



**Instituto Tecnológico Autónomo de México**

Departamento de Ingeniería Industrial y Operaciones

Laboratorio de Automatización y Control de Procesos

*“Proyecto Final”*

**Alumnos:**

Carlos Ignacio Alonso Reyes

Federico Cortes Velhagen

Andrés Saucedo Toledo

Fecha

## 1. Introducción

Para este proyecto se automatizó un proceso que ensambla piezas “base” con su respectiva pieza “tapa” a modo de acoplarse en una pieza “única”. Existen 2 colores diferentes: verde y azul. El sistema solo hace el ensamble si el color de la base coincide con el de la tapa. Se tiene una banda que suministra las bases y otra que suministra las tapas. El objetivo fue que se tuvieran 3 bandas de salida, una con ensambles en azul, la otra en verde y una última banda con las tapas que van sobrando. El acomodo de las bandas de salida y el método para desviar las piezas por dichas bandas fue libre.

Se utilizó el botón de inicio, de pausa y de stop de la caja de control. El botón de inicio debería iniciar el proceso o continuarlo después de una pausa. El botón de pausa debería detener todo el proceso sin borrar variables del proceso de tal manera que al presionar el botón verde se continúe con la ejecución. El botón rojo cumpliría con parar todo el proceso y borrar los valores de las variables del proceso. En la caja de control debía haber 2 displays digitales mostrando el número de ensambles azules completos y ensambles verdes completos. El robot se usaría en su configuración analógica, tal como se encuentra ya configurado y el código debía estar desarrollado en lenguaje escalera.

Al iniciar el proceso se encienden automáticamente los lids y bases emitters. Luego al pulsar el botón de inicio se activan las bandas de las tapas y las bases, a la vez también se activan los stop blade que se encuentran al final de ambas bandas. De esta manera se detienen las tapas y las bases después de pasar por los vision sensor para comparar sus valores. Mientras ambas tapas se encuentran en los stop blade, las bandas se apagan para evitar acumulación de piezas. Si el color de la tapa es distinto al de la base, la banda de las tapas estará activa, el stop blade se desactivará y una vez que la pieza haya pasado a través del stop blade, este se volverá a activar a la vez que enciende la banda de las sobras, la cual se apagará luego de que la pieza haya pasado completamente a través de ella.

Si las piezas son del mismo color, entonces el grab tomará la tapa y la colocará sobre la base para ensamblar las piezas, las soltará y regresará a una posición media. Luego se determina el color del resultado, si el ensamble es azul, el stop blade de la banda de las bases se desactiva y la banda de las bases se enciende simultáneamente solo mientras el ensamble pasa a través del stop blade, una vez ocurrido esto, stop blade se activa y la banda bases se apaga y la banda de los ensambles azules se enciende, para que al final de esta última banda

sea contabilizado por un diffuse sensor y registrado en un display de la caja de control. Una vez que la pieza ha sido contada, la banda de los ensambles azules se desactiva y las de las tapas y de las bases se activan para continuar el proceso.

Si el ensamble es verde, el grab volverá a bajar para tomar la pieza y colocarla en la banda de los ensambles verdes. En cuanto la pieza se encuentre en la banda, el grab regresará a una posición media y las 3 bandas que forman la banda de los ensambles verdes se activarán para que llegue al final y se contabilizada para finalmente parar estas bandas y activar las de las tapas y bases.

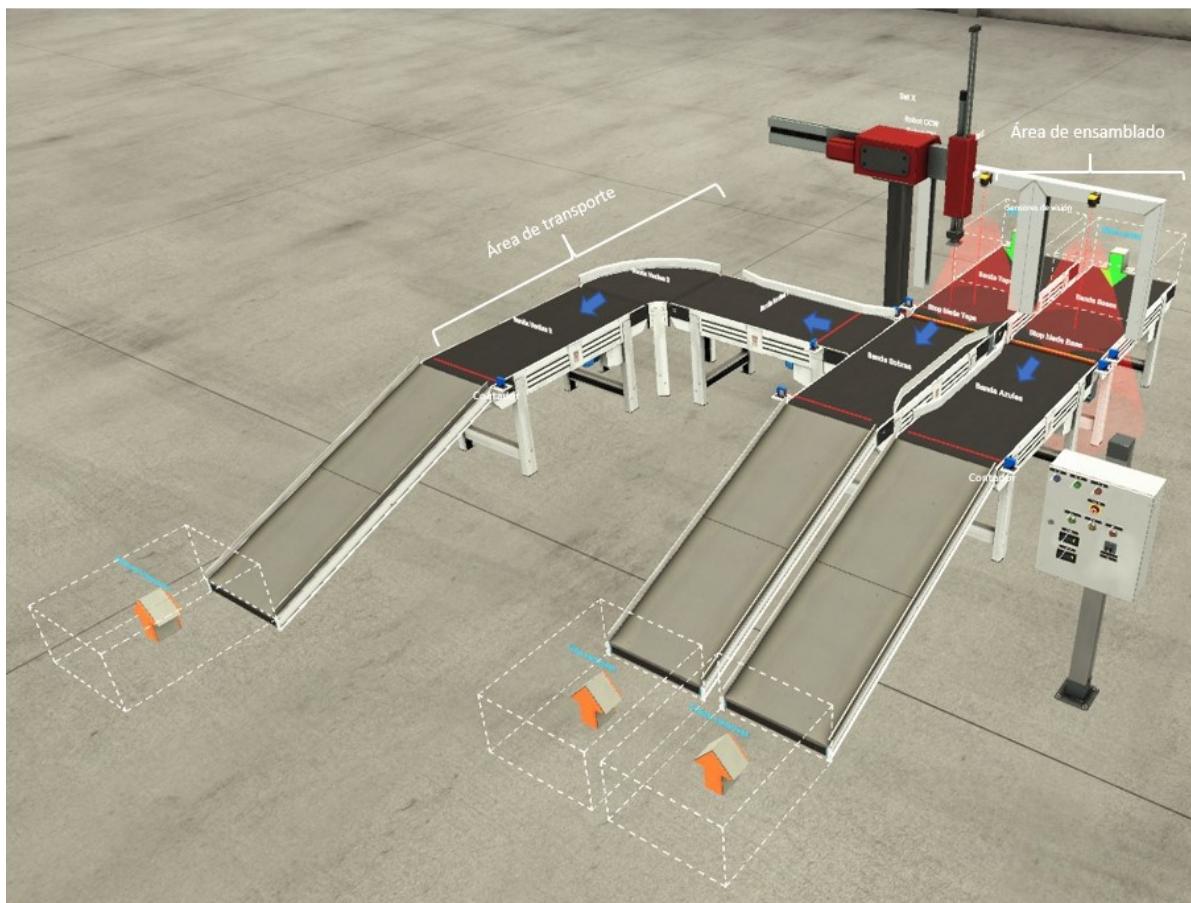


Figura 1. Diagram de proceso.

