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

INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TRABAJO DE FINAL DE GRADO

Morales Jordan, Víctor

ESTUDIO DE LAS ESTAPAS DE AUTOMATIZACIÓN DE UN PROCESO INDUSTRIAL

Director del TFG: Delgado Prieto, Miguel
Co- Director del TFG: Fernández Sobrino, Ángel
Convocatoria: Junio, 2020

I declare that,

the work in this Degree Thesis (*choose one*) is completely my own work,

no part of this Degree Thesis (*choose one*) is taken from other people's work without giving them credit,

all references have been clearly cited,

I'm authorised to make use of the company's related information I'm providing in this document (*select when it applies*).

I understand that an infringement of this declaration leaves me subject to the foreseen disciplinary actions by *The Universitat Politècnica de Catalunya - BarcelonaTECH*.

Víctor Morales Jordan		30/06/2020
Student Name	Signature	Date

Title of the Thesis : ESTUDIO DE LAS ESTAPAS DE AUTOMATIZACIÓN DE UN PROCESO INDUSTRIAL.

Índice

1. Introducción 3

1.1. Resumen 3

1.2. Objetivo 4

1.3. Alcance 5

1.4. Justificación 6

2. Consideraciones previas 8

2.1. Protocolos de comunicación utilizados 8

2.1.1. Protocolo Profibus 8

2.1.2. Protocolo Modbus TCP/IP 9

2.2. Lenguajes de programación utilizados 10

2.2.1. Programación con Texto Estructurado (ST) 10

2.2.2. Diagrama de Contactos (LD) 11

2.3. Plataforma software de programación 12

2.3.1. Bloque de Función (FB) 12

2.3.2. Estructura o Tipo de Dato (STRUCT) 13

3. Análisis. 15

3.1. Deficiencias 16

3.1.1. Secciones del programa 16

3.1.2. Variables globales del programa 18

3.1.2.1. GlovalVar 21

3.1.2.2. In 21

3.1.2.3. Out 22

3.1.2.4. Sensores 22

3.1.2.5. Variables_Visualizacion 23

3.2. Limitaciones 24

3.2.1. Creación de Retenedores 24

3.2.2. Creación de Plataformas 27

3.2.3. Creación de Estaciones de Trabajo. 31

3.3. Carencias 40

3.3.1. VISUALIZACIÓN RETENEDORES..... 43

3.3.2. VISUALIZACIÓN PLATAFORMAS..... 44

3.4. Elementos obsoletos. 46

4. Validación 48

5. Conclusiones 49

Sumario de Figuras

Figura 1- *Esquema del proceso* 7

Figura 2- Esquema ejemplo de Profibus 11

Figura 3- Esquema ejemplo de Modbus 12

Figura 4- Ejemplo programación con Texto Estructurado (ST) 13

Figura 5- *Ejemplo programación con Diagrama de Contactos (LD)* 14

Figura 6- *Ejemplo Bloque de Función (FB)* 15

Figura *7- Aplicación del Bloque de Función (FB)* 16

Figura 8- *Ejemplo Estructura o Tipo de Dato (STRUCT)* 16

Figura 9- Ejemplo definición Estructura o Tipo de Dato (STRUCT) 16

Figura 10 - Árbol del proyecto antiguo (secciones del programa) 18

Figura 11- Árbol del proyecto nuevo (secciones del programa) 19

Figura 12- Arbol del proyecto antiguo (tabla de variables) 20

Figura 13- Tabla de variables (proyecto antiguo) - 1 21

Figura 14- Tabla de variables (proyecto antiguo) - 2 21

Figura 15- Arbol del proyecto nuevo (tabla de variables) 22

Figura 16- Tabla de variables globales 23

Figura 17- Tabla variables IN 23

Figura 18- Tabla variables OUT 24

Figura 19- Tabla variables de Sensores 24

Figura 20- Variables visualización 25

Figura 21- Variables de estado para un retenedor - 1 26

Figura 22- Variables de estado para un retenedor - 2 26

Figura 23- Acción de un retenedor - 1 27

Figura 24- Acción de un retenedor - 2 27

Figura 25- Acción de un retenedor - 3 28

Figura 26- Programación FB Retenedores. 28

Figura 27- Ejemplo uso FB para Retenedores 29

Figura 28- Variables de estado para una plataforma - 1 29

Figura 29- Variables de estado para una plataforma - 2 30

Figura 30- Lineas de programación para una plataforma - 1 30

Figura 31- Lineas de programación para una plataforma - 2 31

Figura 32 - Variables FB para plataformas. 31

Figura 33 - Lineas de programación FB para plataformas. 32
Figura 33 - Uso FB para plataformas - 2 33
Figura 34 - Variables Estaciones de Trabajo - 1 33
Figura 35 - Variables Estaciones de Trabajo - 2 33
Figura 36 - Variables Estaciones de Trabajo - 3 34
Figura 37 - Variables Estructura Estaciones de Trabajo 34
Figura 38 - Variables Estructura SEGUIMIENTO_PRODUCTO 35
Figura 39 - Variables Estructura TIEMPO 35
Figura 40 - Programación estación de trabajo proyecto antiguo 36
Figura 41 - Programación estación de trabajo proyecto nuevo 36
Figura 42 - Aplicación FB estación de trabajo 37
Figura 43 - Lineas programación tiempo Estación de trabajo 37
Figura 44 - Lineas programación FB Cronometro Estación de trabajo 38
Figura 45 - Aplicación FB Cronometro Estación de trabajo 38
Figura 46 - Programación gestión de productos Proyecto antiguo - 1 38
Figura 47 - Programación gestión de productos Proyecto antiguo - 2 39
Figura 48 - Programación FB gestión de productos Proyecto nuevo 39
Figura 49 - Aplicación FB gestión de productos Proyecto nuevo 40
Figura 50 - Visualización del proceso 41
Figura 51 - Esquema del proceso 41
Figura 52 - Tipo de dato visualización Retenedores y Plataformas. 42
Figura 53 - Leyenda retenedores y plataformas 43
Figura 54 - Leyenda retenedores 43
Figura 55 - Programación FB para la visualización de los estados de los Retenedores 44
Figura 56 - Aplicación FB para la visualización de los estados de los Retenedores 44
Figura 57 - Programación FB para la visualización de los estados de las Plataformas 45
Figura 58 - Aplicación FB para la visualización de los estados de las Plataformas 45
Figura 59 - Isla numero 1. Cabecera NIP2311 47
Figura 60 - Isla numero 2. Cabecera NIP2311 48
Figura 61 - Isla numero 3. Cabecera NIP2311 48
Figura 62 - Temporizador simula entrada de Bandeja. 49
Figura 63 - Temporizadores simulan movimiento de Bandeja. 50

1. Introducción

1.1. Resumen

El objetivo de este proyecto es el diseño y programación de un automatismo sobre un proceso industrial. El punto de partida es un laboratorio de automatización operativo ubicado en el aula Schneider Electric, en la Escuela Superior de Ingenierías Industrial, Aeroespacial y Audiovisual de Terrassa, de la Universidad Politécnica de Terrassa (UPC).

Concretamente, el proceso está constituido por unas cintas transportadoras, donde se desplazan unas bandejas que emulan la distribución de diferentes tipos de productos. Esta estructura de cintas dispone de unos elementos fundamentales, llamados retenedores o plataformas, que a través de su control, permiten la gestión de los circuitos que deben recorrer las bandejas. En tres de estos retenedores se encuentran las estaciones de trabajo, que, de la misma manera, emulan procesos de manipulación sobre los productos que se transportan en las bandejas.

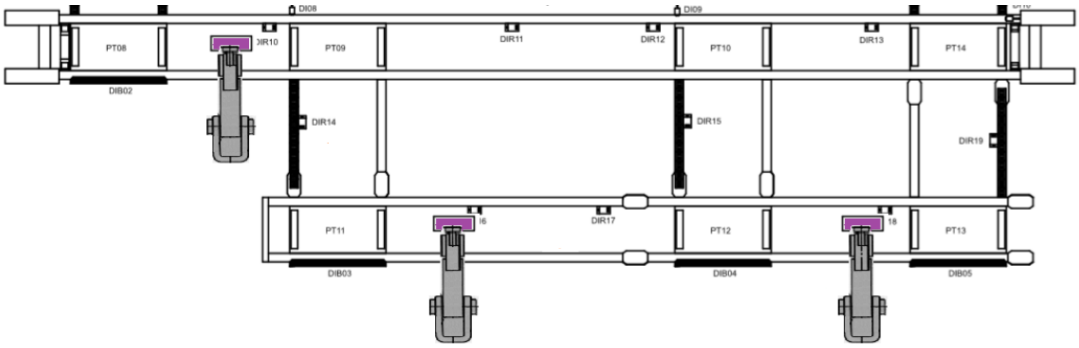


Figura 1- Esquema del proceso

El reto principal del proyecto reside en el análisis de la operación global del proceso, la revisión de los programas existentes, la identificación de errores de operación y limitaciones de funcionamiento, la propuesta de mejoras, la actualización del automatismo programado, la integración del código realizado en un nuevo autómata y, finalmente, la validación del conjunto.

1.2. Objetivo

El objetivo principal es el diseño e implementación de un automatismo programado sobre un proceso industrial.

El entorno de aplicación es un laboratorio de automatización donde se encuentra un proceso industrial basado en la distribución de productos a través de cintas transportadoras.

En este caso nos encontramos con una máquina, con muy buena proyección a la industria 4.0 pero que llega a tener algunas deficiencias de código mejorables.

Como es común en el sector industrial, los procedimientos de mantenimiento, mejora y actualización de los procesos productivos tienen una importancia fundamental para mantener los niveles de eficiencia de la maquinaria y sistemas asociados. En este sentido, el proyecto tiene una fuerte componente de ingeniería industrial, donde se considera la comprensión de un proceso industrial completo (i.e. desde los actuadores hasta el algoritmo de control global). Así, la actualización y mejora de las prestaciones del automatismo representan el objetivo principal de este proyecto, el cual engloba toda una serie subobjetivos, como son: el análisis de la operación global del proceso, la revisión de los programas existentes, la identificación de errores de operación y limitaciones de funcionamiento, la propuesta de mejoras, la actualización del automatismo programado, la integración del código realizado en un nuevo autómata y, finalmente, la validación del conjunto. Concretamente, y en línea con la estructura de este documento, los objetivos específicos son:

1. Reducción del tiempo de entendimiento del programa:
    a. Uso de Listas de variables.
    b. Uso de Carpetas Clasificadoras.

2. Reducción del uso de memoria de la CPU:
    a. Uso de Bloques de Función(FB).
    b. Uso de Estructuras de Datos (STRUCT).

3. Mejora del acabado del proyecto:
    a. Visualización para la simulación del proceso.

4. Mejora del programa para futuras ampliaciones.

5. Actualización del Hardware.

6. Actualización del Software.

1.3. Alcance

El punto de partida del proyecto es un laboratorio de automatización operativo ubicado en el aula Schneider Electric, en la Escuela Superior de Ingenierías Industrial, Aeroespacial y Audiovisual de Terrassa, de la Universidad Politécnica de Terrassa (UPC). En este sentido, se dispone de una infraestructura industrial operativa y un automatismo programado base para su operación.

Para alcanzar los objetivos propuestos, el alcance del proyecto considera:

- Desde el punto de vista del hardware.
  - · Análisis de las mejoras del nuevo hardware en comparación con el antiguo.
  - · Definición de la arquitectura del hardware.

- Desde el punto de vista software.
  - · Análisis de las mejoras del nuevo software en comparación con el antiguo.
  - · Familiarización con el nuevo software.
  - · Análisis del código antiguo.
  - · Definición de la arquitectura del programa.

- Desde el punto de vista de la interfaz de visualización.
  - · Análisis de los elementos necesarios para la visualización.

1.4. Justificación

Desde hace más de un año decidí enfocar mi vida laboral a la programación de autómatas y es por eso mismo que el tema de este trabajo es prácticamente el proceso que cualquier empresa seguiría para mejorar uno de sus productos. Para una empresa, una mejora de sus productos se traduce en mayores ganancias, no solo en lo que al producto se refiere, si no en la reducción de los tiempos de trabajo.

En la actualidad estoy trabajando en una empresa como ingeniero de control, y me dedico básicamente a programar todo aquel elemento de cualquier proceso industrial que tenga que ver con el control del mismo, como pueden ser PLC, SCADAS, HMI... etc.

Este proyecto, por tanto, me abre la posibilidad de aprender y profundizar en otros entornos de programación como es el de Schneider, el cual solo había podido probar en un par de asignaturas en la universidad.

La realización de este proyecto me permite también crecer en creatividad a la hora de programar. El hecho de proponer las mejoras para un código que otra persona ha programado ayuda a visualizar desde otro punto de vista, y mejorar la forma en la que tú mismo programas.

A rasgos globales, se podría decir que el motivo principal de la realización de este proyecto es mi crecimiento en el mundo de la programación de autómatas y puesta en práctica de nuevas técnicas de programación.

2. Consideraciones previas

2.1. Protocolos de comunicación utilizados

2.1.1. Protocolo Profibus

Protocolo utilizado en el proyecto anterior.

El protocolo Profibus (Process Field Bus) es un protocolo abierto de comunicación que se encarga de la transmisión de datos entre los sensores y el sistema de control. Fue desarrollado específicamente para la implementación en industrias de automatización. Es uno de los protocolos más utilizados, y que está en constante evolución.

En este protocolo, los dispositivos del sistema se conectan a una línea central y se dividen en dos tipos, los maestros y los esclavos. Los dispositivos maestros pueden enviar y solicitar datos a otras estaciones. Mientras que los esclavos sólo pueden enviar datos cuando un maestro lo solicite.

Todo esto regido por el método del token o testigo, el cual asigna los derechos de acceso del bus dentro de un intervalo de tiempo definido. El token otorga, a cada estación maestra, la capacidad de transmitir información al bus.

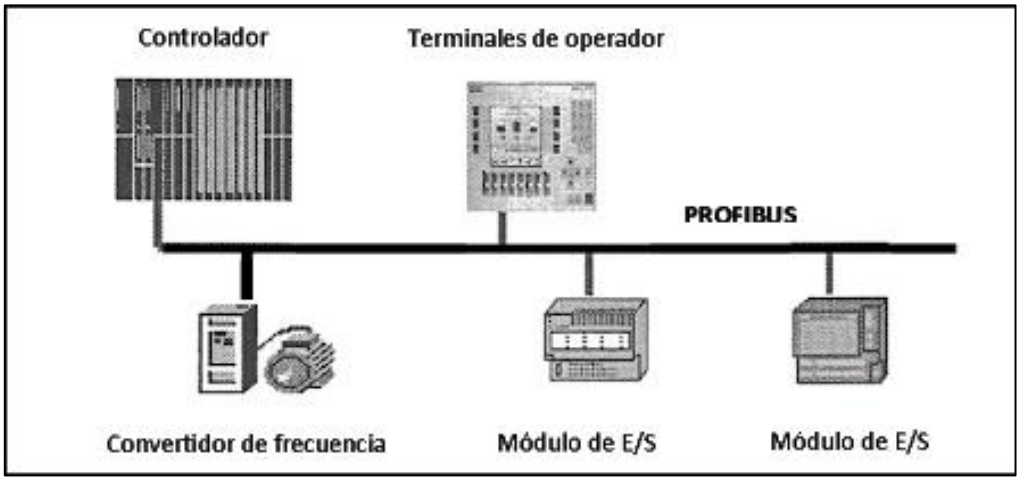


Figura 2- Esquema ejemplo de Profibus

2.1.2. Protocolo Modbus TCP/IP

El protocolo Modbus TCP/IP es un protocolo basado en la arquitectura cliente/servidor conectada en una red Ethernet. Es un protocolo público y gratuito, fácil de interpretar y que requiere poco desarrollo.

