

# **Trabajo Final**

## **AUTOMATIZACIÓN INDUSTRIAL**

*Planta empaquetadora de Cajas Bentō mediante brazo robótico, PLC, interfaz HMI y comunicación TCP/IP*

**Autor del trabajo:** Daniel Béjar Caballero

**Director del trabajo:** Miguel Ángel Montañés Laborda

**Fecha:** Colmenar Viejo, 30 de junio de 2024

 <p><i>estudios abiertos</i> <b>SEAS</b> GRUPO SANVALERO</p>	 <p><i>universidad</i> <b>SANJORGE</b> GRUPO SANVALERO</p>
<i>SEAS, Estudios Abiertos</i>	<i>Universidad San Jorge</i>



# ● Índice

---

● <b>Índice de imágenes .....</b>	<b>6</b>
<b>Índice de tablas .....</b>	<b>8</b>
● <b>Declaración del alumno .....</b>	Error! Bookmark not defined.
● <b>Resumen del trabajo .....</b>	<b>10</b>
1. <b>Palabras Clave.....</b>	<b>11</b>
2. <b>Objeto y objetivos .....</b>	<b>12</b>
3. <b>Alcance .....</b>	<b>13</b>
4. <b>Estado del arte.....</b>	<b>14</b>
5. <b>Funcionamiento del sistema .....</b>	<b>16</b>
6. <b>Convenciones y acrónimos .....</b>	<b>21</b>
7. <b>Descripción formal del sistema mediante GRAFCET .....</b>	<b>22</b>
7.1.     GRAFCET del controlador.....	23
7.1.1.   Lógica del sistema.....	24
7.1.1.1. Detección de llegada de materiales	29
7.1.1.2. Selección de material y mesa según prioridades	31
7.1.2.   Comunicaciones TCP/IP .....	33
7.1.3.   Procesar RX del robot .....	35
7.1.4.   Gestión de errores.....	35
7.1.5.   Actualizar HMI .....	35
7.2.     GRAFCET del robot .....	36
8. <b>Componentes del sistema .....</b>	<b>38</b>
8.1.     PLC SIMATIC S7-1500, CPU 1515-2 PN .....	38
8.1.1.   Especificaciones técnicas.....	38
8.1.2.   Módulos del PLC .....	40
8.1.2.1. Módulo de Alimentación PS 25W 24VDC	40
8.1.2.2. Módulo de entradas digitales DI 32x24VDC BA	42
8.1.2.3. Módulo de salidas digitales DQ 16x24VDC/0.5A BA	43
8.1.3.   Disposición de los módulos .....	44
8.2.     Robot IRB 6700-235/2.65 .....	45
8.2.1.   Controlador del robot IRC5.....	47
8.3.     HMI TP1200 Comfort.....	48
8.4.     Switch EDS-205 .....	49
9. <b>Protocolo de comunicación TCP/IP .....</b>	<b>50</b>
9.1.     Capas del protocolo .....	50
9.2.     Tabla de direcciones de los sistemas .....	52
9.3.     Protocolo PLC-Robot.....	54
9.3.1.   ACK.....	55

9.3.2.	ERROR .....	55
9.3.3.	COLOCA MATERIAL, RECOGIDO, COLOCADO .....	56
9.3.4.	COLOCA CAJA.....	58
9.3.5.	CONEXIÓN COMPLETA, PARÁMETROS.....	59
9.3.6.	ESTADO .....	60
9.3.7.	Ejemplo de intercambio de mensajes .....	61
<b>10.</b>	<b>PLC.....</b>	<b>63</b>
10.1.	Configuración del PLC.....	63
10.2.	Variables simbólicas.....	64
10.2.1.	PLC Tags .....	64
10.2.1.1.	Tabla de tags por defecto (Default tag table)	64
10.2.1.2.	Estados	64
10.2.1.3.	IO Programa	65
10.2.2.	Data Blocks .....	66
10.2.2.1.	Variables del sistema	66
10.2.2.2.	Variables TCP	67
10.2.2.3.	Variables HMI	68
10.2.2.4.	Parámetros de conexión	68
10.3.	Bloques de programa .....	69
<b>11.</b>	<b>Robot.....</b>	<b>70</b>
11.1.	Entorno del robot.....	70
11.2.	El actuador: GarraVentosas .....	73
11.3.	Targets .....	73
11.4.	Entradas y salidas .....	76
11.5.	Objetos inteligentes .....	77
11.5.1.	Ventosas .....	77
11.5.2.	ReponedorPiezas.....	80
11.6.	Código RAPID .....	81
<b>12.</b>	<b>Sistema HMI.....</b>	<b>82</b>
12.1.	Configuración .....	82
12.2.	Variables .....	82
12.3.	Ventanas.....	84
12.3.1.	Pantalla simulación .....	84
12.3.2.	Pantalla real .....	87
12.3.3.	Ventana principal (Root Screen).....	87
12.3.4.	Pantalla configuración .....	87
12.4.	Alarmas .....	88
12.5.	Programación VBA.....	88
<b>13.</b>	<b>Entorno de simulación.....</b>	<b>89</b>
13.1.	Simulación de la planta .....	90
<b>14.</b>	<b>Presupuesto.....</b>	<b>91</b>
<b>15.</b>	<b>Conclusiones.....</b>	<b>92</b>

15.1.	Conclusiones del trabajo .....	92
15.2.	Conclusiones Personales .....	92
<b>16.</b>	<b>Limitaciones y prospectiva.....</b>	<b>93</b>
• <b>Referencias Bibliográficas.....</b>	<b>94</b>	
• <b>Bibliografía.....</b>	<b>95</b>	
• <b>Anexos .....</b>	<b>96</b>	
<b>1.</b>	<b>Anexo 1: Programación del PLC .....</b>	<b>97</b>
1.1.	Selector de Materiales.....	98
1.2.	Conexiones TCP .....	106
1.3.	Procesar mensajes entrantes (TCP).....	109
1.4.	Gestionar errores .....	113
1.5.	Procesar HMI .....	115
1.6.	Funciones extras .....	120
1.6.1.	Crear mensajes salientes (TCP).....	120
1.6.2.	Calcular variables materiales.....	124
1.6.3.	Modificar prioridades .....	125
2.	Anexo 2: Código RAPID .....	127
3.	Anexo 3: Scripts VBA.....	140
3.1.	FuncBotonMesa .....	140
3.2.	IniciarBotonesMesas .....	140
4.	Anexo 4: Túnel de conexión PLC-Robot.....	142
5.	Anexo 5: Presentación.....	145

---

## ● Índice de imágenes

---

FIGURA 1. CAJA BENTO UTILIZADA EN EL TRABAJO.....	16
FIGURA 2. CAJA BENTO UTILIZADA EN EL TRABAJO (SIN TAPA).....	16
FIGURA 3. DIAGRAMA DE LA CAJA BENTO E IDENTIFICACIÓN DE COMPONENTES.....	17
FIGURA 4. DIMENSIONES DE LA CAJA BENTO (EN MILÍMETROS).....	17
FIGURA 5. DETALLE DEL CIERRE HERMÉTICO DE LA CAJA Y DE LA PESTAÑA DE EXTRACCIÓN.....	18
FIGURA 6. DETALLE DE LA CAJA Y LA TAPA CON SUS COMPARTIMENTOS PARA LAS RACIONES.....	19
FIGURA 7. DIAGRAMA DE LA PLANTA COMPLETA.....	20
FIGURA 8. EJEMPLO DE TRANSICIÓN SIN CONDICIÓN.....	22
FIGURA 9. EJEMPLO DE BLOQUE FUNCIONAL.....	22
FIGURA 10. GRACET DEL SISTEMA GENERAL.....	23
FIGURA 11. EJEMPLO DE ENTRADA DE SENsoRES DE PRESENCIA.....	25
FIGURA 12. EJEMPLO DE LISTA DE MESAS.....	26
FIGURA 13. LISTA DE PRIORIDADES PARA EL EJEMPLO ANTERIOR.....	26
FIGURA 14. GRAFCET DE LA LÓGICA DEL CONTROLADOR DE LA PLANTA.....	28
FIGURA 15. ALGORITMO DE SELECCIÓN DE MATERIAL SEGÚN PRIORIDAD.....	31
FIGURA 16. GRAFCET DE CONEXIÓN TCP/IP DEL CONTROLADOR.....	34
FIGURA 17. GRAFCET DEL ROBOT.....	36
FIGURA 18. PLC SIMATIC S7-1500, CPU 1515-2 PN. FUENTE: CATÁLOGO DE SIEMENS.....	38
FIGURA 19. MÓDULO DE ALIMENTACIÓN PS 25W 24VDC. FUENTE: CATÁLOGO DE SIEMENS.....	40
FIGURA 20. MÓDULO DE ENTRADAS DIGITALES DI 32X24VDC BA. FUENTE: CATÁLOGO DE SIEMENS.....	42
FIGURA 21. MÓDULO DE SALIDAS DIGITALES DQ 16X24VDC/0.5A BA. FUENTE: CATÁLOGO DE SIEMENS.....	43
FIGURA 22. DISPOSICIÓN DEL PLC CON SUS MÓDULOS EN EL CARRIL DIN.....	44
FIGURA 23. ROBOT IRB 6700-235/2.65. FUENTE: ABB.....	45
FIGURA 24. RANGO DE TRABAJO DEL ROBOT IRB 6700-235/2.65. FUENTE: ABB.....	46
FIGURA 25. CONTROLADOR IRC5. FUENTE: ABB.....	47
FIGURA 26. PANTALLA HMI TP1200 COMFORT. FUENTE: SIEMENS.....	48
FIGURA 27. SWITCH INDUSTRIAL EDS-205.....	49
FIGURA 28. PROTOCOLO TCP/IP UTILIZADO ENTRE EL PLC Y EL ROBOT (MARCADO EN ROJO). FUENTE: "CPU-CPU COMMUNICATION WITH SIMATIC CONTROLLERS" DE SIEMENS.....	51
FIGURA 29. PROTOCOLO PROFINET UTILIZADO ENTRE EL PLC Y EL HMI. FUENTE: "CPU-CPU COMMUNICATION WITH SIMATIC CONTROLLERS" DE SIEMENS.....	52
FIGURA 30. DIAGRAMA DE CONEXIONADO DE LOS DISPOSITIVOS DE LA PLANTA.....	53
FIGURA 31. TRANSMISIÓN DE MENSAJES PARA "COLOCA MATERIAL".....	56
FIGURA 32. TRANSMISIÓN ANTE FALLO DE ACTUADOR TRAS "COLOCA MATERIAL".....	57

FIGURA 33. TRANSMISIÓN DE MENSAJES PARA "COLOCA CAJA" .....	58
FIGURA 34. ESTABLECIMIENTO DE CONEXIÓN ENTRE PLC Y ROBOT. ....	59
FIGURA 35. INTERCAMBIO DE MENSAJES DE ESTADO Y DESCONEXIÓN. ....	61
FIGURA 36. PLANO GENERAL DEL ENTORNO CON EL ROBOT POSICIONADO EN HOME. ....	70
FIGURA 37. COMPONENTES DEL ENTORNO DEL ROBOT. ....	70
FIGURA 38. EJEMPLO DE GENERACIÓN DE PIEZAS D, E Y F. ....	71
FIGURA 39. ROBOT COLOCANDO MATERIALES EN UNA CAJA. ....	71
FIGURA 40. ESTANDO TODAS LAS RACIONES EN LA CAJA, EL ROBOT SE DIRIGE A RECOGER LA TAPA. ....	72
FIGURA 41. EL ROBOT RECOGE LA CAJA COMPLETA Y LA DEPOSITA EN LA CINTA DE PRECINTADO.....	72
FIGURA 42. ACTUADOR DEL ROBOT: VENTOSA. ....	73
FIGURA 43. TARGETS DEL ROBOT. ....	74
FIGURA 44. CENTROS DE MASA DE LAS PIEZAS. SISTEMA DE REFERENCIA CAJA. ....	74
FIGURA 45. SISTEMA DE COORDENADAS DE LA MESA. ....	75
FIGURA 46. LÓGICA DE LA ESTACIÓN. ....	77
FIGURA 47. VISTA COMPLETA DEL COMPONENTE INTELIGENTE "VENTOSAS". ....	77
FIGURA 48. DETALLE DEL COMPONENTE INTELIGENTE "VENTOSAS". ....	78
FIGURA 49. "LINESENSOR" Y "VOLUMESENSOR" DEL COMPONENTE INTELIGENTE "VENTOSAS". ....	79
FIGURA 50. DETALLE DE LOS DOS "VOLUMESENSOR" PARA DETECTAR LOS MATERIALES D1 Y D2.....	79
FIGURA 51. OR CONJUNTO DE TODOS LOS SENsoRES PARA GENERAR DO_PIEZA. ....	80
FIGURA 52. COMPONENTE INTELIGENTE "REPONEDORPIEZAS". ....	81
FIGURA 53. CONFIGURACIÓN ETHERNET DEL SISTEMA HMI. ....	82
FIGURA 54. PANTALLA SIMULACIÓN, VISTA DESDE TIA PORTAL. ....	84
FIGURA 55. ROBOT EN EL HMI. ....	85
FIGURA 56. CINTAS DE MATERIALES Y CINTA GENERAL EN EL HMI. ....	85
FIGURA 57. PALÉS EN EL HMI. ....	85
FIGURA 58. MESA DESTINO (EN AZUL) MOSTRADA EN EL HMI. ....	86
FIGURA 59. PROGRESO DEL CONTENIDO DE LAS CAJAS POR MESA EN EL HMI. ....	86
FIGURA 60. PANTALLA REAL, VISTA DESDE TIA PORTAL. ....	87
FIGURA 61. PANTALLA DE CONFIGURACIÓN, VISTA DESDE TIA PORTAL. ....	88
FIGURA 62. GUÍA DE SIMULACIÓN, PUNTOS 2 Y 3.....	89
FIGURA 63. GUÍA DE SIMULACIÓN, PUNTOS 4 Y 5.....	89
FIGURA 64. GUÍA DE SIMULACIÓN, PUNTO 6. ....	90
FIGURA 65. GUÍA DE SIMULACIÓN, PUNTO 7 Y 8.....	90
FIGURA 66. CAPTURA DE LA PÁGINA DE GITHUB DEL PROYECTO. ....	96

# Índice de tablas

---

TABLA 1. LISTA DE ACRÓNIMOS Y SIGLAS UTILIZADAS EN EL TRABAJO.....	21
TABLA 2. IDENTIFICADORES DE LOS MATERIALES DE LA CAJA BENTO (EN HEXADECIMAL) .....	25
TABLA 3. LEYENDA DEL ALGORITMO DE SELECCIÓN DE MATERIALES SEGÚN PRIORIDAD.....	32
TABLA 4. IPS DE LOS DISPOSITIVOS DE LA PLANTA. ....	52
TABLA 5. LISTA DE COMANDOS PLC → ROBOT.....	54
TABLA 6. LISTA DE COMANDOS ROBOT → PLC.....	54
TABLA 7. MENSAJES DE ERROR. ....	56
TABLA 8. EJEMPLO DE INTERCAMBIO DE MENSAJES PLC Y ROBOT. ....	61
TABLA 9. EXPLICACIÓN DEL EJEMPLO DE INTERCAMBIO DE MENSAJES.....	62
TABLA 10. CONFIGURACIÓN DEL PLC Y SUS MÓDULOS. ....	63
TABLA 11. TABLA DE TAGS POR DEFECTO.....	64
TABLA 12. TABLA DE TAGS "ESTADOS". ....	65
TABLA 13. TABLA DE TAGS "IO PROGRAMA" .....	65
TABLA 14. PROPIEDADES DE LOS DATA BLOCKS ACCESIBLES POR HMI. ....	66
TABLA 15. VARIABLES DEL SISTEMA EN "SISTEMA_VBES" .....	67
TABLA 16. VARIABLES TCP EN "TCP_VBES" .....	67
TABLA 17. VARIABLES HMI EN "HMI_VBES". ....	68
TABLA 18. PARÁMETROS DE CONEXIÓN EN "DB_PARAMSCONEXION". .....	68
TABLA 19. CENTROS DE MASA DE LAS PIEZAS. ....	76
TABLA 20. TABLA DE ENTRADAS Y SALIDAS DEL ROBOT. ....	76
TABLA 21. VARIABLES COMPARTIDAS DEL HMI CON EL PLC.....	83
TABLA 22. VARIABLES PROPIAS DEL HMI. ....	83
TABLA 23. ALARMAS DEL HMI.....	88
TABLA 24. PRESUPUESTO DEL PROYECTO.....	91
TABLA 25. VARIABLES DEL BLOQUE "SELECCIONARMATERIALES" .....	98
TABLA 26. VARIABLES INTERNAS DE TCP_COMMS. ....	106
TABLA 27. VARIABLES DEL BLOQUE "TCP_COMMS".....	109
TABLA 28. VARIABLES DEL BLOQUE "CALCULARHMI" .....	115
TABLA 29. VARIABLES DEL BLOQUE "TCP_CREARTX".....	121



## ● Resumen del trabajo

---

En el siguiente documento se describe el funcionamiento de un sistema automático de preparación de bandejas de comida, conocido comúnmente en Japón como *Cajas Bentō*.

Estas cajas contendrán por defecto cinco raciones pequeñas, mas este número podrá ser modificado por el usuario. El sistema recibe dichas raciones desde cintas transportadoras individuales con sendos sensores de presencia en sus extremos. Al detectar una de estas raciones, el controlador o PLC decidirá si conducir la ración al brazo robot para colocarla en la caja según unas estrictas condiciones las cuales se explicarán en los siguientes apartados. Una vez esté completa la caja, el brazo robótico la desplazará a una cinta la cual conduce al último paso dentro de la fabricación de las cajas, una máquina empaquetadora.

Se desarrollará en profundidad la programación del PLC “CPU 1515-2 PN” del fabricante *Siemens*, del brazo robótico “IRB 6700” de *ABB*, el diseño del dispositivo HMI “TP1200 Comfort” de *Siemens* y el protocolo utilizado para comunicar todos los componentes entre sí.

Se describirá de igual manera cómo conectar *TIA Portal* y *RobotStudio* para realizar la simulación conjunta del sistema.

## 1. Palabras Clave

- Programación de PLC serie 1500
- Sistema HMI TP1200 Comfort
- Robot IRB6700 de ABB
- TIA Portal
- WinCC Advanced
- RobotStudio
- Conexionado PLC y robot
- Protocolo TCP/IP
- Simulación conjunta de PLC y robot
- Cadena de montaje

## 2. Objeto y objetivos

El trabajo comienza con una descripción formal del sistema además de una explicación del protocolo utilizado para comunicar los diferentes componentes de este.

Se detallará la programación del PLC “CPU 1515-2 PN” del fabricante Siemens mediante el IDE “TIA Portal v17”. Se describirán todas las variables de trabajo y los bloques funcionales del sistema utilizando el propio código como base para las explicaciones.

Sobre el brazo robótico “IRB 6700” de ABB, se hablará sobre su programación mediante el entorno visual y la programación en RAPID mediante el programa “RobotStudio”. Además, se describirán los componentes utilizados para el apartado visual del programa; es decir, para poder movilizar los objetos de la misma manera que lo haría el sistema en la vida real.

En otra sección, se describirá el diseño del dispositivo HMI o SCADA programado en el “TP1200 Comfort” de Siemens, también utilizando el entorno de “TIA Portal v17”. Se hablará sobre las diferentes funciones e integraciones realizadas para poder visualizar de manera rápida y efectiva el estado actual del sistema. Se describirán también las diferentes alarmas que pueden surgir en el funcionamiento del sistema.

También se añadirá una guía para poder comunicar ambos entornos de simulación (TIA Portal y RobotStudio) en un mismo ordenador, sin tener que hacer uso de componentes externos. Esta guía habilitará al lector de este documento para poder verificar el comportamiento descrito en este trabajo.

Finalmente, mediante el uso de capturas de pantalla de los simuladores, se pretenderá demostrar el funcionamiento del sistema completo; es decir, del funcionamiento simultáneo de PLC, robot y HMI.

### 3. Alcance

En las siguientes hojas se describirá exclusivamente el resultado final del proyecto.

No se desarrollarán aquellos pasos que sean naturales del uso de los simuladores; por ejemplo, las acciones para crear un bloque de función en *TIA Portal* o cómo crear un *Target* en *RobotStudio*, puesto que se entiende que el lector tiene ciertos conocimientos del funcionamiento del software; y, en otros aspectos, el autor no desea llenar el documento con hojas que no aporten información sobre el sistema. Esto no quiere decir aun así que aquellos pasos que no sean tan comunes o que sean de suma importancia para el funcionamiento del proyecto no vayan a ser descritos.

No se desarrollará el funcionamiento de los componentes externos al PLC, robot y sistema HMI. Esto engloba las cintas de transporte, los sensores de presencia, el origen de las raciones individuales de comida o el sistema de empaquetado. Se dará una breve y concisa definición de su funcionamiento, pero no se entrará en componentes eléctricos o calificaciones mecánicas. Estos componentes tampoco estarán añadidos a la valoración económica del proyecto.

El proyecto es altamente escalable; es decir, se pueden modificar con facilidad el número de componentes y su disposición en las cajas. Esto está contemplado en la programación del PLC y generará los mensajes de manera correcta. Sin embargo, el robot y el sistema HMI al tener un alto componente visual para poder realizar la simulación, impondrán un límite superior al número de componentes por caja, al máximo número de cajas por mesa y a la posición final de los componentes dentro de la caja.

Pese a que en los anexos se añadirán enlaces para descargar el código fuente de todos los componentes del robot, en el documento de este Trabajo Final de Máster se pretende presentar en un solo medio la totalidad del proyecto. Por lo tanto, gran parte del código, variables usadas y configuraciones serán añadidas a estas hojas en los Anexos. Esto lo hago en consecuencia a la gran cantidad de enlaces de internet que terminan cayendo en el transcurso del tiempo. De no añadir aquí el código, el trabajo quedaría inservible para aquellos posibles lectores del futuro.

## 4. Estado del arte

Para conocer el origen de las cajas bento, hay que remontarse al siglo XII en Japón. Durante este periodo, se cree que apareció el concepto inicial de bento. Los viajeros y cazadores llevaban "*hoshi-i*" (arroz seco cocido) que podía ser comido tal cual o rehidratado con agua.

En el siglo XVI comienzan a aparecer las cajas tradicionales, echas de madera lacada y vendidas normalmente en casas de té. La estética y presentación de las cajas y de las porciones de comida dentro de ellas eran esenciales: mucho esmero se tomaba en decorar las raciones, haciendo muchas veces uso de distintos alimentos con distintivos visuales para formar figuras de animales u objetos cotidianos.

Durante el siglo XVII y XIX se siguen popularizando las cajas bento, haciéndose muy conocidas a finales del siglo XIX las "*ekiben*", unas cajas bento que eran vendidas en las estaciones de tren, normalmente llenas de platos típicos de la zona y que los pasajeros podían consumir dentro del vagón.

Con la expansión del tren y la modernización de Japón durante el siglo XX, las cajas bento se convirtieron en elementos esenciales para los viajeros y la fabricación de ellas se profesionalizó, apareciendo las primeras tiendas especializadas. Posterior a la Segunda Guerra Mundial, toma gran importancia la industrialización y la producción en masa de las cajas bento. Para ello, se utilizan maquinarias avanzadas y procesos automatizados para conseguir alta eficiencia y uniformidad entre cajas. Comienzan a utilizarse aluminio y plásticos para las bases; y, recientemente, materiales ecológicos a base de bambú. El proceso debe además cumplirse en un entorno estéril para garantizar el mayor grado de salubridad posible.

Con esta breve introducción, el propósito de este trabajo es realizar un dispositivo para agilizar la fabricación de cajas bento en un entorno industrial. Tomando inspiración del método JIT (*Just In Time*), el sistema únicamente funcionará cuando disponga de los materiales necesarios para fabricar una caja. La planta podrá cierto número de cajas en paralelo (a petición del usuario), acelerando el proceso de fabricación y reduciendo la dependencia de la planta con el resto de los sistemas.

La planta objeto de este trabajo puede empaquetar cajas bento a una velocidad superior a la de sistemas tradicionales, aumentando la productividad y reduciendo costos laborales.

El sistema permite ajustes rápidos para diferentes diseños de cajas bento, una ventaja significativa, especialmente en un mercado que demanda variedad y personalización.

El proceso de selección y paralelizado en la fabricación de las cajas garantiza el menor tiempo de exposición a agentes externos, luego dispondrá de los más altos estándares de higiene y presentación.

La interfaz de usuario es amigable y no requiere entrenamiento extensivo para hacerla funcionar. Esto reducirá costos y errores operativos.

El sistema es en sí altamente escalable, luego podrá ser adaptado a infinidad de entornos con un amplio grado de personalización, tanto en el diseño como en las velocidades de fabricación.

## 5. Funcionamiento del sistema

En esta sección se describirá de manera más concisa el funcionamiento del sistema y sus diversos componentes. La planta empaquetará cajas bento de cinco raciones como las que se muestran en la Figura 1 y Figura 2.



Figura 1. Caja bento utilizada en el trabajo.

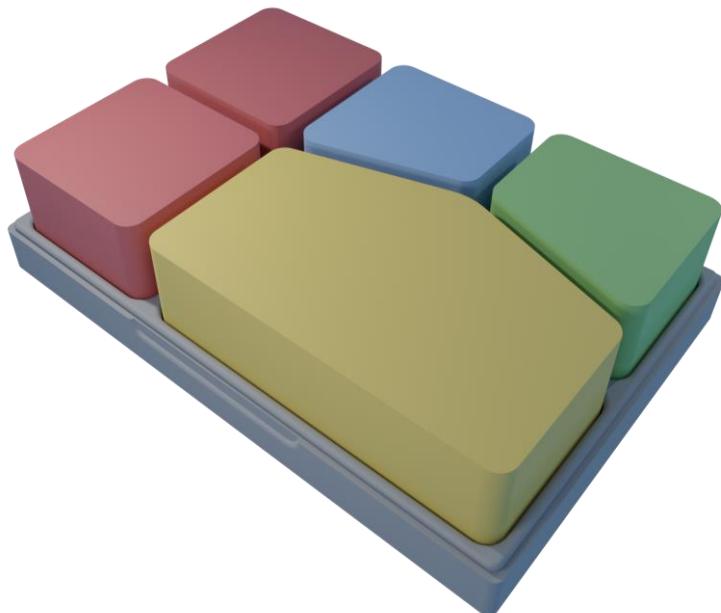
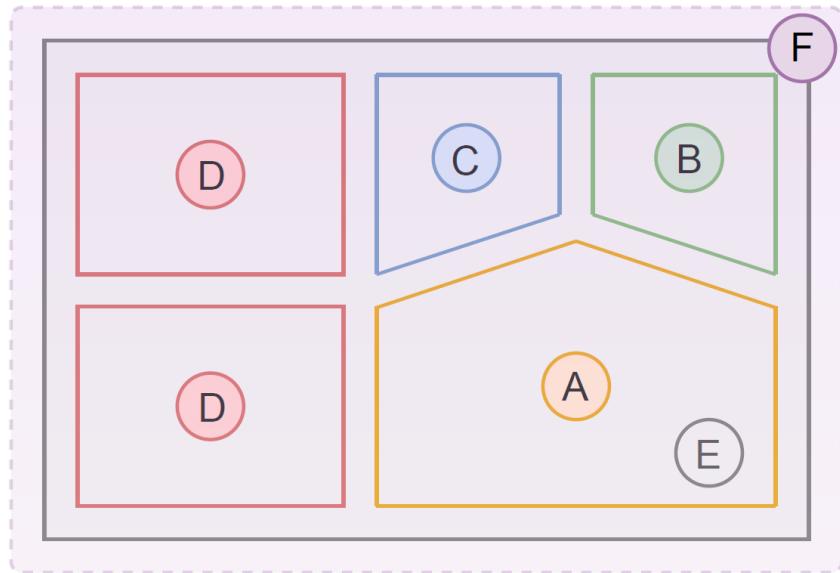


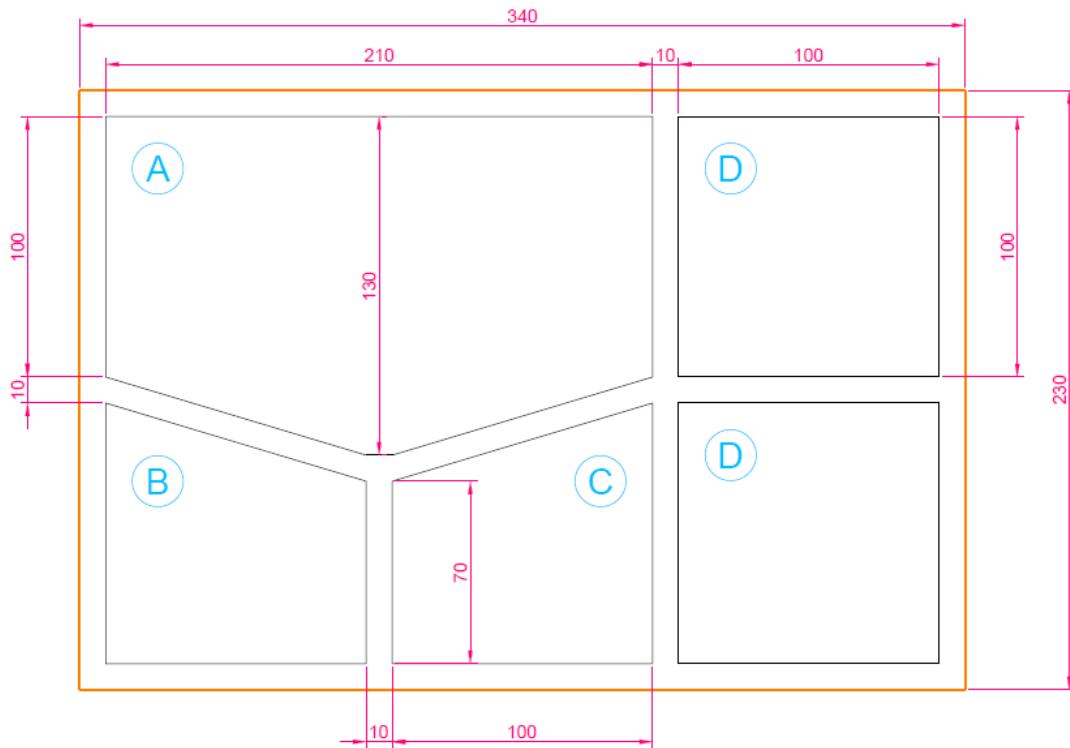
Figura 2. Caja bento utilizada en el trabajo (sin tapa).

Para una fácil identificación de los componentes, los colores utilizados en estos *renders* serán los mismos que se utilizarán en los simuladores y sistema HMI. En la figura inferior se muestra un diagrama con las posiciones de cada componente.



**Figura 3. Diagrama de la caja bento e identificación de componentes.**

Las dimensiones de la caja se muestran en la siguiente figura.



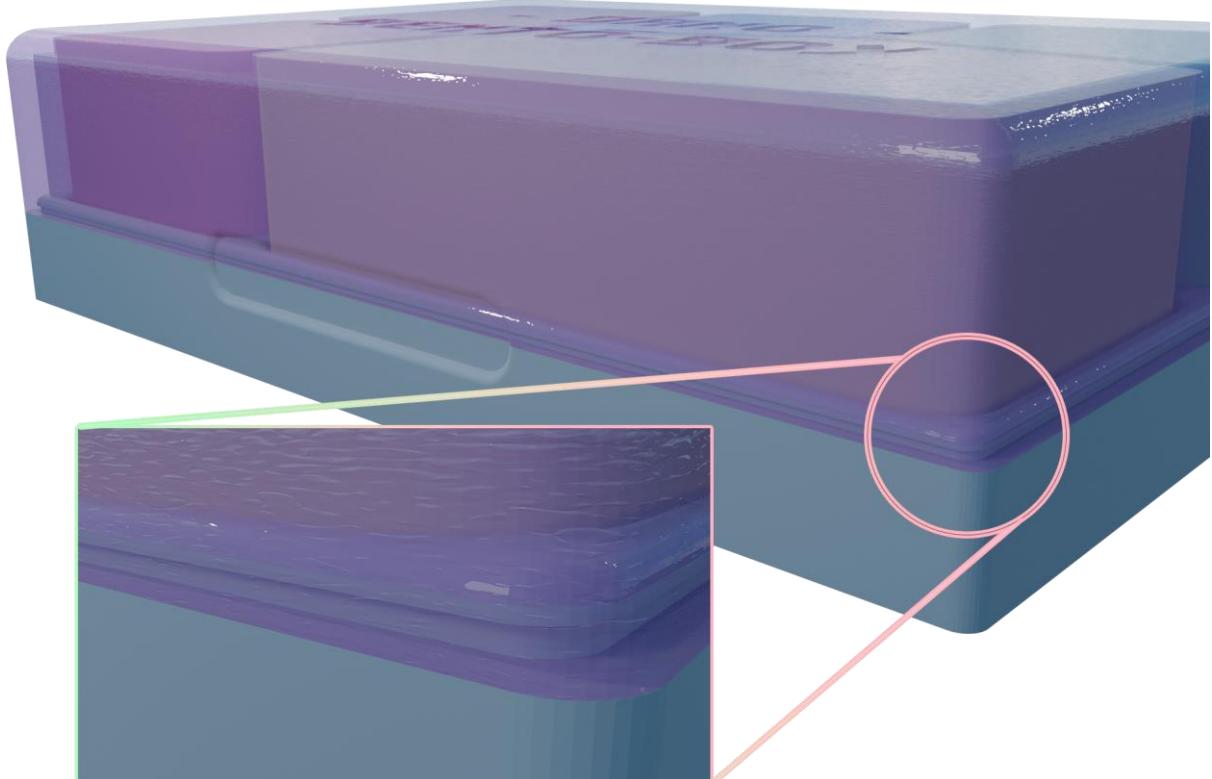
**Figura 4. Dimensiones de la caja bento (en milímetros).**

Las raciones de comida (componentes A, B, C y D) llegarán a la empaquetadora desde cintas transportadoras individuales y de manera interrumpida; es decir, el suministro de raciones no será constante. El PLC escogerá qué componente llevar al brazo robótico mediante una cinta general a la cual desembocan las otras cintas individuales.

El brazo robótico será el encargado del montaje de las cajas. Para ello, deberá recoger primero la caja o base de la caja bento de un palé (componente E, en gris). Después colocará las raciones de comida que serán entregadas por la cinta general. Finalmente, con todas las raciones colocadas, se cerrará la caja mediante la tapadera (componente F, en púrpura) que estará en un palé similar al del componente E. La caja es hermética, luego cuando se presione la tapa, quedará sellada. Esto garantizará un manejo seguro de la caja una vez cerrada.

El canto superior de la base está biselado para que la tapa tenga fácil acceso. El material plástico de la tapa se deformará para poder encajar con la base. El canto sin embargo tiene un ángulo recto en la parte inferior que impedirá que la tapa se pueda extraer. Para facilitar al consumidor la extracción de la tapa, se añade una pestaña que, bajo un poco de presión, permitirá removerla.

Se puede ver el detalle del cierre en la imagen inferior.



**Figura 5. Detalle del cierre hermético de la caja y de la pestaña de extracción.**

El robot dispone de una mesa donde realizar el montaje de varias cajas a la vez. Debido a la existencia irregular de las raciones, el sistema podrá realizar varias cajas en paralelo. El sistema **priorizará aquellas cajas que lleven más tiempo abiertas**; es decir, seguirá un orden FIFO (*First In First Out*).

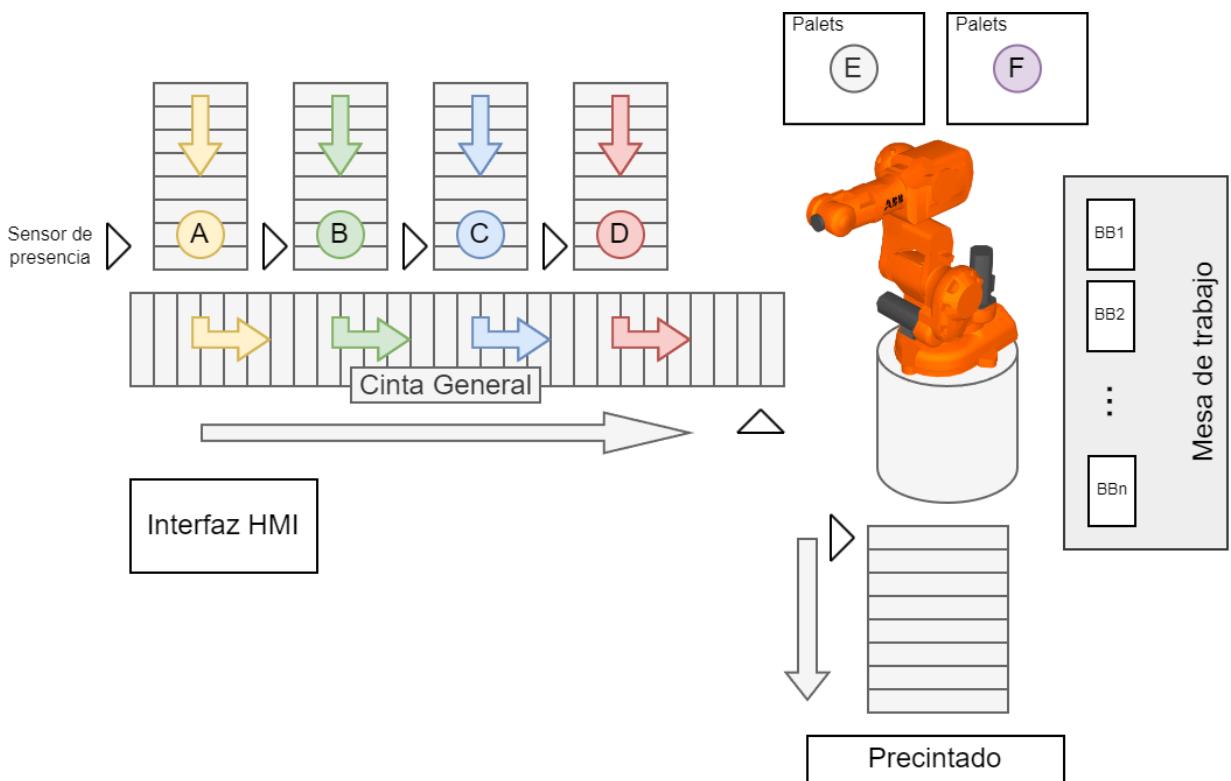
Para garantizar el posicionamiento de las raciones dentro de las cajas y su estabilidad durante el transporte, la base y la tapa traen unas hendiduras que sujetará el contenido de la caja.



**Figura 6. Detalle de la caja y la tapa con sus compartimentos para las raciones.**

Una vez terminada la caja y esté sellada con la tapadera, será transportada a una cinta que llevará a una estación de precintado (no desarrollada en este trabajo). La planta enviará un pulso de cinco segundos a la cinta para indicar a la estación de precintado que tiene una caja pendiente de terminar.

En la figura siguiente se muestra un diagrama del entorno de trabajo del sistema.



**Figura 7. Diagrama de la planta completa.**

Caben destacar los siguientes componentes:

- Las **cintas de transporte** A, B, C y D. Cada cinta tiene una señal de entrada que hace que avance (hacia abajo, según el diagrama). Cada cinta tiene en su parte inferior un **sensor de presencia** que detecta que hay un material sobre ella.
- La **cinta general**. Es una cinta que conecta las cintas de transporte individuales con el robot. Tiene también una señal de entrada que la hace avanzar (hacia la derecha, según el diagrama). Dispone de un **sensor de presencia** que detecta que el material ha llegado a la posición óptima para que el robot la recoja.
- Los **palets** de E y F. Cuentan también con un sensor de presencia que detecta si hay material o no.
- La **mesa de trabajo** donde el robot irá completando las cajas bento.
- Las cajas terminadas acabarán en la **cinta de precintado**. Esta dispone de un sensor de presencia que confirma que hay una caja sobre ella y una señal de entrada que la hace avanzar (hacia abajo, según diagrama).
- La **interfaz HMI** servirá para monitorizar el sistema.

## 6. Convenciones y acrónimos

Una vez descrito el funcionamiento general del sistema, es necesario indicar que a lo largo del trabajo se han tomado las siguientes convenciones en el lenguaje:

- Una **mesa** hace referencia a una posición dentro de la mesa de trabajo. Por ejemplo, si se dice: el material A debe ir a la mesa 2, se refiere a que dicho material debe ir a la caja que se encuentra en la posición 2 de la mesa. En la Sección 11: Robot se describe con detalle cuáles son estas posiciones.
- Un **componente** o **material** hace referencia a lo mismo: es una parte de la caja bento.
- El material D está duplicado, luego para diferenciar ambos D se recurre a su posición: D1 es el material D superior, y D2 el inferior.
- El material E se denominará **base** o **caja**, además de material E.
- El material F se denominará **tapa** o **tapadera**, además de material F.

En la siguiente tabla se recogen algunos de las siglas y acrónimos que se podrán encontrar a lo largo del trabajo.

<b>RX</b>	Recepción de mensajes por TCP/IP
<b>TX</b>	Transmisión de mensajes por TCP/IP
<b>SP</b>	Sensor de Presencia
<b>DB</b>	Data Block (TIA Portal)
<b>NW</b>	Network (TIA Portal)

**Tabla 1. Lista de acrónimos y siglas utilizadas en el trabajo.**

## 7. Descripción formal del sistema mediante GRAFCET

Una vez descrito en grandes rasgos la planta, se procede a desarrollar un GRAFCET del proceso. Comenzando desde el más alto nivel, con funciones abstractas y genéricas, se irá descendiendo en los siguientes apartados en las funciones individuales donde se ahondará en mayor detalle.

**El diseño GRAFCET de esta sección es par al software del PLC y del robot, si bien ciertas simplificaciones han sido tomadas. Puede leer más sobre ello en la Sección 0: Tabla 8.**

**Ejemplo de intercambio de mensajes PLC y robot.**

1	30040M01	COLOCA MATERIAL (TAPA, MESA 1)
2	0ACK3	ACK
3	2MAT_REC	MATERIAL RECOGIDO (TAPA)
4	3MAT_COL	MATERIAL COLOCADO (TAPA)
5	4M01	COLOCA CAJA
6	0ACK4	ACK
7	2STA	ESTADO
8	3MAT_COL	MATERIAL COLOCADO
9	0ACK2	ACK (del mensaje 7)
10	30020M02	COLOCA MATERIAL (BASE, MESA 2)
11	0ACK3	ACK
12	2MAT_REC	MATERIAL RECOGIDO (BASE)
13	3MAT_COL	MATERIAL COLOCADO (BASE)

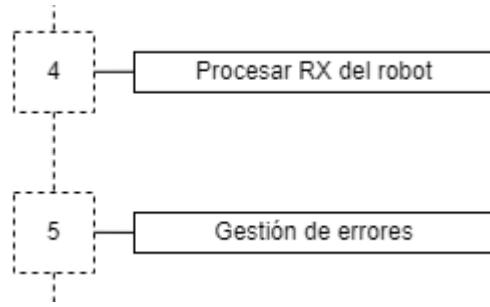
**Tabla 9. Explicación del ejemplo de intercambio de mensajes.**

En este ejemplo se está terminado la caja de la mesa 1. El PLC manda colocar la tapa y, una vez cerrada la caja, manda colocarla en la cinta de precintado. Una vez la caja ha sido colocada, el robot comienza a colocar materiales en una nueva caja en una mesa diferente. El primer material para colocar es la base.

PLC y Sección 11: Robot.

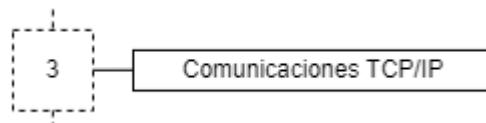
Se ha de advertir también que se han tomado ciertas libertades a la hora de realizar los diagramas GRAFCET ya que estos suelen ser útiles para sistemas secuenciales y, en este programa, hay un alto nivel de paralelismo de tareas. Por ejemplo, el sistema HMI se ha de actualizar constantemente al igual que hay que revisar que el PLC y el robot se mantengan conectados. Además, hay que tener en cuenta las señales asíncronas que pueden surgir ya sea por un funcionamiento anormal del sistema o por indicación del usuario a través del HMI.

Se ha tomado por ello la siguiente convención, una línea rallada no necesitará condición para saltar de un estado a otro y únicamente representa una acción secuencial de bloques que pueden o no estar relacionados entre sí.



**Figura 8. Ejemplo de transición sin condición.**

Los estados rallados no representarán estados del sistema, sino que serán bloques de funciones que tendrán una acción correspondiente, descrita en una caja en el lateral derecho, similar al de una acción en un estado convencional de GRAFCET.



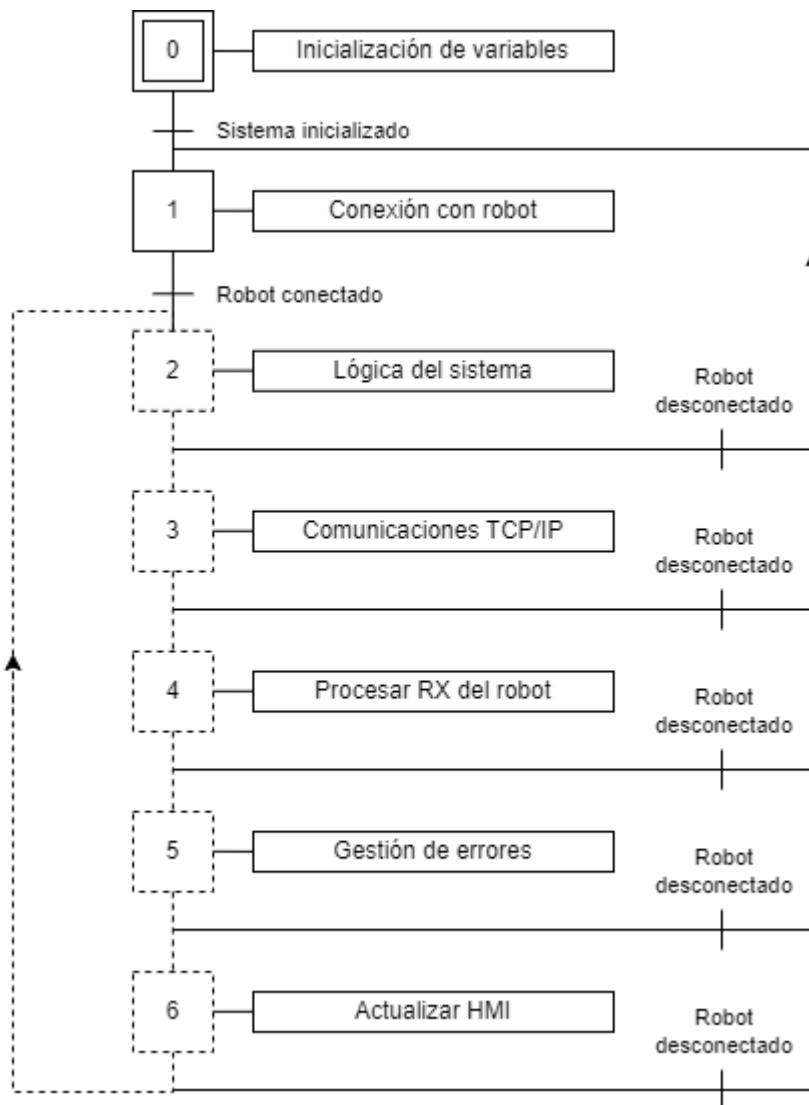
**Figura 9. Ejemplo de bloque funcional.**

Tanto el PLC como el robot no pueden realizar acciones en paralelo, únicamente habrá un hilo de ejecución. Sin embargo, se puede lograr cierto paralelismo si se aíslan las distintas funciones y se suceden sus ejecuciones en orden. Es por ello por lo que se han añadido estas dos convenciones.

Comencemos entonces con la descripción formal del sistema tras este preámbulo.

## 7.1. GRAFCET del controlador

En la siguiente figura se muestra el diagrama de funciones del controlador o PLC.

**Figura 10. GRACET del sistema general.**

El controlador comienza inicializando las variables de trabajo (más sobre ellas en los siguientes apartados) y esperando a la conexión del robot. Una vez está conectado comienza a ejecutar los bloques funcionales:

- La **Lógica del sistema** se encarga de seleccionar qué objetos escoger de las cintas transportadoras o los palés y a qué posición de la mesa de trabajo enviarla. Se encarga de generar los mensajes que serán enviados por TCP al robot en el siguiente bloque funcional.
- El bloque de **Comunicaciones TCP/IP** se encarga de establecer la conexión con el robot, detectar la desconexión de este, obtener los mensajes entrantes del robot y enviar los mensajes salientes hacia el robot. El Estado 1 de la figura donde se conecta con el robot está en verdad dentro de este bloque, pero se ha decidido separarlo para que el esquema se asemeje más al comportamiento real del sistema.

- De haberse recibido algún mensaje del robot, este será procesado en el bloque de **Procesar RX del robot**. En este bloque se comprueba que el mensaje recibido es correcto y actualiza el estado de la lógica del sistema. También se encarga de mandar con cierta periodicidad un mensaje para comprobar que el robot sigue conectado al otro extremo.
- En **Gestión de errores** se comprueban las condiciones necesarias para que el sistema entre en estado de fallo y, de la misma manera, decide si se cumplen las condiciones necesarias para salir de este. Si el sistema está en estado de error, se bloquea la ejecución el bloque “lógica del sistema”.
- En el bloque **Actualizar HMI** se generan las variables necesarias para la correcta visualización de los componentes que forman el sistema HMI.

Una vez el bloque “Actualizar HMI” ha sido ejecutado, vuelve a ejecutar “Lógica del sistema” y así en bucle.

Si el robot se desconectara en alguno de los pasos intermedios, el sistema volvería al estado 1 para volver a establecer la conexión. El estado del robot se podría comprobar desde el HMI en cualquier momento.

### 7.1.1. Lógica del sistema

Para hablar de la lógica del sistema es necesario presentar antes el algoritmo que se utiliza en el controlador para escoger los materiales y las mesas según prioridad.

Una caja bento queda definida por un número de 16 bits (o *Word* en *TIA Portal*). Cada posición o bit representa la ausencia (0) o presencia (1) de un material en la caja. El número de materiales por caja puede ser modificada por el usuario, siendo el máximo permitido 14 componentes. Este límite es así debido a que se reservan los dos últimos bits para la base (componente E) y la tapa (componente F).

En el caso de la caja bento para la cual fue diseñada este trabajo, se pueden comprobar los bits de cada componente en la tabla siguiente.

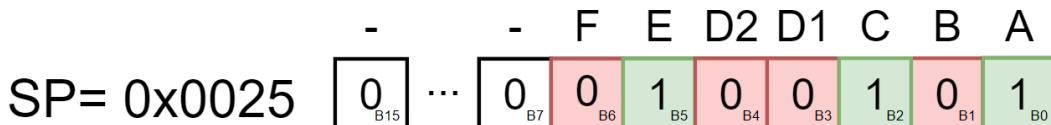
Material A: 0x0001
Material B: 0x0002
Material C: 0x0004
Material D1: 0x0008 (D superior)

	Material D2: 0x0010 (D inferior)
	Material E: 0x0020
	Material F: 0x0040

**Tabla 2. Identificadores de los materiales de la caja bento (en hexadecimal).**

No se debe confundir el tipo de material con la posición del material en la caja. Al controlador en este caso no le importa qué tipo de material sea, sino la posición de este dentro de la caja.

Estos identificadores, además de hacer referencia al material dentro de una caja, también servirán para referenciar los sensores de presencia de las cintas y sus respectivas señales de avance. Por ejemplo, si la variable de entrada de los sensores de presencia de las cintas (SP) fuera 0x0025, esto indica que hay material en las cintas A, C y en el palé E.

**Figura 11. Ejemplo de entrada de sensores de presencia.**

El caso del material D es particular puesto que este proviene de la misma cinta y sin embargo tiene dos identificadores distintos. Para solventar de manera sencilla este problema, el controlador se abstrae de la siguiente manera: los sensores de presencia de la cinta D (que son entradas) y las señales de avance de la cinta transportadora (que son salidas) se conectan de manera duplicada en el PLC.

De esta manera se puede decir que hay un total de siete componentes diferentes que deben de colocarse en una caja bento para completarla.

El número máximo de materiales con los que puede trabajar el sistema son modificables por el usuario. Sin embargo, todas las cajas deben componerse de mínimo una base o caja y una tapa. Por ello, los materiales E y F son especiales porque su identificador no será fijo, sino que su bit se adaptará al número de raciones por bandeja. **E tendrá su bit en el número de raciones, y F en el número de raciones + 1.**

Continuando con la lógica del sistema, dentro del PLC se encuentran dos variables fundamentales:

- **Mesa[ ]**. Es un array de dieciséis *words* donde se almacenan los identificadores de las cajas bento. El índice del array identifica la posición de la caja bento dentro de la mesa.

	-	-	F	E	D2	D1	C	B	A	
Mesa[0] = 0x0025	0 B15	...	0 B7	0 B6	1 B5	0 B4	0 B3	1 B2	0 B1	1 B0
Mesa[1] = 0x0024	0 B15	...	0 B7	0 B6	1 B5	0 B4	0 B3	1 B2	0 B1	0 B0
Mesa[2] = 0x0035	0 B15	...	0 B7	0 B6	1 B5	1 B4	0 B3	1 B2	0 B1	1 B0
Mesa[3] = 0x003F	0 B15	...	0 B7	0 B6	1 B5	1 B4	1 B3	1 B2	1 B1	1 B0
⋮	⋮	⋮								

Figura 12. Ejemplo de lista de mesas.

- **Prioridad[ ]**. Es un array de dieciséis números enteros (tantos como posibles posiciones en la mesa hay). Cada posición almacena el número de una mesa. A menor sea el índice del array, más prioritaria será la mesa. Recuérdese que **la prioridad del sistema es cerrar aquellas cajas que más tiempo lleven abiertas**. Para el ejemplo de la figura anterior, la lista de prioridad sería:

Prioridad[0]	3
Prioridad[1]	2
Prioridad[2]	0
Prioridad[3]	1
⋮	⋮

Figura 13. Lista de prioridades para el ejemplo anterior.

En este ejemplo, puede calcularse la prioridad de cada una de las mesas según el número de materiales que tienen. Sin embargo, este no siempre será el caso: si varias mesas

tienen el mismo número de materiales, será necesario diferenciarlas según el tiempo que lleven abiertas. Es por ello por lo que se utiliza un array de prioridades.

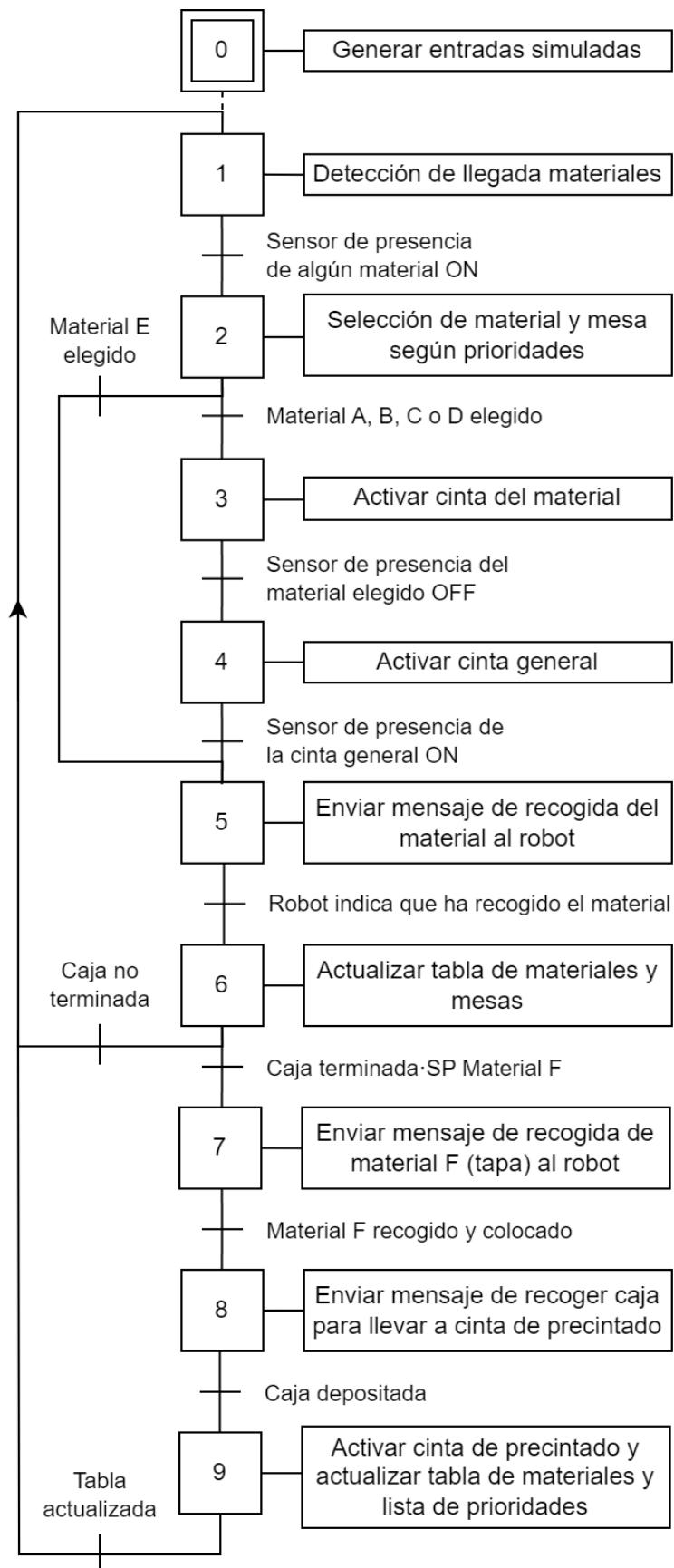
En este ejemplo se observa que la Mesa[3] tiene todas las raciones y que únicamente necesita ser cerrada con la tapa (F) para que la caja esté completa. Una vez la caja sea colocada en la cinta empaquetadora, Mesa[3] pasará a valer cero, puesto que no quedará ningún componente en esa posición de la mesa, y dejará de ser la mesa prioritaria.

