

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/336086650>

Problemas de Automatización Industrial

Book · September 2019

CITATIONS

0

READS

445

3 authors:



José Carlos Castillo
University Carlos III de Madrid

84 PUBLICATIONS 1,049 CITATIONS

[SEE PROFILE](#)



João Valente
Wageningen University & Research

45 PUBLICATIONS 638 CITATIONS

[SEE PROFILE](#)



Dolores Blanco
University Carlos III de Madrid

92 PUBLICATIONS 880 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



When UAV meets the Classroom [View project](#)



Aerial Coverage Path Planning [View project](#)

PROBLEMAS DE AUTOMATIZACIÓN INDUSTRIAL

**José Carlos Castillo, João Valente y María
Dolores Blanco**

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y
AUTOMÁTICA
UNIVERSIDAD CARLOS III DE MADRID

PROBLEMAS DE AUTOMATIZACIÓN INDUSTRIAL

Autor: José Carlos Castillo, João Valente y María Dolores Blanco

Leganés, Noviembre de 2016

Primera Edición

Agradecimientos

A todos los profesores que han pasado y pasarán por la asignatura Automatización Industrial I en la Universidad Carlos III de Madrid.

Índice general

Agradecimientos	I
1. Introducción	1
2. Configuración del autómata	5
2.1. El autómata TSX premium	6
2.2. Entorno de programación	6
2.2.1. Crear nuevo proyecto	8
2.2.2. Configuración	9
2.2.2.1. Ventana del configurador	10
2.2.3. Programación en lenguaje de contactos	12
2.2.3.1. Definición de variables	14
2.2.3.2. Bloques de función	19
2.2.4. Ejecución de programas	22
2.2.5. Visualización de la evolución de un programa	22
2.2.6. Documentación	24
3. Ejercicios de diagramas de estados	27
3.1. Preparado automático de bebidas	28
3.1.1. Solución	29
3.2. Modelado de una ventanilla de coche	32
3.2.1. Solución	33
3.3. Sistema de taponado de latas	36
3.3.1. Solución	38
3.4. Túnel de pintura de piezas	42
3.4.1. Solución	44
3.4.2. Versión optimizada	48
3.5. Sistema de llenado de bidones	49

3.5.1. Solución	51
4. Ejercicios de SFC	57
4.1. Sistema multi-robot para gestión de almacén I	58
4.1.1. Solución	60
4.2. Sistema multi-robot para gestión de almacén II	66
4.2.1. Solución	68
4.3. Sistema automático de fabricación de sidra	74
4.3.1. Solución	77
4.4. Sistema automático de fabricación de cerveza	83
4.4.1. Solución	86
5. Enlaces a los ficheros fuentes y vídeos	97

Índice de figuras

2.1.	Autómata empleado en este libro	7
2.2.	Ventana de configuración del procesador	8
2.3.	Ventana del navegador	9
2.4.	Ventana de configuración	10
2.5.	Catálogo de bastidores	12
2.6.	Catálogo de módulos	13
2.7.	Generación del proyecto	14
2.8.	Introducción de una nueva sección del programa	15
2.9.	Introducción de una nueva sección del programa	16
2.10.	Actividad propuesta	17
2.11.	Propiedades de un elemento	18
2.12.	Editor de datos	19
2.13.	Edición de las variables de un módulo	20
2.14.	Insertar un bloque Compare u Operate	21
2.15.	Asistente de Entrada FFB	21
2.16.	Asistente de entradas de función	21
2.17.	Secuencia de botones para ejecutar un programa	22
2.18.	Visualización de un programa en ejecución	23
2.19.	Nueva tabla de animación	24
2.20.	Tabla de animación	25
2.21.	Inclusión de un tema en la documentación	26
3.1.	Preparado automático de bebidas (Diagrama de estados) . . .	29
3.2.	Ventanilla de coche (Diagrama de estados)	33
3.3.	Sistema de taponado de latas	36
3.4.	Sistema de taponado de latas (Diagrama de estados)	38
3.5.	Túnel de pintura (Solución sin optimizar)	44
3.6.	Túnel de pintura (Solución optimizada)	48
3.7.	Esquema del sistema de llenado de bidones	49

3.8. Sistema de llenado de bidones (Diagrama de estados)	51
4.1. Sistema multi-robot para gestión de almacén I	59
4.2. Sistema multi-robot para gestión de almacén II	67
4.3. Sistema automático de fabricación de sidra	76
4.4. Sistema automático de fabricación de cerveza	85
5.1. Escanear para ir al repositorio.	97
5.2. Escanear para ver el videotutorial de diagramas de estados. . .	98
5.3. Escanear para ver el videotutorial de SFC.	98

Capítulo 1

Introducción

Según la Real Academia Española, se define Automática como la *ciencia que trata de sustituir en un proceso el operador humano por dispositivos mecánicos o electrónicos*. La Automática tiene por objetivo desarrollar dispositivos y procedimientos que permitan que un operador artificial lleve a cabo las mismas tareas físicas o mentales que realizan los operadores humanos con una mínima intervención de estos. Hoy en día la automatización de procesos se extiende a innumerables ámbitos de la vida cotidiana, no solo al campo de la producción industrial. Los transportes, la domótica, la medicina, e incluso las actividades de ocio, están plagados de sistemas automatizados. Es lógico por tanto que los planes de estudio de los diversos Grados de la rama de Ingeniería incluyan asignaturas relacionadas con la *Automatización*. Aunque el concepto es amplio y más aún sus aplicaciones, los conceptos básicos que nos permiten abordar la mayoría de los proyectos de automatización pueden entenderse si se conocen y entienden los sistemas de automatización de procesos industriales que actualmente presentan un alto grado de madurez y una amplísima implantación.

La *Orden CIN/351/2009, de 9 de febrero, por la que se establecen los requisitos para la verificación de los títulos universitarios oficiales que habiliten para el ejercicio de la profesión de Ingeniero Técnico Industrial*, determina que los planes de estudio deben incluir un módulo común a todos los grados de la rama industrial que permita a los alumnos adquirir como competencia *Conocimientos sobre los fundamentos de automatismos y métodos de control*; y, entre las competencias específicas de las tecnologías Eléctrica y Electrónica Industrial se especifican *Conocimientos de regulación automática y técnicas de control y su aplicación a la automatización industrial* y *Capacidad para diseñar sistemas de control y automatización industrial*.

En consecuencia, tanto desde el punto de vista normativo como desde las necesidades formativas de cualquier ingeniero, los conocimientos básicos en *Automatización Industrial* deben formar parte de los programas de estudio de los Grados de la rama industrial y así ocurre en la Universidad Carlos III, donde los alumnos los cursan como asignatura obligatoria.

El objetivo principal de esta asignatura es que el alumno conozca los elementos básicos de un sistema de automatización: sensores, como elemento de *percepción* de la situación de las variables de interés del sistema; actuadores, ejecutores de las acciones requeridas para que el sistema se comporte según las especificaciones; comunicaciones industriales; y, controladores, tecnologías que permiten automatizar el control y toma de decisiones en el lazo de realimentación. Aunque existen muy diversas tecnologías de control en

el entorno industrial, los Autómatas Programables (o PLC, por sus siglas en inglés) constituyen a día de hoy una solución eficaz y muy utilizada en muchos sectores industriales.

Para programar un PLC, como cualquier otro sistema programable, es importante comenzar por describir el comportamiento que deseamos que ejecute el sistema a través de un modelo. Lo más común en automatización industrial es el uso de modelos gráficos, entre otros, los diagramas de estado y los diagramas funcionales secuenciales (SFC) que serán utilizados en este libro. Estos últimos, tan extendidos en el entorno industrial que el software de la mayor parte de los autómatas es capaz de interpretarlos generando las instrucciones del programa a partir de esta representación gráfica.

Los autores de este libro de problemas han impartido la asignatura *Automatización Industrial* durante varios años en los diferentes Grados de la rama Industrial de la Universidad Carlos III de Madrid. Durante estos años se ha puesto de manifiesto la ausencia de materiales de apoyo más allá de los proporcionados como material de la asignatura. Este libro de ejercicios resueltos, junto con contenidos multimedia (video tutoriales), y los ficheros fuente con la programación de los PLC, pretende dar respuesta a esta necesidad proporcionando materiales de apoyo que contribuyan al refuerzo de los contenidos de la asignatura y faciliten un aprendizaje más flexible, que se adapte a las necesidades de cada alumno, permitiendo el trabajo en casa.

En este libro se presentan ejercicios básicos de automatización. El primer paso es el modelado de la dinámica de funcionamiento deseada para el sistema, a partir de este modelo se generan las instrucciones de programación que ejecutará el autómata. Como lenguaje de programación se ha elegido el Diagrama de Contactos (Ladder Diagram, LD). En el capítulo I, el modelado se resuelve por medio del concepto de Diagrama de Estados. A partir de este modelo abstracto del sistema se detallan las instrucciones de programación en LD que permiten al autómata reproducir la dinámica modelada. La metodología seguida en los problemas del capítulo 2 se basa en la representación de la dinámica deseada del sistema a controlar por medio de un Diagrama Funcional Secuencial, (SFC en sus siglas en inglés), herramienta de modelado estándar recogida en la norma *EN-61131*. La programación a partir de un SFC, consiste en: definir las acciones asociadas a cada etapa y los respectivos descriptores que determinan el tiempo de activación con respecto a la etapa a la que están asociadas; y, programar las instrucciones que gobiernan las transiciones, para esta programación se utilizarán fundamentalmente instrucciones en LD.

Dada la falta de estandarización del software de programación de autómatas, cuando se afronta la tarea de realizar un libro de problemas resueltos no queda más remedio que seleccionar una herramienta concreta. Para la resolución de los problemas que aparecen en este libro se ha utilizado la plataforma software UnityPro de Schneider Electric. Se ha seleccionado este software por ser el utilizado para la impartición de la asignatura *Automatización Industrial I* en la UC3M, por lo que se disponen de licencias de uso para los alumnos y profesores. Además, esta herramienta cumple con el estándar *EN-61131*, aspecto muy favorable desde el punto de vista docente ya que permitirá a los alumnos adaptarse de forma sencilla a otras plataformas de programación. Esperamos que este libro sea una herramienta útil para los alumnos y contribuya de forma eficaz a facilitar su formación en un área tan importante como la automatización.

Capítulo 2

Configuración del autómata

El objetivo de este capítulo es conocer el autómata programable que se utilizarán este libro de ejercicios. En el caso del capítulo 3 se utilizará el software en modo simulado, por lo que se escogerá una configuración de autómata que no tiene por qué corresponderse con un sistema real, aunque el funcionamiento será el mismo que veremos en el presente capítulo.

En primer lugar se mostrarán los distintos componentes que forman el autómata y a continuación como se programa y como se ejecutan los programas.

2.1. El autómata TSX premium

El autómata real que se usará en este libro de ejercicios se muestra en la Figura 2.1, y está formado por los siguientes componentes:

- Bastidor
- Fuente de alimentación
- Procesador
- Entradas digitales
- Salidas digitales
- Entradas analógicas
- Salidas analógicas
- Telefast
- Cable comunicación con PC

2.2. Entorno de programación

Las soluciones de los ejercicios siguen la norma *EN-61131*, por lo que se podrían implementar en diversos autómatas. Concretamente en este libro utilizaremos *Unity Pro XL* como entorno de programación ya que permite:

- Configurar el autómata,
- escribir el programa,
- transferirlo al autómata, y

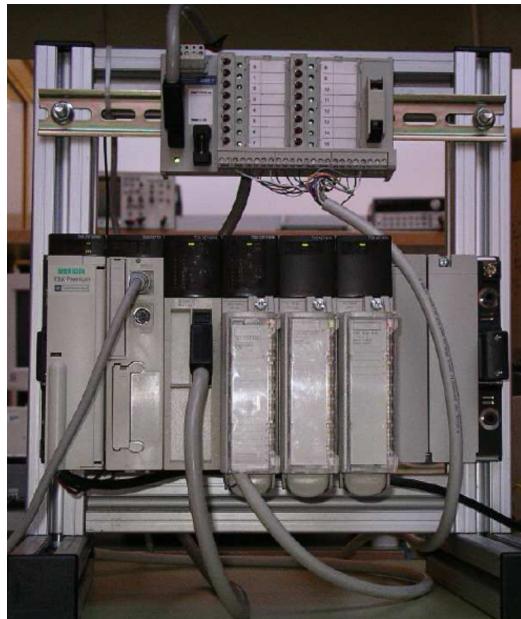


Figura 2.1: Autómata empleado en este libro

- comprobar el funcionamiento.

Unity Pro es un programa para la configuración, programación y depuración de proyectos de autómatas Modicon de Telemecanique. Modicon es el nombre de la familia de un conjunto de plataformas de automatismo complementarias. Las herramientas de Unity Pro sirven para todas las fases de un proyecto de automatización:

- Selección: Software “todo en uno” para programar y depurar una aplicación completa.
- Diseño: vistas funcionales, bibliotecas de funciones, datos estructurados, multi-tareas, 5 lenguajes IEC.
- Depuración: simulación del PLC, pantallas del operador y funciones completas de depuración.
- Operación: Proceso y sistemas de diagnóstico.
- Mantenimiento: Modificaciones múltiples en línea y en marcha y pantallas de diagnóstico.

- Apertura: Tecnología de hipervínculo, importación/exportación en formato XML, migración completa de aplicaciones realizadas con versiones anteriores (Concept, PL7).

2.2.1. Crear nuevo proyecto

Una vez lanzado el programa y seleccionada la opción **Nuevo** del menú **Fichero**, aparece una ventana como la mostrada en la Figura 2.2. En esta ventana se **elige el modelo de procesador** que tiene el autómata (debe seleccionarse según el hardware disponible).

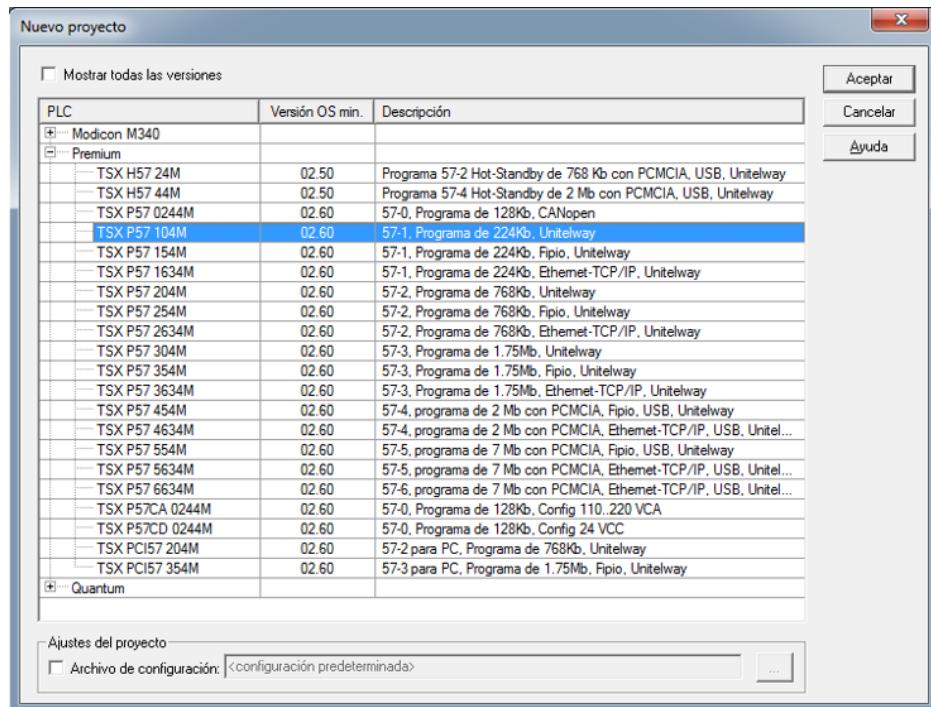


Figura 2.2: Ventana de configuración del procesador

Una vez pulsado el botón **Aceptar**, se genera automáticamente el entorno de trabajo para nuestra aplicación. Aparece la ventana de la Figura 2.3, que es la interfaz de usuario. La interfaz de usuario está compuesta por varias ventanas y barras de herramientas que se pueden organizar de forma libre.

Desde la ventana del **Navegador** que se muestra en la Figura 2.3, es posible seleccionar las herramientas de configuración (hardware y software),

programación, parametrización, prueba y documentación.

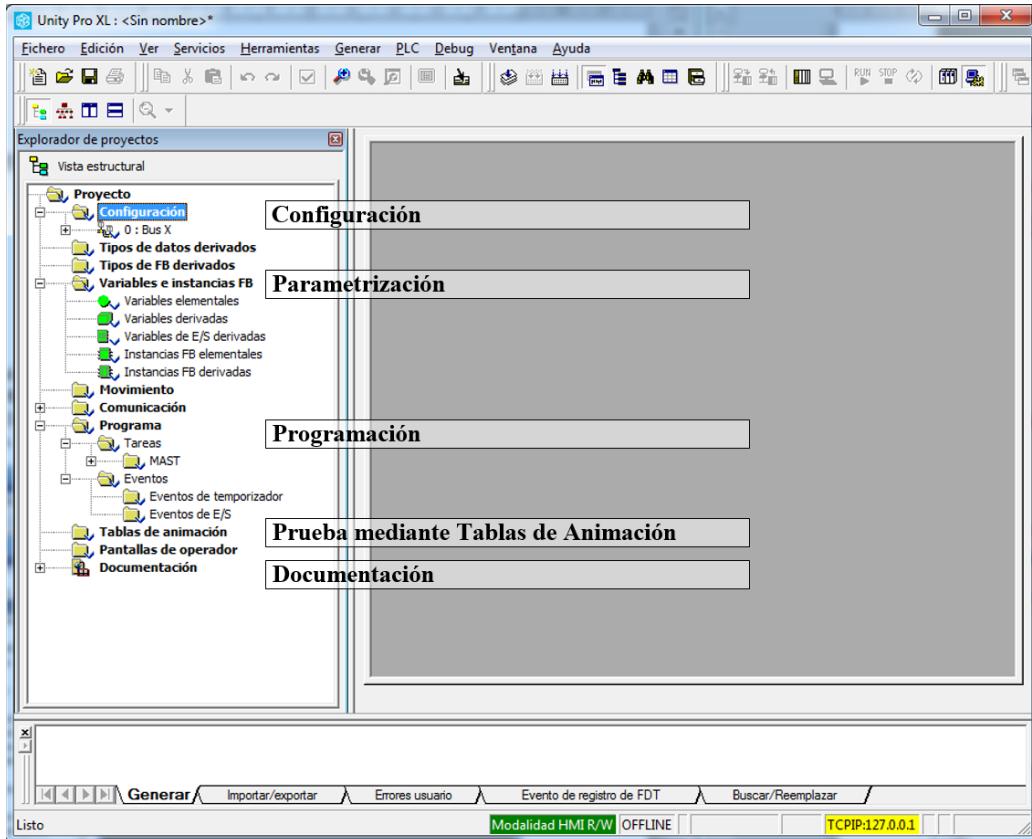


Figura 2.3: Ventana del navegodr

A la izquierda aparece el **Explorador de Proyectos** del que hablaremos posteriormente y que nos permitirá acceder a las herramientas de programación.

2.2.2. Configuración

El primer paso consiste en **configurar el autómata**. Para ello hay que realizar una configuración hardware y una configuración software.

Haciendo doble clic en **Configuración** (véase la anterior Figura 2.3) se despliega la correspondiente ventana de configuración. En esta ventana se **elige el bastidor y los módulos** que van colocados sobre él.

2.2.2.1. Ventana del configurador

La ventana del configurador, Figura 2.4, está dividida en dos ventanas:

- Ventana de catálogo: Desde esta ventana, es posible seleccionar un módulo e insertarlo en la representación gráfica de la configuración del PLC directamente mediante la función Arrastrar y soltar.
- Representación gráfica de la configuración del PLC

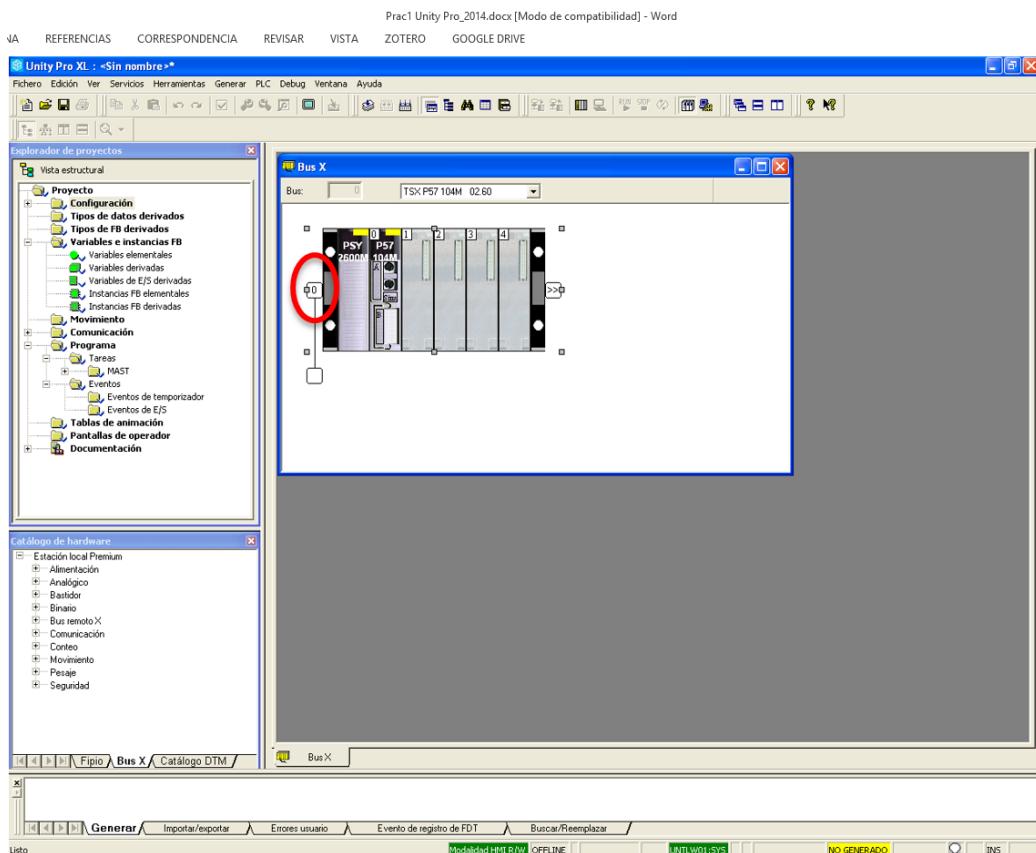


Figura 2.4: Ventana de configuración

En función de la posición del puntero del ratón, se abrirá uno de los siguientes menús contextuales (haciendo doble click):

- Si coloca el **puntero sobre el 0** que está a la izquierda del gráfico que representa el PLC, ver Figura 2.4, se configurará el **bastidor**, para

identificar el bastidor o rack de nuestro autómata consultar la identificación que aparece en la parte trasera del mismo, ver Figura 2.5.

- Si coloca el **puntero sobre el fondo**, podrá modificarse la unidad de la **CPU** y seleccionar diversos factores de zoom.
- Si coloca el **puntero sobre un módulo**, se accederá a las funciones de edición (borrar, copiar, mover), abrir la configuración del módulo para definir los parámetros específicos del módulo y visualizar las propiedades de E/S y la corriente total.
- Si coloca el **puntero sobre un slot vacío**, se podrá **insertar un módulo** del catálogo e insertar un módulo copiado previamente, incluyendo sus propiedades definidas. Para configurar todos los módulos y dejar una configuración adecuada se hará doble click en cada uno de los slots vacíos, y aparecerá un catálogo de módulos (binarios, analógicos, etc), ver Figura 2.6. Cada módulo del autómata tiene un rótulo con su denominación y será el que se vaya seleccionando en los distintos menús.