Este protocolo permite el control de una red de dispositivos y comunicar los datos a un ordenador. Cada dispositivo de la red Modbus tiene una dirección única. Cualquier dispositivo puede enviar órdenes Modbus, aunque lo más normal es que sólo un dispositivo maestro pueda hacerlo.

Cada orden Modbus contiene la dirección del dispositivo destinatario. Aun que todos los dispositivos reciben la orden, solo es el destinatario el que la ejecuta.

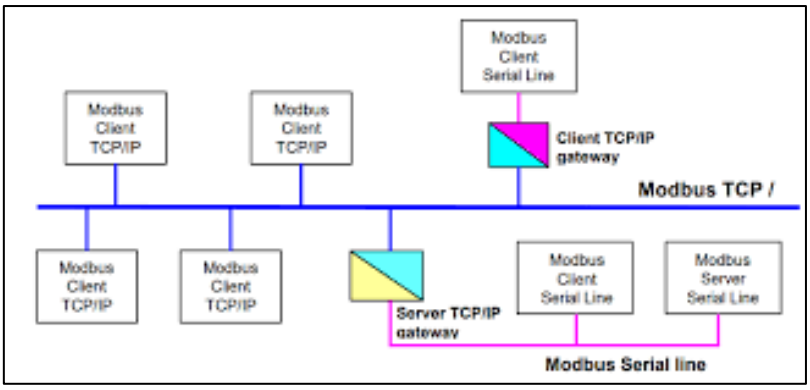


Figura 3- Esquema ejemplo de Modbus

2.2. Lenguajes de programación utilizados

Aun que el software de Schneider permite utilizar diferentes tipos de lenguajes de programación, aquí solo se expondrán los que se han utilizado en este proyecto.

2.2.1. Programación con Texto Estructurado (ST)

La programación de Texto Estructurado es un lenguaje de alto nivel, es decir, la que se adapta al entendimiento de las personas, y no de las máquinas. Este lenguaje nos permite crear algoritmos complejos a partir de instrucciones condicionales, instrucciones repetitivas, funciones... etc.

```
IF Plat.REST AND NOT (Plat.MOV1 OR Plat.MOV2) AND NOT Plat.READY THEN
    PlatVis.Naranja := FALSE;
    PlatVis.Azul := FALSE;
    PlatVis.Verde := TRUE;
    PlatVis.NoNaranja := TRUE;
    PlatVis.NoAzul := TRUE;
    PlatVis.NoVerde := FALSE;
END_IF

IF (Plat.MOV1 OR Plat.MOV2) AND NOT Plat.READY AND NOT Plat.REST THEN
    PlatVis.Naranja := FALSE;
    PlatVis.Azul := TRUE;
    PlatVis.Verde := FALSE;
    PlatVis.NoNaranja := TRUE;
    PlatVis.NoAzul := FALSE;
    PlatVis.NoVerde := TRUE;
END_IF

IF Plat.READY AND NOT Plat.REST AND NOT (Plat.MOV1 OR Plat.MOV2) THEN
    PlatVis.Naranja := TRUE;
    PlatVis.Azul := FALSE;
    PlatVis.Verde := FALSE;
    PlatVis.NoNaranja := FALSE;
    PlatVis.NoAzul := TRUE;
    PlatVis.NoVerde := TRUE;

END_IF
```

Figura 4- Ejemplo programación con Texto Estructurado (ST)

2.2.2. Diagrama de Contactos (LD)

Este lenguaje de programación utiliza elementos gráficos que simulan el esquema eléctrico de control.

Un programa hecho con este lenguaje, en comparación con el diagrama eléctrico, se realiza de forma secuencial y en un tiempo específico que se llama Ciclo de SCAN.

Los elementos mas comunes son los contactos (variables booleanas), temporizadores, contadores...

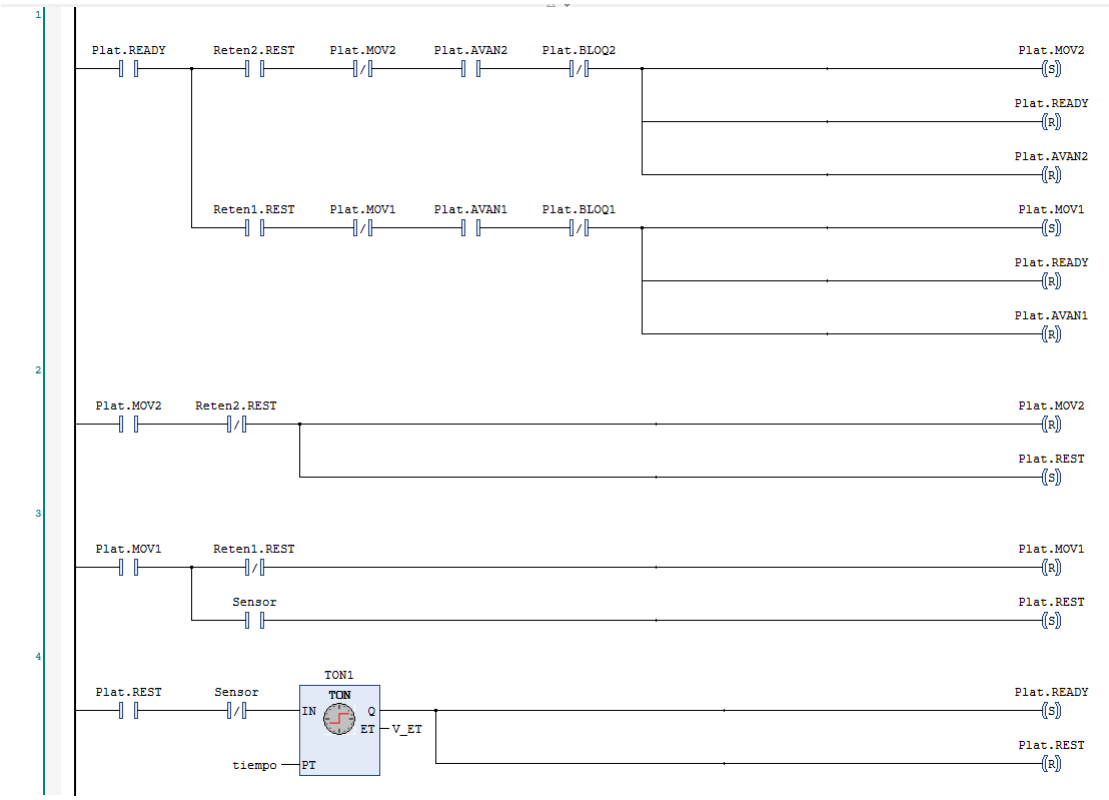


Figura 5- Ejemplo programación con Diagrama de Contactos (LD)

2.3. Plataforma software de programación

En este proyecto se van a utilizar elementos de programación que, necesariamente para el correcto entendimiento del proyecto, se deben explicar antes de entrar en materia.

2.3.1. Bloque de Función (FB)

Un bloque de función es un elemento que tiene asociada unas líneas de programación que se ejecutan cada vez que el ciclo de SCAN lo requiere. Así mismo, un FB tiene una base de datos exclusiva para él, el cual contiene unas variables estáticas.

Para el uso del bloque, se requiere unas variables globales de entrada y otras de salida las cuales se asocian a las variables estáticas.

Las variables de entrada serán aquellas de las que el bloque necesite información para que efectúe su función. Mientras que las de salidas, son aquellas que, dentro del bloque, se modifica su estado o información. En algunos casos, las variables de entrada serán a la vez, de salida. En ese caso, las variables se crearán como variables de entrada y salida.

Un FB también podrá tener variables locales. Variables de las que no se requiera de su información fuera del bloque.

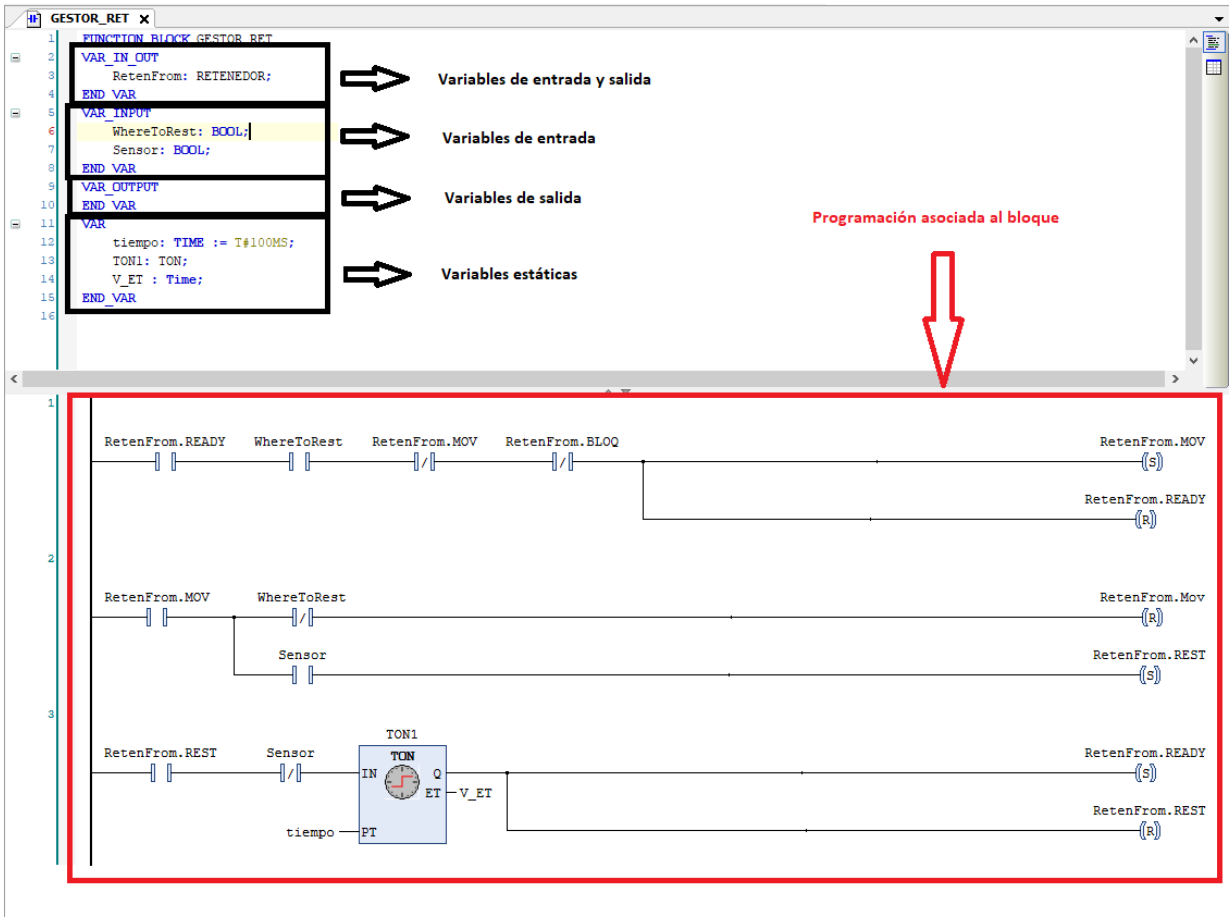


Figura 6- Ejemplo Bloque de Función (FB)

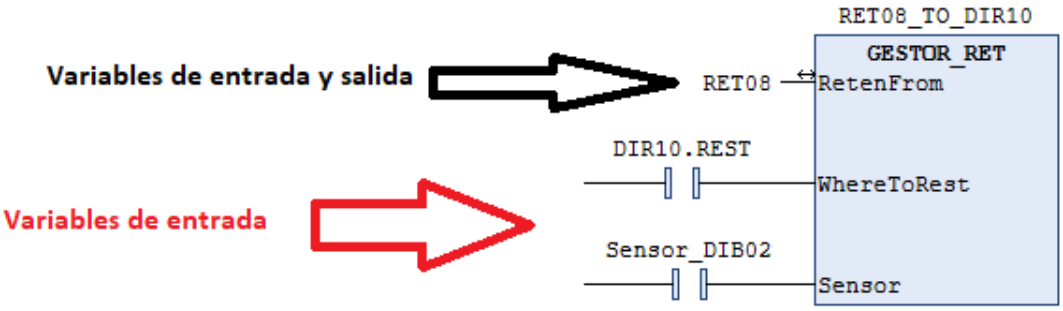


Figura 7- Aplicación del Bloque de Función (FB)

2.3.2. Estructura o Tipo de Dato (STRUCT)

Una estructura o Tipo de Dato es un conjunto de variables que forman una misma variable. Es muy útil cuando tienes muchos elementos del mismo tipo o elementos que tienen las mismas características.

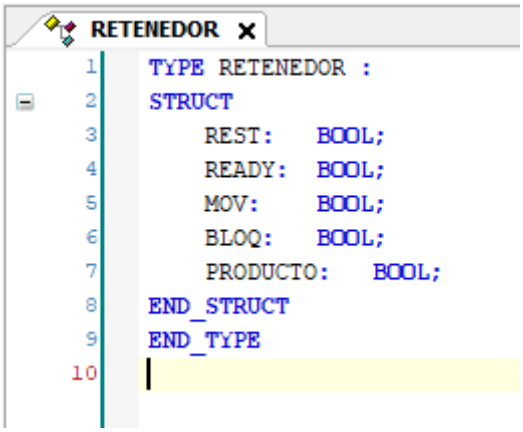


Figura 8- Ejemplo Estructura o Tipo de Dato (STRUCT)

En el ejemplo de la imagen anterior, podemos ver el tipo de dato "Retenedor", el cual engloba 5 variables del tipo Booleano.

```
DIR10: RETENEDOR;
DIR11: RETENEDOR;
DIR12: RETENEDOR;
```

Figura 9- Ejemplo definición Estructura o Tipo de Dato (STRUCT)

Al haber creado un Tipo de Dato, solo deberemos crear una variable tipo "Retenedor". Si no se hubiera creado el Tipo de Dato, se hubiesen requerido la creación de 5 variables booleanas.