## 2. Desarrollo.

Para el desarrollo de este proyecto iniciamos con un diagrama GRAFCET para definir los estados y condiciones necesarias para el proyecto. Pero para poder definir con mayor exactitud los elementos necesarios, primero tenemos que armar nuestra celda en Factory IO. Como se puede ver la Figura 1, nuestro escenario cuenta con 3 bandas, un brazo robótico que tiene movimientos en X,Z y rotación sobre su propio eje. También se cuenta con 8 sensores de posición y 2 sensores identificadores. Los sensores identificadores traducen el tipo y el color de la pieza a valores enteros, o INT, que pueden ser usados mientras la pieza se encuentre bajo estos sensores. Se cuenta con dos topes a la mitad del grupo de bandas “Tapas” y “Bases”, esto con el motivo de posicionar dichas piezas al alcance del brazo robótico. De igual forma contamos con un panel donde se encuentran los botones de “Start”, “Reset” y “Stop”, además que se cuenta con 2 display que muestra la cantidad de piezas Azules y Verdes armadas. Finalmente se encuentran 3 focos que indican si el par de piezas forman un pieza Azul, Verde o no congenian.

Una vez establecida la celda o el escenario nos pusimos a jugar con ellas para obtener valores importantes, como las coordenadas para el robot y los valores de identificación de cada una de las piezas, todos los valores los puede encontrar en la Tabla 1. Estos valores serán usados constantemente a la hora de implementar de usar el código escalera de CCW.

Nombre	Tipo	Valor
Coordenadas en X para recoger Tapa	REAL (FLOAT)	1.1
Coordenadas en Z para recoger Tapa	REAL (FLOAT)	9.4
Coordenadas en X para dejar Tapa	REAL (FLOAT)	10
Coordenadas en Z para dejar Tapa	REAL (FLOAT)	9.5
Coordenadas en X para recoger pieza Verde	REAL (FLOAT)	3.5
Coordenadas en X para dejar pieza Verde	REAL (FLOAT)	8
Identificación de Tapa Azul	INT (INTEGER)	2
Identificación de Tapa Verde	INT (INTEGER)	5

Identificación de Base Azul	INT (INTEGER)	3
Identificación de Base Verde	INT (INTEGER)	6

Tabla 1. Valores interesantes.

Ya conociendo los datos importantes y experimentando con el escenario, nos podemos dar una mejor idea de la secuencia o el proceso que debe seguir nuestro programa, y para hacerlo más visual nos dimos a la tarea de hacer un diagrama Grafset con todos los estados y condiciones necesarios para realizar el proyecto.

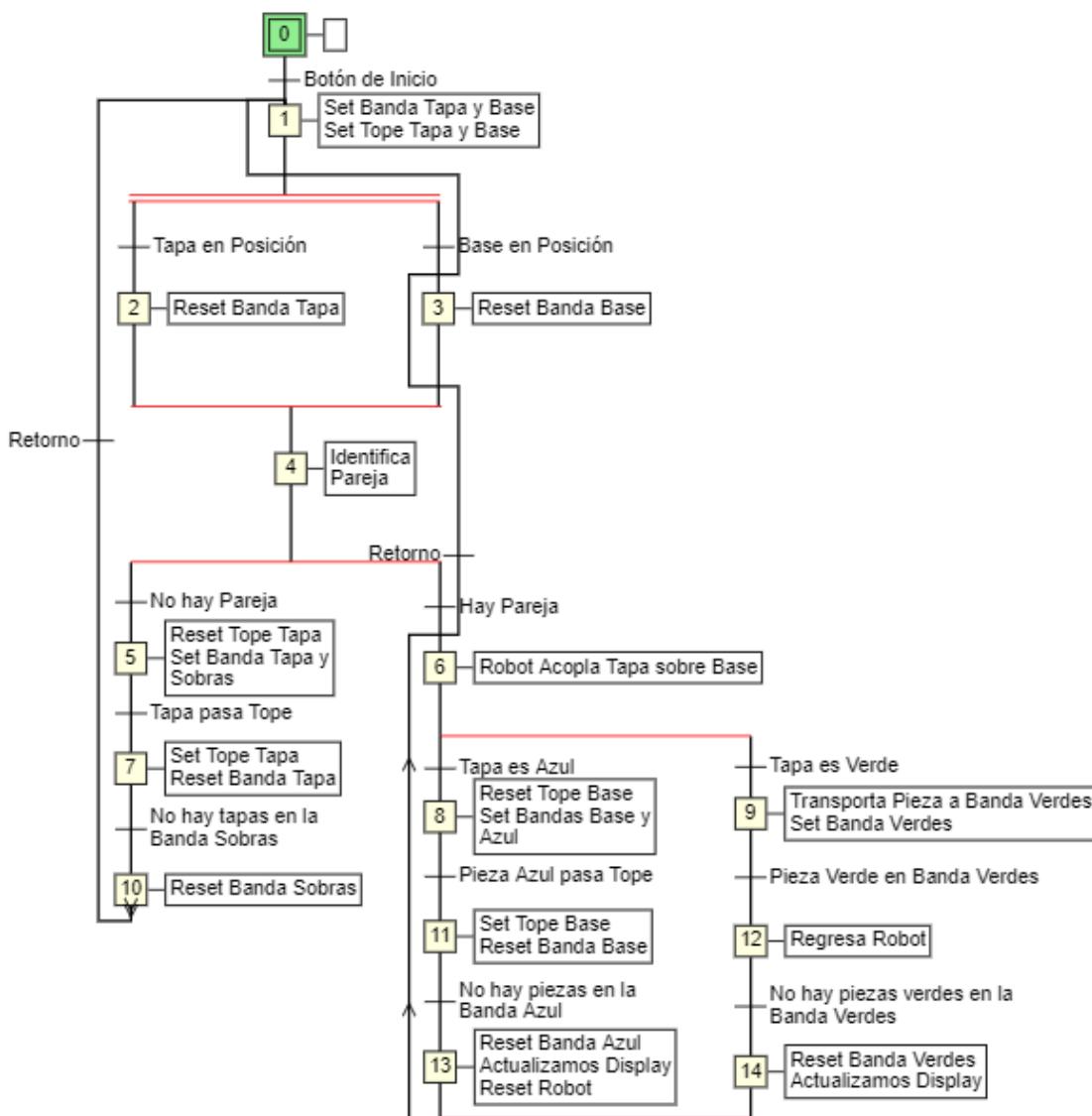


Diagrama 1.Grafset del Modelo

Como se puede ver en el diagrama, nuestro primer estado (E0) solo es posible al iniciar el programa, cuando aún no se le da ninguna instrucción y no se han modificado ninguna variable. Una vez presionado el botón de inicio se activa el estado 1, el cual activa las bandas de tapas y bases, estas son las bandas bajo los “emitters”, y los topes de cada banda. Estas estarán activas hasta que cada uno de sus sensores de posición estén activos, posición tapa y posición base, lo que provocará la desactivación de dicha banda, es decir, el cambio a los estados 2 y 3. Después de esto nos preguntamos si es o no una pareja, en caso de que no lo sea se desecha la tapa actual y esperamos a que la siguiente tapa sea del mismo color que la base. Si resulta que son del mismo color, acoplamos ambas piezas, este caso es interesante en el código escalera que después explicaremos. Una vez acoplada la pieza preguntamos de qué color es la tapa. Si la pieza es azul hacemos una rutina parecida a cuando la tapa no es del mismo color. Si la pieza es verde el brazo robótico transportará la pieza a la banda correspondiente y la dejará, pero antes de volver al estado 1, regresaremos el brazo robótico a su posición original.