El array Prioridad[] será por lo tanto desplazado hacia arriba; es decir, el valor de Prioridad[1] pasará a Prioridad[0], el valor almacenado en Prioridad[2] pasará a estar en Prioridad[1], y así hasta llegar a Prioridad[15], que pasará a estar en Prioridad[14]. En la posición 15 del array se escribirá la mesa que almacenaba anteriormente Prioridad[0].

Cuando se inicializa el sistema, el contenido de todas las posiciones de Mesa[ ] es 0, y en Prioridad[ ] se coloca el número del índice dentro del array ([0] = 0, [1] = 1...).

Con estas variables, ya se puede proceder a la descripción del sistema.

El GRAFCET se encuentra en la siguiente página.



**Figura 14. GRAFCET de la lógica del controlador de la planta.**

Algunas notas sobre el esquema:

- Las indicaciones del robot de que un material ha sido recogido o depositado las genera el bloque funcional “Procesar RX del robot”.
- En la **etapa 0** se crean las entradas simuladas. Esto es para poder realizar las simulaciones sin tener las entradas digitales correspondientes para ello.
- En la **etapa 5, 7 y 8** no se envía la orden al robot como tal, sino que se prepara el mensaje y se indica al bloque funcional de “Comunicaciones TCP/PI” que debe enviar el buffer TX.
- Para actualizar la tabla de materiales en el **estado 6**, simplemente se añade el material que se ha colocado en la posición “mesa” (calculada en la etapa 2) del array Mesa[ ] mediante una operación OR. Sin embargo, **este cambio puede ser puntual si a mitad del proceso el robot manda una señal de error**, por ello, se crean sendas copias de la mesa y el material colocados por si se tiene que deshacer este paso. Cuando llega la confirmación de que el objeto ha sido colocado de manera asíncrona, se ponen a cero ambas copias, indicando que el cambio es final.
- La activación de la cinta de precintado durante cinco segundos de la **etapa 9** se realiza de manera asíncrona. No es un estado en el que se realice la espera, sino que se toma el tiempo del sistema, se le suman cinco segundos y se guarda en una variable. Una comprobación en esta etapa comprueba si el tiempo del sistema es mayor o menor que dicha variable. En el primer caso la cinta estará apagada; y en el segundo, encendida.

En los siguientes subapartados se profundizará un poco más en algunos estados que necesitan mayor aclaración.

#### 7.1.1.1. Detección de llegada de materiales

Los sensores de presencia del sistema están conectados siguiendo el mismo orden que el de los materiales, como se vio en un ejemplo anterior. De esta manera, el sistema recibe los valores de los sensores en una sola variable de entrada tipo *word* que se llamará **SP\_Cintas**.

Es muy útil poder trabajar con palabras puesto que esto permite enmascarar las entradas; es decir, permite “ignorar” las entradas mediante la aplicación de una **máscara**. En una máscara, si un bit es 0, indica que el mismo bit de SP\_Cintas debe ser ignorado. Si el bit 1, el caso es el contrario.

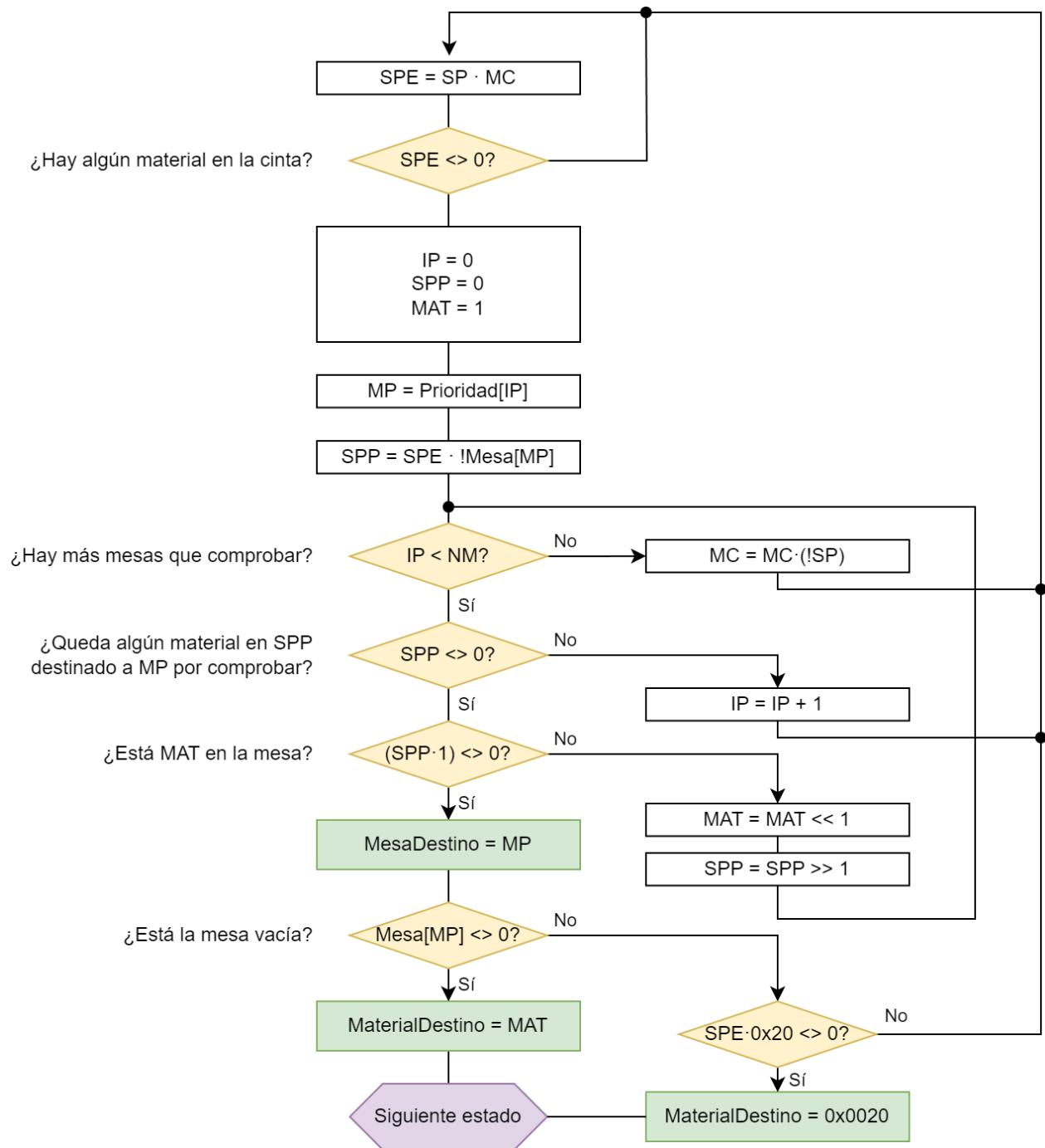
Para aplicar una máscara, simplemente hay que realizar una operación AND entre la máscara y SP\_Cintas.

Para demostrar el uso de una máscara se presenta el siguiente ejemplo: suponiendo que solo hay un sensor activo y que todas las cajas ya tienen colocado ese material, no interesa seguir atendiendo a la lectura de ese sensor en particular. Por ello, se enmascara el sensor; y solo se liberará cuando alguna de las cajas haya sido completada.

Si tras aplicar la máscara a SP\_Cintas, el valor resultante es distinto de cero, indica que hay algún material que puede ser colocado en alguna mesa, y por lo tanto, se procede al siguiente estado.

### 7.1.1.2. Selección de material y mesa según prioridades

A continuación, se presenta el algoritmo de selección de material según prioridades.



**Figura 15. Algoritmo de selección de material según prioridad.**

En la Tabla 3 se muestra la leyenda para este esquema.

Símbolo	Significado	Explicación
<b>IP</b>	Índice de Prioridad	Con este índice se accede a la matriz de prioridades. Es un número entero que incrementa desde el 0 hasta NM-1, inc.
<b>SP</b>	Sensores de Presencia	Entradas de los sensores de presencia, en formato word.
<b>SPE</b>	Sensores de Presencia Enmascarados	Operación AND de SP y MC. Son los sensores que son necesarios para alguna de las mesas.
<b>SPP</b>	Sensor de Presencia Prioritario	Son el conjunto de sensores necesarios para terminar la mesa MP. Si es 0, significa que no hay materiales para MP.
<b>MP</b>	Mesa Prioritaria	El número de la mesa que se está analizando.
<b>NM</b>	Número de Materiales	Número total de materiales por bandeja. Valor seleccionado por el usuario. No incluye ni la base ni la tapa.
<b>MAT</b>	Material	Es una variable temporal que almacena el ID de un material.
<b>MC</b>	Máscara de Cintas	Máscara con las cintas que están habilitadas. Su valor inicial es $2^{NM}-1$ (ningún material enmascarado).

**Tabla 3. Leyenda del algoritmo de selección de materiales según prioridad.**

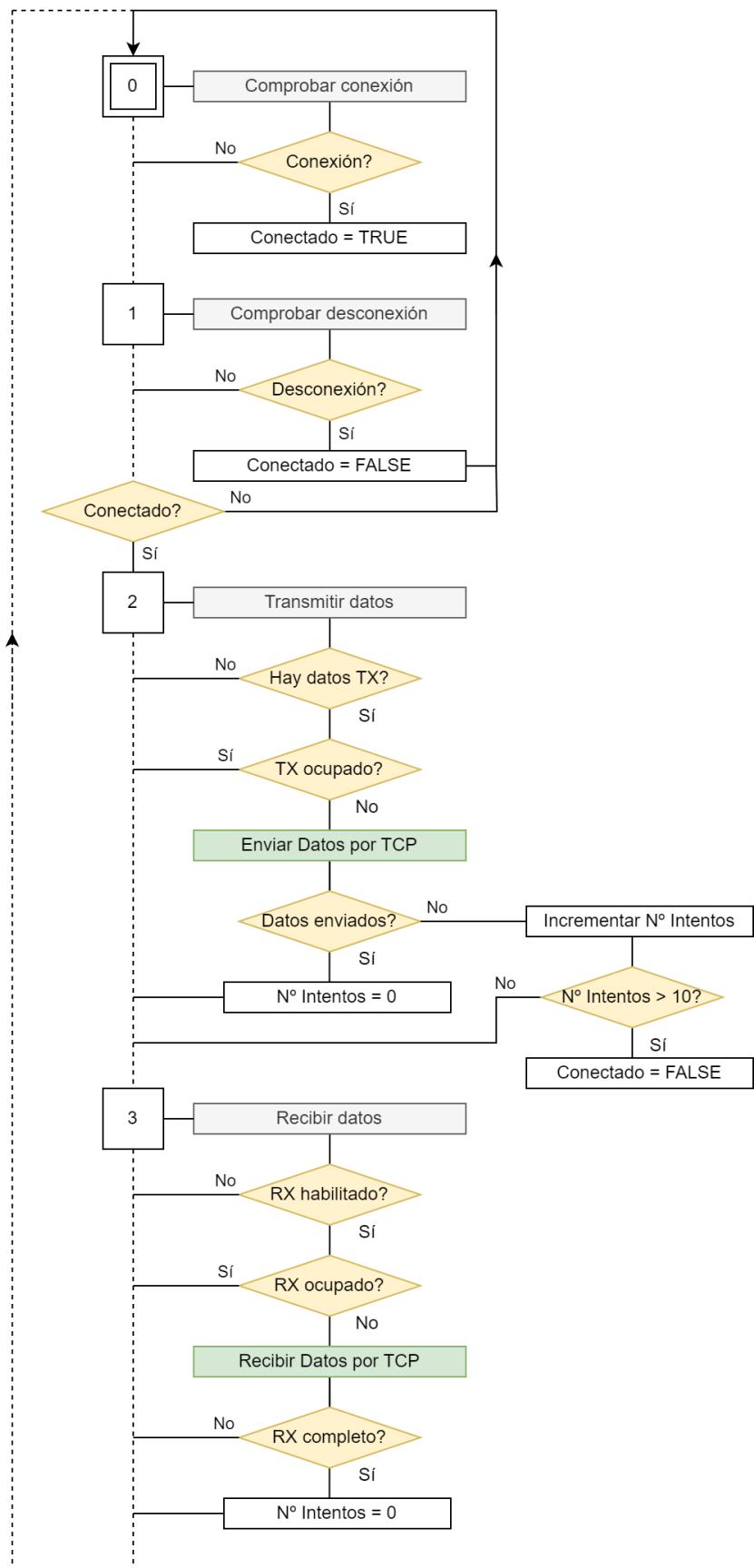
Siguiendo el orden de los bloques de arriba a abajo:

- 1) Calcula SPE haciendo una operación AND de los sensores de presencia (SP) y la máscara (MC). Este valor contiene los materiales necesarios para el algoritmo.
- 2) Si SP es distinto de 0 significa que hay algún material con el que trabajar. Si no, vuelta al paso anterior a esperar que SP cambie.
- 3) Inicializa ciertas variables para el algoritmo:
  - a. IP es un índice que se utilizará para recorrer la lista de prioridades Prioridades[ ].
  - b. SPP es el conjunto de materiales que están en las cintas (y están siendo detectados por SPE) y que no están todavía en la mesa MP.
  - c. MAT almacena la ID de un material. En los puntos siguientes se explica su utilidad.

- 4) Guarda MP, que es el número de la mesa en la posición IP de la lista de prioridades.
- 5) Calcula SPP. Para ello, hay que escoger aquellos materiales dentro de MP **Y** (AND) que **NO** (NOT) están en la mesa MP.
- 6) El valor de IP es un índice que se incrementa si no hay materiales adecuados para la mesa MP.
  - a. Si el índice de IP es válido, continúa al siguiente paso.
  - b. Si llega el momento que se han calculado todas las posibles combinaciones de mesas y materiales presentes en las cintas, entonces IP será igual a NM (en este caso, el array comienza en 0 y acaba en NM-1). Si esto ocurriese, significaría que **los materiales presentes en las cintas** no son necesarios y por lo tanto **pueden ser enmascarados**. Esto es lo mismo que decir que los materiales que se necesitan (la máscara) serán aquellos que se necesitaban de antes **Y** (AND) aquellos que **NO** (NOT) están en la cinta.
- 7) SPP es el conjunto de valores presentes en las cintas y que la mesa MP necesita.
  - a. Si el valor es distinto de 0, entonces **habrá algún bit que es necesario para la mesa MP!** Ahora lo que resta es únicamente encontrar en qué posición se encuentra dicho bit.
  - b. Si este valor es 0 significa que no hay ningún valor necesario y por lo tanto pasaría a la siguiente mesa (en orden de prioridad), por ello se incrementa IP.
- 8) Comienza la búsqueda de la posición del bit que hace SPP distinto de 0. Para ello, se irá desplazando hacia la derecha bit a bit hasta que el bit menos significativo (*LSB*, del inglés, *Less Significative Bit*) sea uno. Para obtener el *LSB*, se enmascara SPP con 0x0001.
  - a. Si el valor es distinto de 0, MAT ya contiene el material al que hace referencia ese bit de SPP.
  - b. Si el *LSB* es cero, se desplaza a la derecha SPP y se pasa al siguiente material MAT. Para esto último, basta con desplazar MAT a la izquierda. Recuérdese que MAT empezó siendo 1; es decir, el material A. Al desplazar a la izquierda, A pasaría a ser B, después C, D1, D2...
- 9) Ya se ha calculado la mesa destino de la pieza. Su valor es MP.
- 10) Dependiendo de si la mesa está vacía o no, se deberá colocar antes la base (material E) o el material MAT.

### 7.1.2. Comunicaciones TCP/IP

En la figura de la página de la página siguiente se muestra el diagrama GRACET del controlador.



**Figura 16. GRAFCET de conexión TCP/IP del controlador.**

Como único detalle, hay que recordar que en el protocolo TCP/IP no hay una señal de “conexión con vida”; es decir, algo que indique que la conexión que se estableció anteriormente prosigue de forma bilateral entre ambos extremos. Únicamente cuando se escribe algo y surge algún error se puede intuir que hay una conexión fallida. Por ello, se cuenta el número de veces que no se han podido transmitir un mensaje para cambiar el estado de conexión del controlador.

### 7.1.3. Procesar RX del robot

Este bloque funcional es muy sencillo. Tiene dos funciones principales:

- Recibe los mensajes provenientes del robot. Hace una comprobación de integridad y, según la ID de los mensajes, realiza cierta acción en el sistema. Más sobre ello en las Secciones 9 y 0.
- Si pasan más de **cinco segundos** sin recibir un mensaje, crea un mensaje de “conexión con vida” que el robot debe responder. Si pasa cierto tiempo y no responde, entonces se considera que el robot no está conectado. Este proceso solo está habilitado cuando hay una conexión (o anteriormente se había establecido) una conexión con el robot.

### 7.1.4. Gestión de errores

Para realizar la gestión de errores se utiliza una cadena de caracteres o *string*. Cuando el sistema encuentra algún problema o el robot manda un mensaje de error, se guarda el código del error dentro de dicha *string*. En este bloque funcional, dependiendo del mensaje de error, entrará en el estado de error. Estando en dicho estado, se bloquea por completo el funcionamiento del sistema y únicamente se puede salir del mismo “rearmando el sistema”. Esta acción retoma el último estado del sistema y vacía el *string* de errores. Si el error perdurase, volvería a surgir el mismo error y el sistema volvería a quedar bloqueado. El rearne del sistema se realiza desde la interfaz HMI.

### 7.1.5. Actualizar HMI

El sistema HMI ha sido programado utilizando “*WinCC Advanced*”. Este trae muchas limitaciones, en especial a la hora de programar en VBA (*Visual Basic for Applications*): casi todas las funciones que se podrían realizar obteniendo primero los valores reales del PLC y procesándolos para mostrar cierta figura en la pantalla, se vuelven imposibles de realizar desde el HMI. Por ello, hay que crear copias individuales de algunas variables y compartir

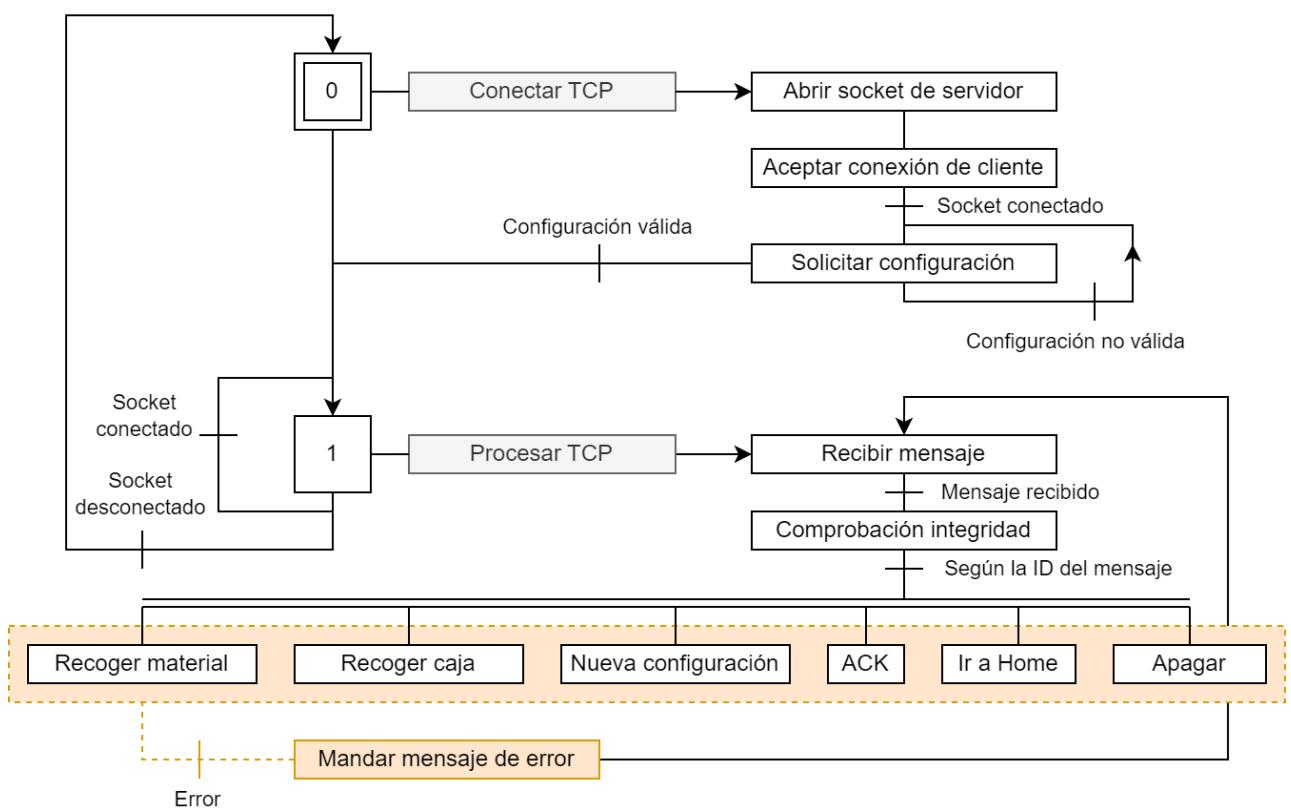
estas con el sistema HMI. Esto afecta no solamente a las variables compartidas desde el PLC al HMI, sino también a las señales que genera el HMI.

En el bloque funcional “Actualizar HMI” se procesan las señales provenientes del HMI y se generan aquellas necesarias para el HMI. Más sobre este bloque en la Sección 0.

Se hablará extensivamente del funcionamiento del HMI en la Sección 12.

## 7.2. GRAFCET del robot

En la figura inferior se muestra un diagrama simplificado del mecanismo del brazo robótico.



**Figura 17. GRAFCET del robot.**

Primeramente, el robot crea un socket donde espera que alguien (en este caso, el PLC) se conecte. Una vez detecta que alguien se ha conectado, manda un mensaje solicitando la configuración del robot (más adelante se hablará sobre qué valores conforman dicha configuración). Si la configuración es válida, el robot la adopta y pasa al siguiente estado.

En el estado “Procesar TCP” el robot espera hasta recibir un mensaje. No está añadido en el diagrama, pero, si pasan **más de 10 segundos**, el robot manda un mensaje de que está con vida. Este tiempo se ha escogido ya que el PLC manda un mensaje de

“conexión con vida” cada cinco segundos, luego de esta manera el robot tendrá tiempo suficiente para recibir el mensaje del PLC y procesarlo. En el caso en el que el PLC pensase que el robot ha quedado desconectado, al recibir el mensaje de “conexión con vida” proveniente del robot reestablecería las comunicaciones.

En el caso en el que un mensaje es recibido, se realiza una comprobación de integridad; es decir, se comprueba si tiene una estructura válida, la longitud es la adecuada al tipo de mensaje, la ID del mensaje es válida...

Siendo el mensaje correcto, el robot puede realizar las siguientes acciones:

- **Recoger material.** El robot recibe la ID de un material y el número de una mesa. Según el material esté situado en la cinta general o en un palé, el robot la recogerá en una posición u otra, y lo desplazará hasta la posición correspondiente de la mesa.
  - **Posibles errores:** Durante este proceso, puede ocurrir que las ventosas que recogen los materiales fallen. En este caso, el robot manda un mensaje indicando que ha ocurrido un fallo y por lo tanto el PLC deshace la escritura del material en la variable Mesa[ ]. Además, manda una alarma al sistema HMI y pausa el sistema.
- **Recoger caja.** Una vez una caja está completada, el PLC manda este mensaje al robot con el número de la caja que debe transportar a la cinta de precintado.
  - **Posibles errores:** Al igual que antes, pueden fallar las ventosas.
- **Nueva configuración.** El usuario puede cambiar desde el HMI la configuración de la planta y esta información necesita tenerla actualizada el robot.
- **ACK.** Es un mensaje que no se utiliza, pero está implementado por seguridad. Más sobre ello en la Sección 9.
- **Ir a Home.** Este comando manda el robot a una posición de home. Esta señal solo se puede realizar cuando el robot no está ocupado; es decir, cuando no está transportando algún componente o realizando alguna instrucción anterior.
- **Apagar.** Detiene por completo la ejecución del programa en el robot; y por lo tanto, la única forma de volver a arrancarlo es encender el robot manualmente.

En la Sección 11 se describe en mayor profundidad, mediante la explicación del código RAPID, el funcionamiento del robot.

## 8. Componentes del sistema

En esta sección serán enumerados los componentes que conforman la planta empaquetadora junto con sus características técnicas.

### 8.1. PLC SIMATIC S7-1500, CPU 1515-2 PN

El *SIMATIC S7-1500*, modelo *CPU 1515-2 PN*, es una unidad central de procesamiento de alta gama diseñada por *Siemens* para aplicaciones avanzadas en sistemas de automatización industrial. Este controlador lógico programable ofrece un rendimiento robusto, alta capacidad de memoria y versatilidad en términos de configuración y comunicación, siendo ideal para entornos industriales exigentes.



Figura 18. PLC SIMATIC S7-1500, CPU 1515-2 PN. Fuente: Catálogo de Siemens.

#### 8.1.1. Especificaciones técnicas

- **Memoria y Rendimiento:** El CPU 1515-2 PN dispone de una memoria de trabajo integrada de 500 KB para programas y 3 MB para datos. Además, soporta una tarjeta de memoria SIMATIC con una capacidad máxima de 32 GB, lo que facilita la expansión y gestión de grandes volúmenes de datos. La CPU destaca por su

capacidad de procesamiento, con tiempos de operación de 30 ns para operaciones de bits, 36 ns para operaciones de palabras, 48 ns para aritmética de punto fijo y 192 ns para aritmética de punto flotante.

- **Interfaces y Comunicación:** Este PLC cuenta con dos interfaces PROFINET integradas. La primera interfaz es PROFINET IRT con un switch de 2 puertos, que permite una comunicación rápida y confiable en redes industriales. La segunda interfaz es PROFINET RT, lo que amplía la capacidad de conectividad del dispositivo. Ambas interfaces soportan múltiples protocolos, incluyendo IPv4, comunicación SIMATIC, comunicación abierta IE, servidor web y redundancia de medios. La CPU 1515-2 PN es capaz de gestionar hasta 256 dispositivos IO a través de PROFINET, con soporte para modos isócronos y priorización de arranque. En este proyecto se utiliza el puerto PROFINET para realizar la conexión con el HMI.
- **Configuración y Diagnóstico:** La configuración del CPU 1515-2 PN se realiza a través de STEP 7 TIA Portal, siendo compatible a partir de la versión V16 (en este trabajo se utiliza la versión V17). El PLC incluye capacidades avanzadas de diagnóstico y monitoreo, como un búfer de diagnóstico con capacidad para 3200 entradas, de las cuales 500 son a prueba de fallos de energía. Además, soporta funciones de forzado y monitoreo de variables, lo que facilita el mantenimiento, la puesta en marcha de los sistemas; y muy importante para el trabajo, la simulación del sistema.
- **Funcionalidad y Extensibilidad:** El PLC soporta una amplia gama de bloques de programa, incluyendo OB, FB, FC y DB, con una capacidad total de 8000 elementos. Los bloques de datos pueden alcanzar un tamaño máximo de 3 MB y pueden ser subdivididos para un uso más eficiente. La CPU permite la creación de alarmas y diagnósticos detallados, incluyendo alarmas de programa y diagnósticos del sistema. Además, soporta funciones avanzadas de contadores y temporizadores, con capacidades de retención ajustables.
- **Energía y Consumo:** El PLC opera con una tensión nominal de 24 V DC, con un rango permisible entre 19.2 V y 28.8 V. Cuenta con protección contra inversión de polaridad y un tiempo de almacenamiento de energía en caso de fallo de tensión de 5 ms. El consumo de corriente nominal es de 0.8 A, con un consumo máximo de 1.1 A y una corriente de irrupción máxima de 2.4 A. La potencia de entrada al bus de *backplane* es de 12 W, con una pérdida de potencia típica de 6.3 W.

- **Capacidades de Red y Redundancia:** El CPU 1515-2 PN soporta la sincronización de relojes y la redundancia de medios a través de MRP (Manager and Client) y MRPD, garantizando una comunicación sin interrupciones incluso en caso de fallos de línea. El dispositivo también es compatible con protocolos de comunicación abiertos como TCP/IP, ISO-on-TCP y UDP, permitiendo la integración en una variedad de redes y sistemas. Será su funcionalidad de TCP/IP la que se utilizará en este trabajo.

El principal motivo para la selección de este PLC es que, mediante S7-PLCSIM Advanced v4.0, pueden simularse conexiones de protocolo abierto TCP/IP. Además, trae un amplio abanico de funciones útiles para la programación.

### 8.1.2. Módulos del PLC

Para el montaje del PLC se han de contar con los módulos de los siguientes apartados. Todos ellos irán montados en el mismo carril DIN que el PLC.

#### 8.1.2.1. Módulo de Alimentación PS 25W 24VDC

El módulo de alimentación *PS 25W 24VDC*, identificado por la referencia *6ES7505-0KA00-0AB0*, es un componente de la familia de sistemas SIMATIC S7-1500 y ET 200MP de Siemens. Este dispositivo está diseñado para suministrar energía a los módulos de estos sistemas, asegurando un funcionamiento fiable y seguro.



**Figura 19. Módulo de alimentación PS 25W 24VDC. Fuente: Catálogo de Siemens.**

El módulo cuenta con las siguientes especificaciones técnicas:

- **Tensión nominal de salida:** 24 V DC.
- **Potencia de salida:** 25 W.
- **Aislamiento galvánico funcional:** Proporciona una barrera segura entre el bus y el módulo, evitando interferencias.
- **Funcionalidades adicionales:** Incluye capacidades de actualización de firmware, datos de identificación I&M (I&M0 a I&M4), reparametrización en RUN, avisos y alarmas de diagnóstico.

El módulo dispone de varios elementos de mando e indicadores situados detrás de su tapa frontal, los cuales incluyen:

- **Indicadores LED:** Muestran el estado operativo actual y los diagnósticos del módulo.
- **Interruptor de conexión y desconexión:** Permite encender o apagar el módulo manualmente.
- **Conecotor de red:** Proporciona una conexión segura y protegida para la alimentación de tensión, evitando el contacto accidental.

La conexión de red debe realizarse utilizando cables flexibles con una sección de 1,5 mm<sup>2</sup> (AWG: 16). Se recomienda que el conductor de protección sea más largo que los otros dos cables para asegurar una correcta conexión a tierra.

El uso del módulo PS 25W 24VDC debe realizarse siguiendo estrictamente las normas de seguridad y las instrucciones de instalación correspondientes. Para ello, hágase referencia al manual anexoado al final de este trabajo.

### 8.1.2.2. Módulo de entradas digitales DI 32x24VDC BA

Para poder introducir señales digitales al sistema, se ha recurrido al módulo de entradas digitales SIMATIC S7-1500/ET 200MP, específicamente el modelo DI 32x24VDC BA (6ES7521 -1BL10 -0AA0) de Siemens.

El módulo cuenta con 32 entradas digitales, aisladas en grupos de 16. Opera con una tensión de entrada nominal de 24 V DC.



**Figura 20. Módulo de entradas digitales DI 32x24VDC BA. Fuente: Catálogo de Siemens.**

En cuanto a su funcionalidad, el módulo soporta datos de Información y Medidas (I&M) desde I&M0 hasta I&M3. Puede ser configurado con STEP 7 (TIA Portal) o con un archivo GSD.

El módulo obtiene energía del bus de la placa trasera, con un consumo de energía típico de 3W. El tipo de entrada es DC con un valor nominal de 24V. Para la señal "0", el rango de tensión de entrada va de -30 a +5V, y para la señal "1", va de +11 a +30V. La corriente de entrada típica para un valor de activo "1" es de 2.7mA.

En términos de diagnóstico e indicadores, el módulo no tiene diagnósticos parametrizables, aunque cuenta con indicadores LED para RUN/ERROR y CHx (estado del canal).

### 8.1.2.3. Módulo de salidas digitales DQ 16x24VDC/0.5A BA

Para controlar las salidas digitales del sistema se utilizará el módulo DQ 16x24VDC/0.5A BA (referencia 6ES7522-1BH10-0AA0) fabricado por Siemens.



**Figura 21. Módulo de salidas digitales DQ 16x24VDC/0.5A BA. Fuente: Catálogo de Siemens.**

Posee 16 salidas digitales aisladas en grupos de 8, capaces de operar con una tensión nominal de salida de 24 V DC y una intensidad nominal de salida de 0.5 A por canal.

Este módulo emplea salidas de estado sólido, protegidas contra cortocircuitos y sobrecargas, e incluye indicadores LED para cada salida que facilitan la visualización de su estado. La conexión se realiza mediante un conector frontal con bornes *push-in*, simplificando la instalación.

El Módulo DQ 16x24VDC/0.5A BA cuenta con un grado de protección IP20, lo que lo hace adecuado para entornos industriales. Su amplio rango de temperatura de funcionamiento, que va desde -20 °C a +60 °C, permite su uso en diversas condiciones climáticas.

### 8.1.3. Disposición de los módulos

En la siguiente figura se puede observar la disposición de los módulos junto con el PLC (de manera esquemática).

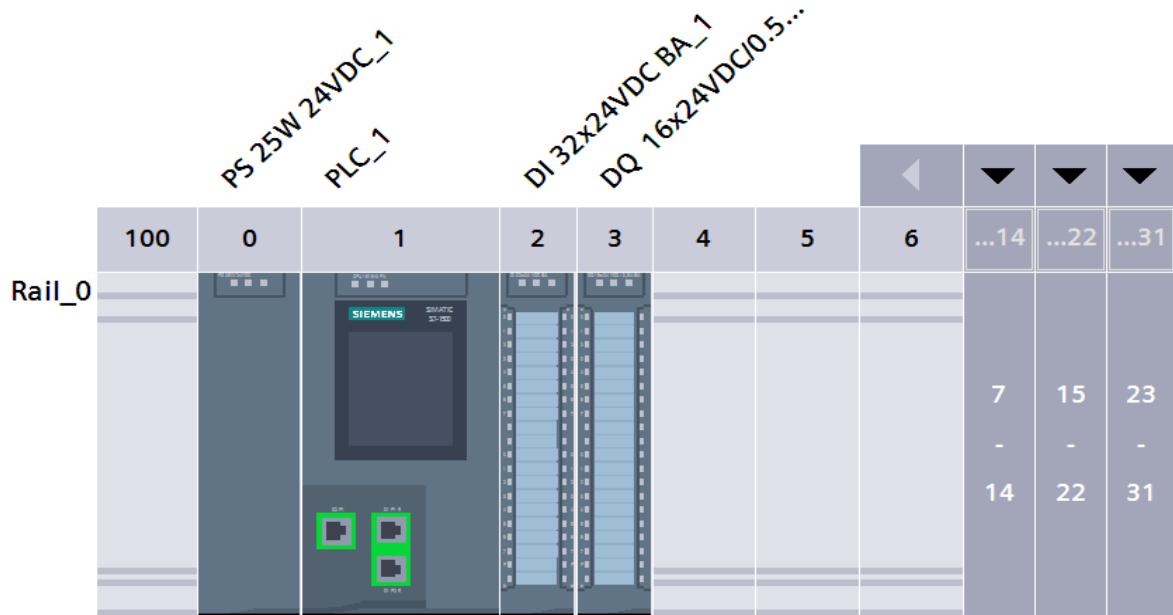


Figura 22. Disposición del PLC con sus módulos en el carril DIN.

## 8.2. Robot IRB 6700-235/2.65

La familia de robots IRB 6700 son la séptima generación de robots industriales de seis ejes de libertad y de gran tamaño de ABB. Estos robots ofrecen un rendimiento superior, una mayor confiabilidad y un menor costo total de propiedad en comparación con la generación anterior.

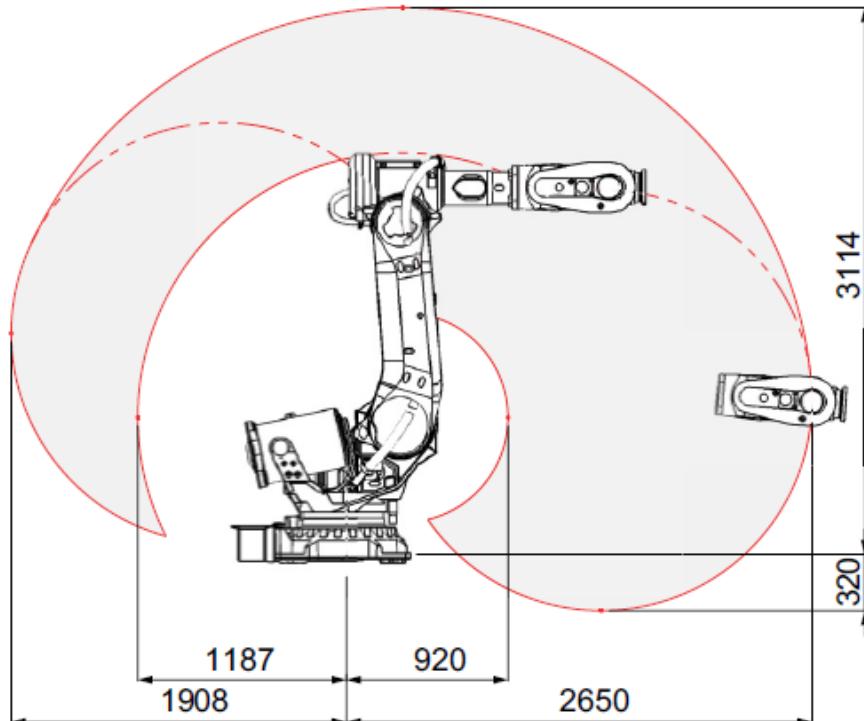


Figura 23. Robot IRB 6700-235/2.65. Fuente: ABB.

Algunas de sus características principales son:

- **Largo tiempo de actividad:** Los robots IRB 6700 tienen una estructura más robusta y motores de nueva generación, lo que reduce el mantenimiento y aumenta la productividad.
- **Carga útil:** Los robots están disponibles con cargas útiles de 150 a 300 kg, lo que los hace adecuados para una amplia gama de aplicaciones.
- **Alcance:** Tienen un alcance de 2.6 a 3.2 metros, lo que permite trabajar en áreas más grandes.
- **Capacidad de repetición de posición:** La capacidad de repetición de posición es una medida de la precisión del robot. Los robots IRB 6700 tienen una capacidad de repetición de posición de 0.05 mm a 0.10 mm.

- **Velocidad máxima del eje:** La velocidad máxima del eje es la velocidad máxima a la que se puede mover cada eje del robot. Los robots IRB 6700 tienen una velocidad máxima del eje de hasta 210 °/s.
- **Protección:** Los robots IRB 6700 están disponibles con protección IP67 y Foundry Plus 2, lo que los hace adecuados para entornos hostiles.



**Figura 24. Rango de trabajo del robot IRB 6700-235/2.65. Fuente: ABB.**

La familia IRB 6700 incluye una variedad de variantes para satisfacer diferentes necesidades de aplicación: montaje en piso, montaje invertido y *Lean ID*.

En este trabajo se ha decidido utilizar un robot IRB 6700 de montaje en piso, en particular, el modelo **IRB 6700-235/2.65**.

Este modelo tiene las siguientes características:

- **Alcance:** 2.65 m.
- **Carga útil:** 235 kg.
- **Torque en la muñeca:** 1324 Nm.
- **Peso:** 1205 kg.
- **Rango de velocidades máximas de rotación:** 90 ~190 °/s.

### 8.2.1. Controlador del robot IRC5

Para dirigir el robot se va a utilizar el controlador IRC5 de ABB, un dispositivo de control de movimiento de alto rendimiento diseñado para robots industriales de ABB.



**Figura 25. Controlador IRC5. Fuente: ABB.**

Entre las características de este controlador, se incluyen:

- **Procesador:** Pentium® CPU.
- **Memoria:** Flash disk para memoria masiva.
- **Interfaz de usuario:** Panel de control en el gabinete o control remoto **FlexPendant**.
- **Interfaces de máquina:** Entradas/salidas digitales y analógicas, bus serie, red Ethernet.
- **Interfaces de campo:** DeviceNet, PROFINET, PROFIBUS, DP y Ethernet/IP.
- **Seguridad:** Seguridad básica y paradas de emergencia, circuitos de seguridad de 2 canales con supervisión, dispositivo de habilitación de 3 posiciones, interruptores de posición electrónicos, 8 salidas de seguridad para monitorización del eje 1-7, **SafeMove**.

### 8.3. HMI TP1200 Comfort

Como sistema de supervisión de la planta se va a utilizar el Panel de Operador Industrial *TP1200 Comfort* de *Siemens* de 12 pulgadas. Este HMI es un dispositivo compacto y versátil diseñado para el control y la monitorización de sistemas industriales.



Figura 26. Pantalla HMI TP1200 Comfort. Fuente: Siemens.

Algunas de sus características son:

- **Pantalla táctil *widescreen* de 12 pulgadas:** La pantalla táctil de alta resolución proporciona una interfaz de usuario intuitiva y fácil de usar.
- **Procesador X86 de alto rendimiento:** El procesador X86 garantiza un rendimiento rápido y fiable.
- **Memoria de usuario de 12 MB:** La memoria de usuario permite almacenar aplicaciones y datos.
- **Interfaz PROFINET:** La interfaz PROFINET permite la comunicación con controladores S7. En este caso, servirá para realizar la simulación en TIA Portal.
- **Interfaces adicionales:** El HMI incluye interfaces RS 485, RS 232, USB y Ethernet.
- **Ampliaciones opcionales:** El HMI se puede ampliar con módulos adicionales para añadir funciones como entradas/salidas analógicas o comunicación en serie.
- **Grado de protección IP65:** El HMI está protegido contra el polvo y el agua, lo que lo hace adecuado para entornos industriales.

## 8.4. Switch EDS-205

El switch EDS-205 de MOXA es necesario para conectar los dispositivos del sistema. Es un switch industrial compatible con los estándares IEEE 802.3/802.3u/802.3x, lo que significa que admite velocidades de red de 10/100 Mbps en modo dúplex completo o semidúplex. Además, cuenta con puertos RJ45 con detección automática MDI/MDIX, que elimina la necesidad de cables especiales.



**Figura 27. Switch industrial EDS-205.**

Diseñada para entornos industriales exigentes, la serie EDS-205 funciona a temperaturas que van desde los -10°C hasta los 60°C y tiene una estructura robusta que la protege de daños. Su instalación es sencilla, ya que se **puede montar en rieles DIN** o directamente en cajas de distribución.

Gracias a la compatibilidad con rieles DIN, el amplio rango de temperatura de funcionamiento, la carcasa con clasificación IP30 y los indicadores LED, la serie EDS-205 son dispositivos confiables, fáciles de usar y listos para funcionar nada más sacarlos de la caja.

Su característica más importante es que soporta tráfico tanto de PROFINET como de TCP/IP. Más sobre ello en la siguiente sección.

## 9. Protocolo de comunicación TCP/IP

Como se ha ido avanzando en apartados anteriores, todas las comunicaciones entre PLC, robot y HMI van a ser realizadas mediante TCP/IP (del inglés, *Transmission Control Protocol/Internet Protocol*).

Con este protocolo se realizan la mayoría de las comunicaciones en Internet. Es un conjunto de protocolos que trabajan en conjunto para permitir que computadoras y dispositivos se comuniquen entre sí de manera eficiente y confiable.

### 9.1. Capas del protocolo

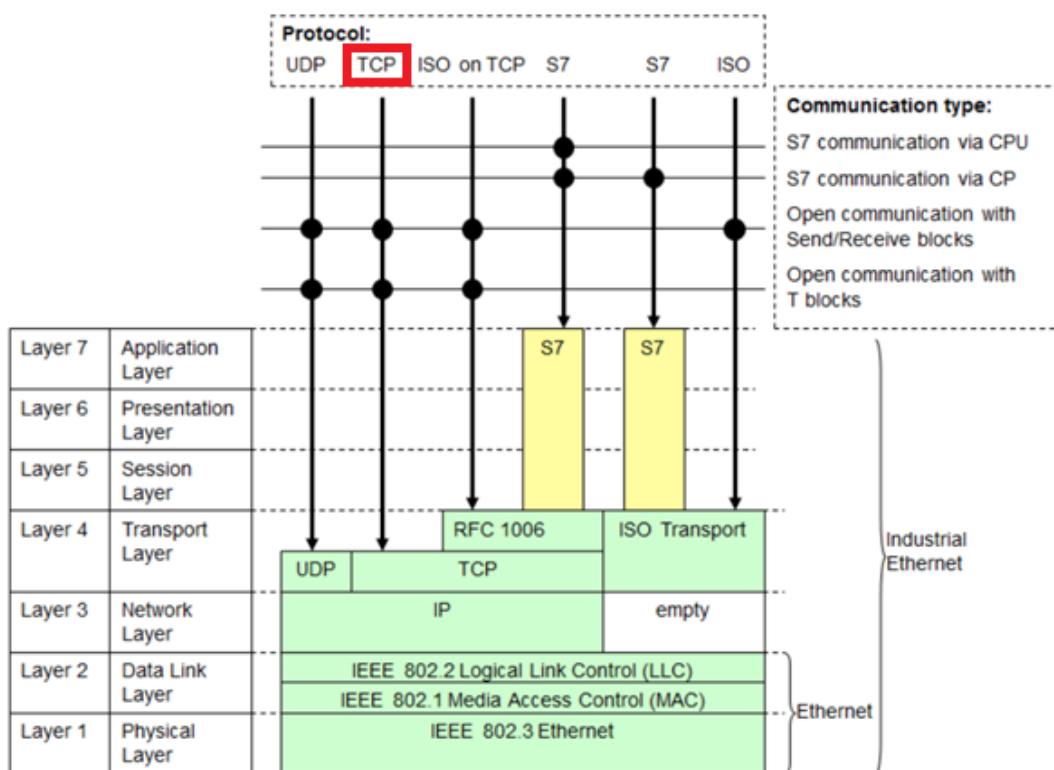
Para definir el protocolo, se deben explicar las funciones de las capas sobre las que se sustenta:

- **Capa de acceso a la red.** Esta capa es el punto de partida donde los datos se convierten en señales que viajan a través de un medio físico (cables o redes inalámbricas). Los protocolos en esta capa, como *Ethernet* o *Wi-Fi*, se encargan de la transmisión de datos en bruto, asegurando una entrega confiable a través de la red física. En el caso del proyecto, en esta capa se va a utilizar *Ethernet*.
- **Capa de Internet.** Cuando los datos atraviesan la capa de acceso a la red, estos pasan al reino de la capa de Internet. Aquí, el protocolo IP (*Internet Protocol*) es fundamental: su función principal es asignar direcciones IP únicas a cada dispositivo conectado a la red, permitiendo que los datos sean dirigidos a su correspondiente destino. Además, IP se encarga del enrutamiento, determinando la mejor ruta para que los paquetes de datos lleguen a su destino final.
- **Capa de Transporte.** La capa de transporte introduce dos protocolos de más alto nivel: **TCP** (*Transmission Control Protocol*) y **UDP** (*User Datagram Protocol*).
  - **UDP** es ideal para aplicaciones que priorizan la velocidad sobre la confiabilidad, como la transmisión de voz o video, ya que no realiza verificaciones de integridad y los paquetes no se tienen por qué recibir o mandar en orden.
  - **TCP** realiza la entrega de datos de manera segura y ordenada, estableciendo una conexión entre el remitente y el receptor antes de la transmisión. Los paquetes de datos se mandan en secuencia y se verifican en su recepción, garantizando la entrega completa sin errores. Es el protocolo sobre el que se va a trabajar en este proyecto.

- **Capa de Aplicación.** En esta capa se encuentran protocolos de alto nivel como el HTTP, FTP y SMTP... En esta capa se pueden diferenciar los protocolos usados entre los dispositivos del proyecto:

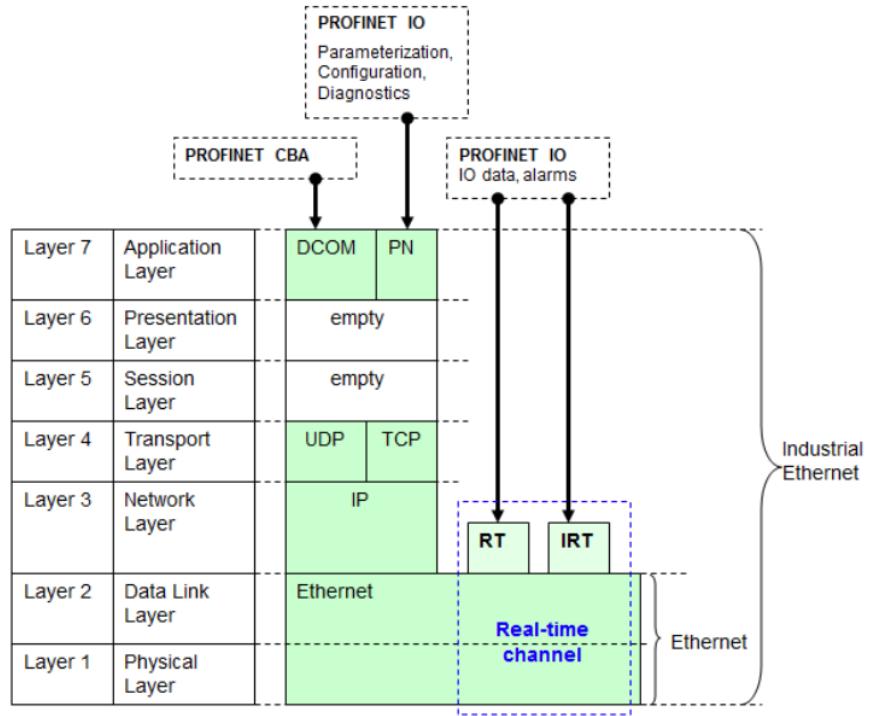
- **Entre el PLC y el robot.** Se ha desarrollado un protocolo sencillo basado en TCP/IP, sobre el cual se hablará extensivamente en los siguientes apartados.
- **Entre el PLC y el HMI.** Se utiliza PROFINET, conocido también como PN. Es el estándar abierto de IE (*Industrial Ethernet*) para automatización. Es una de las comunicaciones por defecto entre el PLC y el HMI. Debido a la facilidad de uso y configuración se ha optado por la misma.

En la siguiente imagen se puede visualizar de manera más sencilla las distintas capas que conforman el *Industrial Ethernet*. En el caso de la comunicación entre el PLC y HMI se va a utilizar TCP que como se puede ver está basado en Ethernet, IP y TCP, además de la capa extra que ha sido desarrollada para este trabajo.



**Figura 28. Protocolo TCP/IP utilizado entre el PLC y el robot (marcado en rojo). Fuente: “CPU-CPU Communication with SIMATIC Controllers” de Siemens.**

Para la comunicación entre el PLC y el HMI se utilizará PROFINET, que a su vez está conformado por distintos protocolos: PROFINET CBA y PROFINET IO, ambos basados también en TCP/IP. En la siguiente imagen se puede observar mejor cómo está estructurado.



**Figura 29. Protocolo PROFINET utilizado entre el PLC y el HMI.** Fuente: “*CPU-CPU Communication with SIMATIC Controllers*” de Siemens.

## 9.2. Tabla de direcciones de los sistemas

Lo más importante que extraer de toda esta información es que todas las comunicaciones van a ser realizadas bajo la misma capa de transporte y, pese a que los protocolos superiores sean diferentes, **los tres sistemas pueden y estarán conectados a la misma red**. Se ha seleccionado la red **192.168.0.x/24** para este TFM.

En el diagrama de la siguiente figura se muestra la conexión a realizar entre los dispositivos con sus respectivas IPs.

En la tabla inferior se enumeran las IPs.

Dispositivo	IP
PLC	192.168.0.10/24
HMI	192.168.0.2/24
IRC5	192.168.0.130/24

**Tabla 4. IPs de los dispositivos de la planta.**



**Figura 30. Diagrama de conexionado de los dispositivos de la planta.**

Para el protocolo entre el PLC y el robot es necesario también indicar los puertos TCP que se van a utilizar y especificar también quién será el servidor y quién el cliente.

El robot es una herramienta del PLC que debe satisfacer las peticiones de este; por lo tanto, tiene sentido que el robot sea el servidor y el PLC el cliente. El robot al arrancar abre su puerto **(9876)** y queda a disposición de conexiones externas. El PLC trata de conectarse con una frecuencia de un segundo al servidor abriendo un socket en el puerto **2000**.

Una vez establecida la conexión, el PLC debe ser el encargado de mantener la conexión con vida; o, mejor dicho, asegurarse de que la conexión no se ha perdido, puesto que está en su interés el mantenerse conectado al robot.

Sin embargo, el robot no tiene esta necesidad, luego únicamente comprueba si el PLC se ha desconectado pasado un largo rato para cerrar el socket de conexión con el cliente y volver a reabrirlo inmediatamente para aceptar nuevas conexiones entrantes.

Es necesario indicar que esta es la configuración del sistema real. Para realizar la simulación se ha tenido que cambiar la IP del robot a la del *localhost* del PC. Esto se debía a que el simulador *S7-PLCSIM Advanced v4* (el simulador del PLC) no permitía la creación de otra IP dentro de la misma red del PLC. Más sobre cómo cambiar la IP del robot en la Sección 11 y sobre cómo realizar la simulación en la Sección 13.

### 9.3. Protocolo PLC-Robot

El protocolo entre el PLC y el robot consiste en una serie de comandos identificados inicialmente por la ID del comando y seguido de los argumentos necesarios para el comando.

A continuación, se muestran los comandos y estructuras de ambos dispositivos. La columna izquierda es el nombre con el que se denominarán los comandos y las siguientes columnas describen el contenido de cada byte.

MENSAJE	0 (ID)	1	2	3	4	5	6	7
ACK	0	A	C	K	ID	-	-	-
ERROR	1	?	MENSAJE ERROR					
ESTADO	2	S	T	A	-	-	-	-
COLOCA MATERIAL	3	Material (0x0000 a 0x8000)				M	Mesa (0x00 a 0x0F)	
COLOCA CAJA	4	M	Mesa (0x00 a 0x0F)					
PARAMETROS	5	Nº Materiales		M	Nº Mesas		-	-
HOME	6	-	-	-	-	-	-	-
APAGAR	7	-	-	-	-	-	-	-

Tabla 5. Lista de comandos PLC → Robot.

MENSAJE	0 (ID)	1	2	3	4	5	6	7
ACK	0	A	C	K	ID	-	-	-
ERROR	1	?	MENSAJE ERROR					
MATERIAL RECOGIDO	2	M	A	T	-	R	E	C
MATERIAL COLOCADO	3	M	A	T	-	C	O	L
CONEXIÓN COMPLETA	4	Material (0x0000 a 0x8000)				M	Mesa (0x00 a 0x0F)	
PARÁMETROS	5	Material (0x0000 a 0x8000)				M	Mesa (0x00 a 0x0F)	

Tabla 6. Lista de comandos Robot → PLC.

Nótese que las IDs no tienen por qué coincidir entre ambos. Esto se debe a que tienen funcionamientos diferentes y no era provechoso utilizar nuevos IDs dejando huecos en el protocolo del otro dispositivo. Sin embargo, aquellos que tienen cierto parentesco o son idénticas sí que tendrán la misma ID.

La estructura de los mensajes es conocida en ambas partes; es decir, el robot conoce de la estructura del PLC para poder mandar mensajes y también conoce la suya propia para poder decodificar los mensajes que les llegue desde el PLC.

En los siguientes puntos se explican cada uno de los comandos.

### 9.3.1. ACK

Todos los mensajes que provengan desde el PLC deben tener su respuesta de confirmación por parte del robot, quedando el PLC bloqueado durante unos cinco segundos hasta recibir confirmación. Dicho mensaje es **ACK** (del inglés, *acknowledge*, confirmar), que debe ir siempre acompañado de la ID del mensaje el cual está confirmando.

De no recibir el ACK en el margen de tiempo fijado, el PLC toma como que ha habido un fallo de conexión y entra en estado de error.

### 9.3.2. ERROR

Indica que ha ocurrido una situación inesperada y que el sistema no está funcionando como debería. Al recibir una señal de error el PLC muestra una alarma por el sistema HMI y queda bloqueado hasta que el usuario lo rearme desde el HMI.

En la siguiente tabla se recogen todos los mensajes de error junto con qué dispositivo los genera.

