Universidad de Oviedo

Memoria del Trabajo Fin de Máster realizado por

DIEGO RODRIGUEZ GARCÍA

DNI: 53552804-H

para la obtención del título de

Máster en Ingeniería de Automatización e Informática Industrial

GENERACIÓN DE CÓDIGO IEC-61131-3 MULTIPLATAFORMA PARA SISTEMAS DE ALMACENAJE AUTOMÁTICO

JULIO DE 2017

TABLA DE CONTENIDO

1. Objetivo y alcance

1.1.- OBJETIVOS

El objetivo de este proyecto es desarrollar una herramienta de generación de código para el ámbito de la programación de sistemas control para soluciones logísticas de almacenaje de Mecalux S.A. Esta implementación tiene como punto de partida la necesidad de generar código multiplataforma para distintos fabricantes, de forma universal, de modo que se pueda cubrir una mayor amplitud de mercado en función de las necesidades, preferencias y regiones geográficas concretas de los clientes.

1.2.- ALCANCE

El resultado final deberá ser una herramienta completamente funcional que sea capaz de cubrir las necesidades de cada cliente, independientemente de las dimensiones de la instalación. Además, deberá ser lo más sencilla posible para el usuario.

Por otro lado, puesto que el código generado deberá servir para distintos fabricantes, para unificar la sintaxis y tener una referencia, el código base de programación estará basado en el lenguaje Texto Estructurado (*Structured Text, ST)* especificado en la norma IEC-61131-3. Por lo tanto, el alcance de este proyecto se restringe a fabricantes que permitan la importación externa de código y que sigan las directrices establecidas por dicha norma y la organización PLCopen (por lo menos, en cuanto a los aspectos básicos de programación), dejando al margen fabricantes con filosofías de automatización independientes.

Las características básicas de esta aplicación serán:

- Herramienta de desarrollo única.
- Etapa de generación multiplataforma.
- Exportación de código.
- Directrices según la norma IEC-61131-3
- Programación unificada en Texto Estructurado (*ST).*
- Aplicación intuitiva para el programador.

1.3.- LOCALIZACIÓN

El proyecto se presentará como Trabajo Fin del Máster en Ingeniería de Automatización e Informática Industrial impartido en la Escuela Politécnica de Ingeniería de Gijón. La empresa beneficiaria y responsable del resultado es Mecalux Software Solutions, localizada en la calle Ataulfo Friera Tarfe número 12, 33211, Gijón. Los tutores responsables del seguimiento del proyecto han sido:

- Tutor académico: Felipe Mateos Martín (Área de Ingeniería de Sistemas y Automática).
- Tutor de empresa: Javier Piñera García (Responsable de equipo de automatización).

1.4.- DOCUMENTACIÓN DEL PROYECTO

El proyecto está compuesto de los siguientes documentos:

- Poster con información y descripción resumida del resultado y ejecución del presente proyecto.

- Memoria descriptiva que constará de:
  - Descripción y objetivos del proyecto.
  - Ámbito de ejecución e información de las necesidades.
  - Estado del arte.
  - Descripción de la implementación.
  - Ejemplo de aplicación de la herramienta de desarrollo.
  - Planificación.
  - Presupuesto estimado.
  - Conclusiones.
  - Bibliografía.

- Presentación PowerPoint para la defensa y exposición del proyecto.

2. Introducción

Para comprender mejor el contexto en el que se enmarca el proyecto, se procederá a comentar algunos puntos básicos sobre la actividad industrial de Mecalux S.A. y el entorno que rodea la empresa (industria dedicada desarrollar sistemas de almacenaje de bienes y mercancías).

2.1.- EL SECTOR LOGÍSTICO EN LA ACTUALIDAD

Dado el crecimiento actual de ciertas empresas en países en desarrollo, las empresas de los países más avanzados necesitan reducir los costes de producción para poder competir con las primeras. A esto se le suma la crisis económica, que ha provocado que las compañías hoy en día deban estar en constante cambio y en una búsqueda permanente de nuevos sistemas, permitiendo reducir el precio de manufacturación de sus productos.

Aquí, es donde ha visto el sector logístico una gran oportunidad. El potencial ahorro (aunque tras una gran inversión) que supone la automatización de almacenes industriales ha llevado a la aparición de grandes empresas dedicadas a prestar estos servicios. Por tanto, aunque la demanda se ha incrementado, la competencia en este sector ha aumentado de forma exponencial.

Aunque puede parecer que la logística se encuentra en un segundo plano en el proceso de producción, esto no es realmente cierto. Mientras que la manufacturación de un producto es sencilla de controlar si se dispone de la maquinaria y el *know-how* adecuado, la demanda de productos terminados presenta altibajos irregulares y ciclos estacionales. Esta gran variabilidad es un problema común para proveedores y fabricantes, suponiendo grandes costes y en ocasiones pérdidas para la propia empresa (por lo menos beneficio que deja de obtener).

Si a todos estos factores se les suma la actual filosofía de compraventa de productos debida a las nuevas tecnologías y, enfoques de mercado que exigen la reducción de tiempo de expedición y entrega (dificultando técnicas de reducción de *stock* como *Just-in-time,* basada en producir en función de los pedidos ya obtenidos) se llega a la conclusión de que en la actualidad las empresas tienen prácticamente obligación de disponer de sistemas de

almacenaje modernos, para ser competitivos en los mercados industriales y enfocados al consumidor. Por tanto, dadas los puntos expuestos, entre otros, el sector logístico espera un gran aumento de demanda y beneficios.

2.2.- DESCRIPCIÓN DE LA EMPRESA

Mecalux S.A. es una empresa inmersa en el sector logístico explicado en el punto anterior. Esta compañía se dedica al diseño, fabricación, venta y realización de proyectos llave en mano de sistemas de almacenaje, incluyendo estanterías metálicas, soluciones automatizadas y proyectos a medida para clientes que requieren innovaciones especiales.

Los orígenes de la empresa se remontan a la década de 1960 en Barcelona, donde partiendo de un pequeño taller de fabricación de estantería ligera se fue desarrollando una red comercial por el territorio español hasta 1980. A partir de entonces, Mecalux comienza a abrir nuevas oficinas por el extranjero (París, Reino Unido, Alemania y Argentina) para agilizar la entrada en mercados internacionales. Mientras tanto, el catálogo de productos ofrecidos se iba ampliando según la demanda de los clientes. Fue entonces cerca del año 2000 cuando Mecalux sufre su expansión más notable. Se abren nuevas oficinas y centros productivos en numerosos países, se crean nuevas delegaciones y se fundan divisiones de robótica y automatización para dar soluciones automáticas. De esta forma también comienza el desarrollo tecnológico del sistema de gestión de almacenes EasyWMS.

Actualmente la compañía se sitúa en el tercer puesto de ventas a nivel mundial dentro del sector logísticos, siendo líder en España. Cuenta con una red de distribución implantada en Alemania, Bélgica, Eslovaquia, España, Francia, Holanda, Italia, Reino Unido, República Checa, Polonia, Portugal, Argentina, Uruguay, Perú, Chile, Panamá, Brasil, México, Canadá, Turquía y Estados Unidos. Por otro lado, la empresa dispone de once centros productivos ubicados estratégicamente. En la Figura 2.1 puede distinguirse la localización de los distintas fábricas y el mercado que Mecalux cubre en la actualidad.

En cuanto a términos económicos, Mecalux S.A. está actualmente valorada en cerca de los 600 millones de euros y ha sufrido un crecimiento de un 13% con respecto al año anterior (sobre todo gracias al mercado norteamericano).

Figura 2.1. Mercado abarcado por Mecalux S.A.

2.3.- DESCRIPCIÓN DE UN ALMACÉN

Para entender el contexto del presente proyecto, es necesario contar con unas nociones básicas de los sistemas de almacenaje. En este apartado se explicarán ciertos conceptos necesarios para comprender el ámbito del proyecto.

2.3.1. Que es un almacén

Un almacén es una instalación que, en conjunto con los equipos de almacenaje, de manipulación, medios humanos y de gestión, permite ajustar las diferencias entre los flujos de entrada de mercancía recibida de proveedores o centros productivos y los flujos de salida (mercancía enviada a los puntos de venta, a producción, etc.). Puesto que estos flujos habitualmente no se encuentran en sintonía, es necesario recurrir al almacenaje para optimizar dicha tarea.

2.3.2. Como se pueden clasificar los almacenes

Existen numerosos tipos de almacenes según la actividad económica de la empresa propietaria (de materias primas, productos semielaborados, productos terminados…), todos ellos enfocados para cubrir las necesidades de funcionamiento a solventar. Según el tipo de edificación pueden existir clasificaciones del tipo: aire libre, naves, silos, cámaras frigoríficas, almacenes autoportantes (estructura formada por la propia estantería), entre otros. Sin embargo, la clasificación habitual suele realizarse en función de la naturaleza del

producto, pudiendo encontrar almacenes especializados para productos inflamables, recambios, productos perecederos, almacenes de uso general, etc.

Por encima de estas clasificaciones, los almacenes pueden ser de tipo clásico (los operarios son los encargados de situar los contenedores en las distintas ubicaciones mediante medios mecánicos, como carretillas y transpaletas) y automatizados, donde son las máquinas las encargadas de realizar los flujos dentro del almacén (transportadores de rodillos, vehículos láserguiados, transelevadores…). El presente proyecto será de aplicación para cualquier tipo de mercancía, pero partiendo siempre de almacenes de tipo automático.

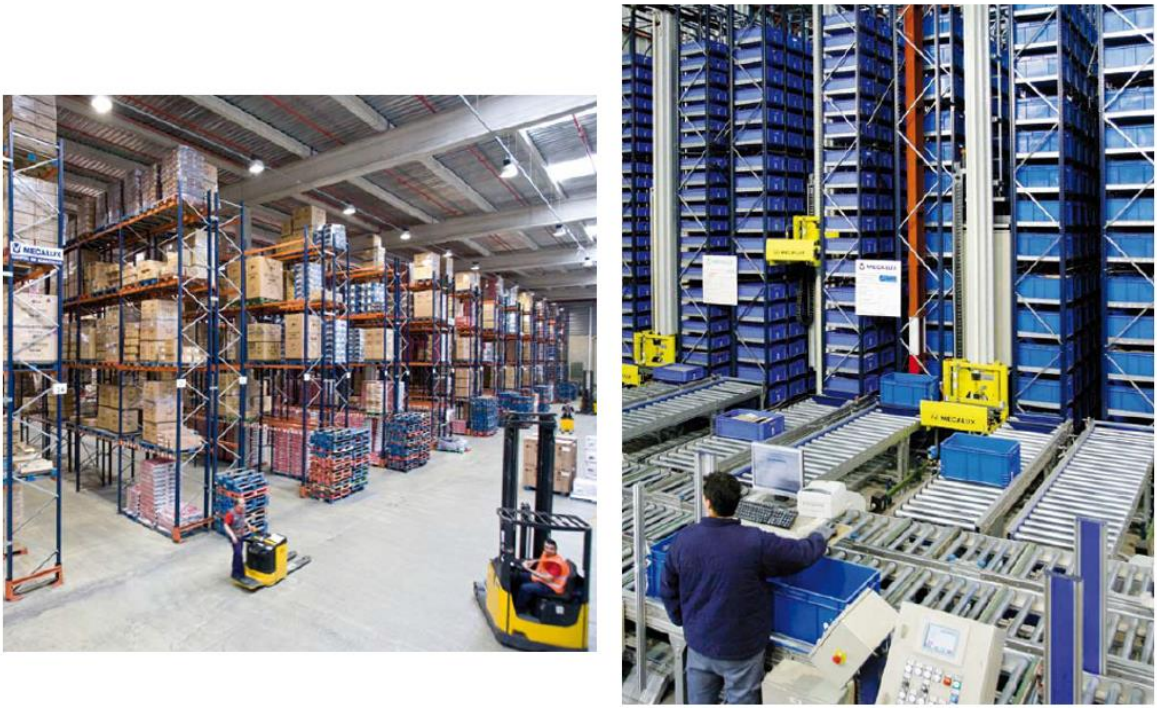
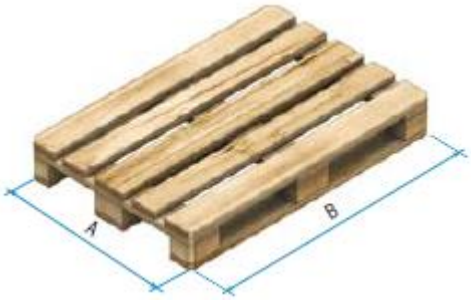


Figura 2.2. Ejemplos de almacenes de tipo clásicos y automatizados.

2.3.3. Unidades de carga

Una unidad de carga es la entidad básica de almacenaje y transporte. Dada la variabilidad de las del tipo de mercancía, se dispone un soporte o embalaje modular (paleta o caja en caso de transporte ligero) para conseguir una manutención eficiente. Lo habitual es que el tamaño de estos sea único en todo el almacén, existiendo tamaños definidos de forma estándar (clasificados en tipos numéricos, siendo los I, II y III los más utilizados globalmente).



LONGITUD (B)	FONDO (A)
1.200	800
	1.000
	1.200

Dimensiones nominales en mm.

Figura 2.3. Paleta estándar (tipo I, II y III) y caja para mercancías ligeras.

2.3.4. Rotación de productos y flujos

Otro concepto decisivo es la forma de clasificar los productos del almacén según su rotación. Esto tiene grandes repercusiones en el diseño del propio sistema de almacenamiento para optimizar el proceso. Esta clasificación puede dividirse en productos de alta rotación (A), media rotación (B) y baja rotación (C), según el tiempo que permanecen en el almacén en función de la demanda de estos. En la mayoría de almacenes se cumple el Diagrama de Pareto (80/20) el cual especifica que el 80% de las ventas se concentra en el 20% de los productos mientras que el 20% restante de las ventas se divide entre el 80% de los productos restantes.

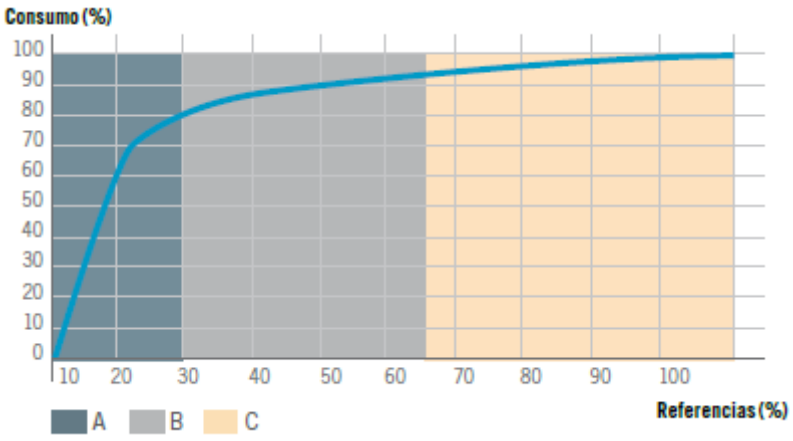


Figura 2.4. Diagrama de Pareto típico de un almacén.

Una vez definida la unidad de carga y la rotación de productos, el último concepto a tener en cuenta son los flujos de mercancía. Puede definirse como la serie de movimientos que tienen lugar dentro de un centro logístico, los cuales consumen tiempo y recursos. Este concepto es muy relevante ya que, el fin último de un almacén es mejorar la actividad de entrada de salida de material para obtener el máximo beneficio económico.

Los movimientos dentro de un almacén pueden ser desde muy sencillos como el representado en la Figura 2.5, de complejidad media (Figura 2.6) y flujos complejos con distintas áreas de trabajo (Figura 2.7).

Figura 2.5. Flujo simple.

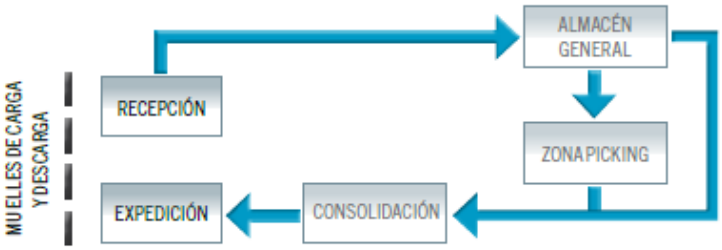


Figura 2.6. Flujo medio.

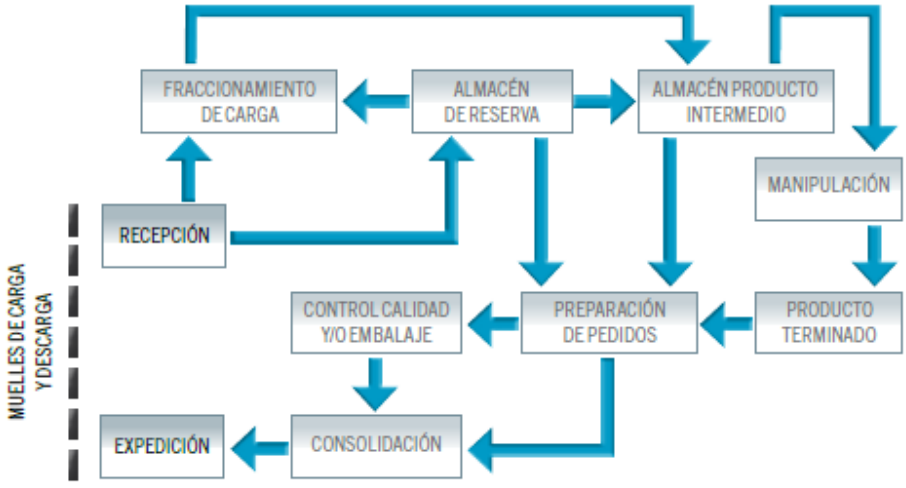


Figura 2.7. Flujo complejo.

2.3.5. Elementos del almacén

En este apartado se explicarán las distintas partes que forman un almacén. El diseño y los espacios asignados a cada zona deben ser adecuados en función de las operaciones que se van a realizar en cada caso particular. A pesar de todas las posibles diferencias, un almacén cuenta básicamente con los elementos señalados en la Figura 2.8.

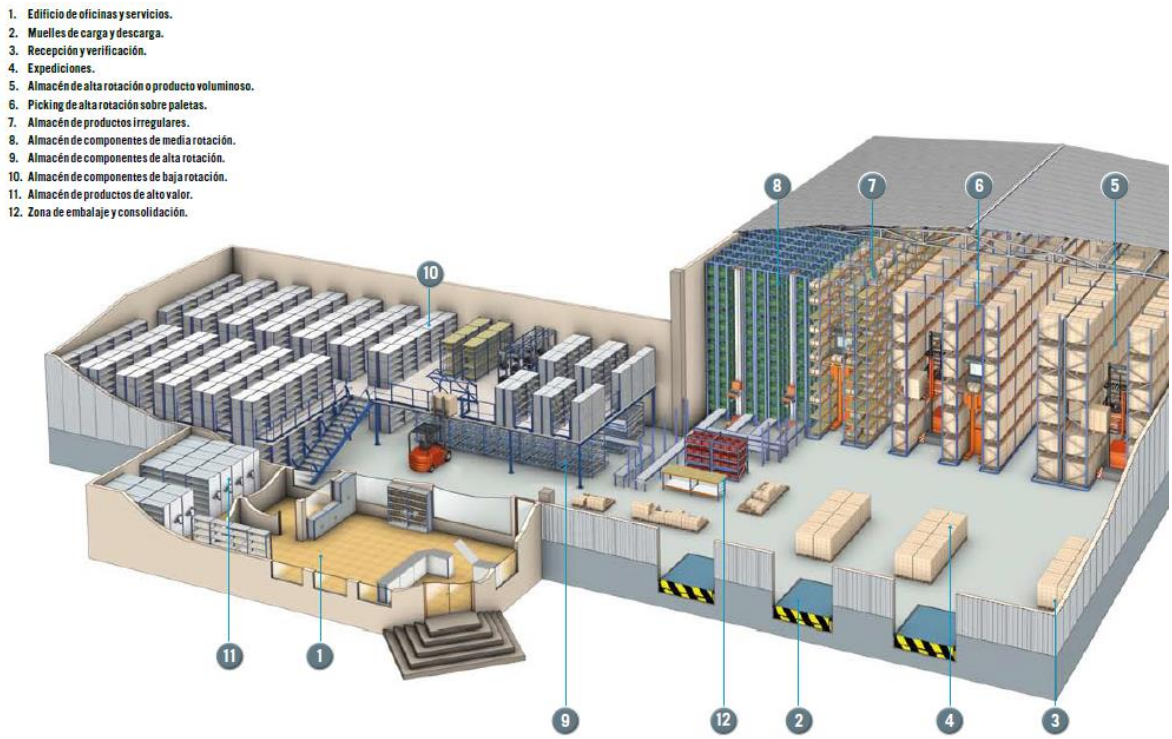


Figura 2.8. Elementos de un almacén.

En la imagen anterior, se contempla una instalación que admite varios tipos de productos y una arquitectura mixta de almacenaje. El material llega por medio de transportadores logísticos a los muelles, donde son descargados y acondicionados para su inspección. Una vez identificados, serán trasladados para su colocación en la ubicación correspondiente en función las especificaciones de la mercancía. Ya que la estancia de los materiales dentro del almacén es temporal, en el caso de su expedición siguen una ruta similar, desde las estanterías a los muelles de carga. Como se ha explicado, estos flujos pueden ser muy simples o extremadamente complejos. En cuanto a los elementos del almacén, pueden combinarse distintas opciones hasta alcanzar instalaciones de un tamaño casi inimaginable.

Concretando más para sistemas de almacenaje automático, aunque el *layout* de un almacén puede ser muy variado, pueden identificarse ciertos elementos comunes en este tipo de instalaciones, tres de los cuales toman vital importancia:

- **Sistema de gestión de almacén (SGA).**

Es el software encargado de gestionar la operativa de la instalación. En la actualidad, prácticamente todos los almacenes cuentan con este sistema, cuyo objetivo es mantener un control sobre los niveles de stock, inventarios y movimientos de la mercancía. Lo habitual es que este sistema esté en consonancia con el resto de la gestión de la empresa (ERP) ya que estos últimos no suelen ser tan específicos para realizar estas labores ya que el SGA debe optimizar todos los movimientos, procesos y operativas dentro del almacén, encargándose de las siguientes tareas:

- Recepción de mercancía: administra la entrada de las mercancías.
- Almacenaje e inventario: gestiona la ubicación de los contenedores.
- Control de stock: registra el estado cantidad, fecha, etc. del stock alojado en el almacén.
- Expediciones: elaboración de pedidos que se envían a los clientes, traspasos…

El sistema de gestión de almacén comercializado por Mecalux es EasyWMS, el cual integra características como: gestión de lectores RFID, impresión de etiquetas, sistema de almacenamiento en la nube, interfaz de usuario web, etc.

- **Sistema de control.**

Como se ha presentado, el SGA podría considerarse como el elemento inteligente dentro de un almacén, el encargado de buscar las rutas más adecuadas. Sin embargo, éste no es el responsable de controlar las máquinas de la instalación para llevar la mercancía de un punto a otro. Este otro participante de más bajo nivel es el sistema de control, que ejecuta el código necesario para realizar los movimientos ordenados por el SGA –por tanto, se necesitará algún tipo de comunicación entre ambos sistemas-. Existen multitud de soluciones para implementar esta tarea, como pueden ser PLCs, *SoftPLCs* u otros sistemas específicos.

Mecalux ofrece tanto soluciones PLC (Siemens, Rockwell…) como propio sistema de control (Galileo) basado en la tecnología *SoftPLC*.

- **Máquinas específicas.**

Aún a más bajo nivel se encuentran las propias máquinas encargadas del movimiento. Aquí se engloban los motores, variadores de velocidad y sensórica necesarios.

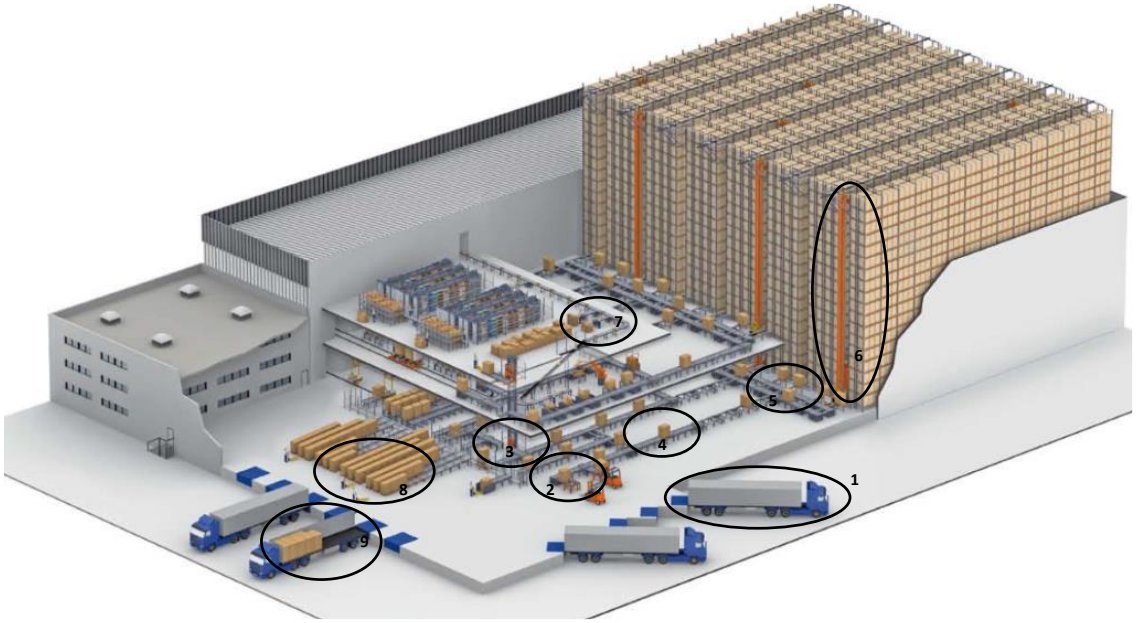


Figura 2.9. Almacén completamente automatizado

En este esquema (Figura 2.9) se distinguen los siguientes elementos:

1. Muelle de recepción. Aquí se reciben las mercancías a almacenar.
2. Puesto de entrada a cabecera. Los pallets de mercancía son descargados desde los camiones e introducidos en este transportador mediante carretillas, donde son reconocidos de forma automática (todas las mercancías llevan un código de barras para su identificación). Una vez son identificados, el sistema informático decide la mejor ubicación para el tipo de producto correspondiente.
3. Elevador para cambio de planta.
4. Transportadores lineales para el transporte de los pallets por el almacén.
5. Punto de decisión y gestión del tráfico. Se calcula una nueva ruta en función de tráfico del almacén y se lleva al punto de entrada al almacén correspondiente.

6. Transelevadores. Son las máquinas encargadas de ubicar los pallets en las estanterías y recogerlos cuando vayan a ser expedidos.
7. Puestos de picking. Mesas donde los operarios pueden recoger productos específicos de los contendedores transportados para preparar nuevos pedidos, montaje de piezas, etc.

3.Necesidades actuales

En este apartado se pretende dar a conocer las necesidades que han hecho que surja el presente proyecto. No obstante, antes hay que dedicar ciertos párrafos a explicar el estado del sistema de control de Mecalux.

3.1.- SOFTWARE DE CONTROL

En primer lugar, debe distinguirse entre sistema de control y las herramientas de desarrollo de control.

3.1.1. Sistema de control – Galileo

La función básica del software de control Galileo es gobernar los equipos electromecánicos de la instalación. Para hacerlo, gestiona los elementos del hardware (buses de campo, variadores, tarjetas de entrada/salida, etc.), registra las averías y se comunica con el nivel superior de gestión (SGA). Todo esto se realiza mediante la ejecución del código máquina generado a partir del programa de control.

En cuanto al tipo de arquitectura, Galileo se incluye dentro de la tecnología *SoftPLC*. Esto significa que, a diferencia de las aplicaciones clásicas que gobiernan las máquinas mediante un controlador lógico programable independiente, se utiliza un programa informático basado en PC para realizar estas tareas. Para realizar las funciones de entrada y salida de periferia se deben emplear tarjetas PCI o PCI-Express dependiendo del bus de campo utilizado. Actualmente Galileo admite: Interbus-S, Profibus, Canopen, Modbus, comunicaciones TCP/IP y comunicaciones inalámbricas Bluetooth. A nivel de ejecución, Galileo se ejecuta como un servicio (proceso en segundo plano) y no como una aplicación de escritorio, de esta forma se garantiza que se ejecute de forma más segura.

3.1.2. Herramienta de desarrollo – Designer

Designer es la herramienta utilizada para programar el código de control que posteriormente será ejecutado por Galileo. Las funciones principales de esta aplicación son:

- Ofrecer un entorno integrado de desarrollo (IDE), de forma que desde el mismo puedan definirse la lógica de programa, las variables y la asignación de direcciones de entrada salida encargadas de gobernar una instalación automatizada.
- Comunicarse con el sistema de control para mostrar visualizaciones creadas por el programador (actuar como SCADA), ver ejecuciones online del programa de control (valor de variables, métodos, etc.) y depurar el código generado.
- Compilación, generación y enlazado del programa de control generado para obtener un fichero con código máquina ejecutable por el *SoftPLC*.

3.1.3. Sistemas externos

Puesto que existen clientes reacios a la utilización de Galileo, Mecalux también ofrece soluciones basadas en sistemas de control clásico (PLC estándar), siguiendo la preferencia de los clientes. De esta forma, existen numerosas instalaciones gobernadas por sistemas de Siemens, Allen Bradley, etc. En estos casos, la tarea de desarrollo se realiza utilizando los IDE de cada fabricante en función de la solución elegida. Cabe destacar que el segundo punto de Designer (visualizaciones) sigue vigente, ya que éste ofrece la posibilidad de conectarse a un PLC externo mediante la comunicación de bloques de datos.

3.2.- JUSTIFICACIÓN DEL PROYECTO

Como se ha comentado, en la actualidad Mecalux oferta proyectos en función de las preferencias de cada cliente particular, ya sea por razones de situación geográfica, experiencia de los operarios, etc. Gracias a esta adaptación, Mecalux pretende cubrir una mayor superficie de ventas. Así, dependiendo del mercado los programas de control se deberán ajustar a:

- Mercado europeo: normalmente caracterizado por la dominación de PLCs de Siemens.
- Mercado americano: donde el personal suele estar más acostumbrado y cualificado para trabajar con sistemas de Allen Bradley, siendo una de las marcas de PLCs con más ventas.
- Clientes que aceptan la utilización del *SoftPLC* propietario de Mecalux: Galileo como sistema de control para sus almacenes.

Top 50 Global Automation Vendors

2015 Global revenue (in millions of U.S. $)

1. Siemens	12,156.74
2. ABB	9,326.01
3. Emerson	8,560.30
4. Schneider Electric	6,356.00
5. Rockwell Automation	5,871.55
6. Mitsubishi Electric	3,522.05
7. GE	3,481.36
8. Honeywell	3,421.67
9. Danaher	3,323.00
10. Yokogawa Electric	3,113.38

Top 50 North American Automation Vendors

2015 North American revenue (in millions of U.S. $)

1. Emerson	3,689.90
2. Rockwell Automation	3,412.08
3. ABB	2,089.09
4. Schneider Electric	1,716.12
5. Danaher	1,462.12
6. GE	1,389.03
7. Siemens	1,103.58
8. Ametek EIG	1,080.80
9. Honeywell	928.62
10. Teledyne Instruments	920.76

Figura 3.1. Ranking de ventas de sistemas de control.

Aunque no existe diferencia a nivel funcional entre los distintos proyectos (ya sea Galileo, o cualquier sistema PLC), sí que existen una serie de pautas diferenciadoras a nivel de desarrollo e implantación. Estas diferencias suelen ser debidas al tipo de tecnología y arquitectura ofrecida por cada uno de los fabricantes de sistemas de control, lo que puede dar lugar a grandes contrastes en el tiempo empleado para el desarrollo de un proyecto en función de la solución de control elegida. Adicionalmente a lo anterior, aunque a nivel funcional los proyectos son equivalentes, actualmente no se puede disponer de un estándar común para todos ellos. Partiendo de estos problemas, Mecalux debe ser capaz de desarrollar sus proyectos para cualquiera de las tecnologías (Siemens, Rockwell y Galileo, por el momento), esperándose para todas ellas unos resultados similares tanto con respecto a estándar de máquinas, tiempo de desarrollo y resultado funcional.

3.3.- OBJETIVOS ESPERADOS

Para conseguir combatir los problemas presentados, es necesario desarrollar un código base para las distintas máquinas ofrecidas por Mecalux utilizable desde cualquiera de los tres sistemas explicados. Este código deberá estar probado y validado mediante una

metodología eficiente y lo más sistemática posible. Además, debería integrarse el propio sistema de generación como un punto dentro de la planificación de la venta de una instalación. Para entender mejor esto, hay que comentar que Mecalux sigue ciertas pautas a la hora de tratar con un cliente, las cuales pasan desde la simulación del *layout* de la instalación (aspecto físico), ofertada por el equipo de ventas, cálculo de ciclos (número de movimientos por hora que el almacén es capaz de realizar), desarrollo del programa de control y modificaciones del comportamiento del SGA para adaptarse a las condiciones de la instalación.

De esta forma, los objetivos para abordar estas necesidades quedan claramente diferenciados:

1. Unificación del entorno de desarrollo independientemente de la solución elegida. De lo contrario deberían existir estándares de desarrollo para cada una de las tecnologías, dificultando la mantenibilidad y flexibilidad de los proyectos. Lo que llevaría a problemas de tiempo necesario por proyecto, discrepancia entre versiones y fallos por código desactualizado.

2. Disponer de un repositorio de máquinas estándar que pueda ser empleado en cualquiera de los sistemas presentados, comportándose igual en cualquiera de ellos. Así se reducirán los tiempos necesarios de desarrollo y se evitarán errores conocidos. También se facilitará la tarea al operario al tener que enfrentarse siempre a un código similar.

