CENTRO DE INGENIERÍA Y DESARROLLO INDUSTRIAL

ESPECIALIDAD DE TECNÓLOGO EN MECATRÓNICA

“Rehabilitación de célula de manufactura IPC 202B SMC implementando la norma IEC 61131-3”

Informe de la Práctica de Entrenamiento Industrial

Nombre de la Empresa o Institución:

CENTRO DE INGENIERÍA Y DESARROLLO INDUSTRIAL

Presenta:

Estudiante: Sergio Armando Molina Gómez

Tutor Académico: Dr. Julio Cesar Solano Vargas

Querétaro, Qro. 15 de agosto del 2018.

Resumen

El presente trabajo muestra el proceso de rehabilitación de la célula de manufactura IPC-202B SMC utilizada para entrenamiento de estudiantes en el Centro de ingeniería y desarrollo industrial. La actualización del módulo comienza con la adaptación del controlador lógico programable CPX-CEC-C1 FESTO. Este controlador es programado en la plataforma CoDeSys, el cual es un ambiente de programación para PLC que sigue la norma IEC 61131-3. Se incluye también el desarrollo de tres programas con CoDeSys, utilizando tres de los cinco lenguajes definidos en la norma, diagrama escalera, diagrama de bloques funcionales y texto estructurado.

A través de este proyecto se prevé recuperar está herramienta necesaria para la practica de programación de controladores lógicos, además de introducir CoDeSys como un software importante para el aprendizaje de los cinco lenguajes de programación definidos en la norma IEC 61131-3.

Índice general

Índice de figuras

1 Introducción

Un controlador lógico programable (PLC) es un microprocesador que usa una memoria programable para almacenar instrucciones e implementar funciones lógicas, secuenciales y aritméticas con la finalidad de controlar maquinas o procesos (Hanssen, 2015). Hoy en día la mayoría de las industrias usan PLCs para controlar sus sistemas de producción, gracias a que estos dispositivos ayudan a optimizar procesos reduciendo costo y tiempo de producción.

Cuando el primer PLC fue desarrollado en 1969 el único lenguaje de programación disponible para estos dispositivos era el lenguaje escalera (ladder) el cual es una representación gráfica de la lógica de relevadores comúnmente usada para controlar máquinas antes de la aparición de los PLC's(Bolton, 2006). Sin embargo a través de los años se fueron desarrollado más lenguajes de programación para estos dispositivos con la finalidad de hacer más fácil su programación.
Debido a que cada fabricante de PLCs tenia que desarrollar su propio conjunto de instrucciones para cada lenguaje de programación, en 1979 la comisión internacional de electrónica (IEC), juntó un grupo de expertos en PLCs con la finalidad de desarrollar una norma para estandarizar los lenguajes de programación. Fue hasta Marzo de 1993 cuando se publico la norma IEC 61131-3 incluyendo los siguientes lenguajes de programación: Texto estructurado (ST), Diagrama de bloques funcionales (FBD), Diagrama escalera (LD), Lista de instrucciones (IL) y Mapa de funciones secuenciales (SFC). En general la norma IEC 61131-3 está orientada a reducir las diferencias entre PLCs, desarrollando un conjunto de instrucciones en un ambiente de programación en común (Hanssen, 2015).

CoDeSys es un ambiente de programación para PLCs que siguen la norma IEC 61131-3, fue desarrollado por la compañía Smart sofware Solution y sus

siglas vienen de las palabras en ingles, Controlled Developmente System (CoDeSys), que se traduce como Desarrollo de sistemas para controladores. Es considerado como uno de los mejores ambientes de programación de PLCs para aprender, ya que permite el uso de los cinco lenguajes de programación definidos en la norma IEC 61131-3, a demás que cuenta con un entorno de simulación visual que facilita la prueba de los códigos incluso cuando no se cuenta con un PLC físico (Solutions, 2003).

Con la finalidad de proporcionar un entrenamiento completo sobre programación de PLCs, CIDESI adquirió la célula de manufactura IPC-202B SMC. Esta célula de manufactura realiza una secuencia de llenado de botes de forma automatizada, controlado por un PLC Micrologix 1500 de la marca Allen-Bradley (*IPC-202B*). El propósito principal de esta máquina es brindar a los alumnos la oportunidad de desarrollar un programa para un sistema automatizado real, ayudando a desarrollar por medio de la practica sus habilidades de programación de PLCs. Sin embargo esta célula de manufactura no ha sido utilizada desde hace tres años, debido a que el PLC Micrologix 1500 se ha vuelto obsoleto.

Este trabajo presenta la implementación de la norma IEC 61131-3 como parte de la rehabilitación de la célula de manufactura IPC-202B SMC usada para entrenamiento. Se ha considerado emigrar todas las funciones del PLC Micrologix 1500 Allen-Bradley a un PLC CPX-CEC-C1 FESTO. A demás de desarrollar el programa necesario para la secuencia de movimientos, usando el ambiente de programación para PLCs CoDeSys.

2 Planteamiento del Problema

Hoy en día el uso de PLCs como dispositivos de control de máquinas y procesos es cada vez más amplio en la industria en general, ya que ayudan a reducir costos y tiempo de producción al realizar los procesos de forma más eficiente y segurar. Esto a la vez demanda la preparación de gente especializada en la programación de estos dispositivos.

El Centro de Ingeniería y Desarrollo Industrial (CIDESI) adquirió la célula de manufactura IPC-202B SMC, con la finalidad de proporcionar un módulo de entrenamiento para los cursos de programación de PLCs. Está máquina es controlada mediante un PLC Micrologix 1500 Allen-Bradley, sin embargo, debido a la rápida evolución de la tecnología este PLC se ha vuelto obsoleto, resaltando como mayor desventaja que solo admite el diagrama escalera como lenguaje de programación. Esto a la vez a provocado que este módulo de entrenamiento no haya sido utilizado en los últimos 3 años.

Siendo una herramienta clave para el desarrollo de conocimiento mediante practicas, se ha propuesto rehabilitar la célula de manufactura mediante la adaptación del PLC CPX-CEC-C1 FESTO el cual es programado mediante el ambiente de programación CoDeSys que a su vez trabaja bajo la norma IEC 61131-3.

3 Justificación

La enseñanza en el uso de los distintos lenguajes de programación de PLCs, requiere forzosamente contar con módulos de entrenamiento, donde se puedan llevar a cabo prácticas sobre un sistema automatizado real. Sin embargo es de gran importancia que dichos módulos se encuentren actualizados, ya que de esta forma el conocimiento transmitido será de mayor utilidad en la vida real.

La rehabilitación de la célula de manufactura IPC-202B SMC tendrá un impacto positivo en los estudiantes interesados en aprender más sobre programación de PLCs, ya que les proporcionará la herramienta necesaria para practicar y desarrollar su conocimiento. A demás, la implementación de la norma IEC 61131-3 permite el aprendizaje de los cinco lenguajes de programación de PLCs que son actualmente usados en la industria.

4 Objetivos

4.1. Objetivo General

Rehabilitar la célula de manufactura IPC-202B SMC mediante la implementación de la norma IEC 61131-3.

4.2. Objetivos Específicos

Para lograr este objetivo, las siguientes tareas se tendrán que cumplir:

- Identificar componentes que integran la célula de manufactura (sensores, válvulas y actuadores).
- Obtener la secuencia de movimientos realizada por los actuadores.
- Corregir fugas de aire en el sistema neumático.
- Adaptar el PLC CPX-CEC-C1 FESTO en el panel frontal de la célula.
- Desarrollar dos programas con lenguajes diferentes usando la plataforma CoDeSys.

5 Marco Teórico

5.1. La norma IEC 61131-3

Un grupo de expertos en PLCs fue formado en de 1979 con la misión de desarrollar la primer version de una norma para PLCs. Esta primer prueba fue presentada en de 1982, la norma fue lo suficientemente comprensible para incluirla en un documento simple. El grupo fue dividido en cinco diferentes secciones(Hanssen, 2015).

1. Información general.
2. Hardware y requerimientos para pruebas.
3. Lenguajes de programación.
4. Interfaz de usuario.
5. Comunicaciones.

En marzo de 1993, la primer norma en lenguajes de programación, designada como IEC 61131-3.2, fue publicada. La ultima actualización de esta norma fue en el año 2003. Esta ultima actualización incluyó los siguientes lenguajes de programación para PLC (Hanssen, 2015).

1. Texto estructurado (ST).
2. Diagrama de bloques funcionales (FBD).
3. Diagrama escalera (LD).
4. Lista de instrucciones (IL).
5. Diagrama de funciones secuenciales (SFC).

En el pasado los fabricantes de PLCs tenían que desarrollar su propio dialecto de programación, esto significa que si una compañía decidía cambiar de PLC, el personal tenía que aprender un nuevo dialecto de programación. En general la norma IEC 61131-3 esta centrada en reducir las diferencias entre PLCs, mediante el desarrollo de un conjunto de instrucciones que se establecen en un ambiente de programación en común.

5.2. Ambiente de programación CoDeSys.

CoDeSys es un ambiente de programación que trabaja bajo la norma IEC 61131-3. Fue desarrollado por la empresa Smart Software Soolutions, y su nombre viene dado por las palabras en ingles Controlled Development System (Hanssen, 2015).

CoDeSys es un sistema de programación independiente para PLC, microcontroladores y otros hardwares. Esto significa que CoDeSys no ha sido desarrollado exclusivamente para PLCs, esto permite que CoDeSys pueda ser utilizado por una gran variedad de fabricantes de PLCs entre los que destacan Beckho, ABB, WAGO, Schneider and Festo, entre otros (viuela, 2013).

Es considerado uno de los mejores ambientes de programación de PLC para estudiantes, debido a que soporta los cinco lenguajes de programación para PLC definidos en la norma IEC 61131-3. Además el software contiene una herramienta de simulación por lo cual los programas pueden ser probados incluso sin la necesidad de tener el hardware (Hanssen, 2015). Otra ventaja importante es que CoDeSys puede ser descargado sin costo y sin limite de tiempo desde la pagina oficial CoDeSys.com.