Mensaje	Remitente	Significado
<b>ID</b>	Robot PLC	El ID del mensaje recibido no ha sido reconocido.
<b>MAT</b>	Robot	El ID del material enviado no es válido o no ha sido reconocido.
<b>MESA</b>	Robot	El número de la mesa no es válida. El número está fuera de rango.
<b>LEN</b>	Robot PLC	La longitud del mensaje recibido no es correcta. El mensaje no es conforme a la especificación.
<b>DISC</b>	Robot PLC	El sistema se ha desconectado. Este mensaje suele mandarse al vacío y no espera respuesta.
<b>FATAL</b>	Robot PLC	Ha ocurrido un fallo no acotado. El sistema puede que se haya desconectado tras esto.
<b>Continúa en la siguiente hoja...</b>		

### Continuación...

<b>CONF_MAT</b>	Robot	El número de materiales por caja enviado durante la configuración no es válido. El robot no dispone de tantos targets para recoger el material.
<b>CONF_MESA</b>	Robot	El número máximo de mesas de trabajo no es válido. La posición de alguna de las mesas acabaría estando fuera del alcance del robot.
<b>ACT</b>	Robot	El actuador del robot ha fallado al activarse/desactivarse.

Tabla 7. Mensajes de error.

### 9.3.3. COLOCA MATERIAL, RECOGIDO, COLOCADO

En la imagen inferior se muestra el comportamiento habitual de la planta ante el comando “COLOCA MATERIAL”.

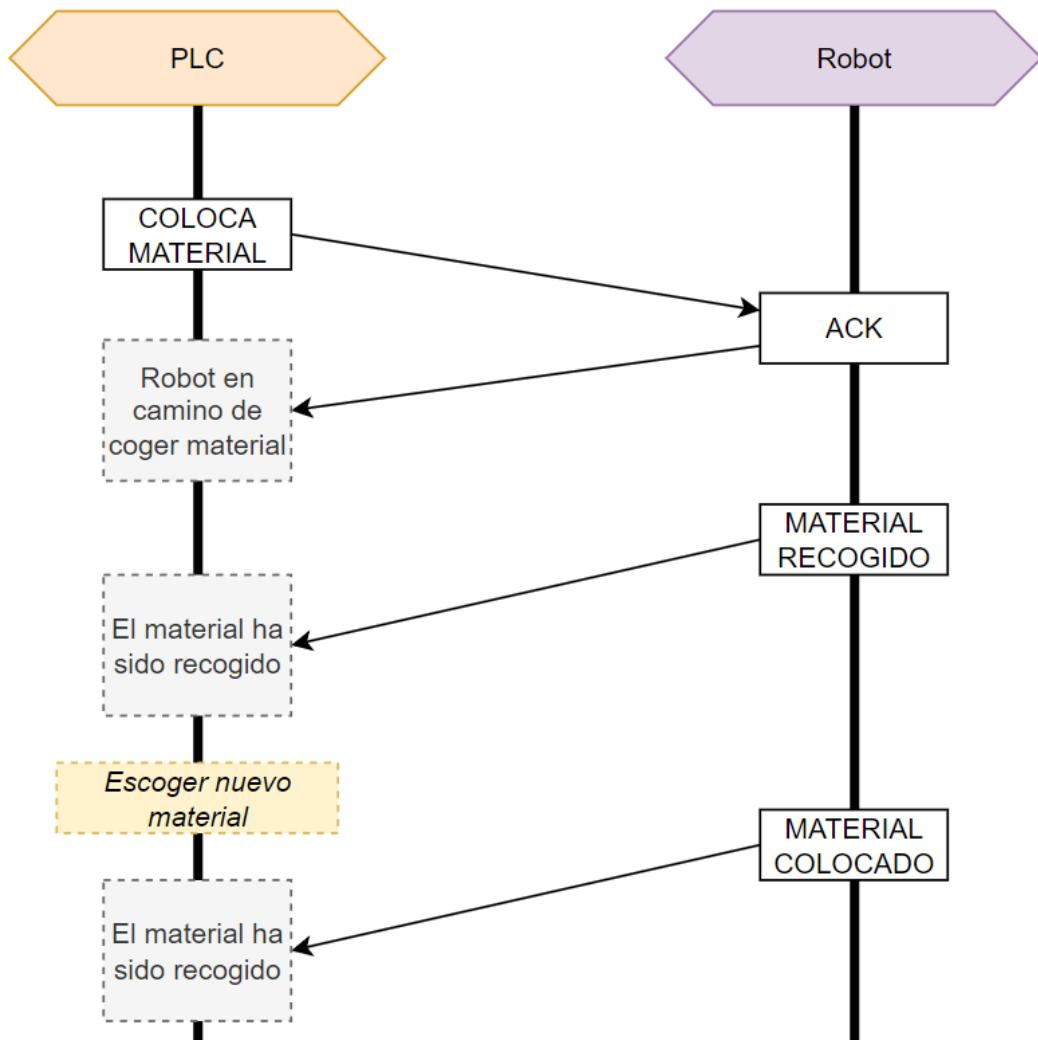


Figura 31. Transmisión de mensajes para “COLOCA MATERIAL”.

Este mensaje necesita dos argumentos:

- El **ID del material** que se desea recoger.
- La posición o **mesa** donde está destinado el material.

En el caso en el que el robot sea capaz de realizar la instrucción y “entienda” el mensaje (su estructura y valores son correctos), manda ACK y comienza la ejecución del comando. Mientras el robot se acerca y comienza a recoger el material, el PLC se mantiene a la espera e indica mediante el HMI que el robot está en movimiento.

Cuando finalmente el robot ha recogido el material lanza el comando “**MATERIAL RECOGIDO**”. El robot comenzará el movimiento hacia la posición de la mesa. Mientras tanto, el PLC escoge la siguiente pieza y ordena su desplazamiento a través de las cintas. De esta manera, el robot no tiene que esperar a que la pieza esté en posición una vez ha dejado la que estaba llevando.

Finalmente, cuando el robot deposita la pieza en la mesa, manda el comando “**MATERIAL COLOCADO**”.

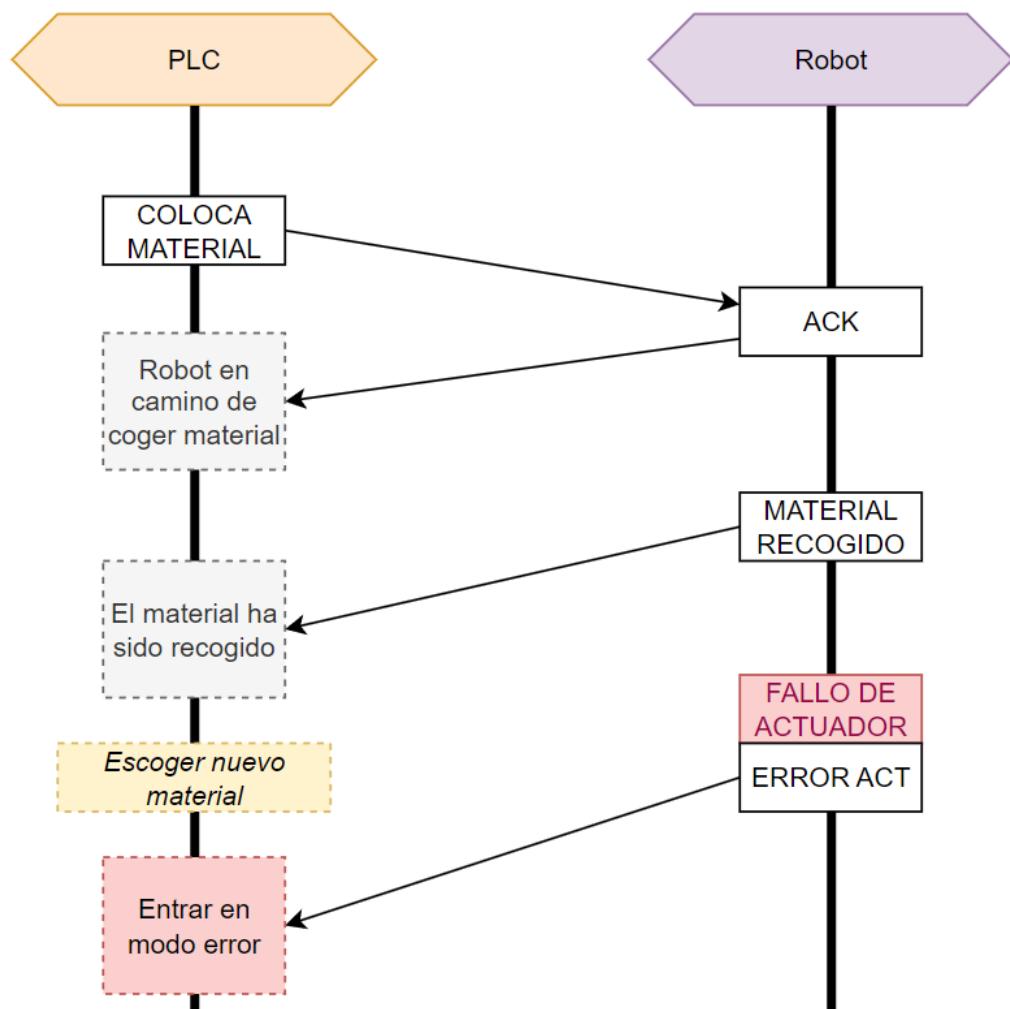


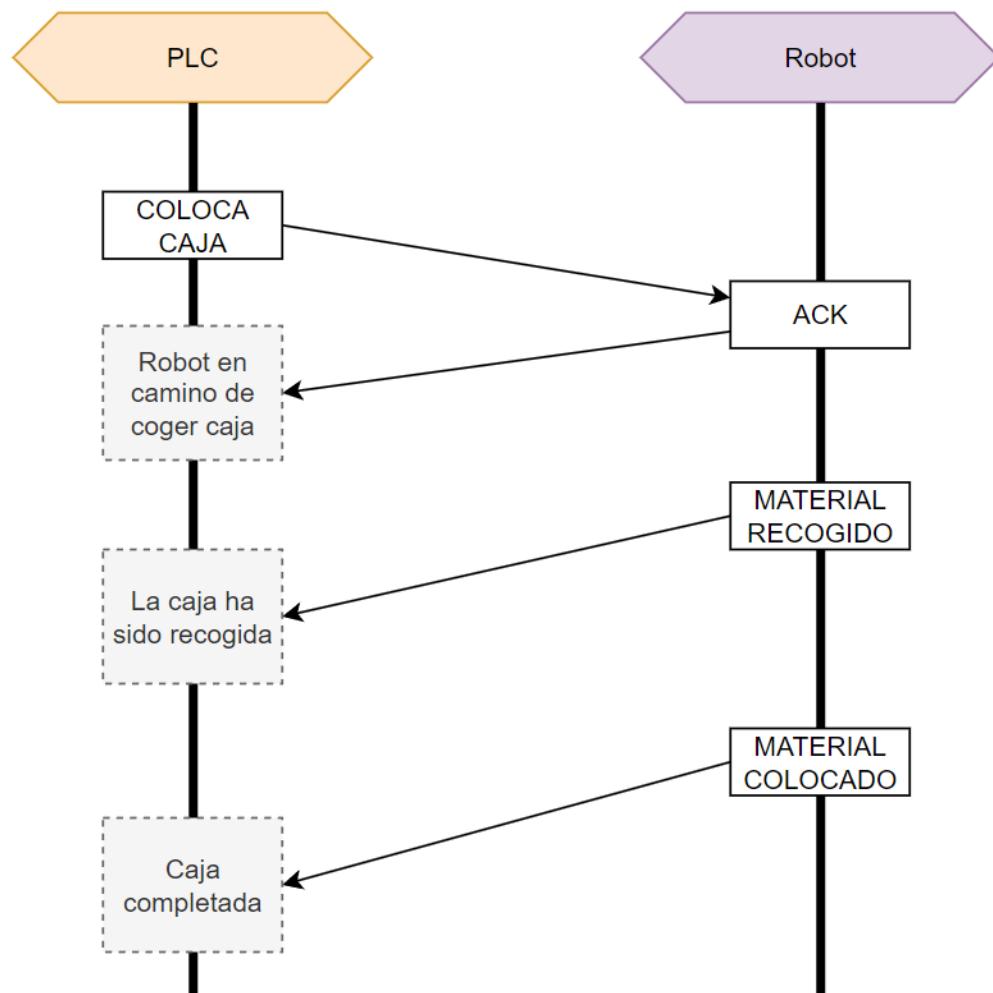
Figura 32. Transmisión ante fallo de actuador tras “COLOCA MATERIAL”.

El movimiento del robot ha terminado y se apaga el piloto del sistema HMI. El robot queda a la espera de nuevas instrucciones. El PLC elimina la copia temporal de los materiales que estaban en tránsito, indicando que ya han sido colocados.

En la Figura 32 se muestra el intercambio de mensajes ante un fallo de actuador por parte del robot. Tras haber recogido el material (o, como norma general, cuando el robot ha comenzado su movimiento), si ocurre un fallo de actuador, el robot se detiene por completo. El PLC recibe el mensaje de ERROR de actuador y entra en estado de fallo. Además, borra de la tabla Mesa[ ] el material que estaba siendo añadido. El PLC queda a la espera de que el usuario lo rearme desde el HMI.

#### 9.3.4. COLOCA CAJA

Cuando una caja está completa; es decir, su tapa acaba de ser puesta, el PLC manda al robot los siguientes mensajes para que la transporte a la cinta de precintado:



**Figura 33. Transmisión de mensajes para “COLOCA CAJA”.**

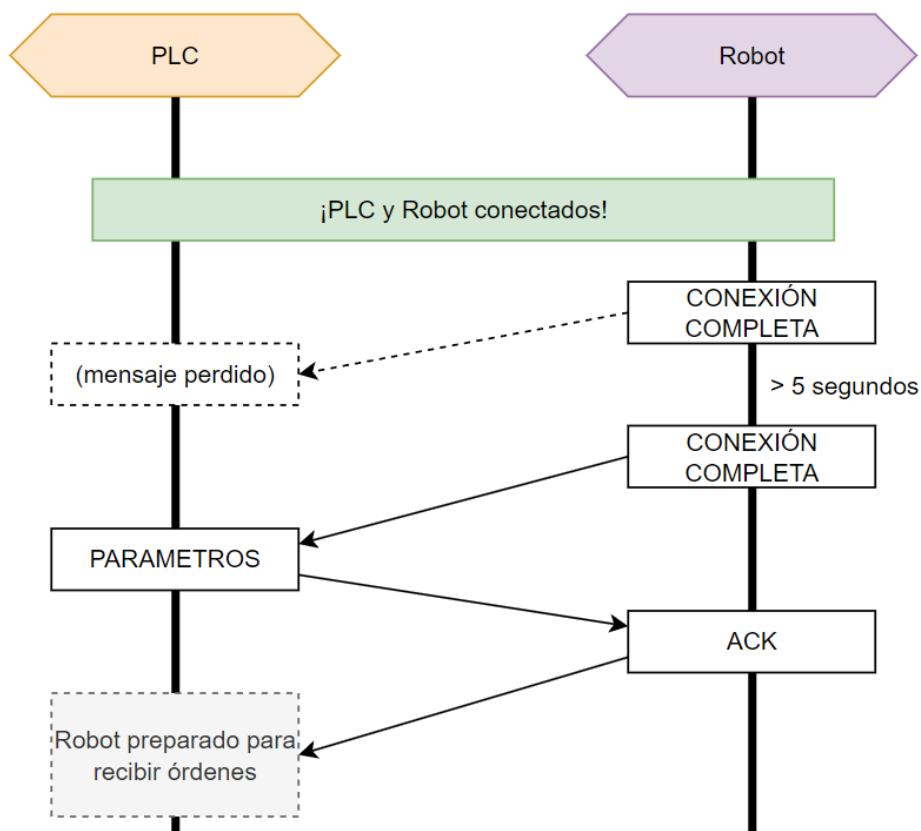
El único parámetro que recibe este comando es el de la mesa que debe recoger.

El funcionamiento es muy similar al de “COLOCA MATERIAL” con la diferencia de que, mientras se está colocando la caja no se escogen nuevos materiales puesto que anteriormente, mientras la tapa estaba siendo colocada, el nuevo material había ya sido elegido y estaba siendo transportado por las cintas.

Cuando se recibe “MATERIAL COLOCADO”, el PLC vacía la posición de la mesa en su lista Mesas [ ] y modifica el array de prioridades.

### 9.3.5. CONEXIÓN COMPLETA, PARÁMETROS

En este apartado se muestra el “*handshake*” del PLC y el robot cuando se conectan o cuando reseñan tras haber perdido conexión.



**Figura 34. Establecimiento de conexión entre PLC y robot.**

En esta imagen se expone la sucesión de mensajes cuando, al principio, el PLC no contesta con los parámetros de simulación. El robot espera durante cinco segundos y si no recibe respuesta, vuelve a mandar el mismo mensaje. Cuando el PLC recibe el comando y lo procesa, contesta al robot con los parámetros de simulación, que son dos:

- **Número de componentes por bandeja.** El robot los necesita para saber a qué posiciones dirigirse a recoger y a depositar las piezas. Puede lanzar un mensaje de **ERROR CONF\_MAT** si el robot no tiene programadas suficientes posiciones para tantas piezas. Este número se manda en decimal, con dos cifras.
- **Número máximo de mesas.** Este valor no es de gran utilidad para el robot, pero, si este número excede el máximo que tiene programado, manda un mensaje de **ERROR CONF\_MESA**, indicando al PLC (y por lo tanto, al usuario mediante HMI) que el número de mesas programado no es válido. Es una medida de seguridad puesto que el robot puede no alcanzar ciertas posiciones y detener el programa en ese caso. Este número se manda en decimal, con dos cifras.

Si los parámetros de configuración son correctos, el robot adopta estos valores y manda ACK, indicando al PLC que el robot está listo para recibir órdenes.

### 9.3.6. ESTADO

El comando ESTADO sirve para corroborar, por parte del PLC, que la conexión con el robot sigue con vida y no se ha perdido.

El mensaje es lanzado por el PLC cada cinco segundos que hayan pasado sin comunicación entre ambos dispositivos. Después de enviarlo, el PLC espera recibir un ACK en los siguientes cinco segundos de haberlo mandado. De no ser así, tomará que el robot se ha desconectado y entrará en estado de error, mandando (al vacío) un mensaje de **ERROR DISC**.

En la figura de la página siguiente puede observarse un diagrama del intercambio.

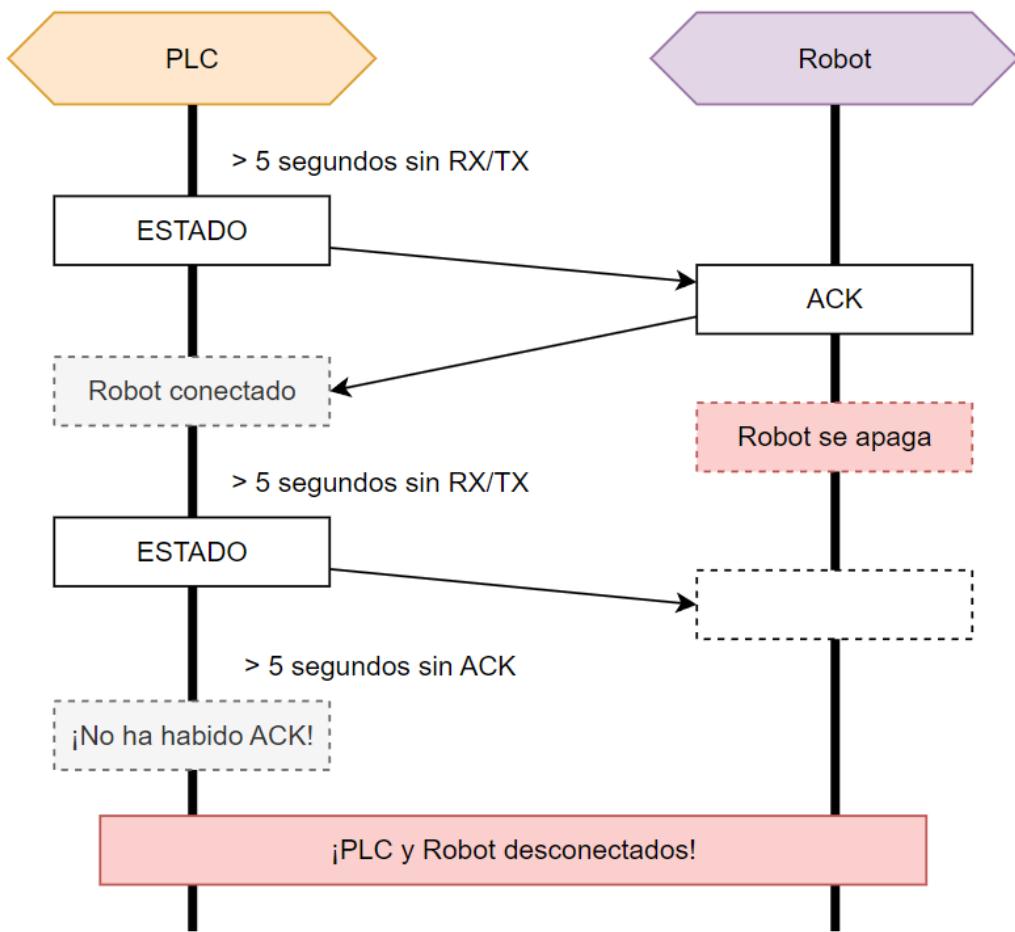


Figura 35. Intercambio de mensajes de ESTADO y desconexión.

### 9.3.7. Ejemplo de intercambio de mensajes

El siguiente es un fragmento real de comunicación entre PLC y robot donde se muestra el emisor, receptor y la longitud del mensaje.

1	TIA => ABB (8): 30040M01
2	ABB => TIA (5): 0ACK3
3	ABB => TIA (8): 2MAT_REC
4	ABB => TIA (8): 3MAT_COL
5	TIA => ABB (4): 4M01
6	ABB => TIA (5): 0ACK4
7	TIA => ABB (4): 2STA
8	ABB => TIA (8): 3MAT_COL
9	ABB => TIA (5): 0ACK2
10	TIA => ABB (8): 30020M02
11	ABB => TIA (5): 0ACK3
12	ABB => TIA (8): 2MAT_REC
13	ABB => TIA (8): 3MAT_COL

Tabla 8. Ejemplo de intercambio de mensajes PLC y robot.

1	30040M01	COLOCA MATERIAL (TAPA, MESA 1)
2	0ACK3	ACK
3	2MAT_REC	MATERIAL RECOGIDO (TAPA)
4	3MAT_COL	MATERIAL COLOCADO (TAPA)
5	4M01	COLOCA CAJA
6	0ACK4	ACK
7	2STA	ESTADO
8	3MAT_COL	MATERIAL COLOCADO
9	0ACK2	ACK (del mensaje 7)
10	30020M02	COLOCA MATERIAL (BASE, MESA 2)
11	0ACK3	ACK
12	2MAT_REC	MATERIAL RECOGIDO (BASE)
13	3MAT_COL	MATERIAL COLOCADO (BASE)

**Tabla 9. Explicación del ejemplo de intercambio de mensajes.**

En este ejemplo se está terminado la caja de la mesa 1. El PLC manda colocar la tapa y, una vez cerrada la caja, manda colocarla en la cinta de precintado. Una vez la caja ha sido colocada, el robot comienza a colocar materiales en una nueva caja en una mesa diferente. El primer material para colocar es la base.

## 10. PLC

En este apartado se va a mostrar y explicar la programación del PLC. Para ello se va a hacer uso directo de capturas del código y la interfaz de TIA PORTAL v17.

### 10.1. Configuración del PLC

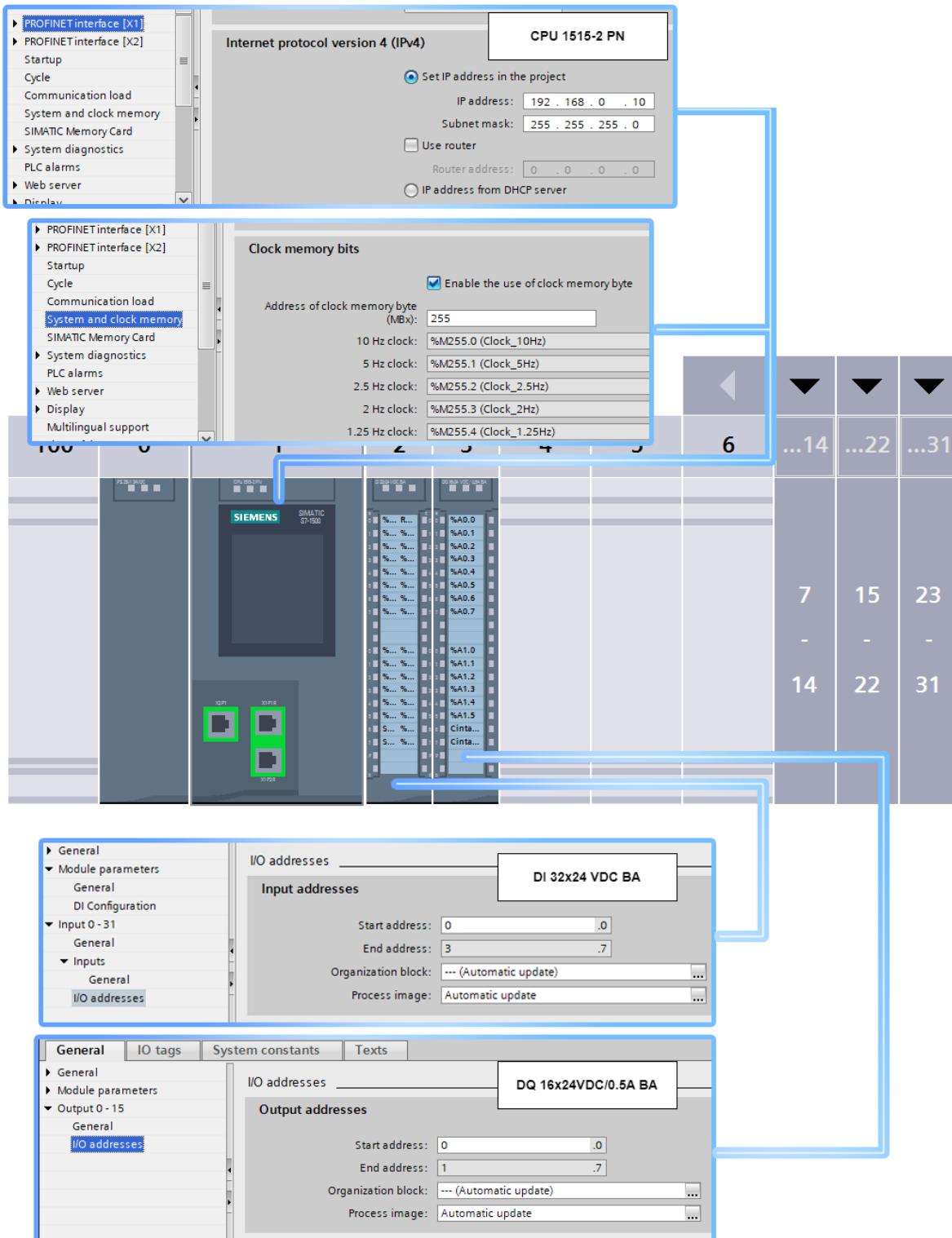


Tabla 10. Configuración del PLC y sus módulos.

Cabe destacar que el PLC ha sido configurado con su IP 192.168.0.10 y que han sido habilitados los bits de reloj. Esto es para poder realizar los intentos de conexión de manera periódica.

Las entradas digitales comienzan desde E0.0 hasta E3.7.

Las salidas digitales comienzan desde A0.0 hasta 1.7.

## 10.2. Variables simbólicas

Se pueden diferenciar los siguientes bloques o tablas de variables simbólicas:

### 10.2.1. PLC Tags

Dentro de esta categoría se han separado las siguientes tablas de variables:

#### 10.2.1.1. Tabla de tags por defecto (Default tag table)

Contiene los bits de reloj.

Default tag table								
	Name	Data type	Address	Retain	Acces...	Writa...	Visib...	
1	DI Clock_Byte	Byte	%MB255	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
2	DI Clock_10Hz	Bool	%M255.0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
3	DI Clock_5Hz	Bool	%M255.1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
4	DI Clock_2.5Hz	Bool	%M255.2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
5	DI Clock_2Hz	Bool	%M255.3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
6	DI Clock_1.25Hz	Bool	%M255.4	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
7	DI Clock_1Hz	Bool	%M255.5	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
8	DI Clock_0.625Hz	Bool	%M255.6	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
9	DI Clock_0.5Hz	Bool	%M255.7	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

Tabla 11. Tabla de tags por defecto.

#### 10.2.1.2. Estados

Contiene los bits que definen el estado del sistema; es decir, sirven para generar el diagrama GRAFCET visto en hojas anteriores.

Se pueden resaltar los estados de pausa (el sistema queda detenido bajo petición del usuario) y el estado de error (donde el PLC espera que el usuario reactive el sistema).

El DWord “Estados” sirve para realizar una comprobación rápida de todos los bits de estado. Esto es útil al arranque, para fijar “Estado 0” a uno si “Estados” es cero.

Estados								
	Name	Data type	Address	Retain	Acces...	Writa...	Visibl...	
1	Estado0	Bool	%M0.0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
2	Estado1	Bool	%M0.1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
3	Estado2	Bool	%M0.2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
4	Estado3	Bool	%M0.3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
5	Estado4	Bool	%M0.4	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
6	Estado5	Bool	%M0.5	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
7	Estado6	Bool	%M0.6	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
8	Estado7	Bool	%M0.7	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
9	Estado8	Bool	%M1.0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
10	Estado9	Bool	%M1.1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
11	Estado10	Bool	%M1.2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
12	Estado11	Bool	%M1.3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
13	Estado12	Bool	%M1.4	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
14	Estado13	Bool	%M1.5	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
15	EstadoPausa	Bool	%M1.6	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
16	EstadoError	Bool	%M1.7	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
17	Estados	DWord	%MD0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Tabla 12. Tabla de tags “Estados”.

### 10.2.1.3. IO Programa

Estos tags son principalmente las entradas y salidas digitales de la planta. Incluyen además las variables de entrada auxiliares para simular la planta desde el sistema HMI. Entrando en el modo de simulación, desde el HMI se podrá simular la presencia de objetos mediante la activación de los sensores de presencia simulados. Estas son todas las variables que comienzan por “HMI\_”. En el código se juntan ambas entradas, reales y simuladas, mediante una puerta OR.

IO_Programa				
	Name	Data type	Address	Comment
1	SP_Cintas	Word	%EW0	Cada bit se activará según los SP del final de las cintas de materiales
2	ActivacionCintas	Word	%AW0	Cada bit indica la activación de una de las cintas
3	CintaGeneral	Bool	%A1.6	Activación de la cinta general
4	SP_CintaGeneral	Bool	%E1.6	Sensor de presencia de la cinta general
5	ReiniciarSistema	Bool	%E2.0	Solicitud desde pulsador de reiniciar sistema
6	SP_CintaEmpaquetado	Bool	%E1.7	Sensor de presencia de la cinta de empaquetado
7	CintaEmpaquetado	Bool	%A1.7	Activación de la cinta de empaquetado
8	HMI_SP_Cintas	Word	%MW10	Entrada simulada del HMI
9	HMI_SP_CintaGeneral	Bool	%M11.6	Entrada simulada del HMI
10	HMI_SP_CintaEmpaquetado	Bool	%M11.7	Entrada simulada del HMI
11	HMI_PaleE	Bool	%M12.0	Entrada simulada del HMI
12	HMI_PaleF	Bool	%M12.1	Entrada simulada del HMI
13	HMI_ReiniciarSistema	Bool	%M12.2	Reiniciar el sistema desde HMI

Tabla 13. Tabla de tags "IO Programa".

## 10.2.2. Data Blocks

Además de los DB (*Data Blocks*) propios de los bloques de comunicaciones (TCON, TDISCON, TRCV y TSEND), se han creado los bloques de las siguientes secciones.

Es importante indicar que todos los DB que tienen que ser accedidos por el sistema HMI tienen desmarcada la opción de “Bloque de acceso optimizado”.

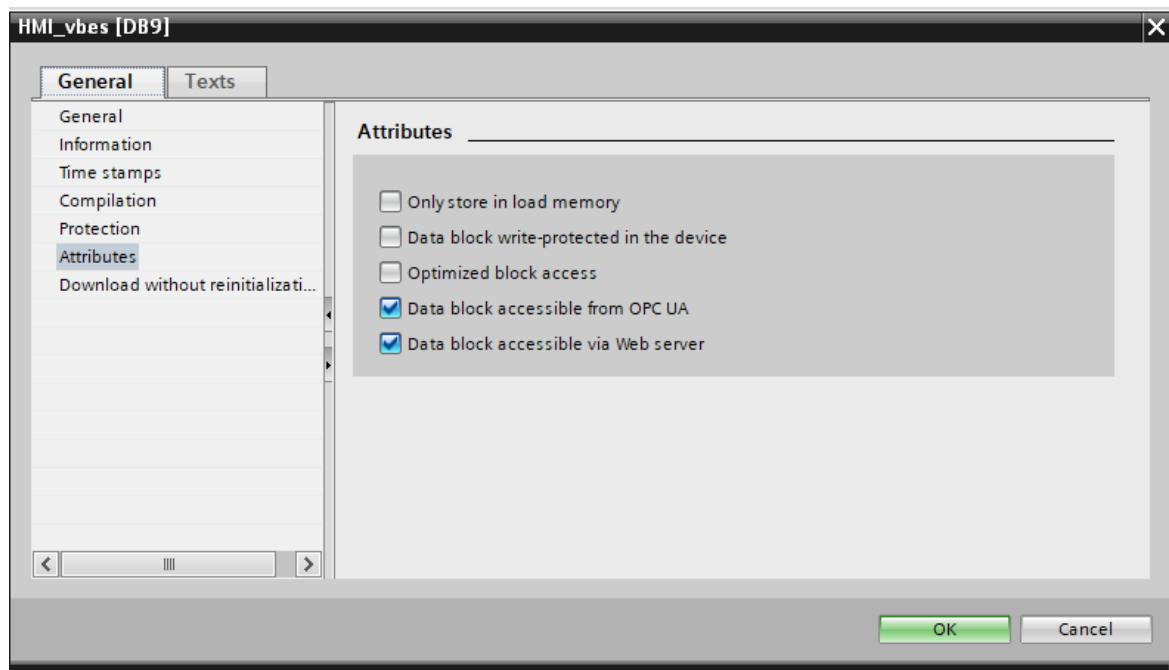


Tabla 14. Propiedades de los Data Blocks accesibles por HMI.

### 10.2.2.1. Variables del sistema

El bloque real se llama “**SISTEMA\_vbes**”, su identificador [DB8].

Contiene todas las variables del algoritmo de selección de materiales y valores auxiliares como tiempos, booleanos de estado del robot, valores de configuración (números de piezas y máximo de mesas), mensajes de error...

En la tabla de la página siguiente se muestran todas las variables del sistema junto con su descripción y valores iniciales.

Nota:

- Mesa [ ] es un Array[0..15] of Word.
- Prioridad [ ] es un Array[0..15] of Int.

SISTEMA_vbes					
	Name	Data type	Offs..	Start v...	Comment
1	► Static				
2	► Mesa	Arra...	0.0		Los bits de Mesa indican si el componente está presente o no en la posición de esa mesa
3	► Prioridad	Arra...	32.0		A menor sea el índice, mayor prioridad tiene dicha mesa. Almacena el número de mesa
4	► Material_Destino	Word	64.0	16#0	Material que será transportado por la cinta general
5	► Mesa_Destino	Int	66.0	0	Mesa en la que irá el material que circula por la cinta general
6	► Material_EnProceso	Word	68.0	16#0	Variable temporal que almacena el material que el robot está colocando
7	► Mesa_EnProceso	Int	70.0	0	Variable temporal que almacena la mesa que el robot está utilizando
8	► Máscaras_Cintas	Word	72.0	16#0	Deshabilita un SP de la cinta de materiales. Ocurre cuando el material no es necesario
9	► NúmeroMesas	Word	74.0	4	Número de mesas reales en el sistema (máximo 16)
10	► SP_Enmascarados	Word	76.0	16#0	Los sensores de presencia tras tener en cuenta cuáles están deshabilitados
11	► IndicePrioridad	Int	78.0	0	Indice de la matriz de prioridades
12	► MatsNecesarios	Word	80.0	16#0	Inversa del contenido en Mesa[]
13	► SP_Prioritarios	Word	82.0	16#0	Los bits de esta vbe indican si el alimento está disponible y es necesario para la mesa prioritaria
14	► NúmeroMateriales	Int	84.0	5	Número de materiales sin contar caja y tapa
15	► RobotOcupado	Bool	86.0	false	True si el robot está realizando alguna operación
16	► RobotConectado	Bool	86.1	false	True si la conexión con el robot es correcta
17	► MsgError	String	88.0	"	Almacena el último mensaje que ha causado un error
18	► ElementoBase	Word	344.0	16#0	El identificador de la base es bit(NumMateriales)
19	► ElementoTapa	Word	346.0	16#0	El identificador de la tapa es bit(NumMateriales+1)
20	► ModoSimulacion	Bool	348.0	false	Determina si el modo simulación está activo o no
21	► TActivacionCintaEmp	Time	350.0	T#0ms	Tiempo hasta el que debe mantenerse activa la cinta de empaquetado
22	► EstadoAnterior	Word	354.0	16#0	Almacena el estado anterior, sirve para retomar el funcionamiento tras un fallo leve

Tabla 15. Variables del sistema en “SISTEMA\_vbes”.

### 10.2.2.2. Variables TCP

El nombre del bloque real es “TCP\_vbes”, su identificador [DB6].

Almacena los mensajes recibidos en el array RX y los mensajes salientes en TX. Tiene varios booleanos que indican si una parte del programa ha mandado a enviar un mensaje (TX\_Enviar), si el TX está siendo realizado (TX\_Procesando), si la recepción de mensajes está habilitada (RX\_Recibir) o si se ha recibido un mensaje y hay que procesarlo (RX\_Procesando). Almacena también la longitud de los mensajes entrantes y salientes (esto es necesario puesto que los mensajes están almacenados en arrays y no en strings). Y por último, los tiempos de recepción y transmisión para poder lanzar errores de desconexión.

TCP_vbes					
	Name	Data type	Offset	Start ..	Comment
1	► Static				
2	► RX	Array[0..49] of Char	0.0		Array del mensaje de entrada (RX)
3	► TX	Array[0..49] of Char	50.0		Array del mensaje de salida (TX)
4	► DesconectarServidor	Bool	100.0	false	Señal para desconectar el servidor
5	► TX_Enviar	Bool	100.1	false	Señal para enviar el mensaje en TX
6	► TX_Procesando	Bool	100.2	false	Hasta no recibir el ACK, esta vbe será true
7	► RX_Recibir	Bool	100.3	true	Habilita la recepción de mensajes
8	► RX_Procesar	Bool	100.4	false	Cuando se recibe un mensaje, se manda a procesar
9	► TX_Len	UDInt	102.0	0	Longitud mensaje TX
10	► RX_Len	UDInt	106.0	0	Longitud mensaje RX
11	► LastID	Char	110.0	''	Último mensaje mandado (para comprobar el ACK)
12	► TiempoTX	Time	112.0	T#0ms	Tiempo del sistema cuando se mandó el último TX
13	► TiempoRX	Time	116.0	T#0ms	Tiempo del sistema cuando se recibió el último RX

Tabla 16. Variables TCP en “TCP\_vbes”.

### 10.2.2.3. Variables HMI

El nombre del bloque en el programa es “HMI\_vbes”, su identificador, [DB6].

Contiene copias de variables ya declaradas en “SISTEMA\_vbes” pero procesadas para que se muestren iconos, mensajes, pilotos o botones en el sistema HMI según conveniencia. Más sobre estas en la Sección 12.

TCP_vbes					
	Name	Data type	Offset	Start ..	Comment
1	► Static				
2	► RX	Array[0..49] of Char	0.0		Array del mensaje de entrada (RX)
3	► TX	Array[0..49] of Char	50.0		Array del mensaje de salida (TX)
4	► DesconectarServidor	Bool	100.0	false	Señal para desconectar el servidor
5	► TX_Enviar	Bool	100.1	false	Señal para enviar el mensaje en TX
6	► TX_Procesando	Bool	100.2	false	Hasta no recibir el ACK, esta vbe será true
7	► RX_Recibir	Bool	100.3	true	Habilita la recepción de mensajes
8	► RX_Procesar	Bool	100.4	false	Cuando se recibe un mensaje, se manda a procesar
9	► TX_Len	UDInt	102.0	0	Longitud mensaje TX
10	► RX_Len	UDInt	106.0	0	Longitud mensaje RX
11	► LastID	Char	110.0	''	Último mensaje mandado (para comprobar el ACK)
12	► TiempoTX	Time	112.0	T#0ms	Tiempo del sistema cuando se mandó el último TX
13	► TiempoRX	Time	116.0	T#0ms	Tiempo del sistema cuando se recibió el último RX

Tabla 17. Variables HMI en “HMI\_vbes”.

### 10.2.2.4. Parámetros de conexión

Este bloque es fundamental para el correcto funcionamiento de la planta puesto que en él se especifica la dirección IP y puerto del robot.

El nombre real del bloque es “DB\_ParamsConexion” y su identificador es [DB2].

DB_ParamsConexion				
	Name	Data type	Start value	Comment
1	► Static			
2	► Interfaceld	HW_ANY	64	Identificador HW para el módulo IE
3	► ID	CONN_OUC	1	Identificador de conexión
4	► ConnectionType	Byte	16#0B	Tipo de conexión (0x0B = TCP/IP)
5	► ActiveEstablished	Bool	true	Establecimiento de conexión activa
6	► RemoteAddress	IP_V4		Dirección IPv4 remota (IP del robot)
7	► ADDR	Array[1..4] of Byte		IPv4 address
8	► ADDR[1]	Byte	192	IPv4 address
9	► ADDR[2]	Byte	168	IPv4 address
10	► ADDR[3]	Byte	0	IPv4 address
11	► ADDR[4]	Byte	130	IPv4 address
12	► RemotePort	UInt	2000	Puerto remoto (puerto del robot)
13	► LocalPort	UInt	0	Puerto local

Tabla 18. Parámetros de conexión en "DB\_ParamsConexion".

Este bloque es generado automáticamente al crear el bloque de función TCON, siendo necesario pasarlo como el argumento CONNECT al bloque.

### **10.3. Bloques de programa**

La programación del PLC, con comentarios realizados por el autor, se puede ver en el Anexo 1 del trabajo.

## 11. Robot

En este apartado se va a mostrar la programación del robot realizada mediante RobotStudio para el robot de ABB **IRB6700-235-265**.

### 11.1. Entorno del robot

Al abrir el programa, se puede ver el entorno de trabajo del robot. Este es un entorno simplificado, pero cumple con los requisitos que deberíamos encontrar en el sistema real.

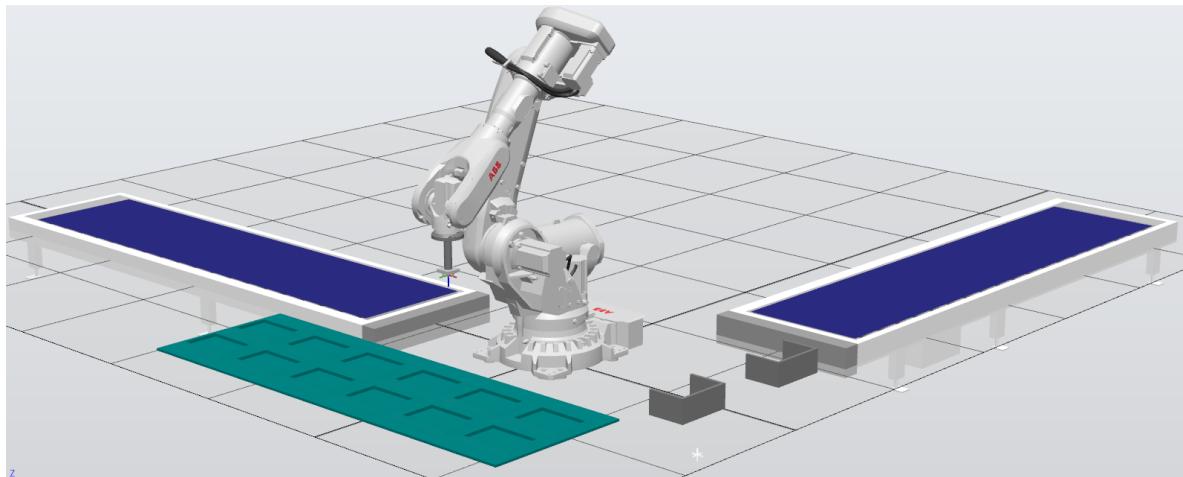


Figura 36. Plano general del entorno con el robot posicionado en Home.

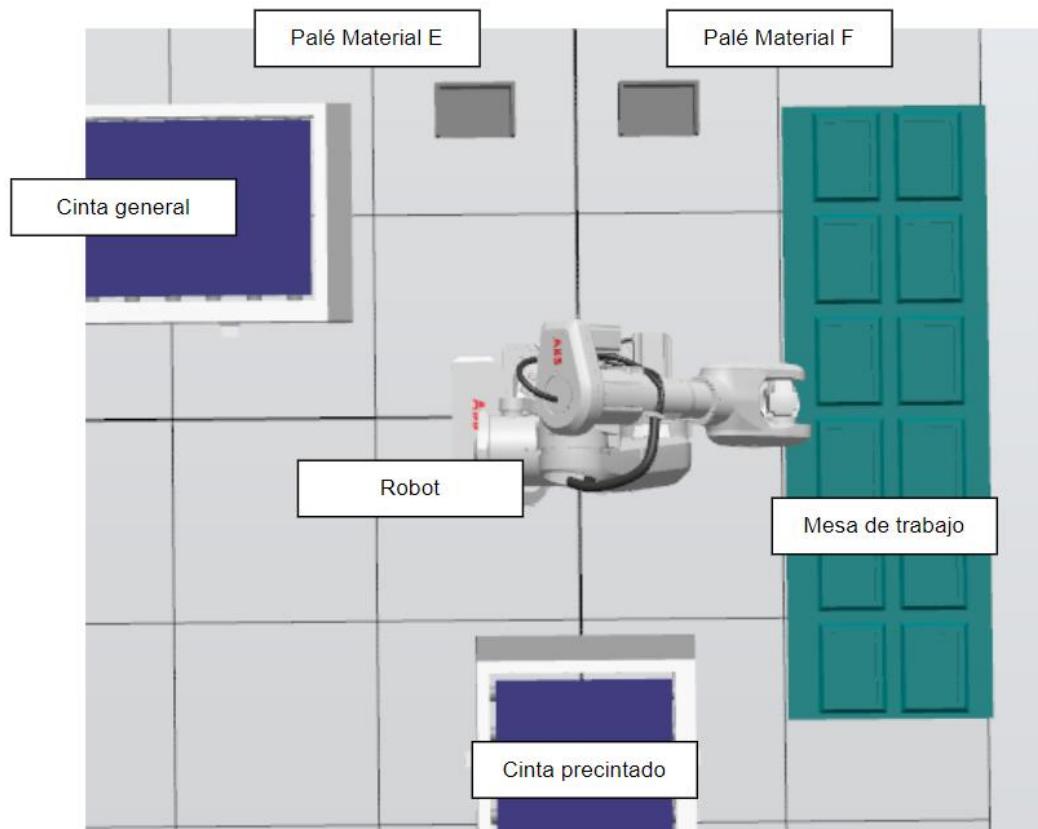


Figura 37. Componentes del entorno del robot.

Cuando el PLC está conectado al robot y manda recoger una pieza, el entorno visual de RobotStudio genera una pieza en su posición correspondiente. Si los materiales provienen de una cinta, el sistema genera el material en la posición de recogida de la cinta general. Si por el contrario es un material E o F, lo genera en los palés.

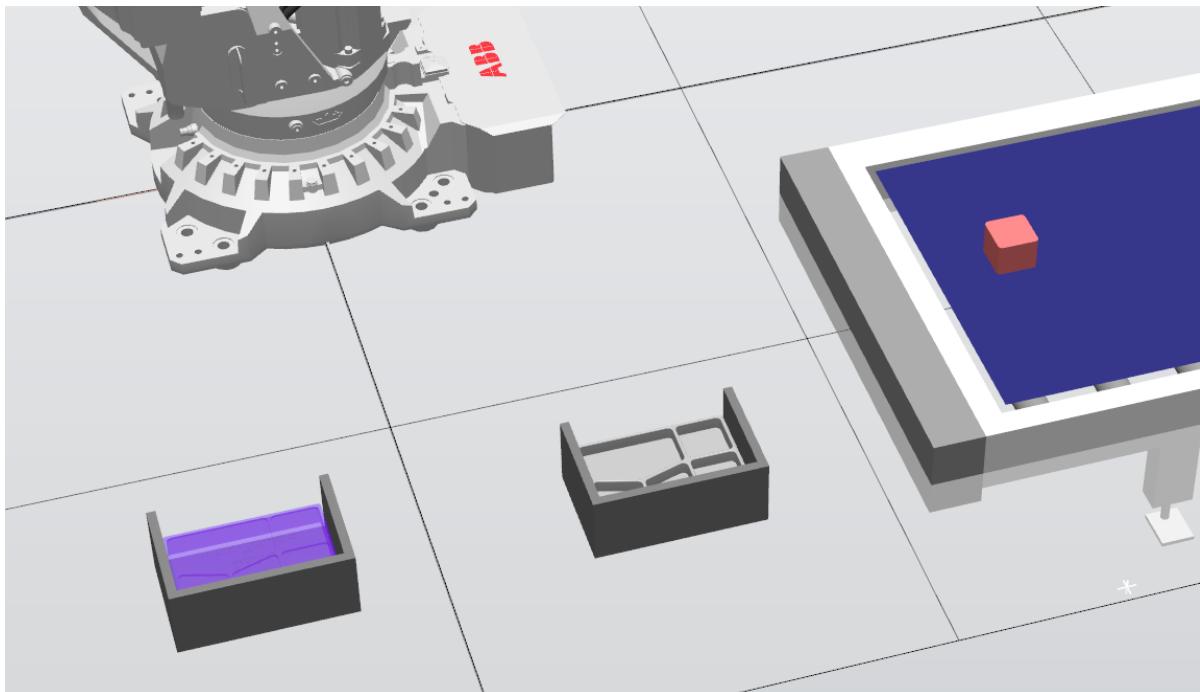


Figura 38. Ejemplo de generación de piezas D, E y F.

El robot recogerá las piezas y las depositará en la mesa según estén disponibles.

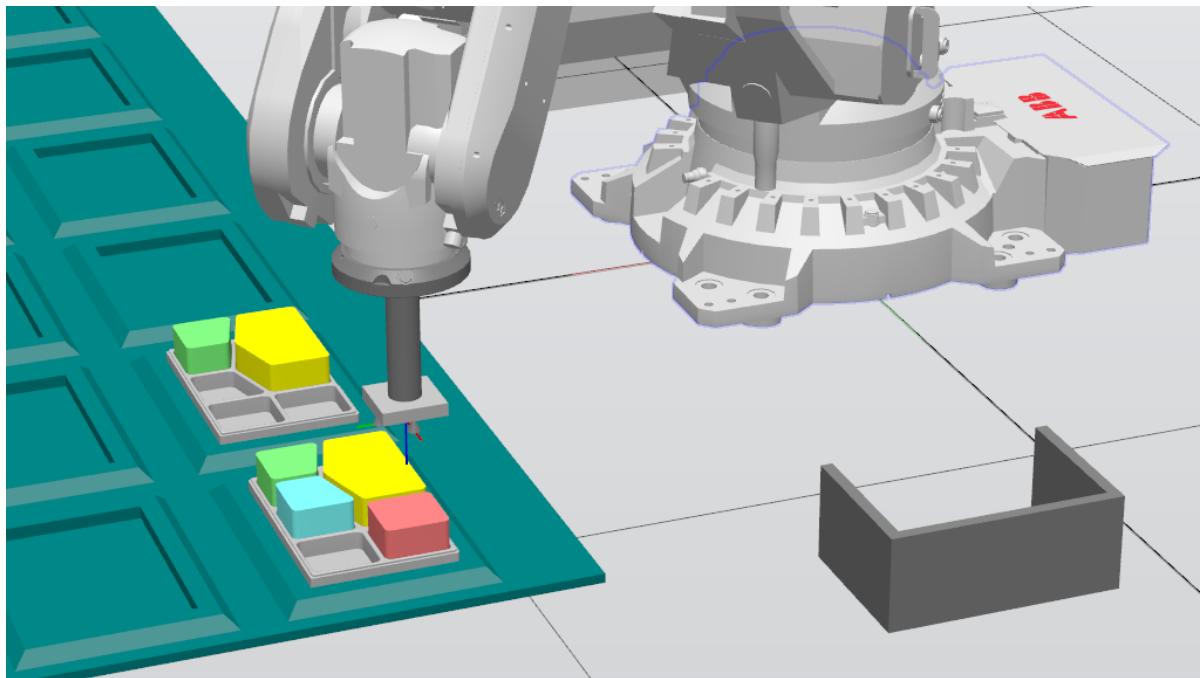


Figura 39. Robot colocando materiales en una caja.

Cuando la última ración está colocada, recoge la tapa y cierra la caja.

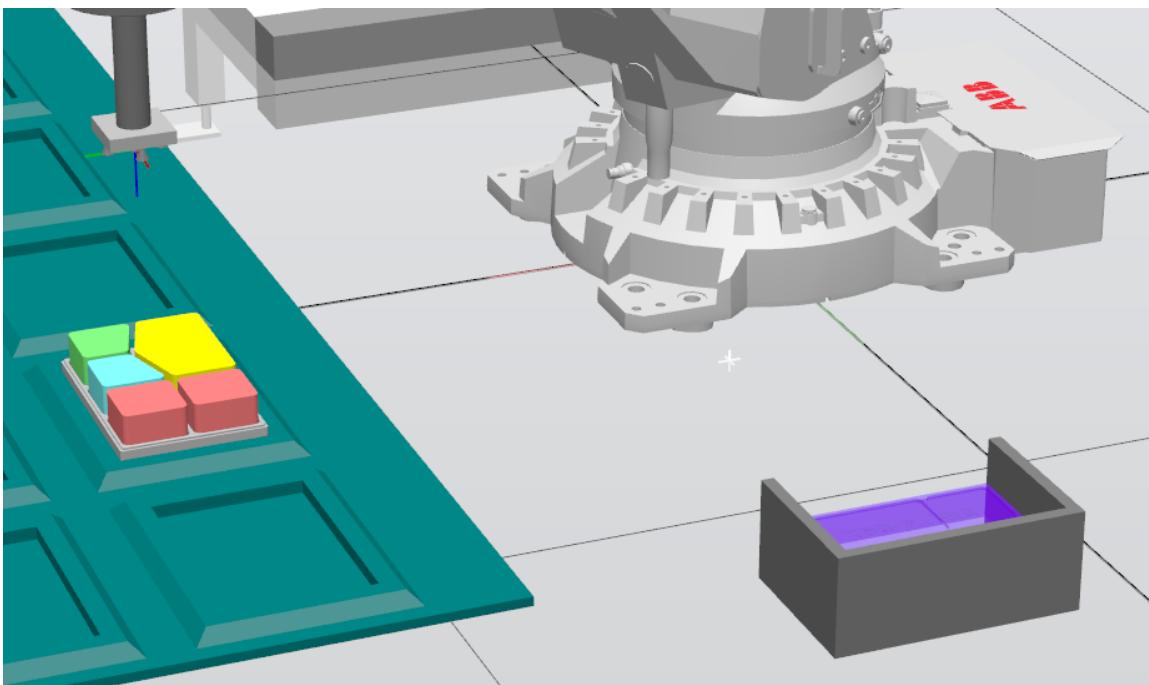


Figura 40. Estando todas las raciones en la caja, el robot se dirige a recoger la tapa.

Con la caja ya cerrada, el robot recoge la caja entera y la deposita en la cinta de precintado.