3. Análisis.

El primer paso para una buena reconstrucción y mejora de nuestro proceso es un análisis de tres aspectos: primero de deficiencias, segundo de las limitaciones y, por último, de las carencias que tiene el proyecto base sobre el que se basa este trabajo.

Así, en primer lugar, como deficiencias, se considera la identificación de mala praxis en la programación, como podría ser la repetición de partes idénticas del código o de declaraciones de variables. El objetivo en esta parte es, por lo tanto, hacer un código estructurado, ordenado e intuitivo, de forma que, en eventuales actualizaciones del programa, el personal técnico que deba modificarlo encuentre un fácil entendimiento y, por lo tanto, lleve a una menor posibilidad de errores.

En segundo lugar, como limitaciones, se considera la falta de estructura y modularidad del código para afrontar posibles expansiones, reducciones o simplemente modificaciones en el proceso de automatización que se realice sobre las máquinas y actuadores. Es decir, en caso de modificaciones físicas sobre el proceso que conlleven una revisión del programa asociado. En este caso, el objetivo considerado es transformar el código en un programa que, en caso de añadir nuevos elementos físicos en la máquina automatizada, se pueda hacer de una manera sencilla, modular, y sin tener que modificar la estructura del código base.

Por último, en tercer lugar, como carencias se incluyen aquellas partes de programa que son necesarias para el entendimiento del proyecto pero que no están implementados. El objetivo en este sentido es identificar y crear esas partes siempre teniendo en cuenta los dos puntos anteriores.

En cuanto a la estructura del Hardware del proceso en cuestión, se podría añadir en limitaciones, ya que, como veremos más adelante, es un elemento que se ha quedado obsoleto al largo de los años, y es posible una gran mejora en cuanto a rendimiento por parte de las islas de hardware.

Por tanto, como elementos obsoletos, nos referiremos a las partes del hardware las cuales se han quedado atrás en cuanto a rendimiento, y sería necesario o muy recomendable el reemplazo por otros mejores.

3.1. Deficiencias

Una de las tareas más complicadas que puede encontrarse un programador cuando se enfrenta por primera vez un código hecho por otro ingeniero, es entenderlo. Una buena estructuración del código del programa puede ahorrar muchas horas de trabajo invertidas en solo entender el código de otra persona.

Nuestro objetivo, como se ha comentado anteriormente, es crear un código estructurado, ordenado e intuitivo. Donde a simple vista se puedan reconocer las diferentes secciones que componen el programa y, en consecuencia, hacerse una idea de cómo está estructurado el código en cuestión. El punto de partida para lograr este objetivo empieza por el árbol del proyecto, donde se encuentran, entre otros elementos, la lista de las variables y las secciones del programa.

3.1.1. Secciones del programa

Fijándonos en el árbol del proyecto base nos podemos dar cuenta que no hay una estructuración clara en lo que al código se refiere.

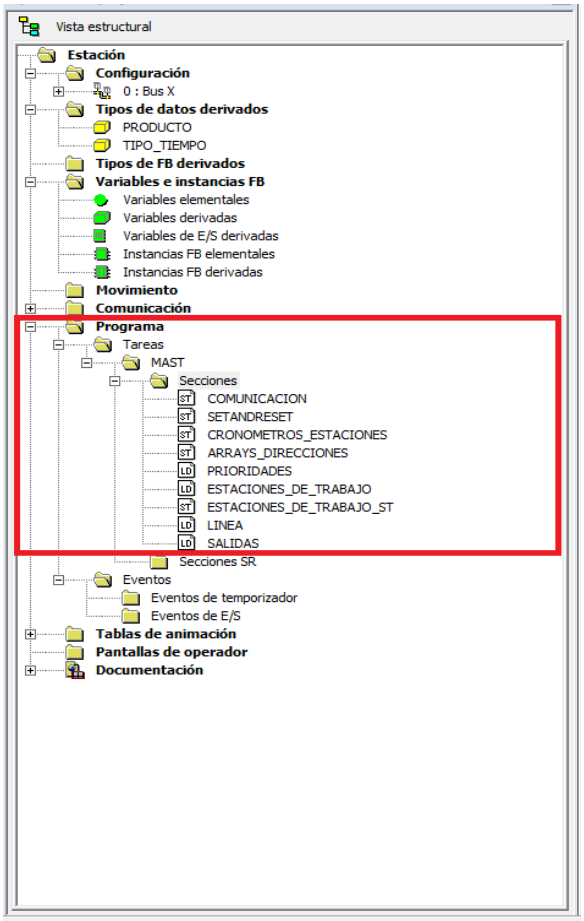


Figura 10 - Árbol del proyecto antiguo (secciones del programa)

En el caso del nuevo programa, se ha decidido separarlo en tres partes bien diferenciadas:

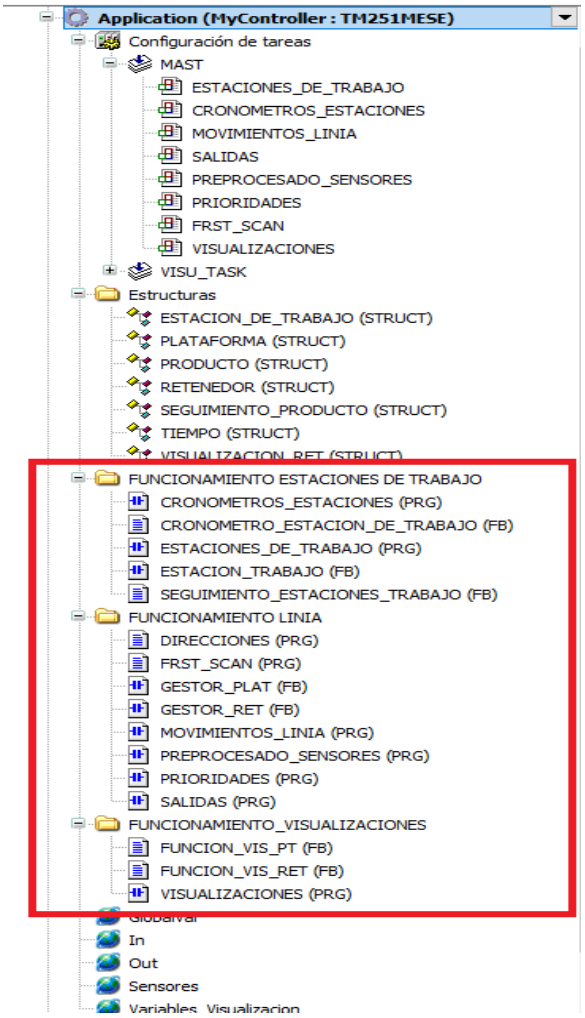


Figura 11- Árbol del proyecto nuevo (secciones del programa)

FUNCIONAMIENTO ESTACIONES DE TRABAJO.

En esta carpeta se han añadido todas las partes del programa, ya sean Bloques de función (FB) o secciones del programa (PRG), que describen el funcionamiento de las estaciones de trabajo, como también los cronómetros, temporizadores y contadores que se usan para este proceso.

FUNCIONAMIENTO LINIA.

En esta carpeta se han añadido todas las partes del programa, ya sean Bloques de función (FB) o secciones del programa (PRG), que describen el funcionamiento de todos los elementos físicos a tener en cuenta en la línea de producción. Ya sean retenedores, plataformas, sensores o actuadores.

FUNCIONAMIENTO VISUALIZACIONES.

En esta carpeta se han añadido todas las partes del programa, ya sean Bloques de función (FB) o secciones del programa (PRG), que describen el funcionamiento de la visualización del proceso. Es decir, describen el comportamiento de los elementos del sinóptico, a partir del comportamiento de los elementos físicos del proceso.

3.1.2. Variables globales del programa

Otra de las partes que es importante tener bajo un orden específico para poder tener una mayor comprensión de la estructuración del programa, son las listas de variables.

En el programa anterior, como podemos ver, todas las variables están agrupadas en una misma tabla.

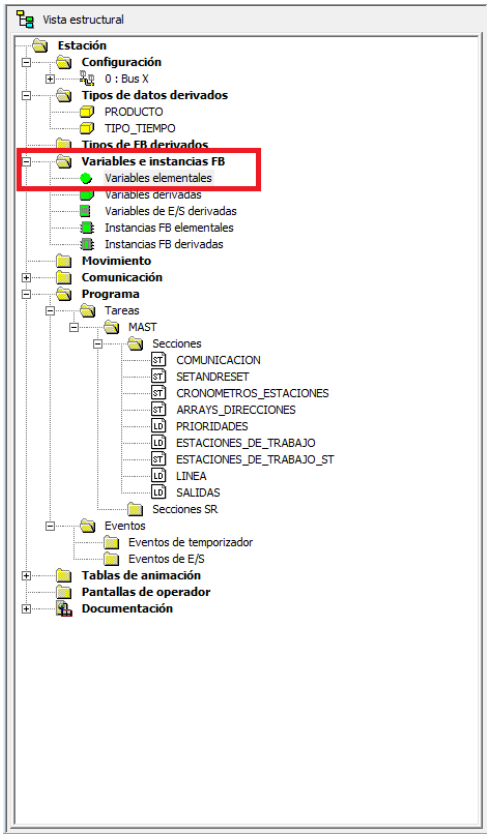


Figura 12- Árbol del proyecto antiguo (tabla de variables)

Esta tabla está compuesta de 237 variables. Sin ningún orden, mezclando variables físicas con variables de programa.

Variables | Tipos de DDT | Bloques de funciones | Tipos de DFB

Filtro — Nombre: *

Nombre	Tipo	Dirección	Valor
AAAAAELIMINAR	INT		
ARRAY_PT09_IN_PT08_OUT	ARRAY[1..10] OF PRODUCTO	%MW305	
ARRAY_PT09_OUT_PT10_IN	ARRAY[1..10] OF PRODUCTO		
ARRAY_PT09_OUT_PT12_IN	ARRAY[1..10] OF PRODUCTO		
ARRAY_PT10_OUT_PT12_IN	ARRAY[1..10] OF PRODUCTO		
ARRAY_PT10_OUT_PT14_IN	ARRAY[1..10] OF PRODUCTO		
ARRAY_PT12_OUT_PT14_IN	ARRAY[1..10] OF PRODUCTO		
ARRAY_PT14_IN_PT12_OUT	ARRAY[1..10] OF PRODUCTO		
ARRAY_PT14_OUT_PT16_IN	ARRAY[1..10] OF PRODUCTO	%MW352	
ARRAY_TIEMPO_E_1	ARRAY[1..4] OF TIPO_TIEMPO	%MW200	
ARRAY_TIEMPO_E_2	ARRAY[1..4] OF TIPO_TIEMPO	%MW212	
ARRAY_TIEMPO_E_3	ARRAY[1..4] OF TIPO_TIEMPO	%MW224	
ARRAY_TIEMPO_VACIA	ARRAY[1..4] OF TIPO_TIEMPO		
BAJA_PT09	BOOL	%QW0.2.0.0.1	
BAJA_PT10	BOOL	%QW0.2.0.4.1	
CONTADOR_PRIORIDAD_PT12	EBOOL		
CONTADOR_PRIORIDAD_PT12_DIR18	EBOOL		
CONTADOR_PRIORIDAD_PT14_DIR19	EBOOL		
CONTADOR_PT07_OUT_PT08_IN	INT		
CONTADOR_PT08_ELIMINAR	INT		
CONTADOR_PT08_OUT	INT		
CONTADOR_PT09_IN_PT08_OUT	INT		
CONTADOR_PT09_OUT_PT10_IN	INT		
CONTADOR_PT09_OUT_PT12_IN	INT		
CONTADOR_PT10_IN_PT09_OUT	INT		
CONTADOR_PT10_OUT_PT12_IN	INT		
CONTADOR_PT10_OUT_PT14_IN	INT		
CONTADOR_PT12_IN_PT09_OUT	INT		
CONTADOR_PT12_IN_PT10_OUT	INT		
CONTADOR_PT12_OUT_PT14_IN	INT		
CONTADOR_PT12_OUT_PT_IN	INT		
CONTADOR_PT14_IN_PT10_OUT	INT		
CONTADOR_PT14_IN_PT12_OUT	INT		
CONTADOR_PT14_OUT_PT15_IN	INT		
CONTADOR_TIEMPO_E_1	INT		
CONTADOR_TIEMPO_E_2	INT		
CONTADOR_TIEMPO_E_3	INT		
DC00	BOOL	%IW0.2.0.0.1	
DC01	BOOL	%IW0.2.0.2.5	
DC02	BOOL	%IW0.2.0.9.2	
DFT6	BOOL	%IW0.2.0.0.2	
DFT7	BOOL	%IW0.2.0.1.1	
DI08	BOOL	%IW0.2.0.0.3	

Figura 13- Tabla de variables (proyecto antiguo) - 1

Nombre	Tipo	Dirección	Valor
DC00	BOOL	%IW0.2.0.0.1	
DC01	BOOL	%IW0.2.0.2.5	
DC02	BOOL	%IW0.2.0.9.2	
DFT6	BOOL	%IW0.2.0.0.2	
DFT7	BOOL	%IW0.2.0.1.1	
DI08	BOOL	%IW0.2.0.0.3	
DI09	BOOL	%IW0.2.0.1.0	
DI10	BOOL	%IW0.2.0.9.0	
DIB02	BOOL	%IW0.2.0.0.0	
DIB03	BOOL	%IW0.2.0.1.3	
DIB04	BOOL	%IW0.2.0.2.3	
DIB05	BOOL	%IW0.2.0.10.0	
DIR10	BOOL	%IW0.2.0.6.0	
DIR10_BLOQUEO	EBOOL		
DIR10_down	BOOL	%QW0.2.0.1.2	
DIR10_MOVE	EBOOL		
DIR10_MOVE_E	EBOOL		
DIR10_MOVE_F_A	EBOOL		
DIR10_READY	EBOOL		
DIR10_REST	EBOOL		
DIR11	BOOL	%IW0.2.0.0.4	
DIR11_BLOQUEO	EBOOL		
DIR11_down	BOOL	%QW0.2.0.1.0	
DIR11_MOVE	EBOOL	%M6	
DIR11_READY	EBOOL	%M5	
DIR11_REST	EBOOL	%M4	
DIR12	BOOL	%IW0.2.0.0.5	
DIR12_BLOQUEO	EBOOL		
DIR12_down	BOOL	%QW0.2.0.1.1	
DIR12_MOVE	EBOOL	%M9	
DIR12_MOVE_E	EBOOL		
DIR12_MOVE_F_A	EBOOL		
DIR12_READY	EBOOL	%M8	
DIR12_REST	EBOOL	%M7	
DIR13	BOOL	%IW0.2.0.10.1	
DIR13_BLOQUEO	EBOOL		
DIR13_down	BOOL	%QW0.2.0.5.2	
DIR13_MOVE	EBOOL	%M12	
DIR13_MOVE_E	EBOOL		
DIR13_MOVE_F_A	EBOOL		
DIR13_READY	EBOOL	%M11	
DIR13_REST	EBOOL	%M10	
DIR14	BOOL	%IW0.2.0.1.2	

Figura 14- Tabla de variables (proyecto antiguo) - 2

En el caso del nuevo programa, se distinguen 5 tablas de variables diferentes.