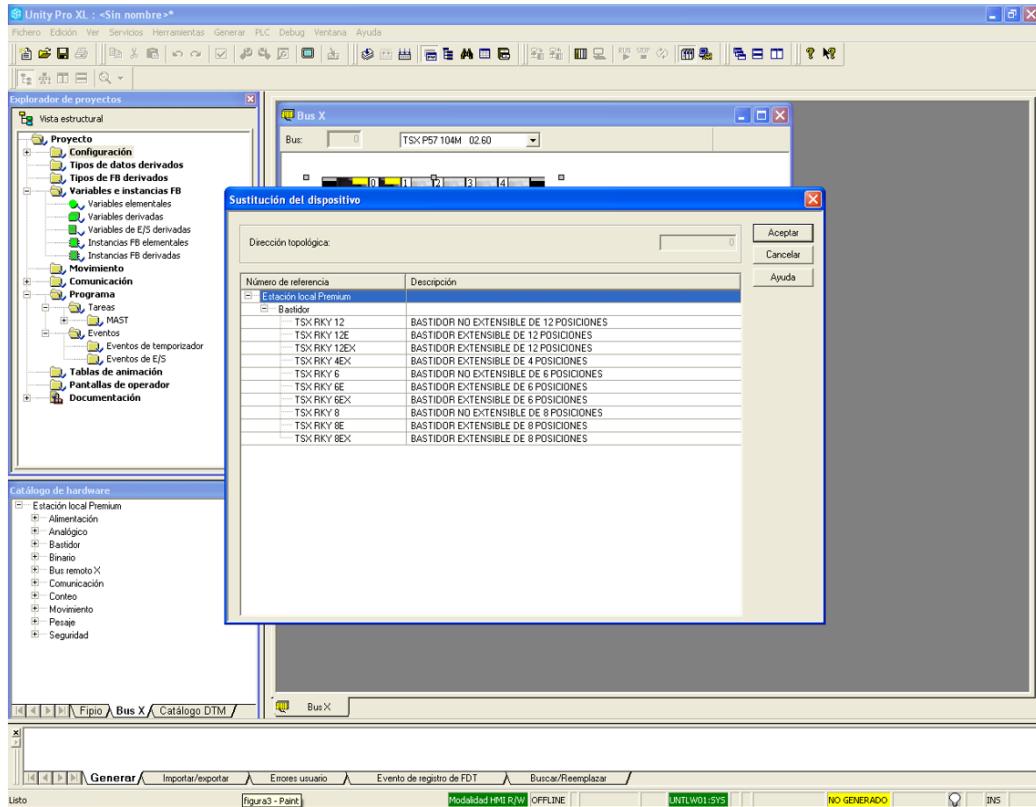


Figura 2.5: Catálogo de bastidores

Después de ajustar todos los valores es necesario **regenerar el proyecto** para aplicar los cambios realizados (menú **Generar → Regenerar todo el proyecto**) (ver Figura 2.7). Este paso es siempre necesario e imprescindible antes de volcar cualquier programa en el autómata.

En cualquier momento es posible (y muy recomendable) guardar nuestro proyecto utilizando los comandos habituales en cualquier aplicación Windows (menú **Archivo → Guardar como**, o botón de Guardar).

2.2.3. Programación en lenguaje de contactos

En esta primera práctica se va a programar en **lenguaje de contactos** (o Ladder, LD) introduciendo el código en una nueva **sección** de la tarea **MAST** (principal) del Programa. Para ello, utilizando de nuevo el Navegador, haremos click con el botón derecho en la carpeta de secciones de la

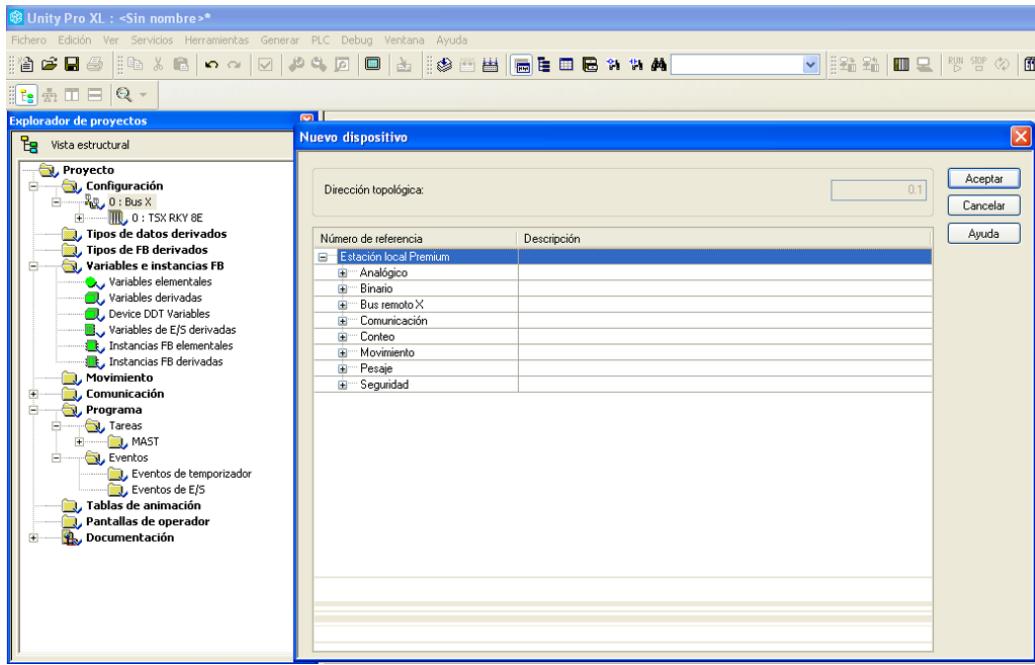


Figura 2.6: Catálogo de módulos

tarea MAST (véase la Figura 2.8). Se mostrará una ventana en la cual escribir un nombre identificativo para la sección y seleccionar el lenguaje de programación (LD en nuestro caso). Tras aceptar, se desplegará una ventana como la de la Figura 2.9. En la parte superior izquierda de dicha ventana está la paleta o barra de herramientas con todos los símbolos que se pueden introducir.

Siempre que modifiquemos nuestro programa, es importante **regenerar** el proyecto como se indicó anteriormente para detectar cualquier posible error de programación o configuración. El resultado de la regeneración se muestra en la ventana inferior de la aplicación Unity Pro XL.

Actividad 1

Introducir el código de la Figura 2.10 como una nueva sección de la tarea MAST.



Figura 2.7: Generación del proyecto

Para poner la variable asignada a cada elemento (bobina o contacto), se debe seleccionar el elemento correspondiente y haciendo después doble click o haciendo click con el botón derecho del ratón, se accede a la ventana mostrada en la Figura 2.11) en la que debe ponerse el nombre de la variable asociada al elemento.

2.2.3.1. Definición de variables

Una variable es una entidad de memoria de diferentes tipos cuyo contenido puede ser modificados por el programa durante la ejecución.

Una **variable no direccionada** es una variable que no está asociada a una referencia de memoria (no es posible conocer esta posición en la memoria). El sistema asigna automáticamente el emplazamiento de memoria de la instancia y puede cambiar cada vez que se genera la aplicación. La instancia tiene como dirección un nombre (símbolo) que elige el usuario.

Una **variable direccionada** es una variable asociada a una referencia de memoria. Por ejemplo, se puede crear la variable presión_agua y asociarla con la palabra de la memoria %MW102, o crear la variable pulsador_arranque y asociarla con el bit de memoria %I0.1.2. Las variables presión_agua y pulsador_arranque son variables direccionadas. El emplazamiento de la memoria de la instancia es fijo, está predefinido y no cambia nunca. La instancia tiene como dirección un nombre (símbolo) que elige el usuario y una dirección topológica que define el fabricante, o bien únicamente la dirección topológica del fabricante.

A la hora de definir una variable, es necesario considerar el tipo de dato necesario para almacenar la variable. Si hablamos de entradas y salidas del autómata, o de las variables que definen los estados (cuando modelemos mediante diagramas de estados), generalmente trabajaremos con tipos binarios (EBOOL). Otro ejemplo podrían ser los contadores en modelado SFC, donde

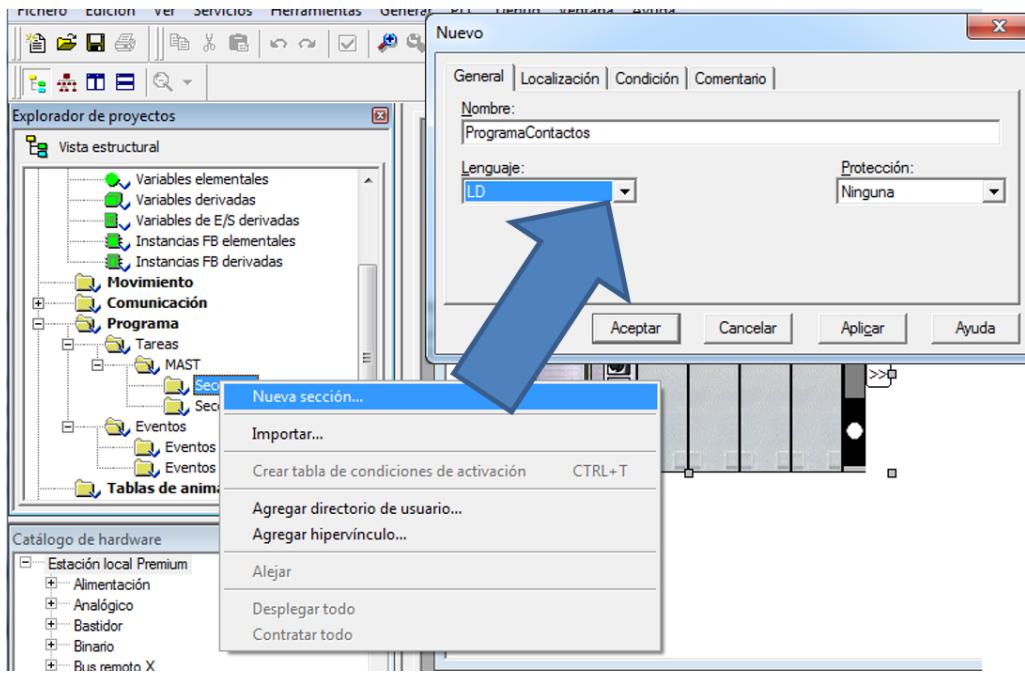


Figura 2.8: Introducción de una nueva sección del programa

un tipo de datos booleano, que tomará valores verdadero o falso no es suficiente, por lo que se deben escoger otros tipos de datos (ej. INT). La tabla 2.1 recoge algunos de los tipos de datos más habituales.

Tabla 2.1: Tipos de datos comunes

Tipo	Descripción	Formato (bits)	Valor por defecto
BOOL	Booleano	8	0 (False)
EBOOL	Booleano con detección de flancos y forzado	8	0 (false)
INT	Entero	16	0
DINT	Entero doble	32	0
UINT	Entero sin signo	32	0
UDINT	Entero doble sin signo	32	0
TIME	Entero doble sin signo	32	T=0s

Por otra parte, es necesario establecer la dirección de memoria de las variables direccionadas, dependiendo por ejemplo de si se trata de una variable de entrada, salida, constante o de otros tipos. Para las entradas y salidas, el patrón para definir una variable es el siguiente:

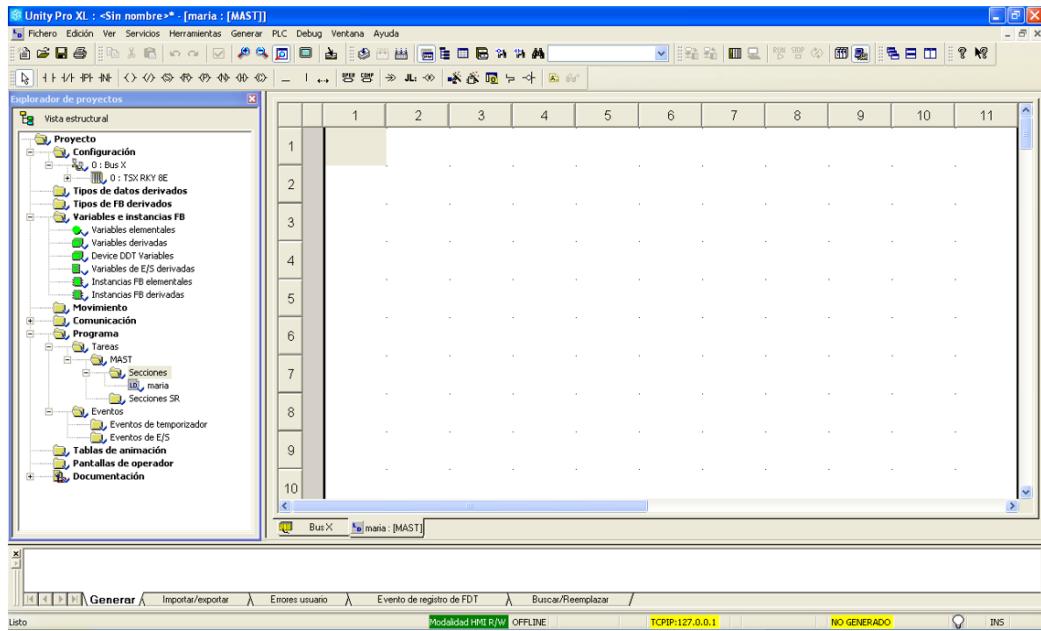


Figura 2.9: Introducción de una nueva sección del programa

$$\% < I|Q > < \text{bastidor} > . < \text{slot_modulo} > . < \text{canal} >$$

donde las variables de entrada tendrán en el primer carácter la letra I y las variables de salida la letra Q. En un PLC con varios bastidores, se asignará en el primer dígito el número del bastidor al que pertenece la entrada que se quiere direccionar. Si solo hay un bastidor, este será el número 0. Lo mismo ocurrirá para la posición del módulo de entrada o salida dentro del bastidor. En este libro hemos seleccionado módulos de entrada y salida digital con 16 valores, por lo que el número de canal estará en el rango 0 – 15. Ejemplos de direcciones de memoria para variables de entrada y salida en nuestros ejercicios podrían ser $\%I0,1,5$ y $\%Q0,2,3$. Otras variables como los contadores se definen con una nomenclatura algo diferente, siguiendo el siguiente patrón:

$$\% < M|K|S > < B|W|D|F > < \text{numero} >$$

Así, un contador se define como una palabra en memoria (Memory Word), por ejemplo $\%MW0$ y sería de tipo entero.



Figura 2.10: Actividad propuesta

Para editar (crear o dar nombre) a las variables, abrimos la **página de variables** a través del Explorador de Proyectos, haciendo un doble click o haciendo un click derecho y después elegir “abrir”. Nos aparecerá la pantalla que se muestra en la figura 2.12 (en nuestro caso, si todavía no hemos creado variables aparece vacío).

En Unity Pro existen 3 métodso para crear variables y asignarles un nombre o símbolo:

Creación de variables a partir de la tabla de variables

Para crear una variable hay que abrir el editor de datos y hacer doble click en la primera columna del Nombre donde podremos escribir el nombre de la variable. La segunda columna permite elegir el tipo de variable (ebool, int, ect.). Finalmente en la tercera columna especificamos la dirección topológica de la variable (%I0.1.0, %MW3, etc).

Se pueden cambiar las columnas que aparecen en esta pantalla haciendo un click derecho en los nombres de las columnas y después elegir en la lista cuales se desea ver y aceptar.

Una variable se caracteriza por:

- Su nombre;
- Su tipo;

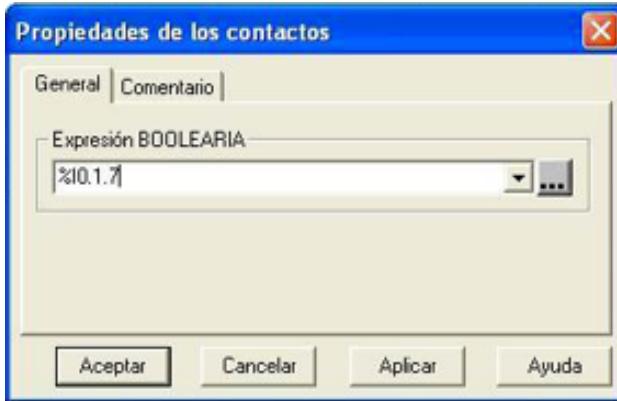


Figura 2.11: Propiedades de un elemento

- Su dirección (para las variables no direccionadas, no se asigna dirección);
- Su valor que es opcional;
- Su comentario que es opcional.

Una variable es el nombre que el programador da a una señal. El usuario puede elegir el nombre (símbolo) que permite acceder a una variable.

Nota: Las variables de entradas, salidas, y las %Mi son de tipo EBOOL.

Creación de variables desde los módulos del autómata

Desde los propios módulos es posible crearse las variables asociadas a las vías de un módulo (ver figura 2.13). Desde la pantalla de configuración del autómata (donde aparecen todos los módulos), haciendo un **doble clic** sobre un **módulo de señales digitales o analógicas**, nos aparece una ventana en la que seleccionamos el nombre del módulo elegido, su referencia, (en la parte superior de la columna que se encuentra a la izquierda) y después la pestaña **Objetos de E/S** (parte derecha de la ventana). Luego se ha de seleccionar la casilla correspondiente al tipo de señal (si son entradas sería %I, si son salidas %Q, etc). Una vez seleccionado el tipo de datos que se desea visualizar se ha de pulsar en **Actualizar cuadricula**. Aparecerá entonces la lista de todas las vías asociadas al tipo de señal que se ha seleccionado previamente.

Una vez escritos los textos en el apartado “Creación variable E/S”, pulsar el botón **Crear**.

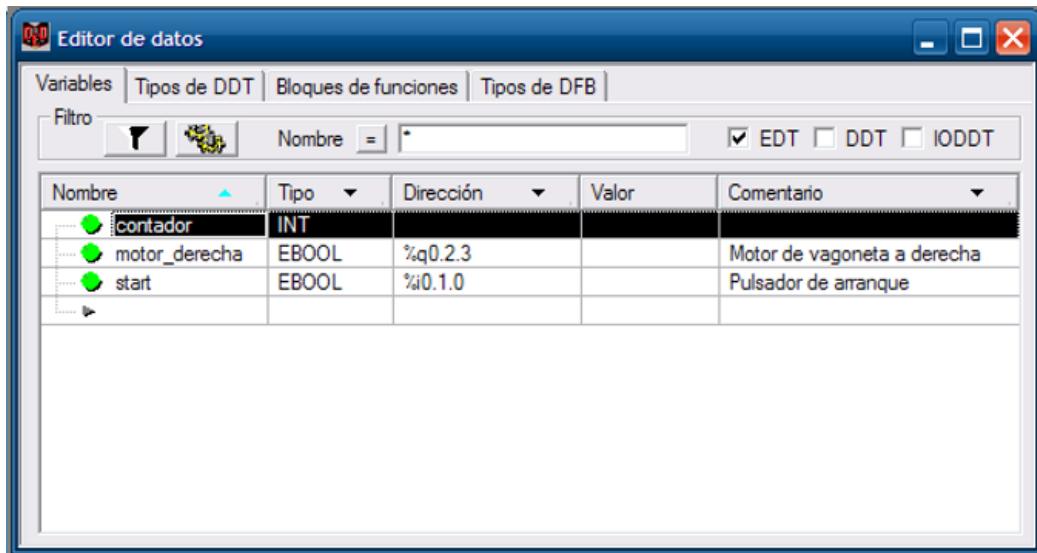


Figura 2.12: Editor de datos

Creación de variables online

La tercera forma de crear variables es a medida que vamos programando nuestro programa. Tanto en lenguaje de contactos como en SFC si llamamos o nos referimos a una variable por su nombre o símbolo y este es nuevo (no lo tiene guardado), el software nos preguntará si queremos crear esa variable nueva con ese nombre y su dirección.

2.2.3.2. Bloques de función

Para insertar un bloque **Compare** o un **Operate**, seleccionar el ícono correspondiente (ver figura 2.14).

Para insertar un bloque de función Contador o Temporizador utilizar el “Asistente de Entrada FFB” pulsando el ícono mostrado en la figura 2.15 o bien, en “Edición” seleccionar esa opción.

Nos aparecerá una pantalla en la que hemos de seleccionar el “Tipo de FFB”, seleccionamos el botón de puntos suspensivos y aparece la ventana de “Selección de tipos de FFB” donde desplegamos la carpeta de “Conjunto de librerías” tal y como se ve en la Figura 2.16.

En la librería **Base_Lib** podemos encontrar la carpeta de **Timers and**

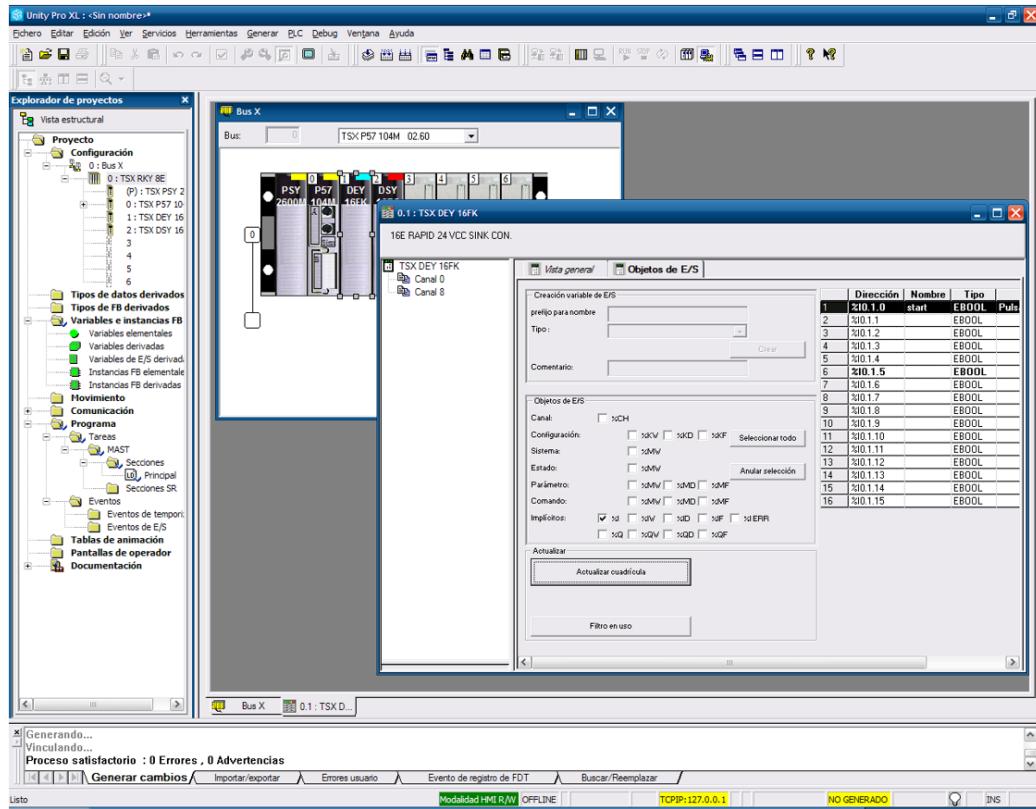


Figura 2.13: Edición de las variables de un módulo

Counters y en ella se encuentran los contadores (CTU) y temporizadores (TON).