3. Integrar un sistema de importación de archivos procedentes de otros programas de la propia empresa (Mecalux) para permitir reducir el tiempo aprovechando el trabajo ya realizado para las fases de simulación y cálculo de ciclos.

4. Planificar una metodología de desarrollo y una arquitectura de pruebas tal que pueda realizarse una correcta validación antes de su uso en la instalación real, garantizando puestas en marcha más cortas y sencillas.

4. Estado del arte

En este apartado, se comentarán los tipos de herramientas, las facilidades y las limitaciones que pueden encontrarse en el mercado a la hora de empezar a plantear este proyecto.

4.1.- ENFOQUE ACTUAL DE LOS PLC

El mercado de los controladores lógicos programables siempre se ha caracterizado por la fuerte competitividad entre las distintas compañías. Esto se traduce en que desde el nacimiento de los PLCs siempre han aparecido dispositivos fuertemente restringidos por sus fabricantes, con el fin de obtener el monopolio de mercado forzando la fidelidad de sus clientes mediante la incompatibilidad con elementos externos. Así, no es raro encontrarse dispositivos de ciertos fabricantes que utilizan arquitecturas, protocolos y sistemas cerrados discordantes con los ofrecidos por la competencia. En cuanto a la filosofía de programación, siempre se ha sido muy conservador en cuanto a la evolución de los lenguajes. Dado que no se esperaban unos altos conocimientos de programación, ni procesos excesivamente complejos para tratar sistemas de control (en comparación con la ingeniería de software, que tiene un ámbito mucho más amplio: consultorías, sistemas de gestión, sistemas bancarios…), los lenguajes han sido muy estáticos, no adaptándose a los cambios sufridos por ejemplo por las herramientas de desarrollo de software para computadores. Por esta razón, aún pueden encontrarse lenguajes de programación muy arcaicos con respecto a otras áreas tecnológicas. Sin embargo, dada la cantidad de demanda desde finales del siglo XX en procesos automáticos y el beneficio que esta filosofía de producción supone, muchas empresas clientes del sector de dispositivos de control industrial han llegado a copar grandes porcentajes del mercado en sus respectivos campos (por ejemplo, la automoción), adquiriendo capitalizaciones multimillonarias. Desde entonces, han cambiado el enfoque de cómo prevén el futuro de la automatización, que pasa por romper la incompatibilidad entre distintos fabricantes y crear un estándar común para distintos elementos (lo cual supondría un ahorro muy importante para ellos). Dada la importancia de estos clientes, las compañías de automatización sufrieron fuerte presión por el miedo a perder gran cantidad de pedidos,

viéndose obligadas a aceptar estas exigencias. Es entonces cuando nace un nuevo estándar en sistemas de control.

4.2.- NORMAS E ORGANIZACIONES INTERNACIONALES

En la última década, se ha establecido de forma bastante estable la filosofía de estandarización de dispositivos de control y todo lo relacionado con ellos. Esto es algo que lleva fomentándose desde finales del siglo XX, y es debido fundamentalmente a dos elementos.

4.2.1. Norma IEC-61131

Después de décadas de incompatibilidad entre sistemas de control, el nacimiento de la norma IEC-61131 (antiguamente IEC-1131) publicada por la Comisión Electrotécnica Internacional (*International Electrotechnical Commission, IEC*), supone un gran logro en la estandarización del entorno de los controladores lógicos programables y todos los dispositivos periféricos relacionados con ellos. Esta se compone de ocho documentos independientes:

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

Concretamente, la tercera parte, la que afecta a la programación de los dispositivos que es objeto del presente proyecto describe, por un lado, los tipos de datos y la especificación de los mismos, es decir la sintaxis de su declaración, así como la representación de cada uno de

ellos (booleanos, enteros 8 bits, reales…). Además, especifica cómo debe ser la estructura del proyecto el cual se divide en diferentes elementos de mayor a menor nivel:

1. Configuración que incluye la especificación del hardware y direccionamiento de entrada salida para el dispositivo.
2. Los recursos que podrían asemejarse con hilos de ejecución independientes.
3. Las tareas que ejecutan programas utilizados típicamente para controlar un proceso bien definido (gestión de alarmas, proceso de llenado de un tanque, etc).
4. Programas, bloques funcionales y funciones que son las unidades de organización de programa (*Program Organization Units,* POU) básicas. Están descritos en los lenguajes de programación establecidos por la norma. Su diferencia radica en la utilización de memoria remanente entre ciclos de ejecución (bloques funcionales) y variables estrictamente temporales (funciones). En cuanto a los programas, están formados por conjuntos de los anteriores.

Por otro lado, se describen los cinco lenguajes de programación admitidos para la realización de POUs. La norma, ha intentado incluir diferentes opciones para adaptarse en la medida de lo posible a los conocimientos ya adquiridos por los profesionales de este sector. De esta forma se recogen y especifican directrices que tienen como base los lenguajes más empleados en distintas partes del mundo:

Sequential Function Chart (SFC): Es un lenguaje gráfico que deriva de los esquemas Grafcet, con ciertas modificaciones para ser más eficientes en el control de procesos. Se divide en etapas (que ejecutan distintos tipos de acciones) y transiciones, es utilizado típicamente para describir comportamientos secuenciales ya que es sencillo e intuitivo para tareas sencillas.

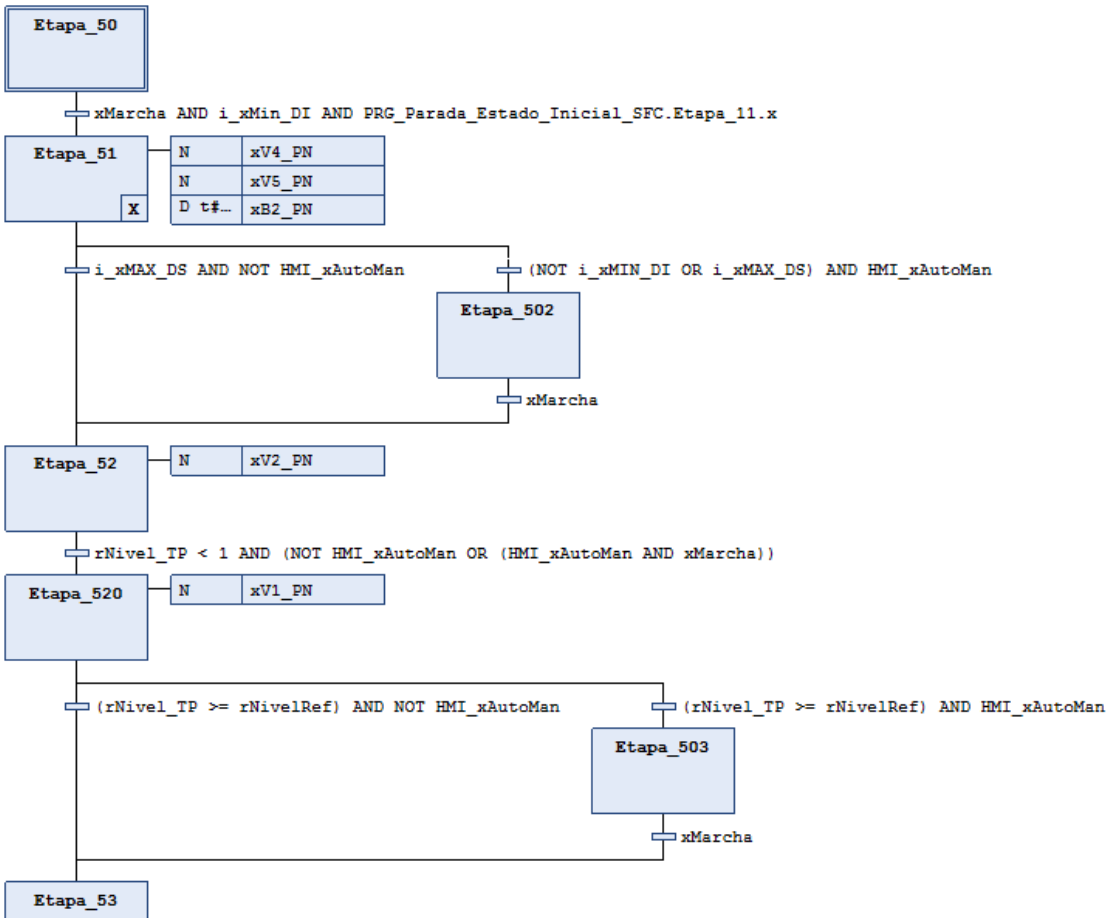


Figura 4.1. Ejemplo de proceso programado en SFC.

Diagrama de contactos (*Diagram Ladder*, LD) típicamente utilizado en el mercado americano de PLCs. Es un lenguaje gráfico que emula la conexión de contactos y bobinas.

Diagrama de bloques funcionales (*Function Block Diagram*, FBD). Una POU descrita según este lenguaje se asemeja con puertas lógicas. Es también un lenguaje de tipo gráfico.

Lista de instrucciones (*Instruction List*, IL) lenguaje de tipo texto que puede identificarse con AWL de Siemens, muy utilizado en Europa.

Texto Estructurado (*Structured Text*, ST) un lenguaje de programación basado en los lenguajes básicos e iniciales de la ingeniería de software (Pascal, C, etc.). Este lenguaje supone una revolución a niveles de los sistemas de control ya que da la posibilidad de programar tareas más complejas de forma estándar.

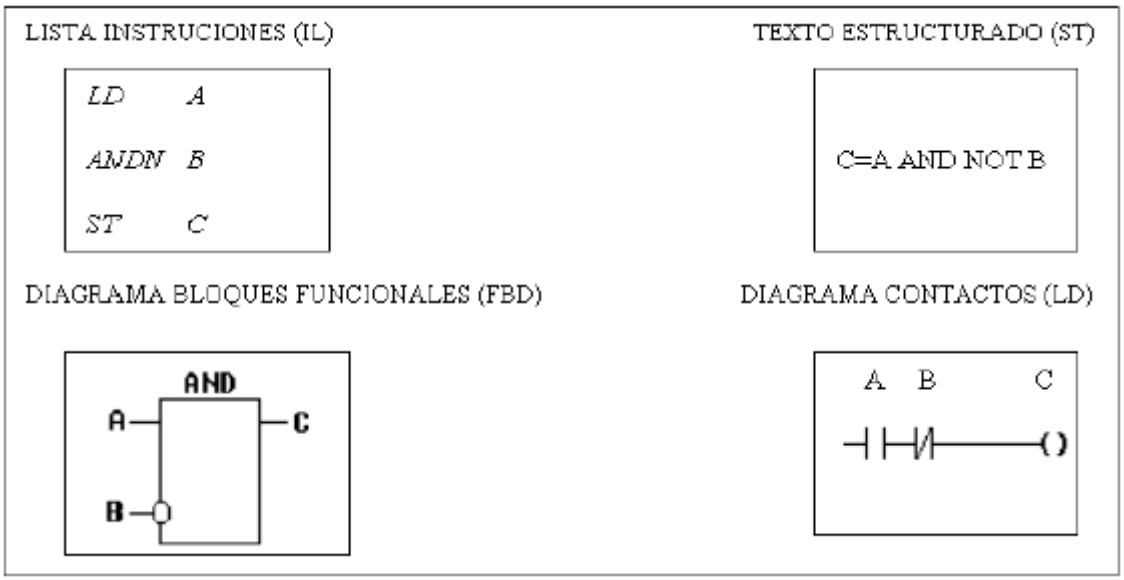


Figura 4.2. Aspecto básico de cada uno de los lenguajes de la norma.

Por último, en este apartado de la norma se especifican otros puntos relacionados con los anteriores que explican cómo deben plantearse lo problemas, como adaptarse a los lenguajes propuestos y que soluciones elegir en función del tipo de proceso.

4.2.2. Organización PLCopen

PLCopen es una organización internacional independiente, nacida en 1992 tras la publicación del estándar IEC-61131-3, formada por distintos fabricantes de PLCs, profesionales de ingeniería de software y distintas instituciones repartidas por todo el mundo. Puesto que los miembros no provienen de un sector único, sino que existe gran pluralidad, se tiene una visión global de las distintas necesidades en temas de automatización, coincidiendo todos ellos en un objetivo común: la importancia de la estandarización para reducir costes y obtener aplicaciones más eficientes y seguras. Para conseguirlo, se han publicado librerías de programación, manuales y recomendaciones en todos los ámbitos de los sistemas de control. Dada la cantidad de empresas inmersas en este proyecto, muchas de ellas de gran envergadura han conseguido que la norma y las directrices propuestas por ellos hayan sido aceptadas, una tarea que años atrás, cuando las empresas realizaban una fidelización “obligada” a sus clientes, parecía imposible.

4.3.- HERRAMIENTAS DE IMPORTACIÓN DE CÓDIGO EXTERNO

Una de las grandes debilidades del estado del sector de los controladores lógicos programables hasta la aparición de la norma IEC-61131 y la organización PLCopen era la imposibilidad de reutilizar programas o partes de los mismos (librerías, funciones, etc.) entre distintos fabricantes. La norma IEC está enfocada a mejorar los aspectos de desarrollo de software de control, sin entrar en cómo han de ser las herramientas de programación de cada uno de los fabricantes (por ejemplo, TIA Portal o RSLogix 5000). Por eso, en este aspecto la organización PLCopen ha tomado la iniciativa, proponiendo un modelado basado en XML (*Extensible Markup Language)* llamado TC6. Aquí se especifican las directrices necesarias para que un proyecto en su conjunto -librerías, POUs, configuraciones…- sean reconocidas desde distintos entornos de desarrollo, independientemente del fabricante. El objetivo de esto es conseguir código reutilizable de forma que sea más eficiente (el código ya podría estar testado en numerosos proyectos anteriormente) y seguro. Este proyecto de intercambio de información, no se queda en la importación de código fuente en los lenguajes propuestos por la norma IEC-61131-3, sino que PLCopen pretende que mediante un texto XML puedan compartirse datos acerca de visualizaciones HMI, documentación, herramientas de compilación, comunicaciones, simulaciones y toda la información contenida en un proyecto de control.

En cuanto al ámbito de las fabricantes del sector de la automatización, en este proyecto se analizarán las herramientas, posibilidades y limitaciones que Siemens y Rockwell ofrecen a este respecto. Cabe destacar que, aunque los esfuerzos de PLCopen son muy importantes y que las grandes compañías de PLCs pertenecen a esta organización, no siempre se aceptan o se complacen todas las especificaciones establecidas. Sin embargo, aunque con ciertas discrepancias con respecto al estándar, si es posible la importación, al menos, de parte de un proyecto de automatización.

4.3.1. Siemens y TIA Portal

Por un lado, Siemens ofrece una API (*Application Programming Interface*) denominada TIA Openness que permite la creación, gestión, modificación de un proyecto desde aplicaciones externas. De forma sencilla, pueden importarse y exportarse en formato XML elementos de un proyecto como gráficos, bloques, tablas de variables, etc. Además,

para aplicaciones más complejas pueden llegar a crearse rutinas para creación de proyectos completos a partir de herramientas como Visual Studio. No obstante, esta API no es sencilla[1] de utilizar y requiere altos conocimientos de programación.

```
<Parts>
  <Access Scope="LocalVariable" Type="Bool" UId="24">
  <Access Scope="LocalVariable" Type="Bool" UId="28">
  <Part Gate="Contact" UId="23">
  <Part Gate="Coil" UId="27">
</Parts>
<Wires>
  <Wire>
    <Powerrail />
    <NameCon UId="23" Name="in" />
  </Wire>
  <Wire>
    <IdentCon UId="24" />
    <NameCon UId="23" Name="operand" />
  </Wire>
  <Wire>
    <NameCon UId="23" Name="out" />
    <NameCon UId="27" Name="in" />
  </Wire>
  <Wire>
    <IdentCon UId="28" />
    <NameCon UId="27" Name="operand" />
  </Wire>
</Wires>
```

Figura 4

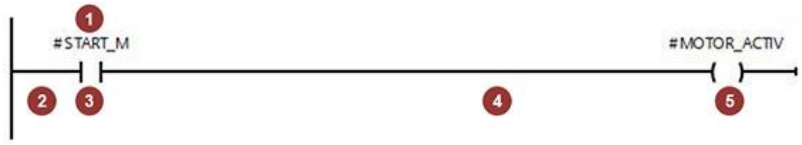


Figura 4.3. Ejemplo de importación de un texto XML para creación de rama LD.

Por otro lado, TIA Portal también admite la importación directa de tablas de variables, UDTs (tipos definidos por el usuario), funciones, bloques funcionales y OBs (identificables con programas de la norma IEC, con ciertas diferencias) a partir de ficheros de texto siempre y cuando el código se encuentre en un lenguaje de tipo texto especificado por la norma (Lista de Instrucciones y Texto Estructurado). Como condición, las variables, bloques funcionales, funciones, etc. deben especificarse tal y como se definen en la norma IEC-61131-3 y poner como extensión del texto plano "*.scl"* (para ST), "*.stl*" (para IL) y "*.udt*" (para datos definidos por el usuario), "*.db*" (bloque de datos), etc.

[1] El propio desarrollo de rutinas de modificaciones de proyectos, no la importación de código XML

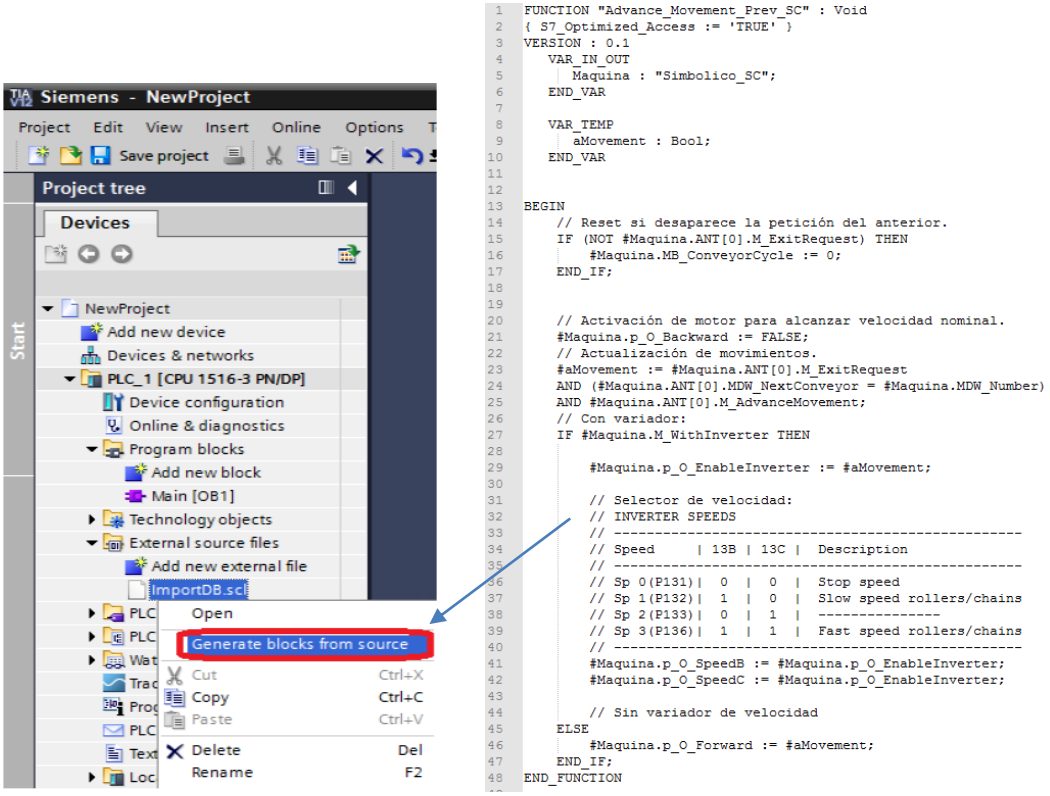


Figura 4.4. Ejemplo de importación de código externo ST (.scl).

4.3.2. Rockwell y RSLogix 5000

En cuanto a Rockwell, siguen una filosofía un poco distinta del anterior. En primer lugar, Rockwell también permite la importación y exportación de las distintas partes de un proyecto de control para mejorar la productividad y la reutilización de código. Entre los elementos reutilizables se permiten rutinas basadas en cualquiera de los lenguajes permitidos (LD, ST, FBD y SFC), UDT, tags, configuraciones, rutinas, etc. La importación y exportación se realiza a través de un fichero de texto ASCII basado en XML, denominado "*.l5x*". Por tanto, aunque existen ciertas diferencias entre el formato de este archivo y el presentado por PLCopen (TC6) y Siemens, adaptando la sintaxis existe la posibilidad de generar código reconocible por los PLCs de Allen Bradley y de esta forma obtener una aplicación multiplataforma que es el objetivo del presente proyecto. En la Figura 4.5 se muestra un ejemplo de fichero de extensión "*l5x"* con un ejemplo de programa en el lenguaje de texto estructurado.

Component	Description
<RSLogix5000Content>	Describes version and export information
<Controller>	The controller
<DataTypes>	User-defined and I/O data structures
<Modules>	Modules in the controller organizer
<AddOnInstructionDefinitions>	Add-On Instructions
<Tags>	Controller-scope tags
<Programs>	Programs
<Routines>	Ladder logic, function block diagram, sequential function chart, and structured text routines
<Tasks>	Controller tasks
<ParameterConnection>	Parameter connections
<ParameterConnections>	Program parameter connections
<Trends>	Any trend configured for the controller project
<QuickWatchLists>	All quick watch lists configured in the controller project
<CommPorts>, <CST>, and <WallClockTime>	Controller configuration objects