La traducción de Grafset a código escalera CCW es casi la misma, con la excepción de algunos casos interesantes. El código completo queda adjuntado en el Anexo, por favor consultar si es necesario. Como decíamos anteriormente, el movimiento del brazo robótico es un caso interesante en el código, pues para no atascar una línea de código con instrucciones, se optó en dividir el movimiento en sub-estados, en el caso del estado 6 donde se acoplan las piezas, lo dividimos en 4 sub-estados o movimientos: movimiento a tapa, movimiento intermedio, movimiento a base y movimiento a espera. De igual forma, en los movimientos del robot se metieron timers para asegurar que el robot está en la posición deseada antes de pasar al siguiente sub estado, consultar las líneas 16,17,18,19 del código de escalera en el Anexo. Todos los movimientos del brazo robótico están divididos en sub-estados para su mejor manipulación y visualización. En el caso de transportar la pieza Verde a la Banda Verde se dividió en: recoger pieza, movimiento intermedio y rotación, dejar pieza, y regreso a posición original, consulte líneas 29 a 32 del código escalera en el Anexo. Nuevamente se agregan timers para asegurar que el movimiento se ha realizado.

Otra parte interesante del código escalera que no se puede ver en el Grafset son los procedimientos de “Pausa” y de “Stop”. Nuestra forma de abordar estos requisitos fue a través de agregar una condición extra a cada estado. Para poder activar un estado tiene que estar activa una condición de “Pausa”. Si “Pausa” se encuentra desactivada, esto solo pasa si presionamos el botón “Reset”, se completará el procedimiento del estado anterior, pero no se realizará el siguiente estado hasta que se active la condición “Pausa”. En el caso del “Stop”, lo que se hace es un RESET a todos los estados, así nos aseguramos que no se realizará ningún estado después de terminar el estado actual. En pocas palabras, nuestras

interrupciones se generan hasta que acabe el estado en proceso, por lo que si se tiene sub-estados no se verá el resultado de la interrupción hasta que todos los sub-estados terminen.

### 3. Resultados



Figura 2. Escenario de ensamble de tapa y base.

En el primer paso los actuadores “lids emitter” y “base emitter”, emitirán una pieza en su correspondiente banda (Figura 1). El color de las piezas es aleatorio. Seguido de la aparición de las piezas, tanto la banda “bases” como la banda “tapas” moverán las piezas a través de los “Vision Sensors”, donde cada color está asociado a una variable tipo “Int” única.



Figura 3. Ensamble de piezas Azules

Después de haber identificado el color de cada pieza, estas avanzará hasta su respectivo “stop blade”, donde si el color coincide, el robot tomará la tapa ubicada en la banda izquierda, y la ensamblará con la base que está en la banda de la derecha. Si el color de la pieza ya ensamblada es azul, esta seguirá por la banda de la derecha y será removida del mismo lado (Figura 2). Caso contrario, si el color de la pieza ensamblada es verde, el robot la tomará y la llevará a la banda 3 que se ubica de manera horizontal a las otras dos. Ahí la pieza pasará por la banda y será removida al final de la misma (Figura 3).



Figura 4. Ensamble de piezas verdes.

Ahora, si al momento de pasar por los “Vision Sensors” los colores de las piezas no son iguales, la pieza “tapa” continuará su camino por la “Banda Sobras”, pero la

pieza “base” es retenida en el “stop blade” (Figura 4). Después de desechar la “tapa” de un color distinto, el sistema genera otra tapa que repetirá el proceso, y si esta al pasar por el “Vision Sensor” es del mismo color que la “base” que está detenida en el “stop blade”, el robot la ensamblará. En el caso de que los colores no coincidan nuevamente, el “stop blade” se bajará y la “tapa” se desecha de nueva cuenta. Este proceso se repite para cada pieza (tapa + base) ensamblada.

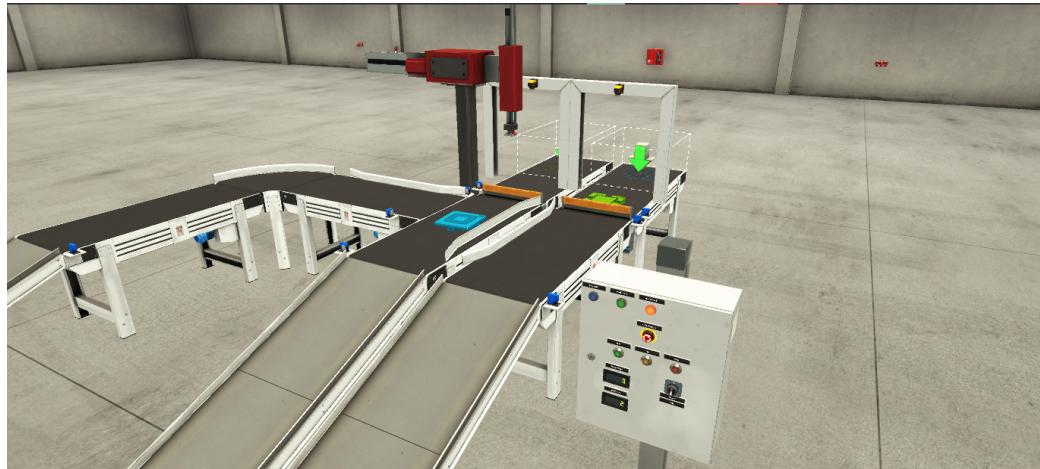


Figura 5. Caso de no tener piezas del mismo color

BOOL_IN_0	Start	BOOL	▼			
BOOL_IN_1	Reset	BOOL	▼			
BOOL_IN_2	Stop	BOOL	▼			
BOOL_IN_3	Base en Posición	BOOL	▼			
BOOL_IN_4	Tapa en Posición	BOOL	▼			
BOOL_IN_5	Sobra Entrando	BOOL	▼			
BOOL_IN_6	Sobra Saliendo	BOOL	▼			
BOOL_IN_7	Contador Azul	BOOL	▼			
BOOL_IN_8	Contador Verde	BOOL	▼			
BOOL_IN_9	Azul Salida	BOOL	▼			
BOOL_IN_10	Verde Salida	BOOL	▼			
BOOL_OUT_0	Robot CCW	BOOL	▼		0	
BOOL_OUT_1	Robot CW	BOOL	▼		0	
BOOL_OUT_2	Grab	BOOL	▼		0	
BOOL_OUT_3	Banda Base 1	BOOL	▼		0	
BOOL_OUT_4	Banda Azul	BOOL	▼		0	
BOOL_OUT_5	Banda Tapa 1	BOOL	▼		0	
BOOL_OUT_6	Banda Sobra	BOOL	▼		0	
BOOL_OUT_7	Banda Verde 1	BOOL	▼		0	
BOOL_OUT_8	Banda Verde 2	BOOL	▼		0	
BOOL_OUT_9	Banda Verde 3	BOOL	▼		0	
BOOL_OUT_10	Tope Base	BOOL	▼		0	
BOOL_OUT_11	Tope Tapa	BOOL	▼		0	
BOOL_OUT_12	Pareja Azul	BOOL	▼		0	
BOOL_OUT_13	Pareja Verde	BOOL	▼		0	
BOOL_OUT_14	Sin Pareja	BOOL	▼		0	
FLOAT_IN_0	X	REAL	▼			
FLOAT_IN_1	Z	REAL	▼			
FLOAT_OUT_0	Set X	REAL	▼			
FLOAT_OUT_1	Set Z	REAL	▼			
INT_IN_0	ID Base	INT	▼			
INT_IN_1	ID Tapa	INT	▼			
INT_OUT_0	Num Piezas Azules	DINT	▼			
INT_OUT_1	Num Piezas Verdes	DINT	▼			