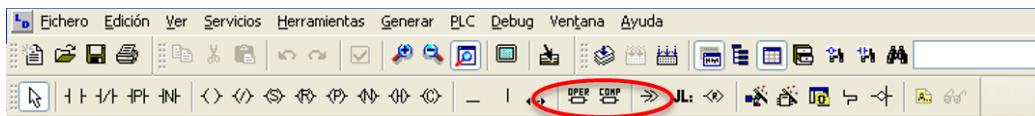


Figura 2.14: Insertar un bloque Compare u Operate

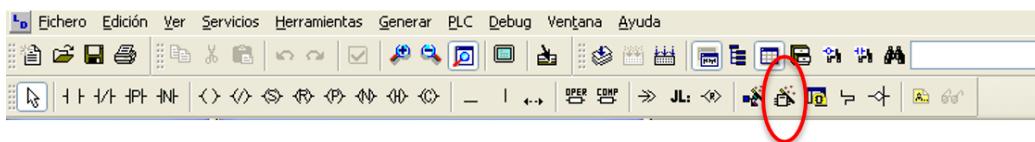


Figura 2.15: Asistente de Entrada FFB

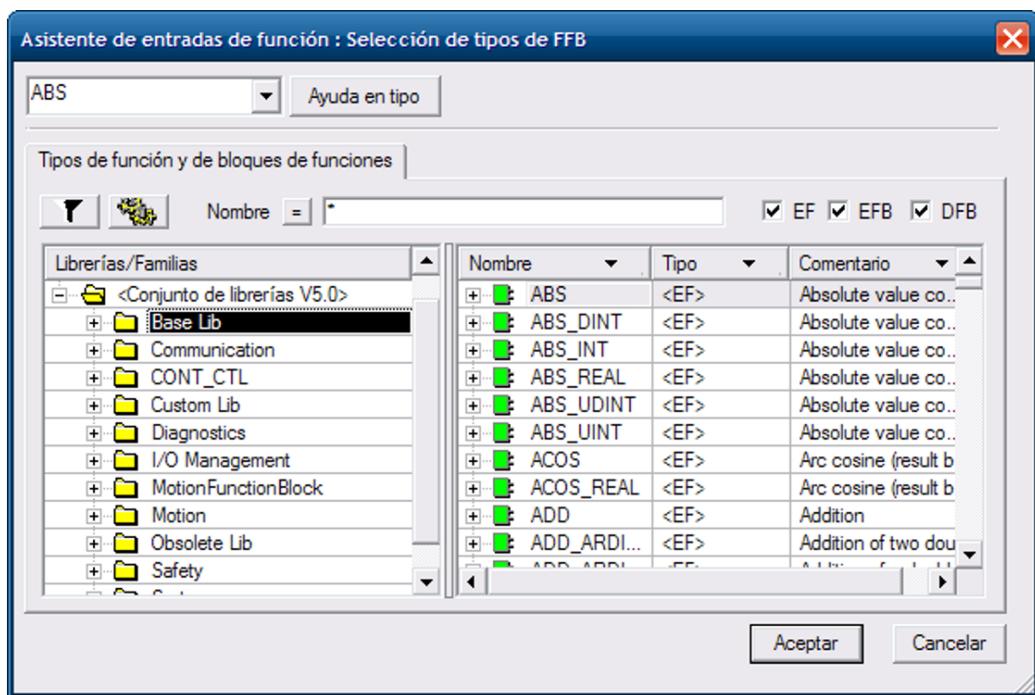


Figura 2.16: Asistente de entradas de función

2.2.4. Ejecución de programas

Para poder ejecutar un programa hay que realizar tres pasos: **conectar** el PC al Autómata (menú PLC - Conectar), **transferir** el programa al autómata (menú PLC - Transferir proyecto a PLC) y finalmente **ejecutar** el programa (menú PLC - Ejecutar/Detener). O bien, pulsando en orden los iconos de la Figura 2.17. En la barra de estado (borde inferior de la aplicación) se puede observar la situación del autómata.



Figura 2.17: Secuencia de botones para ejecutar un programa

Actividad 2

Ejecutar y probar en el autómata el código introducido en la actividad 1.

El objetivo es que, además de familiarizarse con el software, se analice el funcionamiento de los distintos tipos de contactos/bobinas y de las operaciones lógicas (AND y OR).

Diferencias entre:

- Contacto normalmente abierto y normalmente cerrado.
- Bobina abierta y una bobina con un Set.
- Funcionamiento de un bobina con un Set y otra con un Reset.
- Bobina abierta y bobina negada.
- Función AND y función OR.

2.2.5. Visualización de la evolución de un programa

Una vez cargado el programa, en modo conectado se puede observar en la pantalla del ordenador la evolución de las distintas variables. Las variables booleanas aparecen resaltadas en VERDE cuando están activas y en ROJO cuando no (ver Figura 2.18).

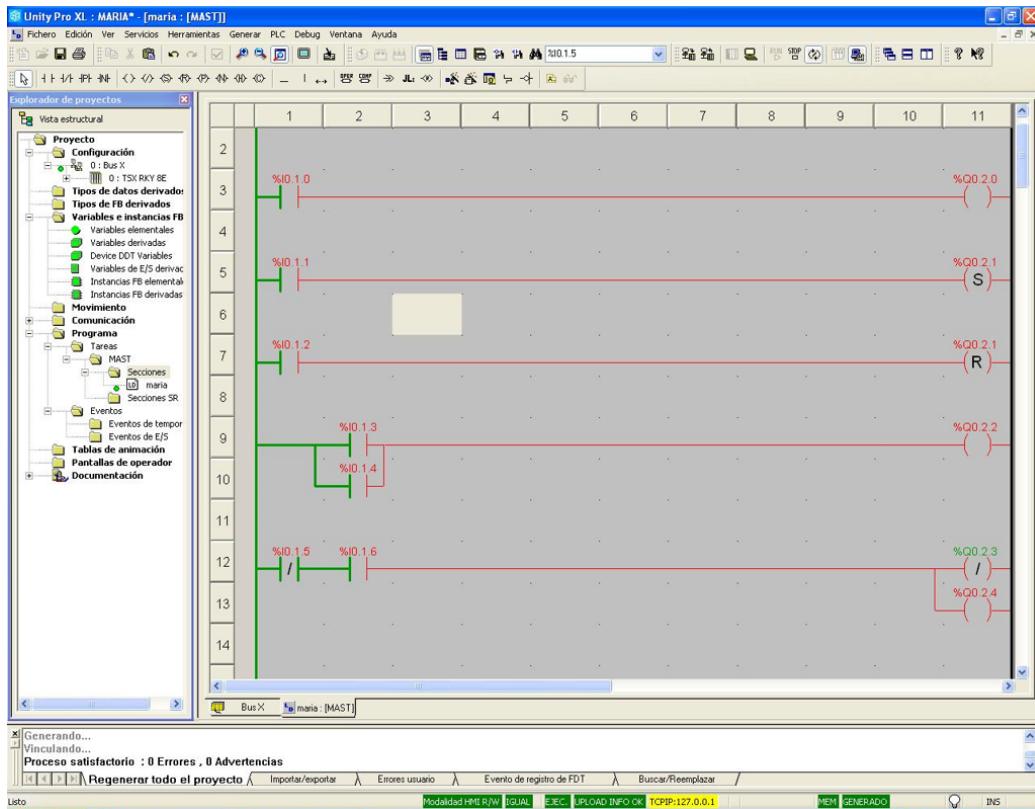


Figura 2.18: Visualización de un programa en ejecución

Otra opción es utilizar una tabla de animación en la que se puede observar de manera tabulada el valor numérico que adquieren las variables seleccionadas. Para agregar una nueva tabla de animación, procédase como se observa en la Figura 2.19, haciendo click con el botón derecho del ratón sobre la carpeta de Tablas de animación del Explorador de proyectos.

Tras asignar un nombre a la nueva tabla de animación, es posible introducir los identificadores de las variables cuyo estado se desea observar (ver Figura 2.20). Es muy importante desconectarse del autómata cada vez que se vaya a hacer un cambio por pequeño que sea.

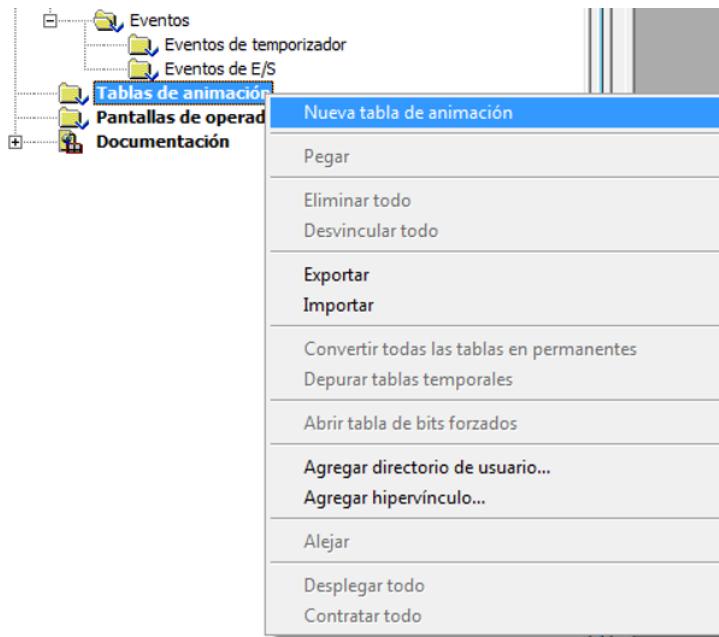


Figura 2.19: Nueva tabla de animación

Actividad 3

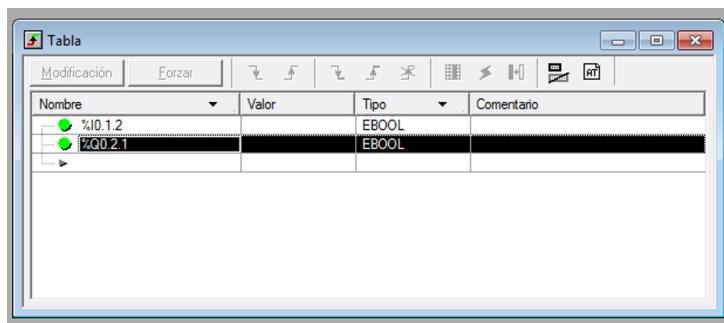
Generar una tabla de animación que muestre los valores de todas las entradas y salidas y ejecutar de nuevo el código introducido en la actividad 1.

Es importante recordar que antes de crear la tabla es necesario desconectarse del autómata y volver a regenerar el proyecto, para posteriormente transferir el programa al PLC de nuevo.

2.2.6. Documentación

Unity Pro permite crear la documentación del proyecto para imprimirla o guardarla en formato PDF (si se dispone una impresora virtual PDF). La estructura de árbol que presenta el Explorador de proyectos permite elegir temas para su impresión o visualización (presentación preliminar).

Desde el programa se puede generar documentación que muestre diferentes elementos como datos de configuración, listados de variables e incluso el



Nombre	Valor	Tipo	Comentario
%Q0.1.2		EBOOL	
%Q0.2.1		EBOOL	

Figura 2.20: Tabla de animación

propio programa. La secuencia de operaciones para generar la documentación es la siguiente:

1. Introducir los datos de **portada** haciendo doble click en el elemento Portada de árbol de documentación.
2. Introducir **comentarios** en el elemento Información General.
3. Seleccionar los elementos que se desea incluir en la documentación. Para ello haremos doble click en el elemento Documentación y en la ventana que se muestra, ver Figura 2.21 **seleccionaremos los elementos** deseados (opción Incluir Encabezamiento del menú contextual de cada elemento).
4. Para Imprimir debemos situarnos en Proyecto → botón derecho y seleccionar → **Imprimir**.

Actividad 4

Generar un pdf con la documentación del código introducido.

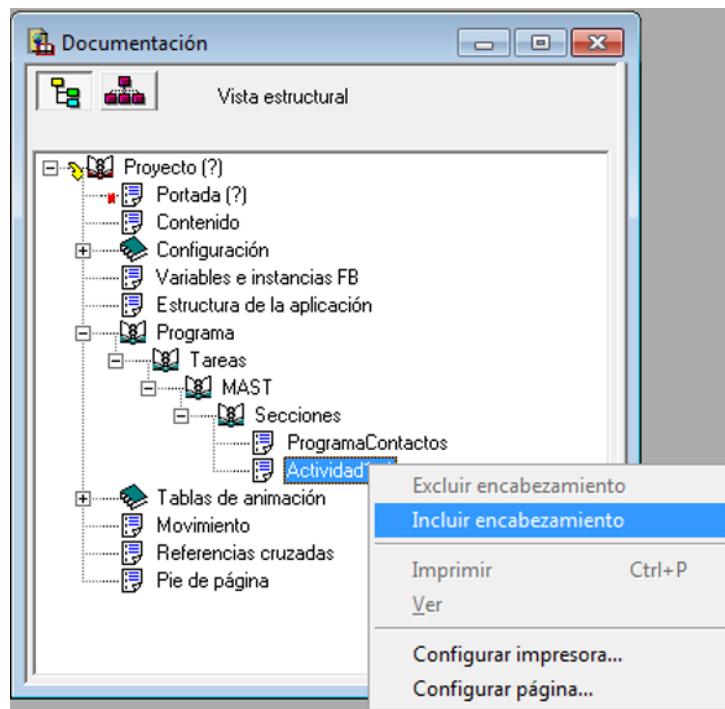


Figura 2.21: Inclusión de un tema en la documentación

Capítulo 3

Ejercicios de diagramas de estados

3.1. Preparado automático de bebidas

Modelar mediante un Diagrama de Estados un subproceso de preparado automático de bebidas. Cuando el usuario pulsa P, se pone en marcha la cinta (activando la señal MC) para traer un vaso a la posición de llenado, donde se detecta por el sensor SP. A la vez, se prepara la dosis de bebida correspondiente, para ello se abre la válvula V1 hasta que el depósito de dosificación se llene (lo cual se detecta con el sensor SL).

Cuando el vaso está en la posición de llenado y el depósito de dosificación lleno, se abre la válvula V2 para dejar caer el contenido del depósito en el vaso, este proceso dura 10 segundos. Después se pone la tapa, accionando el cilindro C.

El sistema vuelve a su situación de reposo cuando el usuario recoge su bebida, es decir, el vaso desaparece de la posición de llenado.

3.1.1. Solución

Este ejercicio es sumamente sencillo, aunque servirá para iniciarnos en el modelado mediante diagramas de estados y la programación en LD. La mayor dificultad reside en modelar dos procesos paralelos (mover la cinta y preparar la dosis de bebida) que no finalizan al mismo tiempo. Para ello, la iniciación de dichos procesos se ha modelado mediante el estado *Cinta Preparar*, en el que se activan las señales correspondientes a ambos procesos. Entonces pueden ocurrir dos cosas: que el vaso llegue a la posición de llenado (detectado por *SP*), o que se llene el depósito (detectado por *SL*). En el primer caso, se debe continuar preparando la dosis, de ahí la transición al estado *Preparar*, mientras que en el segundo caso, se debe continuar moviendo la cinta para posicionar el vaso (estado *Cinta*).

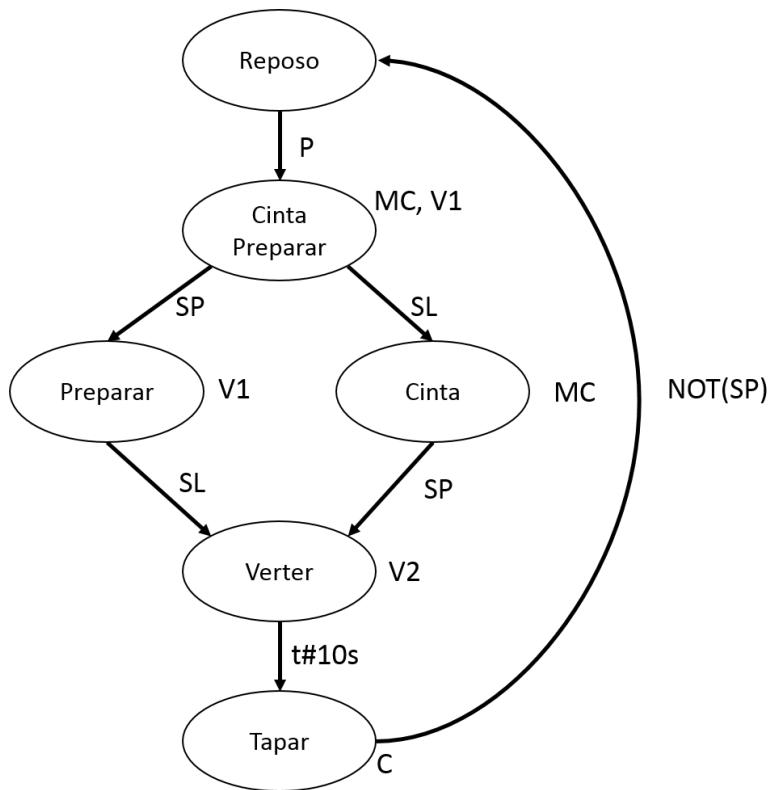


Figura 3.1: Preparado automático de bebidas (Diagrama de estados)

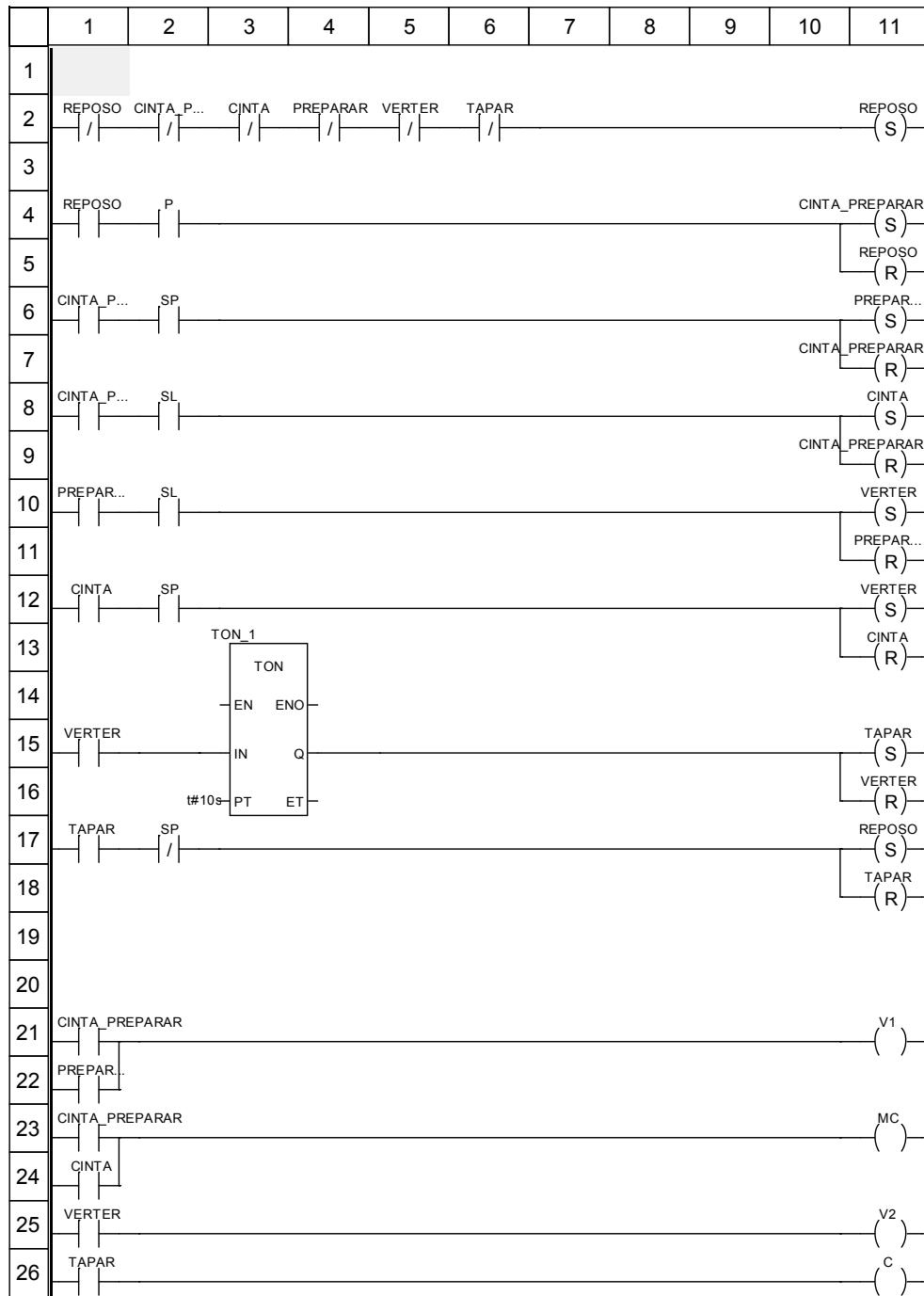
A continuación se muestran las variables asociadas a las entradas y salidas del sistema junto con las direcciones de memoria de los estados.

EBOOL

Nombre	Const	Dirección
C	NO	%Q0.2.3
CINTA	NO	%M3
CINTA_PREPARA	NO	%M1
R		
MC	NO	%Q0.2.0
P	NO	%I0.1.0
PREPARAR	NO	%M2
REPOSO	NO	%M0
SL	NO	%I0.1.2
SP	NO	%I0.1.1
TAPAR	NO	%M5
V1	NO	%Q0.2.1
V2	NO	%Q0.2.2
VERTER	NO	%M4

Una vez definidas las variables, podemos programar el diagrama de estados anterior en lenguaje de contactos.

PREPARADO_BEBIDAS : [MAST]



3.2. Modelado de una ventanilla de coche

Modelar mediante un Diagrama de Estados el sistema de control de una ventanilla de forma que responda a la siguiente dinámica:

- Se dispone de 2 pulsadores: uno para subir la ventanilla (PS) y otro para bajarla (PB).
- El sistema tiene 2 sensores: uno detecta que la ventanilla está completamente cerrada (VC) y otro que la ventanilla está completamente abierta (VA).
- La ventanilla puede moverse por medio de la activación de 2 motores diferentes: motor de subida (activado por la señal MS) y motor de bajada (activado por MB).
- Cuando se activa el pulsador correspondiente la ventanilla subirá o bajará. Si se mantiene pulsado más de 3 segundos, la ventanilla sube completamente o baja completamente. Si se suelta el pulsador antes de esos 3 segundos la ventanilla se para.

3.2.1. Solución

Aunque este ejercicio es sencillo, abordarlo por primera vez puede resultar confuso sobre todo porque no se proporciona una transición inicial desde el estado de reposo con un botón *MARCHA* o *INICIO* como sucede en la mayoría de los problemas. En este caso, desde el estado de *REPOSO* nuestra máquina de estados transitará mediante los pulsadores *PB* y *PS* teniendo en cuenta si la ventanilla está completamente subida o bajada mediante *VC* y *VA*, respectivamente. Una vez aclarado el arranque del sistema, podría ser difícil de abordar el último punto del enunciado, aquel que dice: *Si se mantiene pulsado más de 3 segundos, la ventanilla sube completamente o baja completamente. Si se suelta el pulsador antes de esos 3 segundos la ventanilla se para.* Esto se puede modelar añadiendo una transición desde los estados *SUBIR* y *BAJAR* donde se considere que si están pulsado *PS* y *PB* y transcurre un tiempo igual a tres segundos, transitamos a los estados de subir y bajar completamente, de los cuales se sale mediante los sensores *VC* y *VA*.