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <RSLogix5000Content SchemaRevision="1.0" SoftwareRevision="26.00" TargetName="My_controller"
  TargetType="Controller" ContainsContext="false" Owner="User, RA" ExportDate="Mon May 19
  08:26:42 2014" ExportOptions="DecoratedData ForceProtectedEncoding AllProjDocTrans">
  - <Controller Use="Target" Name="My_controller" ProcessorType="1756-L73" MajorRev="22"
    MinorRev="1" TimeSlice="20" ShareUnusedTimeSlice="1" ProjectCreationDate="Wed Mar 26
    13:55:49 2014" LastModifiedDate="Mon May 19 08:26:35 2014"
    SFCExecutionControl="CurrentActive" SFCRestartPosition="MostRecent" SFCLastScan="DontScan"
    ProjectSN="16#0000_0000" MatchProjectToController="false" CanUseRPIFromProducer="false"
    InhibitAutomaticFirmwareUpdate="0" PassThroughConfiguration="EnabledWithAppend"
    DownloadProjectDocumentationAndExtendedProperties="true">
    <RedundancyInfo Enabled="false" KeepTestEditsOnSwitchOver="false"
      IOMemoryPadPercentage="90" DataTablePadPercentage="50" />
    <Security Code="0" ChangesToDetect="16#ffff_ffff_ffff_ffff" />
    <SafetyInfo />
    <DataTypes />
    - <Modules>
      - <Module Name="Local" CatalogNumber="1756-L73" Vendor="1" ProductType="14"
          ProductCode="94" Major="22" Minor="1" ParentModule="Local" ParentModPortId="1"
          Inhibited="false" MajorFault="true">
          <EKey State="ExactMatch" />
        - <Ports>
          - <Port Id="1" Address="0" Type="ICP" Upstream="false">
              <Bus Size="7" />
            </Port>
          </Ports>
        </Module>
      </Modules>
```

Figura 4.5. Elementos de importación/exportación en "*.l5x*" y ejemplo.

5. Desarrollo de la aplicación

En este apartado de describirá todos los puntos necesarios para comprender el resultado final del proyecto.

5.1.- ASPECTOS PREVIOS

Actualmente, cada fabricante de elementos de control ofrece sus herramientas propietarias de desarrollo para trabajar con sus productos. Esto radica en que los proyectos implementados para una cierta plataforma contarán con ciertas características que dificultarán su utilización en sistemas de otros fabricantes al original, destacando principalmente:

- Cada fabricante dispone de sus propios lenguajes de programación que, aunque son similares en cuanto a usabilidad, suelen ser incompatibles por razones de sintaxis, etc.
- Los mecanismos específicos de cada aplicación (comunicaciones, funciones de tiempo real, temporizadores) son distintos entre distintos sistemas.
- Típicamente, en la automatización de procesos ha habido reacciones de fidelización dificultando la compatibilidad entre dispositivos externos.

Si estos puntos los trasladamos a los sistemas típicos de Mecalux, se encontrará:

- Siemens utiliza TIA Portal como IDE (*Integrated Development Environment*) de programación.
- Rockwell utiliza RSLogix como herramienta de desarrollo.
- El *SoftPLC* propietario de Mecalux (Galileo) emplea Designer para implementar soluciones de control.

Aunque a priori no es posible que todos estos sistemas compartan proyectos comunes -ya que cada uno utiliza sus bloques de organización de programa determinados- gracias a los esfuerzos de la organización PLCOpen sí que es posible reutilizar ciertos elementos entre

los distintos IDE, compartiendo el mismo tipo de formato en ciertas estructuras. De la misma manera, la adopción del estándar internacional IEC-61131-3, y puesto que a los fabricantes les conviene recibir la acreditación de concordancia con dicha norma, ha conseguido que por lo menos se adopten los mismos lenguajes de programación (aunque muchas veces con pequeños matices). Gracias a esta norma, y a su reconocimiento por los fabricantes de PLCs más importantes –entre los que se incluyen Siemens y Rockwell- se dispone de unos pilares fuertes para comenzar a desarrollar aplicaciones multiplataforma.

5.2.- ELECCIÓN DEL LENGUAJE DE PROGRAMACIÓN

Hasta este momento Galileo utilizaba un lenguaje de programación denominado XanaO2, basado en C/C++ con ciertas características limitadas para ajustarse al comportamiento necesario para un sistema de control y facilitar su utilización, mantenibilidad y desarrollo. En adicción, como estructura principal de la programación de aplicaciones de control se empleaban grafos basados en diagramas GRAFCET (*Graphe Fonctionnel de Commande Etape Transition)* con ciertas características modificadas. Como se ha comentado, la norma IEC-61131-3 especifica cuatro lenguajes de programación, no obstante, no todos ellos pueden ser utilizados para el propósito de Mecalux. La discriminación será la siguiente:

- Los lenguajes FBD (*Function Block Diagram*) y LD (*Ladder Diagram*) son lenguajes de tipo gráfico que provocarían que el desarrollo de los programas de control se hiciera lenta y tediosa, rompiendo compatibilidad por completo con XanaO2 el cual es un lenguaje más complejo, completo y de alto nivel.
- El lenguaje IL (*Instruction List*) es más similar a XanaO2 que los anteriores, y se podría llegar a programar de forma más eficiente. Sin embargo, al ser poco intuitivo provocaría mayor esfuerzo en formación, lentitud de desarrollo y menor legibilidad. Por esta razón también se ha descartado
- El lenguaje ST (*Structured Text*) es un lenguaje de alto nivel cuya premisa principal radica en la estructuración de código y datos. De esta forma se asemeja a otros lenguajes utilizados para desarrollo software (aunque más simple y primitivo, suficiente para aplicaciones de control) como Pascal y C. Es la opción ofrecida por el estándar que más se asemeja a XanaO2 y que garantiza una fácil adaptación, mantenimiento y mejor

acogida por parte de los programadores actuales de Mecalux. Además, aunque por el momento continua la inercia de LD o IL, el sentido común dicta que poco a poco y en el futuro se dejarán de lado reemplazados por este lenguaje.

En cuanto a la arquitectura de la lógica principal del programa, podría pensarse en utilizar grafos como se hacía hasta el momento en Designer/Galileo. Esto podría ser completamente viable ya que en la norma IEC-61131-3 se recoge el lenguaje SFC (*Sequential Function Chart*). Por tanto, dado que el esquema de grafo de Galileo no seguía exactamente lo especificado en la norma, podría ser completamente factible.

Para valorar la incorporación de este elemento, se valoró por un lado la dificultad que supone la traducción de este tipo de lenguajes a ciertas directrices importables por las herramientas de desarrollo de Siemens y Rockwell, la cual sería muy elevada. Por otro lado, dada la complejidad de las máquinas del catálogo de Mecalux que serán generadas a partir de esta herramienta (por el momento no se incluyen transelevadores y electrovía). Puesto que estas razones por si solas no son suficientes para descartar la utilización de grafos, dadas sus grandes ventajas en el seguimiento de código *online*, se pasó a la realización de pruebas para evaluar su eficiencia.

5.2.1. Tamaño en generación

Como primera premisa, debe asumirse que el proceso de generación de código multiplataforma y sobre todo a partir de un estándar de máquinas con numerosas posibilidades de configuración, producirá irremediablemente un aumento del tamaño del programa con respecto a soluciones basadas en un proyecto específico. Esto es debido a la adaptación de un código genérico a una estructura sencilla y admitida por los tres sistemas comentados, siempre siguiendo una estructura orientada a objetos para aumentar la legibilidad, la abstracción y el encapsulamiento de partes diferenciadas (Ver apartado 5.5). Por eso, dado que cierta cantidad de memoria será perdida de forma asumible, deberá evaluarse de forma muy importante la memoria que utiliza la arquitectura elegida de los programas de control. Esto es fundamental ya que un pequeño ahorro en una máquina, puesto que en una instalación puede llegar a componerse de cientos de elementos supondrán grandes diferencias en la solución aceptada, pudiendo optar por menor número de sistemas de control, controladores con menos memoria, etc. De esta forma, se procede a comparar la memoria consumida por un programa basado en lenguaje SCL y otro ST. Para la realización

de las pruebas se utilizará la herramienta de desarrollo de Siemens: TIA Portal V13. En cuanto a la CPU, se ha seleccionado un dispositivo sencillo, concretamente el modelo Simatic S7-315-2DP:

En primer lugar, se comprueba la memoria utilizada por un programa vacío. Para ello se ha procedido a cargar el bloque OB1 generado por defecto en la creación de un proyecto genérico. Los resultados obtenidos (a través de la herramienta de depuración *online* de TIA Portal) se presentan en siguiente figura donde las columnas se corresponden con la memoria de carga RAM, memoria de trabajo y memoria remanente de datos.

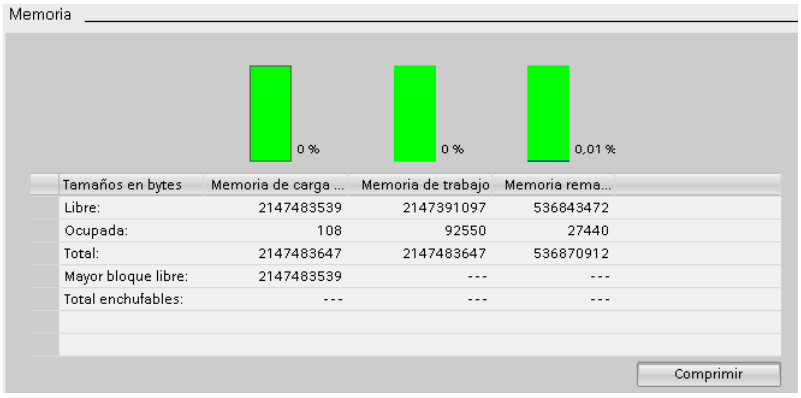


Figura 5.1. Memoria empleada en proyecto vacío.

En segundo lugar, se procede a realizar un programa en lenguaje SFC (*Graph* en TIA Portal) basado en un transportador simple de rodillos. Además, se incluyen ciertos métodos en texto estructurado (SCL en TIA) con la función de añadir instrucciones al código, incluyendo las estructuras típicas (IF-ELSE, FOR, etc.). Los resultados son los siguientes:

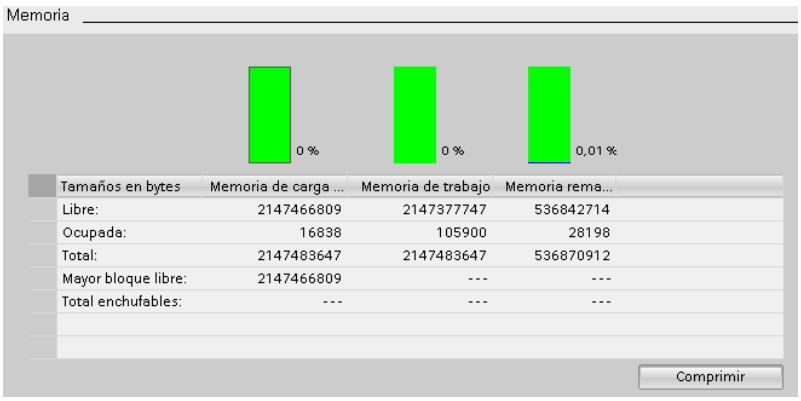


Figura 5.2. Memoria empleada por programa *Graph*.

Por último, se desarrolla un programa en lenguaje estructurado idéntico al implementado anteriormente en lenguaje SFC. Para ello, en líneas generales, se ha traducido la estructura del grafo, empleando las funciones y variables utilizadas en el caso anterior. Una vez cargado en el controlador se han obtenido los siguientes resultados.

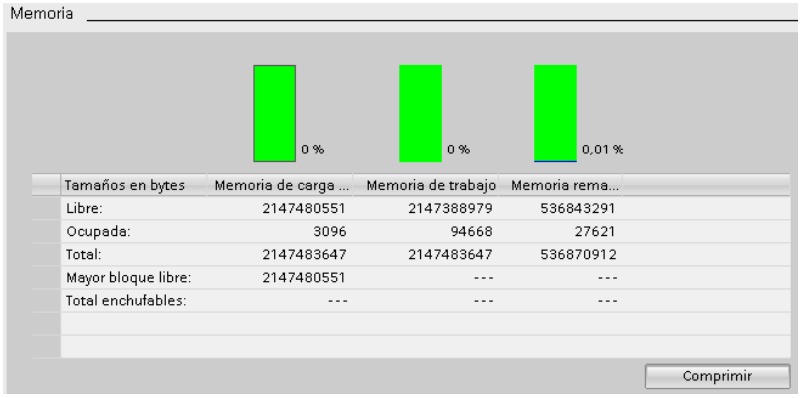


Figura 5.3. Ocupación de memoria por programa SCL.

Puesto que se ha partido de un programa similar para las pruebas, puede suponerse que el aumento de tamaño producido por el uso del lenguaje SFC depende únicamente del PLC. Como se puede observar, la implementación por medio de *Graph* supone una ocupación de memoria cinco veces superior que su equivalente SCL (texto estructurado). Sumado a lo anterior, cabe destacar que la programación de este último se ha hecho ciñéndose estrictamente a la estructura *Graph* (traducción por medio de máquina de estados) y, por tanto, mediante una optimización del programa podría reducirse considerablemente los valores empleados de memoria.

Una limitación adicional que se presenta actualmente respecto a la programación GRAPH radica en la incompatibilidad de este lenguaje con la serie S7-1200:

Programming language	S7-1200	S7-1500
Ladder (LAD)	✓	✓
Function block diagram (FBD)	✓	✓
Structured control language (SCL)	✓	✓
Graph	x	✓
Statement list (STL)	x	✓

Figura 5.4. Incompatibilidad GRAPH y serie S7-1200.

Por tanto, aunque es posible y se ha planteado una solución basada en grafos, esta deberá ser únicamente para la programación en el entorno de desarrollo multiplataforma de Designer. Una vez se desee exportar para PLC, se realizará una traducción automática de ese grafo a texto estructurado (esta ampliación está en fase de estudio).

5.3.- ELECCIÓN DE CONTROLADORES COMPATIBLES

Habitualmente Mecalux ha utilizado distintos modelos de PLCs. Por un lado, las instalaciones europeas que rechazaban el uso del sistema de control Galileo, suelen preferir Siemens. Para estos casos, en función del tamaño de la instalación y las necesidades de la aplicación, se utilizaban controladores de la gama 300 y 400. Por otro lado, para casos similares en el mercado americano, se ha optado por autómatas de Rockwell, los cuales variaban entre modelos de CompactLogix y ControlLogix.

A partir de este momento, una vez que se pretende desarrollar programas estándar para distintas plataformas, debe asumirse que el catálogo de soluciones ofrecidas debe reducirse a aquellos controladores que cumplan una serie de especificaciones. Para esto debe tenerse en cuenta que no se podrán utilizar dispositivos obsoletos, que no soporten ni acepten las directrices de la norma IEC-61131 y el lenguaje de programación de texto estructurado. Por esta razón, para el caso de soluciones basadas en Siemens se restringirá la utilización de la serie 200/300/400 para dejar paso a las nuevas opciones basadas en la serie Simatic S7 1200 (para requisitos simples) y 1500 para instalaciones más exigentes en cuanto a tamaño, comunicaciones, etc. Por supuesto dentro de estas dos gamas, existen multitud de subproductos con diferentes características (memoria de programa, velocidad de procesamiento, conexiones socket simultaneas) y precio. Estos cuentan con los cinco lenguajes propuestos por la norma, admitiendo las funciones presentadas por la norma (rotación de bits, operadores lógicos, temporizadores, etc.) y los tipos de datos admitidos. En cuanto a la estructura de los programas, aunque se desvían ligeramente del estándar, básicamente se utilizan los mismos bloques y la traducción de una a otra especificación no es muy complicada.

En el momento de desarrollo del presente proyecto, para la realización de pruebas se utilizó el modelo S7-1516 PN/DP. La razón de esta selección ha sido la disposición de interfaces Profinet y Profibus integradas, sin necesidad de utilizar módulos de comunicaciones

adicionales, los cuales son los buses de campo más utilizados por Mecalux y porque cuenta con una gran eficiencia en cuanto a comunicaciones (muy utilizadas dado el intercambio de datos con el sistema de gestión de almacén). Cuenta con las siguientes características de procesamiento:

CPU	Performance segment	PROFIBUS interfaces	PROFINET interfaces	Work memory	Processing time for bit operations
CPU 1511-1 PN	Standard CPU for smaller to mid-range applications	--	1	1.15 MB	60 ns
CPU 1511F-1 PN	Fail-safe CPU for smaller to mid-range applications	--	1	1.23 MB	60 ns
CPU 1513-1 PN	Standard CPU for mid-range applications	--	1	1.8 MB	40 ns
CPU 1513F-1 PN	Fail-safe CPU for mid-range applications	--	1	1.95 MB	40 ns
CPU 1515-2 PN	Standard CPU for mid-range to large applications	--	2	3.5 MB	30 ns
CPU 1515F-2 PN	Fail-safe CPU for mid-range to large applications	--	2	3.75 MB	30 ns
CPU 1516-3 PN/DP	Standard CPU for high-end applications and communication tasks	1	2	6 MB	10 ns
CPU 1516F-3 PN/DP	Fail-safe CPU for high-end applications and communication tasks	1	2	6.5 MB	10 ns
CPU 1517-3 PN/DP	Standard CPU for demanding applications and communication tasks	1	2	10 MB	2 ns
CPU 1517F-3 PN/DP	Fail-safe CPU for demanding applications and communication tasks	1	2	11 MB	2 ns
CPU 1518-4 PN/DP	Standard CPU for high-performance applications, demanding communication tasks and very short reaction times	1	3	24 MB	1 ns
CPU 1518F-4 PN/DP	Fail-safe CPU for high-performance applications, demanding communication tasks and very short reaction times	1	3	26 MB	1 ns

Figura 5.5. Autómata para la realización de pruebas de este proyecto.

Para la generación en mercado americano, se seleccionará una serie de autómatas de Rockwell. En este caso, no se ha adquirido todavía un controlador específico para la realización de pruebas, aunque ya se ha hecho una búsqueda y petición de presupuesto a Allen Bradley. Hasta este momento, para desarrollar la aplicación se ha utilizado la herramienta de simulación que Rockwell ofrece: RSLogix Emulator 5000. Con respecto al PLC seleccionado, se ha hecho una investigación en la red para localizar el que mejor se corresponde con las necesidades del proyecto. Uno de los puntos clave es que este fabricante tiene fuertes restricciones con respecto a los modelos compatibles con su software, quedando desactualizado y teniendo que adquirir una nueva licencia (no siendo muy económicas). Dada la versión de que se dispone de RSLogix, por el momento se limitará la búsqueda a las gamas CompactLogix 5370 y ControlLogix 5570. Hay que destacar que la implementación interna de Rockwell tanto en recursos utilizados como en tiempo de procesamiento está

menos optimizada que Siemens. Una comparación entre ambas, con los mismos programas, demuestran que para conseguir un rendimiento igual entre un PLC de Rockwell y el Simatic S7-1516 adquirido, debe seleccionarse la CPU ControlLogix 1756-L73, la cual tras la petición de presupuesto supone una solución casi tres veces más cara que la alemana. Sin contar que Rockwell no fabrica interfaces Profibus, teniendo que adquirir una tarjeta de expansión externa (por ejemplo, Prosoft).

Aunque por el momento se restringe la generación para este tipo de dispositivos, no se descarta aumentar esta selección en un futuro siempre y cuando los controladores de destino cumplan con especificaciones elegidas de partida. Puesto que la aceptación de la norma IEC-61131-3 es cada vez más creciente, se augura un aumento de autómatas compatibles, cada vez de un catálogo más amplio de fabricantes.

5.4.- ESQUEMA GENERAL DE LA APLICACIÓN

La nueva herramienta de programación cuenta con ciertas partes bien definidas, todas ellas imprescindibles para garantizar el resultado esperado. En primer lugar, deberá existir un entorno donde programar las máquinas de una instalación. Esto se relaciona con la propia interfaz de usuario del programa de desarrollo y la arquitectura que seguirán los programas implementados. En segundo lugar, una vez que se tiene el código necesario para una máquina, existirá una etapa de generación. Aquí, inicialmente se comprobarán errores de sintaxis y las normas básicas del lenguaje. Luego, se realiza una síntesis de todo el código utilizado en un proyecto en un mismo archivo, es decir, a partir de las distintas máquinas de la instalación, formada por distintas variables, métodos, etc. se generará un único fichero de texto con un formato específico que contiene todo el código del proyecto en función del tipo de destino (Rockwell, Siemens o Galileo). A grandes rasgos podríamos decir que se trata de una concatenación de datos y funciones siguiendo unos encabezados para hacerlo reconocible por el controlador de destino. Para el caso de autómatas clásicos, a partir de este punto se importarían los archivos generados en las herramientas de desarrollo de cada uno de los fabricantes, RSLogix o TIA Portal. En el caso de Galileo, el proceso continúa con la compilación del archivo generado. En esta etapa, el código fuente del programador, se traducirá a código máquina. Por último, se procede con la fase de empaquetado que genera

el archivo ejecutable por Galileo, conteniendo instrucciones de código máquina e información para depuración y vista online).

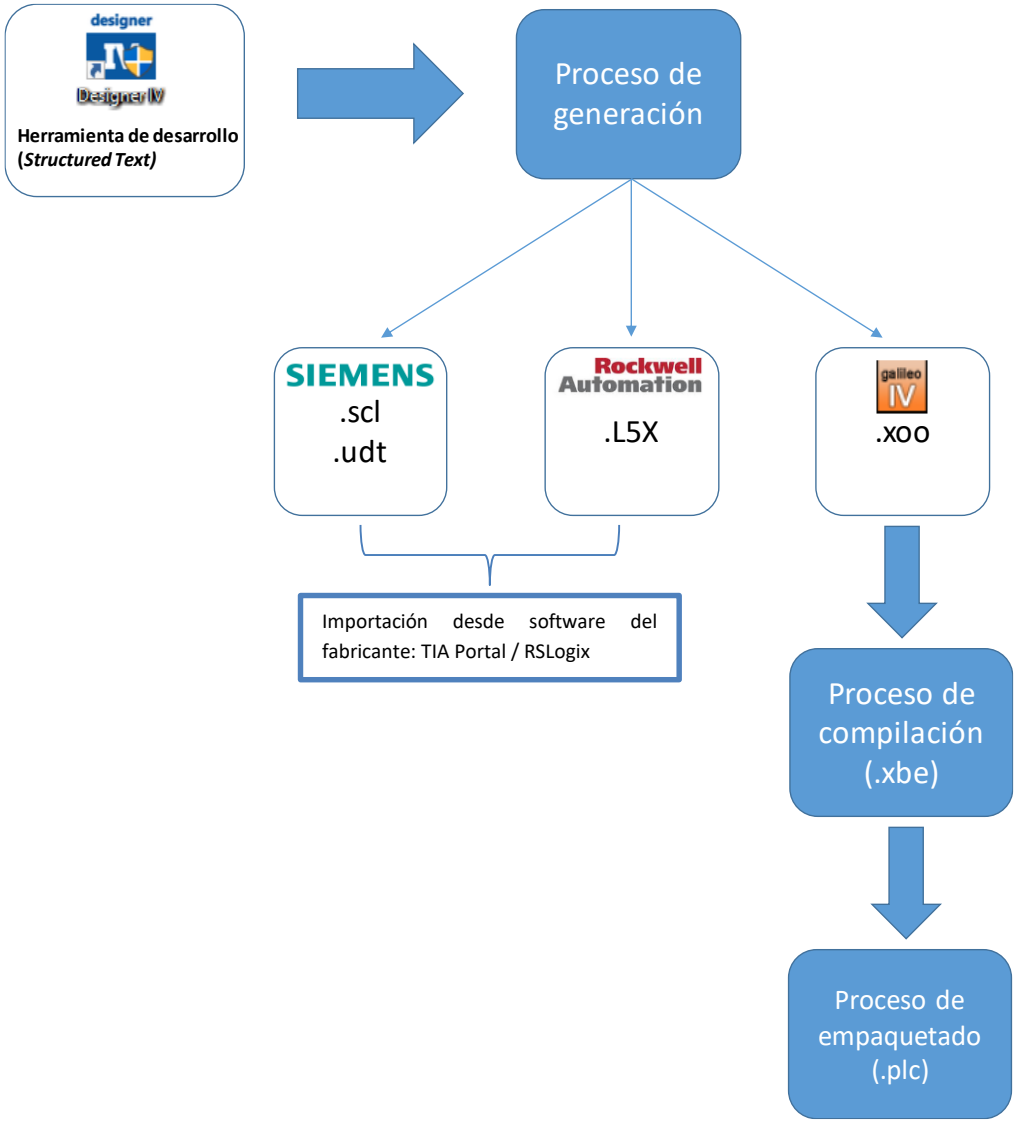


Figura 5.6. Esquema del proceso de generación multiplataforma.

5.5.- ARQUITECTURA DEL PROGRAMA

Aunque la filosofía del tipo de programación no suele ser una de las mayores prioridades a la hora de desarrollar código de control, para esta aplicación se utilizará el paradigma de programación orientada a objetos. Puesto que se quiere disponer de un

repositorio cerrado y probado con las máquinas típicas de una instalación ofrecida por Mecalux, este tipo de solución ofrece numerosas ventajas. Por un lado, se puede enfocar la solución en elementos independientes que disponen de sus variables y métodos y que encapsulan el comportamiento de una máquina concreta. Por otro lado, pueden utilizarse las propiedades de herencia, facilitando la modificación de ciertas máquinas y mejorando el encapsulamiento.

5.5.1. Secuenciador

En primer lugar, se ha denominado secuenciador a la unidad de comportamiento de una máquina concreta. Por ejemplo, la rutina esperada en un transportador de rodillos estará definido y codificado en un secuenciador, mientras que un elevador de paletas a distinto nivel, estará programado en un secuenciador diferente. A continuación, se procederá a detallar los distintos elementos que componen cada uno de los secuenciadores.

En primer lugar, un secuenciador estará formado por distintas variables utilizadas a lo largo del código. Estas agrupaciones de datos, serán definidas como un UDT (*User Defined Type*) en los distintos autómatas compatibles. En cuanto al tipo de parámetros, se admitirán aquellos aceptados por la norma IEC-61131-3, incluyendo temporizadores estándar. En Designer se utilizará un formulario para la introducción de parámetros, mientras que su generación a PLC será un UDT estándar.

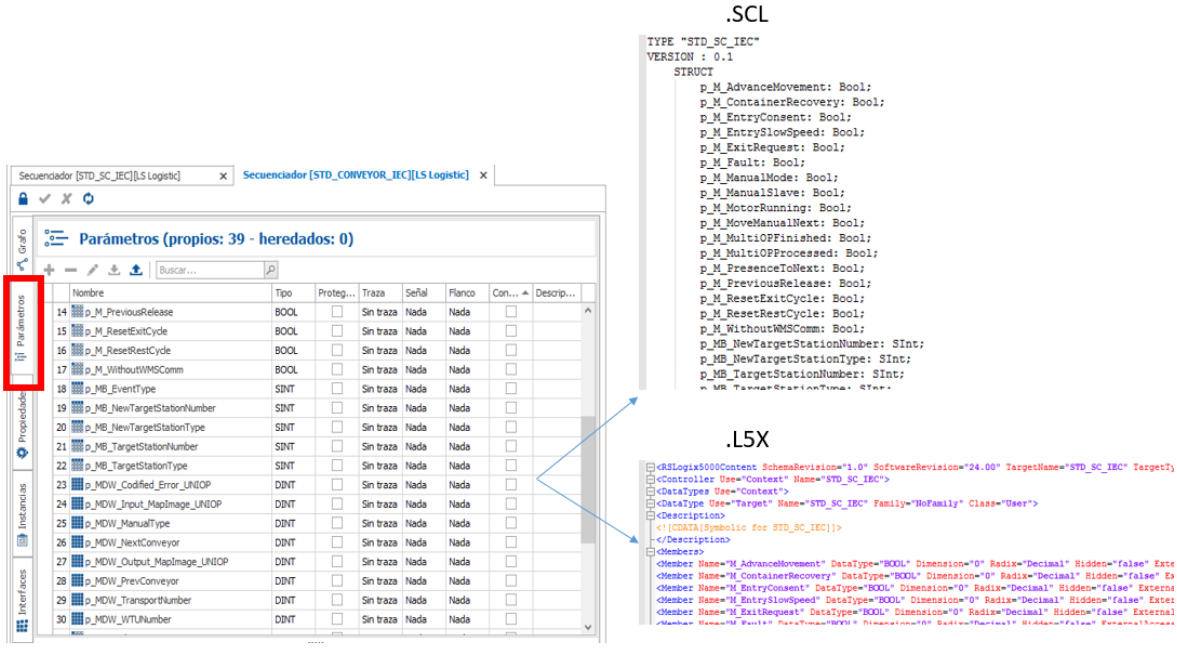


Figura 5.7. Parámetros de un secuenciador.

Los siguientes elementos básicos que componen un secuenciador son los métodos. Pueden entenderse como las distintas funciones que componen la lógica de control de una máquina concreta, dividiéndose en varios tipos:

1. Por un lado, se dispone de funciones tal y como se definen en la norma IEC. Estas estructuras no cuentan con memoria, es decir, las variables declaradas dentro de ellas son temporales, destruyéndose al final de cada ciclo de programa, no necesitando declarar una instancia de ella para su utilización. Estas funciones admiten cualquier tipo de parámetros (tanto por valor como por referencia) definidos en el estándar y pueden devolver, también cualquier tipo de dato. Su llamada se puede realizar desde cualquier parte de código, incluyendo otras funciones.
2. Además de las anteriores, se dispone de un tipo de funciones denominado como *métodos de acción.* Estas, vienen heredadas del antiguo Galileo (basado en otras estructuras similares a los SFC). Su característica principal es que son las funciones que ejecutará el autómata de forma ordenada:
    a. Función de arranque en frío: ejecutada al pasar el PLC de *Stop* a *Run*. Solo dura el primer ciclo de *scan*.
    b. Función de arranque en caliente: tras un reinicio rápido del PLC. Solo dura el primer ciclo de *scan*.
    c. Función anterior: función que se ejecuta en primer lugar en cada ciclo de programa. Típicamente, puede utilizarse para volcar entradas en marcas, calcular distancias (por ejemplo, máquinas con movimientos medidos con telémetro), comprobación de estado de módulos y seguridades, etc.
    d. Función principal. A diferencia de los anteriores, esta función será un bloque de función, lo que quiere decir que contendrá memoria -se pueden declarar variables estáticas- y contiene como tal, una estructura del tipo de datos de esta máquina particular. Es decir, al crear una máquina basada en un secuenciador, se instanciará un FB de este bloque, instanciando, por tanto, los parámetros definidos en el secuenciador (simbólico). Se ejecuta tras la función anterior.

e. Función posterior: ejecutada tras el método principal y que típicamente se utiliza para gestionar información de variables (por ejemplo, para paneles de operador) o volcar marcas de programa en salidas de periferia.

El orden de ejecución se representa en la siguiente Figura:

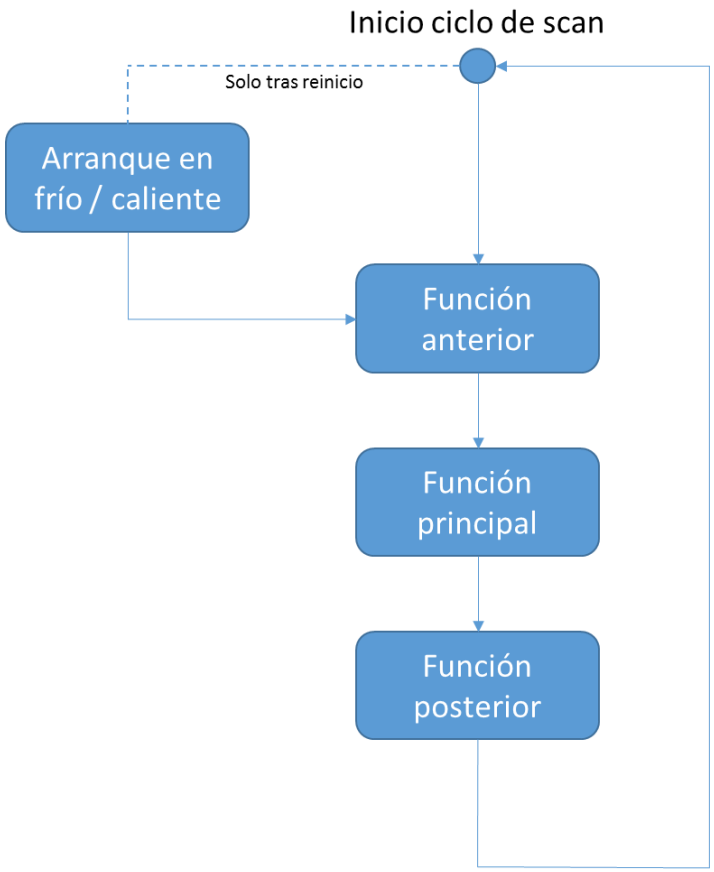


Figura 5.8. Orden de ejecución de un secuenciador.

Por último, existe otro tipo de métodos: *las funciones abstractas*. Como en los lenguajes de programación de ingeniería de software, estos elementos no se encuentran implementados en el propio secuenciador. Sino que cada una de las máquinas instanciadas implementarán este método de forma independiente. De esta forma, se garantiza que, aunque el código base sea el mismo en las distintas instancias de un mismo secuenciador, cada una pueda emplear cierto código personalizado en función de las necesidades de cada una de ellas. El lugar donde se realiza esta acción se denomina simbólico, y se expondrá más adelante. En cuanto

a la forma de obtener este comportamiento en un PLC, se ha tenido que elegir una solución no muy llamativa pero sí funcional, dada la simplicidad de los lenguajes de programación de los autómatas.

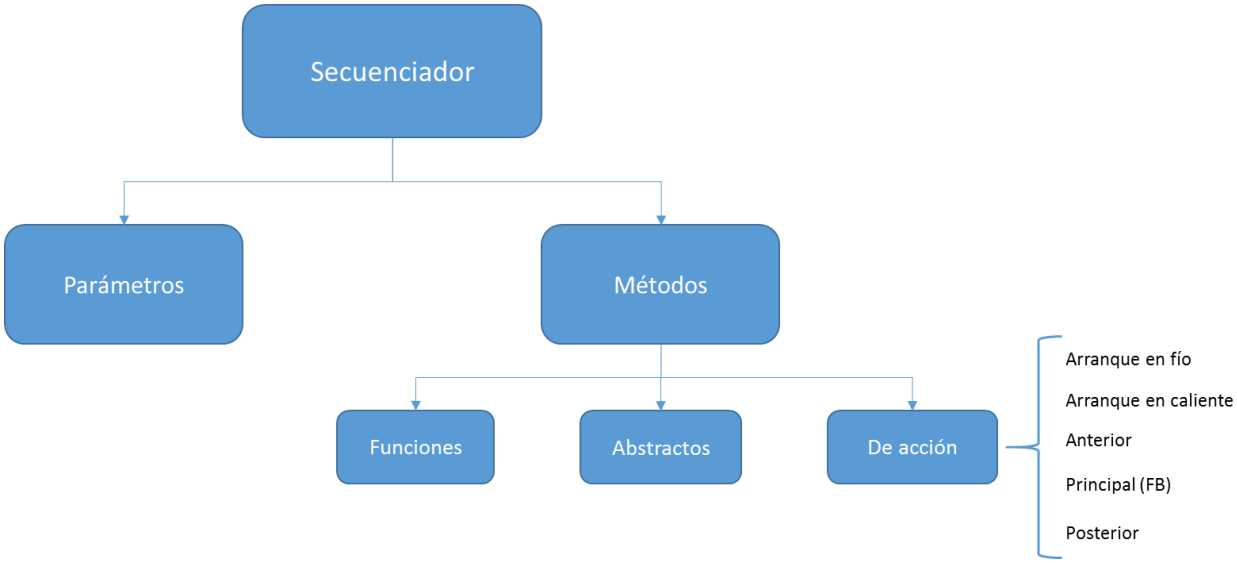


Figura 5.8. Elementos de un secuenciador.

A continuación, se presentará, con más detalle, la arquitectura de los secuenciadores en la solución de autómatas programables convencionales.

Parámetros

Como se ha mencionado, cada secuenciador cuenta con unos parámetros concretos. Estos se instancian junto con la máquina basada en el secuenciador, compartiendo todas ellas la misma plantilla. La solución de este aspecto radica en la generación de un UDT específico para cada secuenciador.

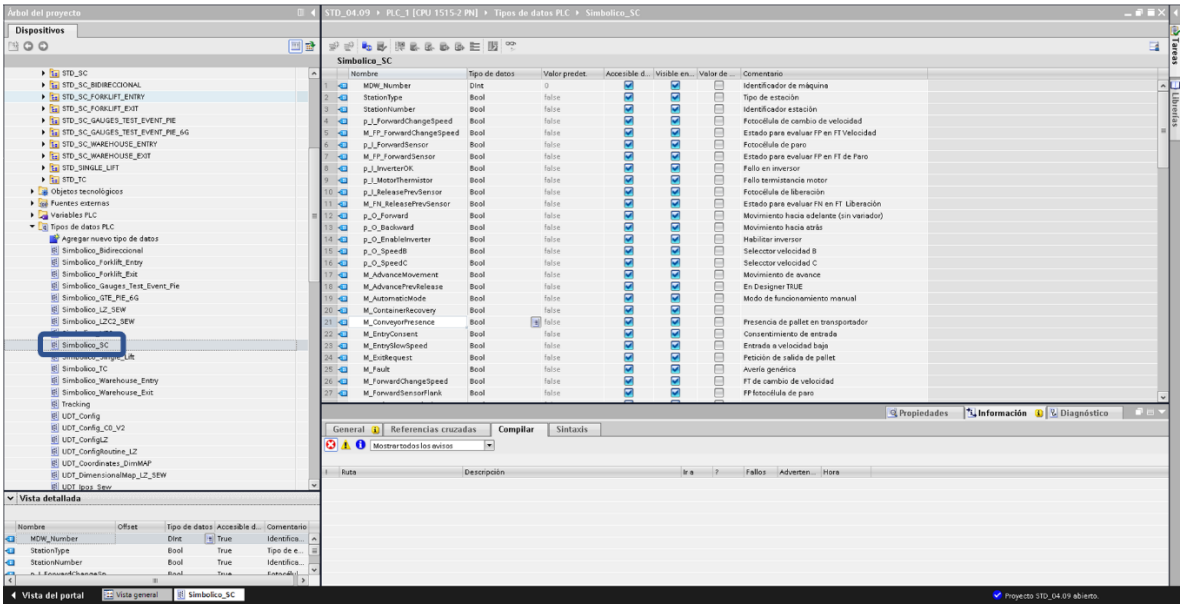


Figura 5.9. Parámetros de secuenciador en PLC.

Métodos

Los métodos se crearán con los mismos tipos de retorno y parámetros de entrada que los especificados en Designer. Segenerará una FC por cada secuenciador con el identificar "*NombreMetodo_Secuenciador"* para evitar conflictos de nombres entre secuenciadores con métodos con el mismo nombre. Todos los métodos recibirán un objeto del tipo UDT de los parámetros del secuenciador, de esta forma cada función actuará sobre las variables específicas de cada máquina.

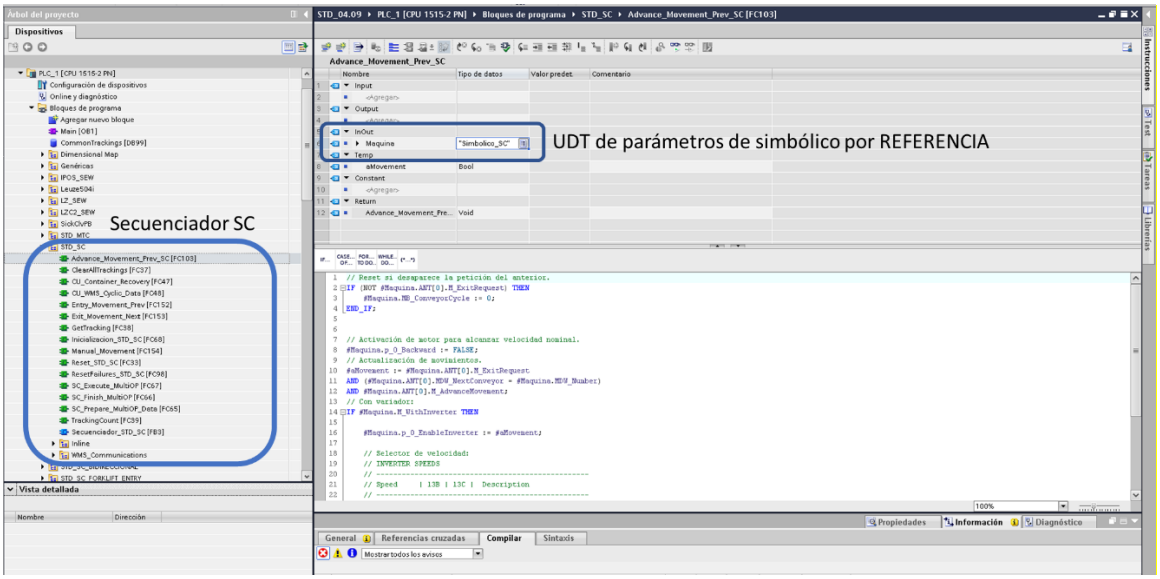


Figura 5.10. Métodos de secuenciador en PLC.

En el caso de las *funciones abstractas* la resolución se realizará mediante la evaluación de una variable común para todas las máquinas: el número de máquina, un identificador único para cada una de ellas. De esta forma, el código específico de cada podrá ser volcado en una función conjunta con un CASE que evalué dicha variable, y así ejecutar cada una de ellas un código distinto, aunque todas ellas estén basadas en el mismo secuenciador.

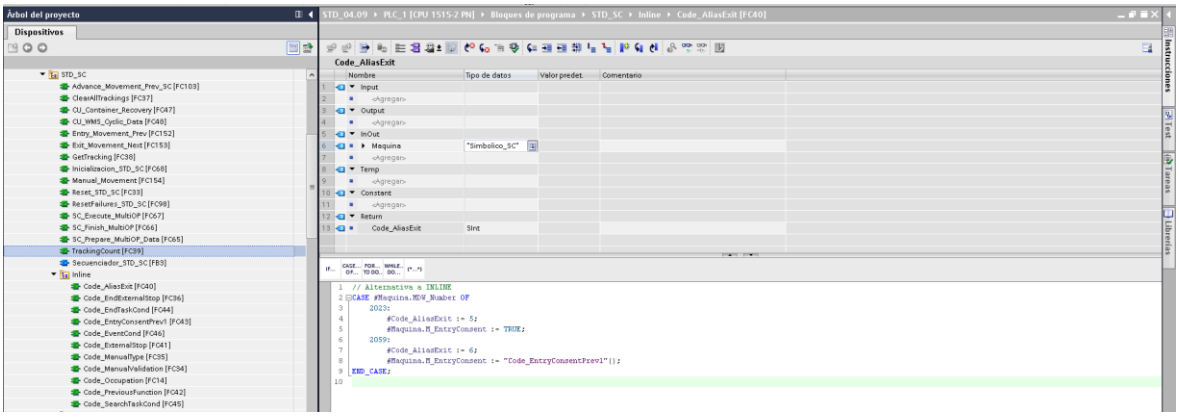


Figura 5.11. Funciones abstractas de secuenciador en PLC.

Timers

Para utilizar los *timers* se han creado funciones personalizadas, de esta forma se garantiza que puedan ser creados tanto para Rockwell como para Siemens, particularizando para cada uno de los casos. Se crearán FB (necesitan instancias) para cada uno de los definidos en los parámetros del secuenciador. Esta instancia pertenecerá al FB principal del secuenciador (a modo de multi-instancia) de forma que cada secuenciador cree los temporizadores utilizados. Se admiten los siguientes: TON, TOF y TP. Estas funciones se han denominado EvalTON, EvalTOF y EvalTP y reciben un objeto IEC_TON/TOF/TP, la condición de habilitación y el tiempo. Devuelven la salida del temporziador.

Flancos

Para los flancos se sigue una metodología similar a la de los temporizadores. Se ha creado una función R_TRIG y F_TRIG (aunque se llaman igual que en la norma, son funciones específicas) para garantizar el funcionamiento en Siemens y Rockwell (en Siemens se admiten estas funciones de la norma), en Rockwell hay que hacerlo codificándolo con una variable auxiliar.

Bloque de función principal

El bloque de función principal equivale a la instancia de cada uno de los secuenciadores. Contiene el UDT de los parámetros del secuenciador en la zona de variables estáticas del mismo. Esta variable siempre se llamará Machine y se pasará como parámetro de referencia en todas las funciones del secuenciador. Así se permitirá modificar la estructura desde cualquier función y garantizar que cada máquina ejecuta el código sobre su propia zona de datos. En el programa principal (rutina en RSLogix o OB1 en Siemens) habrá tantos FB de este tipo declarados como máquinas instancias de ese secuenciador existan.

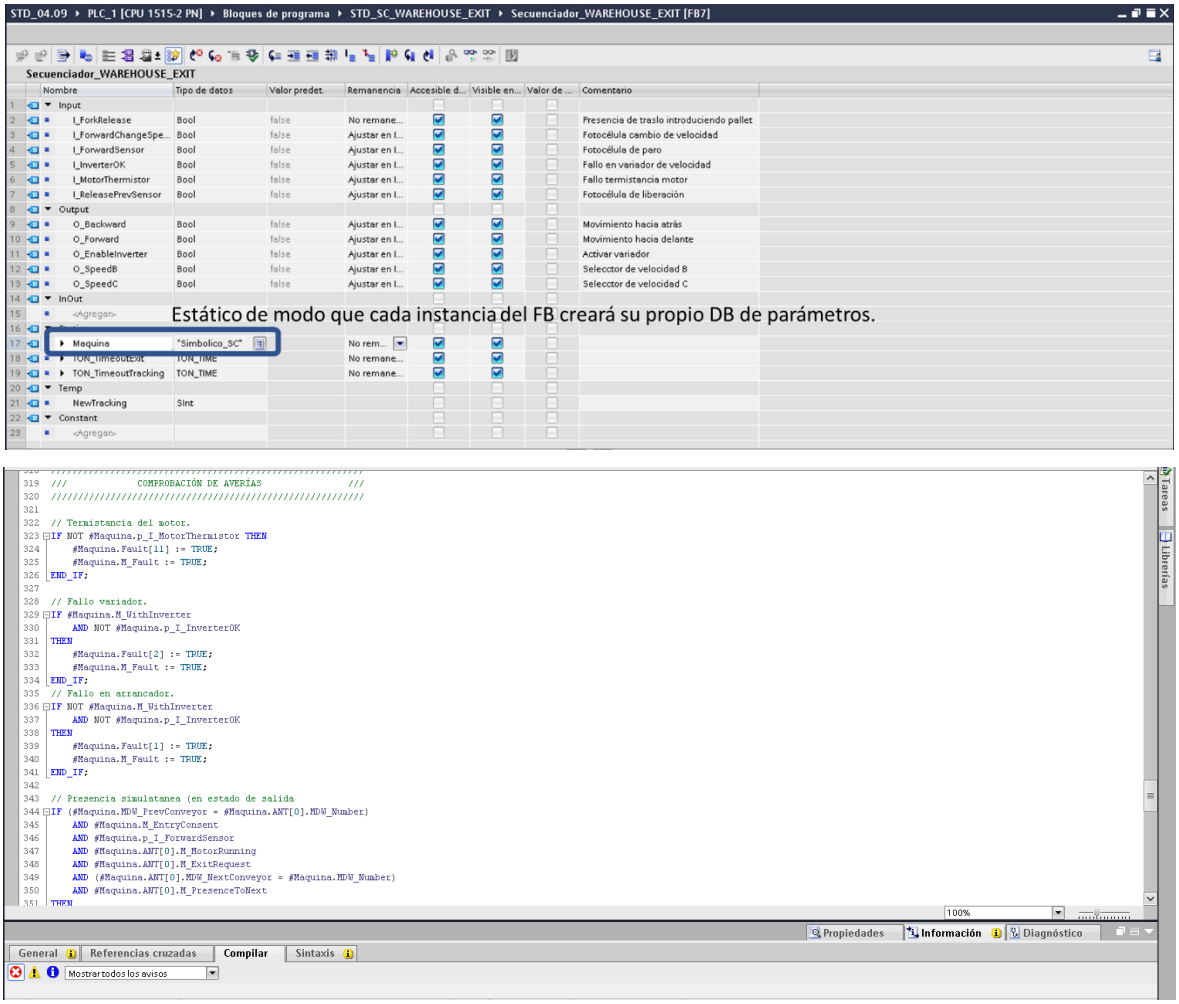


Figura 5.12. FB principal de secuenciador.

Además de los métodos definidos por el usuario para modelar el comportamiento de una máquina, cada secuenciador contiene ciertas librerías adicionales para realizar una serie de acciones típicas. Por ejemplo, se incluyen aquí las librerías de funciones para gestionar

las comunicaciones con el sistema de gestión de almacén, funciones para modificar el *tracking* de los contenedores (incluyendo aquí la altura, el peso, el identificador, origen, destino…). Puesto que estas librerías son únicas para las instalaciones de Mecalux, debe pensarse una forma de generarlas. En caso de que el sistema de control sea Galileo, la generación es inmediata, ya que llevan utilizándose largo tiempo. Sin embargo, para instalaciones que utilicen un PLC convencional, se han implementado todas estas funciones en sus respectivos programas de desarrollo y siguiendo las operativas de comunicaciones, gestión de bloques de datos, etc. que ofrece cada fabricante de forma particular. Posteriormente, se han exportado de manera que su código esté protegido. De esta forma, se podrán importar todas las funciones de estas librerías junto con el resto de elementos generados de forma habitual, garantizando que no pueden ser ni observados ni modificados por un usuario no autorizado. Cabe destacar, que los elementos de estas librerías deberán ser desarrollados en cada una de las herramientas de cada una de las compañías, debiendo implementar cada una de ellas prácticamente de forma independiente, ya que cada PLC dispone de funciones de comunicaciones, trato de DB, etc. de forma distinta.

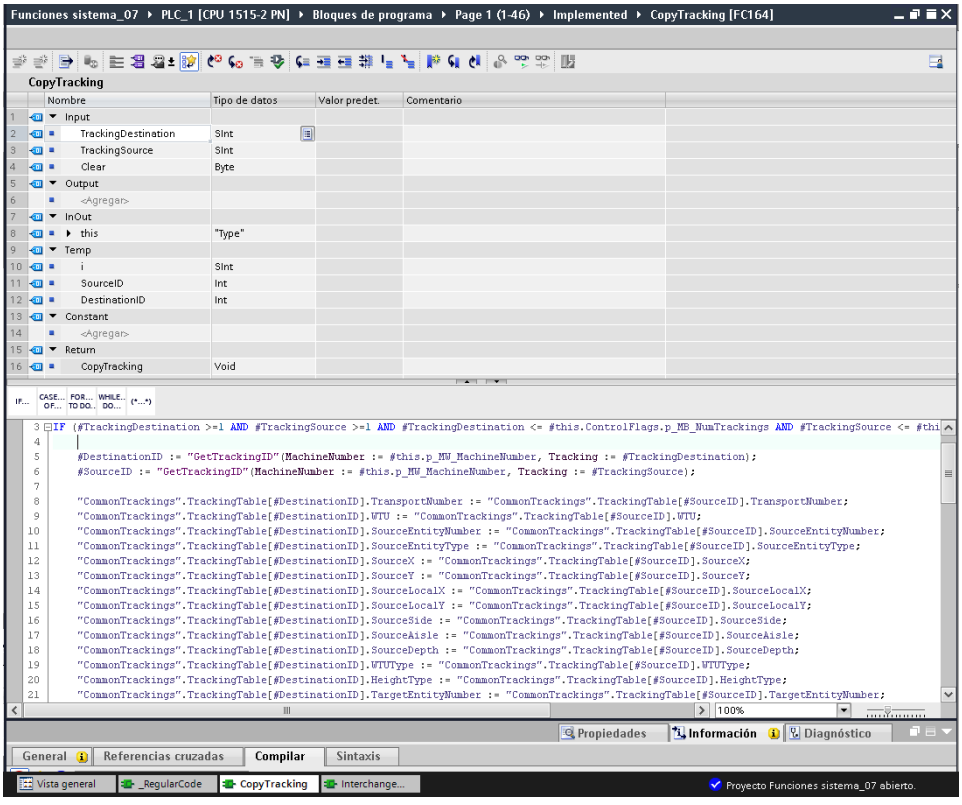


Figura 5.13. Ejemplo de implementación de funciones de librerías particulares.

5.5.2. Elementos auxiliares

En cuanto a la estructura de los programas de control de Mecalux, se siguen unas pautas que benefician el tipo de instalaciones ofrecidas por la empresa. Como es sabido, los proyectos ofrecidos por esta empresa están dedicados a la automatización de centros logísticos y, por tanto, se utilizan una arquitectura muy concreta. Por un lado, las máquinas utilizan una serie de estructuras para realizar un correcto seguimiento de la mercancía. Esto se denomina *Tracking*, estando formado por numerosos campos (tamaño total de 98 bytes) de información de origen y destino, número de transporte, altura del contenedor, peso, identificación, etc. Este elemento viene definido por el sistema de gestión de almacén. El protocolo habitual se da al introducir un pallet en el sistema de almacenaje automático o recuperarlo del almacén, el contenedor pasará por un punto de identificación de entrada y el SGA le asignará una tarea de movimiento y un *tracking* particular. Se ha definido que cada máquina pueda tener hasta un total de 10 estructuras de *T*racking, aunque lo habitual es que, puesto que los transportadores de rodillos y de cadenas son de tamaño unitario de un pallet, solo necesiten uno. El procedimiento típico una vez que se le ha asignado un identificador al pallet en el punto de identificación de entrada, es que las máquinas vayan pasándose el *tracking* de una a otra junto con la mercancía. De esta forma, podemos saber la información del contenedor en todo momento. Para manejar estas estructuras se utilizan funciones englobadas en las librerías propietarias de las que se habló con anterioridad, siendo las típicas: copiar, intercambiar, borrar, consultar campos de la estructura... Con respecto al tratamiento de la información, se ha diseñado una solución pensada en la eficiencia en el aprovechamiento máximo de memoria. Hay que tener en cuenta que 98 bytes por máquina (algunas veces más de una de estas estructuras) multiplicado por cientos de máquinas que forman el *layout* de una instalación, puede concluir en cantidades de espacio necesario muy grandes. El resultado toma dos vertientes en función del autómata destino de ejecución. En caso de Galileo, puesto que es un servicio de un ordenador industrial y programado de forma eficiente en C++, no existe problema de memoria ocupada. Sin embargo, cuando el *target* es un PLC convencional, debe tenerse en cuenta la eficiencia y buscar soluciones adecuadas:

1. Existirá un UDT con todos los campos de un tracking (número de transporte: DINT, Tipo de contenedor: SINT, Altura: INT, etc.)

2. Existirá un DB de datos con todos los Trackings de la instalación (array de UDTs de tipo Tracking). En el peor de los casos, cada una de las máquinas podría llegar a tener un contenedor encima, por tanto, tantas como máquinas.
3. Cada máquina apunta a un elemento del array mediante una variable específica.
4. El contenedor al entrar en el circuito se introduce en el elemento asociado a la mesa de entrada.
5. A partir de ahí el Tracking se irá pasando entre máquinas gracias a intercambios entre las variables que apuntan al Tracking. De esta forma, no se realizan copias de 98 bytes (que es lo que ocupa un tracking) en cada transferencia de contenedor a de una mesa a otra, sino que simplemente se intercambian variables de tipo entero, siendo esto mucho más óptimo.

Existen muchas soluciones de este estilo para destino PLC, ya que, aunque en Galileo se hace de la forma más clara, en PLC ocuparía demasiado espacio o serían tareas muy lentas. Con esto se quiere ilustrar el tipo de innovaciones que han tenido que pensarse estos casos.

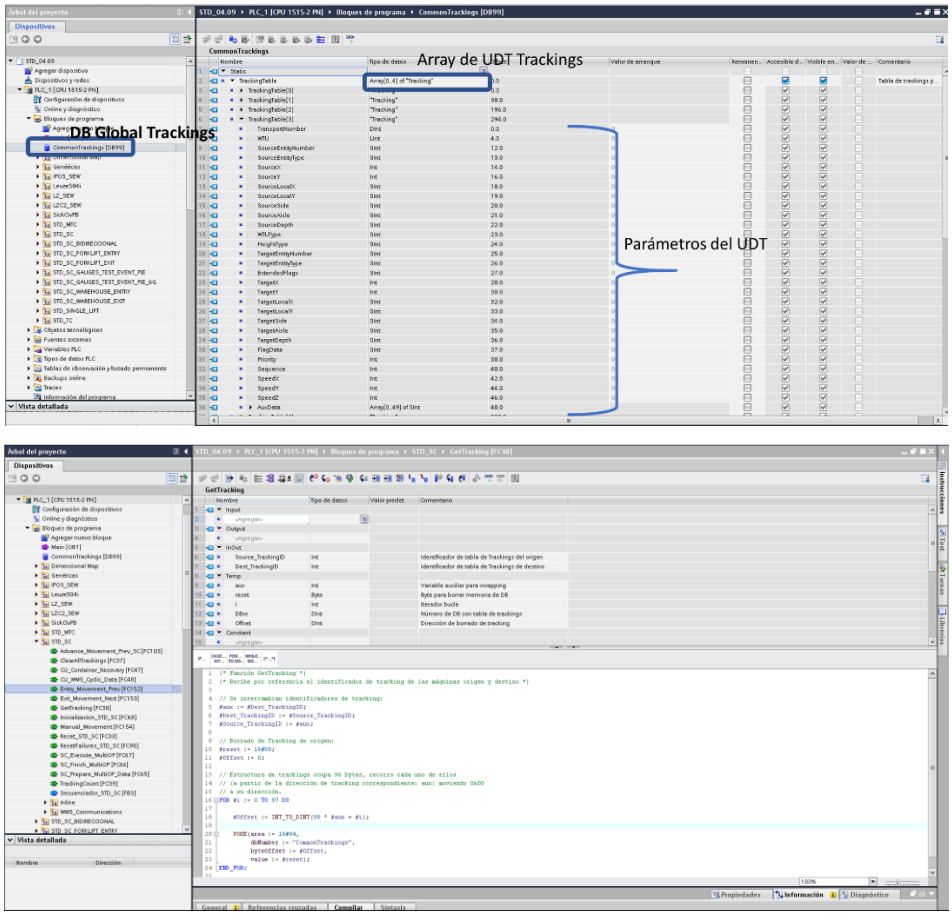


Figura 5.14. DB d trackings y funciones de intercambio.

Otra estructura necesaria es la utilizada para realizar comunicaciones con el sistema de gestión de almacenes. Esta es similar a la anterior, difiriendo en la información almacenada. En este caso, se guardan datos acerca de conexión, búsquedas de orden para realizar movimientos, evento realizados por control para notificar valores de sensórica (peso, tamaño de la mercancía, defectos en los pallets, etc.). En lugar de ser copiado entre máquinas de forma síncrona con la mercancía, estas estructuras de datos son enviadas por comunicación *socket* al sistema de gestión de almacén.

Por último, se han añadido otras estructuras muy importantes para el funcionamiento de la instalación. Puesto que las máquinas suelen ser del tamaño de un pallet unitario, será necesaria algún tipo de elemento de notificación de variables entre máquinas. En lugar de añadir todos los parámetros de los secuenciadores, solo se añadirán las estrictamente necesarias, por lo que es tarea del programador de seleccionar solo aquellas que utilice. La razón de esto es reducir el tamaño de espacio utilizado en los autómatas. Aquí es donde entran los *arrays* de anteriores y posteriores (*ANTPOS)*. Estos son vectores de estructuras contienen todas aquellas variables que es necesario intercambiar entre transportadores, conteniendo tantos elementos como líneas de entrada tenga. Para entender mejor este funcionamiento es necesario explicar la instanciación de máquinas, que se realizará en el siguiente apartado. Aun así, en la siguiente figura puede verse un esquema sencillo de este comportamiento.

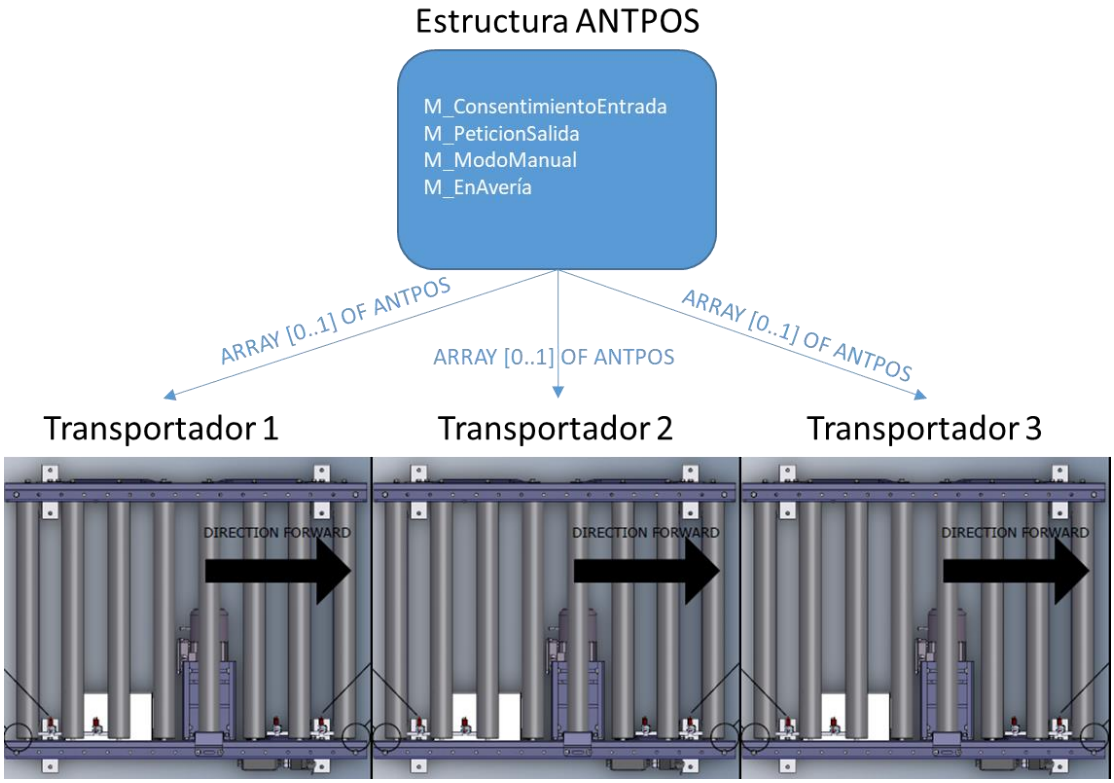


Figura 5.15. Esquema de anteriores y posteriores.

En primer lugar, se deberá definir un UDT con las variables necesarias. Esto se realizará en un espacio reservado a tal efecto en la interfaz gráfica de Designer. Esta estructura es única para todo el proyecto y será necesario que todos los secuenciadores que la utilicen por código, compartan al menos todos los parámetros definidos en ella, tanto nombre como tipo de variable. Posteriormente, se dispone de otro asistente gráfico (al definir transportadores en el *layout* de la instalación para el SCADA) para especificar los anteriores y posteriores de cada máquina. Por ejemplo, en la figura anterior, el Transportador2 tendrá definido como anterior uno al Transportador1 y como posterior uno al Transportador3 (array de dos elementos). Al especificar esta información, el generador de código creará automáticamente tantos elementos de la estructura de anteriores y posteriores como se haya definido, se encargará de forma transparente al usuario de que las variables se intercambien correctamente en cada ciclo de ejecución entre las máquinas interconectadas (realizando copias de estructuras de una máquina hacia sus anteriores y posteriores), todo de forma transparente para el programador y con la sintaxis requerida por cada compañía de PLC. De esta forma, como se verá en los ejemplos de aplicación, se consigue un código muy modular y estructurado, todo ello de forma automática, siendo su utilización tan sencilla como:

