UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

Trabajo Final de Grado

Grado en Ingeniería Electrónica Industrial y Automática

Vizcaíno Cabo, Raúl

ESTUDIO DEL PROCESO DE AUTOMATIZACIÓN DE UNA LÍNEA DE PRODUCCIÓN INDUSTRIAL

Memoria

Director: Delgado Prieto, Miguel

Codirector: Fernández Sobrino, Ángel

Convocatoria: Enero 2022

ÍNDICE

ÍNDICE DE ILUSTRACIONES Y TABLAS

1. Introducción

1.1 Objeto

El principal objetivo es el estudio del desarrollo de la programación de dos controladores lógicos programables (PLC) utilizando una arquitectura basada en el concepto de programación orientada a objetos, con la finalidad de realizar un proceso de automatización de una línea de producción industrial después de un análisis de los entornos de trabajo.

El conjunto de equipos a programar es una celda flexible ubicada en el laboratorio de automatización industrial en las instalaciones de la Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa (ESEIAAT).

La celda está constituida por varias cintas transportadoras las cuales transportan una serie de bandejas entre plataformas. Existen varias plataformas y unos elementos clave llamados retenedores formados por un sensor de presencia y un actuador pneumático para llevar a cabo el control y gestión de las bandejas.

1.2 Alcance

Para cumplir con los objetivos establecidos en este proyecto, será necesario conocer el sistema físico donde se va a desarrollar el proceso y sus componentes (sensores, actuadores, motores, etc.) y el entorno de programación que necesita los controladores incluyendo el programa facilitado.

Por lo tanto, se tratarán los siguientes puntos:

- Estudio de los procesos que desarrolla la celda a automatizar.
- Comprensión de los códigos de programa disponibles usados para la automatización básica de la celda.
- Estudio de los entornos de programación para el desarrollo del proyecto.
- Desarrollo software para la incorporación de la tecnología seleccionada.
- Estudio e implementación física en la celda de los dispositivos necesarios.
- Validaciones operacionales y funcionales del programa. El punto de partida del proyecto está vinculado a la existencia de programas desarrollados y validados anteriormente encargados del control de la celda de automatización.

Se excluye la profundización en la parte de comunicación y adquisición de datos en el que es necesario un sistema SCADA (Supervisory Control And Data Acquistion).

1.3 Requerimientos

- Instalación automatizada compuesta por PLCs, cintas, motores e instrumentación electrónica (p.e. sensórica).
- Softwares de entornos de programación de PLC
- Red física de comunicaciones.
- Conocimientos de programación POO (incluye lenguajes LD y ST) y de procesos industriales.

1.4 Justificación

La programación a desarrollar de un proceso industrial de una línea de producción pretende dar una finalidad a la instalación existente. Esta debe de estar basada en un proceso automatizado de gestión de productos sobre bandejas.

Existen varios tipos de producto los cuales contienen una identificación y depende de esta, tendrán un tipo de procesado diferente.

La instalación existente contiene varios PLCs que facilitan la gestión de dichos productos en lo que se refiere a funcionalidad referente al paso de las bandejas por las cintas, plataformas y retenedores.

Por lo tanto, es necesario realizar una gestión óptima con los programas base facilitados con una estructura definida para controlar toda la celda automatizada con un enfoque diferente en la forma de programar diferenciándose a como se ha hecho tradicionalmente.

Ventajas:

- No existe complejidad en el nivel de proceso.
- Desarrollo único en el nivel de control (PLC).
- Flexibilidad en cambios de funcionalidad.

Desventajas:

- Proceso sin adquisición de datos ni guardado de históricos de sondas.
- Nivel de supervisión poco flexible. El sistema funciona en un cliente local y no en una topología de red distribuida.

2. Consideraciones previas

2.1 Controlador lógico programable (PLC)

2.1.1 Definición

Un controlador lógico programable, más conocido por sus siglas en inglés PLC (Programmable Logic Controller) o por autómata programable, es una computadora utilizada en la ingeniería automática o automatización industrial, para automatizar procesos electromecánicos, electroneumáticos, electrohidráulicos, tales como el control de la maquinaria de la fábrica en líneas de montaje u otros procesos de producción.

Figura 1 : PLC Modicon 340

2.1.2 Componentes

2.1.2.1 Procesador (CPU)

El procesador, la unidad central de procesamiento o la CPU es el "cerebro" del PLC. El tamaño y tipo de CPU determinará cosas como: las funciones de programación disponibles, el tamaño de la lógica de la aplicación disponible, la cantidad de memoria disponible y la velocidad de procesamiento.

Este ejecuta una secuencia de operaciones llamado ciclo de SCAN:

- Lectura de las entradas. Lee la información en los módulos del controlador y se coloca en la memoria de entrada.
- Barrido de rutinas. Ejecutan todas las rutinas en el controlador y actualización de la memoria de datos.
- Actualización de las salidas. Toma la información de la memoria de salida y se escribe en los módulos de salida.
- Comunicaciones. Atiende los requerimientos de comunicaciones.
- Diagnóstico. Verifica si el procesador está funcionando correctamente.

Figura 2 : Ciclo SCAN

2.1.2.2 Módulos de entradas y salidas

Las entradas llevan señales del proceso al controlador, pueden ser interruptores de entrada, sensores de presión, entradas de operador, etc. Estos son como los sensores del PLC.

Las salidas son los dispositivos que el PLC utiliza para enviar los cambios al exterior. Estos son los actuadores que el PLC puede cambiar para ajustar o controlar el proceso (motores, luces, relés, bombas, etc.)

Muchos tipos de entradas y salidas se pueden conectar a un PLC, y todas ellas se pueden dividir en dos grandes grupos: analógicas y digitales. Las entradas y salidas digitales son las que funcionan debido a un cambio de valor discreto o binario (ON/OFF, TRUE/FALSE). Las entradas y salidas analógicas cambian continuamente en un rango variable: presión, temperatura y potencia.

2.1.2.3 Fuente de alimentación

La fuente de alimentación, en un autómata programable, tiene como misión la de suministrar el voltaje que requiere tanto la unidad central de proceso como todos los módulos electrónicos que posea el PLC.

Esta fuente de alimentación suele ser regulada de voltaje de corriente directa, incorporando protecciones contra interferencias electromagnéticas y contra posibles oscilaciones en el voltaje de corriente alterna a cuál se conecta, evitando así, la propagación de estas anomalías a los circuitos internos del autómata.

Algunas de estas fuentes cuentan con baterías de respaldo para que, en caso de que falle el suministro de energía principal, permitir la continuidad de la operación del autómata, a la vez que puede activarse una alarma para dar aviso en el momento justo que el suministro de energía principal ha dejado de operar.

De forma complementaria, la fuente de alimentación puede alimentar todos aquellos captadores de tipo activo que el proceso requiera para su control, siempre que el consumo global de estos no sature la corriente máxima a entregar por la fuente. Ara ello, el autómata suele disponer de una toma de 24 Vcc en forma de bornes de conexión.

2.1.2.4 Rack (Bastidor)

La mayoría de los autómatas medianos y grandes se montan de tal manera que los componentes individuales (CPU, entrada/salida, fuente de alimentación) son módulos que se mantienen unidos dentro de un bastidor o carcasa.

En los PLC más pequeños, todos estos componentes pueden estar contenidos en una sola carcasa.

2.1.2.5 Módulo de comunicación

Cuando el sistema requiera de un tipo de comunicación diferente, se debe de instalar un módulo de comunicación para un protocolo concreto como Ethernet, EtherCAT, CANBus, Modbus, Profibus…

2.2 Protocolos de comunicación

Las redes de comunicaciones en la industria favorecen la descripción de un tipo de sistema de transmisión que ayuda en el intercambio de información entre los distintos elementos que conforman dichas redes. Se crea una conexión entre capas de la factoría, desde las oficinas hasta las máquinas que desarrollan el producto.

Para fines prácticos estas comunicaciones se representan como una pirámide llamada CIM (Computer Integrated Manufacturing), pero en realidad la podemos relacionar mucho con el modelo OSI; ya que CIM se puede separar por segmentos, los cuales realizan distintas operaciones y necesitan requerimientos distintos para operar.

Figura 3 : Pirámide CIM

En este proyecto de estudio se trabaja en los siguientes niveles:

- **Nivel de control**: Se realiza el control individual de cada recurso. Los dispositivos conectados son autómatas de gama baja y media, sistemas de control numérico, transporte automatizado. Se utilizan las medidas proporcionadas por el nivel 0 y se dan las consignas a los actuadores y máquinas de dicho nivel. Se usan buses de campo del tipo: AS-i, Profibus DP, CANBus, Device NET, Modbus, etc.

- **Nivel de proceso**: Se realiza el control directo de las máquinas y sistemas de producción. Los dispositivos conectados son sensores, actuadores, instrumentos de medida, máquinas de control numérico, etc.

La celda de automatización está dividida en dos partes, cada una gobernada por un PLC donde a nivel de campo se ha establecido un protocolo de comunicación diferente; CANopen y Modbus.

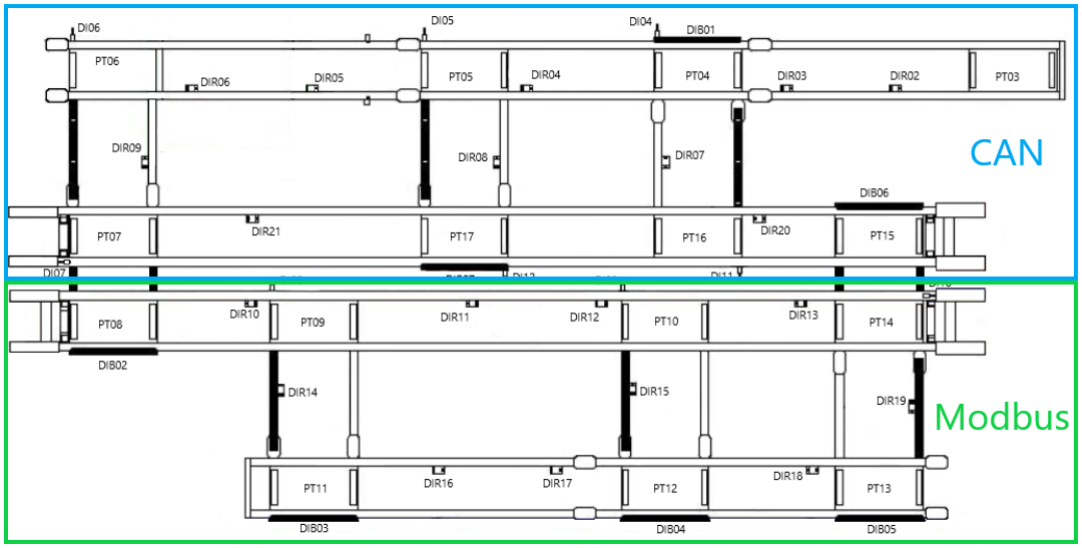


Figura 4 : Protocolos de comunicación en la celda

2.2.1 CANopen

CAN (Controller Area Network) es un protocolo de comunicaciones desarrollado por la firma alemana Robert Bosch GmbH, basado en una topología bus para la transmisión de mensajes en entornos distribuidos. Además, ofrece una solución a la gestión de la comunicación entre múltiples CPUs (unidades centrales de proceso).

Sobre este protocolo estándar se diseñó CANopen que especifica el nivel de aplicación del modelo OSI, resultando una arquitectura simplificada con sólo tres capas, muy común en los buses de campo. En la práctica no significa que las capas intermedias estén totalmente vacías, sino que la mayor parte de los servicios que ofrecen no tienen utilidad para un bus industrial. La solución es aligerar la implementación suprimiendo esas capas y trasladando los servicios necesarios a la capa superior, que es la de aplicación.

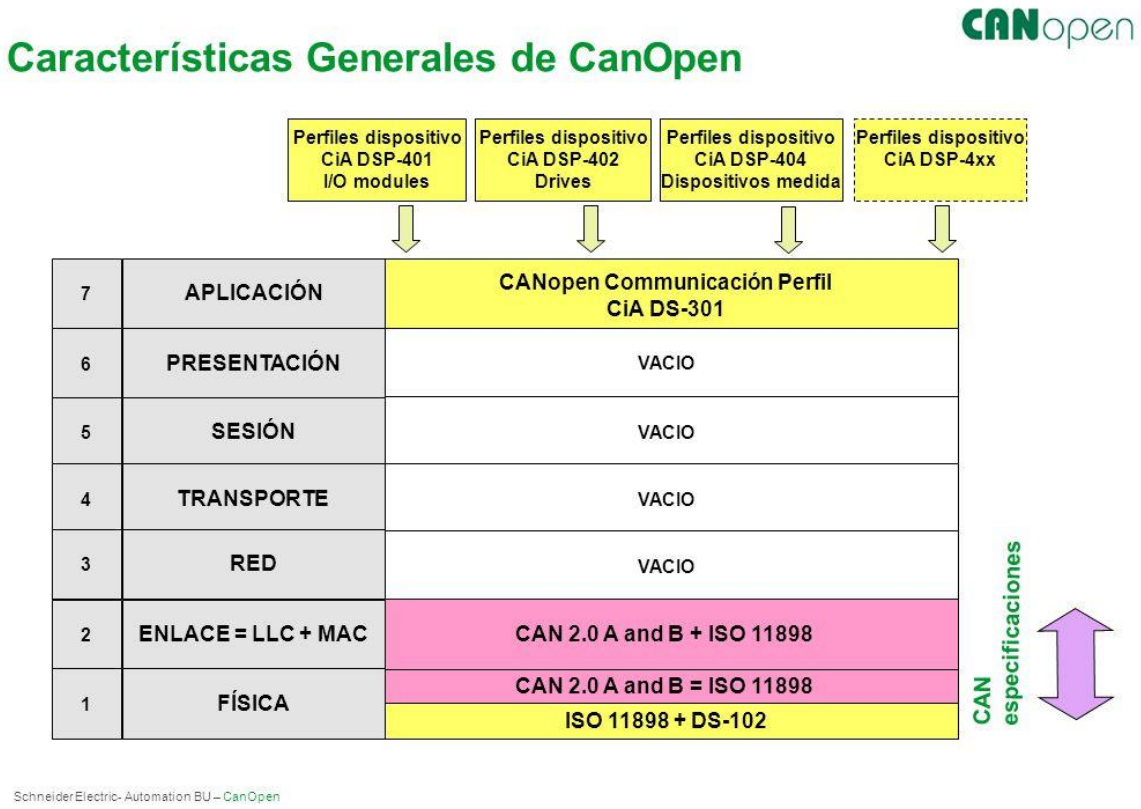


Figura 5 : Capas de comunicación en CANopen

Características principales:

- Distancia: 100 a 500 m.
- Puede tener hasta 64 nodos
- Velocidades de transmisión: 125, 250, 500 y 1000 Kbit/s.
- Puede enviar mensajes de 8 bytes como máximo por nodo y por mensaje.

Ventajas:

- Mejor caracterizado para control de movimiento de alta velocidad, así como para lazos de realimentación cerrados que otros buses CAN.
- Alta fiabilidad, uso eficiente del ancho de banda de la red y alimentación disponible en la misma.

Desventajas:

- Aceptación limitada fuera de Europa.
- Limitación de ancho de banda, tamaño de los mensajes y longitud máxima de la red.

2.2.2 Modbus

Modbus es un protocolo de comunicación abierto, utilizado para transmitir información a través de redes en serie entre dispositivos electrónicos. Fue creado por Modicon, ahora Schneider Electric, a finales de los 70 para la comunicación entre controladores lógicos programables (PLC). En la actualidad, Modbus sigue siendo el protocolo más usado para conectar dispositivos industriales.

El protocolo Modbus intercambia información utilizando un mecanismo de solicitud-respuesta entre un maestro (cliente) y un esclavo (servidor). El principio maestro-esclavo es un modelo de protocolo de comunicaciones en el cual un dispositivo (el maestro) controla uno o más dispositivos (los esclavos). En una red Modbus estándar, hay 1 maestro y hasta 31 esclavos.

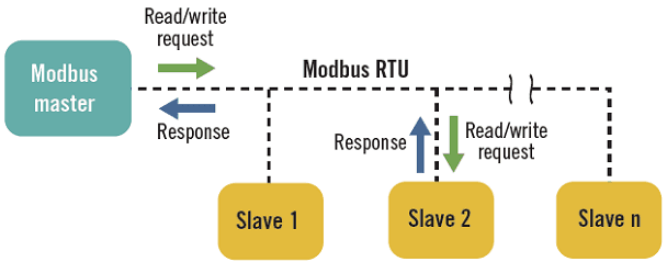


Figura 6 : Comunicación maestro - esclavo

Existen varios tipos de versiones en el protocolo Modbus para el puerto serie y Ethernet, que se utilizan para atender las necesidades específicas de los sistemas de automatización industrial en las empresas. Por ejemplo, Modbus TCP se utiliza para Ethernet, y Modbus RTU y Modbus ASCII para los puertos serie.

Las más comunes son:

- Modbus RTU
- Modbus TCP
- Modbus ASCII
- Modbus Plus

Los diferentes tipos de protocolos abarcan desde el nivel de proceso hasta el de supervisión.

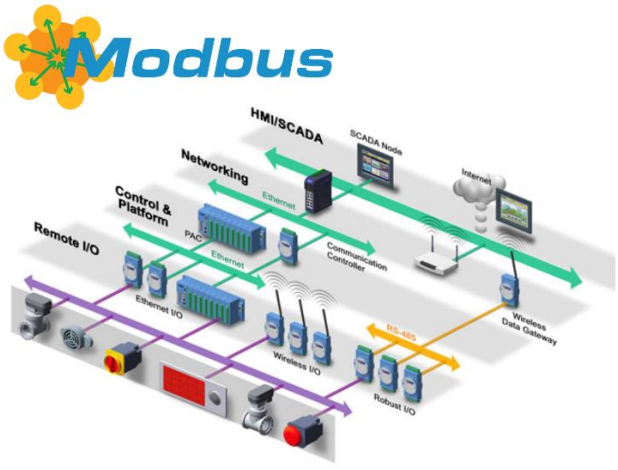


Figura 7 : Campo de aplicación del protocolo Modbus

El diseño de los módulos tecnológicos con Modbus está orientado hacia el modelo ISO/OSI utilizando la capa 2, 3 y 7.

