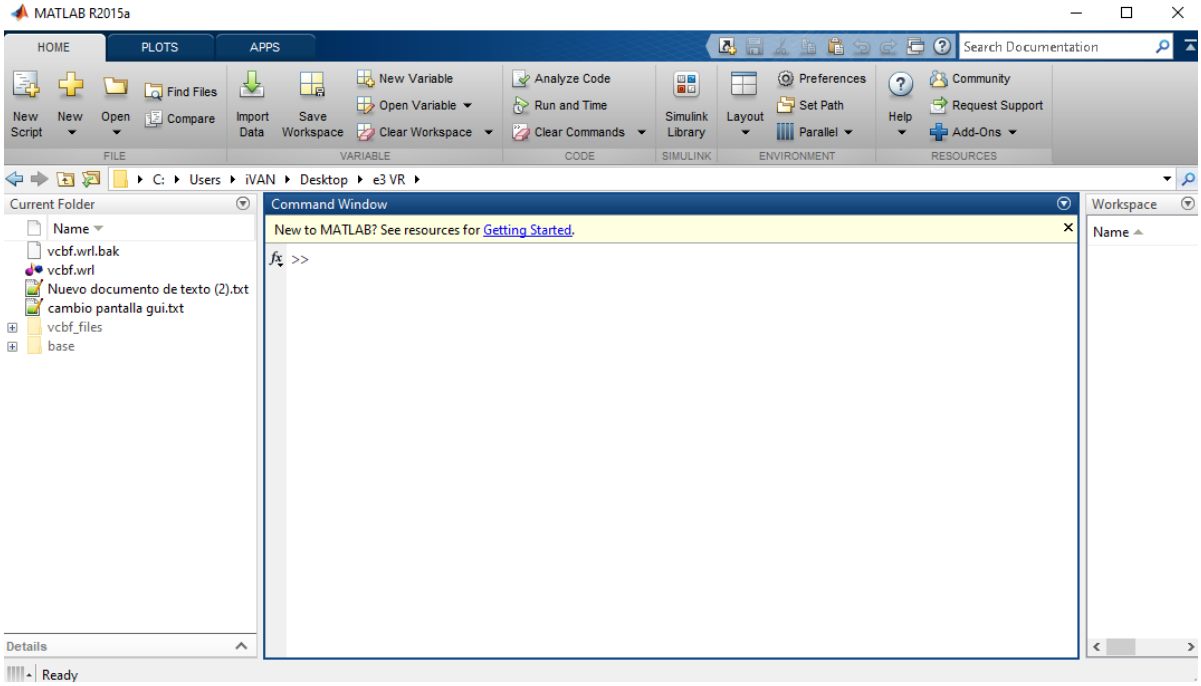


Figura 25.Matlab 2015ª

MATLAB dispone de varias herramientas adicionales que expanden sus prestaciones y que se van a utilizar como seria Simulink y GUIDE (editor de interfaces de usuario-GUI). Además de varias toolboxes; en concreto la **Stateflow** (máquina de estados), **PLC Coder** (generador de código para PLCs), OPC Toolbox (conectar con servidores OPC) y **3D World Editor** (editor de realidad virtual).

-GUIDE

Una interfaz gráfica de usuario (GUI), es una interfaz construida a través de objetos gráficos, tales como menús, botones, listas y barras de desplazamiento. Esos objetos formarán una interfaz en el momento que se defina una acción a realizar cuando se produzca una pulsación de ratón.

Actualmente, las herramientas que nos ofrece GUIDE, permiten además de diseñar la ventana, generar un archivo de extensión .M que contiene el código para controlar el lanzamiento e inicialización de dicha aplicación GUI.

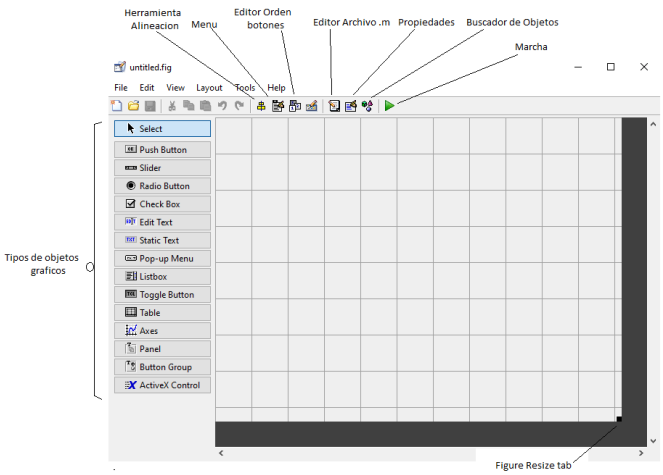


Figura 26.Zona de Trabajo GUIDE

-SIMULINK

Dentro de la plataforma Simulink, existen una serie de librerías por defecto. Dichas librerías contienen los bloques más elementales y por ello no serán explicadas. Y hay otras, que se explicaran a continuación, que tienes que ser instaladas

Librerías especializadas de Simulink.

Stateflow

Stateflow es un entorno para el modelado y simulación de lógica de decisión combinatoria y secuencial basado en máquinas de estado y diagramas de flujo.

El diagrama Stateflow permite definir lógica de decisión compleja en el control de la plataforma de forma gráfica y estructurada, con el fin de simular la respuesta del sistema a entradas externas, eventos y condiciones temporales. Se caracteriza por tener:

- Entorno de modelado, componentes gráficos y motor de simulación para modelar y simular lógica compleja.
- Diagramas de estado, tablas de transición de estado y matrices de transición de estado que representan máquinas de estado finito.
- Diagramas de flujo, funciones de Matlab y tablas de verdad para representar algoritmos.
- Animación de diagrama de estado, registro de actividad de estado, registro de datos y depuración integrada para analizar el diseño y detectar errores de tiempo de ejecución.

Esta librería tiene 3 bloques con los que se puede trabajar. Chart (máquina de estados), State Transition Table (tabla de transiciones de estados) y Truth Table (tabla de verdad).

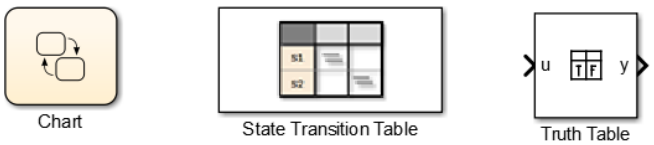


Figura 27.Bloques utilizables de librería Stateflow de Simulink

El bloque utilizado de esta librería será el chart, que no permitirá crear máquinas de estados con los que se controlará cada una de partes de la célula de fabricación.

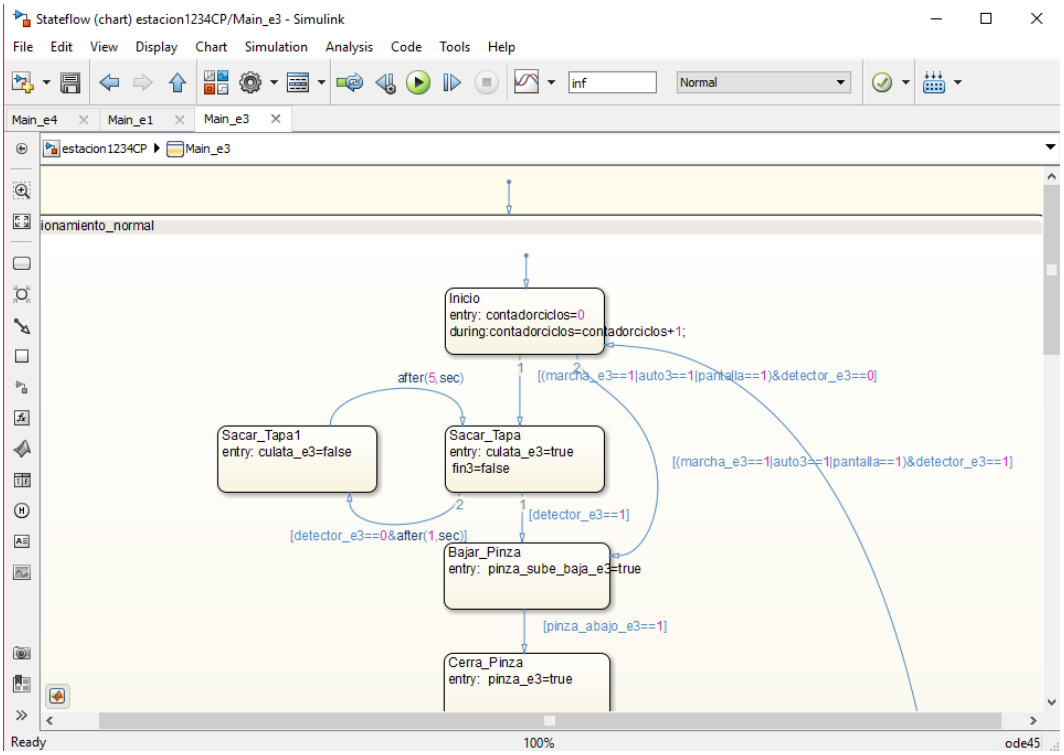


Figura 28.Zona de Trabajo Chart

OPC Toolbox

OPC Toolbox proporciona acceso a datos OPC directamente desde MATLAB y Simulink. Puede leer, escribir y registrar datos OPC desde dispositivos, en nuestro caso, de controladores lógicos programables.

Esta toolbox de Simulink incluye bloques de Simulink que le permiten modelar el control de supervisión en línea.

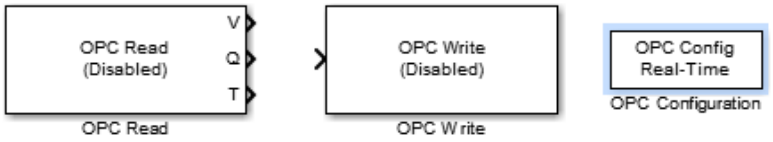


Figura 29.Bloques Simulink OPC Toolbox

- OPC Configuration (configuración OPC)
- OPC Write (Escritura OPC)
- OPC Read (Lectura OPC)

Además de utilizar estos bloques en Simulink, antes de todo hay que instalar en el centro de trabajo el programa del servidor OPC para realizar la configuración del server. El programa utilizado será ICONICS OPC Server, el cual nos permitiré comunicarnos con los autómatas Modicon que se encuentran en las estaciones.

PLC Coder

Simulink PLC Coder genera diagramas de texto estructurado a partir de modelos programados en Simulink. Los diagramas de texto estructurado se generan en PLCopen XML y otros formatos de archivo soportados por entornos de desarrollo integrados (IDE) ampliamente utilizados. Una vez generado el texto estructurado, se puede compilar y transferir la aplicación a diferentes dispositivos de controlador lógico programable (PLC) y programadores de automatización (PAC).

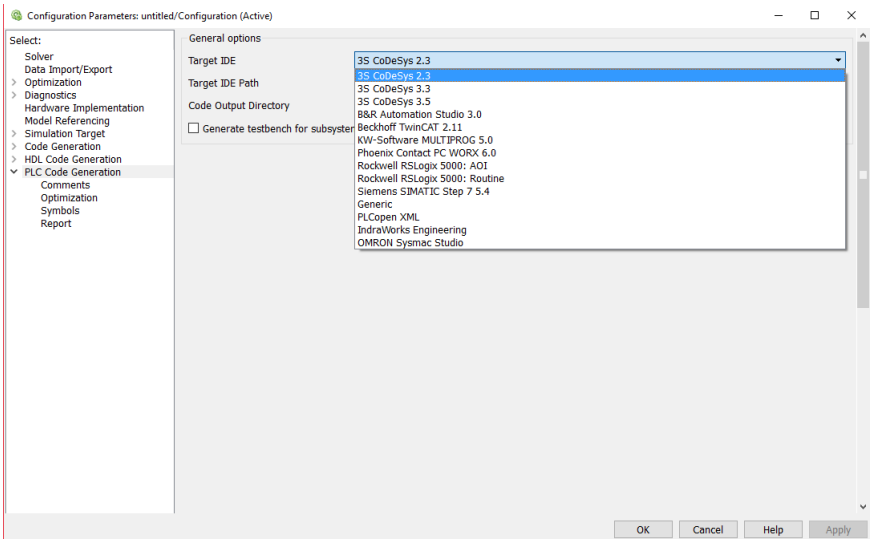


Figura 30.Configuración PLC Coder

Como se puede observar, se puede generar texto para diferentes controladores, entre ellos, Rockwell, pero no aparecen los Schneider. Por lo tanto, se decidió, exportarlo para OMROM Sysmac Studio, que el código es el más parecido al de Schneider, y posteriormente terminar la exportación manualmente.

El flujo de trabajo a seguir para esta herramienta de Simulink es:

- Diseño de un modelo de Simulink del que desea generar código.
- Coloque los componentes en un bloque del subsistema.
- Identifique el IDE del PLC de destino. Es decir, para el tipo de programa que quiere generar texto. Compruebe antes que el modelo es compatible con el software Simulink PLC Coder.
- Seleccione una carpeta de destino.
- Simule su modelo.
- Configure los parámetros del modelo para generar código para su PLC IDE.
- Examine el código generado.
- Importe el código a su PLC deseado, con el programa correspondiente.

Simulink 3D Animation

Simulink® 3D Animation proporciona aplicaciones para vincular modelos Simulink y algoritmos Matlab a objetos gráficos 3D. Le permite visualizar y verificar el comportamiento dinámico del sistema en un entorno de realidad virtual. Los objetos están representados en el lenguaje de modelado de realidad virtual (VRML), un lenguaje de modelado 3D estándar.

Simulink 3D Animation tiene diferentes aplicaciones con las que se puede crear e interactuar con escenas virtuales, Entre ellas 3D World Editor. Con esta aplicación se puede crear escenas detalladas a partir de modelos 3D exportados desde fuentes basadas en CAD o crear tú mismo los diseños.

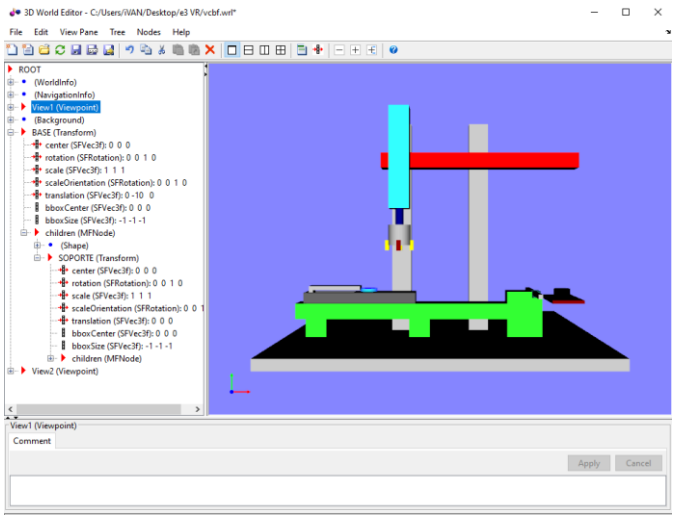


Figura 31.Zona de Trabajo 3D World Editor

4 Programación en Stateflow y Generación de Código

4.1 Introducción

Como ya se dijo en el capítulo anterior, el Stateflow es una herramienta de diseño gráfico interactiva que trabajando con Simulink modela y simula sistemas dirigidos por estados o eventos. Estos sistemas son a menudo utilizados para modelar la lógica que controla dinámicamente dispositivos físicos como ventiladores, motores y otros dispositivos.

La herramienta Stateflow tiene un bloque principal llamado chart el cual se utilizará para toda la programación del proyecto. Dentro de este bloque se pueden utilizar diferentes elementos con los cuales se realizarán las máquinas de estados.

4.2 Elementos de Stateflow – Chart

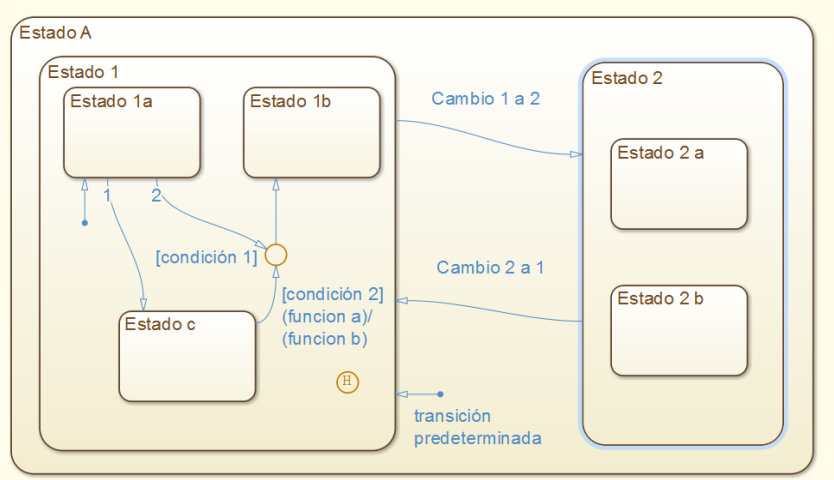


Figura 32.Ejemplo explicativo

4.2.1 Estados (States)

Los estados representan o bien estados propiamente dichos como en el planteamiento clásico de máquinas de estados, o bien engloban una completa máquina de estados que solo se ejecuta cuando dicho estado esté activo.

Este anidamiento de máquinas sobre estados se puede hacer de forma recursiva. En la figura anterior se puede observar como en la máquina de estados incluida en el estado más general EstadoA existen dos estados Estado 1 y Estado 2. En el primero existe una máquina de estados formada por tres estados Estado 1a, Estado A1b y Estado A1c.

Dentro de un estado, por tanto, pueden existir otros que se agruparán entre ellos de una de las dos formas posibles:

1. En paralelo (AND). - Se indican en el gráfico con línea discontinua y se ejecutan todos a la vez, lo que funcionalmente se puede entender como de forma concurrente.
2. Alternativamente (OR exclusive). Se indican en el gráfico con línea continua y se ejecutan de forma exclusiva uno cada vez.

La siguiente figura representa los dos tipos de estados de forma gráfica.

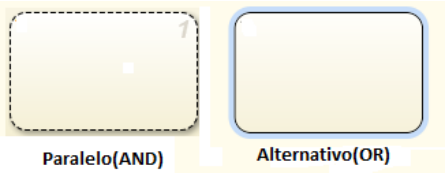


Figura 33.Tipos de Estados

Los estados pueden disponer de acciones a realizar en distintos instantes de su ejecución:

- Entrada (entry).- Cuando se activa el estado una sola vez.
- Durante (during).- Mientras esté activo el estado cada vez que se evalué el diagrama.
- Salida (exit).- Cuando se desactiva el estado una sola vez.
- Cuando (on).- Estando activo si se produce cierto evento.
- Además, también se puede programar sobre ellos funciones if y programación propia de Matlab como sería el SetParam ().

En la figura, se muestra un estado que utiliza las distintas opciones:

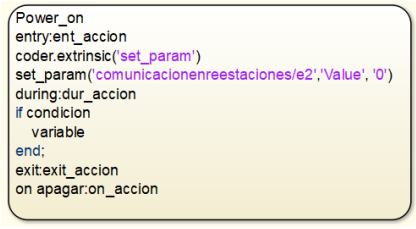


Figura 34.Ejemplos ejecuciones en estados

4.2.2 Transiciones (Transitions)

Las transiciones representadas mediante un segmento orientado indican el siguiente estado a estar activo cuando lo esté el estado de origen, es decir, dado un estado activo y del que se origina una transición a otro, si se cumplen las condiciones adosadas a la transición, dejará de estar activo el primero para pasar a estarlo el segundo.

Si una transición no tiene ninguna condición se produce cada vez que se evalué el diagrama (por ocurrir cualquier evento). En general las transiciones tienen alguna condición que debe cumplirse para que se produzcan.

La nomenclatura utilizada para representar las transiciones es la de la siguiente tabla.

- Evento (event). Indica que evento provocará que sea evaluada la condición de la transición. En ausencia de evento de forma explícita será válido cualquier evento del diagrama, incluido el paso por cada periodo de muestreo de Simulink.
- Condición (condition). Se expresa que condición debe de ser cierta para que se active la transición habiéndose producido el evento correspondiente y se ejecute la acción condicionada.
- Acción condicionada (condition action). Se ejecuta cuando es cierta la condición anterior.
- Acción de transición (Transition action). Se ejecuta la acción antes de ser excitado el estado siguiente.

4.2.3 Transición por defecto ("Default Transitions")

La transición por defecto es aquella que se activa al ser activado la máquina de estados donde está incluida, es decir que cada vez que se active una máquina de estados se activará el estado apuntado por la transición por defecto, este estado será la **posición inicial** de nuestra máquina de estados.

La transición por defecto no tiene por tanto origen pues no viene de ningún estado anterior. Solo tiene sentido aplicarlas cuando los estados están agrupados en forma de activación alternativa, y no cuando lo están en forma paralela, ya que en ese caso no hay ambigüedad posible.

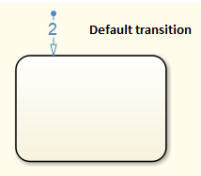


Figura 35.Transicion por defecto y Posición inicial

4.2.4 Memoria de estado ("History Junction")

Cuando se introduce provoca que al reactivarse un estado en vez de activarse el estado apuntado por la transición por defecto se active el último que estuvo activo.

En el siguiente ejemplo se puede observar un ejemplo en el que la memoria de estado corresponde con el símbolo H inscrito en un círculo.

El sistema puede estar en Power_on o en Power_off, y mientras que la primera vez que se activa (Power_on) el estado activo dentro de Power_on es Low, las siguientes veces que se activa el estado activo será el último que lo estuvo. Es decir que, aunque hay una transición por defecto esta ya no actúa después de la primera ejecución de la máquina de estados.

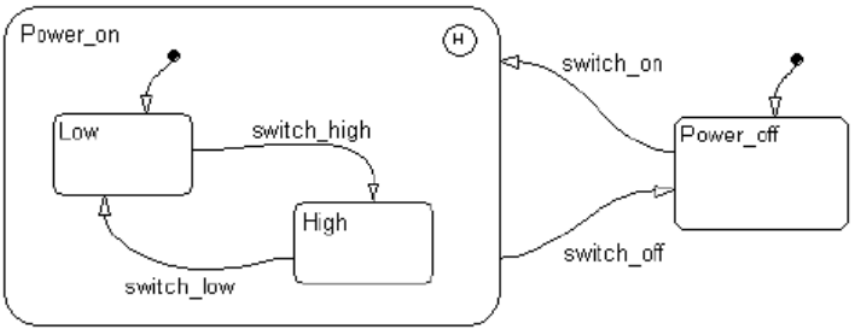


Figura 36.Ejemplo Memoria de estado

4.2.5 Eventos ("Events").

Todas las transiciones reaccionan a los eventos que son las situaciones previamente definidas, y que, en el desarrollo de funcionamiento de la máquina, se producen. Cada vez que se produce un evento provoca que se evalúen los estados y transiciones del diagrama.

Los eventos no se aprecian gráficamente en el diagrama, solo se observan las etiquetas que los definen. En la creación de un diagrama es preciso definir en que consiste cada evento, para ello se utiliza la parte de la herramienta denominada 'explorador de Stateflow'.

Existen diversas formas de generar eventos, pero, sin duda, la más habitual es la de introducir una señal de reloj que periódicamente, y con uno de sus flancos, provoque la evaluación de las condiciones en todas las transiciones. Esto implica directamente implementaciones digitales inmediatas.

Se pueden utilizar múltiples entradas para provocar con sus flancos eventos, eso sí, todas entran por un único pin al bloque de StateFlow en forma de vector de entradas.

Esto puede tener utilidad para disponer por ejemplo de varios relojes de distintas frecuencias, entradas de interrupción por flanco, etc.

El aspecto de la ventana de introducción de eventos, así como de otros elementos, es como la de la figura a continuación en la que se muestra el evento que se producirá cuando se produzca el flanco de bajada de la señal número dos de las que entran por el pin de eventos.

4.2.6 Datos ("Data").

Los datos que internamente para el procesado de la información que el diagrama precise deben ser previamente definidos, salvo que se utilicen referencias al espacio de trabajo de Matlab, que no es necesario.

En la siguiente figura se observa la ventana de introducción de datos para la definición de un dato interno al diagrama.

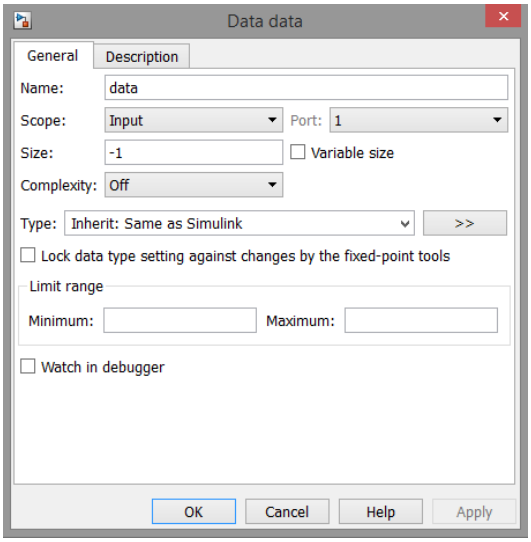


Figura 37.Ventana creación datos

Otros datos habituales son los datos que se comparten con el espacio de Simulink mediante los puertos de entrada y de salida.

4.2.7 Acciones ("Actions")

Se pueden realizar acciones tanto en las transiciones, como en los estados en las distintas opciones que los estados contemplan, ya descritas en el apartado correspondiente.

Dado que StateFlow para su ejecución compila el modelo previamente a la ejecución, para las acciones se puede usar un subconjunto de instrucciones propias de C. También admite la ejecución de scripts de Matlab mediante el operador 'ml.'.

Entre las operaciones que son permitidas en la ejecución de acciones están las operaciones lógicas, desplazamientos, incrementos y decrementos, punteros, etc. Incluso se pueden declarar funciones en C en fichero de código fuente aparte para ser llamadas dichas funciones desde las llamadas a las acciones del diagrama.

Existe también la posibilidad de ejecutar acciones basándose en condiciones temporales.

- Tipo Clock, por ejemplo, after(2,sec)
- Existen otras opciones temporales como: before, at y every.
- Uniones de conexión ("Connective Junctions")

4.2.8 Funciones

Las funciones también se permiten en el Stateflow, tanto escritos en código C en los estados, como con bloques. Las funciones permitidas son las siguientes:

- Estructuras 'if-then-else'.
- Bucle 'for'.
- Tabla de la verdad ("Truth Table Functions")
- Transiciones tanto de múltiples orígenes a un destino, y viceversa.

4.3 Generación de código

4.3.1 Introducción

En el proyecto será necesaria la generación de código para dos autómatas diferentes. Para el autómata Rockwell y para el autómata Schneider.

La herramienta PLC Coder permite realizar la exportación directa para los autómatas Rockwell, pero no para autómatas Schneider. Para los autómatas Schneider se realizará la exportación para Omron y posteriormente se realizarán unos pequeños cambios para adaptar ese código a los autómatas Schneider.

Para hacer la exportación habrá que seguir serie de pasos los cuales serán los mismos para todas las estaciones y transporte, la única diferencia será, que habrá que realizarlos en cada uno de los bloques chart correspondientes de cada una de las estaciones.

4.3.2 Generación de código para Rockwell y Omron

A continuación, se explicarán los diferentes pasos que se realizarán para la obtención de código para autómatas Rockwell.

1. Con el botón derecho del ratón en el bloque Chart y seleccione PLC Code -> Options.

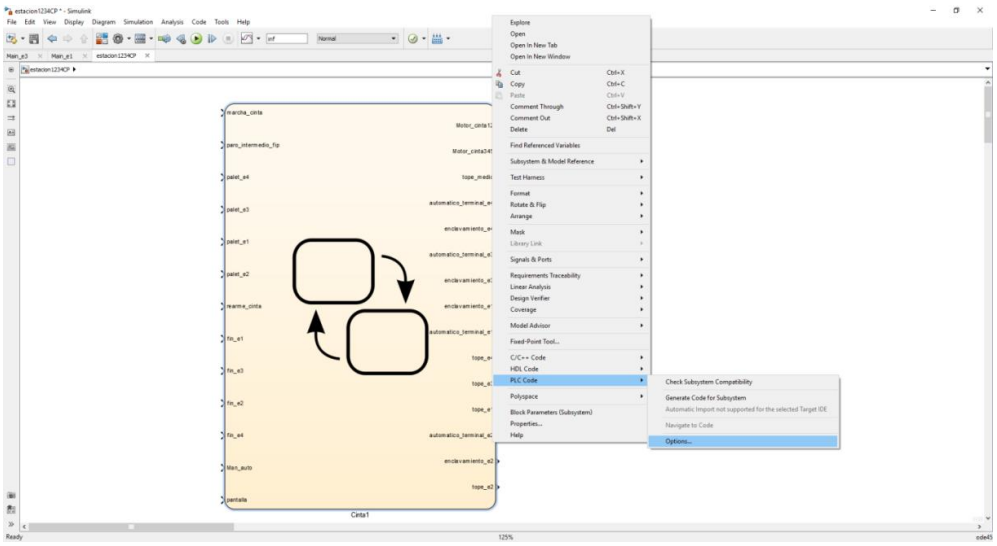


Figura 38.Proceso para abrir panel de configuración

2. Aparecerá la ventana Parámetros de configuración.

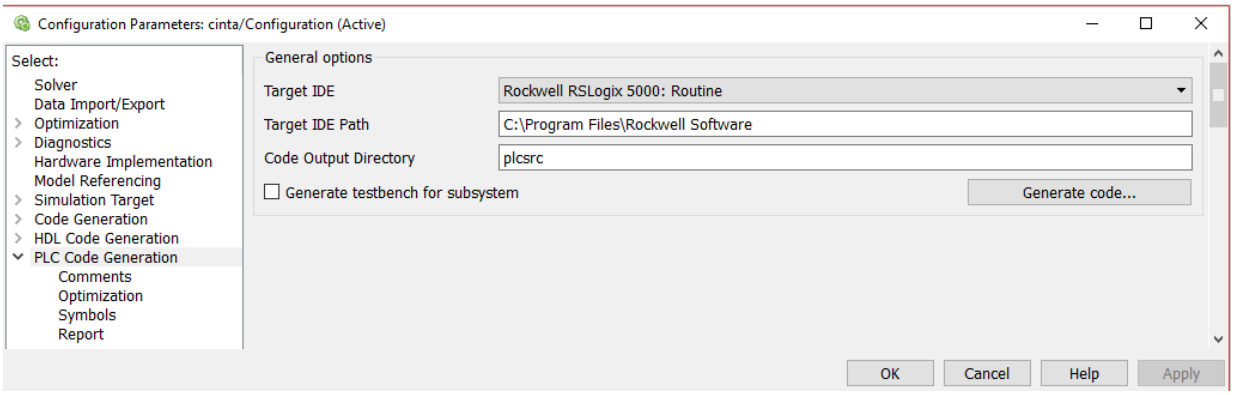


Figura 39.Ventana configuración PLC Coder

3. En el panel **PLC Code Generation**, seleccione una opción de la lista **Target IDE**, para este caso, **Rockwell RSLogix 5000: Routine.** La lista **Target IDE** muestra el conjunto completo de IDE admitidos.

Cuando se valla a controlar con Schneider el **Target IDE** será **OMROM Studio**, y posteriormente, realizar los cambios oportunos para poder utilizarlo en el autómata Schneider. Los cambios son los siguientes:

- En la programación cuando se usa cualquier tipo de variable, Omron adapta al tipo de variable es, que para Schneider habrá que eliminar.

Omron	Schneider
is_c3_ejemplo_1 := USINT#2;	is_c3_ejemplo_1 := 2;
Motor_cinta345 := BOOL#FALSE	Motor_cinta345 := FALSE

Tabla 6.Ejemplos cambios en variables en exportación para Schneider

- Y la segunda modificación será con el temporizador, se eliminará las líneas correspondientes al temporizador exportado y se cambiaran por el temporizador propio de Unity t de tipo TON.

```
temporalCounter_i3(timerAction := INT#2, maxTime := DINT#5000);
 IF temporalCounter_i3.done THEN
```

→

```
temporizador3 (IN := TRUE,PT := t#3s,Q => done3 );
IF done3 THEN
```

Figura 40.Ejemplos cambios en temporizador en exportación para Schneider

4. Se Aplicarán los cambios
5. Y se genera el código pulsando el botón (Generate Code)
   Este botón nos permite generar código de texto estructurado el cual se almacena con el nombre_fichero.L5X (por ejemplo, cinta1.L5X (Rockwell) ó cinta1.XML (OMROM))

Cuando se completa la generación de código, aparece un hipervínculo de diagnóstico en la parte inferior de la ventana del modelo. Haga clic en este hipervínculo para abrir la ventana del Visor de diagnósticos.

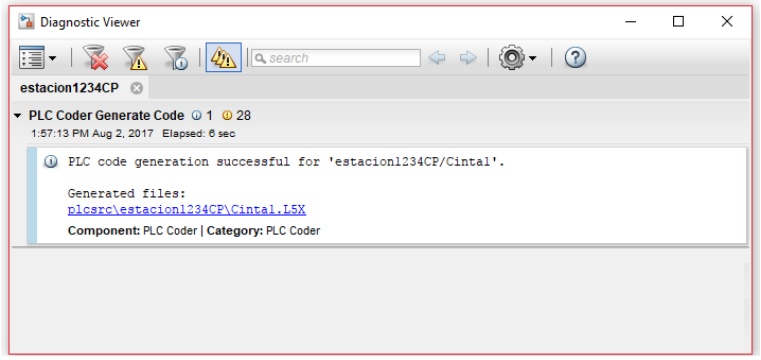


Figura 41.Ventana de diagnostico

5 Control centralizado desde Simulink mediante OPC

5.1 Introducción

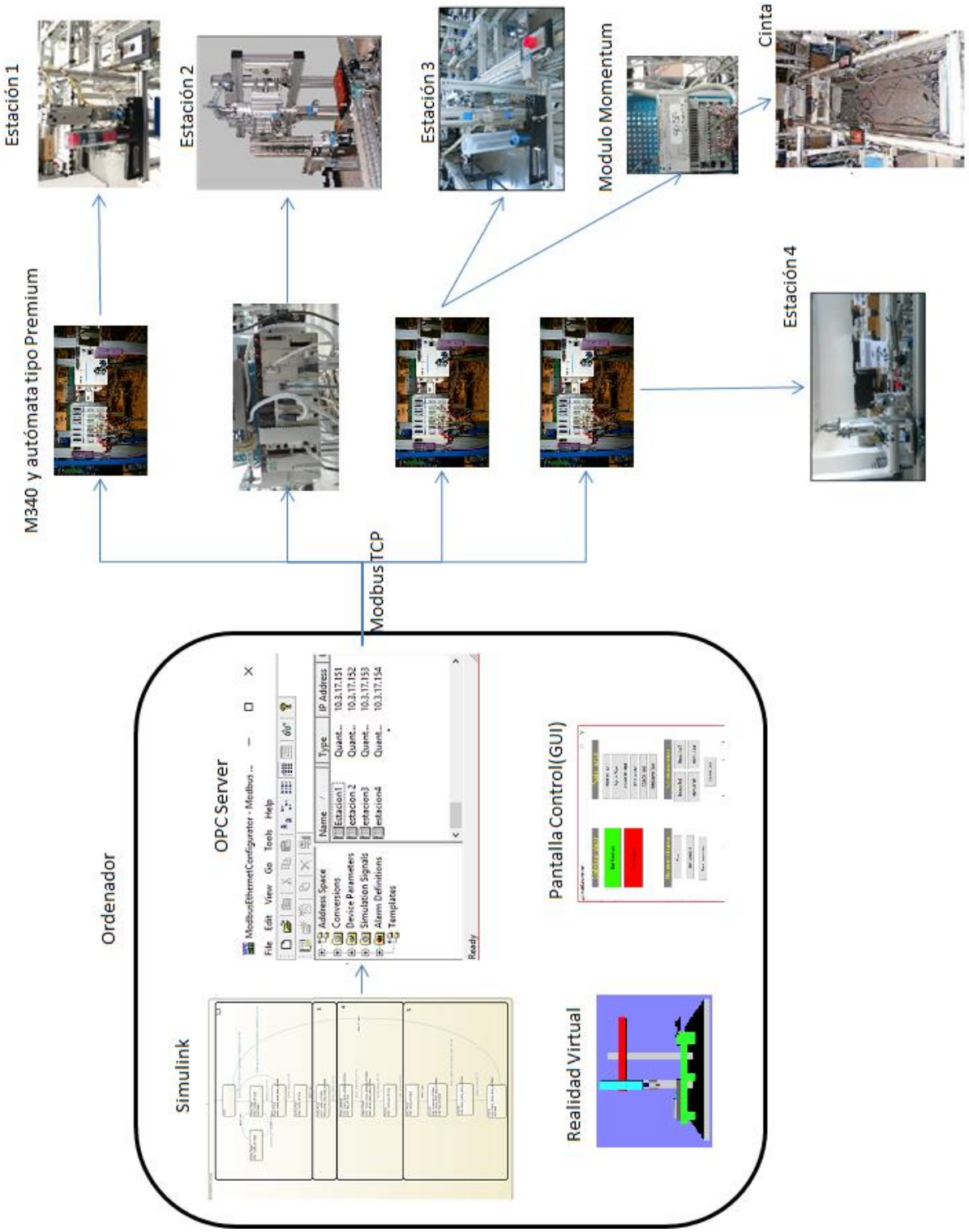


Figura 42.Esquema general Control Simulink

.

El objetivo de este control es poder manejar completamente toda la célula de fabricación, estaciones y transporte, desde Simulink. Para ello, se implementarán las máquinas de estados en StateFlow que controlarán las estaciones y la cinta desde el propio programa de Simulink.

A la hora de realizar el control, se partirá de los programas base realizados de cada una de las estaciones y el transporte, es decir, máquinas de estados en los que solo este implementado el proceso automático.

Para poder realizar este control se tiene que añadir un servidor OPC que comunique Simulink con el autómata, lo que ocasionará tener que configurarlo, crear las variables que se vayan a utilizar y añadir bloques en Simulink que permitan utilizar dichas variables. Este servidor OPC se comunicará con las 4 estaciones mediante los autómatas Schneider, por lo tanto, habrá que realizar un programa para estos autómatas.

Los autómatas de Schneider de cada una de las estaciones 1,3 y 4 se utilizan como si fueran tarjetas de entradas/salidas, es decir, servirán para leer sensores y escribir en los accionadores de cada una de las estaciones.

Ya que no es posible comunicarse directamente con el transporte, la comunicación con él se realizará mediante la estación 3 utilizando un módulo Ethernet. Por lo tanto, a dicha estación habrá también que pasarle las variables del transporte.

La programación se realizará con la ayuda de la herramienta Stateflow de Simulink, con la cual se podrá crear máquinas de estados con las que controlar las estaciones. Para el caso del transporte y de las estaciones 1,3 y 4 se dispondrá de una máquina de estados por estación con el control para realizar el proceso completo de Fabricación. En cambio, para la Estación 2 solo se tendrá una máquina de estados la cual indique a la estación que proceso debe realizar.

Con la estación 2, el control se realizará mediante una SFC que se encontrará en la propia estación, ya que no se puede realizar el control desde Simulink debido a los requisitos temporales del control debido a que hay que manejar un motor paso a paso. En cambio, en el resto de estaciones se trabajará fundamentalmente con entradas salidas digitales y alguna analógica.

Además de todo esto, también se ha creado un interfaz de usuario en Matlab con la cual se podrá poner en marcha las estaciones y elegir la pieza que se pretende fabricar.

Por último, se creó un sistema de realidad virtual en 3D para realizar la supervisión del proceso de las estaciones.

5.2 Configuración del servidor OPC

En primer lugar, cabe decir que habrá que configurar el OPC para que se comunique con las 4 estaciones y crear todas las variables que se vayan a utilizar.

Configuración

En primer lugar, habrá que abrir el programa Modbus Ethernet que se utilizará para la comunicación.

Después de esto, hay que indicar los dispositivos con los cuales se va a conectar. Esto se hará pulsando con el botón derecho sobre el primer término que aparece (Address Space) y se elegirá new device (nuevo dispositivo).

En la parte derecha aparecerá una nueva ventana donde se configurará el dispositivo, ponerle nombre (name), dirección asociada (IP), tipo de dispositivo (Type) y tiempos de salida (Timeout).

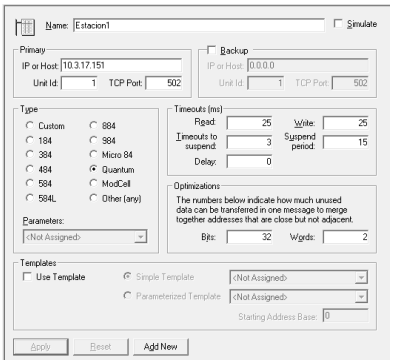


Figura 43.Configuración final Estación 1

Los tiempos se han reducido drásticamente para poder detectar los flancos de subida y bajado con mayor rapidez para que no se produzcan retrasos en la producción.

Para este proyecto se utilizarán 4 autómatas que, aunque tienen sus diferencias, en cuanto, a su configuración en el OPC será la misma, aunque con pequeñas diferencias. Por lo tanto, se explicará para el caso de la estación 1 y para el resto habrá que realizar el mismo proyecto.

Para el resto de estaciones habrá que realizar los mismos pasos, pero habrá que poner el nombre de la estación correspondiente y cambiar la dirección IP.

Nombre	Dirección IP
Estacion1	10.3.17.151
Estacion2	10.3.17.152
Estacion3	10.3.17.153
Estacion4	10.3.17.154

Tabla 7.Direcciones IP Estaciones

Una vez que se han creado los 4 dispositivos para las estaciones, hay que crear todas las variables que se va a utilizar. Como todas las variables se crean del mismo modo y son del mismo tipo solo se va a explicar una de ellas.

Para crear una variable e en primer lugar pulsando con el botón derecho del ratón sobre el dispositivo el cual corresponde la variable. Y se pulsa sobre New data ítem (nuevo objeto).

Una vez creada la variable aparece la siguiente ventana donde lo se modificarán las características cono el nombre deseado, el tipo de variables y la posición que le se le ha otorgado

El tipo de variable utilizado para todas las variables será tipo Coil (booleana), la cual se podrá leer y escribir. Cabe decir que la dirección que se le asocie, posteriormente en Unity habrá que asociarle la misma dirección, por lo tanto, para esta variable, en Unity se creará la variable,

pero la dirección será un valor menor, ya que, en el valor del servidor OPC no indica dirección si no la posición. En consecuencia, la dirección en Unity será %M0

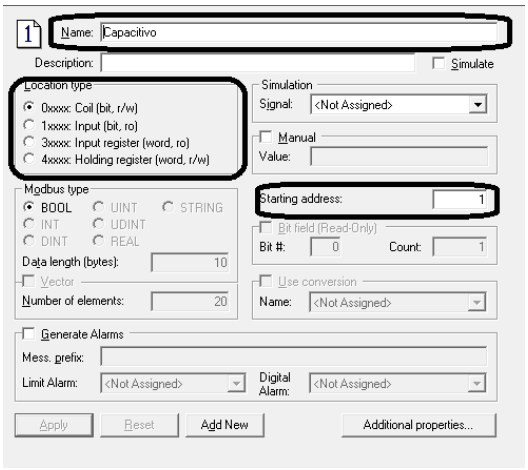


Figura 44.Configuración variable

Con el resto de variables habrá que realizar los mismos pasos, pero poniendo su nombre correspondiente y otra dirección. Las variables creadas son las siguientes:

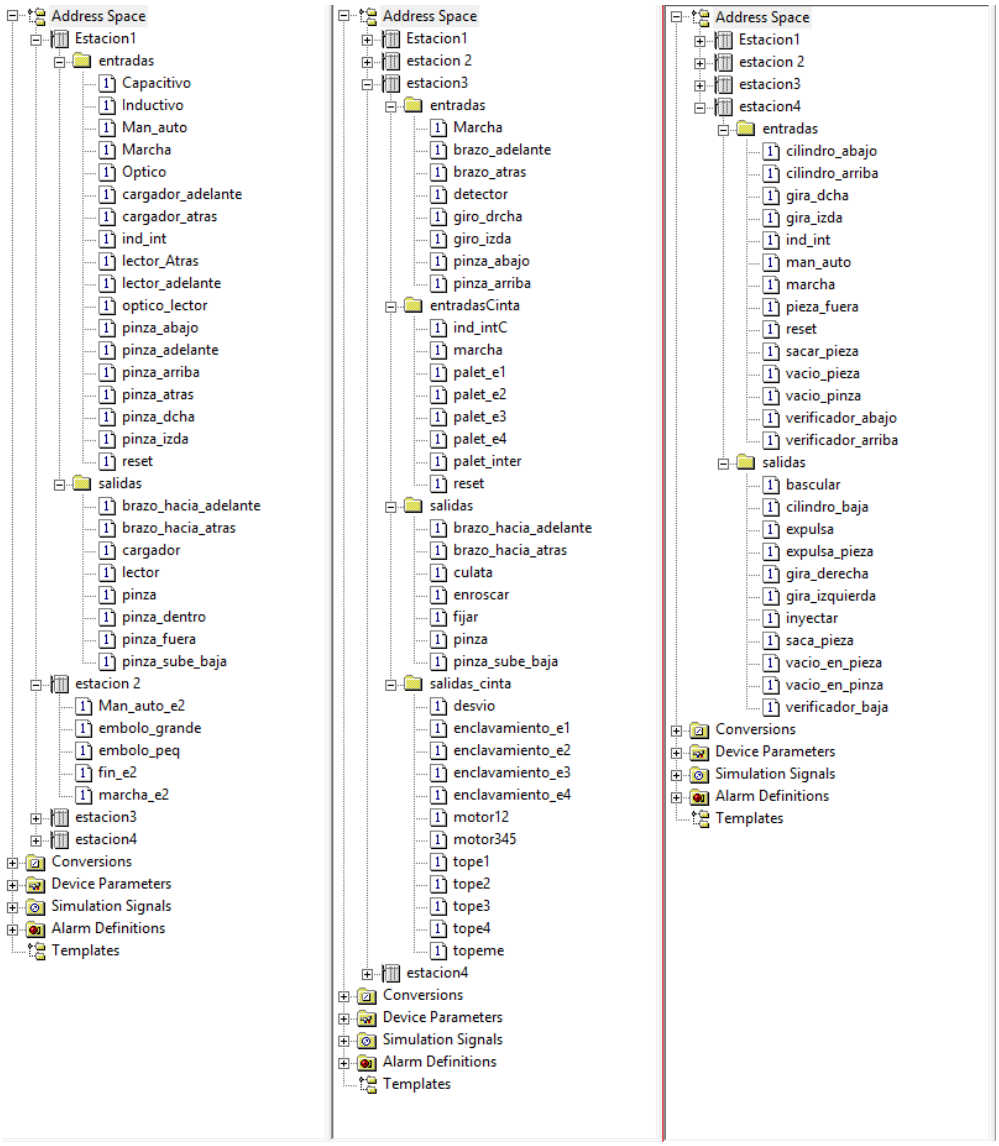


Figura 45.Variables creadas en el servidor OPC

5.3 Programación en Stateflow de la estación 1

Una vez que se han creado todas las variables en el Servidor OPC hay que realizar unas pequeñas modificaciones en la estructura general de Simulink para que este pueda modificar los valores de las variables creadas en el OPC para que esté los comunique a las estaciones.

En primer lugar, habrá que añadir un bloque de configuración del OPC. Al pulsarlo se abrirá una ventana en la cual habrá que pulsar Configure OPC Client. Aparecerá una segunda ventana donde se elegirá el OPC que se quiere utilizar.

Una vez hecho esto, se utilizarán los bloques OPC Read y OPC Write para leer y escribir las variables que se han creado con anterioridad en el apartado anterior.

Al pulsar sobre ellos aparecerá una ventana donde se podrá configurar que variables se van a utilizar. Al pulsar en el botón Add Items (añadir objetos) aparece otra ventana, donde se encuentran todas las variables que se han creada en el programa del servidor OPC con anterioridad.

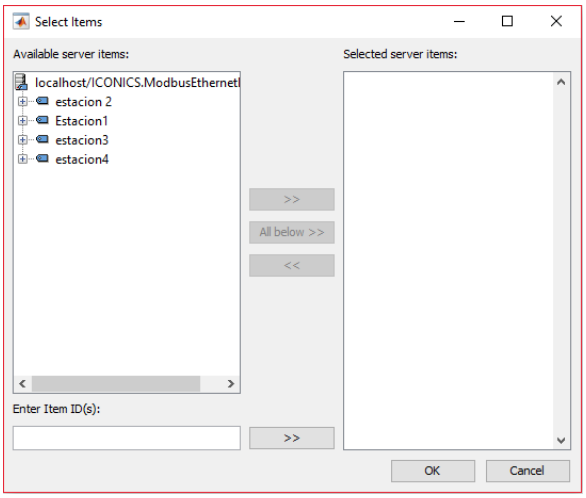


Figura 46.Pantalla de Elección de variables

En ella se elegirá la estación 1 y las variables de pendiendo del bloque OPC. Si es el bloque OPC Read se elegirán las entradas y si es el OPC Write las salidas.

Como la entrada o salida de estos debe ser un vector, a su salida o entrada se debe añadir un bloque Demux o Mux respectivamente. Una vez separadas las variables solo habrá que conectar la entrada del OPC con la entrada al chart y la salida del chart con la salida del OPC. Quedando el esquema para la estación 1 del siguiente modo:

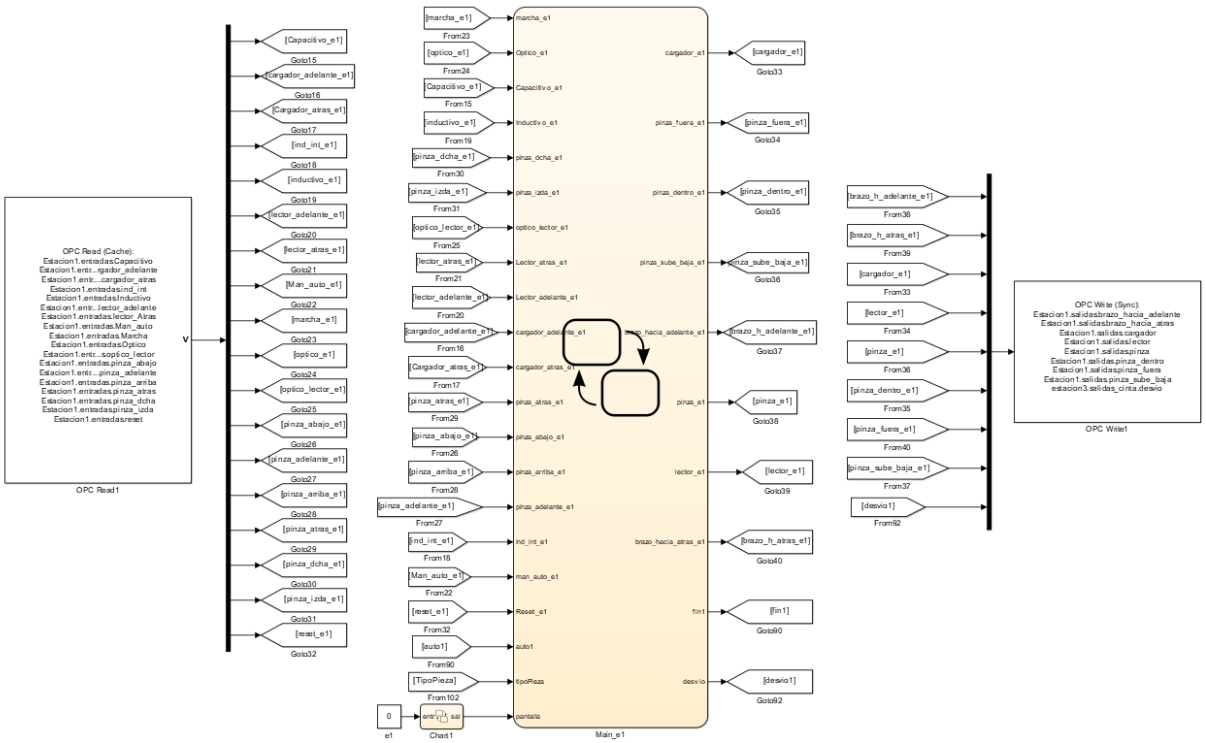


Figura 47.Esquema final para la estación 1

Dentro del bloque chart Main_e1 se encuentra el control para la estación 1 el cual se ha creado con algunos elementos del capítulo 4. A continuación se puede ver la programación del bloque chart junto con su explicación para el control de dicha estación.

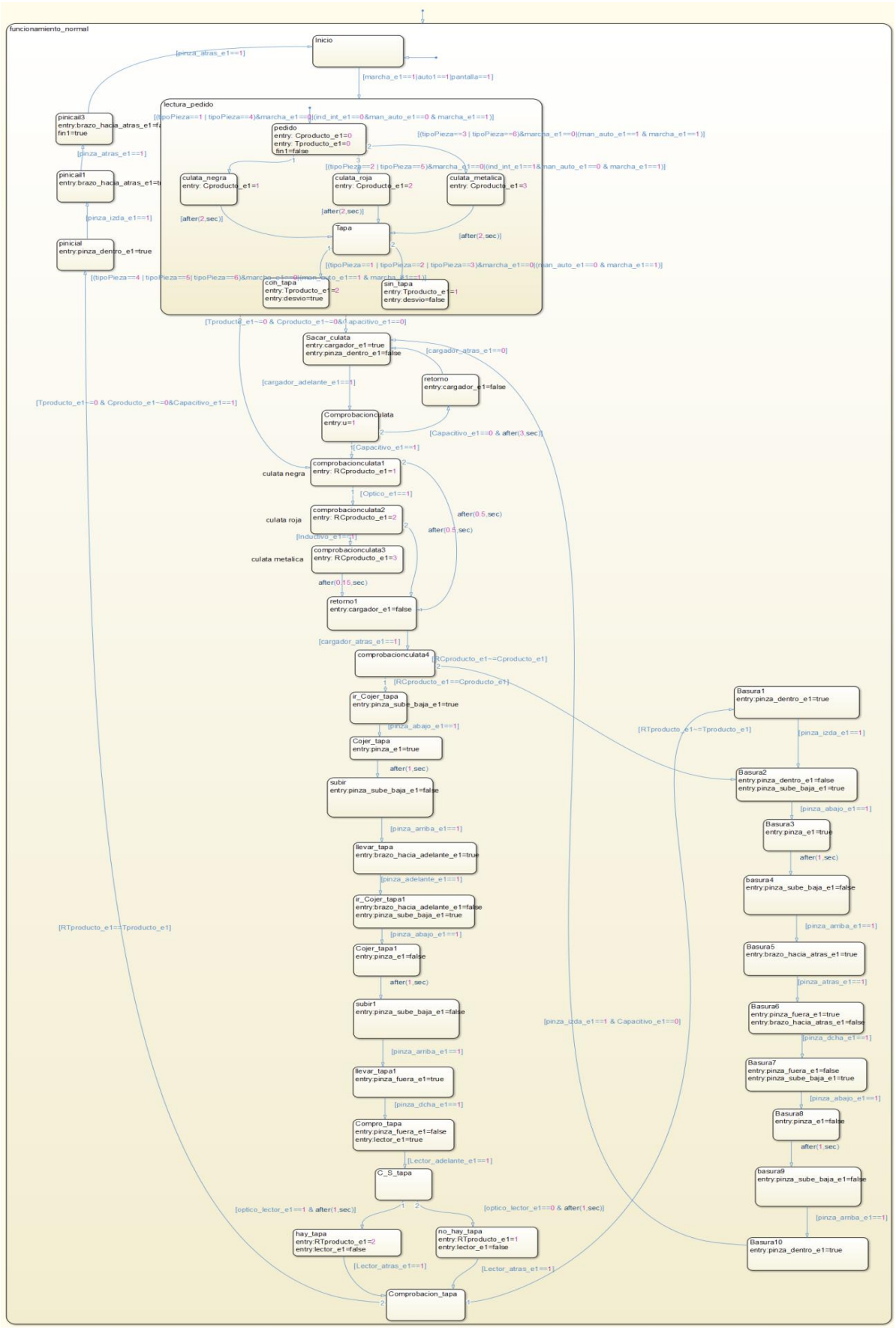


Figura 48.Programa Estación 1 en Simulink

Justo tras pulsar el botón marcha se encuentra el estado lectura del pedido el cual se realizará una conversión de las variables de entrada a unas variables auxiliares las cuales se utilizarán para comparar con los datos leídos de los sensores. Estas dos variables indican el color de la pieza y si tiene o no tiene tapa.

Las variables de entrada utilizadas, serán por un lado tipoPieza la cual indica el tipo de pieza con un valor numérico del 1 al 6 o con los interruptores de la botonera Man_auto e Ind_int. Como solo son dos interruptores y hay 6 tipos de piezas, se han utilizado los dos interruptores para indicar el color y posteriormente únicamente el Man_auto para si tiene o no tapa.

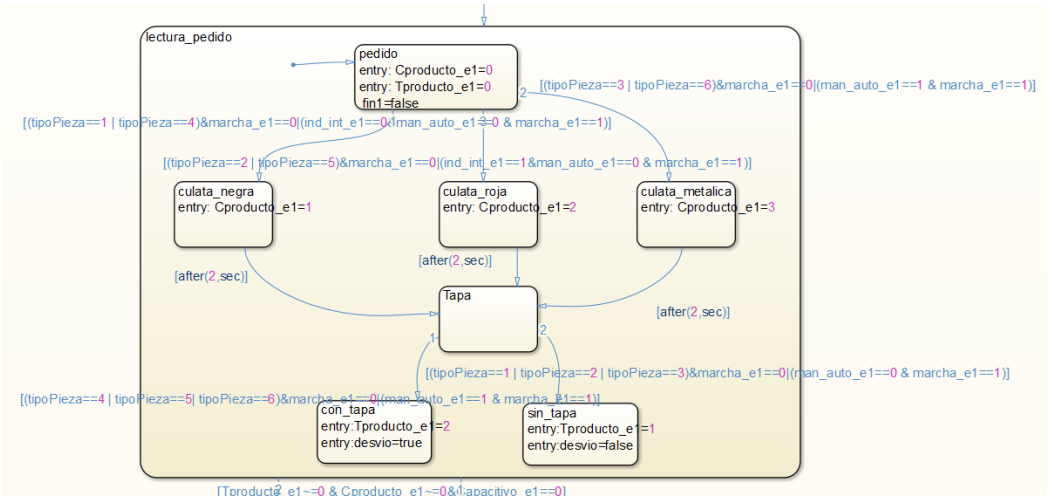


Figura 49.Parte de diagrama estado correspondiente al estado Lectura pedido de la Estación 1

Entradas				Salida lectura pedido	
TipoPieza	Man_auto_e1		Ind_int_e1	CProducto	TProducto
	1º	2º			
1 (Negra Sin Tapa)	0	0	0	1	1
2 (Roja Sin Tapa)	0	0	1	2	1
3 (Metálica Sin Tapa)	1	0	0	3	1
4 (Negra Con Tapa)	0	1	0	1	2
5 (Roja Con Tapa)	0	1	1	2	2
6 (Metálica Con Tapa)	1	1	0	3	2

Tabla 8.Cuadro resumen de las variables auxiliares en función de la pieza pedida en la terminal

El proceso que se ha implementado se correspondería con el siguiente funcionamiento:

1. Cuando el botón de marcha es pulsado o cuando llegue un palet vacío, un cilindro empujador se acciona de manera que la primera pieza del almacén es expulsada, en caso contrario, por fallo del accionamiento o falta de piezas, el empujador volverá a su estado inicial y después de 5 segundos volverá a sacar otra pieza. En este proceso también se realiza la lectura de los sensores para conocer el tipo de pieza, en el caso de ser un cilindro negro, se activa el sensor capacitivo, si el cilindro es rosa, se activa el capacitivo y el inductivo y sí es la pieza metálica se activan los 3 sensores.