```
IF ANT[0].M_ConsentimientoEntrada AND NOT POS[0].M_EnAveria THEN
…
END_IF;
```

Donde ANT[0] es el primer elemento definido como anteriores en la máquina correspondiente del editor gráfico y POS[0] como posteriores. Aunque pueda parecer confuso en este momento, en los ejemplos posteriores con una instalación real, se entenderá de forma sencilla, comprendiendo el potencial de esta estructura.

5.5.3. Simbólico o definición de máquinas

Como se ha explicado, los secuenciadores definen el comportamiento de una máquina particular, de la misma forma que en los lenguajes orientados a objetos se definen clases. Sin embargo, no se dispone de tantos secuenciadores como máquinas existan en una instalación, sino que cada máquina es una instancia de un secuenciador. Es decir, comparten el código base, pero cada una es un elemento único e independiente. Para entender mejor este concepto, pártase de una instalación simplificada, en la que tan solo existen cuatro modelos de máquinas diferentes, sin embargo, existen multitud de máquinas de cada uno de los modelos. En el programa de control, existirían cuatro secuenciadores con el código fuente para conseguir el comportamiento esperado de cada una de ellas. Por otro lado, existirán tantas instancias de dicho secuenciador como máquinas de ese modelo haya en el almacén. Por ejemplo, 50 transportares lineales y 5 mesas giratorias. De esta forma, el reaprovechamiento de código se encuentra optimizado al máximo, siendo igual para los elementos de la misma clase y pudiendo emplear bloques completamente probados anteriormente. Puede apreciarse que, dada esta estructura, contando con un repositorio de secuenciadores estándar, probados y bien definidos, puede conseguirse realizar programas de control muy fiables mediante la modularidad y reaprovechamiento de mejoras incluidas en instalaciones pasadas, todo ello de forma muy simple. Dentro de Designer, la declaración de máquinas se realiza en el Simbólico. Este es un apartado dentro del programa donde se crean las máquinas divididas por zonas (líneas de potencia) de una instalación -asignándole un número de máquina único en el proyecto para su identificación interna-, enlazando las variables de dicha instancia con los parámetros del secuenciador, asociando direcciones

físicas y esclavos de bus (por ejemplo, dentro de una red Profibus) a cada una de las variables de periferia, etc. Es aquí, donde se puede definir el código particular de las funciones declaradas como abstractas de los secuenciadores. En este caso, con estas funciones pueden definirse comportamientos distintos entre instancias de un mismo secuenciador. De todas formas, no debería realizarse de forma descontrolada ya que dado que se tiene que generar una función particular para cada máquina puede llegar a consumir mucha memoria. Como ya se ha explicado, para ejecutar el código particular de cada máquina (llamándose todas ellas igual), en la generación hacia PLC clásico cada función abstracta contiene un *CASE* con el código de cada una de las máquinas que sobrecargan esta función, filtrando el código que tiene que ejecutar cada una por el número de máquina de cada una de ellas.

6. Programación de máquinas

En el capítulo anterior, se ha descrito la arquitectura elegida a la hora de estructurar el código que se generará para los diferentes PLCs. Aunque pueda parecer que este podría ser el primer paso, en realidad, aunque partiendo de una base sólida, muchas elecciones en cuanto a la forma de organizar el programa fueron definiéndose a medida que se iban desarrollando máquinas en los diferentes IDE de Siemens y Rockwell. Al final, el objetivo último es conseguir que las distintas máquinas que ofrece Mecalux en sus soluciones de almacenaje automático, se comporten tal y como se espera. En este capítulo, se describirán sin entrar en mucho detalle en las máquinas desarrolladas y se verá un ejemplo sencillo de aplicación.

6.1.- EL ESTÁNDAR DE MECALUX

En los almacenes automatizados existen dos claras diferenciaciones. Por un lado, existen soluciones para mercancía almacenada en pallets. Es decir, contenedores con un tamaño y peso considerable (hasta 1500 Kg por unidad). Por otro lado, una instalación puede estar enfocada a cajas y mercancía más pequeña, en este caso, el peso habitual por unidad de almacenamiento suele no sobrepasar los 50 Kg. Estas dos filosofías de almacenamiento se conocen como Sistema de Transporte Pesado y Sistema de Transporte Ligero y ofrecen numerosas diferencias en cuanto a su funcionamiento y, por tanto, en cuanto al propio diseño de las máquinas utilizadas. En cuanto a las soluciones ofrecidas por Mecalux, se corresponden con los dos tipos, ajustándose a las necesidades del cliente dentro de lo posible. Aunque en la práctica ha sido necesario crear los programas de control para los dos sistemas de transporte, en este proyecto únicamente se presentará el repositorio de transporte pesado ya que es más complejo y se ha utilizado más para las instalaciones reales en las que se ha trabajado. En la Figura 6.1 pueden verse imágenes de los dos tipos de sistemas.

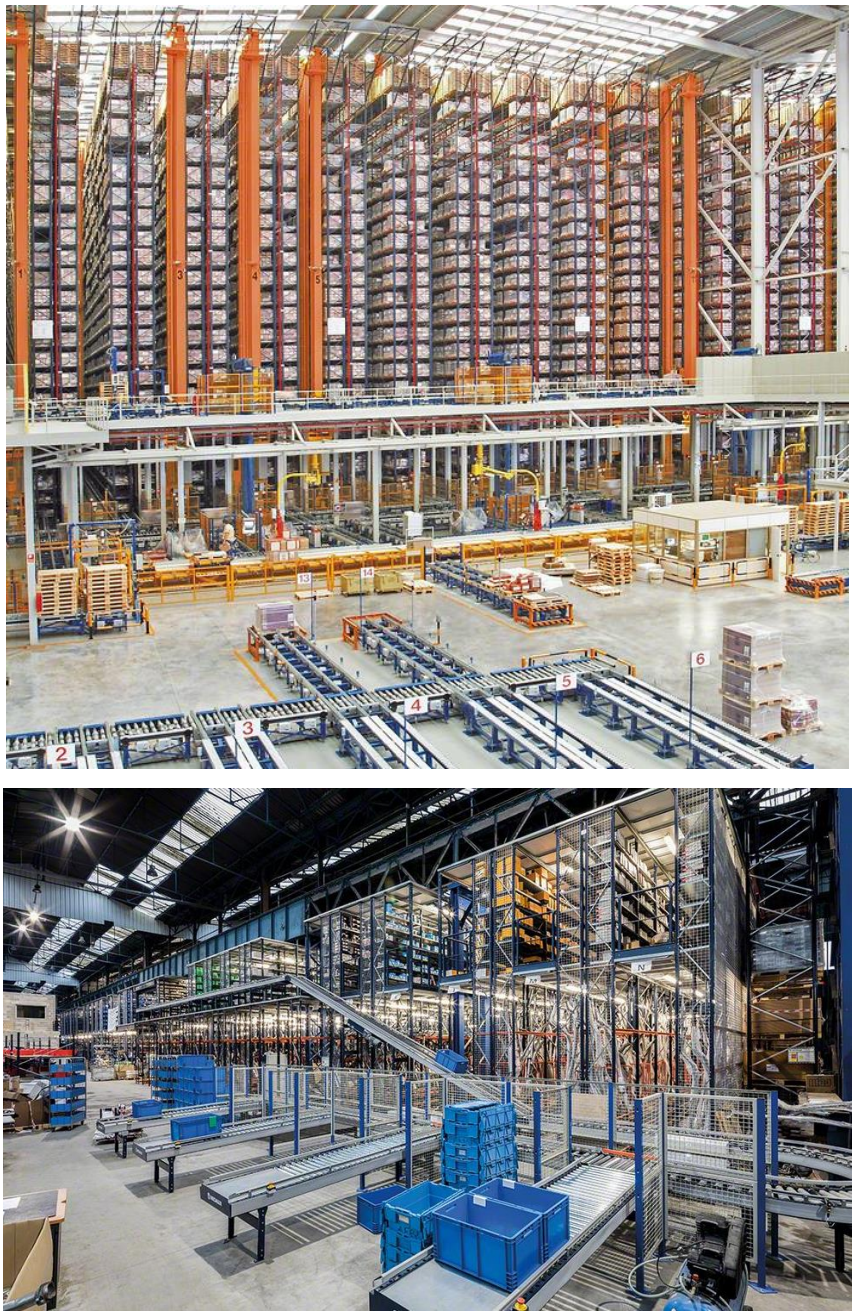


Figura 6.1. Sistema de transporte pesado y ligero respectivamente.

Puede parecer que, en un sistema de transporte de pallets, existe poca variedad de máquinas diferentes, sin embargo, es cierto que las máquinas bases son los trasportadores de rodillos y de cadenas, pero el número de comportamientos diferentes necesarios es muy extenso. A continuación, se describe en pocas líneas el funcionamiento de los diferentes programas de control. En el repositorio de máquinas que fueron programadas en la nueva herramienta de texto estructurado se incluyen:

- STD_CONVEYOR: Máquina que contiene variables base para el resto de transportadores simples. No tiene código, tan solo se utiliza para heredar parámetros y métodos comunes.
- STD_SC: Transportador simple (*Simple Conveyor*) se utiliza para transportar cajas de forma lineal de forma unidireccional. Existen las versiones de rodillos y de cadenas.
- STD_SC_BIDIRECTIONAL: Como su nombre indica, este transportador hijo del simple, añade la bidireccionalidad.
- STD_SC_BIDIRECTIONAL_EVIA. Este secuenciador añade al bidireccional un *handshake* eléctrico con un sistema externo de electrovía, intercambiándose señales de entrada salida para realizar las comunicaciones (consentimiento de entrada, petición de salida…). La electrovía consiste en transportadores guiados sobre carriles que ejecutan trayectorias ovaladas, de tal forma que se puedan repartir pallets desde diferentes y hacia varios transportadores paralelos.
- SC_GAUGES_TEST_EVENT_PIE. Punto de inspección de entrada que añade a un transportador simple numerosa instrumentación para la medida de dimensiones físicas: altura, peso, desplomes laterales de la carga, desplomes frontales y traseros, tacos de pallets rotos, huecos del pallet dañados, medida del tipo de pallet, lectura de código de barras, etc.
- SC_HYDRAULIC_FORKLIFT_ENTRY_EXIT. Esta máquina contiene una unidad hidráulica para permitir la elevación de los pallets. El operario introduce un pallet mediante una traspaleta a nivel de suelo y automáticamente eleva la carga para iniciar el ciclo de movimiento automático.
- SC_FORKLIFT_EXIT. Transportador simple con sensores y código adicional para permitir la retirada controlada de la carga mediante una carretilla guiada por un operario.
- SC_FORKLIG_ENTRY. Punto de entrada de pallets (a altura normal) con carretilla. Contiene detectores para el toro hidráulico, seguridades, pilotos luminosos de indicación, etc.
- SC_RECONDITIONED. Añade al anterior la lógica necesaria para permitir el reacondicionamiento de pallets.
- SC_EXTERNAL_ENTRY_EXIT. Transportador con comunicación flexible por periferia para comunicar con sistemas de transporte externos al de Mecalux.

- SC_WAREHOUSE_ENTRY. Transportador que permite la entrada de la mercancía a un pasillo con transelevador, añadiendo las señales de seguridad y comunicación necesarias.
- SC_WAREHOUSE_EXIT. Transportador que permite la salida de la mercancía a un pasillo con transelevador, añadiendo las señales de seguridad y comunicación necesarias.
- SC_CAROUSEL_ENTRY. Máquina que permite introducir un pallet en un carrousel gestionado por control.
- SC_RECOVERY_STATION. Elemento que permite la recuperación de pallets con tracking desconocido (incluye lectura de etiqueta).
- SC_LZ_ENTRY_EXIT. Entrada y salida a sistemas de lanzadera.
- SC_PALLET_STACKER_DISPENSER. Apilador y desapilador de pallets vacíos, permitiendo agrupar o dispensar unidades de carga.
- SC_EMPTY_PALLET_STACKER. Permite colocar un pallet con carga sobre un pallet vacío, con el objetivo de cambiar el tipo de pallet o corregir algún defecto del primero.
- SC_FULL_PALLET_STACKER. Coloca un contenedor encima de otro para aprovechar mejor el espacio en pedidos con alturas pequeñas.
- MTC. Intercambio entre transportadores de rodillos y de cadenas a 90 grados. Permite cambiar de dirección desde cuatro entradas y cuatro salidas perpendiculares.
- TC. Transportador giratorio. Permite realizar giros de la mercancía. Por ejemplo, a 90 grados. A diferencia del anterior aquí el tipo de máquina tras el desvío sigue siendo la misma (rodillos o cadenas).
- TMC. Transportador mixto giratorio. Permite realizar cambios entre cadenas y rodillos y aplicar giros simultáneamente.
- SINGLE_LIFT. Elevador de contenedores unitario. Permite subir y bajar cargas a distintas alturas desde sistemas de transporte de rodillos o de cadenas.
- TROLLEY. Transportador simple sobre carriles con elementos para permitir el movimiento en traslación sobre los mismos. Permite comunicar varias líneas de transportadores paralelas.
- TRAS_TRAM. Este transportador permite acumular contenedores y compactarlos, contando con dos zonas de motores independientes (entrada de acúmulo y dispensador de salidas).

- TP_OP_UNIOP. Programa que permite gestionar un panel de operador táctil, de tal forma que se puedan controlar las máquinas de la instalación en manual, ver errores, etc.
- FREEING_OP. Como el anterior controla un panel de operador, pero en este caso con pulsadores y selectores para permitir en control en ambientes de frío (con guantes).
- CABINET_C0. Armario eléctrico estándar (versión 1). Se gestionan seguridades, líneas de potencia, señales auxiliares…
- CABINET_CO_V2. Nueva versión del armario eléctrico de Mecalux.
- CROSSWALK_CONTROL. Este secuenciador controla los pasos peatonales sobre los transportadores. Se necesita realizar un control de la carga para evitar accidentes.
- MUTING_CONTROL. Barrera de seguridad para impedir que los operarios accedan a zonas peligrosas.
- CAROUSEL_CONTROL. Este elemento controla la entrada de contenedores a un carrusel.
- UPDATE_ROUTES. Este secuenciador actualiza los estados de las rutas y las estaciones de la instalación, enviando esta información (ocupación, modo manual, avería…) al sistema de gestión de almacén.
- ROLL_SUPPLEMENT. Esclavo de rodillos de pequeño tamaño para salvar distancias. No contiene un control complicado, sino que es esclavo de la máquina anteriores.
- Lanzadera simple y APS. Estas son las máquinas más complejas que se han desarrollado. En el caso de la lanzadera simple, se ha programado un vehículo que puede transportar cargas de hasta 1500 Kg a grandes velocidades (150 m/min) a través de railes y descargar en numerosos transportadores de rodillos o cadenas paralelos. En el caso del APS (*Automatic Pallet Shuttle)* en lugar de un transportador, se cuenta con una plataforma inalámbrica que es capaz de salir del vehículo para recoger cargas situadas en las estanterías. Este último elemento tiene una programación cerrada, utilizando un PLC independiente para su control, de tal forma que para la lanzadera es una caja negra con la que comunica ciertas variables.

Todas las máquinas anteriores han sido programadas utilizando la nueva herramienta de desarrollo y texto estructurado. Además de estas, existen otras máquinas llamadas transelevadores, encargadas de cargar y descargar pallets en estanterías de varios niveles.

Estas máquinas son muy complejas y no se han programado en ST. Por el contrario, quieren venderse como una máquina cerrada con control autónomo.

6.2.- MODELOS DE SIMULACIÓN

Puesto que los centros productivos y tecnológicos de Mecalux se distribuyen por varias partes del mundo, para la prueba del código generado es necesaria una herramienta de simulación que permita mejorar la calidad del código y testear el comportamiento de los programas de control dentro de oficina. Puesto que una parte del desarrollo de proyectos de control se realiza en Gijón y las instalaciones con maquinaria se encuentran en Barcelona, se ha visto indispensable el desarrollo de esta herramienta. El departamento de ingeniería de software dedicado a aspectos de simulación ha sido el encargado de implementar este producto. Como resultado final, se ha obtenido un simulador de máquinas 3D que se comunica con el sistema de control. Por un lado, el sistema de control se ejecuta normalmente, como si de periferia de E/S real se tratara. Por otro lado, el simulador accede a dichas variables de periferia (escritura de sensores y lectura de actuadores), pero siempre de forma externa e independiente, de tal forma que sea físicamente imposible trucar resultados.

El sistema 3D cuenta con varios elementos. En primer lugar, representa los modelos de cada una de las máquinas de la instalación. Estas cuentan con los sensores y actuadores definidos. Por otro lado, están los contenedores, los cuales son objetos independientes de los anteriores. De esta forma, cuando un contendor pasa por la zona de actuación de un sensor, este lo detecta en un hilo aparte, siendo ambos dos elementos exentos entre sí, como si del mundo real se tratase. Todo este desarrollo de ingeniería de software tan complejo ha sido desarrollado por un departamento ajeno al de control, aunque con las supervisión, recomendaciones y retroalimentaciones del segundo.

Para este proyecto, se han elaborado los modelos de todas las máquinas programadas ya comentadas. El punto de partida, han sido los planos y esquemas electromecánicos ofrecidos por el departamento de ingeniería eléctrica e I+D. A continuación, se expondrá el resultado de algunos transportadores y el proceso seguido para su realización.

6.2.1. Generación de modelos

En primer lugar, se parte de un archivo "*.ac*" implementado por las personas dedicadas a la realización de modelos 3D (Solid Works, AutoCad, AC3d, etc). En esta primera etapa tan solo deben modificarse ciertos elementos para ajustarlo mejor a las necesidades (por ejemplo, añadir elementos sencillos como chapas de detección de inductivos, nombres de las mallas, etc.), pero el trabajo importante es realizado por personas externas. El modelo terminado deberá ser exportado como fichero DirectX desde el propio AC3D.

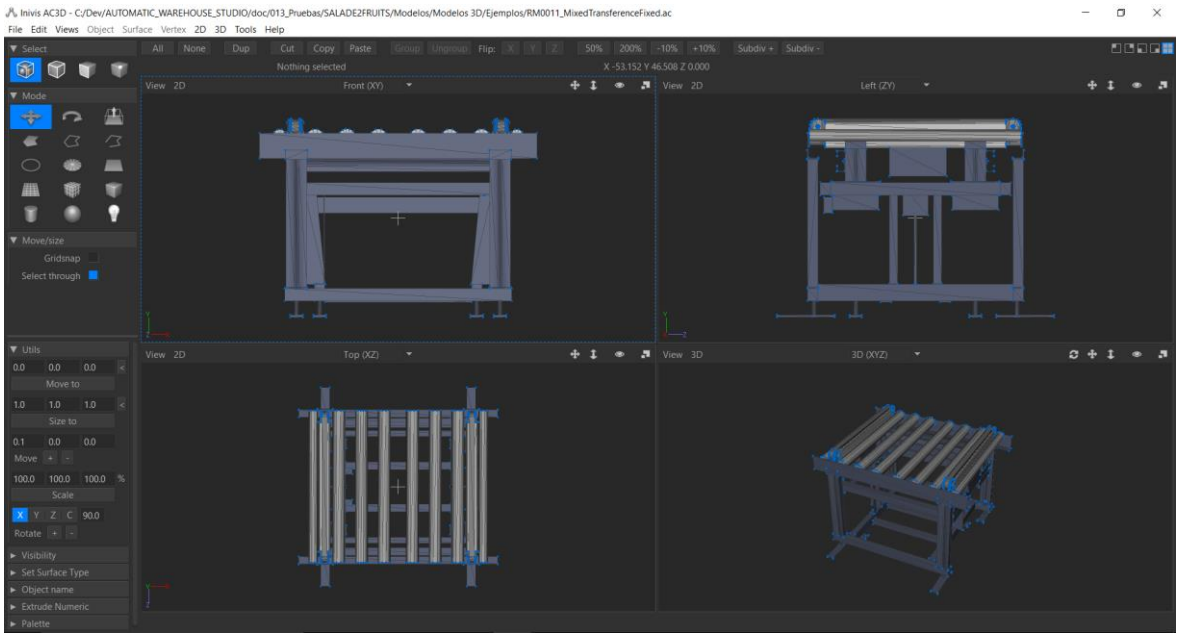


Figura 6.2. Modelo en AC3D.

6.2.2. Importación del archivo *".X"*

El siguiente paso es la importación del fichero generado en el punto anterior. Esta acción se realiza en un nodo del proyecto de control dedicado a tratamiento de los modelos 3D. En este formulario, se importa el "*.X*", se establecen las medidas por defecto y se permite la aportación de código asociado al modelo. Típicamente se introducen sentencias para establecer ciertas acciones de escalado, tamaños por defecto o comportamientos específicos. Esto se realiza mediante lenguaje C#.

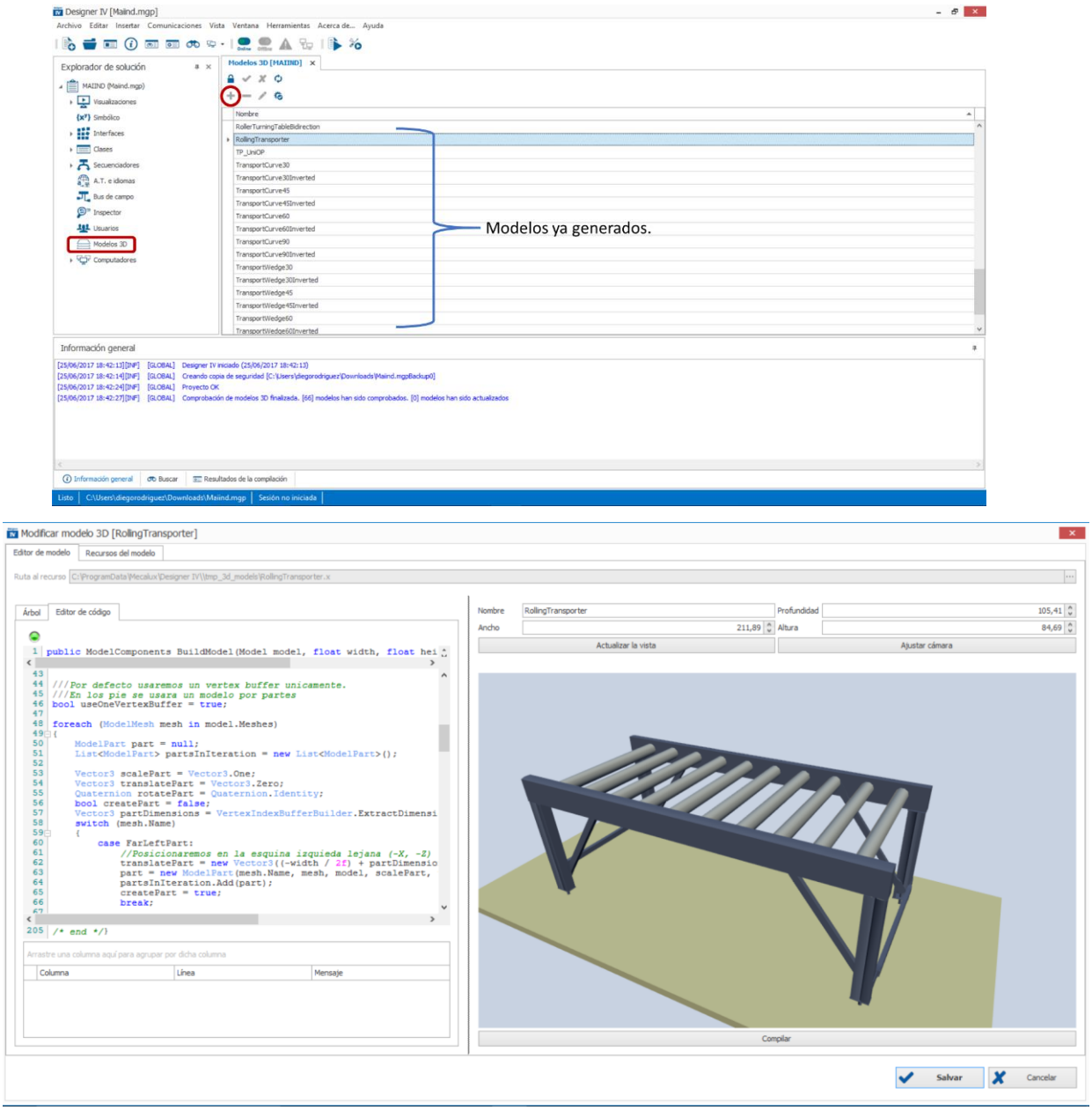


Figura 6.3. Modelos ya generados y vista de importación.

6.2.3. Asignación de modelos al secuenciador y desarrollo

Consecutivamente, se deberá asignar a cada secuenciador de una máquina particular el modelo que se utilizará. Además, se crearán los sensores, actuadores y animaciones que se utilizarán durante la simulación.

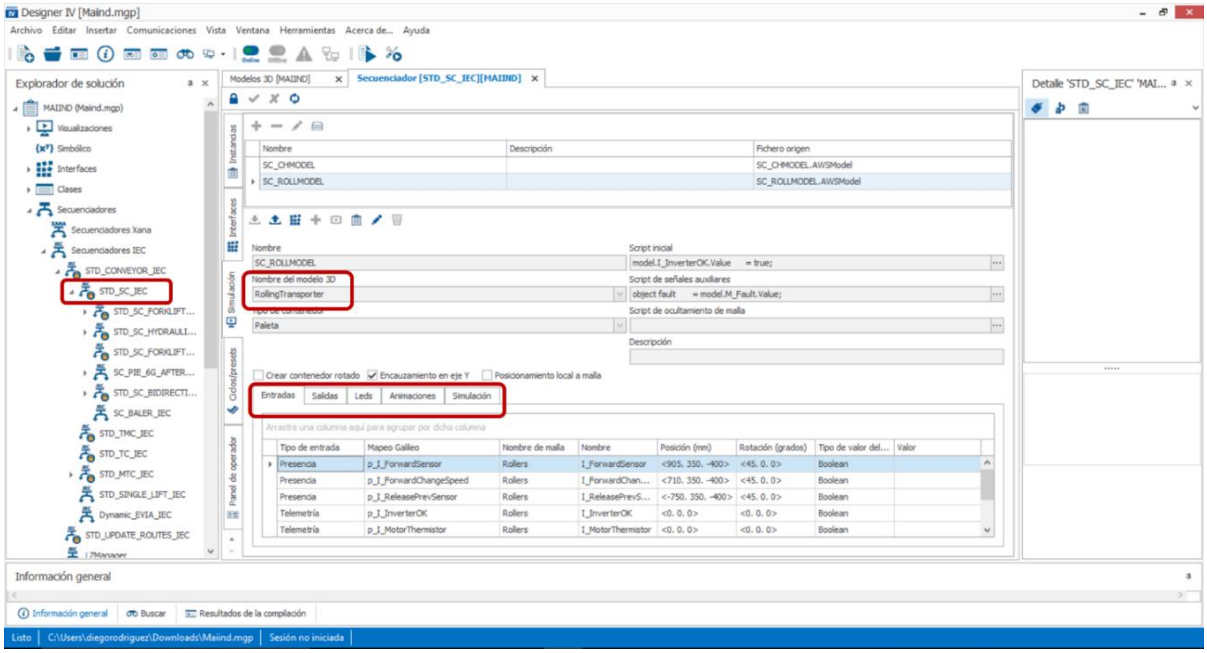


Figura 6.4. Modelo de simulación en el propio secuenciador.

En la última Figura se puede apreciar el apartado de simulación de un secuenciador (concretamente un transportador simple). Además, se ve la selección del modelo a utilizar (el presentado en la imagen anterior) y las pestañas de sensores, salidas, simulaciones… Una vez definidos los sensores y actuadores y los parámetros del secuenciador asociados a estos, se deberá configurar el tipo de sensor (inductivo, fotocélula o telemetría), la distancia de detección, el tipo de variable sobre la que actuará y su nombre, su posición física, etc. Para ajustar las coordenadas, existe una pestaña de simulación donde puede verse una vista previa.