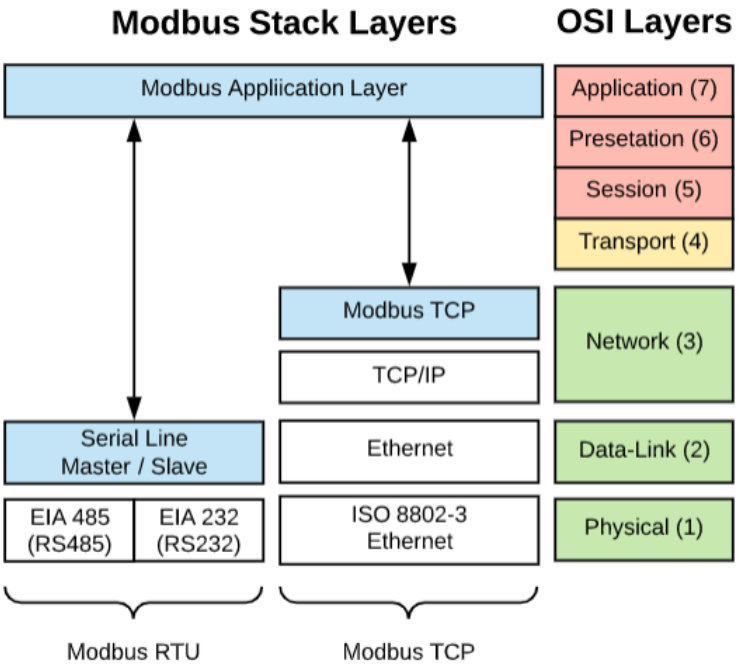


Figura 8 : Capas de comunicación en Modbus

Características principales:

- Control de acceso al medio tipo Maestro/Esclavo.
- El protocolo especifica: formato de trama, secuencias y control de errores.
- Existen variantes en el formato: ASCII, RTU, TCP…
- A cada esclavo se le asigna una dirección fija y única en el rango de 1 a 247.
- La dirección 0 está reservada para mensajes de difusión sin respuesta

Ventajas:

- Alto grado de flexibilidad operativa. Su modo RTU en serie acelera los intercambios de datos con su codificación de 8 bits mientras permite la conexión (con repetidores) de 247 periféricos a una distancia de 1200 metros.
- En su versión TCP/IP, ofrece una velocidad de datos de 10 a 100 Mbits/segundo (y una conectividad casi ilimitada) en topologías de red «clásicas» como el anillo o la estrella.
- El despliegue técnico de su ecosistema se caracteriza por su simplicidad.

Desventajas:

- En el formato RTU tiene un uso limitado como bus de dispositivos, capacidades de diagnóstico limitadas para las aplicaciones y se requiere alimentación independiente.
- En el formato TCP/IP no existe forma estándar para que un nodo encuentre la descripción de un objeto de datos, se limita a direccionar 247 dispositivos que pueden conectarse a una estación maestra y las transmisiones deben ser consecutivas.

2.2.3 Ethernet / IP

EtherNet/IP es un protocolo de red industrial que emplea CIP (Common Industrial Protocol) en Ethernet estándar. Trabaja en una capa de aplicación de red, que es (en los dos modelos conceptuales de redes) la capa 'más alta' de los dispositivos y le facilita al usuario la comunicación entre controles y dispositivos de E/S. Más específicamente, EtherNet/IP es la capa superior de los modelos OSI (Open Systems Interconnection) y TCP/IP (Transmission Control Protocol/Internet Protocol).

El protocolo EtherNet/IP contiene:

- La capa de aplicación ya mencionada;
- Una capa de conexión en red Internet Protocol;
- La capa de enlace Ethernet estándar.

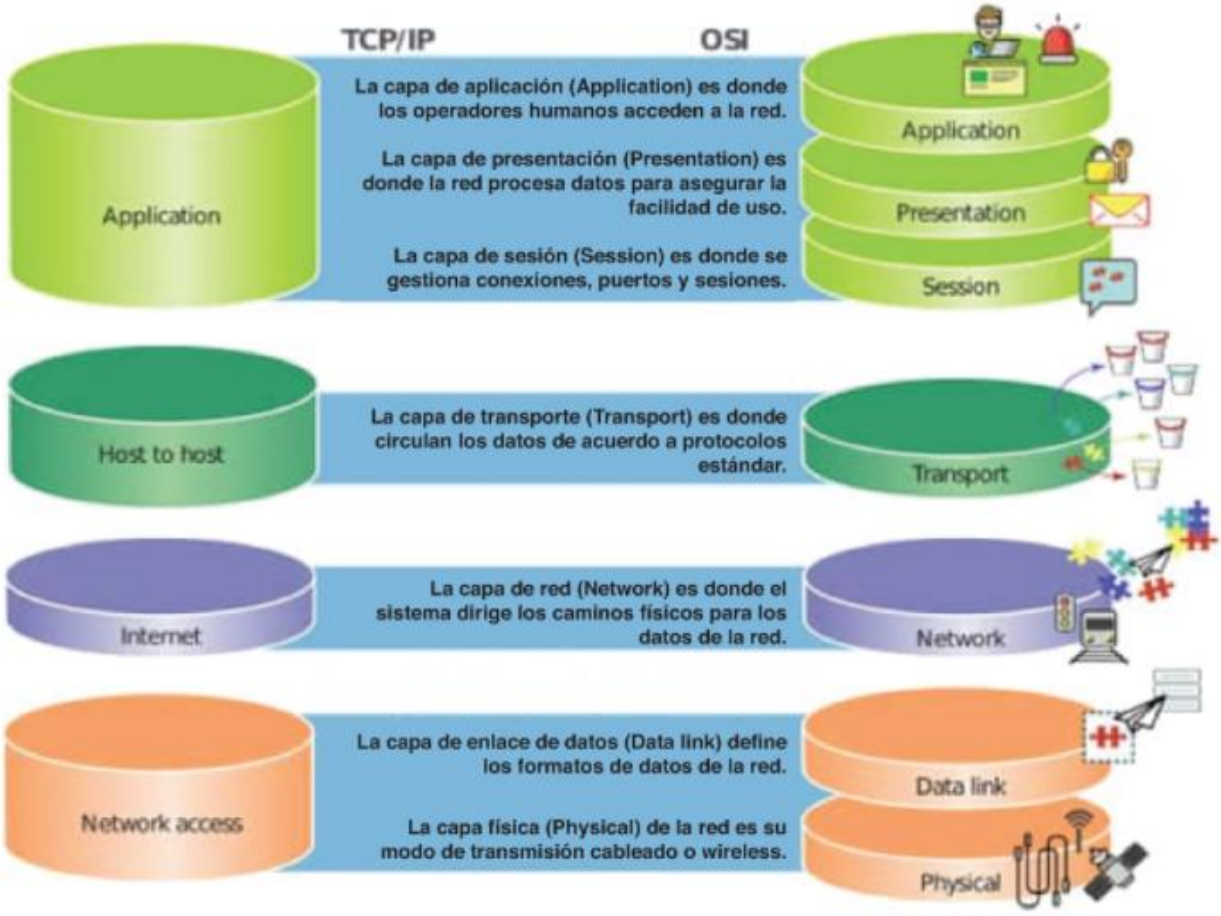


Figura 9 : Capas de comunicación en Ethernet

Ethernet y TCP/IP (Transmisión Control Protocol/Internet Protocol) son dos conceptos distintos pero que se suelen utilizar conjuntamente, Internet es una aplicación de TCP/IP, pero es solo un caso y existen muchas otras muy diferentes.

- TCP/IP se desarrolló en la Universidad de Stanford en 1970 y consiste en un conjunto de protocolos que pueden funcionar sobre diversos medios físicos, cubrirían entre el nivel 3 y 7 del modelo OSI.
- Ethernet es un estándar de comunicaciones que incluye los niveles OSI 1 y 2. La red Ethernet tiene como aplicación básica la gestión e información global de un sistema automatizado.

Características principales:

- Distancia: de 100 (para 10Base-T) a 50 Km (usando fibra óptica).
- Número máximo de nodos: 1024, extensible con routers.
- Velocidad de transmisión: 10 Mbit/s. a 100 Mbit/s.
- Tamaño del mensaje: 46 a 1500 bytes.

Ventajas:

- Es el estándar de red más reconocido internacionalmente.
- Puede tratar con grandes cantidades de información a una velocidad muy rápida sirviendo para instalaciones muy grandes.

Desventajas:

- Para mensajes con poca información no es eficiente.
- No lleva alimentación incorporada.
- Los conectores (RJ45) son vulnerables físicamente.
- No tiene la propiedad de determinismo por el que los buses de campo pueden asegurar la respuesta de la red para cada carga.

2.3 Lenguajes de programación

Los lenguajes de programación de PLC son símbolos, caracteres y reglas de uso que fueron diseñados para poder tener una comunicación de los usuarios con las máquinas. Gracias a este vínculo, es posible crear un programa con instrucciones para controlar el funcionamiento de cualquier proceso o máquina.

En este proyecto se utilizan los dos lenguajes más usados:

- Ladder (LD)
- Structured Text (ST)

2.3.1 Ladder (LD)

Es un lenguaje gráfico, derivado de los esquemas de lógica de relés. Mediante símbolos representa contactos, bobinas, etc. Su principal ventaja es que los símbolos básicos están normalizados según el estándar IEC y son empleados por todos los fabricantes.

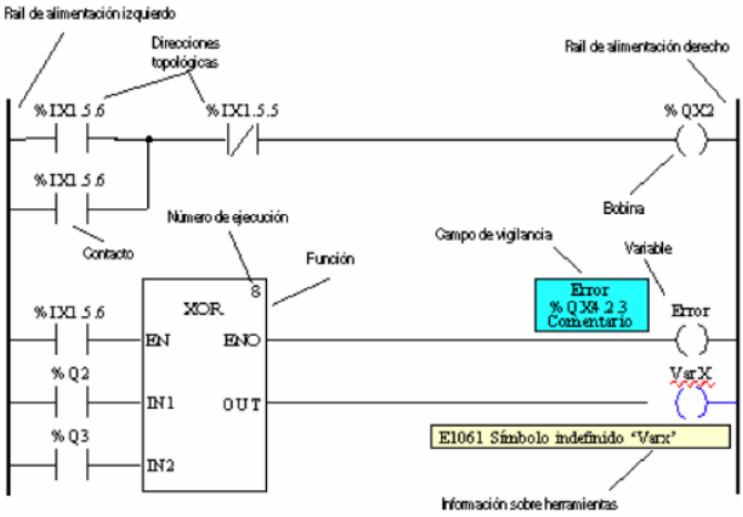


Figura 10 : Ejemplo de código en lenguaje LD

2.3.2 Structured Text (ST)

Este es un lenguaje más parecido a C o Pascal, ya permite utilizar instrucciones más conocidas como IF-ELSE para establecer condicionales o repeat-until, while-do para ejecución de bucles secuenciales. Se suele utilizar también en aplicaciones medianas-pequeñas debido a su grado de complejidad.

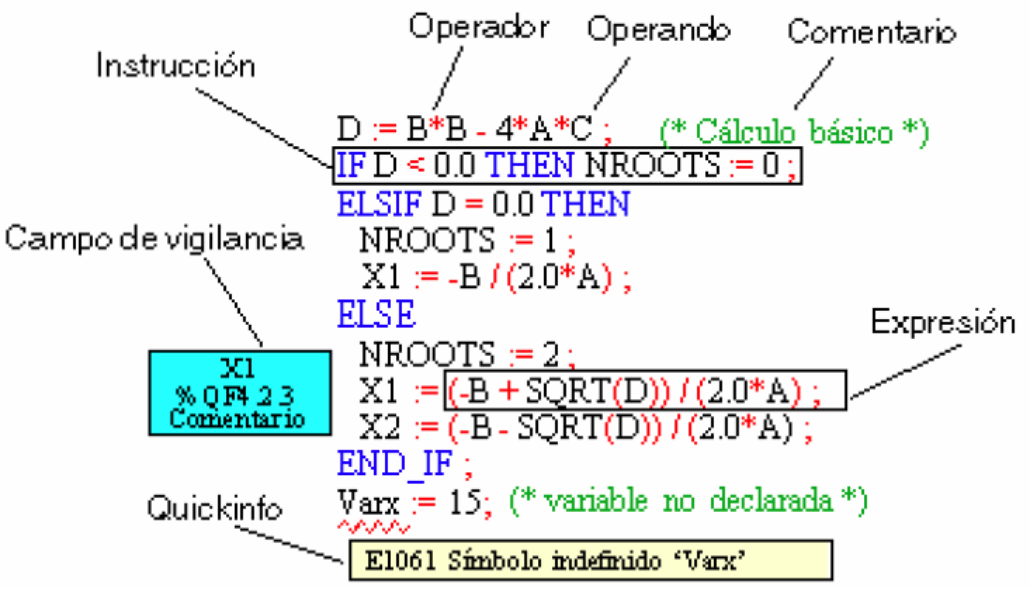


Figura 11 : Ejemplo de código en lenguaje ST

3. Caso de estudio

3.1 Componentes clave de la celda automatizada

La celda del laboratorio está formada por un conjunto de cintas transportadoras las cuales están preparadas para el transporte de bandejas metálicas rectangulares. En el recorrido formado por toda la estructura existen unas plataformas y unos actuadores llamados retenedores para llevar a cabo la gestión de las bandejas.

Figura 12 : Celda automatizada

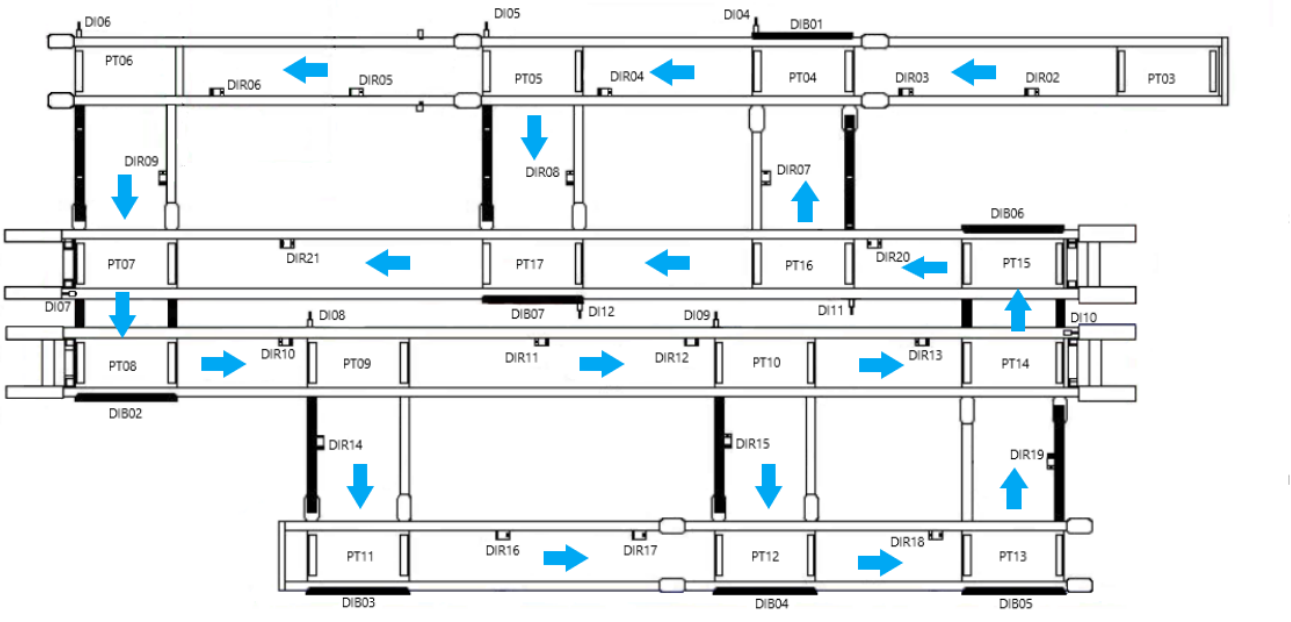


Figura 13 : Direcciones de las cintas transportadoras

3.1.1 Retenedores

Los retenedores tienen la función de detener la bandeja en un punto intermedio entre plataformas con el objetivo de no hacer que colisionen las bandejas y aumentar la capacidad de volumen en la celda.

Están formados por un actuador neumático y un sensor inductivo. Para el avance, el actuador del retenedor debe bajar hasta que la bandeja pase en su totalidad para luego volver a su estado de reposo.

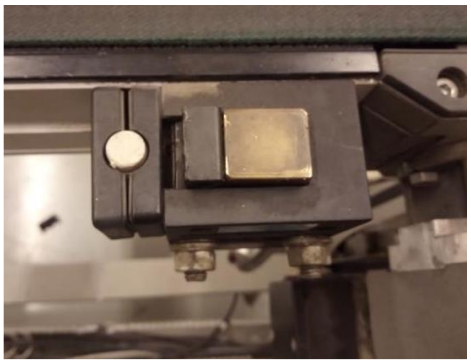


Figura 14: Retenedor

3.1.2 Plataformas

Estas tienen la capacidad de poder cambiar la dirección de la bandeja según el criterio establecido.

Detectan la presencia de la bandeja de dos maneras:

- Sensor inductivo basculante: La bandeja empuja un mecanismo para llevar a cabo la retención.
- Sensor inductivo con retenedor: Se detecta el material metálico de la bandeja para detectar su presencia. La retención se lleva a cabo mediante dos piezas metálicas que impiden su avance.

Las plataformas están compuestas por dos pequeñas cintas que transportan la bandeja en sentido perpendicular de las cintas principales y diisponen de un motor el cual eleva o desciende la plataforma tanto para encarar o recibir la bandeja a la altura de la cinta correspondiente.