Tabla de variables globales

Host: 169.254.102.102			
Start	BOOL_IN_0	BOOL_OUT_0	Robot CCW
Reset	BOOL_IN_1	BOOL_OUT_1	Robot CW
Stop	BOOL_IN_2	BOOL_OUT_2	Grab
Base en Posición	BOOL_IN_3	BOOL_OUT_3	Banda Bases
Tapa en Posición	BOOL_IN_4	BOOL_OUT_4	Banda Azules
Tapa saliendo	BOOL_IN_5	BOOL_OUT_5	Banda Tapas
Sobras Salida	BOOL_IN_6	BOOL_OUT_6	Banda Sobras
Contador Azules	BOOL_IN_7	BOOL_OUT_7	Banda Verdes 1
Contador Verde	BOOL_IN_8	BOOL_OUT_8	Banda Verdes 2
Azul Salida	BOOL_IN_9	BOOL_OUT_9	Banda Verdes 3
Verdes Salida	BOOL_IN_10	BOOL_OUT_10	Stop blade Base
X	FLOAT_IN_0	BOOL_OUT_11	Stop blade Tapa
Z	FLOAT_IN_1	BOOL_OUT_12	Pareja Azul
Vision Sensor Bases (Value)	INT_IN_0	BOOL_OUT_13	Pareja Verde
Vision Sensor Tapas (Value)	INT_IN_1	BOOL_OUT_14	No hay Pareja
		FLOAT_OUT_0	Set X
		FLOAT_OUT_1	Set Z
		INT_OUT_0	Piezas Azules
		INT_OUT_1	Piezas Verdes

Tabla de enlaces en Factory IO

Para enlazar las comunicaciones entre las dos aplicaciones, lo primero que hay que hacer es conectar la aplicación “CCW” con el PLC físico que tenemos. Después de eso descargamos el código escalera al PLC. Seguido de eso, tenemos que escribir la dirección IP del PLC en la sección de “Drivers”. Finalmente se enlazan ambas aplicaciones y el código funciona en “Factory IO”

#### 4. Conclusiones.

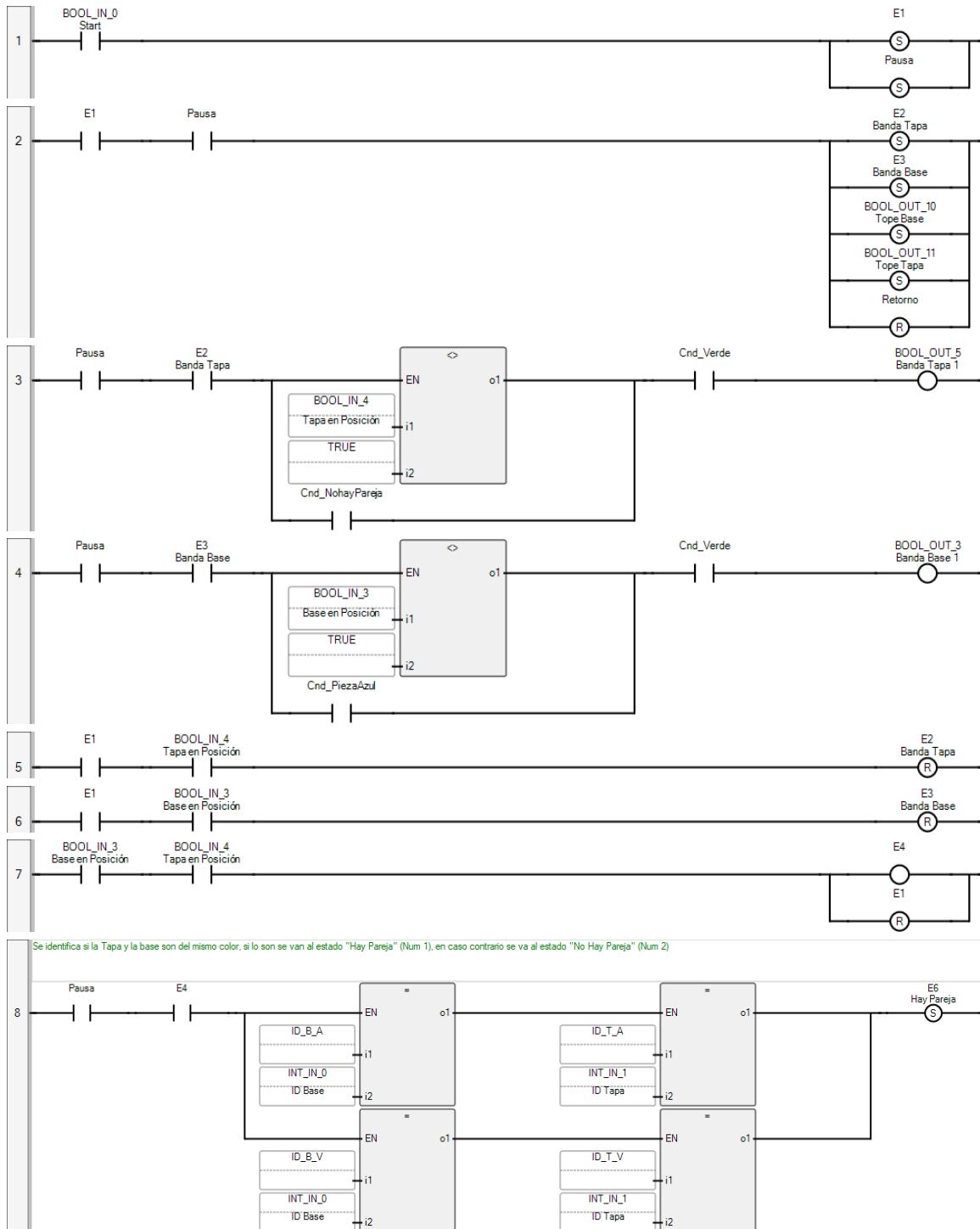
Automatizar un proceso de este tipo fue un gran reto debido a las problemáticas que nos enfrentamos desde el momento de elegir cuál sería la posición ideal de las tres bandas de acuerdo con el método de desviación de ensambles. Al optar por usar al mismo grab para cambiar la posición de los ensambles decidimos que lo mejor era tener las bandas en puntos estratégicos para que los movimientos fueran mínimos. Otro problema que se presentó fue la descoordinación de los movimientos que se le ordenaron al grab al momento de simularlos en Factory OI, esto se resolvió utilizando pequeños timers entre cada movimiento que nos garantiza que el movimiento anterior se habría ejecutado en su totalidad.