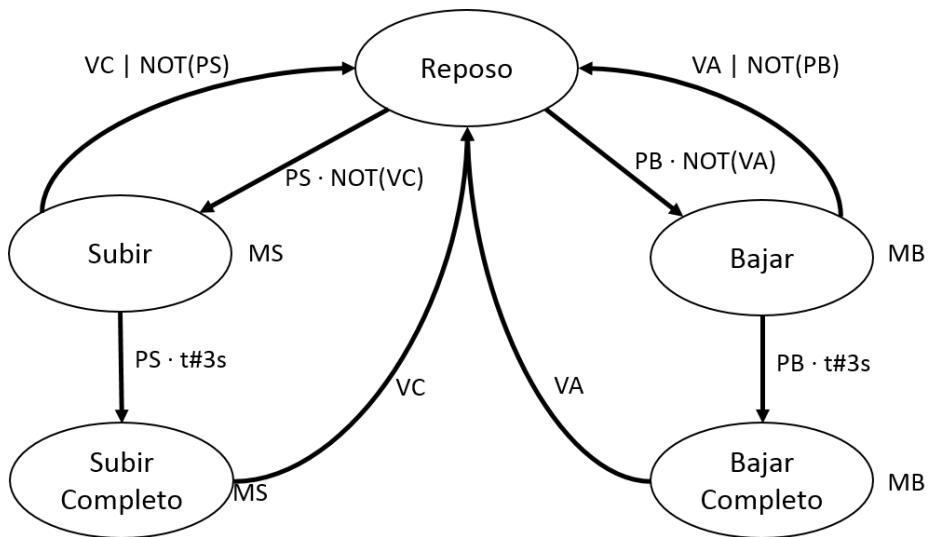


Figura 3.2: Ventanilla de coche (Diagrama de estados)

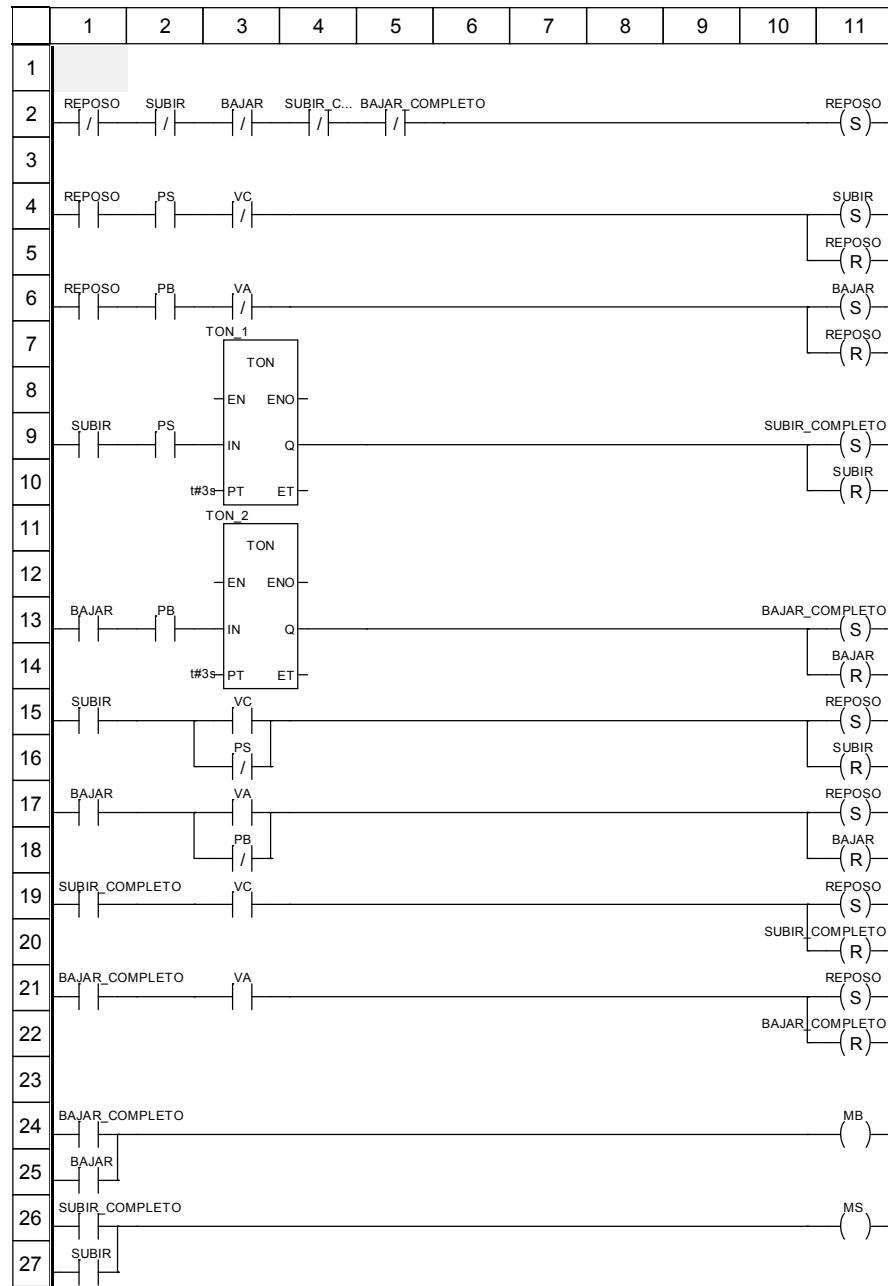
Las variables asociadas al sistema, así como sus direcciones de memoria asociadas se listan a continuación.

EBOOL

Nombre	Const	Dirección
BAJAR	NO	%M2
BAJAR_COMPLETO	NO	%M4
MB	NO	%Q0.2.1
MS	NO	%Q0.2.0
PB	NO	%I0.1.1
PS	NO	%I0.1.0
REPOSO	NO	%M0
SUBIR	NO	%M1
SUBIR_COMPLETO	NO	%M3
VA	NO	%I0.1.3
VC	NO	%I0.1.2

Una vez definidas las variables, se procede a realizar la programación del sistema en lenguaje de contactos.

ventanilla_coche : [MAST]



3.3. Sistema de taponado de latas

Se quiere controlar un sistema de taponado de latas como el de la Figura 3.3 donde las latas pretapadas llegarán a un émbolo que aplicará presión para finalizar el taponado. Después, las latas seguirán por la cinta hasta ser desalojadas.

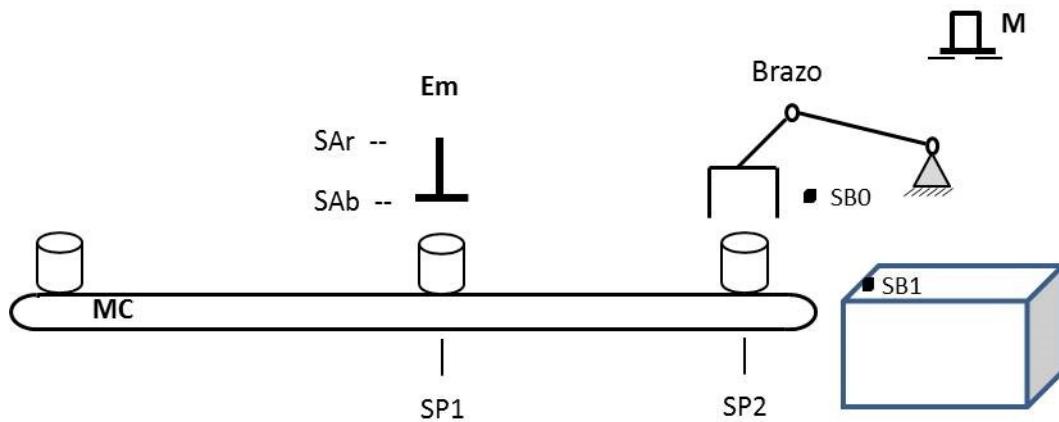


Figura 3.3: Sistema de taponado de latas

El funcionamiento detallado del sistema es el siguiente.

1. Cuando un operario presiona el botón de marcha (M) la cinta comienza a moverse hasta que el sensor SP1 detecta que hay una lata en la posición de tapado.
2. La cinta para y baja el émbolo que se encargará de aplicar presión (señal EmB). La lata estará tapada cuando el émbolo llegue a la posición SAb.
3. Una vez la lata está tapada la cinta comienza a moverse de nuevo y en paralelo el émbolo sube (señal EmS) para colocarse en la posición inicial (SAr).
4. Cuando la lata llegue a la posición SP2 y el émbolo haya subido del todo, un brazo robótico (Br0) llevará la lata tapada a la siguiente área de la cadena de producción. El sensor SB1 informará cuando la lata ha sido depositada. Entonces se ordena al brazo que vaya a la posición

inicial activando la señal Br1. El sensor SB0 indicará cuando el brazo está en la posición inicial.

5. Tras tapar 5 latas, el sistema entra en estado de reposo hasta que se vuelva a pulsar el botón de marcha.

No es necesario modelar lo que pasa con la lata al finalizar este subproceso. Se entiende que la distancia entre latas es suficientemente grande como para causar ambigüedad.

Se pide:

- Listar las entradas y salidas y dar nombre a las variables (los mismos dados en el enunciado) y asignarles las direcciones correspondientes en los módulos de E/S.
- Dibujar el Diagrama de estados de este automatismo.
- Programar en lenguaje de contactos usando los nombres de las variables (No las direcciones %I, %Q).

3.3.1. Solución

El siguiente diagrama de estados se ha generado a partir del planteamiento del problema.

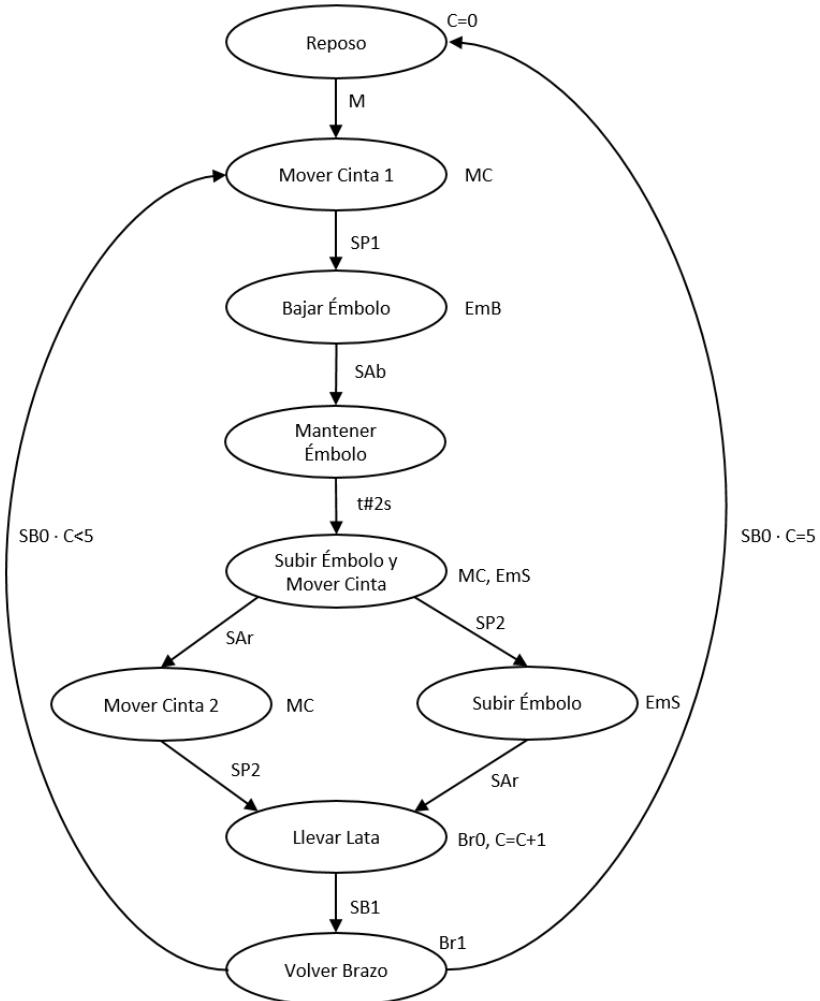


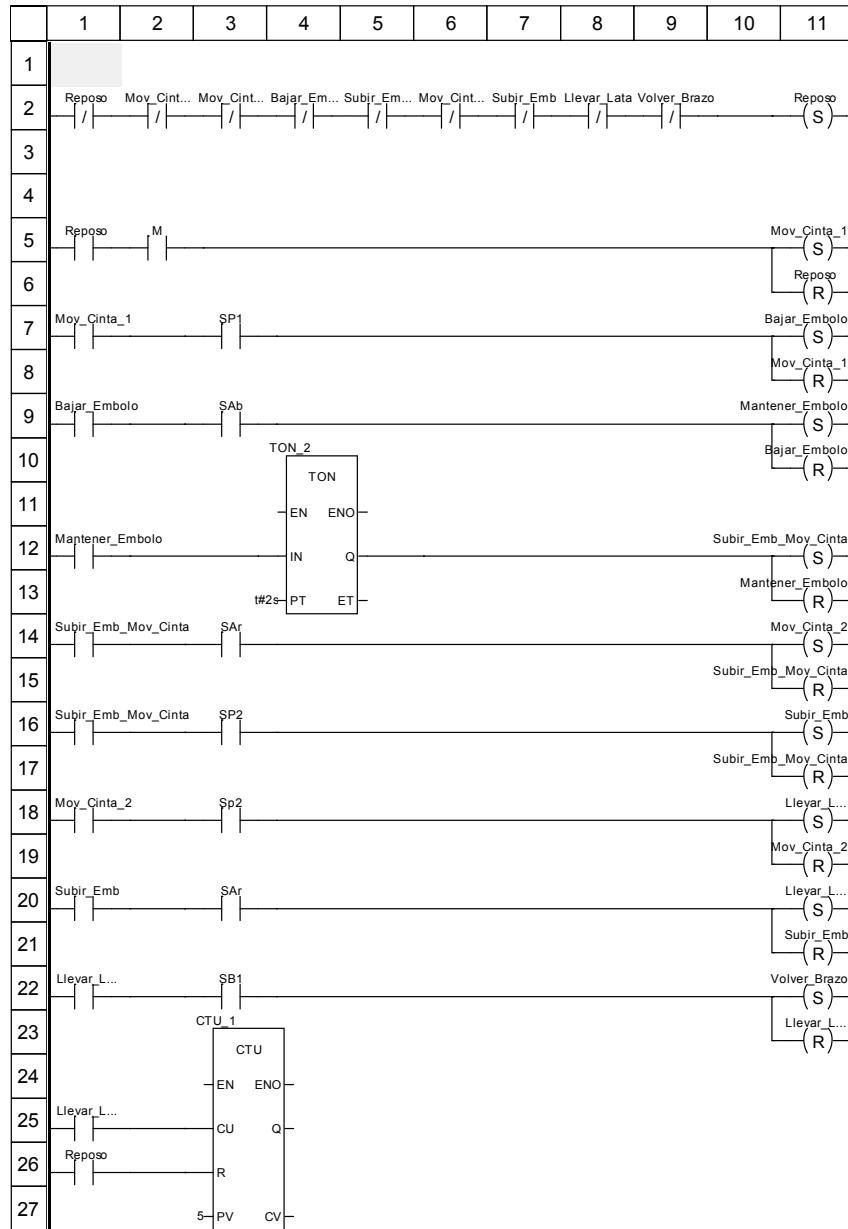
Figura 3.4: Sistema de taponado de latas (Diagrama de estados)

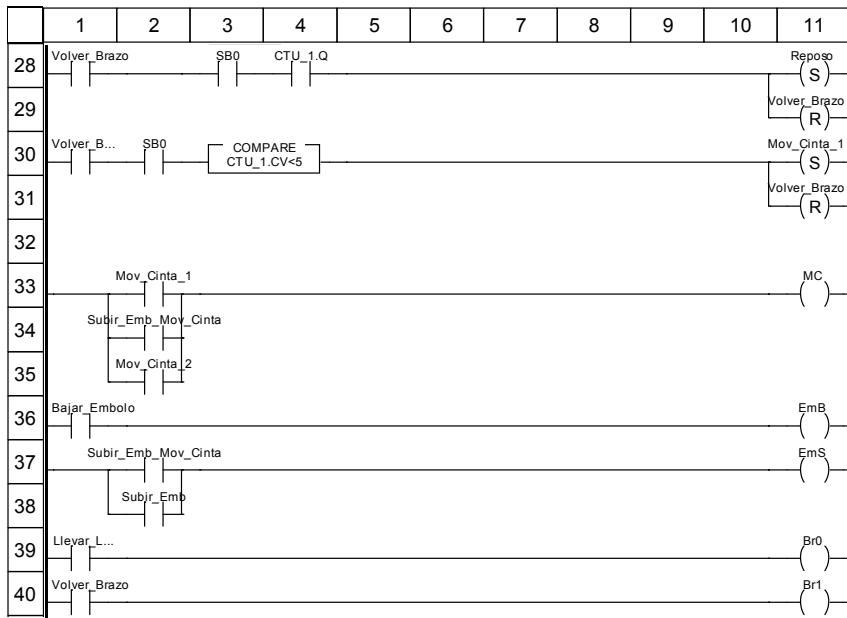
A continuación se muestra el código LD generado. Primero las variables y tras ellas el código.

EBOOL

Nombre	Const	Dirección
Bajar Embolo	NO	%M2
Br0	NO	%Q0.2.2
Br1	NO	%Q0.2.3
EmB	NO	%Q0.2.0
EmS	NO	%Q0.2.1
Llevar Lata	NO	%M6
M	NO	%I0.1.0
Mantener Embolo	NO	%M8
MC	NO	%Q0.2.4
Mov Cinta 1	NO	%M1
Mov Cinta 2	NO	%M4
Reposo	NO	%M0
SAb	NO	%I0.1.1
SAr	NO	%I0.1.2
SB0	NO	%I0.1.6
SB1	NO	%I0.1.5
SP1	NO	%I0.1.3
SP2	NO	%I0.1.4
Subir Emb	NO	%M5
Subir_Emb_Mov_Cinta	NO	%M3
Volver Brazo	NO	%M7

taponado_latas : [MAST]



**Etiquetas truncadas:**

Etiqueta	Posición(es)
Bajar_Embolo	(4, 2)
Llevar_Lata	(11, 18) (11, 20) (1, 22) (11, 23) (1, 25) (1, 39)
Mov_Cinta_1	(2, 2)
Mov_Cinta_2	(3, 2) (6, 2)
Subir_Emb_Mov_Cinta	(5, 2)
Volver_Brazo	(1, 30)

3.4. Túnel de pintura de piezas

Una fábrica pretende instalar un túnel de pintura de piezas para coches. El funcionamiento del sistema y sus componentes viene descrito a continuación:

Después de que un operario presione el botón de INICIO, un brazo robótico, R1, colocará las piezas a pintar sobre una cinta transportadora, C, siempre y cuando la cinta no contenga ninguna pieza. La cinta dispone de cuatro sensores (S1, S2, S3 y S4) que detectan si hay un objeto en distintos lugares de la cinta. El sistema funciona del modo siguiente:

- La cinta, inicialmente parada, no se pondrá en marcha hasta que el brazo robótico, R1 haya colocado la pieza a pintar. Cuando la pieza esté colocada en posición, el brazo robótico emitirá una señal, SR1, al autómata. Además, el sensor S1 detectará que la pieza está efectivamente en la posición correcta.
- Tras esto, la cinta comenzará a moverse hasta que el sensor S2 indique que hay una pieza lista para ser pintada. Un subsistema, AP, procederá a aplicar una capa de pintura a la pieza durante un minuto.
- Despues, la cinta se vuelve a poner en marcha para llevar la pieza a un horno de secado. El sensor S3 detecta que la pieza ha llegado a la posición de secado y se activa el subsistema HS durante 5 minutos.
- Para finalizar el proceso, la cinta se mueve nuevamente hasta que el sensor S4 informa de que hay una pieza en el área de inspección, donde un operario comprobará que la calidad de la pintura es óptima. Si la pieza tiene la calidad requerida, el operario pulsará un botón, B1, y un brazo robótico, R2, retirará la pieza en 30 segundos. Tras esto, el proceso de pintado se repetirá para una nueva pieza. En el caso de que haya imperfecciones en la pintura de la pieza, el operario presionará el botón B2 y R2 llevará la pieza al área de rectificado, donde el subsistema AR retirará la pintura. El área de rectificado tiene un sensor de presión P para detectar cuando hay una pieza en espera y el robot emitirá la señal SR2 al autómata para indicar que la pieza está en posición. El proceso de rectificado dura 2 minutos y tras esto R1 volverá a poner la pieza en el área de recepción de piezas al principio de la cinta.

Cada 100 piezas, el sistema entra en modo mantenimiento y una señal luminosa se activará cada 0,5 segundos hasta que un operario presione el botón SEGUIR, tras comprobar que todo está bien, con lo que R1 vuelve a

colocar piezas para pintar. Si por el contrario, el operario encuentra algún problema, presionará el botón PARAR y el sistema volverá al estado inicial.

Se pide:

- Listar las entradas y salidas y dar nombre a las variables (los mismos dados en el enunciado) y asignarles las direcciones correspondientes en los módulos de E/S.
- Dibujar el Diagrama de estados de este automatismo.
- Programar en lenguaje de contactos usando los nombres de las variables (No las direcciones %I, %Q).

3.4.1. Solución

El siguiente diagrama de estados se ha generado a partir del planteamiento del problema. El parpadeo de la señal luminosa se ha implementando usando un estado, *LUZ*, aparte del *MANTENIMIENTO*. En este caso hay que tener en cuenta que las transiciones para salir del estado de mantenimiento, *PARAR* y *SEGUIR*, hay que incluirlas también en el estado *LUZ*. Para iniciar la ejecución del sistema hay que tener en cuenta que aparte de la señal del botón *INICIO* no debe haber ningún objeto en la cinta. Por eso se crea una función lógica que tiene en cuenta dichas señales: *INICIO&NOT(S1)&NOT(S2)&NOT(S3)&NOT(S4)*.

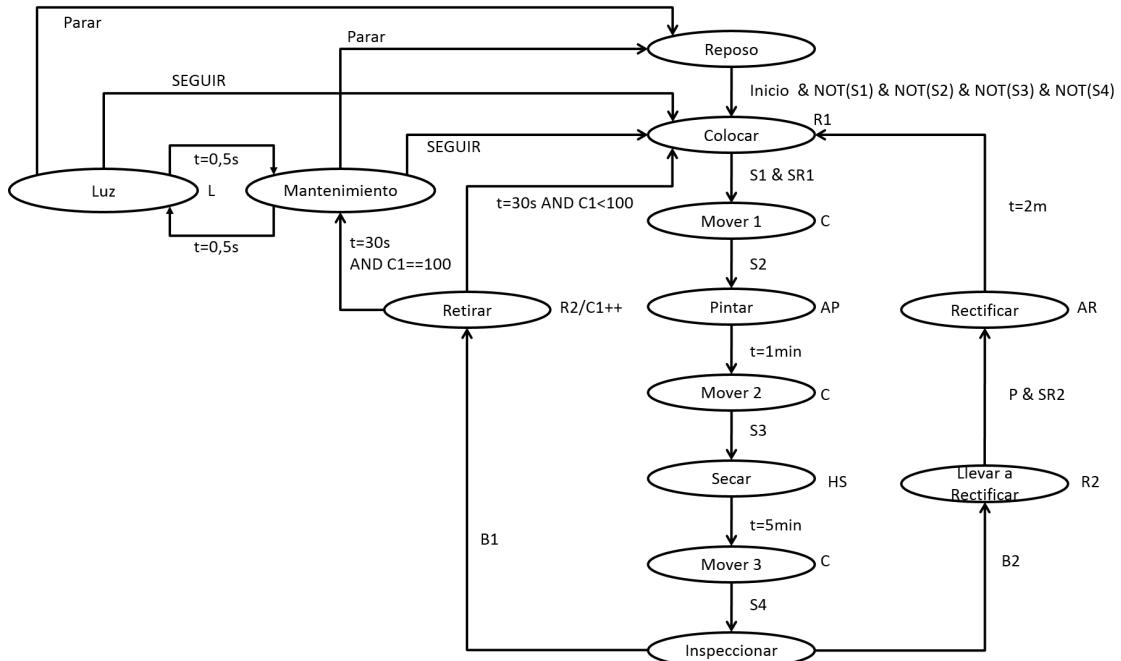


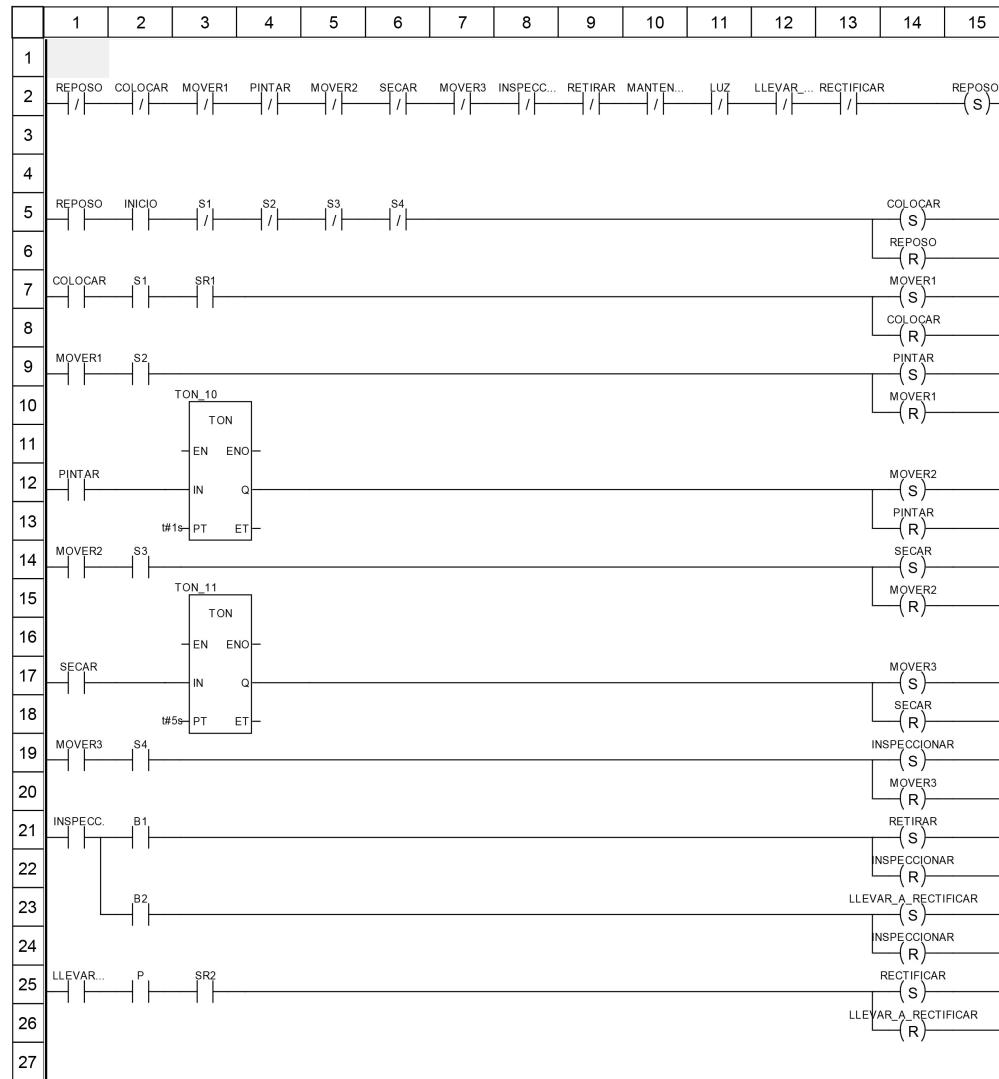
Figura 3.5: Túnel de pintura (Solución sin optimizar)