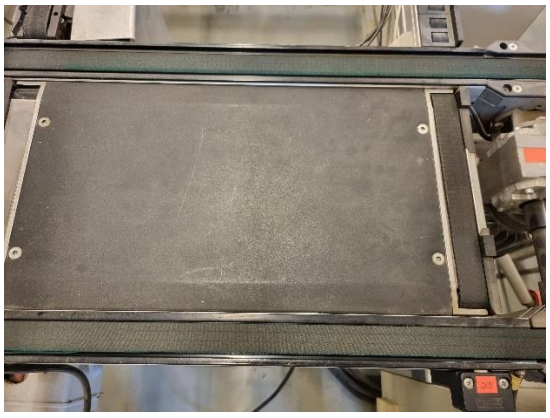


Figura 15: Plataforma con sensor inductivo con retenedor

Figura 16 : Sensor inductivo basculante en plataforma

3.2 Concepto y objetivo

Actualmente la celda no tiene una finalidad establecida aparte de crear un sistema de transporte de bandejas. Por lo tanto, se plantea el objetivo de realizar la programación como si de una línea de producción se tratara.

En primer lugar, las bandejas simulan tener un producto el cual debe pasar por varios procesos industriales.

Existen tres tipos de productos de distinta naturaleza cada uno necesitando diferentes procesos teniendo algunos en común entre ellos.

	Estación de trabajo				
Producto	P1	P2	P3	P4	P5
Prod. Tipo 1	**x**	**x**			**x**
Prod. Tipo 2			**x**	**x**	**x**
Prod. Tipo 3		**x**		**x**	**x**

Tabla 1: Procesos aplicables a cada tipo de producto

Los procesos se llevan a cabo en estaciones de trabajo, situados en varias plataformas pertenecientes a la línea Modbus (menos la P5).

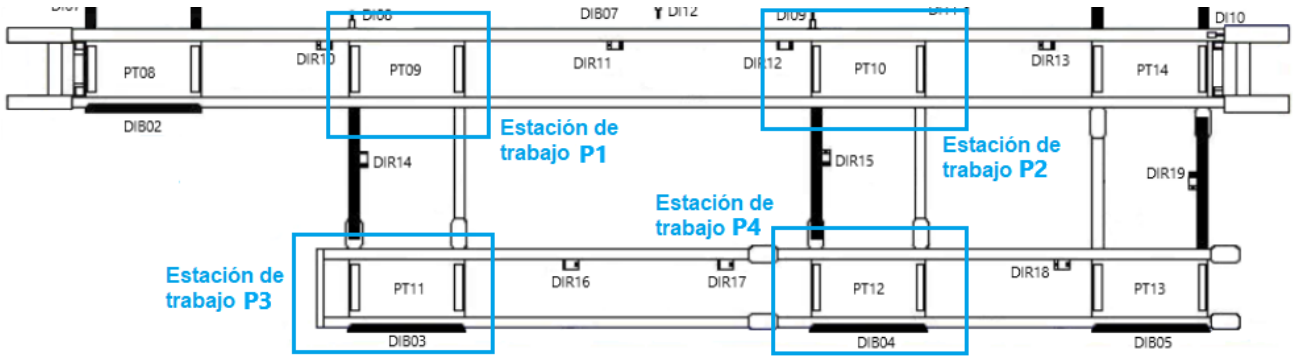


Figura 17 : Estaciones de trabajo

Después de que un producto pase por su correspondiente aplicación de proceso, su estado dependerá del resultado del mismo. Si el resultado ha sido satisfactorio, su identificación de estado cambiará, si no permanecerá con el mismo valor de este.

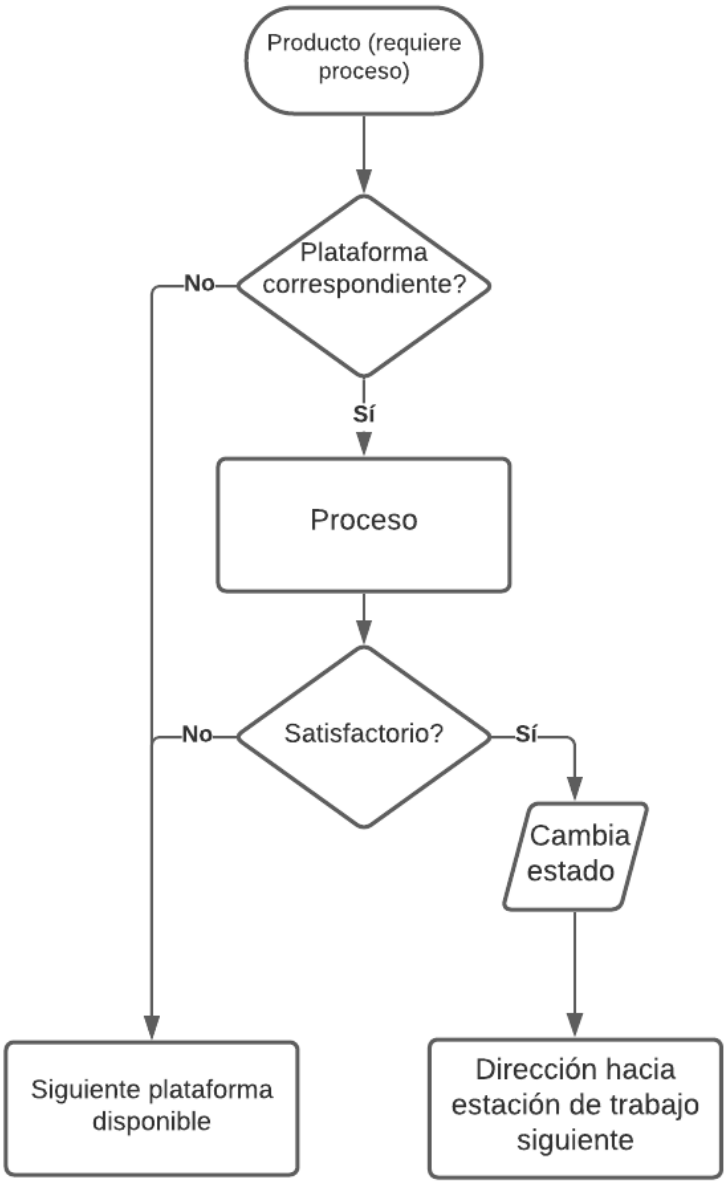


Figura 18 : Diagrama de estados del proceso de un producto

Una vez que el producto haya pasado por todos los procesos correspondientes, se realizará un control de calidad situado en la línea CAN para determinar si ha sido correctamente procesado. Si el resultado es positivo, se procederá a recoger el producto en una de las plataformas, por el contrario, se enviará a otra para realizar su rechazo.

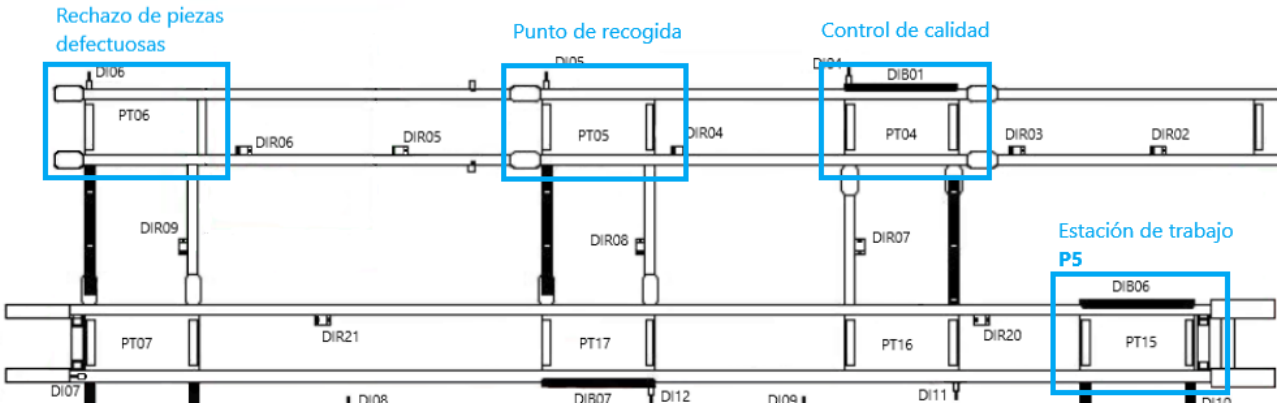


Figura 19 : Estaciones de trabajo y puntos de control

4. Análisis

Conocido el entorno de trabajo en el cual se van a llevar todos los procesos y el comportamiento de cada elemento de la celda, se debe analizar los programas base facilitados para establecer el punto de partida.

4.1 Programa base línea CAN

4.1.1 Configuración de hardware

El software utilizado es EcoStruxure Control Expert de Schneider Electric (actualización del antiguo Unity Pro de Telemechanique) para programar un PLC Modicon M340.

La configuración de hardware del proyecto facilitado consta de dos partes:

- Estructura del PLC: Fuente de alimentación, CPU, módulos de E/S
- Módulos de E/S de la periferia conectados mediante CANOpen.

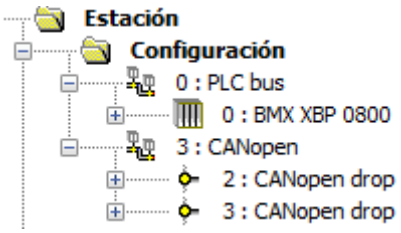


Figura 20 : Configuración de hardware del proyecto de la linea CAN

El PLC está configurado de la siguiente manera:

- Fuente de alimentación.
- Procesador con puerto USB y puertos de comunicación Ethernet y CANOpen.
- Un módulo de 16 entradas digitales y 16 salidas digitales.
- Dos módulos de 8 entradas digitales y 8 salidas digitales.

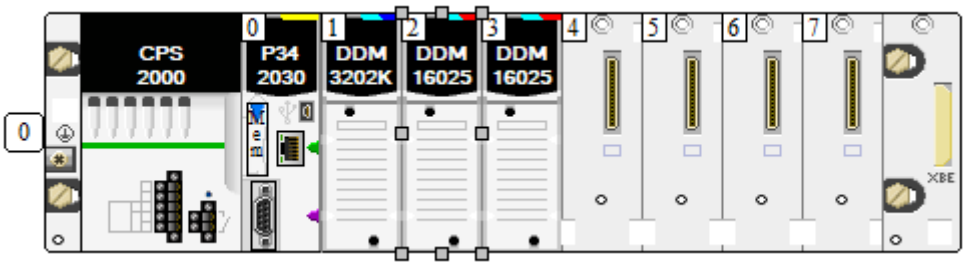


Figura 21 : Configuración PLC línea CAN

La periferia se realiza mediante dos dispositivos llamados islas Advantys que consisten cada uno de:

- Un módulo de 16 entradas digitales.
- Un módulo de 8 entradas digitales.
- Un módulo de 16 salidas digitales.
- Un módulo de 16 salidas digitales.
- Un módulo de 2 canales de entradas analógicas.
- Un módulo de 2 canales de salidas analógicas.

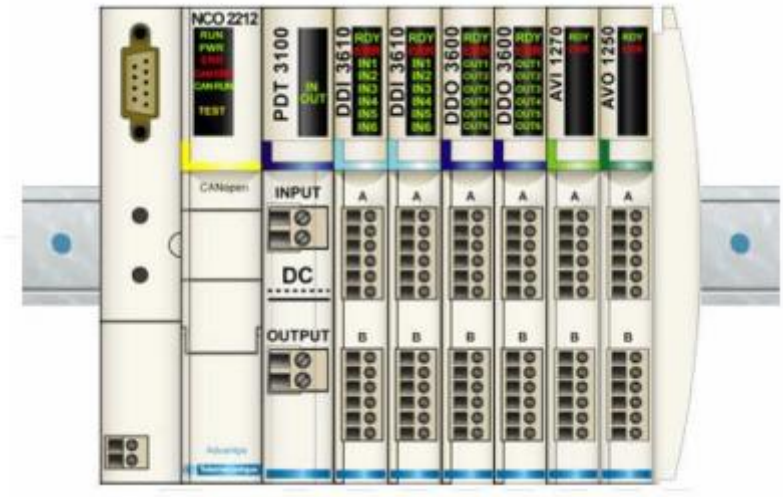


Figura 22 : Módulos de E/S de la periferia con conexión mediante CANBus

4.1.2 Variables del programa

En el programa facilitado vienen declaradas las variables con las direcciones físicas definidas para poder realizar la programación directamente usando estas:

Nombre	Tipo	Comentario	Dirección
PARO_EMERGENCIA	EBOOL		%I0.1.4
REARME	EBOOL		%I0.1.6
EMERGENCY_LOCK_PULMON	BOOL		%MW39.6
DIR04	BOOL	%IW\3.2\0.0.0.167.0	%MW167.0
DI05	BOOL	%IW\3.2\0.0.0.167.1	%MW167.1
DIR05	BOOL	%IW\3.2\0.0.0.167.2	%MW167.2
DIR08	BOOL	%IW\3.2\0.0.0.167.3	%MW167.3
DI06	BOOL	%IW\3.2\0.0.0.167.4	%MW167.4
DIR09	BOOL	%IW\3.2\0.0.0.167.5	%MW167.5
DI07	BOOL	%IW\3.2\0.0.0.169.0	%MW169.0
DIR21	BOOL	%IW\3.2\0.0.0.169.1	%MW169.1
DIB07	BOOL	%IW\3.2\0.0.0.169.2	%MW169.2
DIB06	BOOL	%IW\3.3\0.0.0.167.0	%MW445.0
DI11	BOOL	%IW\3.3\0.0.0.167.2	%MW445.2
DIR07	BOOL	%IW\3.3\0.0.0.167.3	%MW445.3
DIB01	BOOL	%IW\3.3\0.0.0.167.5	%MW445.5
DIR20	BOOL	%IW\3.3\0.0.0.169.2	%MW447.2
DIR03	BOOL	%IW\3.3\0.0.0.169.4	%MW447.4
EMERGENCY_LOCK	BOOL	%MW1234	%MW470
ComunicacionPulmonEscritura	ARRAY[1..30] OF INT		%MW700
ComunicacionPulmonLectura	ARRAY[1..30] OF INT		%MW730
M2	EBOOL		%Q0.2.17
M3	EBOOL		%Q0.2.18
M5	EBOOL		%Q0.2.19
M7	EBOOL		%Q0.2.20
M8	EBOOL		%Q0.2.21
M9	EBOOL		%Q0.2.22
M11	EBOOL		%Q0.2.23
M12	EBOOL		%Q0.3.16
LUZ_ROJA_BALIZA	EBOOL		%Q0.3.20
DIR05_down	BOOL		%QW\3.2\0.0.0.133.0
DIR08_down	BOOL		%QW\3.2\0.0.0.133.1
DIR09_down	BOOL		%QW\3.2\0.0.0.133.2
DIR21_down	BOOL		%QW\3.2\0.0.0.133.3
DIR04_down	BOOL		%QW\3.2\0.0.0.133.4
SUBE_PT05	BOOL		%QW\3.2\0.0.0.133.5
BAJA_PT05	BOOL		%QW\3.2\0.0.0.134.0
SUBE_PT06	BOOL		%QW\3.2\0.0.0.134.1
SUBE_PT07	BOOL		%QW\3.2\0.0.0.134.2
SUBE_PT17	BOOL		%QW\3.2\0.0.0.134.3
CINTAIZQUIERDA	BOOL		%QW\3.2\0.0.0.134.4
CINTACENTRAL	BOOL		%QW\3.2\0.0.0.134.5
DIR03_down	BOOL		%QW\3.3\0.0.0.133.0
DIR07_down	BOOL		%QW\3.3\0.0.0.133.1
SUBE_PT04	BOOL		%QW\3.3\0.0.0.133.2
SUBE_PT15	BOOL		%QW\3.3\0.0.0.133.3
SUBE_PT16	BOOL		%QW\3.3\0.0.0.133.4
BAJA_PT16	BOOL		%QW\3.3\0.0.0.133.5
DIR20_down	BOOL		%QW\3.3\0.0.0.134.0

Figura 23 : Variables del programa base de la linea CAN

4.1.3 Secciones del programa

Se analizan los distintos bloques de programa del programa base:

- Entradas: Al existir varios sensores de diferente tipo (normalmente abiertos y normalmente cerrados) se niegan los tipo NO para tratarlos como NC.

```
Sensor_DIR03    := NOT DIR03;
Sensor_PT04     := DIB01;
Sensor_DIR04    := DIR04;
Sensor_PT05     := DI05;
Sensor_DIR08    := DIR08;
Sensor_PT17     := NOT DIB07;
Sensor_DIR21    := DIR21;
Sensor_PT07     := DI07;
Sensor_DIR05    := DIR05;

Sensor_PT06     := DI06;
Sensor_DIR09    := DIR09;
Sensor_PT15     := DIB06;
Sensor_DIR20    := NOT DIR20;
Sensor_PT16     := NOT DI11;
Sensor_DIR07    := NOT DIR07;
```

Figura 24 : Sección ENTRADAS

- Emergencia rearme: Bloque donde está programado el comportamiento si se realiza un paro de emergencia habilitando un rearme/reset cada vez que se inicia el programa o se pulsa el botón de rearme.

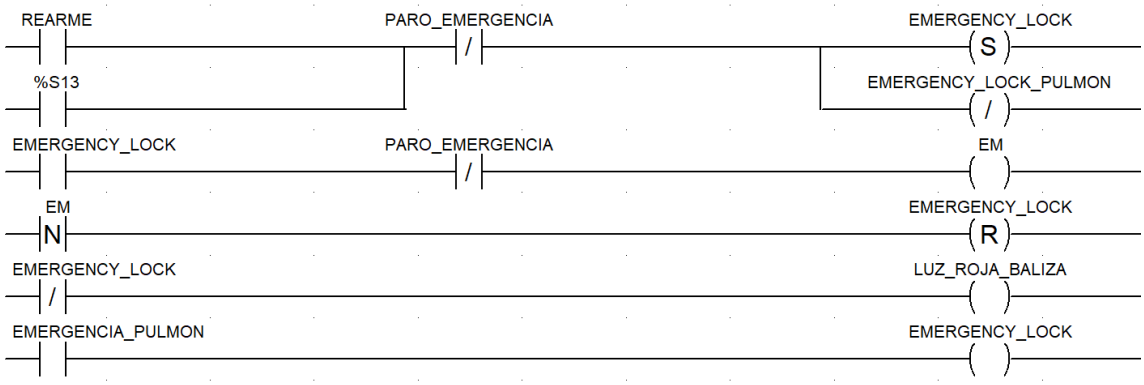


Figura 25 : Sección EMERGENCY

- Salidas: Esta sección consta de las variables de salidas físicas hacia los distintos motores y actuadores de la línea CAN de la celda donde se debe completar con las condiciones necesarias para sus activaciones.