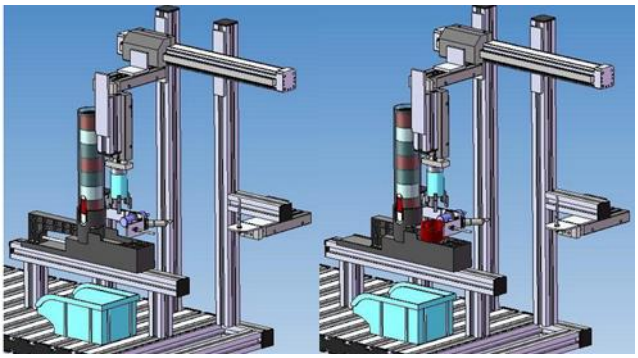


Figura 50.Paso 1 Proceso Estación 1

2. Una vez comprobado el tipo de pieza que ha salido, el empujador se desactiva y por lo tanto comienza el proceso de traslación de la pieza. En primer lugar, y una vez expulsada la pieza, el brazo bajará y, mediante el cierre de la pinza, se encarga de recoger el cilindro y posteriormente el brazo sube.

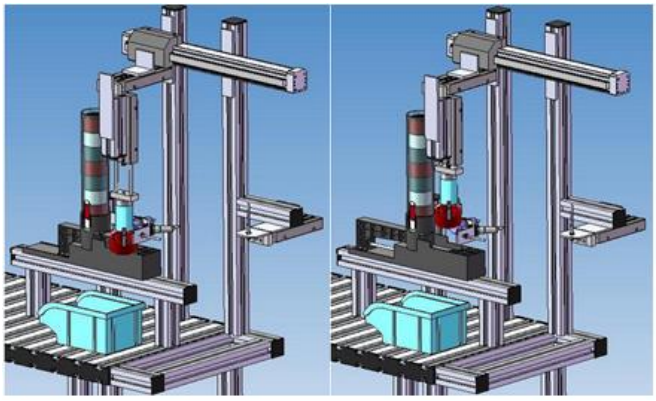


Figura 51.Paso 2 Proceso Estación 1

3. Si el cilindro recogido es el correcto, el brazo se desplaza hacia la zona del palet hasta que el sensor indique que ha llegado al final de su recorrido. En caso contrario, el brazo se desplaza al contenedor situado a la derecha y deposita la pieza.

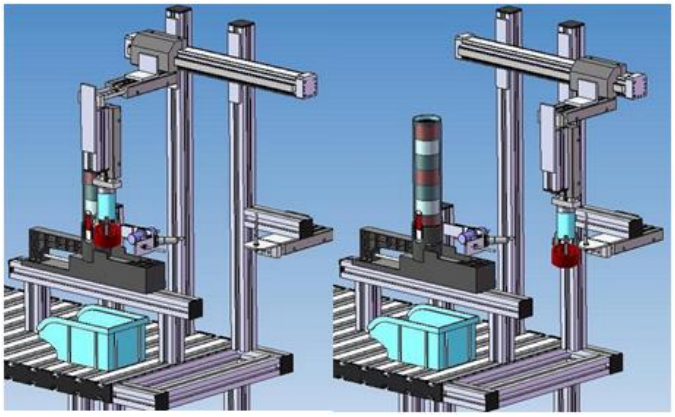


Figura 52.Paso 3 Proceso Estación 1

4. El brazo baja, y depositara el cilindro sobre el palet. La pinza se abre y finalmente el brazo vuelve a subir.

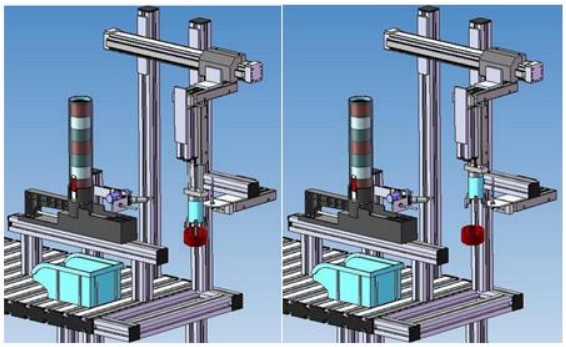


Figura 53.Paso 4 Proceso Estación 1

5. El cilindro vuelve, tras esto, se activa el lector, el cual comprueba si la pieza lleva o no lleva tape. En el caso en el cual la pieza sea correcta, el proceso termina aquí. En caso contrario, se deposita la pieza en el contenedor situado a la derecha de la estación.

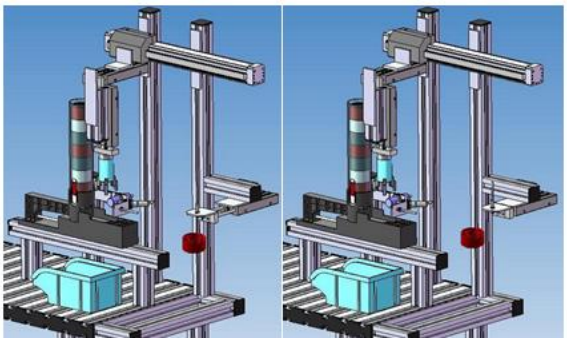


Figura 54.Paso 5 Proceso Estación 1

6. El cilindro vuelve y recoge el cilindro, como se muestra en las siguientes imágenes:

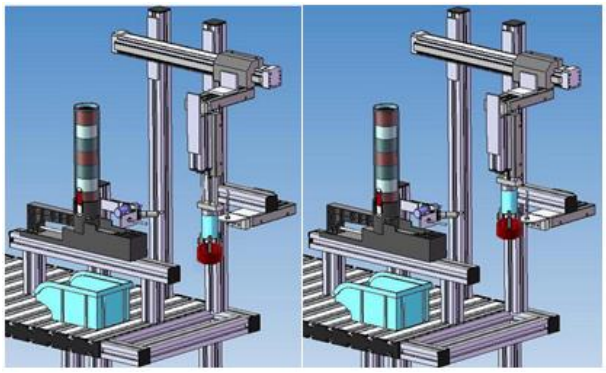


Figura 55.Paso 6 Proceso Estación 1

7. Finalmente, el brazo se desplaza a la derecha, y retrocede hasta que haya recorrido toda la cinta hacia atrás. Una vez realizada esta operación, se deposita el cilindro en el contenedor con la apertura de la pinza.

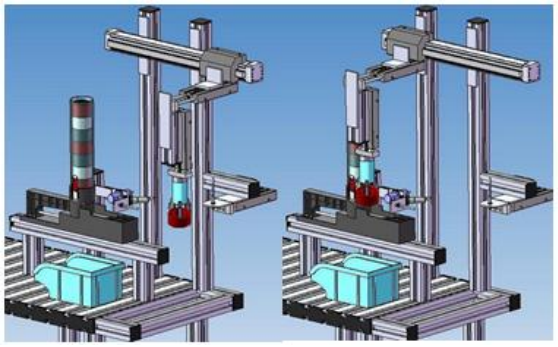


Figura 56.Paso 7 Proceso Estación 1

La ejecución de este control se puede observar en este video.

5.4 Programación en Stateflow de la estación 3

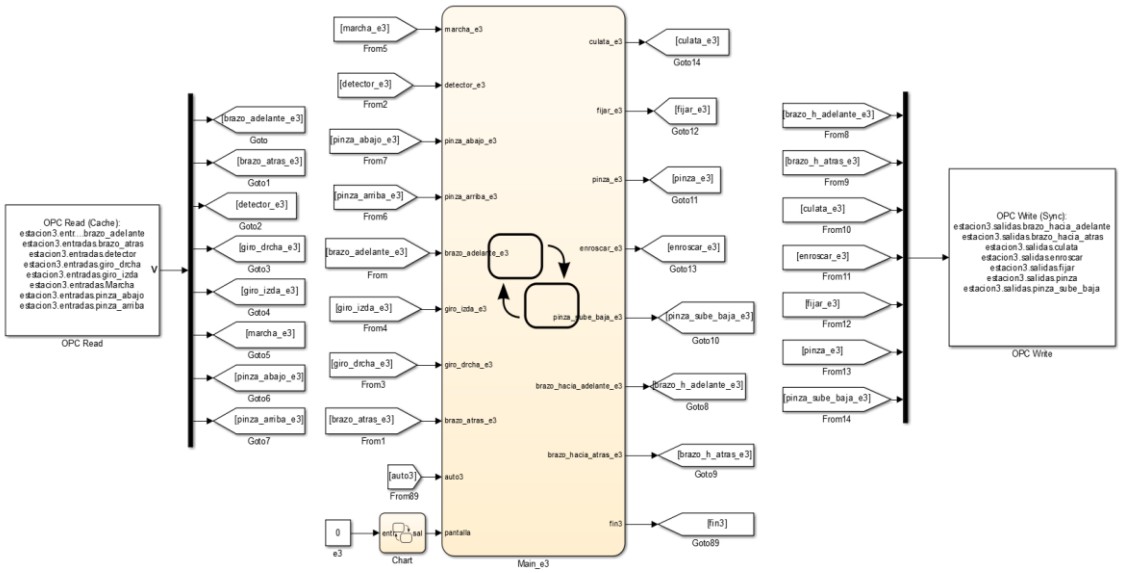


Figura 57.Esquema final para las estaciones 3

La estación 3 es la estación más simple en cuanto al programa, ya que solo tiene un proceso sea cual sea el tipo de pieza sin tapa que llegue a la estación.

El estado automático será el siguiente:

1. Cuando el botón de marcha es pulsado un cilindro empujador se acciona de manera que una de las culatas del almacén es expulsada. En caso contrario, por falta de piezas, se recogerá el empujador y después de 5 segundas se volverá a accionar

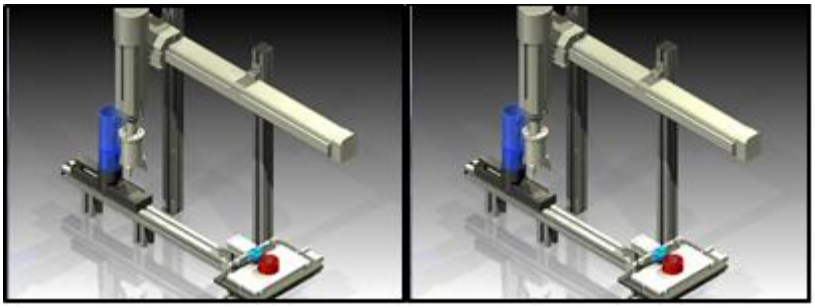


Figura 58.Paso 1 Proceso automático estación 3

2. El lector óptico detecta que la pieza ha salido, el empujador se desactiva y por lo tanto comienza el proceso de traslación de la pieza. En primer lugar, y una vez expulsada la pieza, el brazo bajará y, mediante el cierre de la pinza, se encarga de recoger la culata.

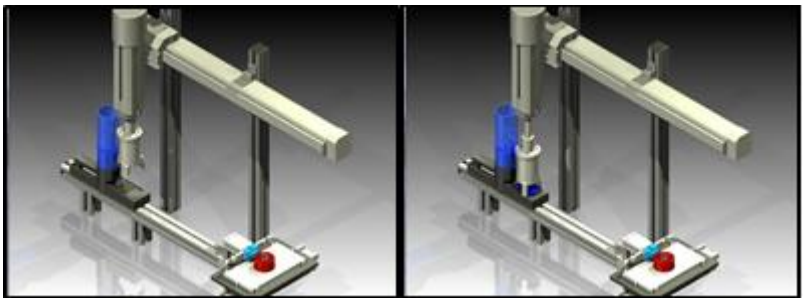


Figura 59.Paso 2 Proceso automático estación 3

3. El brazo sube y se desplaza hacia la zona de la abrazadera. Para ello, se activa el actuador encargado del movimiento traslación, hasta que el sensor nos indique que la pinza ha llegado a dicha zona.

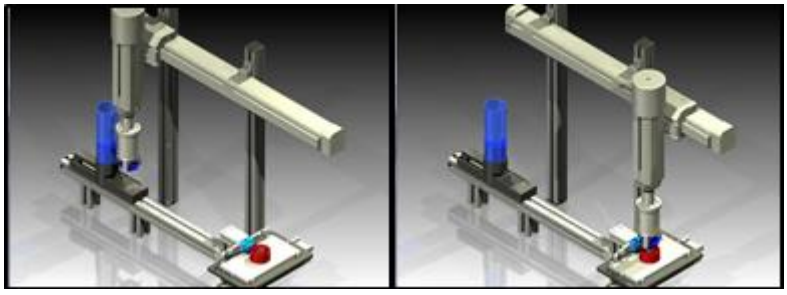


Figura 60.Paso 3 Proceso automático estación 3

4. La mordaza se cierra para sujetar el cilindro y el brazo baja. Se debe cerrar la mordaza antes de realizar el giro. Y después de haber enroscado, se soltará la pieza y el brazo volverá a la posición inicial.

Figura 61.Paso 4 Proceso automático estación 3

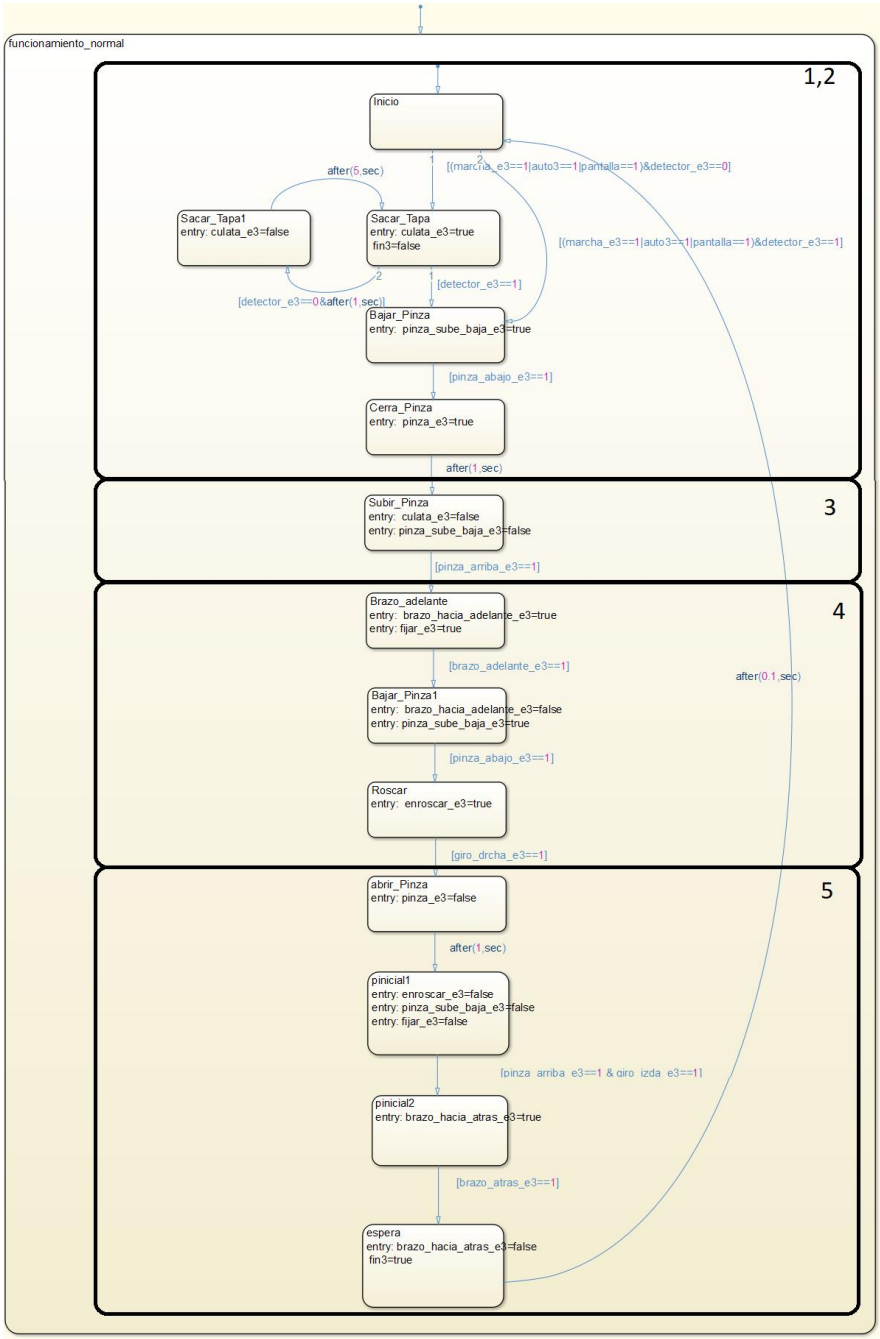


Figura 62.Programa inicial estación 3

Cabe de decir que para el resto de estaciones y el transporte habrá que seguir los mismos pasos para su control, Configurar y crear el servidor OPC, crear el bloque de control y por último añadir los bloques en Simulink del servidor OPC, todo esto y la explicación para el resto de estaciones estará explicado en el Anexo 1. La ejecución de este control se puede observar en este video.

5.5 Pantalla GUI de Matlab

También se creó una pequeña pantalla GUI, que no será exclusiva de este control, sino que también se utilizará en el control desde Schneider que se explicará más adelante.

Esta pantalla se implementó para poder facilitar la elección del tipo de pieza que se quiere fabricar, ya que, si no se implementara, la elección de pieza se elegiría con la combinación de los pulsadores de la botonera de cada estación (explicado capítulo 2).

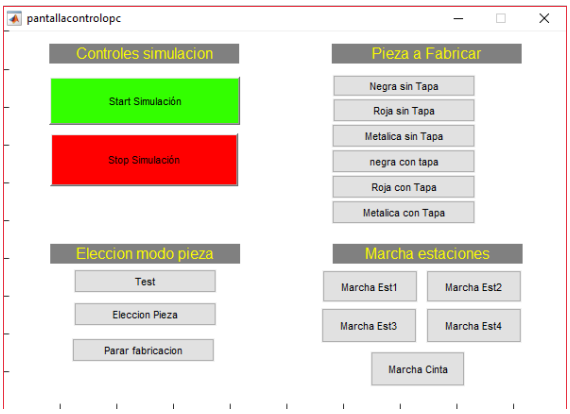


Figura 63.Pantalla Matlab

Cada uno de estos botones modifica el valor de un bloque constant que se encuentra en Simulink. Además, cada uno de estos botones tiene asociado un pequeño código que se escribirá en un fichero .m con el cual modifica los valores, que se explicará en el apartado Pedidos del Anexo 1.

5.6 Lectura de entradas y escritura de salidas en los PLC.

Después de tener configurada las estaciones y creadas todas las variables se pasa a la programación. Esta programación será muy simple ya que solo se necesita hacer llegar las variables de salida de Simulink a los actuadores de la estación y los datos de los sensores de la estación a las variables de entrada de Simulink.

Para ello se volcarán las variables de entrada de los sensores en las variables leídas mediante OPC por Simulink, y las variables salida del PLC son actualizadas con las variables escritas mediante OPC por Simulink.

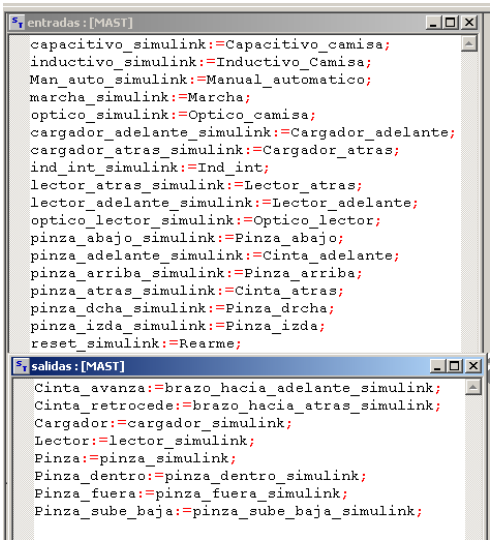


Figura 64.Programación para Estación 1

Para las estaciones 1, 3 y 4, habrá que hacer el mismo proceso, lo único que en la estación 3 también habrá que hacerlo con las variables del transporte.

Por otro lado, en la estación 2 no habrá que hacer este tipo de código ya que el control cae en manos de un SFC programada directamente en Schneider.

La programación del resto de las estaciones para este control se encuentra en los apartados sobre cada una de las estaciones en el Anexo 1.

5.7 3D World editor

Además, del interfaz de usuario (GUI) creada, también se han creado dos modelos con el editor de mundos 3D.

Estos modelos serán utilizados para realizar la supervisión del proceso de las estaciones 1 y 3. A continuación, se puede serían los modelos en 3D de las estaciones:

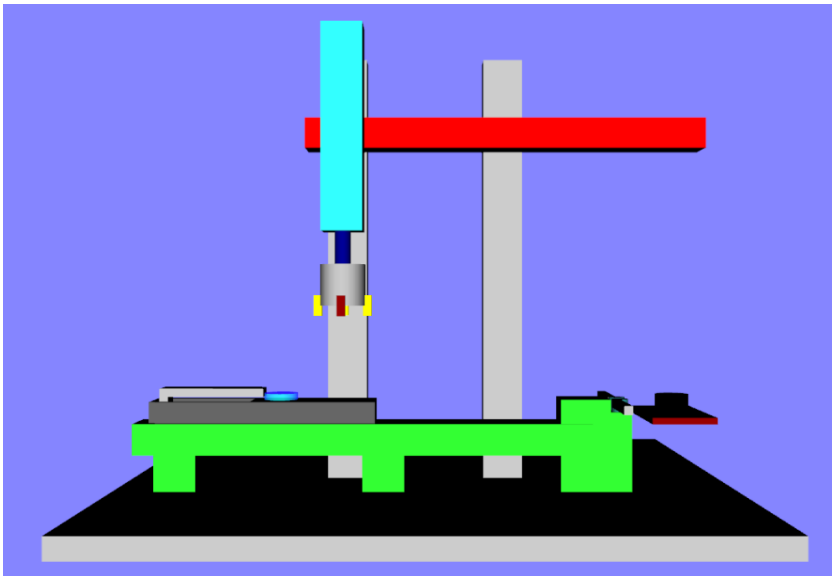


Figura 65.Esquema estación 3 en 3D World editor

La ejecución de este modelo se puede observar en este video.

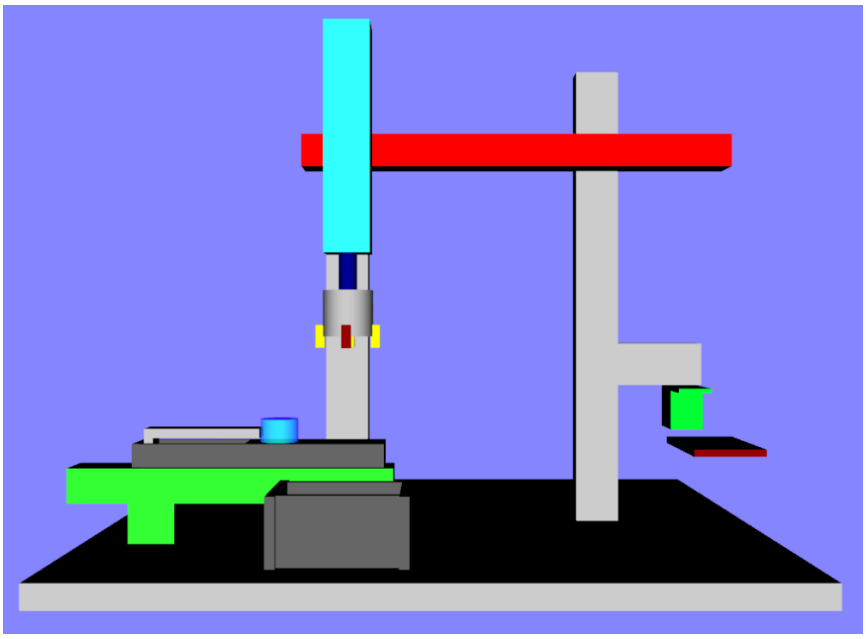


Figura 66.Esquema estación 1 en 3D World editor

La ejecución de este modelo se puede observar en este video.

Además de estos, modelos también se ha creado un mando en 3D con el cual se indicará a las estaciones el tipo de pieza que se quiere fabricar y un botón de marcha para arrancar el proceso de la estación.

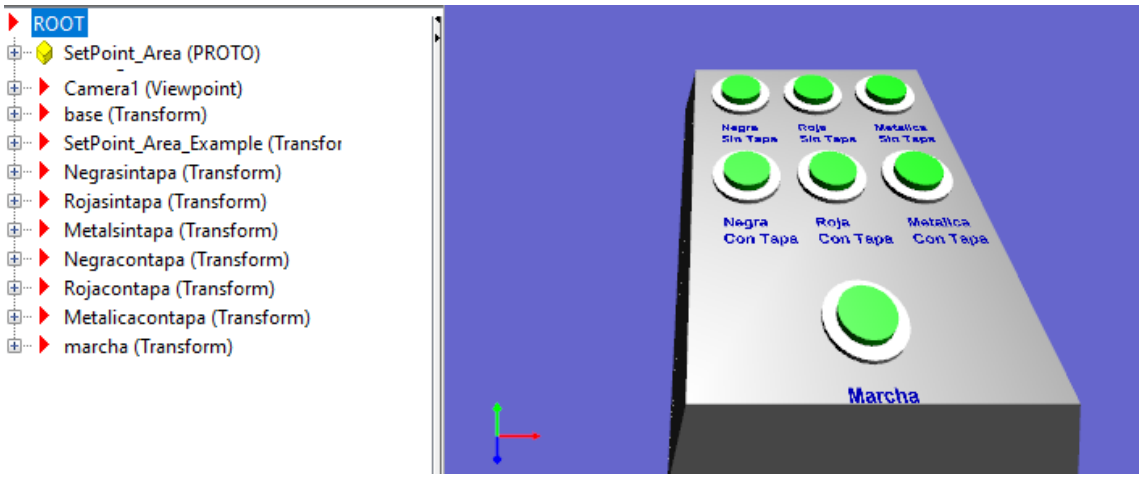


Figura 67.Esquema mando 3D

La creación de estos modelos y la programación a realizar para su funcionamiento se podrá ver en el Anexo 2.

6 Control centralizado desde Rockwell

6.1 Introducción

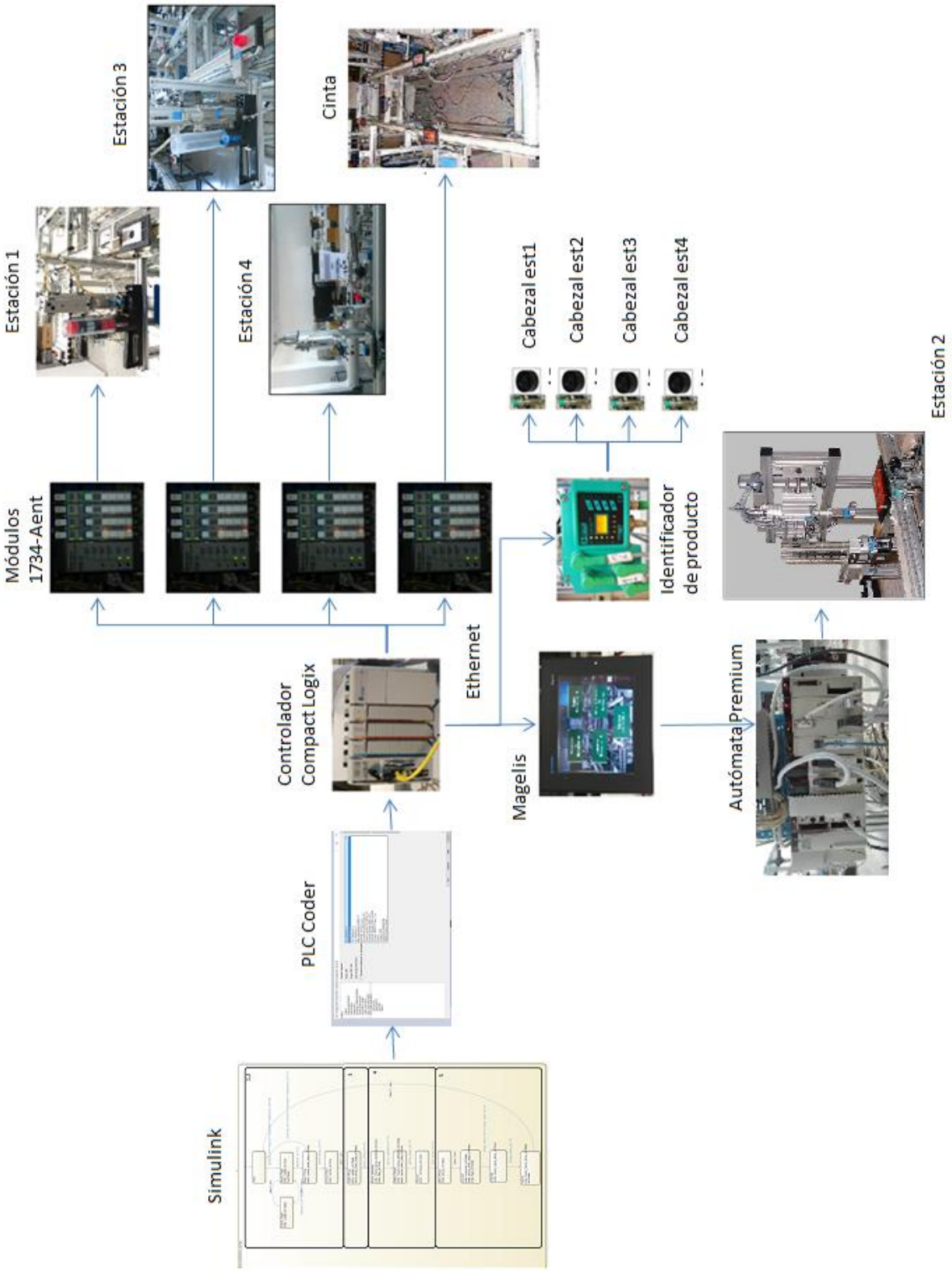


Figura 68.Esquema general control desde Rockwell

Este control consistirá en controlar la célula de fabricación, desde un autómata Compact Logix 1769-L32E de Rockwell el cual se comunicará con las estaciones 1,3, 4 y transporte gracias a

módulos de entradas/salidas por Ethernet instalados en dichas estaciones. Además, este controlador también se comunicará por Ethernet con el identificador de productos y una terminal magelis.

La programación para este autómata, se realizará en Simulink gracias a la herramienta Stateflow con la cual se han programado máquinas de estados para las estaciones y el transporte. Esas máquinas de estados se utilizarán para generar código mediante la herramienta PLC Coder de Simulink que posteriormente será descargado al autómata Rockwell

Con la ayuda de la terminal Magelis XBTGT4330 se ha realizado la supervisión de las estaciones y del transporte. Además, esta terminal, también se ha utilizado para comunicar el autómata de Rockwell con el autómata Schneider tipo Premium TSX instalado en la estación 2. El control para la estación 2 se realizará mediante un SFC creado en el programa Unity pro.

Por último, se añadió el identificador de productos, el cual dispone de unos cabezales de lectura/ escritura que se encuentran en cada una de las estaciones. Estas cabezas lectoras leerán y escribirán la memoria de los palet con el objetivo de conocer en todo momento que pieza se encuentra en el palet, y poder actualizar su estado de producción.

A continuación, se explicará la programación que se ha realizado para las estaciones 1 y 2, el resto de la programación de este control estará en el Anexo 3 del proyecto

6.2 Programación en Stateflow de la estación 1.

Para la programación desde Rockwell se partirá de los diagramas utilizados en el control desde Simulink ya explicados, siguiendo los pasos de generación de código para Rockwell.

En este control, al utilizar un único autómata y tener una magelis, se optó por añadir más elementos al control de las estaciones. En cada una de las estaciones se ha añadido diferentes modos de control como serian el modo manual, un posicionamiento inicial, un estado de emergencia y un modo de test. Además, también se ha añadido el identificador de producto, que nos facilitará la comunicación entre estaciones del tipo de pieza a fabricar.

Con el identificador de Producto se puede producir únicamente la pieza que nosotros se haya elegido o se puede producir cualquier tipo de pieza consecutivamente, con cualquier número de palets.

Las funciones del identificador de producto serán programado en Simulink, pero habrá que hacer pequeñas modificaciones a la hora de introducirlo en el programa RSlogix 5000, ya que las variables que se utilizan para la escritura y lectura de los palets no son accesibles desde Simulink, lo que se hará, será poner variables a 1 en Simulink la cuales se comprobaran cuando se necesario poner ciertos valores en las variables del identificador o, en otros casos, se comprobaran ciertos valores de las variables del identificador y si se cumple ciertas condiciones se pondrá a 1 otra variable auxiliar. De este modo, se podrá simular que el identificador de producto esta implementado en Simulink.

El identificador tiene 2 variables vectoriales los cuales se utilizarán para leer y escribir en el palet (datareadHead y datawirtehead). Cada una de las posiciones de este vector indica una característica de la pieza que se está produciendo, las cuales se modificarán a lo larga de todo el proceso.

La base de datos utilizada será la siguiente:

Distribución posición vectores Datareadhead y Datawritehead		
Posición 0	**Tipo de Pieza**	
	Negra sin tapa=1	Negra con tapa=4
	Roja sin tapa=2	Roja con tapa=5
	Metálica sin tapa=3	Metálica con tapa=6
Posición 1	**Embolo**	
	Embolo Pequeño=1	Embolo Grande=2
	Fallo embolo=9	
Posición 2	**Muelle**	
	Muelle Estándar=1	Fallo Muelle=9
Posición 3	**Tapa**	
	Tapa=1	Fallo Tapa=9
Posición 4	**Verificación OK**	
	Estación Ok=1	Fallo Verificación=9

Tabla 9.Base de Datos variables de Identificador de producto

Al añadir los otros modos de funcionamiento, el identificador de producto y la comunicación con el transporte, se realizaron modificaciones al modo automático que se explican a continuación además de explicar el resto del diagrama de estado y los modos de funcionamiento.

Modificaciones al modo automático

8.-Cuando la pieza ya elegida ya esté en el palet, la estación 1 también se encarga de controlar un desvió localizado en la estación el cual dirige el palet a la estación correspondiente para continuar el ciclo. Además, al tener implementado el identificador de producto, después de modificar el valor de desvió si es necesario, se escribirán los valores en la primera posición del vector en el palet para que la siguiente estación conozca los datos del pedido que se está realizando y está pueda continuar correctamente la producción.

9.-Si se está trabajando sin el identificador de producto (variable desconectar_e1=0), que como ya se explicó anteriormente es posible, habrá otro estado el cual controla el desvió e indica directamente a las otras estaciones que tipo de pieza se encuentra en el palet o se está fabricando.

10.-Ademas hay una variable todas que permite fabricar todo tipo de piezas (variable FabricarTodas=1) en cualquiera de los casos anteriores, ya se trabajando con identificador de producto o sin él. Eso lo consigue saltando las transiciones que comprueban si la pieza se corresponde con la pedida y como a lo largo del ciclo se van leyendo los sensores con ellos se indica al resto de estaciones el tipo de pieza que se está fabricando.

11.-Por último, hay un estado de error en caso de que la escritura realizada en el palet haya fallado.

Estas modificaciones en los diagramas Stateflow tendrán que ser realizadas en todas las máquinas de estados de todas las estaciones

Ahora se van ir explicando el resto de la máquina de estados y posteriormente se mostrará una figura en la cual estará la máquina de estados completa.

En primer lugar, se encuentra el estado de inicio, donde empieza el diagrama de estados y se inicializa alguna variable, y posteriormente se pasará al estado elección, el cual permitirá ir a los diferentes e inicializar más variables

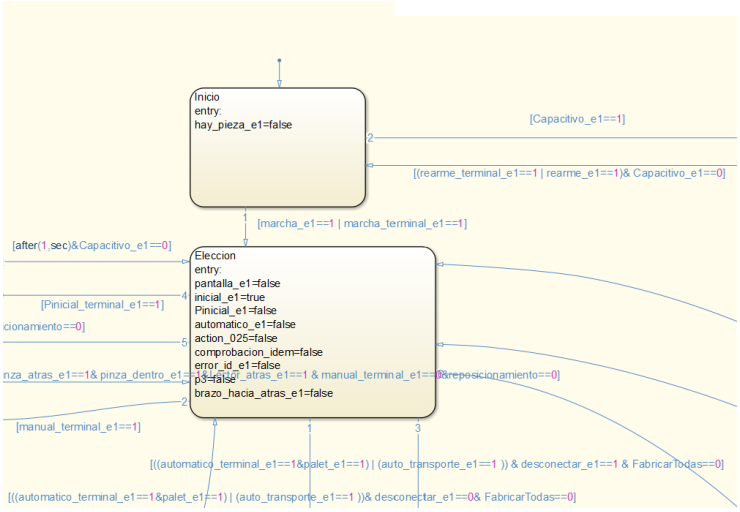


Figura 69.Diagrama correspondiente a los estados Inicio y Elección de la Estación 1

Antes de la entrada al modo automático se debe comprobar que el palet recibido no tiene nada escrito (variables de IdP iguales a 0) para el identificador, para que posteriormente no se tengas problemas en la producción.

Al no estar utilizando RSLogix, no se puede utilizar las mismas variables que se utilizarían para escribir y leer en los palet, por tanto, se han creado variables auxiliares en cada uno de los estados donde había que hacer escritura o lectura del palet para que posteriormente en el fichero de RSLogix sean utilizadas para crear funciones if que modifiquen las variables de escritura y lectura del palet en función de las variables auxiliares creadas en Simulink y que se explicaran más detenidamente más adelante.

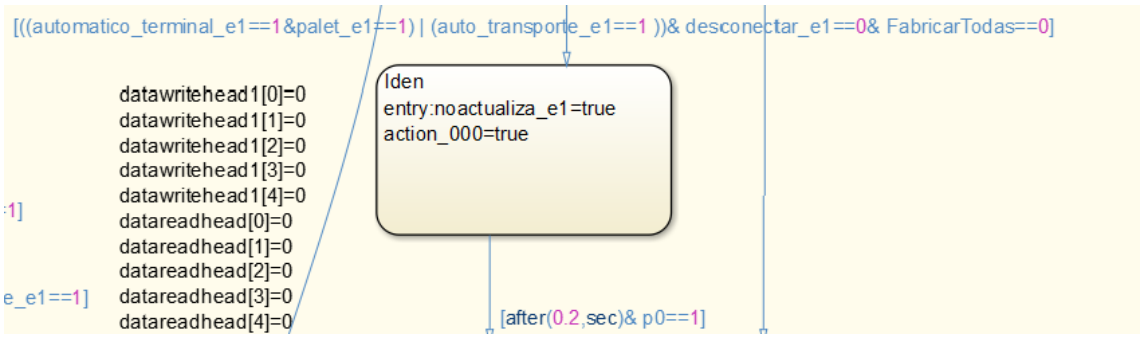


Figura 70.Comprobación de borrado de variables de identificador de producto

Pasada esta comprobación se lee el pedido recibido desde la magelis. Y se ha sacado fuera del proceso automático y se ha colocado justo antes del proceso automático.

En el modo **manual** solo cabe decir que las diferentes acciones que realizará la máquina, provendrán de la terminal magelis. Como se aprecia en la siguiente figura en este estado solo se modifican variables internas, y los actuadores son controlados por la magelis.

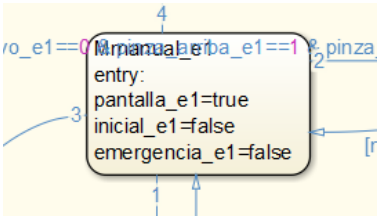


Figura 71.Parte de diagrama de Estado correspondiente al modo manual en la Estación 1

Siempre que se arranque la maquina habrá que hacer un reposicionamiento **inicial** la estación para así evitar problemas en la producción.

La posición inicial propicia para esta estación será cuando el brazo se encuentre arriba, atrás y a la dentro con la pinza abierta (sin pieza) y el cargador se encuentre recogido.

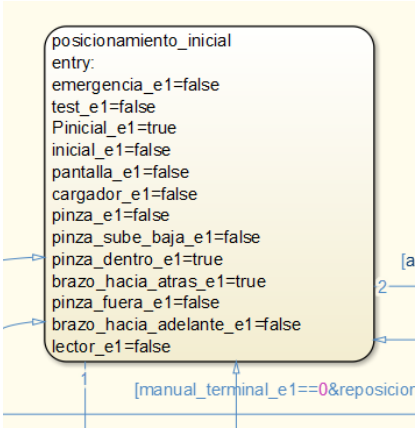


Figura 72.Parte de diagrama de Estado correspondiente Posicionamiento inicial en la Estación 1

También hay un estado de emergencia en el cual la estación para inmediatamente la producción en caso de que se hay pulsado el botón de emergencia de la regleta de la estación o de la terminal. Se podrá salir de este estado cuando sea pulsado el botón reset de la terminal.

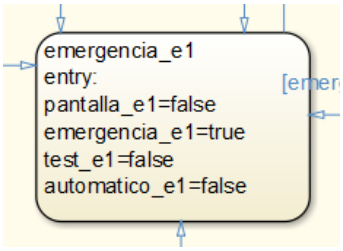


Figura 73.Estado Emergencia

Y, por último, el modo test, en el cual se han implementado todos los movimientos posibles de la estación con el objetivo de comprobar posibles problemas en ella.

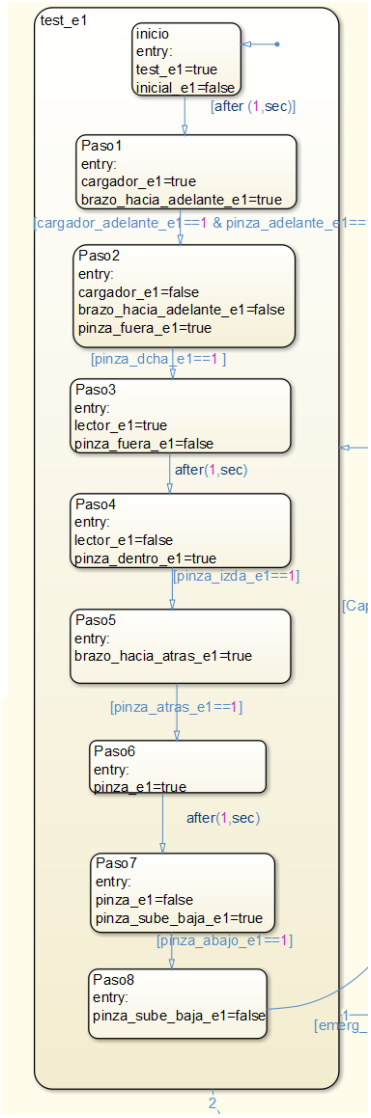


Figura 74.Modo test

Uniendo todos los estados ya explicados se forma el diagrama de estados completo para la estación 1

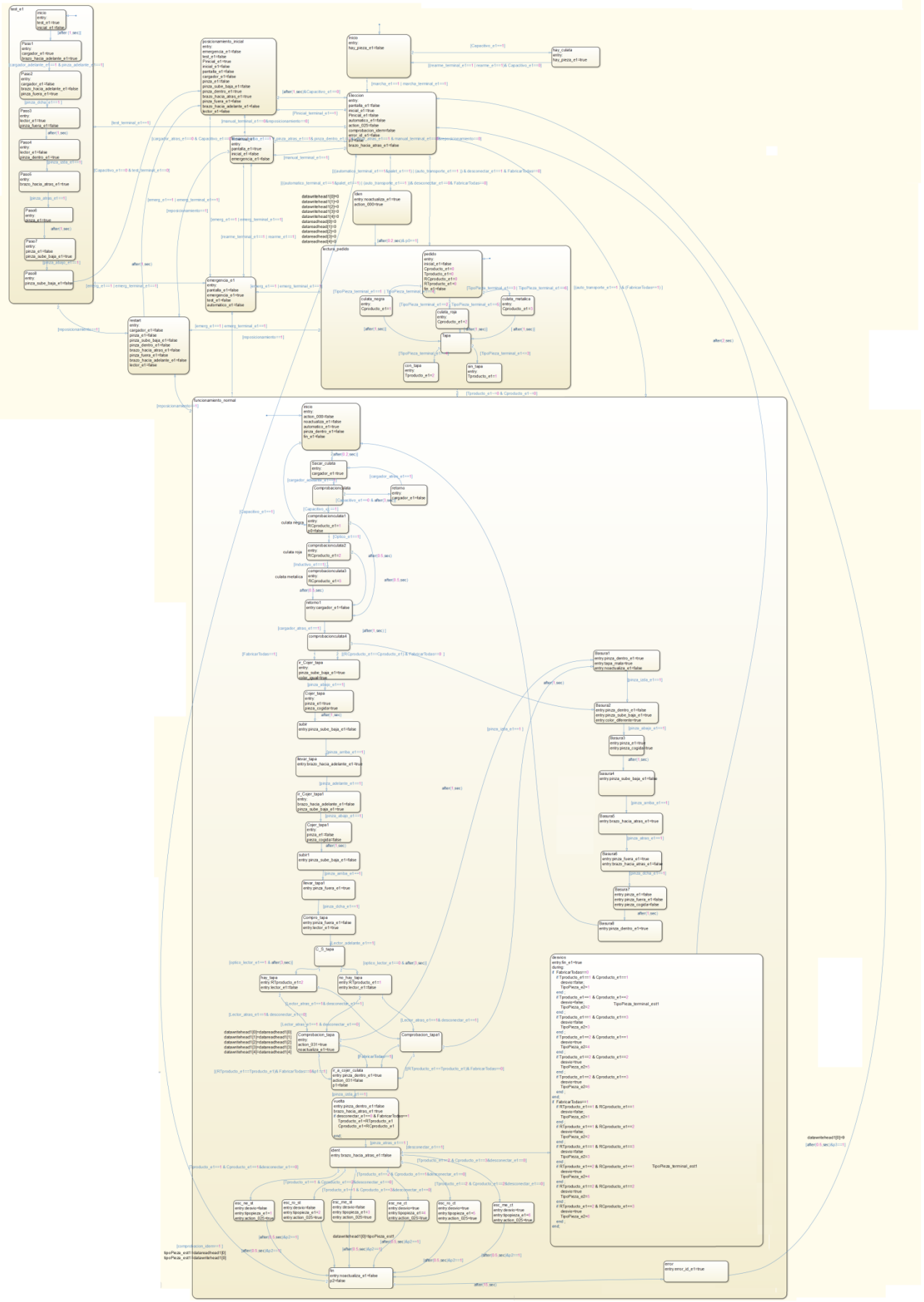


Figura 75.Diagrama de Estado completo de la Estación 1

El resto de diagramas con las modificaciones oportunas se encuentran en los apartados sobre las estaciones en el Anexo 3. La ejecución de todo este modelo se puede observar en este video.

6.3 Programación en el PLC Rockwell

El control de la estación se realizará en el programa antes mencionado RLOGIX 5000. Este programa será descargado en el autómata COMPACT LOGIX 1769-L32E, y este a su vez se comunicará por vía Ethernet/IP con el adaptador Ethernet / IP 1734-AENT.

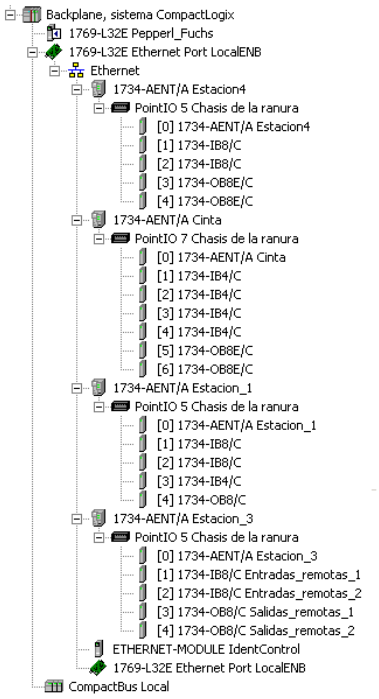


Figura 76.Configuración Global 1

Como ya se ha dicho los programas que controlan las estaciones han sido realizados en Simulink con las modificaciones antes explicadas y posteriormente, exportados a RSlogix 5000 mediante la ayuda de la herramienta PLC Coder. Pero hay algunas subrutinas que se han implementado en RSLogix.

Una vez que se han introducido en RSlogix 5000 todos los módulos de comunicaciones que se han comentado anteriormente para el correcto funcionamiento del programa, se deben crear las variables con las que se trabajaran a lo largo del proyecto. Además de esas variables, se crearon 4 vectores (2 para la lectura y escritura con el identificador de productos, y 2 para su configuración) y otras 2 variables (para activar la lectura y escritura del identificador de productos). A continuación, se explicarán las partes de la implementación para la estación 1.

El primer paso fue crear la rutina principal de nuestro programa, con la cual se controlaría la estación.

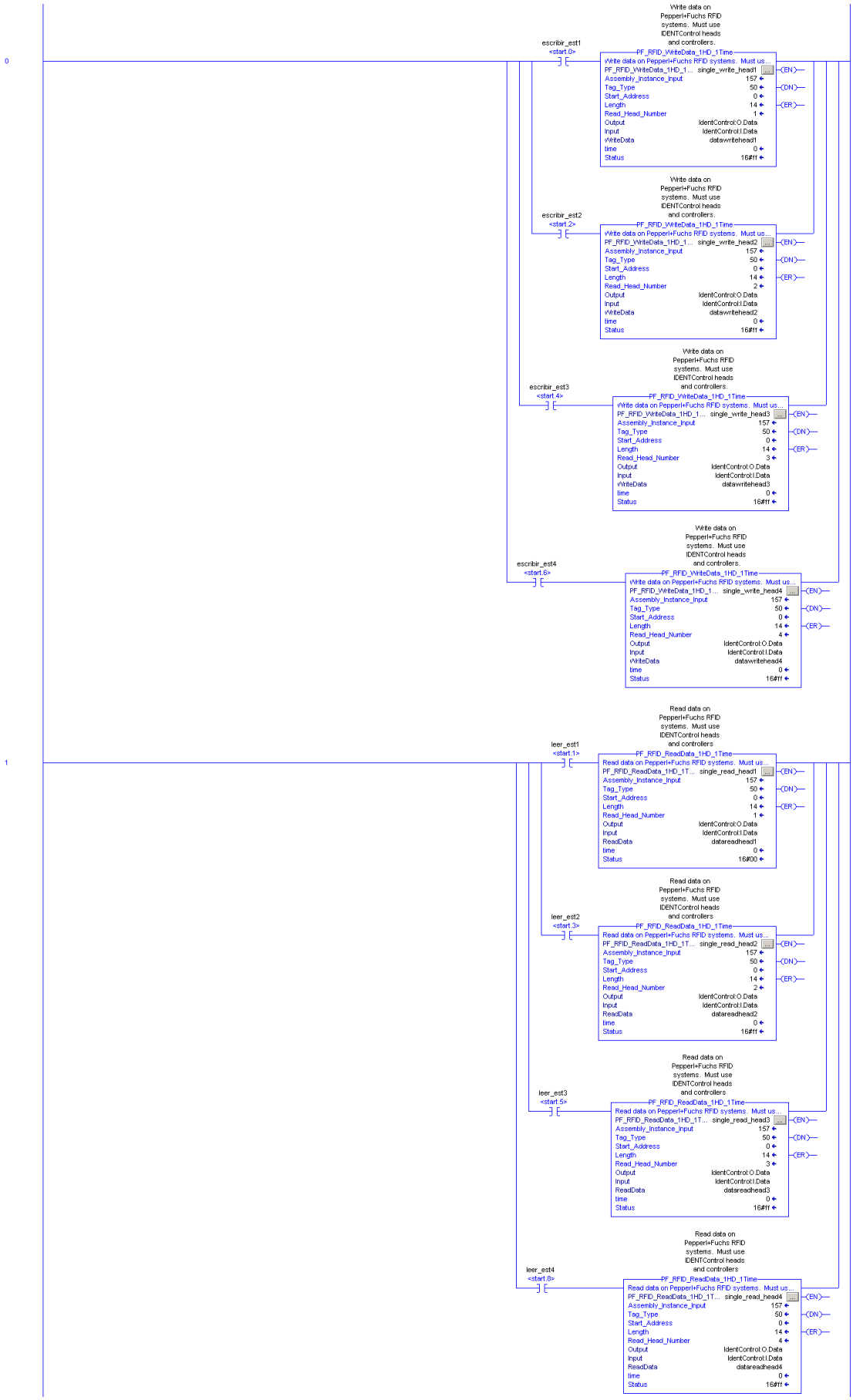


Figura 77.Programa Principal 1

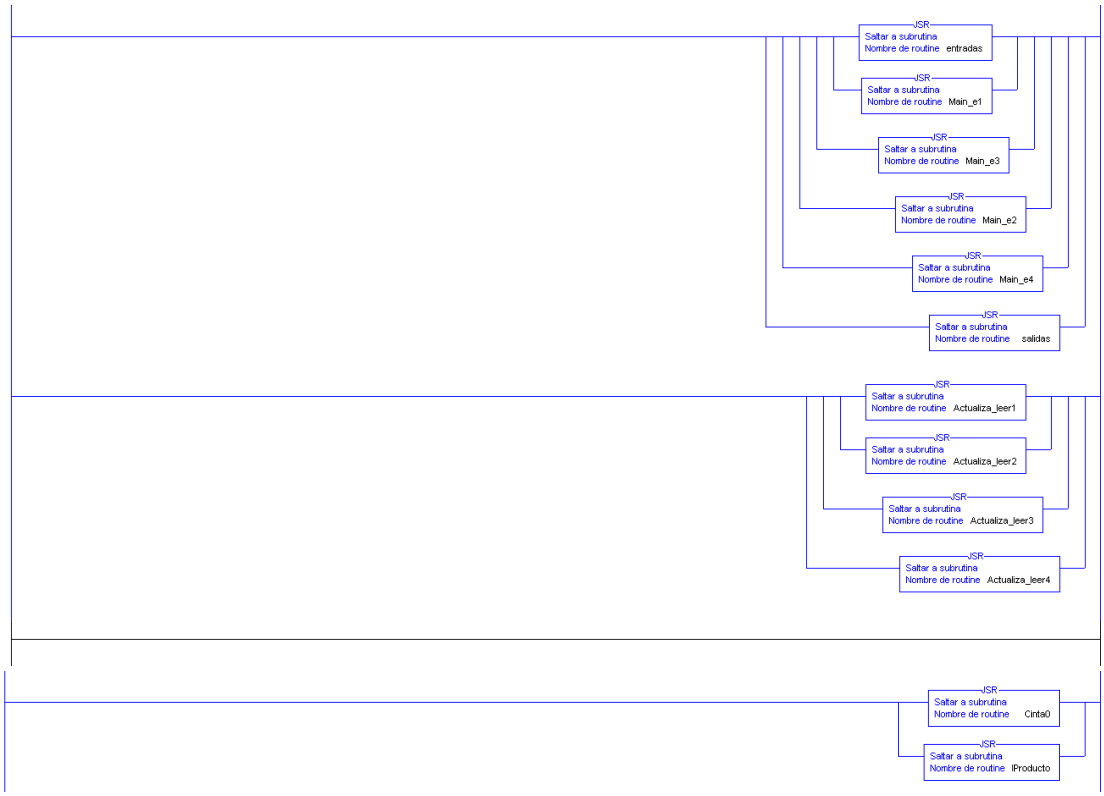


Figura 78.Programa Principal 2

Como se puede ver en esta rutina principal, se leerá la memoria de las estaciones mediante la activación del bit leer_estx, con esto se activa la instrucción Read Data, que lee la memoria del palet de cada estación y descarga su contenido en el vector DataReadHeadx. Lo mismo para escribir, se activa escribir_estx, y esto activa la instrucción Write Data. Con esto, se escribe en la memoria del palet el contenido del vector DataWriteHead1.

Por último, en esta rutina principal, utilizará el salto a otras 5 rutinas, que estarán siempre activas. La primera rutina **entradas** continua con las rutinas de las estaciones, contiene el funcionamiento completo de las estaciones las cuales serán implementados en Simulink, seguido de la rutina **salidas**, después están las rutinas de A**ctulizar_leerx**, que será la encargada de la lectura y escritura del identificador de producto, y por último la rutina **de la cinta** y la rutina **identificación**.

6.3.1 Programación para estación 1

A continuación, solo se explicará todo lo relacionado con la estación 4, ya que para el resto de estaciones tendrá la misma estructura.

Una vez que se ha realizado todo esto, En Simulink se ha generado el código de la máquina de estados de la estación 4, y posteriormente en el programa RSLogix 5000 se ha importado como una rutina.

Main_e1

Esta rutina es la que se ha importado desde Simulink, se ha partido de la máquina de estados completa de Simulink, se ha importado, y el código generado será el utilizado para controlar la estación.

```
CASE i0_Main_e1.is_c3_Main_e1 OF
    1:
        (* During 'Eleccion': '<S1>:381' *)
        IF (((i0_Main_e1.automatico_terminal_e1 AND i0_Main_e1.palet_e1) OR i0_Main_e1.auto_transporte_e1) AND ( NOT i0_Main_e1.desconectar_e1))
            (* Transition: '<S1>:197' *)
            i0_Main_e1.is_c3_Main_e1 := 2;
            PLC_CODER_TIMER(i0_Main_e1.temporalCounter_i1, 1, 0, i0_Main_e1.temp1);
            (* Outport: '<Root>/noactualiza_e1' *)
            (* Entry 'Iden': '<S1>:480' *)
            i0_Main_e1.noactualiza_e1 := 1;
            (* Outport: '<Root>/escribir_e1' *)
            i0_Main_e1.escribir_e1 := 0;
            (* Outport: '<Root>/action_000' *)
            i0_Main_e1.action_000 := 1;
        ELSIF i0_Main_e1.manual_terminal_e1 THEN
            (* Transition: '<S1>:380' *)
            i0_Main_e1.is_c3_Main_e1 := 4;
            (* Outport: '<Root>/pantalla_e1' *)
            (* Entry 'Mmanual_e1': '<S1>:378' *)
            i0_Main_e1.pantalla_e1 := 1;
            (* Outport: '<Root>/inicial_e1' *)
            i0_Main_e1.inicial_e1 := 0;
            (* Outport: '<Root>/emergencia_e1' *)
            i0_Main_e1.emergencia_e1 := 0;
        ELSIF (((i0_Main_e1.automatico_terminal_e1 AND i0_Main_e1.palet_e1) OR i0_Main_e1.auto_transporte_e1) AND i0_Main_e1.desconectar_e1) AND
            (* Transition: '<S1>:532' *)
            i0_Main_e1.is_c3_Main_e1 := 8;
            (* Entry Internal 'lectura_pedido': '<S1>:196' *)
            (* Transition: '<S1>:340' *)
            i0_Main_e1.is_lectura_pedido := 6;
            (* Outport: '<Root>/inicial_e1' *)
            (* Entry 'pedido': '<S1>:339' *)
            i0_Main_e1.inicial_e1 := 0;
            i0_Main_e1.Cproducto_e1 := 0.0;
            i0_Main_e1.Tproducto_e1 := 0.0;
            i0_Main_e1.RCproducto_e1 := 0.0;
            i0_Main_e1.RTproducto_e1 := 0.0;
```

Figura 79.Codigo Main_e1

Entradas /Salidas

Al realizar la exportación desde Simulink y la importación en RSLogix, como ya se ha explicado se crea una variable vectorial cuyo nombre es i0_nombre_chart (para la estación 1 será i0_Main_e1) y las variables de Simulink serán distintas posiciones de ese vector creado en la importación. Por ejemplo: i0_Main_e1.pinza_adelante_e1.