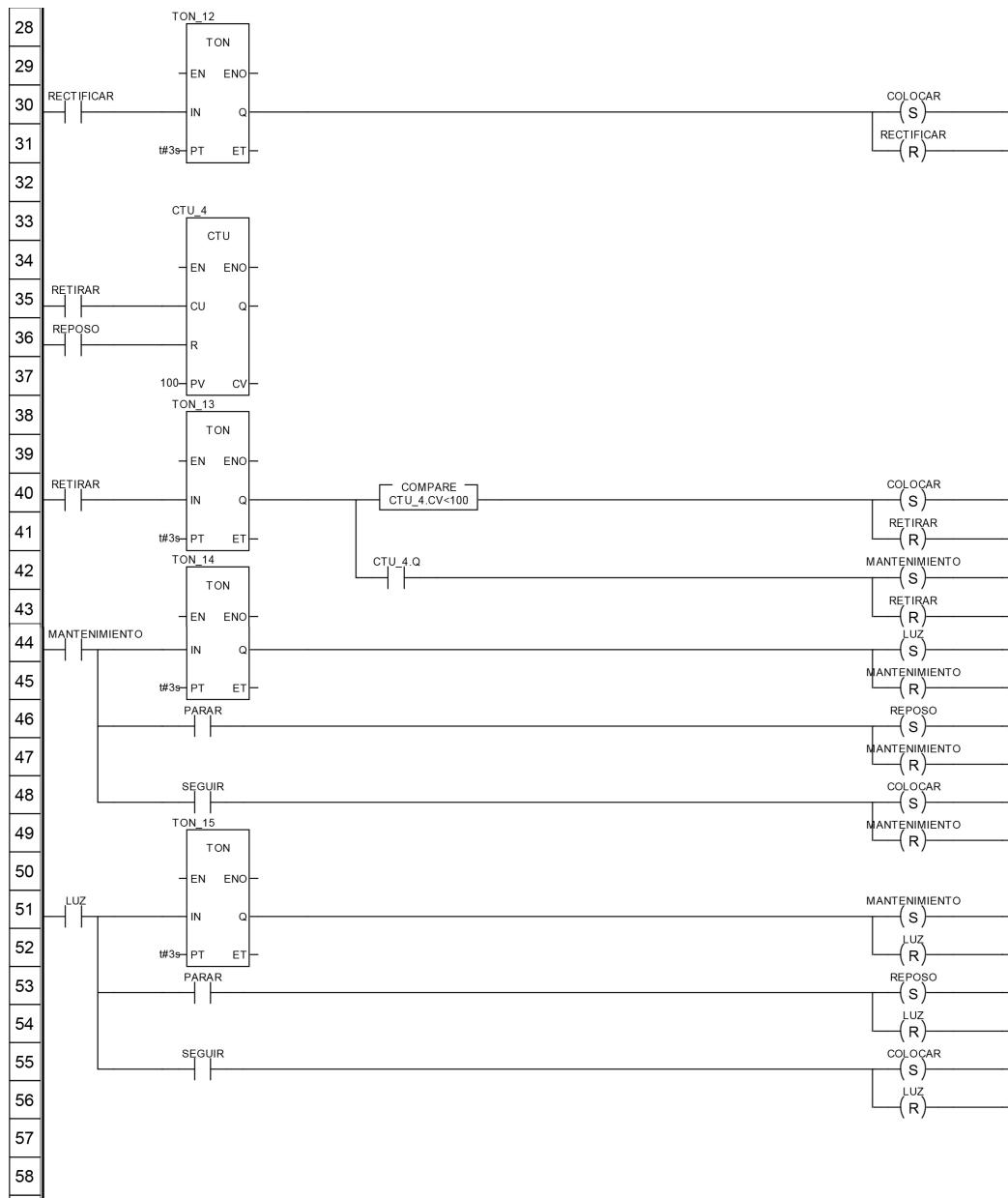
A continuación se muestra una lista de las entradas y salidas del sistema:

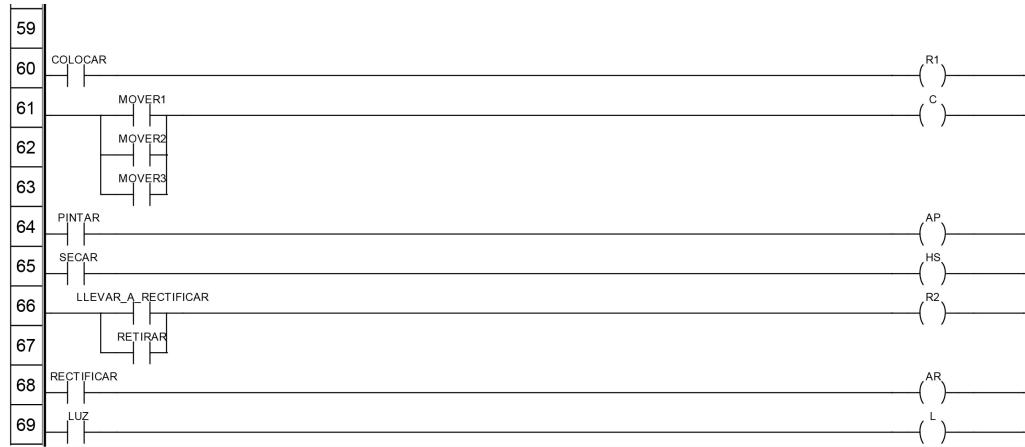
EBOOL

Nombre	Const	Dirección
AP	NO	%Q0.2.2
AR	NO	%Q0.2.5
B1	NO	%I0.1.6
B2	NO	%I0.1.7
C	NO	%Q0.2.0
COLOCAR	NO	%M1
HS	NO	%Q0.2.3
Inicio	NO	%I0.1.0
INSPECCIONAR	NO	%M7
L	NO	%Q0.2.6
LLEVAR_A_RECTIFICAR	NO	%M11
LUZ	NO	%M10
MANTENIMIENTO	NO	%M9
MOVER1	NO	%M2
MOVER2	NO	%M3
MOVER3	NO	%M4
P	NO	%I0.1.8
PARAR	NO	%I0.1.11
PINTAR	NO	%M5
R1	NO	%Q0.2.1
R2	NO	%Q0.2.4
RECTIFICAR	NO	%M12
REPOSO	NO	%M0
RETIRAR	NO	%M8
S1	NO	%I0.1.1
S2	NO	%I0.1.2
S3	NO	%I0.1.3
S4	NO	%I0.1.4
SECAR	NO	%M6
SEGUIR	NO	%I0.1.10
SR1	NO	%I0.1.5
SR2	NO	%I0.1.9

TUNEL_PINTURA_2 : [MAST]







3.4.2. Versión optimizada

Existe una posible optimización a este ejercicio tal y como muestra la figura siguiente. En ella se aprecia cómo se han unificado los tres estados que movían en la solución anterior en uno solo añadiendo las transiciones necesarias para preservar el funcionamiento descrito en el enunciado.

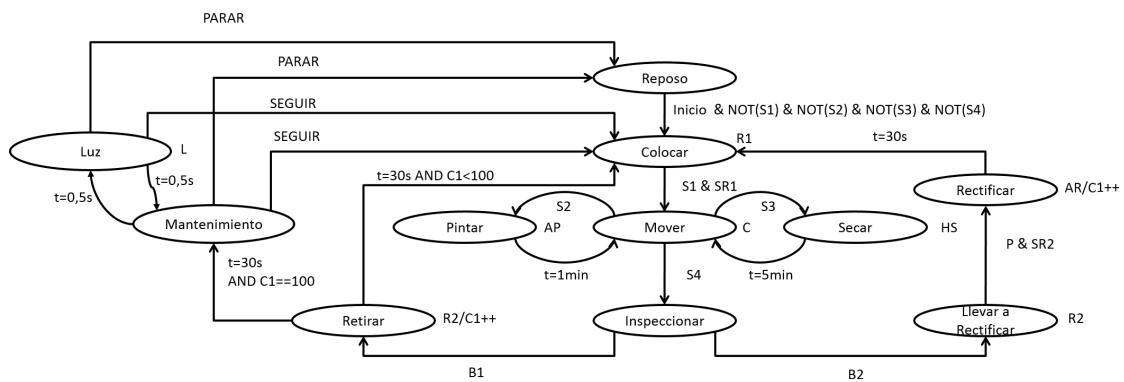


Figura 3.6: Túnel de pintura (Solución optimizada)

Queda como propuesta realizar la programación en LD para esta versión.

3.5. Sistema de llenado de bidones

Modelar con Diagrama de Estados la dinámica del sistema de llenado de bidones según se muestra en la siguiente figura.

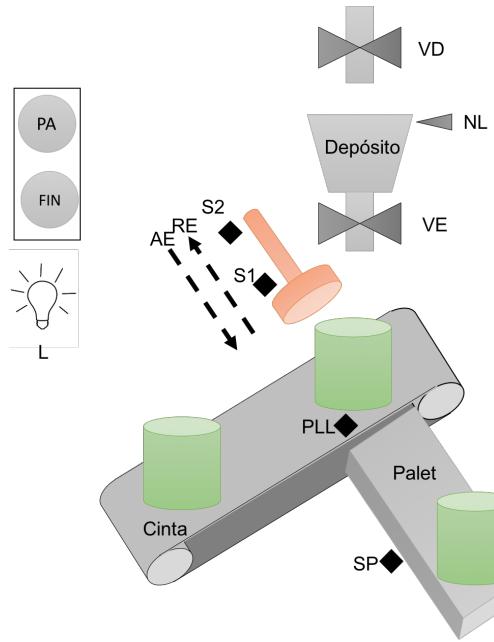


Figura 3.7: Esquema del sistema de llenado de bidones

- Al pulsar *PA* se pone en marcha la Cinta mediante la señal de control *MC* hasta que el bidón se sitúe en la posición de llenado *PLL*. Debe comprobarse que el Depósito está lleno (activado sensor *NL*), si no lo está se abrirá la válvula *VD* para llenarlo mientras se lleva el bidón a la posición de llenado. (NOTA: Se tarda más en posicionar el bidón que en llenar el depósito).
- Cuando el bidón active a *PLL* se detiene la cinta y al cabo de 2 segundos se deberá abrir la válvula *VE* la cual permanecerá abierta durante 30 segundos.
- Cuando *VE* se cierra porque ha permanecido 30 segundos abierta se activará la salida *AE* a fin de que el émbolo desplace al bidón lleno hasta depositarlo en el palé. Al llegar el émbolo al final de su recorrido

(detectado por $S1$) se activará la salida RE para retraer el émbolo hasta la posición detectada por $S2$. No se activará el émbolo si no hay palé (el palé es detectado por el sensor SP). En ese caso, una señal luminosa, L , parpadeará cada segundo para indicar a los operarios que deben poner un palé nuevo.

- Al estar el émbolo totalmente retraído vuelve a ponerse en marcha la Cinta para llenar los siguientes bidones.
- En el palé caben 4 bidones por lo que, al llegar a esta cantidad de bidones sobre el palé, el proceso se detiene hasta que se retire el palé.
- Si tras la retirada del palé no se ha pulsado el botón de FIN , el sistema vuelve a poner un bidón en la posición de llenado. En caso contrario el sistema se para hasta que se vuelva a pulsar PA .

Nota: Con el depósito lleno se pueden llenar 4 bidones.

Nota 2: El parpadeo se entiende que es un segundo con la luz encendida, uno con la luz apagada y así sucesivamente.

3.5.1. Solución

A continuación se muestra una solución posible al problema. Se han implementado algunos estados de espera para modelar la interacción con el usuario (puesta y retirada de palé). En el caso del llenado de los bidones hay que esperar 2 segundos, por lo que se ha incluido un estado, *ESPERA LLENAR BIDON*, con un temporizador.

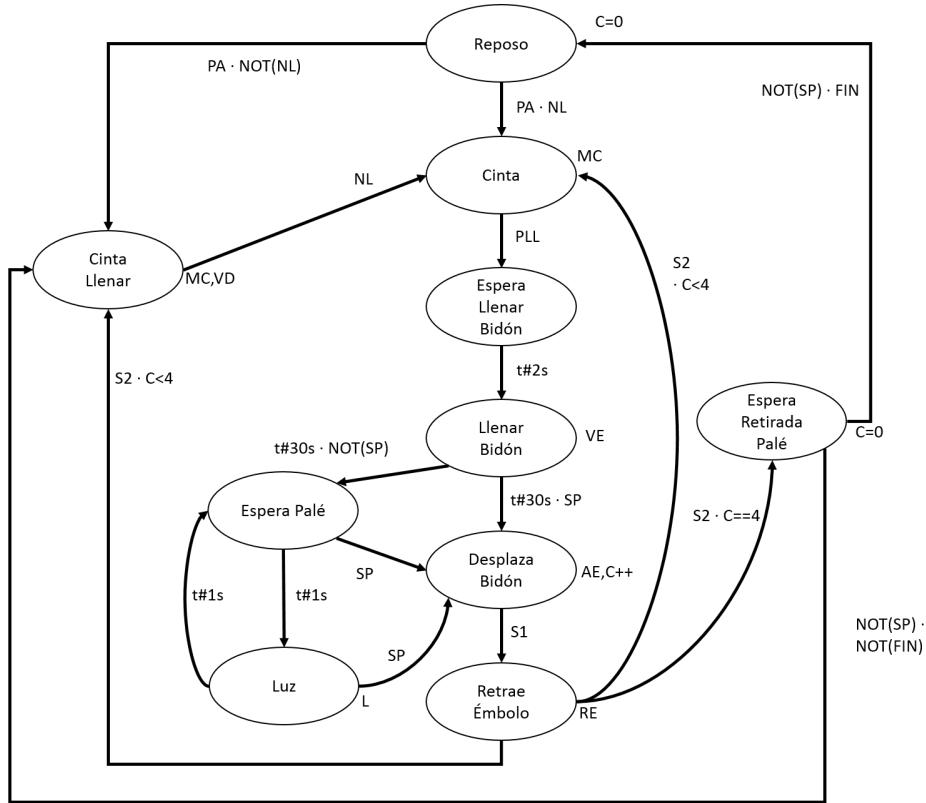


Figura 3.8: Sistema de llenado de bidones (Diagrama de estados)

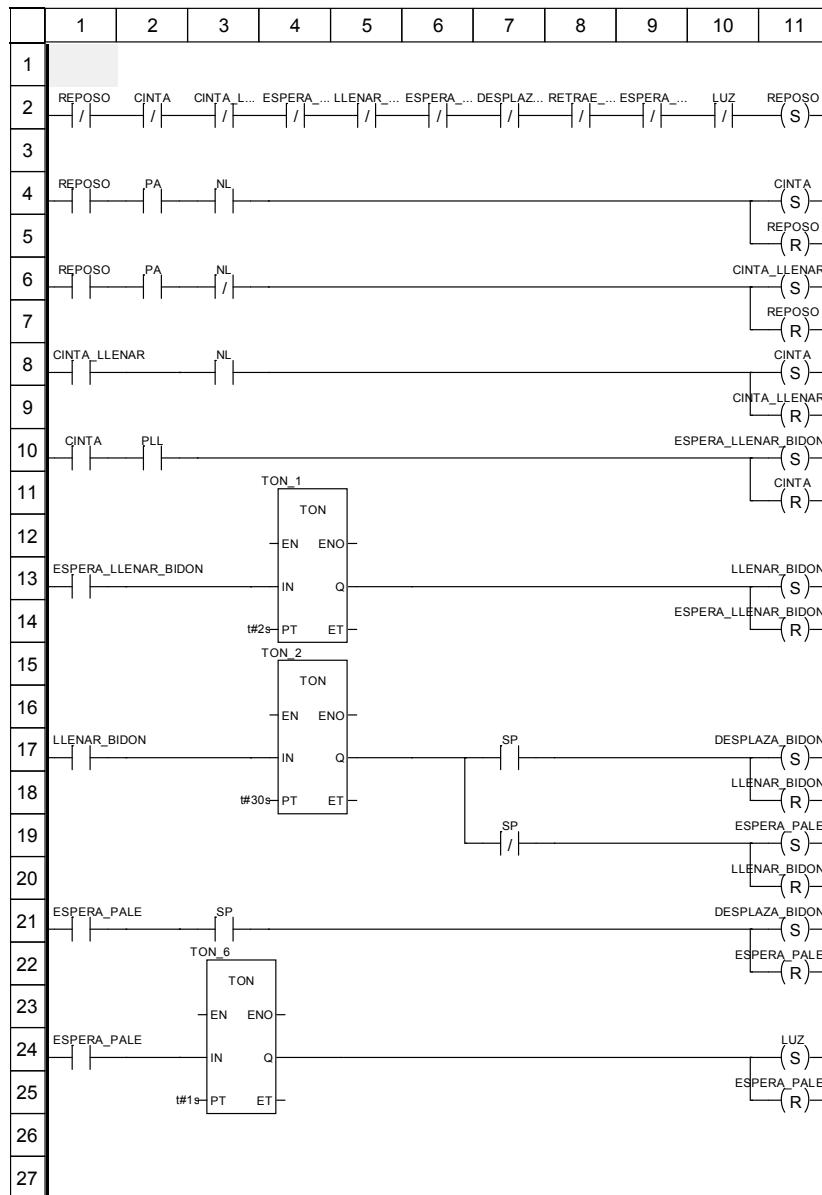
Las variables asociadas al sistema, así como sus direcciones de memoria asociadas se listan a continuación.

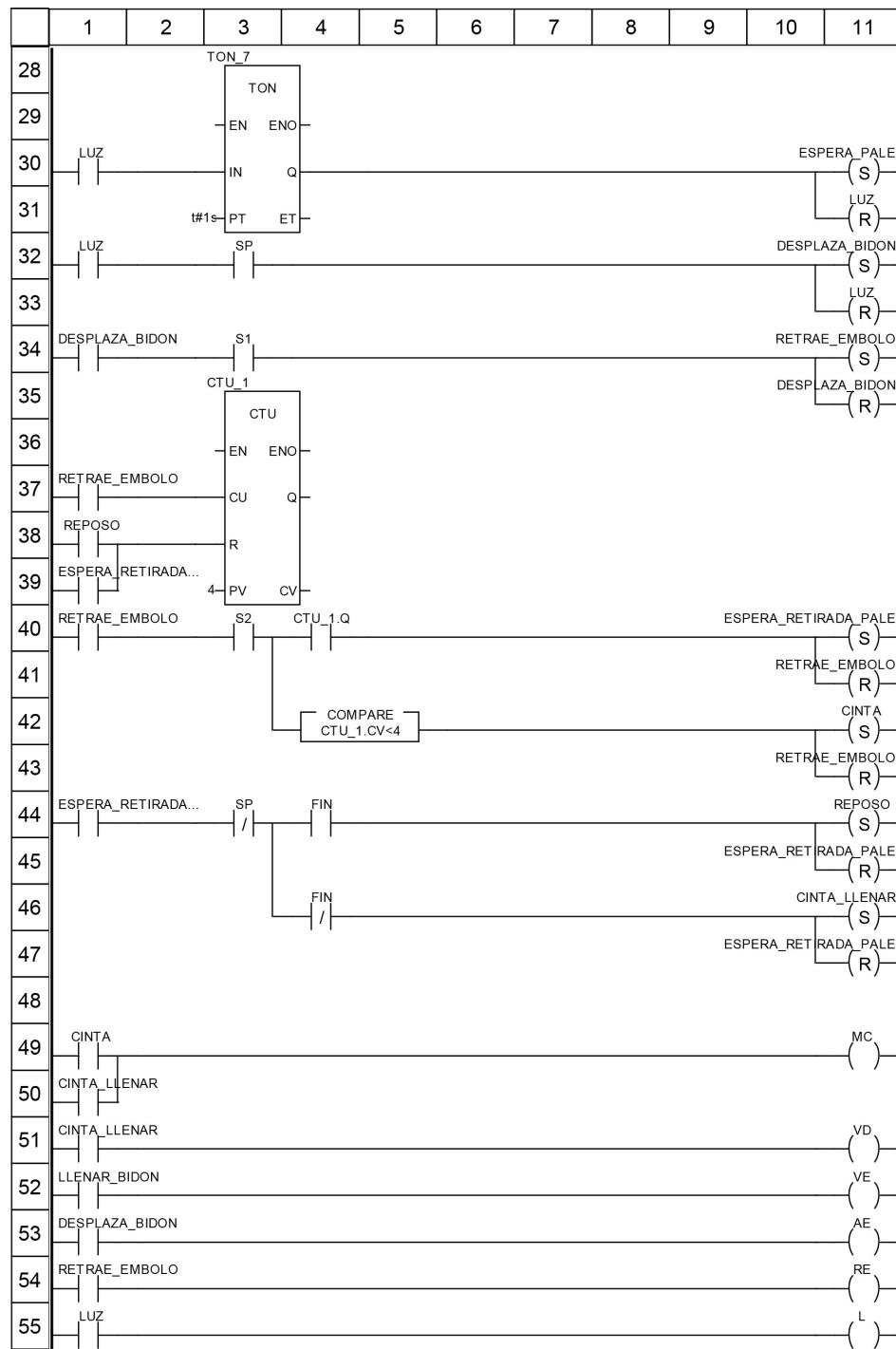
EBOOL

Nombre	Const	Dirección
AE	NO	%Q0.2.3
CINTA	NO	%M1
CINTA_LLENAR	NO	%M2
DESPLAZA_BIDO_N	NO	%M6
ESPERA_LLENAR_BIDON	NO	%M3
ESPERA_PALE	NO	%M5
ESPERA_RETIRA_DA_PALE	NO	%M8
FIN	NO	%I0.1.6
L	NO	%Q0.2.5
LLENAR_BIDON	NO	%M4
LUZ	NO	%M9
MC	NO	%Q0.2.0
NL	NO	%I0.1.2
PA	NO	%I0.1.0
PLL	NO	%I0.1.1
RE	NO	%Q0.2.4
REPOSO	NO	%M0
RETRAEMBOL_O	NO	%M7
S1	NO	%I0.1.3
S2	NO	%I0.1.4
SP	NO	%I0.1.5
VD	NO	%Q0.2.1
VE	NO	%Q0.2.2

Una vez definidas las variables, se procede a realizar la programación del sistema en lenguaje de contactos.

LLENADO_BIDONES : [MAST]





Capítulo 4

Ejercicios de SFC

4.1. Sistema multi-robot para gestión de almacén I

Una empresa de comercio electrónico con almacenes de distribución en Madrid ha adoptado la utilización de drones para gestión de la mercancía dentro del almacén. Como esta iniciativa es muy novedosa y está en fase de prueba, hay solamente disponible un drone para realizar la tarea de transportar paquetes dentro del almacén.

El proceso empieza cuando la central pide un paquete. Para agilizar el proceso de envío del paquete a la central el sistema dispone de un robot industrial junto con el drone. El funcionamiento del sistema se resume en que el robot industrial coge el paquete de la estantería y el drone lo coloca en la cinta que traslada el paquete hasta la central.