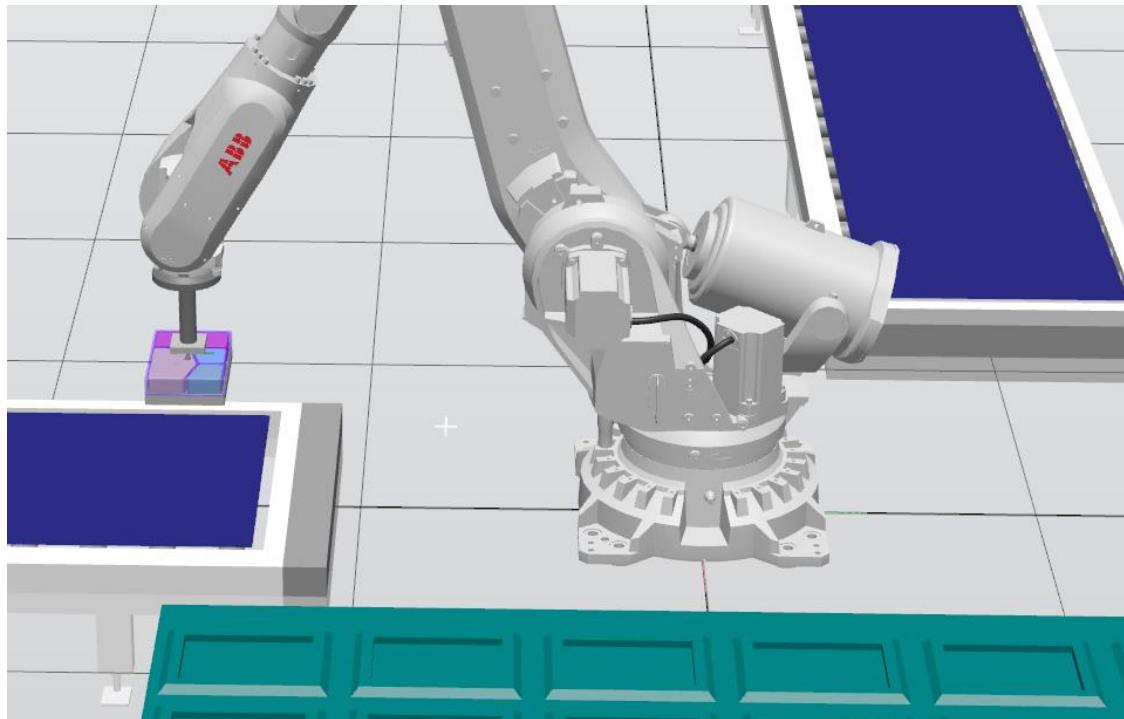
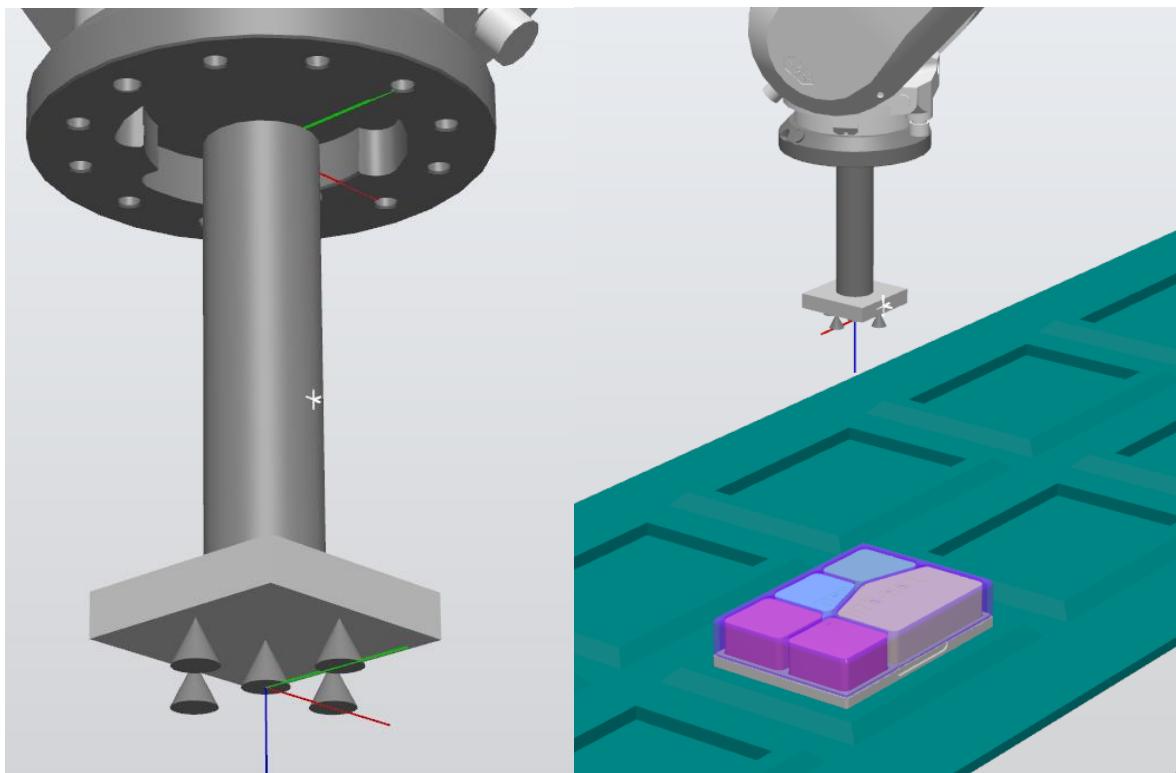


Figura 41. El robot recoge la caja completa y la deposita en la cinta de precintado.

## 11.2. El actuador: GarraVentosas

Para este trabajo se ha decidido recurrir a una herramienta similar al de unas ventosas, ya utilizada en otros ejercicios del Máster. La herramienta ha sido enlazada al robot y se localiza en el extremo de la muñeca final del robot.



**Figura 42. Actuador del robot: ventosa.**

Las ventosas siempre se situarán sobre el centro de masas del objeto para que el agarre sea óptimo. Disponen de dos señales digitales de entrada:

- **DI\_Vacío.** Al recibir un pulso sobre esta señal, activan el vacío de las ventosas y los objetos se unen al robot.
- **DI\_Soplar.** Deshace el vacío y el elemento es soltado.

La herramienta también dispone de un sensor de presencia como señal de salida que indica al robot que se está agarrando la pieza. Esto es importante si a mitad de trayecto el vacío falla y el objeto cae. Ante esta situación, el robot manda un mensaje de error de actuador al PLC, el controlador se detiene y muestra un mensaje de error.

## 11.3. Targets

El robot dispone de los siguientes targets:

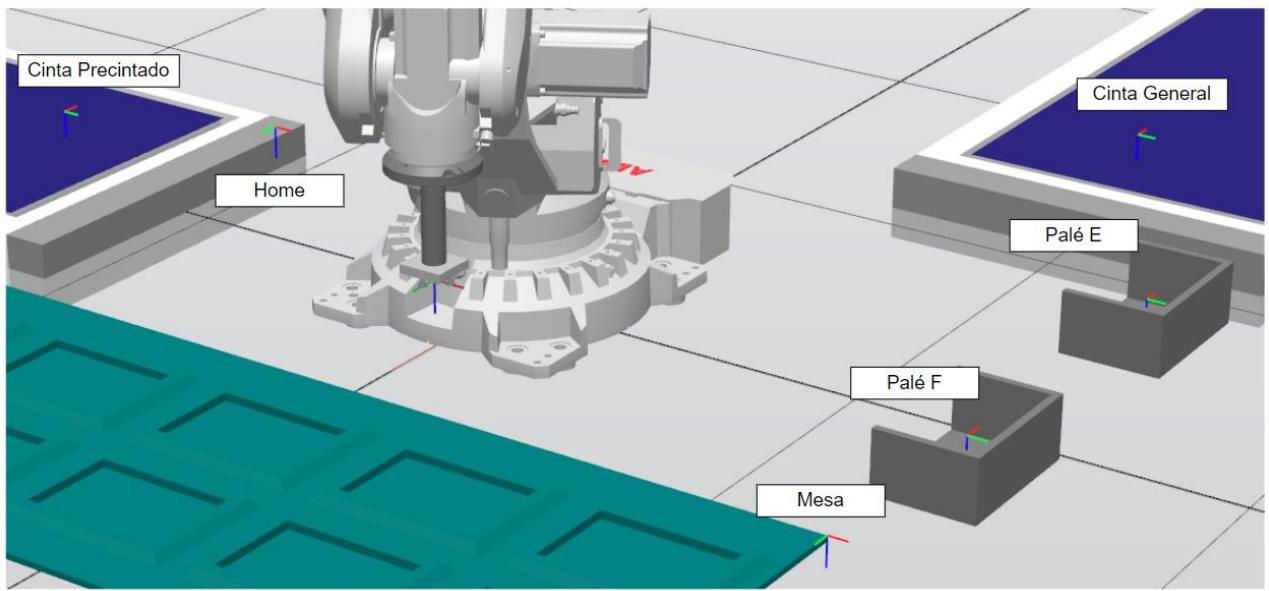


Figura 43. Targets del robot.

Puede observarse que **no se han creado targets distintos para cada una de las posiciones finales de las piezas** en cada una de las posibles posiciones dentro de la mesa de trabajo. Esto sería bastante ineficiente, luego se ha optado por la siguiente solución:

- 1) Calcular los centros de masa de las piezas con respecto el sistema de coordenadas de la caja.** Para ello, se utiliza geometría y análisis integral.

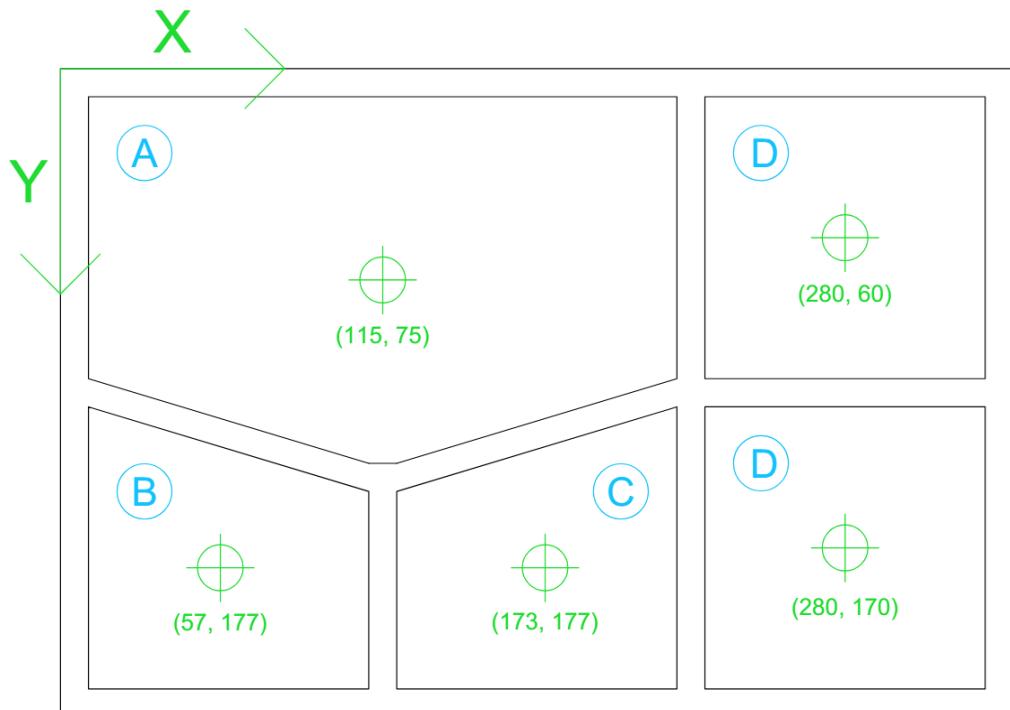


Figura 44. Centros de masa de las piezas. Sistema de referencia caja.

- 2) Calcular el sistema de coordenadas de la caja dada la posición de la mesa en referencia al **target de la mesa**. Para ello, dada la Mesa ( $X_m$ ,  $Y_m$ ) que es el número de fila y columna referida según el diagrama inferior, y conociendo las diferencias entre las posiciones de la mesa, se puede calcular la posición del sistema de coordenadas de la caja según la siguiente ecuación.

$$\begin{cases} x_{Mesa} = 200 + x_M \cdot 400 \\ y_{Mesa} = 80 + y_M \cdot 500 \end{cases}$$

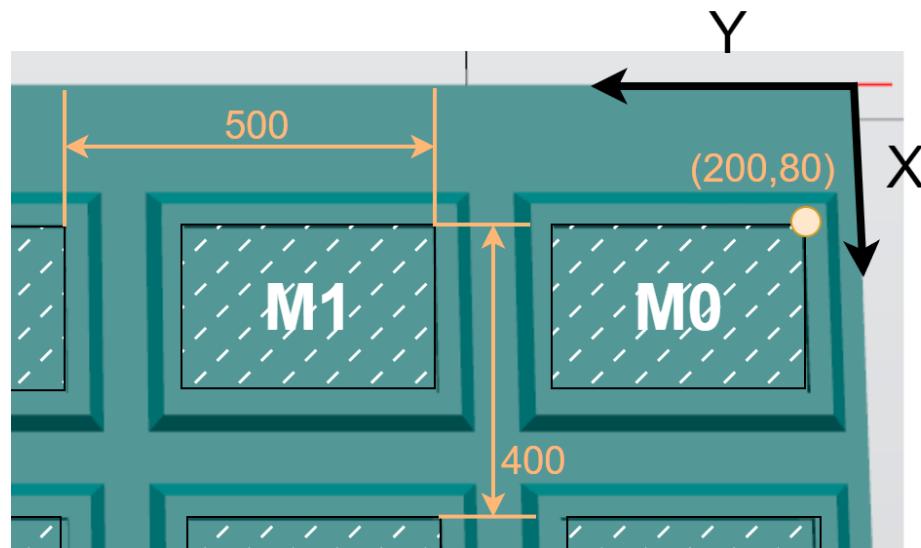
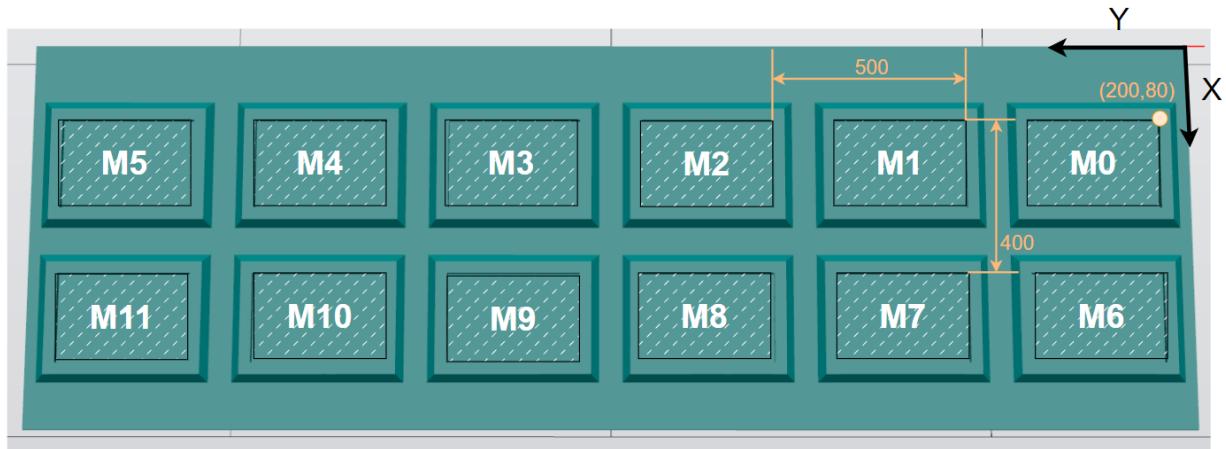


Figura 45. Sistema de coordenadas de la mesa.

- 3) Aplicar las transformaciones en orden: primero Mundo-Mesa, luego Mesa-Caja, y finalmente Caja-Pieza para calcular la posición final de la pieza.

La transformación Mundo-Mesa se debe a que la Y de la figura anterior tiene sentido contrario a la Y del mundo, luego hay que cambiarla de signo básicamente.

Para facilitar algo más las transformaciones, es buena idea orientar el origen de coordenadas de la mesa con el sistema de coordenadas de la caja; es decir, colocando el Y en la horizontal de la esquina superior derecha de la caja, mirando hacia la izquierda; y la X en la vertical de la esquina superior derecha de la caja, apuntando hacia abajo.

De esta manera se consiguen los centros de masa de piezas siguientes:

Material	Posición
A	(75, 225)
B	(177, 283)
C	(177, 167)
D1	(60, 60)
D2	(170, 60)
E	(115, 170)
F	(115, 170)

Tabla 19. Centros de masa de las piezas.

#### 11.4. Entradas y salidas

En la siguiente tabla se recogen las entradas y salidas digitales del sistema.

Nombre	Tipo	Definición
DI_SensorPieza	Entrada	1 si el sensor del actuador detecta una pieza debajo del mismo.
DO_ErrorRX	Salida	1 si ha ocurrido un error durante la transmisión.
DO_PiezaA	Salida	Se está transportando una pieza A.
DO_PiezaB	Salida	Se está transportando una pieza B.
DO_PiezaC	Salida	Se está transportando una pieza C.
DO_PiezaD	Salida	Se está transportando una pieza D.
DO_PiezaE	Salida	Se está transportando una pieza E.
DO_PiezaF	Salida	Se está transportando una pieza F.
DO_Soplar	Salida	Desactiva las ventosas para dejar de transportar una pieza.
DO_SoplarCaja	Salida	Desactiva las ventosas para dejar de transportar una caja.
DO_Vacio	Salida	Activa las ventosas para transportar una pieza.
DO_VacioCaja	Salida	Activa las ventosas para transportar una caja.

Tabla 20. Tabla de entradas y salidas del robot.

## 11.5. Objetos inteligentes

Los objetos inteligentes son herramientas de RobotStudio para facilitar la interacción del robot con otros objetos ajenos a este; en este caso, las piezas.

Ha sido necesaria la creación de dos objetos inteligentes.

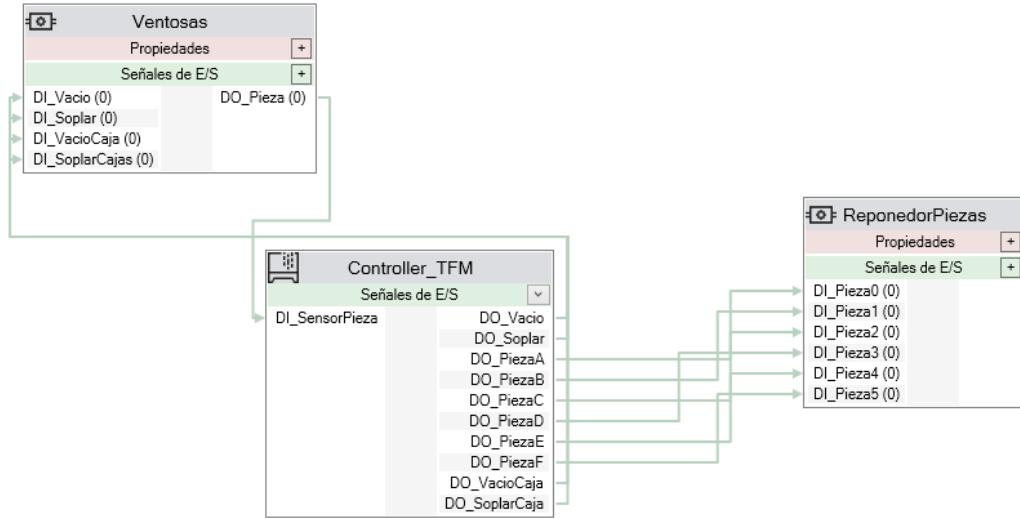


Figura 46. Lógica de la estación.

### 11.5.1. Ventosas

Simula el comportamiento real de las ventosas.

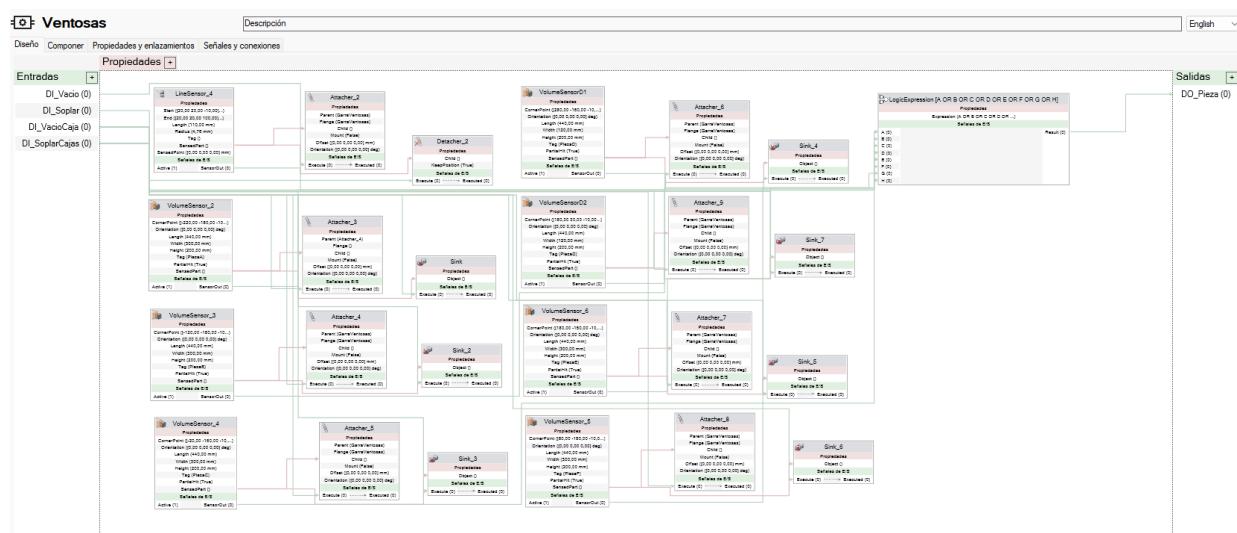
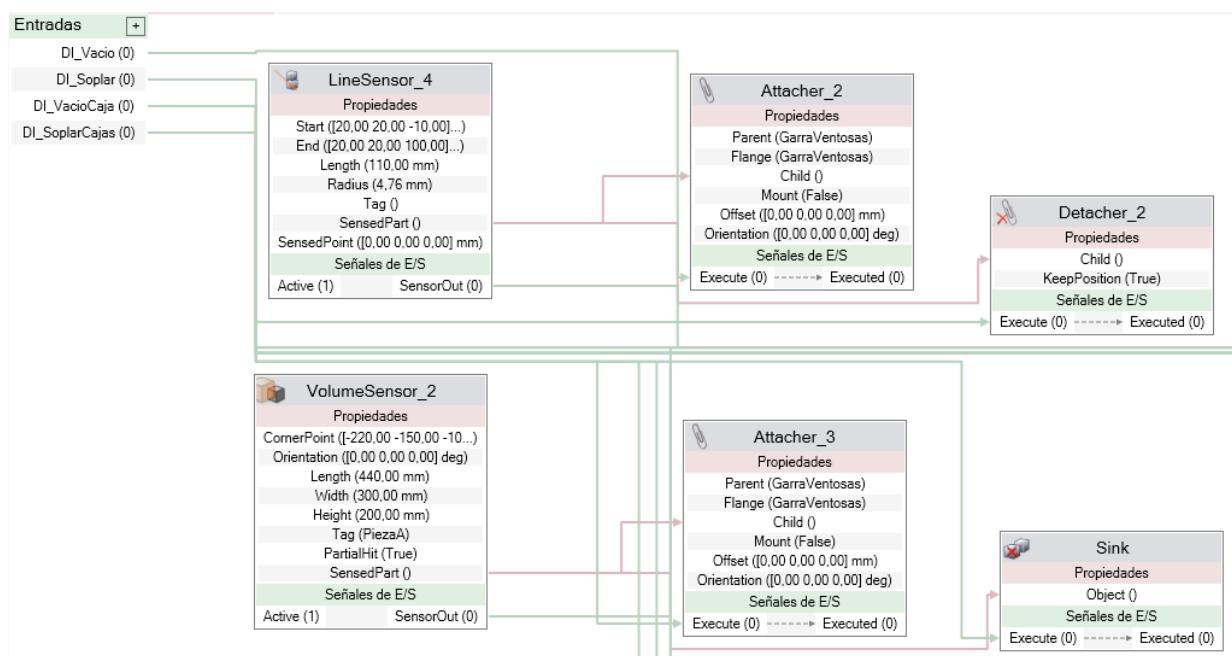


Figura 47. Vista completa del componente inteligente "Ventosas".

La posición de este objeto inteligente está enlazada con la transformada de GarraVentosas, el actuador del robot.

Tiene cuatro entradas: DI\_Vacio, DI\_Soplar, DI\_VacioCaja y DI\_SoplarCajas, con la misma utilidad que las entradas digitales homónimas del sistema. También tiene una salida digital DO\_Pieza, que es la que da el valor real de la señal digital de salida del sistema con el mismo nombre.

Pese a que la figura anterior pueda parecer intimidante, el comportamiento de “Ventosas” es muy sencillo: consta de un “LineSensor” y siete “VolumeSensor” que sirven para detectar la presencia de una pieza bajo el actuador Ventosas. El “LineSensor” sirve para detectar piezas cuando se están transportando individualmente y los “VolumeSensor” sirven para detectar todas las piezas a la vez cuando se transporta la caja completa y todas las piezas se deben mover al unísono.



**Figura 48. Detalle del componente inteligente "Ventosas".**

Sobre esto último, hay que indicar que cada “VolumeSensor” tiene un valor Tag diferente, y hay un sensor por cada Tag. Estos tags o etiquetas se utilizan para poder diferenciar las piezas. Cada pieza tiene su propio tag. Como cada sensor tiene la limitación de que solo puede detectar una pieza a la vez y, al transportar la caja completa, se deben detectar todas las piezas, se han colocado en la misma posición varios “VolumeSensor” con distintas etiquetas para que cada uno detecte una pieza de manera individual.

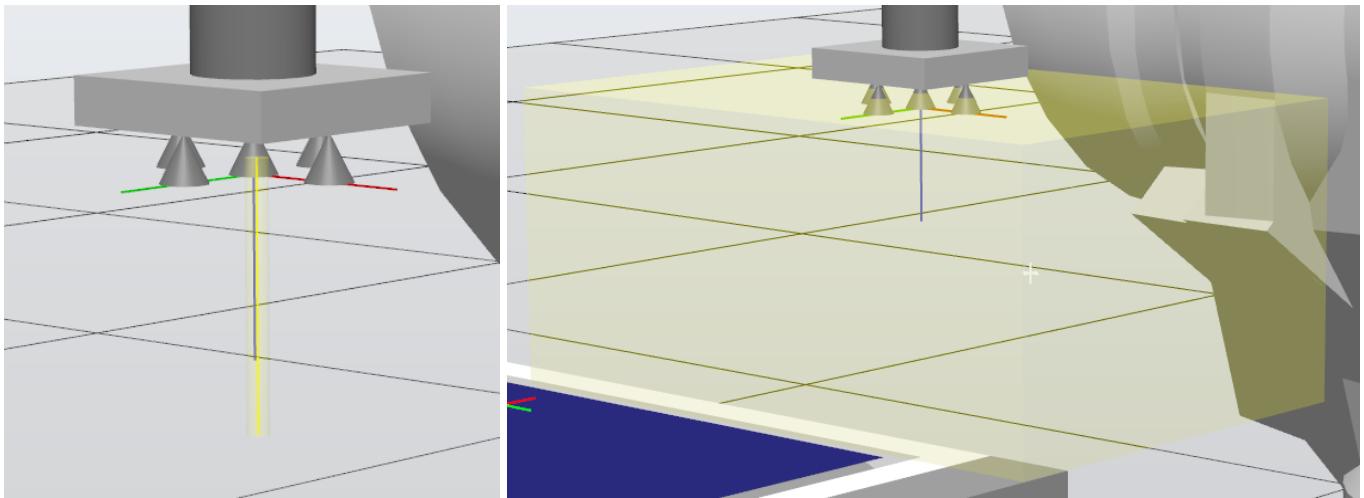


Figura 49. "LineSensor" y "VolumeSensor" del componente inteligente "Ventosas".

Otro detalle más para tener en cuenta es que hay dos componentes D y estos tendrán la misma etiqueta. Para poder detectarlos de manera individual, se han colocado dos "VolumeSensor" más estrechos, aprovechando que los componentes D tendrán una posición diferente en la caja final.

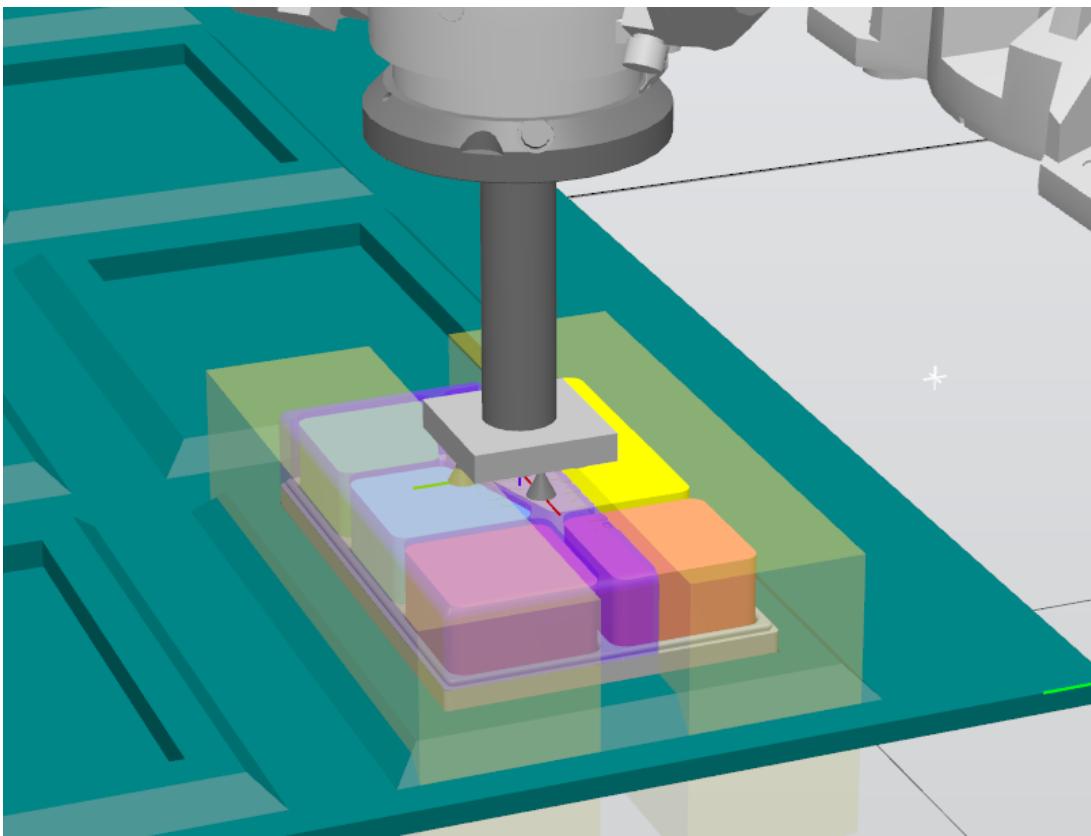


Figura 50. Detalle de los dos "VolumeSensor" para detectar los materiales D1 y D2.

Una vez la pieza ha sido detectada, el valor "SensedPart()" de cada sensor tendrá el identificador de la pieza.

El campo SensedPart del “LineSensor” está conectado a sendos “Attacher” y “Detacher”, los cuales se encargan de enlazar o desenlazar, respectivamente, la transformada del objeto a un *parent*, en este caso, GarraVentosas. Si DI\_Vacio se activa, activa el “Attacher”. Si DI\_Soplar se activa, activa el “Detacher”.

Las entradas DI\_VacioCajas y DI\_SoplarCajas realizan la misma función, pero utilizando un “Attacher” y un “Sink”. “Sink” elimina el objeto de la simulación. Cuando se recoge una caja, se activa DI\_VacioCajas y las transformadas de todas las piezas bajo la ventosa se enlazan con GarraVentosas. Cuando la caja es depositada en la cinta de precintado, se activa DI\_SoplarCajas y se elimina la caja completa con todas las piezas, simbolizando que esa caja ya ha pasado a la estación de precintado.

Finalmente, para crear DO\_Pieza se realiza la operación OR de todas las salidas “SensorOut” de los sensores.

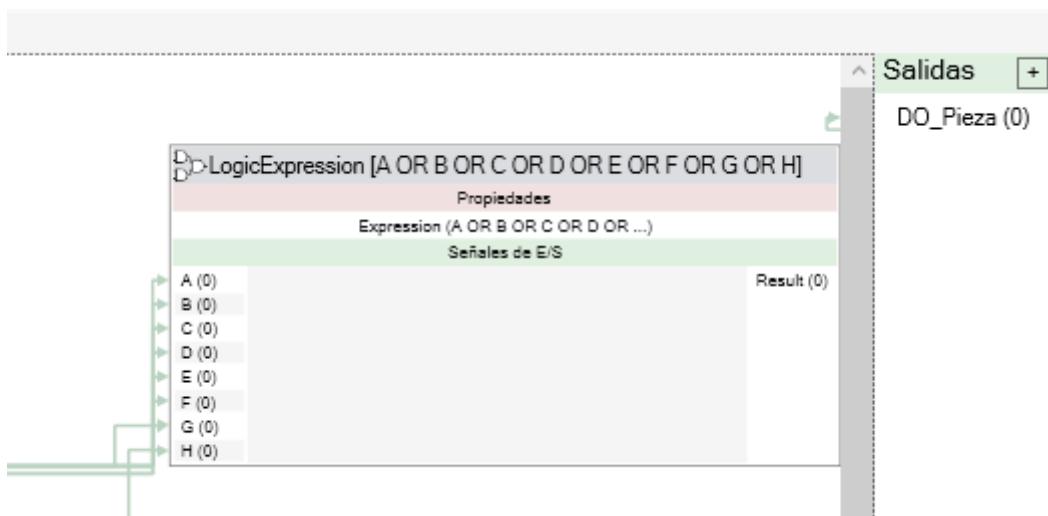


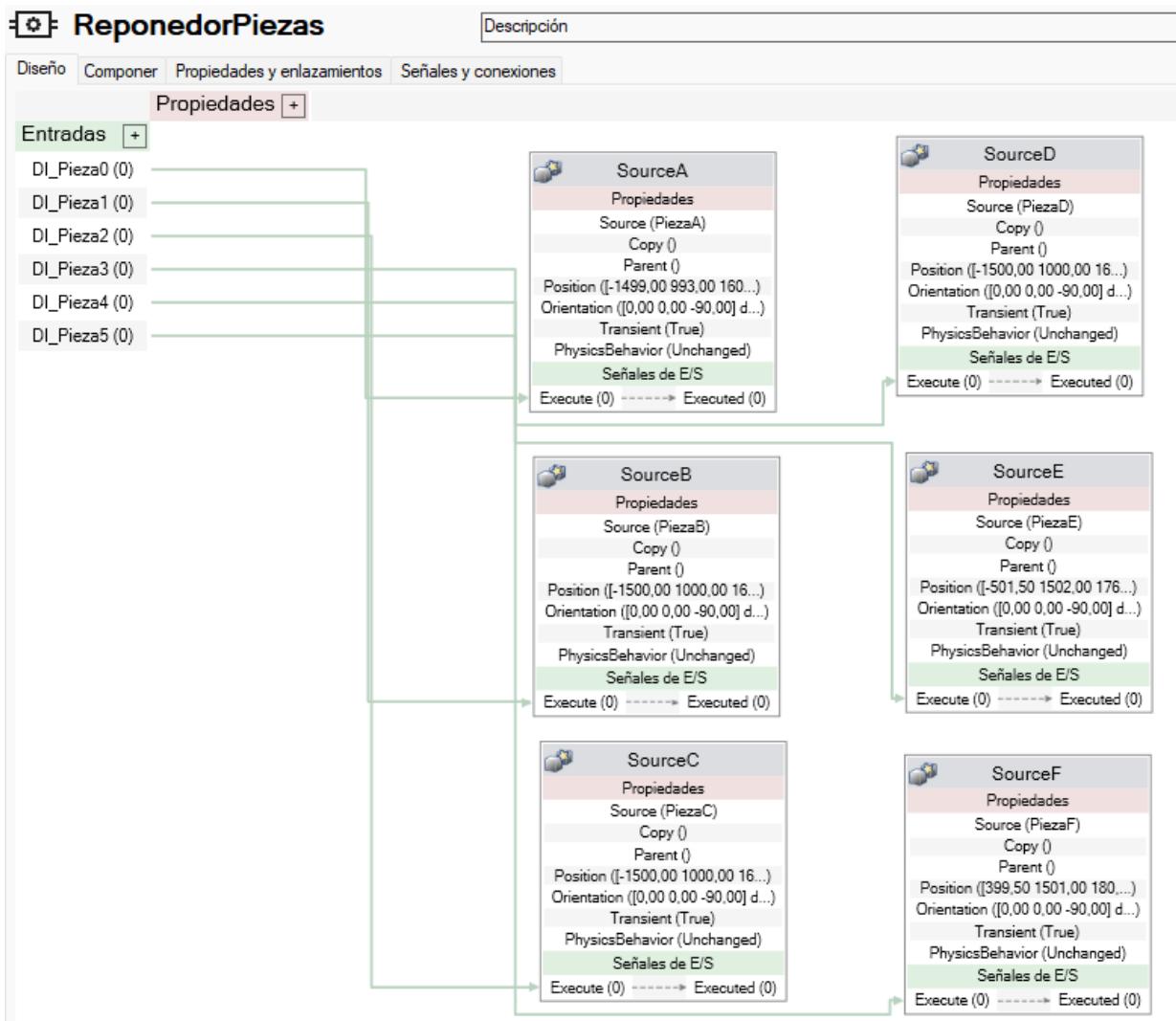
Figura 51. OR conjunto de todos los sensores para generar DO\_Pieza.

### 11.5.2. ReponedorPiezas

Este componente es muy sencillo. Consiste en seis bloques “Source” que generan una pieza dentro de la simulación, dependiendo de qué señal DO\_PiezaX esté activa.

Este bloque sirve para que las piezas se generen en la simulación cuando llegue un mensaje de “ColocarMaterial”.

En la siguiente hoja se muestra el componente.



**Figura 52. Componente inteligente "ReponedorPiezas".**

## 11.6. Código RAPID

En el Anexo 2 se presenta el código RAPID del robot con comentarios para clarificar la función de cada línea.

Con la presentación del código RAPID del robot se concluye la sección sobre el robot.

## 12. Sistema HMI

En esta sección se mostrará la configuración y diversas pantallas, botones, pilotos y mensajes del sistema HMI diseñados para este trabajo.

### 12.1. Configuración

La única configuración realizada para el sistema HMI ha sido la de su conexión ETHERNET en la que se debe fijar la IP del HMI.

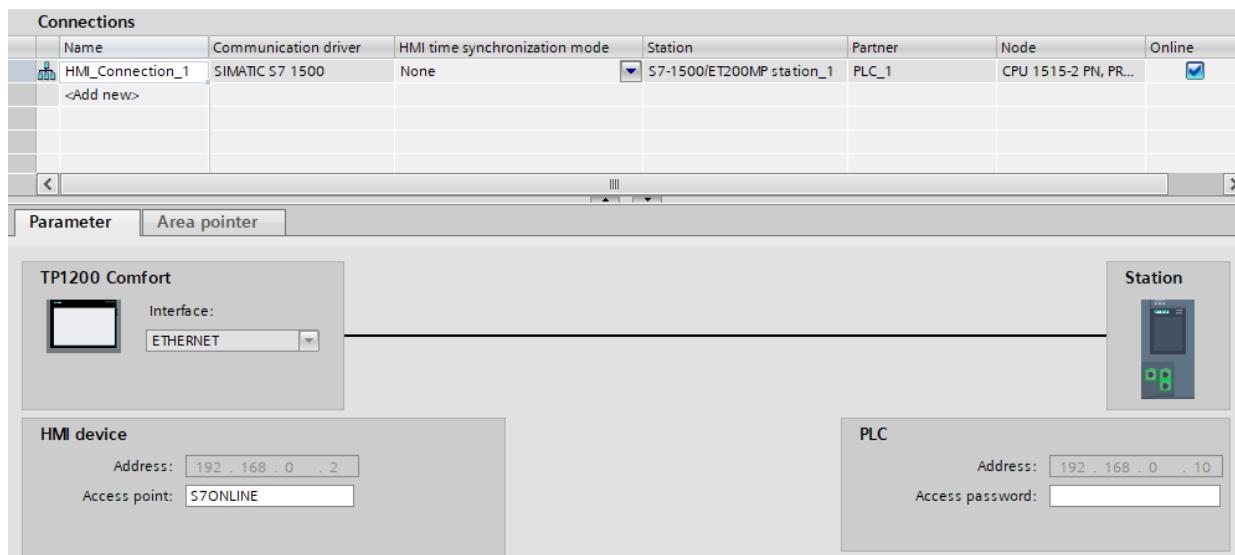


Figura 53. Configuración ethernet del sistema HMI.

### 12.2. Variables

La gran mayoría de variables que maneja el HMI son idénticas a aquellas mostradas en la sección dedicada al PLC. Las conexiones de las variables compartidas se han realizado de manera simbólica y con la tasa de refresco máxima (100 ms).

Se muestra la tabla de variables con sus nombres, tipos y equivalentes en el PLC.

Nombre	Tipo	Tags PLC
CintaEmpaquetado	Bool	CintaEmpaquetado
CintaGeneral	Bool	CintaGeneral
HMI_BaseSeleccionada	Bool	HMI_vbes.HMI_BaseSeleccionada
HMI_MesasDisponiblesDeseadas	Int	HMI_vbes.HMI_NumeroMesasDeseadas
HMI_PaleE	Bool	HMI_PaleE
HMI_PaleF	Bool	HMI_PaleF
HMI_PiezaDisponiblesDeseadas	Int	HMI_vbes.HMI_NumeroMaterialesDeseados
Continúa en la siguiente hoja...		

**Continuación...**

<b>Nombre</b>	<b>Tipo</b>	
<b>HMI_ReiniciarSistema</b>	Bool	HMI_ReiniciarSistema
<b>HMI_SP_CintaEmpaquetado</b>	Bool	HMI_SP_CintaEmpaquetado
<b>HMI_SP_CintaGeneral</b>	Bool	HMI_SP_CintaGeneral
<b>HMI_SP_Cintas</b>	Word	HMI_SP_Cintas
<b>HMI_TapaSeleccionada</b>	Bool	HMI_vbes.HMI_TapaSeleccionada
<b>HMI_vbes_HMI_ApagarRobot</b>	Bool	HMI_vbes.HMI_ApagarRobot
<b>HMI_vbes_HMI_Errores</b>	Word	HMI_vbes.HMI_Errores
<b>HMI_vbes_HMI_MandarHome</b>	Bool	HMI_vbes.HMI_MandarHome
<b>HMI_vbes_HMI_MaterialDestino</b>	Word	HMI_vbes.HMI_MaterialDestino
<b>HMI_vbes_HMI_MesaDestino</b>	Word	HMI_vbes.HMI_MesaDestino
<b>HMI_vbes_HMI_MostrarPiezaE</b>	Bool	HMI_vbes.HMI_MostrarPiezaE
<b>HMI_vbes_HMI_MostrarPiezaF</b>	Bool	HMI_vbes.HMI_MostrarPiezaF
<b>HMI_vbes_HMI_PausarSistema</b>	Bool	HMI_vbes.HMI_PausarSistema
<b>HMI_vbes_HMI_ValorMesaSelecc</b>	Word	HMI_vbes.HMI_ValorMesaSeleccionada
<b>MesaSeleccionada</b>	Bool	HMI_vbes.HMI_MesaSeleccionada
<b>NumeroMesaSeleccionada</b>	Int	HMI_vbes.HMI_NumeroMesaSeleccionada
<b>SP_CintaEmpaquetado</b>	Bool	SP_CintaEmpaquetado
<b>SP_CintaGeneral</b>	Bool	SP_CintaGeneral
<b>SP_Cintas</b>	Word	SP_Cintas
<b>vbes_ElementoBase</b>	Word	SISTEMA_vbes.ElementoBase
<b>vbes_ElementoTapa</b>	Word	SISTEMA_vbes.ElementoTapa
<b>vbes_Material_Destino</b>	Word	SISTEMA_vbes.Material_Destino
<b>vbes_ModoSimulacion</b>	Bool	SISTEMA_vbes.ModoSimulacion
<b>vbes_NumeroMateriales</b>	Int	SISTEMA_vbes.NumeroMateriales
<b>vbes_NumeroMesas</b>	Word	SISTEMA_vbes.NumeroMesas
<b>vbes_RobotConectado</b>	Bool	SISTEMA_vbes.RobotConectado
<b>vbes_RobotOcupado</b>	Bool	SISTEMA_vbes.RobotOcupado

**Tabla 21.** Variables compartidas del HMI con el PLC.

También se añaden las variables propias del PLC.

<b>Nombre</b>	<b>Tipo</b>
<b>Tag_ScreenNumber</b>	UInt
<b>SP_CintaGeneral_Forzado</b>	Bool
<b>SP_CintaEmpaq_Forzado</b>	Bool

**Tabla 22.** Variables propias del HMI.

Estas variables internas sirven para simular los sensores de presencia de las cintas y para mantener su valor de manera constante sin tener que pulsar continuamente un botón.

## 12.3. Ventanas

Además de las ventanas automáticamente creadas al añadir un HMI a un proyecto con PLC en TIA Portal, se han añadido las siguientes al trabajo.

### 12.3.1. Pantalla simulación

En la pantalla de simulación se permiten cambiar los valores de entrada del sistema. Esta pantalla sobre todo ha sido creada para facilitar la realización de las pruebas del sistema, pues permite introducir entradas sin tener que forzar valores desde TIA Portal.

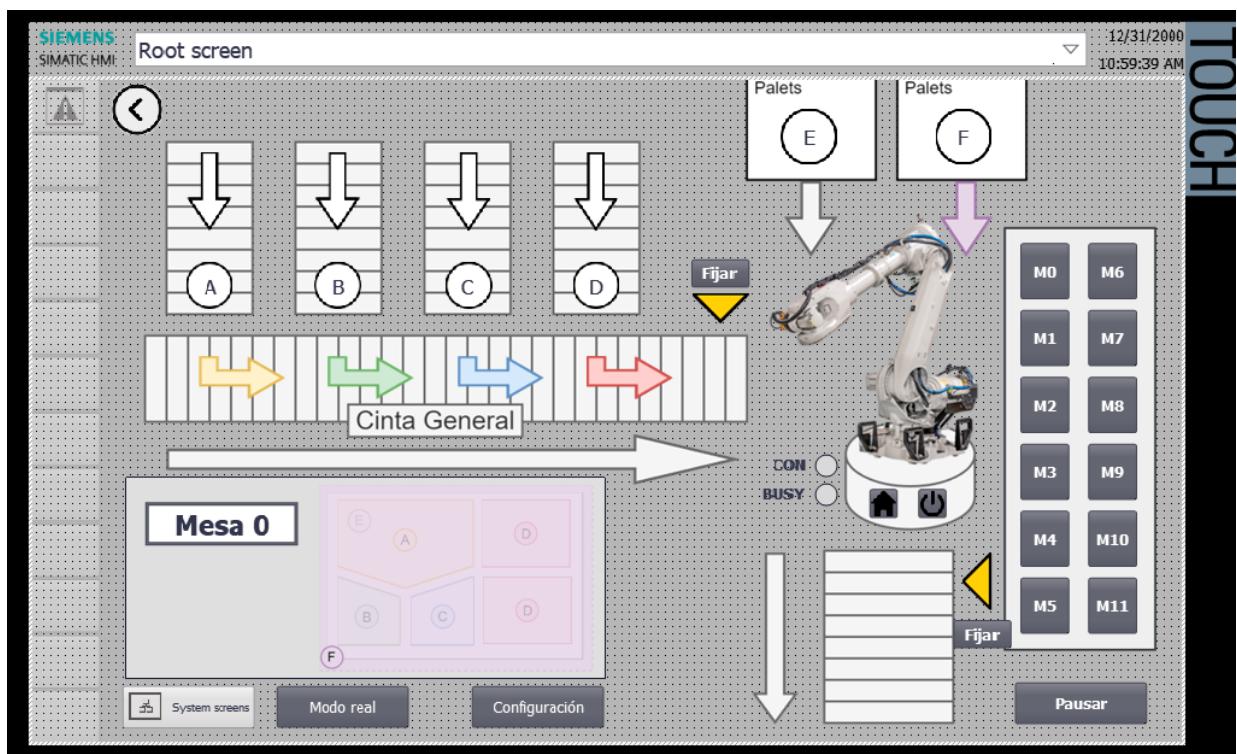


Figura 54. Pantalla Simulación, vista desde TIA Portal.

Este es el funcionamiento de cada uno de los componentes de la ventana de simulación:

- **Modo real.** Cambia del modo de simulación al modo real.
- **Configuración.** Abre la ventana de configuración.
- **Robot.** Dispone de dos pilotos, uno para mostrar si está conectado (rojo desconectado, verde conectado) y otro para mostrar si está realizando algún trabajo (gris robot desocupado, naranja y parpadeando ocupado). Tiene dos botones, uno para mandar a HOME al robot y otro para apagarlo.

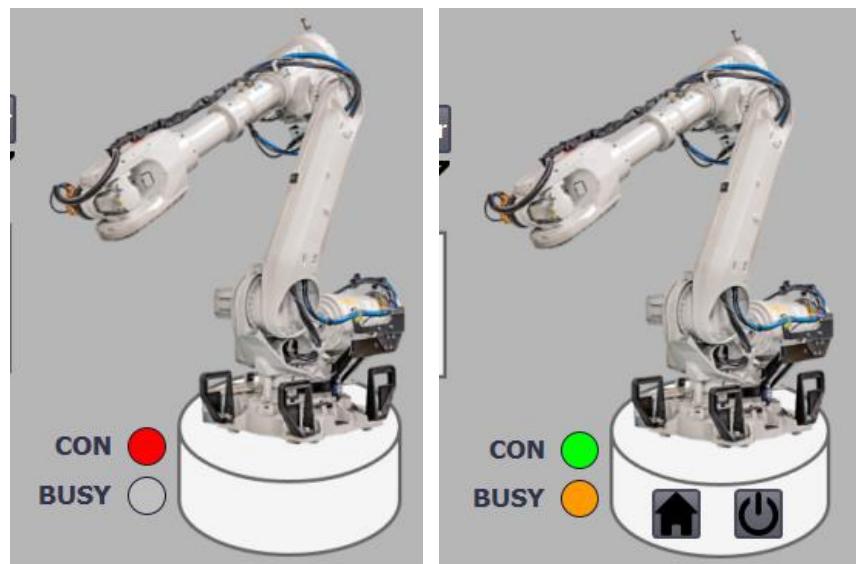


Figura 55. Robot en el HMI.

- **Cintas de materiales.** Cada cinta consta de un botón que simula la presencia de material en la misma. Si el PLC decide escoger ese material para mandarlo a la cinta general, aparece una flecha del mismo color del material bajo la cinta.

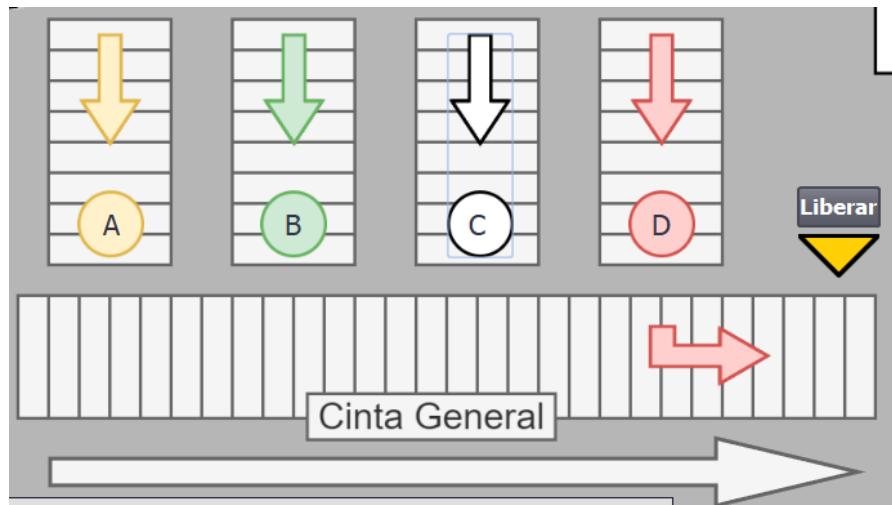


Figura 56. Cintas de materiales y cinta general en el HMI.

- **Palés.** Los palés tienen un comportamiento similar al de las cintas. Cuando el robot está recogiendo uno de los palés aparece una flecha con el color de la pieza.

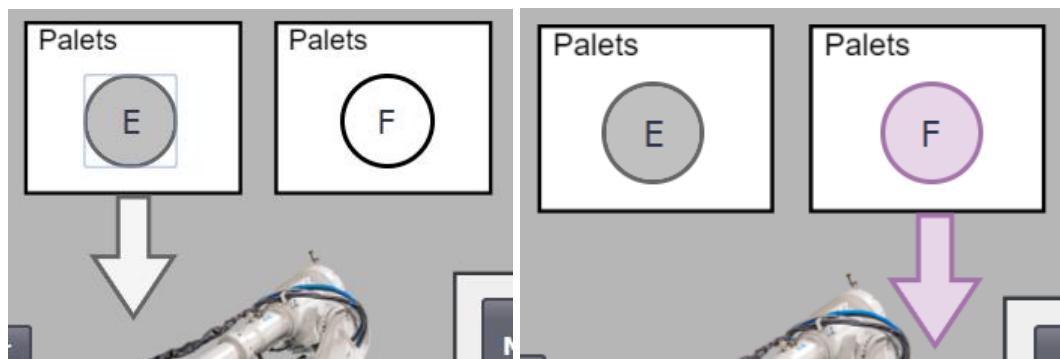


Figura 57. Palés en el HMI.

- **Cinta general.** La cinta general tiene una flecha en la parte inferior que se ilumina cuando esta está siendo activada desde el PLC. En su final, hay un sensor de presencia (triángulo amarillo cuando está activado y gris en caso contrario) que se puede pulsar momentáneamente o dejar fijado mediante el botón “Fijar”.
- **Cinta de empaquetado.** Su funcionamiento es similar al de la cinta general.
- **Mesa de trabajo.** Al entrar en la pantalla simulación, se procesan los doce botones de la mesa para que solo se muestren tantos botones como mesas haya en la configuración del PLC. El botón de la mesa se iluminará en cian cuando el robot esté realizando un trabajo en esta.

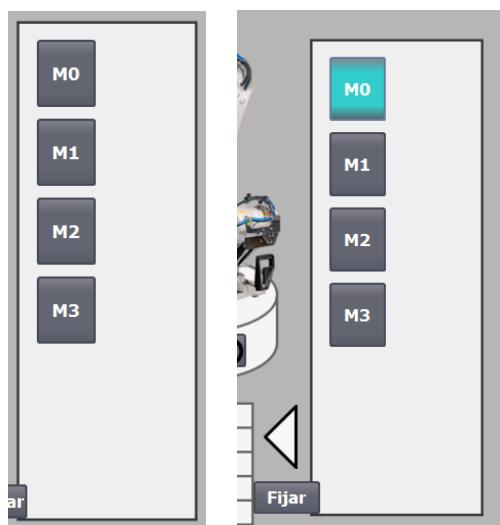


Figura 58. Mesa destino (en azul) mostrada en el HMI.

- **Vista de la caja.** Si se presiona el botón de alguna de las mesas, aparecerá una ventana en la esquina inferior izquierda con el contenido actual de la caja en esa posición. Esto ha sido realizado mediante el script VBA “FuncBotonMesa”.

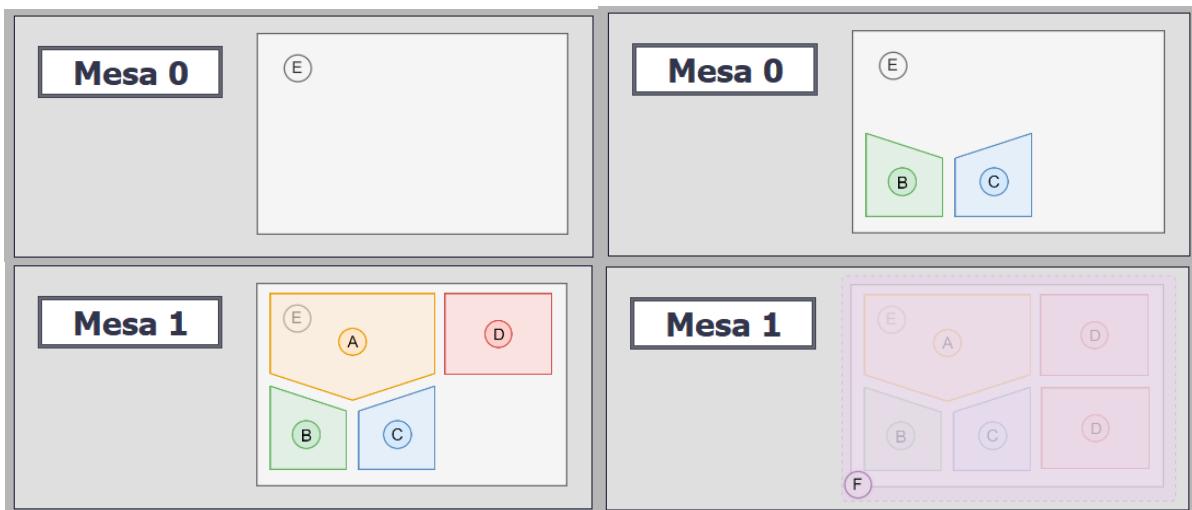


Figura 59. Progreso del contenido de las cajas por mesa en el HMI.