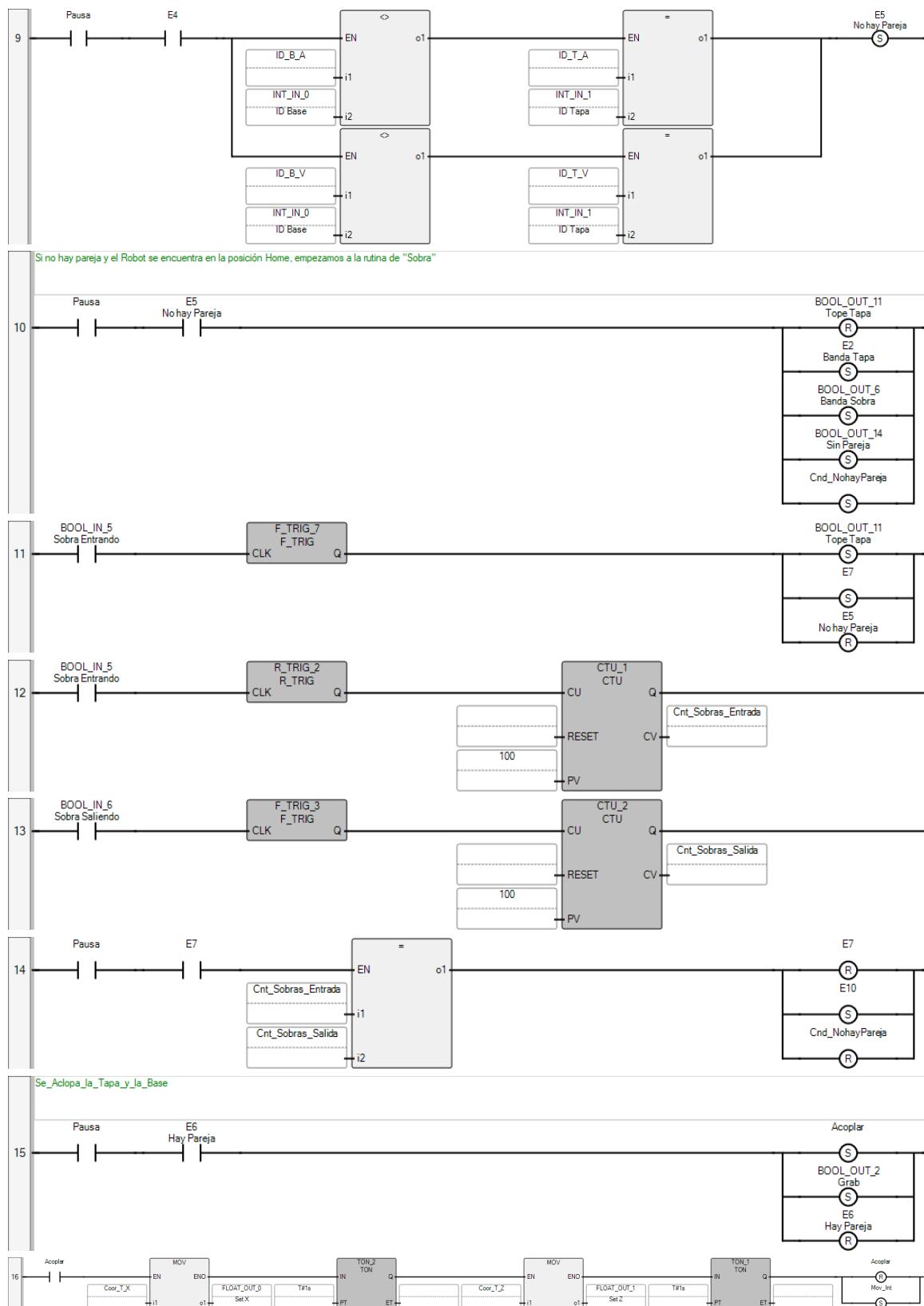
Otra forma en la que se pudo haber resuelto ese problema era mediante la confirmación de la posición antes de ejecutar el siguiente movimiento. Cabe mencionar que entre las dificultades también estuvo el determinar cuándo encenderse o apagarse ciertos componentes para que en la vida real se redujera el consumo de energía, esto fue resuelto implementando sensores en el principio y fin de las bandas, así solo se encendían mientras un objeto pasaba a través de ellas. En cuanto a las mejoras que podrían llevarse a cabo en la implementación destacan la utilidad del botón de stop para detener no solo los estados sino también los movimientos dentro de ellos y el uso de movimientos más precisos en el grab.

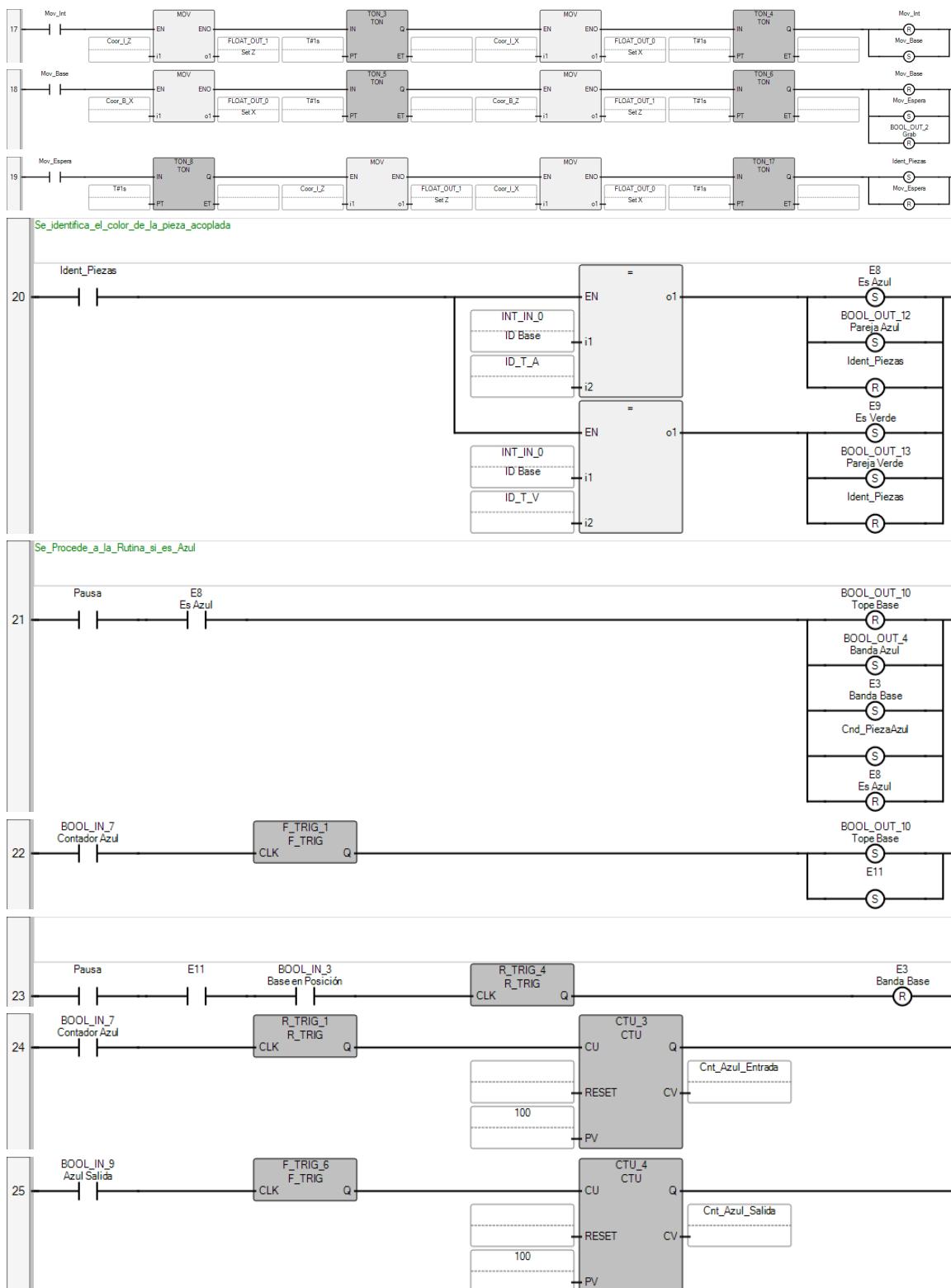
Debemos mencionar que hay problemas que no se tomaron en cuenta para la solución de este proceso automatizado debido a los requerimientos solicitados como el desperdicio de tapas, el largo de las bandas, el tiempo de ensamblado e incluso el tipo de sensores, ya que todos estos elementos serían bastante cruciales a la hora de implementar un proceso de manera eficiente en la vida real, como en líneas de ensamblaje de cajas, de juguetes, de autos, de muebles; de empaquetamiento de objetos varios.