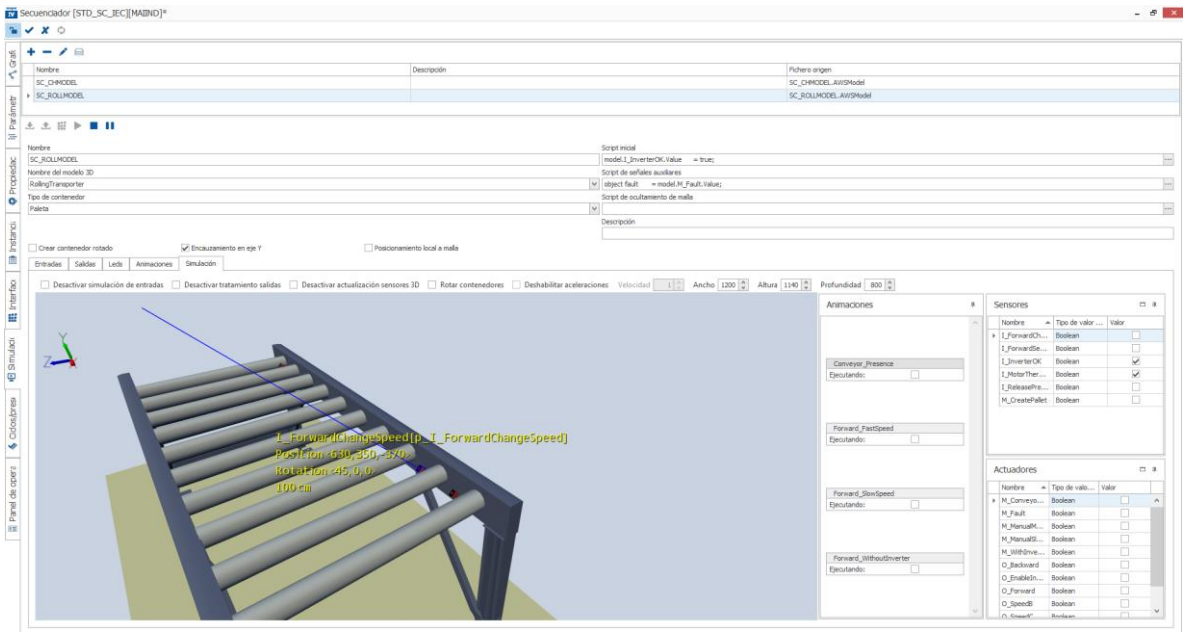


Figura 6.5. Colocación de sensores y aspecto 3D.

Por último, deberán crearse las animaciones del secuenciador. Es decir, los movimientos que realizará la máquina. Estas animaciones se basarán en la activación de los actuadores definidos y tendrán asociado un eje y sentido de movimiento, una velocidad y una aceleración. La activación de esta se realizará programando tres scripts: inicialización, habilitación y desactivación.

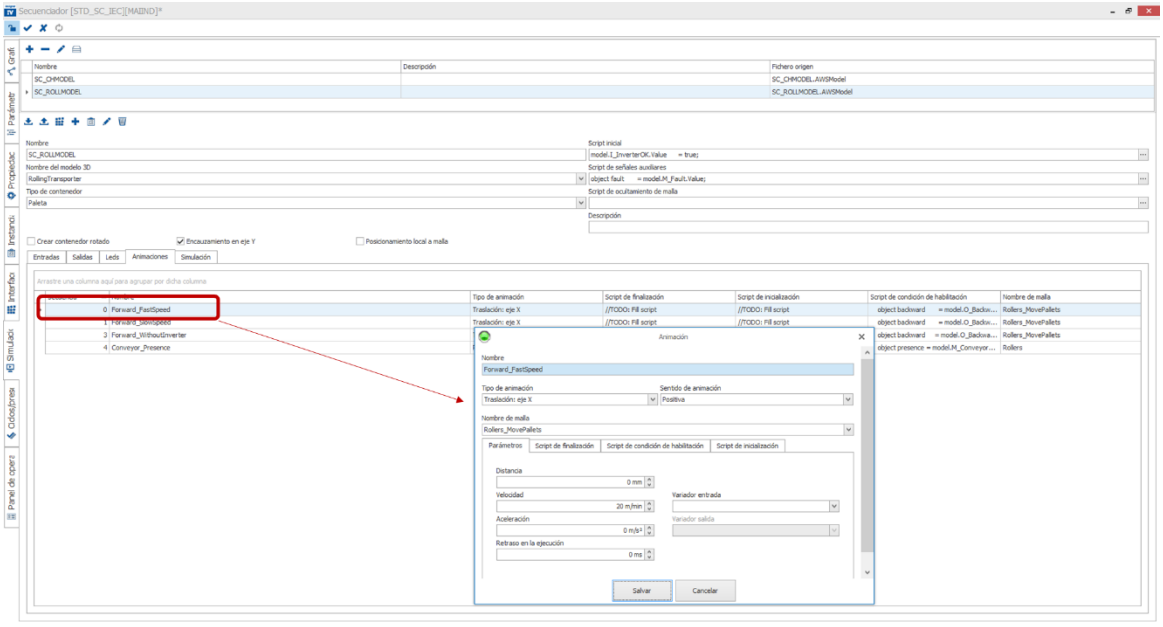


Figura 6.6. Lista de animaciones de la máquina.

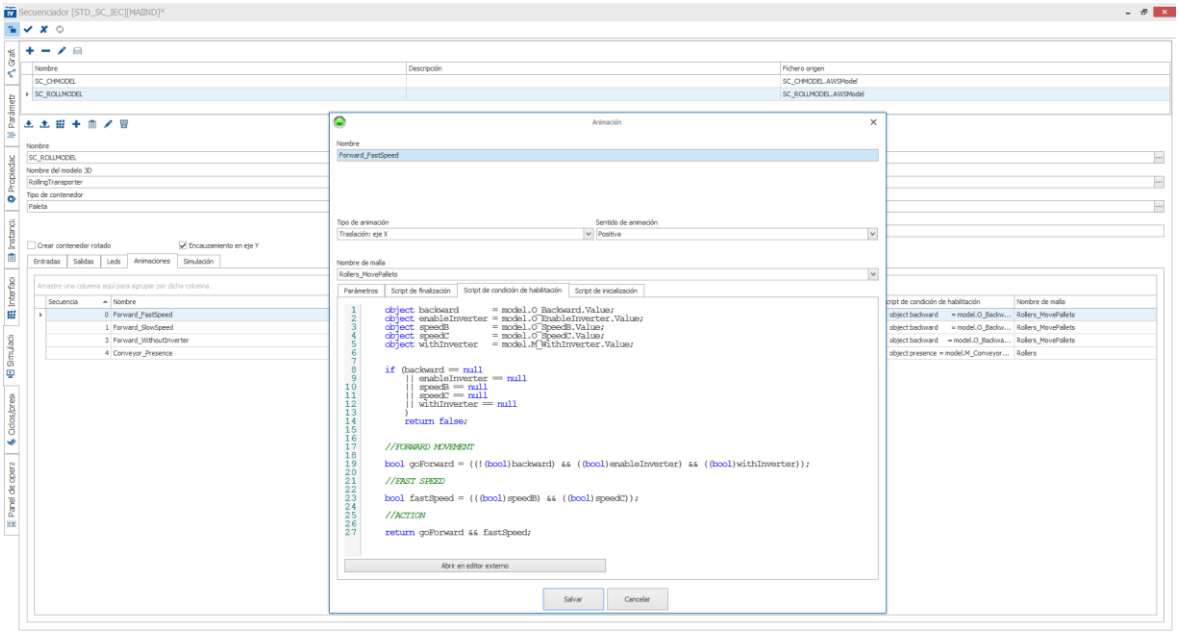


Figura 6.7. Ejemplo de *script* de habilitación de una animación.

6.2.4. Prueba del código de control

Puede realizarse de dos formas diferentes. Por un lado, puede probarse el programa de control en la máquina de forma independiente. Esto se realiza en la propia pestaña de simulación vista en la Figura 6.3. Esto es útil mientras se está empezando a crear la máquina y el comportamiento no se encuentra completamente definido, así como el modelo de simulación no está definitivamente terminado. Así se puede comprobar que las animaciones se muevan con las salidas de los motores, que los sensores detecten presencia en los puntos adecuados, etc. Para ello se crea un pallet que será el que realice los movimientos y sea detectado por las fotocélulas.

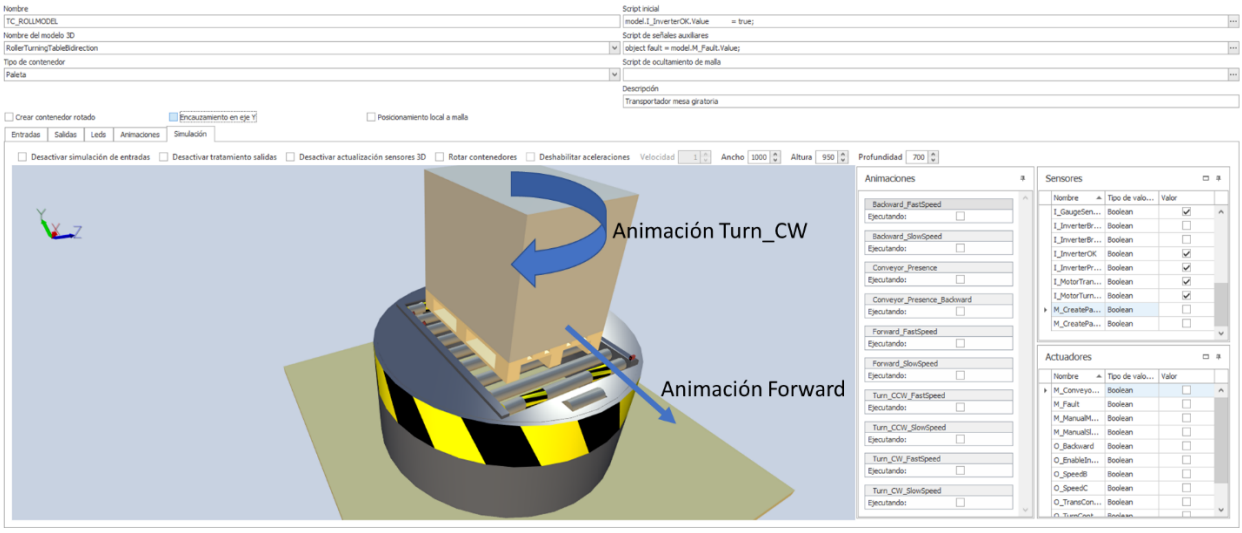


Figura 6.8. Prueba de secuenciador en el entorno de desarrollo del modelo.

Por otro lado, las pruebas importantes de programación no se realizan de forma independiente sino en conjunción con el resto de máquinas de una instalación. Por ese motivo se ha desarrollado una pestaña de visualizaciones del proyecto, que cumple dos funciones. La primera de ella es realizar pruebas en oficina con todas las máquinas del proyecto y, de esta forma, ganar tiempo en las puestas en marcha. En segundo lugar, el propio *layout* del almacén utilizado para pruebas será empleado como SCADA en la instalación real.

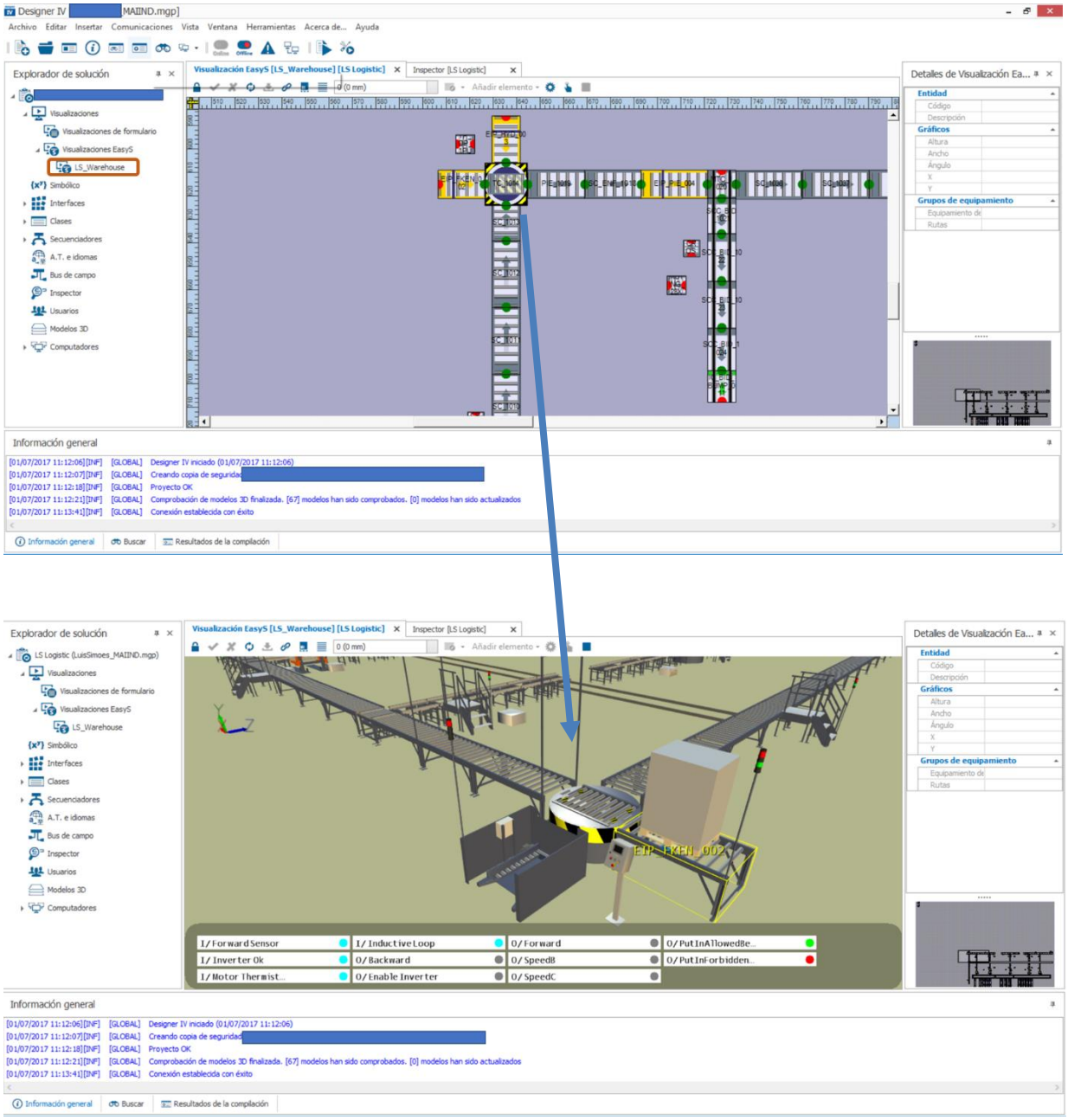


Figura 6.9. Simulación de máquinas conjuntas, también utilizado como SCADA 3D.

A modo de recapitulación, con los pasos anteriores se consigue el modelo de simulación de una máquina que podrá utilizarse para realizar pruebas del comportamiento del código de control en la propia oficina, a la vez que se desarrolla el programa de control. Lo cual ahorra una gran cantidad de tiempo ya que, dada la complejidad del sistema de simulación, su comportamiento puede decirse que es prácticamente igual que en la realidad. La única diferencia podría darse en el comportamiento de cargas muy pesadas (más inercia), deslizamiento de rodillos y problemas mecánicos. Pero a nivel de control, se comporta

exactamente igual. Para este proyecto, se han desarrollado modelos para todos los secuenciadores del sistema de transporte pesado enumeradas en el apartado anterior, colocando los sensores y utilizando actuadores según las máquinas en producción. El resultado ha sido más que favorable.

6.3.- EJEMPLO DE APLICACIÓN REAL

Para entender mejor la aplicación de control desarrollada y la utilidad de esta herramienta, se expondrá un ejemplo de instalación real cuya automatización será realizada exclusivamente con el nuevo Designer y texto estructurado como lenguaje de programación. Puesto que los datos relacionados con la oferta, distribución del almacén, esquemas eléctricos, etc. son confidenciales, en ningún momento se nombrará a dicha compañía. Además, no se expondrá el *layout* completo, sino un fragmento.

6.3.1. Interpretación de la oferta

El punto de partida del desarrollo pasa por comprender el funcionamiento que el comercial de Mecalux y el cliente del almacén han definido. Esto puede parecer una tontería, sin embargo, no siempre se llega a consenso entre las descripciones de unos y otros y, en muchas ocasiones, la información suele ser ambigua. En cualquier caso, el documento firmado por el cliente define como quiere que se comporte el almacén y el rendimiento esperado. Aquí también se expresa como debe actuar el sistema de control, el sistema de gestión de almacén…

En el caso presentado, el cliente será un operador logístico, encargado de distribuir la mercancía almacenada. La forma de trabajar del almacén será:

1. El ERP del cliente se comunicará directamente con el sistema de gestión de almacén, indicándole los productos a introducir en el almacén y las expediciones que deben realizarse. Todo ello por medio de comunicaciones internas.
2. Los operarios introducirán por los puntos de entrada los contenedores preavisados por el ERP. En estos puntos, se realizará una lectura automática de código de barras mediante un escáner. Todas las referencias de productos deberán estar preavisadas por el planificador de recursos del cliente.

3. Una vez en los puntos de entrada los contenedores podrán ser rechazados sobre las propias mesas en caso de lectura incorrecta, etiqueta desconocida, etc. Los contenedores aceptados, serán introducidos de forma automática hacia el almacén y ubicados en el lugar más adecuado posible en función de otra mercancía del mismo proveedor, fecha prevista de expedición, productos relacionados.
4. Cuando sea necesario, normalmente en horario nocturno, el ERP indicará al sistema de gestión de almacén los productos que serán expedidos, teniendo que extraerse del almacén y llevarlos a una zona de recogida de camiones compuesta por mesas dinámicas por gravedad. Todos los productos que se expidan conjuntamente serán colocados en las mismas dinámicas para cargar los camiones situados en los muelles de forma rápida y eficiente.

Con esta información y con las directrices del cliente, el departamento de ventas y el de proyectos genera un plano CAD con el *layout* de la instalación y las medidas perfectamente acotadas. Cuando se llega a un consenso entre las partes se acepta la distribución y el departamento de ingeniería eléctrica y de control, que son los realmente implicados para este proyecto fin de máster, empiezan a desarrollar cada uno su parte.

6.3.2. Planos de implantación

Con las especificaciones de la oferta y los planos de ingeniería técnica, el departamento de ingeniería eléctrica comienza a desglosar los distintos elementos de la instalación. Siempre basándose en el repositorio de máquinas estándar y con una gran experiencia de proyectos ya realizados. El departamento de Control empezará a trabajar una vez se publiquen dos documentos: el plano de implantación eléctrica y los esquemas eléctricos de todos los elementos del almacén. En el primero se describe el *layout* del almacén, presentando todas las máquinas con sus elementos eléctricos (sensores, motores, etc.) perfectamente identificados con una etiqueta. En el segundo, se especifican los aspectos eléctricos, los módulos de entrada/salida, elementos de seguridad… de todos los dispositivos presentes en la instalación. Ambos son de suma importancia ya que definen las direcciones de las señales de E/S que se utilizarán en los programas de control.

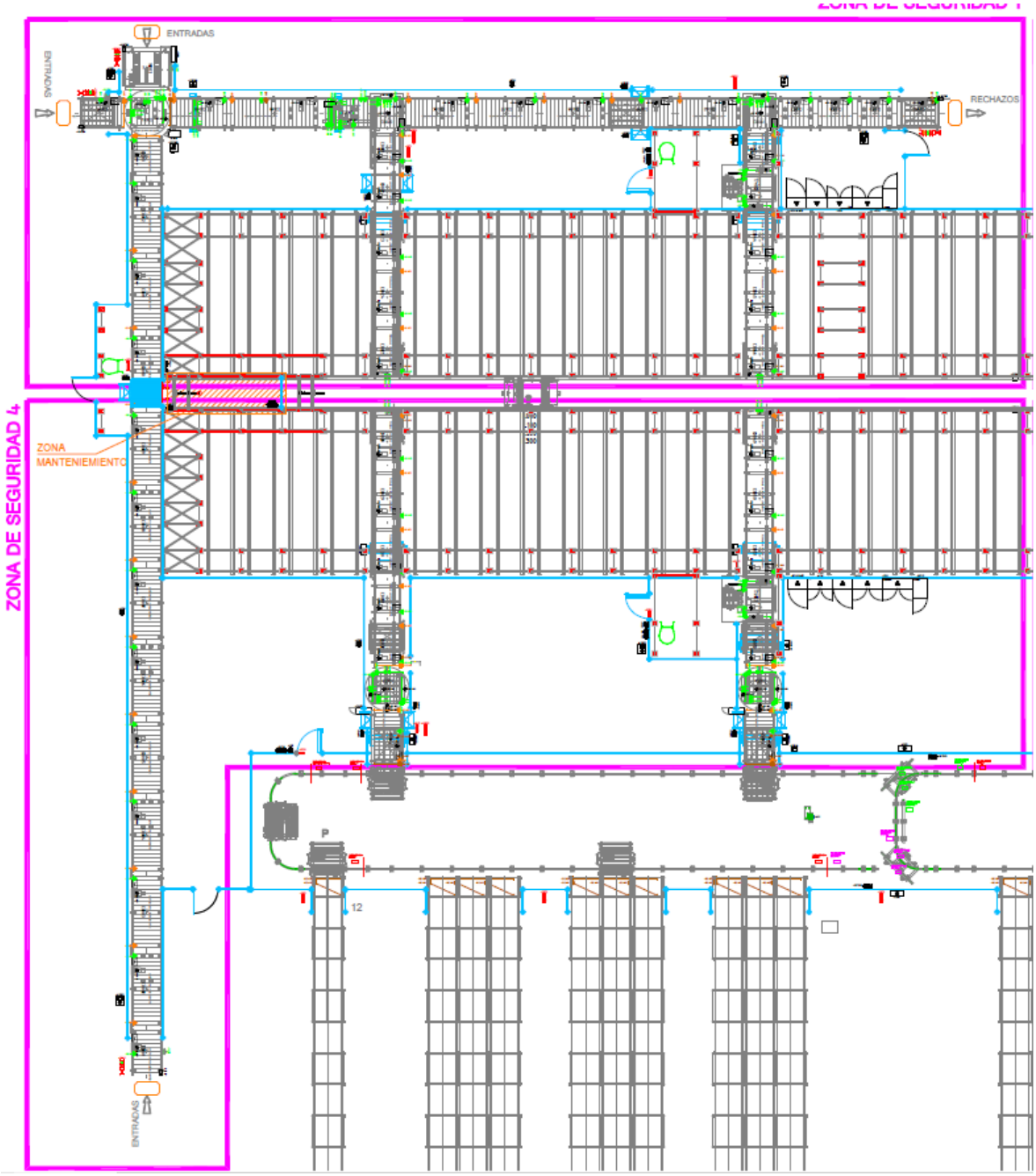


Figura 6.10. Plano de implantación eléctrica de una porción de instalación.

Para este proyecto se utilizará una parte de una instalación real (presentada en la Figura anterior). En principio, parece suficiente para demostrar la utilización de la herramienta de programación. Además, por motivos de confidencialidad no se nombrará la empresa cliente en ningún momento ni se entrará en detalle en ninguno de los documentos protegidos (esquemas, planos, etc.).

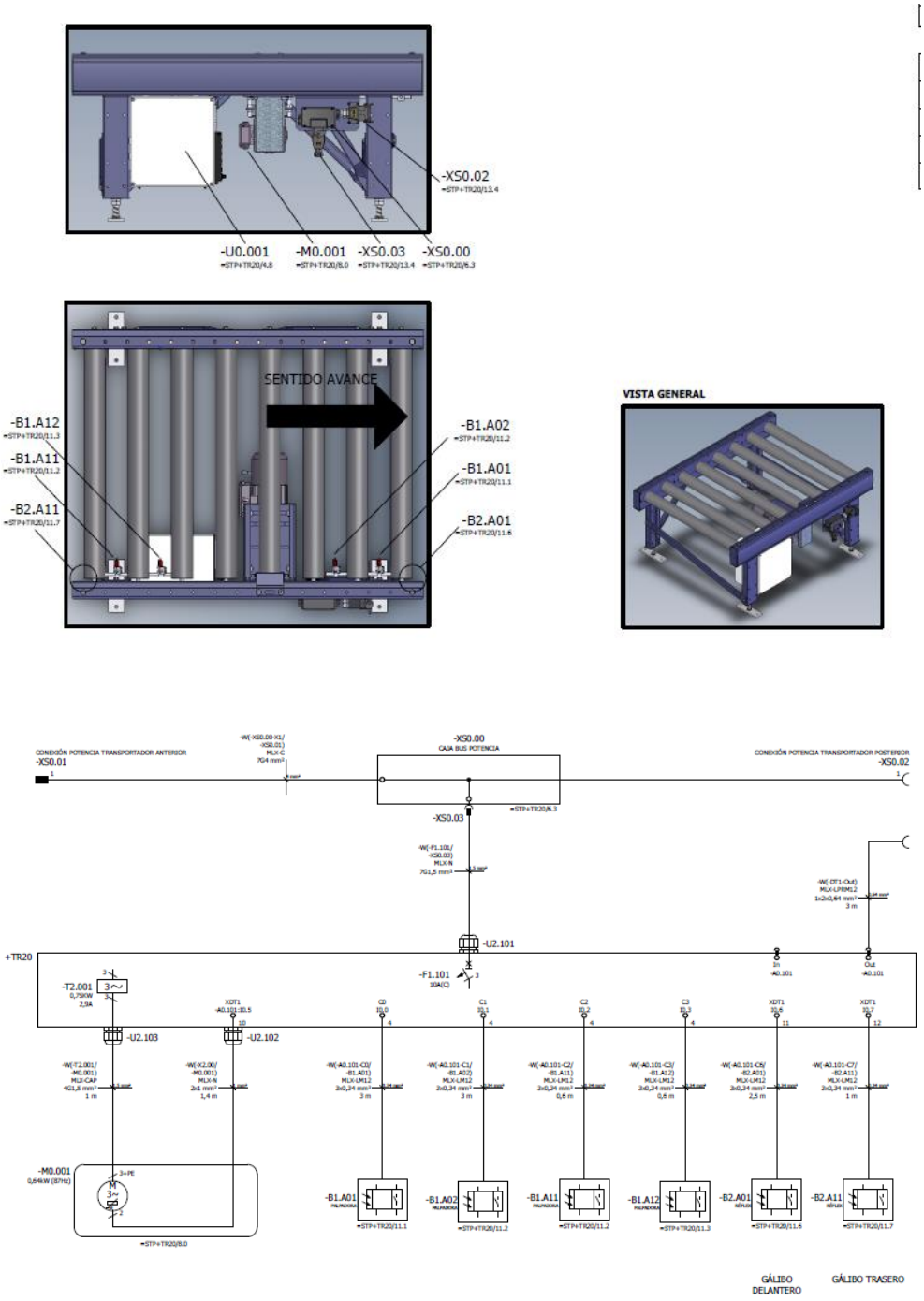


Figura 6.11. Porción de esquemas de un transportador simple de rodillos.

6.3.3. Desarrollo del programa de control

Por último, una vez que se dispone los planos mencionados anteriormente, se procede a la creación del proyecto de control. Para el caso actual, se utilizará el sistema de control Galileo como base de ejecución. No obstante, el desarrollo del proyecto sería similar para

los diferentes autómatas comentados. Un punto que puede llamar la atención es: cómo es posible controlar las máquinas con un ordenador. Esto se consigue mediante la utilización de tarjetas de bus de campo. Cuando se utiliza Galileo, el PC de control deberá contar una (o varias) tarjetas (normalmente se utilizan tarjetas Hilscher) del bus correspondiente, para el caso de esta instalación Profibus. Este dispositivo se conecta directamente a la placa base del PLC mediante PCI o PCI Express. Una vez instalado este elemento (maestro PB/DP) se conectarán los esclavos como en cualquier sistema de automatización normal, utilizando distintos módulos de periferia distribuida. Para esta instalación, se utilizarán cabeceras Phoenix Contact y Beckhoff (en mayor medida).

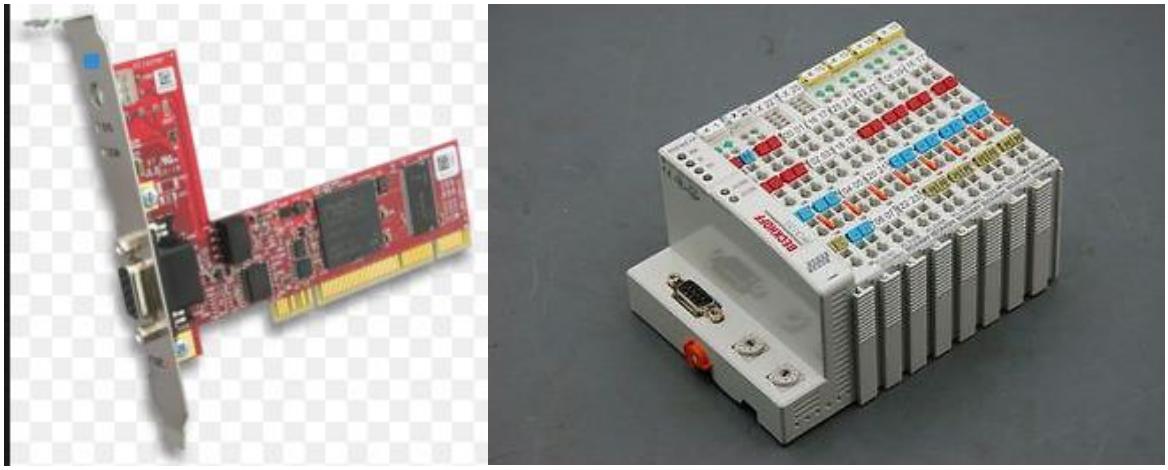


Figura 6.12. Aspecto de tarjeta Hischer CIFX Profibus y cabecera de E/S Bechkoff.

Una vez definido lo anterior, puede pasarse a hablar de la implementación de la solución para el almacén. Como puede apreciarse en el plano de implantación eléctrica, en esta instalación (al menos en la parte presentada) pueden distinguirse varios tipos de máquinas[2]:

- Transportador simple.
- Transportador simple de salida con carretilla.
- Transportador simple con entrada de carretilla y escáner.
- Transportador giratorio.
- Transportador de transferencia mixta.
- Transportador de gravedad.

[2] Existen más de los aquí explicados, como: pasos peatonales, barreras de *muting* (protección de accesos), balizas informativas, armarios eléctricos secundarios, etc. No obstante, para ver el alcance del ejemplo, es suficiente con los presentados.

- Lanzadera.
- Elevador.

El código de estas máquinas no se realiza específicamente para este proyecto, salvo ciertos aspectos personalizados. Por el contrario, se utiliza el repositorio de máquinas ya implementadas y probadas. No obstante, ya que no se ha entrado en detalle, en este apartado se explicará (sin profundizar demasiado) el código de los elementos que aparecen en esta porción del almacén.

El primer punto que se explicará es el comportamiento general de las distintas máquinas, el cual es similar para todas ellas. Cabe destacar que cada una de las máquinas puede entenderse como un proceso independiente, dado que su diseño prioriza el encapsulamiento y la reutilización de código. Se distinguen los siguientes estados.

GDMMA de las máquinas

A6. Inicialización en condiciones iniciales. Sería el estado por defecto al arrancar la instalación de cero, una vez hay tensión, por ejemplo, tras caída de protecciones del armario de distribución, tras accionamiento del seccionador, etc.

A1. Parada en estado inicial. Etapa de reposo tras un rearme del armario eléctrico de zona (si no hay averías ni dispositivos de emergencia accionados) o tras fin de ciclo de entrada/salida de contenedor.

F4. Modo manual donde se pueden accionar todos los actuadores por medio de un panel de operador.

F1. Producción normal. Comportamiento en automático de entrada/salida de contenedores.

A4. Parada desde cualquiera de los anteriores para garantizar que no existen movimientos. Aquí suele configurarse el selector del pupitre en modo mantenimiento, así como pasos peatonales seguros.

A2. Fin de ciclo tras proceso de tránsito de contenedor o forzado a etapa inicial por parte del operario.

D2. Supervisión de averías. Una vez rearmado continuará en el estado en el que se encontraba.

D1. Paro seguro tras accionamiento de dispositivos de emergencia. Tira potencia por lo que se necesitará un rearme eléctrico en el armario de zona.

SOLUCIÓN GDMMA

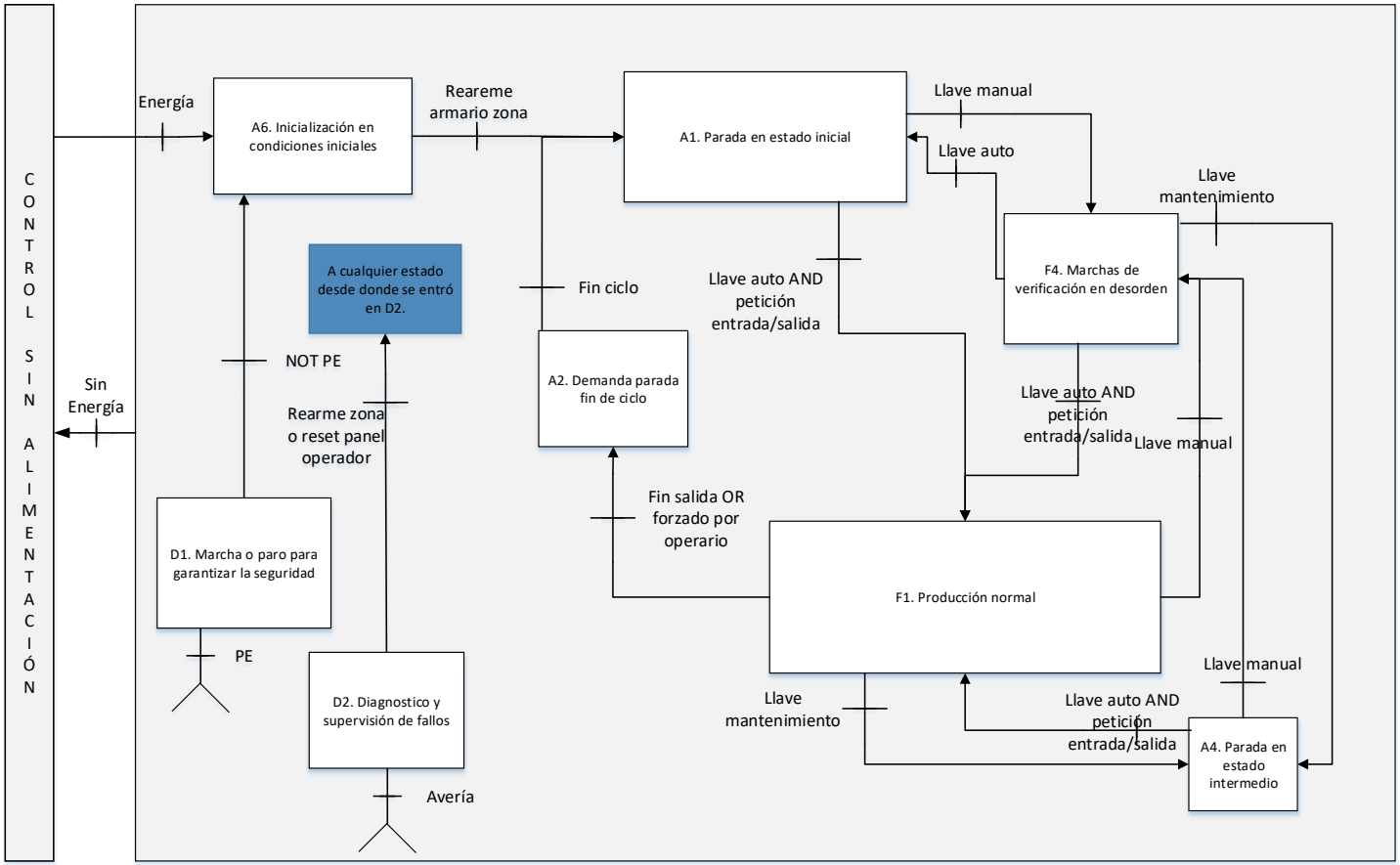
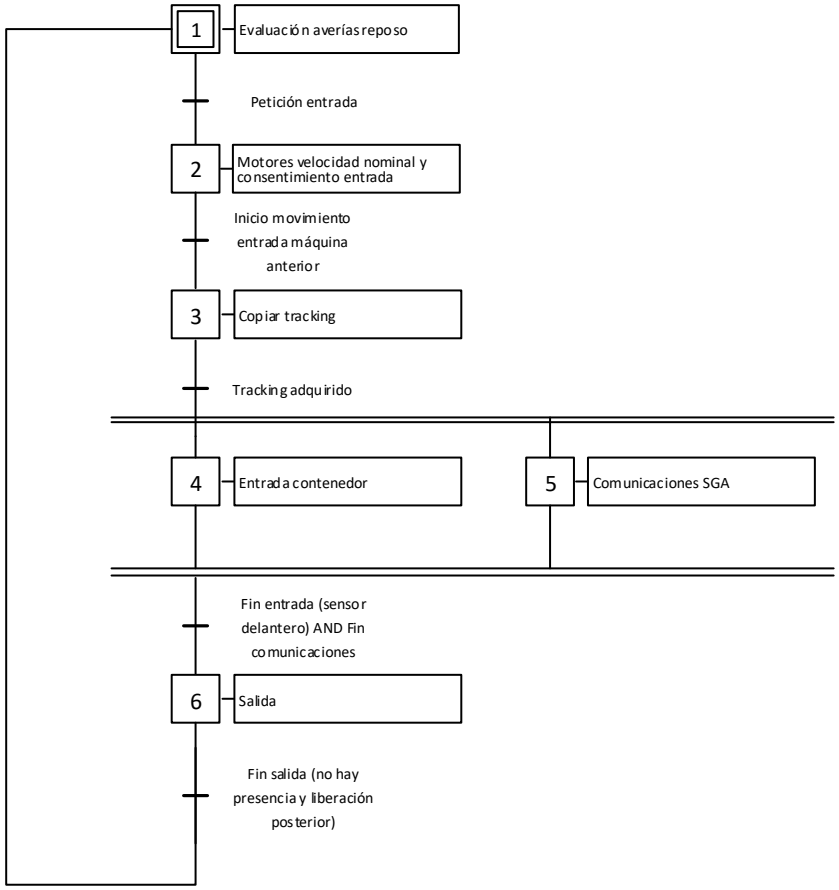


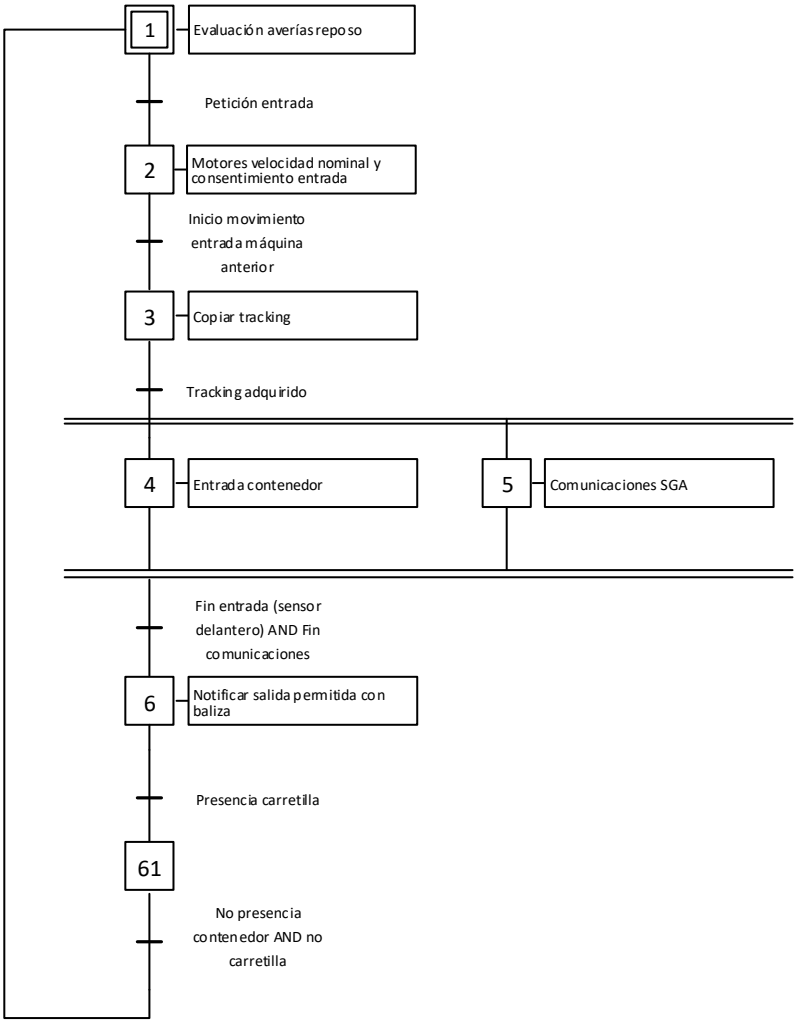
Figura 6.13. Diagrama GDMMA de las máquinas.

A continuación, se describen los comportamientos de las distintas máquinas según GRAFCET de primer nivel. Los programas no se han desarrollado en SFC, sin embargo, es una buena herramienta para identificar el comportamiento.

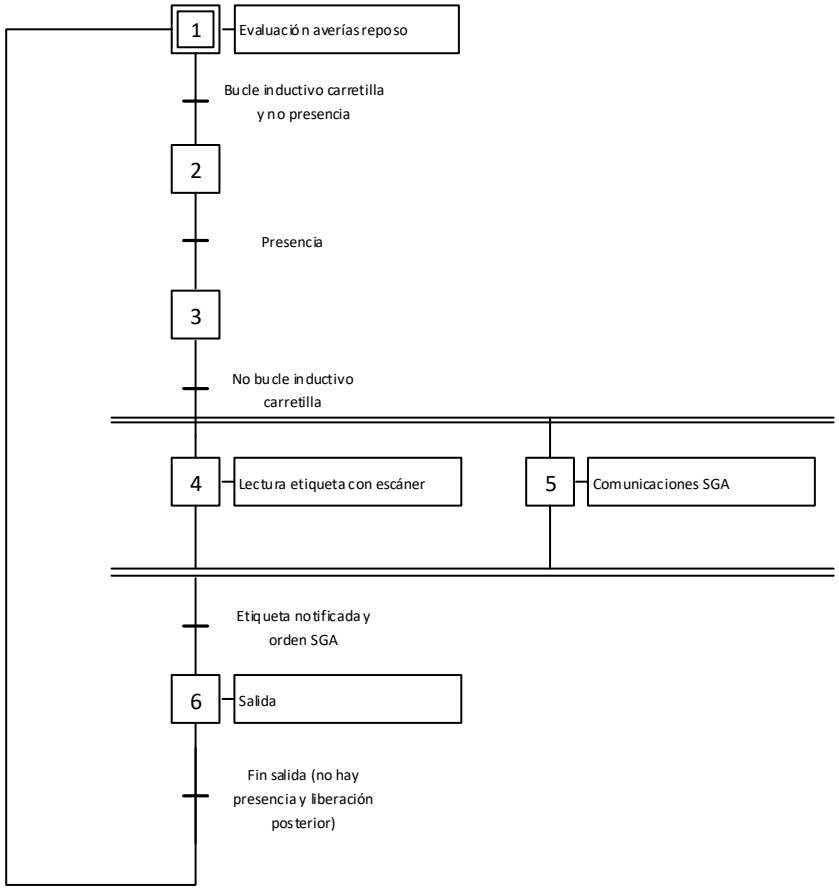
Transportador simple.



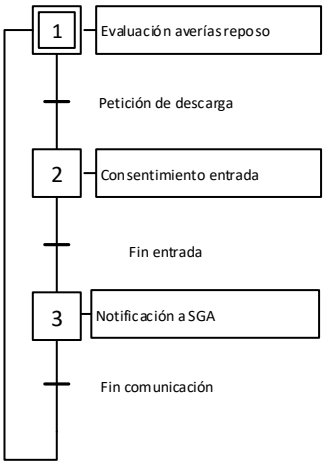
Transportador simple con salida carretilla



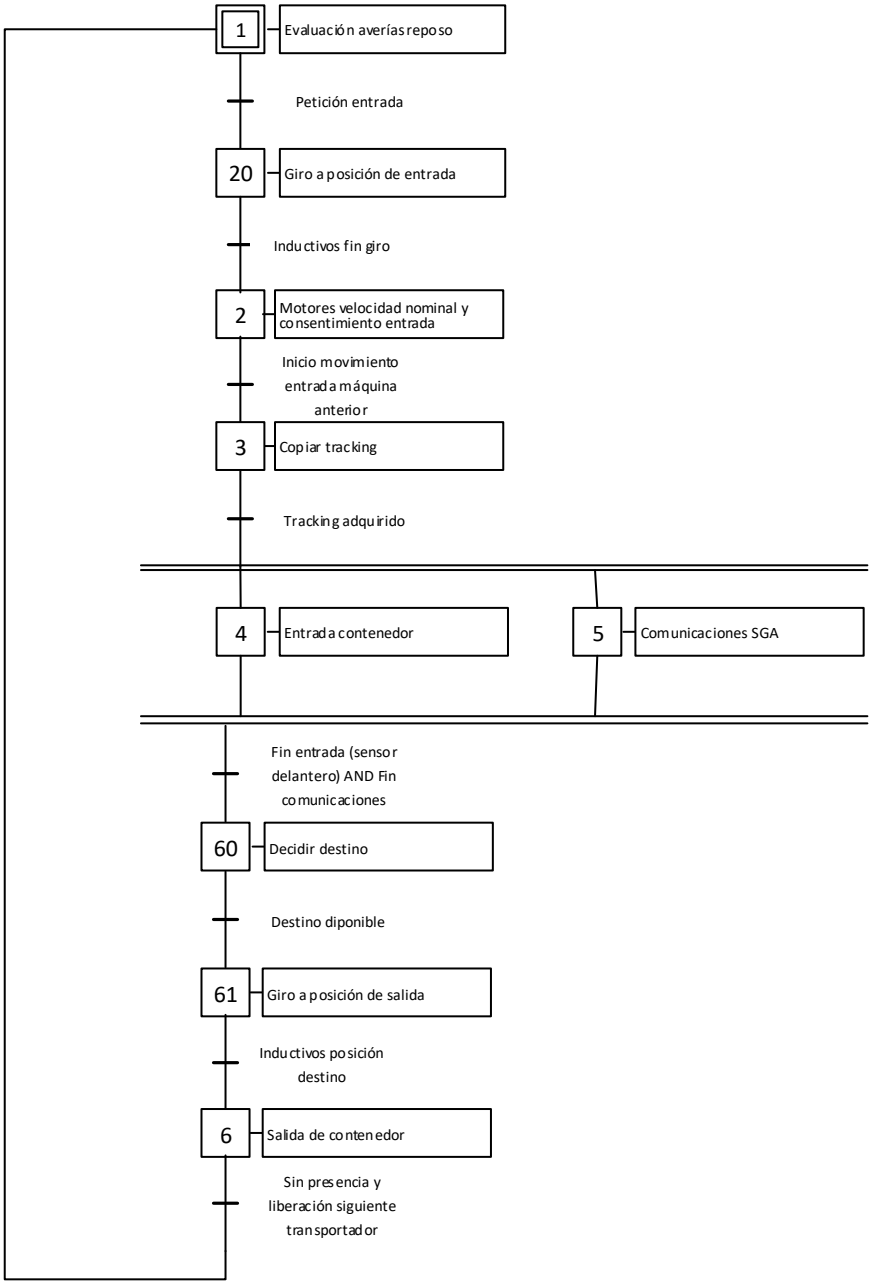
Transportador simple con entrada carretilla



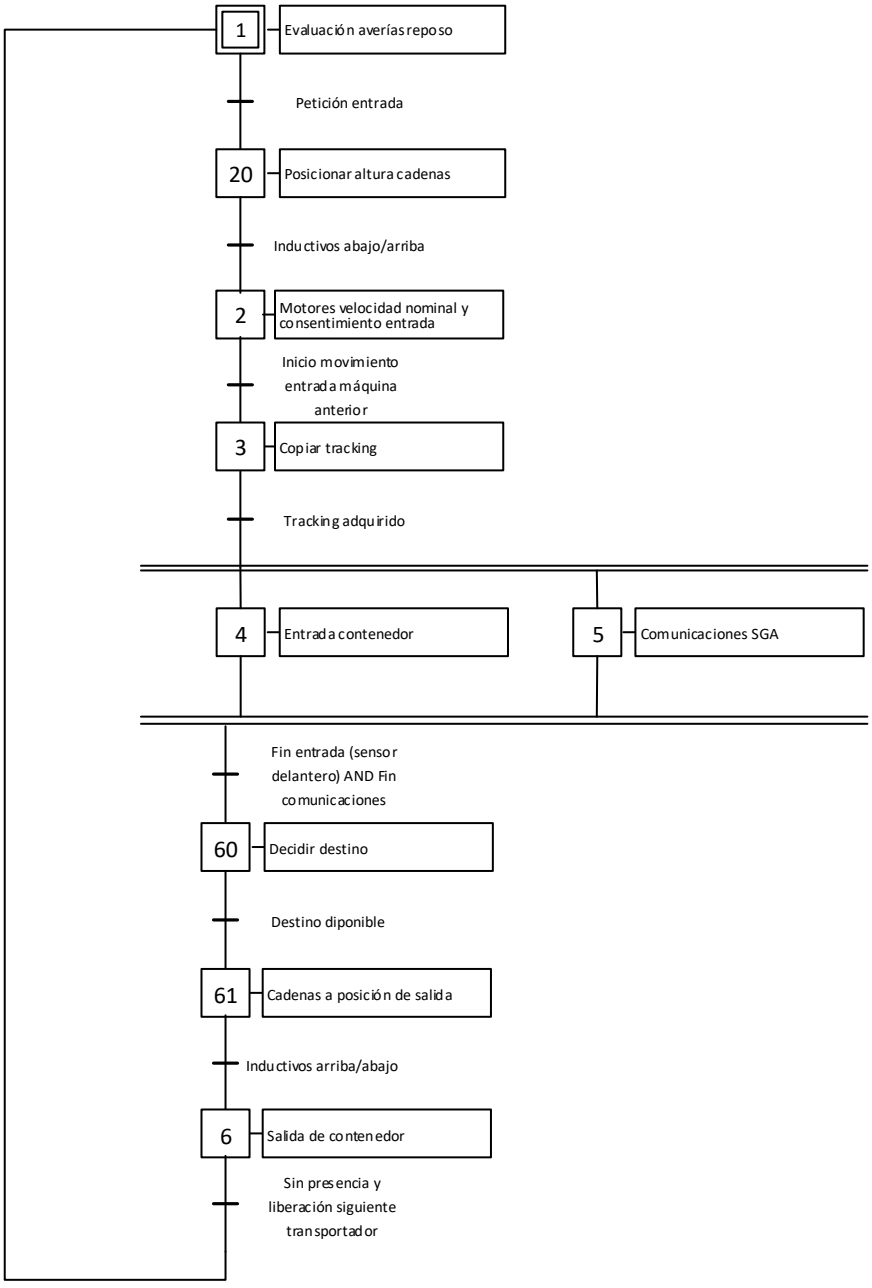
Transportador de descarga por gravedad



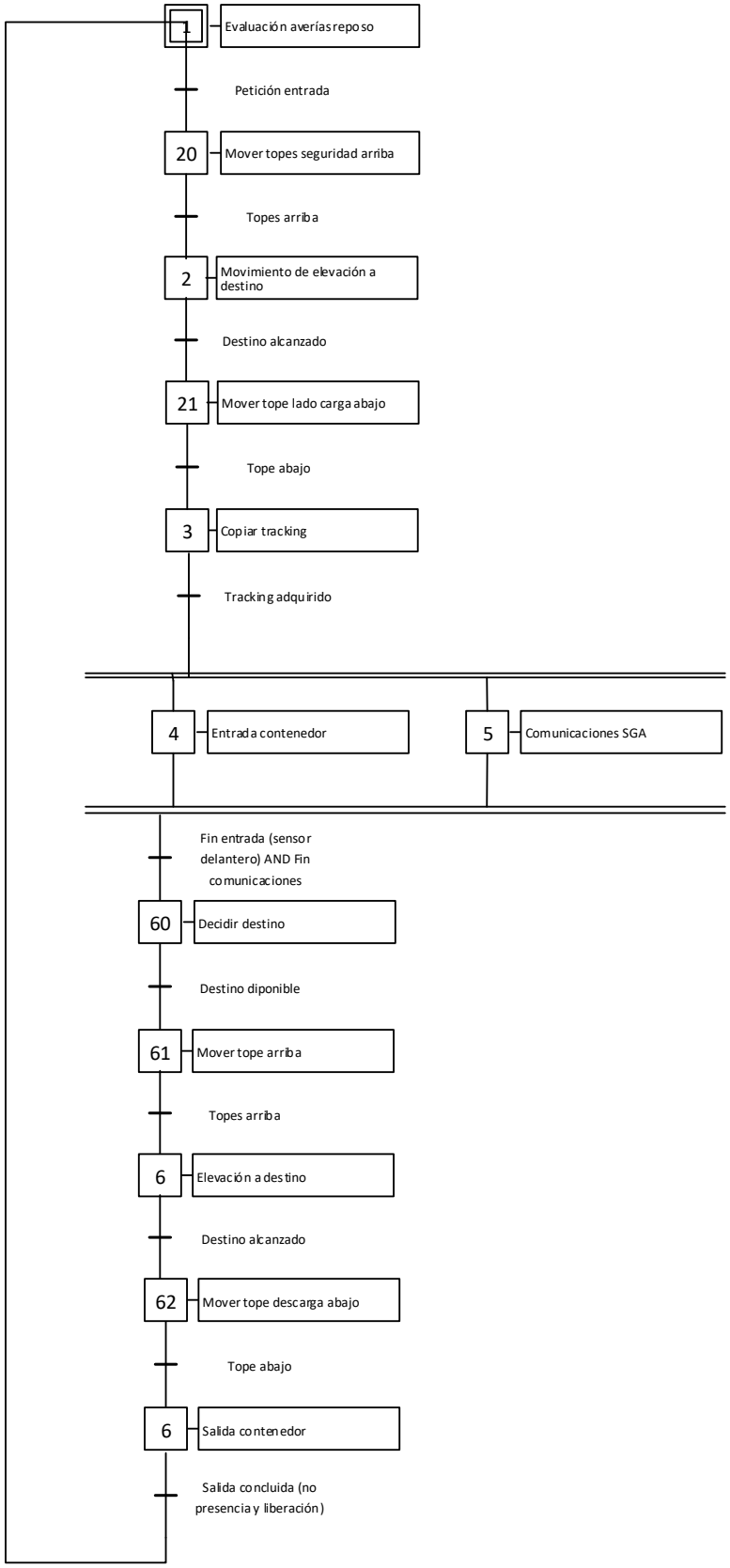
Transportador giratorio



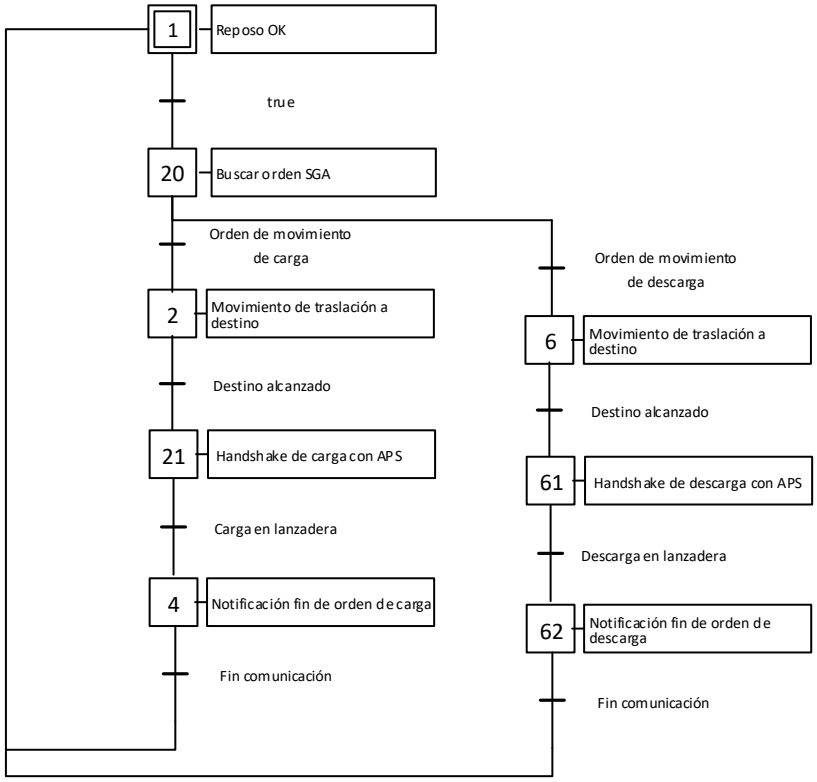
Transportador de transferencia mixta.



Elevador

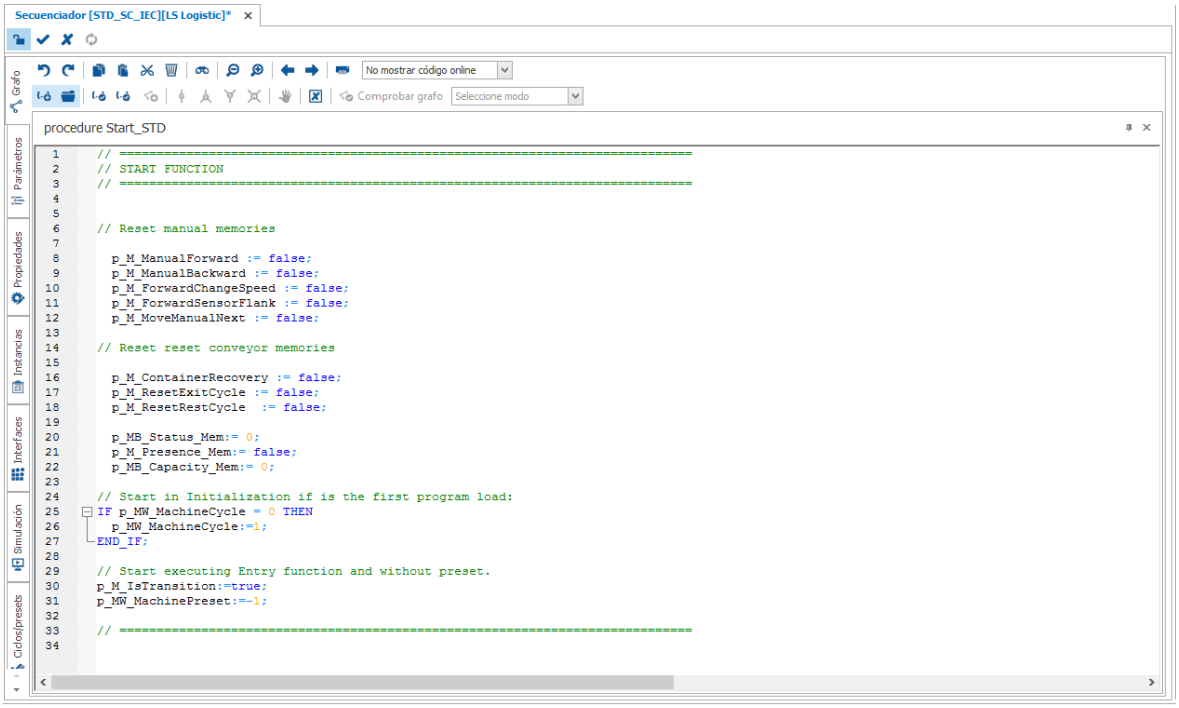

1
Evaluación averías reposo
Petición entrada
20
Mover topes seguridad arriba
Topes arriba
2
Movimiento de elevación a destino
Destino alcanzado
21
Mover tope lado carga abajo
Tope abajo
3
Copiar tracking
Tracking adquirido
4
Entrada contenedor
5
Comunicaciones SGA
Fin entrada (sensor delantero) AND Fin comunicaciones
60
Decidir destino
Destino diponible
61
Mover tope arriba
Topes arriba
6
Elevación a destino
Destino alcanzado
62
Mover tope descarga abajo
Tope abajo
6
Salida contenedor
Salida concluida (no presencia y liberación)

Lanzadera

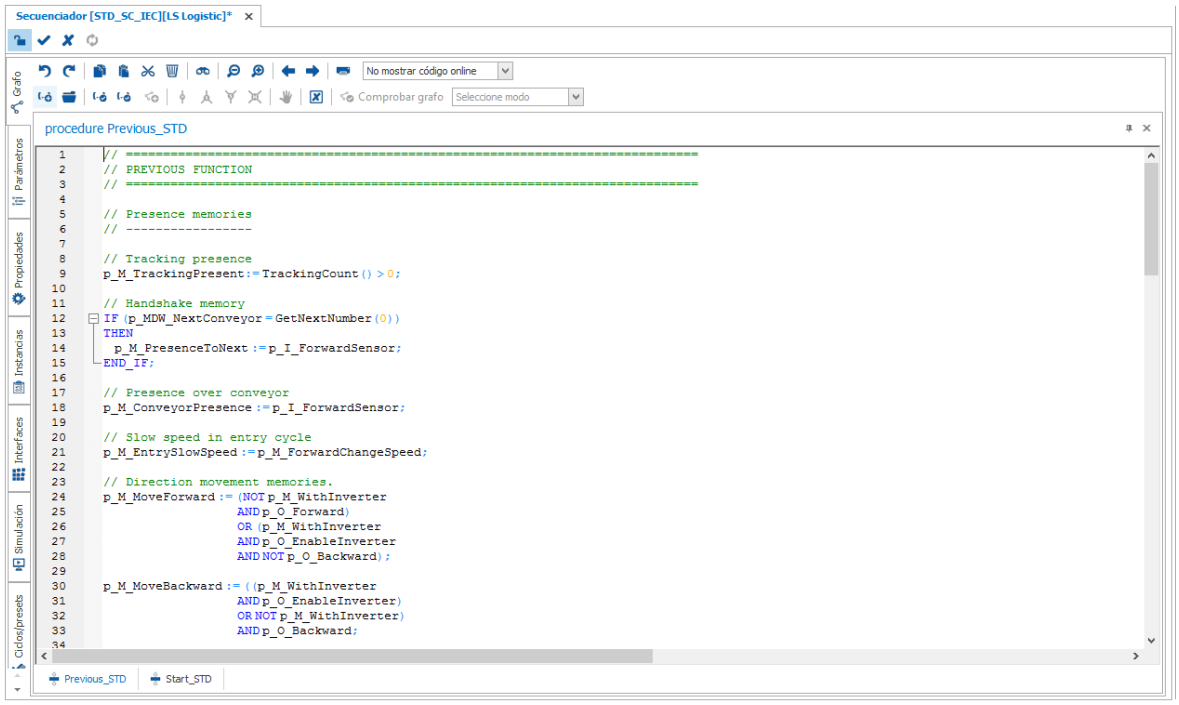


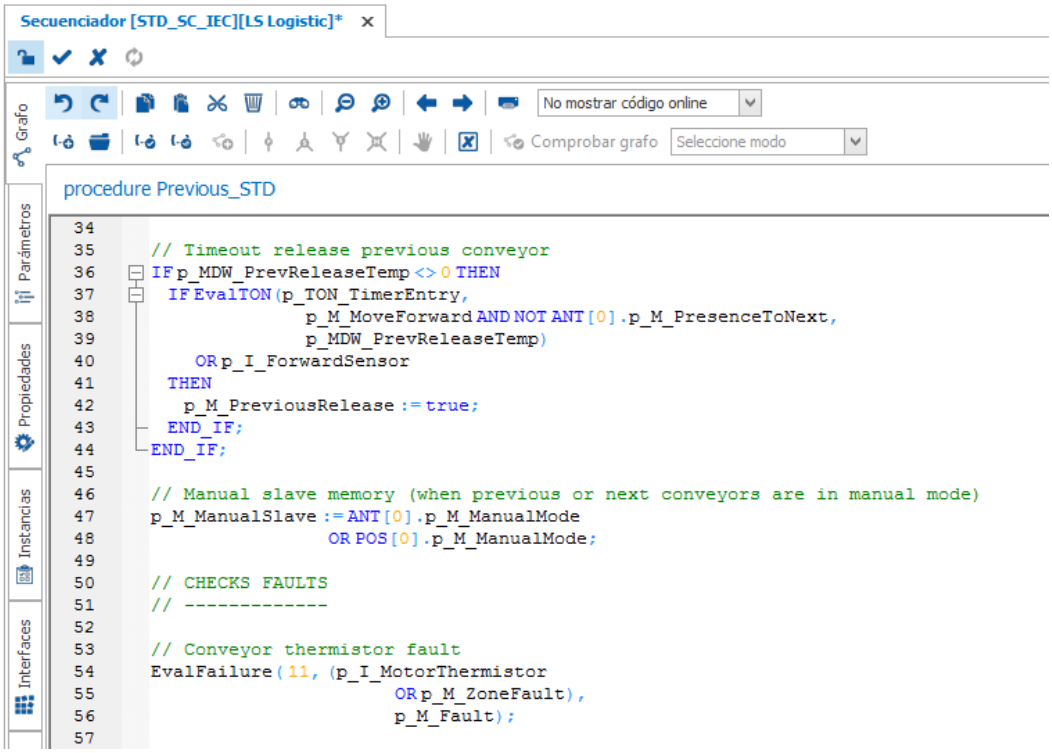
Por último, se expondrá el código del transportador simple para presentar la estructura de los programas. Aunque con ciertas diferencias de comportamiento, siempre se sigue una estructura similar para garantizar la legibilidad y la mantenibilidad del código durante la puesta en marcha y producción. No se expondrán todas las funciones sino una porción de las mismas para orientar sobre el comportamiento del programa.

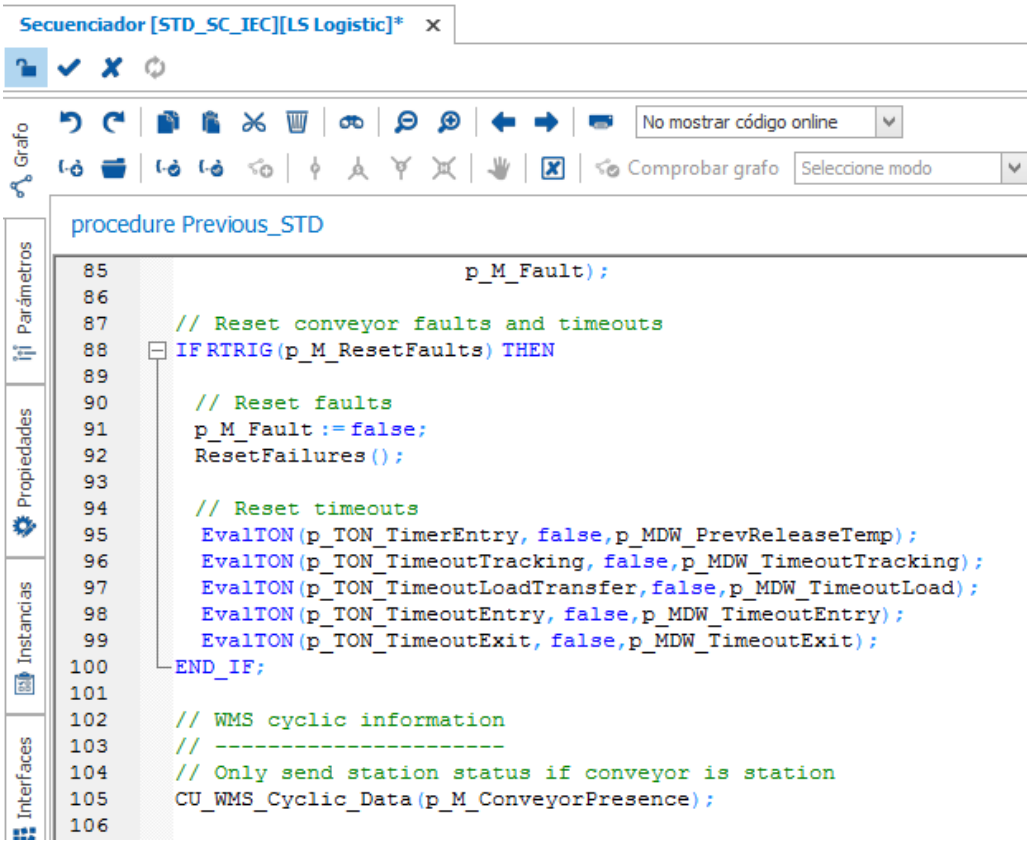
1. En primer lugar, se expone la función *Start*_STD. Ésta se ejecuta tras un arranque de la CPU posterior a una parada de la misma (equivale al típico OB100 de Siemens). Como se puede observar, aquí se inicializan todas las variables



2. La primera función que se ejecuta en cada ciclo de *scan* normal del programa (salvo ciclo de arranque), es la función *Previous_STD*. Aquí se evalúan ciertas variables a utilizar posteriormente (por ejemplo, presencia de Tracking), se comprueban averías necesarias en cualquier parte del proceso, el reset de las averías (rearme), etc.

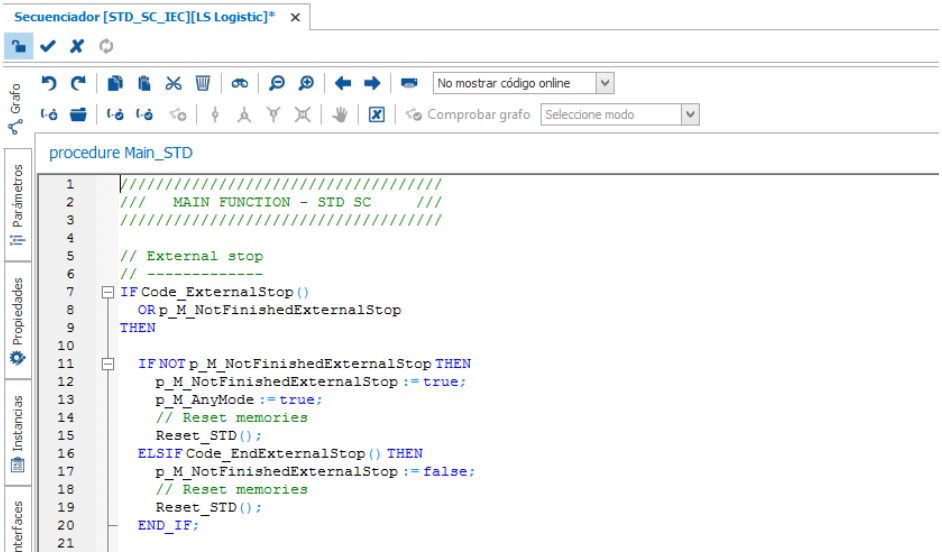




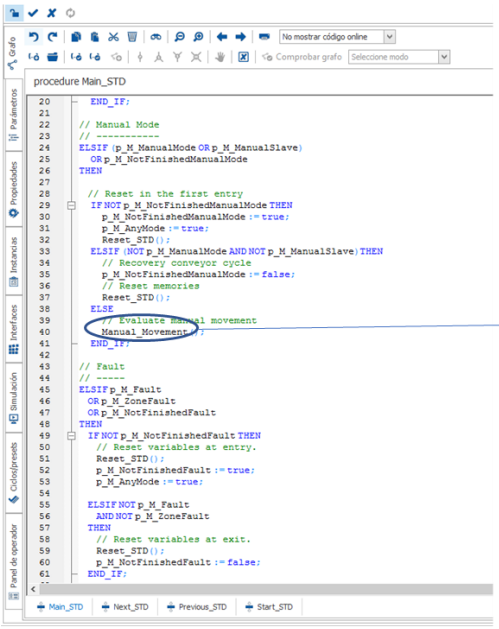


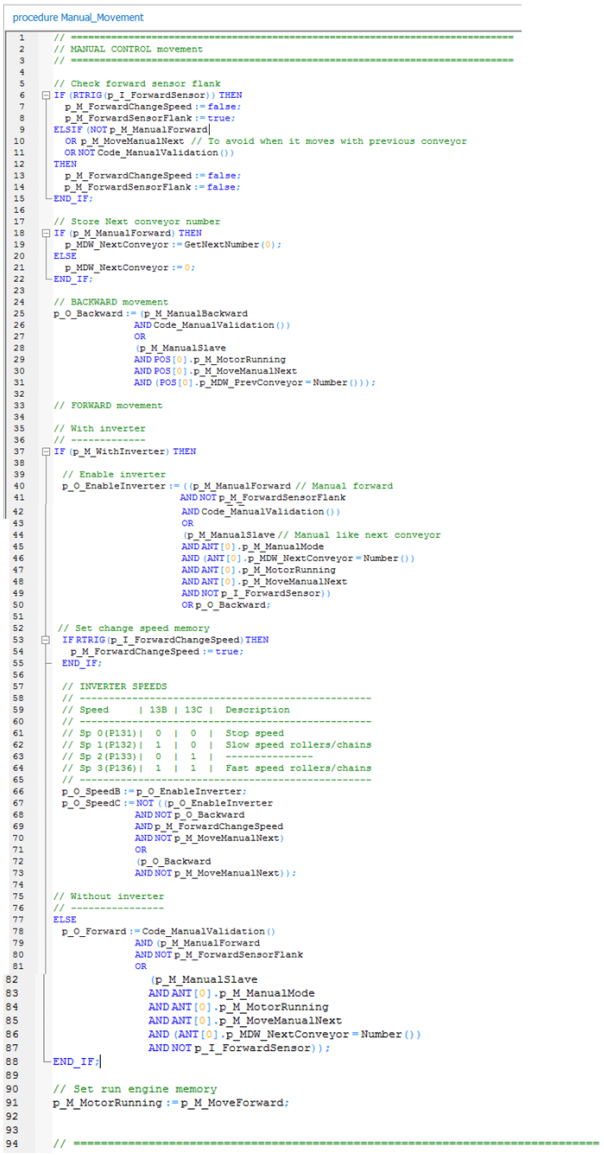
3. La siguiente función a ejecutarse, es la función MAIN_STD. Aquí se evalúan los ciclos de la máquina ya presentados en el GDMMA. En primer lugar, se evalúa el

estado de máxima prioridad: Parada externa (llave de mantenimiento, pasos peatonales, etc). Como puede observarse, en estos estados especiales (*external stop*, avería y manual) primero se ejecuta una función de Reset la cual limpia cualquier salida que pudiera haber quedado a *true*, así como las marcas necesarias.



El siguiente estado es el modo manual. Al entrar se resetean marcas y salidas. Posteriormente, se evalúa la función Manual_Movement que gestiona los movimientos. En la siguiente figura se muestra esta función, así como el estado Manual en el MAIN. También se aprecia el modo avería, el cual resetea las marcas y salidas, quedando en este estado hasta que se rearman las avería.

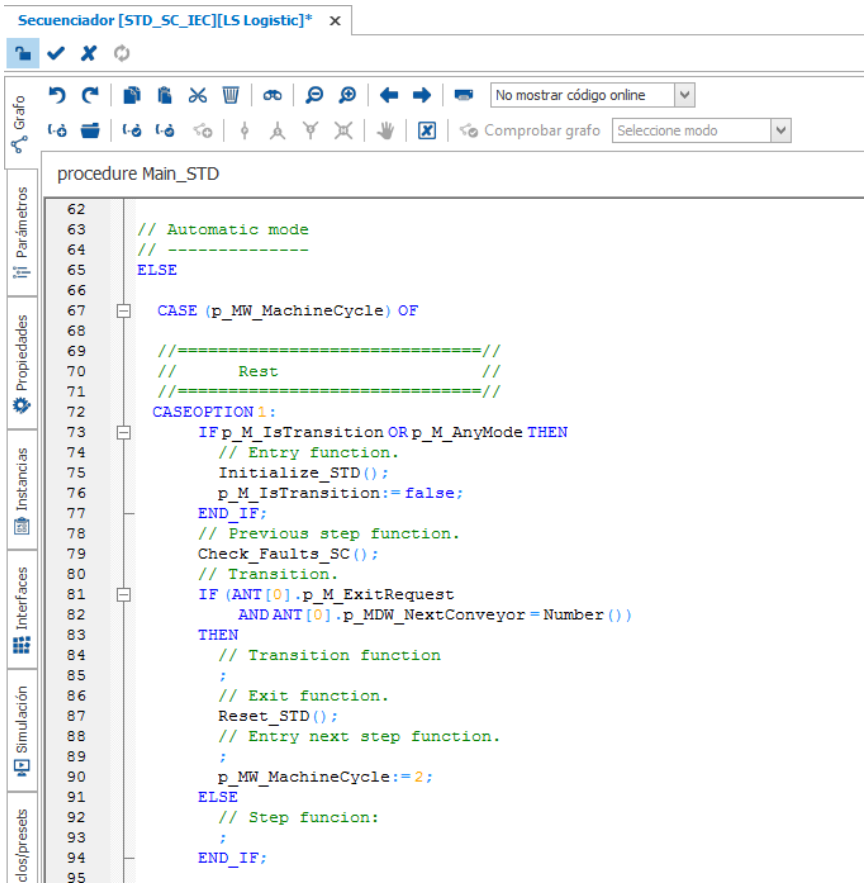




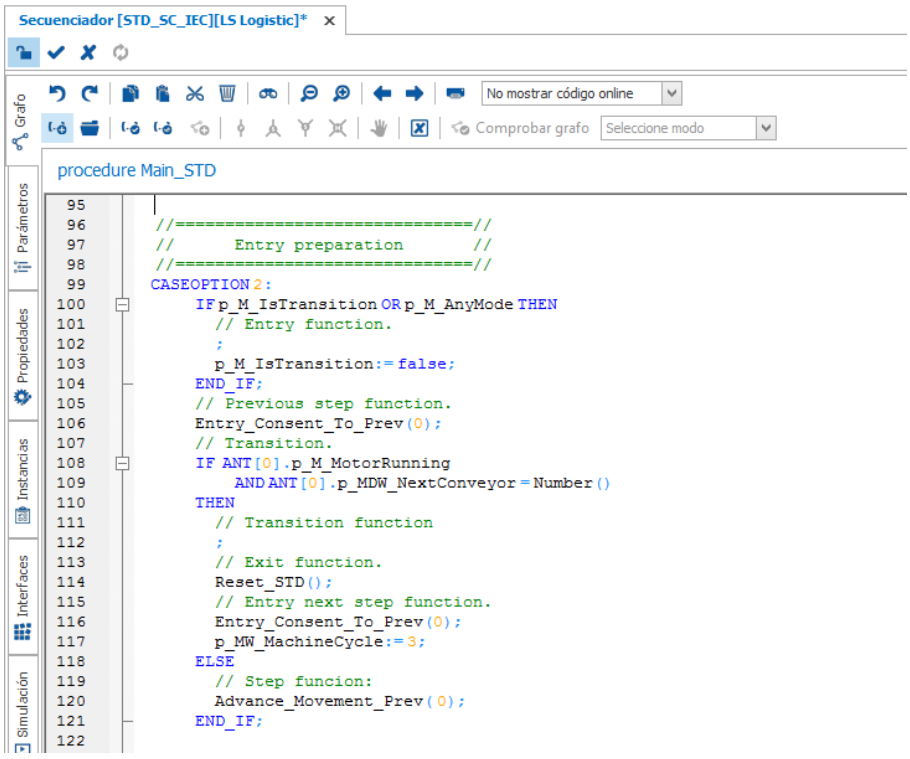
Por último, y si ninguno de los anteriores casos está activo, se ejecuta el estado automático. La estructura de este modo es una máquina de estados secuencial por medio de una estructura CASE y distintos estados del proceso. El modo de ejecución de cada etapa es la siguiente:

a. La primera vez en entrar en la nueva etapa se ejecuta una función (si es necesario).
b. En todo ciclo de *scan* se ejecuta una función específica (si es necesario).
c. Posteriormente se ejecuta una evaluación de cambio de estado, en caso de que se cumpla se modifica la variable del CASE y se ejecutan las funciones necesarias.
d. En caso de que no haya cambio de estado se puede ejecutar una función posterior.

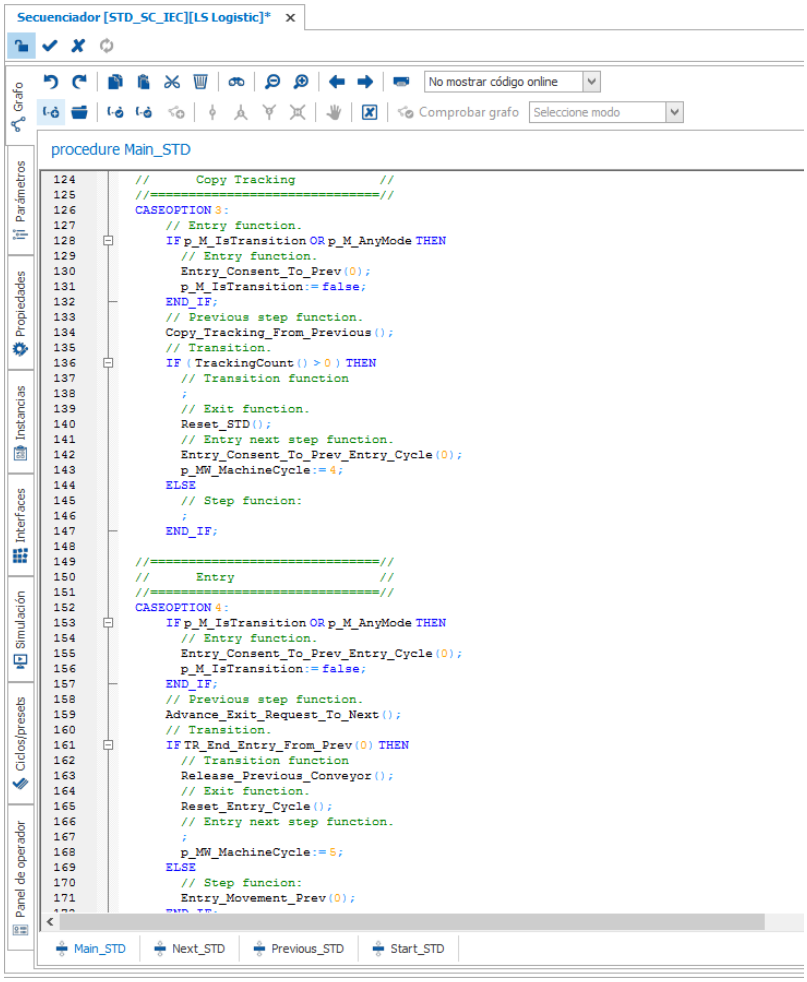
Todos los estados se corresponden a lo presentado en los GRAFCET anteriores.



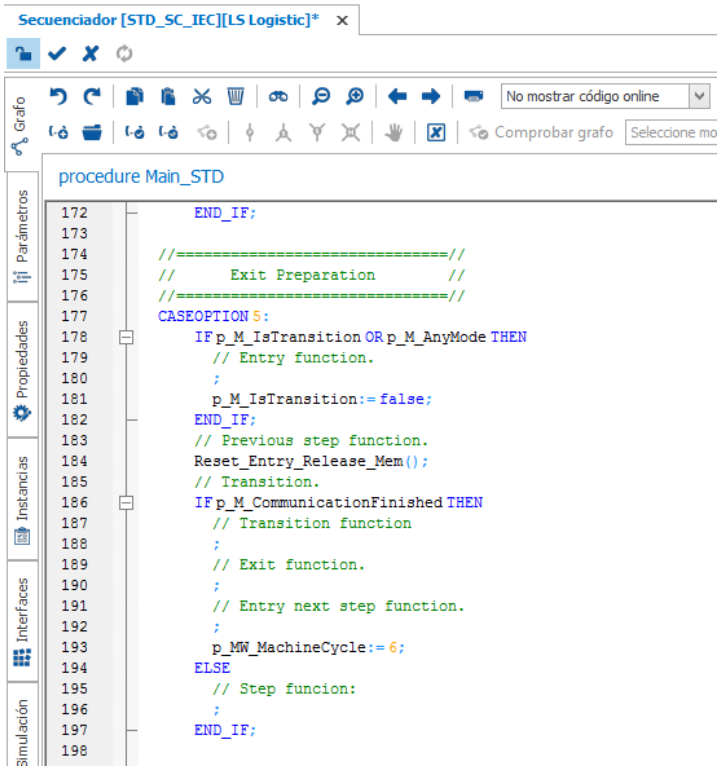