### 12.3.2. Pantalla real

En esta pantalla se muestra la situación actual de la planta utilizando las entradas reales del sistema. Las señales reales corresponden a las entradas digitales de los sensores de presencia de las cintas de transporte.

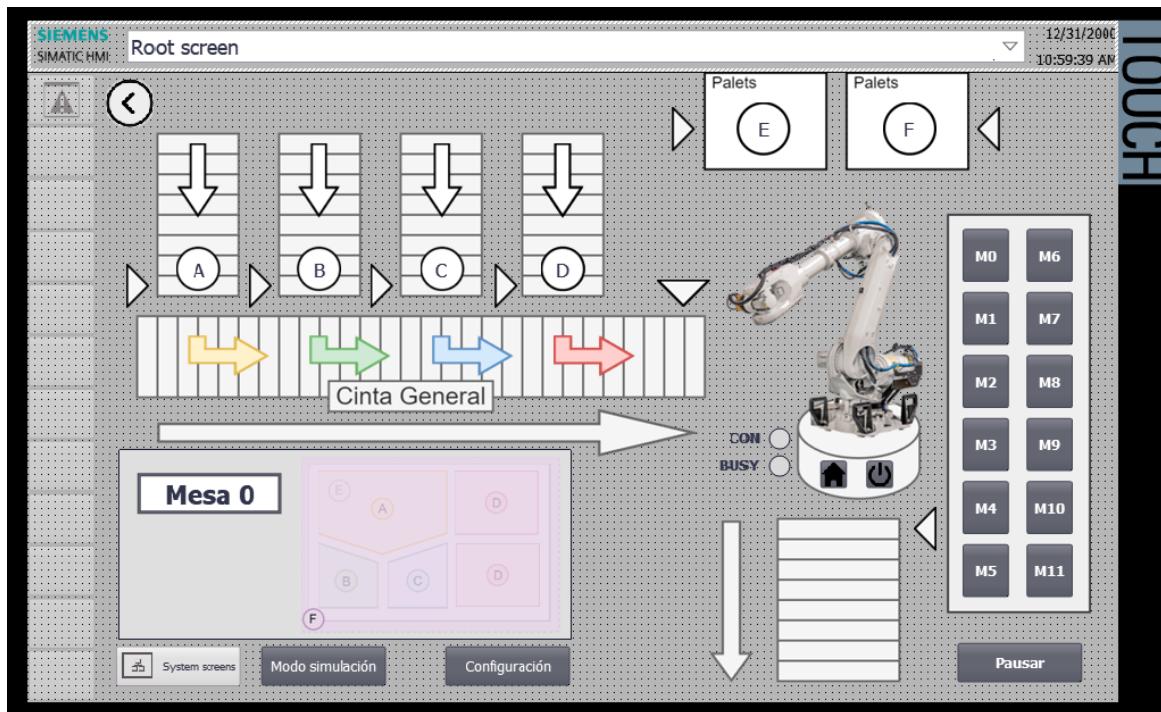


Figura 60. Pantalla real, vista desde TIA Portal.

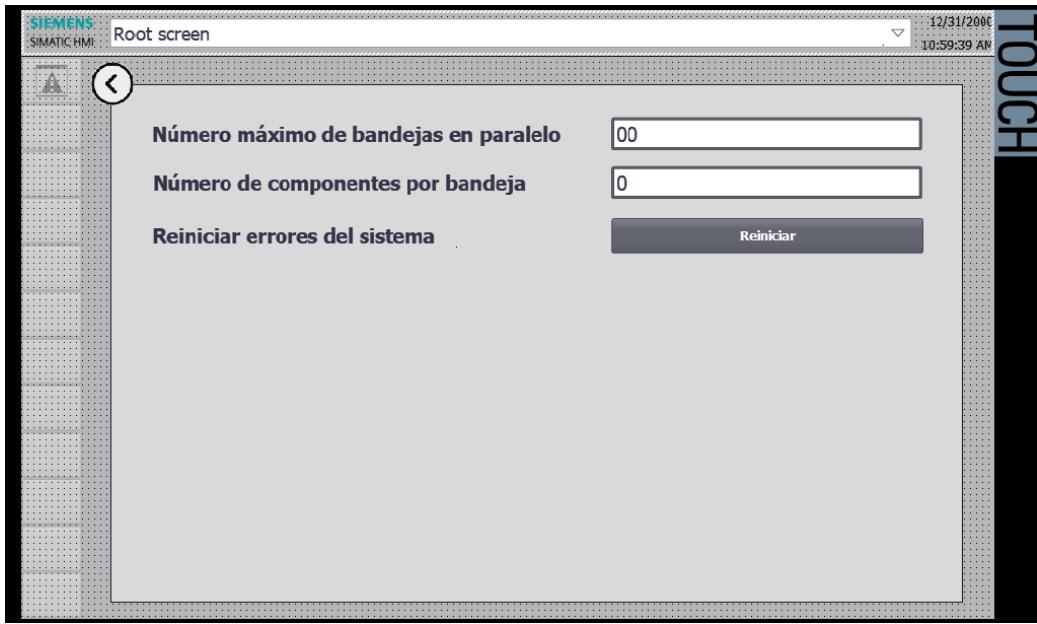
Esta pantalla es muy similar a la del modo de simulación con la diferencia de que no se pueden forzar los sensores de presencia de las cintas.

### 12.3.3. Ventana principal (Root Screen)

La ventana principal consta únicamente de un botón de inicio para arrancar el sistema. Esta ventana tiene una función principal y es que, al cargar la pantalla, se llama al script “IniciarBotonesMesas” el cual se encarga de ocultar los botones de la mesa según el número de mesas que haya en la configuración del PLC. Si en vez de empezar el HMI en esta ventana comenzase en la “Pantalla real”, por algún motivo, no se ejecuta correctamente el script y aparecen todos los botones de las mesas.

### 12.3.4. Pantalla configuración

En la pantalla de configuración se permite modificar los parámetros del PLC. Para acceder a ella, se debe pulsar el botón “Configuración”.



**Figura 61.** Pantalla de configuración, vista desde TIA Portal.

## 12.4. Alarmas

En la siguiente tabla se recogen todas las alarmas del HMI. Todas se obtienen del tag “HMI\_vbes\_HMI\_Erroros”.

ID	Nombre	Texto de alarma	Bit
1	Error_ID	La instrucción enviada al robot no fue reconocida por el mismo	0
2	Error_Mat	El ID del material enviado al robot no es válido o no fue reconocido por el robot	1
3	Error_Mesa	El número de mesa enviado al robot no es válido o no fue reconocido por el robot	2
4	Error_Len	La longitud del mensaje enviado no es correcta	3
5	Error_Disc	El robot se ha desconectado	4
6	Error_Fatal	El robot ha tenido un fallo fatal y deberá ser rearmado	5
7	Error_Conf_Mat	Error de numero de componentes por bandeja enviado en la configuración al robot	6
8	Error_Conf_Mesa	Error de numero de mesas enviado en la configuración al robot	7
9	Error_Act	El sensor de material del robot ha fallado mientras transportaba material.	8

**Tabla 23.** Alarmas del HMI.

## 12.5. Programación VBA

En el Anexo 3 se añade al trabajo el código de los scripts VBA con explicaciones y comentarios sobre su funcionamiento.

## 13. Entorno de simulación

En este apartado se explica cómo configurar el ordenador para poder realizar las simulaciones del sistema.

1. Instalar la versión v4.0 del S7-PLCSIM Advanced.
2. Abrir el “Panel de control” > “Centro de redes y recursos compartidos”.
3. Acceder a “Cambiar configuración del adaptador”.

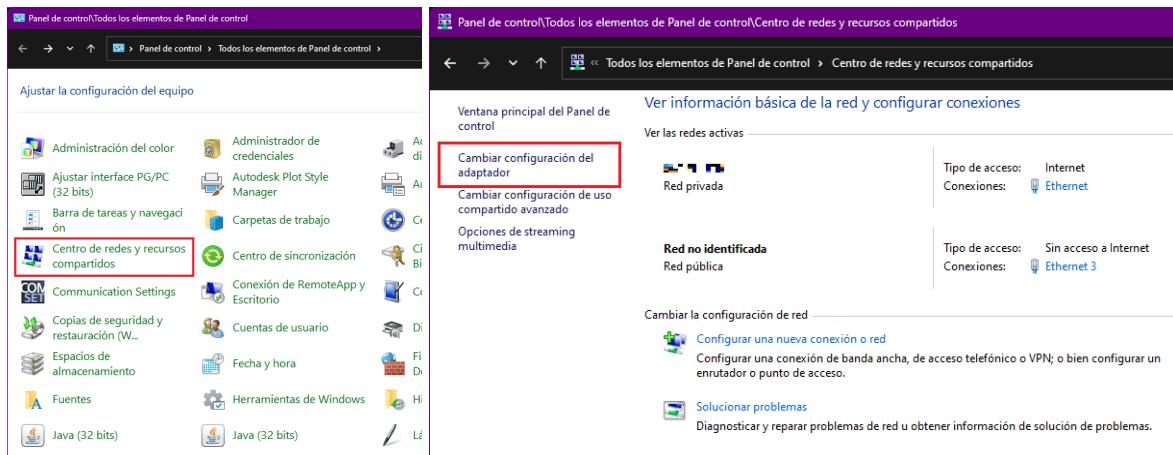


Figura 62. Guía de simulación, puntos 2 y 3.

4. Click derecho sobre el adaptador de nombre “Siemens PLCSIM Virtual Ethernet Adapter” > “Propiedades”.
5. Desciende dentro de la lista hasta encontrar “Protocolo de Internet Versión 7 (TCP/IPv4)”.

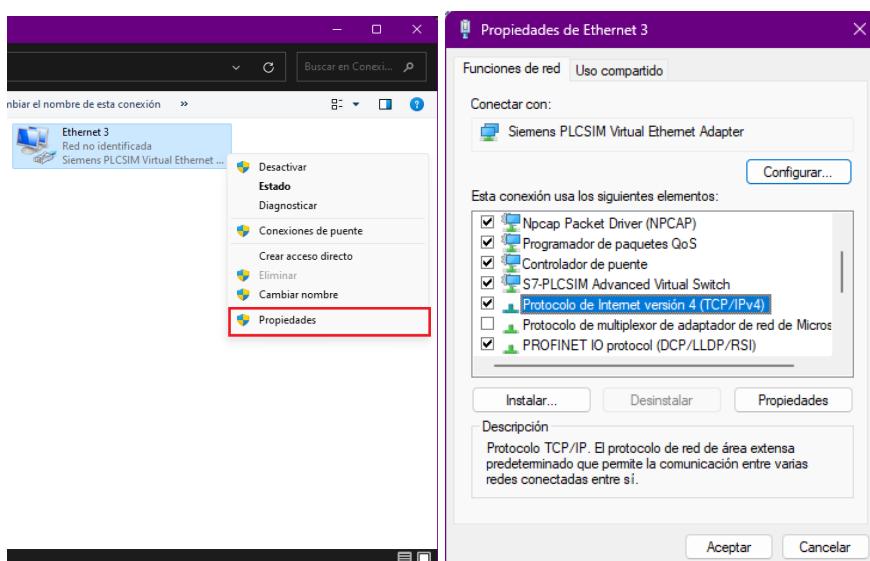
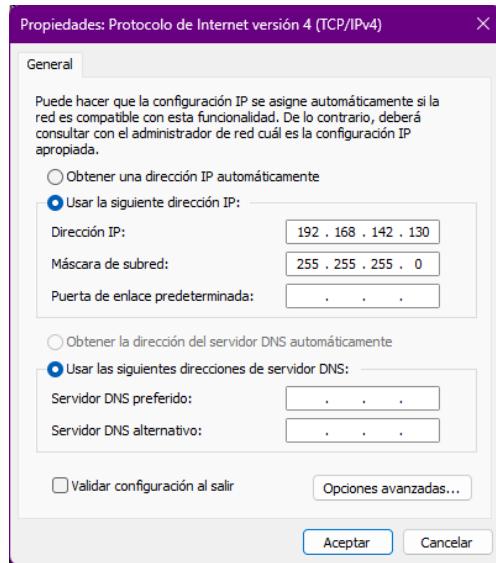


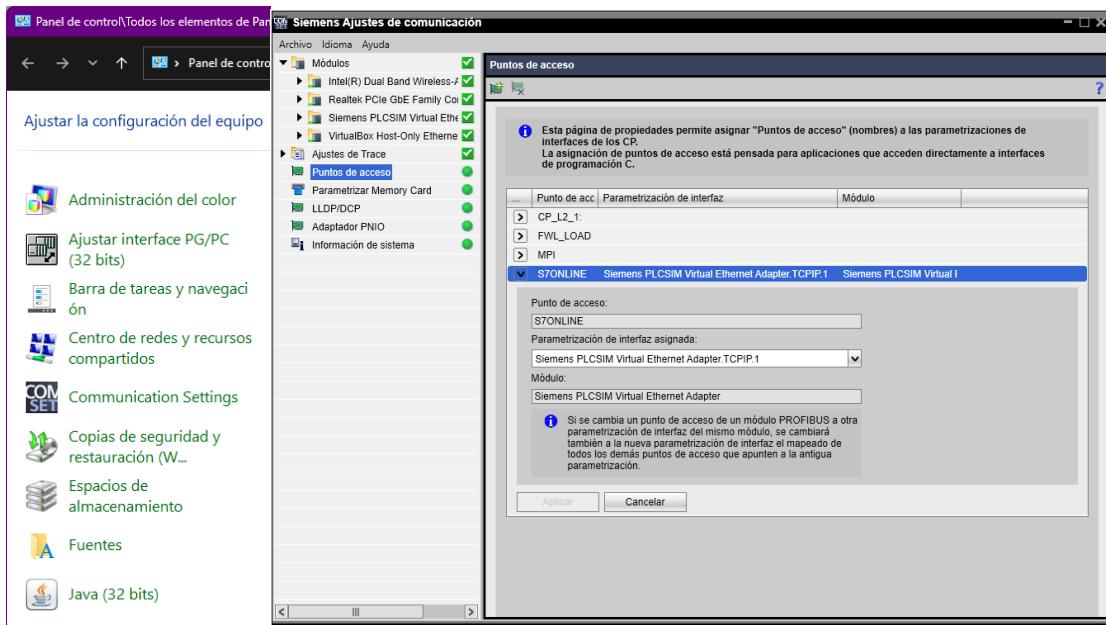
Figura 63. Guía de simulación, puntos 4 y 5.

6. Doble click sobre el mismo. Introduce la IP y la máscara de red del **SERVER** (no es la IP del PLC); sino la que se encuentra dentro de “DB\_PropsConexion” en el PLC.



**Figura 64. Guía de simulación, punto 6.**

7. Abrir ahora el “Communication Settings” del Panel de Control.
8. Seleccionar puntos de acceso > S7ONLINE > Siemens PLCSIM Virtual Ethernet Adapter.TCP/IP.1.



**Figura 65. Guía de simulación, punto 7 y 8.**

### 13.1. Simulación de la planta

Para realizar la simulación es necesario un nexo o túnel entre la interfaz de red del PLC y la del robot. Para ello, se debe utilizar el programa en Python incluido en el Anexo 4.

En los enlaces de los anexos se añaden vídeos donde se simula la planta completa.

## 14. Presupuesto

Como se detalla en el alcance del proyecto, únicamente se tendrán en cuenta los componentes de la planta que se detallan en el proyecto, y no los otros componentes que rodean a la planta.

<b>PLC</b>	RelePro.com	
CPU 1515-2 PN	3,179.67 €	3,683.53 €
PS 25W 24VDC	250.52 €	
DI 32x24VDC BA	126.67 €	
DQ 16x24VDC	126.67 €	
<b>HMI</b>	RelePro.com	2,601.26 €
TP1200 Comfort	2,601.26 €	
<b>Robot</b>	ABB	23,000.00 €
IRB 6700-235/2.65	18,000.00 €	
IRC5	5,000.00 €	
<b>Licencias software</b>	plc-city.com	2,964.62 €
TIA Portal V17	1,358.33 €	
S7-PLCSIM Advanced V4.0 (90 días)	276.93 €	
WinCC Advanced	1,329.36 €	
Switch Moxa EDS-205	109.98 €	109.98 €
<b>Margen de beneficio</b>		10,000.00 €
<b>Coste total del proyecto</b>		<b>42,359.39 €</b>

Tabla 24. Presupuesto del proyecto.

## 15. Conclusiones

### 15.1. Conclusiones del trabajo

Tras haber concluido el trabajo y utilizando el entorno de simulación desarrollado, se ha comprobado que el proyecto cumple con todos los requisitos que se especificaron durante la propuesta del mismo.

- Realizar el diagrama GRAFCET del sistema e implementarlo en un PLC.
- Programación del brazo robótico el cual deberá recoger las piezas y colocarlas en su lugar correspondiente siguiendo el orden de prioridad.
- Simulación individual del brazo robótico.
- Diseño de una interfaz HMI descriptiva del sistema con controles para usuario.
- Simulación individual del PLC y sistema HMI en TIA Portal.
- Investigación de la comunicación entre RobotStudio y TIA Portal mediante protocolo TCP/IP para simular el entorno real.
- Desarrollo del protocolo de comunicación entre PLC y robot.
- Simulación conjunta del PLC, HMI y robot.

### 15.2. Conclusiones Personales

Este trabajo me ha servido para poner en práctica todos los conocimientos que he adquirido a lo largo del máster e incluso me ha hecho ir más allá, recibiendo una experiencia similar a aquella que tendría a la hora de trabajar por mi cuenta para realizar proyectos de automatización o robótica. He tenido que investigar mucho sobre tópicos de los que desconocía y, sobre todo, he aprendido a ser autosuficiente y resolutivo.

Pienso que el algoritmo de este sistema parece sencillo una vez desarrollado, pero creo que en la simplicidad de la solución radica la belleza del mismo: no ha sido necesario recurrir a muchos bloques o funciones complejas para hacer funcionar el sistema y creo que eso es algo importante de valorar. A más sencillo el sistema, más sencillo también de mantener.

## 16. Limitaciones y prospectiva

Una de las mayores limitaciones de este trabajo ha sido el número máximo de hojas a escribir. El lector podrá haberse percatado que no se han dedicado muchas hojas al robot o al sistema HMI, pese a que han tomado mucho tiempo en realizarse y tienen mucho contenido que describir. Esto ha sido por el límite de hojas, el cuál no estuve teniendo cuenta hasta haber terminado la sección del robot.

También me hubiese gustado añadir una sección de resultados de simulación, pero va a tener que quedar relegado a unos enlaces a videos externos.

Pienso que, con la ayuda de los anexos, más o menos he podido completar el trabajo en estas hojas. Sin embargo, creo que hubiese sido mejor haber introducido el código como parte de las propias secciones para hacer una lectura más amena, no como en este caso en el que habrá que saltar muchas hojas para llegar al código y después otras más para volver atrás. Entiendo que la norma estará por algo, pero pienso que aquellos trabajos que necesiten de más texto, como ha sido mi caso, acabarán reducidos drásticamente por el tope impuesto y por lo tanto no quedarán tan completos como deberían.

Sobre el futuro o posible aplicación del trabajo creo que hay una barrera gigantesca que necesitaría ser cubierta para poder dar salida a esta planta empaquetadora, y esa es el lenguaje. Casi no he podido añadir nada concreto en la sección del estado del arte porque buscaba por páginas en inglés o en español y claro, las páginas que yo buscaba estarían en japonés. Es algo para tener en cuenta por si alguno de los estimados lectores de este trabajo desea poner en práctica lo leído en todas estas páginas.

Para el futuro también me gustaría que a este proyecto se le añadiese un

## ● Referencias Bibliográficas

---

ABB, 2010. Technical reference manual RAPID Instructions, Functions and Data types. *ABB AB.*

HMI, S., 2016. WinCC V7.4 WinCC: Scripting (VBS, ANSI-C, VBA). *System Manual.*

ROBOTICS, 2017. Technical reference manual: RAPID Instructions, Functions and Data. *RobotWare 6.06.*

ROBOTICS, 2024. Product manual: IRC5. *ABB.*

RobotWare, 2007. Manual del operador: Introducción a RAPID. *ABB.*

SIEMENS, 2009. TIA Portal STEP 7 Basic V10.5: Getting Started. *SIMATIC.*

SIEMENS, 2017. Programming Guideline for S7-1200/S7-1500. *Background and System Description.*

SIEMENS, 2023. CPU-CPU: Communication with SIMATIC Controllers. *SIMATIC S7.*

## ● Bibliografía

---

I/O setup on the ABB IRB 120 using the IRC5 compact controller.

[https://youtu.be/JG3\\_HsDkwcw](https://youtu.be/JG3_HsDkwcw)

S7 1200 TCP/IP Communication with windows terminal

<https://youtu.be/iO6BftFtias>

ROBOTICS, 2017. Technical reference manual: RAPID Instructions, Functions and Data. *RobotWare 6.06*.

ROBOTICS, 2024. Product manual: IRC5. *ABB*.

ABB, 2010. Technical reference manual RAPID Instructions, Functions and Data types. *ABB AB*.

HMI, S., 2016. WinCC V7.4 WinCC: Scripting (VBS, ANSI-C, VBA). *System Manual*.

SIEMENS, 2009. TIA Portal STEP 7 Basic V10.5: Getting Started. *SIMATIC*.

SIEMENS, 2017. Programming Guideline for S7-1200/S7-1500. *Background and System Description*.

RobotWare, 2007. Manual del operador: Introducción a RAPID. *ABB*.

SIEMENS, 2023. CPU-CPU: Communication with SIMATIC Controllers. *SIMATIC S7*.

**Datasheets de todos los componentes del sistema** (descargables desde el enlace de git del proyecto en la sección Anexos).

## ● Anexos

Pese a que en los anexos se han añadido enlaces para descargar el código fuente de todos los componentes del robot, en el documento de este Trabajo Final de Máster se pretende **presentar en un solo medio la totalidad del proyecto**. Por lo tanto, gran parte del código, variables usadas y configuraciones serán añadidas a estas hojas de Anexos. Esto lo hago en consecuencia a la gran cantidad de enlaces de internet que terminan cayendo en el transcurso del tiempo. De no añadir aquí el código, el **trabajo quedaría inservible para aquellos posibles lectores del futuro**.

**Vídeo de la defensa (con DNI)**. Este enlace será borrado al ser corregido el TFM.

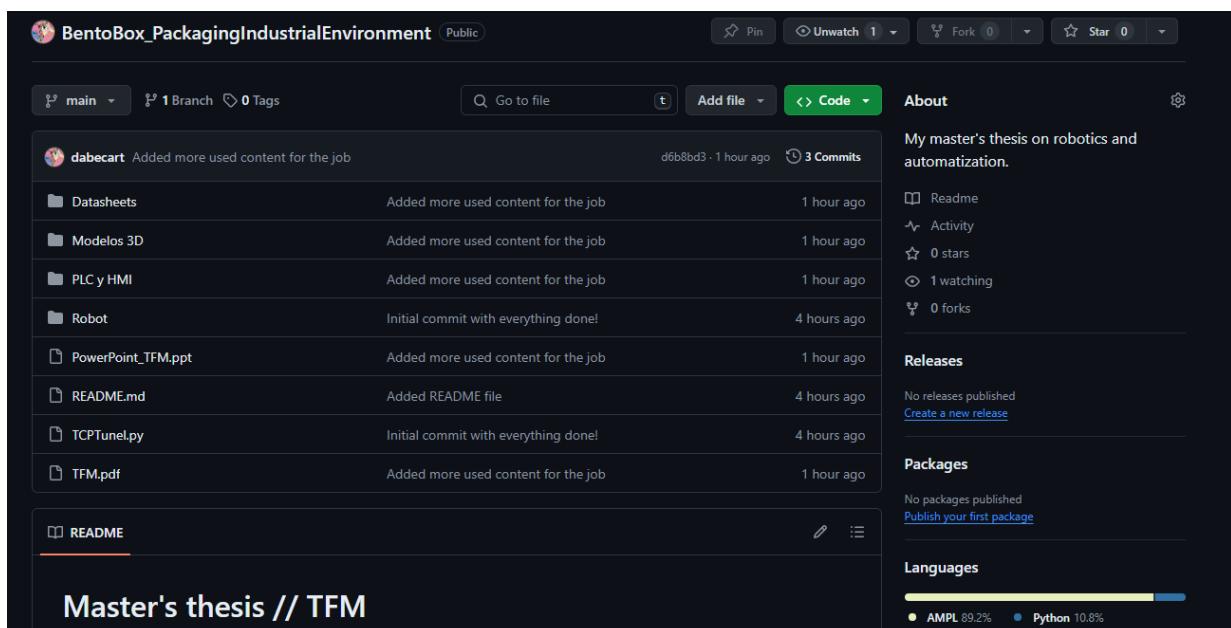
[https://youtu.be/\\_40QNaJSeB0](https://youtu.be/_40QNaJSeB0)

**Vídeo de la defensa (sin DNI)**

<https://youtu.be/ntla3Tn9hl>

**Código fuente del proyecto y todos los recursos del programa**

[https://github.com/dabecart/BentoBox\\_PackagingIndustrialEnvironment](https://github.com/dabecart/BentoBox_PackagingIndustrialEnvironment)



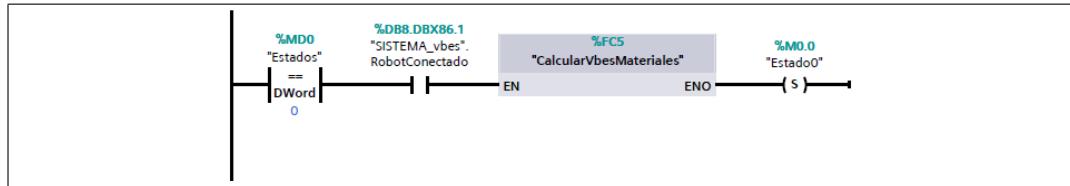
**Figura 66. Captura de la página de GitHub del proyecto.**

## 1. Anexo 1: Programación del PLC

El PLC comienza la ejecución del programa desde el bloque Main.

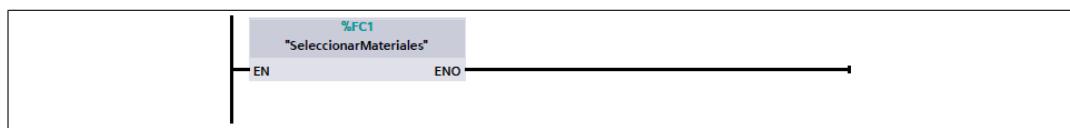
### Network 1: Inicialización del sistema

Si el sistema acaba de arrancar (todos los estados están a 0), vuelve al estado inicial 0.



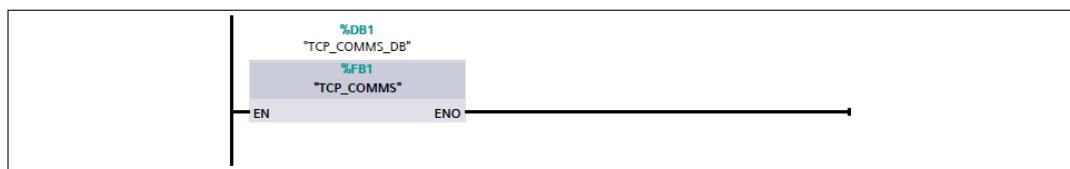
### Network 2: Selector de materiales

Este bloque espera recibir un material desde las cintas de materiales y lo envía al final de la cinta general para que lo recoja el robot.



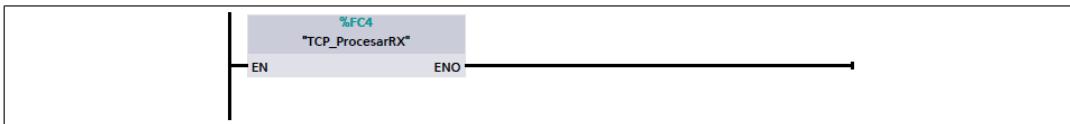
### Network 3: Conexiones RX/TX por TCP

Encargado de leer los mensajes de entrada y generar los mensajes de salida por TCP.



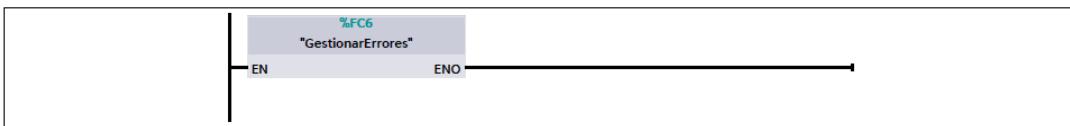
### Network 4: Procesar mensajes recibidos por TCP

Una vez recibido los mensajes, estos son procesados.



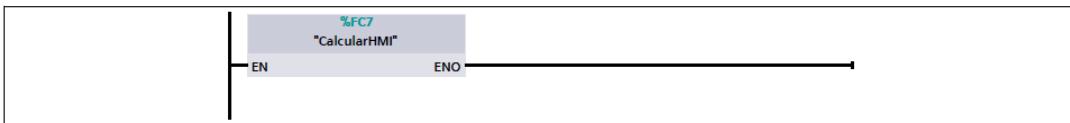
### Network 5: Error

Rutina ante errores



### Network 6: Actualizar valores para HMI

Genera las variables necesarias para el HMI.



En NW1 (Network 1) se comprueba si el sistema está recién arrancado (“Estados” es 0) y si el robot está conectado, calcula ciertos valores necesarios para el sistema (CalcularVbesMateriales) y pasa al sistema al estado 0 (“Estados” = 1).

## 1.1. Selector de Materiales

El bloque de “SeleccionarMateriales” utiliza las siguientes variables temporales:

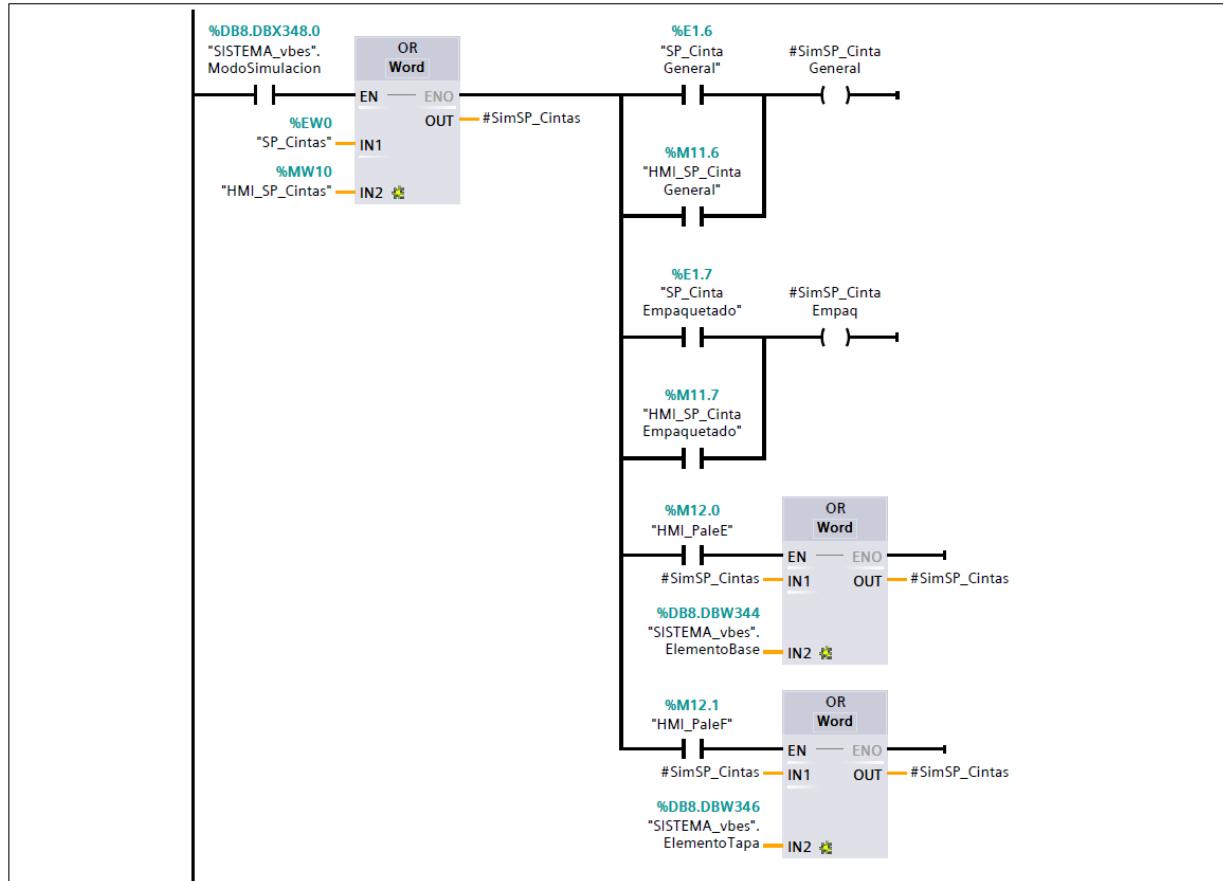
Nombre	Tipo	Descripción
<b>AuxVar</b>	Word	Word auxiliar
<b>TX_Realizado</b>	Bool	True si el TX se ha realizado correctamente
<b>SimSP_Cintas</b>	Word	OR de las entradas reales y simuladas
<b>SimSP_CintaGeneral</b>	Bool	OR de los sensores reales y simulados de la cinta general
<b>SimSP_CintaEmpaq</b>	Bool	OR de los sensores reales y simulados de cinta empaquetadora
<b>TiempoAux</b>	Time	Time auxiliar

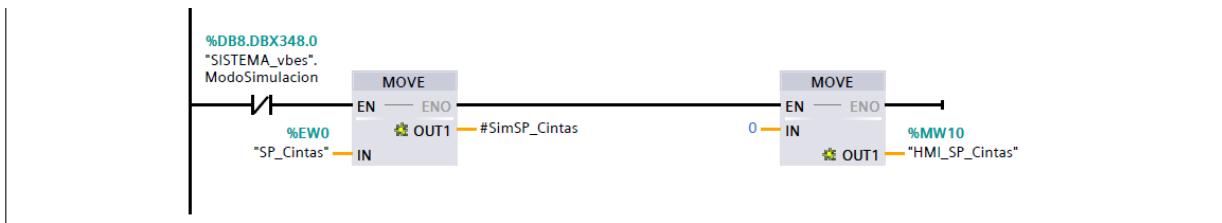
**Tabla 25. Variables del bloque "SeleccionarMateriales".**

A continuación, se muestra el código del bloque “SeleccionarMateriales”.

### Network 1: Genera las entradas simuladas de estar en el modo de simulación

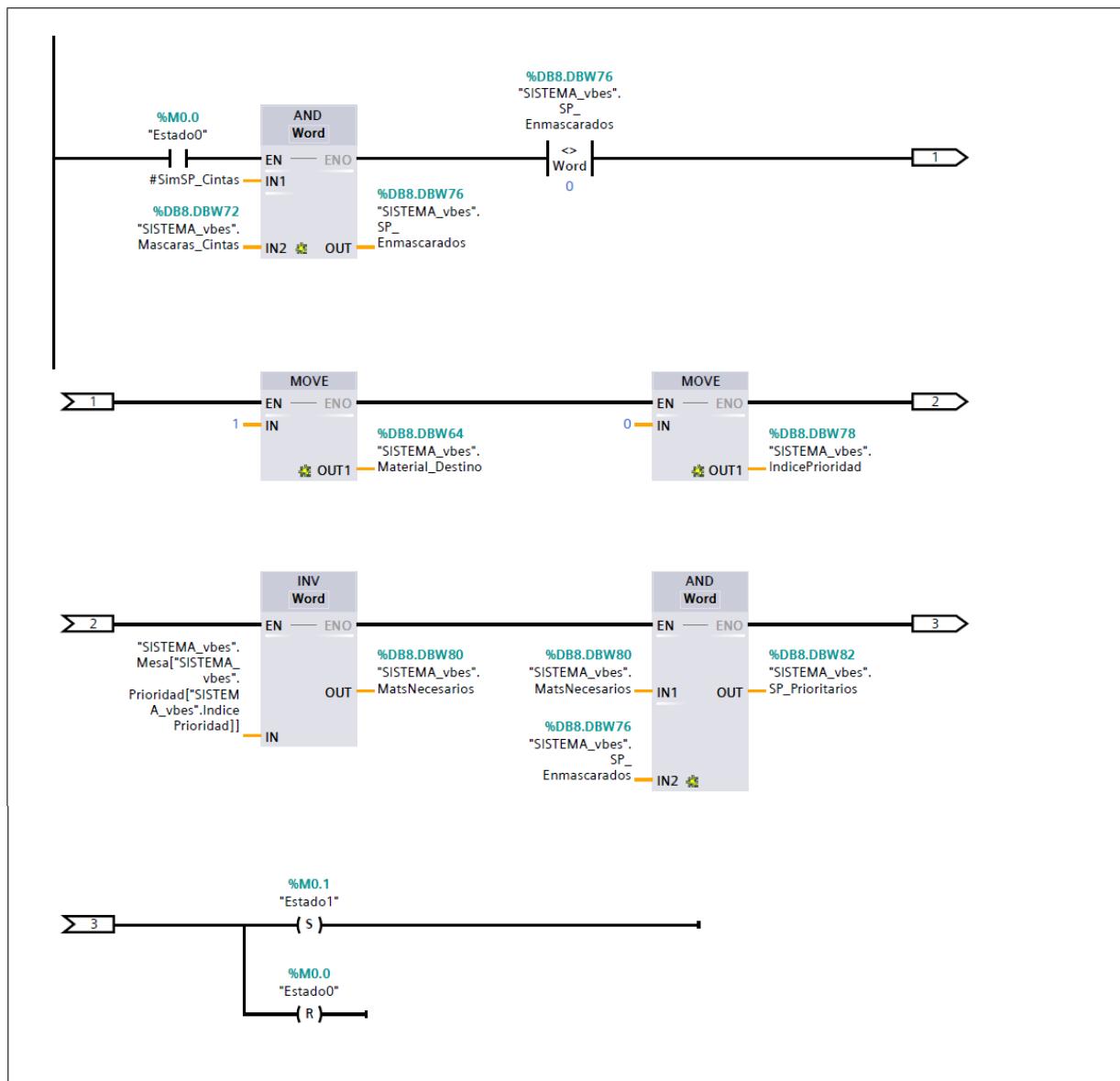
Se unen las entradas HMI con las entradas digitales del sistema en el modo de simulación.





## Network 2: Detección de llegada de material

Si llega algún material, los bits de los sensores de posición SP\_Cintas se activarán. El material deberá estar habilitado (según Mascaras\_Cintas). Esto indica que el material es necesario por alguna de las mesas. Si al analizar todas las mesas, ese material no es necesario, quedará enmascarado hasta que finalice alguna caja. El valores asignad a Material\_Destino es temporal.



### Network 3: Selección de un solo material según mesa prioritaria

Si por lo que fuera llegase más de un material, habría que elegir uno solo que comprobar. Habrá que seleccionar además aquél material que necesiten las mesas en orden de prioridad. Aquí se itera dentro del array Prioridad[] mediante el índice IndicePrioridad. El máximo de mesas queda definido por NumeroMesas.

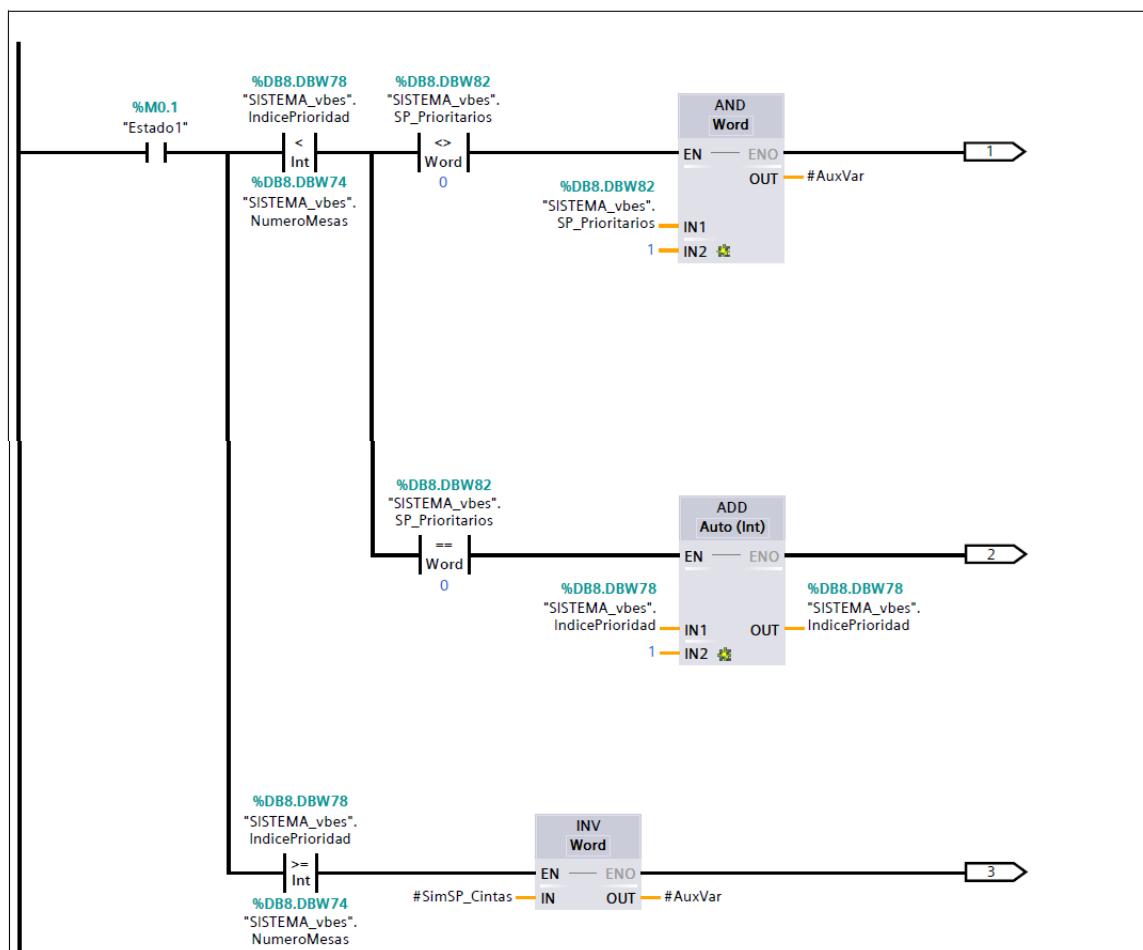
Si el índice supera el número de mesas indica que los materiales que habían por procesar y que han disparado los SP NO son necesarios hasta que se liberen alguna de las cajas, por lo tanto, se deshabilitan todos los materiales que hayan disparado el SP.

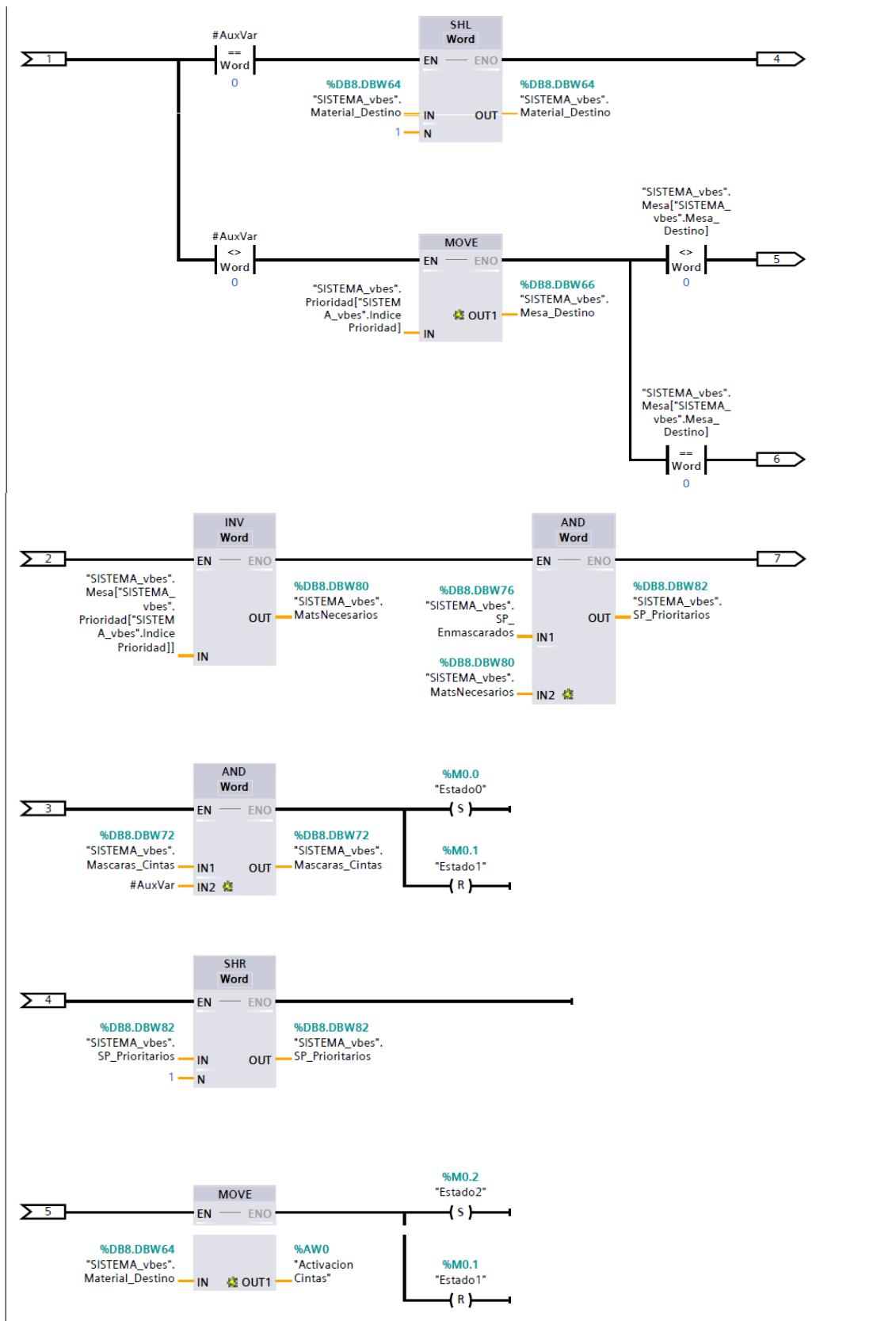
La variable SP\_Prioritarios almacena aquellos componentes que la Mesa[IndicePrioridad] necesita Y que están dentro de las cintas. Si es distinto de cero, indica que hay un material dentro de las cintas que es necesario para la mesa. Únicamente falta encontrar qué material es; es decir, en qué posición de SP\_Prioritarios se encuentra.

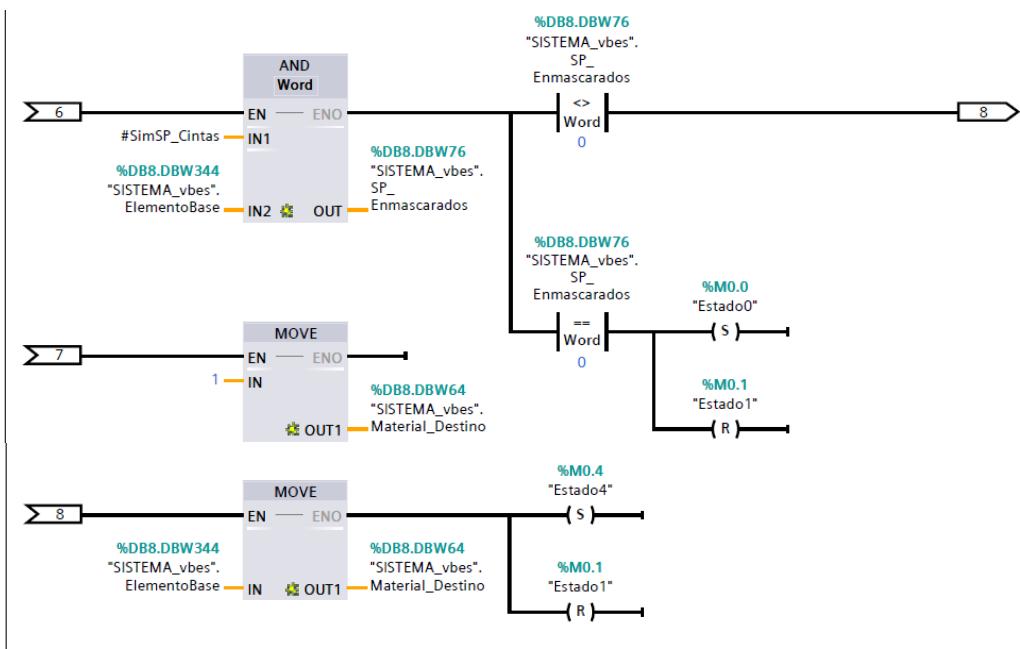
La función AND (SP\_Prioritarios AND 1) comprueba que el LSB es 1 o 0. Si es uno, ya se ha encontrado el material y por lo tanto, se actualiza el valor de Mesa\_Destino, deja activada la cinta correspondiente a ese material y cambia de estado. Antes de hacer esto, se debe comprobar si ya hay una bandeja en la mesa que se dispone a colocar.

Si el valor es cero, hay que pasar al siguiente Material\_Destino (son potencias de 2, luego bitshift a la izquierda) y pasar a comprobar el siguiente bit de SP\_Prioritarios haciendo un bitshift a la derecha.

Cuando SP\_Prioritarios es 0, indica que NO queda ningún material que sea necesario para esa Mesa[IndicePrioridad], por lo tanto, se incrementa el contador IndicePrioridad y se vuelve a calcular SP\_Prioritarios pero para la nueva mesa.



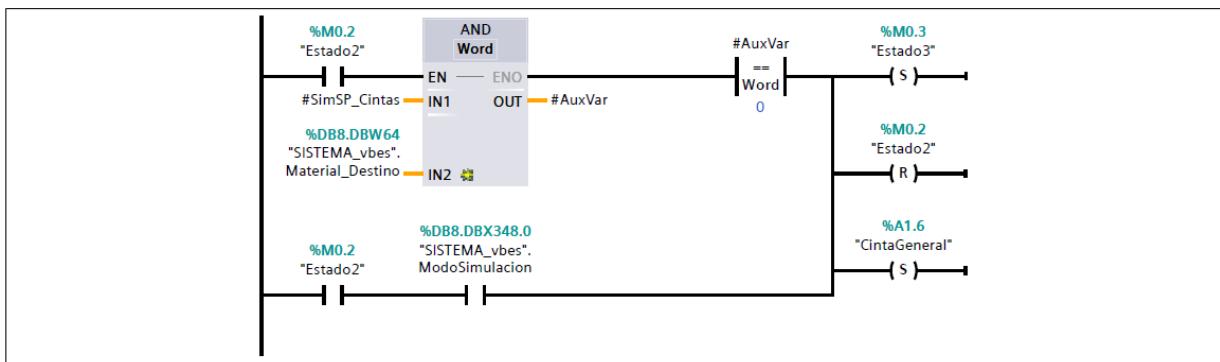




#### Network 4: Activar la cinta del material

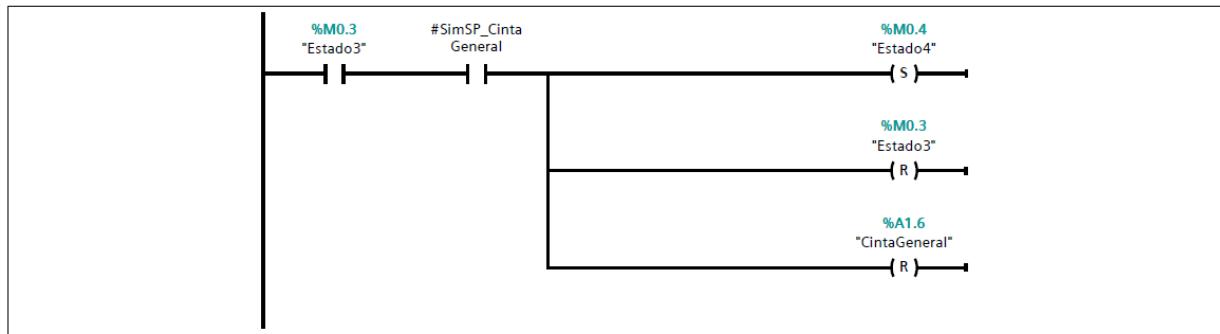
Una vez se tiene seleccionado el material, hay que activar su correspondiente cinta. Para ello, se comprueba la lectura de los SP\_Cintas y se enmascara con el Material\_Destino. Cuando este valga cero, significará que el material ha pasado el sensor.

Si está en modo simulación no simular el paso de los materiales por los sensores.



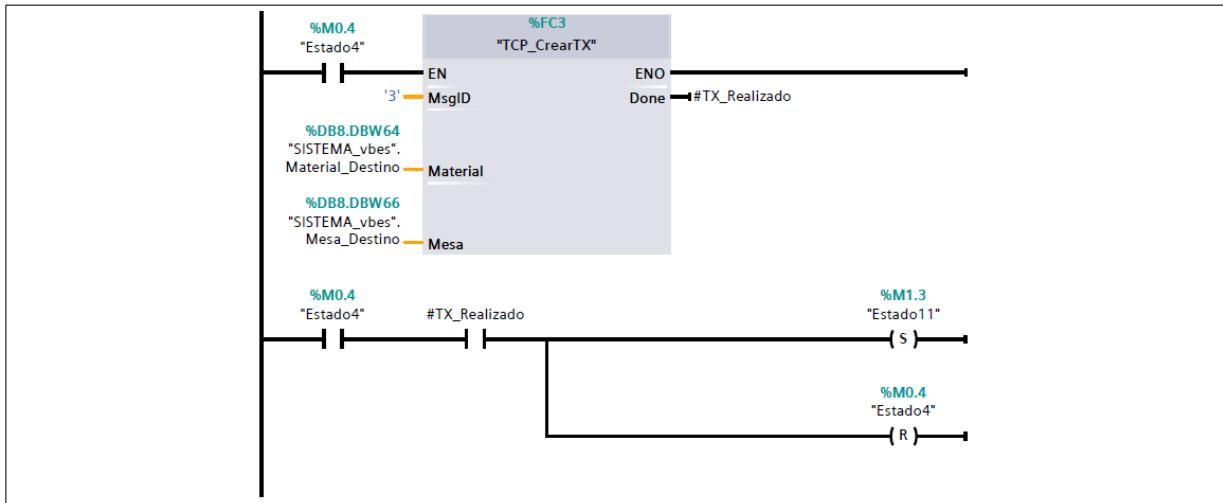
#### Network 5: Activar la cinta general

La cinta general alcanza el material al brazo robótico.