5.3. Diagrama escalera LD

El lenguaje escalera ha sido el lenguaje de programación de PLC más utilizado desde la aparición del primer PLC. Otro concepto utilizado para este código escrito es diagrama de relevadores. Incluso con la aparición de otros lenguajes de programación más eficientes, el diagrama escalera sigue siendo ampliamente utilizado debido a que es es un lenguaje gráfico relativamente fácil de usar. (Hanssen, 2015).

Las funciones básicas necesarias para implementar un sistema de control pequeño pueden ser aprendidas relativamente rápido y la presentación gráfica se puede entender de forma intuitiva (Hanssen, 2015).

En la figura 5.1 se muestra el principio de funcionamiento del código escalera. Ambos lados del código están limitados por lineas verticales que llamamos rieles de potencia. El riel de la izquierda es considerado como el riel de voltaje (+24V), mientras que el riel de la derecha terminal neutra o tierra.

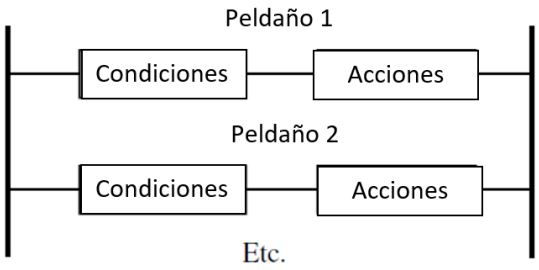


FIGURA 5.1: Estructura del diagrama escalera

En general, el código escalera sigue el siguiente principio.

- Si una condición o una combinación de condiciones se cumple, entonces una o más acciones serán ejecutadas.

Las condiciones consisten en contactos o combinación de contactos. Un contacto es un elemento gráfico que es asociado una variable booleana, por lo cual solo puede tener valores de 0 ó 1, es decir el contacto tiene estado abierto o cerrado.
Cuando las condiciones se cumplen, activan la bobina del lado derecho, que

también esta asociada a un valor booleano, (activo o inactivo). Un ejemplo de este código se puede observar en la figura 5.2.

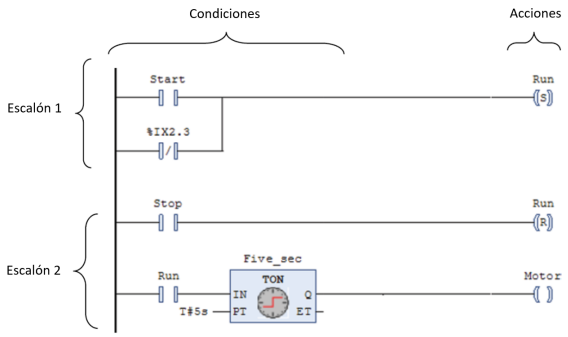


FIGURA 5.2: Ejemplo de código en diagrama escalera.

5.4. Diagrama de bloques funcionales (FBD).

El segundo lenguaje gráfico que esta definido en la norma IEC 61131-3 es el diagrama de bloques funcionales. Sigue los mismos lineamientos definidos para el diagrama escalera respecto a gráficos y estructura. Este lenguaje incluye funciones lógicas tales como AND, OR, NOT, entre otras en forma gráfica, las personas que tienen conocimiento de electrónica digital encuentran a este lenguaje fácil de utilizar(Hanssen, 2015).

En este lenguaje de programación las variables son usadas directamente como argumentos de entrada para las funciones en los bloques. Como en los peldaños en el diagrama escalera, el código FBD está también divido en lineas de trabajo, colocadas verticalmente una sobre otra. Un ejemplo de este código se muestra en la figura 5.3.

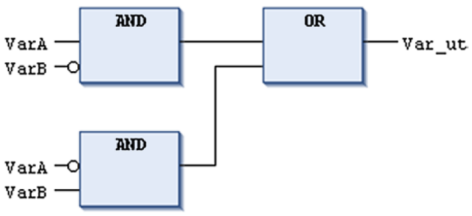


FIGURA 5.3: Ejemplo de código FBD

5.5. Texto Estructurado ST

El Texto estructurado es un lenguaje de alto nivel que se asemeja mucho al lenguaje Pascal. Pero para cualquiera que haya trabajado en programación, ya sea C o C++, puede identificar mucha de la sintaxis y en consecuencia podrá ser fácil adaptarse a ST. El lenguaje contiene muchos elementos, comandos e instrucciones que suelen ser comunes en otros lenguajes de programación de alto nivel (Hanssen, 2015).

La principal fortaleza del texto estructurado es ante todo con cálculos aritméticos, procesamiento de números y manejo de datos de tipo estructurado. Por estas razones la programación es mas rápida, y el código es mucho mas comprimido en comparación a diagrama escalera. En la figura 5.4 se muestra un ejemplo de texto estructurado.

```
WHILE index1 < 100 DO
    index2 := 0;
    Value := Table1[index1];
    REPEAT
        Table2[index2] := Value + Table2[index2];
        IF Table2[index2] > 32767 THEN
            EXIT;
        END_IF
        index2 := index2 + 1;
    UNTIL index2 >25
    END_REPEAT
    index1 := index1 + 1;
END_WHILE
```

FIGURA 5.4: Ejemplo de texto estructurado ST

5.6. Módulo de llenado con alimentador completo IPC-202B

Este módulo corresponde con la etapa de llenado dentro del sistema que emula una planta de producción y embotellado de bebidas en forma automatizada (figura 5.5).

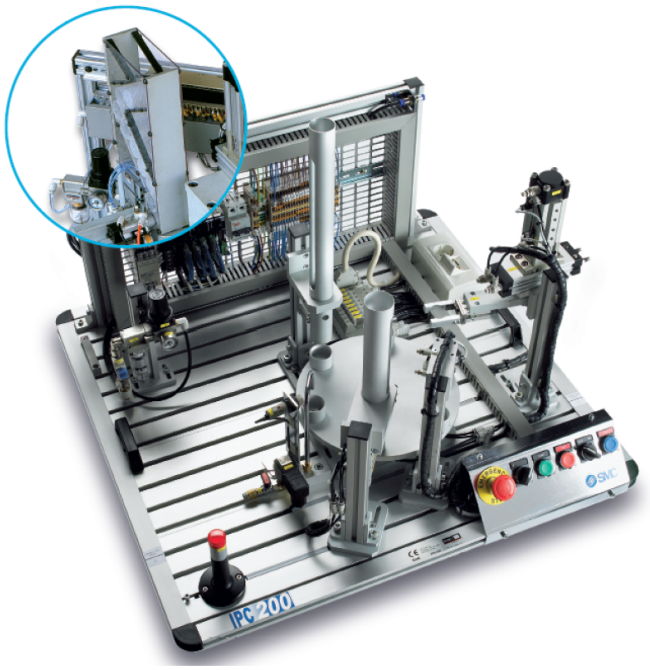


FIGURA 5.5: Módulo IPC-202B

Todos los componentes utilizados en la fabricación del modulo son industriales (*IPC-202B*).

5.6.1. Alimentación de botes completo

Fabricado en aluminio anodizado y metacrilato se encarga de alimentar los botes en la posición correcta e incluye los siguientes elementos:

- 2 cilindros compactos con guías y regulador de caudal de carrera 10 y diámetro 12 mm controlados por electroválvula 5/2.
- 2 cilindros compactos de doble efecto de carreras 15 y 75 mm y diámetro 12mm con reguladores de caudal y detector de posición final reed controlados por electroválvula 5/2.
- 1 actuador de giro neumático tipo piñón – doble cremallera con reguladores de caudal y detectores de posición 0^o y 180^o controlados por electroválvulas 5/2.
- 5 detectores magnéticos tipo reed.

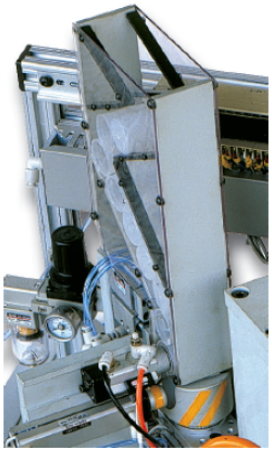


FIGURA 5.6: Alimentador de botes completo

5.6.2. Inserción en plato giratorio

Se encarga de trasladar el bote desde el alimentador hasta la posición de recogida del plato divisor e incluye los siguientes elementos:

- 1 cilindro compacto de movimiento lineal y rotativo de carrera 50 y diámetro 32mm, con reguladores de caudal y detectores de posición inicial y final reed en el movimiento lineal y de 0^o y 180^o en el rotativo, controlado por 2 electroválvulas 5/2.
- 1 unidad de pinza giratoria de doble efecto con dos dedos de apertura paralela, controlada por electroválvula 5/2.
- 1 acoplamiento rotolineal.
- 4 detectores magnéticos tipo reed.

6 Metodología

La metodología será dividida en 3 secciones:

(a) Acondicionamiento. En esta sección se ejecutan los pasos necesarios para realizar el cambio de PLC de la célula de manufactura.

(b) Programación. En esta sección se desarrollan tres programas diferentes para la célula de manufactura, usando la plataforma CoDeSys.

(c) Pruebas. En esta sección se realiza la simulación de cada código y posteriormente la prueba de funcionamiento en el PLC.

6.1. Acondicionamiento

1. Antes de desconectar cualquier componente del PLC Allen Bradley, es necesario determinar cuantos sensores, válvulas y actuadores se tienen y donde están conectados. Esta actividad se ve resumida en la imagen 6.1, donde se muestra la relación de cada actuador con los sensores correspondientes y la válvula que lo activa.