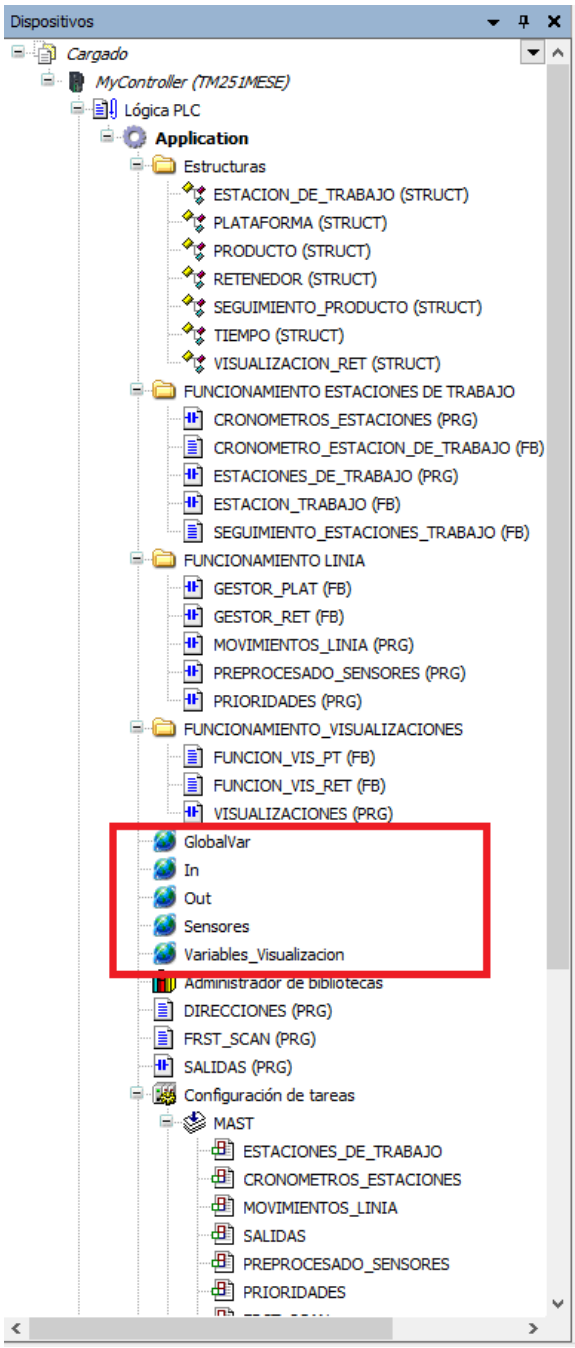


Figura 15- Árbol del proyecto nuevo (tabla de variables)

3.1.2.1. GlovalVar

En esta lista de variables se pueden encontrar las variables globales. Es decir, esas variables donde se encuentran en gran parte del programa. En ella se encuentran los elementos básicos de nuestro proceso: Los retenedores, plataformas y estaciones de trabajo.

```
GlobalVar
1   VAR_GLOBAL
2       PT09: PLATAFORMA;
3       PT10: PLATAFORMA;
4
5       RET08: RETENEDOR;
6       RET11: RETENEDOR;
7       RET12: RETENEDOR;
8       RET13: RETENEDOR;
9       RET14: RETENEDOR;
10      RET15: RETENEDOR;
11
12      DIR10: RETENEDOR;
13      DIR11: RETENEDOR;
14      DIR12: RETENEDOR;
15      DIR13: RETENEDOR;
16      DIR14: RETENEDOR;
17      DIR15: RETENEDOR;
18      DIR16: RETENEDOR;
19      DIR17: RETENEDOR;
20      DIR18: RETENEDOR;
21      DIR19: RETENEDOR;
22
23      Estacion_T1: ESTACION_DE_TRABAJO;
24      Estacion_T2: ESTACION_DE_TRABAJO;
25      Estacion_T3: ESTACION_DE_TRABAJO;
26
27  END_VAR
```

Figura 16- Tabla de variables globales

3.1.2.2. In

En esta lista de variables podemos encontrar todas las variables con dirección física al PLC. Son todos aquellos elementos de nuestro proceso que den algún tipo de señal de entrada al PLC, como pueden ser los sensores.

```
In
1   VAR_GLOBAL
2       DIB02   AT  %IX0.0: BOOL;
3       DC00    AT  %IX0.1: BOOL;
4       DFT6    AT  %IX0.2: BOOL;
5       DI08    AT  %IX0.3: BOOL;
6       S_DIR11 AT  %IX0.4: BOOL;
7       S_DIR12 AT  %IX0.5: BOOL;
8       DI09    AT  %IX1.0: BOOL;
9       DFT7    AT  %IX1.1: BOOL;
10      S_DIR14 AT  %IX1.2: BOOL;
11      DIB03   AT  %IX1.3: BOOL;
12      E6      AT  %IX1.4: BOOL;
13      P_ONE_Plate1    AT  %IX1.5: BOOL;
14      P_ALL_Plates1   AT  %IX2.0: BOOL;
15      S_DIR16 AT  %IX2.1: BOOL;
16      S_DIR17 AT  %IX2.2: BOOL;
17      DIB04   AT  %IX2.3: BOOL;
18      S_DIR15 AT  %IX2.4: BOOL;
19      DC01    AT  %IX2.5: BOOL;
20      PULSADOR_REAL_MOTOR_CENTR   AT  %IX3.0: BOOL;
21      PULSADOR_REAL_MOTOR_DER AT  %IX3.1: BOOL;
22      S_DIR10 AT  %IX6.0: BOOL;
23      DI10    AT  %IX9.0: BOOL;
24      PRESOSTATO_SEGURIDAD_MINIMA AT  %IX9.1: BOOL;
25      DC02    AT  %IX9.2: BOOL;
26      P_ONE_Plate2    AT  %IX9.3: BOOL;
27      P_ALL_Plates2   AT  %IX9.4: BOOL;
28      E7  AT  %IX9.5:     BOOL;
29      DIB05   AT  %IX10.0:    BOOL;
30      S_DIR13 AT  %IX10.1:    BOOL;
31      S_DIR19 AT  %IX10.2:    BOOL;
32      S_DIR18 AT  %IX10.3:    BOOL;
33  END_VAR
```

Figura 17- Tabla variables IN

3.1.2.3. Out

En esta lista de variables podemos encontrar todas las variables con dirección física al PLC. Son todos aquellos elementos de nuestro proceso que den algún tipo de señal de salida al PLC, como pueden ser los motores de las cintas o los movimientos de los retenedores y plataformas.

```
Out
VAR_GLOBAL
    SUBE_PT09    AT  %QX0.0: BOOL;
    BAJA_PT09    AT  %QX0.1: BOOL;
    DIR14_down   AT  %QX0.2: BOOL;
    SUBE_PT11    AT  %QX0.3: BOOL;
    DIR16_down   AT  %QX0.4: BOOL;
    DIR17_down   AT  %QX0.5: BOOL;
    DIR11_down   AT  %QX1.0: BOOL;
    DIR12_down   AT  %QX1.1: BOOL;
    DIR10_down   AT  %QX1.2: BOOL;
    MOTOR_CENTRALBAND    AT  %QX1.4: BOOL;
    MOTOR_RIGHTBAND      AT  %QX1.5: BOOL;
    M21      AT  %QX2.0: BOOL;
    M16      AT  %QX2.1: BOOL;    (* motor PT09 to DIR14 *)
    M17      AT  %QX2.2: BOOL;    (* motor PT10 to DIR15 *)
    M19      AT  %QX2.3: BOOL;
    M20      AT  %QX2.4: BOOL;
    M20_reverse AT  %QX2.5: BOOL;
    M22      AT  %QX3.0: BOOL;
    SUBE_PT10    AT  %QX4.0: BOOL;
    BAJA_PT10    AT  %QX4.1: BOOL;
    DIR15_down   AT  %QX4.2: BOOL;
    SUBE_PT12    AT  %QX4.3: BOOL;
    DIR18_down   AT  %QX4.4: BOOL;
    SUBE_PT13    AT  %QX4.5: BOOL;
    DIR19_down   AT  %QX5.0: BOOL;
    SUBE_RET14   AT  %QX5.1: BOOL;
    DIR13_down   AT  %QX5.2: BOOL;
    SUBE_PT08    AT  %QX5.3: BOOL;


END_VAR
```

Figura 18- Tabla variables OUT

3.1.2.4. Sensores

En esta lista encontramos las variables de programa donde van dirigidas las señales de los sensores. Para la manipulación de las señales de los sensores a la hora de simular el proceso, son necesarias estas variables.

```
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 19- Tabla variables de Sensores

3.1.2.5. Variables_Visualizacion

En esta tabla encontramos las variables relacionadas con la visualización del proceso. Es decir, todos los elementos visuales que están descritos en el sinóptico para la simulación.

```
Variables_Visualizacion
1   VAR_GLOBAL
2       RT08:   VISUALIZACION_RET_AND_PLAT;
3       P09:    VISUALIZACION_RET_AND_PLAT;
4       P10:    VISUALIZACION_RET_AND_PLAT;
5       RT11:   VISUALIZACION_RET_AND_PLAT;
6       RT12:   VISUALIZACION_RET_AND_PLAT;
7       RT13:   VISUALIZACION_RET_AND_PLAT;
8       RT14:   VISUALIZACION_RET_AND_PLAT;
9       RT15:   VISUALIZACION_RET_AND_PLAT;
10
11      DR10:   VISUALIZACION_RET_AND_PLAT;
12      DR11:   VISUALIZACION_RET_AND_PLAT;
13      DR12:   VISUALIZACION_RET_AND_PLAT;
14      DR13:   VISUALIZACION_RET_AND_PLAT;
15      DR14:   VISUALIZACION_RET_AND_PLAT;
16      DR15:   VISUALIZACION_RET_AND_PLAT;
17      DR16:   VISUALIZACION_RET_AND_PLAT;
18      DR17:   VISUALIZACION_RET_AND_PLAT;
19      DR18:   VISUALIZACION_RET_AND_PLAT;
20      DR19:   VISUALIZACION_RET_AND_PLAT;
21
22  END_VAR
```

Figura 20- Variables visualización

Como se puede apreciar en las imágenes de los proyectos, el cambio en la estructura ha cambiado y ahora, es más intuitiva que antes.

Debido a la separación de las secciones del programa, en tres partes claramente diferenciadas, podemos asegurar que la próxima persona que deba modificar este programa, tendrá las cosas mucho más fáciles. Lo que se traduce en menos horas de entendimiento, y, por tanto, menos dinero destinado a horas perdidas.

Además, con la diferenciación entre tablas de variables, es mucho más fácil poder encontrar una variable que se desee modificar o eliminar.

3.2. Limitaciones

Como ya hemos expuesto anteriormente, nuestro código base, es un código que, en el caso de necesitar una modificación física en la máquina, supondría un gran esfuerzo y horas de dedicación para la persona que vaya a programarlo. En una empresa productora de maquinaria, dónde el código es un 90% idéntico en la mayoría de ellas, sería un gran costo en horas de trabajo el hecho de hacer las modificaciones en un código con estas características.

Para evitar ese costo, debemos crear un código fácil, que no requiera demasiado tiempo para entender su estructura, y en el que, por ejemplo, añadir un elemento físico a la máquina, no suponga un costo elevado en tiempo y trabajo, en comparación a lo que realmente estamos haciendo.

3.2.1. Creación de Retenedores

En el caso de nuestro programa antiguo, de querer añadir un retenedor a nuestra línea de producción, sería necesario la creación de 8 variables y más de 20 líneas de programación nuevas.

Como vemos en la siguiente imagen de la tabla de variables, para un solo retenedor, es necesario crear 8 variables.

DIR10	BOOL
DIR10_BLOQUEO	EBOOL
DIR10_down	BOOL
DIR10_MOVE	EBOOL
DIR10_MOVE_E	EBOOL
DIR10_MOVE_F_A	EBOOL
DIR10_READY	EBOOL
DIR10_REST	EBOOL

Figura 21- Variables de estado para un retenedor - 1

Para el caso de nuestro nuevo programa, se ha creado una estructura de variables, que define todas las variables de un retenedor.

RETENEDOR