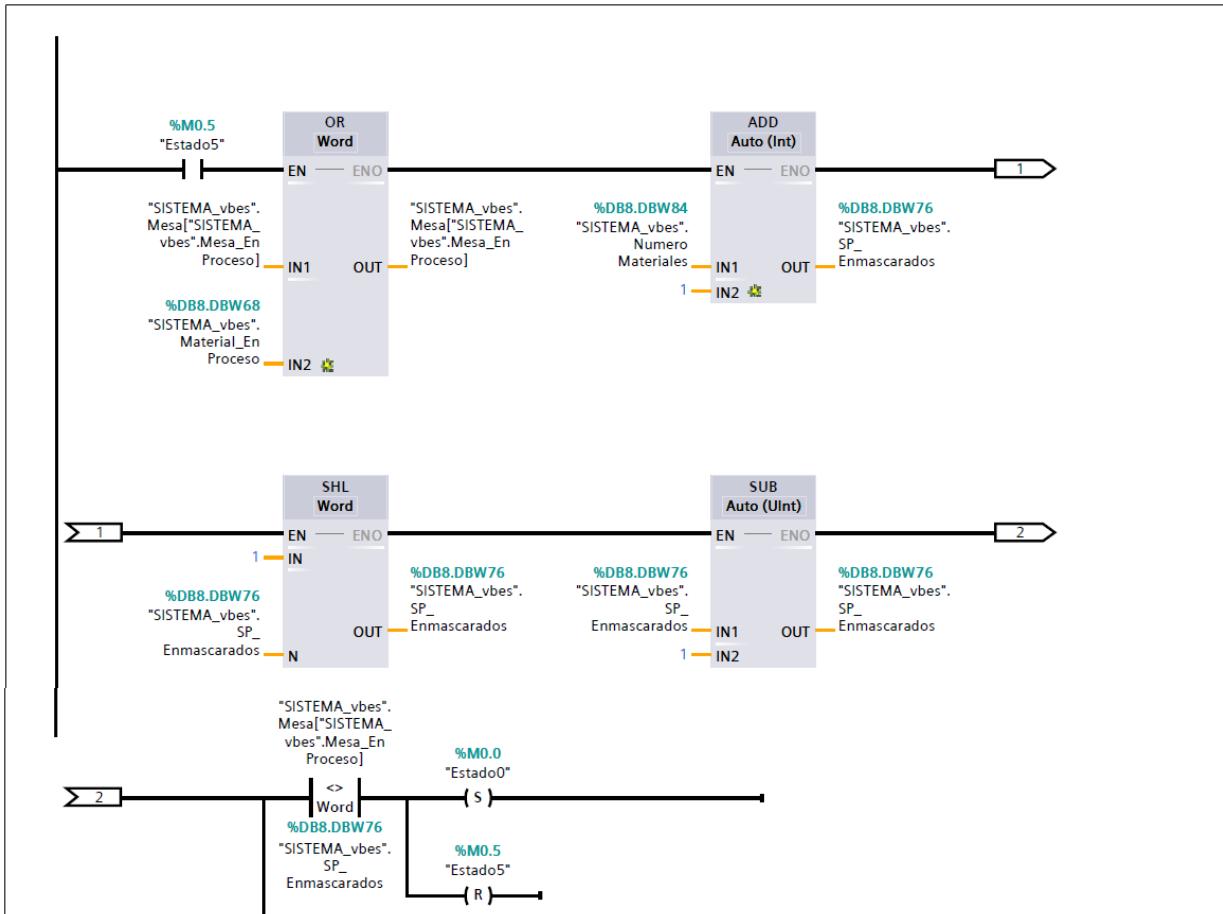
### Network 6: Enviar mensaje por TCP

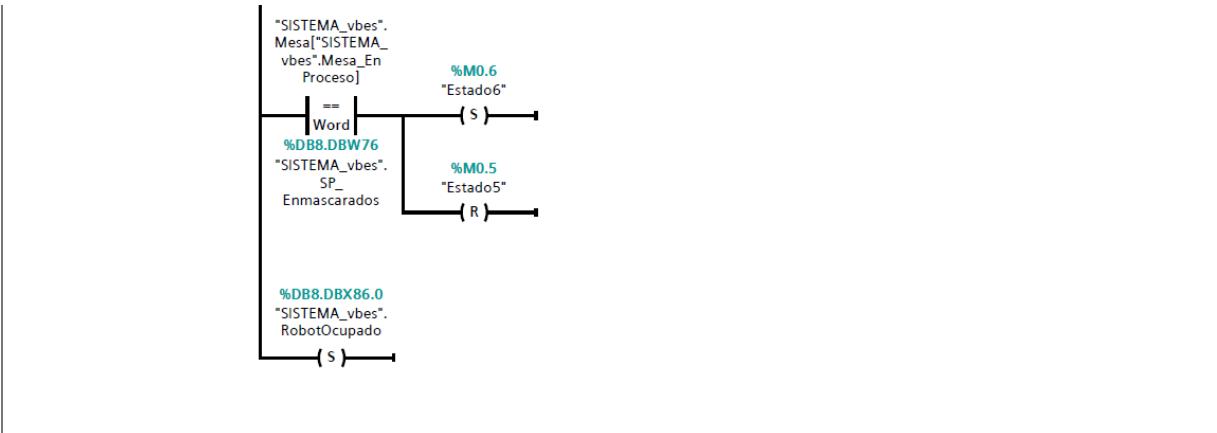
Manda el mensaje con ID 3: Colocar material. Una vez se ha enviado correctamente, continúa al siguiente estado.



### Network 7: Espera de recepción del mensaje de material recogido

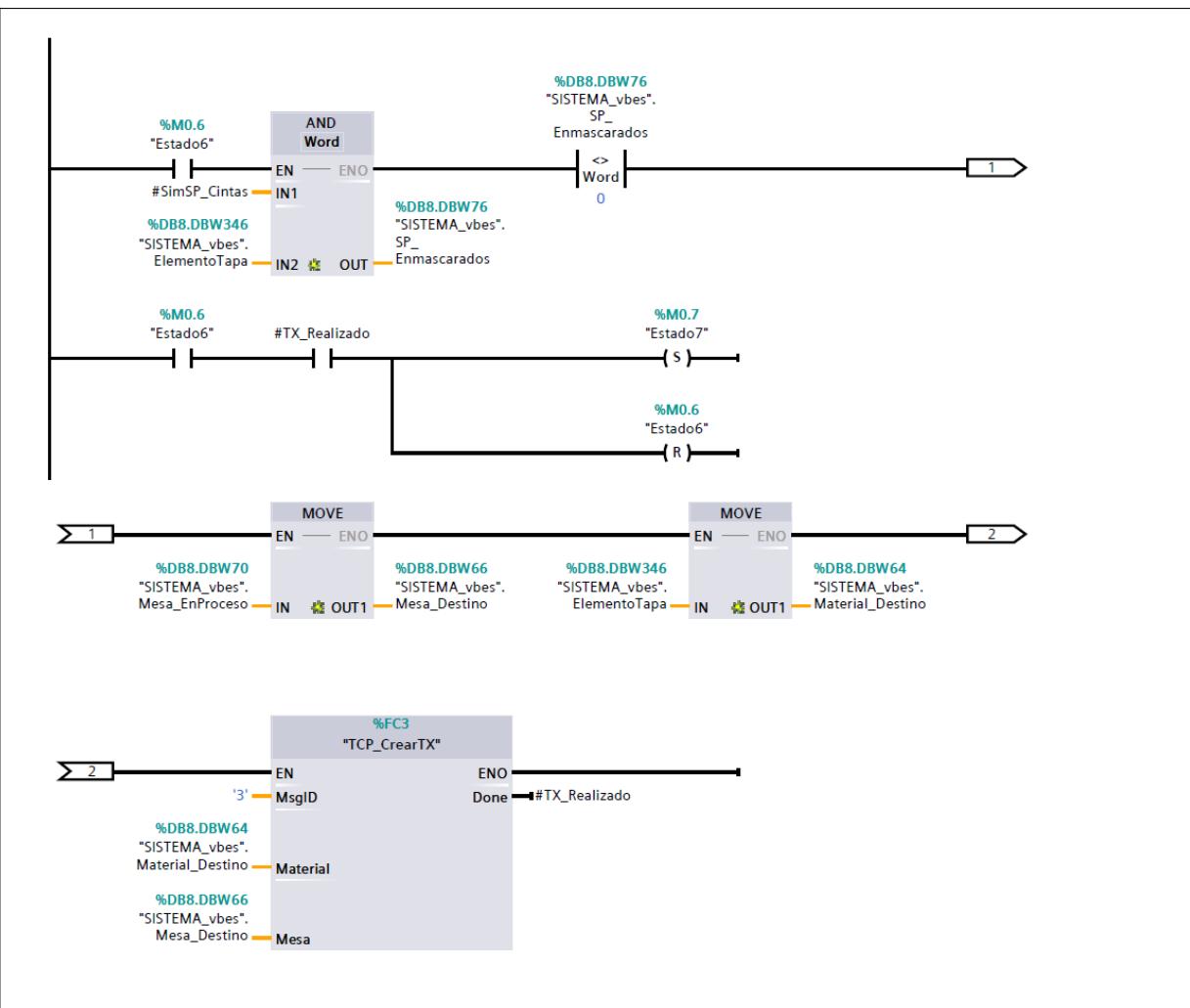
Una vez es recibido el mensaje del robot de que ha recogido el material de la cinta, se guarda de manera temporal el material y la mesa que se han mandado. Dejará de ser temporal cuando el robot confirme que el componente está bien colocado. Para añadir un componente a una Mesa, simplemente se hace un OR con el contenido de la mesa y el material. Una vez colocado el material en la mesa, se comprueba si ya están todos los materiales en esa caja para cerrarla y mandarla a la empaquetadora.





### Network 8: Colocar tapadera

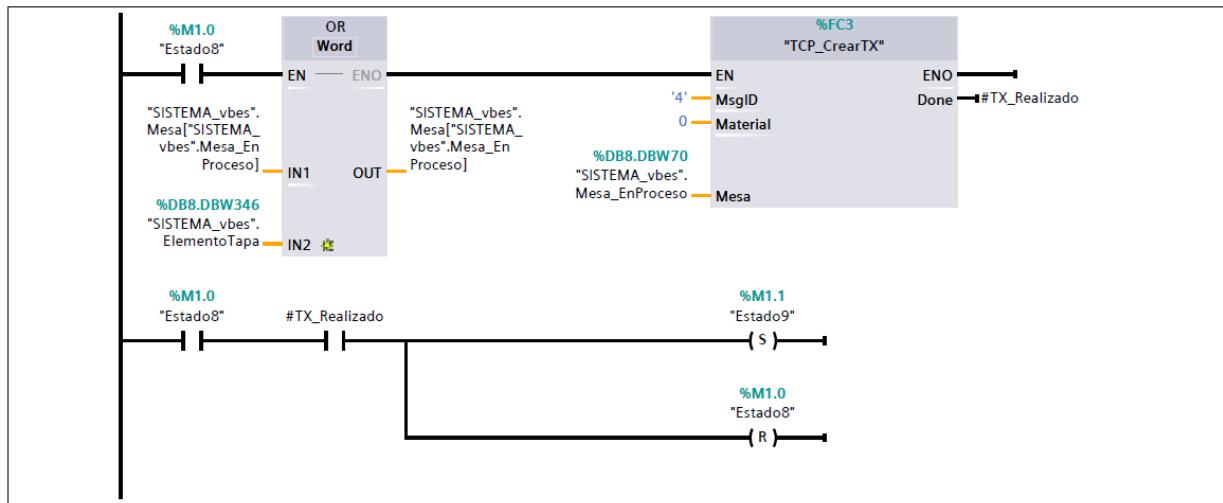
Cuando todos los componentes están colocados, se manda a poner una tapadera.



### Network 9: Mandar comando de colocar en cinta de empaquetado

El estado 7 es un estado de espera en el que no se hace nada. Únicamente se espera que al recibir el mensaje de material colocado. En la función TCP\_ProcesarRX, se detecta que se ha terminado de colocar la tapa cuando el robot lo indica y si el PLC se encuentra en el estado 7, activa el estado 8 (desactivando el 7).

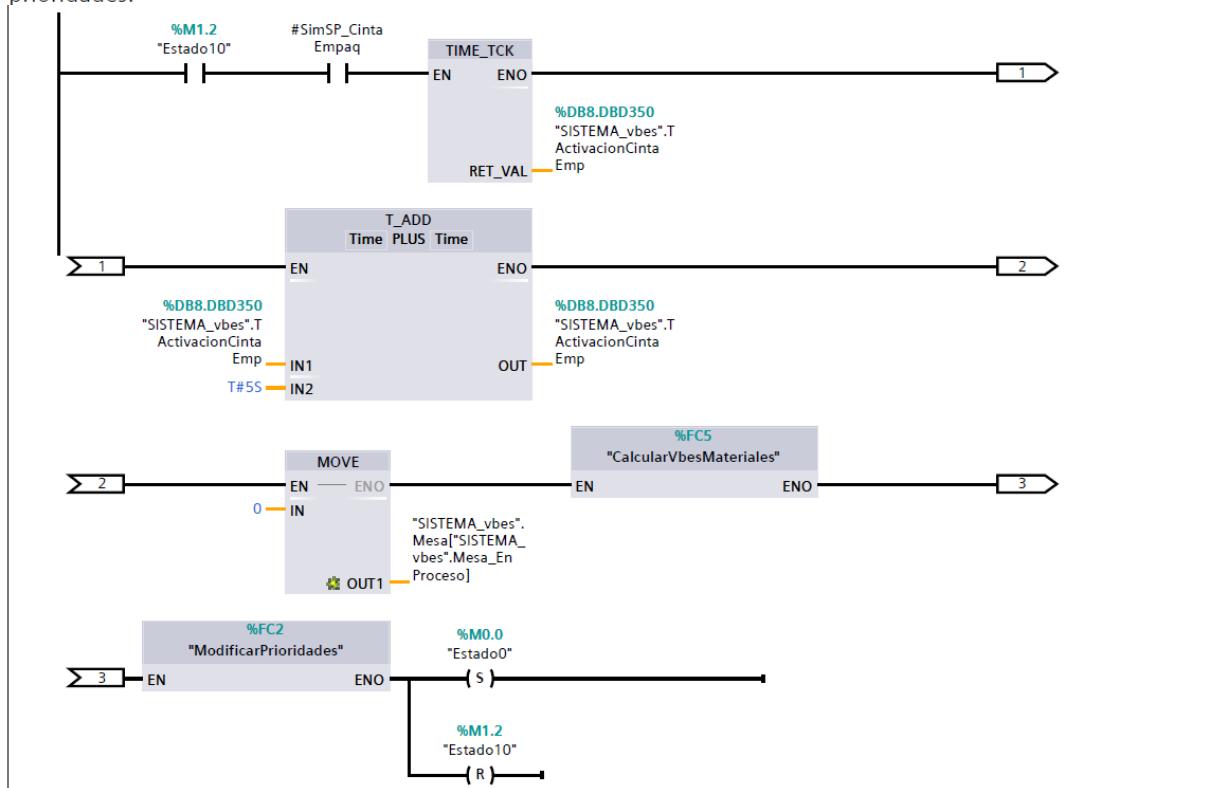
En el estado 8, se manda a colocar la caja en la cinta de empaquetado. Cuando se recibe el ACK no se vuelve al estado inicial, sino que se espera a que llegue el mensaje de material colocado que mandará el robot para enviar el comando para colocar la caja en la cinta.



### Network 10: Confirmación de presencia de la caja en la cinta de empaquetado

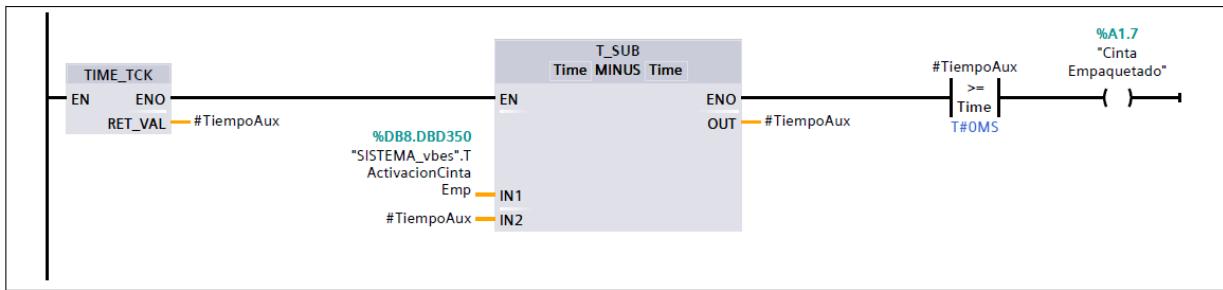
Una vez el robot confirma que ha depositado la caja, se comprueba el sensor de presencia en la cinta de empaquetado.

Se modifica el tiempo hasta el cual debe estar activa la cinta de empaquetado (durante 5 segundos). Si está presente el material, recalculamos las máscaras de los materiales (antes se podrían haber enmascarado aquellos materiales que no eran necesarios, pero ahora al haber una caja libre, todos los materiales vuelven a ser necesarios) y se modifica la lista de prioridades.



### Network 11: Activar la cinta de empaquetado

Se utiliza el tiempo del sistema en vez de temporizadores para no bloquear el funcionamiento del sistema.



## 1.2. Conexiones TCP

El bloque TCP\_COMMS es el encargado de establecer las comunicaciones TCP con el robot, desconectarse si es requerido, enviar y recibir mensajes por el puerto TCP. El PLC trabaja como cliente. Los datos necesarios para este bloque se encuentran en TCP\_DATA.

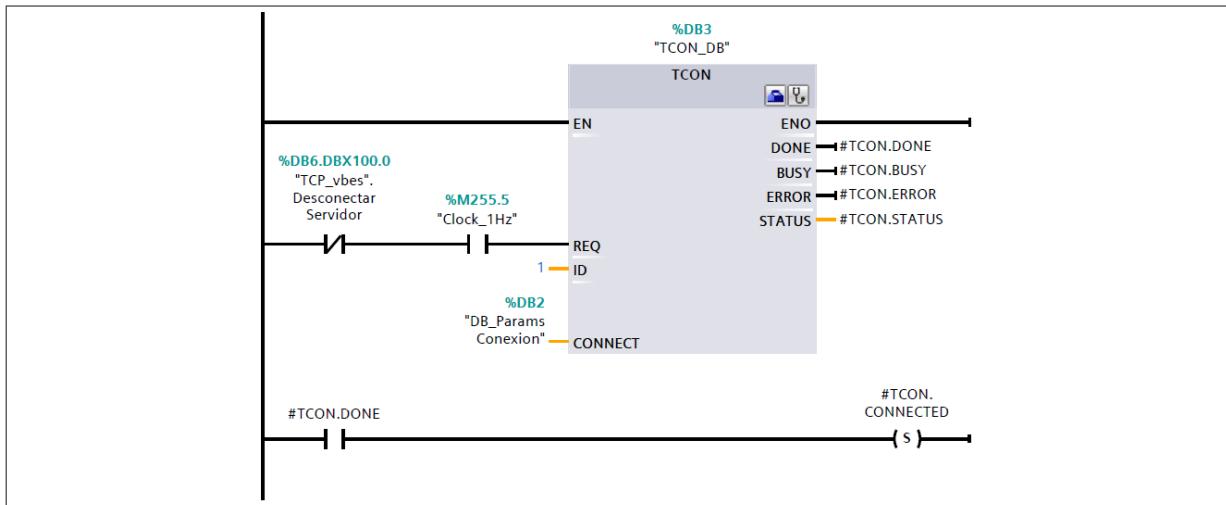
Nombre	Campo	Tipo	Valor inicial
<b>TCON</b>		Struct	
	REQ	Bool	false
	DONE	Bool	False
	BUSY	Bool	False
	ERROR	Bool	False
	STATUS	UInt	0
	CONNECTED	Bool	False
<b>TDISCON</b>		Struct	
	REQ	Bool	False
	DONE	Bool	False
	BUSY	Bool	False
	ERROR	Bool	False
	STATUS	UInt	0
<b>TSEND</b>		Struct	
	REQ	Bool	False
	DONE	Bool	False
	BUSY	Bool	False
	ERROR	Bool	False
	STATUS	UInt	0
<b>TRCV</b>		Struct	
	EN_R	Bool	False
	NDR	Bool	False
	BUSY	Bool	False
	ERROR	Bool	False
	STATUS	UInt	0
	RCVD_LEN	UInt	0
<b>IntentosTX</b>		Int	0 (máximo 10)

Tabla 26. Variables internas de TCP\_COMMS.

El código del bloque TCP\_COMMS se presenta a continuación.

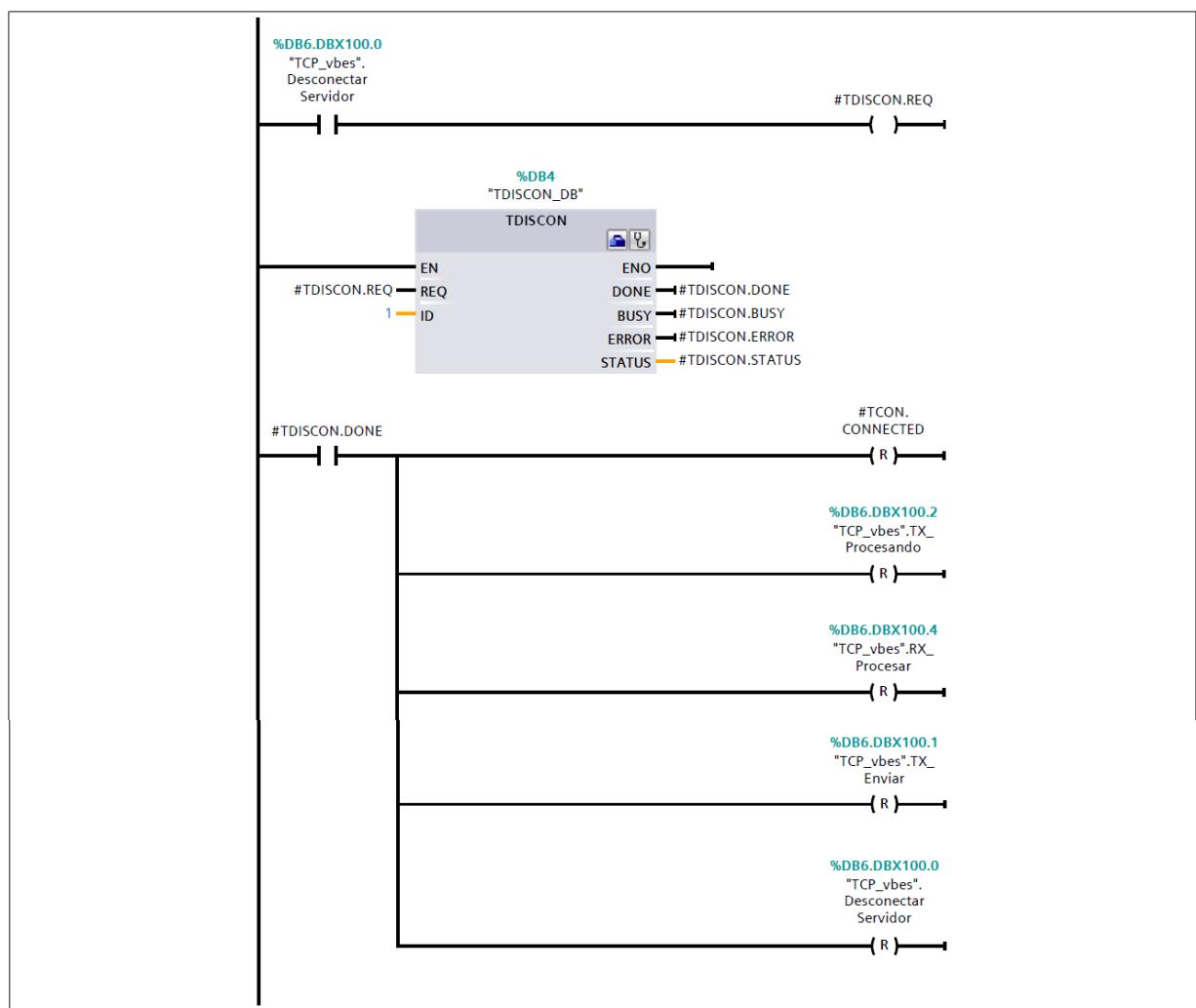
#### Network 1: Conexión con el servidor TCP

Si se habilita, permite la conexión del PLC con el server.



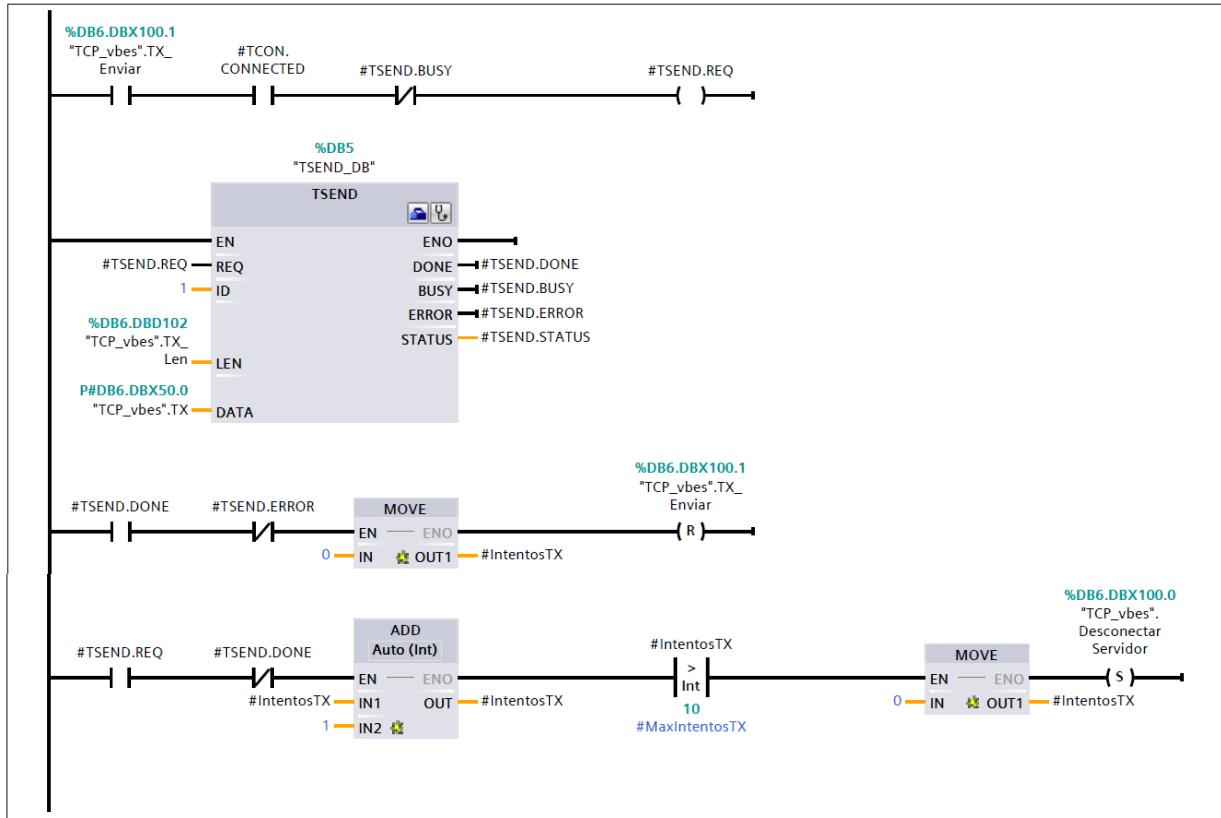
#### Network 2: Desconectar del servidor TCP

Al desconectarse el PLC del server, actualiza el estado del dispositivo.



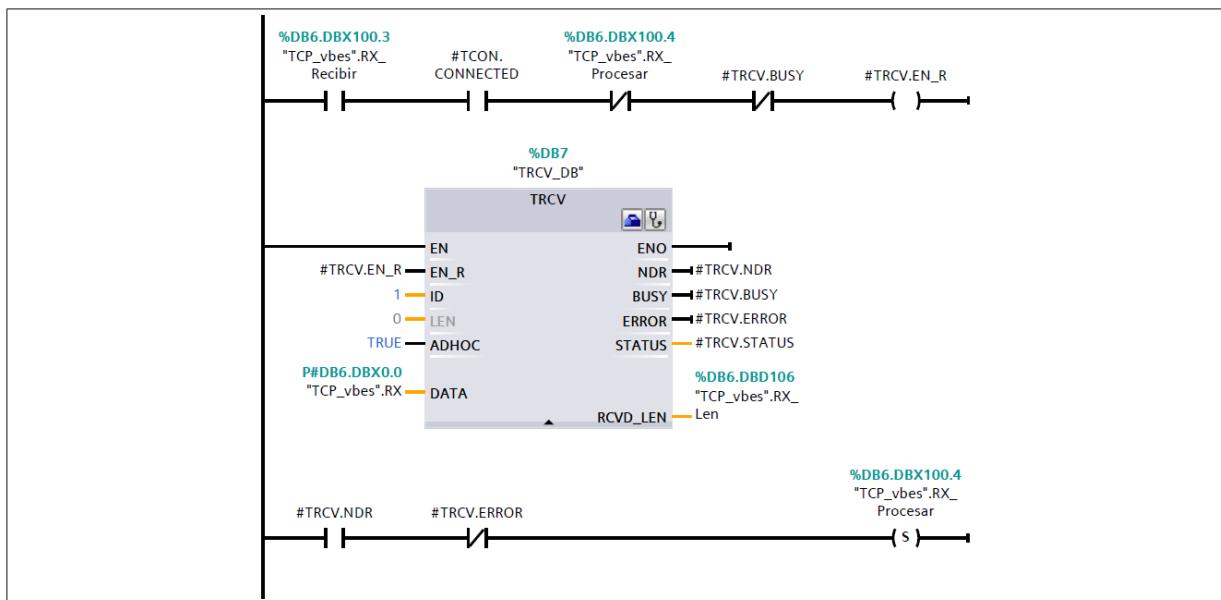
### Network 3: Transmitir datos hacia el servidor (TX)

Si hay datos que enviar, el sistema está conectado y no se está procesando una escritura anterior, se manda el buffer TX.



### Network 4: Recibir datos desde el servidor (RX)

Los mensajes de entrada se procesarán de uno en uno. Si llega un mensaje y no se ha procesado el anterior, este se descartará. El mensaje de salida se encuentra en el buffer RX.



### 1.3. Procesar mensajes entrantes (TCP)

Este bloque es el encargado de procesar los mensajes recibidos desde TCP\_COMMS.

Tiene las siguientes variables internas.

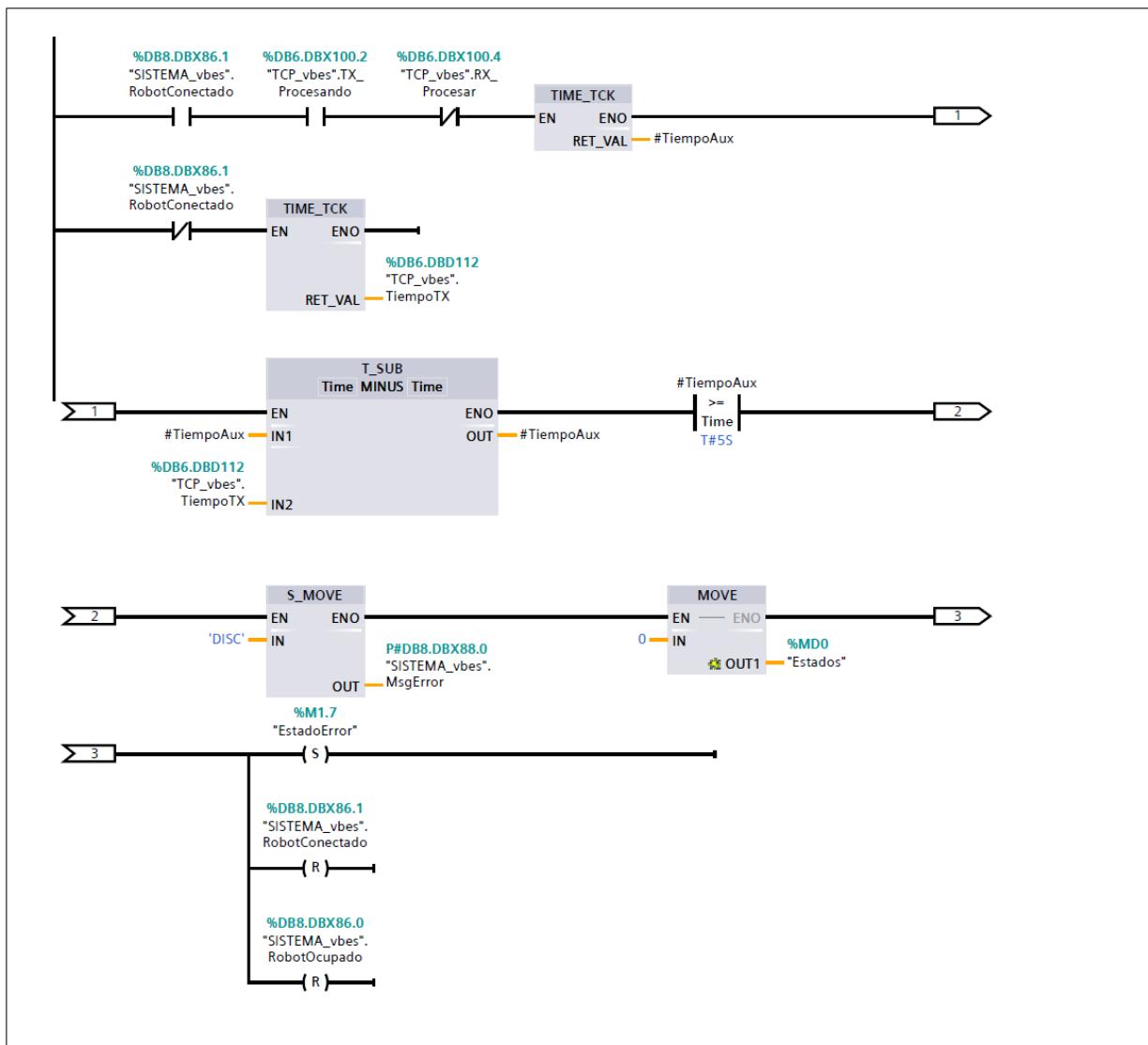
<b>Nombre</b>	<b>Tipo</b>	<b>Descripción</b>
<b>MsgID</b>	Char	ID del mensaje recibido
<b>MsgOK</b>	Bool	True si el mensaje es correcto
<b>TiempoAux</b>	Time	Variable auxiliar de tipo Time
<b>LongStr</b>	UDInt	Variable auxiliar para almacenar la longitud de un string

**Tabla 27. Variables del bloque “TCP\_COMMS”.**

A continuación, se presenta el código del bloque.

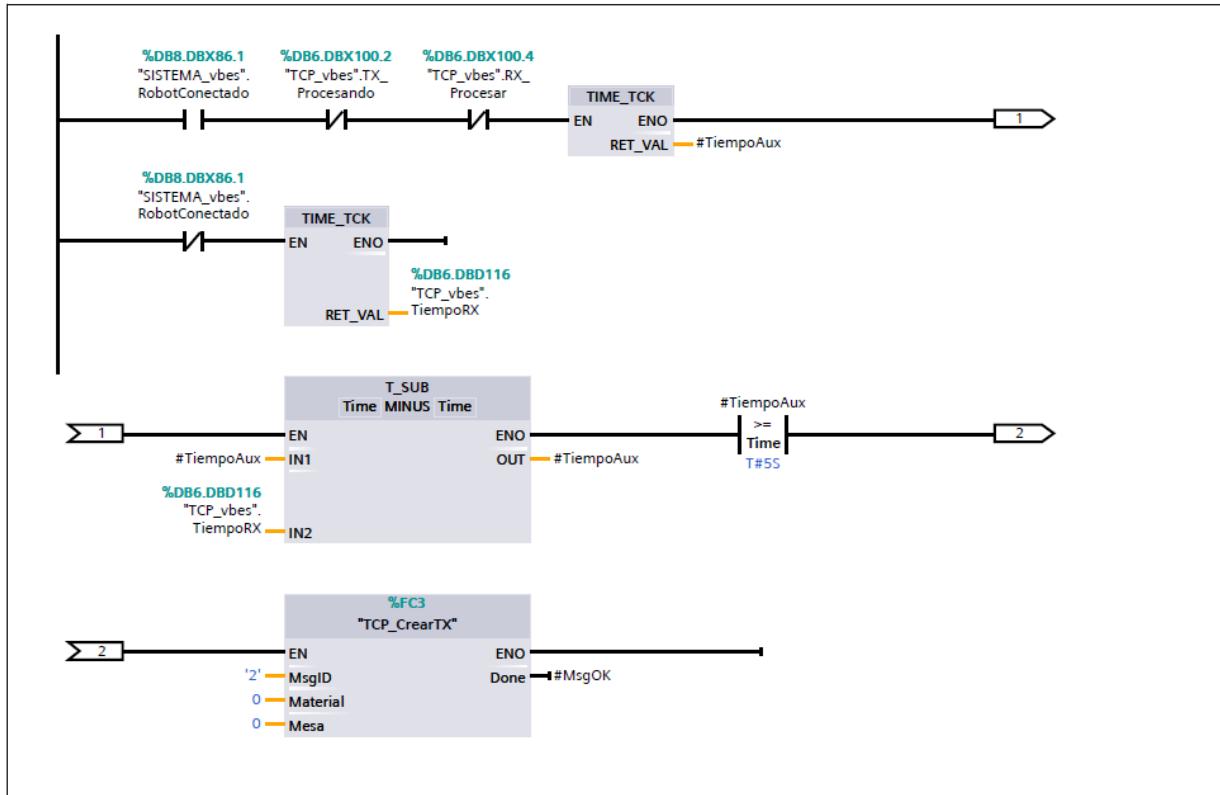
Network 1: Comprobar el tiempo del último TX para ver si el sistema se ha desconectado

Se espera que el sistema devuelva un ACK. Si pasan más de cinco segundos se tomará que el sistema está desconectado.

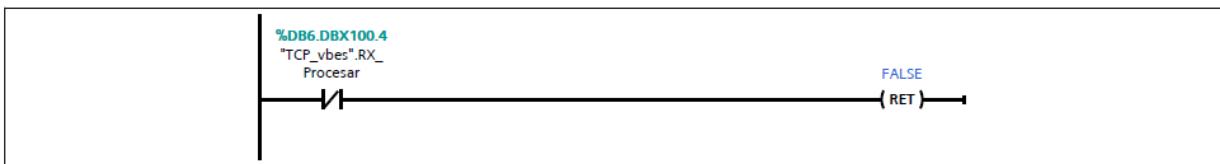


## Network 2: Comprobar el tiempo del último RX

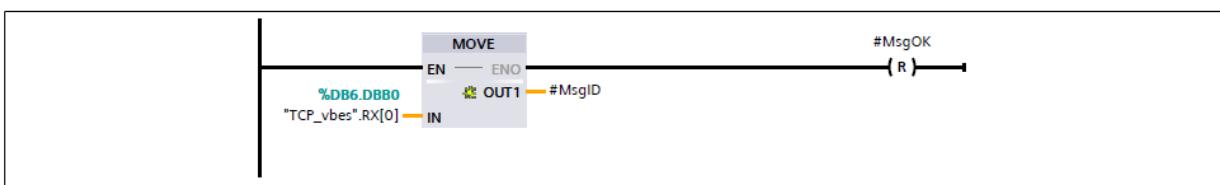
Si ha pasado mucho tiempo desde la última vez que se recibió algo, mandar un mensaje para comprobar el estado del robot.



## Network 3: Si la recepción de mensajes está deshabilitada, salir del programa.

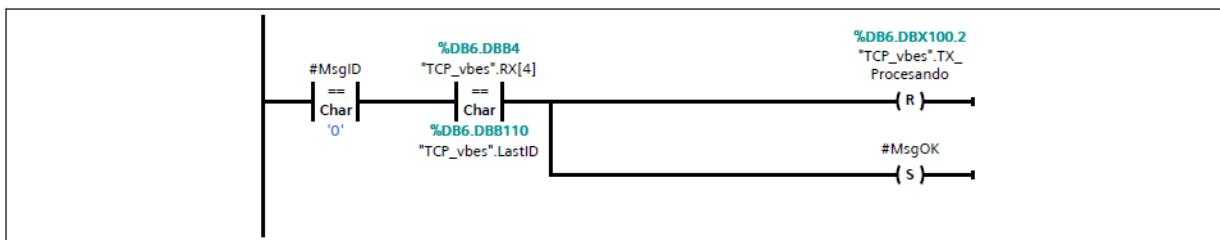


## Network 4: Separar la ID del mensaje recibido e Inicializa las variables

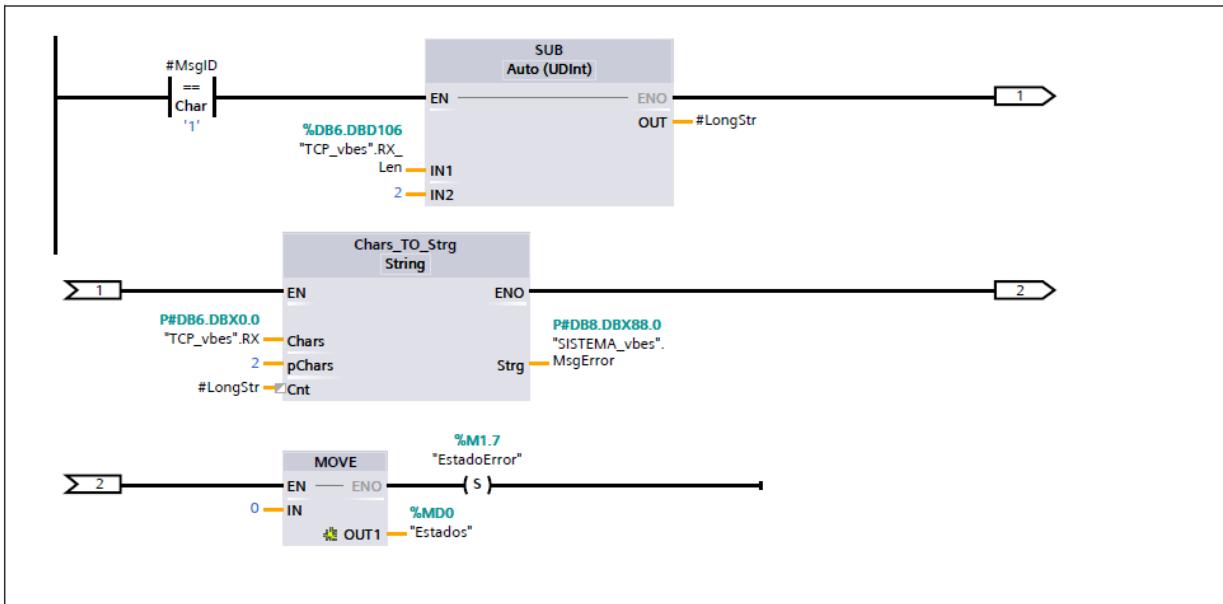


## Network 5: Procesar mensaje 0: ACK

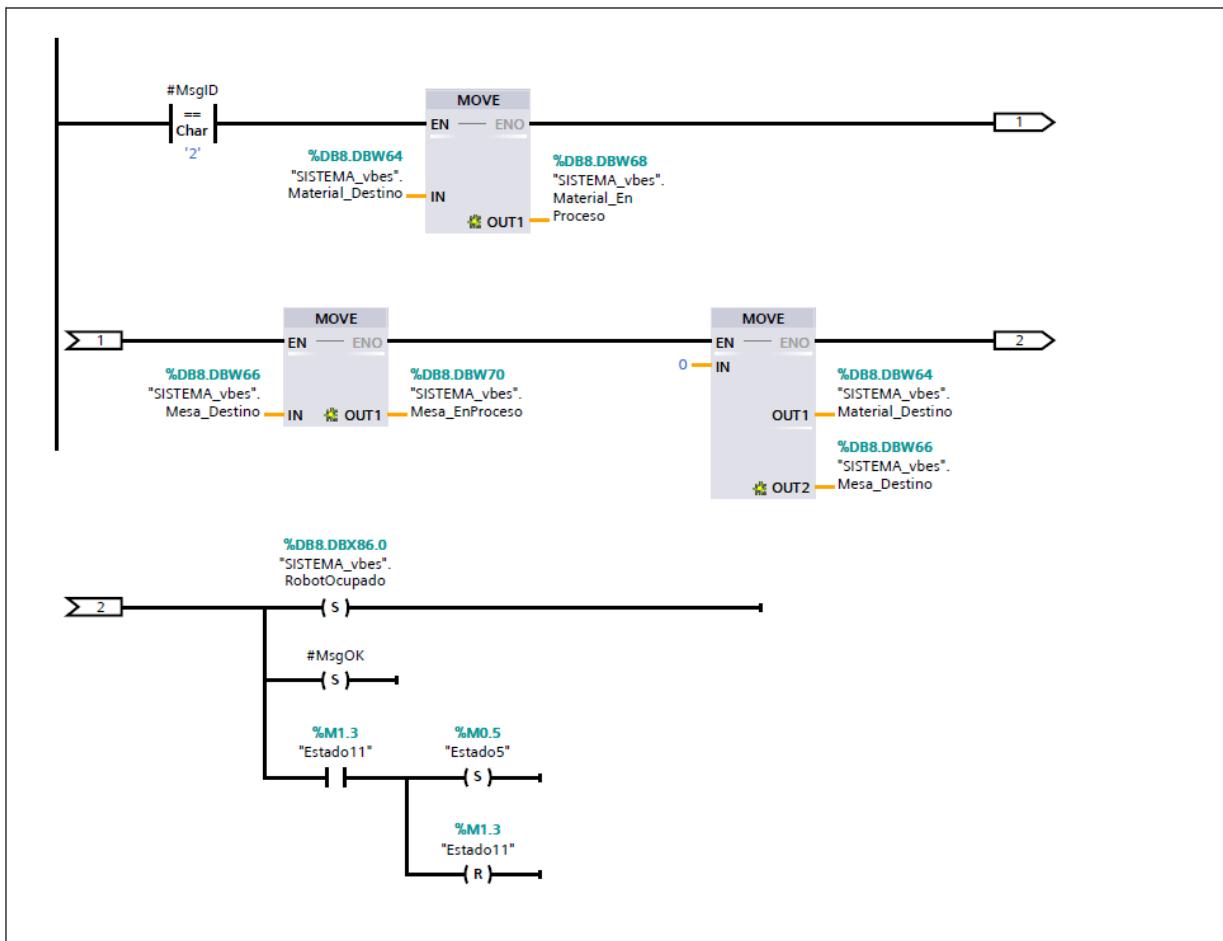
Ante un ACK, únicamente se libera el bit de RX\_Procesando (el otro sistema ya ha terminado de procesar o está en ello).



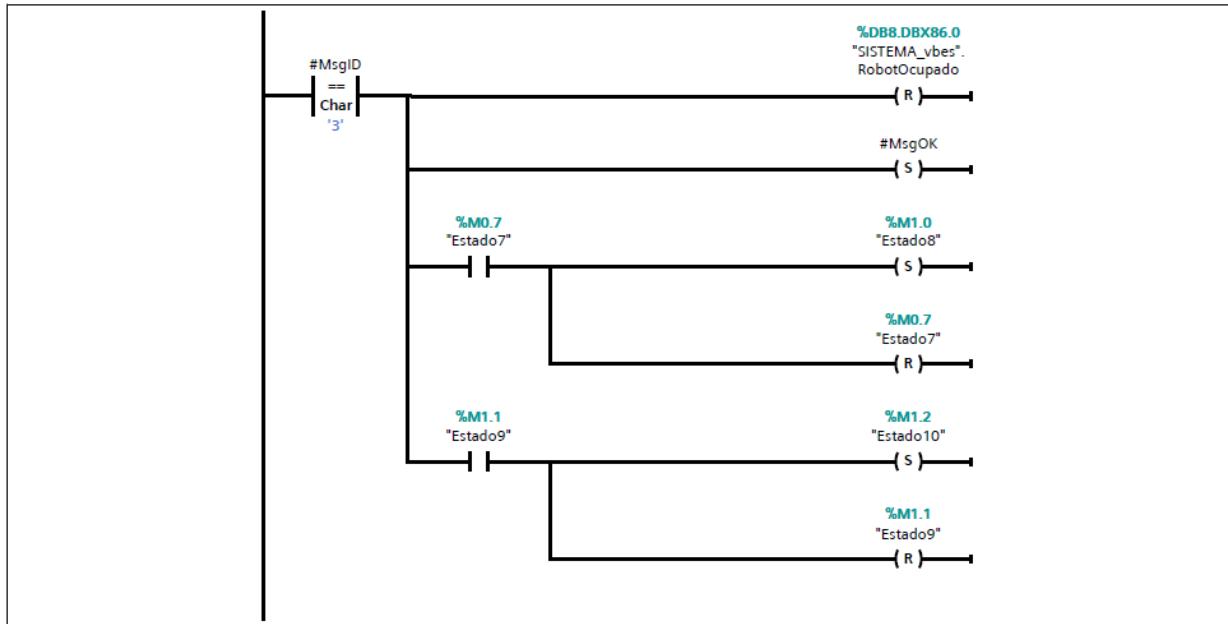
### Network 6: Procesar mensaje 1: ERROR



### Network 7: Procesar mensaje 2: Material recogido

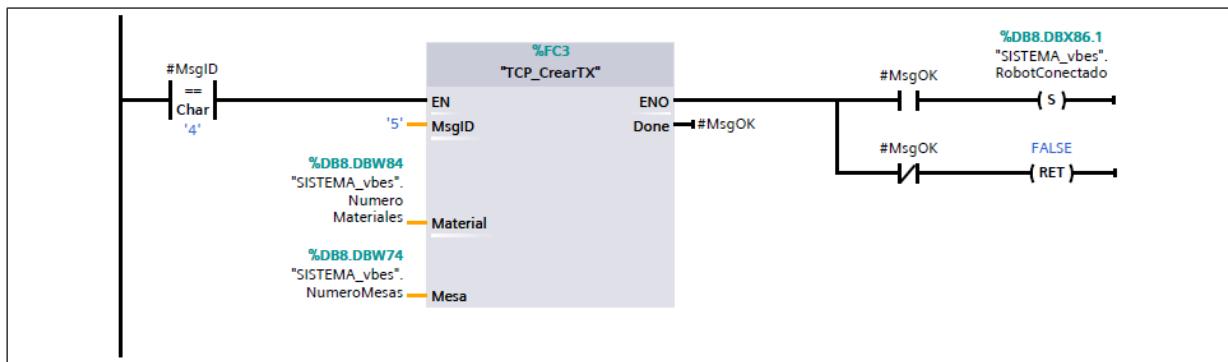


### Network 8: Procesar mensaje 3: Material colocado

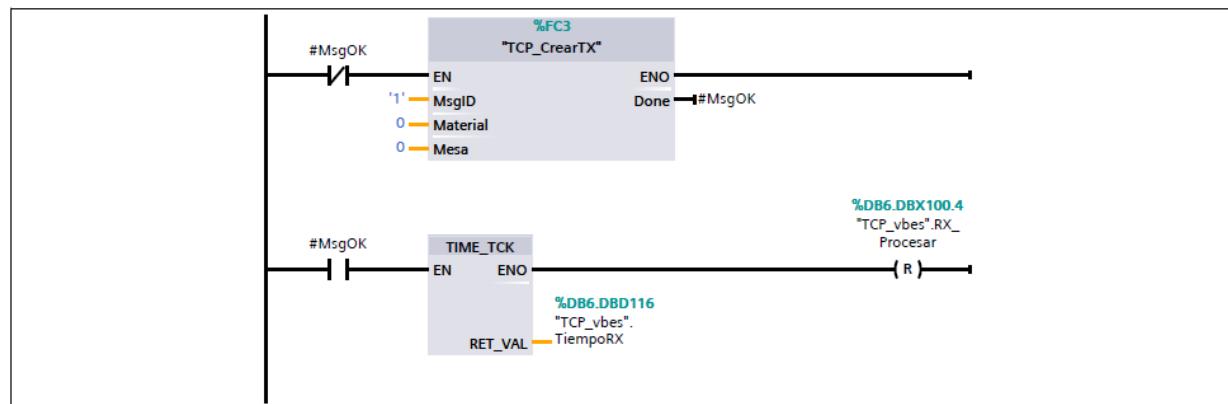


### Network 9: Procesar mensaje 4: Conexion establecida

Cuando se establezca conexión, mandar mensaje con los numeros de mesa y materiales configurados en el programa.



### Network 10: Mandar mensaje de error



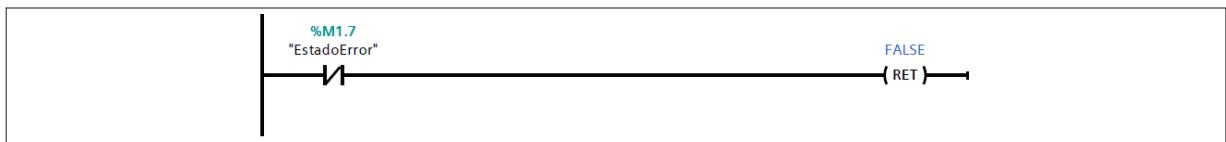
## 1.4. Gestionar errores

El bloque “GestionarErrores” se encarga de procesar los mensajes de error entrantes y mostrar la alarma correspondiente en el sistema HMI. Una vez el error es gestionado por el usuario, el sistema vuelve a ser rearmado (o reiniciado) desde el sistema HMI o desde una entrada física.

El bloque tiene una variable booleana auxiliar llamada “TempBool”.

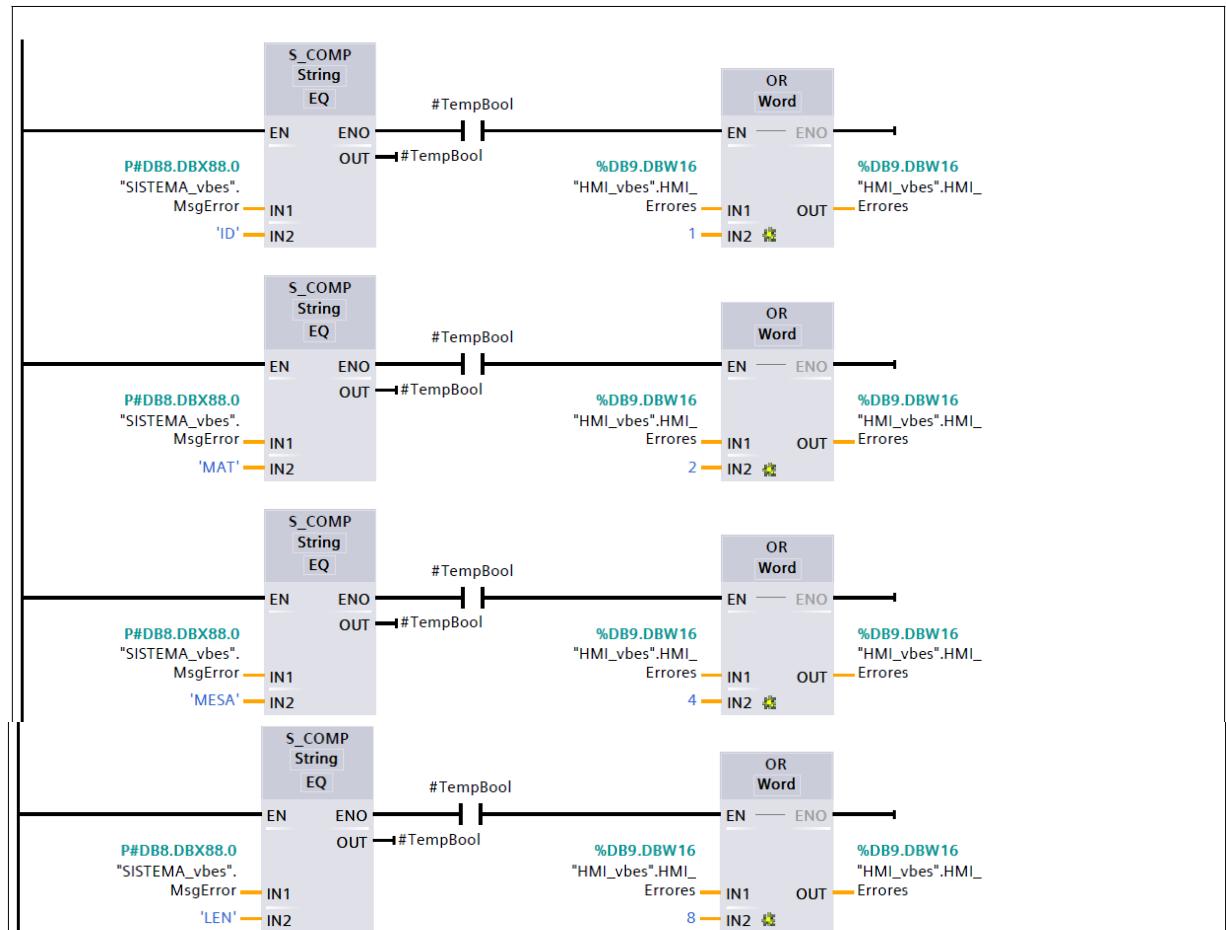
A continuación, se presenta el código de este bloque.