```
62
63  // Automatic mode
64  // --------------
65  ELSE
66
67    CASE (p_MW_MachineCycle) OF
68
69    //==============================//
70    //      Rest                    //
71    //==============================//
72   CASEOPTION 1:
73        IF p_M_IsTransition OR p_M_AnyMode THEN
74          // Entry function.
75          Initialize_STD();
76          p_M_IsTransition:= false;
77        END_IF;
78        // Previous step function.
79        Check_Faults_SC();
80        // Transition.
81        IF (ANT[0].p_M_ExitRequest
82            AND ANT[0].p_MDW_NextConveyor = Number())
83        THEN
84          // Transition function
85          ;
86          // Exit function.
87          Reset_STD();
88          // Entry next step function.
89          ;
90          p_MW_MachineCycle:= 2;
91        ELSE
92          // Step funcion:
93          ;
94        END_IF;
95
```



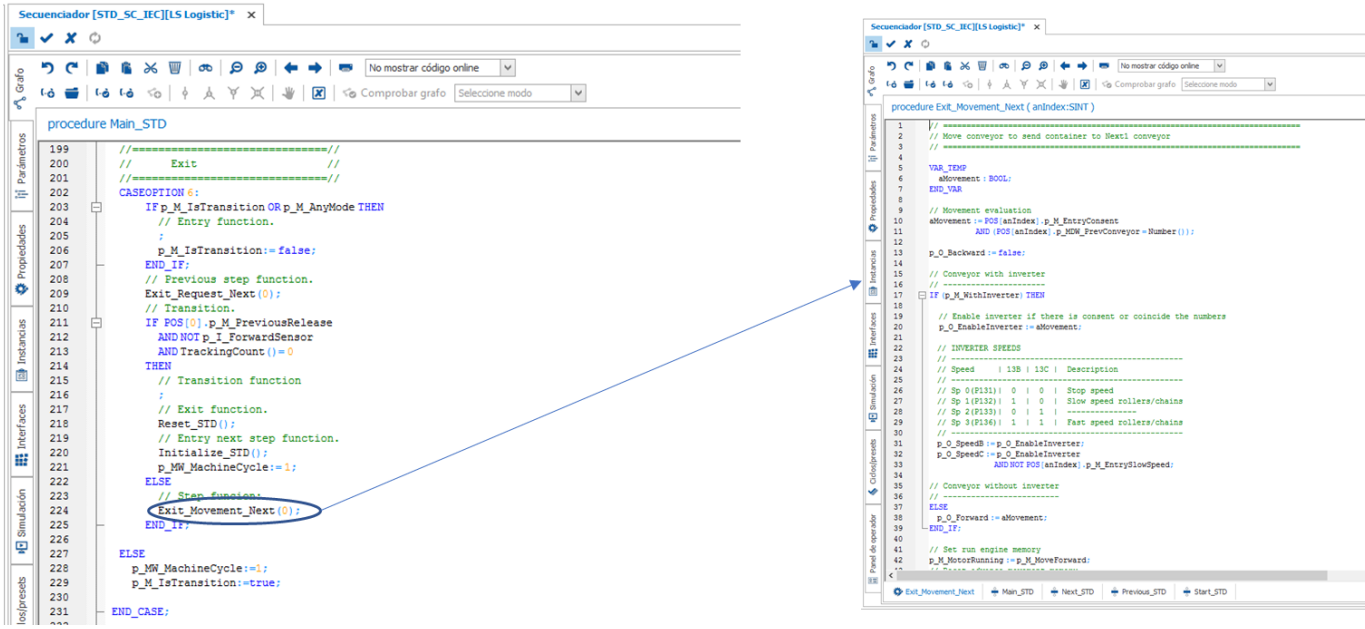