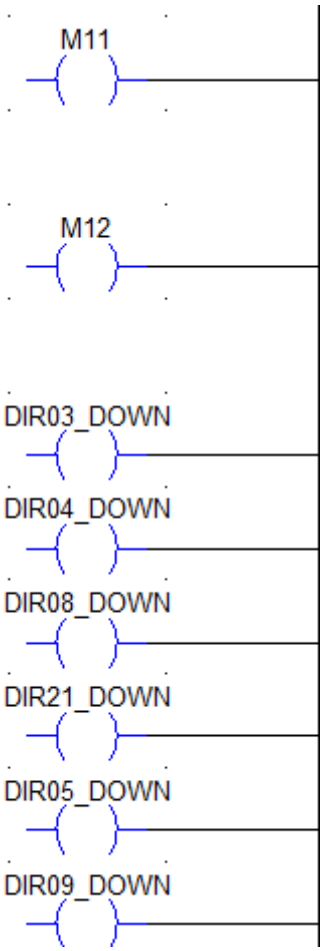


Figura 26 : Varias salidas de la sección SALIDAS sin completar

4.1.4 Bloque de funciones (FB) y estructuras

El programa contiene un bloque de funciones el cual va a ser el objeto principal de funcionamiento en la celda. En este incluye el uso de estructuras creadas "Bandeja" y "DatosEstado".

Figura 27 : Estructuras de datos y FB existentes y sus secciones

Las estructuras creadas están formadas de la siguiente manera:

Nombre	Tipo
Bandeja	<Estruct.>
IDPRODUCTO	INT
TIPOPRODUCTO	INT
ESTADO	INT
DatosEstado	<Estruct.>
BandejaEntrada	Bandeja
BandejaSalida1	Bandeja
BandejaSalida2	Bandeja
Estado	INT

Figura 28 : Estructura de Bandeja y Datos Estado

El bloque de funciones "Plataforma" contiene la siguiente estructura de variables:

Nombre	Nº	...	Comentario
Plataforma		<DFB>	
<entradas>			
ReposoDestino1	1	BOOL	Estado de Reposo del Destino en la dirección 1
ReposoDestino2	2	BOOL	Estado de Reposo del Destino en la dirección 2
AvanceAux1	3	BOOL	Evitar el avance simultaneo hacia una plataforma con dos entradas, conectar al estado de avance del otro acceso a la plataforma
AvanceAux2	4	BOOL	Evitar el avance simultaneo hacia una plataforma con dos entradas, conectar al estado de avance del otro acceso a la plataforma
PresenciaBandeja	5	BOOL	Sensor de la sección correspondiente que indicará si hay una bandeja
BandejaVacia	6	Bandeja	Variable para asignar el estado de la Bandeja de Entrada a 0
Tiempo_ACK	10	TIME	Tiempo a esperar el ack cuando el sensor detecta (una vez sobrepasado se activará el avance aun y no haber obtenido un ack)
Emergencia	11	BOOL	Emergencia del sistema
Rearme	12	BOOL	Para poner a zero todos los valores internos y volver al estado de reposo
Especial1	13	BOOL	Bit para las plataformas que envian una bandeja lateralmente en el destino 1
Especial2	14	BOOL	Bit para las plataformas que envian una bandeja lateralmente en el destino 2
<salidas>			
Retenedor1	1	BOOL	Retenedor o Plataforma a activar durante el avance al destino 1
Retenedor2	2	BOOL	Retenedor o Plataforma a activar durante el avance al destino 2
Motor1_1	3	BOOL	Activación del motor en Avance 1
Motor1_2	4	BOOL	Activación del motor en Avance 1
Motor2_1	5	BOOL	Activación del motor en Avance 2
Motor2_2	6	BOOL	Activación del motor en Avance 2
<entradas/salidas>			
DatosEstado	7	Datos...	Variable sobre la cual se van a reescribir ciertos datos internos para su accesibilidad des del exterior del bloque
BandejaDestino1	8	Bandeja	Bandeja a enviar al destino 1
BandejaDestino2	9	Bandeja	Bandeja a enviar al destino 2
ConfirmacionEnvioInfo1	15	BOOL	Bit para confirmar al Destino 1 que ya se le ha hecho la transmisión de datos acerca de la bandeja
ConfirmacionEnvioInfo2	16	BOOL	Bit para confirmar al Destino 2 que ya se le ha hecho la transmisión de datos acerca de la bandeja
<público>			
BandejaEntrada		Bandeja	Variable sobre la cual el retenedor sobreescrivirá los valores de la bandeja que le envie
BandejaSalida1		Bandeja	Datos de la Bandeja que iran a la Salida 1
BandejaSalida2		Bandeja	Datos de la Bandeja que iran a la Salida 2
Reposo		BOOL	Estado base de la Plataforma
Peticion		BOOL	Estado cuando se tiene una bandeja y se espera poder enviarla a la siguiente plataforma
Avance1		BOOL	Movimiento en la dirección 1
Avance2		BOOL	Movimiento en la dirección 2
Bloqueo1		BOOL	Variable para bloqueo del Avance 1
Bloqueo2		BOOL	Variable para bloqueo del Avance 2
AcuseEnvioInfo		BOOL	Bit para la confirmación de que se ha recibido la información de la bandeja
<privado>			
FReposo		BOOL	
FAvance1		BOOL	
FAvance2		BOOL	
FILTRO_SENSOR		TON	
FBI_0		TOF	
FBI_1		TOF	
Bloqueo_ACK_temp		BOOL	
FILTRO_SENSOR_0		TON	
<secciones>			
Control		<LD>	
Seguimiento		<ST>	
Almacenaje		<ST>	
Setandreset		<ST>	

Figura 29 :Estructura del FB "Plataforma"

El bloque de funciones "Plataforma" contiene diferentes secciones de las cuales se debe analizar su comportamiento:

La sección de control de la plataforma consiste de tres estados:

- Reposo
- Petición
- Avance

No es posible la simultaneidad de varios estados.

Debido a la posibilidad de plataformas dónde pueden realizar desvíos en dos direcciones distintas, la sección está duplicada a lo que se refiere al estado de Avance.

Cuando la plataforma o retenedor no contiene ninguna bandeja (o esta la abandona) teniendo esta información mediante "PresenciaBandeja", se activa el estado de Reposo.

Dependiendo de si la plataforma debe trasladar la bandeja por una dirección la cual necesita tener las salidas activadas, la variable "EspecialX" no desactiva el reposo hasta que la bandeja no haya llegado al destino. Pues al existir una altura diferente entre cintas transversales, se requiere de esta funcionalidad para su correcto traslado.

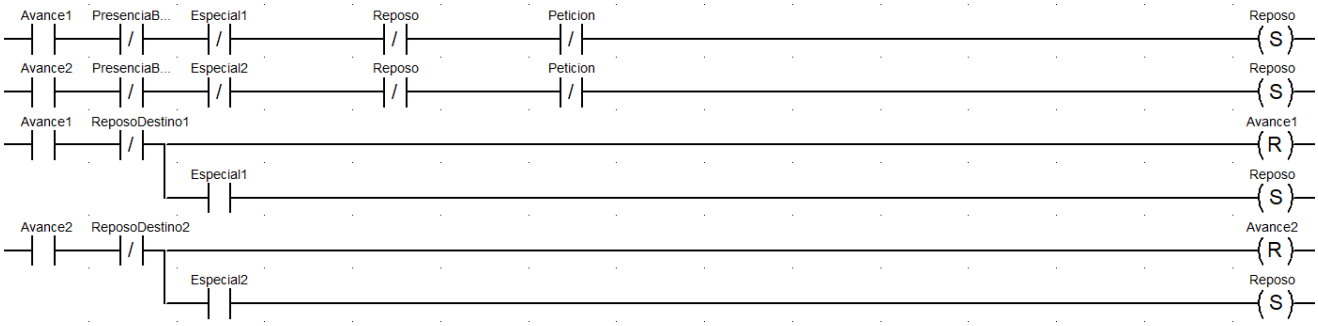


Figura 30 : Activación de Reposo y desactivación de Avance

Si la plataforma o retenedor permanece en estado de reposo y mediante el sensor de presencia se detecta la llegada de una bandeja, se realiza un cambio de estados pasando de Reposo a Petición.

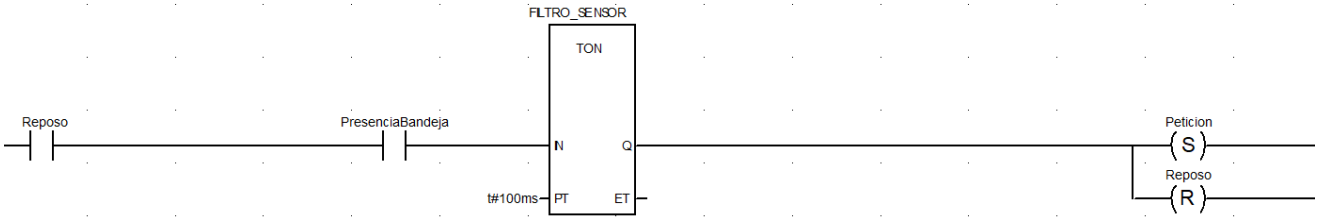


Figura 31 : Cambio de Reposo a Petición

Las condiciones para que la plataforma pase a estado de Avance son las siguientes:

- La plataforma o retenedor siguiente debe estar en estado de reposo.
- El pulsador de emergencia no debe de estar pulsado.
- No debe existir un avance simultaneo con otra bandeja que comparte el mismo destino la cual se debe definir en la variable "AvanceAuxX".
- La variable "BloqueoX" debe de estar desactivada.
- La variable "AcuseEnvioInfo" debe de estar activada.

Estas dos últimas variables públicas, se puedes activar/desactivar desde fuera de la sección de control lo que hace posible una manipulación según se requiera.

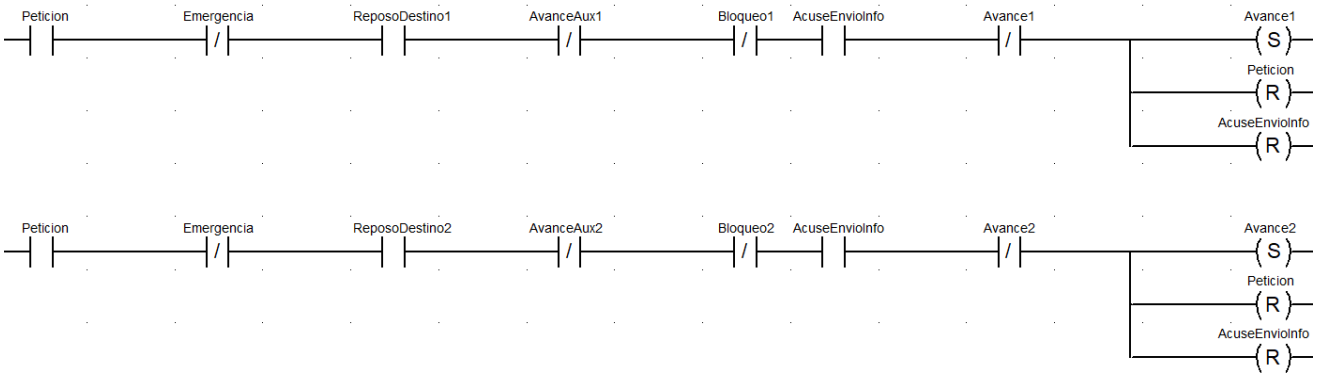
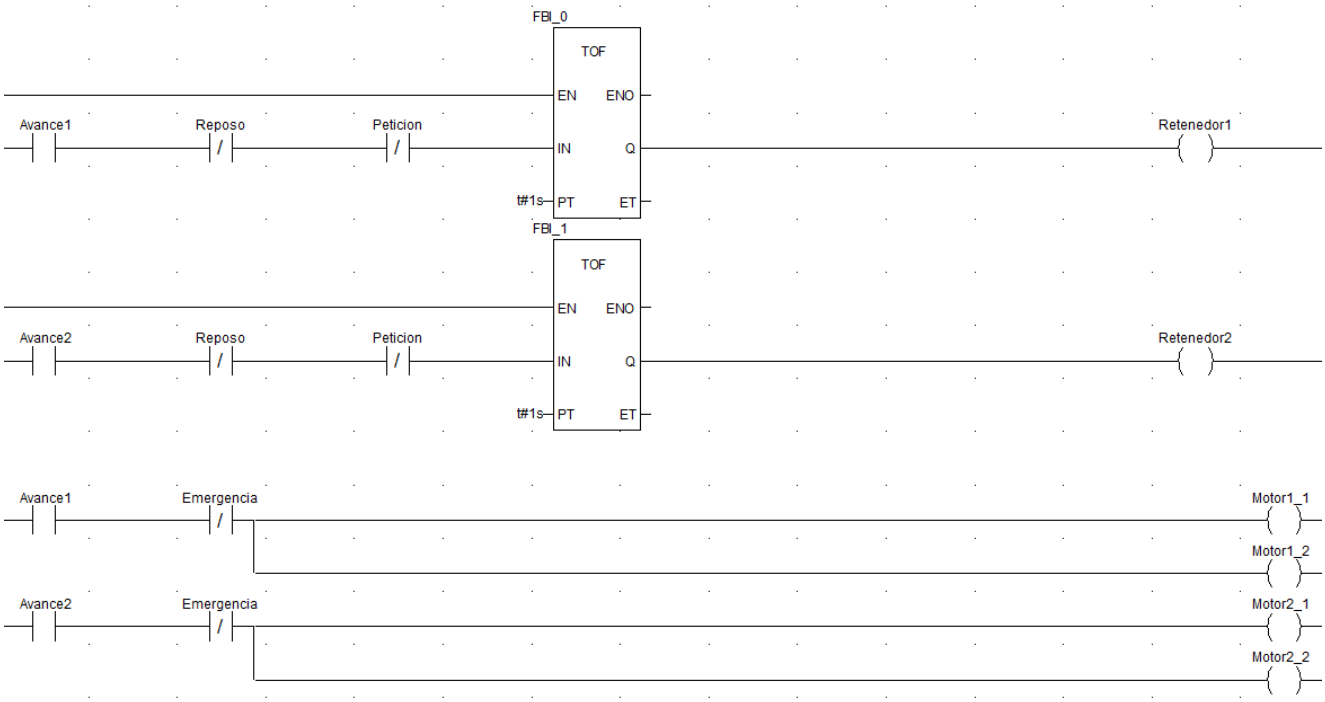


Figura 32 : Cambio de Petición a Avance

Una vez activado el estado de Avance, activa los bits de "RetenedorX" a la vez que "MotorX_X".



La variable "AcuseEnvioInfo" está diseñada para poder realizar una confirmación de envío de los datos de la bandeja mediante el bit "ConfirmacionEnvioInfoX" de la sección "Seguimiento" enviada desde el destino:

```
IF Avance1 AND NOT FAvance1 THEN
        BandejaSalida1 := BandejaEntrada;
END_IF;

IF Avance2 AND NOT FAvance2 THEN
        BandejaSalida2 := BandejaEntrada;
END_IF;

IF FAvance1 AND NOT Avance1 THEN
        BandejaDestino1 := BandejaSalida1;
        ConfirmacionEnvioInfo1 := TRUE;
END_IF;

IF FAvance2 AND NOT Avance2 THEN
        BandejaDestino2:= BandejaSalida2;
        ConfirmacionEnvioInfo2 := TRUE;
END_IF;

IF Reposo AND NOT FReposo THEN
        BandejaEntrada := BandejaVacia;
END_IF;

FAvance1 := Avance1;
FAvance2 := Avance2;
FReposo :=Reposo;
```

Figura 33 : Sección Seguimiento

La sección "Setandreset" está diseñada para realizar un reinicio estableciendo el estado de Reposo en la plataforma en el caso del inicio del programa o un rearme después de haberse realizado un paro de emergencia.

Este cambio de estado afecta al envío de datos, por lo tanto, en el caso de rearme se establece a 0 toda la información de bandeja que provenga de la plataforma anterior.

```
IF %S13 OR Rearme  THEN
        Reposo  := TRUE;
        Peticion := FALSE;
        Avance1  := FALSE;
        Avance2  := FALSE;
        Bloqueo1 := FALSE;
        Bloqueo2 := FALSE;
END_IF;

IF Rearme THEN
        BandejaEntrada := BandejaVacia;
END_IF;
```

Figura 34 : Sección "Setandreset"

Por último, la sección de "Almacenaje" tiene el objetivo de realizar un tratamiento de los datos de la bandeja existente de la plataforma para poder usar externamente en aplicaciones para la supervisión. En este proyecto no está contemplado el uso de dicha sección la cual usa la estructura "DatosEstado".

```
DatosEstado.BandejaEntrada := BandejaEntrada;
DatosEstado.BandejaSalida1 := BandejaSalida1;
DatosEstado.BandejaSalida2 := BandejaSalida2;

DatosEstado.Estado := BOOL_TO_INT(Reposo) + BOOL_TO_INT(Peticion)*2 +
                      BOOL_TO_INT(Avance1)*2*2 + BOOL_TO_INT(Avance2)*2*2*2;
```

Figura 35 : Sección "Almacenaje"

La llamada del bloque con los parámetros establecidos tiene la siguiente forma en los diferentes lenguajes de programación:

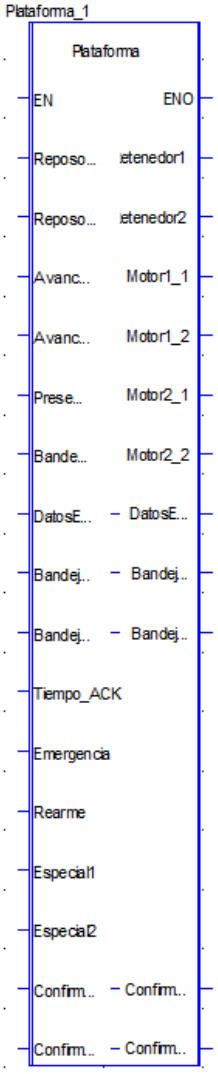


Figura 36 : Llamada del FB "Plataforma" en lenguaje LD

```
Plataforma_1 (ReposoDestino1 := (*BOOL*),
              ReposoDestino2 := (*BOOL*),
              AvanceAux1 := (*BOOL*),
              AvanceAux2 := (*BOOL*),
              PresenciaBandeja := (*BOOL*),
              BandejaVacia := (*Bandeja*),
              DatosEstado := (*DatosEstado*),
              BandejaDestino1 := (*Bandeja*),
              BandejaDestino2 := (*Bandeja*),
              Tiempo_ACK := (*TIME*),
              Emergencia := (*BOOL*),
              Rearme := (*BOOL*),
              Especial1 := (*BOOL*),
              Especial2 := (*BOOL*),
              ConfirmacionEnvioInfo1 := (*BOOL*),
              ConfirmacionEnvioInfo2 := (*BOOL*),
              Retenedor1 => (*BOOL*),
              Retenedor2 => (*BOOL*),
              Motor1_1 => (*BOOL*),
              Motor1_2 => (*BOOL*),
              Motor2_1 => (*BOOL*),
              Motor2_2 => (*BOOL*));
```

Figura 37 : Llamada del FB "Plataforma" en lenguaje ST

4.2 Programa base línea Modbus

4.2.1 Configuración de hardware

El software utilizado es SoMachine de Schneider Electric, un entorno de desarrollo basado en **Codesys**, para programar un PLC Modicon M251.