El PLC controla todos los procesos, envía los señales al drone mediante una conexión inalámbrica; activa el programa del robot industrial, etc. Los posibles comandos que pueden ser enviados al drone son:

Mandos	Acciones
C1	Despegar de la plataforma
C2	Volar al área de almacenamiento y coger el paquete
C3	Dejar el paquete encima de la cinta
C4	Aterrizar en la plataforma
CH	Cargar

El proceso se describe en detalle a continuación:

- El proceso empieza cuando la central pide un paquete mediante la señal *Signal2R*.
- Después, se realizan los siguientes procesos en paralelo:
 - El programa del robot industrial (**R**) es activado, con lo que el robot coge el paquete de la estantería y lo coloca en el suelo (detectado por el sensor *S_{floor}*).
 - Se envía un señal al drone para que despegue de la plataforma. Esta acción tarda 10s.
- Después, se envían las siguientes órdenes al drone:

- Coge el paquete del área de almacenamiento. Esta acción tarda 10s.
- Coloca el paquete encima de la cinta (detectado por el sensor de peso S_{weight}).
- Aterriza en la plataforma. Esta acción tarda 10s.

Por motivos de seguridad, una luz (**Light**) se enciende durante los primeros 5s cada vez que el drone está despegando y aterrizando en la plataforma.

Cuando termina el proceso, el sistema vuelve a el estado de reposo y espera a que la central le pida otro paquete. El drone solamente puede repetir este proceso 2 veces. Después tiene que cargar las baterías durante 15s antes de volver al estado de reposo. En la Figura 4.1 se ilustra el proceso que se pide modelar.

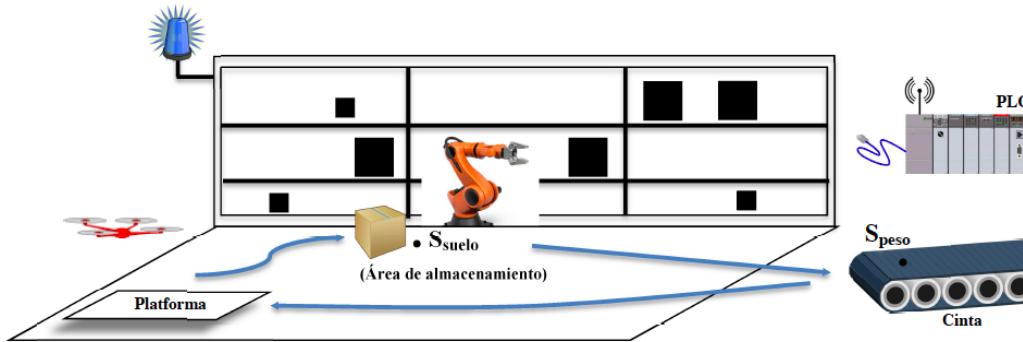


Figura 4.1: Sistema multi-robot para gestión de almacén I

Se pide:

1. Definir las entradas y las salidas. Definir las direcciones correspondientes en los módulos de E/S.
2. Diseñar el proceso en SFC
3. Programar en unity Pro el proceso.

4.1.1. Solución

Las variables asociadas al sistema, así como sus direcciones de memoria asociadas se listan a continuación.

BOOL

Nombre	Const	Dirección	Comentario	Valor	Utilizado	DG
Empty	NO			2	NO	
Full	NO			2	NO	
T_10s_1	NO			2	NO	
T_10s_2	NO			2	NO	
T_15s	NO			2	NO	

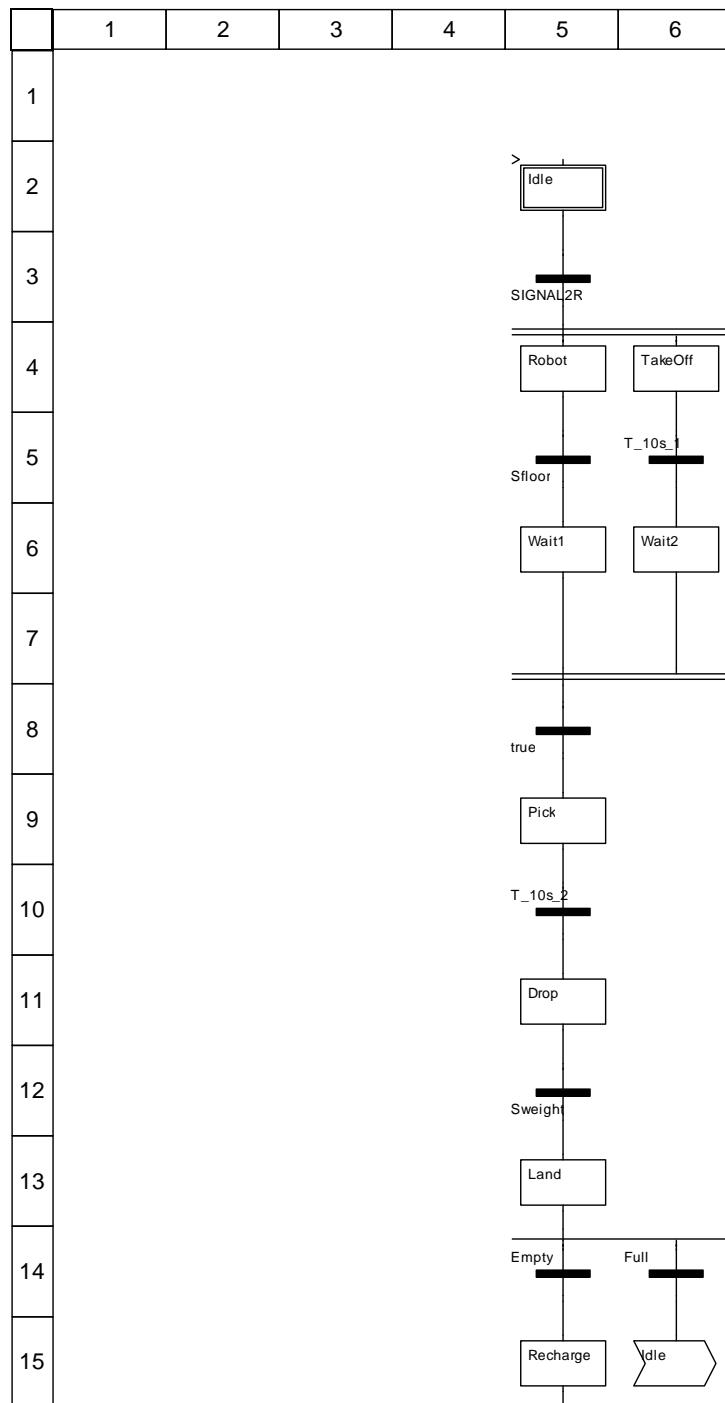
EBOOL

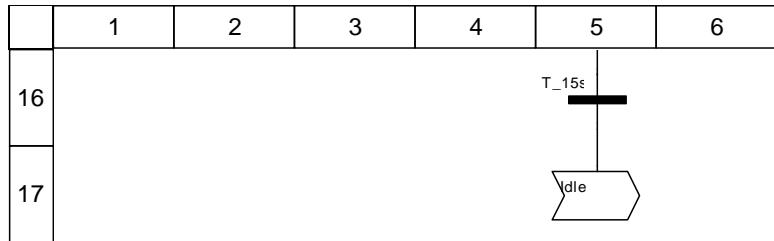
Nombre	Const	Dirección	Comentario	Valor	Utilizado	DG
C1	NO	%Q0.2.2		1	NO	
C2	NO	%Q0.2.3		1	NO	
C3	NO	%Q0.2.4		1	NO	
C4	NO	%Q0.2.5		1	NO	
CH	NO	%Q0.2.6		1	NO	
I	NO	%Q0.2.0		1	NO	
Light	NO	%Q0.2.15		2	NO	
R	NO	%Q0.2.1		1	NO	
Sfloor	NO	%IO.1.1		1	NO	
SIGNAL2R	NO	%IO.1.0		1	NO	
Sweight	NO	%IO.1.2		1	NO	

INT

Nombre	Const	Dirección	Comentario	Valor	Utilizado	DG
count	NO			4	NO	

Una vez definidas las variables, se procede a realizar la programación del sistema en diagrama funcional secuencial.





Los pasos, transiciones así como las variables que resultan del diagrama funcional secuencial, se detallan a continuación.

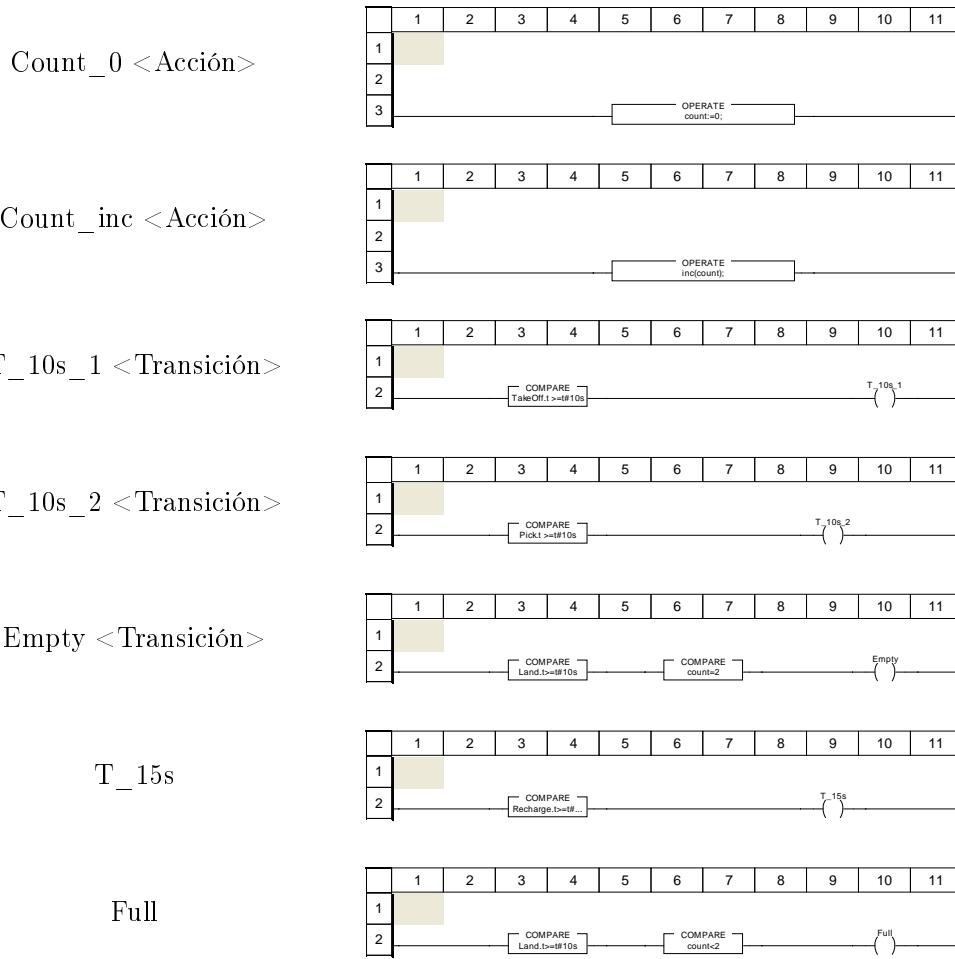
Drop	(5, 11)			
Tiempo de supervisión mín./máx.:	Tiempo de retardo:			
Comentario:				
Acciones:				
Descriptor: N	Tiempo:	Variable: C3		
Idle (paso inicial)	(5, 2)			
Tiempo de supervisión mín./máx.:	Tiempo de retardo:			
Comentario:				
Acciones:				
Descriptor: N	Tiempo:	Variable: I		
Land	(5, 13)			
Tiempo de supervisión mín./máx.:	Tiempo de retardo:			
Comentario:				
Acciones:				
Descriptor: P1	Tiempo:	Sección: LD :: count_inc		
Descriptor: N	Tiempo:	Variable: C4		
Descriptor: L	Tiempo: t#5s	Variable: Light		
Pick	(5, 9)			
Tiempo de supervisión mín./máx.:	Tiempo de retardo:			
Comentario:				
Acciones:				
Descriptor: N	Tiempo:	Variable: C2		
Recharge	(5, 15)			
Tiempo de supervisión mín./máx.:	Tiempo de retardo:			
Comentario:				
Acciones:				
Descriptor: P1	Tiempo:	Sección: LD :: count_0		
Descriptor: N	Tiempo:	Variable: CH		
Robot	(5, 4)			
Tiempo de supervisión mín./máx.:	Tiempo de retardo:			
Comentario:				
Acciones:				

Descriptor: N	Tiempo:	Variable: R
TakeOff	(6, 4)	
Tiempo de supervisión mín./máx.:		Tiempo de retardo:
Comentario:		
Acciones:		
Descriptor: N	Tiempo:	Variable: C1
Descriptor: L	Tiempo: t#5s	Variable: Light
Wait1	(5, 6)	
Tiempo de supervisión mín./máx.:		Tiempo de retardo:
Comentario:		
Wait2	(6, 6)	
Tiempo de supervisión mín./máx.:		Tiempo de retardo:
Comentario:		

Transiciones:

Nombre	Tipo de condición	Posición	Comentario
LD :: Empty	Sección	(5, 14)	
LD :: Full	Sección	(6, 14)	
SIGNAL2R	Variable	(5, 3)	
Sfloor	Variable	(5, 5)	
Sweight	Variable	(5, 12)	
LD :: T_10s_1	Sección	(6, 5)	
LD :: T_10s_2	Sección	(5, 10)	
LD :: T_15s	Sección	(5, 16)	
true	Constante	(5, 8)	

Veamos ahora, como se ha programado cada sección en lenguaje Ladder,



4.2. Sistema multi-robot para gestión de almacén II

Una empresa de comercio electrónico con almacenes de distribución en Madrid ha adoptado la utilización de drones para gestión de la mercancía dentro del almacén.

Como esta iniciativa es muy novedosa y está en fase de prueba, hay solamente disponible un drone para realizar la tarea de transportar paquetes dentro del almacén.

El proceso empieza cuando desde la central piden 1 o 2 paquetes. Cuando el pedido es enviado, el programa robot industrial es activado para recoger el (los) paquete(s) de la estantería y dejarlo(s) en el suelo. Después el drone recoge el(los) paquete(s) y lo(s) coloca en la cinta 1 o 2 según el numero de paquetes. La cinta 1 soporta el peso de 1 paquete y la cinta 2 soporta el peso de 2 paquetes.

El proceso es descrito en detalle a continuación:

- El proceso empieza cuando se presionan los botones P_1 o P_2 para pedir 1 o 2 paquetes.
- Después, el programa robot industrial (**Rob**) es activado durante 10s y hasta que se detecte 1 paquete en el suelo (sensor S_1) o 2 paquetes en suelo (sensor S_2).
- Si la mercancía dejada en el suelo por el robot industrial es detectada por el sensor S_1 entonces la PLC envía una señal al drone (**Drone_i**) para que recoja 1 paquete del suelo y lo ponga en la cinta cinta 1 (detectado por el sensor S_{peso_1}). El motor M_1 es activado durante 10s una vez detectado el paquete en la cinta 1.
- Si la mercancía dejada en el suelo por el robot industrial es detectada por el sensor S_2 entonces la PLC envía otro comando al drone (**Drone_d**) para que recoja 2 paquete del suelo y los ponga en la cinta 2. Cada vez que un paquete es colocado encima de la cinta 2 es detectado por el sensor S_{peso_2} . El drone solamente puede transportar un paquete a la vez. Además antes de ir por el segundo paquete el proceso debe ser monitorizado durante 2s para asegurar que todo va bien. Cuando los dos paquetes estén encima de la cinta 2, el motor M_2 es activado durante 20s.

- Finalmente el drone carga las baterías durante 15s antes de que el proceso vuelva al estado inicial de reposo. Durante el proceso de cargar las baterías una luz estará encendida durante los primeros 10s indicando que el drone está en modo carga. En la Figura 4.2 ilustra el proceso que se pide modelar.

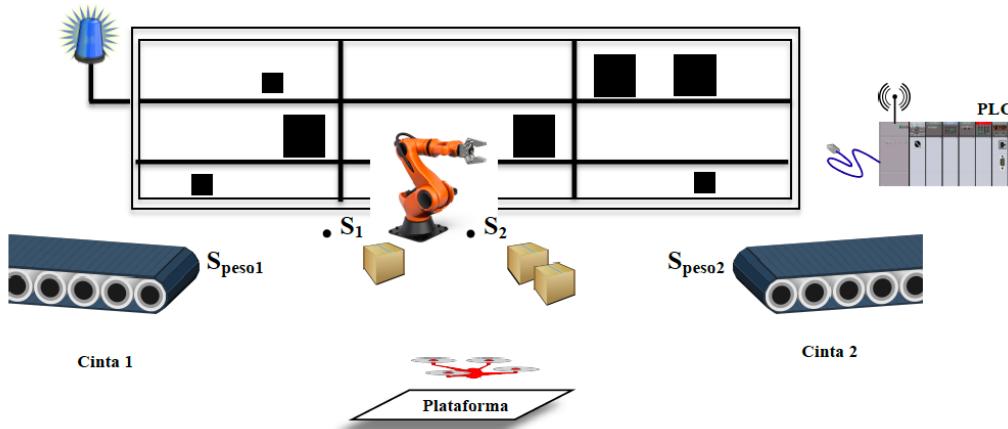


Figura 4.2: Sistema multi-robot para gestión de almacén II

Se pide:

- Definir las entradas y las salidas. Definir las direcciones correspondientes en los módulos de E/S.
- Diseñar el proceso en SFC
- Programar en unity Pro el proceso.

4.2.1. Solución

Las variables asociadas al sistema, así como sus direcciones de memoria asociadas se listan a continuación.

BOOL

Nombre	Const	Dirección	Comentario	Valor	Utilizado	DG
DOS	NO			2	NO	
NoTerminado_d	NO			2	NO	
NoTerminado_i	NO			1	NO	
T_2s	NO			2	NO	
T_10s	NO			2	NO	
T_15s	NO			2	NO	
T_20s	NO			2	NO	
Terminado_d	NO			2	NO	
UNO	NO			2	NO	

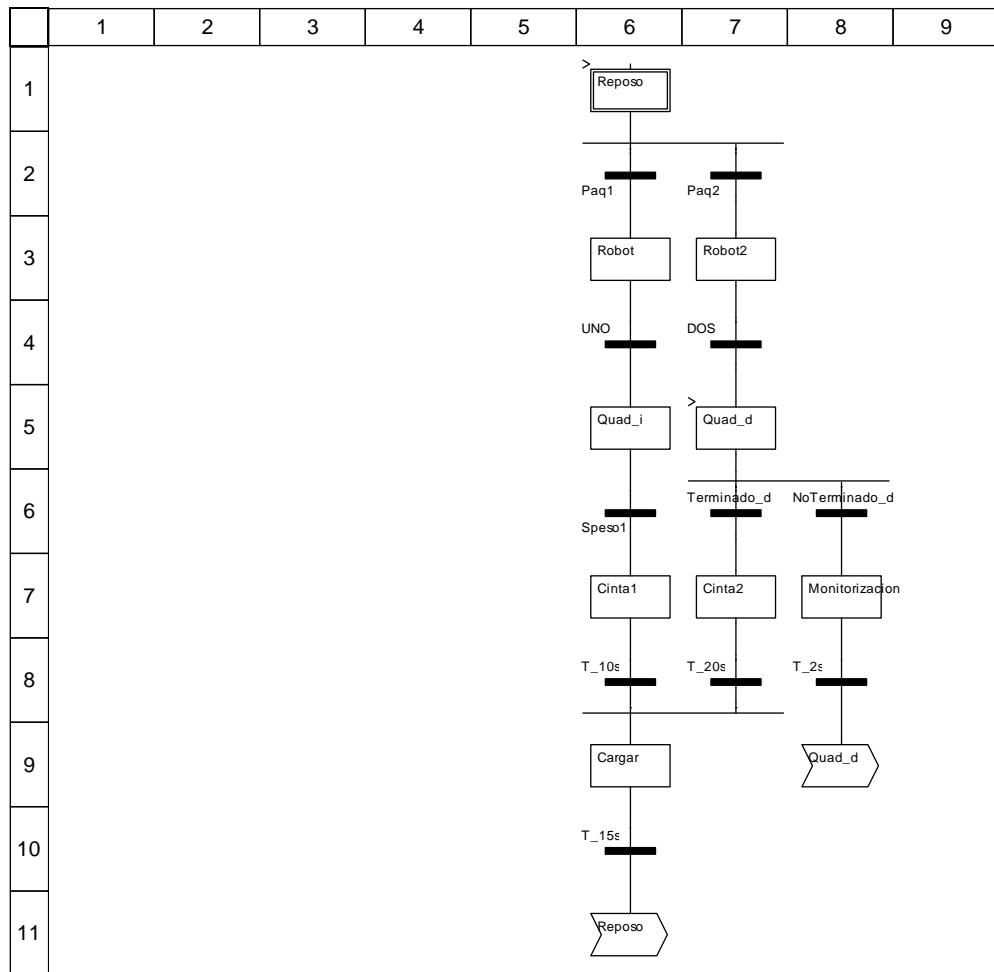
EBOOL

Nombre	Const	Dirección	Comentario	Valor	Utilizado	DG
Cag	NO	%Q0.2.6		1	NO	
Drone_d	NO	%Q0.2.3		1	NO	
Drone_i	NO	%Q0.2.2		1	NO	
Luz	NO	%Q0.2.7		1	NO	
M1	NO	%Q0.2.4		1	NO	
M2	NO	%Q0.2.5		1	NO	
Mon	NO			1	NO	
Paq1	NO	%I0.1.7		1	NO	
Paq2	NO	%I0.1.8		1	NO	
Rep	NO	%Q0.2.0		1	NO	
Rob	NO	%Q0.2.1		0	NO	
S1	NO	%I0.1.5		1	NO	
S2	NO	%I0.1.6		1	NO	
Signal2R	NO	%I0.1.0		0	NO	
Speso1	NO	%I0.1.3		2	NO	
Speso2	NO	%I0.1.4		2	NO	

INT

Nombre	Const	Dirección	Comentario	Valor	Utilizado	DG
count	NO			6	NO	

Una vez definidas las variables, se procede a realizar la programación del sistema en diagrama funcional secuencial.



Los pasos, transiciones así como las variables que resultan del diagrama funcional secuencial, se detallan a continuación.