```
95
96    //==============================//
97    //      Entry preparation       //
98    //==============================//
99   CASEOPTION 2:
100       IF p_M_IsTransition OR p_M_AnyMode THEN
101         // Entry function.
102         ;
103         p_M_IsTransition:= false;
104       END_IF;
105       // Previous step function.
106       Entry_Consent_To_Prev(0);
107       // Transition.
108       IF ANT[0].p_M_MotorRunning
109           AND ANT[0].p_MDW_NextConveyor = Number()
110       THEN
111         // Transition function
112         ;
113         // Exit function.
114         Reset_STD();
115         // Entry next step function.
116         Entry_Consent_To_Prev(0);
117         p_MW_MachineCycle:= 3;
118       ELSE
119         // Step funcion:
120         Advance_Movement_Prev(0);
121       END_IF;
122
```



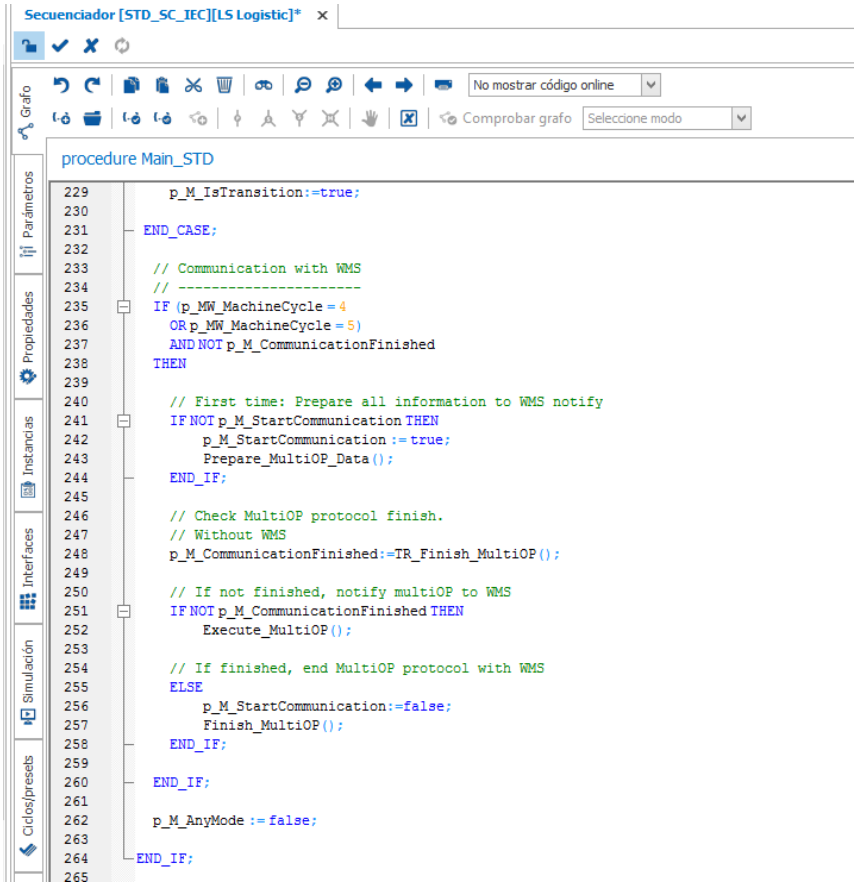