La configuración de hardware del proyecto facilitado es más compacta, ya que dispone de un puerto serie y dos puertos Ethernet. Uno está conectado al PLC de la linea CAN con la IP configurada en el mismo rango para poder realizar la comunicación entre ambos y el otro conectado a los módulos de la periferia descentralizada por diferentes islas Advantys por protocolo Modbus.

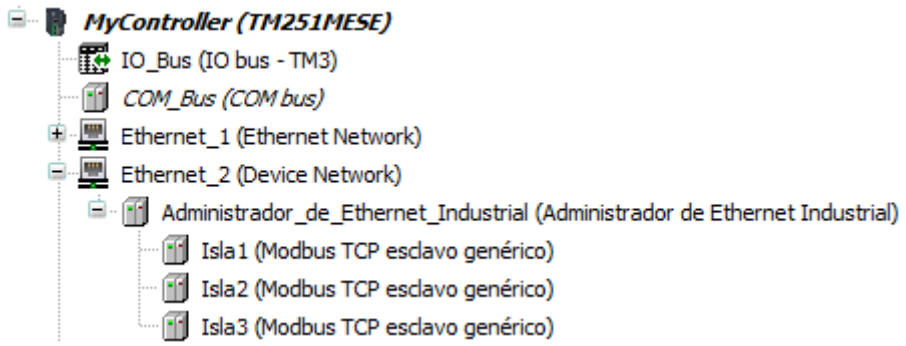


Figura 38 : Configuración hardware linea Modbus

4.2.2 Variables del programa

La gestión de variables en este entorno de desarrollo difiere bastante al tradicional como se ha visto en EcoStruxure.

Codesys ofrece un entorno de desarrollo dónde se facilita la programación orientada a objetos ya que las variables se pueden declarar en cada bloque de programa.

Se pueden declarar las variables públicas por separado, en este caso separándose por entradas físicas, salidas físicas, sensores y variables globales:

- Variables globales:

```
VAR_GLOBAL
    EMERGENCIA: BOOL;
    PRODUCT_0: PRODUCTO;
    REARME: BOOL;
    S12: BOOL;
    S13: BOOL:=TRUE;
END_VAR
```

Figura 39 : Variables globales linea Modbus

- Variables de entrada:

```
VAR_GLOBAL
    DC01    AT  %IX38.5:    BOOL;
    DFT3    AT  %IX30.2:    BOOL;
    DFT4    AT  %IX34.1:    BOOL;
    DI08    AT  %IX30.3:    BOOL;
    DI09    AT  %IX34.0:    BOOL;
    DI10    AT  %IX42.0:    BOOL;
    DIB03   AT  %IX34.3:    BOOL;
    DIB04   AT  %IX38.3:    BOOL;
    DIB05   AT  %IX46.0:    BOOL;

    (*Variable para pulsador de emergencia lateral izquierdo*)
    EM      AT  %IX34.4:    BOOL;
    (*Pulsador de Emergencia del medio, al lado de PT12*)
    EM2     AT  %IX42.5:    BOOL;

    P_ALL_Plates1   AT  %IX38.0:    BOOL;
    (*Pulsador 1 lateral izquierdo*)
    P01     AT  %IX34.5:    BOOL;
    (*Pulsador 2 lateral izquierdo*)
    P02     AT  %IX34.6:    BOOL;
    (*Pulsador 3, al lado de la emergencia 2*)
    P03     AT  %IX42.3:    BOOL;
    P04     AT  %IX42.4:    BOOL;

    S_DIR10 AT  %IX54.0:    BOOL;
    S_DIR11 AT  %IX30.4:    BOOL;
    S_DIR12 AT  %IX30.5:    BOOL;
    S_DIR13 AT  %IX46.1:    BOOL;
    S_DIR14 AT  %IX34.2:    BOOL;
    S_DIR15 AT  %IX38.4:    BOOL;
    S_DIR16 AT  %IX38.1:    BOOL;
    S_DIR17 AT  %IX38.2:    BOOL;
    S_DIR18 AT  %IX46.3:    BOOL;
    S_DIR19 AT  %IX46.2:    BOOL;
    DIB02   AT  %IX30.0:    BOOL;
END_VAR
```

Figura 40 : Variables de entrada linea Modbus

- Variables de salida:

```
VAR_GLOBAL
    DIR10_BAJA  AT  %QX22.2:    BOOL;
    DIR11_BAJA  AT  %QX22.0:    BOOL;
    DIR12_BAJA  AT  %QX22.1:    BOOL;
    DIR13_BAJA  AT  %QX30.2:    BOOL;
    DIR14_BAJA  AT  %QX20.2:    BOOL;
    DIR15_BAJA  AT  %QX28.2:    BOOL;
    DIR16_BAJA  AT  %QX20.4:    BOOL;
    DIR17_BAJA  AT  %QX20.5:    BOOL;
    DIR18_BAJA  AT  %QX28.4:    BOOL;
    DIR19_BAJA  AT  %QX30.0:    BOOL;

    M16     AT  %QX34.1:    BOOL;   (* motor PT09 to DIR14 *)
    M17     AT  %QX34.2:    BOOL;   (* motor PT10 to DIR15 *)
    M19     AT  %QX34.3:    BOOL;
    M20     AT  %QX34.4:    BOOL;
    M20_reverse AT  %QX34.5:    BOOL;
    M21     AT  %QX34.0:    BOOL;
    M22     AT  %QX36.0:    BOOL;

    MOTOR_CENTRALBAND   AT  %QX22.4:    BOOL;
    MOTOR_RIGHTBAND     AT  %QX22.5:    BOOL;

    PT08_SUBE   AT  %QX30.3:    BOOL;
    PT09_BAJA   AT  %QX20.1:    BOOL;
    PT09_SUBE   AT  %QX20.0:    BOOL;
    PT10_BAJA   AT  %QX28.1:    BOOL;
    PT10_SUBE   AT  %QX28.0:    BOOL;
    PT11_SUBE   AT  %QX20.3:    BOOL;
    PT12_SUBE   AT  %QX28.3:    BOOL;
    PT13_SUBE   AT  %QX28.5:    BOOL;
    PT14_SUBE   AT  %QX30.1:    BOOL;
END_VAR
```

Figura 41 : Variables de salida linea Modbus

- Sensores:

```
VAR_GLOBAL
    Sensor_DIR10: BOOL;
    Sensor_DIR11: BOOL;
    Sensor_DIR12: BOOL;
    Sensor_DIR13: BOOL;
    Sensor_DIR14: BOOL;
    Sensor_DIR15: BOOL;
    Sensor_DIR16: BOOL;
    Sensor_DIR17: BOOL;
    Sensor_DIR18: BOOL;
    Sensor_DIR19: BOOL;

    Sensor_DIB02: BOOL;
    Sensor_DIB03: BOOL;
    Sensor_DIB04: BOOL;
    Sensor_DIB05: BOOL;

    Sensor_DI08: BOOL;
    Sensor_DI09: BOOL;
    Sensor_DI10: BOOL;
END_VAR
```

Figura 42 : Variables para el tratamiento de sensores

4.2.3 Secciones del programa

Como en el programa de la línea CAN, aparecen definidos varios bloques de programas equivalentes como el tratamiento de sensores para un idéntico comportamiento y las activaciones de las salidas sin completar.

En este caso, el programa base está menos completo que el anterior.

4.2.4 Bloque de funciones (FB) y estructuras

En el programa facilitado existe el mismo bloque de función "Plataforma" con las mismas estructuras de variables.

En este caso "PRODUCTO" es equivalente a "Bandeja" del programa de la línea CAN.

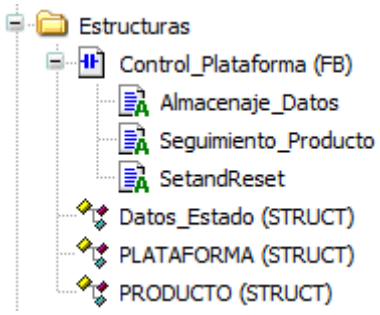


Figura 43 : Estructuras de datos y FB existentes y sus secciones

5. Diseño del programa

5.1 Línea CAN

En primer lugar, es necesario definir el correcto funcionamiento del traslado de bandejas entre plataformas y retenedores mediante los bloques de función "Plataforma".

Se ha creado un bloque de programa para agrupar todas las instancias creadas mediante el bloque de funciones para cada plataforma/retenedor e introducir los parámetros correspondientes:

```
PT04_FB (ReposoDestino1 := DIR04_FB.Reposo,
         PresenciaBandeja := Sensor_PT04,
         BandejaVacia := BandejaVacia,
         BandejaDestino1 := DIR04_FB.BandejaEntrada,
         Emergencia := EMERGENCY_LOCK,
         Rearme := REARME,
         ConfirmacionEnvioInfo1 := DIR04_FB.AcuseEnvioInfo,
         Retenedor1 => RET_PT04,
         Motor1_1 => ActivaMotor_PT04);

DIR04_FB (ReposoDestino1 := PT05_FB.Reposo,
          PresenciaBandeja := Sensor_DIR04,
          BandejaVacia := BandejaVacia,
          BandejaDestino1 := PT05_FB.BandejaEntrada,
          Emergencia := EMERGENCY_LOCK,
          Rearme := REARME,
          ConfirmacionEnvioInfo1 := PT05_FB.AcuseEnvioInfo,
          Retenedor1 => RET_DIR04,
          Motor1_1 => ActivaMotor_DIR04);

PT05_FB (ReposoDestino1 := DIR05_FB.Reposo,
         ReposoDestino2 := DIR08_FB.Reposo,
         PresenciaBandeja := Sensor_PT05,
         BandejaVacia := BandejaVacia,
         BandejaDestino1 := DIR05_FB.BandejaEntrada,
         BandejaDestino2 := DIR08_FB.BandejaEntrada,
         Emergencia := EMERGENCY_LOCK,
         Rearme := REARME,
         Especial1 := TRUE,
         ConfirmacionEnvioInfo1 := DIR05_FB.AcuseEnvioInfo,
         ConfirmacionEnvioInfo2 := DIR08_FB.AcuseEnvioInfo,
         Retenedor1 => RET_PT05_1,
         Retenedor2 => RET_PT05_2,
         Motor1_1 => ActivaMotor_PT05_11,
         Motor2_1 => ActivaMotor_PT05_21,
         Motor2_2 => ActivaMotor_PT05_22);
```

Figura 44 : Instancias de plataformas y retenedores de la línea CAN

Los principales parámetros que se necesitan son:

- **ReposoDestino**: Estado de reposo de la instancia de la plataforma destino.
- **PresenciaBandeja**: Detector de presencia correspondiente.
- **BandejaVacia**: Instancia con estructura "Bandeja" con valor 0 en todo el array para el reset después de un rearme.
- **BandejaDestino**: Estructura "Bandeja" que contiene la información del producto (ID, tipo y estado).
- **Rearme**: Variable activada por el botón de rearme.
- **Especial**: Bit para dejar activado el estado de Avance cuando se requiere de un desvío transversal.
- **ConfirmaciónEnvioInfo**: Bit de **AcuseEnvioInfo** de la instancia de la plataforma de destino para la confirmación del traslado de información de datos.
- **Retenedor**: Activación del actuador que deja avanzar la bandeja alojada.
- **MotorX_X**: Bit para la activación de las cintas correspondientes para el traslado de la bandeja.

Las variables introducidas como parámetros de salida en cada objeto de plataforma se utilizan como condiciones en la sección de salidas las cuales activan los bits de las variables con dirección física.

A continuación, un ejemplo de varias salidas:

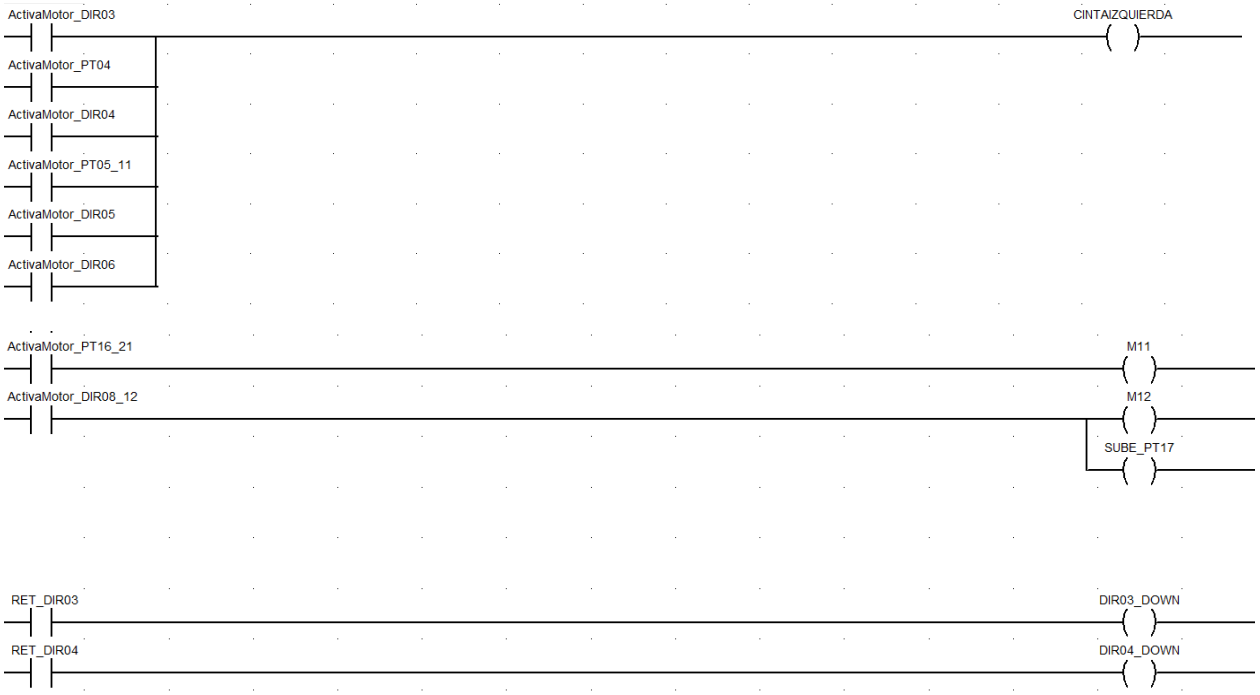


Figura 45 : Activación de salidas físicas de la línea CAN

Para poder realizar la automatización de procesos según el objetivo se necesita agrupar todas las casuísticas y condiciones varias que puedan existir. Establecer todo esto en un diagrama de estados es bastante útil para tener claro los pasos a seguir.

Una característica a tener en cuenta es que los procesos descritos en el concepto, tanto estaciones de trabajo como controles de estados de pieza no existen físicamente, por lo tanto, se procede a definir una simulación mediante la generación de resultados aleatorios, estableciendo una probabilidad distinta en cada uno de dichos procesos obteniendo un resultado diferente.

Una vez realizada la programación aplicada al traslado de bandejas, se procede a crear distintos bloques de programa para tratar y modificar los datos de información sobre el producto.

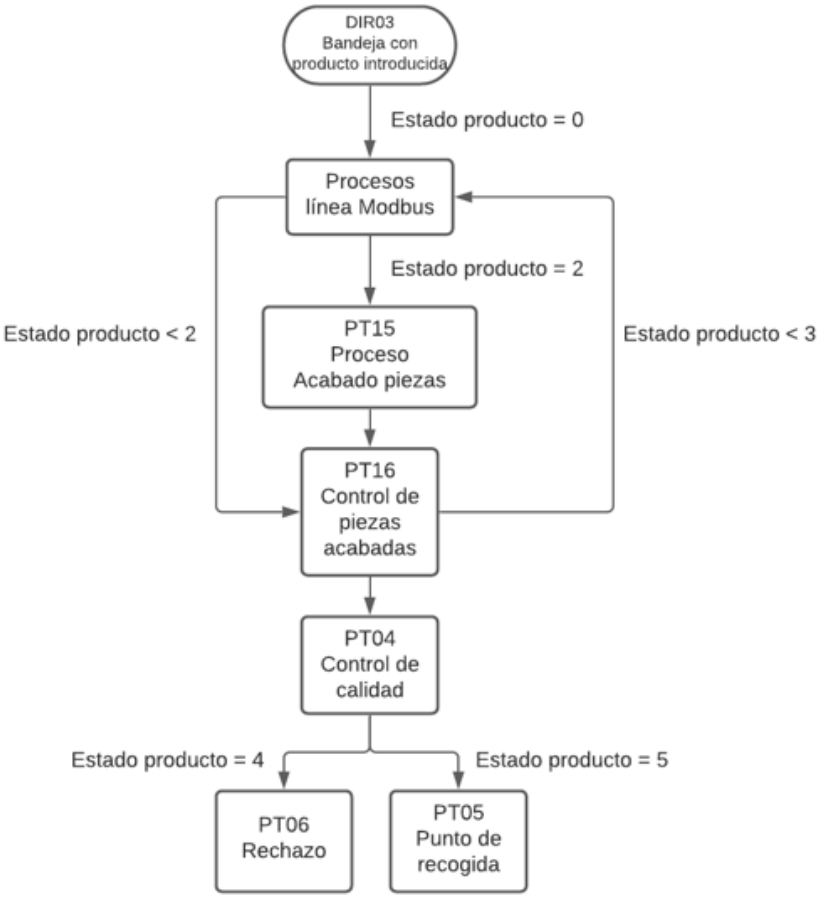


Figura 46 : Tratamiento de producto en línea CAN

Para poder realizar una generación de un número aleatorio se necesita importar una librería externa que contiene el bloque de función.