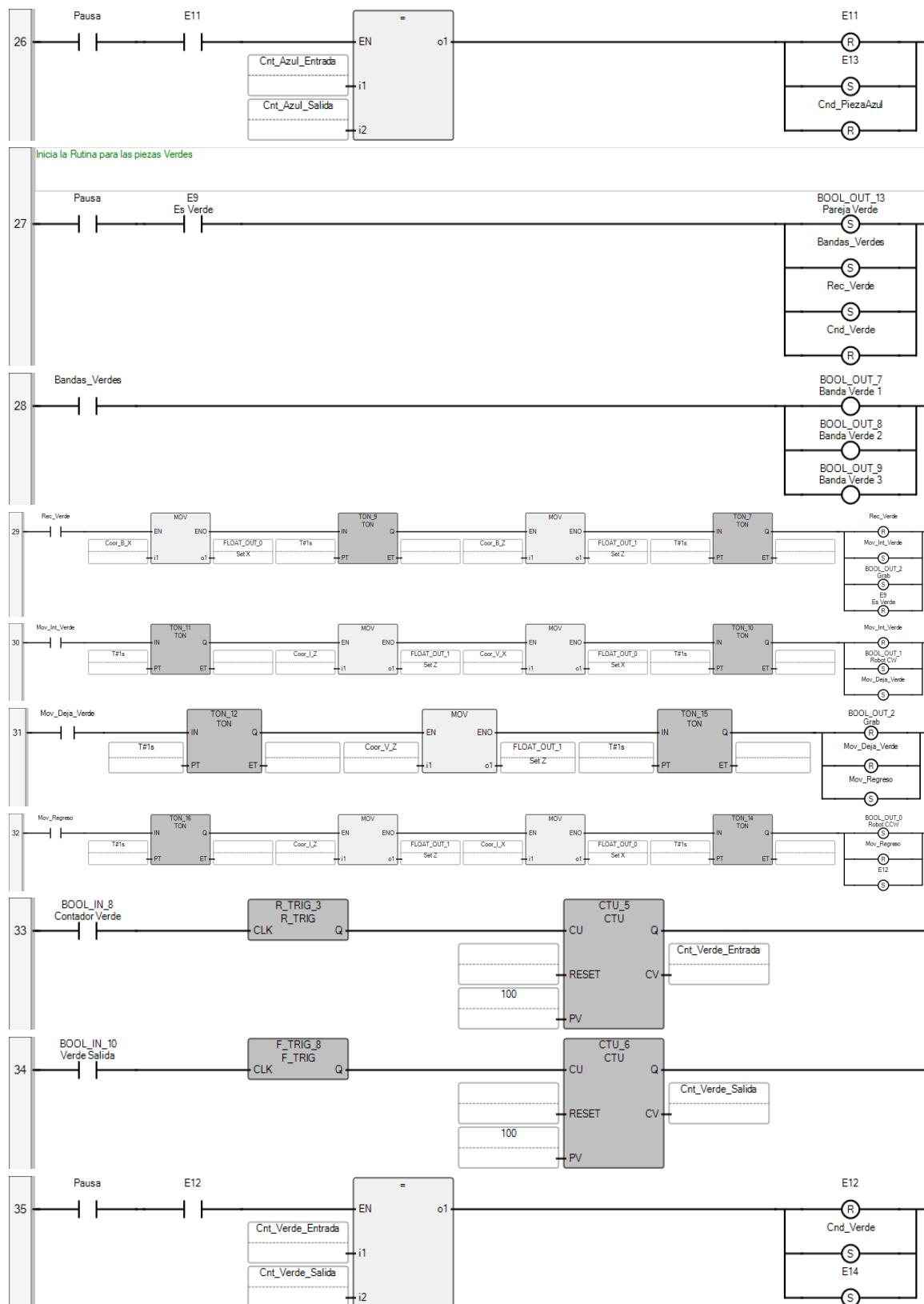
## 5. Anexo

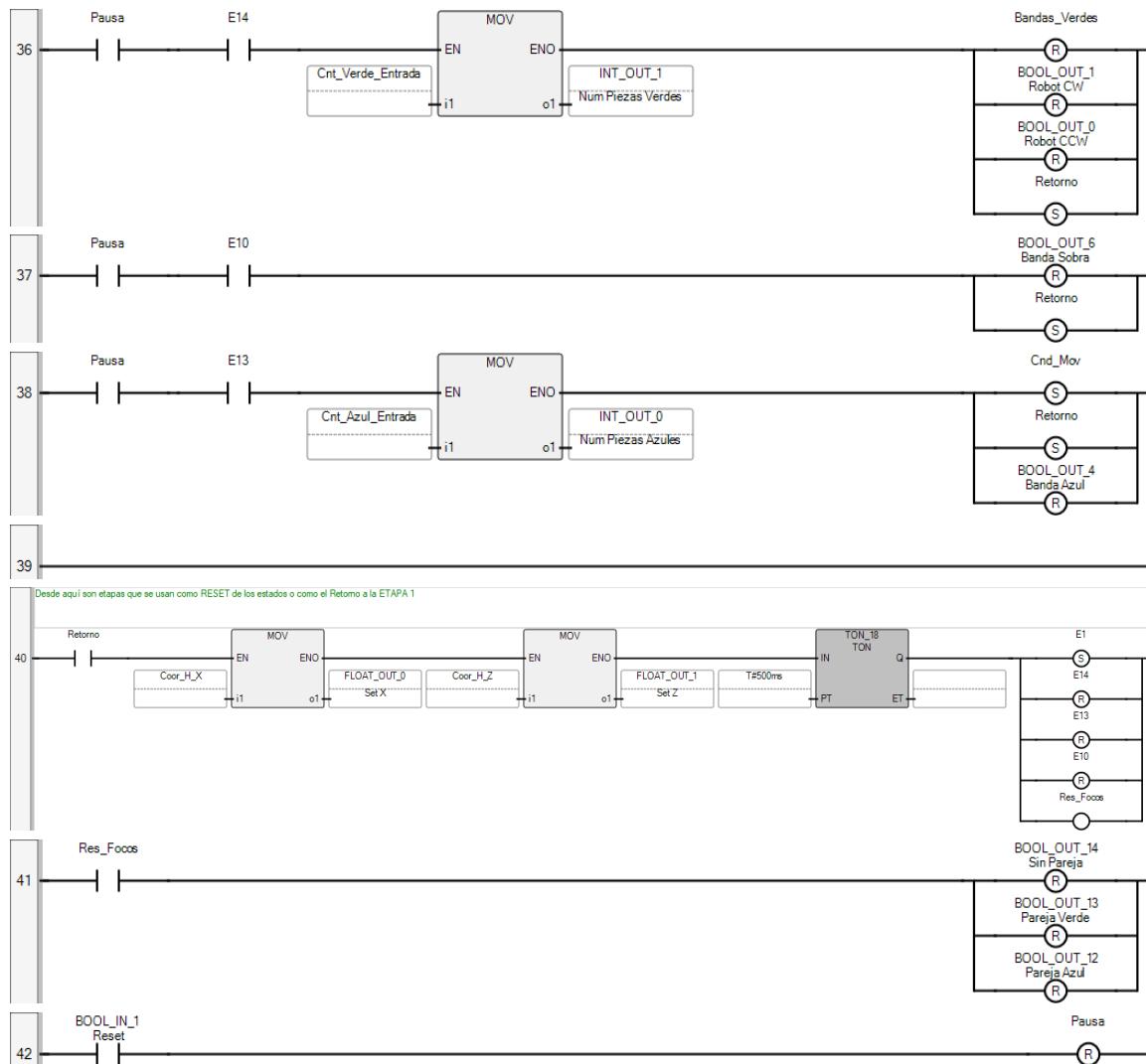
### Código Escalera Proyecto Final

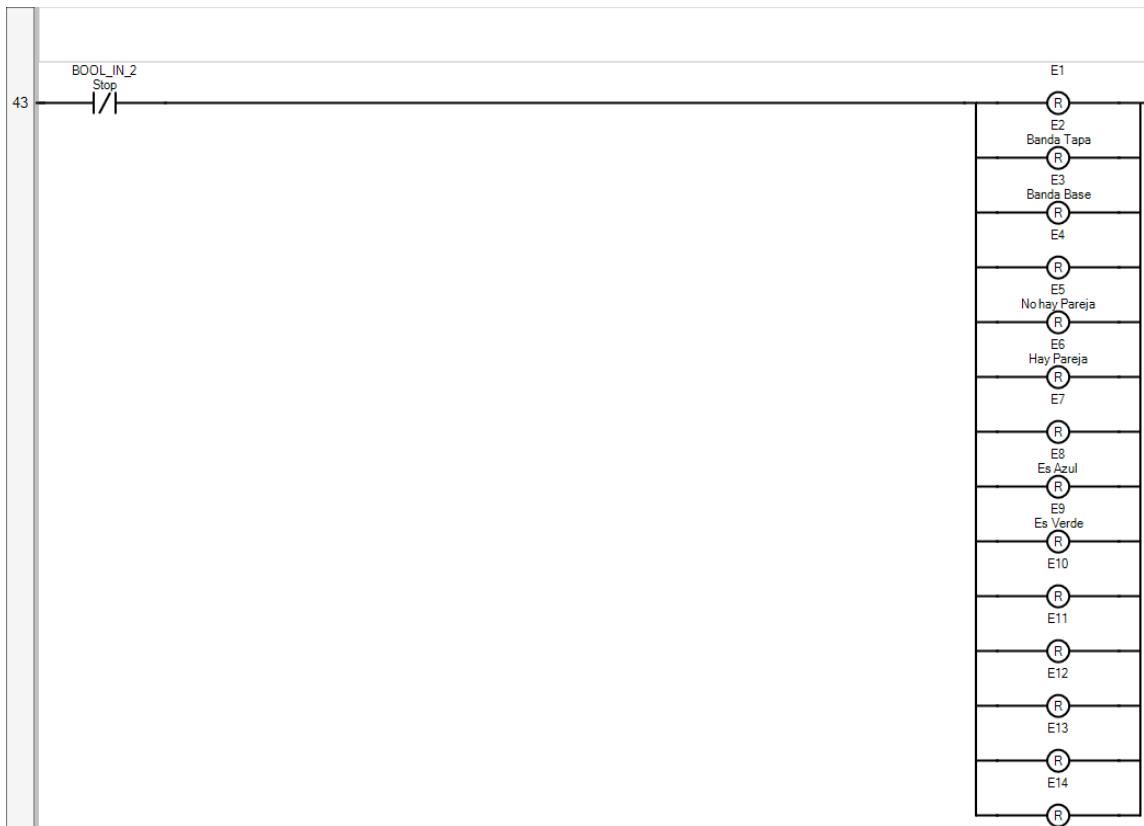












## POU Proyecto Final

The POU defines 84 variable(s).

### Variable E1

(\* \*)

Direction: Var  
 Data type: BOOL  
 Attribute: Read/Write

### Variable E2

(\* \*)

Direction: Var  
 Alias: Banda Tapa  
 Data type: BOOL  
 Attribute: Read/Write

### Variable E3

(\* \*)

Direction: Var  
 Alias: Banda Base  
 Data type: BOOL

Attribute: Read/Write

### Variable E4

(\* \*)  
Direction: Var  
Data type: BOOL  
Attribute: Read/Write

### Variable E5

(\* \*)  
Direction: Var  
Alias: No hay Pareja  
Data type: BOOL  
Attribute: Read/Write

### Variable E6

(\* \*)  
Direction: Var  
Alias: Hay Pareja  
Data type: BOOL  
Attribute: Read/Write

### Variable E7

(\* \*)  
Direction: Var  
Data type: BOOL  
Attribute: Read/Write

### Variable E8

(\* \*)  
Direction: Var  
Alias: Es Azul  
Data type: BOOL  
Attribute: Read/Write

### Variable E9

(\* \*)  
Direction: Var  
Alias: Es Verde  
Data type: BOOL  
Attribute: Read/Write

**Variable E10**

(\* \*)  
Direction: Var  