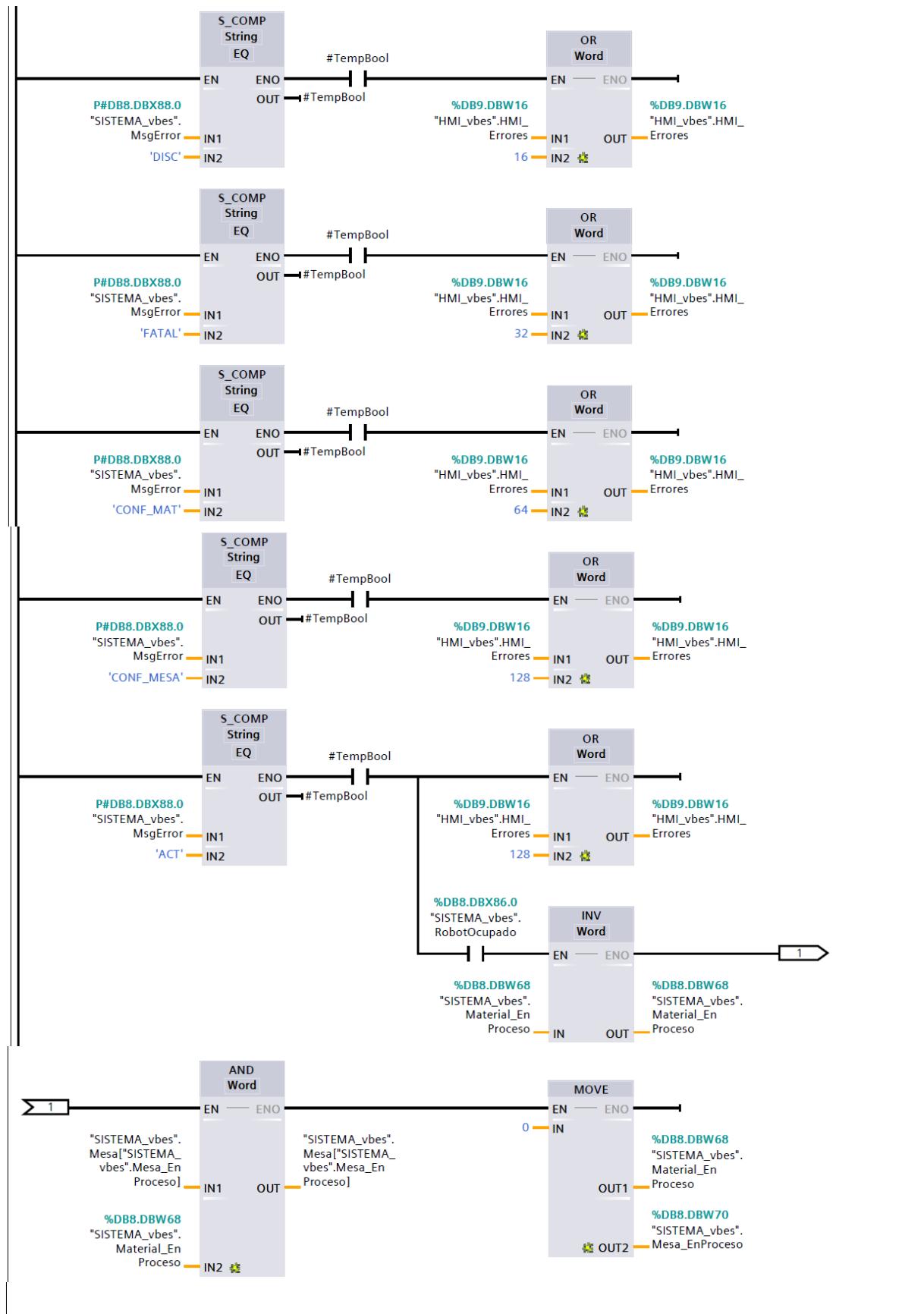
**Network 1: Si no se encuentra en el estado de error, ignorar**



**Network 2: Crear bits para alarmas de estado en HMI**

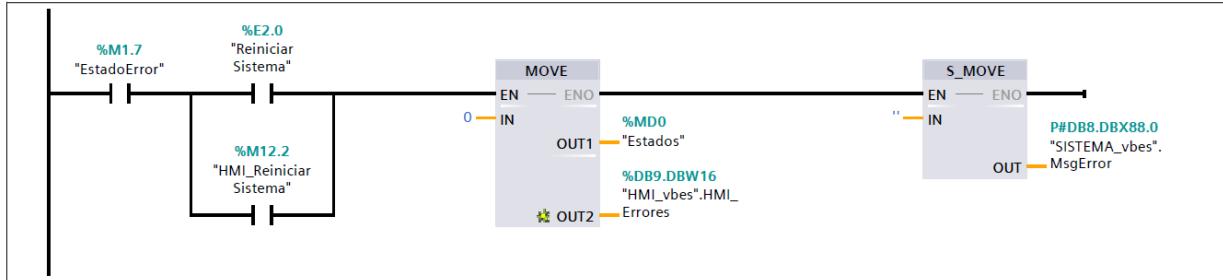
Se compara el string almacenado en MsgError con los mensajes conocidos y si es el mismo, se añade a los bits de errores.





### Network 3: Reiniciar errores y estados

En todos los casos, ante error el usuario debe reiniciar el sistema, aunque estaría mejor dicho rearmarlo. Cuando el sistema es rearmado se vuelve al estado inicial pero se siguen almacenando los valores dentro de SISTEMA\_vbes, luego el sistema continúa por dónde se quedó antes del error.



### 1.5. Procesar HMI

El bloque “CalcularHMI” está encargado de generar las señales propias del HMI para crear la interfaz visual; y, a su vez, es el encargado de procesar las entradas del usuario que realice desde la pantalla HMI. Muchos de estas operaciones deberían realizarse desde el HMI, pero debido a las limitaciones impuestas por la licencia de *WinCC Advanced*, hay que realizar la gran mayoría del procesamiento en el PLC.

Tiene las siguientes variables internas.

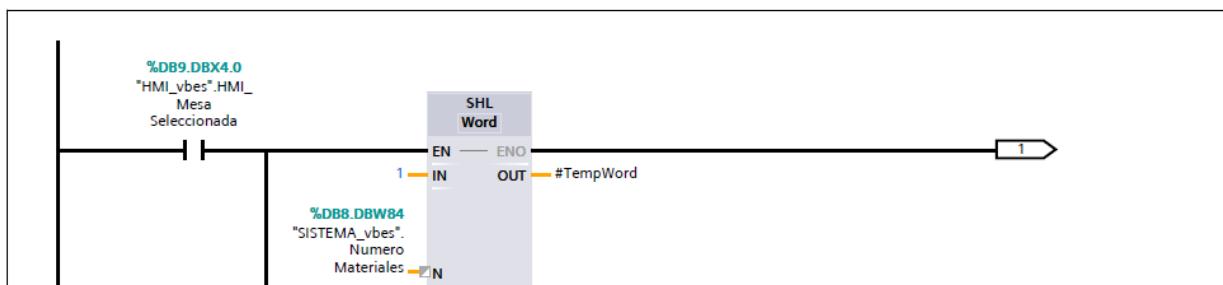
Nombre	Tipo	Descripción
TempWord	Word	Variable temporal de tipo Word.
MandarConfRobot	Bool	True si ha habido un cambio y se debe mandar una nueva configuración.

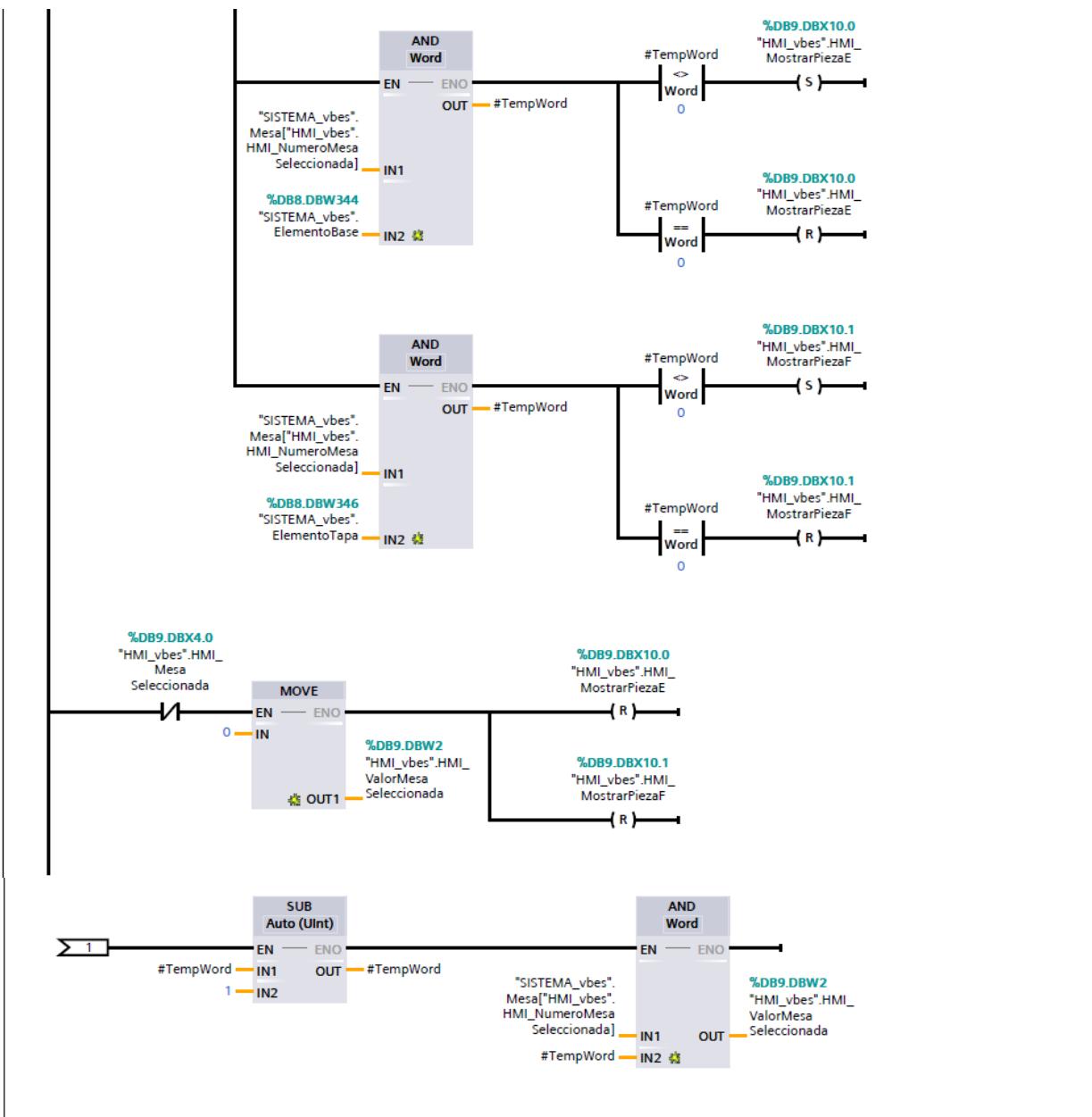
Tabla 28. Variables del bloque “CalcularHMI”.

A continuación, se presenta el código del bloque.

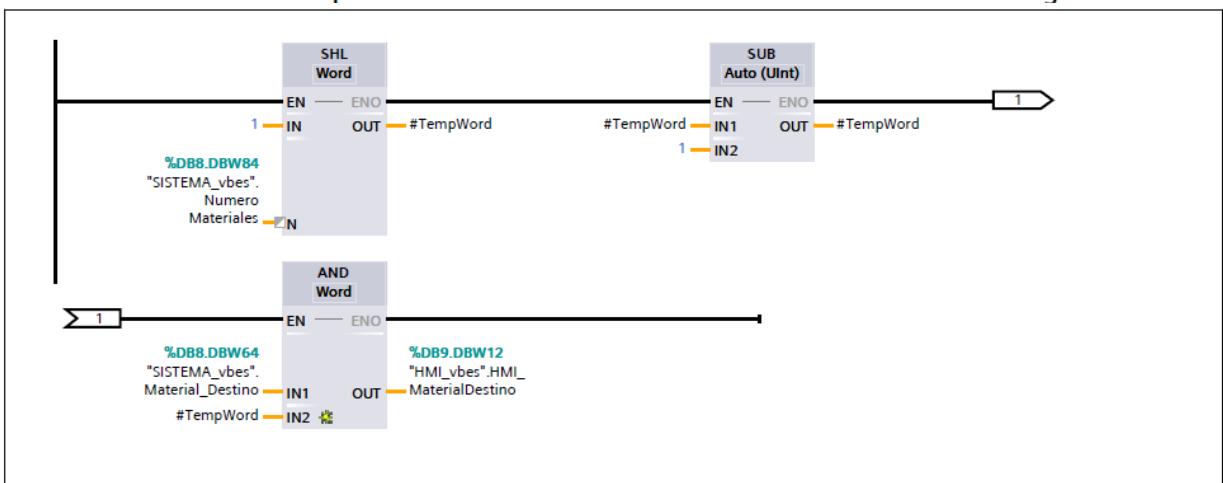
### Network 1: Generar los bools para que se muestren los componentes por mesa

En la primera rama se seleccionan únicamente los materiales (para el caso en el que el número de materiales de la bandeja es menor), para ello, se enmascaran.





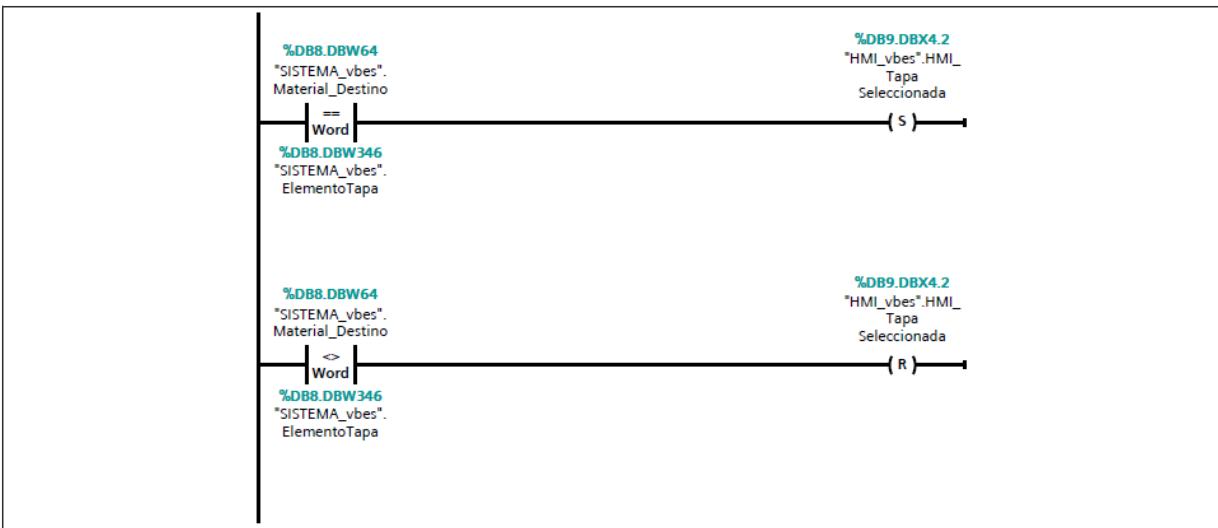
Network 2: Generar bools para mostrar flechas de selección de material en la cinta general



### Network 3: Generar bools para mostrar flechas de selección material E (base)

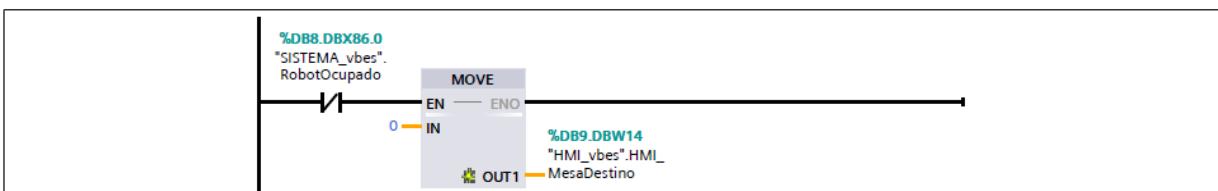


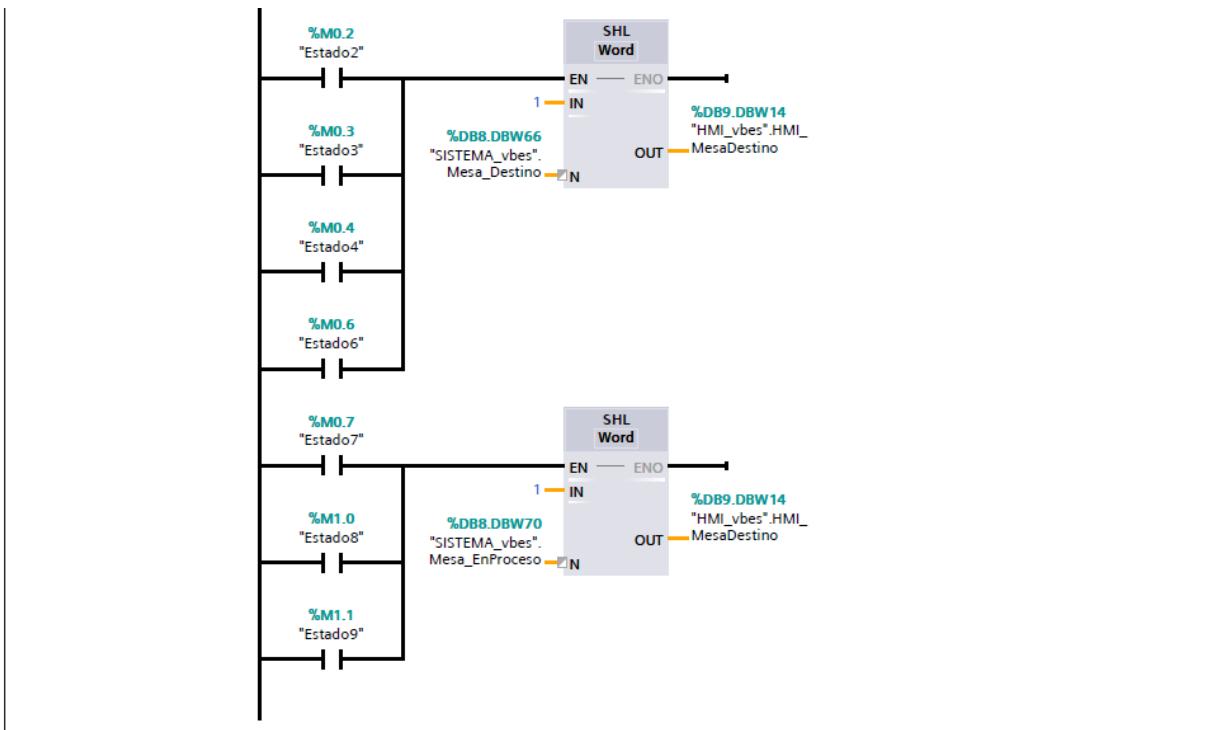
### Network 4: Generar bools para mostrar flechas de selección material F (tapa)



### Network 5: Mostrar colores de la mesa seleccionada

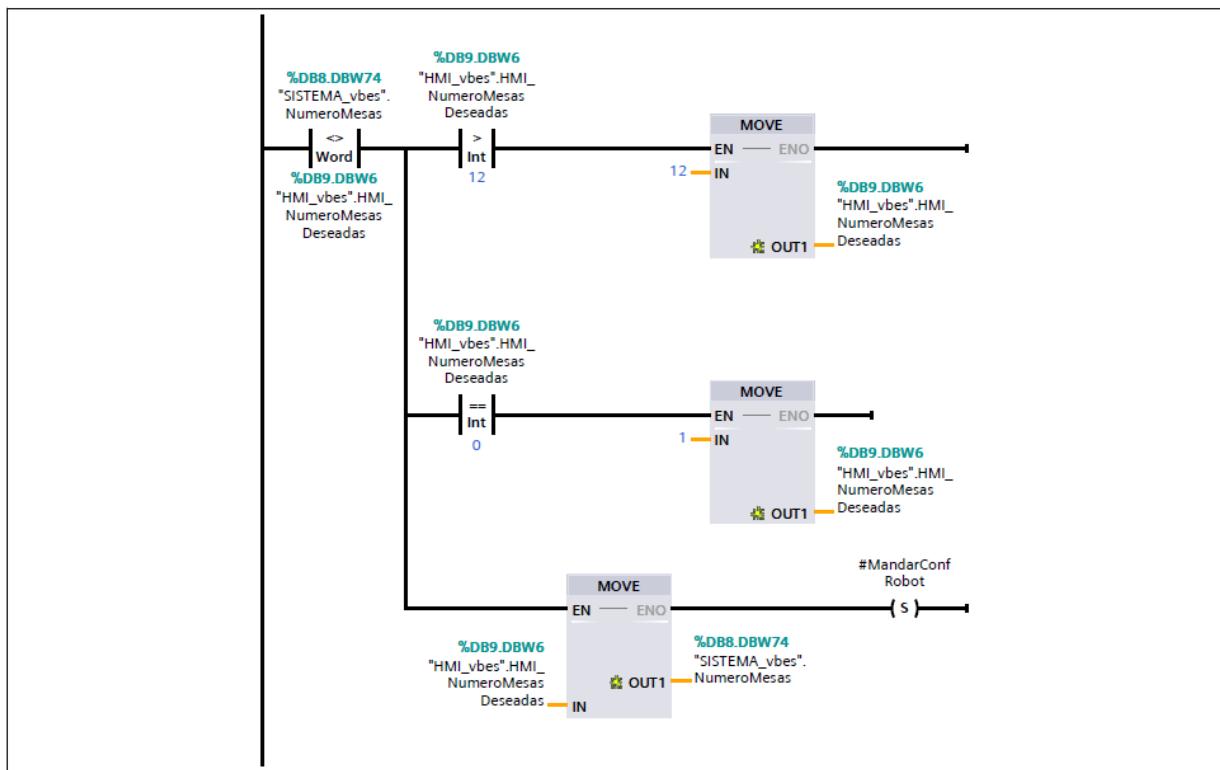
Se coloca un número grande para que no se seleccione ningún botón (esto es porque las mesas empiezan desde el número cero, luego ese valor neutral encendería el botón de la mesa cero). Si el sistema se encuentra en alguno de los estados donde tiene una mesa seleccionada, se pasa el valor real de la mesa al HMI.





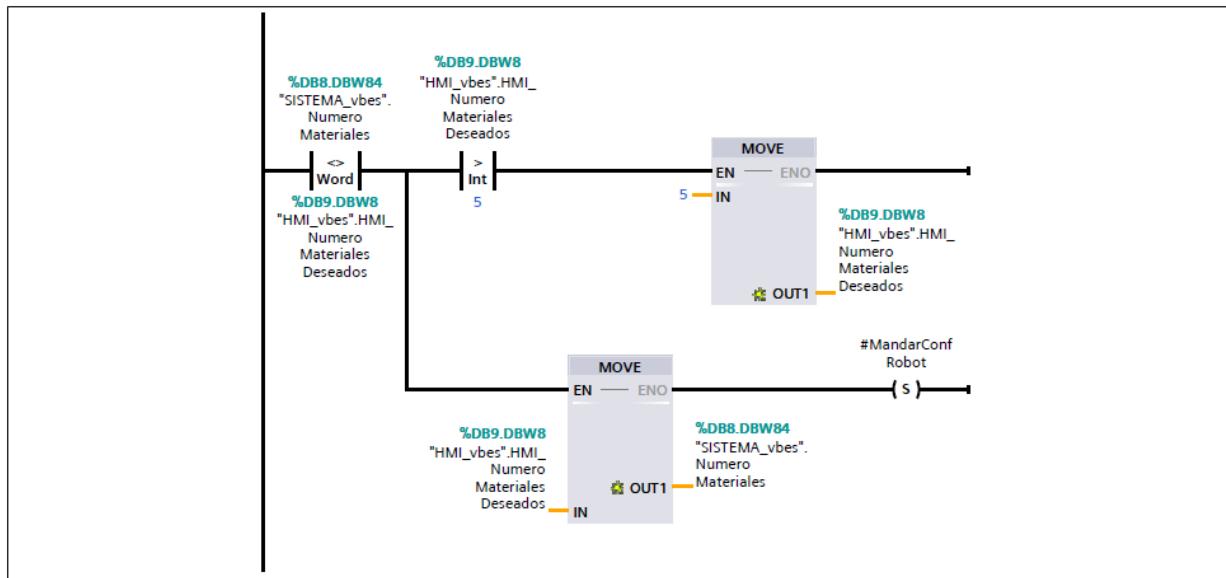
### Network 6: Comprobar los rangos máximos de número de mesas

Antes de aceptar la configuración introducida por el usuario desde HMI, comprobar si el valor es válido.

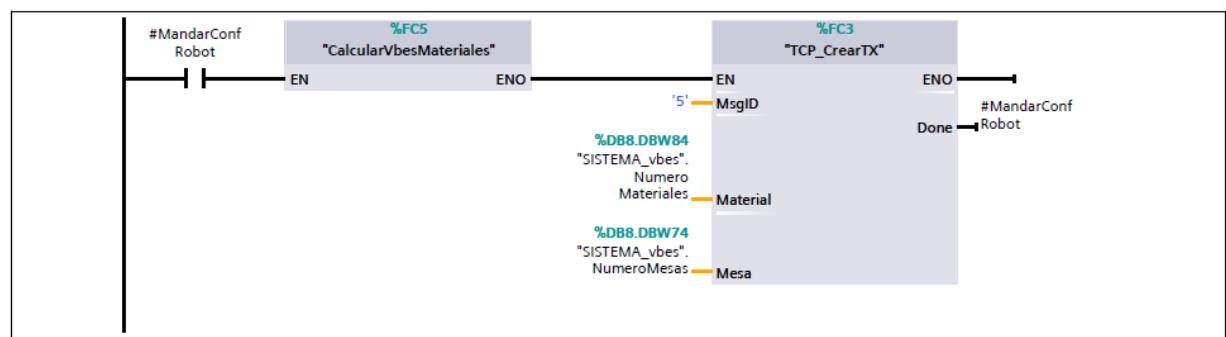


### Network 7: Comprobar los rangos máximos de número de componentes

Antes de aceptar la configuración introducida por el usuario desde HMI, comprobar si el valor es válido.

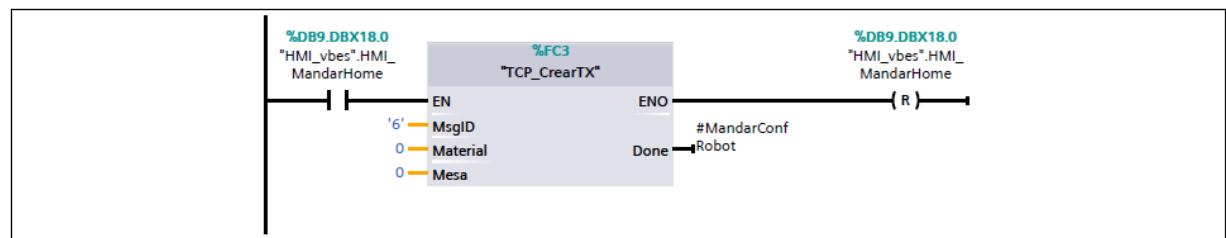


### Network 8: Si hay configuración nueva, mandarla al robot



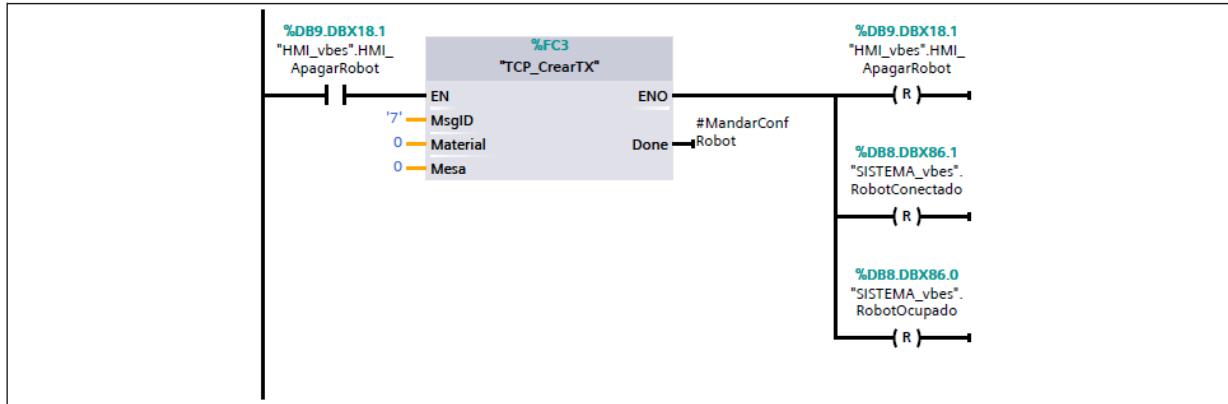
### Network 9: Procesar HOME

Desde el HMI, el robot puede ser mandado a HOME cuando acabe de realizar lo que esté haciendo. Esta condición se realiza desde el lado del robot.



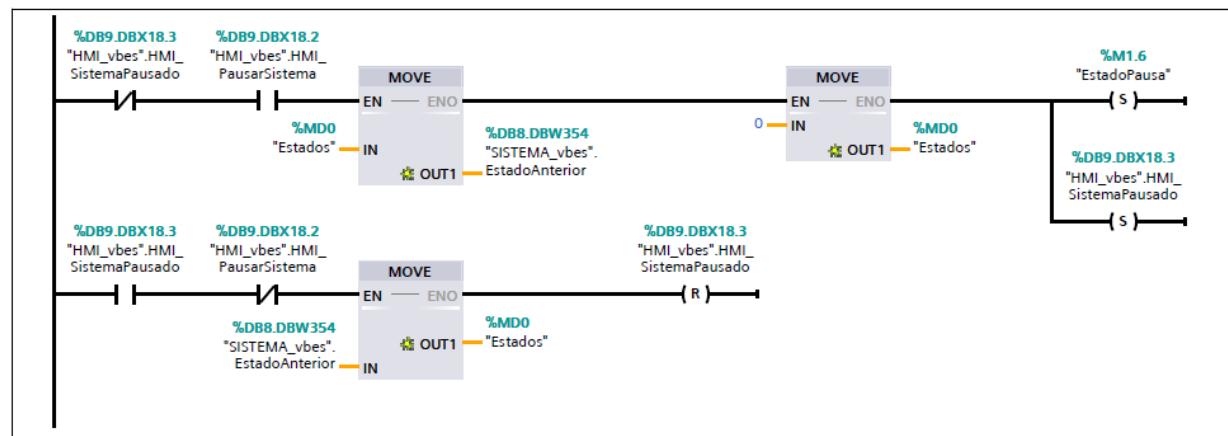
## Network 10: Procesar Apagar robot

Desde el HMI, el robot puede ser apagado cuando este no esté transportando ningún componente. Esta condición se realiza desde el lado del robot.



## Network 11: Pausar el sistema si se requiere desde HMI

El sistema HMI dispone de un botón que puede pausar el sistema completo.



## 1.6. Funciones extras

En esta sección se muestra el código de algunas de las funciones auxiliares que no forman parte del programa Main, pero aun así son llamadas esporádicamente.

### 1.6.1. Crear mensajes salientes (TCP)

En el bloque de función TCP\_CrearTX, se generan los mensajes a enviar al robot dados tres parámetros de entrada:

- **MsgID:** El ID del mensaje.
- **Material:** El número de material. Opcional si el tipo de mensaje no lo requiere.
- **Mesa:** El número de mesa. Opcional si el tipo de mensaje no lo requiere.

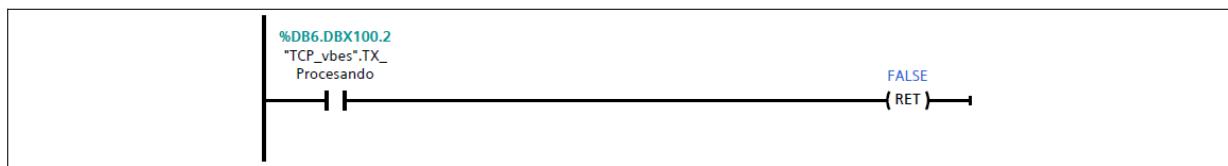
Las variables auxiliares que utiliza este bloque son las siguientes.

Nombre	Tipo	Descripción
NoACK	Bool	Si el mensaje no necesita ACK marcar como TRUE
TempCnt	UInt	Variable auxiliar de tipo UInt
ArgsStr	String	Variable auxiliar donde unir los distintos argumentos para después pasarlo al array TX[ ]
MesaByte	Byte	Variable temporal para pasar el valor de entrada Mesa (Word) a byte.
MsgListo	Bool	True si el mensaje está listo para ser enviado.

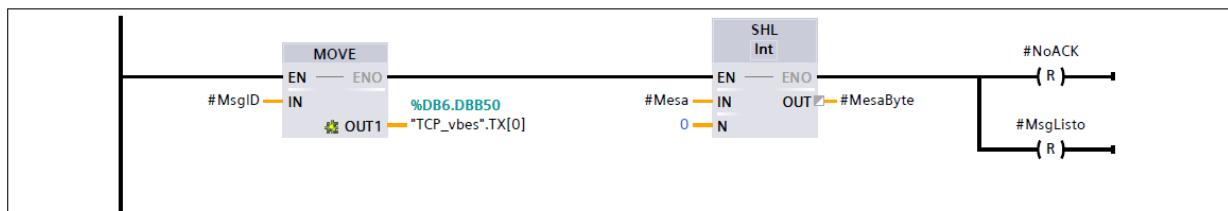
Tabla 29. Variables del bloque "TCP\_CrearTX".

A continuación, se muestra el código del programa.

Network 1: Si ya se está procesando un mensaje, el bloque no continuará su ejecución

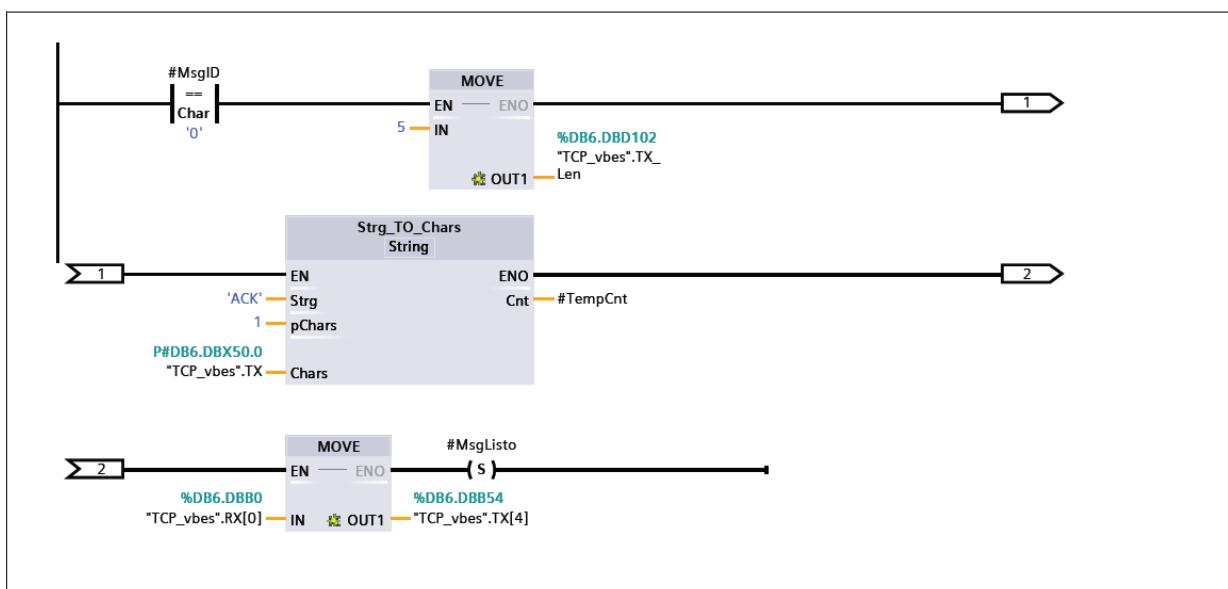


Network 2: Añadir ID del mensaje a la cabecera e inicializar variables

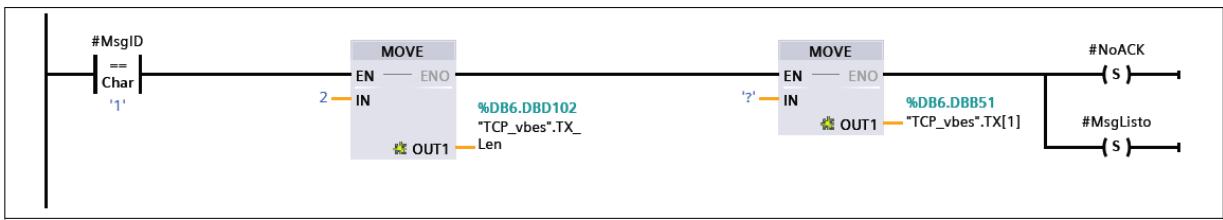


Network 3: Mensaje 0: ACK

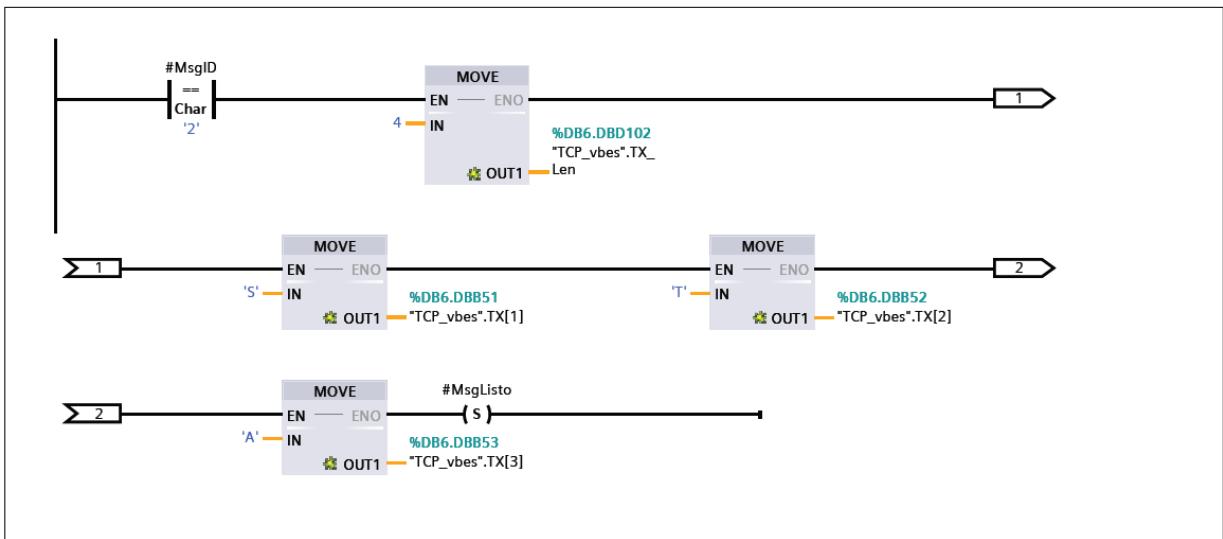
Manda un mensaje con únicamente la cabecera.



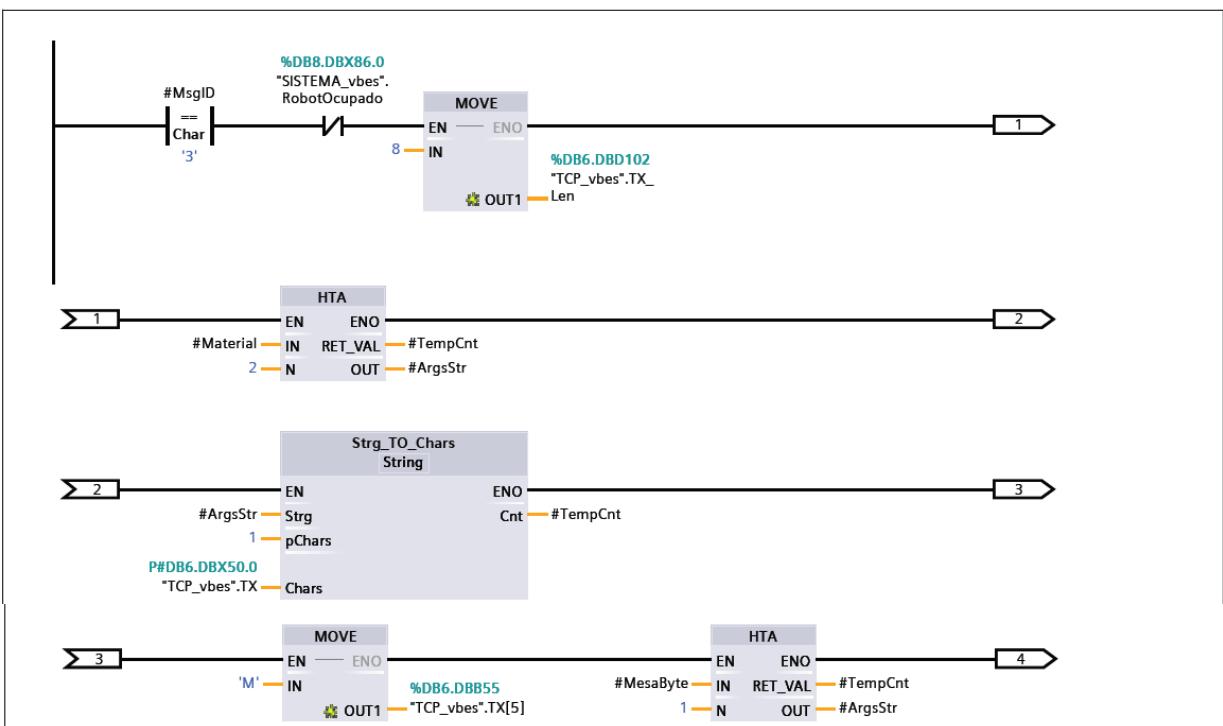
#### Network 4: Mensaje 1: Error

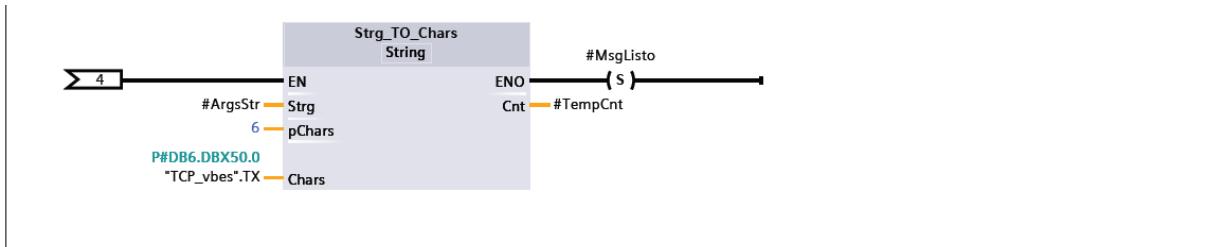


#### Network 5: Mensaje 2: Status

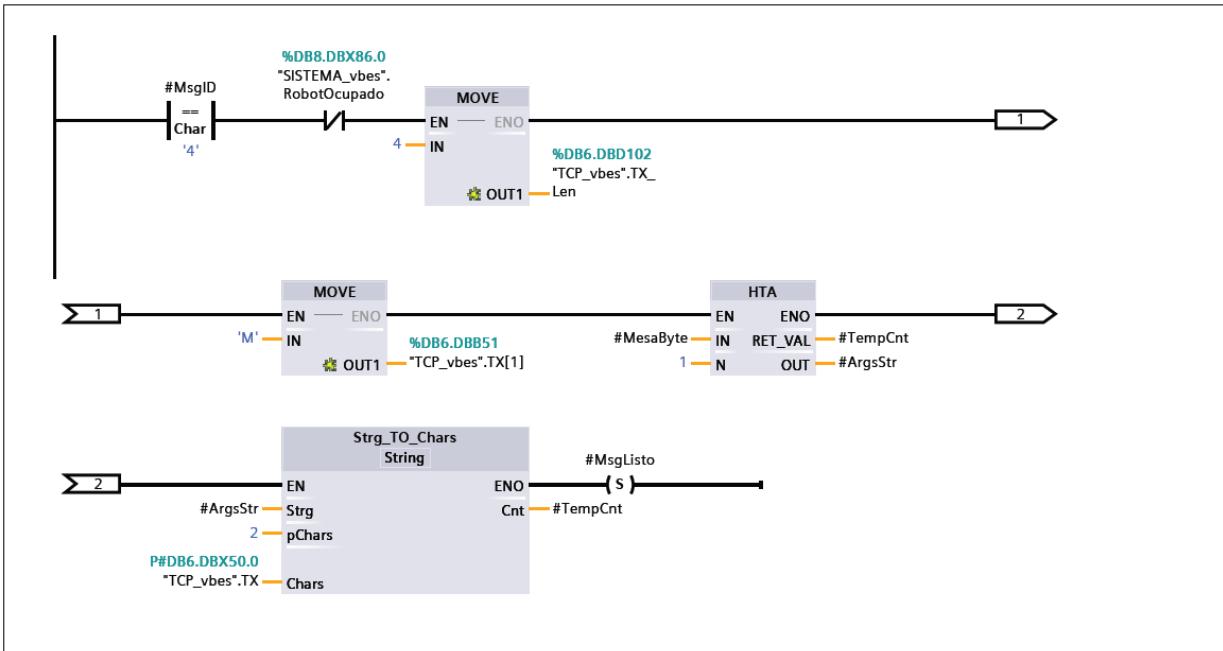


#### Network 6: Mensaje 3: Colocar material

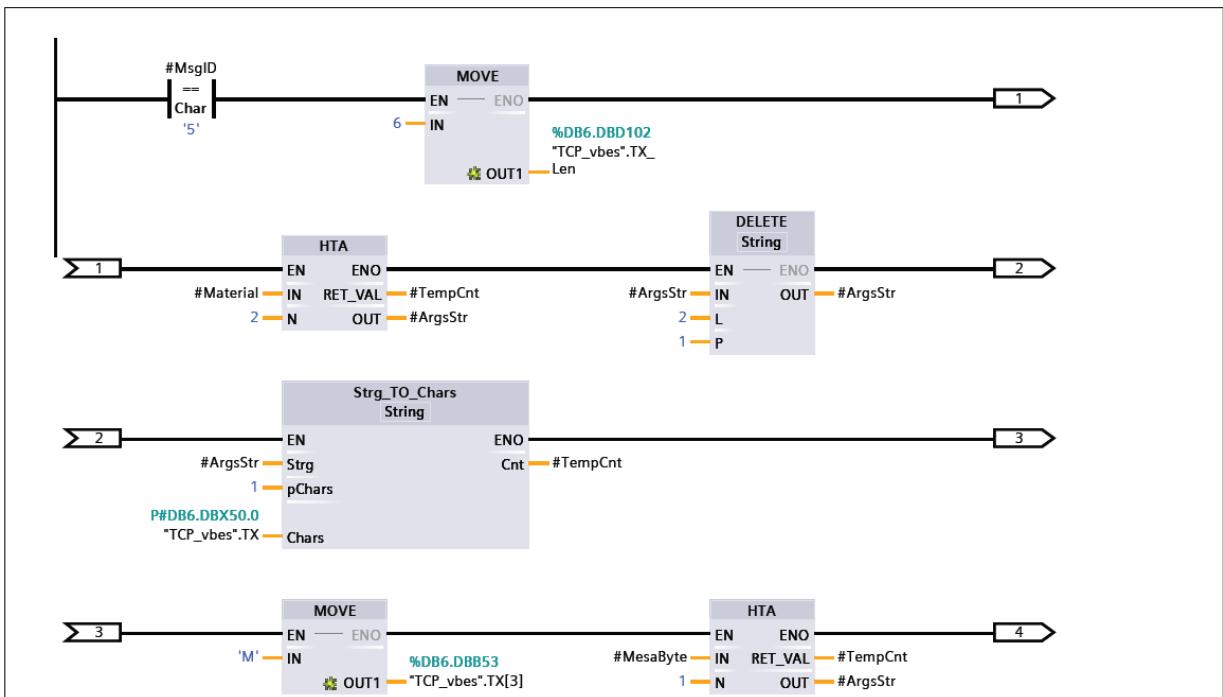




Network 7: Mensaje 4: Colocar caja en cinta de empaquetado

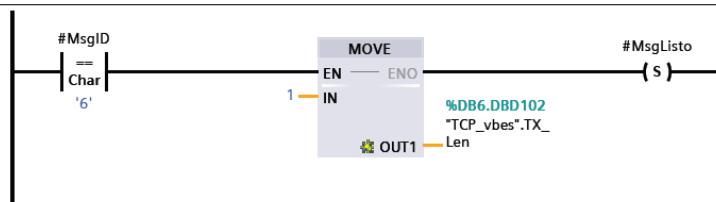


Network 8: Mensaje 5: Mandar valores de configuración

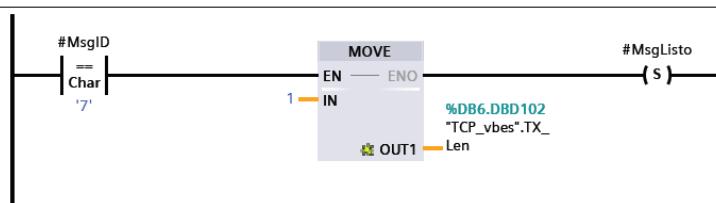




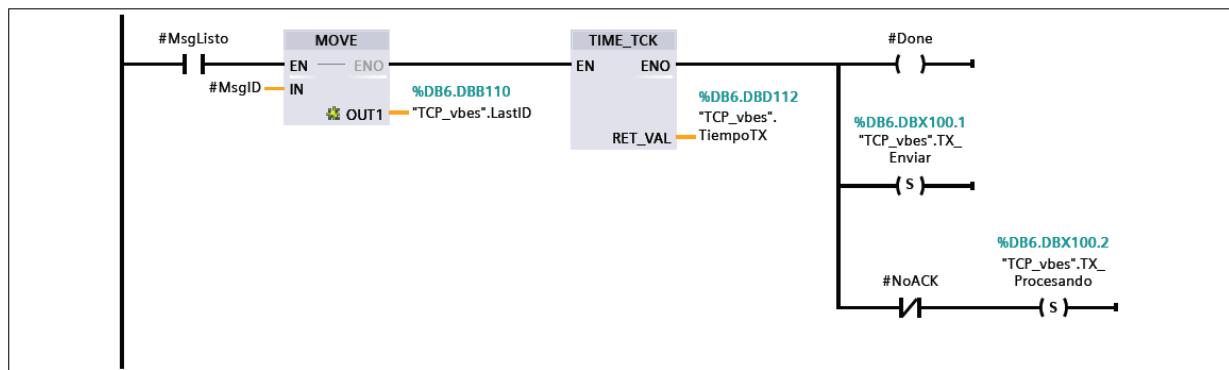
Network 9: Mensaje 6: Ir a HOME



Network 10: Mensaje 7: Apagar robot



Network 11: Enviar mensaje a TCP

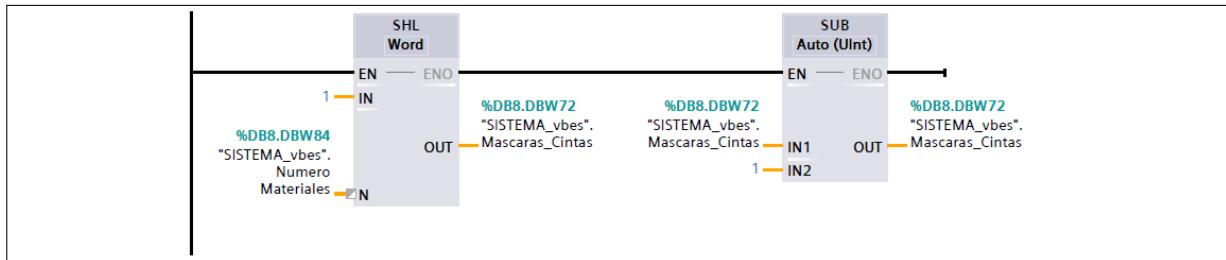


## 1.6.2.Calcular variables materiales

El bloque de función “CalcularVbesMateriales” se encarga de generar las máscaras de los sensores de presencia de las cintas según el número de materiales por bandeja que se vayan a utilizar en la planta. También genera las variables auxiliares “ElementoBase” y “ElementoTapa”, que sirven para identificar de manera individual la base y la tapa.

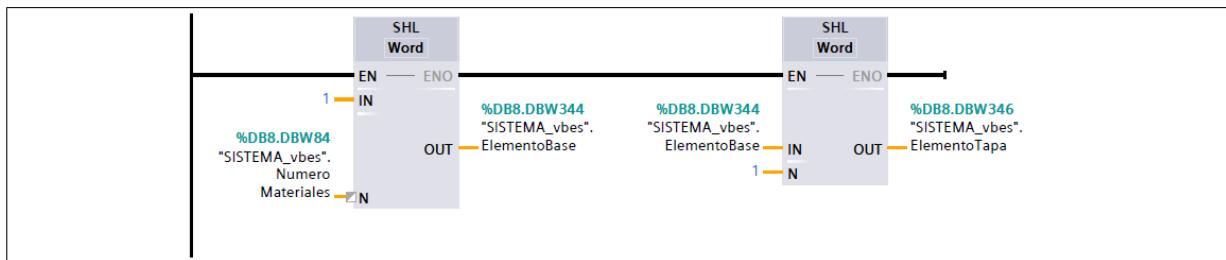
En la siguiente hoja se muestra el código del bloque.

### Network 1: Cálculo de la máscara de COMPONENTES (excluyen base y tapa)



### Network 2: Calculo del identificador de la base y tapa

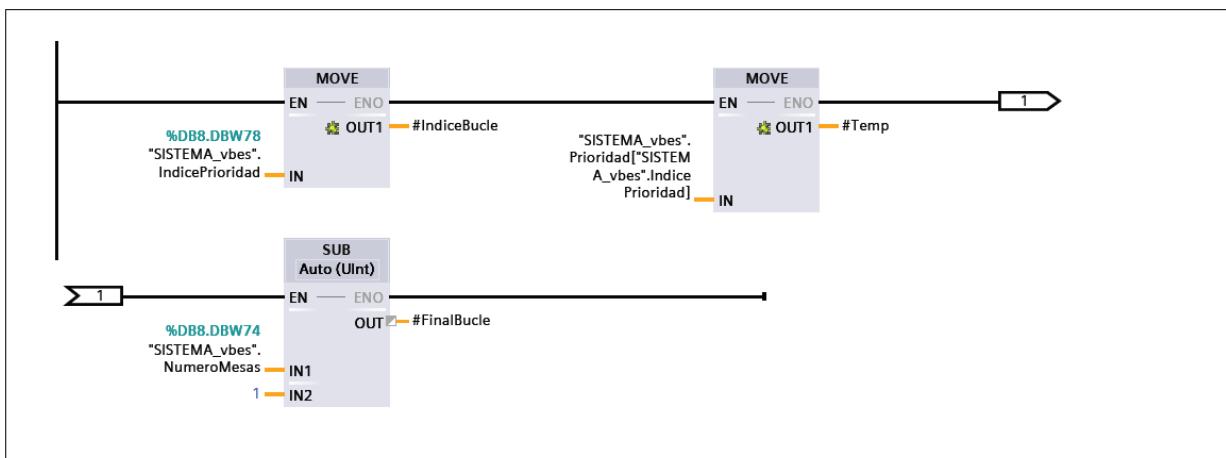
El bit de la base está en la posición NumeroMateriales. La tapa un bit a la izquierda más.



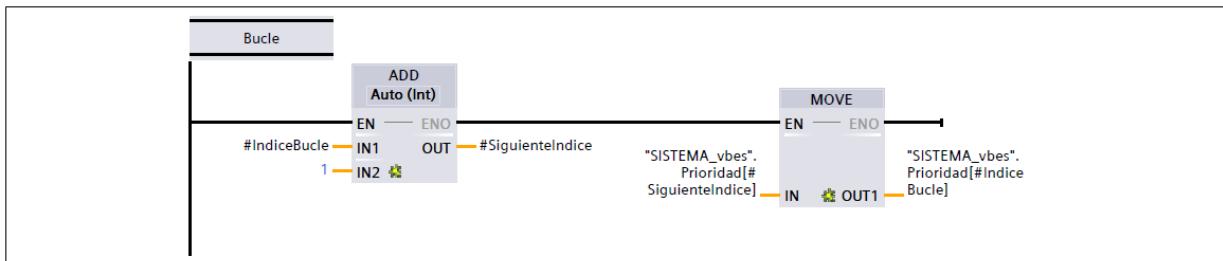
### 1.6.3.Modificar prioridades

Una vez se ha cerrado una caja, debe modificarse el orden de prioridad, manteniendo aun así el orden relativo entre componentes. Esta es la función del bloque "ModificarPrioridades". Esta desplaza la lista de prioridades cuando una caja está completada. La mesa actual pasa al final de la lista.

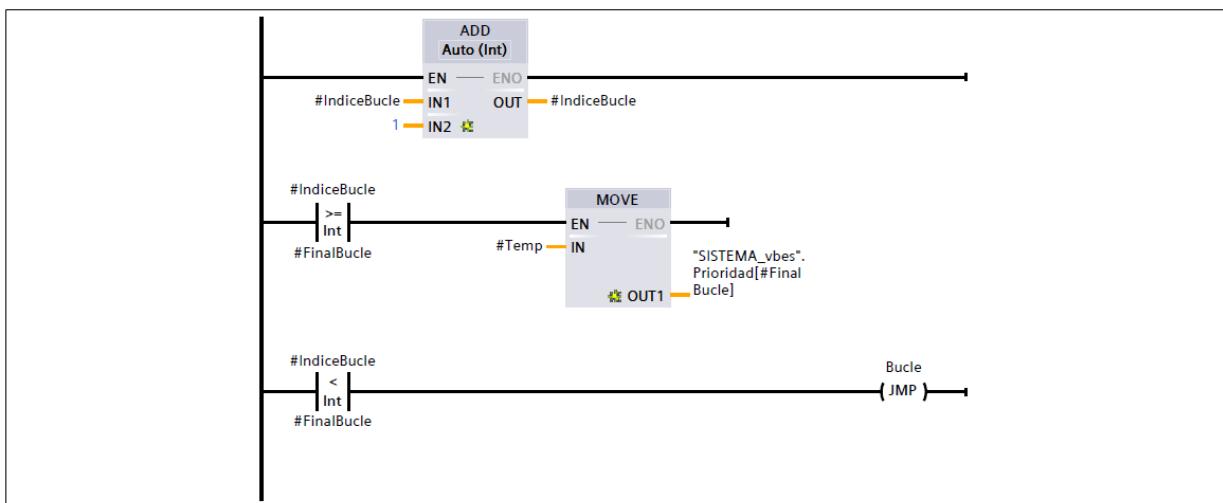
#### Network 1: Inicialización de variables



## Network 2: Mover a la izquierda los componentes del array



## Network 3: Avanzar índices



**(Final del Anexo 1)**

## 2. Anexo 2: Código RAPID

En este anexo se presenta el código RAPID del robot con comentarios para clarificar la función de cada línea. RobotStudio no permite guardar caracteres especiales en el código por ello no hay ningún acento en los comentarios.

```

1 MODULE Module1
2 ! Declaracion de los targets.
3 CONST robtarget Targ_CajaE:=[[-500,1500,200],[0,0,1,0],[1,0,1,0],
4 [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
5 CONST robtarget Targ_CajaF:=[[400,1500,200],[0,0,1,0],[0,0,0,0],
6 [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
7 CONST robtarget Targ_CintaEmpaq:=[[-0.000065567,-1500.000031808,200.000065078],
8 [0,0,1,-0.000000008],[-2,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
9 CONST robtarget Targ_CintaLlegada:=[[-1500.000032056,1000.00001355,200.000069515],
10 [-0.00000007,0,1,0.00000004],[1,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
11 CONST robtarget Targ_Mesa:=[[999.999977515,1500.00004795,215.000069515],
12 [0.00000008,0.707106781,0.707106781,0.00000002],[0,-1,1,0],
13 [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
14 ! Numero maximo de posiciones dentro de la mesa.
15 CONST num maximoNumMesas := 12;
16 ! Numero maximo de materiales para los que el robot ha sido programado.
17 CONST num maximoNumMateriales := 5;
18
19 VAR socketdev serverSocket;
20 VAR socketdev clientSocket;
21 VAR bool servidorConectado := FALSE;
22 VAR string inputData;
23
24 ! Almacena el numero de componentes (sin contar base y tapa) que maneja el robot.
25 VAR num numeroMateriales := maximoNumMateriales;
26 ! Almacena el numero maximo de mesas que esta manejando el sistema en ese momento.
27 VAR num numeroMesas := maximoNumMesas;
28 ! TRUE si hay algun material pegado a las ventosas. Variable necesaria por si falla la ventosa.
29 VAR bool transportandoMaterial := FALSE;
30
31 ! Interrupcion para gestionar la deteccion de piezas.
32 VAR intnum int_DISensorPieza;
33
34 ! **** main() es la funcion de comienzo del programa. Si termina, se vuelve a ejecutar desde el
35 ! comienzo.
36 ! **** PROC main()
37 ! Enlazar interrupcion con su funcion de interrupcion.