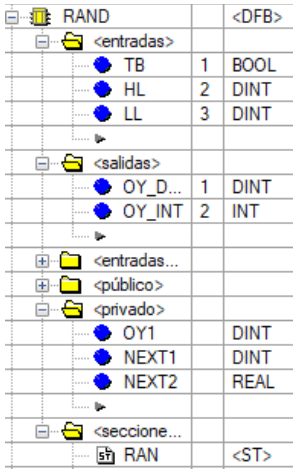


Figura 47 : Estructura del FB "RAND"

Este bloque de funciones consiste en la generación de un valor de tipo DINT (doble entero) aleatorio dentro del rango introducido.

```
(*If %S6=1 then If OY<LL
then OY:=LL;
else OY:=OY1;
END_IF;
If OY>HL
then OY:=HL;
else OY:=Oy1;
END_IF;
END_IF;*)

next1 := next1 * 1103515245 + 12345;
next2 := DINT_TO_REAL (IN := next1);
    OY1:=Real_to_DInt(((next2/65536.0)*(1.0/100.0))*32767.0);
OY_DINT := LIMIT (MN := ll,IN := OY1,          MX := HL);

OY_INT := DINT_TO_INT (OY_DINT);
```

Figura 48 : FB "RAND"

Se ha añadido una conversión para que el valor devuelto sea de tipo INT (entero) para que el posterior uso de este resultado se pueda tratar fácilmente.

Las distintas secciones programadas para llevar a cabo el comportamiento deseado son:

- IINICIO
- PT04
- PT05
- PT06
- PT15
- PT16

En la sección de INICIO se establece un pulso cada vez que se detecta que se ha introducido una bandeja en el retenedor DIR03 donde se genera un numero aleatorio del1 al 3. El resultado definirá el tipo de producto que alojará dicha bandeja.

El estado siempre tendrá el valor de 0 ya que es un producto sin procesar y el valor de ID (identificación) será un número mayor que la anterior.

```
R_TRIG_0 (CLK := Sensor_DIR03,
          Q => Sensor_DIR03_pulso);

IF Sensor_DIR03_pulso THEN

        RAND_0 (TB := TRUE,
                HL := 3,
                LL := 1,
                OY_INT => TipoProducto_Random);

        DIR03_FB.BandejaEntrada.IDPRODUCTO := DIR03_FB.BandejaEntrada.IDPRODUCTO + 1;
        DIR03_FB.BandejaEntrada.TIPOPRODUCTO := TipoProducto_Random;
        DIR03_FB.BandejaEntrada.ESTADO := 0;

        DIR03_FB.AcuseEnvioInfo          := TRUE;

END_IF;
```

Figura 49 : Sección INICIO

En la plataforma PT04 se define el control de calidad. Este detiene la bandeja durante 15 segundos si el estado del producto es igual a 3, lo que quiere decir que el producto ha pasado por todos los procesos correspondientes.

Hay una probabilidad de que el 25% del producto no pase el control de calidad utilizando el bloque de funciones RAND.

Si el resultado ha sido satisfactorio, el estado del producto pasa a tener un valor de 5, por el contrario, se establece un valor de 4, lo que se traduce como que el producto debe ser rechazado ya que no ha pasado correctamente el control de calidad

.

```
(* CONTROL CALIDAD

  Hay una probabilidad del 25% de que el producto no pase el control de calidad.
  Si ControlCalidad_Resultado <= 3, el producto debe volver a pasar de nuevo.
  Si ControlCalidad_Resultado = 1, el producto debe pasar por control de calidad.

  0 = Sin procesar
  1, 2 = No acabado
  3 = Acabado
  4 = Defectuoso -> Avanza por DIR04 para retirar en PT06
  5 = Procesado correctamente -> Avanza por DIR04 para retirar en PT05
*)


IF PT04_FB.BandejaEntrada.ESTADO = 3 THEN

        PT04_FB.Bloqueo1 := TRUE;

        TON_ControlCalidad (IN := DIB01,
                    PT := t#15s,
                    Q => ControlCalidad_OK,
                    ET => ControlCalidad_Tiempo);

END_IF;

IF ControlCalidad_OK THEN

        RAND_1 (TB := TRUE,
                HL := 4,
                LL := 1,
                OY_INT => ControlCalidad_Resultado);

        IF ControlCalidad_Resultado = 1 THEN
                PT04_FB.BandejaEntrada.ESTADO := 4;
        ELSE
                PT04_FB.BandejaEntrada.ESTADO := 5;
        END_IF;

        PT04_FB.Bloqueo1 := FALSE;

END_IF;
```

Figura 50 : Sección PT04 – Control de calidad

La PT05 se ha definido como punto de recogida de productos correctamente procesados. En el programa se realiza un bloqueo del avance de la bandeja cuando el producto ha pasado el control de calidad. Cuando la bandeja es retirada, vuelve a su comportamiento normal.

Si el producto tiene un valor en su estado equivalente a rechazo, se fuerza la dirección de la plataforma hacia DIR05 para ser rechazado en la PT06.

```
(* RECOGIDA DEL PRODUCTO *)

(* 0 = Sin procesar -> Avanza por RET5 o RET8 según disponibilidad
   1, 2 = No acabado -> Pasar por PR5, avanza por RET5 o RET8 según disponibilidad
   3 = Acabado
   4 = Defectuoso -> Avanza por DIR05 para retirar en PT06
   5 = Procesado correctamente
 *)


IF PT05_FB.BandejaEntrada.ESTADO = 5 THEN

        PT05_FB.Bloqueo1:= TRUE;
        PT05_FB.Bloqueo2 := TRUE;

ELSE

        IF Sensor_DIR08 OR PT05_FB.BandejaEntrada.ESTADO = 4 THEN
                PT05_FB.Bloqueo2 := TRUE;
                PT05_FB.Bloqueo1 := FALSE;

        ELSE
                PT05_FB.Bloqueo1 := TRUE;
                PT05_FB.Bloqueo2 := FALSE;
        END_IF;

END_IF;

IF NOT Sensor_PT05 THEN

        PT05_FB.Bloqueo1:= FALSE;
        PT05_FB.Bloqueo2 := FALSE;

END_IF;
```

Figura 51 : Sección PT05 - Recogida del producto

```
(* RECOGIDA DEL PRODUCTO DEFECTUOSO

  0 = Sin procesar
  1, 2 = No acabado
  4 = Defectuoso -> Retirar
 *)


IF PT06_FB.BandejaEntrada.ESTADO = 4 THEN

        PT06_FB.Bloqueo1:= TRUE;

ELSE IF PT06_FB.BandejaEntrada.ESTADO < 4 OR NOT Sensor_PT06 THEN

        PT06_FB.Bloqueo1:= FALSE;

END_IF;
END_IF;
```

Figura 52 : Sección PT06 - Recogida del producto defectuoso

En la PT15 se aloja la estación de trabajo P5, que aplica a todos los productos si estos han sido correctamente procesados en las estaciones de trabajo correspondientes en la linea Modbus.

Posteriormente avanza hasta la PT17, donde se realiza un control de piezas acabadas. Si el valor del estado del producto no es equivalente al de una pieza que ha pasado por el acabado, será devuelto a la línea Modbus para volver a ser procesado correctamente.

```
(* ESTACIÓN DE TRABAJO 5 (P5) - ACABADO

  El PR5 consiste en el acabado de cualquier tipo de producto.
  Hay una probabilidad del 33% de que el acabado sea NO satisfactorio.
  Si PR5_Resultado <= 2, el producto debe volver a pasar de nuevo.
  Si PR5_Resultado = 3, el producto debe pasar por control de calidad.

  1, 2 = No acabado
  3 = Acabado
*)

IF PT15_FB.BandejaEntrada.ESTADO = 2 THEN

        PT15_FB.Bloqueo1 := TRUE;

        TON_PR5 (IN := DIB06,
                 PT := t#10s,
                 Q => PR5_OK,
                 ET => PR5_Tiempo);
END_IF;

IF PR5_OK THEN

        RAND_1 (TB := TRUE,
                HL := 3,
                LL := 1,
                OY_INT => PR5_Resultado);

        IF PR5_Resultado <> 1 THEN
                PT15_FB.BandejaEntrada.ESTADO := 3;
        END_IF;

        PT15_FB.Bloqueo1 := FALSE;

END_IF;
```

Figura 53 : Sección PT15 - Estación de trabajo P5 – Acabado

```
(* CONTROL DE PIEZAS ACABADAS

  0, 1, 2 = No acabado -> Avanzar hacia PT17
  3 = Acabado -> Avanzar por DIR07
 *)


IF PT16_FB.BandejaEntrada.ESTADO < 3 THEN

        PT16_FB.Bloqueo1:= FALSE;
        PT16_FB.Bloqueo2:= TRUE;

ELSE

        PT16_FB.Bloqueo1:= TRUE;
        PT16_FB.Bloqueo2:= FALSE;

END_IF;
```

Figura 54: Sección PT17 - Control de piezas acabadas

En último lugar, se crean dos bloques de programa para la comunicación con la línea Modbus. Utilizando los bloques necesarios para establecer una conexión mediante Ethernet.

En dichos bloques de funciones se definen los parámetros para que lea y escriba en una dirección de memoria concreta donde se aloja el buffer de cada PLC.

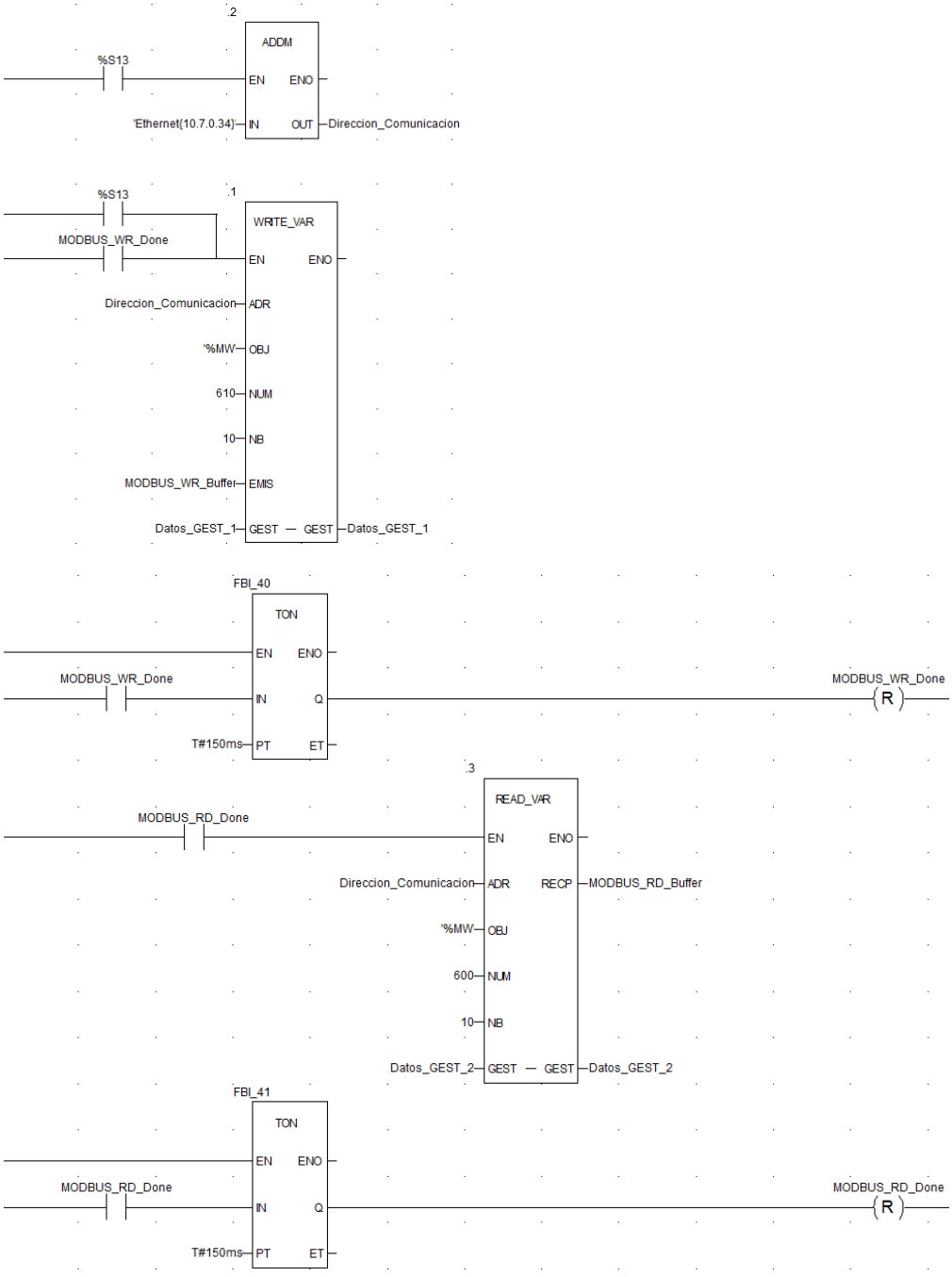


Figura 55 : Sección para la comunicación con la línea Modbus

```
PT08_FB.Reposo                          := MODBUS_RD_Buffer[1].0;
PT08_FB.Avance1                         := MODBUS_RD_Buffer[1].1;
PT15_FB.AcuseEnvioInfo                  := MODBUS_RD_Buffer[1].2;
PT15_FB.BandejaEntrada.IDPRODUCTO       := MODBUS_RD_Buffer[2];
PT15_FB.BandejaEntrada.TIPOPRODUCTO     := MODBUS_RD_Buffer[3];
PT15_FB.BandejaEntrada.ESTADO           := MODBUS_RD_Buffer[4];
MODBUS_RD_Done := TRUE;

IF Bandeja_PT08_Previa.ESTADO <> PT08_FB.BandejaEntrada.ESTADO
OR Bandeja_PT08_Previa.IDPRODUCTO <> PT08_FB.BandejaEntrada.IDPRODUCTO
OR Bandeja_PT08_Previa.TIPOPRODUCTO <> PT08_FB.BandejaEntrada.TIPOPRODUCTO
OR PT08_Acuse_Previo <> PT08_FB.AcuseEnvioInfo
OR PT15_Reposo_Previo <> PT15_FB.Reposo
OR PT15_Avance_Previo <> PT15_FB.Avance1 THEN

        MODBUS_WR_Buffer[1].0   := PT15_FB.Reposo;
        MODBUS_WR_Buffer[1].1   := PT15_FB.Avance1;
        MODBUS_WR_Buffer[1].2   := PT08_FB.AcuseEnvioInfo;
        MODBUS_WR_Buffer[2]     := PT08_FB.BandejaEntrada.IDPRODUCTO;
        MODBUS_WR_Buffer[3]     := PT08_FB.BandejaEntrada.TIPOPRODUCTO;
        MODBUS_WR_Buffer[4]     := PT08_FB.BandejaEntrada.ESTADO;
        MODBUS_WR_Done := TRUE;

END_IF;

PT08_Acuse_Previo := PT08_FB.AcuseEnvioInfo;
Bandeja_PT08_Previa := PT08_FB.BandejaEntrada;
PT15_Reposo_Previo := PT15_FB.Reposo;
PT15_Avance_Previo := PT15_FB.Avance1;
```

Figura 56 : Sección para la comunicación con el buffer de la línea Modbus

5.2 Línea Modbus

Para realizar un comportamiento del funcionamiento del traslado de bandejas idéntico al de la línea CAN, se ha procedido de igual manera instanciando mediante los bloques de función "Plataforma" de cada plataforma/retenedor.

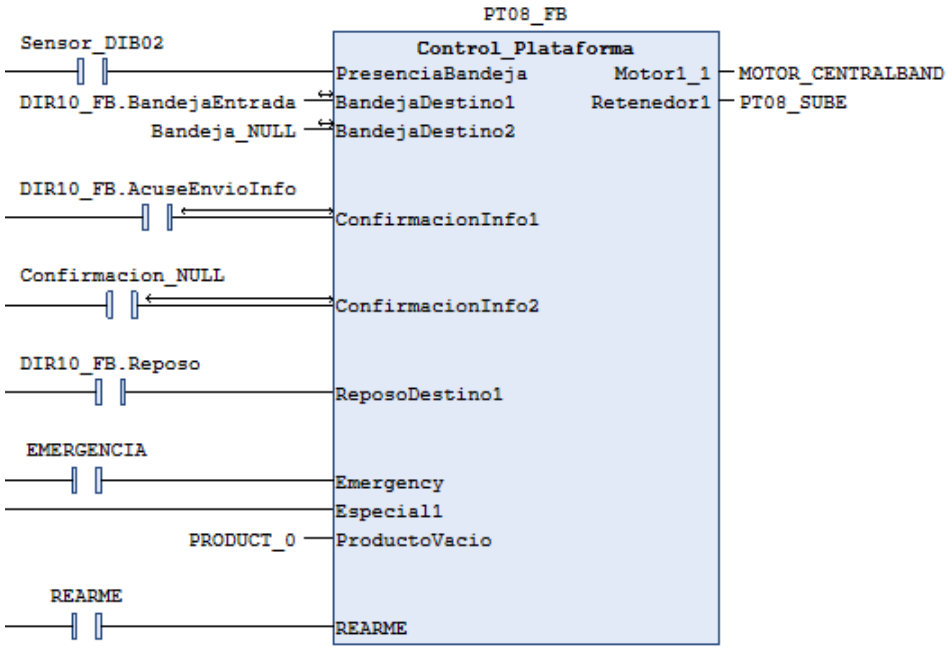


Figura 57 : Instancia de una plataforma con parámetros introducidos de la línea Modbus

En este caso, para simplificar el número de bloques de programa se ha establecido la salida del bloque a la dirección física directamente.

Como en el programa base de esta línea no existe ningún comportamiento definido referente al paro de emergencia y su rearme, se crea una sección parecida al de la línea CAN.