```
1   TYPE RETENEDOR :
2   STRUCT
3       REST:    BOOL;
4       READY:   BOOL;
5       MOV:     BOOL;
6       BLOQ:    BOOL;
7       PRODUCTO:    BOOL;
8   END_STRUCT
9   END_TYPE
10
```

Figura 22- Variables de estado para un retenedor - 2

Esto nos permite crear variables de tipo retenedor. Lo que conlleva la creación de solo una variable para el añado de un solo retenedor físico a nuestra línea.

En el caso de las líneas de programación, como podemos ver en las imágenes siguientes, la programación de las acciones para un retenedor en la línea de trabajo es en todos los casos, idéntica. El único cambio apreciable es la polaridad del sensor.

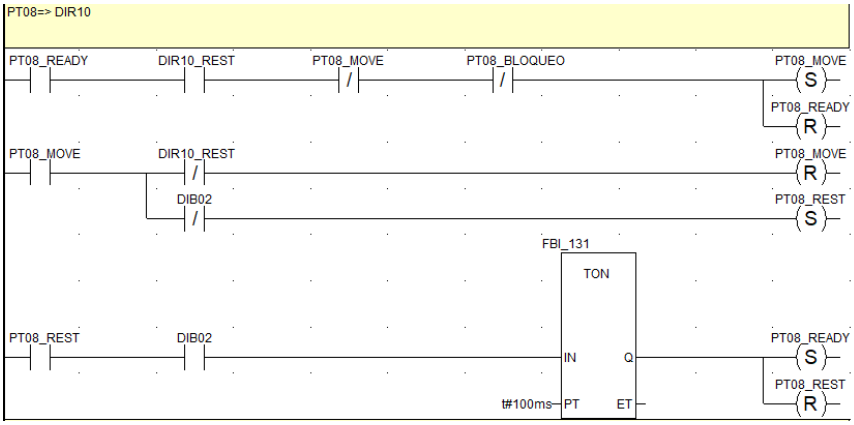


Figura 23- Acción de un retenedor - 1

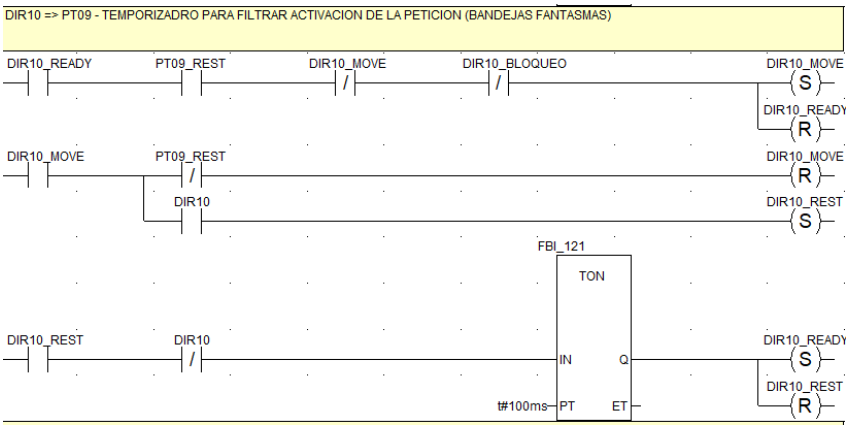


Figura 24- Acción de un retenedor - 2

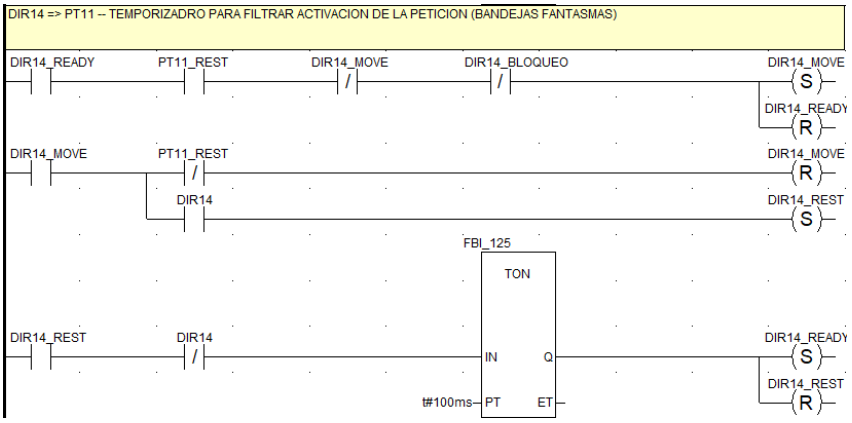


Figura 25- Acción de un retenedor - 3

Para no tener que repetir esta programación tantas veces como retenedores haya o se quieran añadir, en el nuevo programa se ha añadido un Bloque de Función (FB) que realice contenga esas líneas de programación.

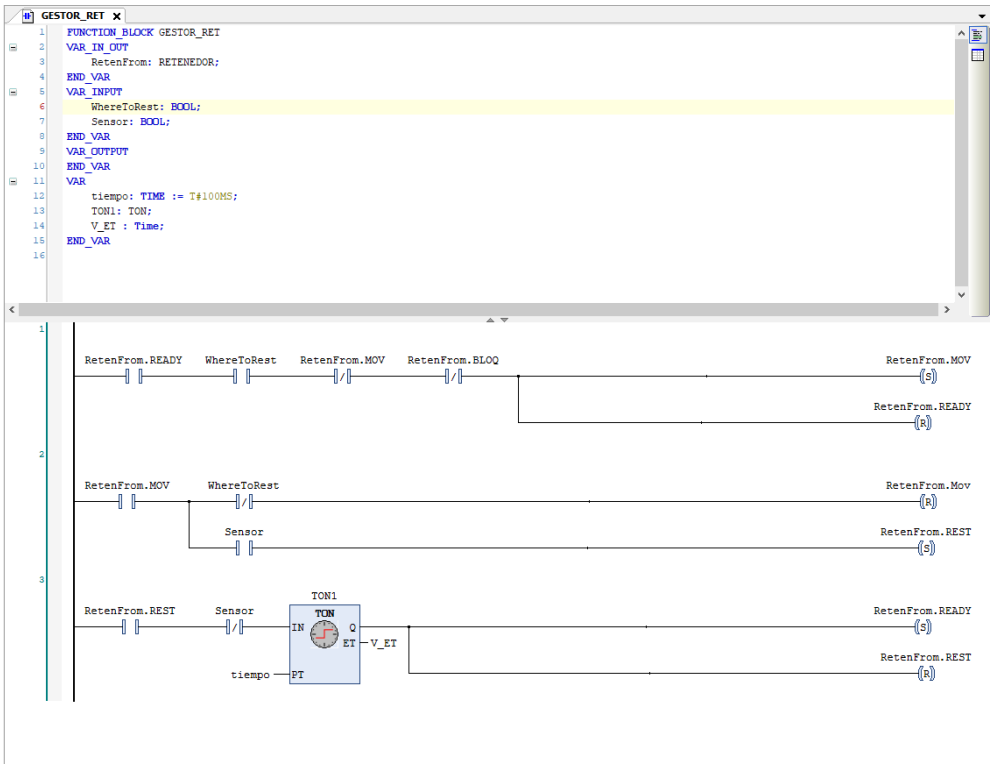


Figura 26- Programación FB Retenedores.

La programación de todos los retenedores, como ya hemos dicho anteriormente, es la misma. Por lo tanto, para añadir un retenedor a la programación, simplemente deberíamos crear una variable tipo Retenedor y usar el bloque de función.

El resultado del uso de FB es un programa bien estructurado, con todos los movimientos de la línea bien definidos y con una posibilidad de error mínima, debido a nuestra poca cantidad de líneas de programación.

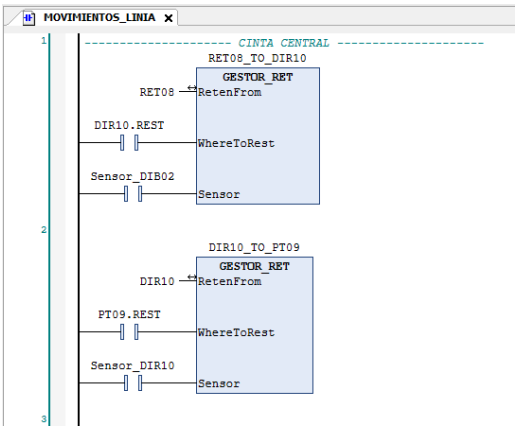


Figura 27- Ejemplo uso FB para Retenedores.

3.2.2. Creación de Plataformas.

Como en el caso de los retenedores, las líneas de programación en nuestro programa base son totalmente idénticas para todos los casos de plataformas. Pero en este caso, son necesarias, solo en variables de estado, 14 posiciones de memoria. Y en el caso de líneas de programación, más de 10, que como ya hemos dicho, son idénticas para todas las plataformas.

PT09_DIR11_BLOQUEO	EBOOL
PT09_DIR14_BLOQUEO	EBOOL
PT09_DIRECCION_DIR11	EBOOL
PT09_DIRECCION_DIR14	EBOOL
PT09_down	EBOOL
PT09_MOVE	EBOOL
PT09_MOVE_DIR11	EBOOL
PT09_MOVE_DIR11_E	EBOOL
PT09_MOVE_DIR11_F_A	EBOOL
PT09_MOVE_DIR14	EBOOL
PT09_MOVE_DIR14_E	EBOOL
PT09_MOVE_DIR14_F_A	EBOOL
PT09_READY	EBOOL
PT09_REST	EBOOL

Figura 28- Variables de estado para una plataforma - 3

Para nuestro nuevo programa, el uso de las variables para las plataformas, como en el caso de los retenedores, también se ha simplificado con el uso de una estructura de datos.

```
PLATAFORMA
1   TYPE PLATAFORMA :
2   STRUCT
3       REST:    BOOL;
4       READY:   BOOL;
5       MOV1:    BOOL;
6       MOV2:    BOOL;
7       BLOQ1:   BOOL;
8       BLOQ2:   BOOL;
9       AVAN1:   BOOL;
10      AVAN2:   BOOL;
11  END_STRUCT
12  END_TYPE
13
```

Figura 29- Variables de estado para una plataforma - 2

Con esta estructura de datos, permitimos, a los siguientes programadores del código, crear una plataforma con tan solo crear una variable tipo Plataforma, donde ya incluye todas las variables de estado

En el caso de las líneas de programación, para cada plataforma será necesario la creación de más de 10 líneas idénticas, y como vemos en el programa base, se han ido copiando para todas las plataformas existentes.

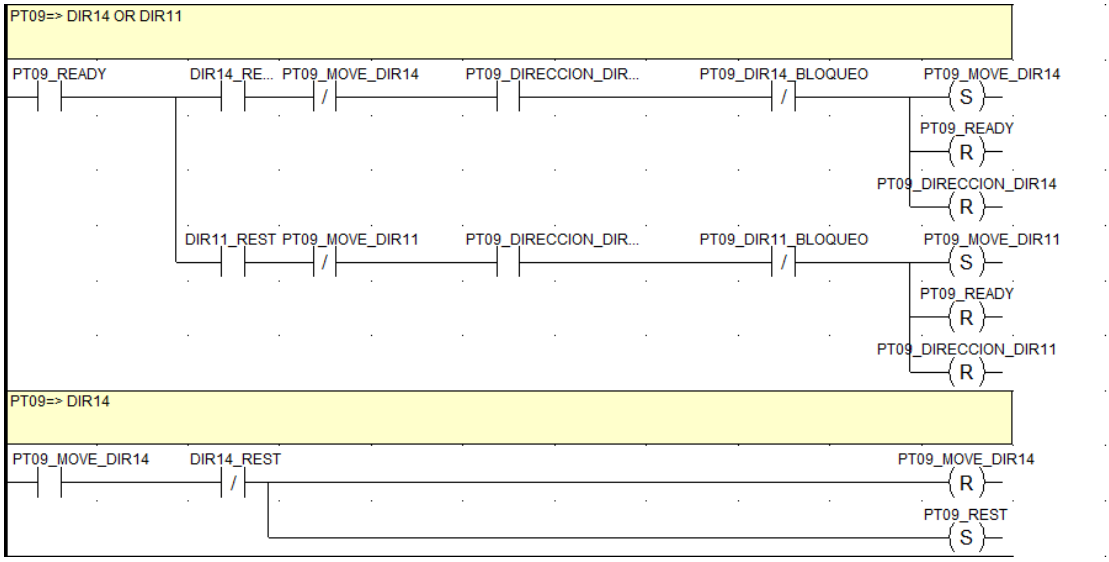


Figura 30- Líneas de programación para una plataforma - 1

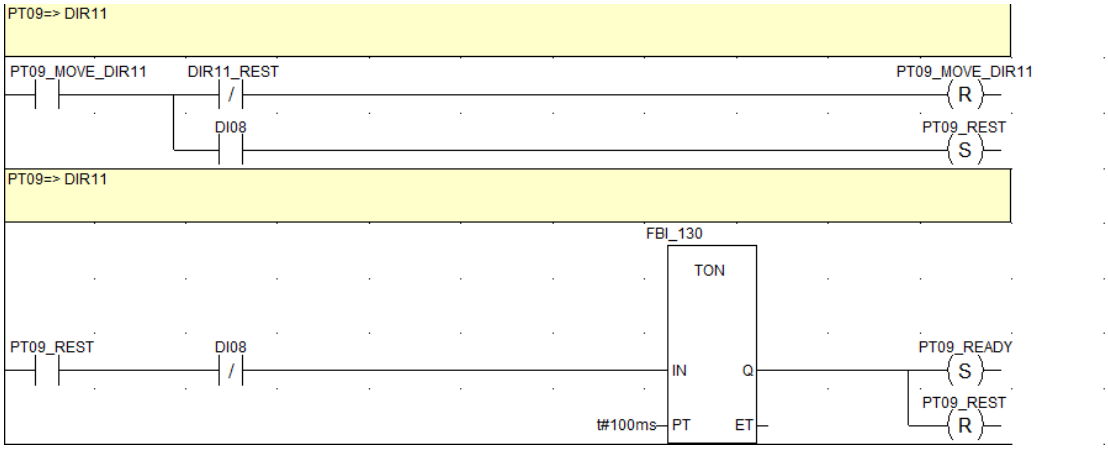


Figura 31- Líneas de programación para una plataforma - 2

Para este caso también se ha recurrido a las FB. Así nos ahorraremos tiempo en copiar esas líneas de programación, una y otra vez, por cada plataforma que se quiera añadir a la línea.

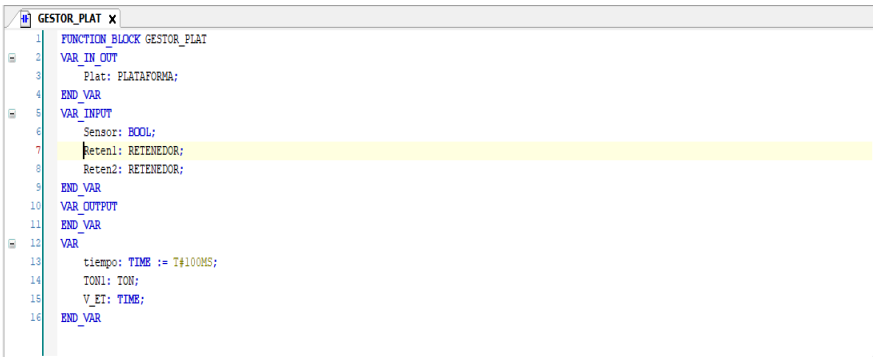


Figura 32 - Variables FB para plataformas.

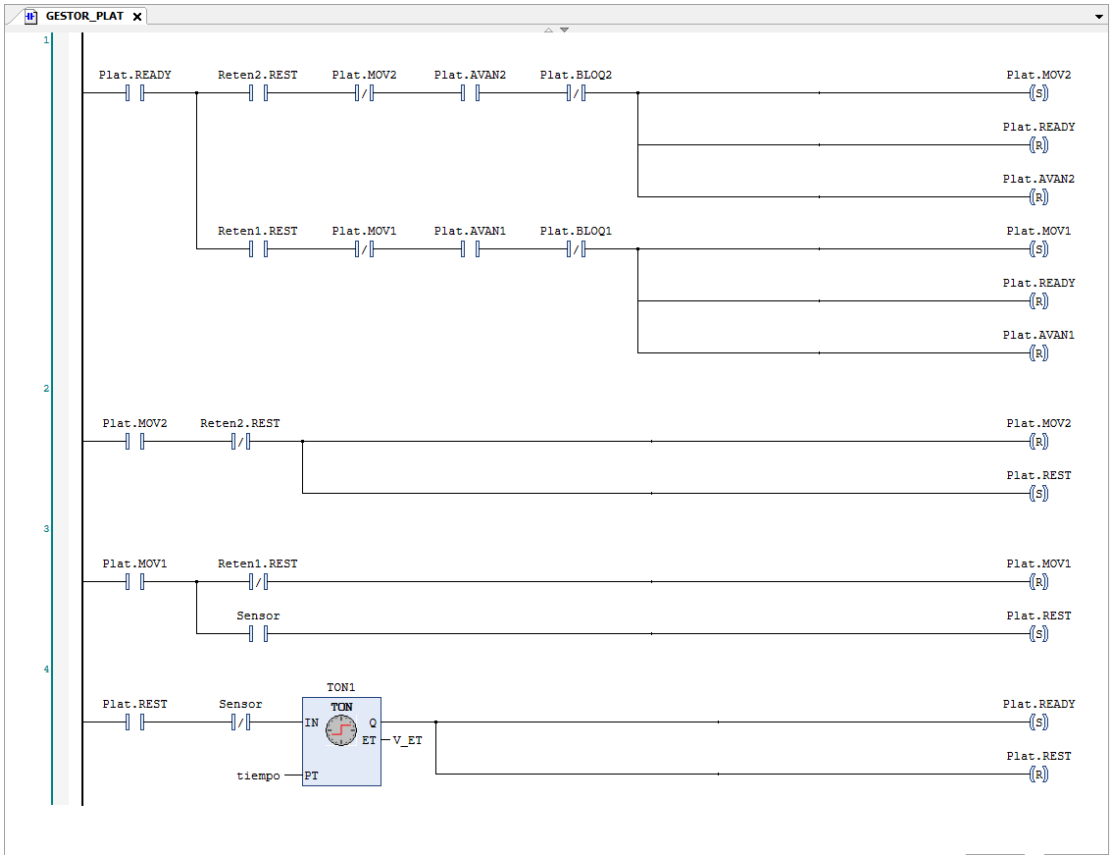


Figura 33 - Líneas de programación FB para plataformas.

Como ya hemos visto en el caso de los Retenedores, el uso de FB deja un código de programación intuitivo y fácil a la hora de detectar cualquier tipo de error en el programa.

Y hemos pasado de tener más de 20 líneas de programación para dos Plataformas, a tener solamente dos, como podemos ver en las siguientes imágenes.

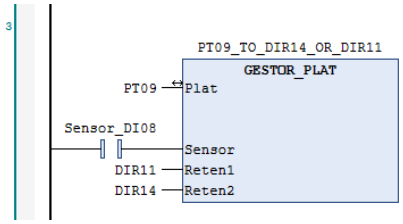


Figura 32 - Uso FB para plataformas - 1

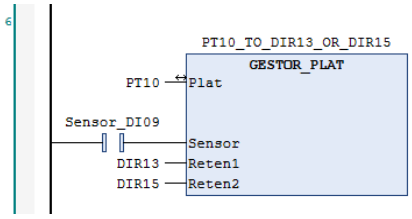


Figura 33 - Uso FB para plataformas - 2

3.2.3. Creación de Estaciones de Trabajo.

En comparación con los retenedores y plataformas, las estaciones de trabajo tienen mucho más que la programación para el estado físico de las mismas. Tienen un seguimiento de las bandejas que pasan por ellas, cuantas han pasado, el producto que llevan, la dirección que debería tomar en la cinta, contadores...

Por ello, la optimización del código en la creación de Estaciones de Trabajo es notoriamente más elevada que con los retenedores y plataformas.

Para empezar con la comparación, miraremos las variables de estado de las estaciones de trabajo en el proyecto base.

TEMPO_ESTACION_1	EBOOL
TEMPO_ESTACION_1_E	EBOOL
TEMPO_ESTACION_1_F_A	EBOOL
TEMPO_ESTACION_2	EBOOL
TEMPO_ESTACION_2_E	EBOOL
TEMPO_ESTACION_2_F_A	EBOOL
TEMPO_ESTACION_3	EBOOL
TEMPO_ESTACION_3_E	EBOOL
TEMPO_ESTACION_3_F_A	EBOOL

Figura 34 - Variables Estaciones de Trabajo - 1

TIEMPO_E_1	TIME
TIEMPO_E_2	TIME
TIEMPO_E_3	TIME
TIPO_PRODUCTO	EBOOL
TRABAJO_ESTACION_1	EBOOL
TRABAJO_ESTACION_2	EBOOL
TRABAJO_ESTACION_3	EBOOL

Figura 35 - Variables Estaciones de Trabajo - 2

CONTADOR_PT09_IN_PT08_OUT	INT	
CONTADOR_PT09_OUT_PT10_IN	INT	
CONTADOR_PT09_OUT_PT12_IN	INT	
CONTADOR_PT10_IN_PT09_OUT	INT	
CONTADOR_PT10_OUT_PT12_IN	INT	
CONTADOR_PT10_OUT_PT14_IN	INT	
CONTADOR_PT12_IN_PT09_OUT	INT	
CONTADOR_PT12_IN_PT10_OUT	INT	
CONTADOR_PT12_OUT_PT14_IN	INT	
CONTADOR_PT12_OUT_PT_IN	INT	
CONTADOR_PT14_IN_PT10_OUT	INT	
CONTADOR_PT14_IN_PT12_OUT	INT	
CONTADOR_PT14_OUT_PT15_IN	INT	
CONTADOR_TIEMPO_E_1	INT	
CONTADOR_TIEMPO_E_2	INT	
CONTADOR_TIEMPO_E_3	INT	

Figura 36 - Variables Estaciones de Trabajo - 3

Como vemos, hay una cantidad de variables sustancialmente mayor en comparación con las variables de los retenedores y plataforma. En el caso de querer añadir una estación de trabajo en un futuro, solamente en creación de variables, ya se debería invertir mucho tiempo. Si hablamos de llegar a entender el código y el propósito de cada variable, la inversión de horas sería, sin duda, elevada.

Para facilitar la faena en cuanto a creación de las variables para una estación de trabajo, se ha creado una estructura de datos, con todas las variables necesarias para el correcto funcionamiento de las estaciones.

ESTACION_DE_TRABAJO