Cargar	(6, 9)
Tiempo de supervisión mín./máx.:	Tiempo de retardo:
Comentario:	
Acciones:	
Descriptor: N Tiempo:	Variable: Cag
Descriptor: L Tiempo: t#7s	Variable: Luz
Cinta1	(6, 7)
Tiempo de supervisión mín./máx.:	Tiempo de retardo:

Comentario:	
Acciones:	
Descriptor: N Tiempo:	Variable: M1

Cinta2	(7, 7)
Tiempo de supervisión mín./máx.:	Tiempo de retardo:
Comentario:	
Acciones:	
Descriptor: N Tiempo:	Variable: M2

Monitorizacion	(8, 7)
Tiempo de supervisión mín./máx.:	Tiempo de retardo:
Comentario:	
Acciones:	
Descriptor: N Tiempo:	Variable: Mon

Quad_d	(7, 5)
Tiempo de supervisión mín./máx.:	Tiempo de retardo:
Comentario:	
Acciones:	
Descriptor: N Tiempo:	Variable: Drone_d
Descriptor: P1 Tiempo:	Sección: LD :: count_inc

Quad_i	(6, 5)
Tiempo de supervisión mín./máx.:	Tiempo de retardo:
Comentario:	
Acciones:	
Descriptor: N Tiempo:	Variable: Drone_i
Descriptor: P1 Tiempo:	Sección: LD :: count_inc

Robot	(6, 3)
Tiempo de supervisión mín./máx.:	Tiempo de retardo:
Comentario:	

Robot2	(7, 3)
Tiempo de supervisión mín./máx.:	Tiempo de retardo:
Comentario:	

Transiciones:

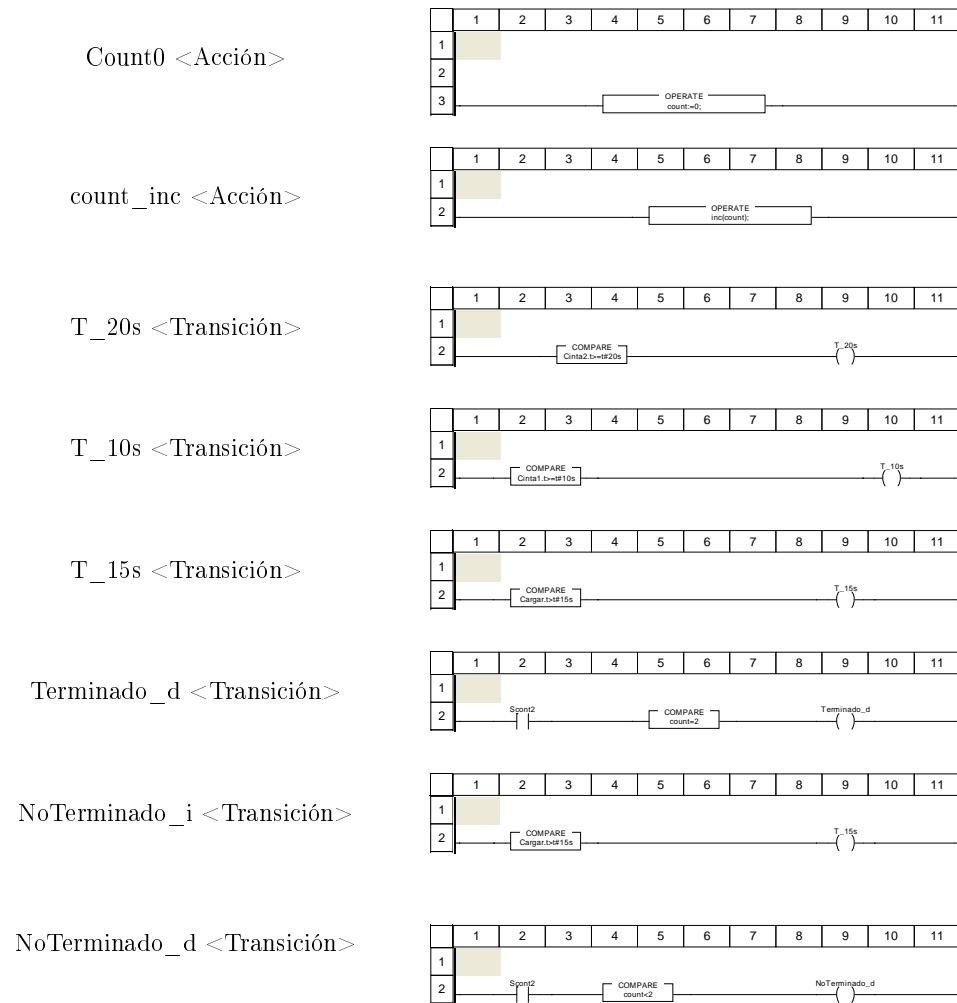
Nombre	Tipo de condición	Posición	Comentario
LD :: DOS	Sección	(7, 4)	
LD :: NoTerminado_d	Sección	(8, 6)	
Paq1	Variable	(6, 2)	
Paq2	Variable	(7, 2)	
Speso1	Variable	(6, 6)	
LD :: T_10s	Sección	(6, 8)	
LD :: T_15s	Sección	(6, 10)	
LD :: T_20s	Sección	(7, 8)	

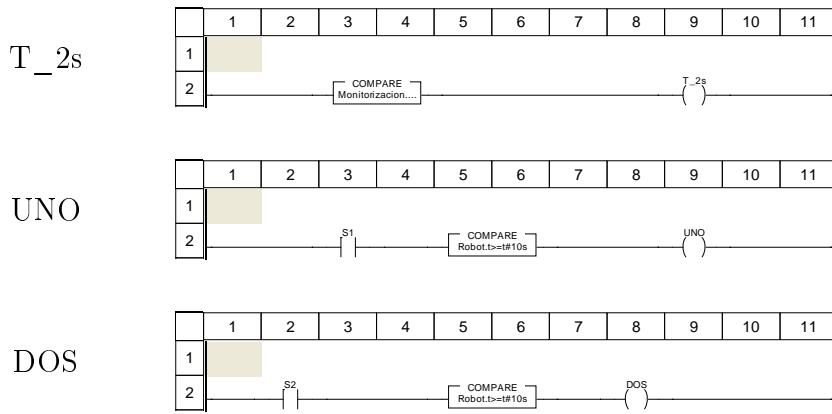
LD :: T_2s	Sección	(8, 8)	
LD :: Terminado_d	Sección	(7, 6)	
LD :: UNO	Sección	(6, 4)	

Saltos:

Nombre	Posición	Comentario
Quad_d	(8, 9)	
Reposo	(6, 11)	

Veamos ahora, como se ha programado cada sección en lenguaje Ladder,





4.3. Sistema automático de fabricación de sidra

Un productor de sidra Asturiana ha decidido contratar un equipo de alumnos de grado de la UC3M para automatizar el proceso de elaboración de su sidra. Los sistemas utilizados en el proceso de elaboración de la cerveza se enseñan a la continuación. El proceso consta de las siguientes etapas:

1. El sistema comenzará a funcionar de forma automática cuando se pulse **Start**.
2. Se abre la válvula **VA** para llenar de agua la primera caldera (detectado por **S1H**).
3. Se calienta la agua (activando la resistencia **R1**) durante 10 segundos y a la falta de 5 segundos para terminar de calentar se introducen los trozos de manzana en la caldera, activando el motor de la cinta **M1** y el cortador de manzanas **M2**.
4. La pulpa que resulta del proceso anterior pasa al prensador para obtener la sidra dulce (mosto) y va directamente a la segunda caldera, abriendo la válvula **VB** y activando el motor de la prensa **M3** hasta que **S1L** detecte que la primera caldera está vacía.
5. Se abre la válvula **VC** para llenar de agua la segunda caldera durante 10 segundos, se echa la levadura abriendo la compuerta **A1** en los primeros 5 segundos y se activa el motor del mezclador **M4** en los últimos 5 segundos.
6. Después y en simultáneo:
 - a) Se embotella la sidra:
 - 1) Se activa el motor de la cinta **M5** hasta que el sensor **S2** detecta la botella.
 - 2) Se echa azúcar abriendo la válvula **VE** durante 1 segundo.
 - 3) Se activa nuevamente el motor de la cinta **M5** hasta que el sensor **S3** detecta la botella.
 - 4) Se echa la sidra abriendo la válvula **VD** durante 2 segundo. Con el contenido de la caldera se llenan 3 botellas. El proceso de embotellado se repite entonces 3 veces.

- b) Se limpia la primera caldera:
- 1) Se abre la válvulas **VA** para llenar la caldera de agua hasta que el agua es detectada por el sensor **S2H**.
 - 2) Se calienta el agua durante 5 segundos activando **R1**.
 - 3) Se vacía la caldera (detectado por el sensor **S1L**) abriendo la válvula **VF**.

7. Cuando los procesos terminan vuelve a la situación inicial y, si el pulsador de **INICIO** sigue activado, el proceso vuelve a recomenzar.

Para esterilizar las botellas, se enciende una luz ultravioleta (**LU**) desde el momento que el motor de la cinta **M5** es activado hasta que termine todo el proceso de embotellado.

NOTA:

S1L: Si es TRUE, detecta que la primera caldera esta vacía

S1H: Si es TRUE, detecta que la primera caldera esta llena

Se pide:

1. Definir las entradas y las salidas. Definir las direcciones correspondientes en los módulos de E/S.
2. Diseñar el proceso en SFC
3. Programar en unity Pro el proceso.

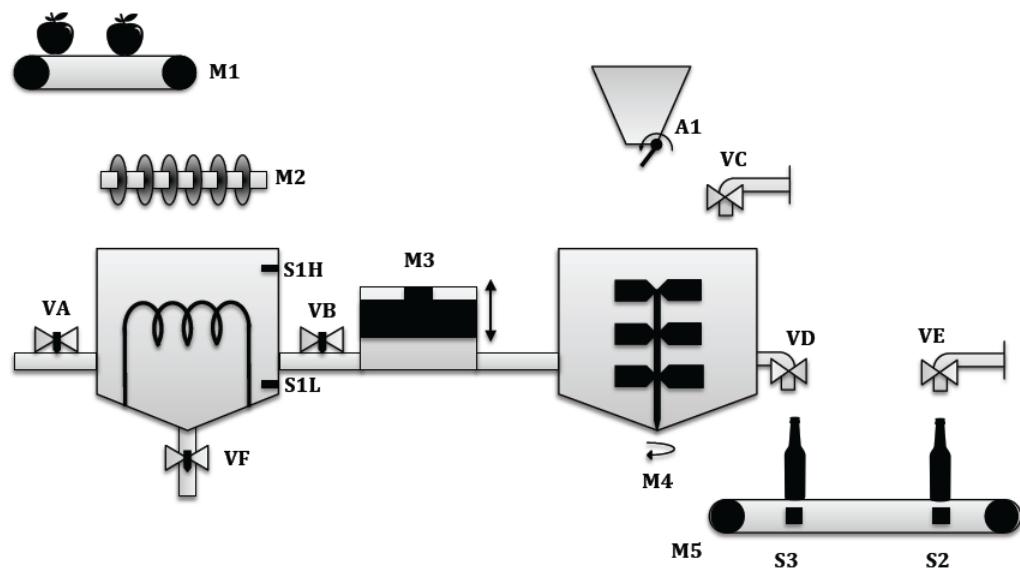


Figura 4.3: Sistema automático de fabricación de sidra

4.3.1. Solución

Las variables asociadas al sistema, así como sus direcciones de memoria asociadas se listan a continuación.

BOOL

Nombre	Const	Dirección	Comentario	Valor	Utilizado	DG
done	NO			2	NO	
incomplete	NO			2	NO	
T_1s	NO			2	NO	
T_5s	NO			2	NO	
T_10IIs	NO			2	NO	
T_10s	NO			2	NO	

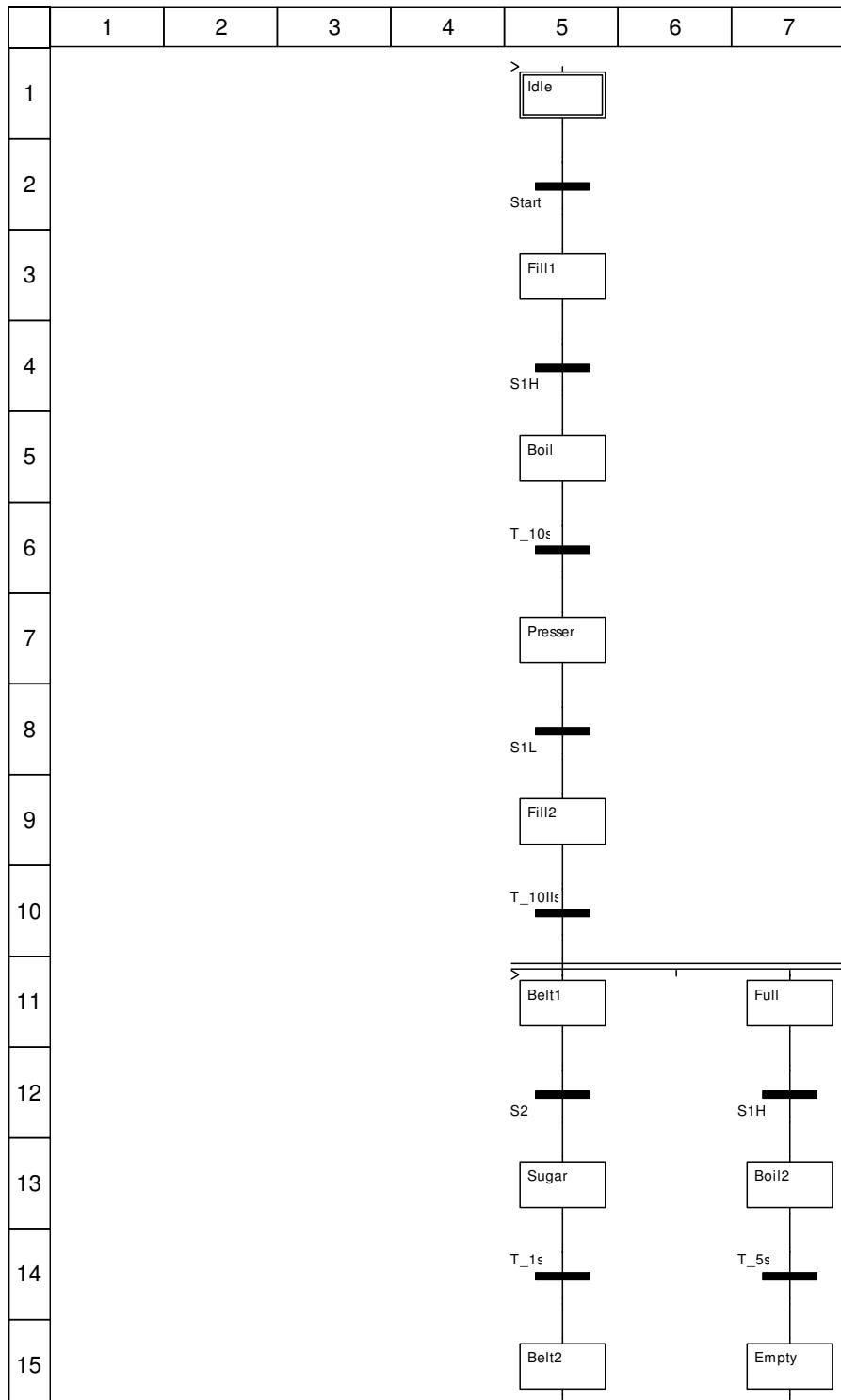
EBOOL

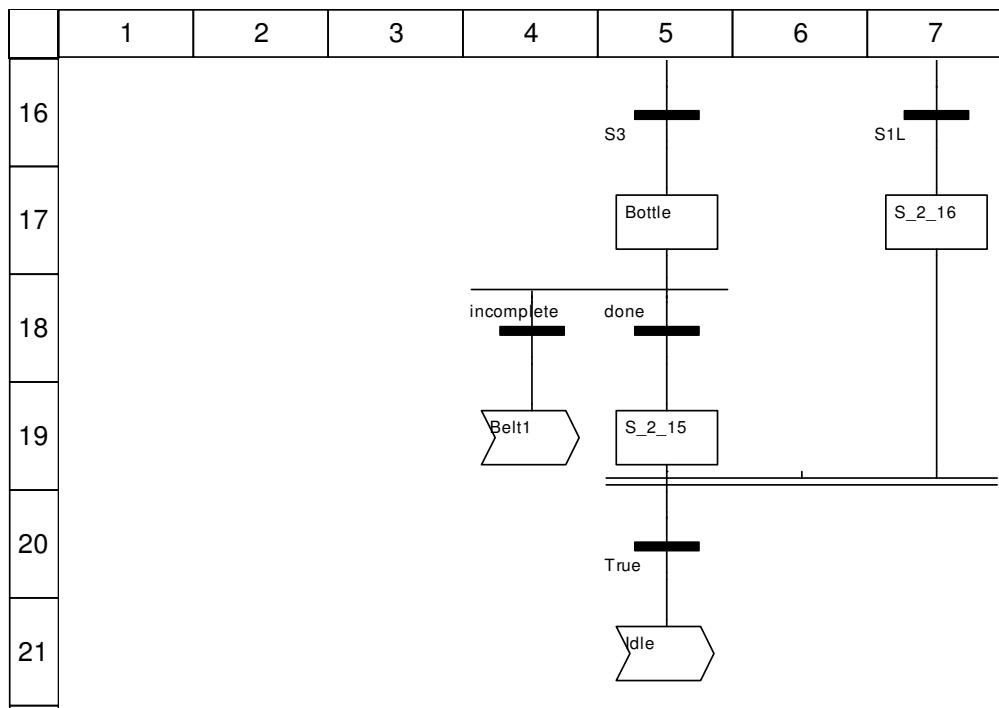
Nombre	Const	Dirección	Comentario	Valor	Utilizado	DG
A1	NO	%Q0.2.6		1	NO	
Finish	NO			0	NO	
LU	NO	%Q0.2.13		2	NO	
M1	NO	%Q0.2.1		1	NO	
M2	NO	%Q0.2.2		1	NO	
M3	NO	%Q0.2.4		1	NO	
M4	NO	%Q0.2.7		1	NO	
M5	NO	%Q0.2.8		2	NO	
R1	NO	%Q0.2.11		2	NO	
S1H	NO	%I0.1.1		2	NO	
S1L	NO	%I0.1.2		2	NO	
S2	NO	%I0.1.3		1	NO	
S3	NO	%I0.1.4		1	NO	
Start	NO	%I0.1.0		1	NO	
VA	NO	%Q0.2.0		2	NO	
VB	NO	%Q0.2.3		1	NO	
VC	NO	%Q0.2.5		0	NO	
VD	NO	%Q0.2.10		1	NO	
VE	NO	%Q0.2.9		1	NO	
VF	NO	%Q0.2.12		1	NO	

INT

Nombre	Const	Dirección	Comentario	Valor	Utilizado	DG
count	NO			4	NO	

Una vez definidas las variables, se procede a realizar la programación del sistema en diagrama funcional secuencial.





Los pasos, transiciones así como las variables que resultan del diagrama funcional secuencial, se detallan a continuación.

Belt1	(5, 11)
Tiempo de supervisión mín./máx.:	Tiempo de retardo:
Comentario:	
Acciones:	
Descriptor: N	Tiempo: Variable: M5
Descriptor: S	Tiempo: Variable: LU

Belt2	(5, 15)
Tiempo de supervisión mín./máx.:	Tiempo de retardo:
Comentario:	
Acciones:	
Descriptor: N	Tiempo: Variable: M5

Boil	(5, 5)
Tiempo de supervisión mín./máx.:	Tiempo de retardo:
Comentario:	
Acciones:	
Descriptor: D	Tiempo: t#5s Variable: M1
Descriptor: D	Tiempo: t#5s Variable: M2
Descriptor: N	Tiempo: Variable: R1

Boil2	(7, 13)
Tiempo de supervisión mín./máx.:	Tiempo de retardo:
Comentario:	
Acciones:	

Descriptor: N	Tiempo: Variable: R1
---------------	----------------------

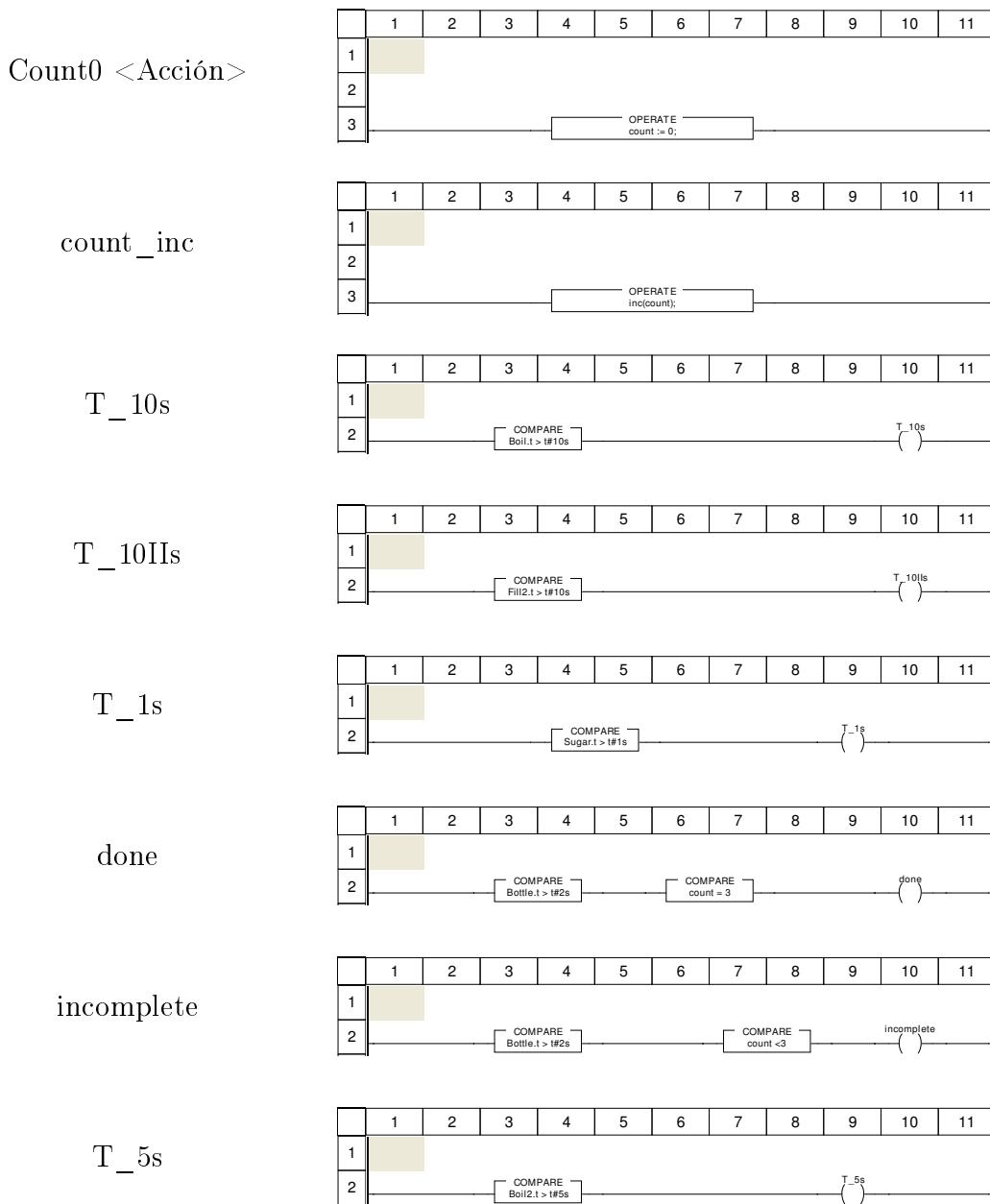
Bottle	(5, 17)
Tiempo de supervisión mín./máx.:	Tiempo de retardo:
Comentario:	
Acciones:	
Descriptor: N	Tiempo: Variable: VD
Descriptor: P1	Tiempo: Sección: LD :: count_inc

S_2_16	(7, 17)
Tiempo de supervisión mín./máx.:	Tiempo de retardo:
Comentario:	
Sugar	(5, 13)
Tiempo de supervisión mín./máx.:	Tiempo de retardo:
Comentario:	
Acciones:	
Descriptor: N	Tiempo: Variable: VE

Transiciones:

Nombre	Tipo de condición	Posición	Comentario
S1H	Variable	(5, 4)	
S1H	Variable	(7, 12)	
S1L	Variable	(5, 8)	
S1L	Variable	(7, 16)	
S2	Variable	(5, 12)	
S3	Variable	(5, 16)	
Start	Variable	(5, 2)	
LD :: T_10IIs	Sección	(5, 10)	
LD :: T_10s	Sección	(5, 6)	
LD :: T_1s	Sección	(5, 14)	
LD :: T_5s	Sección	(7, 14)	
True	Constante	(5, 20)	
LD :: done	Sección	(5, 18)	
LD :: incomplete	Sección	(4, 18)	

Veamos ahora como se ha programado cada sección en lenguaje Ladder:



4.4. Sistema automático de fabricación de cerveza

Una nueva marca de cerveza Española pretende contratar un equipo de alumnos de grado de la UC3M para automatizar el proceso de elaboración de su cerveza. Los sistemas utilizados en el proceso de elaboración de la cerveza se enseñan a la continuación.

El proceso consta de las siguientes etapas:

1. El sistema comenzará a funcionar de forma automática cuando se pulse INICIO.
2. Se abre la válvula **V1** para llenar de agua la caldera A (detectado por **S1H**).
3. Se calienta el agua (activando la resistencia **R1**) durante 10 segundos.
4. Se introduce la Malta en la caldera A abriendo la compuerta **A1** durante 3 segundos.
5. Se remuelve la mezcla de la caldera A cada 5 segundos durante 20 segundos. Para remover se activa **M1** durante 2 segundos.
6. Se abre la válvula **V2** para que la mezcla entre en la caldera B durante 3 segundos.
7. Se procede al filtrado para separar los cereales del mosto (líquido) activando **M2** durante 5 segundos. Al faltar 3 segundos para terminar de aspirar todo el cereal se abre la válvula **V1** para llenar de agua la caldera A.
8. Se vuelve a calentar la caldera A (activando la resistencia **R1**) durante 10 segundos.
9. Después y en simultáneo, se abren las válvulas **V2** y **V3** para terminar el proceso de filtrado hasta que la caldera A se quede vacía (detectado por **S1L**).
10. Se calienta el mosto en la caldera C (activando la resistencia **R2**) y se remueve el mosto continuamente (activando **M3**) durante 10 segundos. En los últimos 3 segundos se le añade el lúpulo al mosto (activando la compuerta **A2**).