Por lo tanto, hay que coger esas variables de entrada y salida de Simulink e igualarlas a las variables de entrada y salida del RSLogix para poder mover la estación. Un ejemplo de entrada seria: i0_Main_e1.pinza_adelante_e1:=cinta_adelante_est1;

Y otro de salida seria: Pinza_est1:=i0_Main_e1.pinza_e1;

Las rutinas Salidas y entradas son 2 rutinas en las cuales están implementadas esas igualaciones explicadas anteriormente.

```
Cinta_avanza_est1:=i0_Main_el.brazo_hacia_adelante_el;
Cinta_retrocede_est1:=i0_Main_el.brazo_hacia_atras_el;
Pinza_fuera_est1:=i0_Main_el.pinza_fuera_el;
Pinza_dentro_est1:=i0_Main_el.pinza_dentro_el;
Pinza_sube_baja_est1:=i0_Main_el.pinza_sube_baja_el;
Cargador_est1:=i0_Main_el.cargador_el;
Pinza_est1:=i0_Main_el.pinza_el;
Lector_est1:=i0_Main_el.lector_el;
F4Gemma_est1:=i0_Main_el.pantalla_el;
F1Gemma_est1:=i0_Main_el.automatico_el;
F6Gemma_est1:=i0_Main_el.test_el;
D1Gemma_est1:=i0_Main_el.emergencia_el;
A6Gemma_est1:=i0_Main_el.Pinicial_el;
A1Gemma_est1:=i0_Main_el.inicial_el;
(*manual el*)
IF(i0_Main_el.pantalla_el) THEN
i0_Main_el.brazo_hacia_adelante_el:=MaCintaAvanza_est1_terminal;
i0_Main_el.brazo_hacia_atras_el:=MaCintaRetrocede_est1_terminal;
i0_Main_el.pinza_fuera_el:=MaPinza_fuera_est1_terminal;
i0_Main_el.pinza_dentro_el:=MaPinza_dentro_est1_terminal;
i0_Main_el.pinza_sube_baja_el:=MaPinza_sube_baja_est1_terminal;
i0_Main_el.cargador_el:=MaCargador_est1_terminal;
i0_Main_el.pinza_el:=MaPinza_est1_terminal;
i0_Main_el.lector_el:=MaLector_est1_terminal;

Cinta_avanza_est1:=i0_Main_el.brazo_hacia_adelante_el;
Cinta_retrocede_est1:=i0_Main_el.brazo_hacia_atras_el;
Pinza_fuera_est1:=i0_Main_el.pinza_fuera_el;
Pinza_dentro_est1:=i0_Main_el.pinza_dentro_el;
Pinza_sube_baja_est1:=i0_Main_el.pinza_sube_baja_el;
Cargador_est1:=i0_Main_el.cargador_el;
Pinza_est1:=i0_Main_el.pinza_el;
Lector_est1:=i0_Main_el.lector_el;
END_IF;
```

```
(*Entradas estacion 1*)
i0_Main_el.desconectar_el:=Desconectar_el;
i0_Main_el.marcha_el:=Marcha_est1;
i0_Main_el.pinza_atras_el:=cinta_atras_est1;
i0_Main_el.pinza_adelante_el:=cinta_adelante_est1;
i0_Main_el.pinza_izda_el:=Pinza_izda_est1;
i0_Main_el.pinza_dcha_el:=Pinza_dcha_est1;
i0_Main_el.pinza_arriba_el:=Pinza_arriba_est1;
i0_Main_el.pinza_abajo_el:=Pinza_abajo_est1;
i0_Main_el.cargador_adelante_el:=Cargador_adelante_est1;
i0_Main_el.cargador_atras_el:=Cargador_atras_est1;
i0_Main_el.Rearme_el:=Rearme_est1;
i0_Main_el.Capacitivo_el:=Capacitivo_camisa_est1;
i0_Main_el.Optico_el:=Optico_camisa_est1;
i0_Main_el.Inductivo_el:=Inductivo_camisa_est1;
i0_Main_el.Lector_adelante_el:=Lector_adelante_est1;
i0_Main_el.Lector_atras_el:=Lector_atras_est1;
i0_Main_el.optico_lector_el:=Optico_lector_est1;

i0_Main_el.pantalla_el:= F4Gemma_est1;
i0_Main_el.marcha_terminal_el:=Marcha_est1_terminal;
i0_Main_el.rearme_terminal_el:=rearme_est1_terminal;
i0_Main_el.automatico_terminal_el:=automatico_est1_terminal;
i0_Main_el.manual_terminal_el:=manual_est1_terminal;
i0_Main_el.TipoPieza_terminal_el:=TipoPieza_Terminal_est1;
i0_Main_el.emerg_terminal_el:=Emergencia_est1_terminal;
i0_Main_el.Pinicial_terminal_el:=Pinicial_est1_terminal;
i0_Main_el.test_terminal_el:=Test_est1_terminal;
i0_Main_el.Palet_el:=Palet_est1;
i0_Main_el.auto_transporte_el:=i0_Cinta0.automatico_terminal_el;
i0_Main_el.FabricarTodas:=FabricarTodasPiezas;
i0_Main_el.reposicionamiento:=reposicionamiento_est1;
```

Figura 80.Programación Rutinas salidas/entradas estación 1

Rutina Identificación

El identificador tiene 2 variables vectoriales para cada estación, las cuales serán utilizadas para leer y escribir en el palet (para esta estación datareadHead1 y datawirtehead1). Cada una de las posiciones de este vector indicará una característica de la pieza que se está produciendo, las cuales se irán modificando a lo larga de todo el proceso.

El modulo del identificador de producto ha sido también programado en Simulink, pero habrá que realizar pequeños cambios (rutina identificación) a la hora de introducirlo en el programa RSlogix 5000, ya que las variables que se utilizan para la escritura y lectura de los palets no son accesibles desde Simulink.

Por lo tanto, se han utilizado variables auxiliares que se utilizarán en Simulink las cuales se modificaran dependiendo de las lecturas del identificador de producto y, viceversa, habrá otras variables que ocasionaran modificaciones en las variables de escritura del identificador.

Para la estación 1 habrá que realizar 4 bloques if los cuales modificaran los valores de las variables auxiliares y de las variables del identificador. Son los siguientes:

```
if i0_Main_e1.is_c3_Main_e1 = 2 THEN
    if i0_Main_e1.action_000 THEN
    datawriteheadl[0]:=0;
    datawriteheadl[1]:=0;
    datawriteheadl[2]:=0;
    datawriteheadl[3]:=0;
    datawriteheadl[4]:=0;
    datareadheadl[0]:=0;
    datareadheadl[1]:=0;
    datareadheadl[2]:=0;
    datareadheadl[3]:=0;
    datareadheadl[4]:=0;
    i0_Main_e1.p0:=1;
    END_IF;
END_IF;

IF i0_Main_e1.is_funcionamiento_normal = 14 THEN
    if i0_Main_e1.action_031 THEN
    datawriteheadl[0]:=datareadheadl[0];
    datawriteheadl[1]:=datareadheadl[1];
    datawriteheadl[2]:=datareadheadl[2];
    datawriteheadl[3]:=datareadheadl[3];
    datawriteheadl[4]:=datareadheadl[4];
    i0_Main_e1.p1:=1;
    END_IF;
END_IF;

IF i0_Main_e1.is_funcionamiento_normal>= 25 AND i0_Main_e1.is_funcionamiento_normal<= 30THEN
    if i0_Main_e1.action_025 THEN
    datawriteheadl[0]:=i0_Main_e1.tipopieza_e1;
    i0_Main_e1.p2:=1;
    END_IF;
END_IF;

IF i0_Main_e1.is_funcionamiento_normal = 24 THEN
    if i0_Main_e1.error_id_e1 THEN
    datawriteheadl[0]:=9;
    i0_Main_e1.p3:=1;
    END_IF;
END_IF;

if i0_Main_e1.is_funcionamiento_normal=2 then
    if i0_Main_e1.tipoPieza_e1=datareadheadl[0] and i0_Main_e1.tipoPieza_e1=datawriteheadl[0] then
        i0_Main_e1.comprobacion_idem:=1;
        i0_Main_e1.comprobacion_idemt:=1;
    end_if;
end_if;
```

Figura 81.Programación rutina identificación para la estación 1

Como se puede observar, delante de cada una de los bloques if otro bloque if, el cual comprueba en la programación Main_e1 la variable io_Main_e1.is_funcionamiento_normal, la cual indica en qué estado del proceso se encuentra el proceso.

Rutina Actualizar_leer1

Esta rutina es utilizada para actualizar los valores de las variables del identificador de producto, hay una rutina igual por cada estación.

La función principal de esta rutina será la lectura de la memoria incluida en el Palet de la estación 1. Se actualizará cada segundo. Con esta rutina, también se conseguido hacer la escritura de la memoria del Palet mediante el botón (Escribir Palet1) incluido en la pantalla de la Magelis. La última función de esta rutina, será la escritura mediante programa.

Al final de cada estación se activará la variable noactualiza_est1 durante un periodo superior a 500ms, consiguiendo que se ejecute la operación de escritura en la memoria del Palet.

La rutina Actualizar_leer1 es la siguiente:

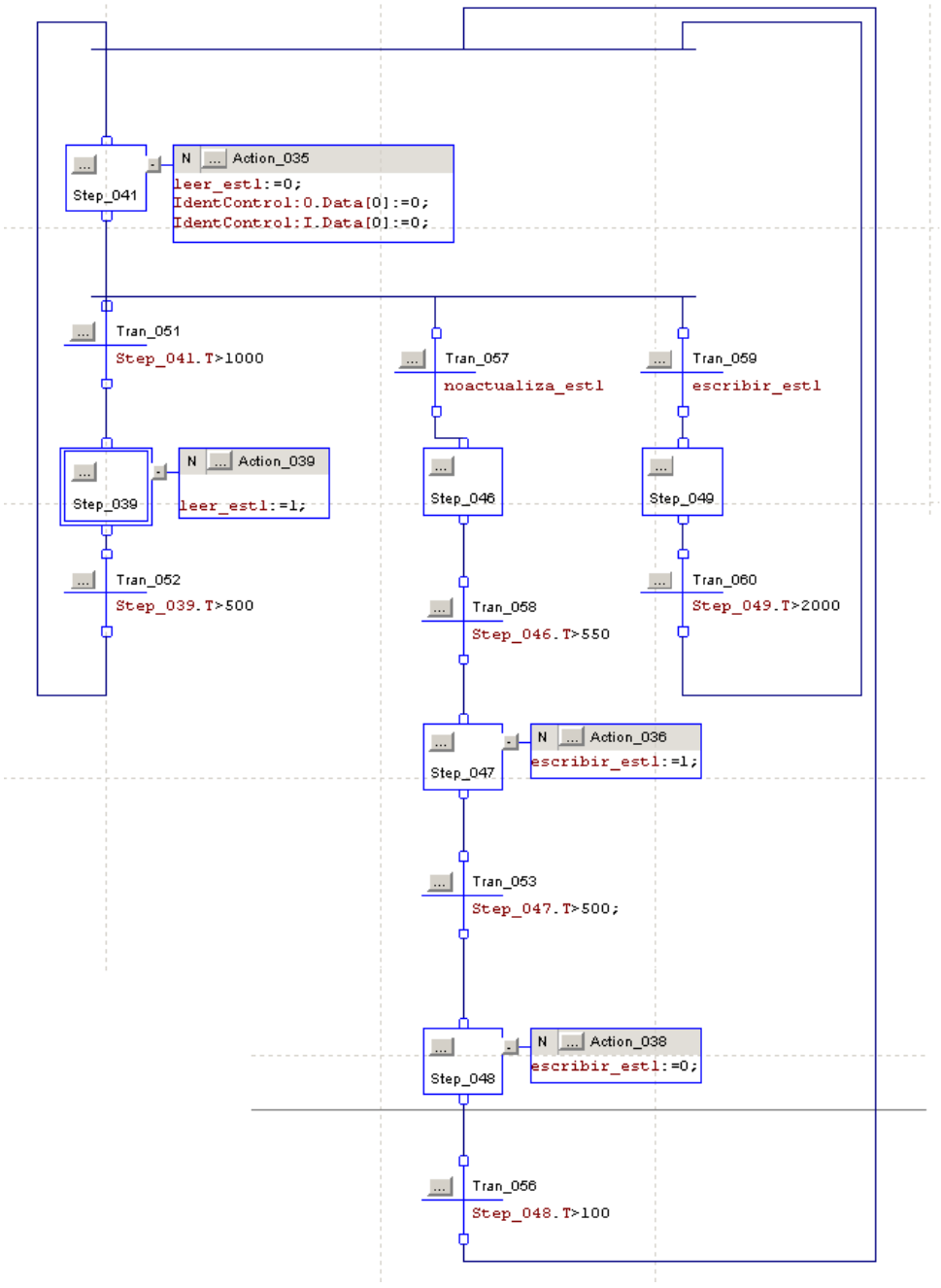


Figura 82.Rutina actualizar Estación 1

6.4 Supervisión mediante Magelis.

En este punto, se hablará de la supervisión de la estación mediante una Magelis XBTGT4330. El programa ha sido creado en Vijeo Designer y posteriormente descargado en ella. A continuación, se describirán las distintas pantallas creadas en la Magelis.

- **Pantalla Menú:**

Lo primero que se ha creado fue un menú, que nos llevará a las distintas pantallas del programa. Desde el Menú puedes acceder al control de la estación, al funcionamiento de la

estación, a la guía gemma, al identificador de productos y finalmente a la elección de pieza a fabricar, como se muestra en la siguiente figura:

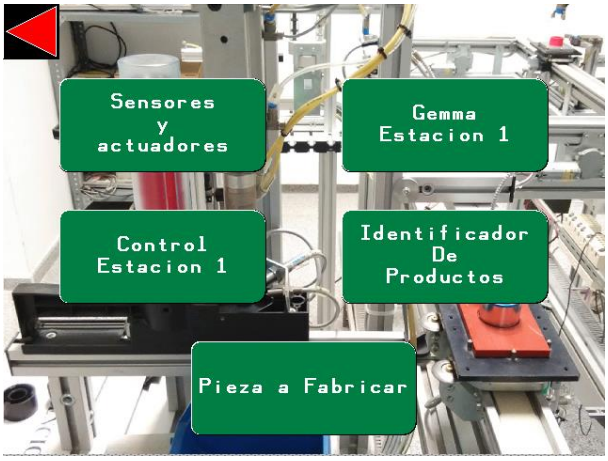


Figura 83.Pantalla menú Estación 1

- **Pantalla Sensores y Actuadores.**

En esta pantalla se puede ver una imagen de la estación 1, que cambia en tiempo real dependiendo de las diferentes posiciones de la estación. Esta pantalla también incorpora unos pilotos que indican si los actuadores o los sensores están activados o desactivados. Incluye también una imagen actualizada de la pieza que hay actualmente en el Palet. Y por último incorpora un piloto de emergencia, que se cambia a color rojo si ocurre alguna emergencia.

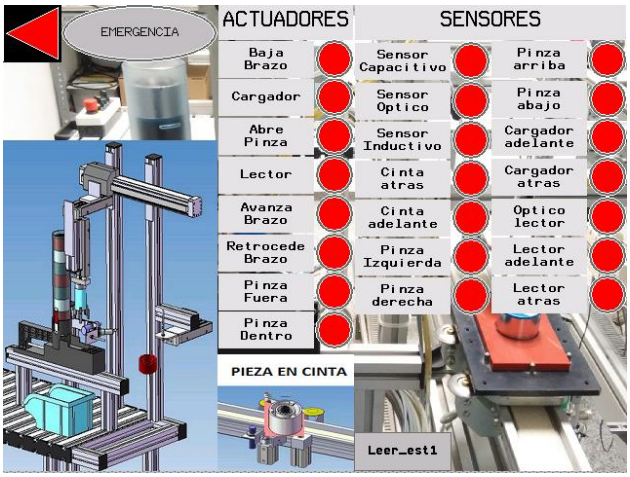


Figura 84.Pantalla Sensores y Actuadores de la Estación 1

- **Pantalla Control Estación 1**

En esta pantalla se puede ver diferentes botones para controlar la estación en todo momento. En esta pantalla se podrá cambiar entre los diferentes modos de funcionamiento, ya sea modo automático, modo manual, o test. Además de eso incorpora también los sensores y actuadores que aparecen en la pantalla anterior para cuando se controle la estación desde el modo manual.

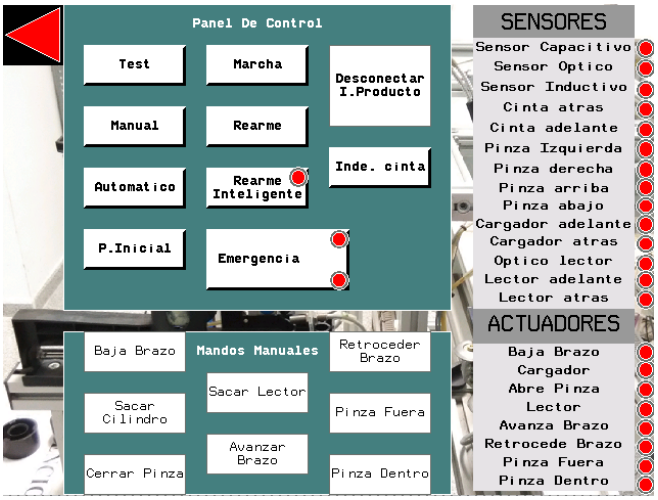


Figura 85.Pantalla de Control de la Estación 1

- **Pantalla guía gemma**

Esta pantalla indicaría el funcionamiento global del programa controlado por la pantalla anterior, al igual que la pantalla anterior, esto tampoco está implementado en nuestro programa.

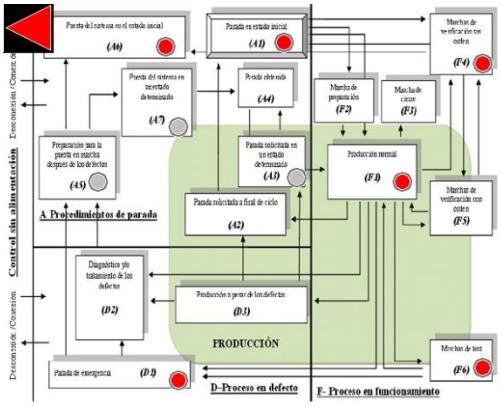


Figura 86.Pantalla Guía Gemma Estación 1

- **Pantalla Identificador de productos**

Con esta pantalla se ha conseguido leer y escribir mediante RFID en la memoria incluida en el palet. Está compuesta de 3 botones (leer Palet1, escribir Palet1 y borrar Palet1).

- Leer Palet1→ Con este botón se podrá leer la memoria del Palet 1, descargando sus datos en el vector DataReadHead1, que es mostrado en la parte derecha de la pantalla.
- Escribir Palet1→ Con este botón se escribirá en la memoria los datos que estén escritos en el vector DataWriteHead1, este vector se puede modificar, ya que está definido en la parte central de la pantalla.
- Borrar Palet1→ Con este botón se cambian todos los valores del vector DaraWriteHead1 a 0, y después se activa el botón Escribir Palet 1 automáticamente, consiguiendo así, poner a 0 los 9 primeros valores de la memoria por RFID.

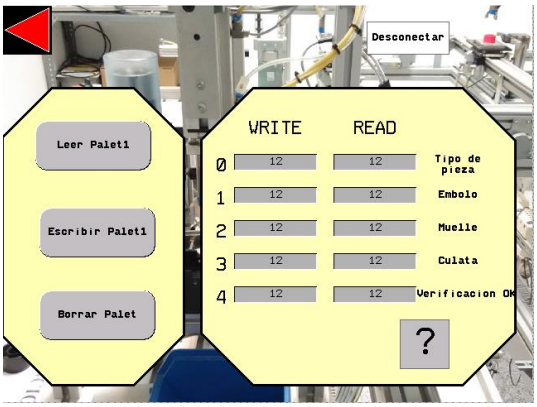


Figura 87.Pantalla Identificador de Productos Estación 1

Esta pantalla también cuenta con dos ventanas emergentes, la primera solo salta si se intenta activar leer Palet1 y escribir Palet1 al mismo tiempo. Esto es debido a que no puedes escribir y leer al mismo tiempo, solo se puede realizar una acción al mismo tiempo. La segunda es informativa, y salta al pulsar el interrogante, informando del significado de los diferentes números que pudieran aparecer en la memoria del palet.

Además de estas operaciones, a la derecha de la imagen, aparece el vector DataReadHead1, que informa del contenido del Palet, ya que es la lectura de su memoria. La base de datos utilizada en el identificador en la ventana emergente es la siguiente:

Posición 0	**Tipo de Pieza**	
	Negra=1	Negra con Tapa=4
	Pieza Rosa=2	Rosa con Tapa=5
	Pieza Metálica=3	Metálica con Tapa=6
	Fallo Pieza=9	
Posición 1	**Embolo**	
	Embolo Tipo1=1	Embolo Tipo2=2
	Fallo Embolo=9	
Posición 2	**Muelle**	
	Muelle estándar=1	Fallo Muelle=9
Posición 3	**Culata**	
	Culata=1	Fallo Culata=9
Posición 4	**Verificación OK**	
	Estación 4 Ok=1	Fallo Verificación=9

Figura 88.Ventana emergente base de datos de la memoria

- **Pantalla de Fabricación**

Mediante esta pantalla se elegirá el tipo de pieza a fabricar. Una vez seleccionada la pieza a fabricar, se pulsará el botón de marcha de la botonera, o el propio botón de marcha disponible en esta misma pantalla. También incorpora un botón “Fabricar Todas Las Piezas” que hace que salgan todas las piezas, sin desechar ninguna.

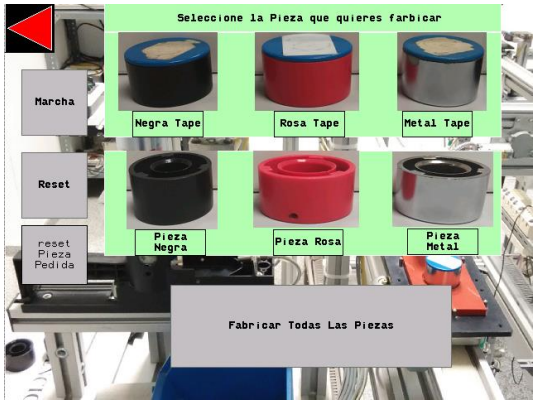


Figura 89.Pantalla de fabricación.

7 Control descentralizado con PLCs Schneider

7.1 Introducción

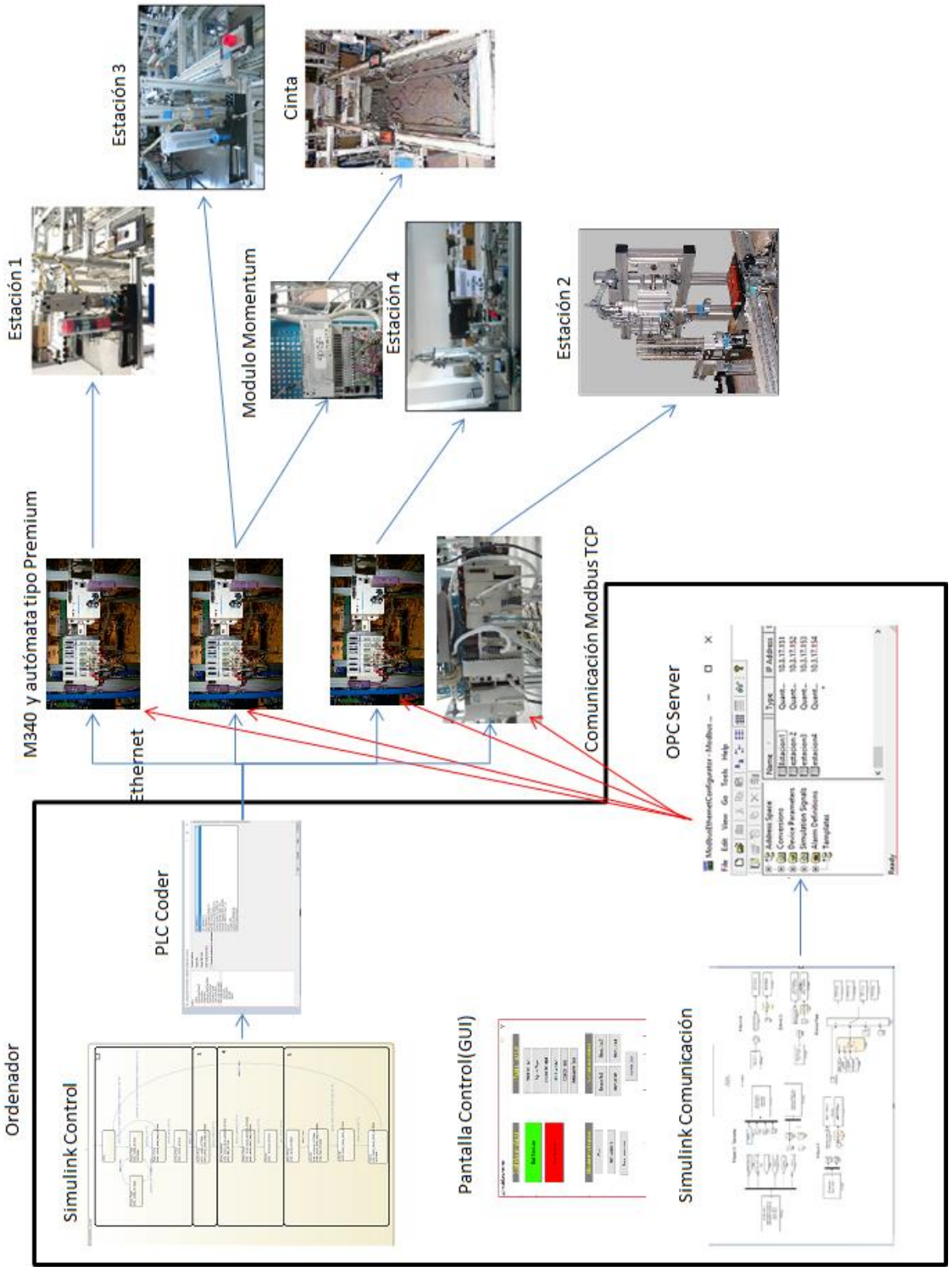


Figura 90.Esquema general Control desde Schneider

El objetivo es realizar un control completo de toda la célula de fabricación desde los autómatas Schneider instalados en cada una de ellas.

La programación de Simulink, se ha realizado gracias a la herramienta Stateflow con la cual se han programado máquinas de estados para cada una de las estaciones y del transporte, con ellas se generará código para los autómatas Schneider mediante la herramienta PLC Coder de Simulink.

Pero hay un problema, ya que Simulink no genera código directamente para los autómatas que controlan las estaciones, por lo que se ha procedido a la generación de código para el fabricante OMRON, donde posteriormente este código se introdujo en Unity Pro, adaptándolo y creando las variables necesarias, para el correcto funcionamiento en dichos autómatas.

Por otro lado, para conseguir un control global de toda la célula, las estaciones tienen que estar en todo momento comunicadas con el transporte, eso se consiguió mediante un servidor OPC por Modbus TCP y un programa de Simulink.

Este programa se encargará de transmitir variables a las estaciones para que estás conozcan cuando ha llegado el palet y que la estación pueda empezar el proceso de fabricación, e indicará al transporte cuando ha finalizado el proceso la estación para que el palet se marcha de la estación donde se encontraba.

Por último, se utilizará un interfaz de usuario en Matlab, para poner en marcha las estaciones y para elegir la pieza a fabricar.

7.2 Servidor OPC

Partiendo de la misma configuración, habiendo creado los dispositivos con sus direcciones IP, ahora se pasará a crear las variables que serán utilizadas por el servidor.

Las variables habrá que reducirlas a estas, ya que, este servidor solo se encargará de comunicar entre estaciones el tipo de pieza y cuando acaba y empieza el proceso de la estación.

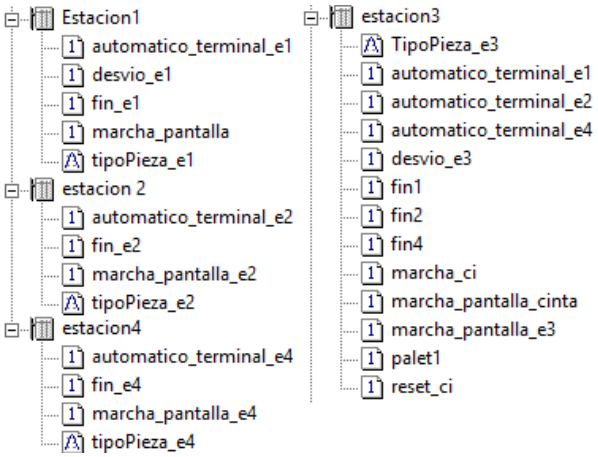


Figura 91.Variables creadas en el servidor OPC

7.3 Simulink

Una vez que se han creado todas las variables en el Servidor OPC, se creará el sistema de comunicación entre estaciones en Simulink. La comunicación consistirá en mandar a las estaciones el tipo de pieza a fabricar y cuando se encuentra el palet en dicha estación se enviará la instrucción de marcha y estas devolverán una variable que indicará cuando ha acabado el proceso.

En el esquema de Simulink solo se encontrarán los bloques del OPC (OPC Read y OPC Write), bloques goto/from para unir salidas de las estaciones con entradas de otras, y, por último, el esquema que ya se explicó para elegir el tipo de pieza que se quiere explicar. Esto no se volverá a explicar ya que esta explicado en el capítulo 4. El esquema quedaría así:

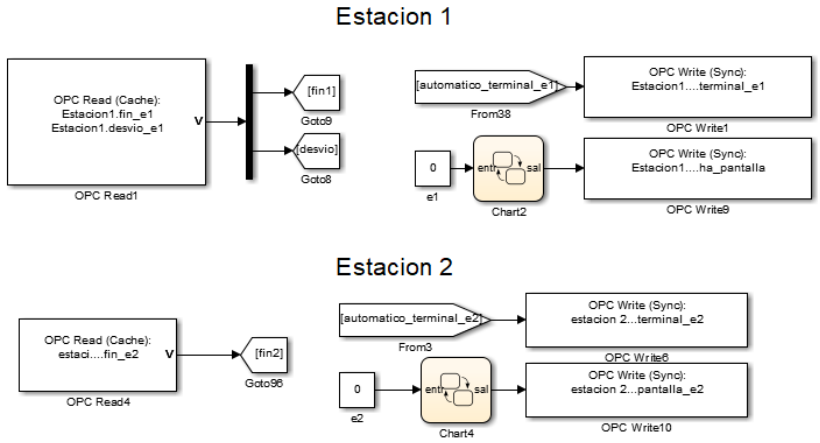


Figura 92.Esquema comunicación Simulink 1

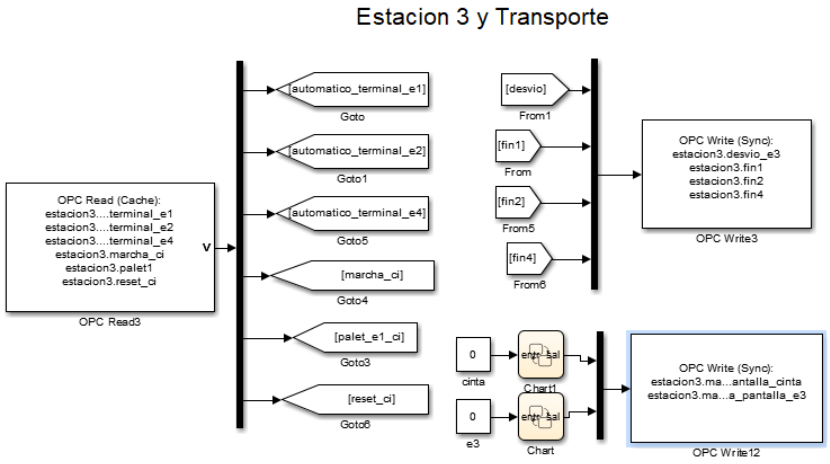


Figura 93.Esquema de comunicación Simulink 2

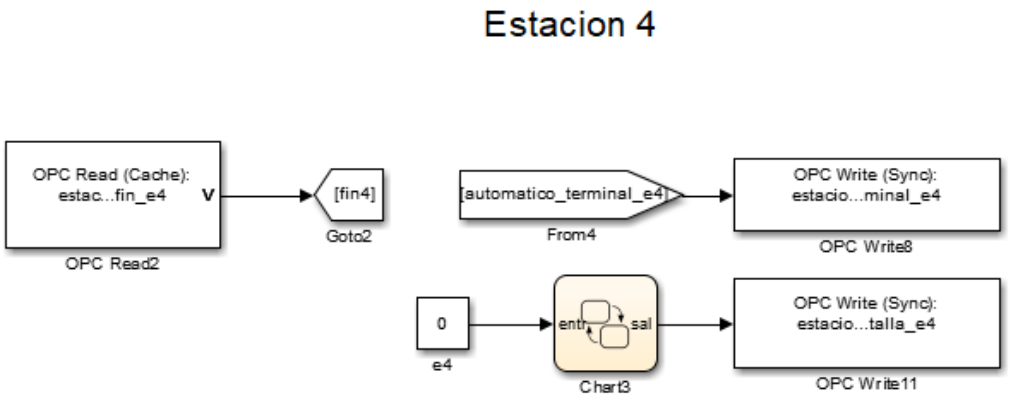


Figura 94.Esquema de comunicación Simulink 3

Al utilizar también la pantalla GUI, se utilizará el sistema de elección de pieza, explicado ya en el apartado 4.

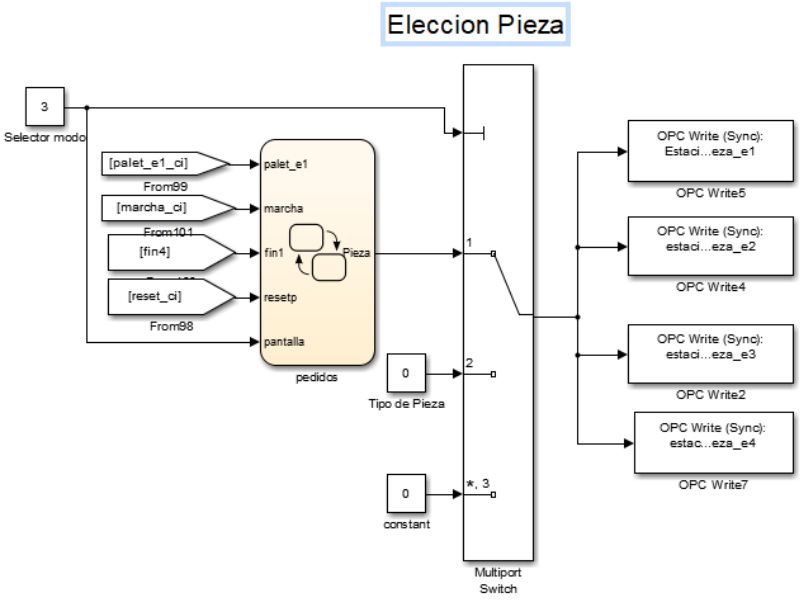


Figura 95.Esquema Elección Pieza

7.3.1 Estación 2

En la estación 2, ya no se empleará la SFC que se ha utilizado en el resto de controles si no que la será sustituida por una máquina de estados hecha en Simulink y exportada para ese autómata.

Hay algunas variables propias de dicha estación que no se podrán utilizar, pero si se podrán utilizar variables auxiliares que hagan la misma función que las variables originales.

Como seria las variable SMOVE (motor, 0, 90, 9, 481, 4000,0) que es utilizada para mover el motor paso a paso instalado en la estación. Lo que se hizo fue que cuando haya que utilizar dicha variable en la máquina de estado se pondrá a 1 una variable (smove1) la cual hará una vez este el código en Schneider que el motor se mueva.

Esto habría que hacerlo también con las variables Q0609, Q0605… y el resto de variables de ese estilo.

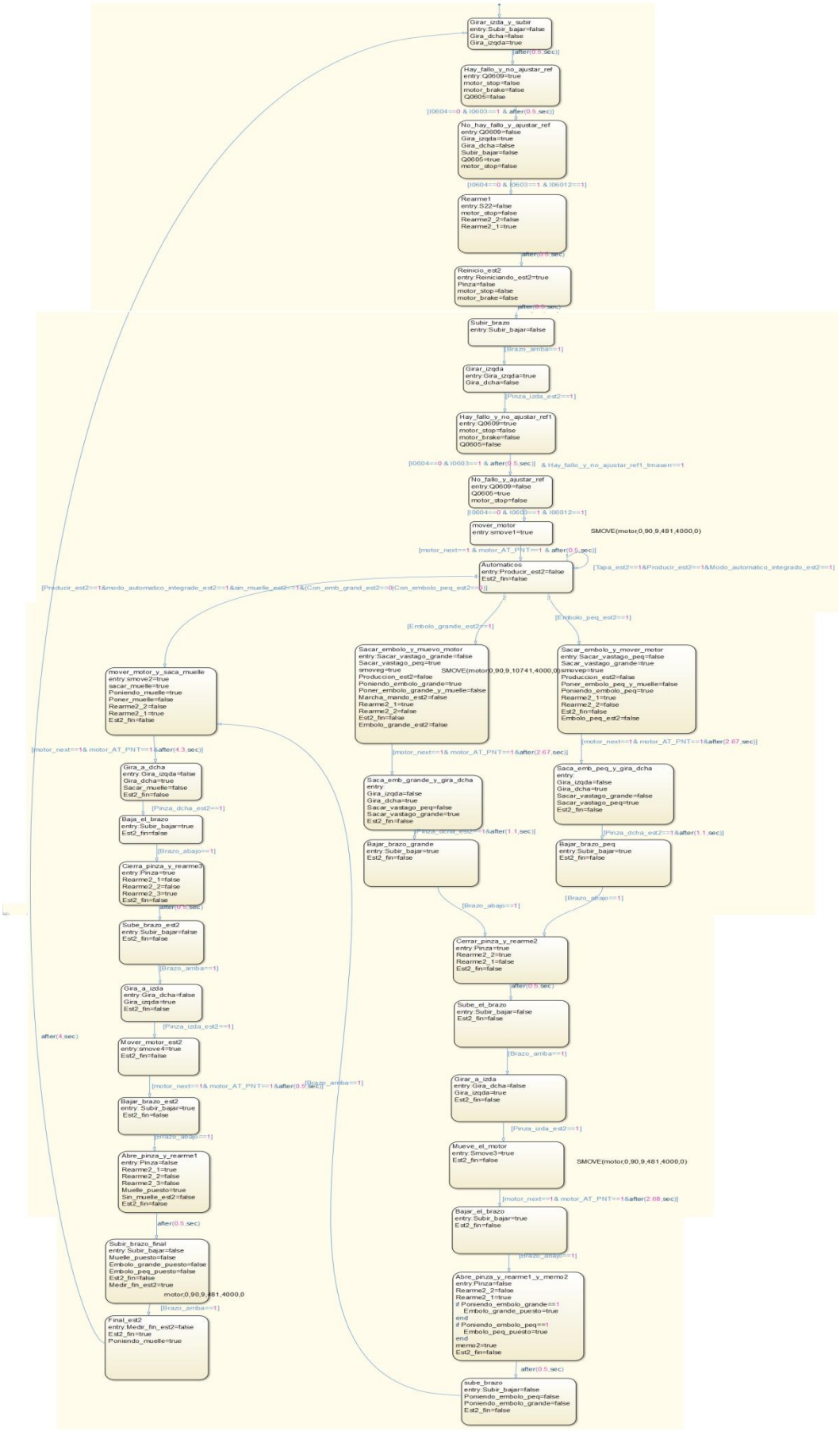


Figura 96.Máquina de estados Estación 2

Como se puede ver solo se ha explicado la programación para la estación 2, ya que, para el resto de estaciones, la programación de Simulink utilizada será la misma que para el control desde Simulink, y se puede encontrar en el correspondiente capítulo sobre Control centralizado desde Simulink mediante OPC y en el Anexo 1

Pantalla GUI de Matlab

Se utilizara la misma pantalla GUI que para el control desde Simulink. Esta pantalla se implementó para poder facilitar la elección del tipo de pieza que se quiere fabricar. La función de los botones será la misma que para el control desde Simulink.

7.4 Unity

En primer lugar para la programación en Unity, habrá que configurar las estaciones, esta configuración será la misma que para el control desde Simulink donde cada una de las estaciones será controlada por un autómata Schneider salvo para la estación 3 que también se encargara del control de la cinta de transporte.

A continuación, se explica la programación que se ha seguido para hacer el control de la estación 1, para el resto de estaciones ser realizara del mismo modo.

7.4.1 Programación Estación 2

Para la programación de este control se partirá del modelo inicial en Simulink de cada una de las estaciones. Se realizará la generación de código para OMRON Studio y posteriormente se modificará para Schneider.

La estación 2 solo puede ser controlada por un autómata Schneider, como es el caso, por tanto, se ha optado por implementar en Simulink la SFC que se utilizó en el resto de controles, por lo tanto, y para no realizar modificaciones, en esta estación se dispondrá 2 bloques funcionales DFB.

El primero de ellos será con la máquina de estados inicial utilizado en el resto de controles y la segunda con el código generado por Simulink de la máquina de estados de la estación, explicada en la memoria.

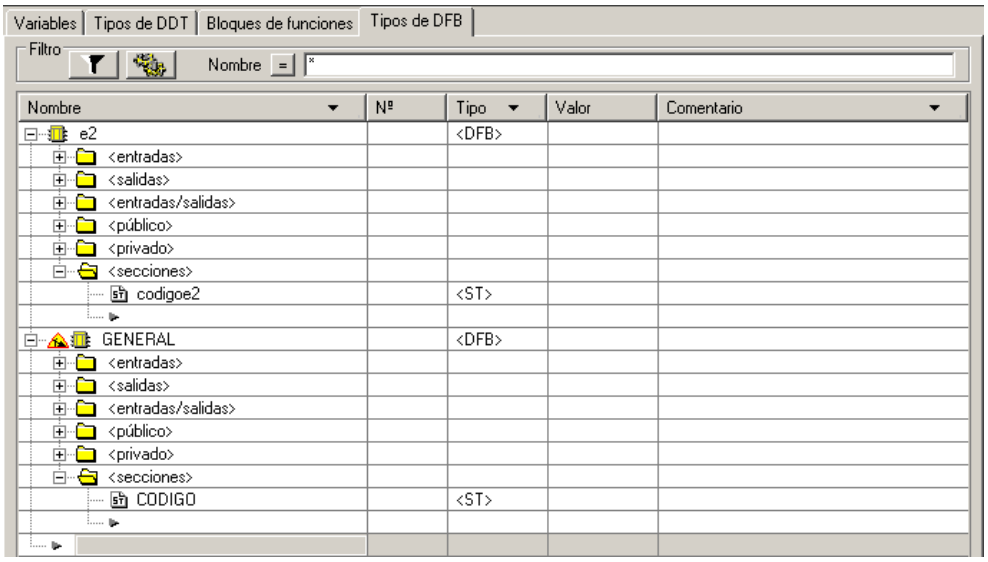
Las nuevas variables utilizadas para la comunicación por el OPC serán las siguientes:

Nombre		Tipo	Dirección	
OPC	Unity		OPC	Unity
Automatico_terminal_e2	Automatico_terminal_e2	Booleana	1	%M0
tipoPieza_e2	tipoPieza	Real	2	%MW1
Marcha_pantalla	pantalla	Booleana	15	%M14
Fin_e2	Est2_fin	Booleana	291	%M291

Tabla 10.Relación entre variables entradas del servidor OPC y Unity estación control Schneider

Después de haber creado las variables y tener el código generado con las modificaciones antes comentadas, se pasará a implementar ese código en un bloque funcional de tipo DFB, los cuales serán creados del mismo modo que en la estación 1

Una vez creados ambos bloques, se creará una nueva sección en cada bloque donde será copiado el código final tras la exportación de Simulink.



Nombre	Nº	Tipo	Valor	Comentario
e2		<DFB>		
<entradas>				
<salidas>				
<entradas/salidas>				
<público>				
<privado>				
<secciones>				
codigoe2		<ST>		
GENERAL		<DFB>		
<entradas>				
<salidas>				
<entradas/salidas>				
<público>				
<privado>				
<secciones>				
CODIGO		<ST>		

Figura 97.Implementación bloques 1 funcionales DFB

Después de esto, hay que indicar al bloque que variables de las escritas en el código son salidas, entradas o ambas o si son auxiliares (privadas) y de qué tipo son en cada uno de los bloques.

GENERAL		<DFB>
<entradas>		
fin_e2	1	BOOL
Pieza	2	REAL
automatico_terminal_e2	3	BOOL
man_auto_e2	4	BOOL
marcha_e2	5	BOOL
pantalla	6	BOOL
<salidas>		
poner_embolo_grande	1	BOOL
poner_embolo_peq	2	BOOL

GENERAL		<DFB>
<entradas>		
<salidas>		
<entradas/salidas>		
<público>		
<privado>		
ssMethodType		DINT
is_active_c5_Main_e2		UINT
is_c5_Main_e2		UINT
temporizador		TON
done		BOOL

Figura 98.Variables Bloque general control estación 2

e2		<DFB>
<entradas>		
Tapa_est2	1	BOOL
Brazo_abajo	2	BOOL
modo_automatico_integrado_e...	3	BOOL
Pinza_dcha_est2	4	BOOL
Pinza_izda_est2	5	BOOL
I0604	8	BOOL
I0603	9	BOOL
Brazo_arriba	10	BOOL
Con_embolo_peq_est2	11	BOOL
Con_emb_grand_est2	13	BOOL
I06012	15	BOOL
<salidas>		
<entradas/salidas>		
Embolo_grande_est2	6	EBOOL
Embolo_peq_est2	7	EBOOL
motor	32	T_STEPP...
<público>		
<privado>		
ssMethodType		DINT
is_active_c3_e2		UINT
is_c3_e2		UINT
temporizador		TON
done		BOOL
<secciones>		
codigoe2		<ST>

e2		<DFB>
<entradas>		
<salidas>		
Subir_bajar	1	BOOL
Gira_dcha	2	BOOL
Gira_izqda	3	BOOL
Muelle_puesto	4	BOOL
Embolo_peq_puesto	5	BOOL
Poniendo_embolo_peq	8	BOOL
Poniendo_embolo_grande	9	BOOL
Producir_est2	10	BOOL
memo2	11	BOOL
Q0609	12	BOOL
Sacar_vastago_grande	13	BOOL
Sacar_vastago_peq	14	BOOL
Marcha_mando_est2	15	BOOL
Poner_embolo_grande_y_muelle	16	BOOL
Poniendo_muelle	17	BOOL
Rearme2_1	18	BOOL
Rearme2_2	19	BOOL
Embolo_grande_puesto	20	BOOL
Medir_fin_est2	21	BOOL
Poner_embolo_peq_y_muelle	22	BOOL
S22	23	BOOL
Est2_fin	24	BOOL
sacar_muelle	25	BOOL
Q0605	26	BOOL
Poner_muelle	27	BOOL
Pinza	28	BOOL
Rearme2_3	29	BOOL
Reiniciando_est2	30	BOOL
Sin_muelle_est2	31	BOOL
<entradas/salidas>		

Figura 99.Variables Bloque e2 control estación 2

Para finalizar la programación, solo nos falta crear la rutina principal. Esta rutina será de tipo LD en la cual aparecerá el bloque que se han creado antes, solo bastara con unir las entradas y salidas con las variables correspondientes que controlan la estación.

Figura 100.Rutina principal control Estación 2

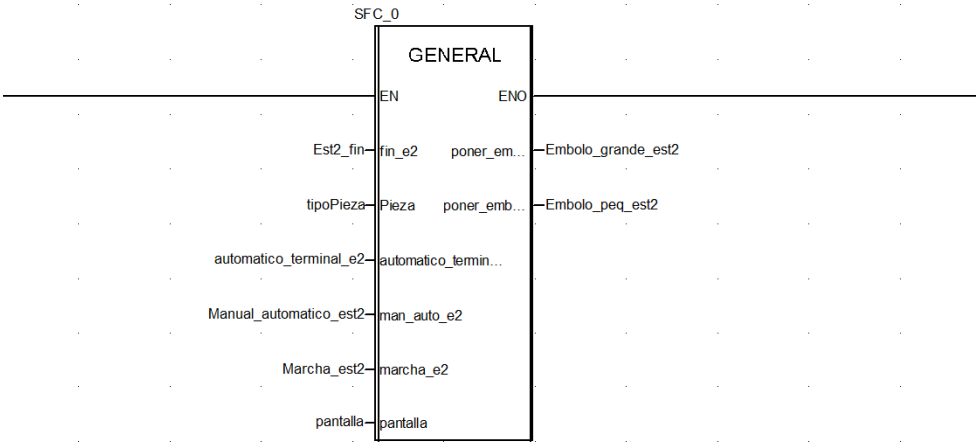


Figura 101.Rutina principal control general Estación 2

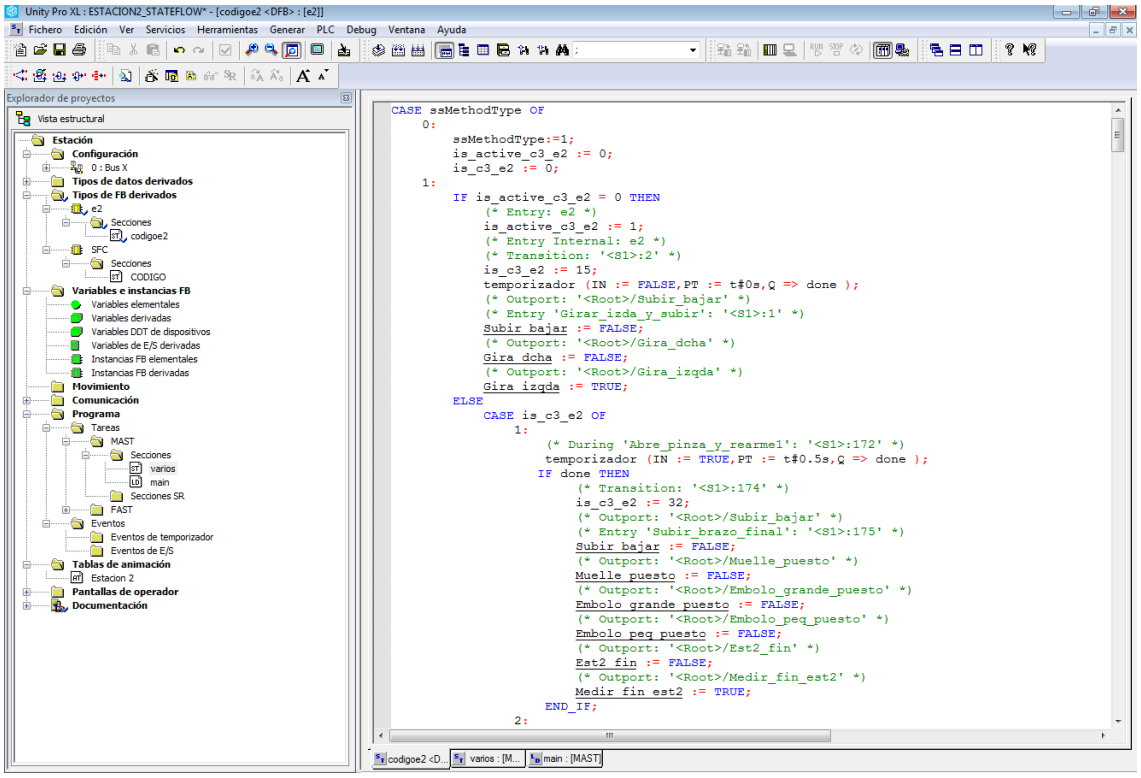


Figura 102.Código bloque DFB rutina principal estación 2

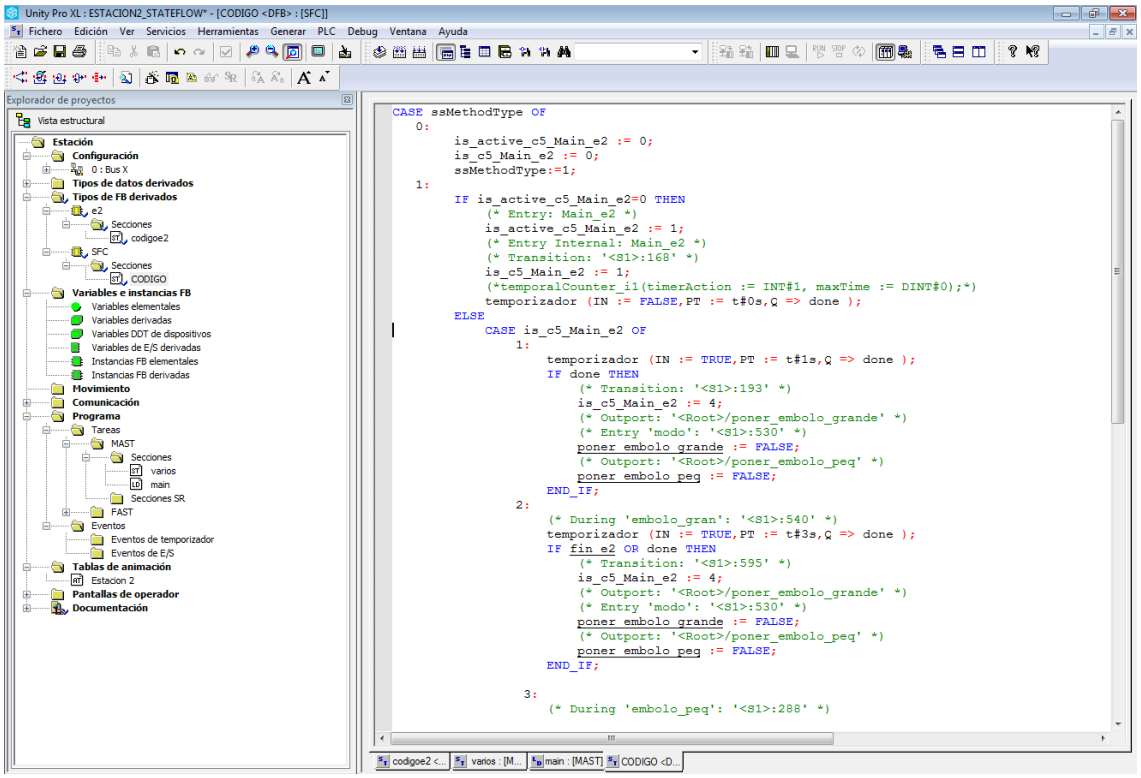


Figura 103.Código bloque DFB control general estación 2

La programación en Unity para el resto de estaciones y el transporte se encuentra en el Anexo 4 del proyecto. La ejecución de este control se pueden observar en este video.

8 Comparación controles y análisis del código generado

8.1 Comparación entre los diferentes controles

Para comparar ambos controles se ha optado por calcular el número de ciclos por segundo que realiza el PLC. En función del número de ciclos de ambas implementaciones se verá cual es más rápida y se obtendrán las conclusiones pertinentes.

8.1.1 Comparación en control desde Rockwell mediante un SFC y el código generado por StateFlow

A continuación, se van a comparar desde el PLC de Rockwell dos implementaciones. La primera de ellas consiste en una SFC tomada del primer trabajo indicado en la Bibliografía y la segunda será el código generado de la máquina de estados realizada en Stateflow en nuestro proyecto.

Para calcular el número de ciclos se añadió una variable al control, la cual se incrementa en 1 cada vez que el PLC realiza un ciclo. Posteriormente se calculó cuanto tiempo costaba que esa variable se incrementará cierta cantidad de veces. Posteriormente, se ha dividido el número de ciclos por el tiempo trascurrido

El PLC de Rockwell necesito 20.78 seg en incrementarla variable a 10000 en el SFC. En cambio, tardo 24.82 seg con la implementación importada del Stateflow. Para calcular los ciclos por segundos se hará lo siguiente

- SFC: 10000 Ciclos/24.28 Sg=402.90 ciclos/seg
- Stateflow: 10000 Ciclos/20.78 Sg=481.23 ciclos/seg

Como se puede ver, al utilizar la SFC, el autómata realiza menos ciclos por segundos, que al utilizar el código del Stateflow, debido a que la SFC es menos eficiente que el código ST generado para Rockwell incluso aunque esté tenga más modos implementados de funcionamiento. Estas ejecuciones se pueden observar en este video.

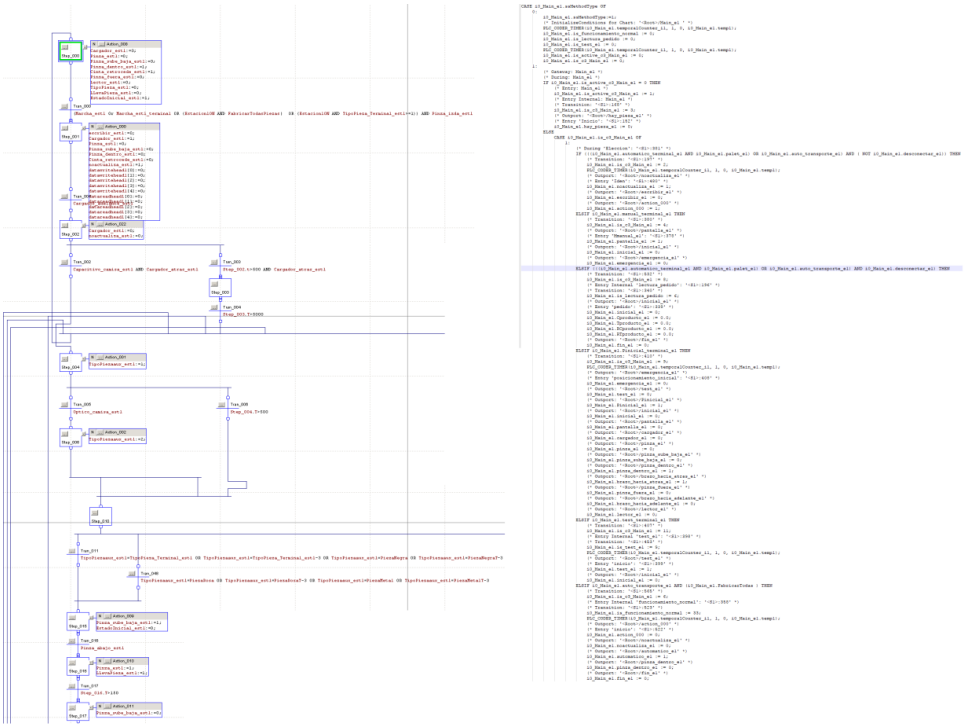


Figura 104.SFC comparada con Código generado en Rockwell

8.1.2 Comparación en control desde Schneider mediante un SFC y el código generado por StateFlow

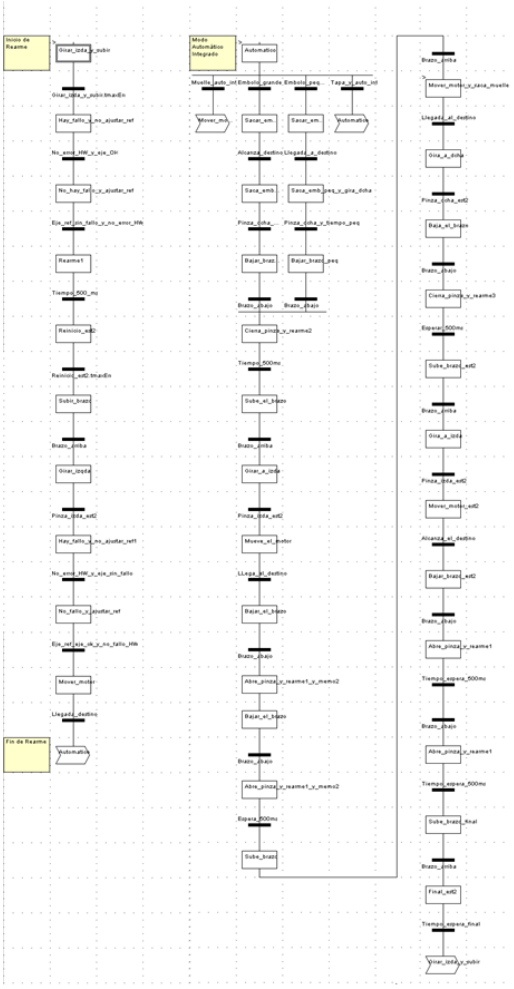
A continuación, se van a comparar desde el PLC de Schneider dos implementaciones. La primera de ellas consiste en la SFC utilizada en el control desde Simulink y la segunda será el código generado de la máquina de estados realizada en Stateflow.