```
1   TYPE ESTACION_DE_TRABAJO :
2   STRUCT
3       (*////////////////////////////////////////*)
4       (*Variables para el LD *)
5       ESTACION: BOOL;
6       TRABAJO_ESTACION: BOOL; (* estación trabajando *)
7       TIEMPO_ESTACION: BOOL; (*tiempo de trabajo realizado ladder *)
8       TIEMPO_TEMPORIZADOR: TIME; (* tiempo TON para guardar en array *)
9       (*////////////////////////////////////////*)
10      (* Variables para el ST *)
11      ARR_PROD_EST: ARRAY[1..10] OF SEGUIMIENTO_PRODUCTO;
12      CONT_BANDEJAS_EST: INT;
13      ARR_TIEMPO_EST: ARRAY[1..4] OF TIEMPO;
14      CONT_TIEMPO_EST: INT;
15      TIEMPO_TEMPORIZADOR_ST: TIME;
16      FLANCO_D: BOOL;
17  END_STRUCT
18  END_TYPE
```

Figura 37 - Variables Estructura Estaciones de Trabajo

Estas son las variables necesarias para cada Estación de Trabajo que tenga nuestro proyecto. En el caso de querer crear las variables para una nueva Estación de Trabajo, solo se debe crear una variable tipo ESTACION_DE_TRABAJO.

A parte de esta estructura de datos, se ha visto necesario la creación de otras dos estructuras de datos, destinadas al seguimiento de las bandejas.

SEGUIMIENTO_PRODUCTO:

```
SEGUIMIENTO_PRODUCTO
1  TYPE SEGUIMIENTO_PRODUCTO :
2  STRUCT
3      ID: INT;
4      TIPO_DE_PRODUCTO: INT;
5      BANDEJA: INT;
6  END_STRUCT
7  END_TYPE
8
```

Figura 38 - Variables Estructura SEGUIMIENTO_PRODUCTO

Esta estructura contiene las variables necesarias para el seguimiento completo de los productos que llevan las bandejas y que pasan por cada estación de trabajo.

TIEMPO:

```
TIEMPO
1  TYPE TIEMPO :
2  STRUCT
3      CRONO: TIME;
4      ID: INT;
5  END_STRUCT
6  END_TYPE
7
```

Figura 39 - Variables Estructura TIEMPO

Esta estructura contiene las variables necesarias para el seguimiento del tiempo de trabajo de cada una de las bandejas que pasa por las estaciones.

En el caso de las líneas de programación, como en los casos anteriores a los que nos hemos referido a nuestro programa base, son idénticas para cada estación de trabajo que tengamos en la línea. Por ello, en el caso que queramos añadir otra estación de trabajo al proceso, se deberían crear las mismas líneas que para las ya creadas.

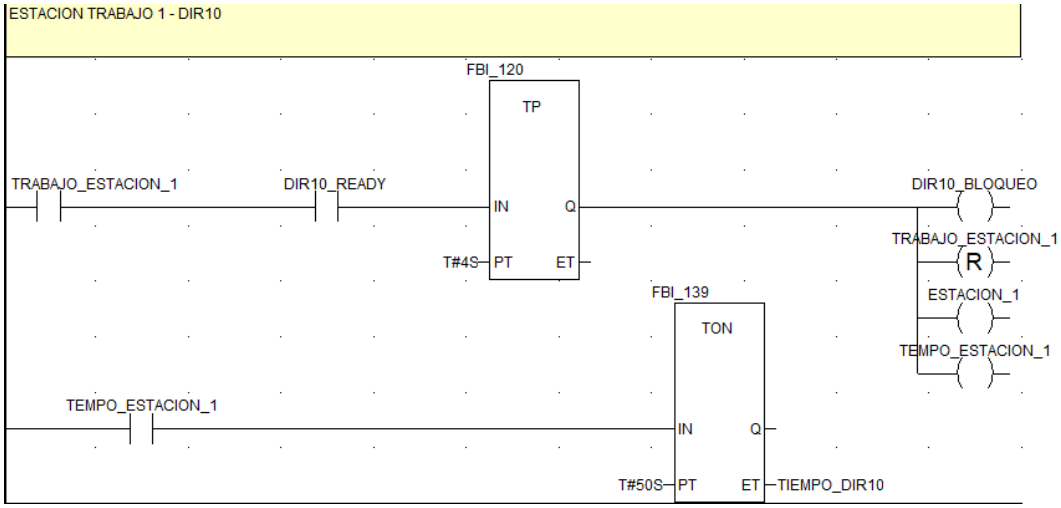


Figura 40 - Programación estación de trabajo proyecto antiguo

Para este caso, como en los problemas anteriores de repetición de código, se ha creado un Bloque de Función con la programación para las estaciones de trabajo.

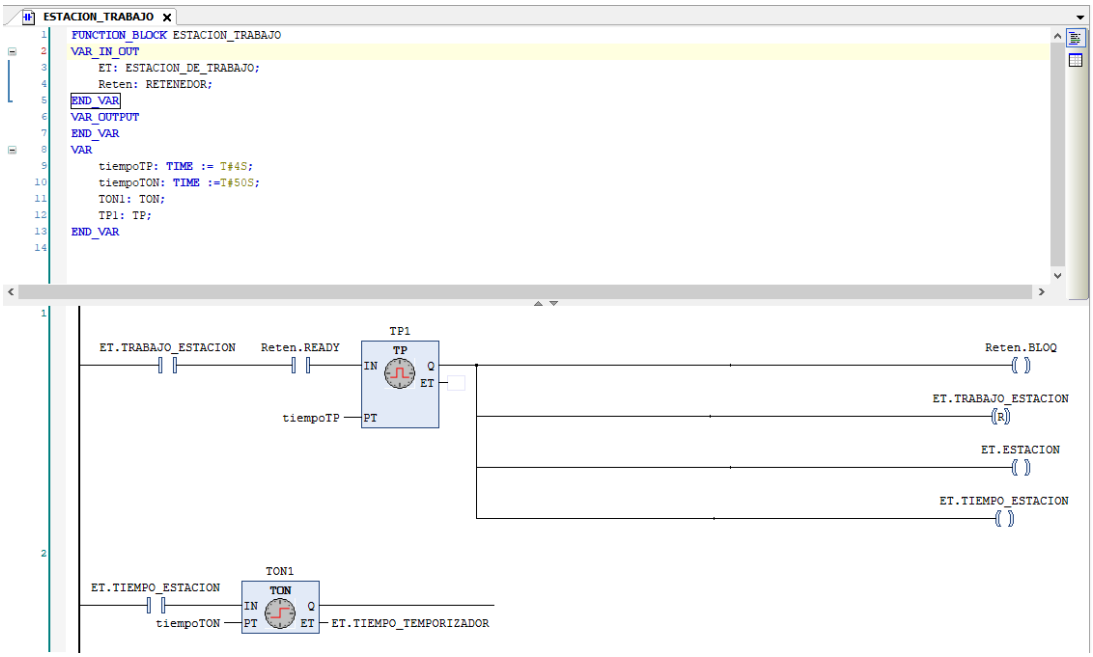


Figura 41 - Programación estación de trabajo proyecto nuevo

Como se puede ver, el código es el mismo que en el proyecto antiguo, pero en este caso solo se ha tenido que crear una vez, y aplicar el FB por cada una de las estaciones de trabajo que tenga el proceso, como se puede ver en la imagen siguiente.

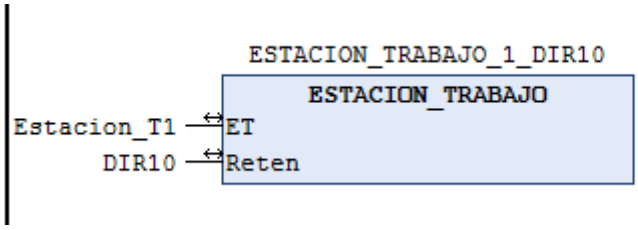


Figura 42 - Aplicación FB estación de trabajo

Para hacer el seguimiento de los tiempos de trabajo de cada una de las estaciones, en el proyecto anterior se repetían las siguientes líneas de programación:

```
IF TEMPO_ESTACION_1 = TRUE THEN

        TIEMPO_E_1 := TIEMPO_DIR10;
        FLANCO_D_E_1 := FALSE;
END_IF;
```

Figura 43 - Líneas programación tiempo Estación de trabajo

Estas líneas de programación están repetidas por cada estación de trabajo que contenga nuestro proceso.

Para nuestro nuevo proyecto, se ha creado un FB con las mismas líneas de programación y se ha aplicado tantos, como estaciones de trabajo hay.