Componentes de la célula de manufactura IPC 202B SMC							
N	Actuadores	Sensor inicio	Sensor final	Válvula	Descripción	Botónes	Identificador
1	A	a0 / I-04	NA	A / O-01	Avance de plato central	Botón Start	I-00
2	B	NA	NA	B / O-02	Anclaje de plato central	Botón Stop	I-01
3	C	c0 / I-05	c1 / I-06	C / O-03	Sube o baja pinza de entrega	Botón Auto/man	I-02
4	D	d0 / I-07	d1 / I-08	D / O-04	Rota la pinza a la posición de entrega	Botón Reset	I-03
5	E	NA	e1 / I-09	E / O-05	Abre y cierra la pinza de entrega		
6	F	f0 / I-10	f1 / I-11	F / O-06	Sostine columna de tapas		
7	G	g0 / I-12	g1 / I-13	G / O-07	Presión para cerrar la tapa		
8	H	h0 / I-14	h1 / I-15	H / O-08	Sube o baja pinza de alimentador		
9	I	i0 / I-16	i1 / I-17	I / O-09	Rota pinza del alimentador		
10	J	NA	NA	J / O-010	Cierra pinza del alimentador		
11	K	NA	NA	K / O-011	Retiene columna de botes		
12	L	NA	l1 / I-18	L / O-012	Determina orientación del bote		
13	M	m0 / I-19	m1 / I-20	M / O-013	Entrega el bote a la pinza del alimentador		
14	N	n0 / I-21	n1 / I-22	N / O-014	Corrige la orientación del bote		

FIGURA 6.1: Lista de componentes de la célula de manufactura IPC-202B

2. Para tener una idea de las instrucciones que se deben de programar, se realizan diagramas de flujo como los que se muestran en las figuras 6.2 Y 6.3, que corresponden al alimentador y a la pinza de entrega a plato central, respectivamente.En ellos se expresa de forma gráfica la secuencia de movimientos que realizan los actuadores.

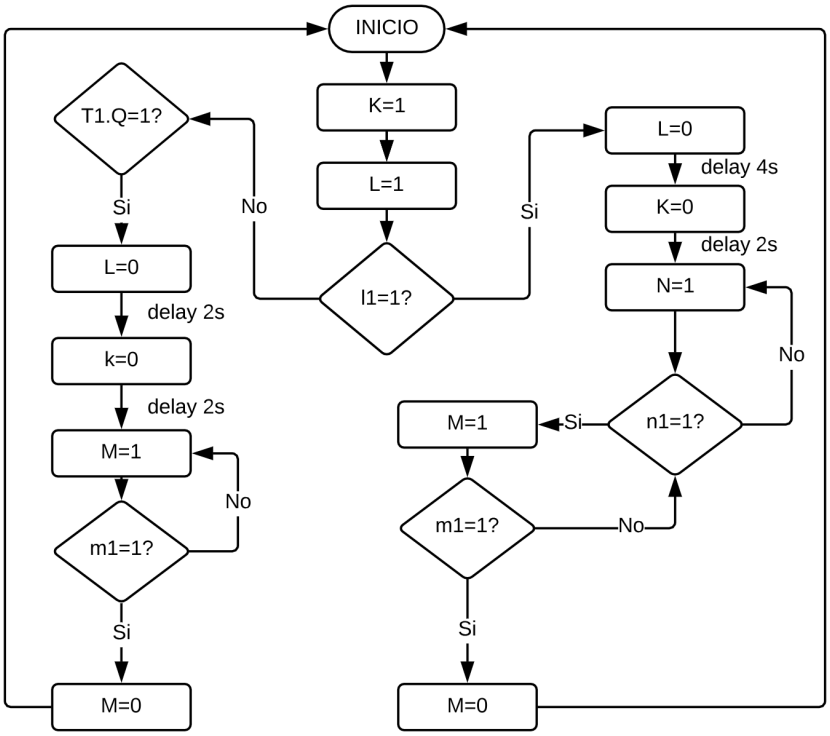


FIGURA 6.2: Diagrama de flujo del alimentador

Una vez definidos estos pasos podemos desconectar todos los componentes y extraer el PLC Allen Bradley.

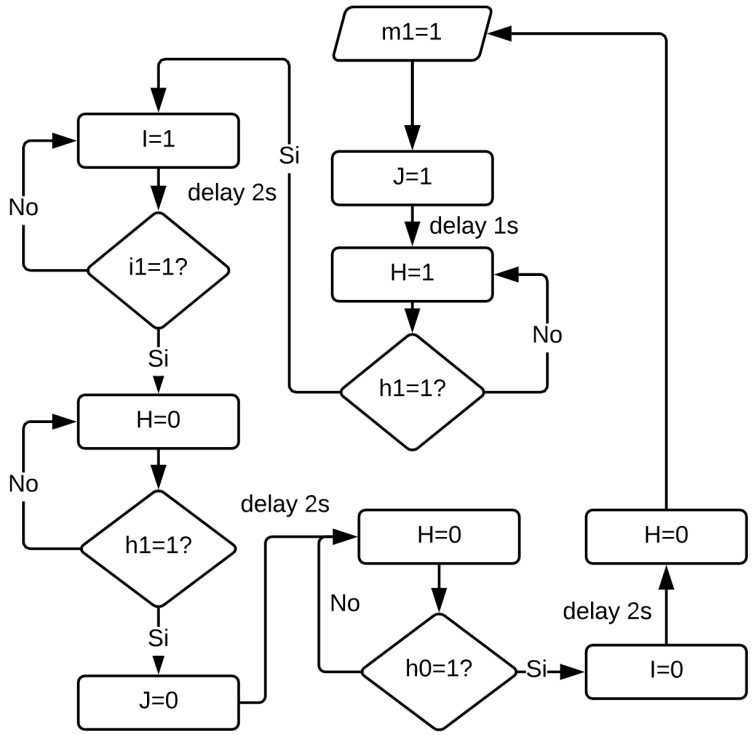
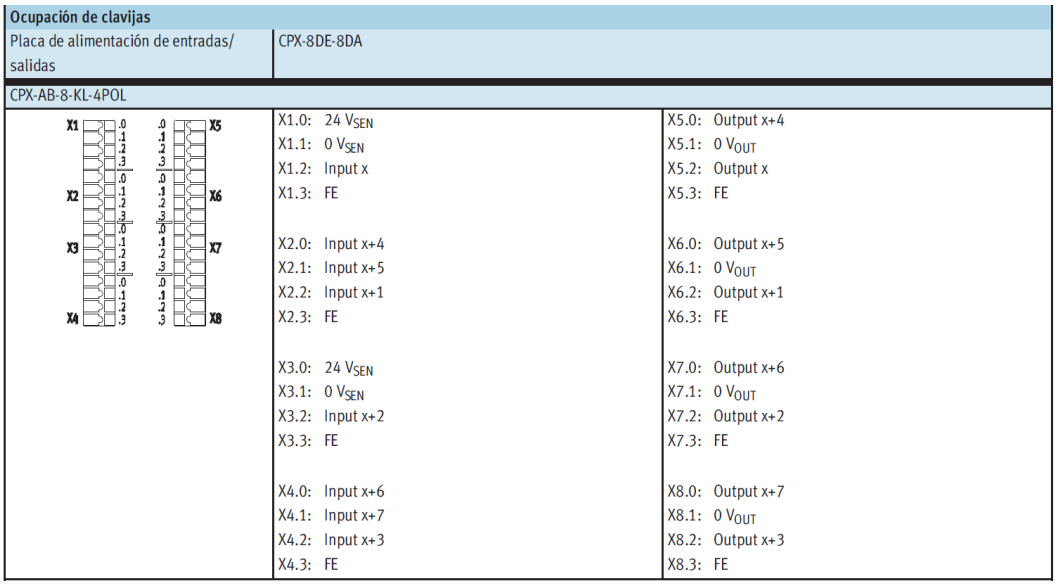


FIGURA 6.3: Diagrama de flujo "Pinza de entrega a plato central"

3. Antes de conectar el PLC CPX-CEC-C1 FESTO, debemos estudiar las condiciones eléctricas a las cuales trabaja este PLC, así como contar con un mapa de distribución de entradas y salidas en los módulos. En este caso el PLC es alimentado a 24 Volts en corriente continua, y el esquema de conexiones se muestra en la imagen 6.4.



Ocupación de clavijas		
Placa de alimentación de entradas/ salidas	CPX-8DE-8DA	
CPX-AB-8-KL-4POL		
X1 .0 .1 .2 .3 X5 .0 .1 .2 .3 X2 .0 .1 .2 .3 X6 .0 .1 .2 .3 X3 .0 .1 .2 .3 X7 .0 .1 .2 .3 X4 .0 .1 .2 .3 X8 .0 .1 .2 .3	X1.0: 24 V_{SEN} X1.1: 0 V_{SEN} X1.2: Input x X1.3: FE X2.0: Input x+4 X2.1: Input x+5 X2.2: Input x+1 X2.3: FE X3.0: 24 V_{SEN} X3.1: 0 V_{SEN} X3.2: Input x+2 X3.3: FE X4.0: Input x+6 X4.1: Input x+7 X4.2: Input x+3 X4.3: FE	X5.0: Output x+4 X5.1: 0 V_{OUT} X5.2: Output x X5.3: FE X6.0: Output x+5 X6.1: 0 V_{OUT} X6.2: Output x+1 X6.3: FE X7.0: Output x+6 X7.1: 0 V_{OUT} X7.2: Output x+2 X7.3: FE X8.0: Output x+7 X8.1: 0 V_{OUT} X8.2: Output x+3 X8.3: FE

FIGURA 6.4: Distribución de entradas y salidas PLC CPX-CEC-C1 FESTO

Con ayuda de este mapa de distribución podemos proceder a conectar los sensores y las válvulas al nuevo PLC en el orden que definimos anteriormente. Es importante seguir un orden al momento de conectar los sensores y las válvulas, esto nos ayudara a evitar errores cuando desarrollemos los programas.