```

```

36   CONNECT int_DisSensorPieza WITH IntSensorPiezas;
37   ! Disparar interrupcion unicamente cuando falle el sensor de piezas (pase de uno a cero la
38   ! entrada DI_SensorPieza).
39   ISignalDI DI_SensorPieza, 0, int_DisSensorPieza;
40
41   ! Esperar conexion entrante del PLC
42   ConectarTCP;
43   ! Mientras este conectado, realizar las peticiones del PLC.
44   WHILE servidorConectado DO
45       ProcesarMensajesTCP;
46   ENDWHILE
47   ENDPROC
48
49   !*****!
50   ! IntSensorPiezas es la funcion que se ejecuta cuando el sensor da un fallo mientras se esta
51   ! transportando material.
52   !*****!
53   TRAP IntSensorPiezas
54   IF transportandoMaterial THEN
55       EscribirTCP "1?ACT";
56       StopMove;
57   ENDIF
58   ENDTRAP
59
60   !*****!
61   ! ProcesarMensajesTCP() recibe los mensajes por TCP y realiza los movimientos u operaciones que
62   ! el PLC instruya al robot.
63   !*****!
64   PROC ProcesarMensajesTCP()
65       VAR bool msgOK;
66       VAR num inputID;
67       VAR num matDestino;
68       VAR num mesaDestino;
69       VAR bool mensajeRecibido;
70
71       ! Recepcion de datos desde el TCP (bloquea el programa durante 10 segundos). Si no se recibe
72       ! nada envia un mensaje para tratar de conectarse.
73       mensajeRecibido := LeerTCP(10);
74       IF NOT mensajeRecibido THEN
75           ! Trata de reconectar.
76           SolicitarConfiguracion;
77           RETURN;
78       ENDIF
79

```

```

80 ! Convertir el numero inicial del mensaje a numero.
81 msgOK := StrToVal(StrPart(inputData,1,1), inputID);
82 IF NOT msgOK THEN
83     ! Mandar mensaje de error de transmision.
84     ErrorRX "ID";
85     RETURN;
86 ENDIF
87
88 ! Coger material.
89 IF inputID = 3 THEN
90     IF StrLen(inputData) < 8 THEN
91         ErrorRX "LEN";
92         RETURN;
93     ENDIF
94
95     ! Obtener la ID del material que se desea coger.
96     matDestino := ConvertirStringMaterial(StrPart(inputData,2,4));
97     IF matDestino < 0 OR matDestino >= (numeroMateriales+2) THEN
98         ! Error al parsear la ID del material.
99         ErrorRX "MAT";
100        RETURN;
101    ENDIF
102
103    ! Activar senales digitales para que ReponedorPiezas genere las piezas
104    ! correspondientes.
105    IF matDestino = numeroMateriales THEN
106        PulseDO \High,\PLength:=0.5, DO_PiezaE;
107    ELSEIF matDestino = numeroMateriales+1 THEN
108        PulseDO \High,\PLength:=0.5, DO_PiezaF;
109    ELSEIF matDestino = 0 THEN
110        PulseDO \High,\PLength:=0.5, DO_PiezaA;
111    ELSEIF matDestino = 1 THEN
112        PulseDO \High,\PLength:=0.5, DO_PiezaB;
113    ELSEIF matDestino = 2 THEN
114        PulseDO \High,\PLength:=0.5, DO_PiezaC;
115    ELSEIF matDestino = 3 THEN
116        PulseDO \High,\PLength:=0.5, DO_PiezaD;
117    ELSEIF matDestino = 4 THEN
118        PulseDO \High,\PLength:=0.5, DO_PiezaD;
119    ELSE
120        ! Mandar mensaje de error fatal, no deberia ser posible entrar aqui.
121        ErrorRX "FATAL";
122        RETURN;
123    ENDIF

```

```

124
125      ! Obtener la ID de la mesa destino de la pieza.
126      mesaDestino := StrToByte(StrPart(inputData,7,2)\Hex);
127      IF mesaDestino >= maximoNumMesas THEN
128          ErrorRX "MESA";
129          RETURN;
130      ENDIF
131
132      ! Enviar el ACK.
133      EscribirTCP "0ACK3";
134
135      ! Comprobacion rapida por si hay un mensaje para detener el programa.
136      mensajeRecibido := LeerTCP(0.5);
137      IF mensajeRecibido THEN
138          ! Convertir el numero inicial del mensaje a numero.
139          msgOK := StrToInt(StrPart(inputData,1,1), inputID);
140          IF NOT msgOK THEN
141              ! Mandar mensaje de error de transmision.
142              ErrorRX "ID";
143              RETURN;
144          ENDIF
145          ComprobarAsincronas(inputID);
146      ENDIF
147
148      ! Algoritmo de posicionamiento de objetos en mesas.
149      ColocarEnMesa matDestino, mesaDestino;
150
151      ! Una vez colocado el objeto, manda senal de MaterialColocado.
152      EscribirTCP "3MAT_COL";
153
154      ! Coloca una caja completa en la cinta de empaquetado.
155      ELSEIF inputID = 4 THEN
156          IF StrLen(inputData) < 4 THEN
157              ErrorRX "LEN";
158              RETURN;
159          ENDIF
160
161      ! Obtener la ID de la posicion donde se encuentra la caja que se desea colocar
162      ! en la cinta general.
163      mesaDestino := StrToByte(StrPart(inputData,3,2)\Hex);
164      IF mesaDestino > maximoNumMesas THEN
165          ErrorRX "MESA";
166          RETURN;
167      ENDIF

```

```

168
169      ! Enviar el ACK.
170      EscribirTCP "0ACK4";
171
172      ! Comprobacion rapida por si hay un mensaje para detener el programa.
173      mensajeRecibido := LeerTCP(0.5);
174      IF mensajeRecibido THEN
175          ! Convertir el numero inicial del mensaje a numero.
176          msgOK := StrToVal(StrPart(inputData,1,1), inputID);
177          IF NOT msgOK THEN
178              ! Mandar mensaje de error de transmision.
179              ErrorRX "ID";
180              RETURN;
181          ENDIF
182          ComprobarAsincronas(inputID);
183      ENDIF
184
185      ! Algoritmo de posicionamiento de cajas completas en la cinta de empaquetado.
186      ColocarCajaCompleta mesaDestino;
187
188      ! Una vez colocada la caja, manda senal de MaterialColocado.
189      EscribirTCP "3MAT_COL";
190
191      ! Ir a HOME y otras senales.
192  ELSE
193      ComprobarAsincronas(inputID);
194  ENDIF
195 ENDPROC
196
197 ****
198 ! ColocarEnMesa(num mat, num mesa) recibe la ID del material y de la mesa donde se desea colocar
199 ! dicho material y genera las instrucciones de movimiento del brazo del robot.
200 ****
201 PROC ColocarEnMesa(num mat, num mesa)
202     VAR num x;
203     VAR num y;
204     VAR num xPos;
205     VAR num yPos;
206     VAR robtarget posicionFinal;
207
208     ! Obtener la posicion X,Y dentro de la mesa. La esquina superior izquierda desde el punto
209     ! de vista de la cinta de empaquetado es el 0,0. El numero de la mesa se puede recalcular
210     ! como X*6+Y.
211     x := mesa DIV 6;

```

```

212 y := mesa MOD 6;
213
214 ! Obtener las esquinas de la posicion de la mesa que se indica. A partir de estas se
215 ! desplazaran las coordenadas donde se encuentran individualmente las piezas con respecto la
216 ! bandeja.
217 xPos := 200 + x*400;
218 yPos := 80 + y*500;
219
220 ! Punto final donde se colocara el componente. En esta linea es la posicion de la esquina
221 ! superior izquierda de la mesa que se esta pasando a esta funcion en referencia al mundo.
222 posicionFinal := Offs(Targ_Mesa, xPos, -yPos, 0);
223
224 ! Dependiendo del material, la posicion donde se colocara variara. Esto ira en referencia
225 ! a la posicion calculada anteriormente.
226 IF mat = numeroMateriales THEN
227     ! Base de la caja.
228     posicionFinal := Offs(posicionFinal, 115,-170,40);
229 ELSEIF mat = numeroMateriales+1 THEN
230     ! Tapadera de la caja.
231     posicionFinal := Offs(posicionFinal, 115,-170,93);
232 ELSEIF mat = 0 THEN
233     posicionFinal := Offs(posicionFinal, 75,-225,85);
234 ELSEIF mat = 1 THEN
235     posicionFinal := Offs(posicionFinal, 177,-283,85);
236 ELSEIF mat = 2 THEN
237     posicionFinal := Offs(posicionFinal, 177,-167,85);
238 ELSEIF mat = 3 THEN
239     posicionFinal := Offs(posicionFinal, 60,-60,85);
240 ELSEIF mat = 4 THEN
241     posicionFinal := Offs(posicionFinal, 170,-60,85);
242 ENDIF
243
244 ! Seleccion de la posicion donde se recogera el material.
245 IF mat = numeroMateriales THEN
246     ! Posicion de recogida de la caja.
247     MoveJ Offs(Targ_CajaE,0,0,400),v5000,z20,GarraVentosas\WObj:=wobj0;
248     MoveL Targ_CajaE,v500,fine,GarraVentosas\WObj:=wobj0;
249     PulseDO \High,\PLength:=0.5, D0_Vacio;
250     transportandoMaterial := TRUE;
251     MoveL Offs(Targ_CajaE,0,0,400),v1000,z20,GarraVentosas\WObj:=wobj0;
252 ELSEIF mat = numeroMateriales+1 THEN
253     ! Posicion de recogida de la tapa.
254     MoveJ Offs(Targ_CajaF,0,0,400),v5000,z20,GarraVentosas\WObj:=wobj0;
255     MoveL Targ_CajaF,v500,fine,GarraVentosas\WObj:=wobj0;

```

```

256     PulseDO \High,\PLength:=0.5, DO_Vacio;
257     transportandoMaterial := TRUE;
258     MoveL Offs(Targ_CajaF,0,0,400),v1000,z20,GarraVentosas\WObj:=wobj0;
259 ELSE
260     ! Componentes de la cinta general.
261     MoveJ Offs(Targ_CintaLlegada,0,0,400),v5000,z20,GarraVentosas\WObj:=wobj0;
262     MoveL Targ_CintaLlegada,v500,fine,GarraVentosas\WObj:=wobj0;
263     PulseDO \High,\PLength:=0.5, DO_Vacio;
264     transportandoMaterial := TRUE;
265     MoveL Offs(Targ_CintaLlegada,0,0,400),v1000,z20,GarraVentosas\WObj:=wobj0;
266 ENDIF
267
268 ! Mandar señal de material recogido.
269 EscribirTCP "2MAT_REC";
270
271 ! Dejar el material en la posición final.
272 MoveJ Offs(posicionFinal,0,0,400),v5000,z20,GarraVentosas\WObj:=wobj0;
273 MoveL posicionFinal,v500,fine,GarraVentosas\WObj:=wobj0;
274 PulseDO \High,\PLength:=0.5, DO_Soplar;
275 transportandoMaterial := FALSE;
276 MoveL Offs(posicionFinal,0,0,400),v1000,fine,GarraVentosas\WObj:=wobj0;
277 ENDPROC
278
279 ! ****
280 ! ColocarCajaCompleta(num mat) coloca una caja completa en la cinta de empaquetado.
281 ! ****
282 PROC ColocarCajaCompleta(num mesa)
283     VAR num x;
284     VAR num y;
285     VAR num xPos;
286     VAR num yPos;
287     VAR robtarget posicionFinal;
288
289     ! Obtener la posición X,Y dentro de la mesa. La esquina superior izquierda desde el punto
290     ! de vista de la cinta de empaquetado es el 0,0. El número de la mesa se puede recalcular
291     ! como X*6+Y.
292     x := mesa DIV 6;
293     y := mesa MOD 6;
294
295     ! Anadir la posición del centro de la caja (115, 170); es decir, la mitad de las
296     ! dimensiones.
297     xPos := 200 + x*400 + 115;
298     yPos := 80 + y*500 + 170;
299

```

```

300 ! Posicion del centro de la caja en la mesa.
301 posicionFinal := Offs(Targ_Mesa, xPos, -yPos, 93);
302
303 MoveJ Offs(posicionFinal,0,0,400),v5000,z20,GarraVentosas\WObj:=wobj0;
304 MoveL posicionFinal,v500,fine,GarraVentosas\WObj:=wobj0;
305 PulseDO \High,\PLength:=0.5, DO_VacioCaja;
306 transportandoMaterial := TRUE;
307 MoveL Offs(posicionFinal,0,0,400),v1000,z20,GarraVentosas\WObj:=wobj0;
308
309 ! Dejar el material en la cinta de empaquetado.
310 MoveJ Offs(Targ_CintaEmpaq,0,0,400),v5000,z20,GarraVentosas\WObj:=wobj0;
311 MoveL Targ_CintaEmpaq,v500,fine,GarraVentosas\WObj:=wobj0;
312 transportandoMaterial := FALSE;
313 PulseDO \High,\PLength:=0.5, DO_SoplarCaja;
314 MoveL Offs(Targ_CintaEmpaq,0,0,400),v1000,fine,GarraVentosas\WObj:=wobj0;
315 ENDPROC
316
317 ****
318 ! ConectarTCP() esta funcion realiza la conexion mediante TCP al socket del cliente. Una vez
319 ! realizada la conexion, manda un mensaje con su configuracion inicial al PLC y espera una
320 ! respuesta para confirmar que esta correctamente conectado.
321 ****
322 PROC ConectarTCP()
323     ! Inicializacion de los sockets.
324     SocketCreate serverSocket;
325     SocketBind serverSocket, "127.0.0.1", 9876;
326     SocketListen serverSocket;
327     SocketAccept serverSocket, clientSocket,\Time:=WAIT_MAX;
328     SolicitarConfiguracion;
329 ENDPROC
330
331 ****
332 ! SolicitarConfiguracion() manda al PLC un mensaje con la configuracion actual del robot. Espera
333 ! que el PLC le devuelva la configuracion que esta utilizando y la adopta si es valida.
334 ****
335 PROC SolicitarConfiguracion()
336     VAR bool msgOK;
337     VAR num inputID;
338
339 SolicitarConfig:
340     ! Mandar mensaje de conexion completada junto con el numero maximo de mesas y materiales
341     ! permitidos.
342     EscribirTCP "405M0C";
343     ! Espera la respuesta del sistema con la configuracion.

```

```

344     msgOK := LeerTCP(5);
345     IF NOT msgOK THEN
346         GOTO SolicitarConfig;
347     ENDIF
348
349     ! Comprobar la longitud de la cadena.
350     IF StrLen(inputData) < 6 THEN
351         ErrorRX("LEN");
352         GOTO SolicitarConfig;
353     ENDIF
354
355     ! Comprobar que la ID es la de Parametros.
356     msgOK := StrToInt(StrPart(inputData,1,1), inputID);
357     IF inputID <> 5 THEN
358         GOTO SolicitarConfig;
359     ENDIF
360
361     ! Obtener el numero maximo de materiales por bandeja.
362     numeroMateriales := StrToInt(StrPart(inputData,2,2)\Hex);
363     IF numeroMateriales > maximoNumMateriales THEN
364         ErrorRX("CONF_MAT");
365         GOTO SolicitarConfig;
366     ENDIF
367
368     ! Obtener el numero de mesas que puede haber.
369     numeroMesas := StrToInt(StrPart(inputData,5,2)\Hex);
370     IF numeroMesas > maximoNumMesas THEN
371         ErrorRX("CONF_MESA");
372         GOTO SolicitarConfig;
373     ENDIF
374
375     ! Enviar ACK. La conexion con el PLC ya ha sido establecida.
376     EscribirTCP "0ACK5";
377     servidorConectado := TRUE;
378 ENDPROC
379
380 ****
381 ! ErrorRX(string errMsg) es una funcion que envia al PLC un mensaje de error tras la recepcion de
382 ! un mensaje que el PLC ha mandado.
383 ****
384 PROC ErrorRX(string errMsg)
385     EscribirTCP "1?"+errMsg;
386     PulseDO \High,\PLength:=1, D0_ErrorRX;
387 ENDPROC

```

```

388
389 ! ****
390 ! Reconectar() cierra los sockets y vuelve a intentar realizar una conexion con el PLC.
391 !
392 PROC Reconectar()
393     SocketClose serverSocket;
394     SocketClose clientSocket;
395     servidorConectado := FALSE;
396     ConectarTCP;
397 ENDPROC
398
399 ! ****
400 ! bool LeerTCP(num time) sirve para recibir un mensaje del PLC mediante TCP. Puede ser especificado
401 ! un tiempo limite de espera, tras el cual se lanza una excepcion que es recogida en la seccion de
402 ! ERROR. Devuelve true cuando el mensaje recibido es valido y false si no se ha recibido ningun
403 ! mensaje antes de que acabe el tiempo de espera. Si el socket se desconecta intenta reconectarse
404 ! y vuelve a leer el TCP.
405 !
406 FUNC bool LeerTCP(num time)
407     ! Comprobacion rapida por si hay un mensaje para detener el programa.
408     SocketReceive clientSocket, \Str:=inputData, \Time:=time;
409     RETURN TRUE;
410     ERROR
411     IF errno=ERR_SOCK_TIMEOUT THEN
412         RETURN FALSE;
413     ELSEIF errno=ERR_SOCK_CLOSED THEN
414         Reconectar;
415         RETRY;
416     ELSE
417         ! Error desconocido, el sistema se apagara.
418         Stop;
419     ENDIF
420 ENDFUNC
421
422 ! ****
423 ! EscribirTCP(string data) escribe por TCP mensajes hacia el PC. Si el socket se desconectara
424 ! trata de reconectarse y vuelve a lanzar el mensaje.
425 !
426 PROC EscribirTCP(string data)
427     SocketSend clientSocket \Str:=data;
428     ERROR
429     IF errno=ERR_SOCK_CLOSED THEN
430         Reconectar;
431         RETRY;

```

```

432    ELSE
433        ! Error desconocido, el sistema se apagara.
434        Stop;
435    ENDIF
436 ENDPROC
437
438 ! ****
439 ! ComprobarAsincronas(num inputID) procesa aquellos mensajes que no estan relacionados con el
440 ! funcionamiento normal del sistema; por ejemplo, las instrucciones enviadas por operarios desde
441 ! el HMI.
442 ! ****
443 PROC ComprobarAsincronas(num inputID)
444     VAR bool msgOK;
445
446     ! Estado del sistema. Mandar simplemente un ack.
447     IF inputID = 2 THEN
448         EscribirTCP "0ACK2";
449         ! Parametros de configuracion.
450     ELSEIF inputID = 5 THEN
451         ! Comprobar la longitud de la cadena.
452         IF StrLen(inputData) < 6 THEN
453             ErrorRX("LEN");
454             RETURN;
455         ENDIF
456
457         ! Comprobar que la ID es la de Parametros.
458         msgOK := StrToInt(StrPart(inputData,1,1), inputID);
459         IF inputID <> 5 THEN
460             RETURN;
461         ENDIF
462
463         ! Comprobar que el numero de materiales por bandeja no supera el maximo.
464         numeroMateriales := StrToInt(StrPart(inputData,2,2)\Hex);
465         IF numeroMateriales > maximoNumMateriales THEN
466             ErrorRX("NUM_MAT");
467             RETURN;
468         ENDIF
469
470         ! Comprobar que el numero de mesas no supera el maximo.
471         numeroMesas := StrToInt(StrPart(inputData,5,2)\Hex);
472         IF numeroMesas > maximoNumMesas THEN
473             ErrorRX("NUM_MES");
474             RETURN;
475         ENDIF

```

```

476
477     EscribirTCP "0ACK5";
478     servidorConectado := TRUE;
479
480     ! Ir a HOME
481     ELSEIF inputID = 6 THEN
482         ! Enviar el ACK.
483         EscribirTCP "0ACK6";
484
485     MoveJ Targ_Home,v500,fine,GarraVentosas\WObj:=wobj0;
486
487     ! Apagar la maquina
488     ELSEIF inputID = 7 THEN
489         ! Enviar el ACK.
490         EscribirTCP "0ACK7";
491         ! Ir a home y apagar.
492         MoveJ Targ_Home,v500,fine,GarraVentosas\WObj:=wobj0;
493         Stop;
494     ENDIF
495 ENDPROC
496
497 !*****
498 ! num ConvertirStringMaterial(string strMat) recibe un string con el identificador del material y
499 ! lo convierte a un numero que reconoce el robot. En caso de haber algun error, devuelve -1.
500 !*****
501 FUNC num ConvertirStringMaterial(string strMat)
502     VAR num numMatEntrada;
503     VAR num numMat := 0;
504     numMatEntrada := BitLsh(StrToByte(StrPart(strMat,1,2)\Hex),8) +
505                     StrToByte(StrPart(strMat,3,2)\Hex);
506     WHILE BitAnd(numMatEntrada, 1) = 0 DO
507         numMatEntrada := BitRsh(numMatEntrada, 1);
508         numMat := numMat + 1;
509     ENDWHILE
510
511     IF BitRsh(numMatEntrada, 1) <> 0 THEN
512         RETURN -1;
513     ENDIF
514     RETURN numMat;
515 ENDFUNC
516
517 !*****
518 ! Funcion no utilizada. Sirve para poder transferir puntos desde la ventanda de posicion inicial
519 ! a RAPID.

```

520	! ****
521	PROC Path_10()
522	MoveL Targ_CajaE,v500,z20,GarraVentosas\WObj:=wobj0;
523	MoveL Targ_CajaF,v500,z20,GarraVentosas\WObj:=wobj0;
524	MoveL Targ_CintaEmpaq,v500,z20,GarraVentosas\WObj:=wobj0;
525	MoveL Targ_CintaLlegada,v500,z20,GarraVentosas\WObj:=wobj0;
526	MoveL Targ_Home,v500,z20,GarraVentosas\WObj:=wobj0;
527	MoveL Targ_Mesa,v500,z20,GarraVentosas\WObj:=wobj0;
528	ENDPROC
529	
530	ENDMODULE

(Final del Anexo 2)

### 3. Anexo 3: Scripts VBA

En este anexo se muestran los scripts utilizados en el HMI.

#### 3.1. FuncBotonMesa

Este script se ejecuta al pulsar alguno de los botones de la mesa. Sirve para abrir la ventana de los materiales y para indicar al PLC cuál de todas las mesas se ha seleccionado. De no utilizarse, habría que asignar una acción a cada uno de los botones que fije el valor de la variable “NumeroMesaSeleccionada”.

Primero, lo que hace es cambiar el valor de “MesaSeleccionada”. Esto hace que se muestre la ventana de las piezas.

Después, se obtiene el número correspondiente a la mesa según su nombre. Todos los botones tienen de nombre “BotonMesa\_x” donde x es el número de la mesa que representan.

Una vez se tiene el número de mesa, se pasa a la variable “NumeroMesaSeleccionada”. Esta variable sirve para que el PLC pueda seleccionar qué piezas mostrar en la ventana de la caja.

1	Sub FuncBotonMesa()
2	
3	Dim nombreBoton, numMesa, valorMesa
4	
5	SmartTags("MesaSeleccionada") = Not SmartTags("MesaSeleccionada")
6	nombreBoton = HmiRuntime.ActiveScreen.ActiveScreenItem.ObjectName
7	numMesa = Split(nombreBoton, "_")(1)
8	SmartTags("NumeroMesaSeleccionada") = numMesa
9	
10	End Sub

#### 3.2. IniciarBotonesMesas

Este script es necesario para poder mostrar/ocultar de manera automática los botones de las mesas según el valor máximo de mesas configurado.

El script se llama cada vez que se carga alguna pantalla.

Primero, obtiene el número de mesas configuradas leyendo la variable “vbes\_NumeroMesas”.

Después, realiza un bucle con índice i desde 0 a numMesas donde consigue la referencia del botón “BotonMesa\_i” y la hace visible.

Finalmente, realiza el proceso contrario con los botones restantes, ocultándolos.

1	Sub IniciarBotonesMesas()
2	
3	Dim numMesas, i
4	
5	numMesas = SmartTags("vbes_NumeroMesas")
6	For i = 0 To numMesas
7	HmiRuntime.ActiveScreen.ScreenItems("BotonMesa_" & i).Visible = True
8	Next
9	For i = numMesas To 12
10	HmiRuntime.ActiveScreen.ScreenItems("BotonMesa_" & i).Visible = False
11	Next
12	
13	SmartTags("MesaSeleccionada") = False
14	SmartTags("NumeroMesaSeleccionada") = 0
15	
16	End Sub

**(Final del Anexo 3)**

#### 4. Anexo 4: Túnel de conexión PLC-Robot

TCPTunel.py es un programa extra desarrollado para este proyecto. Sirve para recibir paquetes de una interfaz red y mandarlos a otra. Su uso es necesario puesto que no se permite abrir el socket del servidor del robot en la red que crea S7-PLCSIM Advanced v4.0.

Además, otra utilidad del túnel es que permite observar en tiempo real los mensajes transmitidos entre las interfaces, lo cual ha facilitado mucho el desarrollo de la planta.

Para ejecutarlo, hace falta tener Python instalado y llamarlo mediante:

```
$ python TCPTunel.py
```

Código:

```
1 #####  
2 # TCPTunel es un nexo de conexión entre las interfaces de red del  
3 # PLC y la del robot. Es necesario puesto que el PLC no permite  
4 # crear una IP nueva dentro de la red (que correspondería al  
5 # robot).  
6 #####  
7  
8 #####  
9 # LIBRERIAS  
10 #####  
11 import socket  
12 import threading  
13 import time  
14  
15 #####  
16 # VARIABLES GLOBALES Y CONSTANTES  
17 #####  
18 stopProgram = False  
19  
20 # IP y puerto de TIA Portal  
21 tiaHost = '192.168.0.130'  
22 tiaPort = 2000  
23 # IP y puerto de RobotStudio  
24 abbHost = "127.0.0.1" # localhost  
25 abbPort = 9876  
26  
27 #####  
28 # FUNCIONES
```

```

29 ######
30
31 # Hilo de conexión TIA --> ABB
32 def tiaToABB(tia_conn, tia_addr, abbSocket):
33     global stopProgram
34
35     while not stopProgram:
36         try:
37             tia_conn.settimeout(1.0)
38             recv = tia_conn.recv(1024)
39             if recv:
40                 print(f"TIA => ABB ({len(recv)}): {recv}")
41                 abbSocket.send(recv)
42         except socket.timeout:
43             pass
44         except ConnectionResetError:
45             print("TIA disconnected!")
46             stopProgram = True
47
48 # Hilo de conexión ABB --> TIA
49 def abbToTia(tia_conn, tia_addr, abbSocket):
50     global stopProgram
51
52     while not stopProgram:
53         try:
54             abbSocket.settimeout(1.0)
55             recv = abbSocket.recv(1024)
56             if recv:
57                 print(f"ABB => TIA ({len(recv)}): {recv}")
58                 tia_conn.sendall(recv)
59         except socket.timeout:
60             pass
61         except ConnectionResetError:
62             print("ABB disconnected!")
63             stopProgram = True
64
65 # Main
66 try:
67     tiaSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
68     tiaSocket.bind((tiaHost, tiaPort))
69     print(f"TIA PORTAL conectado en {tiaHost}:{tiaPort}")
70     tiaSocket.listen()

```

```

71     tia_conn, tia_addr = tiaSocket.accept()
72
73     abbSocket = socket.socket()
74     abbSocket.connect((abbHost, abbPort))
75     print(f"ROBOT STUDIO conectado en {abbHost}:{abbPort}")
76
77     tiaToABBThread = threading.Thread(target=tiaToABB, args=(tia_conn, tia_addr,
78                                         abbSocket))
78
79     abbToTiaThread = threading.Thread(target=abbToTia, args=(tia_conn, tia_addr,
80                                         abbSocket))
80
81     tiaToABBThread.start()
82     abbToTiaThread.start()
83
84     while tiaToABBThread.is_alive() and abbToTiaThread.is_alive():
85         time.sleep(0.1)
86
86 except KeyboardInterrupt:
87     stopProgram = True

```

**(Final del Anexo 4)**

## 5. Anexo 5: Presentación

A continuación, se adjunta las diapositivas utilizadas durante la presentación.



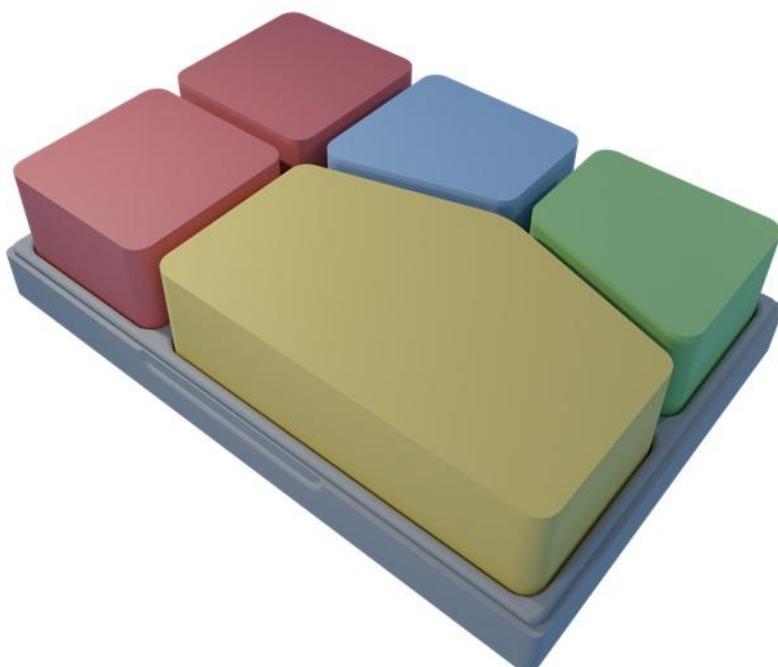
# Caja Bento



[www.seas.es](http://www.seas.es)

---

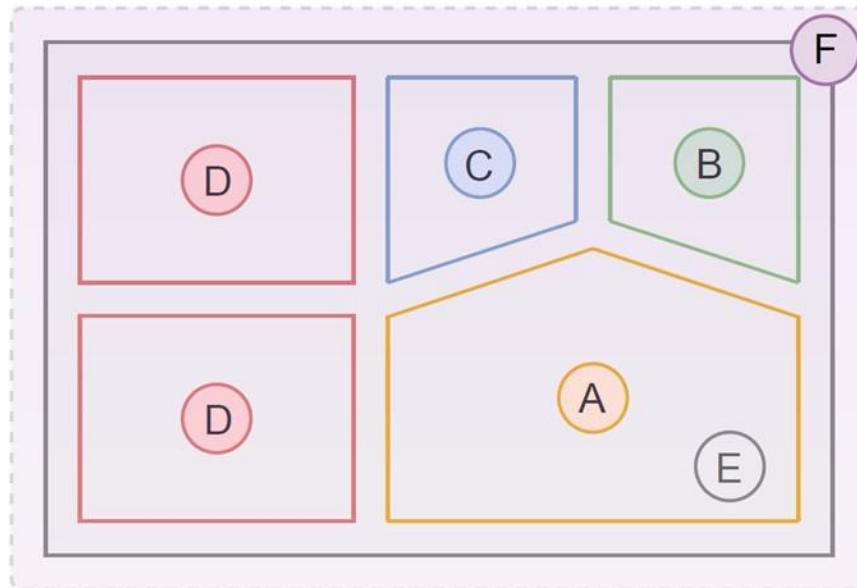
# Caja Bento



[www.seas.es](http://www.seas.es)

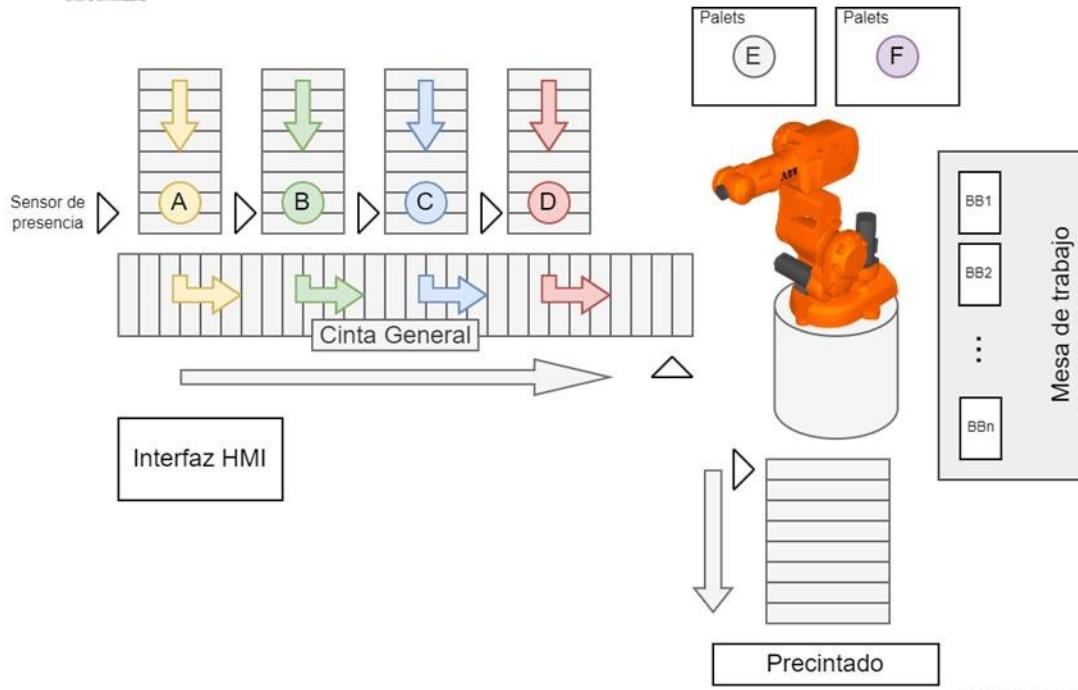
---

## Diagrama de la caja



[www.seas.es](http://www.seas.es)

## Planta



[www.seas.es](http://www.seas.es)

## PLC

### CPU 1515-2 PN



• **Procesador y Memoria:** CPU de doble núcleo, 1 MB de memoria de trabajo para programas y 5 MB para datos.

• **Comunicación:** Interfaces integradas PROFINET IO IRT con 2 puertos y soporte para PROFINET CBA y OPC UA.

• **Escalabilidad y Flexibilidad:** Diseño modular que permite una fácil expansión.

• **Diagnósticos y Seguridad.**

Capacidades de diagnóstico extensas, características de seguridad integradas

• **Ingeniería y Entorno:** Programado mediante TIA Portal.

[www.seas.es](http://www.seas.es)

## PLC

### Complementos



PS 25W 24VDC

DQ 16x24VDC/0.5A BA



DI 32x24VDC BA

[www.seas.es](http://www.seas.es)



# PLC

## Disposición de los componentes



[www.seas.es](http://www.seas.es)

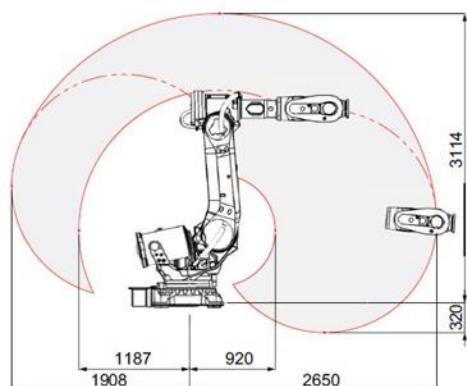


# Robot

IRB 6700-235/2.65



- Alcance: 2.65 m.
- Carga útil: 235 kg.
- Torque en la muñeca: 1324 Nm.
- Peso: 1205 kg.
- Rango de vel. máx. rot: 90 ~190 °/s.



[www.seas.es](http://www.seas.es)

# Robot

## IRC5



[www.seas.es](http://www.seas.es)

# Robot

## IRC5



[www.seas.es](http://www.seas.es)



## HMI

HMI TP1200 Comfort

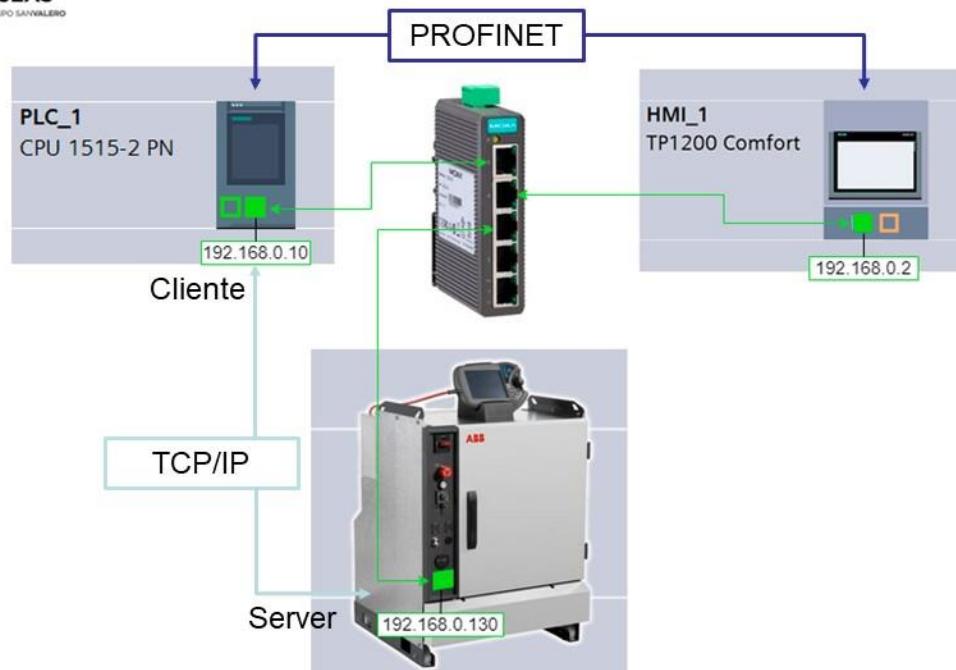


- Pantalla táctil widescreen de 12 pulgadas.
- Procesador x86.
- Memoria de usuario de 12 MB.
- Interfaz PROFINET.
- Grado IP65.

[www.seas.es](http://www.seas.es)



## Conexión entre dispositivos



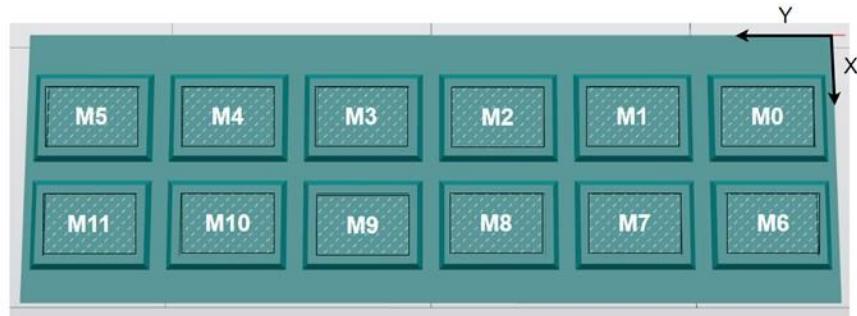
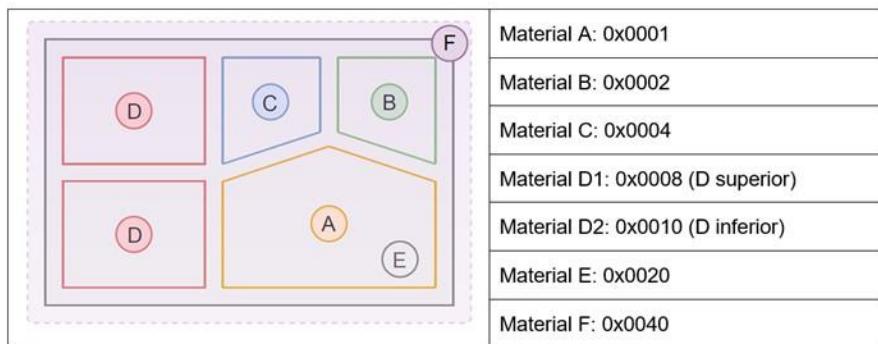
[www.seas.es](http://www.seas.es)

# Mensajes PLC → Robot

MENSAJE	0 (ID)	1	2	3	4	5	6	7
ACK	0	A	C	K	ID	-	-	-
ERROR	1	?		MENSAJE ERROR				
ESTADO	2	S	T	A	-	-	-	-
COLOCA MATERIAL	3			Material (0x0000 a 0x8000)		M	Mesa (0x00 a 0xF)	
COLOCA CAJA	4	M						
PARAMETROS	5	Nº Materiales		M	Nº Mesas	-	-	
HOME	6	-	-	-	-	-	-	-
APAGAR	7	-	-	-	-	-	-	-

[www.seas.es](http://www.seas.es)

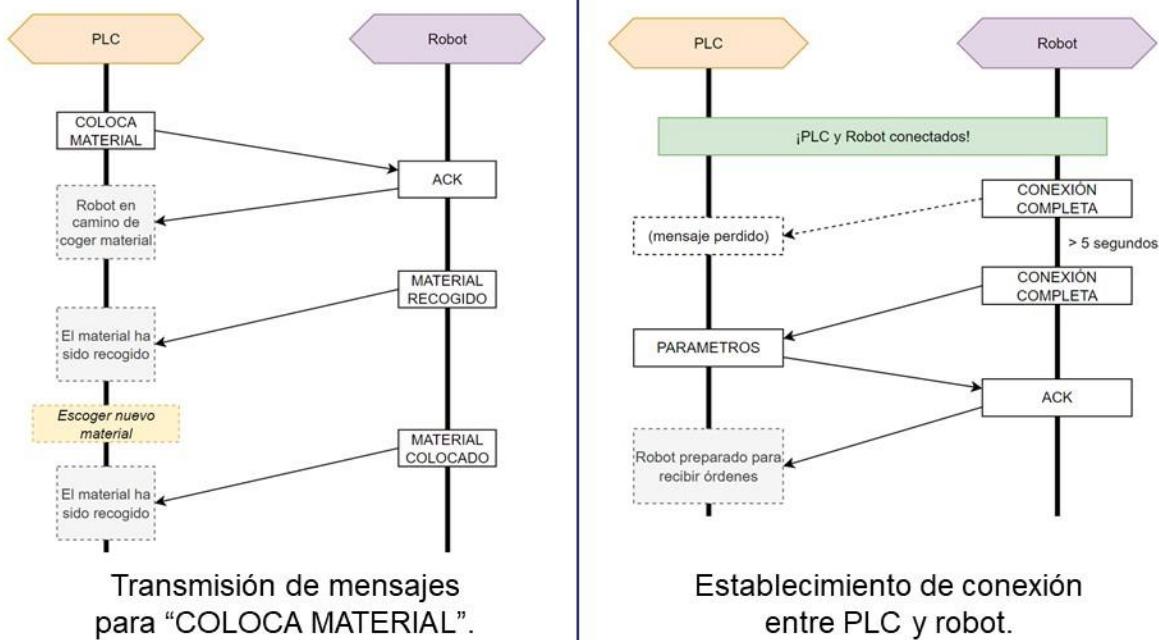
## Materiales y Mesas



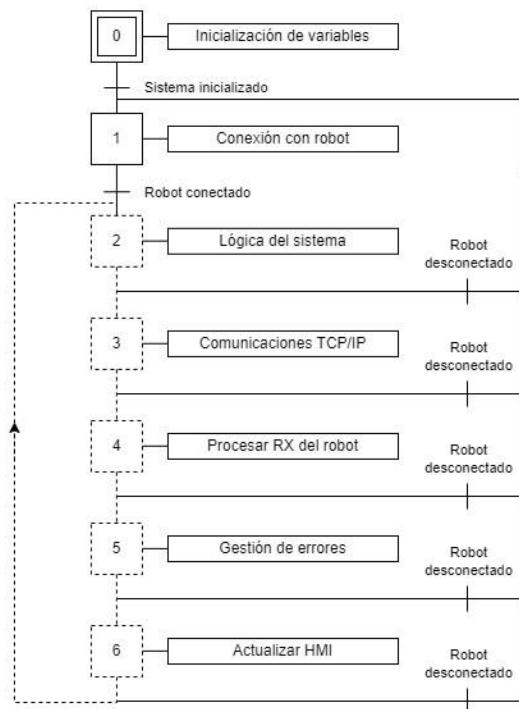
# Mensajes Robot → PLC

MENSAJE	0 (ID)	1	2	3	4	5	6	7
<b>ACK</b>	0	A	C	K	ID	-	-	-
<b>ERROR</b>	1	?	MENSAJE ERROR					
<b>MATERIAL RECOGIDO</b>	2	M	A	T	-	R	E	C
<b>MATERIAL COLOCADO</b>	3	M	A	T	-	C	O	L
<b>CONEXIÓN COMPLETA</b>	4	Material (0x0000 a 0x8000)				M	Mesa (0x00 a 0x0F)	
<b>PARÁMETROS</b>	5	Material (0x0000 a 0x8000)				M	Mesa (0x00 a 0x0F)	

## Ejemplo de mensajes

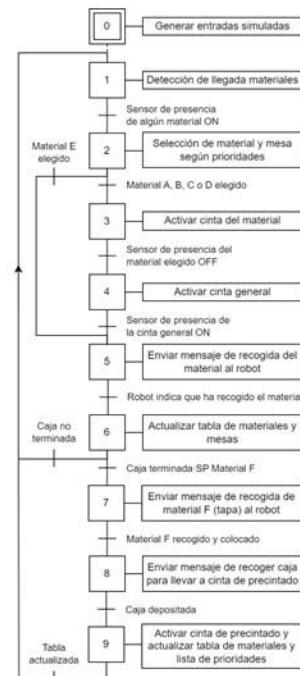


# Funcionamiento del PLC



[www.seas.es](http://www.seas.es)

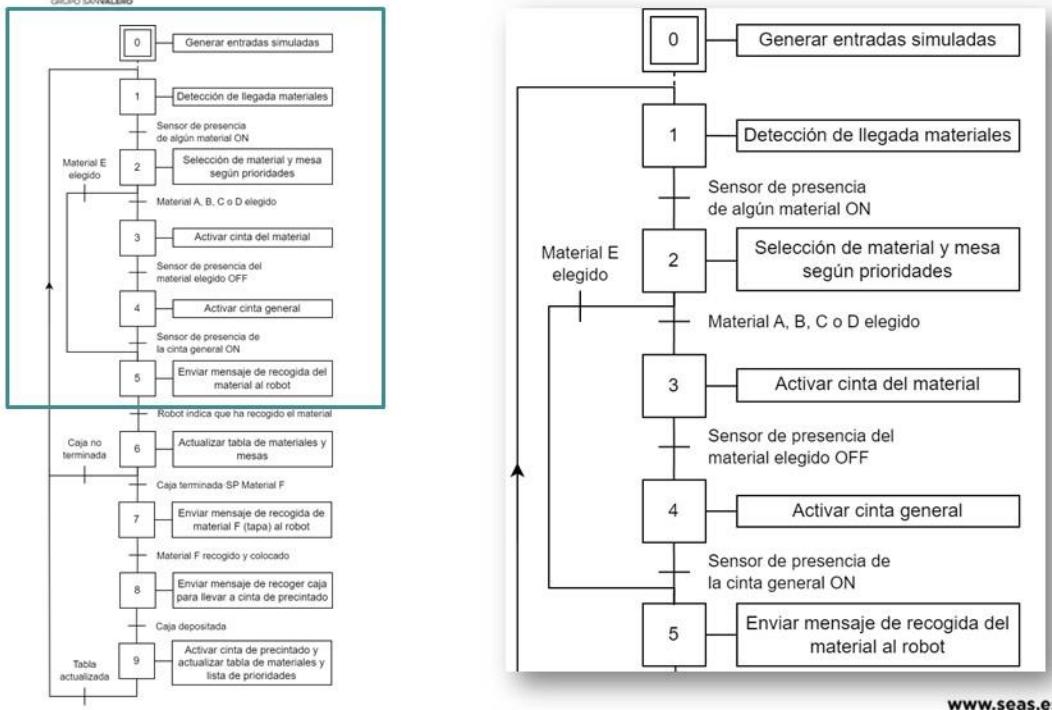
# Lógica de la planta (I)



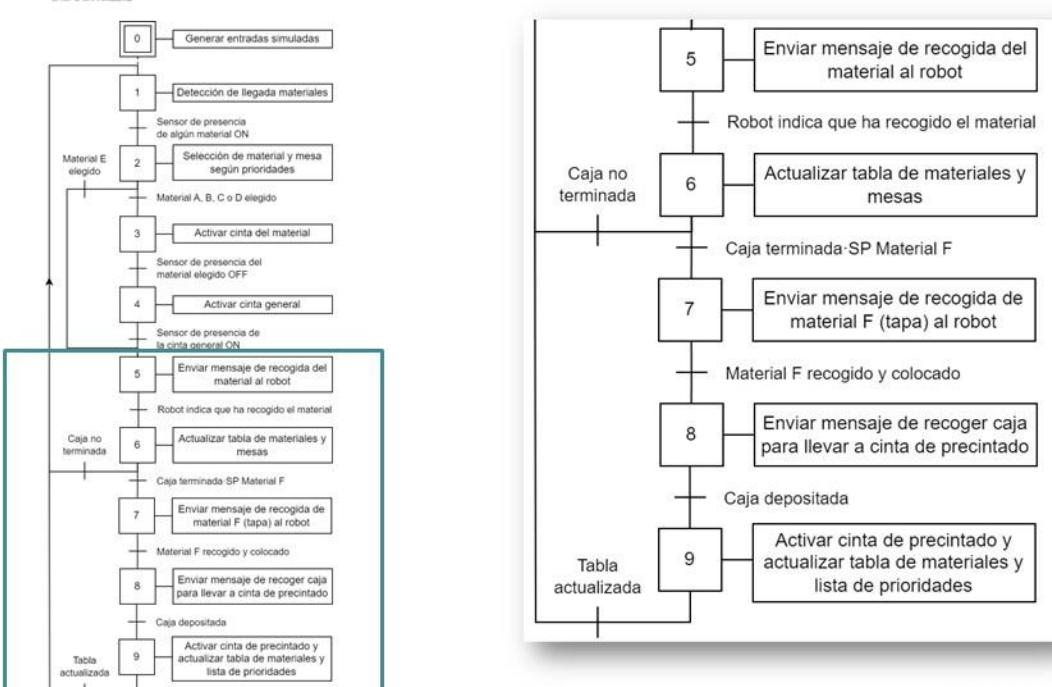
[www.seas.es](http://www.seas.es)



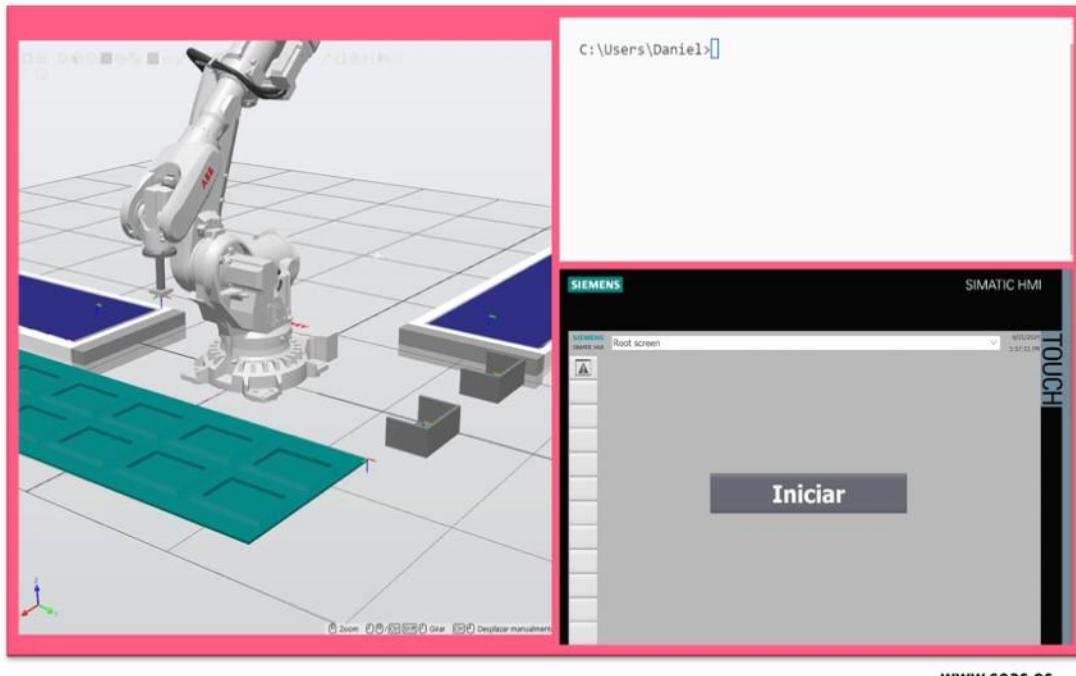
## Lógica de la planta (II)



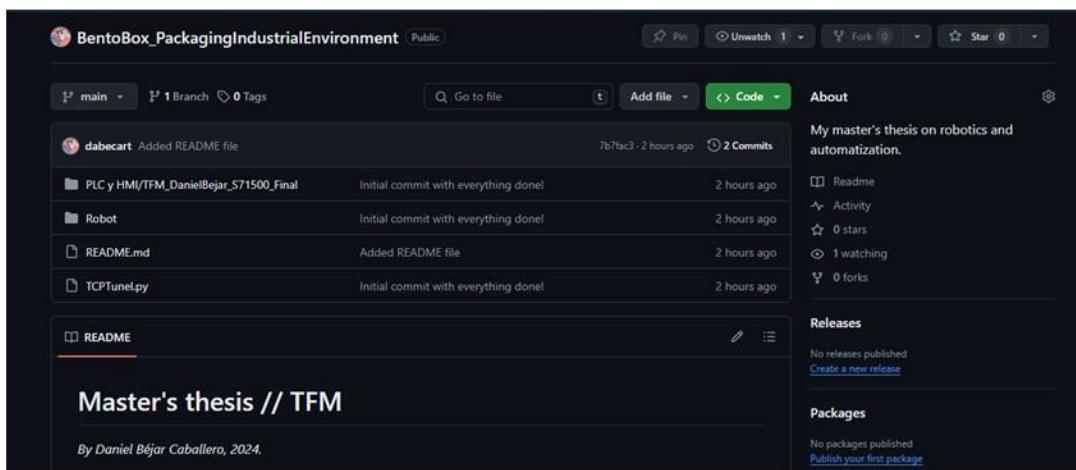
## Lógica de la planta (III)



# Simulación



Todo el contenido usado para el TFM en

[https://github.com/dabecart/BentoBox\\_PackagingIndustrialEnvironment](https://github.com/dabecart/BentoBox_PackagingIndustrialEnvironment)



**www.seas.es**



SEAS, Estudios Superiores Abiertos  
Vicente Ferrer, 9. Edificio SEAS  
50015 Zaragoza  
T. 976 700 660 • F. 976 228 793

**(Final del Anexo 5)**

## **FINAL DEL TFM**

“Planta empaquetadora de Cajas Bentō mediante brazo robótico, PLC, interfaz HMI y comunicación TCP/IP”

Escrito y realizado por Daniel Béjar Caballero.

Colmenar Viejo, 21 de junio de 2024.