```
CRONOMETRO_ESTACION_DE_TRABAJO
1  FUNCTION_BLOCK CRONOMETRO_ESTACION_DE_TRABAJO
2  VAR_IN_OUT
3      EST: ESTACION_DE_TRABAJO;
4  END_VAR
5  VAR_OUTPUT
6  END_VAR
7  VAR
8  END_VAR
9
```

```
1  IF EST.TIEMPO_ESTACION = TRUE THEN
2      EST.TIEMPO_TEMPORIZADOR_ST := EST.TIEMPO_TEMPORIZADOR;
3  END_IF
```

Figura 44 - Líneas programación FB Cronometro Estación de trabajo

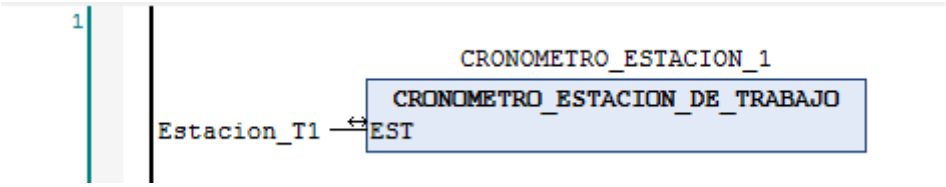


Figura 45 - Aplicación FB Cronometro Estación de trabajo

La última función de las estaciones de trabajo es el seguimiento de los productos de cada bandeja que pasa por cada una de ellas.

Esto, en nuestro programa base, se gestiona a través de unas líneas de programación, que como la mayoría del proyecto, se repiten por cada elemento. En nuestro caso, se repiten por cada estación de trabajo.

```
IF PT08_MOVE_F_A = TRUE AND ARRAY_PT09_IN_PT08_OUT[CONTADOR_PT08_OUT].TIPO_PRODUCTO > 0 THEN
        TRABAJO_ESTACION_1 := TRUE;
END_IF;
```

Figura 46 - Programación gestión de productos Proyecto antiguo - 1

```
TEMPO_ESTACION_1_E := TEMPO_ESTACION_1;

TEMPO_ESTACION_1_F_A := FE(TEMPO_ESTACION_1_E);

IF (TEMPO_ESTACION_1_F_A) = TRUE THEN

        INC(CONTADOR_TIEMPO_E_1);
        IF CONTADOR_TIEMPO_E_1 = 5 THEN
        ARRAY_TIEMPO_E_1 := ARRAY_TIEMPO_VACIA;
        CONTADOR_TIEMPO_E_1 := 1;
        END_IF;

        ARRAY_TIEMPO_E_1[CONTADOR_TIEMPO_E_1].CRONO := TIEMPO_E_1;
        ARRAY_TIEMPO_E_1[CONTADOR_TIEMPO_E_1].ID := ARRAY_PT09_IN_PT08_OUT[CONTADOR_PT08_OUT].ID;

        FLANCO_D_E_1 := TRUE;

END_IF;
```

Figura 47 - Programación gestión de productos Proyecto antiguo - 2

En este caso, y como en los anteriores, se han implementado FB para evitar la repetición de líneas de programación. El FB queda programado de la forma en la que se ve en la siguiente imagen.

```
SEGUIMIENTO_ESTACIONES_TRABAJO
FUNCTION_BLOCK SEGUIMIENTO_ESTACIONES_TRABAJO
VAR_IN_OUT
    MovFrom: BOOL;
    Est: ESTACION_DE_TRABAJO;
END_VAR
VAR_OUTPUT
END_VAR
VAR
    PT_MOVE_E: BOOL;
    PT_MOVE_FLANCO: BOOL;
    PT_TIME_E: BOOL;
    PT_TIME_FLANCO: BOOL;
    ARRAY_TIEMPO_VACIA: ARRAY[1..4] OF TIEMPO;
    RTRIGInst: R_TRIG;
    FTRIGInst: F_TRIG;

PT_MOVE_E := MovFrom;
RTRIGInst( CLK:= PT_MOVE_E );
PT_MOVE_FLANCO := RTRIGInst.Q;

IF PT_MOVE_FLANCO = TRUE THEN
    EST.CONT_BANDEJAS_EST := EST.CONT_BANDEJAS_EST + 1;
END_IF

IF PT_MOVE_FLANCO = TRUE AND EST.ARR_PROD_EST[EST.CONT_BANDEJAS_EST].TIPO_DE_PRODUCTO > 0 THEN
    EST.TRABAJO_ESTACION := TRUE;
END_IF


PT_TIME_E := EST.TIEMPO_ESTACION;
FTRIGInst(CLK := PT_TIME_E);
PT_TIME_FLANCO := FTRIGInst.Q;

IF ( PT_TIME_FLANCO ) = TRUE THEN
    EST.CONT_TIEMPO_EST := EST.CONT_TIEMPO_EST + 1;

    IF EST.CONT_TIEMPO_EST = 5 THEN
        EST.ARR_TIEMPO_EST := ARRAY_TIEMPO_VACIA;
        EST.CONT_TIEMPO_EST := 1;
    END_IF

    EST.ARR_TIEMPO_EST[EST.CONT_TIEMPO_EST].CRONO := EST.TIEMPO_TEMPORIZADOR_ST;
    EST.ARR_TIEMPO_EST[EST.CONT_TIEMPO_EST].ID := EST.ARR_PROD_EST[EST.CONT_BANDEJAS_EST].ID;
    EST.FLANCO_D := TRUE;
END_IF
```

Figura 48 - Programación FB gestión de productos Proyecto nuevo

Como se puede ver, las líneas de programación son idénticas a las del programa base. Pero en este caso, solo se han escrito una vez.

Entonces, en el caso de querer añadir una estación de trabajo, la gestión de productos de las bandejas en esa estación de trabajo se programará simplemente implementando el FB SEGUIMIENTO_ESTACIONES_TRABAJO:

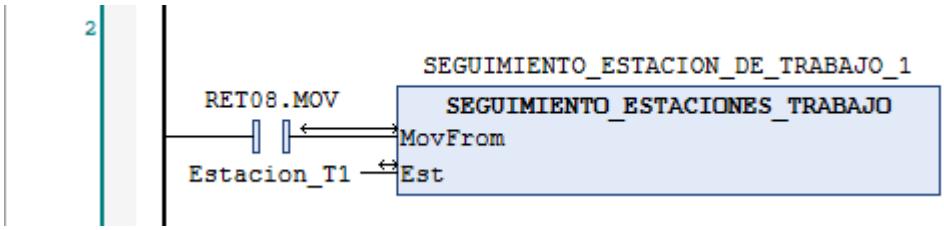


Figura 49 - Aplicación FB gestión de productos Proyecto nuevo

3.3. Carencias

En nuestro programa anterior, se echa en falta algún tipo de visualización o simulación para poder corroborar que nuestro código realmente funciona, sin tener que poner en riesgo la máquina, físicamente hablando.

Todo y que en la mayoría de los softwares hay una herramienta de simulación, ninguna de ellas es lo suficientemente intuitiva como para corroborar el código de una manera fiable. Para resolver esto, se ha creado una visualización que recrea la máquina lo más fielmente posible, con una simbología sencilla.

Esta simbología está ligada a variables del programa, que describen el estado de cada uno de los elementos visualizados.

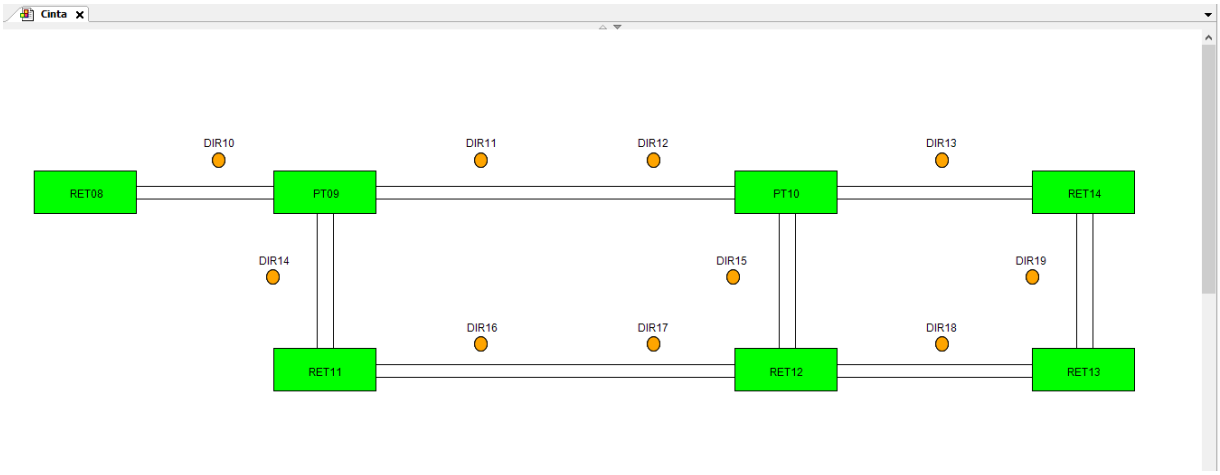


Figura 50 - Visualización del proceso

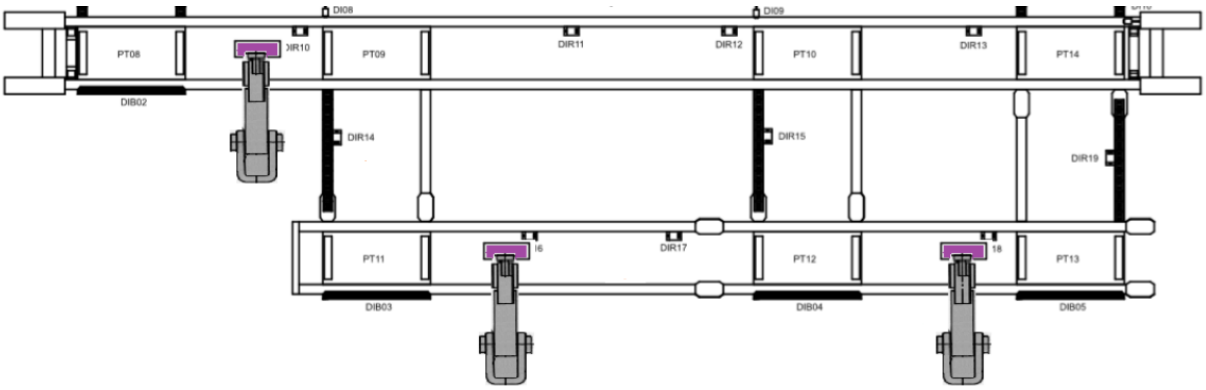


Figura 51 - Esquema del proceso

Como podemos ver, la visualización que se ha creado en el proyecto es una proyección fiel de la máquina, donde se pueden ver los principales elementos del proceso. Los elementos que aparecen en la visualización son los elementos esenciales para poder entender como está funcionando el proceso.

Estos son, los retenedores, las plataformas y los sensores.

Tanto para las Plataformas como para los Retenedores, los estados físicos son los mismos:

Avance, Petición y Reposo. Por lo tanto, se ha creado un tipo de dato para estos estados.

```
VISUALIZACION_RET_AND_PLAT
1   TYPE VISUALIZACION_RET_AND_PLAT :
2   STRUCT
3       Azul:       BOOL; (*AVANCE*)
4       Naranja:    BOOL; (*PETICION*)
5       Verde:      BOOL; (*REPOSO*)
6       NoAzul:     BOOL;
7       NoNaranja:  BOOL;
8       NoVerde:    BOOL;
9   END_STRUCT
10  END_TYPE
11
```

Figura 52 - Tipo de dato visualización Retenedores y Plataformas.

Las variables con nombre de color, azul, naranja y verde hacen referencia a los estados de avance, petición y reposo respectivamente, tanto de plataformas como de retenedores.

La visualización de los retenedores y plataformas, son tres rectángulos de diferente color, superpuestos, los cuales se vuelven invisibles y visibles, en función de sus variables asociadas. Estas variables son aquellas con el prefijo 'No' y el nombre del color, que hacen referencia al estado de invisibilidad del elemento.

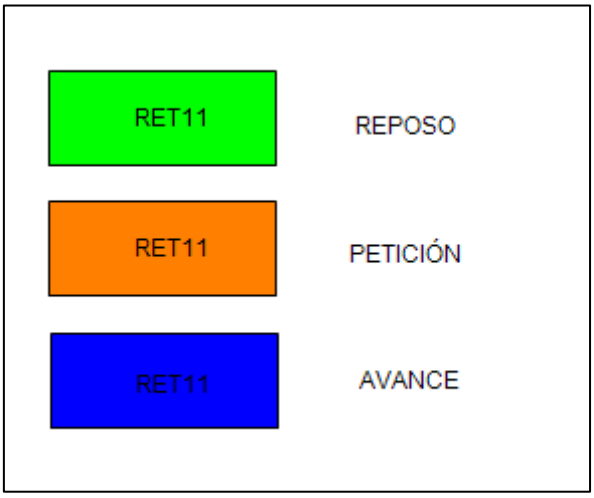


Figura 53 - Leyenda retenedores y plataformas

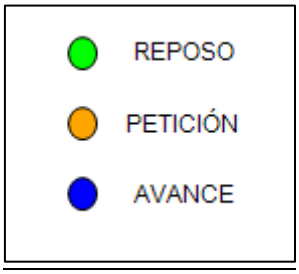


Figura 54 - Leyenda retenedores

3.3.1. VISUALIZACIÓN RETENEDORES

```
FUNCION_VIS_RET
FUNCTION_BLOCK FUNCION_VIS_RET
VAR_INPUT
    Reten: RETENEDOR;
END_VAR
VAR_IN_OUT
    RetVis: VISUALIZACION_RET_AND_PLAT;
END_VAR
VAR
END_VAR

IF Reten.REST AND NOT Reten.MOV AND NOT Reten.READY THEN
    RetVis.Naranja := FALSE;
    RetVis.Azul := FALSE;
    RetVis.Verde := TRUE;
    RetVis.NoNaranja := TRUE;
    RetVis.NoAzul := TRUE;
    RetVis.NoVerde := FALSE;
END_IF

IF Reten.MOV AND NOT Reten.READY AND NOT Reten.REST THEN
    RetVis.Naranja := FALSE;
    RetVis.Azul := TRUE;
    RetVis.Verde := FALSE;
    RetVis.NoNaranja := TRUE;
    RetVis.NoAzul := FALSE;
    RetVis.NoVerde := TRUE;
END_IF

IF Reten.READY AND NOT Reten.REST AND NOT Reten.MOV THEN
    RetVis.Naranja := TRUE;
    RetVis.Azul := FALSE;
    RetVis.Verde := FALSE;
    RetVis.NoNaranja := FALSE;
    RetVis.NoAzul := TRUE;
    RetVis.NoVerde := TRUE;

END_IF
```

Figura 55 - Programación FB para la visualización de los estados de los Retenedores

Con esta FB, se controlarán las variables de visualización de los retenedores, en función de las variables de estado de los mismos.

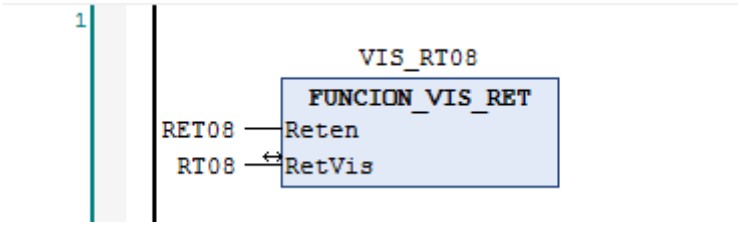


Figura 56 - Aplicación FB para la visualización de los estados de los Retenedores

3.3.2. VISUALIZACIÓN PLATAFORMAS