6.2. Programación

Se desarrollan tres programas diferentes usando tres de los cinco lenguajes definidos dentro de la norma IEC 61131-3 usando la plataforma de programación CoDeSys. Los lenguajes seleccionados para esta sección son:

1. Diagrama escalera. En la imagen 6.5 se muestra parte del código desarrollado, el código completo se deja como anexo.

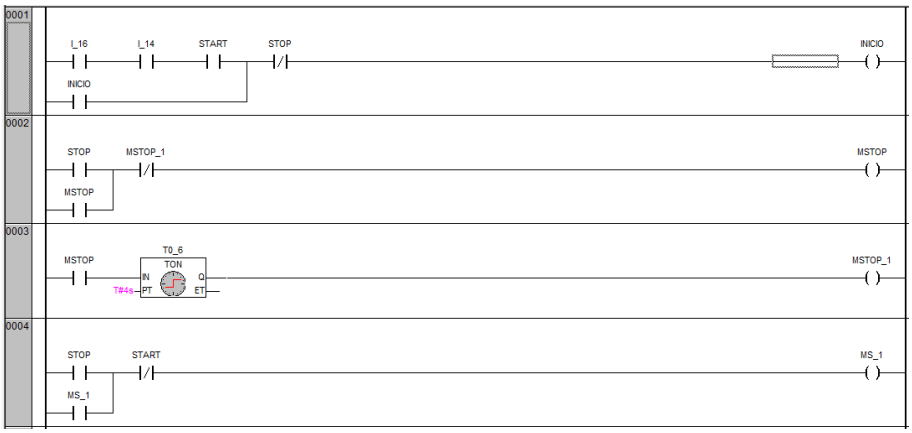


FIGURA 6.5: Parte del programa en diagrama escalera

2. Diagrama de bloques funcionales. En la imagen 6.6 se muestra parte del código desarrollado, el código completo se deja como anexo.

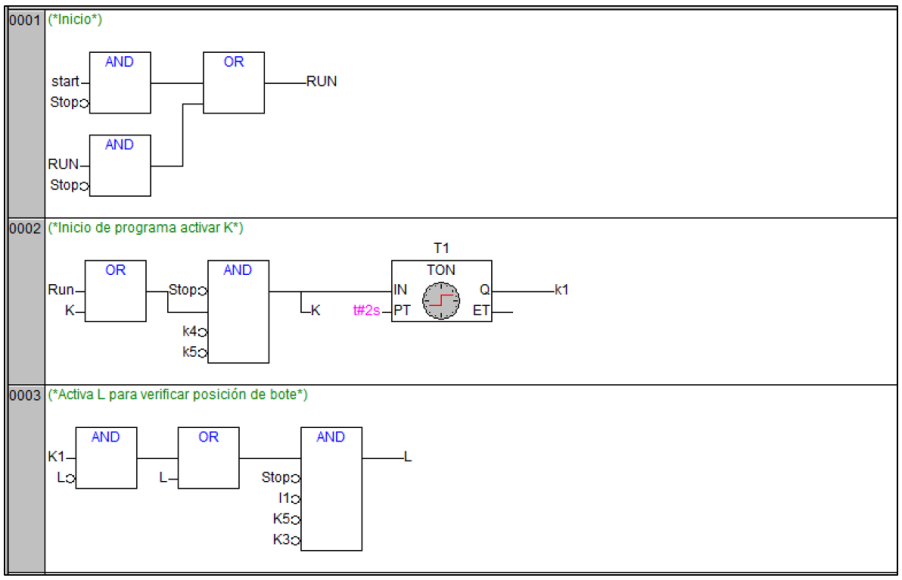


FIGURA 6.6: Parte del código en Diagrama de bloques funcionales.

3. Texto estructurado. En la imagen 6.7 se muestra parte del código desarrollado, el código completo se deja como anexo.

```
0036 (*Corregir posición de bote*)
0037
0038 IF N=TRUE AND n1=TRUE THEN;
0039    TON5(IN:=n1, PT:=T#1S);
0040    M:= TON5.Q;
0041 END_IF;
0042
0043 IF m1=TRUE THEN;
0044   M:=FALSE;
0045   mr:=TRUE;
0046 END_IF;
0047
0048 IF mr = TRUE THEN;
0049   TON6(IN:=mr, PT:=T#2S);
0050    nr:=TON6.Q;
0051 END_IF;
0052
0053 IF nr=TRUE THEN;
0054   N:=FALSE;
0055 END_IF;
```

FIGURA 6.7: Parte del código en Texto estructurado

6.3. Pruebas

Antes de descargar cada uno de los códigos en el PLC y realizar pruebas directas sobre la célula de manufactura, se realiza una simulación usando la herramienta de visualización de CoDeSys. Con esta herramienta podemos correr el código para verificar que la secuencia de movimientos es correcta y de esta forma evitamos que existan colisiones entre actuadores si en dado caso la secuencia de movimientos es incorrecta. En la imagen 6.8 se muestra un ejemplo de la simulación.

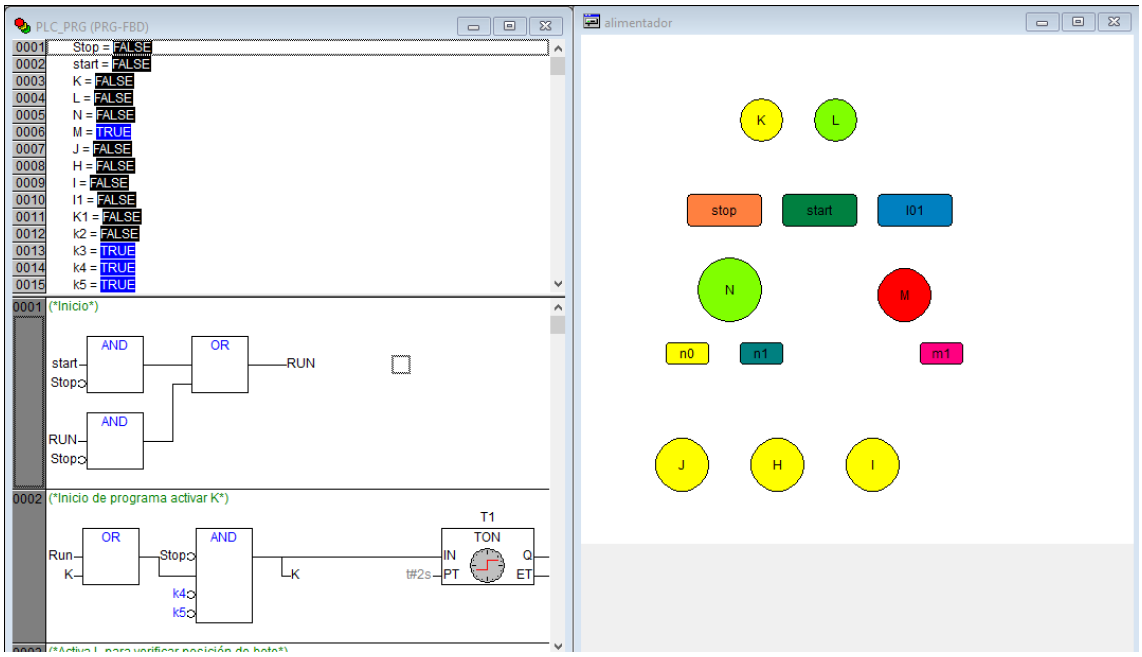


FIGURA 6.8: Ejemplo de simulación

Después de asegurarnos que la secuencia de movimientos se ejecuta de forma correcta, podemos descargar cada uno de los códigos en el PLC y hacer pruebas de funcionamiento. En esta etapa regularmente se realizan correcciones en tiempo de ejecución.

7 Resultados

7.1. Acondicionamiento de la Célula de manufactura IPC 202B SMC

Como parte de los resultados obtenidos con este proyecto se realiza la comparación del módulo antes y después de la rehabilitación. Estas imágenes se muestran en las figuras 7.1 y 7.2 respectivamente.