11. Se enfriá el mosto haciéndole pasar por el enfriador (abriendo válvula **V4** y **V7**) y activando el motor de refrigeración **M4** hasta que el sensor **S2L** no detecte más líquido en el tanque C.
12. Una vez que la mezcla final llega al depósito de fermentación se le añade la dosis de levadura (activando **A3**) durante 3 segundos.
13. Una vez terminado todo el proceso es necesario lavar la caldera C y retirar de la caldera B los restos de cereal que han quedado del filtrado. De este modo y en simultáneo: Se activa el motor **M5** del aspirador hasta que no quede más cereal en el tanque 2 (detectado por el sensor de peso **SP1**), y se abre la válvula de agua **V5** y la válvula de salida del tanque **V6** durante 5 segundos.

Cuando termina vuelve a la posición inicial y el proceso vuelve a recomenzar si el tanque de fermentación no está lleno (detectado por el sensor **S2H**). En el caso de estar lleno, se debe vaciar el tanque (detectado por el sensor **S2L**) abriendo la válvula **V8**.

Por motivos de higiene, se debe encender una luz ultravioleta (**LU**) desde el filtrado hasta que se echa la levadura. Debe también sonar un pitido (**B**) mientras se está vaciando el depósito de fermentación.

NOTA:

SxL: Si es TRUE, detecta que la primera caldera esta vacía

SxH: Si es TRUE, detecta que la primera caldera esta llena

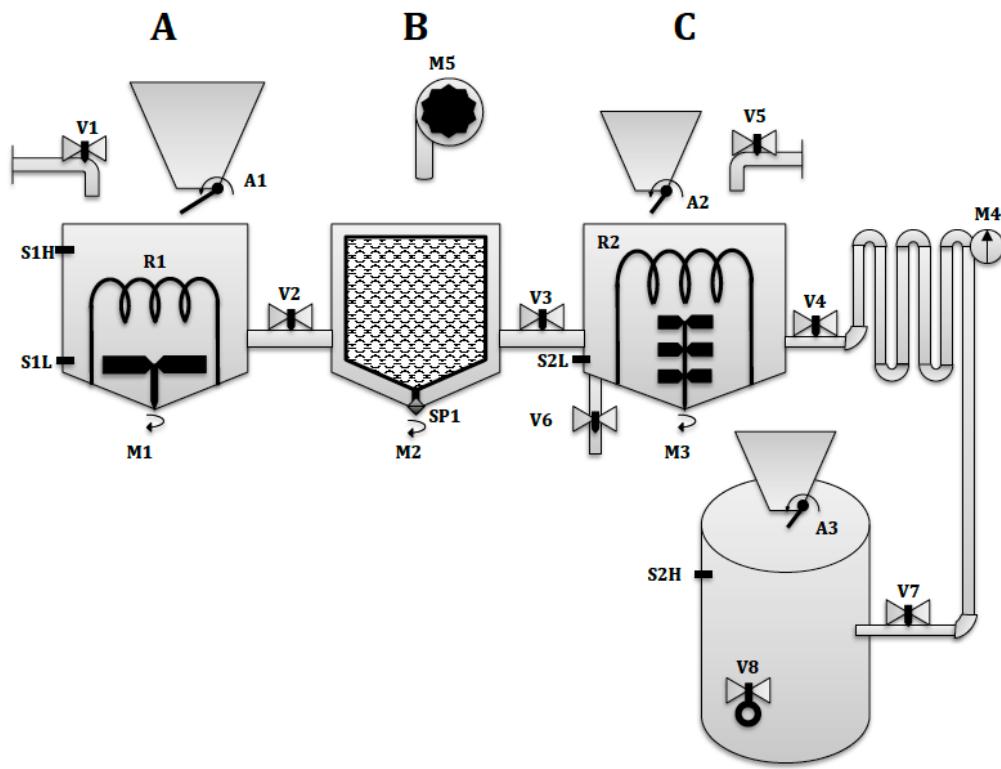


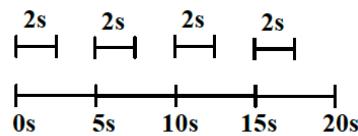
Figura 4.4: Sistema automático de fabricación de cerveza

Se pide:

1. Definir las entradas y las salidas. Definir las direcciones correspondientes en los módulos de E/S.
2. Diseñar el proceso en SFC
3. Programar en unity Pro el proceso.

4.4.1. Solución

Aunque este ejercicio es sencillo, el punto 5 puede resultar confuso. En este punto se indica “Se remueve la mezcla en la caldera A cada 5 segundos durante 20 segundos. Para removerla se activa M1 durante 2 segundos”. La siguiente figura aclara lo que se pretende:



Por tanto, para modelar este sub-proceso utilizaremos un bucle: cada pasada del bucle durará 5 segundos y se remueve activando M1 durante los 2 primeros segundos de ese tiempo; en total con 4 intervalos de 5 segundos (4 pasadas del bucle) se completan los 20 segundos requeridos para el sub-proceso de mezcla.

Las variables asociadas al sistema, así como sus direcciones de memoria asociadas se listan a continuación.

BOOL

Nombre	Const	Dirección	Comentario	Valor	Utilizado	DG
Lleno	NO			2	NO	
No	NO			2	NO	
si	NO			2	NO	
T_3IIs	NO			2	NO	
T_3Is	NO			2	NO	
T_3s	NO			2	NO	
T_5Is	NO			2	NO	
T_5s	NO			2	NO	
T_10IIs	NO			2	NO	
T_10Is	NO			2	NO	
T_10s	NO			2	NO	
Vacio	NO			2	NO	

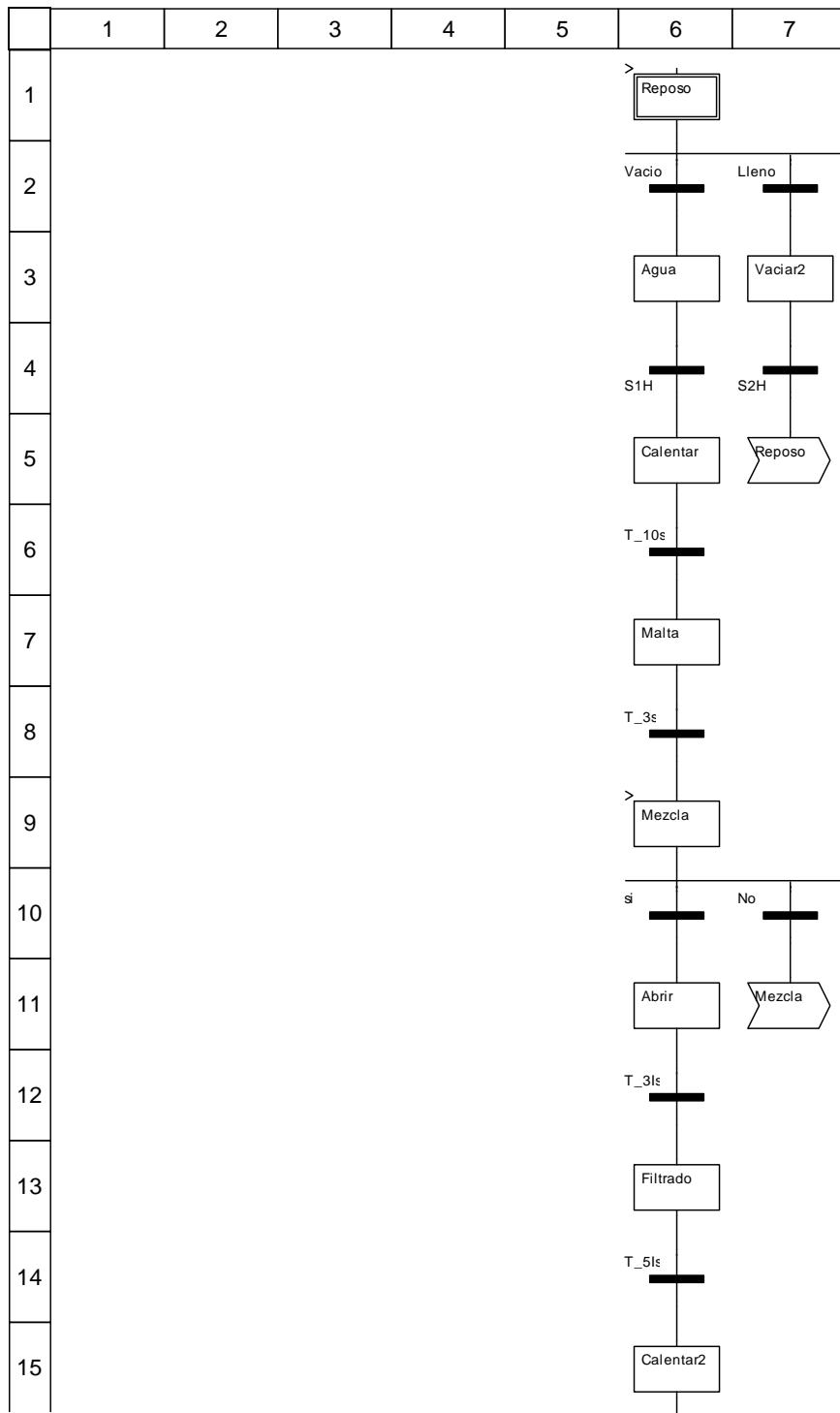
EBOOL

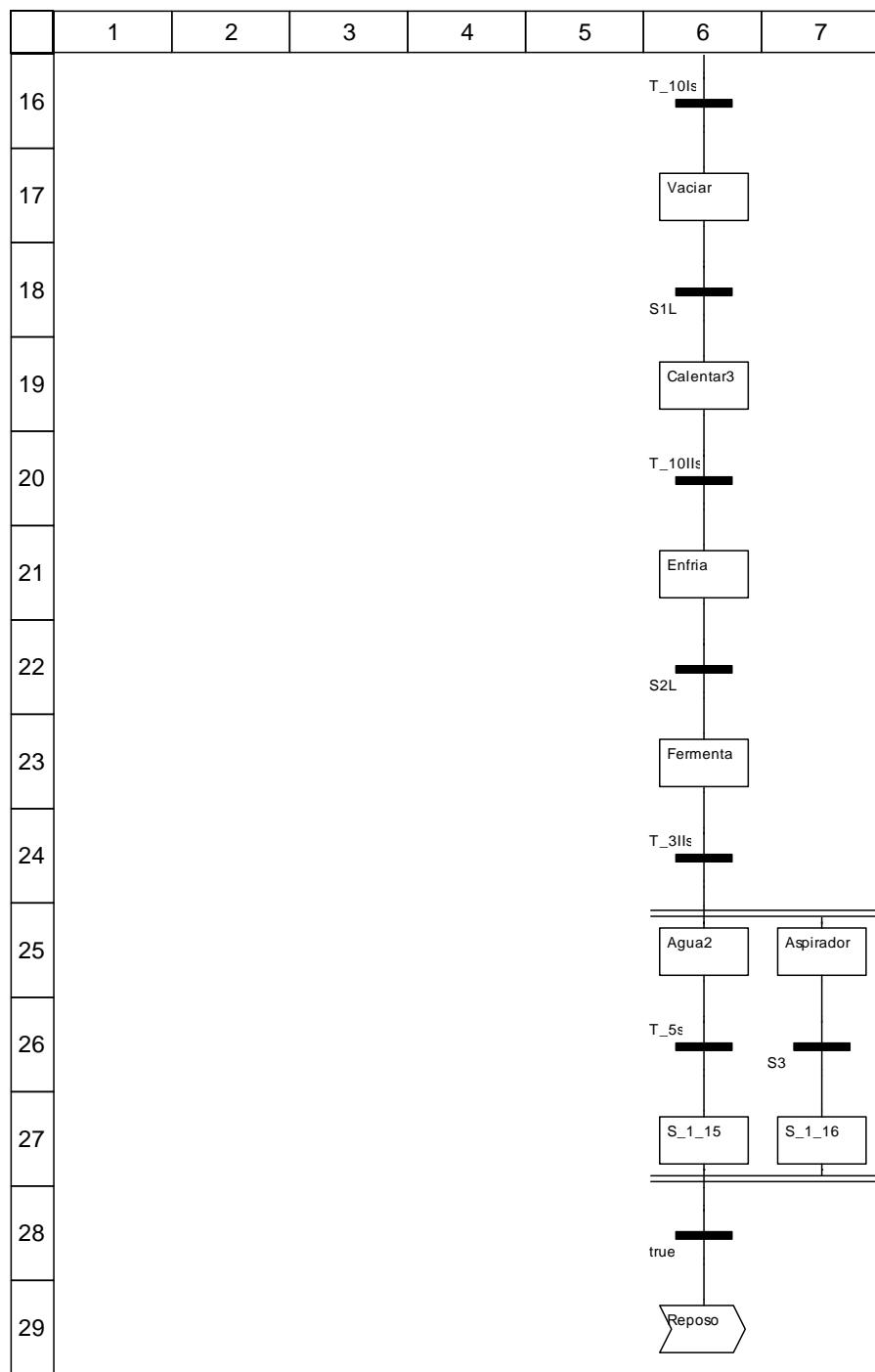
Nombre	Const	Dirección	Comentario	Valor	Utilizado	DG
A1	NO	%Q0.3.0		1	NO	
A2	NO	%Q0.3.1		1	NO	
A3	NO	%Q0.3.2		1	NO	
Inicio	NO	%I0.1.5		2	NO	
LU	NO	%Q0.2.15		2	NO	
M1	NO	%Q0.2.8		1	NO	
M2	NO	%Q0.2.9		1	NO	
M3	NO	%Q0.2.10		1	NO	
M4	NO	%Q0.2.11		1	NO	
M5	NO	%Q0.2.12		1	NO	
R1	NO	%Q0.2.13		2	NO	
R2	NO	%Q0.2.14		1	NO	
S1H	NO	%I0.1.0		1	NO	
S1L	NO	%I0.1.1		1	NO	
S2H	NO	%I0.1.4		3	NO	
S2L	NO	%I0.1.2		1	NO	
S3	NO	%I0.1.3		1	NO	
V1	NO	%Q0.2.0		2	NO	
V2	NO	%Q0.2.1		2	NO	
V3	NO	%Q0.2.2		1	NO	
V4	NO	%Q0.2.3		1	NO	
V5	NO	%Q0.2.4		1	NO	
V6	NO	%Q0.2.5		1	NO	
V7	NO	%Q0.2.6		1	NO	
V8	NO	%Q0.2.7		1	NO	

INT

Nombre	Const	Dirección	Comentario	Valor	Utilizado	DG
count	NO			4	NO	

Una vez definidas las variables, se procede a realizar la programación del sistema en diagrama funcional secuencial.





Los pasos, transiciones así como las variables que resultan del diagrama funcional secuencial, se detallan a continuación.

Abrir	(6, 11)
Tiempo de supervisión mín./máx.:	Tiempo de retardo:
Comentario:	
Acciones:	

Descriptor: N	Tiempo:	Variable: V2
---------------	---------	--------------

Agua	(6, 3)	
Tiempo de supervisión mín./máx.:	Tiempo de retardo:	
Comentario:		
Acciones:		
Descriptor: N	Tiempo:	Variable: V1

Agua2	(6, 25)	
Tiempo de supervisión mín./máx.:	Tiempo de retardo:	
Comentario:		
Acciones:		
Descriptor: N	Tiempo:	Variable: V6
Descriptor: N	Tiempo:	Variable: V5

Aspirador	(7, 25)	
Tiempo de supervisión mín./máx.:	Tiempo de retardo:	
Comentario:		
Acciones:		
Descriptor: N	Tiempo:	Variable: M5

Calentar	(6, 5)	
Tiempo de supervisión mín./máx.:	Tiempo de retardo:	
Comentario:		
Acciones:		
Descriptor: N	Tiempo:	Variable: R1

Calentar2	(6, 15)	
Tiempo de supervisión mín./máx.:	Tiempo de retardo:	
Comentario:		
Acciones:		
Descriptor: N	Tiempo:	Variable: R1

Calentar3	(6, 19)	
Tiempo de supervisión mín./máx.:	Tiempo de retardo:	
Comentario:		
Acciones:		
Descriptor: N	Tiempo:	Variable: R2
Descriptor: N	Tiempo:	Variable: M3
Descriptor: D	Tiempo: t#7s	Variable: A2

Enfria	(6, 21)			
Tiempo de supervisión mín./máx.:	Tiempo de retardo:			
Comentario:				
Acciones:				
Descriptor: N	Tiempo:	Variable: V4		
Descriptor: N	Tiempo:	Variable: V7		
Descriptor: N	Tiempo:	Variable: M4		

Fermenta	(6, 23)	
Comentario:		

Tiempo de supervisión mín./máx.:	Tiempo de retardo:			
Comentario:				
Acciones:				
Descriptor: N	Tiempo:	Variable: A3		
Descriptor: R	Tiempo:	Variable: LU		

Filtrado	(6, 13)			
Tiempo de supervisión mín./máx.:	Tiempo de retardo:			
Comentario:				
Acciones:				
Descriptor: N	Tiempo:	Variable: M2		
Descriptor: D	Tiempo: t#2s	Variable: V1		
Descriptor: S	Tiempo:	Variable: LU		

Malta	(6, 7)			
Tiempo de supervisión mín./máx.:	Tiempo de retardo:			
Comentario:				
Acciones:				
Descriptor: N	Tiempo:	Variable: A1		

Mezcla	(6, 9)			
Tiempo de supervisión mín./máx.:	Tiempo de retardo:			
Comentario:				
Acciones:				
Descriptor: L	Tiempo: t#2s	Variable: M1		
Descriptor: P1	Tiempo:	Sección: LD :: Count_inc		

Reposo (paso inicial)	(6, 1)			
Tiempo de supervisión mín./máx.:	Tiempo de retardo:			
Comentario:				
Acciones:				
Descriptor: P1	Tiempo:	Sección: LD :: Count_0		

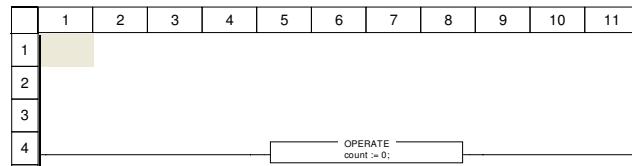
Reposo (paso inicial)	(6, 1)
Tiempo de supervisión mín./máx.:	Tiempo de retardo:
Comentario:	
Acciones:	
Descriptor: P1	Tiempo: Sección: LD :: Count_0
S_1_15	(6, 27)
Tiempo de supervisión mín./máx.:	Tiempo de retardo:
Comentario:	
S_1_16	(7, 27)
Tiempo de supervisión mín./máx.:	Tiempo de retardo:
Comentario:	
Vaciar	(6, 17)
Tiempo de supervisión mín./máx.:	Tiempo de retardo:
Comentario:	
Acciones:	
Descriptor: N	Tiempo: Variable: V2
Descriptor: N	Tiempo: Variable: V3
Vaciar2	(7, 3)
Tiempo de supervisión mín./máx.:	Tiempo de retardo:
Comentario:	
Acciones:	
Descriptor: N	Tiempo: Variable: V8

Transiciones:

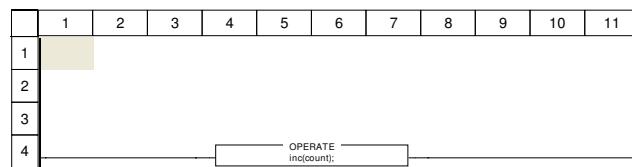
Nombre	Tipo de condición	Posición	Comentario
LD :: Lleno	Sección	(7, 2)	
LD :: No	Sección	(7, 10)	
S1H	Variable	(6, 4)	
S1L	Variable	(6, 18)	
S2H	Variable	(7, 4)	
S2L	Variable	(6, 22)	
S3	Variable	(7, 26)	
LD :: T_10IIs	Sección	(6, 20)	
LD :: T_10Is	Sección	(6, 16)	
LD :: T_10s	Sección	(6, 6)	
LD :: T_3IIs	Sección	(6, 24)	
LD :: T_3Is	Sección	(6, 12)	
LD :: T_3s	Sección	(6, 8)	
LD :: T_5Is	Sección	(6, 14)	
LD :: T_5s	Sección	(6, 26)	
LD :: Vacío	Sección	(6, 2)	
LD :: si	Sección	(6, 10)	
true	Constante	(6, 28)	

Veamos ahora, como se ha programado cada sección en lenguaje Ladder,

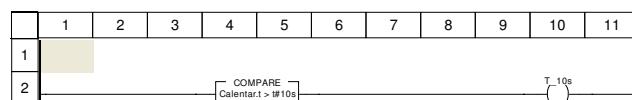
Count_0 <Acción>



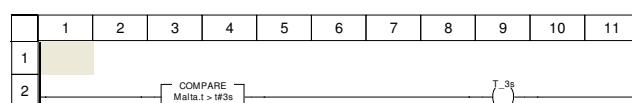
Count_inc <Acción>



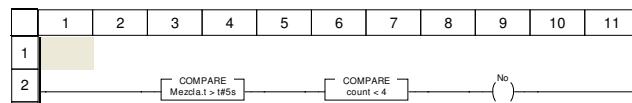
T_10s <Transición>



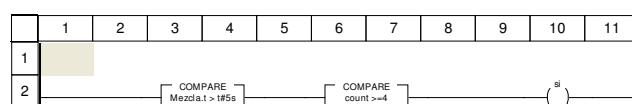
T_3s <Transición>



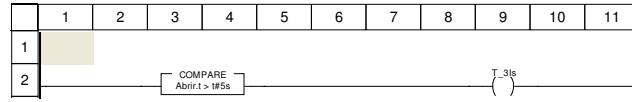
No <Transición>



si <Transición>

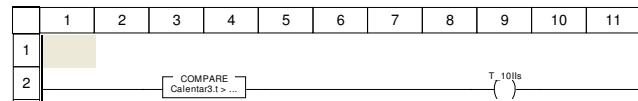


T 3Is <Transición>



T_10Is <Transición>

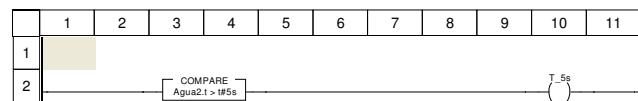
T_10IIs <Transición>



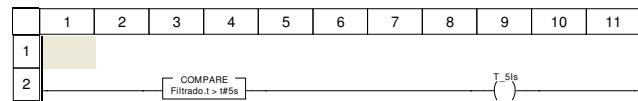
T_3IIs <Transición>



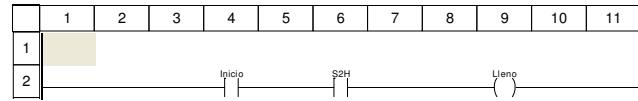
T_5s <Transición>



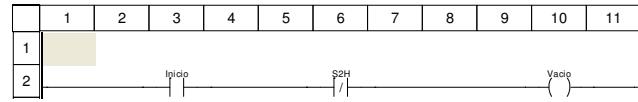
T_5IIs <Transición>



Lleno <Transición>



Vacio <Transición>



Capítulo 5

Enlaces a los ficheros fuentes y videos

Ficheros fuente

Los ficheros que contienen las soluciones de los problemas de este libro programados en Unity Pro se encuentran subidos a un repositorio GitHub: https://github.com/jccmontoya/libro_automatizacion_industrial. Dichos ficheros fuente pueden descargarse libremente y distribuirse bajo la Licencia Académica Social Robotics Lab - UC3M (LASR-UC3M), una licencia abierta no comercial que permite descargar, modificar y distribuir los ficheros siempre y cuando se distribuyan las fuentes originales.



Figura 5.1: Escanear para ir al repositorio.

Video tutoriales

Además, existen dos videotutoriales que se corresponden con el problema 3.3 de los ejercicios de diagramas de estados y con el problema 4.3 de los ejercicios resueltos de SFC.

El problema de diagramas de estados se ha resuelto usando la función de simulación de Unity Pro en lugar del autómata real. Este es el enlace al vídeo: <https://youtu.be/BcpNXYafDRQ>



Figura 5.2: Escanear para ver el videotutorial de diagramas de estados.

Por otro lado, el problema modelado mediante SFC se ha ejecutado sobre un autómata real. Así, se proporcionan ejemplos de ejecución en las dos opciones que proporciona la plataforma. Este es el enlace al vídeo: https://youtu.be/Q9li_oEibmY



Figura 5.3: Escanear para ver el videotutorial de SFC.