```
FUNCION_VIS_PT
FUNCTION_BLOCK FUNCION_VIS_PT
VAR_INPUT
    Plat: PLATAFORMA;
END_VAR
VAR_IN_OUT
    PlatVis: VISUALIZACION_RET_AND_PLAT;
END_VAR
VAR
END_VAR

IF Plat.REST AND NOT (Plat.MOV1 OR Plat.MOV2) AND NOT Plat.READY THEN
    PlatVis.Naranja := FALSE;
    PlatVis.Azul := FALSE;
    PlatVis.Verde := TRUE;
    PlatVis.NoNaranja := TRUE;
    PlatVis.NoAzul := TRUE;
    PlatVis.NoVerde := FALSE;
END_IF

IF (Plat.MOV1 OR Plat.MOV2) AND NOT Plat.READY AND NOT Plat.REST THEN
    PlatVis.Naranja := FALSE;
    PlatVis.Azul := TRUE;
    PlatVis.Verde := FALSE;
    PlatVis.NoNaranja := TRUE;
    PlatVis.NoAzul := FALSE;
    PlatVis.NoVerde := TRUE;
END_IF

IF Plat.READY AND NOT Plat.REST AND NOT (Plat.MOV1 OR Plat.MOV2) THEN
    PlatVis.Naranja := TRUE;
    PlatVis.Azul := FALSE;
    PlatVis.Verde := FALSE;
    PlatVis.NoNaranja := FALSE;
    PlatVis.NoAzul := TRUE;
    PlatVis.NoVerde := TRUE;

END_IF
```

Figura 57 - Programación FB para la visualización de los estados de las Plataformas

Con esta FB, se controlarán las variables de visualización de las plataformas, en función de las variables de estado de estas.

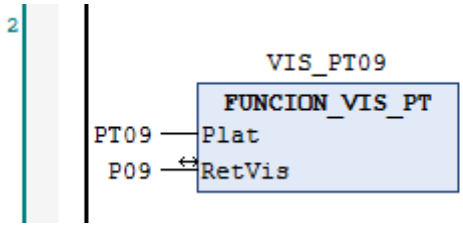


Figura 58 - Aplicación FB para la visualización de los estados de las Plataformas

3.4. Elementos obsoletos.

Como pasa con la mayoría de los elementos electrónicos y eléctricos hoy en día, con el paso del tiempo se van añadiendo mejoras y las más antiguas van siendo sustituidas por las más nuevas.

En nuestro caso, nuestras islas de PLC tienen cabeceras STBNDP2212, los cuales trabajan con el protocolo Profibus DP. La fecha de lanzamiento de este elemento fecha del año 2012.

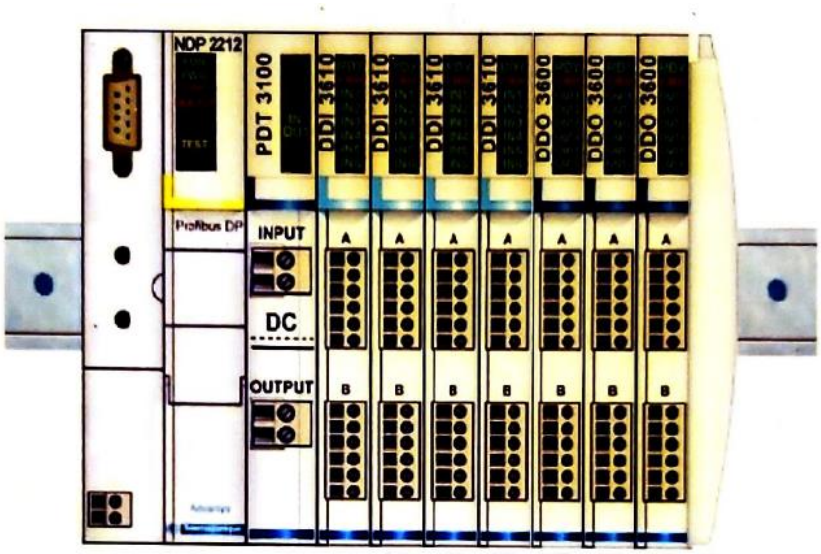


Figura 59 - Isla número 1. Cabecera NDP2212

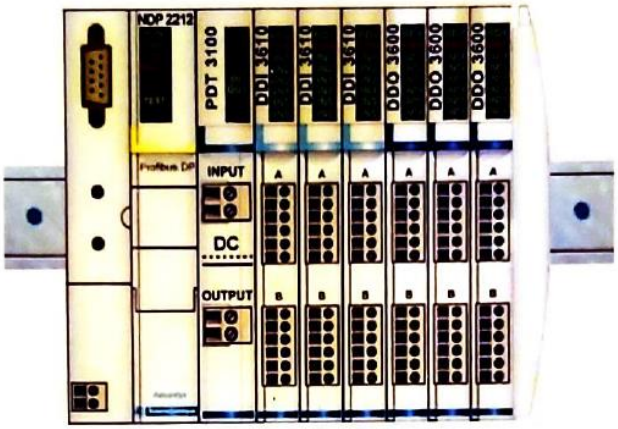
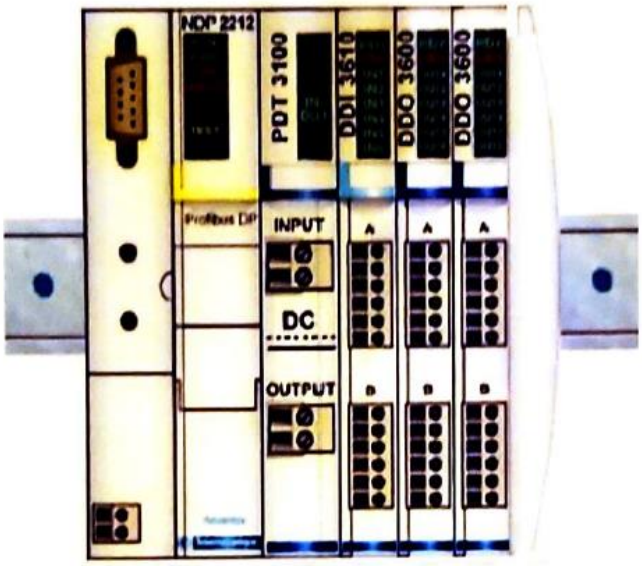


Figura 60 - Isla número 2. Cabecera NDP2212



Isla número 3. Cabecera NDP2212

El reemplazo para las cabeceras antiguas, serán un STBNIP2311, que trabaja con Modbus TCP/IP. La fecha de lanzamiento de este elemento fecha del año 2019.

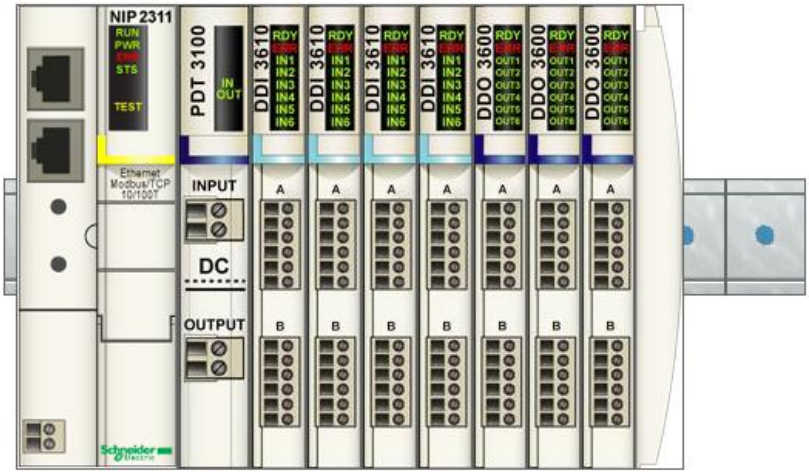


Figura 61 - Isla número 1. Cabecera NIP2311

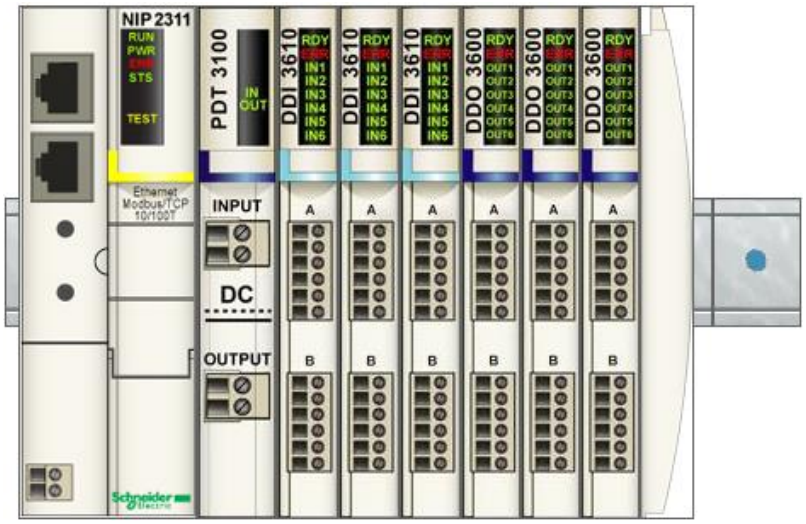


Figura 62 - Isla número 2. Cabecera NIP2311

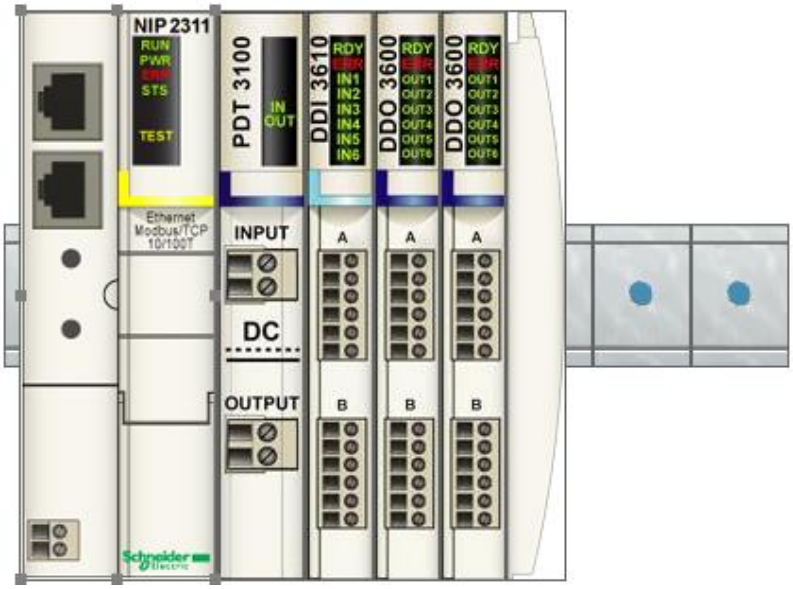


Figura 63 - Isla número 3. Cabecera NIP2311

Una de las diferencias más importantes entre ellos dos, a parte del protocolo de comunicación, es la velocidad en la que estos transmiten los datos.

La velocidad máxima de transmisión de datos para el STBNDP2212 son 12MBits/s. Mientras que la velocidad máxima de transmisión para el STBNIP2311 es de 100MBits/s.

4. Validación

Por la situación actual extraordinaria que nos deja el Covid-19, las pruebas prácticas de la máquina, tales como, comprobación de las comunicaciones, comprobación de las direcciones a los elementos, correcto funcionamiento práctico del código... no han sido posible su comprobación.

Por lo tanto, podremos dar por bueno un correcto funcionamiento del código, en el caso que la simulación teórica salga de forma satisfactoria.

Para una correcta simulación, se deberán añadir temporizadores que simulen el tiempo físico del movimiento de las bandejas. Como también, la información que debería llegar desde la cinta anterior (producto de la bandeja, id ...).

La siguiente línea de programación, simula la entrada de la primera bandeja por el primer retenedor. Dos segundos después del inicio de la simulación, una bandeja entra en el primer retenedor.

Figura 64 - ***Temporizador simula entrada de Bandeja.***

Dentro de los bloques de función de retenedores y plataformas, se añaden dos temporizadores. El primero de ellos, simula el tiempo que pasa desde que sale la bandeja del retenedor en cuestión, hasta que el sensor de este deja de detectarla. El otro, simula el tiempo en que una bandeja tarda en llegar de un retenedor al siguiente.

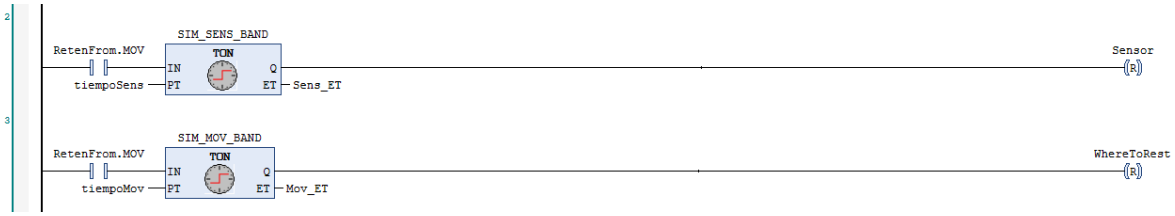


*Figura 65 - **Temporizadores simulan movimiento de Bandeja.***

Por tanto, una simulación satisfactoria será la que, en todo momento haga un seguimiento correcto de las bandejas, y no se produzcan estados imposibles físicamente o incongruencias en los mismos.

Por último, en referencia a la visualización que se ha creado para el seguimiento de los estados de las bandejas, se deberá hacer una simulación de los movimientos de las bandejas, para poder corroborar que la visualización funciona correctamente. Como en la simulación que hablamos anteriormente, será necesario añadir temporizadores que simulen el movimiento de las bandejas y el estado de los sensores.

5. Conclusiones

La realización de este proyecto, obviando los objetivos que han sido imposibles de realizar su comprobación debido a la situación del Covid19, ha sido un éxito tanto en la realización de los objetivos del proyecto, como en los objetivos a nivel personal de crecimiento en este ámbito de la ingeniería de control.

Para empezar, la sustitución del hardware no se pudo implementar físicamente, pero se dejan programadas correctamente todas las islas, tanto en comunicaciones como en estructura. Y aunque no se pudieron programar, sí se pudieron configurar para su uso en un futuro.

En cuanto al software, se ha podido corroborar su buen funcionamiento a través de la simulación del código. Añadiendo las líneas necesarias para la simulación, las pruebas para concluir que la aplicación funciona correctamente, han sido un éxito.

Y dejando de lado las partes técnicas del proyecto, se puede afirmar, que mis conocimientos en la programación de autómatas, ha mejorado en todos los ámbitos. El uso de bloques de función, de estructuras... han hecho que mi forma de programar sea mucho más limpia que antes y me han ayudado a entender un poco más sobre el funcionamiento de la programación en Ladder y ST.