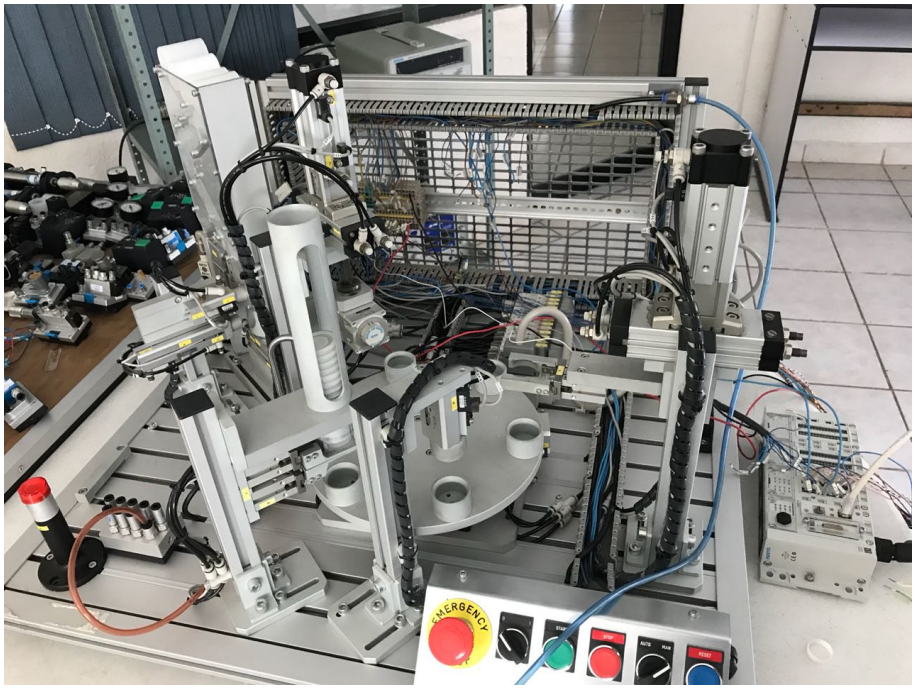


FIGURA 7.1: Foto del módulo IPC 202B antes de la rehabilitaicón

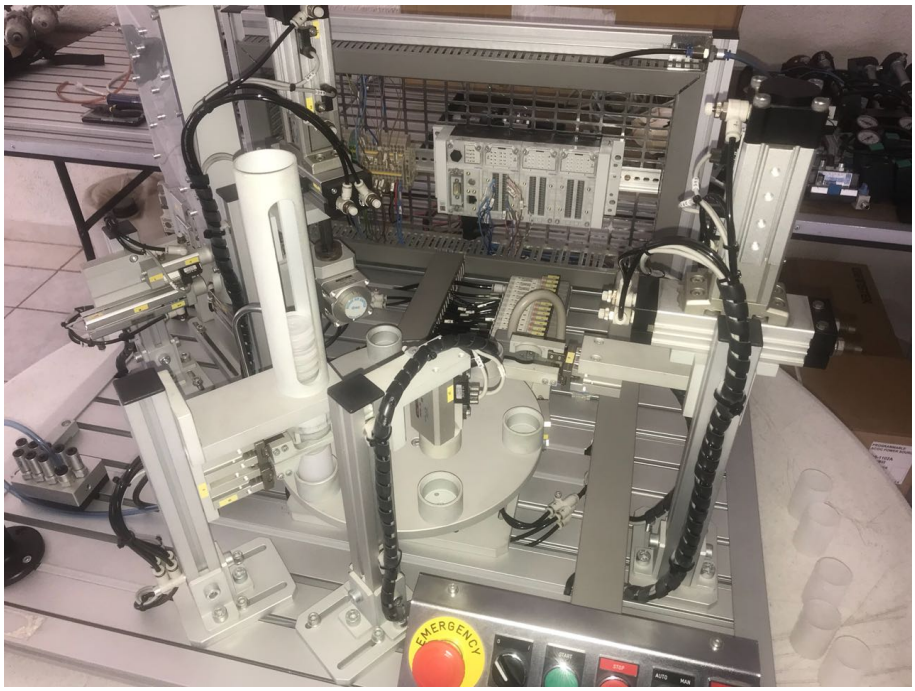


FIGURA 7.2: Foto del módulo IPC 202B después de la rehabilitación

7.2. Funcionamiento del alimentador

La secuencia de movimientos del alimentador inicia con la activación del pistón K el cual retiene la columna de botes, después de esto se activa el pitón L, el cual tiene la finalidad de determinar la orientación del bote. Este proceso se ilustra en la figura 7.3.

FIGURA 7.3: Funcionamiento inicial del alimentador pistón K y L

Después de determinar la orientación del bote, se desactiva el pistón K dejando caer el bote al cilindro giratorio N, si la orientación del bote es correcta, este no se activa, pero si esta mal orientado, el cilindro N se activará para girar la pieza y orientarla correctamente. Este proceso se ilustra en las figuras 7.4 y 7.5.

FIGURA 7.4: El cilindro N recibe el bote.

FIGURA 7.5: El cilindro N se activa para rotar el bote y orientarlo a la salida

Finalmente cuando el bote ya esta orientado correctamente, se activa el pistón M para empujar el bote fuera del alimentador. Este proceso se muestra en la figura 7.6.

FIGURA 7.6: Expulsión del bote.

7.3. Funcionamiento de pinza de entrega a plato central

La secuencia de movimientos de la pinza de entrega al plato central comienza con la activación de la pinza J, la cual sujeta al bote durante todo el proceso. Este proceso se muestra en la figura 7.6.

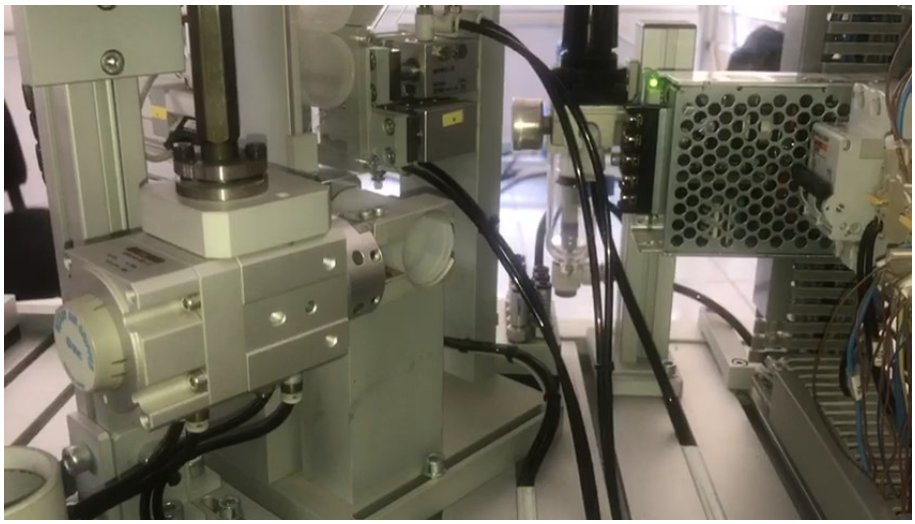


FIGURA 7.7: Pinza J sujeción del bote

Después de la activación de la pinza, se activa el cilindro H para subir la pinza y evitar que colisione con la base del alimentador. Este proceso se muestra en la figura 7.7.

FIGURA 7.8: Subir la pinza, activación de cilindro H

El siguiente paso es rotar la pinza y entregar el bote en el plato central. En este paso se desactiva la pinza J para liberar el bote y dejarlo en el plato. Este proceso se muestra en la figura 7.8.

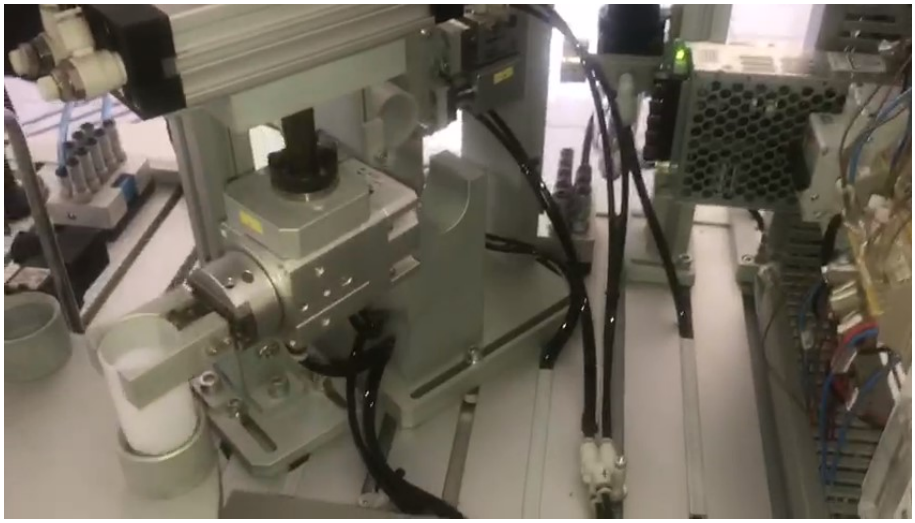


FIGURA 7.9: Entrega del bote a plato central

Una vez que el bote esta en el plato central, todos los actuadores regresan a su posición original y el proceso comienza de nuevo.

8 Conclusiones

Por medio de este trabajo se logró rehabilitar la célula de manufactura IPC 202B que figura como una herramienta muy importante para el aprendizaje de programación de PLCs, permitiendo el desarrollo de practicas en una máquina automatizada real.

Por otra parte mediante la implementación de la norma IEC 61131-3 se logró expandir el conocimiento transmitido hacia los estudiantes, ya que con el uso de CoDeSys se asegura el aprendizaje de los cinco lenguajes de programación definidos en la norma, lo cual representa una actualización importante ya que en la actualidad estos lenguajes son cada vez más usados en la industria.

A Código en diagrama escalera

En este apartado se anexan las imágenes correspondientes al código en lenguaje escalera realizado en CoDeSys.