El conteo de la variable se realizara del mismo modo que en la comparación anterior, con una variable, la cual se incrementará en cada ciclo del PLC.

El PLC de Schneider necesito 16.38 seg en incrementarla variable a 10000 en el SFC. En cambio, tardo 16.76 seg con la implementación importada del Stateflow. Para calcular los ciclos por segundos se hará lo siguiente

- SFC
  10000/16.38=610 ciclos/seg
- Stateflow
  10000/16.76=596.66 ciclos/seg

Como se puede ver, al utilizar la SFC, el autómata realiza unos pocos ciclos más por segundos, que al utilizar el código del Stateflow, esto es debido a que en la SFC y en el Stateflow está implementado el mismo proceso, por lo tanto, la computación que debe realizar es similar en ambos casos. Estas ejecuciones se pueden observar en este video.



```
CASE ssMethodType OF
    0:
        ssMethodType:=1;
        (* InitializeConditions for Chart: '<Root>/e2' *)
        is_active_c3_e2 := 0;
        is_c3_e2 := 0;
    1:
        (* Gateway: e2 *)
        (* During: e2 *)
        IF is_active_c3_e2 = 0 THEN
            (* Entry: e2 *)
            is_active_c3_e2 := 1;
            (* Entry Internal: e2 *)
            (* Transition: '<S1>:2' *)
            is_c3_e2 := 15;
        temporizador (IN := FALSE,PT := t#0s,Q => done );
            (* Outport: '<Root>/Subir_bajar' *)
            (* Entry 'Girar_izda_y_subir': '<S1>:1' *)
            Subir_bajar := FALSE;
            (* Outport: '<Root>/Gira_dcha' *)
            Gira_dcha := FALSE;
            (* Outport: '<Root>/Gira_izqda' *)
            Gira_izqda := TRUE;
        ELSE
            CASE is_c3_e2 OF
                1:
                    (* During 'Abre_pinza_y_rearmel': '<S1>:172' *)
                    temporizador (IN := TRUE,PT := t#0.5s,Q => done );
                    IF done THEN
                        (* Transition: '<S1>:174' *)
                        is_c3_e2 := 32;
                        (* Outport: '<Root>/Subir_bajar' *)
                        (* Entry 'Subir_brazo_final': '<S1>:175' *)
                        Subir_bajar := FALSE;
                        (* Outport: '<Root>/Muelle_puesto' *)
                        Muelle_puesto := FALSE;
                        (* Outport: '<Root>/Embolo_grande_puesto' *)
                        Embolo_grande_puesto := FALSE;
                        (* Outport: '<Root>/Embolo_peq_puesto' *)
                        Embolo_peq_puesto := FALSE;
                        (* Outport: '<Root>/Est2_fin' *)
                        Est2_fin := FALSE;
                        (* Outport: '<Root>/Medir_fin_est2' *)
                        Medir_fin_est2 := TRUE;
                    END_IF;
                2:
                    (* During 'Abre_pinza_y_rearmel_y_memo2': '<S1>:146' *)
                    temporizador (IN := TRUE,PT := t#0.5s,Q => done );
                    IF done THEN
                        (* Transition: '<S1>:140' *)
                        is_c3_e2 := 35;
                        (* Outport: '<Root>/Subir_bajar' *)
                        (* Entry 'sube_brazo': '<S1>:131' *)
                        Subir_bajar := FALSE;
                        (* Outport: '<Root>/Poniendo_embolo_peq' *)
                        Poniendo_embolo_peq := FALSE;
                        (* Outport: '<Root>/Poniendo_embolo_grande' *)
                        Poniendo_embolo_grande := FALSE;
                        (* Outport: '<Root>/Est2_fin' *)
                        Est2_fin := FALSE;
                    END_IF;
```

Figura 105.Comparacion SFC y Código generado en Schneider

8.1.3 Tiempos ejecución de Simulink

También se han calculado el tiempo de ejecución del control desde Simulink. Para calcular los ciclos por seg en Simulink se realizará del mismo modo que en los anteriores apartados. Se medirán para el caso de que se esté o no utilizando el servidor OPC pero en ambos casos en tiempo real.

- Con OPC

  300/53=5.66 ciclos/seg

  Esta limitación viene dada por el servidor OPC, ya que este servidor necesita tiempo para enviar y recibir las variables que comunica con las estaciones-

- Sin OPC

  20000/20.13=993.54 ciclos/seg

  Al eliminar el OPC para ver qué capacidad de computación tenia Simulink se observó que era capaz de realizar unos 100000 ciclos/seg como eran demasiados se configuro Simulink para que realizara 1000 ciclos/seg. La configuración se puede ver en la siguiente imagen:

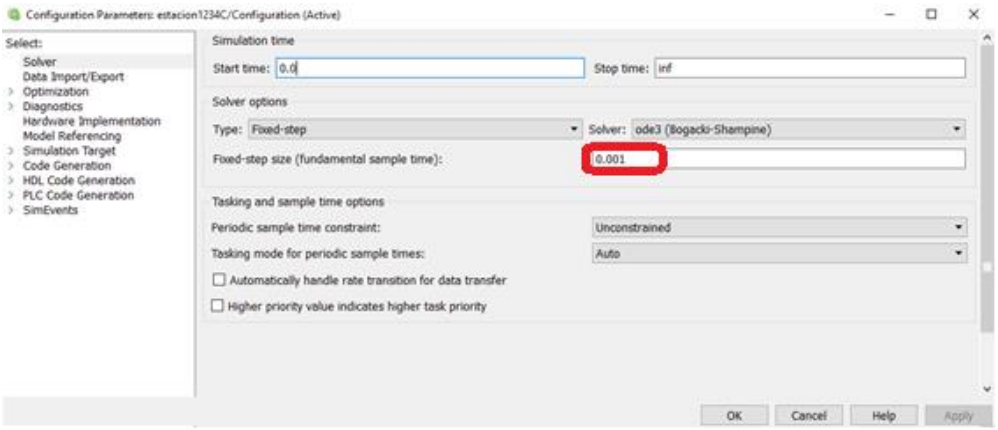


Figura 106.Configuración Simulink sin OPC

Como se puede ver Simulink tiene un alto poder computacional, ya que, ante la misma implementación Simulink es capaz de casi un millar de ciclos por segundo

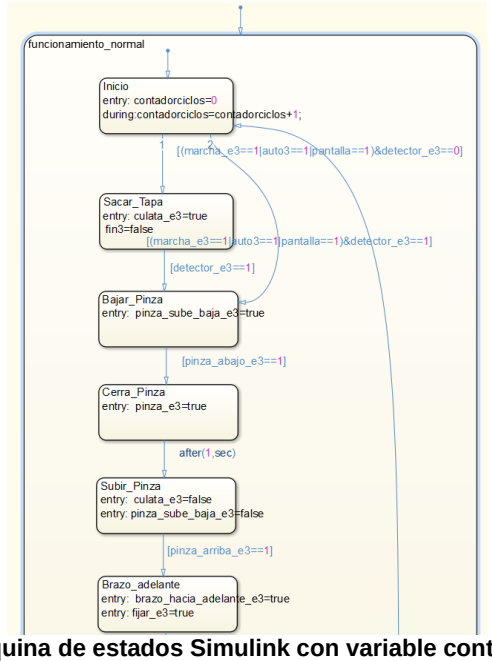


Figura 107.Máquina de estados Simulink con variable contadora de ciclos

8.2 Análisis del código generado

Para analizar el código generado por la herramienta PLC Coder se han implementado en Simulink con la herramienta Stateflow varias máquinas de estados con diferentes estructuras. Pero cabe decir que el código tiene la misma estructura. En primer lugar, se realiza una inicialización de variables y posteriormente se pasa a una función case donde cada caso es un estado programado de la máquina de estados.

El primer lugar se implementó una SFC continua con varios estados. Cada estado solo tiene una posibilidad para avanzar a otro si se cumple cierta condición. La máquina de estados será la siguiente:

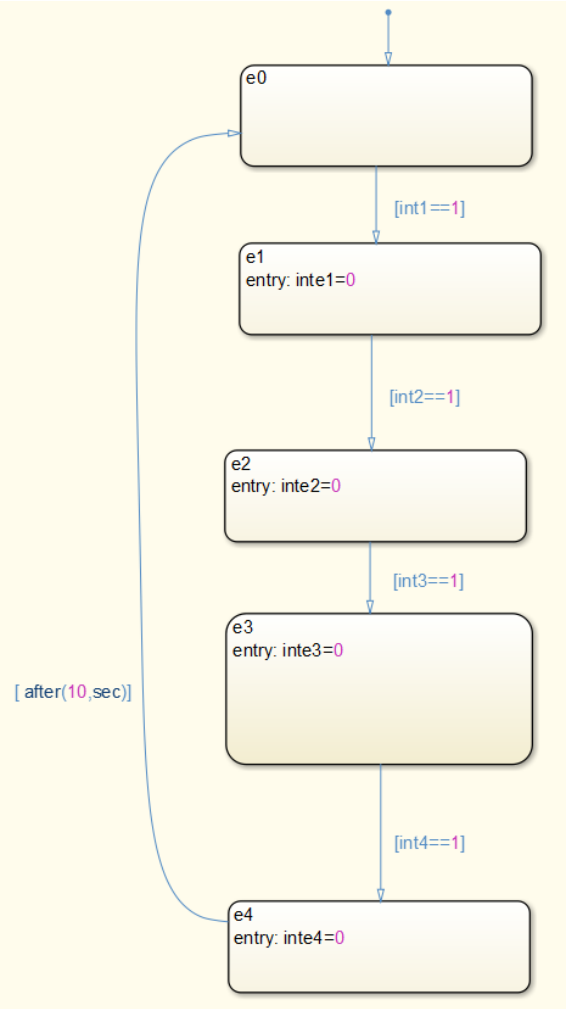


Figura 108.Ejemplo SFC

```
CASE ssMethodType OF
    SINT#0:
        (* InitializeConditions for Chart: '<Root>/SFC ' *)
        temporalCounter_i1(timerAction := INT#1, maxTime := DINT#0);
        is_active_c3_SFC0 := USINT#0;
        is_c3_SFC0 := USINT#0;
    SINT#1:
        (* Gateway: SFC *)
        (* During: SFC *)
        IF USINT_TO_UDINT(is_active_c3_SFC0) = UDINT#0 THEN
            (* Entry: SFC *)
            is_active_c3_SFC0 := USINT#1;
            (* Entry Internal: SFC *)
            (* Transition: '<S1>:219' *)
            is_c3_SFC0 := USINT#1;
        ELSE
            CASE is_c3_SFC0 OF
                USINT#1:
                    (* During 'e0': '<S1>:228' *)
                    IF int1 = LREAL#1.0 THEN
                        (* Transition: '<S1>:227' *)
                        is_c3_SFC0 := USINT#2;
                        (* Outport: '<Root>/inte1' *)
                        (* Entry 'e1': '<S1>:221' *)
                        inte1 := LREAL#0.0;
                    END_IF;
                USINT#2:
                    (* During 'e1': '<S1>:221' *)
                    IF int2 = LREAL#1.0 THEN
                        (* Transition: '<S1>:229' *)
                        is_c3_SFC0 := USINT#3;
                        (* Outport: '<Root>/inte2' *)
                        (* Entry 'e2': '<S1>:225' *)
                        inte2 := LREAL#0.0;
                    END_IF;
                USINT#3:
                    (* During 'e2': '<S1>:225' *)
                    IF int3 = LREAL#1.0 THEN
                        (* Transition: '<S1>:220' *)
                        is_c3_SFC0 := USINT#4;
                        (* Outport: '<Root>/inte3' *)
                        (* Entry 'e3': '<S1>:222' *)
                        inte3 := LREAL#0.0;
                    END_IF;
                USINT#4:
                    (* During 'e3': '<S1>:222' *)
                    IF int4 = LREAL#1.0 THEN
                        (* Transition: '<S1>:224' *)
                        is_c3_SFC0 := USINT#5;
                        temporalCounter_i1(timerAction := INT#1, maxTime := DINT#0);
                        (* Outport: '<Root>/inte4' *)
                        (* Entry 'e4': '<S1>:223' *)
                        inte4 := LREAL#0.0;
                    END_IF;
                ELSE
                    (* During 'e4': '<S1>:223' *)
                    temporalCounter_i1(timerAction := INT#2, maxTime := DINT#10000);
                    IF temporalCounter_i1.done THEN
                        (* Transition: '<S1>:226' *)
                        is_c3_SFC0 := USINT#1;
                    END_IF;
            END_CASE;
        END_IF;
        (* End of Chart: '<Root>/SFC ' *)
END_CASE;
```

Figura 109.Codigo Generado para ejemplo SFC

En primer lugar, hay una variable llamada ssMethodType que se utilizara para inicializar las variables para recorrer los diferentes estados y los temporizadores. Y posteriormente se pasará a la programación como tal

Posteriormente, como se puede ver en el código, la estructura generada es una estructura tipo CASE donde cada valor de la variable is_c3_SFC0 indica un estado de la máquina de estados. Dentro de cada case se puede ver una función if la cual representa la transición de salida del estado en el que te encuentres. Dentro de cada case se actualiza los estados, cuando eso ocurre se produce el disparo de la transición. Si se cumple la condición de la transición, se modifica el valor de la variable is_c3_SFC0 para continuar con el siguiente estado y se modifican las variables de salida que se encuentran en el siguiente estado.

A continuación, se muestra otro ejemplo SFC, pero en este caso se dispone de una SFC donde ciertos estados tienen la posibilidad de avanzar por diferentes ramas en función de las condiciones que se cumplan.

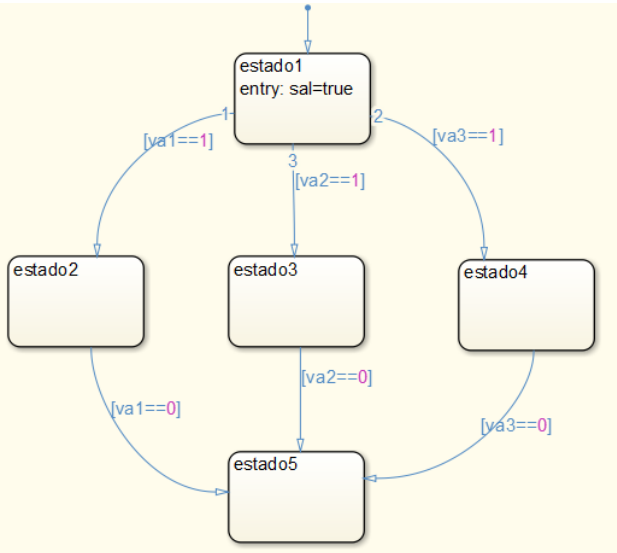


Figura 110.Ejemplo 1 SFC con salida a varios estados y entrada desde variaos estados

```
CASE ssMethodType OF
    SINT#0:
        (* InitializeConditions for Chart: '<Root>/ejemplo_1 ' *)
        is_active_c3_ejemplo_1 := USINT#0;
        is_c3_ejemplo_1 := USINT#0;
    SINT#1:
        (* Gateway: ejemplo_1 *)
        (* During: ejemplo_1 *)
        IF USINT_TO_UDINT(is_active_c3_ejemplo_1) = UDINT#0 THEN
            (* Entry: ejemplo_1 *)
            is_active_c3_ejemplo_1 := USINT#1;
            (* Entry Internal: ejemplo_1 *)
            (* Transition: '<S1>:189' *)
            is_c3_ejemplo_1 := USINT#1;
            (* Outport: '<Root>/int0' *)
            (* Entry 'el': '<S1>:188' *)
            int0 := BOOL#TRUE;
        ELSE
            CASE is_c3_ejemplo_1 OF
                USINT#1:
                    (* During 'el': '<S1>:188' *)
                    IF int1 = LREAL#1.0 THEN
                        (* Transition: '<S1>:193' *)
                        is_c3_ejemplo_1 := USINT#2;
                    ELSIF int2 = LREAL#1.0 THEN
                        (* Transition: '<S1>:194' *)
                        is_c3_ejemplo_1 := USINT#3;
                    ELSIF int3 = LREAL#1.0 THEN
                        (* Transition: '<S1>:195' *)
                        is_c3_ejemplo_1 := USINT#4;
                    END_IF;
                USINT#2:
                    (* During 'estado2': '<S1>:663' *)
                    IF va1 = LREAL#0.0 THEN
                        (* Transition: '<S1>:671' *)
                        is_c12_pedidos1 := USINT#5;
                    END_IF;
                (* During 'estado3': '<S1>:664' *)
                    IF va2 = LREAL#0.0 THEN
                        (* Transition: '<S1>:670' *)
                        is_c12_pedidos1 := USINT#5;
                    END_IF;
                USINT#4:
                    (* During 'estado4': '<S1>:665' *)
                    IF va3 = LREAL#0.0 THEN
                        (* Transition: '<S1>:674' *)
                        is_c12_pedidos1 := USINT#5;
                    END_IF;
            END_CASE;
        END_IF;
        (* End of Chart: '<Root>/ejemplo_1 ' *)
END_CASE;
```

Figura 111.Código generado para ejemplo 1 SFC

Al igual que en el anterior ejemplo, en primer lugar, se utiliza una variable llamada ssMethodType para inicializar las variables para recorrer los diferentes estados y los temporizadores. Y posteriormente se pasará a la programación como tal

La programación para esta máquina de estados será semejante a la anterior, pero con la diferencia de que en el estado que tenga varios estados para continuar como seria e1, se comprueban la condición de cada una de las transiciones, cuando una de ellas sea se modificara la variable is_c3_ejemplo_1 para cambiar de estado

Por último, hay un último ejemplo donde se realizan 3 estados, e21, e31 y e41, en paralelo, cuando se cumpla cierta condición se termina el proceso concurrente.

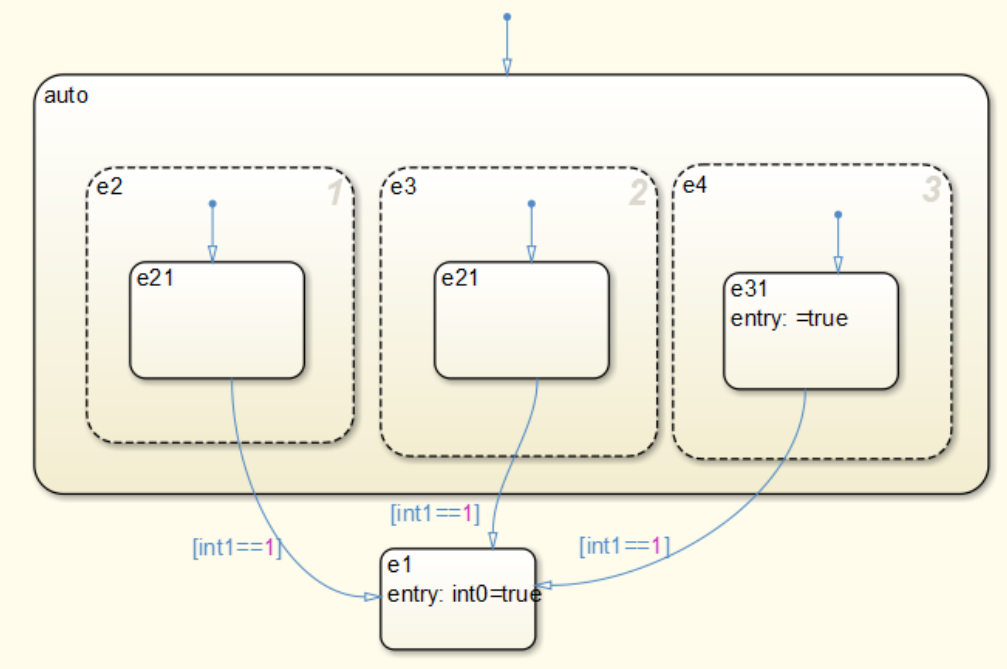


Figura 112.Ejemplo 2 con SFC concurrente

El principio del código es igual que en los ejemplos anteriores, don se inicializan las variables que se van a utilizar en el código.

Para esta máquina de estados ya no aparece la estructura del case, si no que aparecen funciones if. Por lo tanto cada vez que se cumpla una transición (condición del if) se realizaran las acciones del estado siguiente a la transición y se vuelve a la posición anterior para en caso de que se cumpla la transición de los otros estados se puedan realizar.

En caso de que la maquina tendría una SFC normal dentro de cada estado concurrente, ese código estaría ubicad con una función case dentro de los if antes mencionados.

```
CASE ssMethodType OF
    SINT#0:
        (* InitializeConditions for Chart: '<Root>/ejemplo_2 ' *)
        is_active_e2 := USINT#0;
        is_e2 := USINT#0;
        is_active_e3 := USINT#0;
        is_e3 := USINT#0;
        is_active_e4 := USINT#0;
        is_e4 := USINT#0;
        is_active_c3_ejemplo_2 := USINT#0;
        is_c3_ejemplo_2 := USINT#0;
    SINT#1:
        (* Gateway: ejemplo_2 *)
        (* During: ejemplo_2 *)
        IF USINT_TO_UDINT(is_active_c3_ejemplo_2) = UDINT#0 THEN
            (* Entry: ejemplo_2 *)
            is_active_c3_ejemplo_2 := USINT#1;
            (* Entry Internal: ejemplo_2 *)
            (* Transition: '<S1>:205' *)
            is_c3_ejemplo_2 := USINT#1;
            (* Entry Internal 'auto': '<S1>:204' *)
            (* Transition: '<S1>:207' *)
            is_active_e2 := USINT#1;
            (* Entry Internal 'e2': '<S1>:190' *)
            (* Transition: '<S1>:212' *)
            is_e2 := USINT#1;
            is_active_e3 := USINT#1;
            (* Entry Internal 'e3': '<S1>:215' *)
            (* Transition: '<S1>:213' *)
            is_e3 := USINT#1;
            is_active_e4 := USINT#1;
            (* Entry Internal 'e4': '<S1>:218' *)
            (* Transition: '<S1>:216' *)
            is_e4 := USINT#1;
        ELSIF USINT_TO_DINT(is_c3_ejemplo_2) = USINT_TO_DINT(USINT#1) THEN
            (* During 'auto': '<S1>:204' *)
            (* During 'e2': '<S1>:190' *)
            (* During 'e21': '<S1>:210' *)
            IF int1 = LREAL#1.0 THEN
                (* Transition: '<S1>:193' *)
                (* Exit Internal 'auto': '<S1>:204' *)
                (* Exit Internal 'e4': '<S1>:218' *)
                is_e4 := USINT#0;
                is_active_e4 := USINT#0;
                (* Exit Internal 'e3': '<S1>:215' *)
                is_e3 := USINT#0;
                is_active_e3 := USINT#0;
                (* Exit Internal 'e2': '<S1>:190' *)
                is_e2 := USINT#0;
                is_active_e2 := USINT#0;
                is_c3_ejemplo_2 := USINT#2;
                (* Outport: '<Root>/int0' *)
                (* Entry 'e1': '<S1>:188' *)
                int0 := BOOL#TRUE;
            END_IF;
            IF USINT_TO_DINT(is_c3_ejemplo_2) = USINT_TO_DINT(USINT#1) THEN
                (* During 'e3': '<S1>:215' *)
                (* During 'e31': '<S1>:214' *)
                IF int2 = LREAL#1.0 THEN
                    (* Transition: '<S1>:194' *)
                    (* Exit Internal 'auto': '<S1>:204' *)
                    (* Exit Internal 'e4': '<S1>:218' *)
                    is_e4 := USINT#0;
                    is_active_e4 := USINT#0;
                    (* Exit Internal 'e3': '<S1>:215' *)
                    is_e3 := USINT#0;
                    is_active_e3 := USINT#0;
                    (* Exit Internal 'e2': '<S1>:190' *)
                    is_e2 := USINT#0;
                    is_active_e2 := USINT#0;
                    is_c3_ejemplo_2 := USINT#2;
                    (* Outport: '<Root>/int0' *)
                    (* Entry 'e1': '<S1>:188' *)
                    int0 := BOOL#TRUE;
                END_IF;
                IF  NOT ((USINT_TO_DINT(is_c3_ejemplo_2) <> USINT_TO_DINT(USINT#1)) OR ( NOT (int3 = LREAL#1.0))) THEN
                    (* During 'e4': '<S1>:218' *)
                    (* During 'e41': '<S1>:217' *)
                    (* Transition: '<S1>:195' *)
                    (* Exit Internal 'auto': '<S1>:204' *)
                    (* Exit Internal 'e4': '<S1>:218' *)
                    is_e4 := USINT#0;
                    is_active_e4 := USINT#0;
                    (* Exit Internal 'e3': '<S1>:215' *)
                    is_e3 := USINT#0;
                    is_active_e3 := USINT#0;
                    (* Exit Internal 'e2': '<S1>:190' *)
                    is_e2 := USINT#0;
                    is_active_e2 := USINT#0;
                    is_c3_ejemplo_2 := USINT#2;
                    (* Outport: '<Root>/int0' *)
                    (* Entry 'e1': '<S1>:188' *)
                    int0 := BOOL#TRUE;
                END_IF;
            END_IF;
        END_IF;
        (* End of Chart: '<Root>/ejemplo_2 ' *)
END_CASE;
```

Figura 113.Código generado ejemplo 2 SFC concurrente

9 Conclusiones y líneas futuras

En este capítulo se comentan las conclusiones finales del proyecto, así como las posibles líneas futuras de trabajo sobre la maqueta.

9.1 Conclusiones

En este proyecto se ha conseguido controlar la célula de Fabricación desde tres plataformas diferentes. Se ha realizado la programación completa de la célula de fabricación en Stateflow y se han puesto en marcha tres plataformas de control distintas:

- La primera ejecutando directamente en Simulink y comunicando con el hardware de control mediante OPC.
- En la segunda el control se ejecuta en un PLC Rockwell que de forma centralizada controla toda la zona de elaboración de pedidos, excepto la estación 2, controlada por un PLC Schneider. El programa del PLC Rockwell se ha generado exportando los Stateflow mediante la librería PLC Coder.
- En la tercer el control se ejecuta en cuatro PLCs Schneider de forma descentralizada. Los programas de los PLCs Schneider también se han generado exportando los Stateflow mediante la librería PLC Coder.

En una terminal Magelis se han creado pantallas para el control y la supervisión de todas las estaciones y de la cinta de transporte. Y se ha añadido la lectura y escritura de las memorias de los palets que pasan por cada una de las estaciones. Esta lectura y escritura ha sido realizada mediante RFID. También se han realizado pantallas de mando y de animación en 3D en Simulink.

Resaltar también que se ha conseguido realizar el control mediante Stateflow de la Estación 2 de la célula de fabricación, la cual no solo tiene accionadores todo/nada, sino que hay que realizar el control de un accionamiento con motor paso a paso.

Se ha realizado una evaluación de los tiempos de ejecución de las diferentes plataformas. Se ha realizado también una evaluación del código generado por el PLC Coder para las estructuras típicas que aparecen en el modelado de los sistemas de eventos discretos.

Como conclusión final resaltar la versatilidad que ofrece Simulink para poder implementar el control de Sistemas de Eventos Discretos mediante Stateflow, permitiendo mediante PLC Coder la puesta en marcha, en este trabajo fin de grado, de tres plataformas de control diferentes.

9.2 Líneas futuras

En el punto de líneas futuras, hay que mencionar la posibilidad de ampliación de los controles implementados en StateFlow de Simulink, incorporando nuevos modos de funcionamiento a los controles, así como, la ampliación de la supervisión de más estaciones mediante modelos virtuales.

Se podría realizar el control de las estaciones mediante micro controlador Arduino, realizando toda la programación en Simulink mediante Stateflow y generando código para el micro utilizando la herramienta PLC Coder, al igual que en este proyecto.

Por otro lado, en las nuevas versiones de Matlab/Simulink se ha incorporado la opción de exportar a lenguaje de contactos y se podría evaluar el código generado en este lenguaje.

Por último, quedaría incorporar el resto la célula, añadiendo la zona de almacén intermedio y la zona de expedición de pedidos.

Bibliografía

[1] Trabajo final de grado “AUTOMATIZACIÓN Y SUPERVISIÓN DE LA CÉLULA DE FABRICACIÓN”. Grado ingeniería electrónica y automática. 2015. Aut. Santiago Gracia Mateo

[2] Proyecto de Fin de Carrera “Programación de autómatas en Unity: Programación de autómatas en Unity”. Ingeniería industrial curso 2007. Aut. Sergio Alegre Bernad

[3] Proyecto de Fin de Carrera “AUTOMATIZACIÓN DE LA CÉLULA DE FABRICACIÓN”. Ingeniería industrial curso 2011/12. Aut. Emilio Gimenez Gaudes

[4] Explicación de la lógica temporal de la herramienta Stateflow de Simulink https://es.mathworks.com/help/stateflow/ug/using-temporal-logic-in-state-actions-and-transitions.html?requestedDomain=www.mathworks.com

[5] Explicación estados de la herramienta Stateflow de Simulink https://es.mathworks.com/help/stateflow/ug/states.html

[6] Explicación estados de la herramienta Stateflow de Simulink https://es.mathworks.com/help/stateflow/ug/states.htmll

[7] Generación de código mediante PLC Coder https://es.mathworks.com/help/plccoder/plc-coder-general.html

[8] Página principal de Simulink 3D Animation https://es.mathworks.com/help/sl3d/index.html

[9] Guía básica construcción y conexión de un mundo virtual en Simulink https://es.mathworks.com/help/sl3d/example-of-building-a-virtual-world-and-connecting-it-to-a-simulink-model.html

[10] Guía avanzada de programación para 3D World Editor http://docplayer.net/4576980-Simulink-3d-animation-user-s-guide.html

[11] Ejemplo modelo 3D de puente grúa con mando https://es.mathworks.com/help/sl3d/examples/portal-crane-with-control-panel.html

[12] Ejemplo modelo 3D de brazo moviendo una carga https://es.mathworks.com/help/sl3d/examples/manipulator-moving-a-load-with-use-of-global-coordinates.html

[13] Ejemplo creación pantalla GUIDE de Matlab https://es.mathworks.com/help/matlab/creating_guis/about-the-simple-guide-gui-example.html

[14] Creación de Interfaces Gráficas de Usuario (GUI) con Matlab http://webpersonal.uma.es/de/gfdc/docencia/GuiSection.pdf

[15] Prácticas de Informativa Industrial “Célula de Fabricación Flexible”, Aut.: Ramón Piedrafita.

[16] Ramón Piedrafita. (2004) INGENIERIA DE LA AUTOMATIZACION INDUSTRIAL. Zaragoza. España. Editorial RA-M.

Anexo 1 Control desde Simulink

A continuación se explicaran el resto de la programación para el control desde Simulink. Para esta programación se realizará el mismo procedimiento que se realizó para la estación 1, primero se mostrará el esquema general de Simulink y posteriormente las máquinas de estados asociados a los bloques chart que aparecen en ese esquema.

Configuración estaciones

En primer lugar habrá que configurar la dirección del PLC a la cual se va a transferir las implementaciones. Para ello, la conexión se puede realizar a través del Ethernet. Para configurar la comunicación, se accede a la opción "Establecer comunicación" desde el menú desplegable PLC.

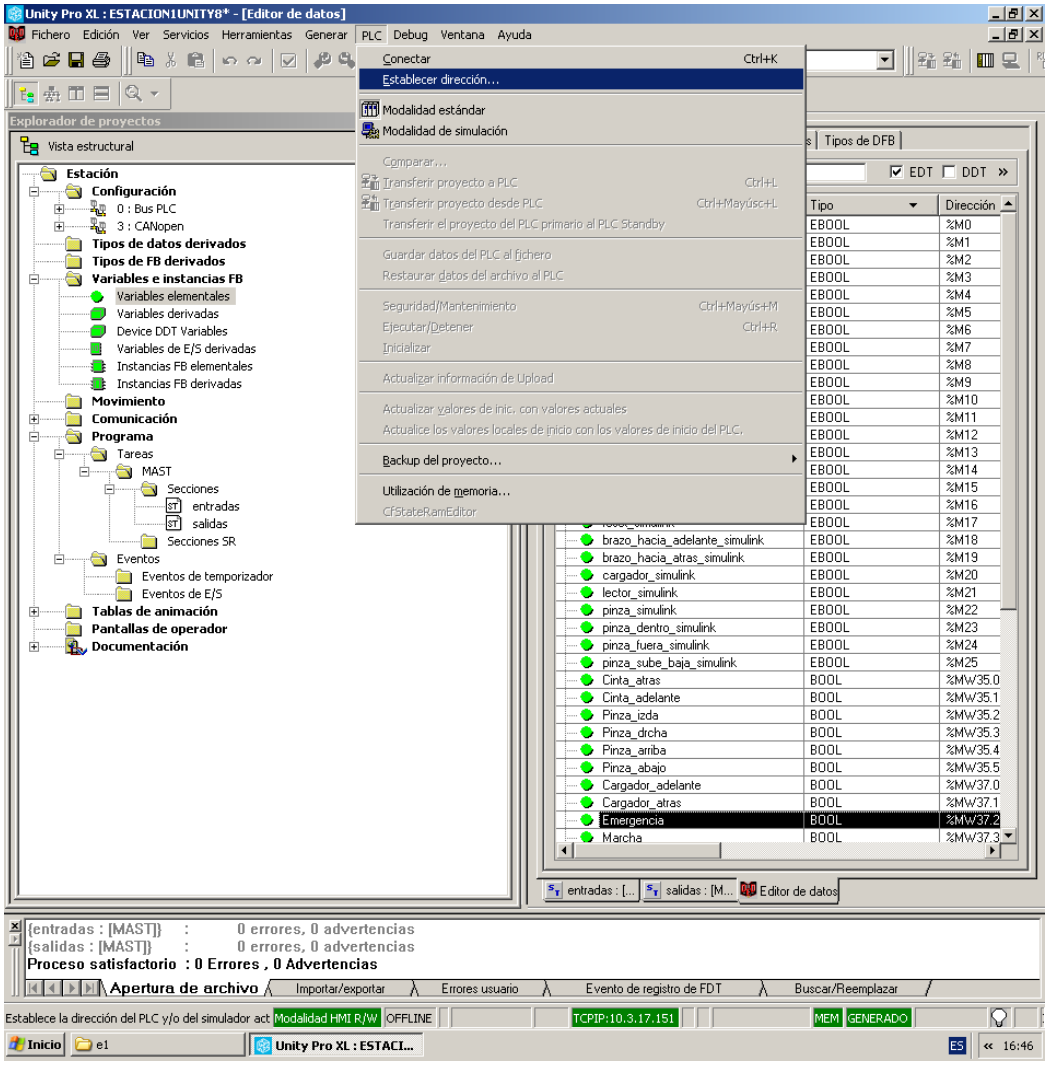


Figura 114.Ventana Unity configuración dirección 1

Aparecerá la siguiente ventana, como se va a utilizar Ethernet, el medio será TCPIP y la dirección dependerá de la estación a la cual va dirigido. Las direcciones de cada una de las estaciones se han indicado con anterioridad.

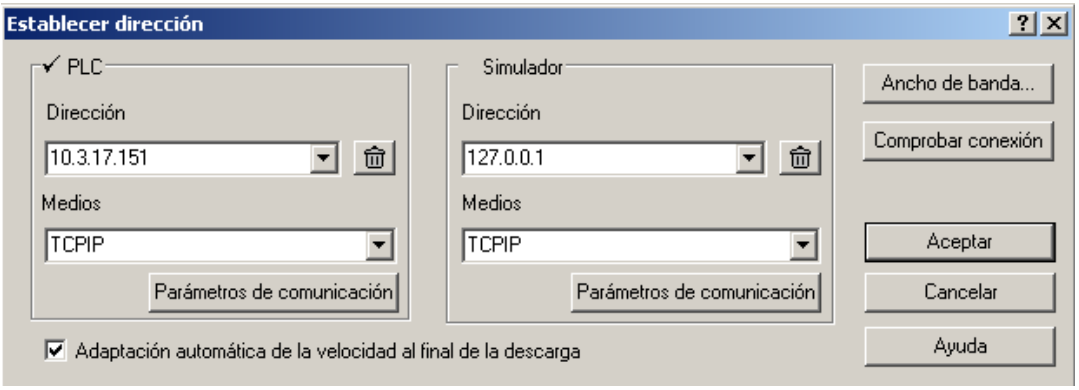


Figura 115.Ventana Unity configuración dirección 1

Nuevas Variables creadas para OPC en Unity y relación con las variables utilizadas por la estación

Además de configurar las estaciones, habrá que crear unas variables las cuales las serán utilizadas para el OPC y poder controlar las estaciones.

Estación 1

Entradas				
Nombre		Tipo	Dirección	
OPC	Unity		OPC	Unity
Capacitivo	Capacitivo_simulink	Booleana	1	%M0
Inductivo	Inductivo_simulink	Booleana	2	%M1
Man_auto	Man_auto_simulink	Booleana	3	%M2
Marcha	Marcha_simulink	Booleana	4	%M3
Optico	Optico_simulink	Booleana	5	%M4
Cargador_adelante	Cargador_adelante_simulink	Booleana	6	%M5
Cargador_atras	Cargador_atras_simulink	Booleana	7	%M6
Ind_int	Ind_int_simulink	Booleana	8	%M7
Lector_atras	Lector_atras_simulink	Booleana	9	%M8
Lector_adelante	Lector_adelante_simulink	Booleana	10	%M9
Óptico_lector	Óptico_lector_simulink	Booleana	11	%M10
Pinza_abajo	Pinza_abajo_simulink	Booleana	12	%M11
Pinza_adelante	Pinza_adelante _simulink	Booleana	13	%M12
Pinza_arriba	Pinza_arriba _simulink	Booleana	14	%M13
Pinza_atras	Pinza_atras _simulink	Booleana	15	%M14
Pinza_dcha	Pinza_dcha _simulink	Booleana	16	%M15
Pinza_izda	Pinza_izda _simulink	Booleana	17	%M16
reset	reset _simulink	Booleana	18	%M17
Salidas				
Nombre		Tipo	Dirección	
OPC	Unity		OPC	Unity
Brazo_hacia_atras	Brazo_hacia_atras_simulink	Booleana	20	%M19
Cargador	Cargador_simulink	Booleana	21	%M20
Lector	Lector_simulink	Booleana	22	%M21
Pinza	Pinza _simulink	Booleana	23	%M22
Pinza_dentro	Pinza_dentro_simulink	Booleana	24	%M23

Pinza_fuera	Pinza_fuera _simulink	Booleana	25	%M24
Pinza_sube_baja	Pinza_sube_baja _simulink	Booleana	26	%M25

Tabla 11.Relación entre variables entradas/salidas del servidor OPC y Unity Estación 1

Estación 2

Cómo es la única estación que no tiene el control en Simulink, no será necesario crear variables auxiliares de Simulink, sino que, se utilizaran directamente las variables utilizadas en el SFC que se explicara a continuación estará implementado. Las nuevas variables creadas serán las siguientes:

Nombre OPC	Tipo	Dirección	
		OPC	Unity
Man_auto_e2	Booleana	16	%M15
Embolo_grande	Booleana	231	%M230
Embolo_peq	Booleana	232	%M231
Fin_est2	Booleana	292	%M291
Marcha_e2	Booleana	15	%M14

Tabla 12.Relación entre variables servidor OPC y Unity Estación 2

Estación 3

Al igual que la estación 1, esta estación también será controlada desde Simulink por medio del OPC, por tanto, habrá que crear las variables en Unity que se utilizan para recibir la información. Además en esta estación, también llegaran las variables para el transporte, ya que, como se ha explicado antes, será la encargada de transmitir los valores a la cinta.

Las nuevas variables creadas serán las siguientes:

Entradas Estación				
Nombre		Tipo	Dirección	
OPC	Unity		OPC	Unity
Marcha	Marcha_simulink	Booleana	1	%M0
Brazo _adelante	Brazo _adelante_simulink	Booleana	2	%M1
Brazo_atras	Brazo_atras_simulink	Booleana	3	%M2
Detector	Detector_simulink	Booleana	4	%M3
Gira_drcha	Gira_drcha_simulink	Booleana	5	%M4
Gira_izda	Gira_izda_simulink	Booleana	6	%M5
Pinza_abajo	Pinza_abajo_simulink	Booleana	7	%M6
Pinza_arriba	Pinza_arriba_simulink	Booleana	8	%M7
Salidas Estación				
Nombre		Tipo	Dirección	
OPC	Unity		OPC	Unity
Brazo_hacia_adelante	Brazo_hacia_adelante_simulink	Booleana	9	%M8
Brazo_hacia_atras	Brazo_hacia_atras _simulink	Booleana	10	%M9
culata	culata _simulink	Booleana	11	%M10
enroscar	enroscar _simulink	Booleana	12	%M11
fijar	fijar	Booleana	13	%M12
pinza	Pinza_simulink	Booleana	14	%M13
Pinza_sube_baja	Pinza_sube_baja_simulink	Booleana	15	%M14

Tabla 13.Relación entre variables entradas/salidas del servidor OPC y Unity Estación 3

Entradas cinta				
Nombre		Tipo	Dirección	
OPC	Unity		OPC	Unity
Ind_intC	Ind_int_cin	Booleana	28	%M27
Palet_e1	Palet_e1	Booleana	29	%M28
Palet_e2	Palet_e2	Booleana	30	%M29
Palet_e3	Palet_e3	Booleana	31	%M30
Palet_e4	Palet_e4	Booleana	32	%M31
Palet_inter	Palet_inter	Booleana	33	%M32
reset	Rearme_cin	Booleana	34	%M33
Marcha	Marcha_cin	Booleana	35	%M34
Salidas cinta				
Nombre		Tipo	Dirección	
OPC	Unity		OPC	Unity
Motor12	Motor12	Booleana	16	%M15
Enclavamiento_e1	Enclavamiento_e1	Booleana	17	%M16
Enclavamiento_e2	Enclavamiento_e2	Booleana	18	%M17
Enclavamiento_e3	Enclavamiento_e3	Booleana	19	%M18
Enclavamiento_e4	Enclavamiento_e4	Booleana	20	%M19
desvio	desvió	Booleana	21	%M20
Tope1	Tope1	Booleana	22	%M21
Tope2	Tope2	Booleana	23	%M22
Tope3	Tope3	Booleana	24	%M23
Tope4	Tope4	Booleana	25	%M24
Topeme	Topeme	Booleana	26	%M25
Motor345	Motor345	Booleana	27	%M26

Tabla 14.Relación entre variables entradas/salidas del servidor OPC y Unity transporte

Estación 4

Entradas				
Nombre		Tipo	Dirección	
OPC	Unity		OPC	Unity
Cilindro_abajo	Cilindro_abajo_simulink	Booleana	1	%M0
Cilindro_arriba	Cilindro_arriba_simulink	Booleana	2	%M1
Gira_dcha	Gira_dcha_simulink	Booleana	3	%M2
Gira_izda	Gira_izda_simulink	Booleana	4	%M3
Ind_int	Ind_int_simulink	Booleana	5	%M4
Man_auto	Man_auto_simulink	Booleana	6	%M5
Marcha	Marcha_simulink	Booleana	7	%M6
Pieza_fuera	Pieza_fuera_simulink	Booleana	8	%M7
Reset	Reset_simulink	Booleana	9	%M8
Sacar_pieza	Sacar_pieza_simulink	Booleana	10	%M9
Vacio_pieza	Vacio_pieza_simulink	Booleana	11	%M10
Vacio_pinza	Vacio_pinza_simulink	Booleana	12	%M11
Verificador_abajo	Verificador_abajo_simulink	Booleana	13	%M12
Verificador_arriba	Verificador_arriba_simulink	Booleana	14	%M13

Salidas				
Nombre		Tipo	Dirección	
OPC	Unity		OPC	Unity
bascular	bascular_simulink	Booleana	15	%M14
Cilindro_baja	Cilindro_baja_simulink	Booleana	16	%M15
Expulsa	Expulsa_simulink	Booleana	17	%M16
Expulsa_pieza	Expulsa_pieza_simulink	Booleana	18	%M17
Gira_derecha	Gira_derecha_simulink	Booleana	19	%M18
Gira_izquierda	Gira_izquierda_simulink	Booleana	20	%M19
inyectar	inyectar_simulink	Booleana	21	%M20
Saca_pieza	Saca_pieza_simulink	Booleana	22	%M21
Vacio_en_pieza	Vacio_en_pieza_simulink	Booleana	23	%M22
Vacio_en_pinza	Vacio_en_pinza_simulink	Booleana	24	%M23
Verificador_baja	Verificador_baja_simulink	Booleana	25	%M24

Tabla 15.Relación entre variables entradas/salidas del servidor OPC y Unity Estación 4

Estación 2

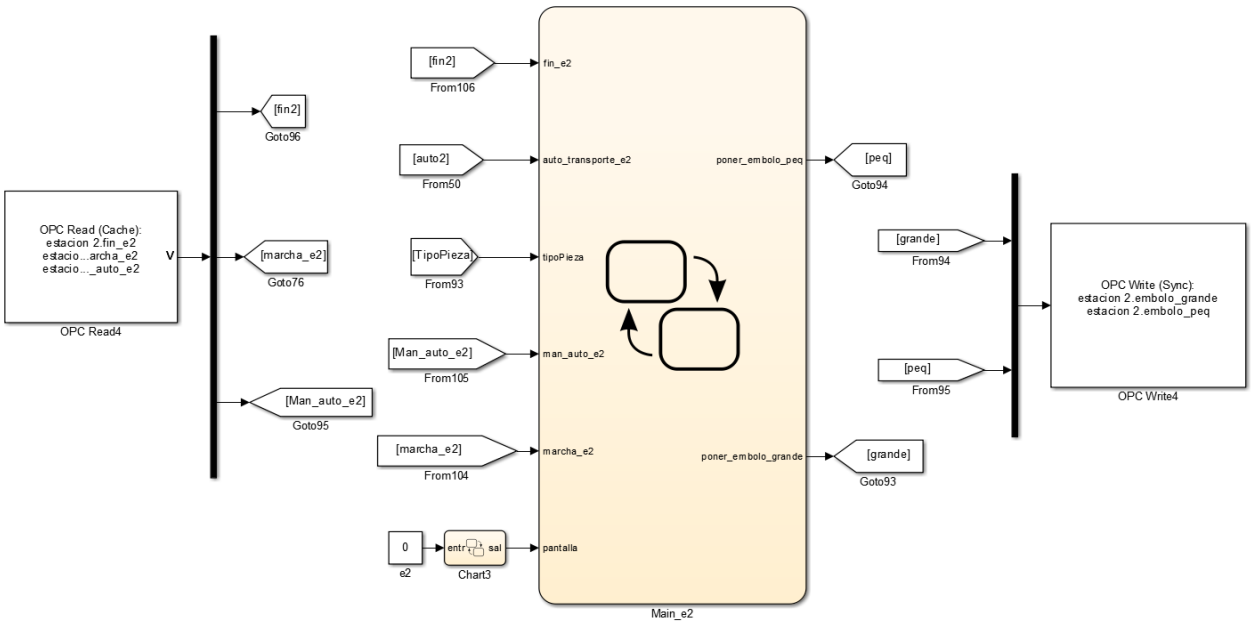


Figura 116.Esquema final para las estación 2

Como esta estación no tiene instalado un módulo de entradas/salidas por Ethernet no puede ser controlada desde Rockwell directamente, por lo tanto, se optó por realizar la implementación en Unity pro para el autómata Schneider que tiene instalado, y mediante la pantalla el servidor OPC, se realizará la comunicación con la estación.

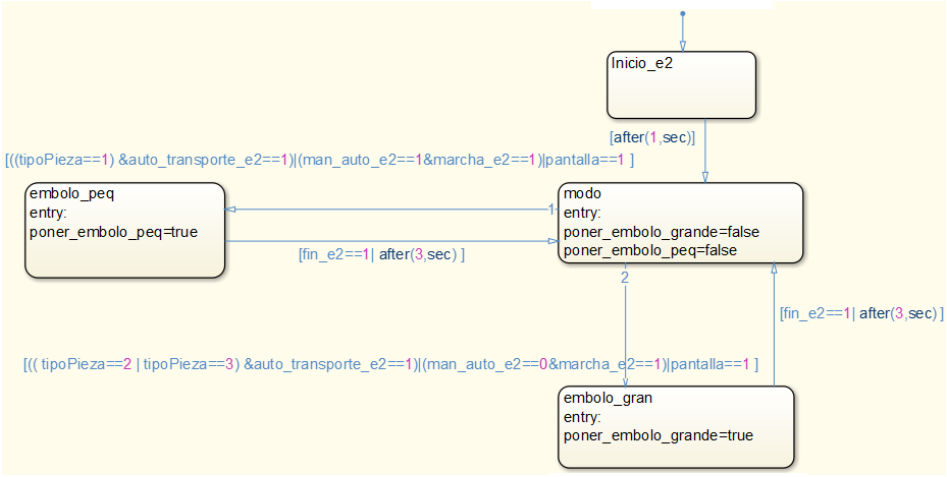


Figura 117.Programa inicial Estación 2

Como se puede observar, el diagrama es simple. Un estado inicial y uno de elección de producción de la estación (modo), el cual resetea 2 variables, poner_embolo_peq o poner_embolo_grande. Después hay otros dos estados que indican el tipo de vástago que tendría que poner la estación 2. Según el valor de estas variables, la estación tendrá que realizar un proceso u otro.

Además de esta pequeña máquina de estados, para la estación 2, se ha utilizado un SFC en Unity pro que controlará la estación, la cual se utilizará para realizar los controles desde de Simulink y desde Rockwell.

A esta SFC se le indicara el proceso a fabricar por medio de una máquina de estados implementada en Simulink la cual le indica el vástago a colocar en función del tipo de pieza que se quiere fabricar por medio del servidor OPC o la magelis en función del control realizado.

La SFC consta de varias partes. En primer lugar se encuentran una serie de estados cuyo objetivo el colocar la estación en la posición inicial para empezar el proceso. Después de esto la SFC se quedara en el estado automático esperando a que el servidor o la magelis le indiquen el proceso a realizar mediante las variables Poner_embolo_grande o Poner_embolo_peq.

Una vez que la estación ya conoce el vástago a colocar, el proceso consistiría en:

1. El brazo se dirige al dispensador de émbolos. El dispensador en función de que la variable está a 1, Poner_embolo_peq o Poner_embolo_grande, sacara un embolo pequeño o grande respectivamente.
2. Una vez que el embolo está en posición, la pinza bajara, lo cojera y volverá a subir.
3. El brazo volverá a la posición encima del palet y dejara el embolo en la pieza.
4. El brazo ira al dispensador de muelle, una vez que haya llegado, el dispensador sacara un muelle.
5. La pinza bajara, lo cojera y volverá a subir. Y el brazo volverá a la posición encima del palet y lo dejara y abra acabado el proceso.

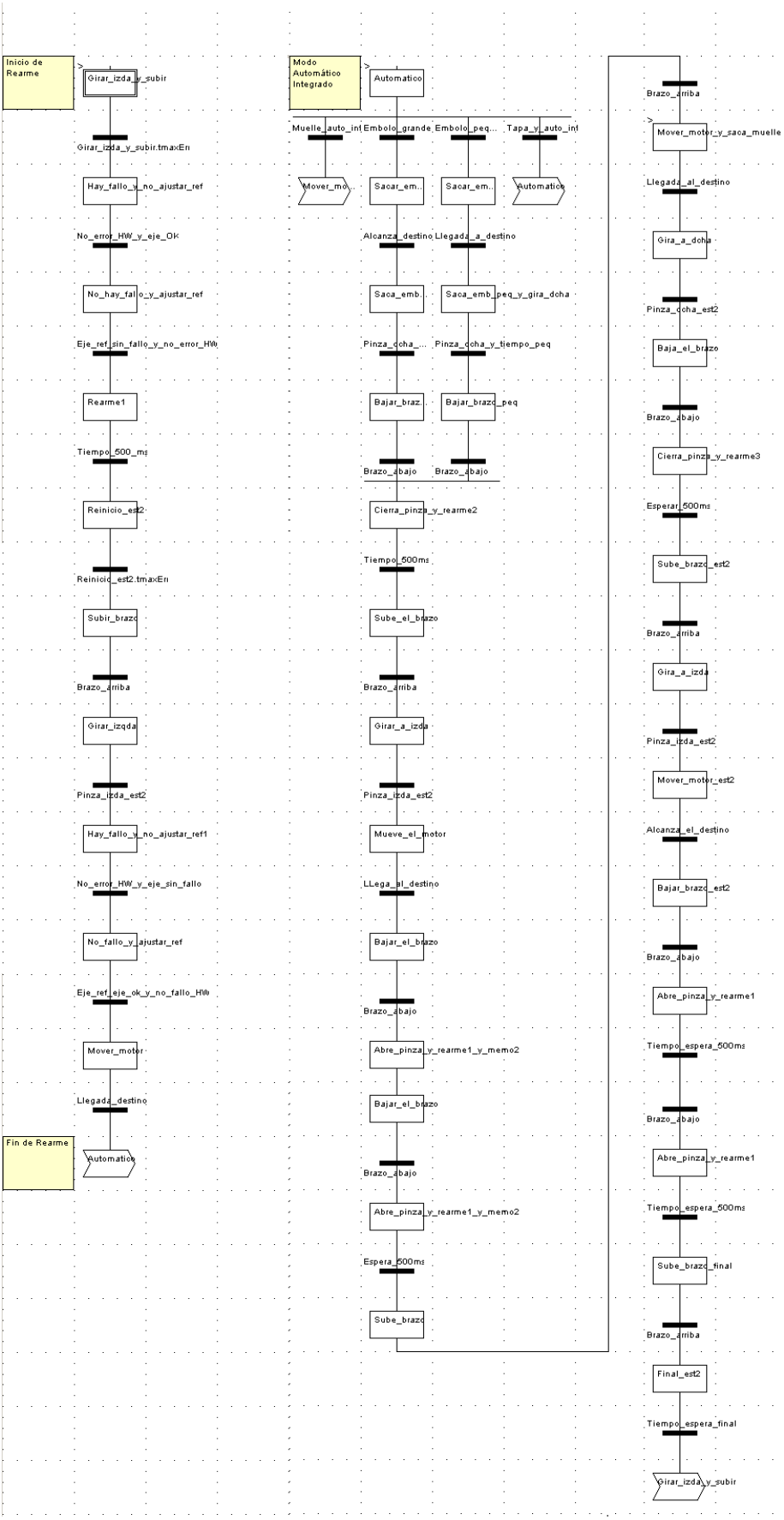


Figura 118.SFC de control para estación 2 para los controles desde Simulink y Rockwell

Estación 3

Además de la implementación explicada en la memora se tiene que realizar una configuración adicional para que la estación se pueda comunicar con la cinta por medio del adaptador de entradas/salidas por Ethernet.

En el fichero destinado para la estación 3 hay que hacer una configuración adicional para que esta se pueda comunicar con el transporte.

En primer lugar en el explorador del proyecto se abrirá la carpeta comunicación y aparecerá una nueva carpeta llamada Redes. Al Pulsar con el botón derecho sobre ella se creará una nueva red de tipo Ethernet.

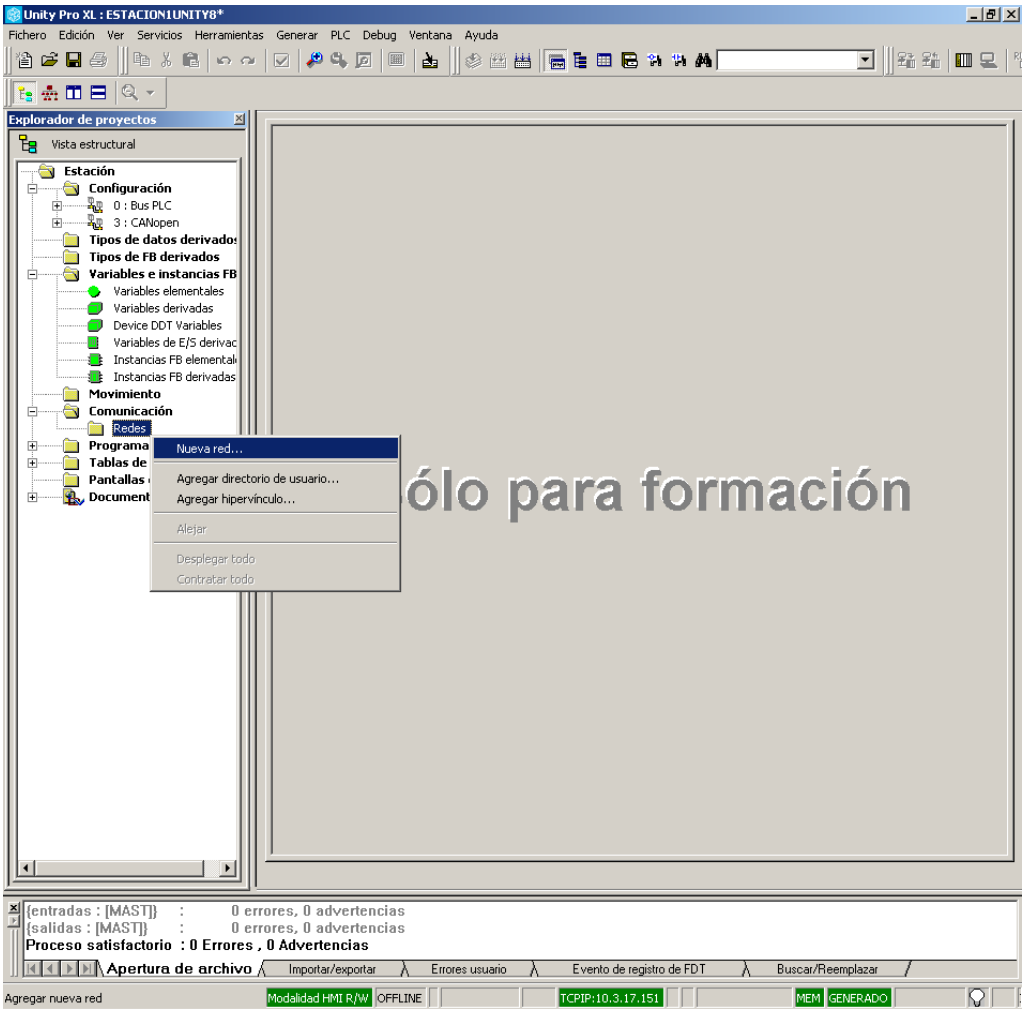


Figura 119.Ventana Unity configuración red

Una vez creada la red de tipo Ethernet con el nombre deseado. Aparecerá una ventana como la siguiente, donde harba que realizar algunos cambios:

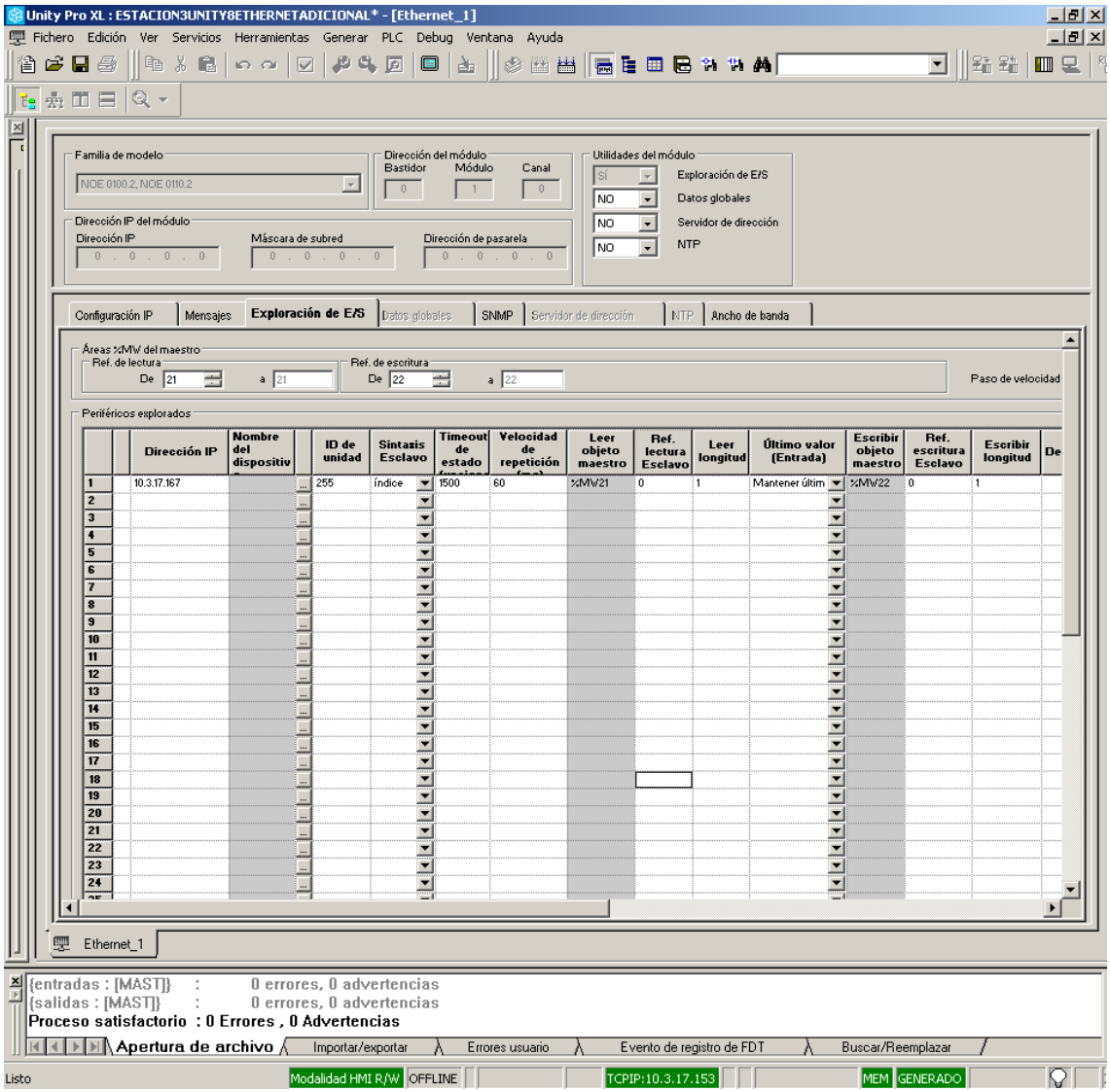


Figura 120.Configuración de la red creada

Primero, en utilidades de modulo, en la pestaña de Exploración de E/S se elegirá SI, inicialmente estará a NO. Así aparecerá en la parte inferior una pestaña con el nombre de Exploración de E/S.