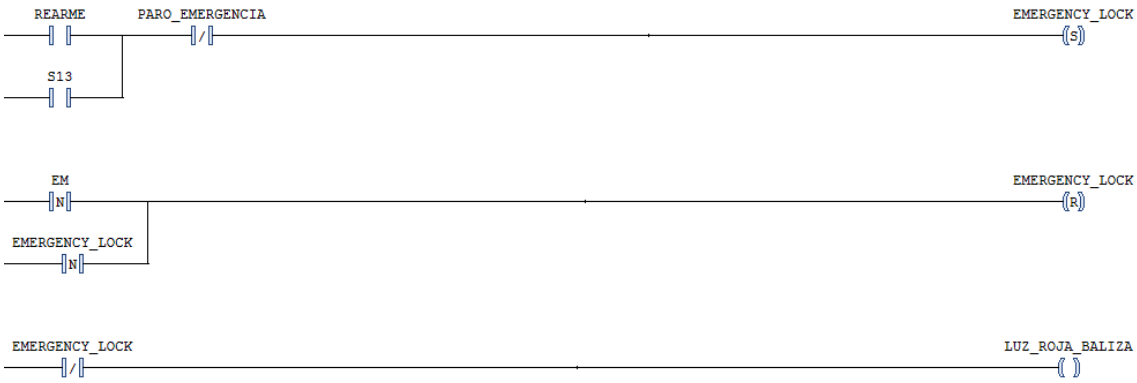


Figura 58 : Bloque de programa EM_REARME

Como se ha podido observar anteriormente, esta línea aloja las estaciones de trabajo P1, P2, P3 y P4. Estas estaciones de trabajo, al contrario de las secciones creadas por separado en el programa de la línea CAN, usarán un bloque de función creado específicamente para ellas. De esta manera se consigue minimizar el tamaño de código ya que el objetivo es cumplir con una estructura de programación orientadas a objetos.

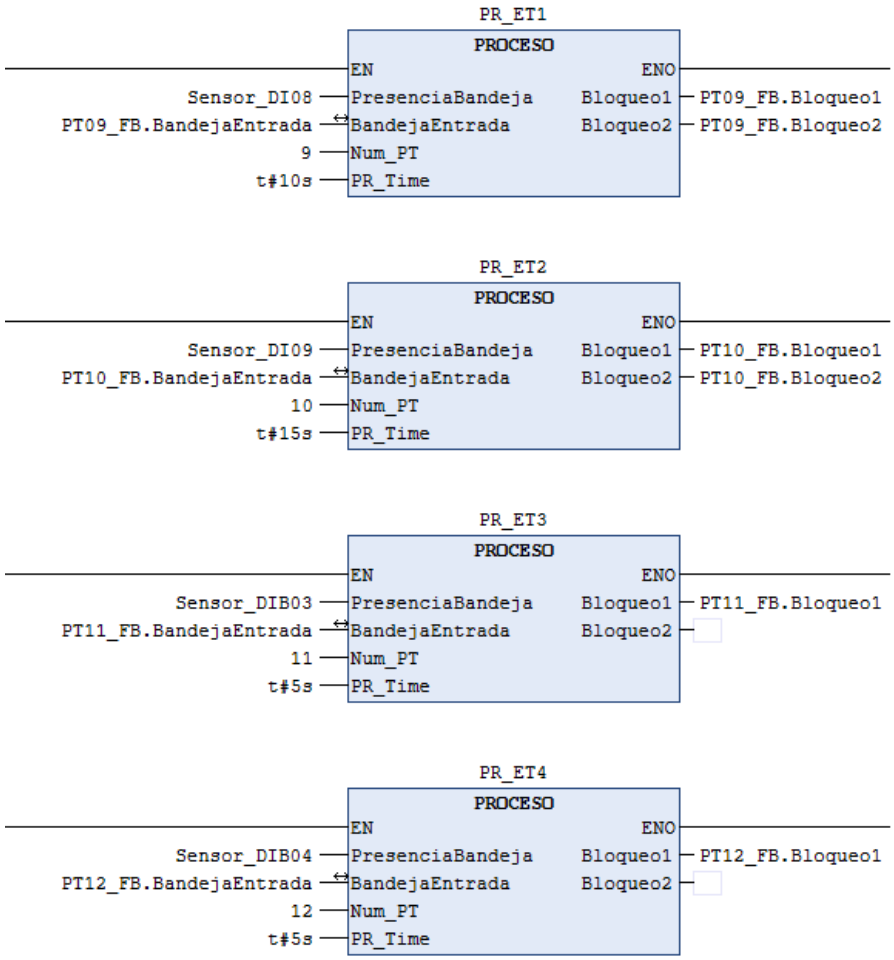


Figura 59 : Instancias creadas de las estaciones de trabajo de la línea Modbus

Principalmente los parámetros claves para dicho bloque de función llamado "PROCESO" son:

- **PresenciaBandeja:** Sensor correspondiente de la plataforma donde se aloja la estación de trabajo.
- **BandejaEntrada:** Estructura de tipo "Bandeja" para llevar a cabo el tratamiento de datos que se verán afectados según corresponda.
- **Num_PT:** Número de plataforma para distinguir el tipo de producto para aplicar el proceso.
- **PR_Time:** Tiempo de proceso.
- **BloqueoX:** Salidas para realizar el bloqueo mientras dura el proceso y poder desviar la bandeja de dirección si corresponde.

El bloque de función tiene la estructura siguiente:

```
FUNCTION_BLOCK PROCESO
VAR_INPUT
    PresenciaBandeja: BOOL;
    Tipo_Prod_1: INT;
    Tipo_Prod_2: INT;
    PR_Time: TIME;
END_VAR
VAR_OUTPUT
    Bloqueo1: BOOL;
    Bloqueo2: BOOL;
END_VAR
VAR_IN_OUT
    BandejaEntrada: PRODUCTO;
END_VAR
VAR
    PR_No_Aplica: BOOL;
    Time_IN: BOOL;
    TON_PR: TON;
    PR_RndNum: INT;
    PR_END: BOOL;
    PR_Time_OK: BOOL;
    PR_Succes: BOOL;
    Desvio_DIR1: BOOL;
    Desvio_DIR2: BOOL;
END_VAR
```

Figura 60 : Estructura del FB "PROCESO"

Cuando la bandeja pasa por el sensor de presencia de la plataforma en cuestión se realiza el bloqueo en las dos posibles direcciones. Seguidamente una sección de código chequea si se debe aplicar el producto al proceso de la estación de trabajo en la que se aloja.

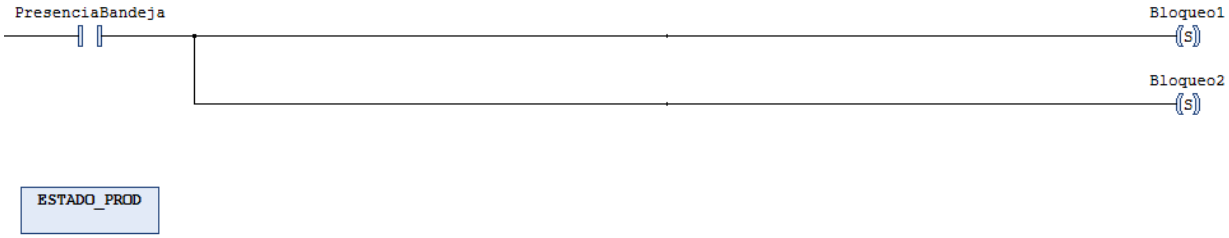


Figura 61 : Inicio de la rutina del FB "PROCESO"

```
IF NOT PR_END AND PresenciaBandeja AND(
    (BandejaEntrada.TIPOPRODUCTO = 1 AND BandejaEntrada.ESTADO = 0 AND Num_PT = 9) OR
    (BandejaEntrada.TIPOPRODUCTO = 1 AND BandejaEntrada.ESTADO = 1 AND Num_PT = 10) OR
    (BandejaEntrada.TIPOPRODUCTO = 2 AND BandejaEntrada.ESTADO = 0 AND Num_PT = 11) OR
    (BandejaEntrada.TIPOPRODUCTO = 2 AND BandejaEntrada.ESTADO = 1 AND Num_PT = 12) OR
    (BandejaEntrada.TIPOPRODUCTO = 3 AND BandejaEntrada.ESTADO = 0 AND Num_PT = 10) OR
    (BandejaEntrada.TIPOPRODUCTO = 3 AND BandejaEntrada.ESTADO = 1 AND Num_PT = 12)) THEN

    PR_No_Aplica := FALSE;
    Time_IN := TRUE;

ELSE

    PR_No_Aplica := TRUE;

END_IF
```

Figura 62 : Subrutina ESTADO_PROD del FB "PROCESO"

Si el producto tiene un estado al cual se le debe aplicar el proceso, se activará el temporizador con el tiempo establecido en los parámetros de entrada.

Una vez transcurrido el tiempo, se genera un número aleatorio el cual va a definir si el proceso ha sido satisfactorio en la sección del bloque "RESULTADO".

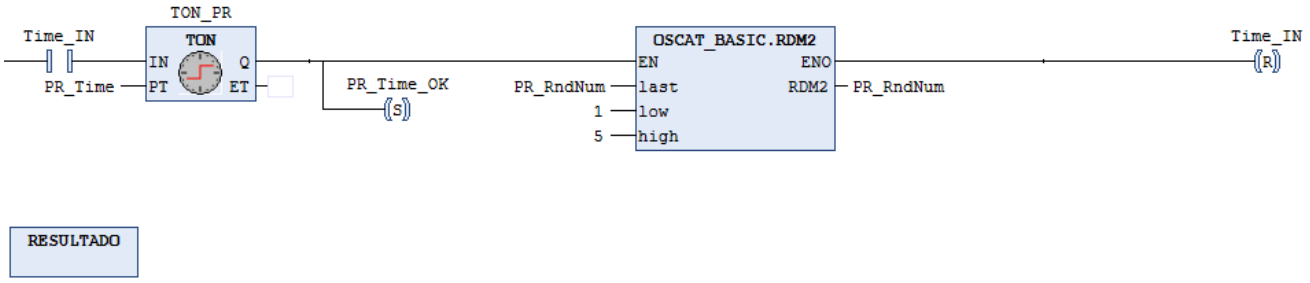


Figura 63 : Temporizador y generación aleatorio de resultado del FB "PROCESO"

Todos los procesos tienen una probabilidad de que el resultado no sea satisfactorio del 20% simulando esta mediante la generación del número aleatorio del bloque RDM2 importado de la librería externa OSCAT_BASIC.

En la sección "RESULTADO" se introducen las condiciones que se necesitan dependiendo del resultado del proceso llevando a cabo el desvío de la bandeja en la dirección hacía la estación de trabajo que corresponde al producto.

También se define el tratamiento de datos en el estado del producto. Si no ha sido procesado correctamente, su valor no cambiará.

```
IF PR_Time_OK AND PR_RndNum <5 THEN

    BandejaEntrada.ESTADO := BandejaEntrada.ESTADO + 1;

    PR_Succes := TRUE;
    PR_Time_OK := FALSE;

    IF (BandejaEntrada.TIPOPRODUCTO = 1 AND Num_PT = 9) OR
        (BandejaEntrada.TIPOPRODUCTO = 1 AND Num_PT = 10) THEN

        Desvio_DIR1 :=TRUE;

    ELSIF (BandejaEntrada.TIPOPRODUCTO = 3 AND Num_PT = 10) OR
        (BandejaEntrada.TIPOPRODUCTO = 2 AND BandejaEntrada.ESTADO <3 AND Num_PT = 9) THEN

        Desvio_DIR2 :=TRUE;

     END_IF;

     PR_END :=TRUE;

ELSIF PR_Time_OK AND PR_RndNum = 5 THEN

    BandejaEntrada.ESTADO := BandejaEntrada.ESTADO;

    PR_Time_OK := FALSE;
    PR_END := TRUE;

END_IF;
```

Figura 64 : Subrutina "RESULTADO" del FB "PROCESO"

El desbloqueo total o parcial depende de si se ha aplicado el proceso descrito anteriormente.

Figura 65 : Final de la rutina FB "PROCESO"

Por último, se necesita establecer la lectura y escritura de igual manera que en la línea CAN:

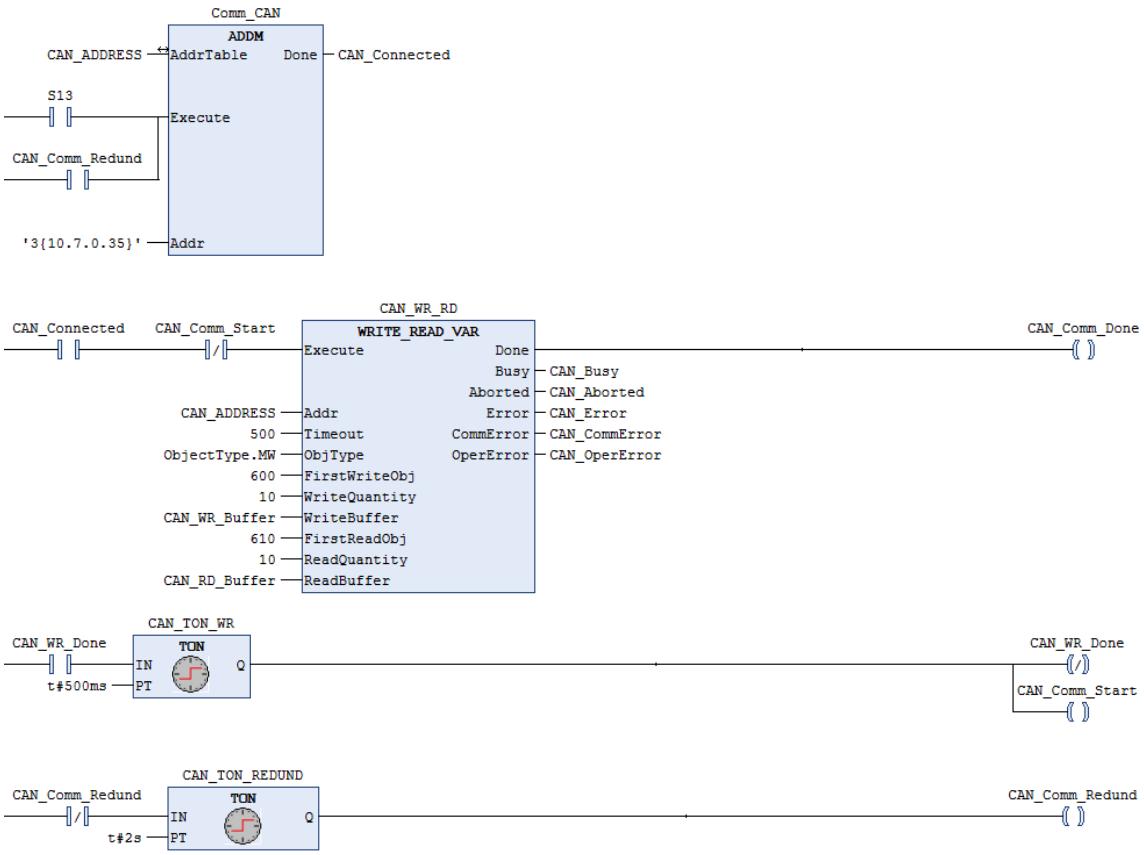


Figura 66 : Bloque de programa para la comunicación con la línea CAN

```
PT08_FB.Reposo                    := CAN_RD_Buffer[1].0;
PT08_FB.Avance1                   := CAN_RD_Buffer[1].1;
PT08_FB.AcuseEnvioInfo            := CAN_RD_Buffer[1].2;
PT08_FB.BandejaEntrada.IDPRODUCTO   := CAN_RD_Buffer[2];
PT08_FB.BandejaEntrada.TIPOPRODUCTO := CAN_RD_Buffer[3];
PT08_FB.BandejaEntrada.ESTADO       := CAN_RD_Buffer[4];;
CAN_RD_Done := TRUE;

CAN_WR_Buffer[1].0  := PT15_FB.Reposo;
CAN_WR_Buffer[1].1  := PT15_FB.Avance1;
CAN_WR_Buffer[1].2  := PT15_FB.AcuseEnvioInfo;
CAN_WR_Buffer[2]    := PT15_FB.BandejaEntrada.IDPRODUCTO;
CAN_WR_Buffer[3]    := PT15_FB.BandejaEntrada.TIPOPRODUCTO;
CAN_WR_Buffer[4]    := PT15_FB.BandejaEntrada.ESTADO;

CAN_WR_Done := TRUE;
```

Figura 67 : Bloque de programa para la comunicación con el buffer de la línea CAN

6. VALIDACIONES

Los programas deben estar sujetos a pruebas que validen su correcto funcionamiento o en su caso detectar los defectos que no se han tenido en cuenta en el diseño de la programación.

Existen dos tipos de validaciones:

- **Pruebas operacionales:** Se realizan para verificar el funcionamiento de un sistema o sub-sistema que se requieren mínimamente (Entradas/salidas, bloques de funciones, etc.).

- **Pruebas funcionales**: Verificación de la capacidad del producto, funcionalidades que la aplicación debe tener y los datos que deberá manipular, así como límites y conexiones con otras aplicaciones (Comportamiento global de la celda según estado de las bandejas).

Se llevará a cabo una validación operacional y funcional por cada programa.

Las pruebas referentes a la línea CAN se han podido realizar físicamente en la celda automatizada del laboratorio sin necesidad de usar el simulador.

6.1 Pruebas operacionales

6.1.1 Línea CAN

Se procede a probar el funcionamiento del FB "Plataforma" mediante la introducción de bandejas en el primer retenedor "DIR03".

Se observa que una vez que el sensor detecta la presencia de la bandeja, el retenedor actúa dejándola pasar hacia el retenedor "DIR04" teniendo este un comportamiento idéntico avanzando hacia la plataforma "PT04".

Para comprobar el feedback entre los estados de las plataformas y retenedores, se procede a llenar toda la línea con bandejas para cerciorar de que cuando cada una de las plataformas y retenedores de destino no estén en estado de reposo, la anterior se quede en estado de petición teniendo un resultado satisfactorio.