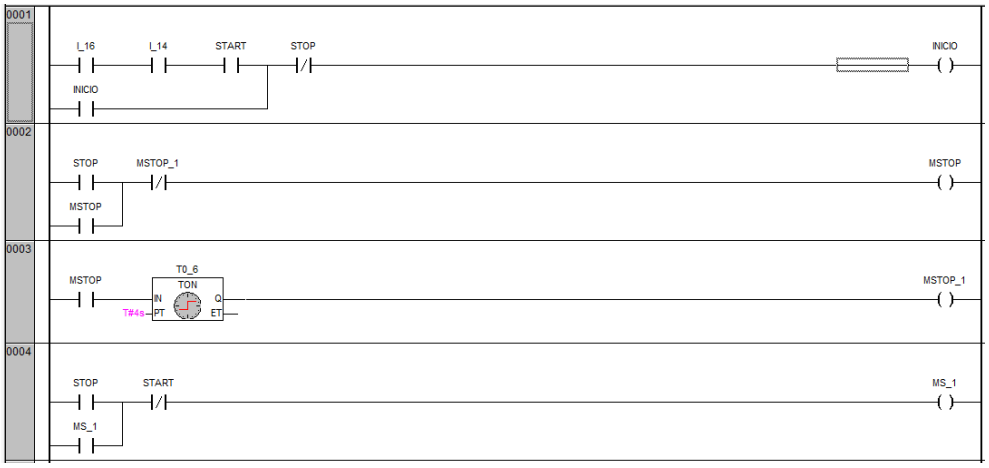


FIGURA A.1: Código escalera parte 1

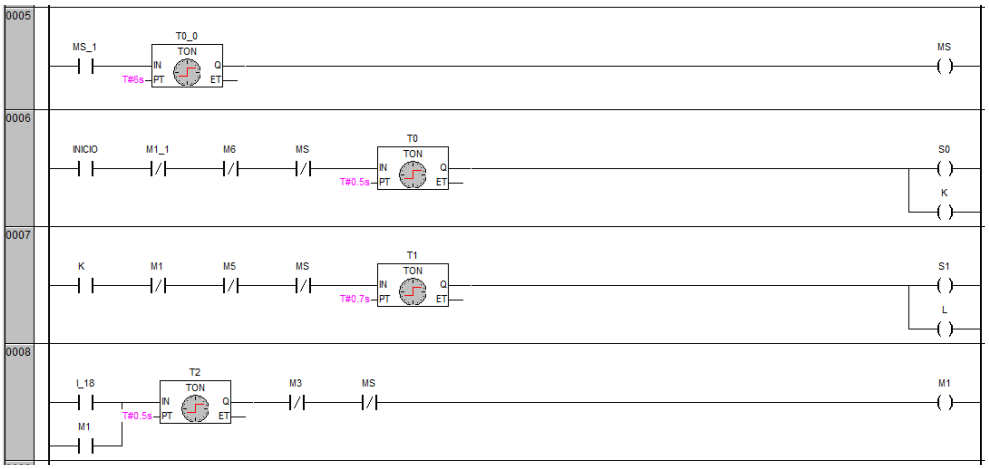


FIGURA A.2: Código escalera parte 2

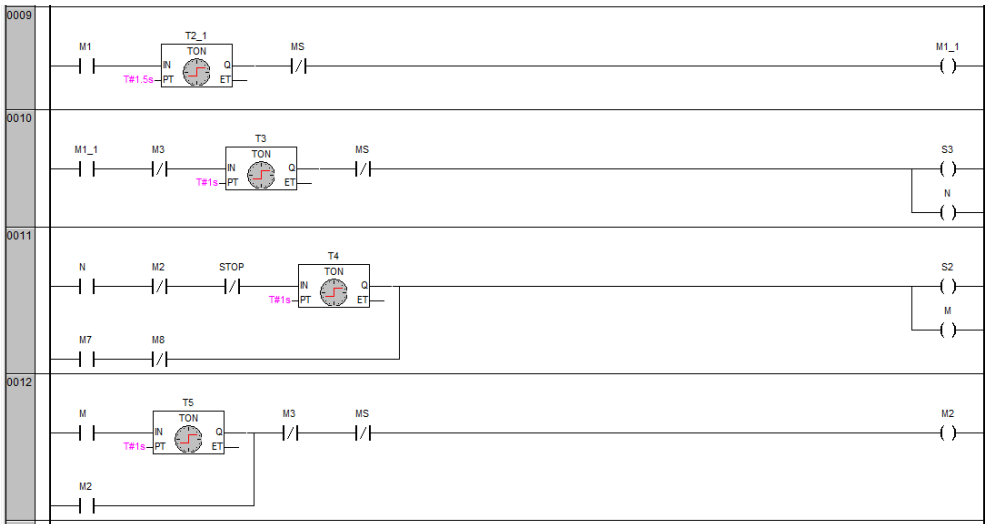


FIGURA A.3: Código escalera parte 3

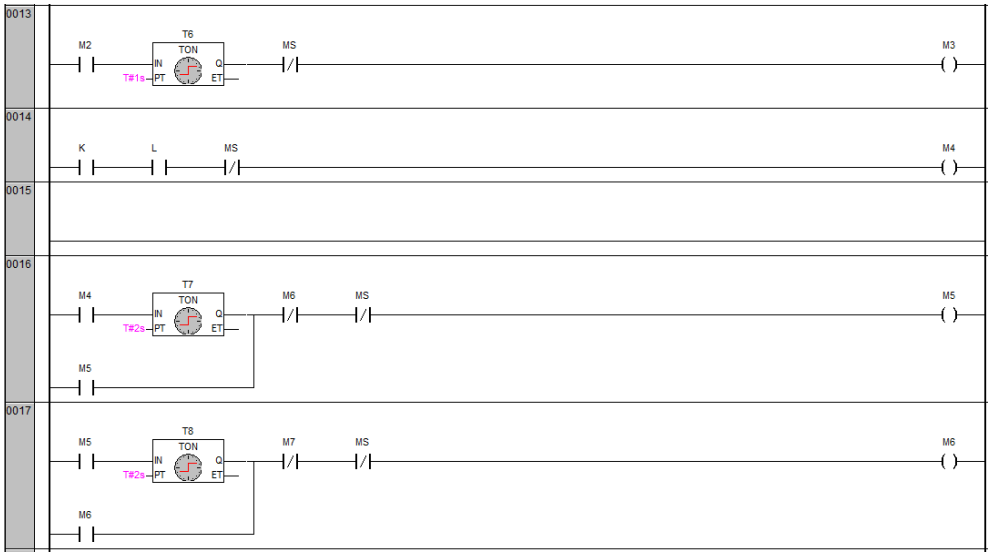


FIGURA A.4: Código escalera parte 4

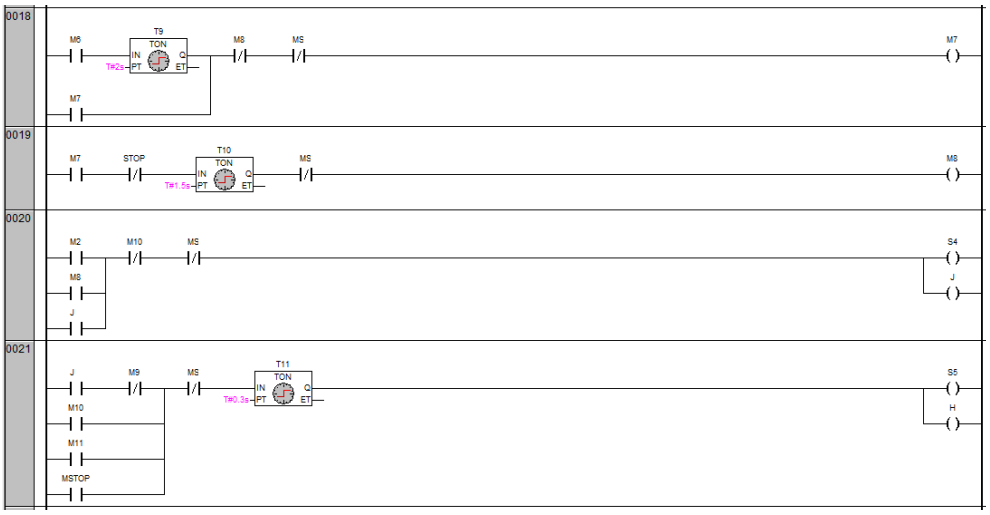


FIGURA A.5: Código escalera parte 5

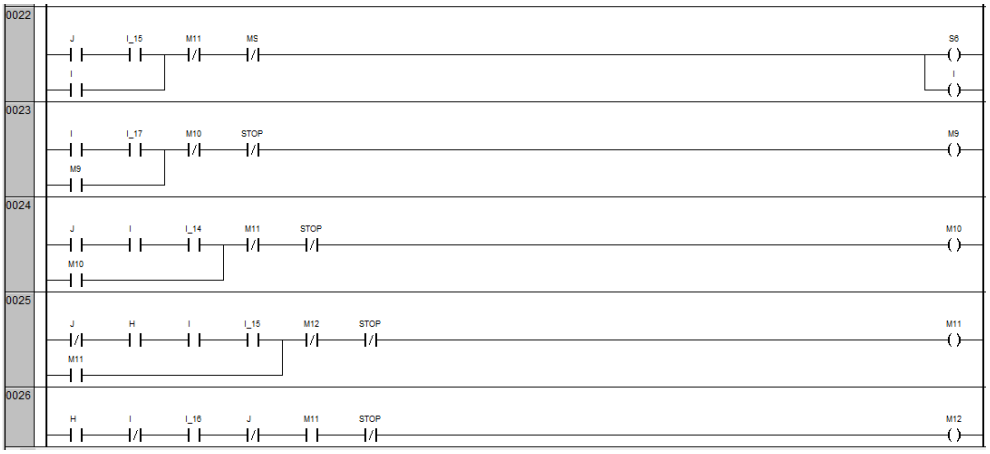


FIGURA A.6: Código escalera parte 6

B Código en diagrama de bloques funcionales

En este apartado se anexan las imágenes correspondientes al código en lenguaje diagrama de bloques funcionales realizado en CoDeSys.