Segundo, al ir a esa pestaña y se añadirá en el primer puesto la dirección del transporte que será 10.3.17.167, el resto de valores aparecen por defecto, de todos ellos solo se va modificar 2 de ellos. En Leer objeto maestro se cambiará la dirección a %MW21y en Escribir objeto maestro se elegirá %MW22. En estas direcciones se habrán creado con anterioridad 2 variables que serán utilizadas para el control del transporte. Cada uno de los bits de estas variables corresponderá con las entradas y salidas de la cinta respectivamente

Estación 4

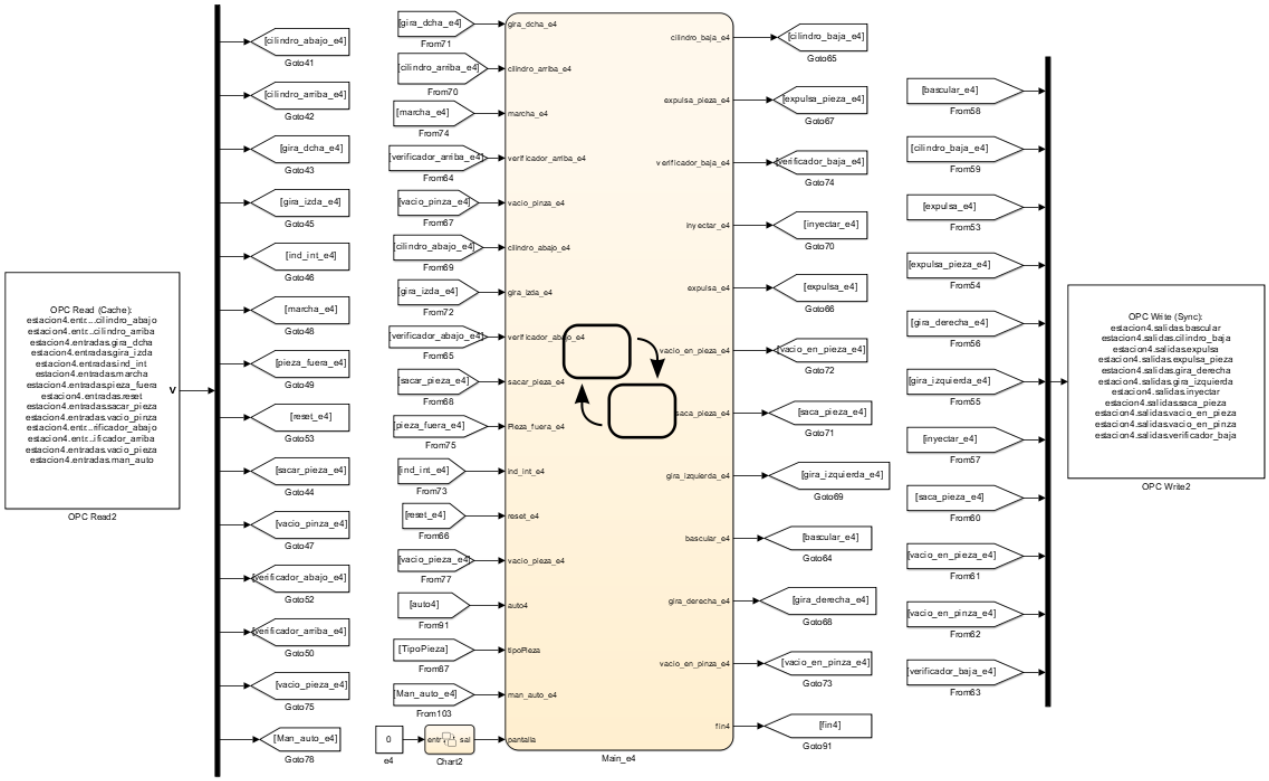


Figura 121.Esquema final para las estación 4

A continuación, se explicara el chart implementado para esta estación.

Una vez pulsado el botón marcha se pasará al estado Pinicial consiste en colocar la estación en la posición óptima para empezar el trabajo. Esa configuración es con el brazo arriba y encima de la cinta sin hacer vacío, para poder coger la pieza con mayor rapidez, con el verificador arriba y con el brazo giratorio fuera del verificador sin hacer vacío. Una vez en esa posición se comenzara el modo automático como tal

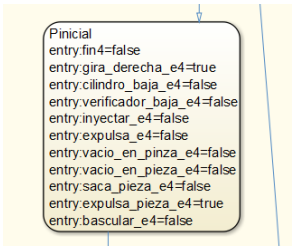


Figura 122.Estado posición inicial

Aunque el modo automático tiene dos producciones distintas dependiendo del tipo de pieza, ambos son muy parecidas excepto en un estado se realiza vacío y ara el otro no, por tanto, se explicaran dos 2 procesos como uno solo aunque en el diagrama estén separados para mayor comprensión

El estado **automático** será el siguiente:

1. Cuando una pieza llega, mediante el brazo (el cual dispone de dos ventosas) se procede a recoger la pieza y depositarla en la zona del verificador. Todo esto es posible tras activar el equipo de vacío y hasta que se haya producido el vacío.

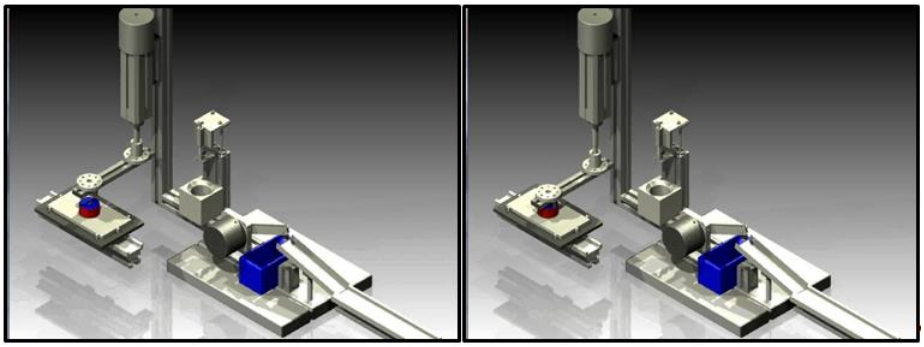


Figura 123.Paso 1 Proceso fabricación estación 4

1.1 Tras haberse recogido la pieza por tal efecto, esta será llevada hasta el verificador.

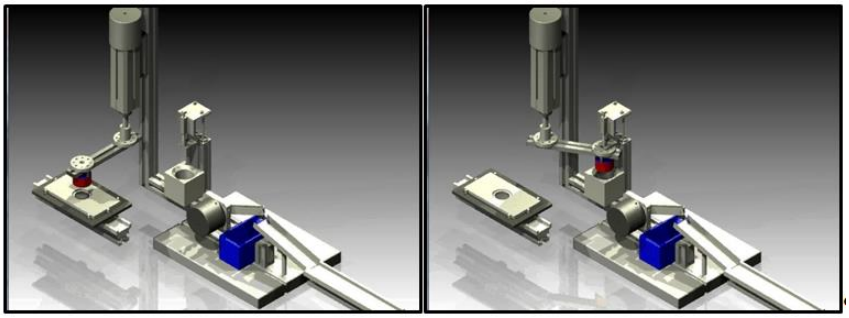


Figura 124.Paso 1.1 Proceso fabricación estación 4

1.2 Depositada ya la pieza en el verificador, en primer lugar se retira el brazo de dicha zona pues de lo contrario, al bajar el verificador, produce un choque entre ambas partes.

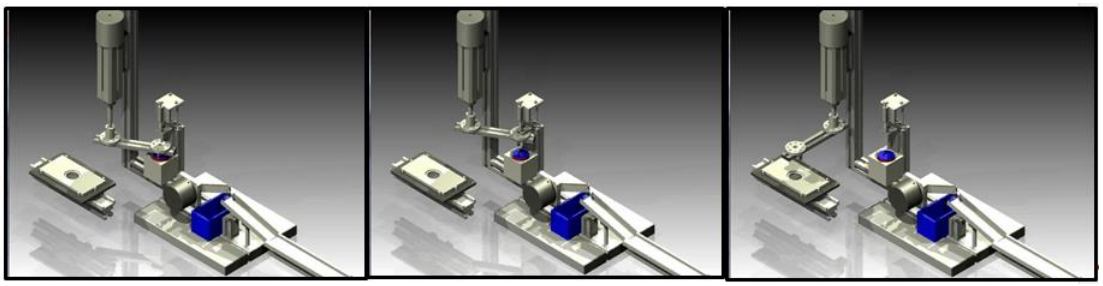


Figura 125.Paso 1.2 Proceso fabricación estación 4

2. Una vez la pieza esté situada, comenzará el proceso de verificación del cilindro el verificador bajará e inyectará aire dentro del cilindro, provocando que el vástago salga hacia afuera. Dicha activación será continuada de manera que pueda medirse el recorrido del émbolo.

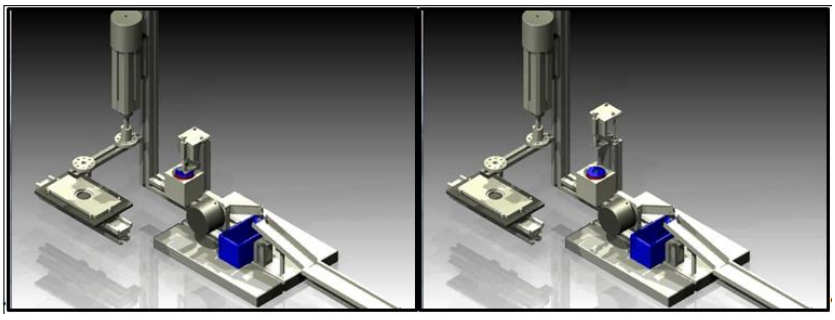


Figura 126.Paso 2 Proceso fabricación estación 4

3. Por último, y en caso en el cual la verificación haya sido positiva, la pieza se enviará hacia la cinta, de modo que el brazo giratorio deberá recoger la pieza mediante absorción por medio de las ventosas que este tiene.

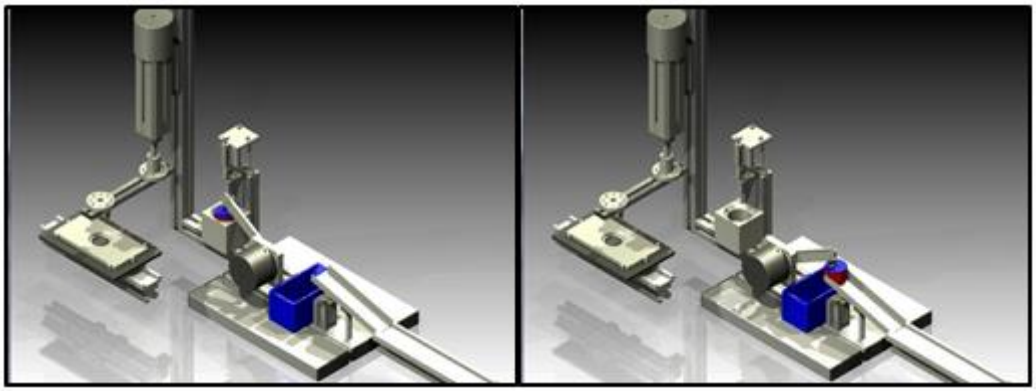


Figura 127.Pao 3.1 Proceso fabricación estación 4

3.1 En el caso de haber sido negativo, la pieza debería haber sido desechada de modo que la inclinación de la rampa tendría que haber cambiado, y por consiguiente, la pieza haber sido enviada a la cubeta de piezas defectuosas.

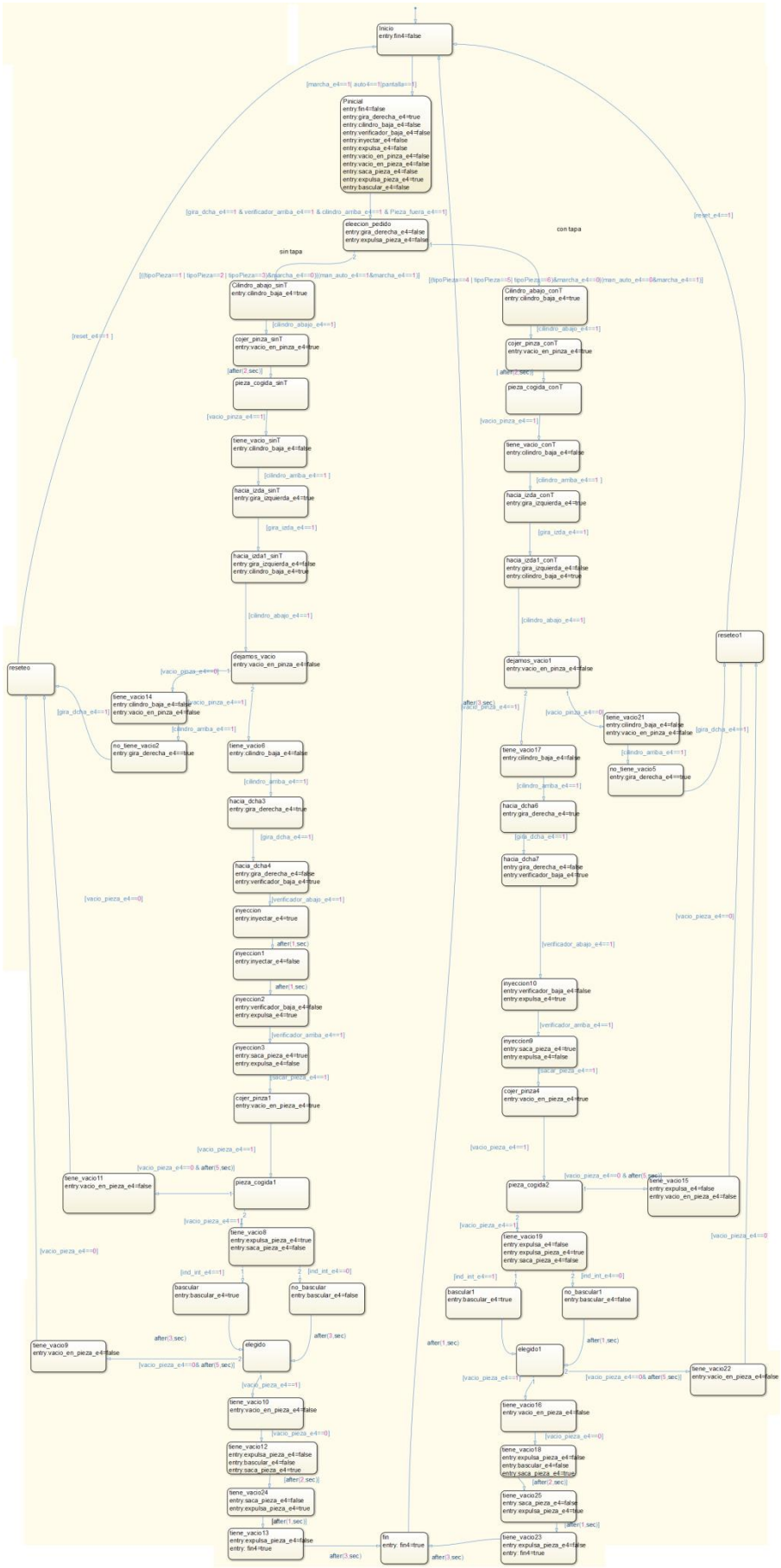


Figura 128.Programa inicial estación 4

Cinta

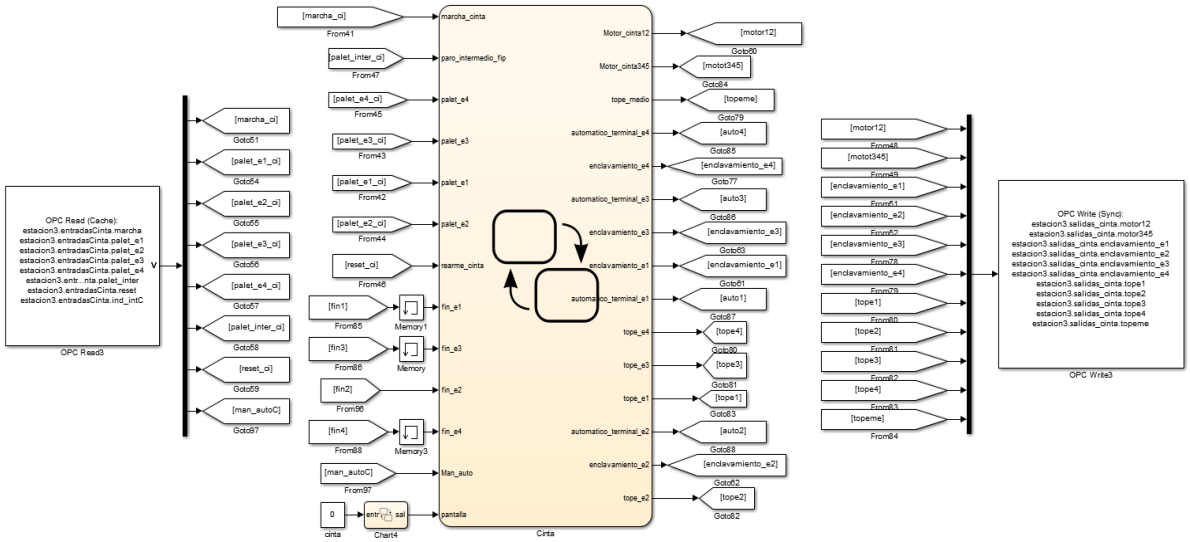


Figura 129.Esquema Simulink para Cinta

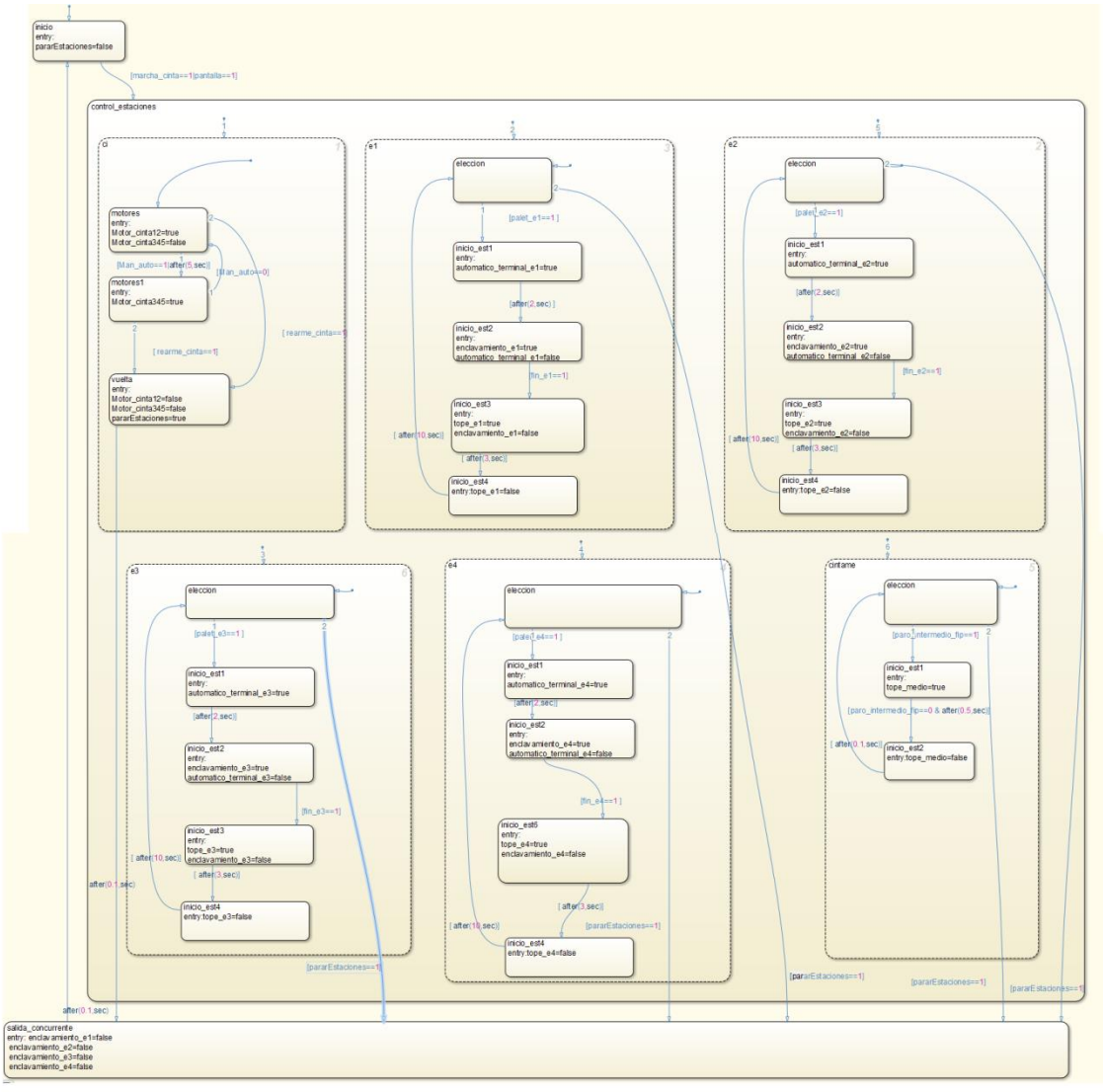


Figura 130.Programación chart cinta

Pedidos

Al añadir la pantalla de Matlab se añadió otra máquina de estados con el cual se controlara la elección de la pieza en función de los botones de la pantalla.

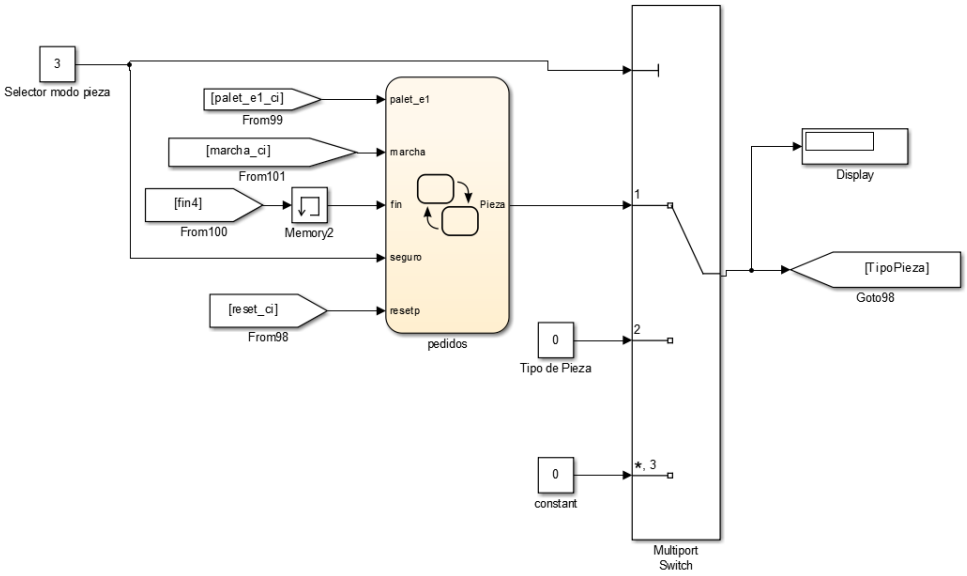


Figura 131.Programa Simulink destinado a elección de Pedidos

Para finalizar, se va a explicar una máquina de estados la cual solo se utilizara en el control desde Simulink y Schneider.

Esta máquina de estados simplemente en un modo elección de pedido automático en el cual se ha implementado todos los tipos posibles de pieza para su fabricación. Para ponerlo en funcionamiento hay que darle al botón marcha de la botonera de la cinta o al botón de marcha de la pantalla de Matlab (GUI). El tipo de pieza cambia cada vez que la estación 4 termina su proceso (variable fin =1).

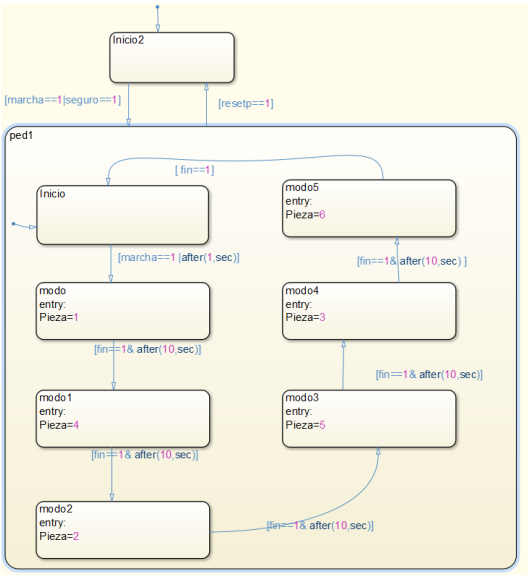


Figura 132.Máquina de estados modo test elección de pedidos

Programación fichero .m pantalla GUI Matlab

Los botones de control de simulación se utilizaran para arrancar o parar la simulación de Simulink. El código asociado a estos botones es el siguiente:

```
% --- Executes on button press in simu_on.
function simu_on_Callback(hObject, eventdata, handles)
% hObject    handle to simu_on (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set_param('estacion1234CP','SimulationCommand','start');
% --- Executes on button press in simu_off.
function simu_off_Callback(hObject, eventdata, handles)
% hObject    handle to simu_off (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set_param('estacion1234CP','SimulationCommand','stop');
```

Figura 133.Código que pone en marcha o detiene la simulación

Los 3 botones de elección de modo pieza son utilizados para cambiar el modo de elección. Pudiendo elegir entre elegir nosotros mismo las piezas o con un test implementado en Simulink (ya explicado) que elaborara una pieza de cada con las estaciones, o parar la fabricación. EL nombre del constant que modificara será Selector modo Pieza.

```
% --- Executes on button press in ciclo.
function ciclo_Callback(hObject, eventdata, handles)
% hObject    handle to ciclo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set_param('estacion1234CP/Selector modo pieza','Value', '1')

% --- Executes on button press in Pieza.
function Pieza_Callback(hObject, eventdata, handles)
% hObject    handle to Pieza (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set_param('estacion1234CP/Selector modo pieza','Value', '2')

% --- Executes on button press in Pausa.
function Pausa_Callback(hObject, eventdata, handles)
% hObject    handle to Pausa (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set_param('comunicacionenreestaciones/Selector modo pieza','Value', '3')
```

Figura 134.Código que modifica el valor Selector modo pieza

Como se puede ver en función del valor de Selector modo Pieza el valor de TipoPieza provendrá o de la máquina de estados, o del Constant Tipo Pieza que se modificara con los botones que se explicaran a continuación o será 0 para parar el proceso.

Los 6 botones de Pieza a fabricar modificaran el constant Tipo de Pieza, esta variable tendrá un valor de 1 a 6 (base de datos de piezas), y cada valor indicará un tipo de pieza.

El Código de estos 6 botones será el siguiente:

```
function NST_Callback(hObject, eventdata, handles)
% hObject    handle to NST (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set_param('estacion1234CP/Tipo de Pieza','Value', '1')


% --- Executes on button press in RST.
function RST_Callback(hObject, eventdata, handles)
% hObject    handle to RST (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set_param('estacion1234CP/Tipo de Pieza','Value', '2')

% --- Executes on button press in MST.
function MST_Callback(hObject, eventdata, handles)
% hObject    handle to MST (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set_param('estacion1234CP/Tipo de Pieza','Value', '3')

% --- Executes on button press in NCT.
function NCT_Callback(hObject, eventdata, handles)
% hObject    handle to NCT (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set_param('estacion1234CP/Tipo de Pieza','Value', '4')

% --- Executes on button press in RCT.
function RCT_Callback(hObject, eventdata, handles)
% hObject    handle to RCT (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set_param('estacion1234CP/Tipo de Pieza','Value', '5')

% --- Executes on button press in MCT.
function MCT_Callback(hObject, eventdata, handles)
% hObject    handle to MCT (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set_param('estacion1234CP/Tipo de Pieza','Value', '6')
```

Figura 135.Código que modifica el valor Tipo de pieza

Los 5 botones restantes sirven para dar marcha a cada una de las estaciones y al transporte. Si es pulsado el botón de marcha de las estaciones se realizara el trabajo individual pero Si se pulsa la marcha del transporte se pone en marcha el proceso completo. Al igual que los anteriores botones, estos modificarán el valor de una constante que pondría en marcha la estación. El código asociado a los botones es el siguiente:

```
% --- Executes on button press in Estacion1.
function Estacion1_Callback(hObject, eventdata, handles)
% hObject    handle to Estacion1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set_param('estacion1234CP/e1','Value', '1')

% --- Executes on button press in Estacion2.
function Estacion2_Callback(hObject, eventdata, handles)
% hObject    handle to Estacion2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set_param('estacion1234CP/e2','Value', '1')

% --- Executes on button press in Estacion4.
function Estacion4_Callback(hObject, eventdata, handles)
% hObject    handle to Estacion4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set_param('estacion1234CP/e4','Value', '1')

% --- Executes on button press in Estacion3.
function Estacion3_Callback(hObject, eventdata, handles)
% hObject    handle to Estacion3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set_param('estacion1234CP/e3','Value', '1')

% --- Executes on button press in cinta.
function cinta_Callback(hObject, eventdata, handles)
% hObject    handle to cinta (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set_param('estacion1234CP/cinta','Value', '1')
```

Figura 136.Código que modifica el valor e1, e2, e3, e4 y cinta

Anexo 2 3D World editor

A la hora de crear toda la estructura, todo parte de bloques más pequeños que se han ido uniendo para formarlo. Por tanto, a continuación se explicaran los pasos a realizar para crear un único objeto y como crear el siguiente para que este se encuentre acoplado al primero

En primer lugar, se creara un nuevo grupo de transformación (Group->Transform) al cual le cambiará el nombre que deseado. Esto se hace pulsando con el botón derecho sobre ROOT.

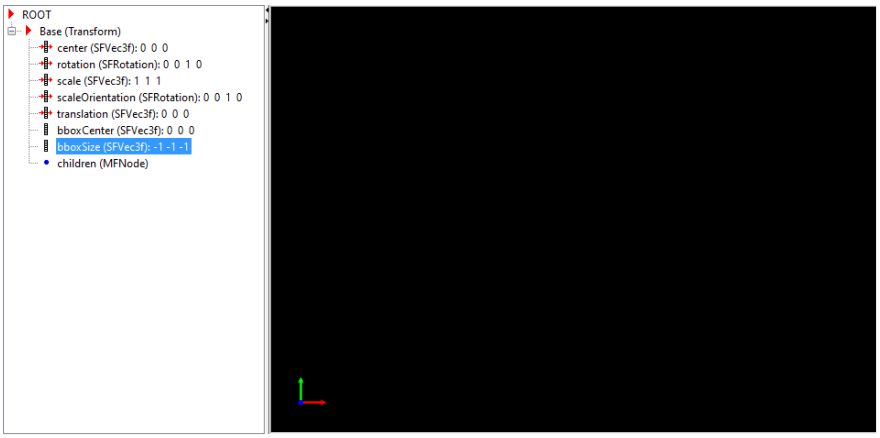


Figura 137.Primer Objeto creado en 3d World editor

Lo siguiente será añadir la geometría y la apariencia. Al Pulsar con botón derecho sobre children y se añadirán dos nuevos modos, el primero del tipo appearance y otro de tipo geometry, en este último habrá que elegir entre una serie de objetos predefinidos, con los cuales se pueden hacer formas más complejas. Como seria box (Cajas), Sphere (Esferas), cylinder (Cilindros) y cone (cono) que tendrá unas características específicas. Para este proyecto, se ha elegido la caja.

Para el bloque del tipo Appearance, se elegirá un bloque que se llama también Appearance. En él se podrá indicar las texturas y el material el cual va a ser el objeto que se va a utilizar.

Una vez hecho esto, se añadirá en el nodo material, un material perteneciente al nodo appearance. El esquema después de esto, quedaría del siguiente modo:

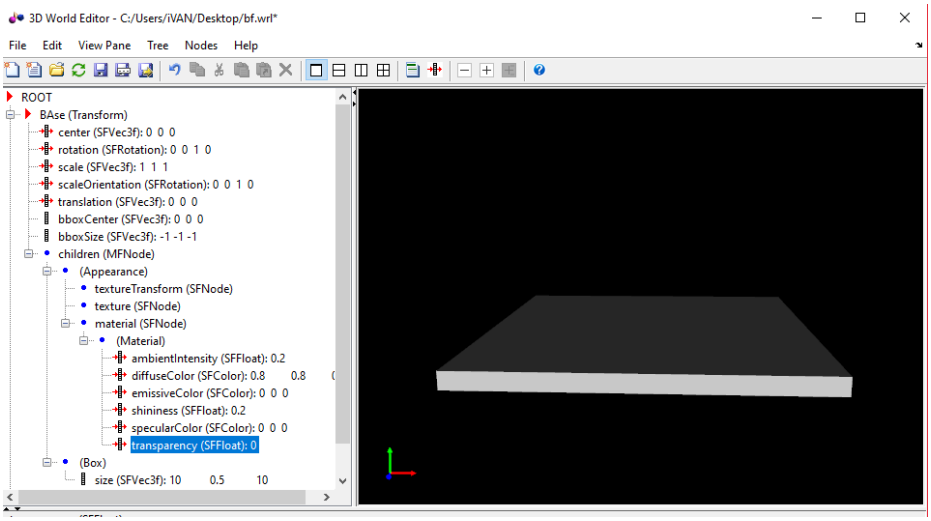


Figura 138.Creacion de los nodos geometry y appearance en 3d World editor

Para modificar las dimensiones del objeto, se cambiarán los valores que aparecen en la parte inferior izquierda de la pantalla, si se pulsa sobre ellos y en la parte inferior aparecerá una tabla con los 3 valores a modificar.

Una vez realizados los cambios, se podrá trasladar y girar el objeto como se desee, Para ello, en las características del nodo con el nombre Base, nos aparece 2 características, una llamada traslation(SFVec3f) y otra llamada rotation (SFRotation), que pueden ser utilizadas para mover y girar el objeto.

Para modificar los valores, ocurrirá igual que para el tamaño aparecerá una tabla en la parte inferior donde se podrá modificar los valores.

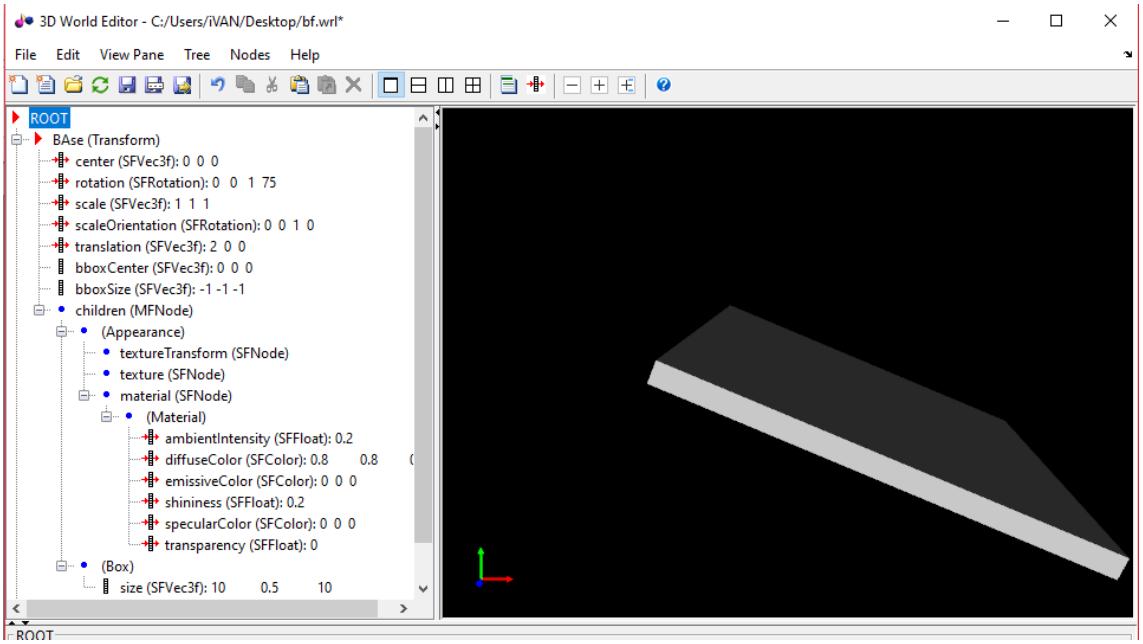


Figura 139.Pieza girada en 3d World editor

Una vez hecho esto, se añadirán otros bloques, que dependa de este primero, es decir, que si el primer bloque gira, el segundo lo haga con él.

Para ello en la característica llamada children, pulsando con el botón derecho se añadirán otro nodo de tipo Transform. Aparece otro objeto con el primero, totalmente vacío al cual habrá que añadir del mismo modo los bloques de geometry y appearance. Solo cabe decir que en este caso en vez de añadir otra caja, se ha añadido un cilindro.

Después de crear todo lo antes mencionado, nos quedaría del siguiente modo. Y ahora en caso de que se produzca un giro, el cilindro también realizara ese giro.

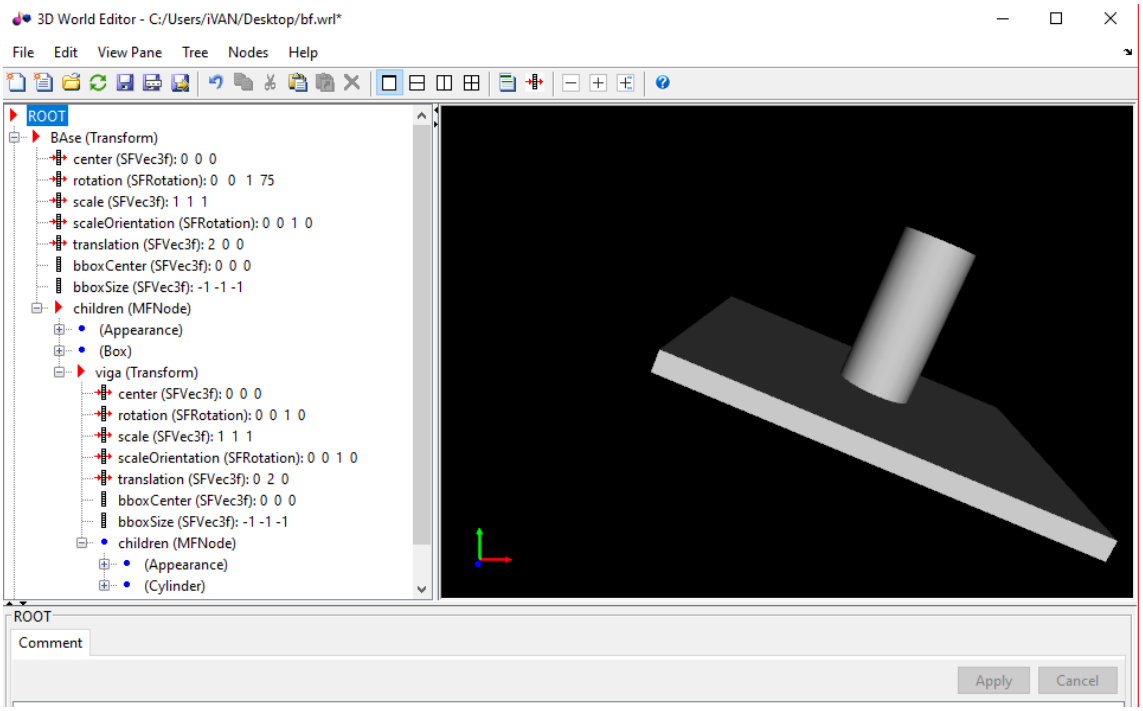


Figura 140.Segundo objeto creado perteneciente al primero en 3d World editor

También cabe explicar, otro nodo importante como sería el Viewpoint en el cual se podrá configurar una vista predeterminada con la que ver el montaje con mayor facilidad solo modificando la posición (position) de la cámara.

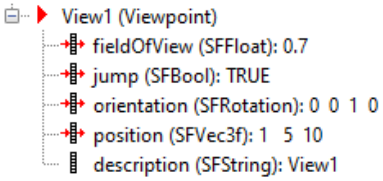


Figura 141.Creacion de punto de vista en 3d World editor

Para el Proyecto se ha realizado esto, cuantas veces ha sido necesario para crear un esquema simple de las estaciones. El cual posteriormente podrá ser movido por variables de Simulink.

El esquema para nuestras estaciones consta de unos 25 objetos, que han sido creados del mismo modo que lo anterior descrito.

Esquema Estacion 3

```
ROOT
(WorldInfo)
(NavigationInfo)
View1 (Viewpoint)
(Background)
BASE (Transform)
  center (SFVec3f): 0 0 0
  rotation (SFRotation): 0 0 1 0
  scale (SFVec3f): 1 1 1
  scaleOrientation (SFRotation): 0 0 1 0
  translation (SFVec3f): 0 -10 0
  bboxCenter (SFVec3f): 0 0 0
  bboxSize (SFVec3f): -1 -1 -1
  children (MFNode)
    (Shape)
    SOPORTE (Transform)
      center (SFVec3f): 0 0 0
      rotation (SFRotation): 0 0 1 0
      scale (SFVec3f): 1 1 1
      scaleOrientation (SFRotation): 0 0 1 0
      translation (SFVec3f): 0 0 0
      bboxCenter (SFVec3f): 0 0 0
      bboxSize (SFVec3f): -1 -1 -1
      children (MFNode)
        S1 (Transform)
          center (SFVec3f): 0 0 0
          rotation (SFRotation): 0 0 1 0
          scale (SFVec3f): 1 1 1
          scaleOrientation (SFRotation): 0 0 1 0
          translation (SFVec3f): 2 5 0
          bboxCenter (SFVec3f): 0 0 0
          bboxSize (SFVec3f): -1 -1 -1
          children (MFNode)
        S2 (Transform)
        SOPORTECARGADOR (Transform)
          center (SFVec3f): 0 0 0
          rotation (SFRotation): 0 0 1 0
          scale (SFVec3f): 1 1 1
          scaleOrientation (SFRotation): 0 0 1 0
          translation (SFVec3f): -1.5 1.5 2
          bboxCenter (SFVec3f): 0 0 0
          bboxSize (SFVec3f): -1 -1 -1
          children (MFNode)
        (Transform)
        (Transform)
        PALET (Transform)
View2 (Viewpoint)
```

Esquema Estacion 1

```
ROOT
(WorldInfo)
(NavigationInfo)
View1 (Viewpoint)
(Background)
BASE (Transform)
  center (SFVec3f): 0 0 0
  rotation (SFRotation): 0 0 1 0
  scale (SFVec3f): 1 1 1
  scaleOrientation (SFRotation): 0 0 1 0
  translation (SFVec3f): 0 -10 0
  bboxCenter (SFVec3f): 0 0 0
  bboxSize (SFVec3f): -1 -1 -1
  children (MFNode)
    (Shape)
    SOPORTE (Transform)
      center (SFVec3f): 0 0 0
      rotation (SFRotation): 0 0 1 0
      scale (SFVec3f): 1 1 1
      scaleOrientation (SFRotation): 0 0 1 0
      translation (SFVec3f): 0 0 0
      bboxCenter (SFVec3f): 0 0 0
      bboxSize (SFVec3f): -1 -1 -1
      children (MFNode)
        S1 (Transform)
        S2 (Transform)
        SOPORTECARGADOR (Transform)
          center (SFVec3f): 0 0 0
          rotation (SFRotation): 0 0 1 0
          scale (SFVec3f): 1 1 1
          scaleOrientation (SFRotation): 0 0 1 0
          translation (SFVec3f): -4.3 1.5 3
          bboxCenter (SFVec3f): 0 0 0
          bboxSize (SFVec3f): -1 -1 -1
          children (MFNode)
        (Transform)
        (Transform)
View2 (Viewpoint)
```

Figura 142.Esquemas finales para las estaciones 1 y 3

A continuación, se puede ver cómo quedaría el montaje final Perteneciente a estos esquemas.

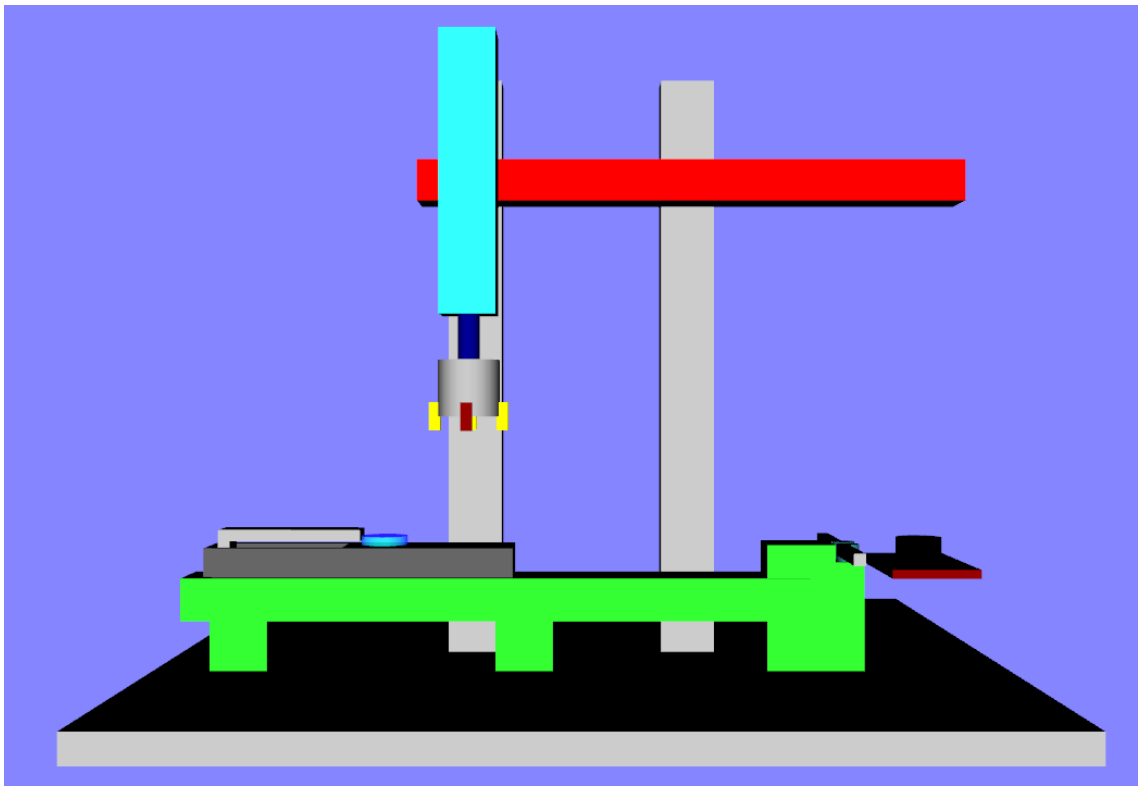


Figura 143.Esquema estación 3 en 3D World editor

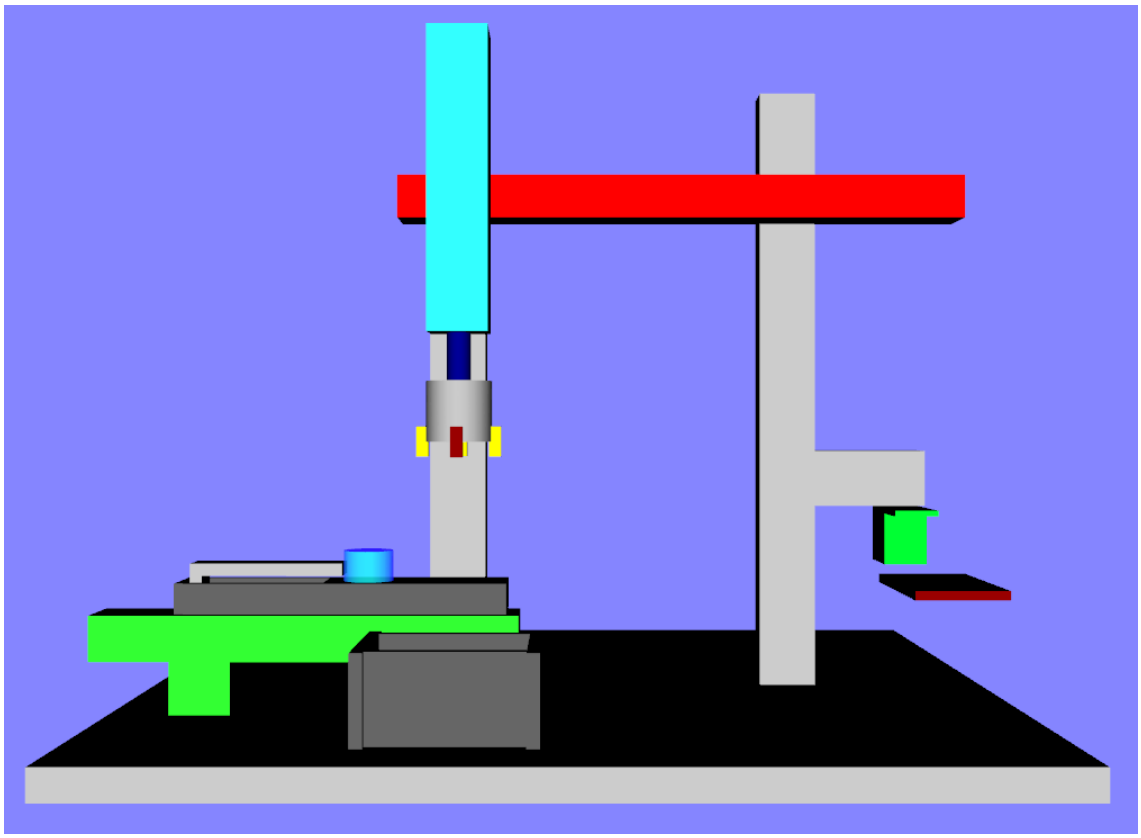


Figura 144.Esquema estación 1 en 3D World editor

A parte de crear estos 2 esquemas, también se creó un mando en 3D para indicar a las estaciones el tipo de pieza que se quiere fabricar y poner en marcha las estaciones

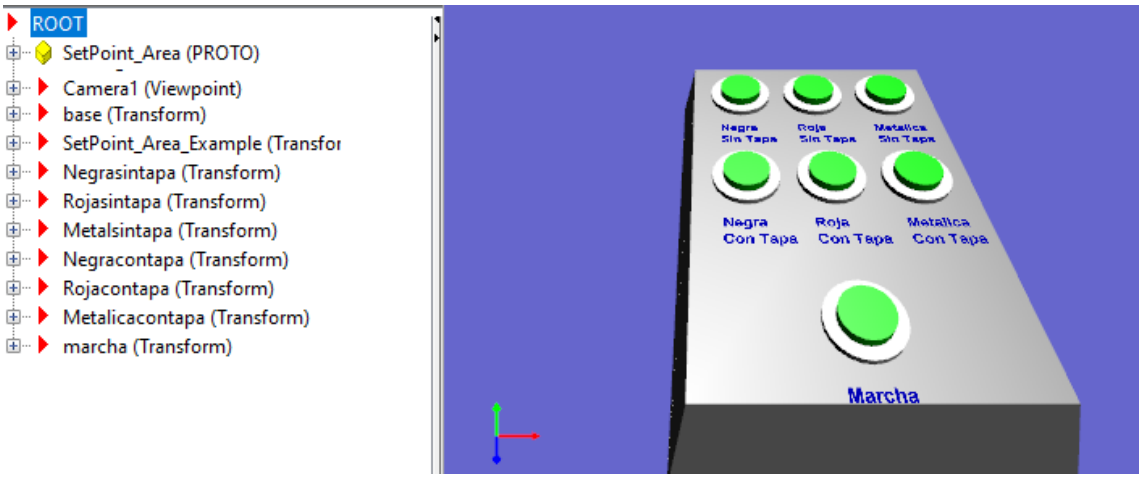


Figura 145.Esquema mando

En este esquema, simplemente es una base (parte gris) con 7 botones (cilindros verdes), una vista, y una área de setpoint. Esta área de setpoint será la que detectara la pulsación en cada uno de los botones. Esa pulsación se indica con una variable la cual se le pasara a la estación para que realice el proceso consecuente.

Para poder controlar desde Simulink la estructura generada en 3d World editor será necesario un bloque llamado VR Sink, el cual habrá que configurar para que abra el proyecto de realidad virtual creado, e indicando las variables las cuales se quieren controlar, ya sea, traslación en cualquier eje de un objeto o la rotación sobre un eje.

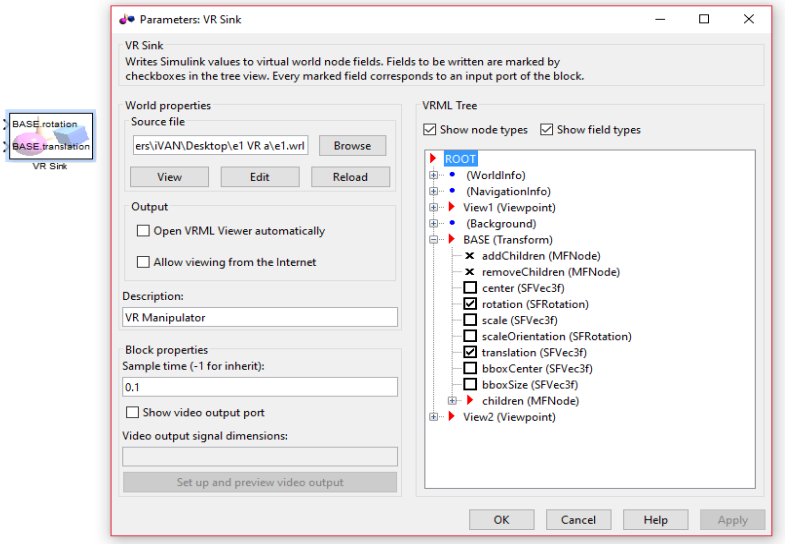


Figura 146.Configuración Bloque VR Sink

Para nuestras estaciones se controlaran principalmente los desplazamientos hechos por los brazos y un giro para la estación 3. Estos desplazamiento se realizaran a través de unas variables que se incrementaran o reducirán en ciertos estados determinados. Una vez elegidas las variables el bloque quedaría del siguiente modo:

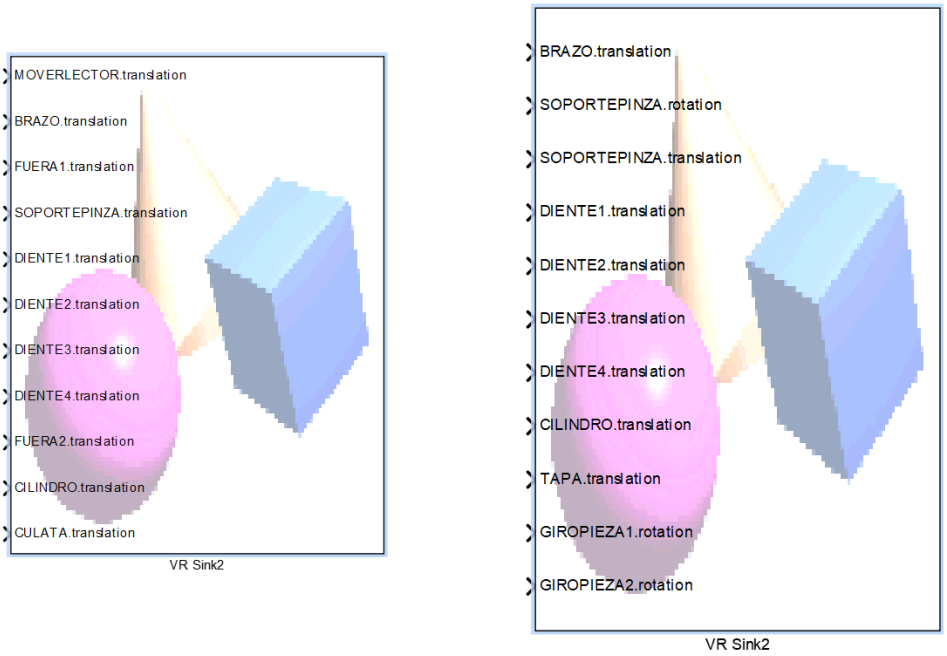


Figura 147.Bloque VR Sink con variables a controlar estación 1 y 3 respectivamente

El problema reside en que cada una de estas entradas es un vector de 3 coordenadas para la traslación y de 4 coordenadas para la rotación, por lo tanto, si se desea hacer una traslación

en el eje x, los ejes z e y deben permanecer constantes. Quedando el esquema final de Simulink junto con el chart antes explicado.

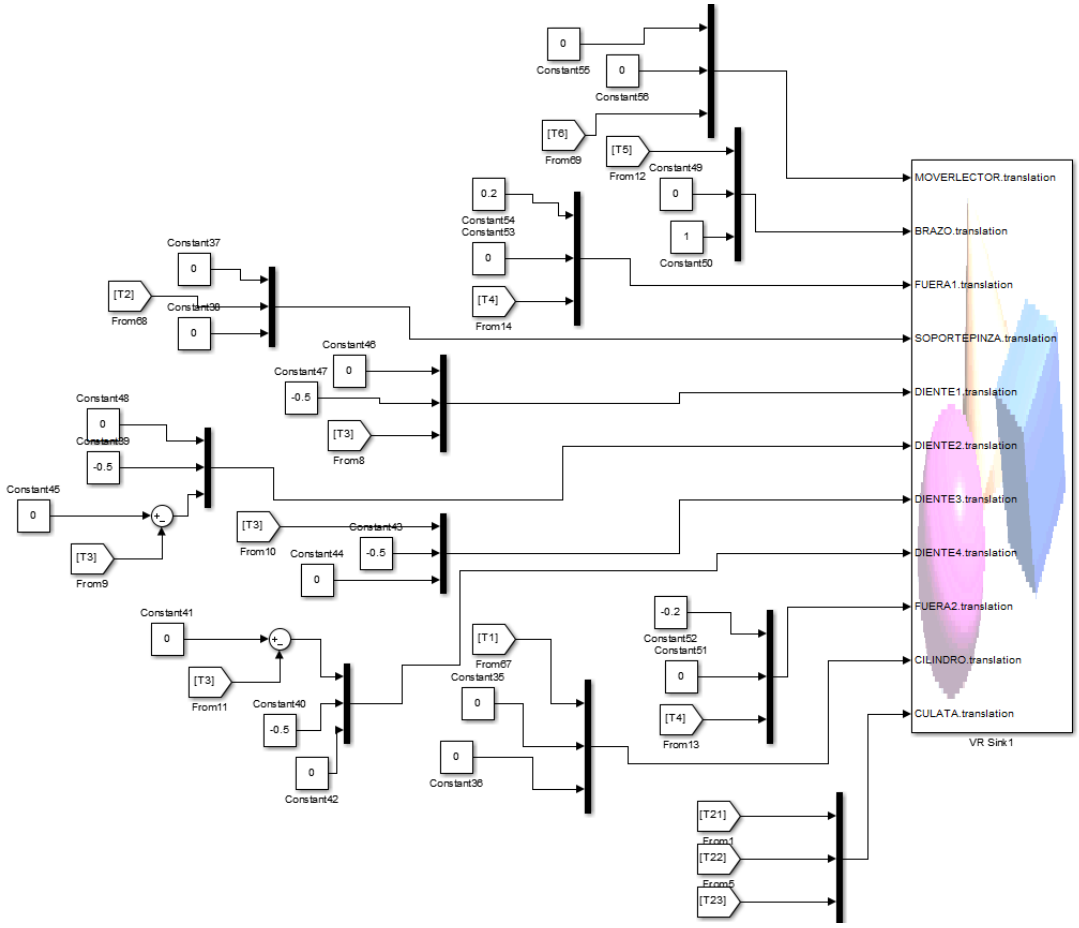


Figura 148.Esquema final 3D estación 1

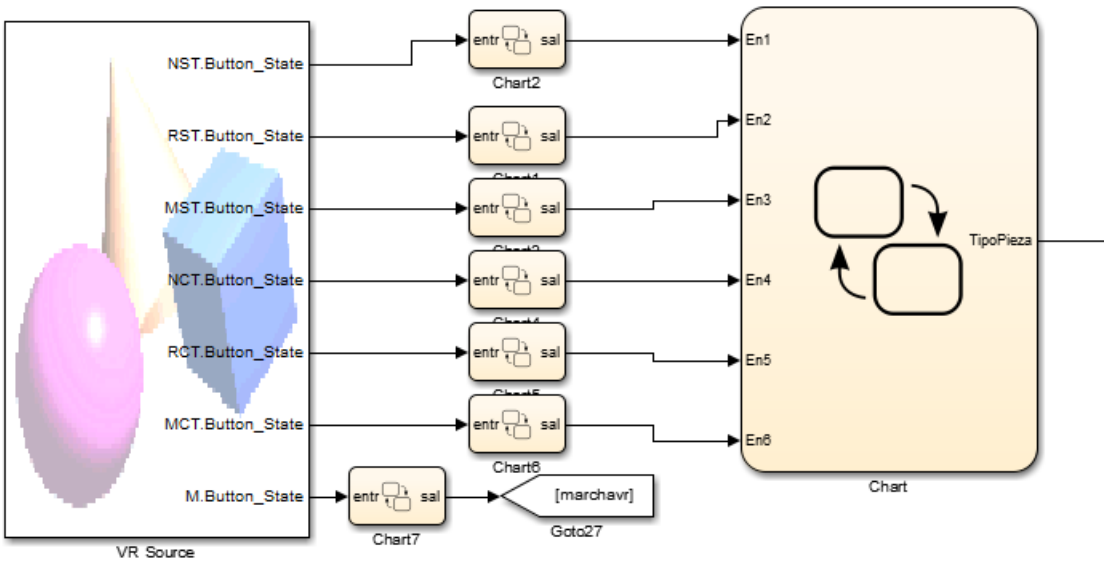


Figura 149.Esquema Mando 3d

Para este mando se tuvo que crear un StateFlow, en el cual se coloca un valor a tipoPieza en función del botón pulsado en el mando.

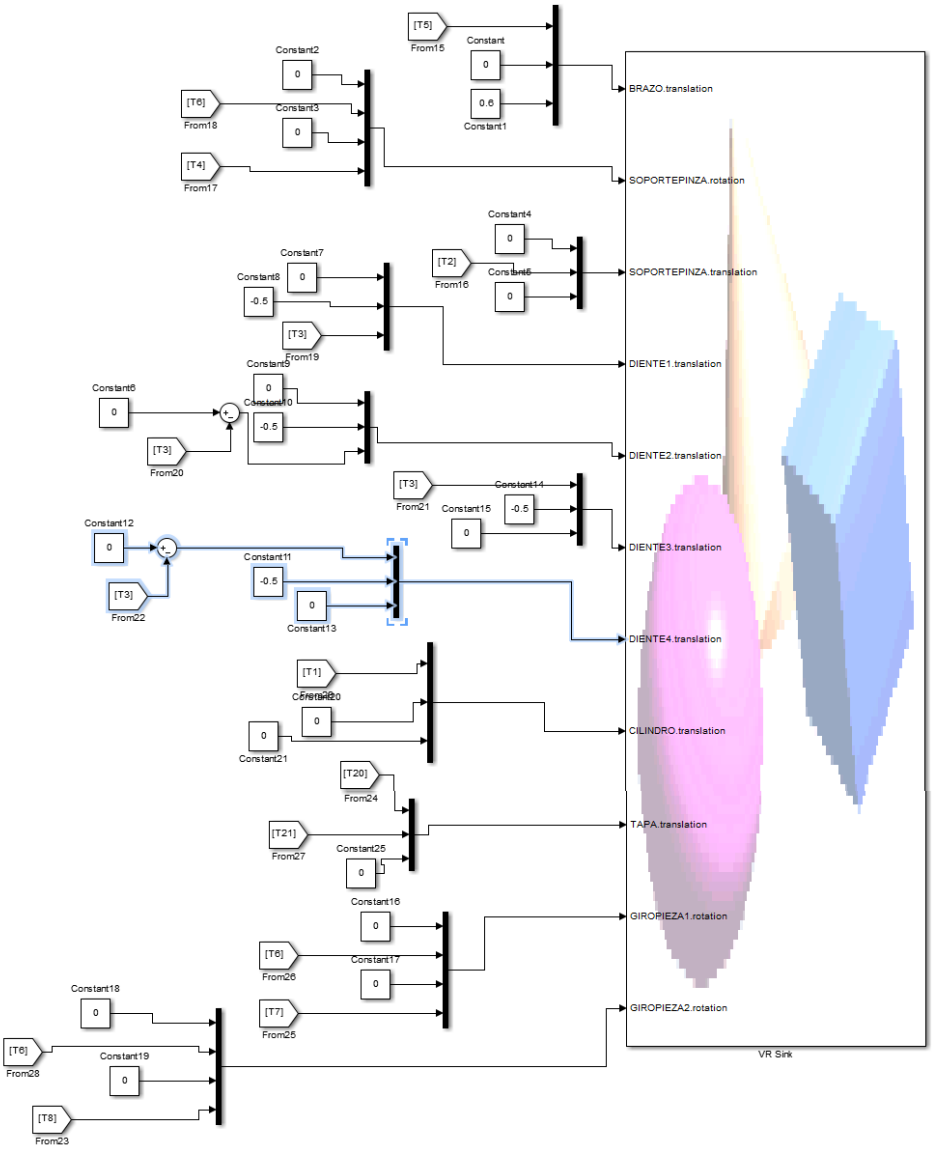


Figura 150Esquema Final estación 3

Además habrá que añadir las variables en las máquinas de estados para realizar los movimientos. A continuación, se muestra en ejemplo de una de esas variables encargada del movimiento de subir y bajar la pinza del modelo.

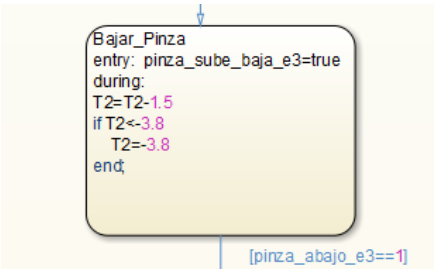


Figura 151.Ejemple estado Stateflow con variable utilizada en movimiento modelo 3D

Anexo 3 Control desde Rockwell

Para realizar la implementación para este control se partirá de las máquinas de estados implementadas en el control desde Simulink, y se realizaran una serie de modificaciones al añadir diferentes modos de funcionamiento y el identificador de producto. Una vez explicado como quedara la máquina de estados de Simulink, se explicara la programación de Rockwell y Vijeo.

Estación 2

Al añadir el identificador de producto, dos estados de error

El primer estado de error es en el caso de que por diversas razones llegue una pieza con tapa (4<=tipoPieza<=6), la estación permitirá que el palet siga camino con la pieza ya que con esas piezas no se tiene que realizar ningún proceso.

El segundo caso de error en caso de que la producción no se produzca correctamente, se escribirá en el identificador un código de error en el palet, explicado en la introducción.

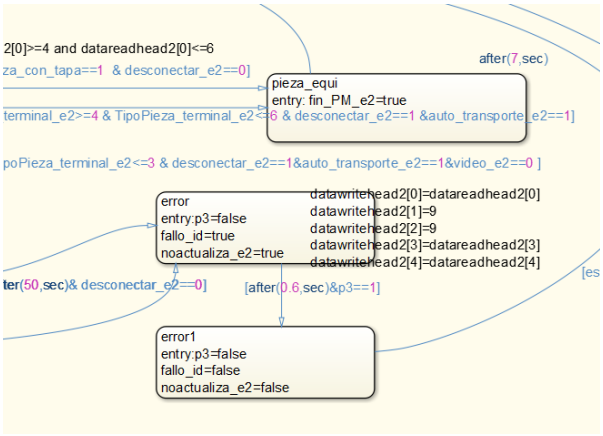


Figura 152.Estado error por pieza errónea (pieza_equi) y estado error de producción (error y error1)

Además de que el identificador se encarga de los estados de error también se encarga de leer los datos del palet para conocer el tipo de pieza que se encuentra en la palet y después de que la estación haya realizado el proceso escribirá en el palet el resultado, es decir, escribirá en las posiciones del muelle y embolo el valor numérico correspondiente en función de la base de datos explicado en la introducción.

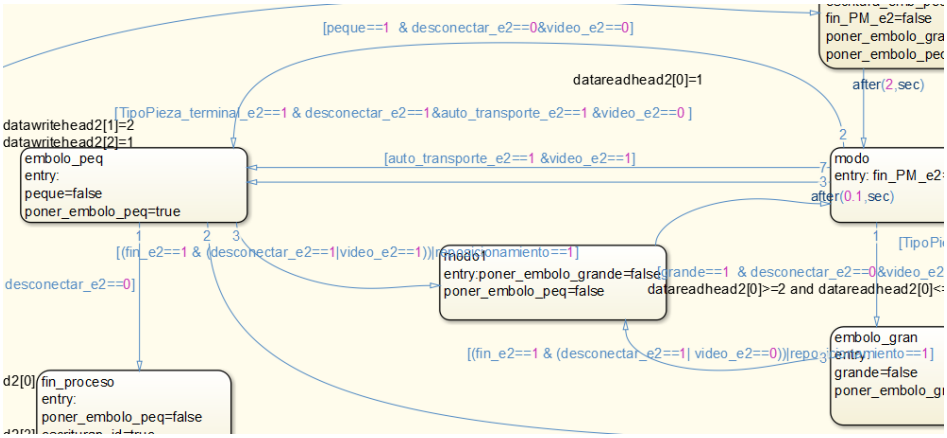


Figura 153.Transiciones a implementar el Rockwell, Comprobación valores I. Producto Estación 2

Como ya sea dicho, en Simulink no se puede utilizar las variables del identificador pero en Rockwell si, por tanto, se optó por hacer comprobaciones a esas variables y que cuando se cumplan ciertas condiciones, se pongan a 1 ciertas variables auxiliares que se utilizarán en Simulink como es el caso de las variables peque y grande de esta estación.

Y por último, hay otros 2 posibles errores en caso de que la escritura del identificador haya sido errónea, done se volverían a escribir los códigos de error en las variables del identificador.

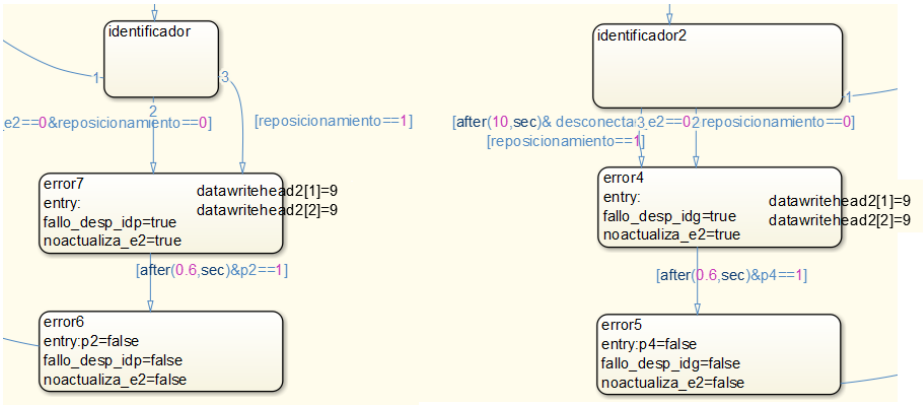


Figura 154.Estados correspondientes a los errores por fallo de escritura

Por lo tanto, el diagrama de estado de la estación 2 quedaría del siguiente modo:

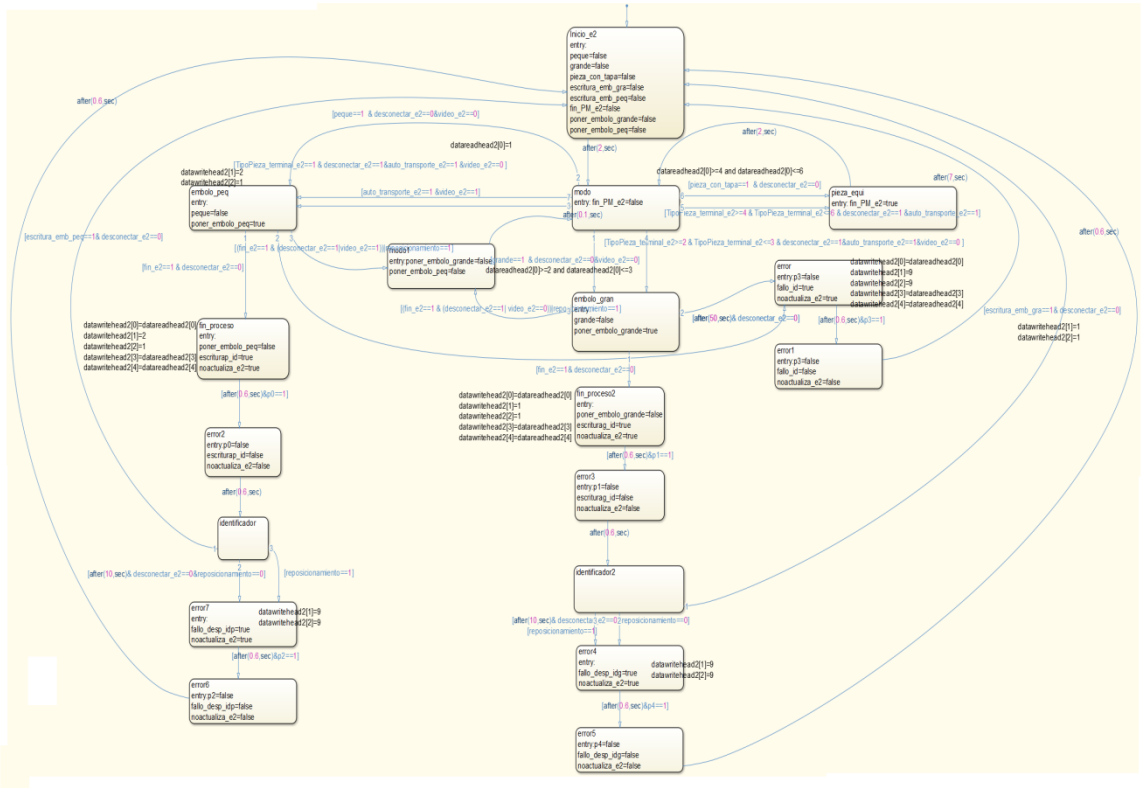


Figura 155.Diagrama completo estación 2

Una vez hecha la implementación se ha generado código para Rockwell con PLC Coder, el cual estará programado en la rutina Main_e2. A continuación, se pasará a la programación de Rockwell

El primer paso fue crear la rutina principal de nuestro programa, con la cual se controlará nuestra estación.

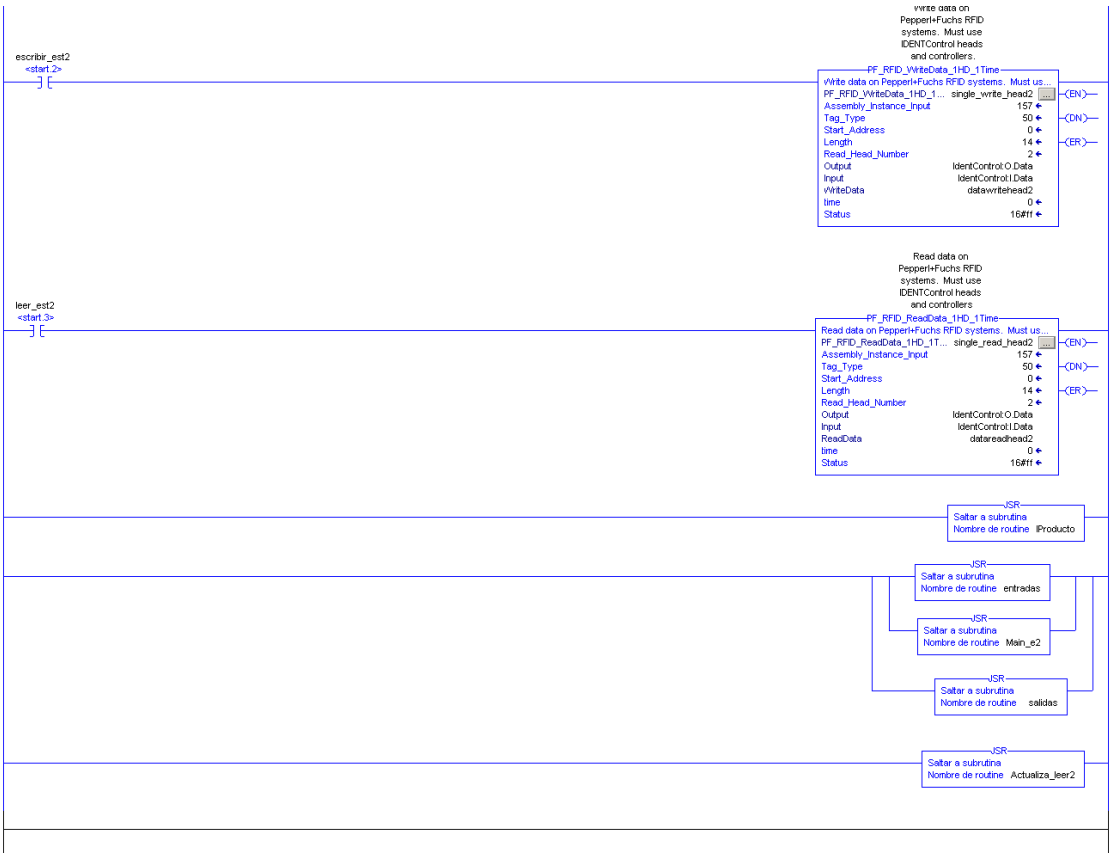


Figura 156.Programa Principal Estación 2

Como se puede observar en esta rutina principal, se lee la memoria de las estaciones mediante la activación del bit leer_est2, con esto se activa la instrucción Read Data, que lee la memoria del palet de cada estación y descarga su contenido en el vector DataReadHead2. Lo mismo para escribir, al activar escribir_est2, y esto activa la instrucción Write Data. Con esto, se escribe en la memoria del palet el contenido del vector DataWriteHead2.

Por último, en esta rutina principal, se puede ver el salto a otras 5 rutinas, que estarán siempre activas. La primera rutina (**Main_e2**), contiene el funcionamiento completo de la estación 2 la cual se ha implementado en Simulink, que se describirá a continuación, la siguiente rutina **entradas,** continua con **Actulizar_leer2**, que será la encargada de la lectura y escritura del identificador de producto, la rutina **salidas** y por último, la rutina **identificación**.

Rutinas para Estación 2

Una vez que se ha realizado todo esto, En Simulink se ha generado el código de la máquina de estados de la estación 2, y posteriormente en el programa RSLogix 5000 se ha importado como rutina.

Main_e2

Esta rutina es la que se ha importado desde Simulink, se partirá desde la máquina de estado completa de Simulink, se importará el código generado.

```
CASE i0_Main_e2.is_c3_Main_e2 OF
    1:
        (* During 'Inicio_e2': '<S1>:152' *)
        PLC_CODER_TIMER(i0_Main_e2.temporalCounter_i1, 2, 2000, i0_Main_e2.temp1);
        IF i0_Main_e2.temp1 THEN
            (* Transition: '<S1>:193' *)
            i0_Main_e2.is_c3_Main_e2 := 16;
            i0_Main_e2.fin_PM_e2 := 0;
            (* Outport: '<Root>/poner_embolo_grande' *)
            (* Entry 'modo': '<S1>:530' *)
            (* i0_Main_e2.poner_embolo_grande := 0;*)
                (* Outport: '<Root>/poner_embolo_peq' *)
                (*i0_Main_e2.poner_embolo_peq := 0;*)
        END_IF;
    2:
        (* During 'embolo_gran': '<S1>:540' *)
        IF i0_Main_e2.fin_e2 AND ( NOT i0_Main_e2.desconectar_e2) THEN
            (* Transition: '<S1>:536' *)
            i0_Main_e2.is_c3_Main_e2 := 13;
            PLC_CODER_TIMER(i0_Main_e2.temporalCounter_i1, 1, 0, i0_Main_e2.temp1);
            (* Outport: '<Root>/poner_embolo_grande' *)
            (* Entry 'fin_proceso2': '<S1>:539' *)
            i0_Main_e2.poner_embolo_grande := 0;
            (* Outport: '<Root>/escriturag_id' *)
            i0_Main_e2.escriturag_id := 1;
            (* Outport: '<Root>/noactualiza_e2' *)
            i0_Main_e2.noactualiza_e2 := 1;
        ELSIF (i0_Main_e2.fin_e2 AND (i0_Main_e2.desconectar_e2 OR ( NOT i0_Main_e2.video_e2))) OR i0_Main_e2.reposicionamiento THEN
            (* Transition: '<S1>:595' *)
            i0_Main_e2.is_c3_Main_e2 := 17;(*i0_Main_e2.is_c3_Main_e2 := 16;*)
            (* Outport: '<Root>/poner_embolo_grande' *)
            (* Entry 'modo1': '<S1>:530' *)
           i0_Main_e2.poner_embolo_grande := 0;
                (* Outport: '<Root>/poner_embolo_peq' *)
            i0_Main_e2.poner_embolo_peq := 0;
        ELSE
            PLC_CODER_TIMER(i0_Main_e2.temporalCounter_i1, 2, 50000, i0_Main_e2.temp1);
            IF i0_Main_e2.temp1 AND ( NOT i0_Main_e2.desconectar_e2) THEN
                (* Transition: '<S1>:532' *)
                i0_Main_e2.is_c3_Main_e2 := 4;
                PLC_CODER_TIMER(i0_Main_e2.temporalCounter_i1, 1, 0, i0_Main_e2.temp1);
                (* Entry 'error': '<S1>:531' *)
                i0_Main_e2.p3 := 0;
                (* Outport: '<Root>/fallo_id' *)
                i0_Main_e2.fallo_id := 1;
```

Figura 157.Rutina Main_e2

Salidas/Entradas

Al realizar la exportación desde Simulink y la importación en RSLogix, como ya se ha explicado se crea una variable vectorial cuyo nombre es i0_nombre_chart (para la estación 2 será i0_Main_e2) y las variables de Simulink serán distintas posiciones de ese vector creado en la importación. Por ejemplo: i0_Main_e2.palet_e2.

Por lo tanto, hay que coger esas variables de entrada y salida de Simulink e igualarlas a las variables de entrada y salida del RSLogix para poder mover la estación. Un ejemplo de entrada seria: i0_Main_e2.palet_e2:=Palet_est2;

Y otro de salida seria: Poner_embolo_peq:=i0_Main_e2.poner_embolo_peq;

Las rutinas Salidas y entradas son 2 rutinas en las cuales están implementadas esas igualaciones explicadas anteriormente.

```
(*Estacion 2*)
Poner_embolo_peq:=i0_Main_e2.poner_embolo_peq;
Poner_embolo_grande:=i0_Main_e2.poner_embolo_grande;
```

```
(*Estacion 2*)
i0_Main_e2.fin_e2:=Fin_est2;
i0_Main_e2.auto_transporte_e2:=i0_Cinta0.automatico_terminal_e2;

i0_Main_e2.palet_e2:=Palet_est2;
i0_Main_e2.desconectar_e2:=Desconectar_e2;
i0_Main_e2.TipoPieza_terminal_e2:=TipoPieza_Terminal_est1;
i0_Main_e2.reposicionamiento:=reposicionamiento_est1;
```

Figura 158.Programación rutinas salidas/entradas

Rutina Actualizar_leer2

La rutina Actualizar_leer2 es la siguiente:

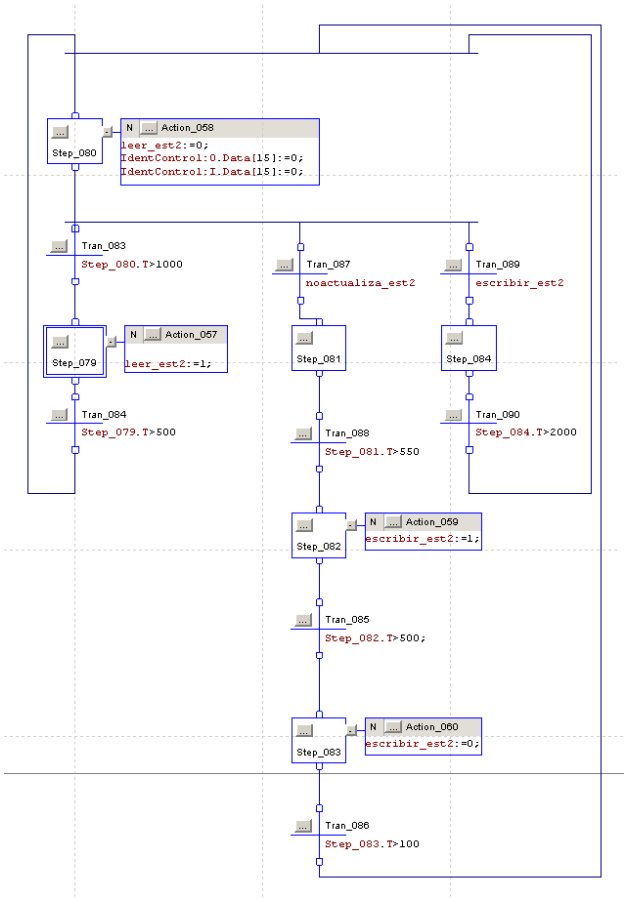


Figura 159.Rutina actualizar Estación 2

La función principal de esta rutina será la lectura de la memoria incluida en el Palet de la estación 2. Se actualizará cada un segundo. Con esta rutina, también se ha podido conseguir hacer la escritura de la memoria del Palet mediante el botón (Escribir Palet2) incluido en la pantalla de la Magelis. La última función de esta rutina, será la escritura mediante programa.
Se activará al final de cada estación la variable noactualiza_est2 durante un periodo superior a 500ms, consiguiendo que se ejecute la operación de escritura en la memoria del Palet.

Rutina Identificación

Como ya se ha ido explicando durante el capítulo 3, el modulo del identificador de producto, el cual será también se programara en Simulink, pero habrá que hacer pequeñas modificaciones (rutina identificación) a la hora de introducirlo en el programa RSlogix 5000, ya que las variables que se utilizan para la escritura y lectura de los palets no son accesibles desde Simulink.

Por lo tanto, se han utilizado variables auxiliares en Simulink las cuales se modificaran dependiendo de las lecturas del identificador de producto y, viceversa, habrá otras variables que ocasionaran modificaciones en las variables de escritura del identificador.

Para la estación 2 habrá que realizar varios bloques if los cuales modificaran los valores de las variables auxiliares y de las variables del identificador. Son los siguientes:

```
if i0_Main_e2.is_c3_Main_e2 = 16 then
    if datareadhead2[0]=1 then
    i0_Main_e2.peque:=1;
    end_if;
end_if;

if i0_Main_e2.is_c3_Main_e2 = 16 then
    if datareadhead2[0]>=2 and datareadhead2[0]<=3 then
    i0_Main_e2.grande:=1;
    end_if;
end_if;

if i0_Main_e2.is_c3_Main_e2 = 16 then
    if datareadhead2[0]>=4 and datareadhead2[0]<=6 then
    i0_Main_e2.pieza_con_tapa:=1;
    end_if;
end_if;

if i0_Main_e2.is_c3_Main_e2 = 15 then
    if datareadhead2[1]=1 and datareadhead2[2]=1 then
    i0_Main_e2.escritura_emb_gra:=1;
    end_if;
end_if;

if i0_Main_e2.is_c3_Main_e2 = 14 then
    if datareadhead2[1]=2 and datareadhead2[2]=1 then
    i0_Main_e2.escritura_emb_peq:=1;
    end_if;
end_if;

if i0_Main_e2.is_c3_Main_e2 = 12 then
    if i0_Main_e2.escriturap_id then
    datawritehead2[0]:=datareadhead2[0];
    datawritehead2[1]:=2;
    datawritehead2[2]:=1;
    datawritehead2[3]:=datareadhead2[3];
    datawritehead2[4]:=datareadhead2[4];
    i0_Main_e2.p0:=1;
    end_if;
end_if ;

if i0_Main_e2.is_c3_Main_e2 = 11 then
    if i0_Main_e2.fallo_desp_idp then
    datawritehead2[1]:=9;
    datawritehead2[2]:=9;
    i0_Main_e2.p2:=1;
    end_if;
end if;
```

Figura 160.Programación rutina identificación para estación 2

Como se puede ver, delante de cada una de los bloques if otro bloque if, el cual comprueba en la programación Main_e2 la variable io_Main_e2.is_ce_Main_e2, la cual indica en qué estado del proceso se encuentra.

Comunicación Magelis

En primer lugar se crearan 3 variables en cada uno de los autómatas que se refieran a lo mismo, es decir, se crean 3 variables en el autómata Rockwell, en nuestro caso, Fin_est2, indica que el proceso ha acabado, Poner_embolo_peq, indica que hay que realizar el proceso para poner un embolo pequeño en la pieza del palet, y, por último, Poner_embolo_grande, indica que hay que realizar el proceso para poner un embolo grande en la pieza del palet. Y luego se crearán esas 3 variables en el autómata Rockwell.

Una vez creadas las variables, en la magelis habrá que crear una decisión para modificar variables de un autómata en función de las variables del otro. En nuestro caso, se modificarán las 2 variables de poner embolo del autómata Schneider con los valores de las variables del autómata Rockwell, y con la variable de fin de proceso será al revés, se modificarán las variables de Rockwell con el valor de las variables de Schneider.

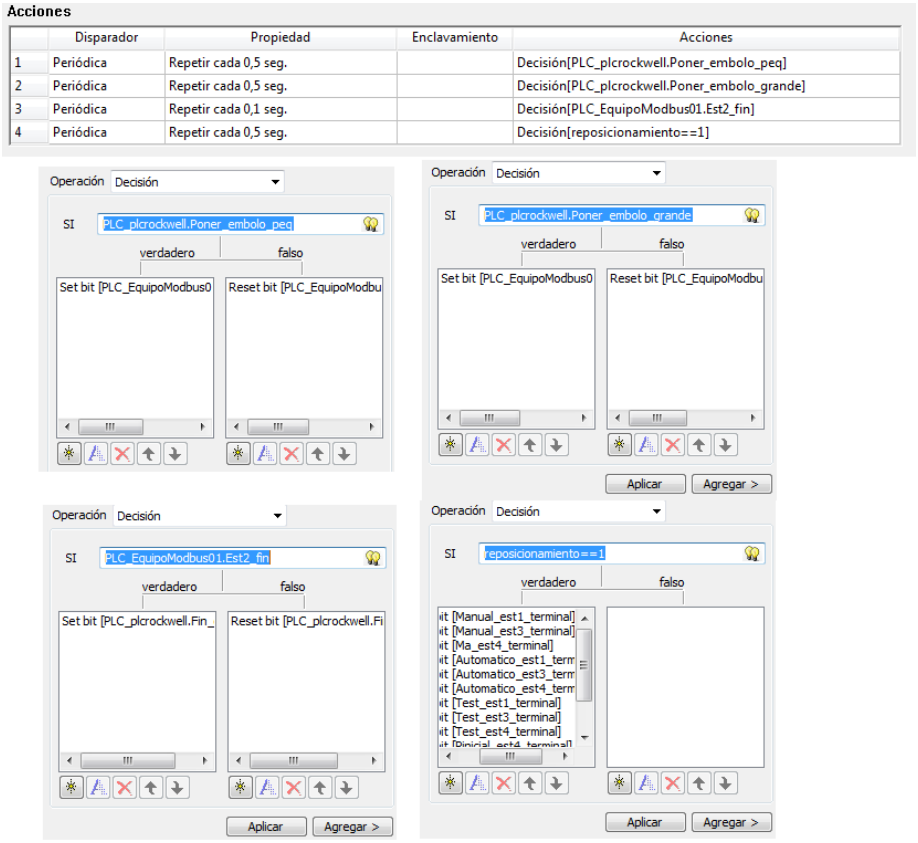


Figura 161.Decisiones implementadas en magelis.

Supervisión mediante Magelis.

En este punto se hablará de la supervisión de la estación mediante una Magelis XBTGT4330. El programa ha sido creado en Vijeo Designer. A continuación, se describirán las distintas pantallas creadas para la Magelis.

-Pantalla Menú:

Lo primero que hemos creado ha sido un menú, que nos llevará a las distintas pantallas del programa. Desde el Menú puedes acceder al control de la estación, al funcionamiento de la estación, a la guía gemma, al identificador de productos y finalmente a la elección de pieza a fabricar, como se muestra en la siguiente figura:

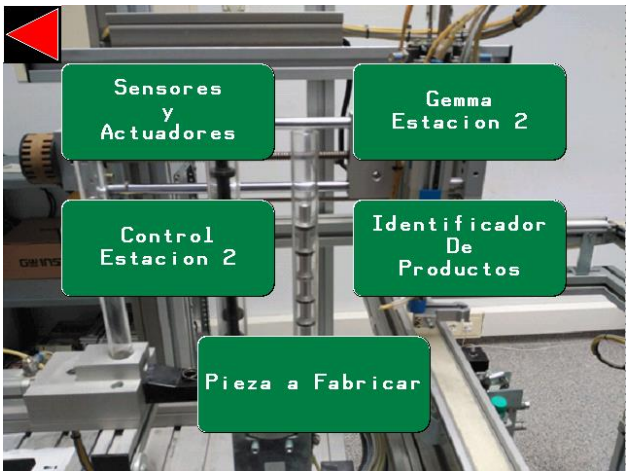


Figura 162.Pantalla menú Estación 2

-Pantalla Sensores y Actuadores.

En esta pantalla se puede ver una imagen de la estación 2, que cambia en tiempo real dependiendo de las diferentes posiciones de la estación. Esta pantalla también incorpora unos pilotos que indican si los actuadores o los sensores están activados o desactivados. Incluye también una imagen actualizada de la pieza que hay actualmente en el Palet. Y por último incorpora un piloto de emergencia, que se cambia a color rojo si ocurre alguna emergencia.

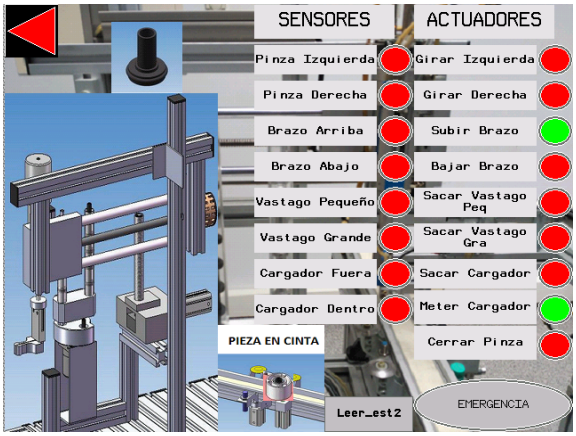


Figura 163.Pantalla Sensores y Actuadores Estación 2

-Pantalla Control Estación 2

En esta pantalla se puede ver diferentes botones para controlar la estación en todo momento. En esta pantalla se podrá cambiar entre los diferentes modos de funcionamiento, ya sea modo automático, modo manual, o test. Además de eso incorpora también los sensores y actuadores que aparecen en la pantalla anterior para cuando nos encontremos en modo manual.

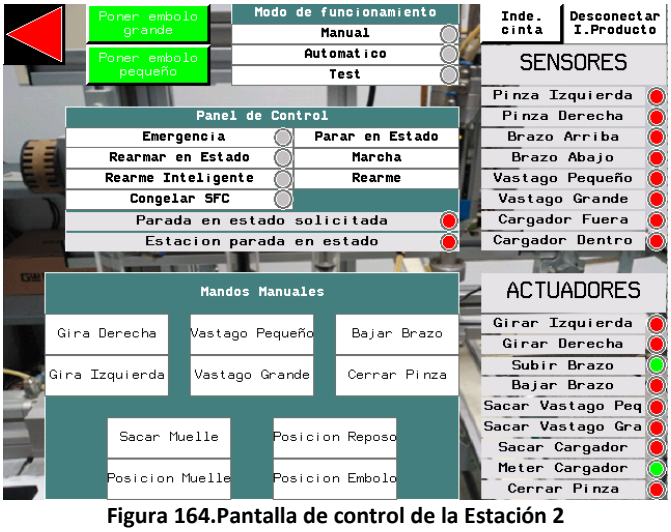


Figura 164.Pantalla de control de la Estación 2

-Pantalla guía gemma

Esta pantalla indicaría el funcionamiento global del programa controlado por la pantalla anterior, al igual que la pantalla anterior, esto tampoco está implementado en nuestro programa.

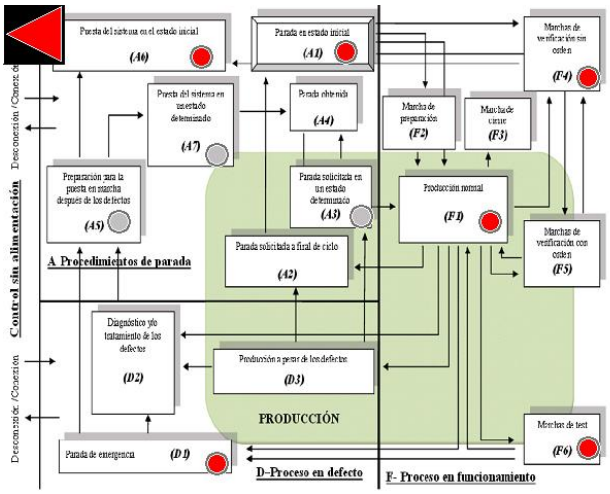


Figura 165.Pantalla Guía Gemma Estación 2

-Pantalla Identificador de productos

Con esta pantalla conseguimos leer y escribir mediante RFID en la memoria incluida en el palet. Está compuesta de 3 botones (leer Palet2, escribir Palet2 y borrar Palet2).

- Leer Palet2→ Con este botón leemos la memoria del Palet 2, descargando sus datos en el vector DataReadHead2, que es mostrado en la parte derecha de la pantalla.
- Escribir Palet2→ Con este botón escribimos en la memoria los datos que estén escritos en el vector DataWriteHead2, este vector se puede modificar, ya que está definido en la parte central de la pantalla.
- Borrar Palet2→ Con este botón cambiamos todos los valores del vector DaraWriteHead2 a 0, y después se activa el botón Escribir Palet 2 automáticamente, consiguiendo así, poner a 0 los 9 primeros valores de la memoria por RFID.

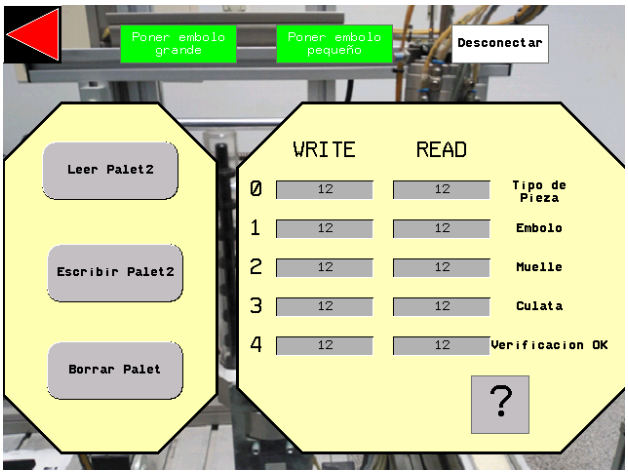
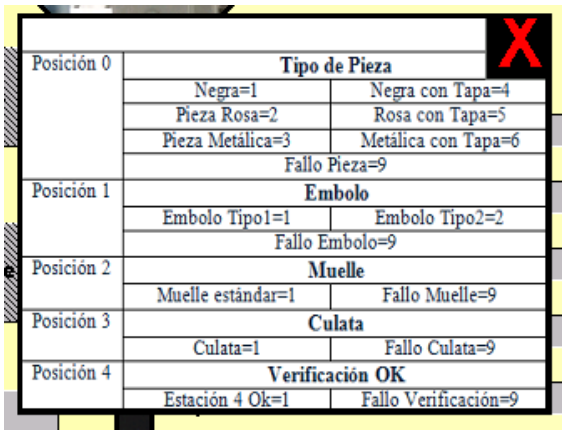


Figura 166.Pantalla Identificador de Productos Estación 2

Esta pantalla también cuenta con dos ventanas emergentes, la primera solo salta si se intenta activar leer Palet2 y escribir Palet2 al mismo tiempo. Esto es debido a que no puedes escribir y leer al mismo tiempo, solo se puede realizar una acción al mismo tiempo. La segunda es informativa, y salta al pulsar el interrogante, informando del significado de los diferentes números que pudieran aparecer en la memoria del palet.

Además de estas operaciones, a la derecha de la imagen, aparece el vector DataReadHead2, que informa del contenido del Palet, ya que es la lectura de su memoria. La base de datos utilizada como podemos ver en la ventana emergente es la siguiente:



Posición 0	**Tipo de Pieza**	
	Negra=1	Negra con Tapa=4
	Pieza Rosa=2	Rosa con Tapa=5
	Pieza Metálica=3	Metálica con Tapa=6
	Fallo Pieza=9	
Posición 1	**Embolo**	
	Embolo Tipo1=1	Embolo Tipo2=2
	Fallo Embolo=9	
Posición 2	**Muelle**	
	Muelle estándar=1	Fallo Muelle=9
Posición 3	**Culata**	
	Culata=1	Fallo Culata=9
Posición 4	**Verificación OK**	
	Estación 4 Ok=1	Fallo Verificación=9

Figura 167.Ventana emergente base de datos de la memoria

- **Pantalla de Fabricación**

Mediante esta pantalla elegiremos el tipo de pieza a fabricar. Una vez seleccionada la pieza a fabricar, deberemos pulsar el botón de marcha de la botonera, o el propio botón de marcha disponible en esta misma pantalla. También incorpora un botón "Fabricar Todas Las Piezas" que hace que salgan todas las piezas, sin desechar ninguna.

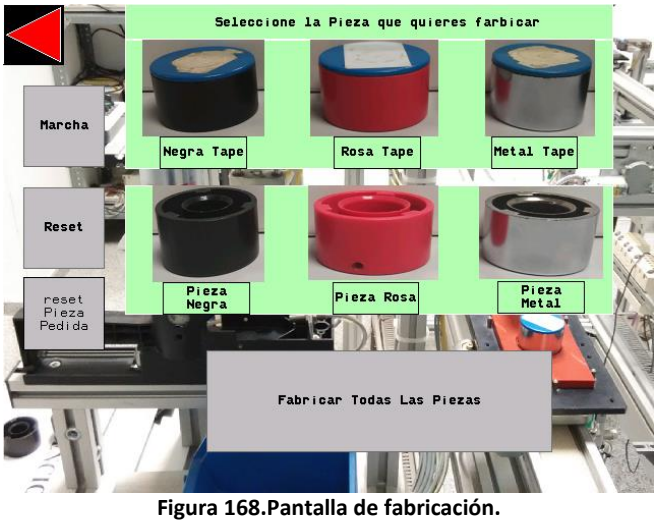


Figura 168.Pantalla de fabricación.

Estación 3

Después de añadir las modificaciones realizadas a la hora de introducir los otros modos de funcionamiento, el identificador de producto y la comunicación con el transporte, se realizaron modificaciones al modo automático que se explican a continuación además de explicar el resto del diagrama de estado y los modos de funcionamiento.

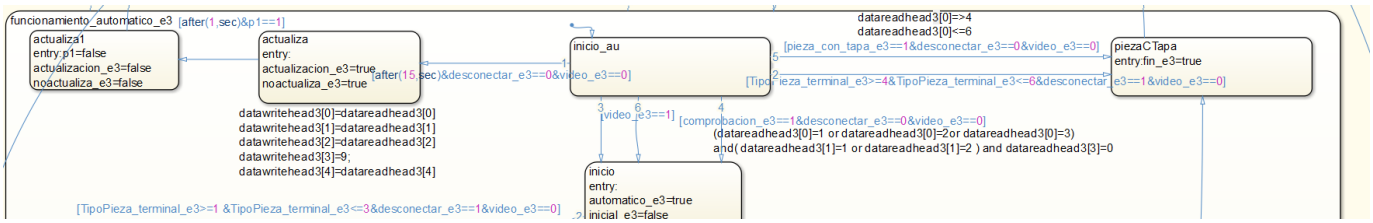


Figura 169.Estados iniciales añadidos para la lectura del palet

Al finalizar el ciclo del modo automático, se añadió varios estados cuyo destino es escribir en el la variable del identificador, que la tapa esta puesta (posición 4 del vector igual a 1) o, si por el contrario, se ha producido un error en cuyo caso se escribirá un código de error (posición 4 del vector igual a 9).

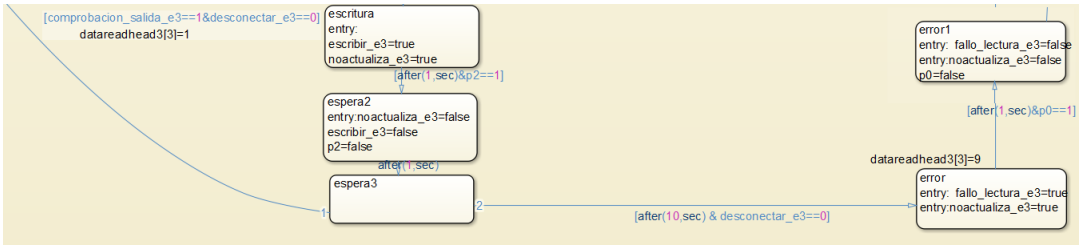


Figura 170.Estados de escritura en el palet

Al igual para la estación 1, tenemos los mismos estados programados aunque con las variables para esta estación.

En Primer lugar tenemos el estado de inicio, donde empieza el diagrama de estados e inicializamos alguna variable, y posteriormente tenemos el estado modo, el cual nos podremos elegir con la ayuda de la magelis en qué modo de funcionamiento queremos producir.

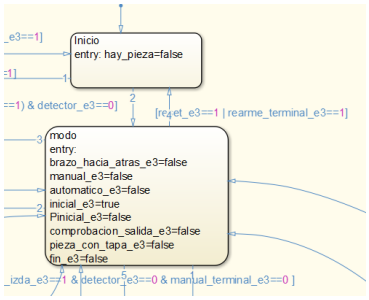


Figura 171.Parte de diagrama estado Inicio y Elección de la Estación 3

En el modo **manual** solo se modifican variables internas y el movimiento de la estación se realizara desde la magelis.

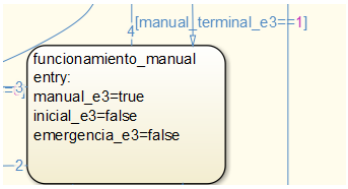


Figura 172.Parte de diagrama de Estado correspondiente al modo manual en la Estación 3

Siempre habrá que hacer un **reposicionamiento inicial** la estación para así evitar problemas en la producción. La posición inicial propicia para esta estación será cuando el brazo se encuentre arriba y atrás con la mordaza abierta y el cargador se encuentre recogido.

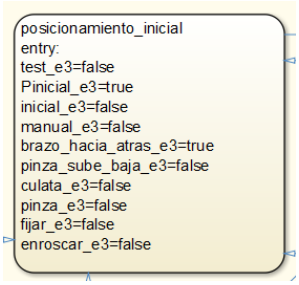


Figura 173.Parte de diagrama de Estado correspondiente Posicionamiento inicial en la Estación 3

También tenemos un estado de **emergencia** en el cual la estación para inmediatamente la producción en caso de que se hay pulsado el botón de emergencia de la regleta de la estación o de la terminal. Podremos salir de este estado cuando pulsemos el botón reset de la terminal.

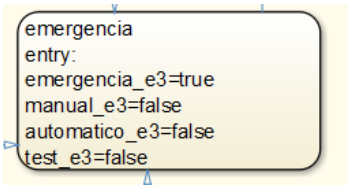


Figura 174. Estado Emergencia

Y por último, el modo **test**, en el cual se han implementado todos los movimientos posibles de la estación con el objetivo de comprobar posibles problemas en ella.

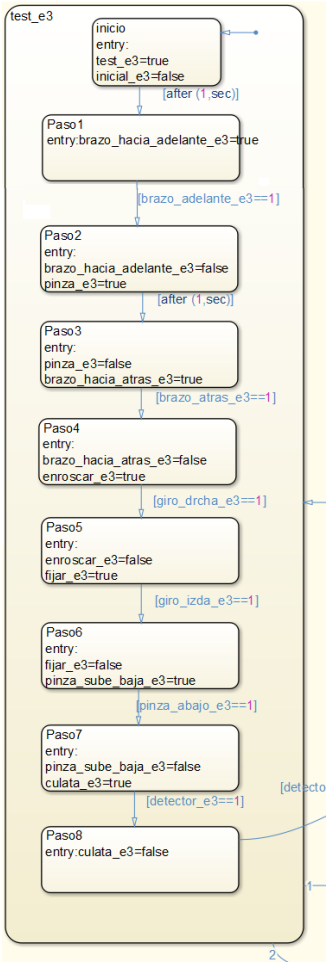


Figura 175.Modo test

Uniendo todos los estados ya explicados obtenemos el diagrama de estados completo para la estación 3.

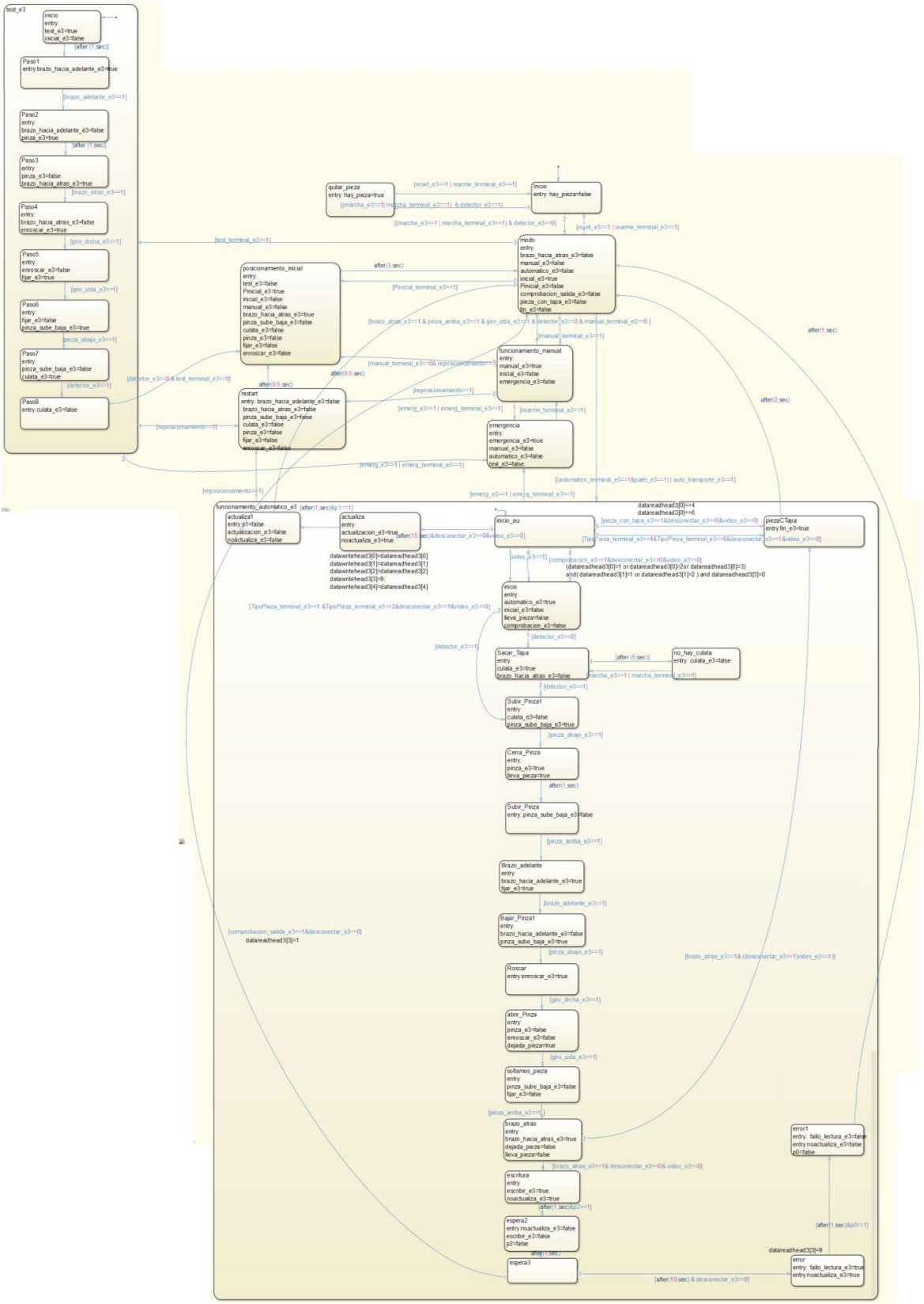


Figura 176.Programa completo Estación 3

Una vez hecha la implementación se ha generado código para Rockwell con PLC Coder, el cual estará programado en la rutina Main_e3. A continuación, se pasará a la programación de Rockwell

El primer paso fue crear la rutina principal de nuestro programa, con la cual controlaríamos el control de nuestra estación.

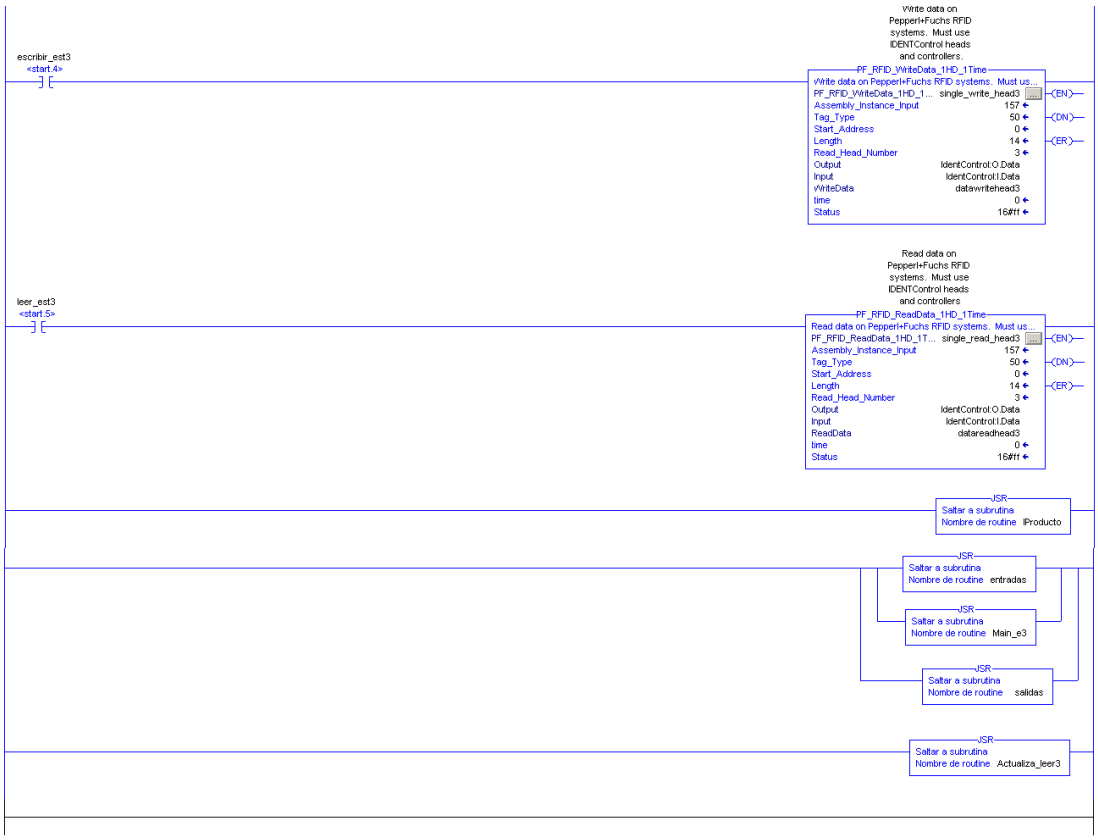


Figura 177.Programa Principal Estación 3

Como se puede observar en esta rutina principal, se lee la memoria de las estaciones mediante la activación del bit leer_est3, con esto se activa la instrucción Read Data, que lee la memoria del palet de cada estación y descarga su contenido en el vector DataReadHead3. Lo mismo para escribir, al activar escribir_est3, y esto activa la instrucción Write Data. Con esto, se escribe en la memoria del palet el contenido del vector DataWriteHead3.

Por último, en esta rutina principal, se puede ver el salto a otras 5 rutinas, que estarán siempre activas. La primera rutina (**Main_e3**), contiene el funcionamiento completo de la estación 2 la cual se ha implementado en Simulink, que se describirá a continuación, la siguiente rutina **entradas,** continua con **Actulizar_leer3**, que será la encargada de la lectura y escritura del identificador de producto, la rutina **salidas** y por último, la rutina **identificación**.

Rutinas estación 3

Una vez que se ha realizado todo esto, En Simulink se ha generado el código de la máquina de estados de la estación 3, y posteriormente en el programa RSLogix 5000 se ha importado como rutina.

Main_e3

Esta rutina es la que se ha importado desde Simulink, se partirá desde la máquina de estado completa de Simulink, se importará el código generado.

```
CASE i0_Main_e3.is_funcionamiento_automatico_e3 OF
    1:
        (* During 'Bajar_Pinzal': '<S1>:32' *)
        IF i0_Main_e3.pinza_abajo_e3 THEN
            (* Transition: '<S1>:33' *)
            i0_Main_e3.is_funcionamiento_automatico_e3 := 4;
            (* Outport: '<Root>/enroscar_e3' *)
            (* Entry 'Roscar': '<S1>:34' *)
            i0_Main_e3.enroscar_e3 := 1;
        END_IF;
    2:
        (* During 'Brazo_adelante': '<S1>:30' *)
        IF i0_Main_e3.brazo_adelante_e3 THEN
            (* Transition: '<S1>:31' *)
            i0_Main_e3.is_funcionamiento_automatico_e3 := 1;
            (* Outport: '<Root>/brazo_hacia_adelante_e3' *)
            (* Entry 'Bajar_Pinzal': '<S1>:32' *)
            i0_Main_e3.brazo_hacia_adelante_e3 := 0;
            (* Outport: '<Root>/pinza_sube_baja_e3' *)
            i0_Main_e3.pinza_sube_baja_e3 := 1;
        END_IF;
    3:
        (* During 'Cerra_Pinza': '<S1>:20' *)
        PLC_CODER_TIMER(i0_Main_e3.temporalCounter_i1, 2, 1000, i0_Main_e3.temp1);
        IF i0_Main_e3.temp1 THEN
            (* Transition: '<S1>:25' *)
            i0_Main_e3.is_funcionamiento_automatico_e3 := 6;
            (* Outport: '<Root>/pinza_sube_baja_e3' *)
            (* Entry 'Subir_Pinza': '<S1>:26' *)
            i0_Main_e3.pinza_sube_baja_e3 := 0;
        END_IF;
```

Figura 178.Rutina Main_e3

Salidas/Entradas

Al realizar la exportación desde Simulink y la importación en RSLogix, como ya se ha explicado se crea una variable vectorial cuyo nombre es i0_nombre_chart (para la estación 3 será i0_Main_e3) y las variables de Simulink serán distintas posiciones de ese vector creado en la importación. Por ejemplo: i0_Main_e3.palet_e3.

Por lo tanto, hay que coger esas variables de entrada y salida de Simulink e igualarlas a las variables de entrada y salida del RSLogix para poder mover la estación. Un ejemplo de entrada seria: i0_Main_e3.palet_e3:=Palet_est3;

Y otro de salida seria: Pinza_est3:=i0_Main_e3.pinza_e3;

Las rutinas Salidas y entradas son 2 rutinas en las cuales están implementadas esas igualaciones explicadas anteriormente.

```
(*Salidas estacion 3*)
    Cinta_avanza_est3:=i0_Main_e3.brazo_hacia_adelante_e3;
    Cinta_retrocede_est3:=i0_Main_e3.brazo_hacia_atras_e3;
    Culata_est3:=i0_Main_e3.culata_e3;
    Roscar_est3:= i0_Main_e3.enroscar_e3;
    Fijar_est3:= i0_Main_e3.fijar_e3;
    Pinza_est3:= i0_Main_e3.pinza_e3;
    Pinza_baja_est3:= i0_Main_e3.pinza_sube_baja_e3;
    F4Gemma_est3:=i0_Main_e3.manual_e3;
    F1Gemma_est3:=i0_Main_e3.automatico_e3;
    F6Gemma_est3:=i0_Main_e3.test_e3;
    D1Gemma_est3:=i0_Main_e3.emergencia_e3;
    A6Gemma_est3:=i0_Main_e3.Pinicial_e3;
    A1Gemma_est3:=i0_Main_e3.inicial_e3;
    ReInte_est3_terminal:=i0_Main_e3.ReInte_e3_e3 ;


IF(i0_Main_e3.manual_e3) THEN
    i0_Main_e3.brazo_hacia_adelante_e3:=MaAvanzarBrazo_est3_terminal;
    i0_Main_e3.brazo_hacia_atras_e3:=MaRetrocederBrazo_est3_terminal;
    i0_Main_e3.culata_e3:=MaSacarTapa_est3_terminal;
    i0_Main_e3.enroscar_e3:= MaRoscar_est3_Terminal;
    i0_Main_e3.fijar_e3:= MaCerrarMordaza_est3_terminal;
    i0_Main_e3.pinza_e3:= MaCerrarPinza_est3_terminal;
    i0_Main_e3.pinza_sube_baja_e3:= MaBajarPinza_Est3_terminal;

    Cinta_avanza_est3:=i0_Main_e3.brazo_hacia_adelante_e3;
    Cinta_retrocede_est3:=i0_Main_e3.brazo_hacia_atras_e3;
    Culata_est3:=i0_Main_e3.culata_e3;
    Roscar_est3:= i0_Main_e3.enroscar_e3;
    Fijar_est3:= i0_Main_e3.fijar_e3;
    Pinza_est3:= i0_Main_e3.pinza_e3;
    Pinza_baja_est3:= i0_Main_e3.pinza_sube_baja_e3;
END_IF;
```

```
(*Entradas estacion 3*)

i0_Main_e3.brazo_adelante_e3:=Cinta_adelante_est3;
i0_Main_e3.brazo_atras_e3:=Cinta_atras_est3;
i0_Main_e3.detector_e3:=Cargador_est3;
i0_Main_e3.emerg_e3:=Emergencia;
i0_Main_e3.giro_drcha_e3:=Gira_dcha_est3;
i0_Main_e3.giro_izda_e3:=Gira_izda_est3;
i0_Main_e3.marcha_e3:=Marcha_est3;
i0_Main_e3.pinza_abajo_e3:=Pinza_abajo_est3;
i0_Main_e3.pinza_arriba_e3:=Pinza_arriba_est3;
i0_Main_e3.reset_e3:=Reset_est3;
i0_Main_e3.Palet_e3:=Palet_est3;
i0_Main_e3.desconectar_e3:=Desconectar_e3;
i0_Main_e3.TipoPieza_terminal_e3:=TipoPieza_Terminal_est1;

i0_Main_e3.automatico_terminal_e3:=automatico_est3_terminal;
i0_Main_e3.manual_terminal_e3:=Manual_est3_terminal;
i0_Main_e3.estacion30N:=Estacion30N;
i0_Main_e3.emerg_terminal_e3:=Emergencia_est3_terminal;
i0_Main_e3.marcha_terminal_e3:=Marcha_est3_terminal;
i0_Main_e3.rearme_terminal_e3:=Rearme_est3_terminal;
i0_Main_e3.test_terminal_e3:=Test_est3_terminal;
i0_Main_e3.Pinicial_terminal_e3:=Pinicial_est3_terminal;
i0_Main_e3.rearme_inte_terminal_e3:=RearmeInteligente_est3_terminal;
i0_Main_e3.auto_transporte_e3:=i0_Cinta0.automatico_terminal_e3;
i0_Main_e3.reposicionamiento:=reposicionamiento_est1;
```

Figura 179.Programación rutinas salidas/entradas

Rutina Actualizar_leer3

La rutina Actualizar_leer3 es la siguiente:

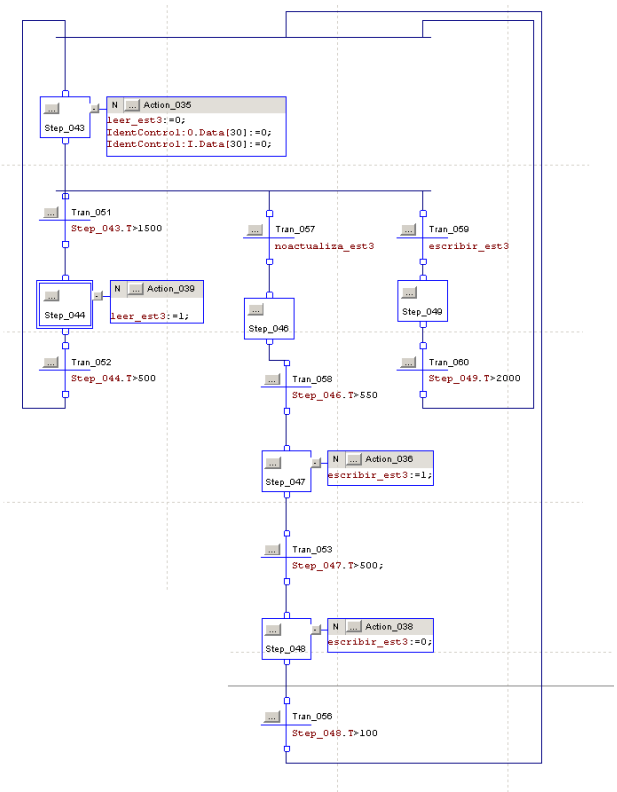


Figura 180.Rutina actualizar estación 3

La función principal de esta rutina será la lectura de la memoria incluida en el Palet de la estación 3. Se actualizará cada un segundo. Con esta rutina, también se ha podido conseguir hacer la escritura de la memoria del Palet mediante el botón (Escribir Palet3) incluido en la pantalla de la Magelis. La última función de esta rutina, será la escritura mediante programa.

Se activará al final de cada estación la variable noactualiza_est3 durante un periodo superior a 500ms, consiguiendo que se ejecute la operación de escritura en la memoria del Palet.

Rutina Identificación

Como ya se ha ido explicando durante el capítulo 3, el modulo del identificador de producto, el cual será también se programara en Simulink, pero habrá que hacer pequeñas modificaciones (rutina identificación) a la hora de introducirlo en el programa RSlogix 5000, ya que las variables que se utilizan para la escritura y lectura de los palets no son accesibles desde Simulink.

Por lo tanto, se han utilizado variables auxiliares en Simulink las cuales se modificaran dependiendo de las lecturas del identificador de producto y, viceversa, habrá otras variables que ocasionaran modificaciones en las variables de escritura del identificador.

Para la estación 3 habrá que realizar varios bloques if los cuales modificaran los valores de las variables auxiliares y de las variables del identificador. Son los siguientes:

```
(*estacion 3*)
if i0_Main_e3.is_funcionamiento_automatico_e3 = 9 then
    if i0_Main_e3.actualizacion_e3 then
    datawritehead3[0]:=datareadhead3[0];
    datawritehead3[1]:=datareadhead3[1];
    datawritehead3[2]:=datareadhead3[2];
    datawritehead3[3]:=9;
    datawritehead3[4]:=datareadhead3[4];
    i0_Main_e3.p1:=1;
    end_if;
end_if;

if i0_Main_e3.is_funcionamiento_automatico_e3 = 18 then
    if (datareadhead3[0]=1 or datareadhead3[0]=2 or datareadhead3[0]=3) and(datareadhead3[1]=1 or datareadhead3[1]=2) and datareadhead3[3]=0 then
    i0_Main_e3.comprobacion_e3:=1;
    end_if;
end_if;

if i0_Main_e3.is_funcionamiento_automatico_e3 = 18 then
    if datareadhead3[0]>=4 and datareadhead3[0]<=6 then
    i0_Main_e3.pieza_con_tapa_e3:=1;
    end_if;
end_if;

if i0_Main_e3.is_funcionamiento_automatico_e3 = 16 then
    if datareadhead3[3]=1 then
    i0_Main_e3.comprobacion_salida_e3:=1;
    end_if;
end_if;

if i0_Main_e3.is_funcionamiento_automatico_e3 = 12 then
    if i0_Main_e3.fallo_lectura_e3 then
    datareadhead3[3]:=9;
    i0_Main_e3.p0:=1;
    end_if;
end_if;
if i0_Main_e3.is_funcionamiento_automatico_e3 = 14 then
    if i0_Main_e3.escribir_e3 THEN
    datawritehead3[0]:=datareadhead3[0];
    datawritehead3[1]:=datareadhead3[1];
    datawritehead3[2]:=datareadhead3[2];
    datawritehead3[3]:=1;
    datawritehead3[4]:=datareadhead3[4];
    i0_Main_e3.p2:=1;
    end_if;
end_if;
```

Figura 181.Programación rutina identificación para estación 3

Como se puede ver, al igual que en la estación 1, delante de cada una de los bloques if otro bloque if, el cual comprueba en la programación Main_e3 la variable io_Main_e3.is_ce_Main_e3, la cual indica en qué estado del proceso se encuentra.

Supervisión mediante Magelis.

En este punto hablaremos de la supervisión de la estación mediante una Magelis XBTGT4330. El programa ha sido creado en Vijeo Designer y posteriormente descargado en la Magelis. A continuación describiremos las distintas pantallas creadas en la Magelis.

-Pantalla Menú:

Lo primero que hemos creado ha sido un menú, que nos llevará a las distintas pantallas del programa. Desde el Menú puedes acceder al control de la estación, al funcionamiento de la estación en tiempo real, a la guía gemma, al identificador de productos y finalmente a las alarmas, como se muestra en la siguiente figura:

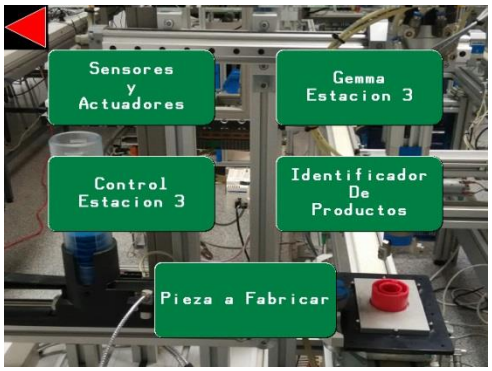


Figura 182.Pantalla menú Estación 3

-Pantalla Sensores y Actuadores.

En esta pantalla se puede ver una imagen de la estación 3, que cambiará en tiempo real dependiendo de las diferentes posiciones de la estación. Esta pantalla también incorpora unos pilotos que indican si los actuadores o los sensores están activados o desactivados. Incluye también una imagen actualizada de la pieza que hay actualmente en el Palet. Y por último, incorpora un piloto de emergencia, que se cambia a color rojo si ocurre alguna emergencia.

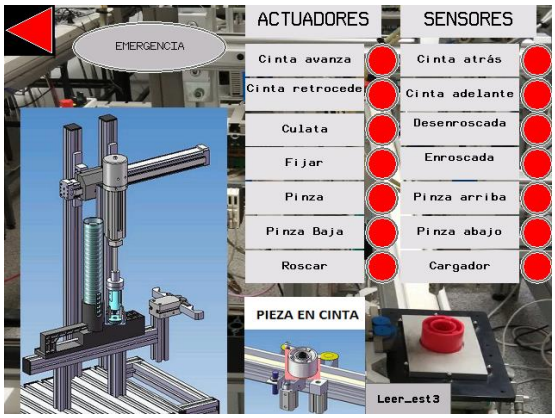


Figura 183.Pantalla Sensores y Actuadores estación 3

-Pantalla Control Estación 3

En esta pantalla se puede ver diferentes botones para controlar la estación en todo momento. En esta pantalla se podrá cambiar entre los diferentes modos de funcionamiento, ya sea modo automático, modo manual, o test. Solo está creada la pantalla y las variables, pero solo está

programado el modo automático, ya que los demás modos no entraban dentro de los objetivos de este trabajo. Además de eso incorpora también los sensores y actuadores que aparecen en la pantalla anterior.

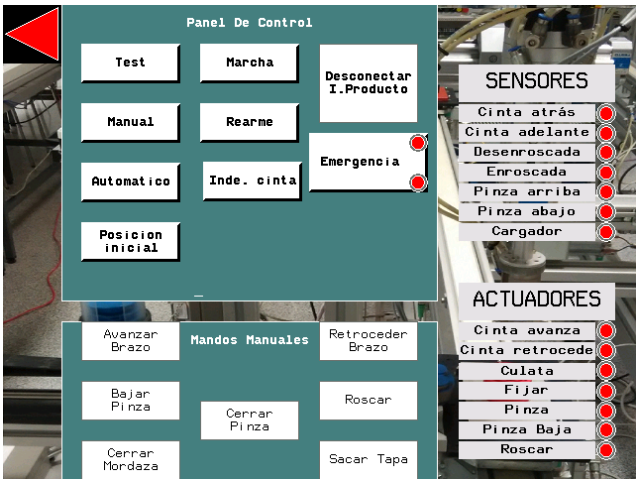


Figura 184.Pantalla de control de la estación 3

-Pantalla guía gemma

Esta pantalla indicaría el funcionamiento global del programa controlado por la pantalla anterior, al igual que la pantalla anterior, esto tampoco está implementado en nuestro programa.

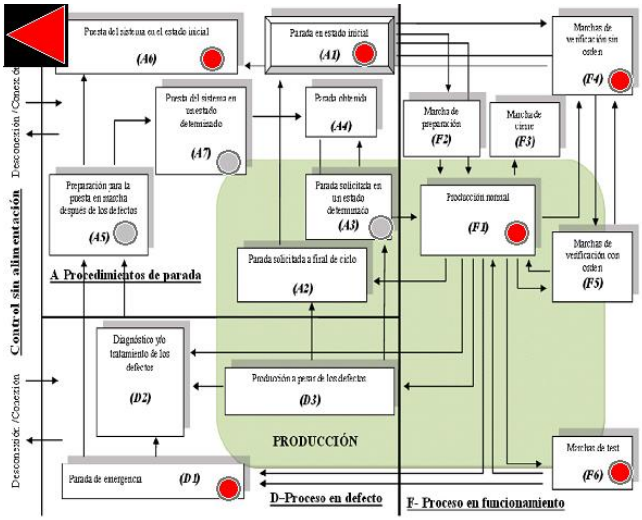


Figura 185.Pantalla Guía Gemma estación

-Pantalla Identificador de productos

Con esta pantalla conseguimos leer y escribir mediante RFID en la memoria incluida en el palet. Está compuesta de 3 botones (leer Palet3, escribir Palet3 y borrar Palet3).

- Leer Palet3→ Con este botón se lee la memoria del Palet 3, descargando sus datos en el vector DataReadHead3, que es mostrado en la parte derecha de la pantalla.
- Escribir Palet3→ Con este botón escribimos en la memoria los datos que estén escritos en el vector DataWriteHead3, este vector se puede modificar, ya que está definido en la parte central de la pantalla.

- Borrar Palet3→ Con este botón cambiamos todos los valores del vector DaraWriteHead3 a 0, y después se activa el botón Escribir Palet 3 automáticamente, consiguiendo así, poner a 0 los 9 primeros valores de la memoria por RFID.

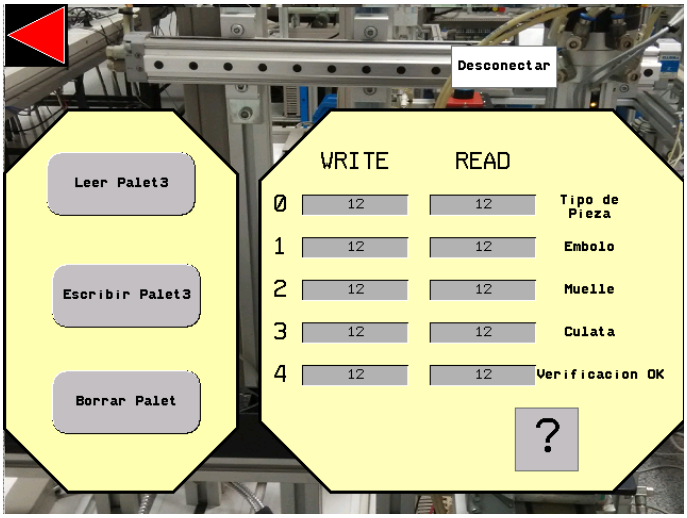


Figura 186.Pantalla Identificador de Productos estación 3

Esta pantalla también cuenta con dos ventanas emergentes, la primera solo salta si se intenta activar leer Palet3 y escribir Palet3 al mismo tiempo. Esto es debido a que no puedes escribir y leer al mismo tiempo, solo se puede realizar una acción al mismo tiempo. La segunda es informativa, y salta al pulsar el interrogante, informando del significado de los diferentes números que pudieran aparecer en la memoria del palet.

Además de estas operaciones, a la derecha de la imagen, aparece el vector DataReadHead3, que informa del contenido del Palet, ya que es la lectura de su memoria. La base de datos utilizada como podemos ver en la ventana emergente es la siguiente:

Posición 0	**Tipo de Pieza**	
	Negra=1	Negra con Tapa=4
	Pieza Rosa=2	Rosa con Tapa=5
	Pieza Metálica=3	Metálica con Tapa=6
	Fallo Pieza=9	
Posición 1	**Embolo**	
	Embolo Tipo1=1	Embolo Tipo2=2
	Fallo Embolo=9	
Posición 2	**Muelle**	
	Muelle estándar=1	Fallo Muelle=9
Posición 3	**Culata**	
	Culata=1	Fallo Culata=9
Posición 4	**Verificación OK**	
	Estación 4 Ok=1	Fallo Verificación=9

Figura 187.Ventana emergente base de datos de la memoria

- **Pantalla de Fabricación**

Mediante esta pantalla elegiremos el tipo de pieza a fabricar. Una vez seleccionada la pieza a fabricar, deberemos pulsar el botón de marcha de la botonera, o el propio botón de marcha

disponible en esta misma pantalla. También incorpora un botón “Fabricar Todas Las Piezas” que hace que salgan todas las piezas, sin desechar ninguna.

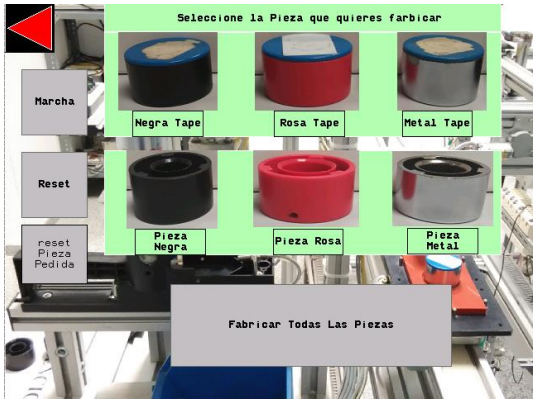


Figura 188.Pantalla de fabricación.

Estación 4

Después de añadir las modificaciones realizadas a la hora de introducir los otros modos de funcionamiento, el identificador de producto y la comunicación con el transporte, se realizaron modificaciones al modo automático que se explican a continuación además de explicar el resto del diagrama de estado y los modos de funcionamiento

Modificaciones al modo automático

Se ha añadido un estado anterior al inicio en el cual hacemos la lectura del palet y realizamos una comprobación.
En primer lugar, como en esta estación solo habría que realizar la producción si la pieza no tiene tapa, comprobamos si la primera posición del vector de lectura del identificados tiene valor entre 1 y3 si es el caso se realiza la producción normal. Si el valor esta entre 4 y 6 se finalizaría el proceso y se dejaría partir al palet. Y si por cualquier motivo no se ha podido realizar la lectura después de 15 segundos se iría a un estado en el cual se escribirá un código de error (un 9) en la posición 4 del vector de lectura

Tras añadir los modos de funcionamiento que son los mismos que para el resto de estaciones, la estación quedaría del siguiente modo:

El primer paso fue crear la rutina principal de nuestro programa, con la cual controlaríamos el control de nuestra estación.

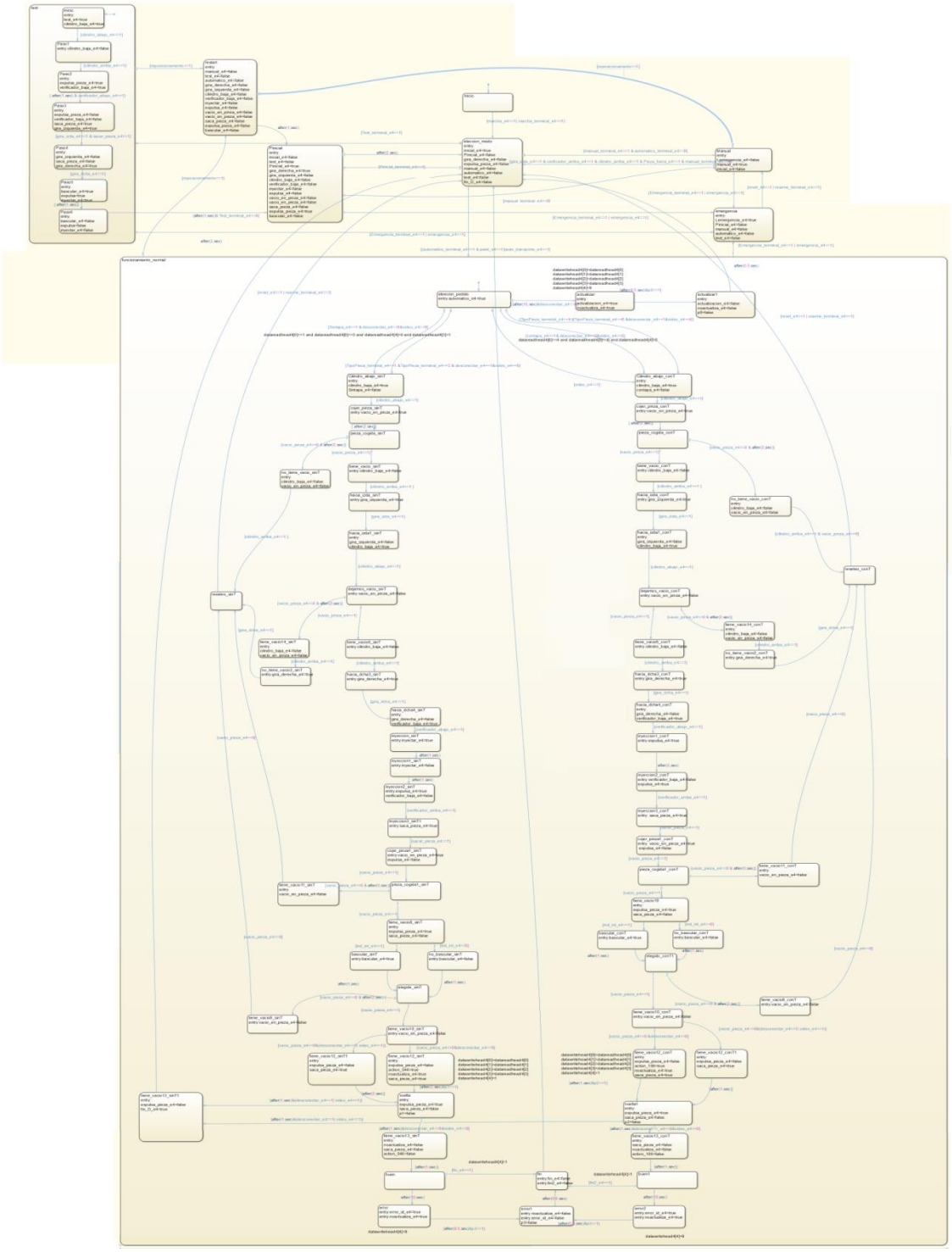


Figura 189.Esquema Simulink Estación 4

Una vez hecha la implementación se ha generado código para Rockwell con PLC Coder, el cual estará programado en la rutina Main_e3. A continuación, se pasará a la programación de Rockwell

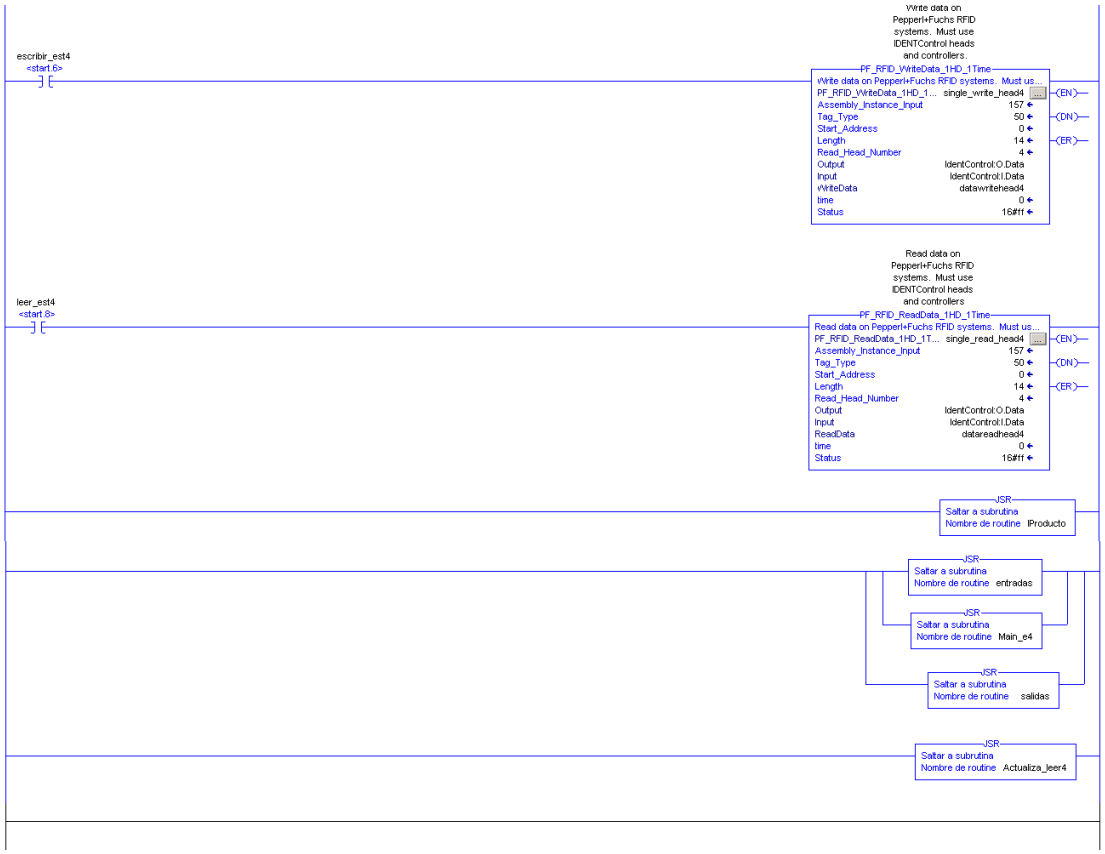


Figura 190.Programa Principal Estación 4

Como se puede observar en esta rutina principal, se lee la memoria de las estaciones mediante la activación del bit leer_est4, con esto se activa la instrucción Read Data, que lee la memoria del palet de cada estación y descarga su contenido en el vector DataReadHead4. Lo mismo para escribir, al activar escribir_est4, y esto activa la instrucción Write Data. Con esto, se escribe en la memoria del palet el contenido del vector DataWriteHead4.

Por último, en esta rutina principal, se puede ver el salto a otras 5 rutinas, que estarán siempre activas. La primera rutina (**Main_e4**), contiene el funcionamiento completo de la estación 4 la cual se ha implementado en Simulink, que se describirá a continuación, la siguiente rutina **entradas,** continua con **Actulizar_leer4**, que será la encargada de la lectura y escritura del identificador de producto, la rutina **salidas** y por último, la rutina **identificación**.

Rutinas para estación 4

Una vez que se ha realizado todo esto, En Simulink se ha generado el código de la máquina de estados de la estación 4, y posteriormente en el programa RSLogix 5000 se ha importado como rutina.

Main_e4

Esta rutina es la que se ha importado desde Simulink, se partirá desde la máquina de estado completa de Simulink, se importará el código generado.

```
IF (i0_Main_e4.automatico_terminal_e4 AND i0_Main_e4.palet_e4) OR i0_Main_e4.auto_transporte_e4 THEN
    (* Transition: '<S1>:187' *)
    i0_Main_e4.is_c3_Main_e4 := 6;
    (* Entry Internal 'funcionamiento_normal': '<S1>:410' *)
    (* Transition: '<S1>:650' *)
    i0_Main_e4.is_funcionamiento_normal := 15;
    PLC_CODER_TIMER(i0_Main_e4.temporalCounter_i1, 1, 0, i0_Main_e4.temp1);
    (* Outport: '<Root>/automatico_e4' *)
    (* Entry 'eleecion_pedido': '<S1>:649' *)
    i0_Main_e4.automatico_e4 := 1;
ELSIF i0_Main_e4.Pinicial_terminal_e4 THEN
    (* Transition: '<S1>:664' *)
    i0_Main_e4.is_c3_Main_e4 := 3;
    PLC_CODER_TIMER(i0_Main_e4.temporalCounter_i1, 1, 0, i0_Main_e4.temp1);
    (* Outport: '<Root>/inicial_e4' *)
    (* Entry 'Pinicial': '<S1>:557' *)
    i0_Main_e4.inicial_e4 := 0;
    (* Outport: '<Root>/test_e4' *)
    i0_Main_e4.test_e4 := 0;
    (* Outport: '<Root>/Pinicial_e4' *)
    i0_Main_e4.Pinicial_e4 := 1;
    (* Outport: '<Root>/gira_derecha_e4' *)
    i0_Main_e4.gira_derecha_e4 := 1;
    (* Outport: '<Root>/gira_izquierda_e4' *)
    i0_Main_e4.gira_izquierda_e4 := 0;
    (* Outport: '<Root>/cilindro_baja_e4' *)
    i0_Main_e4.cilindro_baja_e4 := 0;
    (* Outport: '<Root>/verificador_baja_e4' *)
    i0_Main_e4.verificador_baja_e4 := 0;
    (* Outport: '<Root>/inyectar_e4' *)
    i0_Main_e4.inyectar_e4 := 0;
    (* Outport: '<Root>/expulsa_e4' *)
    i0_Main_e4.expulsa_e4 := 0;
    (* Outport: '<Root>/vacio_en_pinza_e4' *)
    i0_Main_e4.vacio_en_pinza_e4 := 0;
    (* Outport: '<Root>/vacio_en_pieza_e4' *)
    i0_Main_e4.vacio_en_pieza_e4 := 0;
    (* Outport: '<Root>/saca_pieza_e4' *)
    i0_Main_e4.saca_pieza_e4 := 0;
    (* Outport: '<Root>/expulsa_pieza_e4' *)
    i0_Main_e4.expulsa_pieza_e4 := 1;
    (* Outport: '<Root>/bascular_e4' *)
    i0_Main_e4.bascular_e4 := 0;
ELSIF i0_Main_e4.manual_terminal_e4 AND ( NOT i0_Main_e4.automatico_terminal_e4) THEN
```

Figura 191.Rutina Main_e4

Salidas/Entradas

Al realizar la exportación desde Simulink y la importación en RSLogix, como ya se ha explicado se crea una variable vectorial cuyo nombre es i0_nombre_chart (para la estación 4 será i0_Main_e4) y las variables de Simulink serán distintas posiciones de ese vector creado en la importación. Por ejemplo: i0_Main_e4.palet_e4.

Por lo tanto, hay que coger esas variables de entrada y salida de Simulink e igualarlas a las variables de entrada y salida del RSLogix para poder mover la estación. Un ejemplo de entrada seria: i0_Main_e4.palet_e4:=Palet_est4;

Y otro de salida seria: Vacio_en_pieza_est4:=i0_Main_e4.vacio_en_pieza_e4;

```
(*Salidas estacion 4*)

Cilindro_baja_est4:=i0_Main_e4.cilindro_baja_e4;
Expulsa_pieza_est4:=i0_Main_e4.expulsa_pieza_e4;
Verificador_baja_est4:=i0_Main_e4.verificador_baja_e4;
Inyecta_est4:=i0_Main_e4.inyectar_e4;
Expulsa_est4:=i0_Main_e4.expulsa_e4;
Vacio_en_pieza_est4:=i0_Main_e4.vacio_en_pieza_e4;
Saca_pieza_est4:=i0_Main_e4.saca_pieza_e4;
Gira_izquierda_est4:=i0_Main_e4.gira_izquierda_e4;
Bascular_est4:=i0_Main_e4.bascular_e4;
Gira_derecha_est4:=i0_Main_e4.gira_derecha_e4;
Vacio_en_pinza_est4:=i0_Main_e4.vacio_en_pinza_e4;
F4Gemma_est4:=i0_Main_e4.manual_e4;
F1Gemma_est4:=i0_Main_e4.automatico_e4;
A1Gemma_est4:=i0_Main_e4.inicial_e4;
A6Gemma_est4:=i0_Main_e4.Pinicial_e4;
F6Gemma_est4:=i0_Main_e4.test_e4;
D1Gemma_est4:=i0_Main_e4.Lemergencia_e4;


IF(i0_Main_e4.manual_e4) THEN
i0_Main_e4.cilindro_baja_e4:=MaCilindro_Baja_est4_terminal;
i0_Main_e4.expulsa_pieza_e4:=MaExpulsa_pieza_est4_terminal;
i0_Main_e4.verificador_baja_e4:=MaVerificador_baja_est4_terminal;
i0_Main_e4.inyectar_e4:=MaInyecta_est4_terminal;
i0_Main_e4.expulsa_e4:=MaExpulsa_est4_terminal;
i0_Main_e4.vacio_en_pieza_e4:=MaVacio_en_Pieza_est4_terminal;
i0_Main_e4.saca_pieza_e4:=MaSaca_Pieza_est4_terminal;
i0_Main_e4.gira_izquierda_e4:=MaGira_Izquierda_est4_terminal;
i0_Main_e4.bascular_e4:=MaBascular_est4_terminal;
i0_Main_e4.gira_derecha_e4:=MaGira_Derecha_est4_terminal;
i0_Main_e4.vacio_en_pinza_e4:=MaVacio_en_Pinza_est4_terminal;
```

```
(*Entradas estacion 4*)
i0_Main_e4.rearme_terminal_e4:=Rearme_est4_terminal;
i0_Main_e4.marcha_e4:=Marcha_est4;
i0_Main_e4.cilindro_arriba_e4:=Cilindro_arriba_est4;
i0_Main_e4.gira_dcha_e4:=Gira_dcha_est4;
i0_Main_e4.verificador_arriba_e4:=Cilindro_arriba_est4;
i0_Main_e4.vacio_pinza_e4:=Vacio_pinza_est4;
i0_Main_e4.cilindro_abajo_e4:=Cilindro_abajo_est4;
i0_Main_e4.gira_izda_e4:=Gira_izda_est4;
i0_Main_e4.verificador_abajo_e4:=Verificador_abajo_est4;
i0_Main_e4.sacar_pieza_e4:=Sacar_pieza_est4;
i0_Main_e4.pieza_fuera_e4:=Pieza_fuera_est4;
i0_Main_e4.ind_int_e4:=Ind_Int_est4;
i0_Main_e4.reset_e4:=Reset_est4;
i0_Main_e4.TipoPieza_terminal_e4:=TipoPieza_Terminal_est1;
i0_Main_e4.vacio_pieza_e4:=Vacio_pieza_est4;
i0_Main_e4.desconectar_e4:=Desconectar_e4;
i0_Main_e4.Emergencia_e4:=Emergencia4;
i0_Main_e4.marcha_terminal_e4:=marcha_est4_terminal;
i0_Main_e4.manual_terminal_e4:=Manual_est4_terminal;
i0_Main_e4.Emergencia_terminal_e4:=Emergencia_est4_terminal;
i0_Main_e4.automatico_terminal_e4:=automatico_est4_terminal;
i0_Main_e4.Pinicial_terminal_e4:=Pinicial_est4_terminal;
i0_Main_e4.Test_terminal_e4:=Test_est4_terminal;
i0_Main_e4.Palet_e4:=Palet_est4;
i0_Main_e4.auto_transporte_e4:=i0_Cinta0.automatico_terminal_e4;
i0_Main_e4.desconectar_e4:=Desconectar_e4;
i0_Main_e4.reposicionamento:=reposicionamiento_est1;
```

Figura 192.Programación rutinas salidas/entradas

Rutina Actualizar_leer4

La rutina Actualizar_leer4 es la siguiente:

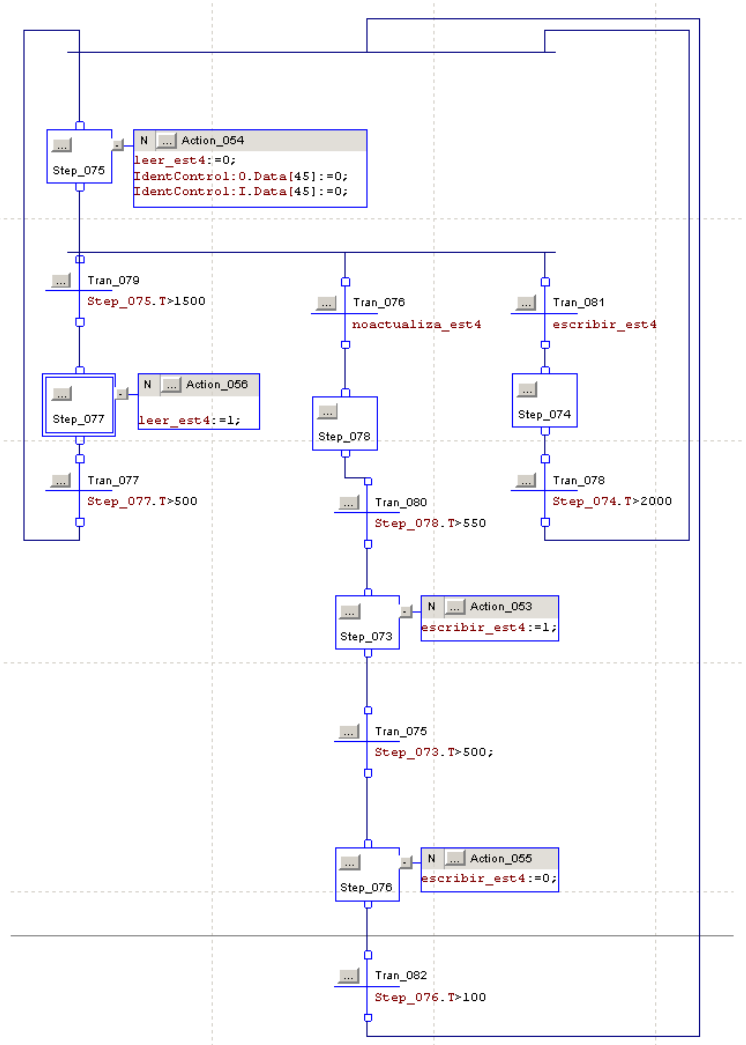


Figura 193.Rutina actualizar Estación 4

La función principal de esta rutina será la lectura de la memoria incluida en el Palet de la estación 4. Se actualizará cada segundo y medio. Con esta rutina, también se ha podido conseguir hacer la escritura de la memoria del Palet mediante el botón (Escribir Palet4) incluido en la pantalla de la Magelis. La última función de esta rutina, será la escritura mediante programa.

Se activará al final de cada estación la variable noactualiza_est4 durante un periodo superior a 500ms, consiguiendo que se ejecute la operación de escritura en la memoria del Palet.

Rutina Identificación

Como ya se ha ido explicando durante el capítulo 3, el modulo del identificador de producto, el cual será también se programara en Simulink, pero habrá que hacer pequeñas modificaciones (rutina identificación) a la hora de introducirlo en el programa RSlogix 5000, ya que las variables que se utilizan para la escritura y lectura de los palets no son accesibles desde Simulink.

Por lo tanto, se han utilizado variables auxiliares en Simulink las cuales se modificaran dependiendo de las lecturas del identificador de producto y, viceversa, habrá otras variables que ocasionaran modificaciones en las variables de escritura del identificador.

Para la estación 4 habrá que realizar varios bloques if los cuales modificaran los valores de las variables auxiliares y de las variables del identificador. Son los siguientes:

```
(*estacion 4*)
if i0_Main_e4.is_funcionamiento_normal = 3 then
    if i0_Main_e4.actualizacion_e4  then
    datawritehead4[0]:=datareadhead4[0];
    datawritehead4[1]:=datareadhead4[1];
    datawritehead4[2]:=datareadhead4[2];
    datawritehead4[3]:=datareadhead4[3];
    datawritehead4[4]:=9;
    i0_Main_e4.p0:=1;
    end_if;
end_if;

if i0_Main_e4.is_funcionamiento_normal = 15 then
    if datareadhead4[0]>=1 and datareadhead4[0]<=3 and datareadhead4[4]=0 and datareadhead4[3]=1 then
    i0_Main_e4.Sintapa_e4:=1;
    end_if;

end_if;

if i0_Main_e4.is_funcionamiento_normal = 15 then
    if datareadhead4[0]>=4 and datareadhead4[0]<=6 (*and datareadhead4[4]=0 *)then
    i0_Main_e4.contapa_e4:=1;
    end_if;
end_if;

if i0_Main_e4.is_funcionamiento_normal = 55 then
    if i0_Main_e4.action_046 then
    datawritehead4[0]:=datareadhead4[0];
    datawritehead4[1]:=datareadhead4[1];
    datawritehead4[2]:=datareadhead4[2];
    datawritehead4[3]:=datareadhead4[3];
    datawritehead4[4]:=1;
    i0_Main_e4.p1:=1;
    end_if;
end_if;

if i0_Main_e4.is_funcionamiento_normal = 53 then
    if i0_Main_e4.action_106 then
    datawritehead4[0]:=datareadhead4[0];
    datawritehead4[1]:=datareadhead4[1];
    datawritehead4[2]:=datareadhead4[2];
    datawritehead4[3]:=datareadhead4[3];
    datawritehead4[4]:=1;
    i0_Main_e4.p2:=1;
    end_if;
end_if;

 if i0_Main_e4.is_funcionamiento_normal = 18 or i0_Main_e4.is_funcionamiento_normal = 20 then
     if i0_Main_e4.error_id_e4 then
     datawritehead4[4]:=9;
     i0_Main_e4.p3:=1;
     end_if;
 end_if;

 if i0_Main_e4.is_funcionamiento_normal = 7 then
     if datawritehead4[4]=1 then
     i0_Main_e4.fin_e4:=1;
     end_if;
 end_if;

 if i0_Main_e4.is_funcionamiento_normal = 8 then
     if datawritehead4[4]=1 then
     i0_Main_e4.fin2_e4:=1;
     end_if;
 end_if;
```

Figura 194.Programación rutina identificación para estación 4

Como se puede ver, al igual que en la estación 4, delante de cada una de los bloques if otro bloque if, el cual comprueba en la programación la variable io_Main_e3.is_ce_Main_e3, la cual indica en qué estado del proceso se encuentra.

Supervisión mediante Magelis.

En este punto hablaremos de la supervisión de la estación mediante una Magelis XBTGT4330. El programa ha sido creado en Vijeo Designer y posteriormente descargado en la Magelis. A continuación describiremos las distintas pantallas creadas en la Magelis.

- **Pantalla Menú:**

Lo primero que hemos creado ha sido un menú, que nos llevará a las distintas pantallas del programa. Desde el Menú puedes acceder al control de la estación, al funcionamiento de la estación en tiempo real, a la guía gemma, al identificador de productos y finalmente a las alarmas, como se muestra en la siguiente figura:

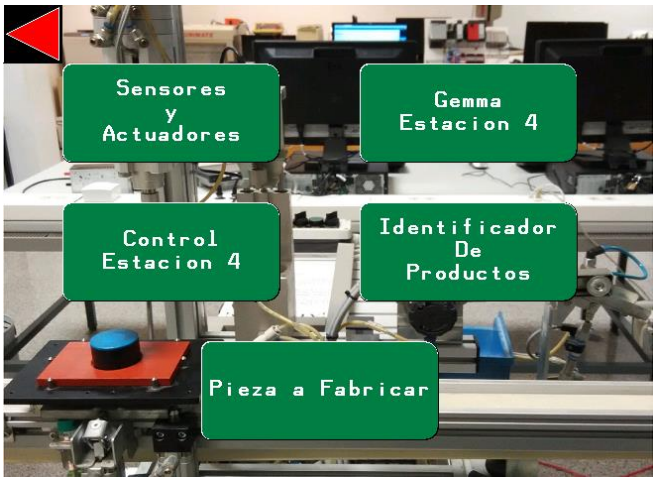


Figura 195.Pantalla menú Estación 4

-Pantalla Sensores y Actuadores.

En esta pantalla se puede ver una imagen de la estación 4, que cambiará en tiempo real dependiendo de las diferentes posiciones de la estación. Esta pantalla también incorpora unos pilotos que indican si los actuadores o los sensores están activados o desactivados. Además de todo esto incorpora un piloto de emergencia, que se cambia a color rojo si ocurre alguna emergencia.

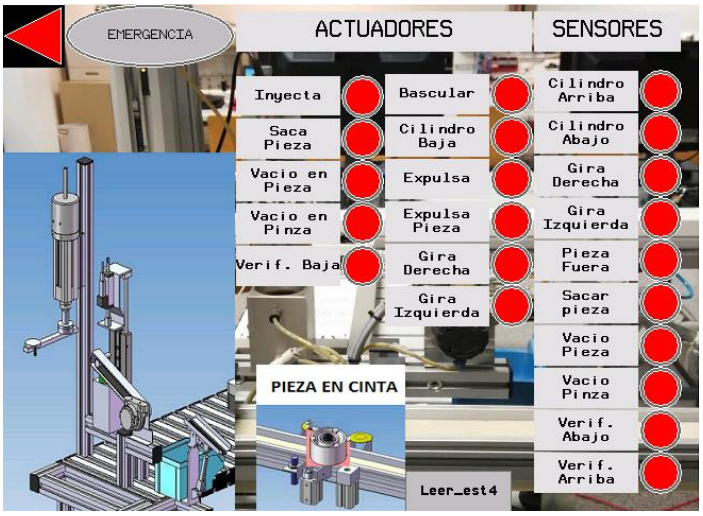


Figura 196.Pantalla Sensores y Actuadores estación 4

-Pantalla Control Estación 4

En esta pantalla se puede ver diferentes botones para controlar la estación en todo momento, en esta pantalla se podrá cambiar entre los diferentes modos de funcionamiento, ya sea modo automático, modo manual, o test. Solo está creada la pantalla y las variables, pero solo está programado el modo automático, ya que los demás modos no entraban dentro de los objetivos de este trabajo. Además de eso incorpora también los sensores y actuadores que aparecen en la pantalla anterior.

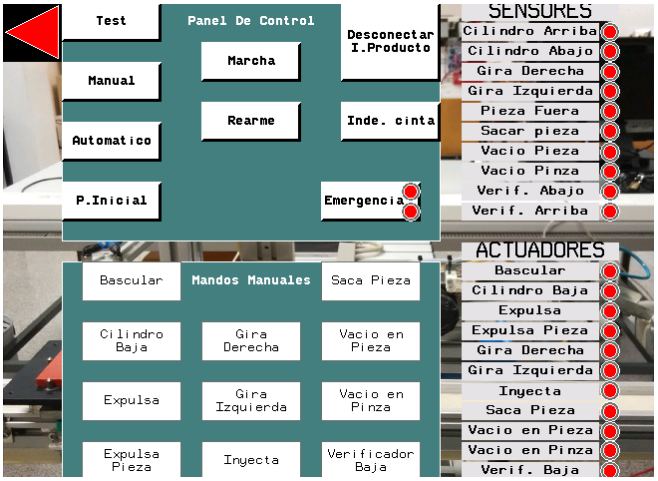


Figura 197.Pantalla de control de la estación 4

-Pantalla guía gemma

Esta pantalla indicaría el funcionamiento global del programa controlado por la pantalla anterior, al igual que la pantalla anterior, esto tampoco está implementado en nuestro programa.

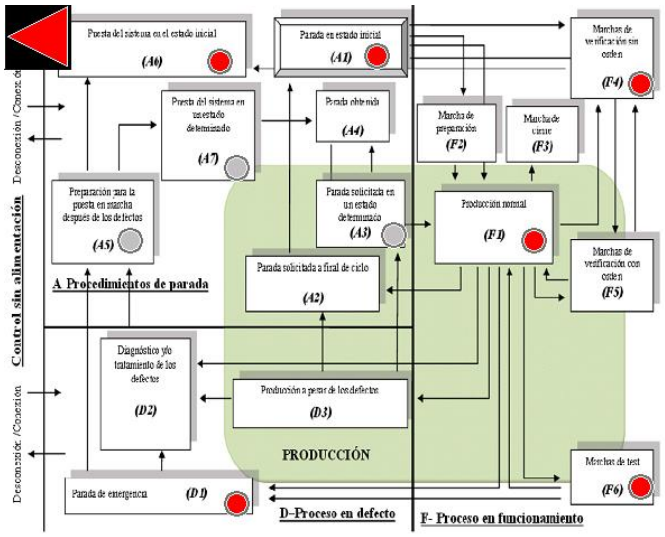


Figura 198. Pantalla Guía Gemma estación 4

-Pantalla Identificador de productos

Con esta pantalla conseguimos leer y escribir mediante RFID en la memoria incluida en el palet. Está compuesta de 3 botones (leer Palet4, escribir Palet4 y borrar Palet4).

- Leer Palet4→ Con este botón leemos la memoria del Palet 4, descargando sus datos en el vector DataReadHead4, que es mostrado en la parte derecha de la pantalla.
- Escribir Palet4→ Con este botón escribimos en la memoria los datos que estén escritos en el vector DataWriteHead4, este vector se puede modificar, ya que está definido en la parte central de la pantalla.
- Borrar Palet4→ Con este botón cambiamos todos los valores del vector DaraWriteHead4 a 0, y después se activa el botón Escribir Palet 4 automáticamente, consiguiendo así, poner a 0 los 9 primeros valores de la memoria por RFID.

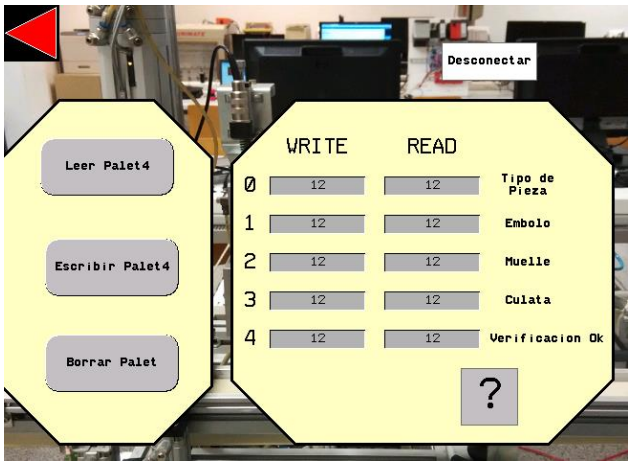
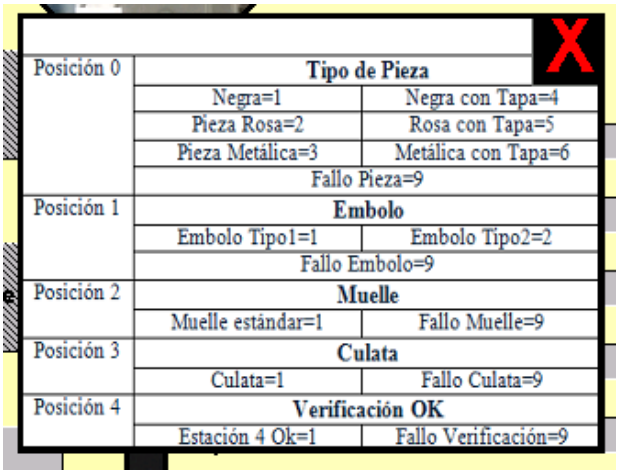


Figura 199.Pantalla identificador de productos estación 4

Esta pantalla también cuenta con dos ventanas emergentes, la primera solo salta si se intenta activar leer Palet4 y escribir Palet4 al mismo tiempo. Esto es debido a que no puedes escribir y leer al mismo tiempo, solo se puede realizar una acción al mismo tiempo. La segunda es

informativa, y salta al pulsar el interrogante, informando del significado de los diferentes números que pudieran aparecer en la memoria del palet.

Además de estas operaciones, a la derecha de la imagen, aparece el vector DataReadHead4, que informa del contenido del Palet, ya que es la lectura de su memoria. La base de datos utilizada como podemos ver en la ventana emergente es la siguiente:



Posición 0	**Tipo de Pieza**	
	Negra=1	Negra con Tapa=4
	Pieza Rosa=2	Rosa con Tapa=5
	Pieza Metálica=3	Metálica con Tapa=6
	Fallo Pieza=9	
Posición 1	**Embolo**	
	Embolo Tipo1=1	Embolo Tipo2=2
	Fallo Embolo=9	
Posición 2	**Muelle**	
	Muelle estándar=1	Fallo Muelle=9
Posición 3	**Culata**	
	Culata=1	Fallo Culata=9
Posición 4	**Verificación OK**	
	Estación 4 Ok=1	Fallo Verificación=9

Figura 200.Ventana emergente base de datos de la memoria

Cinta

Transporte

Para esta rutina se optó por hacer directamente el modelo final de la máquina de estados y contemplar con las variables desconectar_e que se pueda desconectar el uso del identificador.

Se mostrara una imagen de su programación por bloques y se describirá:

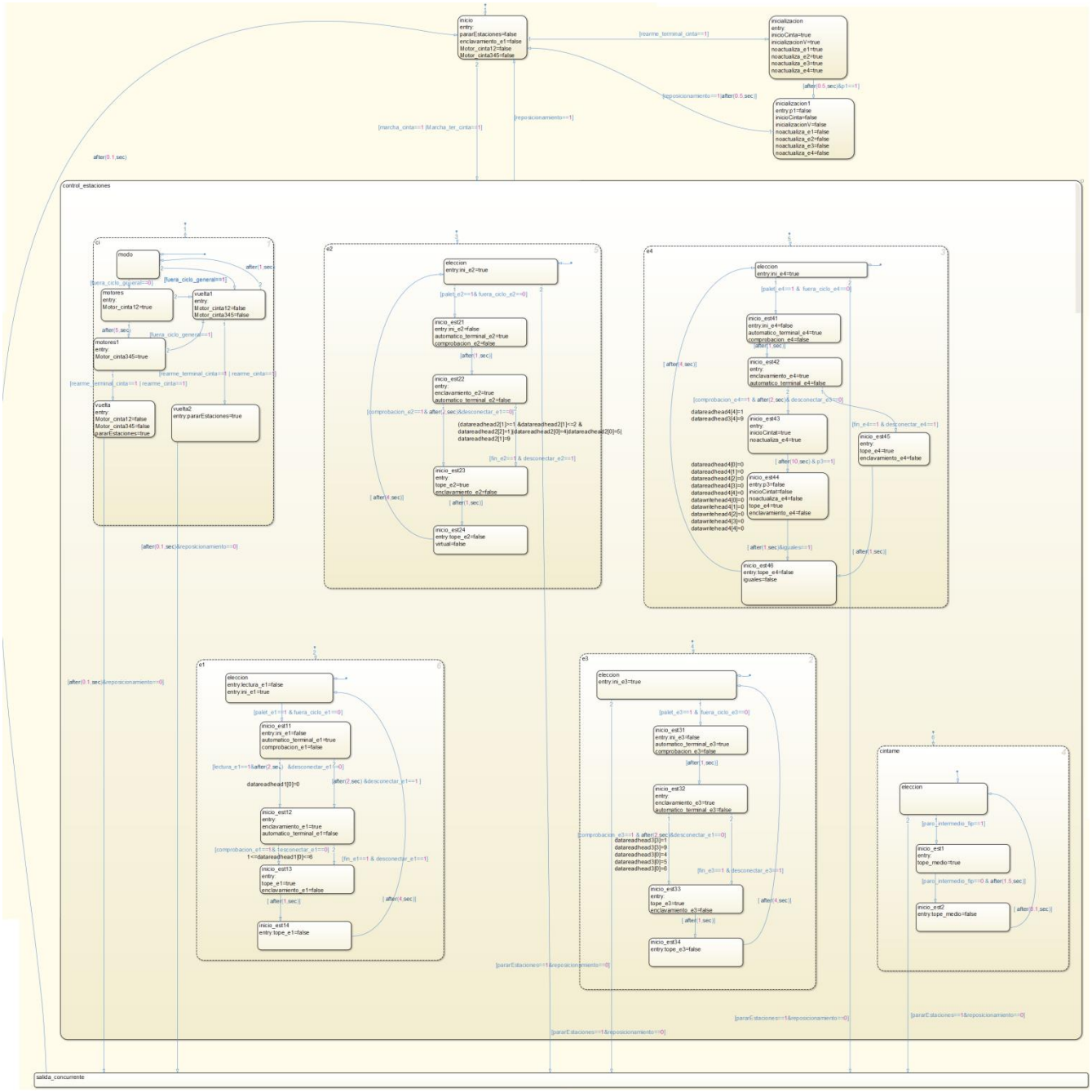


Figura 201.Máquina de estados completa transporte

- Incluye un estado inicial en el cuál la cinta de transporte está apagada. Al activar marcha (de la botonera o del terminal), se pone la cinta de transporte en funcionamiento. Además de la cinta, también se activa el paso de piezas intermedio y los bloques 1, 2, 3 y 4 que se explicará a continuación. Si se pulsa reset, la cinta se detendrá inmediatamente, y los bloques 1, 2, 3 y 4 terminarán la pieza que estén fabricando antes de detenerse.

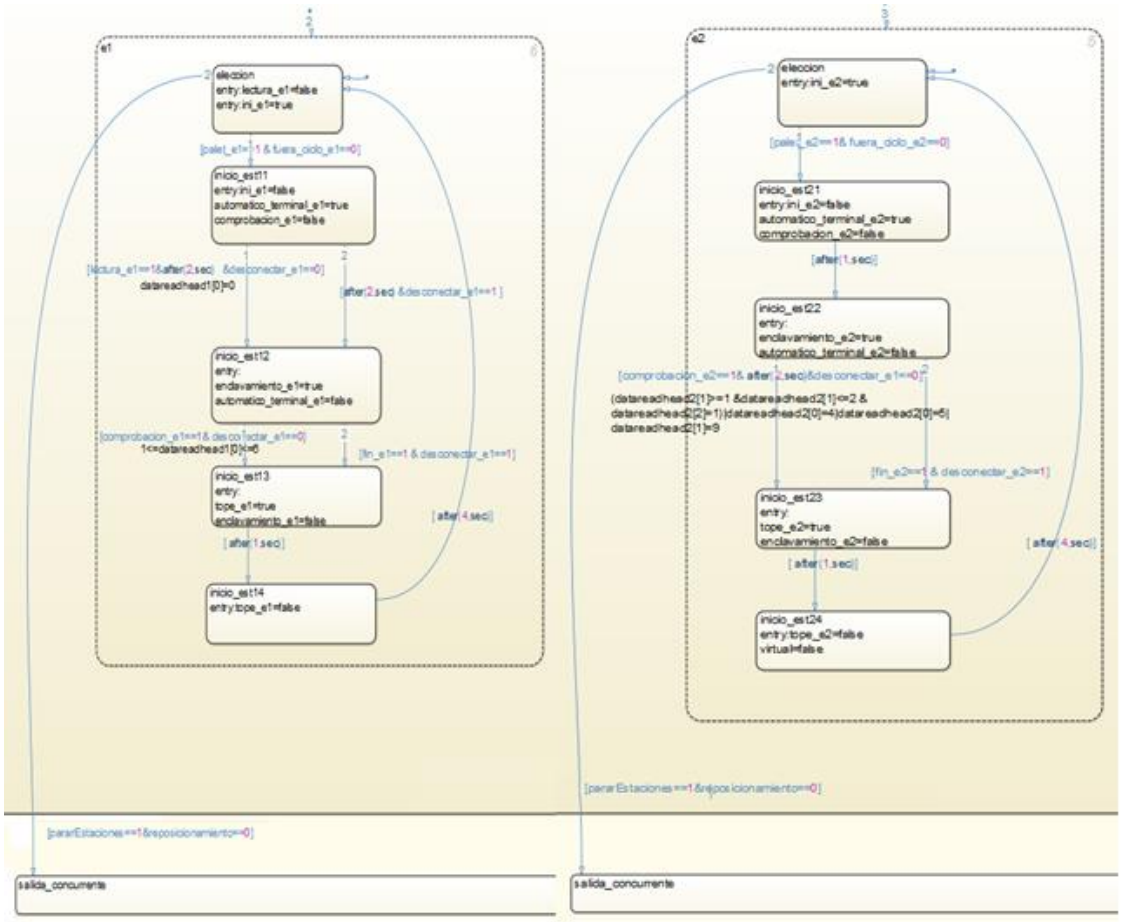


Figura 202.Rutina Cinta de transporte Estación 1 y 2

-Para la estación 1. En el momento en que se detecte un palet, comenzará la fabricación de una pieza. Se activa el enclavamiento para fijar el palet, y se mantiene activado hasta que coloque el cilindro deseado en el palet. Una vez colocado el cilindro, el palet continua hasta la siguiente estación, y esta queda disponible para la fabricación de la siguiente pieza.

Para la estación 2. En el momento en que se detecte un palet, comienza a funcionar la estación 2. Se activa el enclavamiento para fijar el palet, y se mantiene activado hasta que se escribe que el muelle y el embolo han sido colocados. Una vez realizada la operación, el palet continua hasta la siguiente estación, y esta queda disponible para la fabricación de la siguiente pieza. La operación también puede terminar si no se necesita hacer ninguna operación en la pieza, o si el palet está vacío.

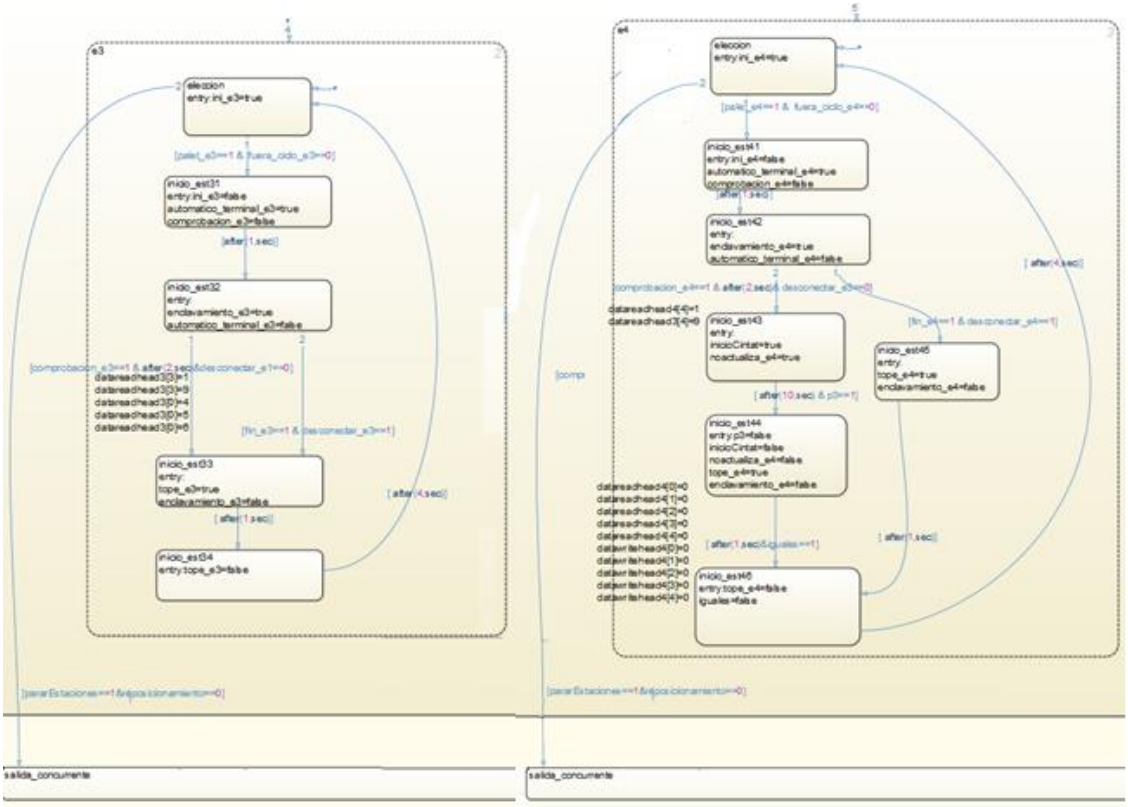


Figura 203. Rutina Cinta de transporte Estación 3 y 4

-Para la estación 3. En el momento en que se detecte un palet, comienza a funcionar la estación 3. Se activa el enclavamiento para fijar el palet, y se mantiene activado hasta que se coloca la culata. Una vez realizada la operación, el palet continua hasta la siguiente estación, y esta queda disponible para la fabricación de la siguiente pieza. La operación también puede terminar si no se necesita hacer ninguna operación en la pieza, o si el palet está vacío.

-Para la estación 4. En el momento en que se detecte un palet, comienza a funcionar la estación 4. Se activa el enclavamiento para fijar el palet, y se mantiene activado hasta que se realiza la verificación de la pieza. Una vez realizada la operación, el palet continua hasta la siguiente estación, y esta queda disponible para la fabricación de la siguiente pieza. La operación también puede terminar si no se necesita hacer ninguna operación en la pieza, o si el palet está vacío.

- Funcionamiento global de la célula

Como el funcionamiento individual de las estaciones ya está realizado, solo habrá que realizar una rutina que permita combinar la cinta de transporte con cada una de ellas. El objetivo es que cada estación lea el palet cuando sea detectado por el lector capacitivo que se encuentra en la cinta de transporte y mediante la lectura de su memoria, realizar las operaciones necesarias. Una vez terminado este proceso, el palet continua por la cinta de transporte hasta la siguiente estación.

Como se puede observar, la estación consta de tres partes principales:

- Cintas transportadoras: son las encargadas de realizar el transporte de los Palets de una estación a otra.
- Topes: Son los encargados de retener a los palets en las estaciones, ya que sin ellos, los palets nunca se quedarían en las estaciones.
- Lectores de Palets: Son los encargados de detectar los palets que pasan por las estaciones.

- Programación célula de fabricación

El programa ha sido realizado en Simulink, y se importará al igual que todas las rutinas empleadas en las estaciones anteriores, ya que en este apartado hay que crear una rutina principal que englobe a todas ellas. Se han añadido a las rutinas de las estaciones las variables, automatico_terminal_e1, automatico_terminal_e2, automatico_terminal_e3 y automatico_terminal_e4, y se han incluido en la transición en la que incorpora el botón de marcha, consiguiendo así que pueda funcionar automáticamente sin estar pulsando manualmente cada vez que la estación necesita realizar una operación.

Como se puede ver en esta rutina principal, se leerá la memoria mediante la activación de los bits leer_est1, leer_est2, leer_est3 y leer_est4. Con esto se activa la instrucción Read Data, que lee la memoria de los palets y descarga su contenido en los vectores correspondientes a cada palet. Lo mismo para escribir, al activar escribir_est1, escribir_est2, escribir_est3 y escribir_est4 y esto activa la instrucción Write Data. Con esto, se escribe en la memoria del palet el contenido del vector DataWriteHead1.

Por último, en esta rutina principal, se puede observar el salto a otras 12 rutinas, que estarán siempre activas. Cinco de ellas se corresponden con las rutinas de funcionamiento de las 4 estaciones y de la cinta de transporte. Las cuatro siguientes se corresponden con las rutinas de actualización de los 4 palets. Las restantes son rutinas utilizadas para adaptación entre variables (entradas, salidas, identificación).

Rutina cinta

Al igual que el resto de estaciones, el transporte también se ha implementado en Simulink y después se ha exportado para RSLogix.

A continuación, se mostrara una imagen de su programación:

```
CASE i0_Cinta0.is_ci OF
    1:
        (* During 'modo': '<S1>:153' *)
        IF  NOT i0_Cinta0.fuera_ciclo_general THEN
            (* Transition: '<S1>:28' *)
            i0_Cinta0.is_ci := 2;
            PLC_CODER_TIMER(i0_Cinta0.temporalCounter_i4, 1, 0, i0_Cinta0.temp1);
            (* Outport: '<Root>/Motor_cinta12' *)
            (* Entry 'motores': '<S1>:27' *)
            i0_Cinta0.Motor_cinta12 := 1;
        ELSE
            (* Transition: '<S1>:154' *)
            i0_Cinta0.is_ci := 5;
            PLC_CODER_TIMER(i0_Cinta0.temporalCounter_i4, 1, 0, i0_Cinta0.temp1);
            (* Outport: '<Root>/Motor_cinta12' *)
            (* Entry 'vueltal': '<S1>:164' *)
            i0_Cinta0.Motor_cinta12 := 0;
            (* Outport: '<Root>/Motor_cinta345' *)
            i0_Cinta0.Motor_cinta345 := 0;
        END_IF;
    2:
        (* During 'motores': '<S1>:27' *)
        PLC_CODER_TIMER(i0_Cinta0.temporalCounter_i4, 2, 5000, i0_Cinta0.temp1);
        IF i0_Cinta0.temp1 THEN
            (* Transition: '<S1>:194' *)
            i0_Cinta0.is_ci := 3;
            (* Outport: '<Root>/Motor_cinta345' *)
            (* Entry 'motoresl': '<S1>:193' *)
            i0_Cinta0.Motor_cinta345 := 1;
        ELSIF i0_Cinta0.fuera_ciclo_general THEN
            (* Transition: '<S1>:171' *)
            i0_Cinta0.is_ci := 5;
            PLC_CODER_TIMER(i0_Cinta0.temporalCounter_i4, 1, 0, i0_Cinta0.temp1);
            (* Outport: '<Root>/Motor_cinta12' *)
            (* Entry 'vueltal': '<S1>:164' *)
            i0_Cinta0.Motor_cinta12 := 0;
            (* Outport: '<Root>/Motor_cinta345' *)
            i0_Cinta0.Motor_cinta345 := 0;
        END_IF;
    3:
```

Figura 204.Codigo Programación cinta transporte

- Rutinas restantes

El resto de rutinas no serán descritas debido a que ya han sido explicadas con anterioridad en los puntos correspondientes a cada estación.

- Pantallas célula de fabricación

Las pantallas incluidas en la célula de fabricación, son las incluidas en las 4 estaciones. Adicionalmente a estas, se han añadido 3. Un menú creado para moverte entre los menús individuales de cada estación, y una pantalla de transporte, en la cual se muestra el contenido de los 4 palets y los sensores y actuadores de la cinta de transporte.

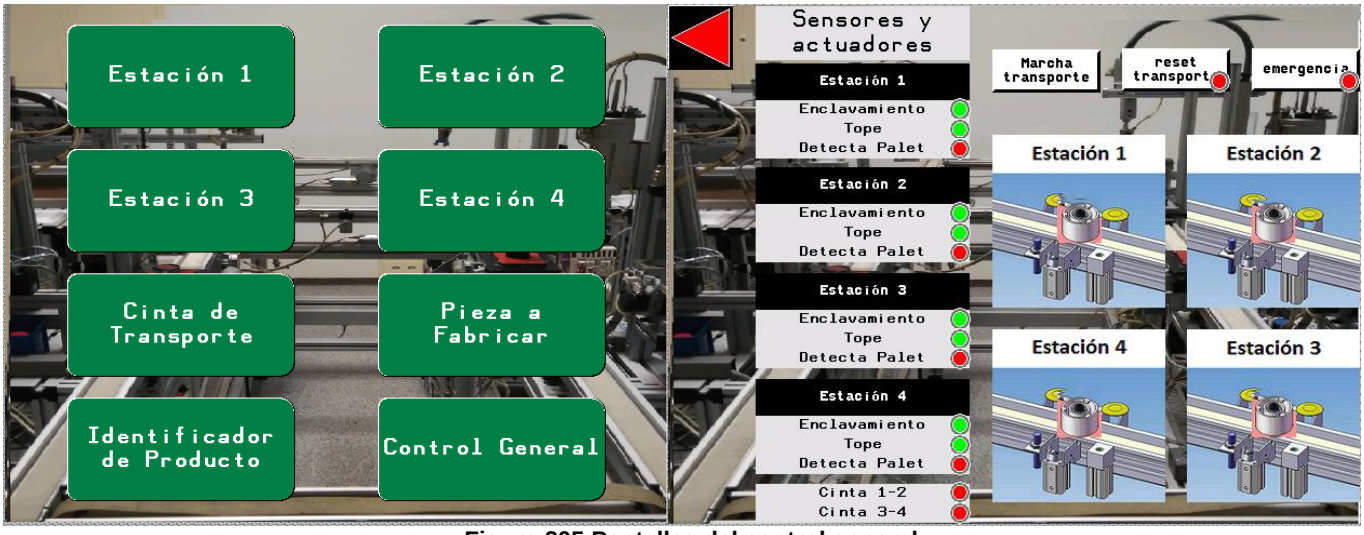


Figura 205.Pantallas del control general

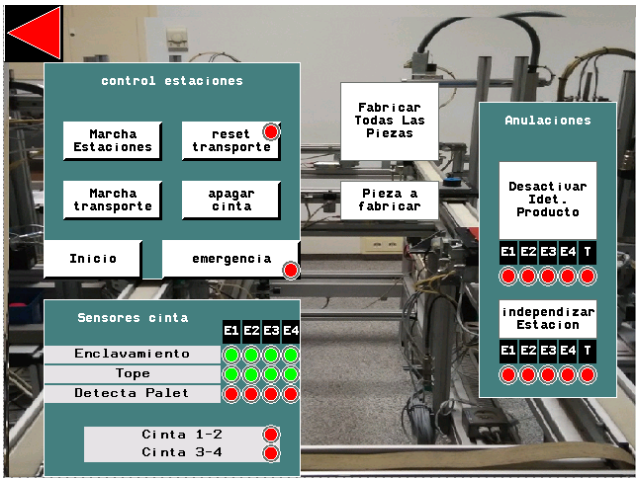


Figura 206.Control General

Anexo 4 Control desde Schneider

Para realizar la implementación para este control se partirá de las máquinas de estados implementadas en el control desde Simulink, a partir de ellas, se generará código para los autómatas Schneider.

Estación 1

Como ya se ha dicho Simulink se comunicara con la estación por medio del servidor OPC, en el capítulo 4, ya se ha explicado cómo crear las variables en el servidor pero ahora esa variables habrá que crearlas en Unity para poder utilizarlas además de tener ya creadas las variables que se indicaron en el capítulo 2.

Las nuevas variables creadas serán las siguientes para la comunicación serán:

Nombre		Tipo	Dirección	
OPC	Unity		OPC	Unity
Desvio_e1	Desvió	Booleana	1	%M0
Fin_e1	Fin_e1	Booleana	2	%M1
Automatico_terminal_e1	Automatico_terminal_e1	Booleana	3	%M2
Marcha_pantalla	pantalla	Booleana	4	%M3
tipoPieza_e1	tipoPieza_e1	Real	31	%MW30

Tabla 16.Relación entre variables entradas del servidor OPC y Unity

Después de haber creado las variables y tener el código generado con las modificaciones antes comentadas, se implementará el código en un bloque funcional de tipo DFB.

Para crearlo, en el panel izquierda de Unity y al pulsar sobre Tipos de FB derivados. Aparecerá la siguiente ventana, en la cual se creara un nuevo objeto de Tipo DFB.

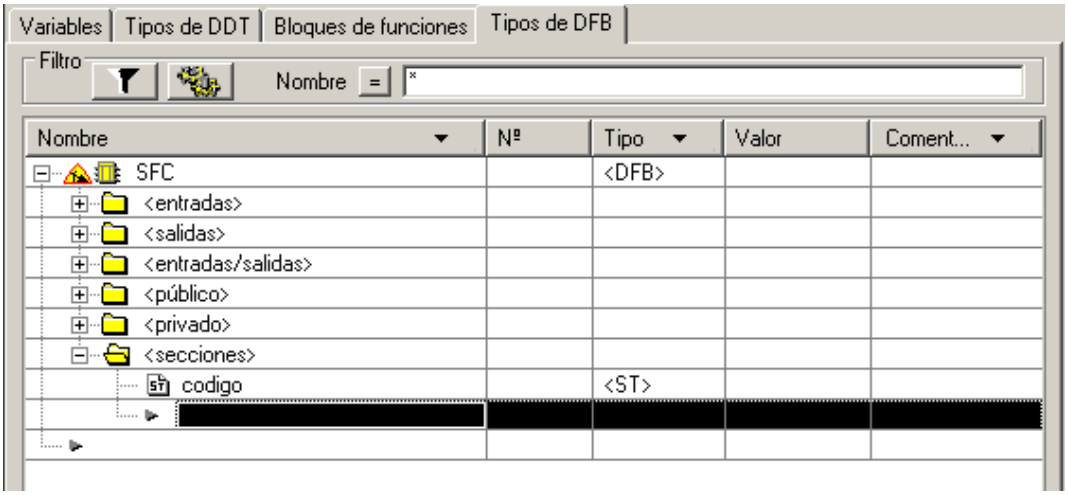


Figura 207.Creación bloque funcional DFB e implementación

En primer lugar en la pestaña secciones, se creará una nueva sección donde se copiará el código final tras la exportación de Simulink y las modificaciones. Para ello, se despegara la pestaña secciones y se creará una rutina de tipo ST. Al abrir esa rutina nos aparecerá un fichero ST donde se introducirá el código tras haber realizado las modificaciones.

Después de esto, habrá que indicar al bloque que variables de las escritas en el código son salidas, entradas o son auxiliares (privadas) y de qué tipo son.

SFC		<DFB>
<entradas>		
pinza_izda_e1	1	BOOL
Capacitivo_e1	2	BOOL
pinza_abajo_e1	3	BOOL
pinza_atras_e1	4	BOOL
pinza_dcha_e1	5	BOOL
optico_lector_e1	6	BOOL
Lector_adelante_e1	7	BOOL
marcha_e1	8	BOOL
cargador_adelante_e1	9	BOOL
pinza_arriba_e1	10	BOOL
Optico_e1	11	BOOL
Inductivo_e1	12	BOOL
Lector_atras_e1	13	BOOL
pinza_adelante_e1	14	BOOL
cargador_atras_e1	15	BOOL
automatico_terminal_e1	16	BOOL
Pieza	17	REAL
man_auto_e1	18	BOOL
ind_int_e1	19	BOOL
pantalla	20	BOOL

SFC		<DFB>
<entradas>		
<salidas>		
pinza_dentro_e1	1	BOOL
pinza_sube_baja_e1	2	BOOL
cargador_e1	3	BOOL
pinza_e1	4	BOOL
pinza_fuera_e1	5	BOOL
brazo_hacia_atras_e1	6	BOOL
lector_e1	7	BOOL
fin1	8	BOOL
brazo_hacia_adelante_e1	9	BOOL
desvio	10	BOOL
<entradas/salidas>		
<público>		
<privado>		
ssMethodType		DINT
is_funcionamiento_normal		UINT
is_lectura_pedido		UINT
is_active_c1_Main_e1		UINT
is_c1_Main_e1		UINT
temporizador		TON
done		BOOL
Tproducto_e1		REAL
Cproducto_e1		REAL
RCproducto_e1		REAL
RTproducto_e1		REAL

Figura 208.Variables Bloque control estación 1

Para finalizar la programación, solo nos falta crear la rutina principal. Esta rutina será de tipo LD en la cual aparecerá el bloque que se ha creado antes, solo bastara con unir las entradas y salidas con las variables correspondientes que controlan la estación.

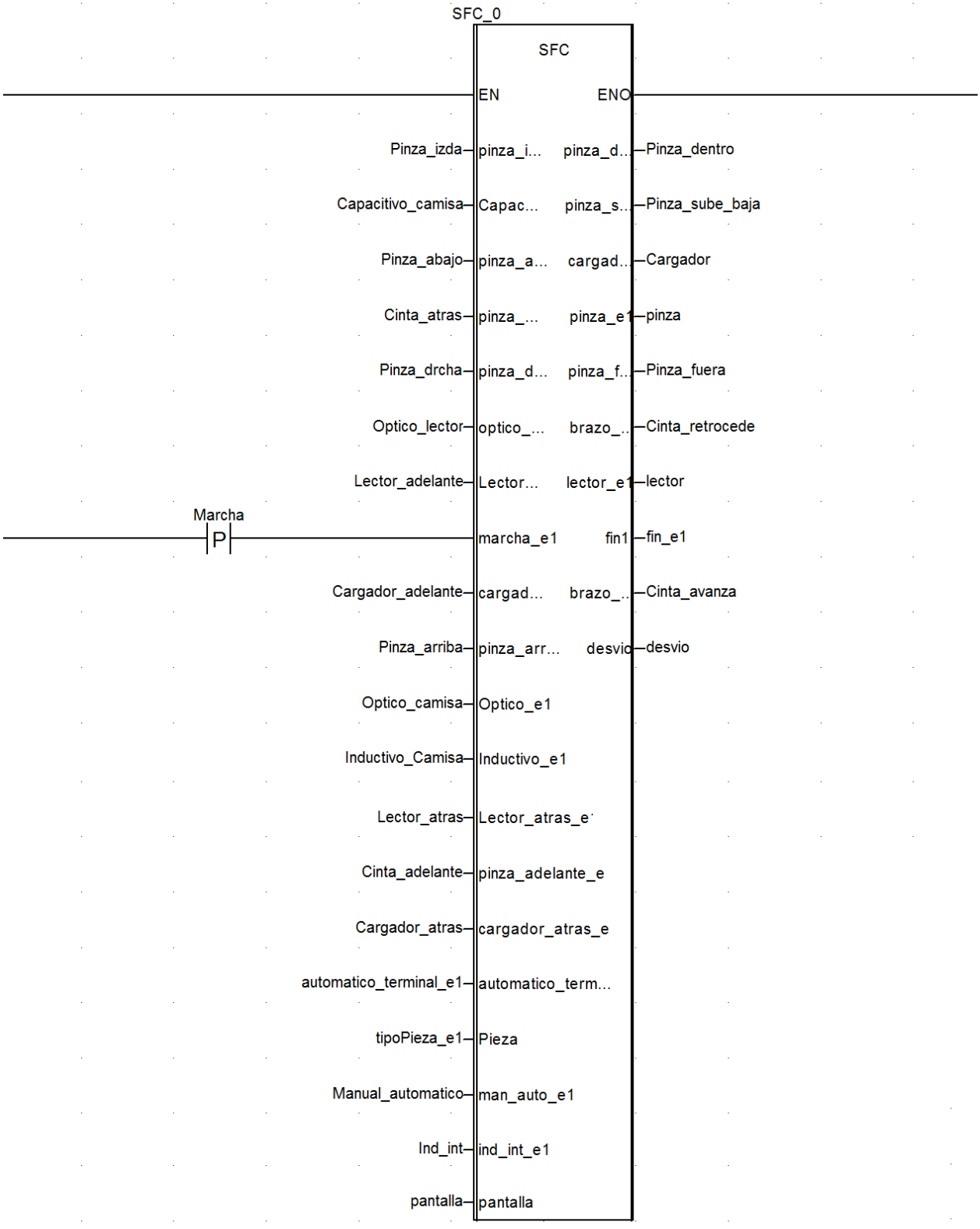


Figura 209.Rutina principal en DFB para Estación 1

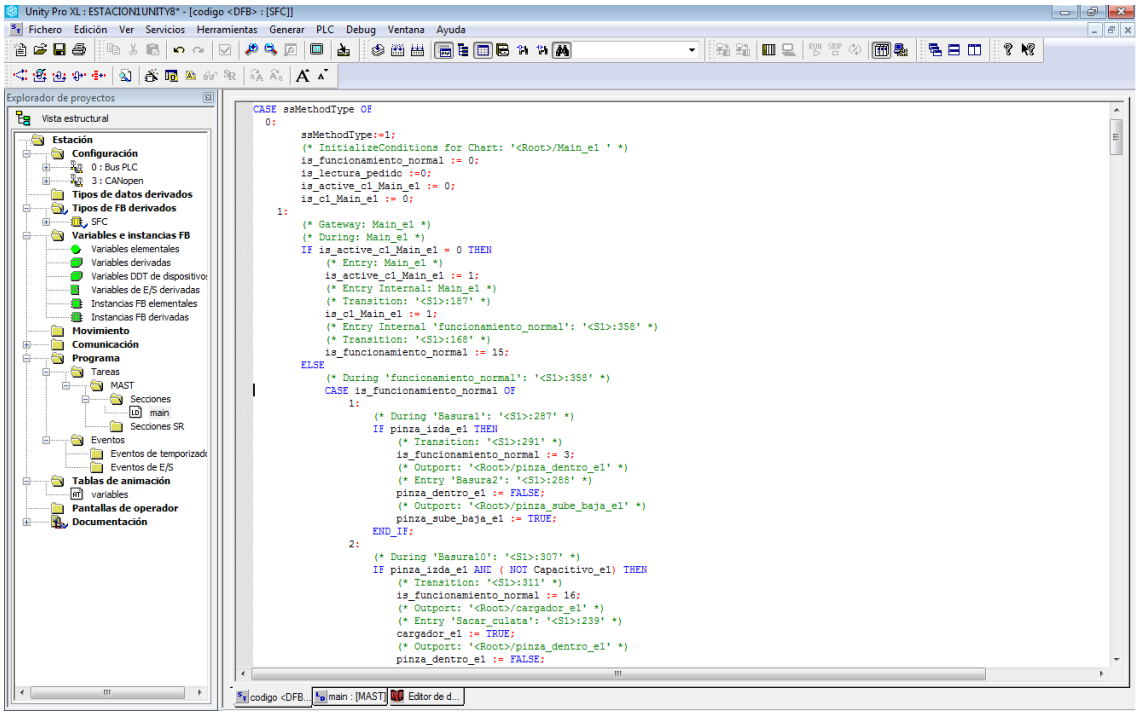


Figura 210.Codigo programado en el bloque DFB

Estación 3 y Cinta

Al igual que en la estación 1 y la 2, habrá que crear las variables para la comunicación con Simulink. Después, se creará el bloque funcional DFB y para acabar la rutina principal.

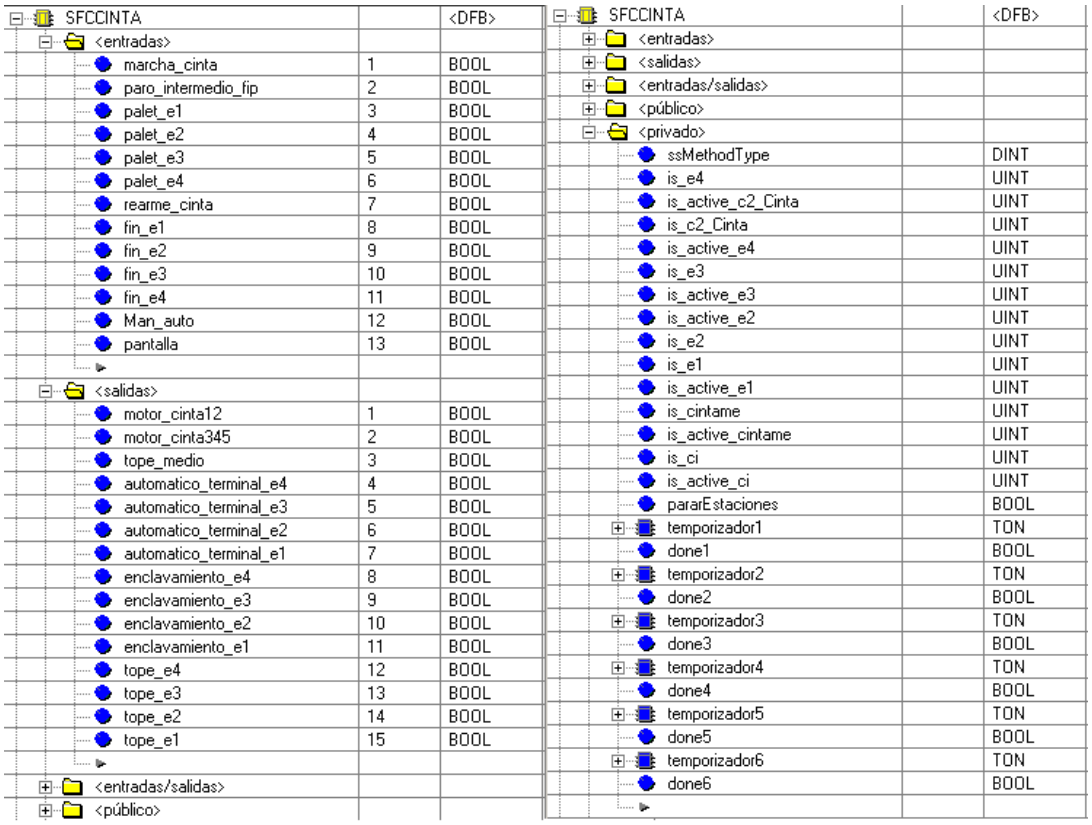
La estación3 controlara por medio de un módulo Ethernet, la cinta, por tanto, en esta estación también habrá 2 bloques funcionales DFB. El primero de ellos será la máquina de estados inicial de la estación 3 y la segunda con la cinta.

Las nuevas variables utilizadas para la comunicación por el OPC serán las siguientes:

Nombre		Tipo	Dirección	
OPC	Unity		OPC	Unity
Desvio_e3	Desvió	Booleana	7	%M6
Fin1	Fin1	Booleana	1	%M0
Fin2	Fin2	Booleana	2	%M1
Fin4	Fin4	Booleana	3	%M2
Automatico_terminal_e1	Automatico_terminal_e1	Booleana	5	%M4
Automatico_terminal_e2	Automatico_terminal_e2	Booleana	6	%M5
Automatico_terminal_e4	Automatico_terminal_e4	Booleana	7	%M6
Marcha_ci	Marcha_cin	Booleana	9	%M8
Marcha_pantalla_cinta	Pantalla_c	Booleana	12	%M11
Marcha_pantalla_e3	Pantalla_3	Booleana	8	%M7
Palet1	Palet_e1	Booleana	10	%M9
Reset_ci	Rearme_cin	Booleana	11	%M10
tipoPieza_e3	tipoPieza	Real	31	%MW30

Tabla 17.Relación entre variables entradas del servidor OPC y Unity

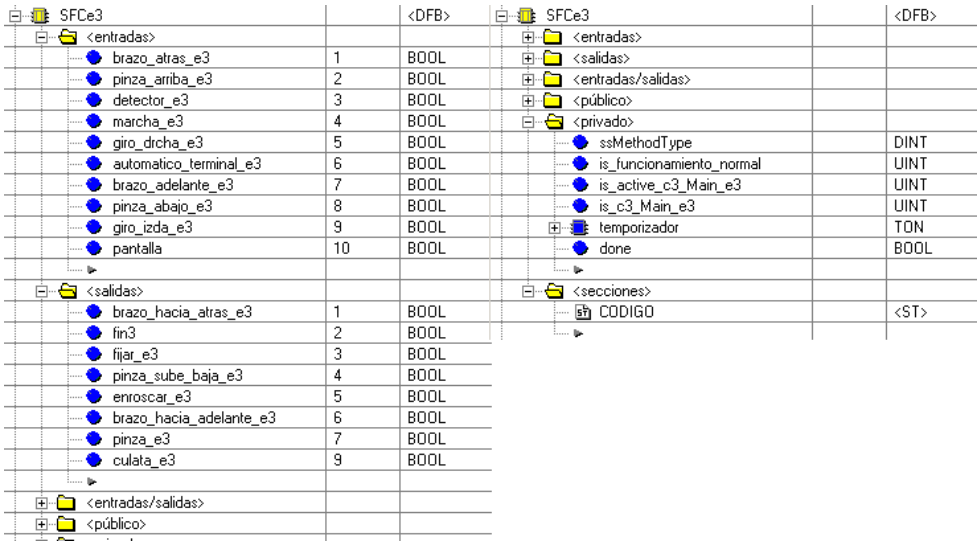
Después de haber creado las variables y tener el código, se pasará a implementar los códigos en dos bloques funcionales de tipo DFB, los cuales se han creado del mismo modo que en la estación 1.



SFCCINTA		<DFB>
<entradas>		
marcha_cinta	1	BOOL
paro_intermedio_fip	2	BOOL
palet_e1	3	BOOL
palet_e2	4	BOOL
palet_e3	5	BOOL
palet_e4	6	BOOL
rearme_cinta	7	BOOL
fin_e1	8	BOOL
fin_e2	9	BOOL
fin_e3	10	BOOL
fin_e4	11	BOOL
Man_auto	12	BOOL
pantalla	13	BOOL
<salidas>		
motor_cinta12	1	BOOL
motor_cinta345	2	BOOL
tope_medio	3	BOOL
automatico_terminal_e4	4	BOOL
automatico_terminal_e3	5	BOOL
automatico_terminal_e2	6	BOOL
automatico_terminal_e1	7	BOOL
enclavamiento_e4	8	BOOL
enclavamiento_e3	9	BOOL
enclavamiento_e2	10	BOOL
enclavamiento_e1	11	BOOL
tope_e4	12	BOOL
tope_e3	13	BOOL
tope_e2	14	BOOL
tope_e1	15	BOOL
<entradas/salidas>		
<público>		

SFCCINTA		<DFB>
<entradas>		
<salidas>		
<entradas/salidas>		
<público>		
<privado>		
ssMethodType		DINT
is_e4		UINT
is_active_c2_Cinta		UINT
is_c2_Cinta		UINT
is_active_e4		UINT
is_e3		UINT
is_active_e3		UINT
is_active_e2		UINT
is_e2		UINT
is_e1		UINT
is_active_e1		UINT
is_cintame		UINT
is_active_cintame		UINT
is_ci		UINT
is_active_ci		UINT
pararEstaciones		BOOL
temporizador1		TON
done1		BOOL
temporizador2		TON
done2		BOOL
temporizador3		TON
done3		BOOL
temporizador4		TON
done4		BOOL
temporizador5		TON
done5		BOOL
temporizador6		TON
done6		BOOL

Figura 211.Variables Bloque control cinta



SFCe3		<DFB>
<entradas>		
brazo_atras_e3	1	BOOL
pinza_arriba_e3	2	BOOL
detector_e3	3	BOOL
marcha_e3	4	BOOL
giro_drcha_e3	5	BOOL
automatico_terminal_e3	6	BOOL
brazo_adelante_e3	7	BOOL
pinza_abajo_e3	8	BOOL
giro_izda_e3	9	BOOL
pantalla	10	BOOL
<salidas>		
brazo_hacia_atras_e3	1	BOOL
fin3	2	BOOL
fijar_e3	3	BOOL
pinza_sube_baja_e3	4	BOOL
enroscar_e3	5	BOOL
brazo_hacia_adelante_e3	6	BOOL
pinza_e3	7	BOOL
culata_e3	9	BOOL
<entradas/salidas>		
<público>		

SFCe3		<DFB>
<entradas>		
<salidas>		
<entradas/salidas>		
<público>		
<privado>		
ssMethodType		DINT
is_funcionamiento_normal		UINT
is_active_c3_Main_e3		UINT
is_c3_Main_e3		UINT
temporizador		TON
done		BOOL
<secciones>		
CODIGO		<ST>

Figura 212.Variables Bloque control estación 3

Para finalizar la programación, solo nos falta crear la rutina principal. Esta rutina será de tipo LD en la cual aparecerá el bloque que se ha creado antes, solo bastara con unir las entradas y salidas con las variables correspondientes que controlan la estación.

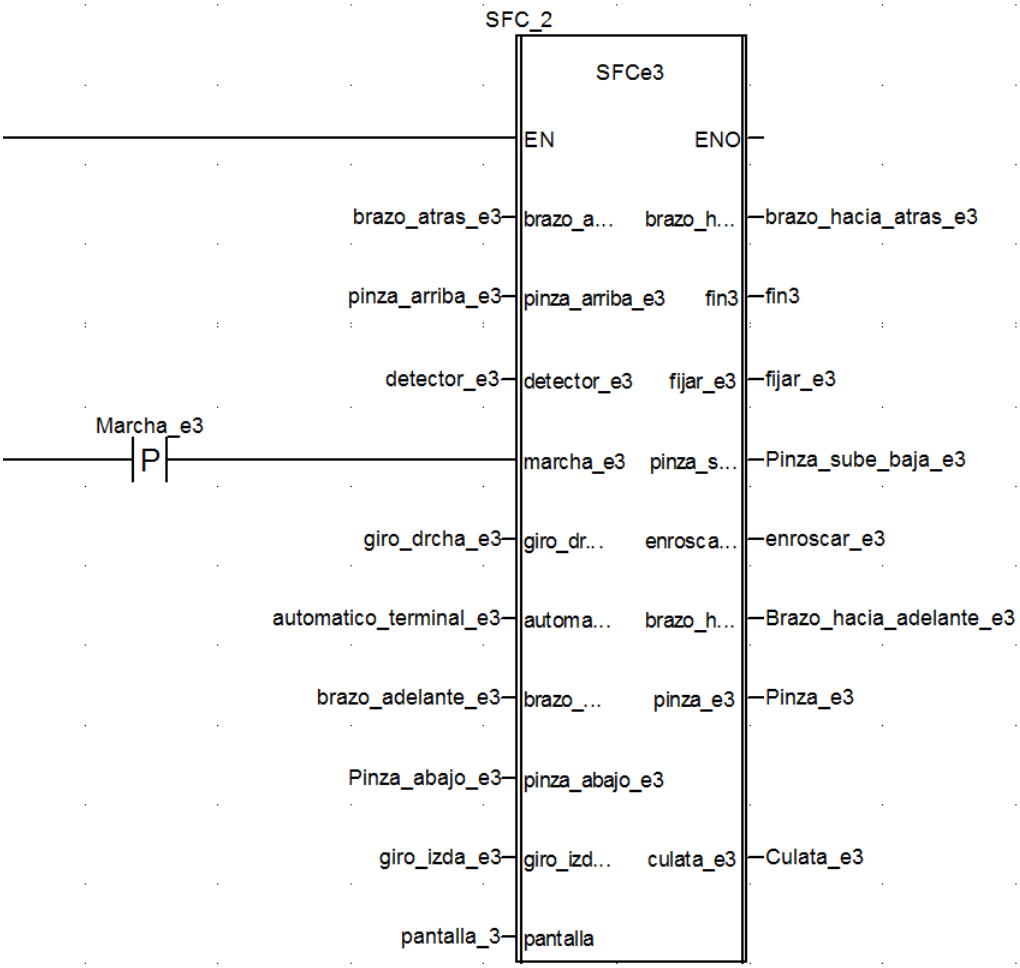


Figura 213.Rutina principal bloque estación 3

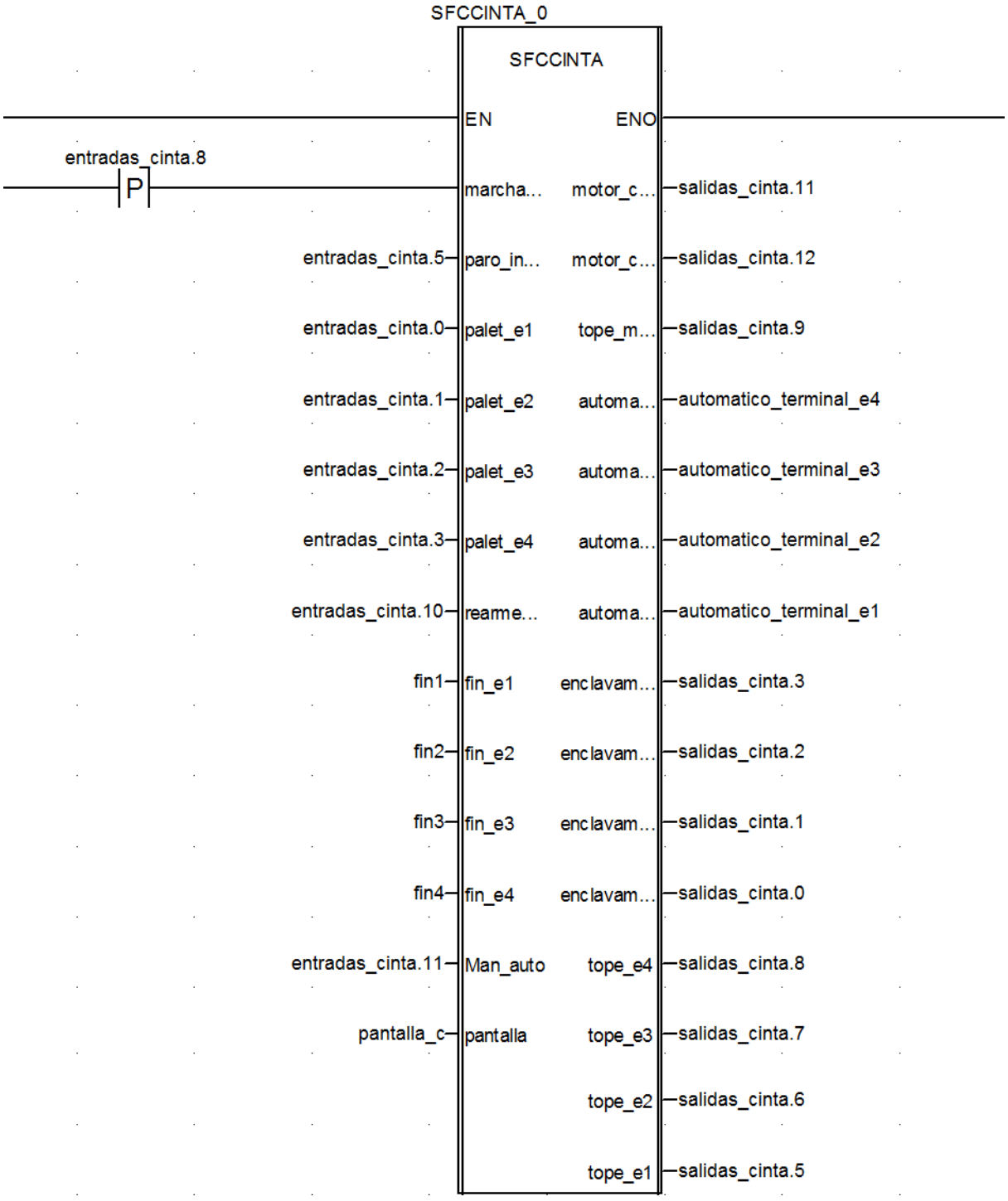


Figura 214.Rutina principal bloque transporte

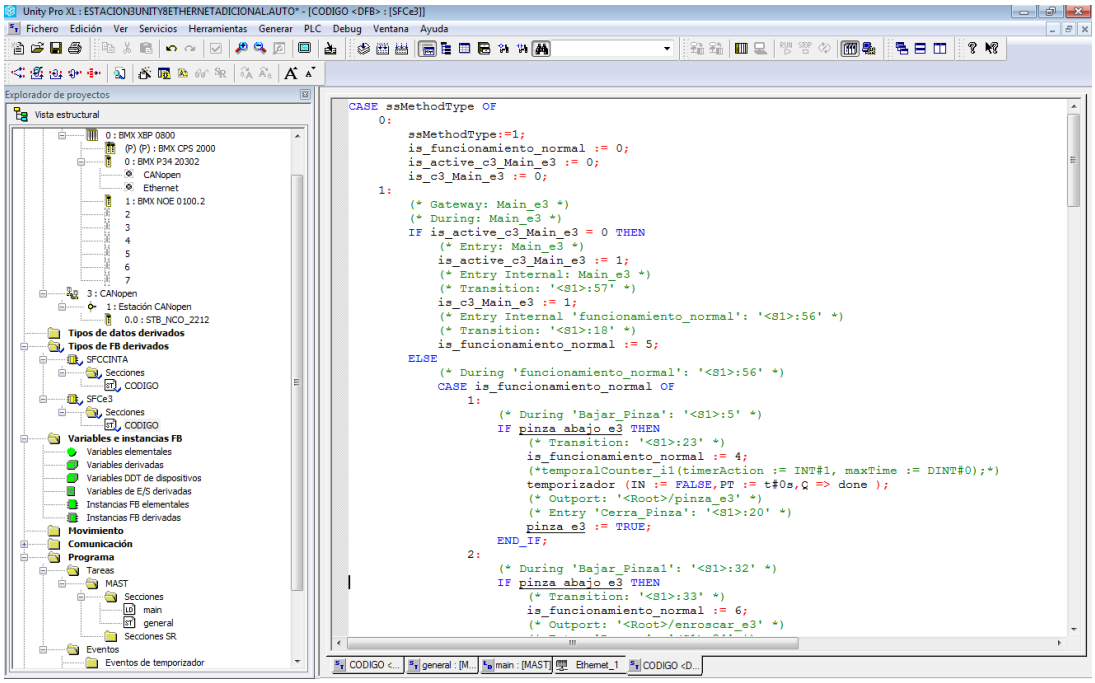


Figura 215.Código bloque DFB Estación 3

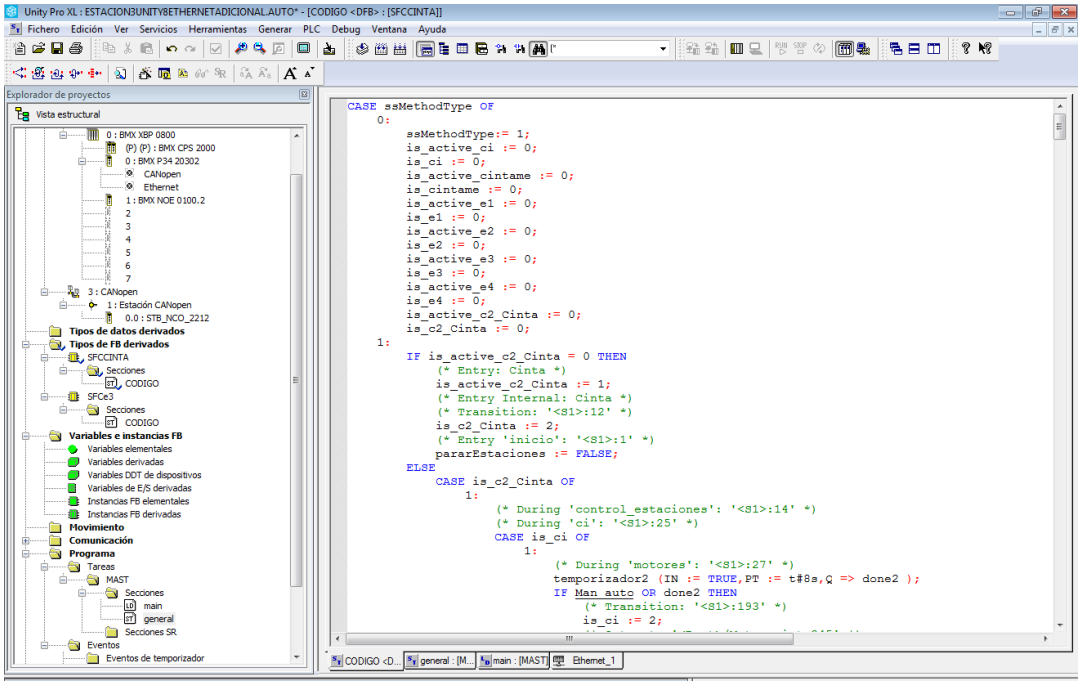


Figura 216.Código bloque DFB Cinta

Estación 4

Al igual que el resto de estaciones, habrá que crear las variables para la comunicación con Simulink. Después, se creará el bloque funcional DFB y para acabar la rutina principal.

Las nuevas variables utilizadas para la comunicación por el OPC serán las siguientes:

Nombre		Tipo	Dirección	
OPC	Unity		OPC	Unity
Fin_e4	Fin4	Booleana	2	%M1
Automatico_terminal_e4	Automatico_terminal_e4	Booleana	4	%M3

Marcha_pantalla_e4	Pantalla	Booleana	1	%M0
tipoPieza_e4	tipoPieza	Real	31	%MW30

Tabla 18.Relación entre variables entradas del servidor OPC y Unity

Después de haber creado las variables y tener el código, se pasará a implementar el código en un bloque funcional de tipo DFB, el cual se ha creado del mismo modo que en la estación 1.

Una vez creado el bloque, se creará una nueva sección en cada bloque donde se copiará el código final tras la exportación de Simulink y las modificaciones realizadas.

Después de esto, habrá que indicar al bloque que variables de las escritas en el código son salidas, entradas o son auxiliares (privadas) y de qué tipo son.

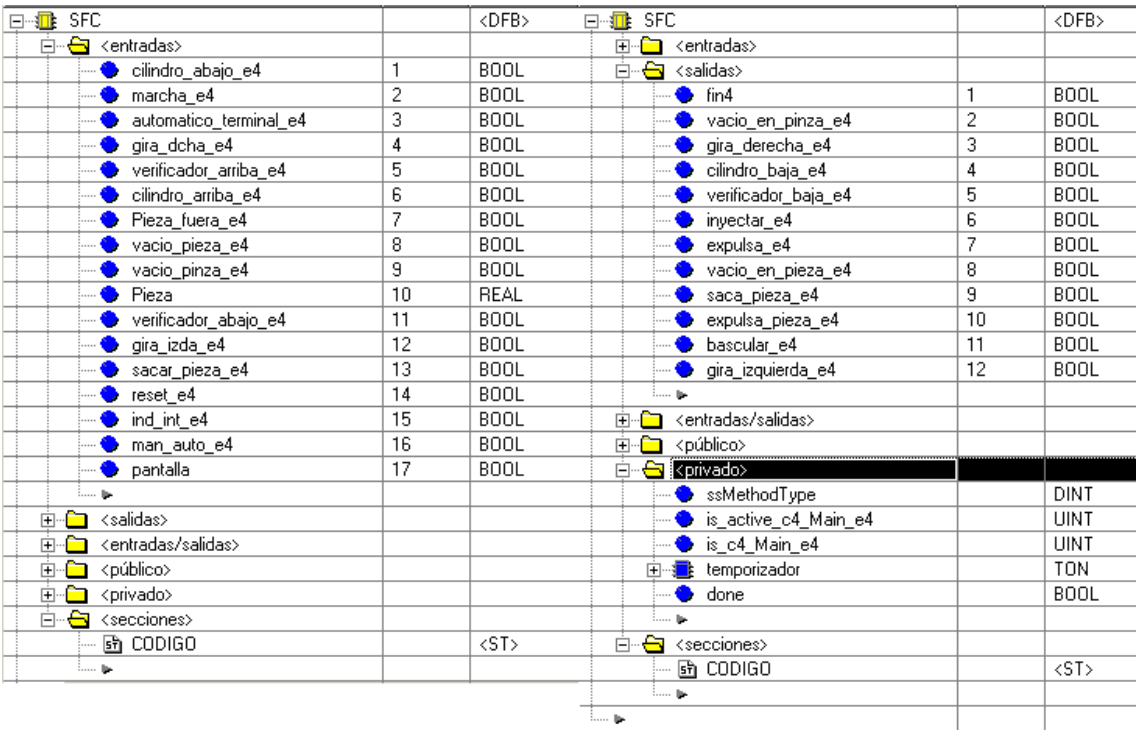


Figura 217.Variables Bloque control estación 4

Para finalizar la programación, solo nos falta crear la rutina principal. Esta rutina será de tipo LD en la cual aparecerá el bloque que se ha creado antes, solo bastara con unir las entradas y salidas con las variables correspondientes que controlan la estación.

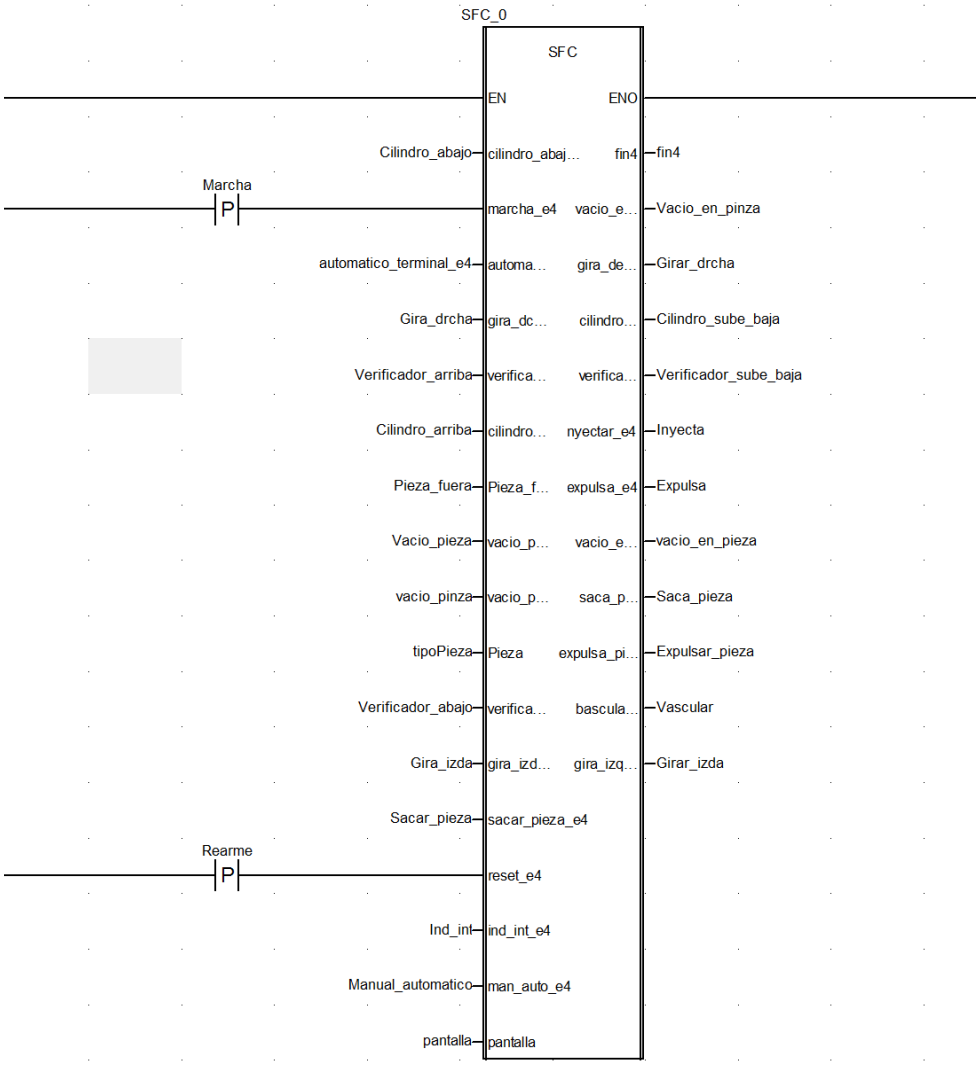


Figura 218.Rutina principal bloque estación 4

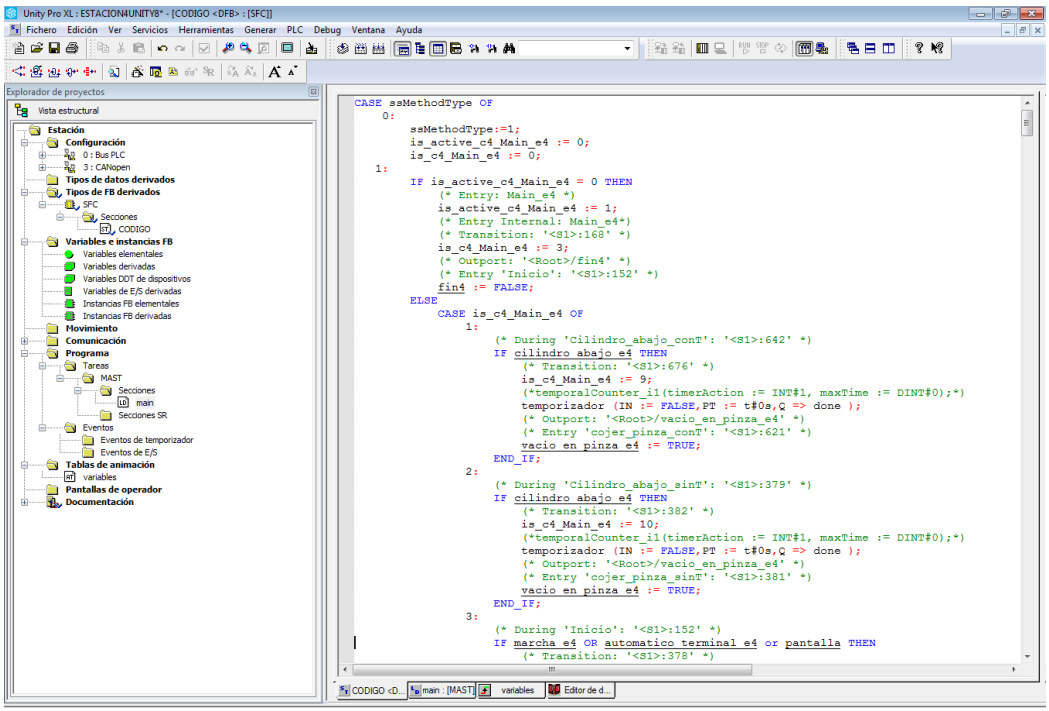


Figura 219.Código bloque DFB estación 4