Data type: BOOL  
Attribute: Read/Write

**Variable E11**

(\* \*)  
Direction: Var  
Data type: BOOL  
Attribute: Read/Write

**Variable E12**

(\* \*)  
Direction: Var  
Data type: BOOL  
Attribute: Read/Write

**Variable E13**

(\* \*)  
Direction: Var  
Data type: BOOL  
Attribute: Read/Write

**Variable E14**

(\* \*)  
Direction: Var  
Data type: BOOL  
Attribute: Read/Write

**Variable Coor\_T\_X**

(\* \*)  
Direction: Var  
Data type: REAL  
Attribute: Read/Write

**Variable Coor\_T\_Z**

(\* \*)  
Direction: Var  
Data type: REAL  
Attribute: Read/Write

**Variable Coor\_B\_X**

(\* \*)

Direction: Var  
Data type: REAL  
Attribute: Read/Write

**Variable Coor\_B\_Z**

(\* \*)

Direction: Var  
Data type: REAL  
Attribute: Read/Write

**Variable Coor\_I\_X**

(\* \*)

Direction: Var  
Data type: REAL  
Attribute: Read/Write

**Variable Coor\_I\_Z**

(\* \*)

Direction: Var  
Data type: REAL  
Attribute: Read/Write

**Variable Coor\_H\_X**

(\* \*)

Direction: Var  
Data type: REAL  
Attribute: Read/Write

**Variable Coor\_H\_Z**

(\* \*)

Direction: Var  
Data type: REAL  
Attribute: Read/Write

**Variable Coor\_V\_X**

(\* \*)