Para validar el envío de datos entre los FB, una vez se introduce la bandeja en el primer retenedor "DIR03" se detecta la generación del ID (identificación) que es un número mayor y el tipo de producto es aleatorio. Mediante el modo de supervisión online en el software de PLC de las instancias de FB de los retenedor "DIR03" y "DIR04" se comprueba como una vez la bandeja se detiene en un instante, este pasa a tener los valores de ID, tipo de producto y estado del retenedor anterior.

6.1.2 Línea Modbus

Se procede a validar el funcionamiento de la FB "Proceso" que se aplica a las plataformas donde se simulan las estaciones de trabajo mediante modo simulación.

Una vez cargado el programa en el simulador se abre la instancia del "PR_ET1".

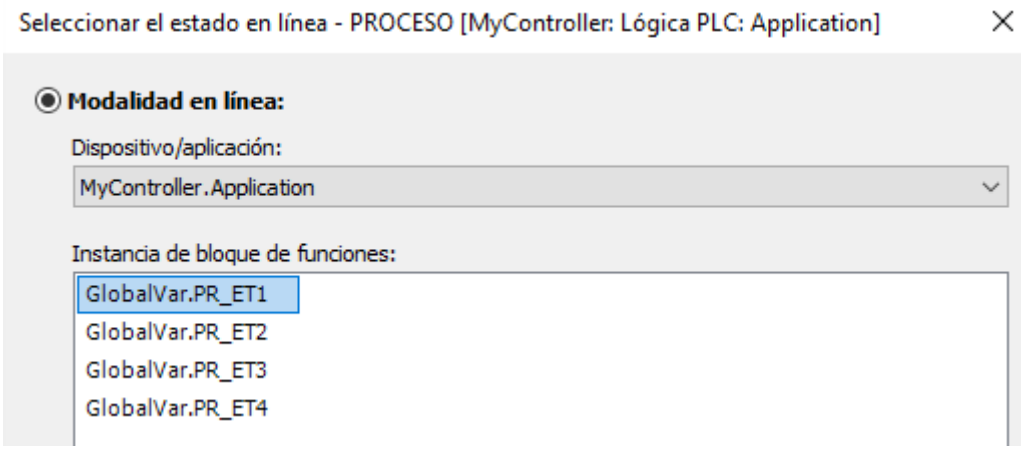


Figura 68 : Selección instancias de FB en línea

Para que la instancia tenga efecto se debe modificar la variable "TipoProducto" de la estructura "BandejaEntrada" del bloque correspondiente.

Se simula que la bandeja recibe un producto tipo 1 con un estado 0 (sin procesar):

Expresión	Tipo de datos	Valor
MyController.Application.ESTACIONES_TRABAJO.PT09_FB.BandejaEntrada	PRODUCTO	
IDPRODUCTO	INT	100
TIPOPRODUCTO	INT	1
ESTADO	INT	0

Figura 69 : Datos del producto antes de pasar por el proceso

Una vez introducido los valores mencionados, se procede a forzar el bit del sensor de presencia del bloque para cumplir la primera condición de bloqueo y el inicio de procesamiento.

Se repite este proceso varias veces para comprobar que se cumple el tiempo establecido antes de realizar el cambio de estado y la probabilidad de que el resultado no sea satisfactorio no lo cambie.

Expresión	Tipo de datos	Valor
MyController.Application.ESTACIONES_TRABAJO.PT09_FB.BandejaEntrada	PRODUCTO	
IDPRODUCTO	INT	100
TIPOPRODUCTO	INT	1
ESTADO	INT	1

Figura 70 : Datos del producto después de pasar por el proceso

Se observa que el funcionamiento del bloque de funciones es el esperado.

6.2 Pruebas funcionales

6.2.1 Línea CAN

Se comprueba el funcionamiento de varias plataformas según el estado del producto de cada bandeja entrante en cada plataforma.

El resultado es satisfactorio. Algunos ejemplos son:

Plataforma "PT05":

- Desvía hacía la dirección alternativa si la principal tiene una bandeja en el destino.
- Direcciona hacía el retenedor "DIR05" si el estado del producto es el equivalente a rechazo.
- Se detiene si el estado del producto está correctamente procesado.

Plataforma "PT04":

- No detiene la bandeja si el estado del producto es 0.
- Realiza el control de calidad si el resultado del producto es equivalente a acabado.

6.2.2 Línea Modbus

Con una pantalla creada mediante las herramientas que facilita SoMachine se lleva a cabo la validación del orden correcto de funcionamiento del traslado de bandejas y sus datos, así como el desvío correspondiente según el tipo de producto y el resultado después de pasar por una estación de trabajo.

En primer lugar, se introducen los valores en "BandejaEntrada" de la plataforma "PT08" simulando la llegada de una bandeja procedente de la línea CAN. Forzando el valor de entrada del sensor, se procede a acusar la confirmación del envío de datos para poder pasar al estado de avance:

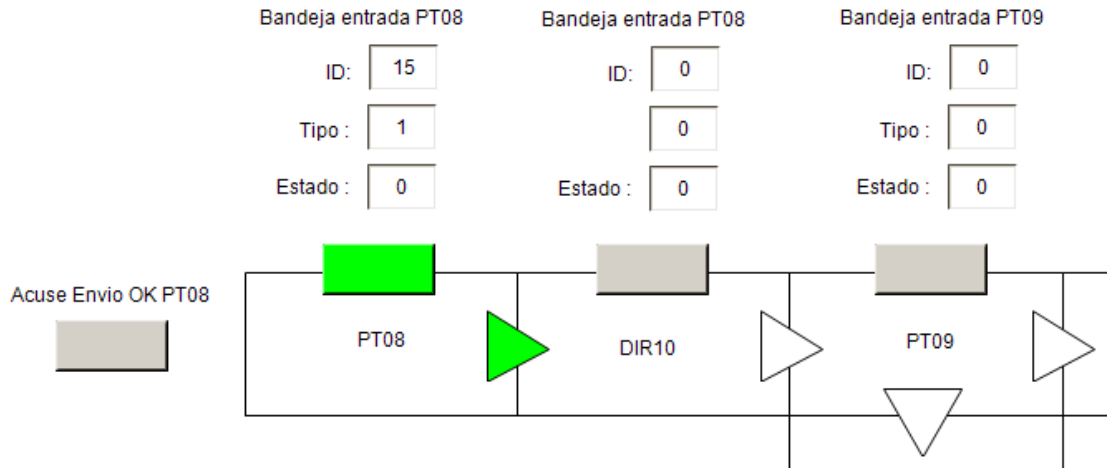


Figura 71 : Pantalla de simulación de la línea Modbus (1)

Simulando la activación y desactivación de los detectores de presencia se puede observar cómo cada plataforma funciona como se esperaba.

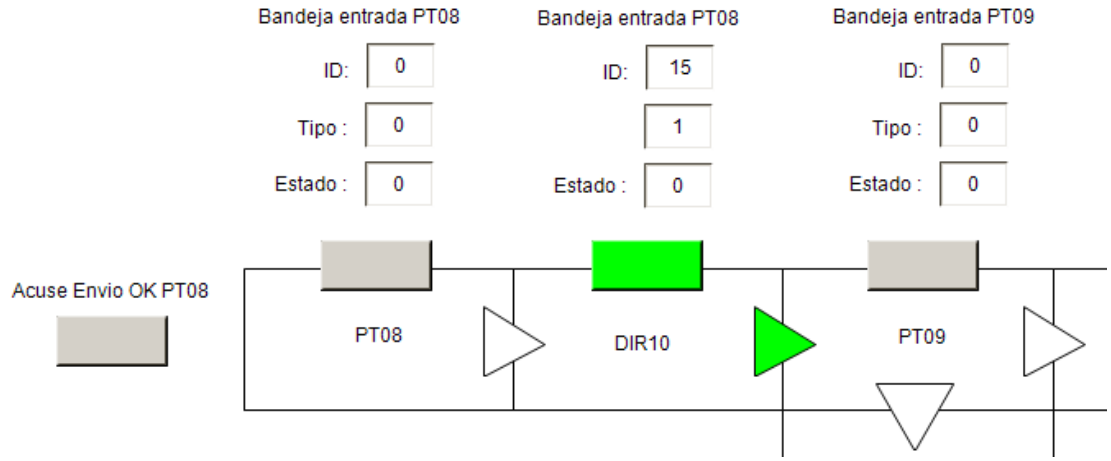


Figura 72 : Pantalla de simulación de la línea Modbus (2)

Una vez que la bandeja llega a la plataforma donde se aloja la estación de trabajo se comprueba el bloqueo de avance.

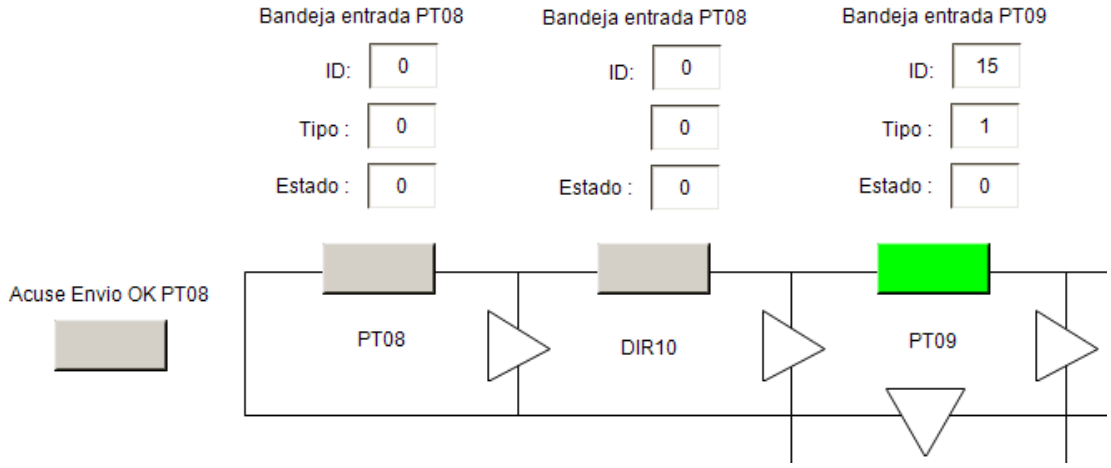


Figura 73 : Pantalla de simulación de la línea Modbus (3)

Una vez pasado el tiempo de procesado establecido, el estado del producto se ve afectado y se realiza el desbloqueo del avance hacia la dirección correspondiente para la próxima estación de trabajo.

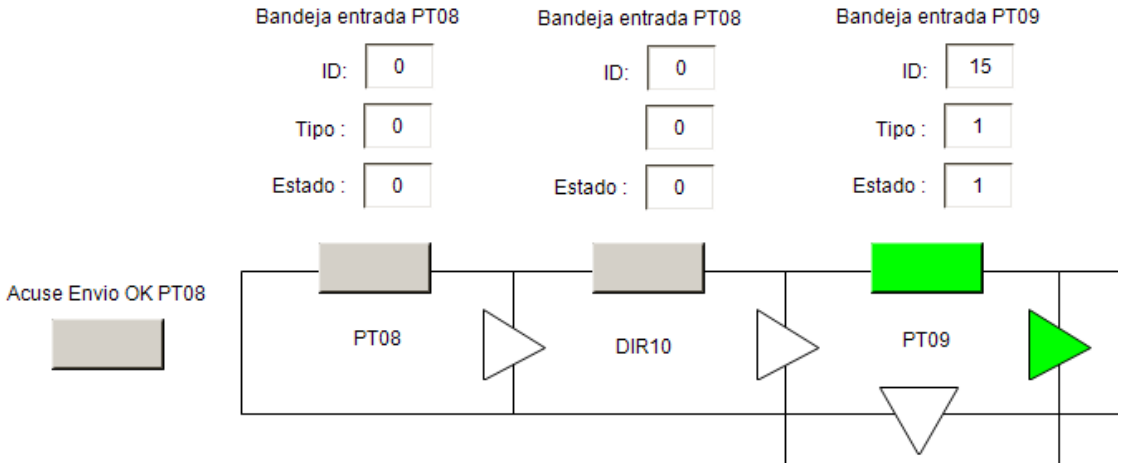


Figura 74 : Pantalla de simulación de la línea Modbus (4):

Se repite el proceso cambiando el valor del tipo de producto para comprobar el desvío inmediato hacia la dirección correspondiente y la no aplicación de proceso.

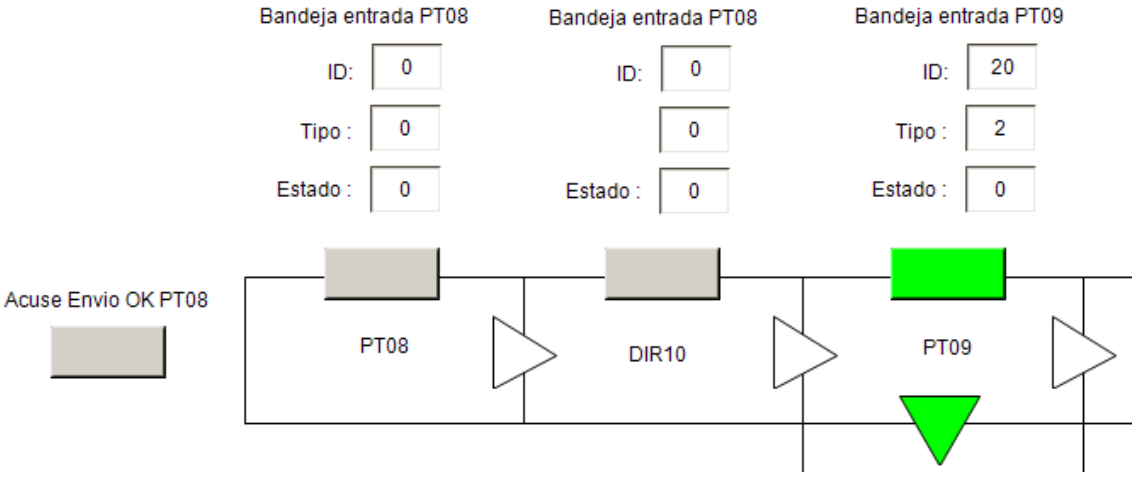


Figura 75 : Pantalla de simulación de la línea Modbus (5):

7. Conclusiones

Se debe de tener en cuenta el orden de los procedimientos a seguir para conseguir una optimización en el transcurso del proyecto y un buen resultado final. En primer lugar, se debe tener claro el concepto del caso de estudio, definir el comportamiento y funcionalidades que se quiere realizar para pasar a la siguiente etapa. Es muy importante previo análisis para conocer el entorno donde se va a trabajar, tanto instalaciones eléctricas y neumáticas como los softwares de programación con el fin de tener todo conocimiento posible al implementar las soluciones.

La previsión para realizar el testeo con anticipación de los programas tiene un impacto elevado a posibles modificaciones ya sean por errores o mejoras de estos, pues un testeo tardío puede llevar un atraso en las previsiones, no poder pasar de etapa o en su peor caso no poder completar el estudio.

En este caso, no se ha podido realizar las validaciones del programa de la línea Modbus en las instalaciones ni su correspondiente comunicación entre controladores obligando a crear un entorno de simulación con herramientas limitadas ergo la falta de visualización del comportamiento real.

Cabe destacar que al usar un nuevo entorno de programación el cual requiere de documentación y de inversión de tiempo para poder familiarizarse con este, se adquiere experiencia aún fuera del contexto del objetivo lo cual induce a una mejora profesional y del know-how en el sector.

8. Presupuesto

Concepto	Cantidad [h]	Precio [€/h]	Total [€]
Diseño	10,00	42,00	420,00
Desarrollo	150,00	42,00	6.300,00
Validación	20,00	42,00	840,00
Memoria	30,00	42,00	1.260,00
			8.820,00
		Costes indirectos …… 10%	882,00
			9.702,00
		IVA…… 21%	2.037,42
		TOTAL PRESUPUESTO	**11.739,42**

Tabla 2 : Coste del proyecto

9. Bibliografia

[1] srcsl, Que es un PLC. [En línea] [01/2022]. Disponible en:

https://srcsl.com/que-es-un-plc/

[2] cursoaula21, Funcionamiento del PLC. [En línea] [01/2022]. Disponible en:

https://cursosaula21.com/que-es-un-automata-programable-o-plc-y-como-funciona/

[3] industriasgsl, Funcionamiento del PLC y el ciclo SCAN. [En línea] [01/2022]. Disponible en:

https://industriasgsl.com/blog/post/que-es-un-plc-y-como-funciona

[4] controlreal, El ciclo SCAN. [En línea] [01/2022]. Disponible en:

https://controlreal.com/es/memoria-y-ciclo-de-escan/

[5] infoplc, La pirámide CIM. [En línea] [01/2022]. Disponible en:

http://infoplc.net/files/documentacion/comunicaciones/infoPLC_net_Historia_Comunicaciones_Industriales.pdf

[6] automatismosmundo, Los lenguajes de programación de PLC. [En línea] [01/2022]. Disponible en:

https://automatismosmundo.com/los-lenguajes-de-programacion-de-plc/

[7] wikipedia, Bus CAN. [En línea] [01/2022]. Disponible en:

https://es.wikipedia.org/wiki/Bus_CAN

[8] ingenieriaymecanicaautomotriz, Funcionamiento del CANBus. [En línea] [01/2022]. Disponible en:

https://www.ingenieriaymecanicaautomotriz.com/que-es-el-can-bus-y-como-funciona/

[9] infoplc, Origen del CANBus. [En línea] [01/2022]. Disponible en:

http://www.infoplc.net/files/documentacion/comunicaciones/infoPLC_net_Historia_Comunicaciones_Industriales.pdf

[10] profesionalreview, El protocolo IP/TCP. [En línea] [01/2022]. Disponible en:

https://www.profesionalreview.com/2020/03/21/protocolo-tcp-ip/

[11] cursoaula21, Modbus: Que es y cómo funciona. [En línea] [01/2022]. Disponible en:

https://www.cursosaula21.com/modbus-que-es-y-como-funciona/

[12] Schneider Electric, EcoStruxure Control Expert - Lenguajes y estructura del programa, Manual de referencia. [En línea] [01/2022]. Disponible en:

https://www.se.com/es/es/download/document/35006147K01000/

[13] Schneider Electric, SoMachine, Guía de programación. [En línea] [01/2022]. Disponible en:

https://www.se.com/es/es/download/document/EIO0000000071/