```
//      Copy Tracking           //
//==============================//
CASEOPTION 3:
    // Entry function.
    IF p_M_IsTransition OR p_M_AnyMode THEN
      // Entry function.
      Entry_Consent_To_Prev(0);
      p_M_IsTransition:= false;
    END_IF;
    // Previous step function.
    Copy_Tracking_From_Previous();
    // Transition.
    IF (TrackingCount() > 0) THEN
      // Transition function
      ;
      // Exit function.
      Reset_STD();
      // Entry next step function.
      Entry_Consent_To_Prev_Entry_Cycle(0);
      p_MW_MachineCycle:= 4;
    ELSE
      // Step funcion:
      ;
    END_IF;

//==============================//
//      Entry                   //
//==============================//
CASEOPTION 4:
    IF p_M_IsTransition OR p_M_AnyMode THEN
      // Entry function.
      Entry_Consent_To_Prev_Entry_Cycle(0);
      p_M_IsTransition:= false;
    END_IF;
    // Previous step function.
    Advance_Exit_Request_To_Next();
    // Transition.
    IF TR_End_Entry_From_Prev(0) THEN
      // Transition function
      Release_Previous_Conveyor();
      // Exit function.
      Reset_Entry_Cycle();
      // Entry next step function.
      ;
      p_MW_MachineCycle:= 5;
    ELSE
      // Step funcion:
      Entry_Movement_Prev(0);
```



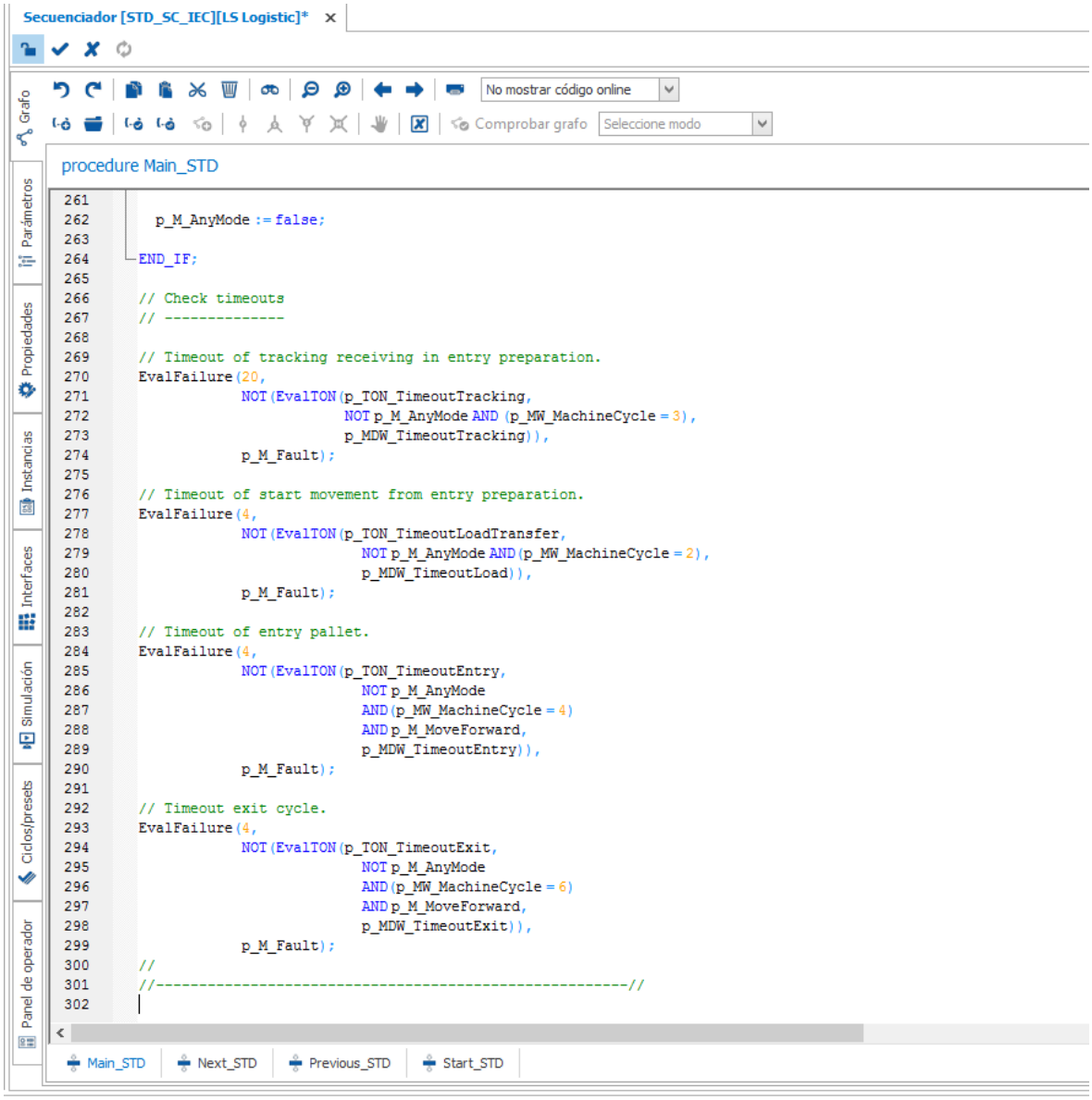

```
    END_IF;

//==============================//
//      Exit Preparation        //
//==============================//
CASEOPTION 5:
    IF p_M_IsTransition OR p_M_AnyMode THEN
      // Entry function.
      ;
      p_M_IsTransition:= false;
    END_IF;
    // Previous step function.
    Reset_Entry_Release_Mem();
    // Transition.
    IF p_M_CommunicationFinished THEN
      // Transition function
      ;
      // Exit function.
      ;
      // Entry next step function.
      ;
      p_MW_MachineCycle:= 6;
    ELSE
      // Step funcion:
      ;
    END_IF;
```



Tras el modo automático se evalúan acciones paralelas al mismo. Como se había explicado en los *Grafcet*, durante el proceso de entrada se realizan las comunicaciones con el SGA. No se entrará en detalle de estas funciones, básicamente realizan búsquedas de movimientos, notificaciones de evento o finales de orden.

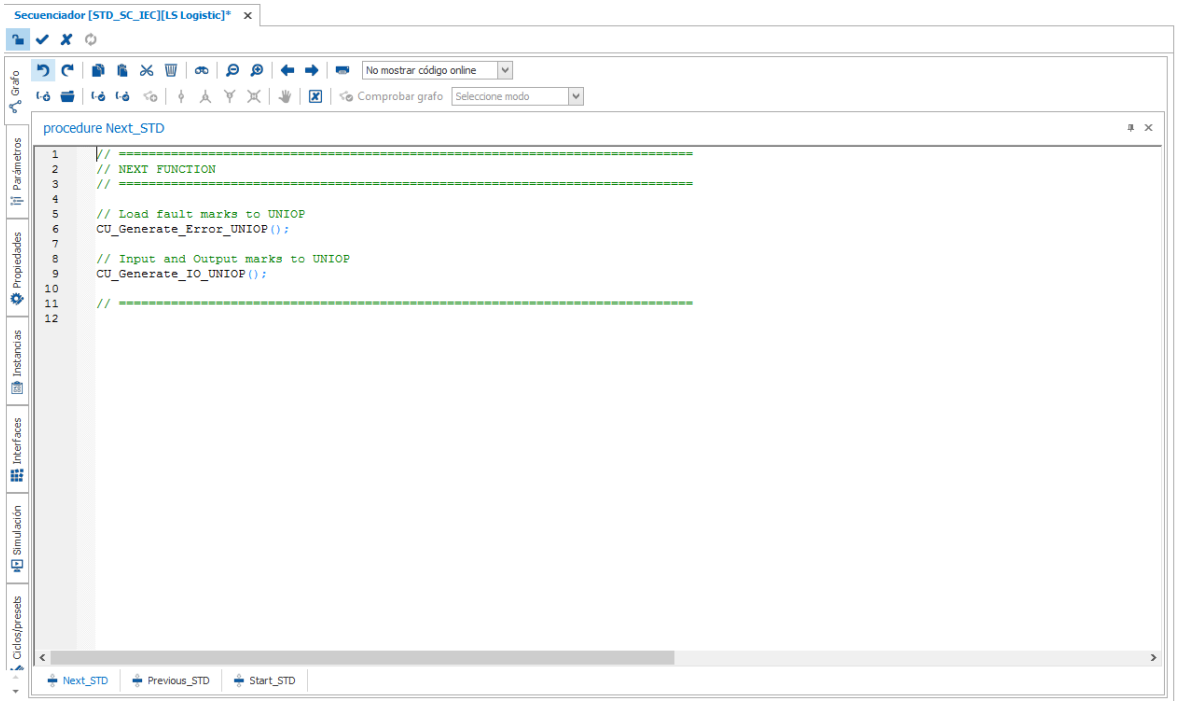


Para finalizar se evalúan averías asociadas al modo automático. Estos son llamados *timeouts* y se lanzan en caso de exceso de tiempo en alguna de las etapas. Por ejemplo, han pasado más de 30 segundos en etapa de salida, con los motores en marcha y sin haber perdido la presencia (podría ser indicio de motor bloqueado, pallet atascado, etc.).




```
261
262       p_M_AnyMode := false;
263
264   END_IF;
265
266   // Check timeouts
267   // --------------
268
269   // Timeout of tracking receiving in entry preparation.
270   EvalFailure(20,
271               NOT(EvalTON(p_TON_TimeoutTracking,
272                           NOT p_M_AnyMode AND (p_MW_MachineCycle = 3),
273                           p_MDW_TimeoutTracking)),
274               p_M_Fault);
275
276   // Timeout of start movement from entry preparation.
277   EvalFailure(4,
278               NOT(EvalTON(p_TON_TimeoutLoadTransfer,
279                           NOT p_M_AnyMode AND(p_MW_MachineCycle = 2),
280                           p_MDW_TimeoutLoad)),
281               p_M_Fault);
282
283   // Timeout of entry pallet.
284   EvalFailure(4,
285               NOT(EvalTON(p_TON_TimeoutEntry,
286                           NOT p_M_AnyMode
287                           AND(p_MW_MachineCycle = 4)
288                           AND p_M_MoveForward,
289                           p_MDW_TimeoutEntry)),
290               p_M_Fault);
291
292   // Timeout exit cycle.
293   EvalFailure(4,
294               NOT(EvalTON(p_TON_TimeoutExit,
295                           NOT p_M_AnyMode
296                           AND(p_MW_MachineCycle = 6)
297                           AND p_M_MoveForward,
298                           p_MDW_TimeoutExit)),
299               p_M_Fault);
300   //
301   //-----------------------------------------------------------//
302
```

4. Por último, tras el MAIN se ejecuta la función posterior. En este caso solo se realiza la llamada a dos funciones que actualizan el estado de la máquina en los paneles de operador.



6.3.4. Simulación

Una vez se dispone del código de control, es hora de realizar pruebas para realizar un ajuste más fino del código y evaluar errores. Esto se realiza, como ya ha sido comentado, mediante un *plug-in* de simulación 3D. Definiendo el *layout* del almacén (que también será utilizado como SCADA) se puede probar el programa de control. La fiabilidad y comportamiento tan real conseguidos mediante esta tarea hace que realmente se consiga reducir el tiempo de puesta en marcha en obra y se obtengan programas más estables.

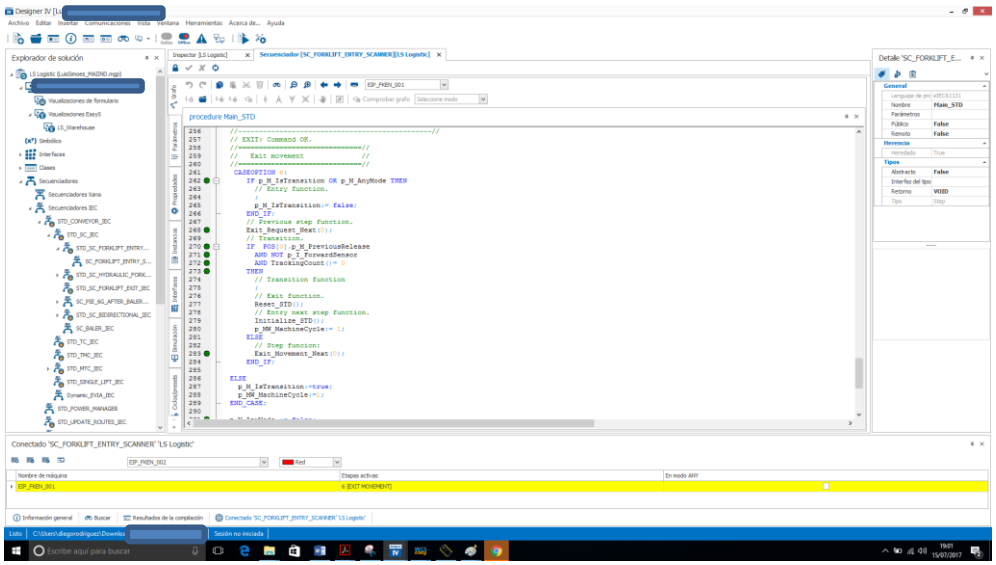


Figura 6.14. Código online durante una simulación.

Figura 6.15. Simulación de la instalación.

6.3.5. Puesta en marcha

Tras realizar las pruebas en oficinas y una vez que se decide que se tiene el código final del proyecto, es el momento de realizar la puesta en marcha. El momento idóneo para que el departamento de control se desplace para optimizar el tiempo y que todo vaya lo más fluido posible, es una vez que todas las máquinas estén montadas y cableadas, ya que la instalación tiene que cumplir con unos requisitos mínimos eléctricos. Toda esta labor, es llevada a cabo por montadores mecánicos y eléctricos que pueden ser de Mecalux o de una empresa subcontratada. A partir de este momento, el departamento de control con este programa ya desarrollado, viajará al lugar y comenzará a validar las distintas señales de entrada/salida, movimientos en manual y poco a poco ir realizando movimientos en automático. Una vez que está todo validado, será el turno del departamento de informática, encargado de implantar el sistema de gestión de almacén. Aunque existen personas escépticas, después de este primer proyecto con esta nueva herramienta se ha demostrado la reducción de tiempos desde que control va a obra y hasta que se deja validado para el sistema de gestión de almacén. Principalmente por los avances realizados en las pruebas en oficina.

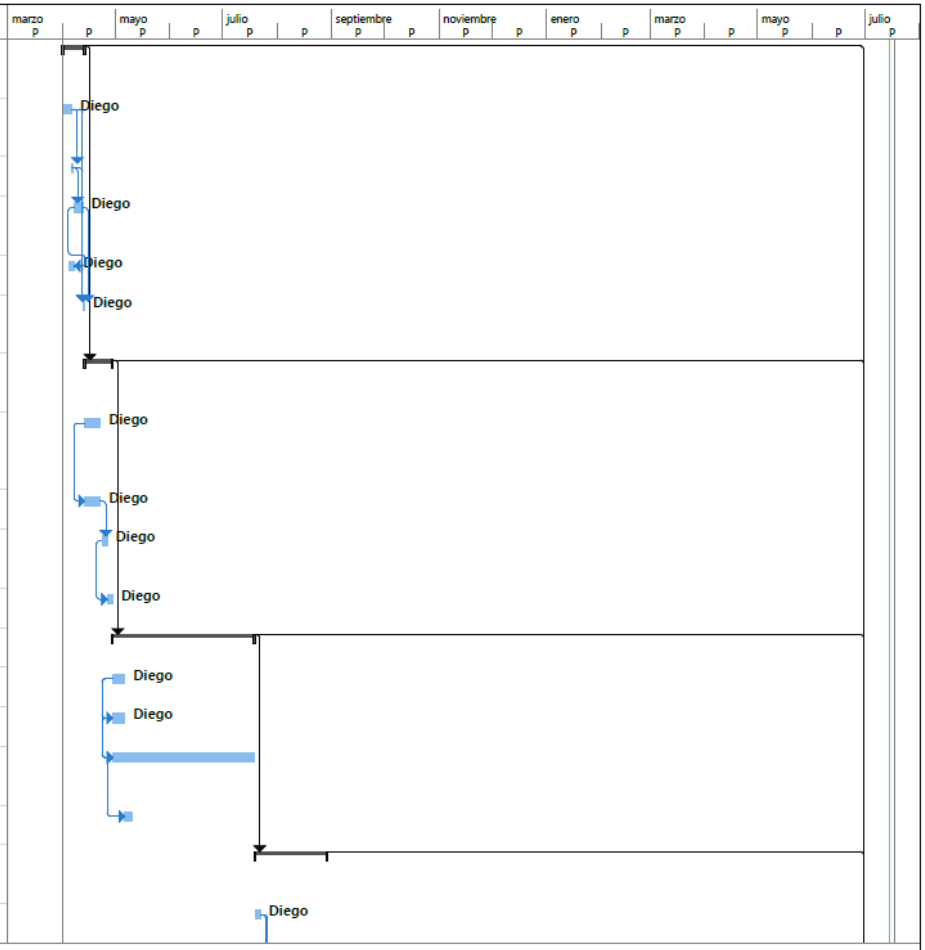
Figura 6.16. Aspecto de la instalación durante la puesta en marcha.

Figura 6.17. Aspecto de la instalación durante la puesta en marcha.

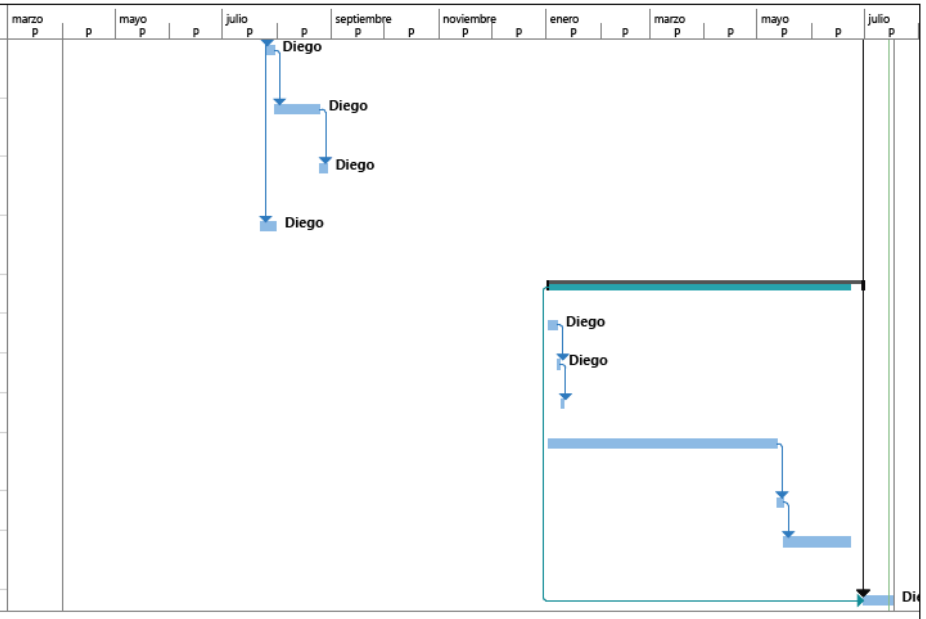
7. Planificación

En este capítulo se presenta un diagrama de Gantt (se ha utilizado Microsoft Project) con las tareas que componen el proyecto. Cabe destacar que se ha añadido un mes de vacaciones en agosto, aunque realmente las vacaciones fueron repartidas a lo largo del año, ya que de esta forma es más sencillo. Además, se ha establecido una jornada de 37 horas semanales porque parece un valor más fiable a las 40 que realmente son (en la jornada existen descansos, llamadas inesperadas, etc.). Por otro lado, hay que indicar que el lector debe tomar estos datos como orientativos ya que en un día laboral normal no solo se trabajaba en este proyecto, sino que paralelamente se realizan otras tareas (resolución de incidencias, soporte a cliente, pruebas, fallos no controlados en PC, entre otras). Por otro lado, existe un intervalo temporal desde que se finalizó el proyecto actual y hasta que se empezó con la instalación real. Entre medias hubo otros proyectos, mejoras, etc.

Id	EDT	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	1	**Estudio Inicial**	**8,93 días**	**vie 01/04/16**	**mié 13/04/16**	
2	1.1	Formación	3,2 días	vie 01/04/16	mié 06/04/16	
3	1.2	Instalación RSLogix/TIA	3 horas	mié 06/04/16	mié 06/04/16	2
4	1.3	Compatibilidad exportación código RS/TIA	3,2 días	jue 07/04/16	mar 12/04/16	3
5	1.4	Evaluación tamaño de código	2,13 días	lun 04/04/16	jue 07/04/16	4CF+1,07 días
6	1.5	Documentación información recopilada	5 horas	mar 12/04/16	mié 13/04/16	2;3;4;5
7	2	**Estructuración inicial**	**12,93 días**	**mié 13/04/16**	**vie 29/04/16**	**1**
8	2.1	Programación proyecto base transportador simple TIA/RS	5,33 días	mié 13/04/16	vie 22/04/16	
9	2.2	Diseño código estándar ST	5,33 días	mié 13/04/16	vie 22/04/16	8CC
10	2.3	Ayuda sintaxis y estructura compilador ST	2,13 días	sáb 23/04/16	mar 26/04/16	9
11	2.4	Pruebas proyectos compilados sencillos	2,13 días	mar 26/04/16	vie 29/04/16	10CC+1,07 días
12	3	**Programación máqinas estándar**	**60 días**	**vie 29/04/16**	**lun 18/07/16**	**7**
13	3.1	Adquisición planos eléctricos	4 días	vie 29/04/16	vie 06/05/16	
14	3.2	Información sobre comportamiento	4 días	vie 29/04/16	vie 06/05/16	13CC
15	3.3	Programación ST	60 días	vie 29/04/16	lun 18/07/16	13CC;14CC
16	3.4	Documentación asociada	2 días	vie 06/05/16	mar 10/05/16	15CC+5 días
17	4	**Modelos de simulación**	**31 días**	**mar 19/07/16**	**lun 29/08/16**	**12**
18	4.1	Prototipos máquinas en 3D	2 días	mar 19/07/16	vie 22/07/16	



Id	EDT	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
19	4.2	Información posicion sensores, motores, etc	2 días	mar 26/07/16	sáb 30/07/16	18
20	4.3	Generación modelos para máquinas estándar	15 días	sáb 30/07/16	jue 25/08/16	19
21	4.4	Reporte bugs	2 días	jue 25/08/16	lun 29/08/16	20
22	4.5	Proyectos pruebas	5 días	vie 22/07/16	dom 31/07/16	18
23	**5**	**Proyecto instalación real**	**138 días**	**lun 02/01/17**	**vie 30/06/17**	
24	5.1	Análisis ingeniería eléctrica	3 días	lun 02/01/17	sáb 07/01/17	
25	5.2	Análisis planos implantación	1 día	sáb 07/01/17	dom 08/01/17	24
26	5.3	Layout SCADA a partir de CAD	2 días	lun 09/01/17	mar 10/01/17	25
27	5.4	Generación de proyecto, pruebas, resolucion bugs	100 días	lun 02/01/17	jue 11/05/17	
28	5.5	Simulación en oficina	2 días	jue 11/05/17	lun 15/05/17	27
29	5.6	Puesta en marcha	30 días	lun 15/05/17	jue 22/06/17	28
30	**6**	**Documentación TFM**	**10 días**	**vie 30/06/17**	**lun 17/07/17**	**1;7;12;17;23CC**



Proyecto: Planificación Tempor
Fecha: sáb 15/07/17

Tarea
División
Hito
Resumen
Resumen del proyecto
Tarea inactiva
Hito inactivo
Resumen inactivo
Tarea manual
solo duración
Informe de resumen manual
Resumen manual
solo el comienzo
solo fin
Tareas externas
Hito externo
Fecha límite
Progreso
Progreso manual

Página 2

8. Presupuesto

8.1.- COSTE DE MATERIALES

Aquí se incluyen las herramientas que se han utilizado para permitir avanzar y desarrollar el proyecto. No se incluye el precio de los elementos hardware de Rockwell ya que por el momento solo se tiene la petición de oferta, sin haber adquirido los productos a día de hoy.

NÚMERO	CONCEPTO	CANTIDAD	COSTE/UNIDAD	COSTE TOTAL
1	Licencia TIA Portal	1	2362 €	2362 €
2	Licencia RSLogix Studio 5000	1	2.930,50 €	2.930,50 €
3	Fuente de alimentación SIMATIC S7-1500	1	138,75 €	138,75 €
4	CPU SIMATIC 1516-3 PN/DP	1	2550 €	2550 €
5	Perfil soporte SIMATIC S7-1500	1	13,77 €	13,77 €
6	Tarjeta de memoria S7 para modelos 1X00	1	158,10 €	158,10 €
			TOTAL	**8153,12 €**

8.2.- COSTE DE MANO DE OBRA

En este segundo capítulo se recogen los costes asociados a la mano de obra empleada para la ejecución del proyecto. Aunque numerosas personas de varios departamentos se han visto implicadas, solo se tendrán en cuenta aquellas desde el punto de vista del departamento

de automatización. Además, puesto que se trabaja como profesional contratado, el precio de la hora no concuerda completamente con el presentado, ya que el sueldo es mensual y no se cobra por horas de ingeniero. De todas maneras, se presenta de esta última forma como es habitual.

NÚMERO	CONCEPTO	HORAS	COSTE/UNIDAD	COSTE TOTAL
1	Estudio previo, formación y recopilación de información	60	22 €	1.320 €
2	Diseño y estructuración de código	80	22 €	1.760 €
3	Compatibilidad exportación de código	8	22 €	176 €
4	Diseño junto a informático de compilador ST (solo guía sobre sintaxis, estructuras, etc.)	30	22 €	660 €
5	Programas de control repositorio	320	22 €	7.040 €
6	Modelos 3D simulación repositorio	100	22 €	2.200 €
7	Proyecto beta instalación real	800	22 €	17.600 €
8	Puesta en marcha instalación real	240	22 €	5.280 €
			TOTAL COSTE DE MANO DE OBRA	**36.036 €**

8.3.- PRESUPUESTO FINAL

Coste total de la ejecución material del proyecto. Hay que entender este presupuesto como el valor del proyecto en su conjunto (no como el valor cobrado por el trabajador).

NÚMERO	CONCEPTO	SUBTOTAL	TOTAL
1	Materiales y Equipos		8.153,12 €
2	Mano de obra		3.6036 €
3	Gastos generales (6 %)		2.162,16 €
	Subtotal 1	46351,28 €	
4	Beneficio industrial (13 %)		6.025,67 €
	Subtotal 2	52.376,9464 €	
5	I.V.A. (21 %)		10.999,1587 €
		IMPORTE FINAL	**63.376,10 €**

Ascendiendo el importe final a la cantidad de:

63.376,1051 € - Sesenta y tres mil trescientos setenta y seis con diez céntimos

Gijón, julio de 2017

Diego Rodríguez García

9. Conclusiones

En el presente proyecto se ha diseñado un sistema de generación de código multiplataforma para PLCs. Básicamente, se permite codificar el comportamiento de máquinas de instalaciones de Mecalux para facilitar la reutilización de código y obtener programas más sencillos, seguros, probados y reducir los tiempos de puesta en marcha. Para la consecución de este trabajo se ha trabajado en conjunto con diferentes departamentos de ingeniería de software, diseño 3D e ingenieros de control. Como prueba del resultado, se ha presentado la utilización de esta solución para una instalación real. Los aspectos más relevantes de este proyecto son los siguientes.

9.1.- COMPATIBILIDAD

Actualmente se admite compatibilidad con PLCs de Allen Bradley y Siemens, los cuales siguen en mayor o menor medida las directrices establecidas por la norma IEC-61131. No obstante, se ha tenido que modificar la generación para una y otra plataforma ya que, aunque la base es la misma, ni el código generado (sintaxis de llamadas a funciones, etc.) es completamente igual, ni el lenguaje en el que se deben exportar los ficheros fuente. En un futuro, no sería complicado añadir a la lista otras compañías de elementos de control, siempre y cuando admitan el lenguaje de Texto Estructurado y ciertas restricciones para permitir la importación de código. No obstante, todo esto queda supeditado a la demanda de nuevos clientes con PLCs específicos, ya que, por el momento, las dos marcas citadas, junto con el *SoftPLC* Galileo, abarcan toda la demanda.

9.2.- HERRAMIENTA DE DESARROLLO

Hasta este momento existía una herramienta ya implementada y utilizada en Mecalux que era específica para programas desarrollados en Galileo. Con el resultado que aquí se presenta, se ha conseguido portar dicho entorno a otro distinto, compatible con la norma actual más relevante de lenguajes de programación de PLCs, y que, por ello, abre un abanico de posibilidades en la exportación y portabilidad de código. Esta herramienta, gracias a la

labor de todos los departamentos implicados, ha conseguido llegar a mejorar su rendimiento, facilidad para el usuario y fiabilidad.

9.3.- ESTÁNDAR DE CONTROL

Una vez con los instrumentos adecuados, se han desarrollado programas de control estándar para todas las máquinas ofrecidas por Mecalux. El propósito de esto ha sido obtener programas estándar, ya probados, que faciliten la creación de nuevos proyectos mediante la reutilización de código. De esta forma se llegarán a conseguir soluciones más estables y con menos errores de funcionamiento. Este estándar no es definitivo y se va optimizando a medida que se va utilizando, recomendando a todos los usuarios de los departamentos de operaciones de control de Mecalux y subcontratas que informen sobre mejoras que ellos consideren. Por último, el departamento de ingeniería de software ha ofrecido un sistema de simulación para observar el comportamiento del código y corregir aspectos de forma rápida en oficina.

9.4.- IMPRESIONES FINALES

Para el desarrollo del proyecto se ha consultado una amplia variedad bibliográfica y distintos conocimientos técnicos de multitud de personas con gran experiencia. Esto aporta a la memoria una base teórica plenamente válida. Apoyándose en ella se han conseguido superar con éxito las dificultades surgidas durante su progresión, consiguiendo implementar una instalación real de gran complejidad.

El resultado final es un sistema completamente funcional que, a través de la presente documentación, queda perfectamente identificado y definido (siempre manteniendo la confidencialidad). Este cumple con las expectativas inicialmente previstas, consiguiendo satisfacer todos y cada uno de los aspectos importantes concretados en el alcance y en los objetivos preliminares del proyecto. Se concluye, por tanto, que se está ante un sistema que facilita la labor de programación de soluciones de automatización, garantizando un correcto funcionamiento y reduciendo el coste de explotación de nuevos proyectos.

10.Bibliografía

- SIEMENS SIMATIC CONTROLLERS. http://w3.siemens.com/mcms/programmable-logic-controller/en/pages/default.aspx
- ROCKWELL AUTOMATION. http://www.rockwellautomation.com/es_ES/products-technologies/overview.page?
- MECALUX S.A., Manual de Uso Software: Sistema de Control Galileo IV.
- MECALUX S.A., Manual de sistemas de almacenaje.
- ORGANIZACIÓN PLCOPEN. http://www.plcopen.org/
- WIKIPEDIA, Programación Orientada a Objetos. https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos
- VÍCTOR MANUEL GONZÁLEZ SUÁREZ, Metodología de Análisis y Modelado de Sistemas de Evento Discretos Mediante Tecnicas Orientadas a Objetos.
- KARL-HEINZ JOHN y MICHAEL TIEGELKAMP, IEC-61131-3 Programming Industrial Automation Systems
- ASIGNATURA AeISA MAIIND, PLCopen: estandarización en la programación de los sistemas de control.
- ASIGNATURA Seguridad en Plantas Industriales MAIIND, Safety Bridge y elementos de seguridad.
- AUTOMATION STORE. https://www.theautomatinstore.com
- ASIGNATURA Proyectos y oficina técnica. Presupuestos para proyectos industriales.
- ASIGNATURA Proyectos y oficina técnica. Planificación con MS Project