Direction: Var  
Data type: REAL  
Attribute: Read/Write

**Variable Coor\_V\_Z**

(\* \*)  
Direction: Var  
Data type: REAL  
Attribute: Read/Write

**Variable ID\_T\_A**

(\* \*)  
Direction: Var  
Data type: INT  
Attribute: Read/Write

**Variable ID\_T\_V**

(\* \*)  
Direction: Var  
Data type: INT  
Attribute: Read/Write

**Variable ID\_B\_A**

(\* \*)  
Direction: Var  
Data type: INT  
Attribute: Read/Write

**Variable ID\_B\_V**

(\* \*)  
Direction: Var  
Data type: INT  
Attribute: Read/Write

**Variable Cnt\_Sobras\_Salida**

(\* \*)  
Direction: Var  
Data type: DINT  
Attribute: Read/Write

**Variable Cnt\_Sobras\_Establecida**

(\* \*)  
Direction: Var  
Data type: DINT  
Attribute: Read/Write

**Variable Cnt\_Azul\_Entrada**

(\* \*)

Direction: Var  
Data type: DINT  
Attribute: Read/Write

**Variable Cnt\_Azul\_Salida**

(\* \*)

Direction: Var  
Data type: DINT  
Attribute: Read/Write

**Variable Cnt\_Verde\_Entrada**

(\* \*)

Direction: Var  
Data type: DINT  
Attribute: Read/Write

**Variable Cnt\_Verde\_Salida**

(\* \*)

Direction: Var  
Data type: DINT  
Attribute: Read/Write

**Variable Retorno**

(\* \*)

Direction: Var  
Data type: BOOL  
Attribute: Read/Write

**Variable Acoplar**

(\* \*)

Direction: Var  
Data type: BOOL  
Attribute: Read/Write

**Variable Mov\_Int**

(\* \*)

Direction: Var  
Data type: BOOL  
Attribute: Read/Write

**Variable Mov\_Base**

(\* \*)

Direction: Var  
Data type: BOOL  
Attribute: Read/Write

**Variable Mov\_Espera**

(\* \*)

Direction: Var  
Data type: BOOL  
Attribute: Read/Write

**Variable Ident\_Piezas**

(\* \*)

Direction: Var  
Data type: BOOL  
Attribute: Read/Write

**Variable Res\_Focos**

(\* \*)

Direction: Var  
Data type: BOOL  
Attribute: Read/Write

**Variable Rec\_Verde**

(\* \*)

Direction: Var  
Data type: BOOL  
Attribute: Read/Write

**Variable Mov\_Int\_Verde**

(\* \*)

Direction: Var  
Data type: BOOL  
Attribute: Read/Write

**Variable Mov\_Deja\_Verde**

(\* \*)

Direction: Var  
Data type: BOOL  
Attribute: Read/Write

**Variable Bandas\_Verdes**

(\* \*)

Direction: Var  
Data type: BOOL  
Attribute: Read/Write

**Variable Cnd\_NohayPareja**

(\* \*)

Direction: Var  
Data type: BOOL  
Attribute: Read/Write

**Variable Cnd\_PiezaAzul**

(\* \*)

Direction: Var  
Data type: BOOL  
Attribute: Read/Write

**Variable Cnd\_PiezaVerde**

(\* \*)

Direction: Var  
Data type: BOOL  
Attribute: Read/Write

**Variable TON\_8**

(\* \*)

Direction: Var  
Data type: TON  
Attribute: Read/Write

**Variable TON\_6**

(\* \*)

Direction: Var  
Data type: TON  
Attribute: Read/Write

**Variable TON\_5**

(\* \*)

Direction: Var  
Data type: TON  
Attribute: Read/Write

**Variable TON\_4**

(\* \*)

Direction: Var  
Data type: TON  
Attribute: Read/Write

**Variable TON\_3**

(\* \*)

Direction: Var  
Data type: TON  
Attribute: Read/Write

**Variable TON\_1**

(\* \*)

Direction: Var  
Data type: TON  
Attribute: Read/Write

**Variable TON\_2**

(\* \*)

Direction: Var  
Data type: TON  
Attribute: Read/Write

**Variable CTU\_2**

(\* \*)

Direction: Var  
Data type: CTU  
Attribute: Read/Write

**Variable F\_TRIG\_3**

(\* \*)

Direction: Var  
Data type: F\_TRIG  
Attribute: Read/Write

**Variable CTU\_1**

(\* \*)

Direction: Var  
Data type: CTU  
Attribute: Read/Write

**Variable R\_TRIG\_2**

(\* \*)  
Direction: Var  
Data type: R\_TRIG  
Attribute: Read/Write

**Variable F\_TRIG\_7**

(\* \*)  
Direction: Var  
Data type: F\_TRIG  
Attribute: Read/Write

**Variable TON\_9**

(\* \*)  
Direction: Var  
Data type: TON  
Attribute: Read/Write

**Variable TON\_7**

(\* \*)  
Direction: Var  
Data type: TON  
Attribute: Read/Write

**Variable TON\_11**

(\* \*)  
Direction: Var  
Data type: TON  
Attribute: Read/Write

**Variable TON\_10**

(\* \*)  
Direction: Var  
Data type: TON  
Attribute: Read/Write

**Variable TON\_15**

(\* \*)  
Direction: Var  
Data type: TON  
Attribute: Read/Write

**Variable R\_TRIG\_3**

(\* \*)  
Direction: Var  
Data type: R\_TRIG  
Attribute: Read/Write

**Variable CTU\_5**

(\* \*)  
Direction: Var  
Data type: CTU  
Attribute: Read/Write

**Variable F\_TRIG\_8**

(\* \*)  
Direction: Var  
Data type: F\_TRIG  
Attribute: Read/Write

**Variable CTU\_6**

(\* \*)  
Direction: Var  
Data type: CTU  
Attribute: Read/Write

**Variable TON\_12**

(\* \*)  
Direction: Var  
Data type: TON  
Attribute: Read/Write

**Variable TON\_16**

(\* \*)  
Direction: Var  
Data type: TON  
Attribute: Read/Write

**Variable TON\_14**

(\* \*)  
Direction: Var  
Data type: TON  
Attribute: Read/Write

**Variable Cnd\_Verde**

(\* \*)  
Direction: Var  
Data type: BOOL  
Attribute: Read/Write

**Variable TON\_17**

(\* \*)  
Direction: Var  
Data type: TON  
Attribute: Read/Write

**Variable CTU\_4**

(\* \*)  
Direction: Var  
Data type: CTU  
Attribute: Read/Write

**Variable F\_TRIG\_6**

(\* \*)  
Direction: Var  
Data type: F\_TRIG  
Attribute: Read/Write

**Variable CTU\_3**

(\* \*)  
Direction: Var  
Data type: CTU  
Attribute: Read/Write

**Variable R\_TRIG\_1**

(\* \*)  
Direction: Var  
Data type: R\_TRIG  
Attribute: Read/Write

**Variable R\_TRIG\_4**

(\* \*)  
Direction: Var  
Data type: R\_TRIG  
Attribute: Read/Write

**Variable F\_TRIG\_1**

(\* \*)

Direction: Var  
Data type: F\_TRIG  
Attribute: Read/Write

**Variable Mov\_Regreso**

(\* \*)

Direction: Var  
Data type: BOOL  
Attribute: Read/Write

**Variable TON\_18**

(\* \*)

Direction: Var  
Data type: TON  
Attribute: Read/Write

**Variable Pausa**

(\* \*)

Direction: Var  
Data type: BOOL  
Attribute: Read/Write