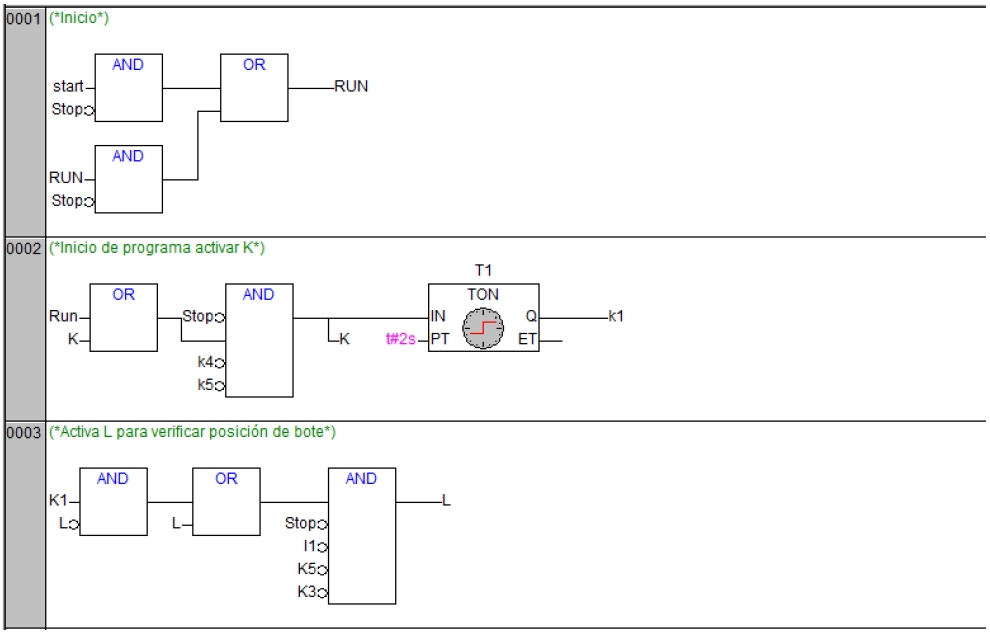


FIGURA B.1: Diagrama de bloques funcionales parte 1

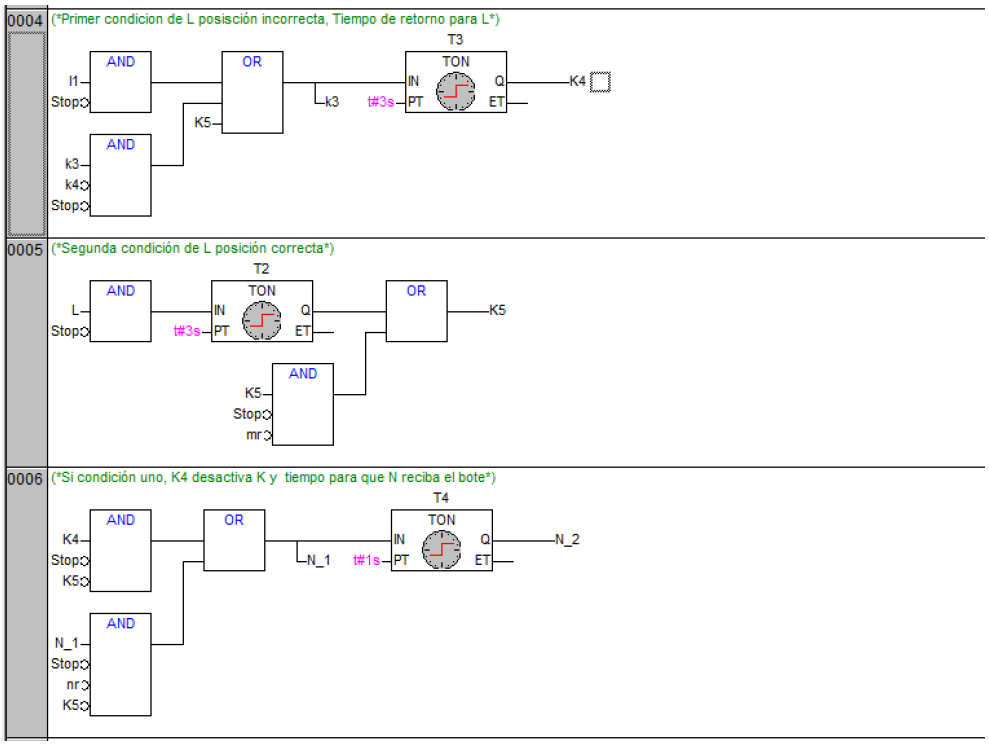


FIGURA B.2: Diagrama de bloques funconales parte 2

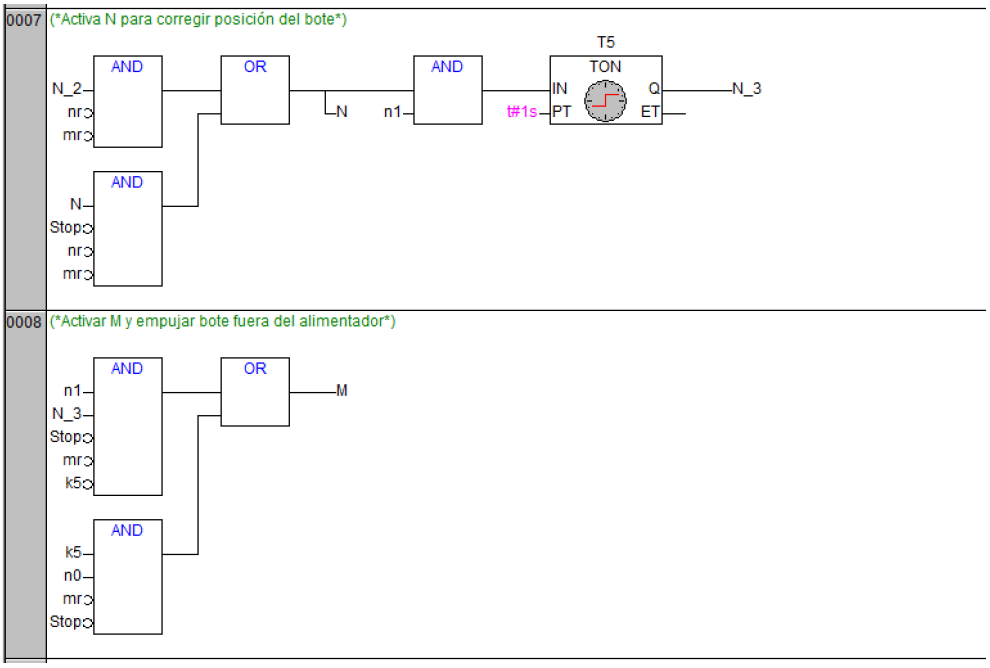


FIGURA B.3: Diagrama de bloques funcionales parte 3

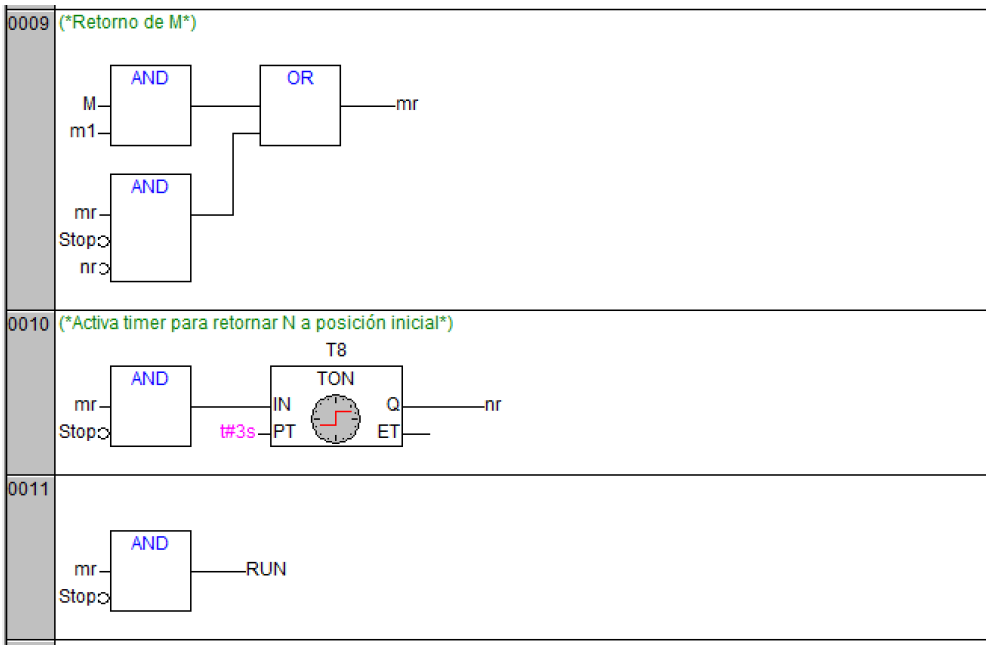


FIGURA B.4: Diagrama de bloques funcionales parte 4

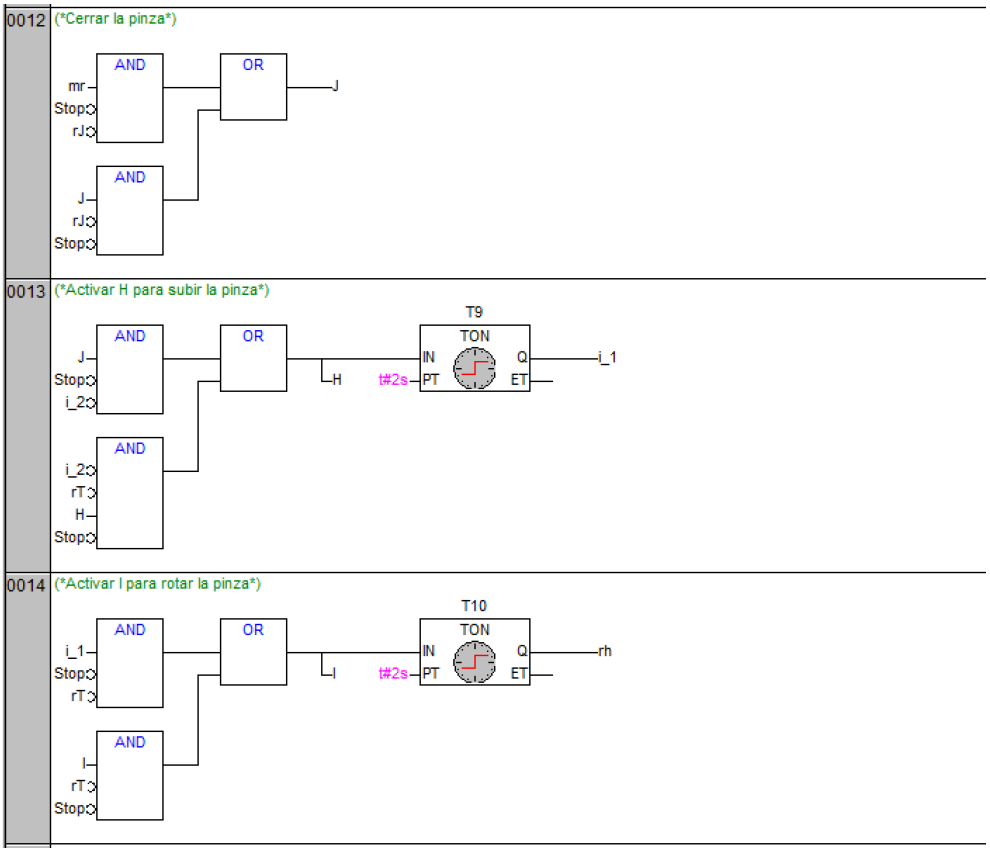


FIGURA B.5: Diagrama de bloques funcionales parte 5

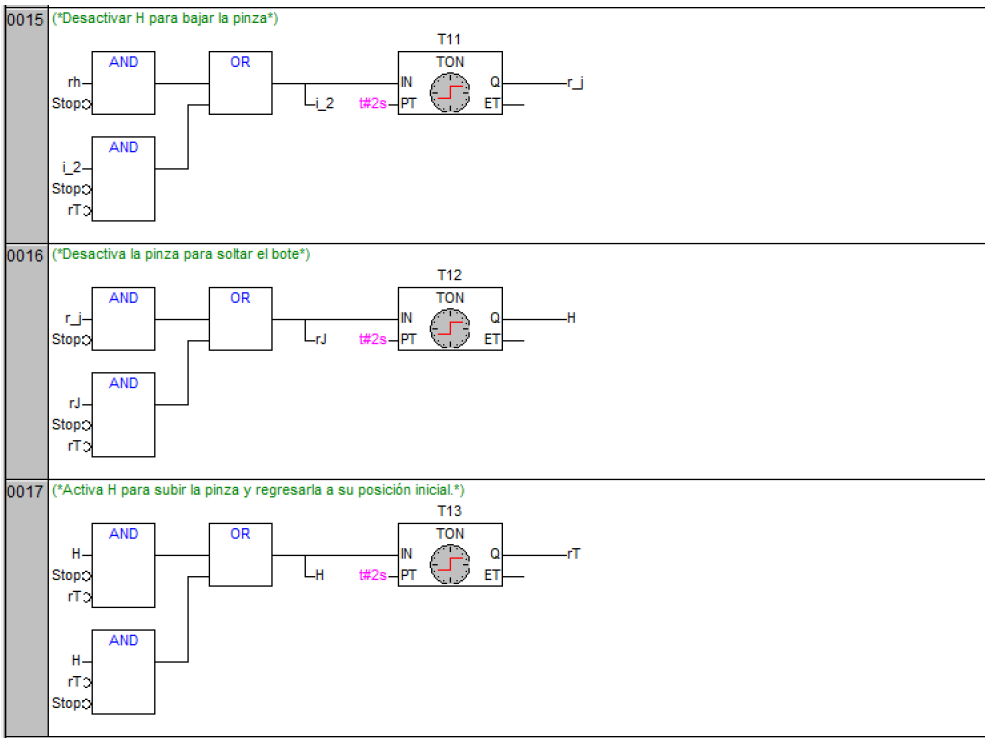


FIGURA B.6: Diagrama de bloques funcionales parte 6

C Código en Texto estructurado

En este apartado se anexan las imágenes correspondientes al código en lenguaje Texto estructurado realizado en CoDeSys.

```
0001 (*Inicio de programa*)
0002 IF start = TRUE  THEN;
0003       INICIO := TRUE;
0004 END_IF;
0005
0006  IF INICIO=TRUE THEN;
0007     K:=TRUE;
0008 END_IF;
0009
0010  IF K=TRUE THEN;
0011        TON1(IN :=  K, PT:=T#1s);
0012        L :=  TON1.Q;
0013 END_IF;
0014
0015  IF L = TRUE AND I1=FALSE THEN;
0016       TON2(IN := L, PT := T#3s);
0017       T2:=TON2.Q;
0018 END_IF;
0019
0020 (*Primer condición*)
0021  IF  I1=TRUE THEN;
0022        Ir := TRUE;
0023 END_IF;
0024 IF Ir=TRUE THEN;
0025    L :=  FALSE;
0026    TON3(IN := Ir, PT := T#4S);
0027    K1:= TON3.Q;
0028 END_IF;
0029
0030 IF K1=TRUE THEN;
0031    K := FALSE;
0032
0033     N:=TRUE;
0034 END_IF;
```

FIGURA C.1: Texto estructurado parte 1

```
0036 (*Corregir posición de bote*)
0037
0038 IF N=TRUE AND n1=TRUE THEN;
0039    TON5(IN:=n1, PT:=T#1S);
0040    M:= TON5.Q;
0041 END_IF;
0042
0043 IF m1=TRUE THEN;
0044    M:=FALSE;
0045    mr:=TRUE;
0046 END_IF;
0047
0048  IF mr = TRUE THEN;
0049    TON6(IN:=mr, PT:=T#2S);
0050    nr:=TON6.Q;
0051 END_IF;
0052
0053 IF nr=TRUE THEN;
0054    N:=FALSE;
0055 END_IF;
```

FIGURA C.2: Texto estructurado parte 2

```
0057 (*Segunda condición*)
0058 IF T2 = TRUE THEN;
0059       L:=FALSE;
0060      Ir2:= TRUE;
0061      Ir:= FALSE;
0062      T2:=FALSE;
0063    IF Ir2=TRUE  THEN;
0064      TON4(IN:=Ir2, PT:= T#1S);
0065       T4:=TON4.Q;
0066    END_IF;
0067    IF T4=TRUE THEN;
0068       K:=FALSE;
0069       Ir2:=FALSE;
0070      T4:=FALSE;
0071    END_IF;
0072 END_IF;
0073 (*Activar M*)
0074 IF TON2.Q=TRUE THEN;
0075    ma2:=TRUE;
0076 END_IF;
0077 IF ma2=TRUE THEN;
0078    TON7(IN:=ma2, PT:=T#2S);
0079    ma3:=TON7.Q;
0080 END_IF;
0081
0082 IF ma3=TRUE AND n0=TRUE THEN;
0083    M:=TRUE;
0084 END_IF;
0085
0086 IF m1=TRUE THEN;
0087    M:=FALSE;
0088 END_IF;
```

FIGURA C.3: Texto estructurado parte 3

```
0092 (*Transladar bote a plato central*)
0093 IF m1=TRUE THEN;
0094   GIN:=TRUE;
0095 END_IF;
0096 IF GIN=TRUE THEN;
0097   TON9(IN:=GIN, PT:=T#2S);
0098   J:=TON9.Q;
0099 END_IF;
0100 IF J=TRUE THEN;
0101   TON10(IN:=J, PT:=T#2S);
0102   H:=TON10.Q;
0103 END_IF;
0104 IF H=TRUE THEN;
0105   TON11(IN:=H, PT:=T#3S);
0106   I:=TON11.Q;
0107 END_IF;
0108 IF I=TRUE THEN;
0109   TON12(IN:=I, PT:=T#2S);
0110   hr:=TON12.Q;
0111 END_IF;
0112 IF hr=TRUE THEN;
0113   H:=FALSE;
0114   Th:=TRUE;
0115 END_IF;
0116 IF Th=TRUE THEN;
0117   TON13(IN:=Th, PT:=T#3S);
0118   rj:=TON13.Q;
0119 END_IF;
0120 IF rj=TRUE THEN;
0121   J:=FALSE;
0122 END_IF;
0123 IF rj=TRUE THEN;
0124   TON14(IN:=rj, PT:=T#1S);
0125   h1:=TON14.Q;
```

FIGURA C.4: Texto estructurado parte 4

D Esquema de conexiones en módulo

En este apartado se anexan el esquema de conexiones correspondiente al módulo CPX-CEC-C1.

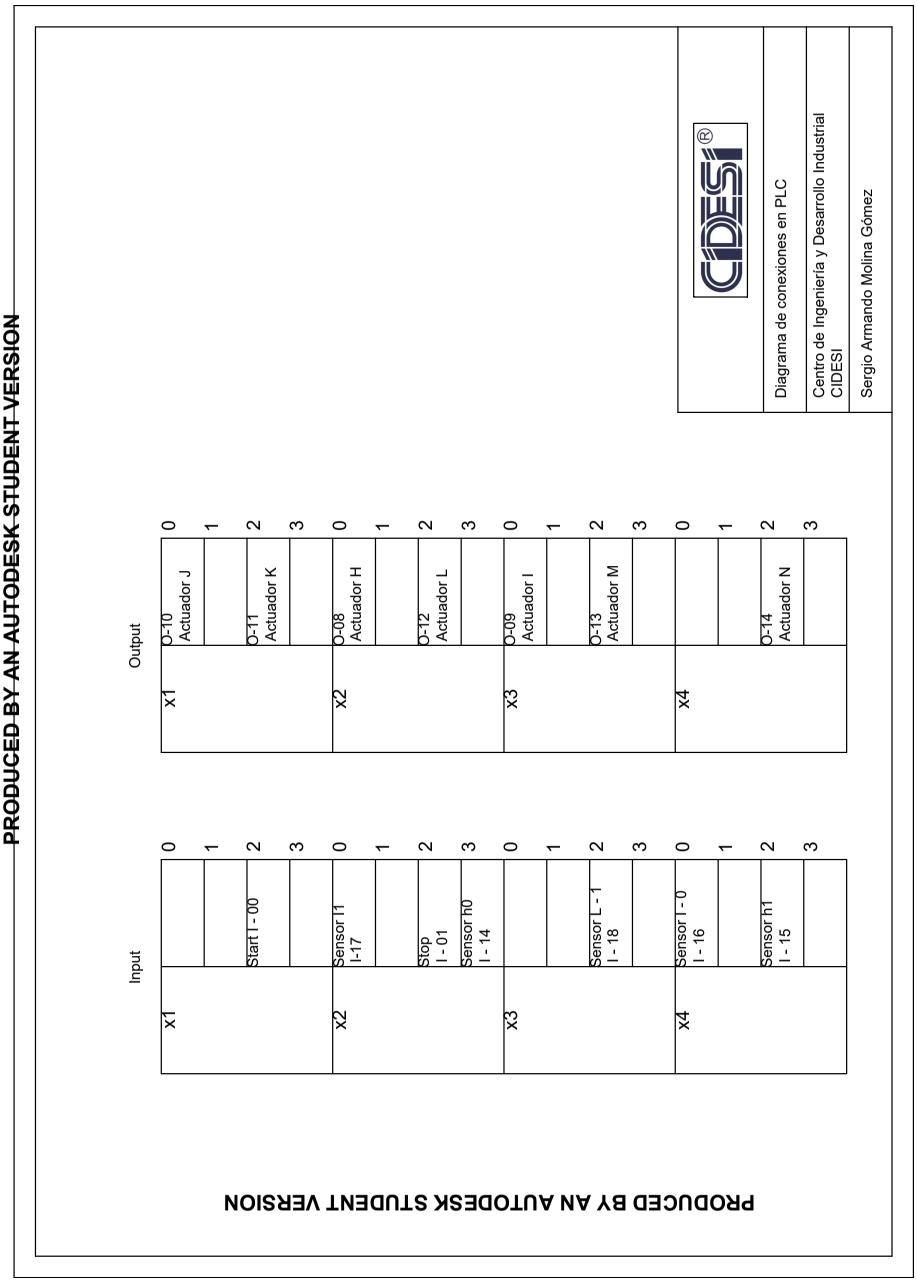


FIGURA D.1: Esquema de conexiones módulo CPX-CEC-C1

Bibliografía

Bolton, W. (2006). *Programmable logic Controllers*. cuarta edición. Elsevier.

Hanssen, Dag H. (2015). *Programmable logic controllers*. primer edición. Wiley.

Solutions, Smart Software (2003). «User Manual for PLC programming with CoDeSys 2.3». En:

training, SMC International. *IPC-202B*. SMC.

viuela, E. Blanco (2013). «Integrating Complex or Diverse Systems». En: