UNIVERSIDAD JAUME I

TRABAJO FIN DE GRADO

ESCUELA SUPERIOR DE TECNOLOGÍA Y CIENCIAS EXPERIMENTALES

DESARROLLO DE UN SIMULADOR RECONFIGURABLE DE UNA LÍNEA DE AZULEJOS Y EL CONTROL AUTOMÁTICO DE LA LÍNEA UTILIZANDO CODESYS

Autor:
Mihai Andrei Gherca
Director:
Roberto Sanchis Llopis

2017

ÍNDICE GENERAL

MEMORIA DESCRIPTIVA

Índice

1. OBJETO

El objetivo de este proyecto es diseñar un simulador reconfigurable de una línea de azulejos simplificada, para su uso posterior en ámbitos académicos. La herramienta desarrollada se pretende usar como complemento didáctico para el análisis de la interacción entre sensores y actuadores sencillos presentes en la industria cerámica.

Además del simulador se requiere el diseño del programa de control automático a nivel de PLC de todos los procesos existentes en la línea simulada. Gracias a la reconfigurabilidad el usuario tiene la posibilidad de desarrollar algoritmos de control alternativos, ya sea para todos los procesos en conjunto o por separado, y comprobar su funcionamiento.

2. ALCANCE

Tal como se deja indicado en el objeto del presente proyecto, el ámbito de aplicación de este simulador es exclusivamente didáctico. El simulador a diseñar es una herramienta gráfica que junta varios conceptos vistos en asignaturas de programación y sistemas de control automático, permitiendo además el análisis, la modificación o la creación de algoritmos para el control de un sistema de producción real como es el cerámico.

Como consecuencia surge la necesidad de diseñar un simulador sencillo e intuitivo con una interfaz amigable que llegue al usuario, permitiendo a este contemplar detalladamente los pasos seguidos durante el desarrollo del proyecto pero sobretodo las razones que han llevado a ensamblar el simulador de tal modo.

3. ANTECEDENTES

Dada la índole de este trabajo, las condiciones de partida son casi nulas teniéndose como base solo la idea inicial referente al proceso de producción a simular. La realización de actividades laborales en el sector cerámico local y los conocimientos de programación de autómatas programables (a partir de ahora PLC) adquiridos a lo largo de la carrera universitaria influyen de forma favorable en la realización de este proyecto.

Cabe destacar además que en las asignaturas relacionadas con la automatización los estudiantes solo pueden probar los controles diseñados por ellos mismos en las pocas maquetas de laboratorio, por lo que disponer de un simulador como el propuesto en este proyecto

puede mejorar mucho el aprendizaje de esta asignatura.

4. REQUISITOS DE DISEÑO

Una de las características imprescindibles del simulador en torno al cual se centra este proyecto es su representación gráfica. Es necesario que la simulación proporcionada con este proyecto sea lo más fidedigna posible en cuanto a concepto con una línea cerámica real, facilitando al usuario una visualización lo más intuitiva posible. Sin embargo es importante representar los elementos de forma simplificada evitando posibles confusiones en el funcionamiento de los procesos desarrollados.

En segundo lugar se exige que el simulador desarrollado en este proyecto sea reconfigurable, es decir, permita que mediante parámetros internos, a configurar ya sea antes o durante la ejecución, se obtengan simulaciones de distintas partes de una misma línea cerámica. Esta es una característica de alto interés también ya que aporta alta versatilidad en su uso, teniendo al alcance varios simuladores con solo cambiar un determinado número de parámetros.

Por último, un tercer requisito a cumplir enfoca la necesidad de usar un software que permita programar a nivel de PLC distintos controladores, que además cumplan con el estándar industrial internacional IEC 61131-3. De este modo el usuario puede programar sus propios algoritmos y probarlos en el simulador siempre siguiendo la norma, universalizando así el programa.

5. ANÁLISIS DE ALTERNATIVAS

Teniendo en cuenta los objetivos expuestos y los requisitos de diseño descritos las alternativas tangibles solo se pueden dar a nivel de software.

Una posible opción que cubre parte de las necesidades es emplear como software de partida Matlab, una herramienta generalmente matemática con lenguaje de programación propio. Aunque sea posible conseguir una interfaz gráfica gracias a las librerías que este software ofrece, el principal inconveniente de esta opción es la imposibilidad de programar un control automático con los lenguajes basados en la norma IEC 611313.

Otra posible alternativa es programar todo el entorno del simulador con el lenguaje de programación java, ya que es un lenguaje orientado a objetos, característica que puede

facilitar el desarrollo de la visualización en tiempo real. Aunque esta opción sea bastante más atractiva que la anterior, dejando de lado la complejidad a nivel de programación que puede implicar, se considera inviable conseguir que el simulador siga todas las pautas de la norma IEC 61131-3.

Por último la tercera opción es el software CodeSys, plataforma que proporciona un entorno de desarrollo específico para controladores lógicos programables, que además se rige por la norma IEC 611313.

6. VIABILIDAD

6.1. Viabilidad técnica

Analizados los objetivos a alcanzar junto con los requisitos anteriores, se considera que este proyecto es viable técnicamente ya que para su elaboración son necesarios recursos físicos disponibles o de fácil acceso, que junto con los conocimientos de programación adecuados pueden llevar a resultados satisfactorios.

6.2. Viabilidad económica

A nivel económico este proyecto se puede considerar viable ya que, teniendo en cuenta los objetivos académicos que se quieren alcanzar, da la posibilidad al estudiante a interactuar mediante el simulador con un proceso industrial real, evitando una posible inversión en maquetas de laboratorio para los mismos fines.

7. SOLUCIÓN ADOPTADA

7.1. Introducción

Para la realización de este proyecto se opta por el software CodeSys, dadas las ventajas que la plataforma ofrece, entre las cuales se destaca la definición interna de lenguajes de programación para PLC que siguen la norma IEC 61131-3. El software proporciona además las herramientas necesarias para crear elementos de visualización dinámicos de forma intuitiva, que además puede ejecutar en tiempo real en un PLC virtual que viene con su

instalación por defecto. Por ello se considera esta la opción más acertada, pretendiéndose conseguir un simulador como el propuesto sin elevada complejidad.

7.2. Estructura del programa

Considerando los requisitos de diseño y parte de los objetivos, es necesario desarrollar un programa que gestione tanto la parte del simulador, asociada directamente con la visualización representada y los cambios que esta sufre, como la parte del control automático también exigida. Por ello el conjunto que forma este programa se divide en distintas unidades de organización del programa (a partir de ahroa POU) que son recorridas sin discriminación alguna en cada ciclo del maintask.

7.3. Desarrollo del simulador

7.3.1. Descripción general

El simulador es el objetivo principal de este proyecto, ya que gracias a esta parte del programa se consigue mostrar de forma interactiva el comportamiento de los sensores y actuadores introducidos, así como los desplazamientos y tratamientos sufridos por las piezas cerámicas durante su paso por la línea de producción.

Por otro lado es la parte más compleja en cuanto a programación, no sólo por el hecho de conseguir la correcta visualización descrita en el párrafo anterior, sino además por la necesidad de interpretar las órdenes del usuario en cuanto a la reconfigurabilidad, cambiando así según el caso los elementos mostrados y controlados por el sistema.

Otra de las características destacables del simulador es su universalidad en cuanto a los algoritmos del control automático, función imprescindible dado el otro objetivo del proyecto: el poder implementar y probar alternativas al control automático propuesto.

A lo largo de los siguientes puntos del apartado 7.3 se va a describir de forma detallada tanto los elementos de la visualización como el código que hace posible la evolución de dicha visualización, además de los bloques de código que permiten la reconfigurabilidad.

7.3.2. Estructura de la visualización

La visualización es uno de los elementos fundamentales del simulador, ya que es el mecanismo responsable de representar los elementos reales involucrados en la simulación en

marcha (procesos y azulejos). Además, a través de la visualización el usuario puede introducir órdenes puntuales al sistema; operación imprescindible para conseguir un análisis adecuado.

En CodeSys la visualización se crea como un objeto aparte de la aplicación en marcha. En la visualización se introducen los elementos uno a uno, empleando el panel de visualización que permite escoger qué tipo de elemento se quiere introducir así como sus propiedades (nombre, posición, dimensiones, visibilidad, desplazamientos, etc.).

La visualización de este proyecto se consigue a partir de elementos sencillos: generalmente figuras geométricas simples, elementos indicativos (como lámparas) y elementos que permiten introducir órdenes puntuales (interruptores de varias clases); todos son elementos que el software con su instalación básica lleva incorporados. Además, para algunas órdenes puntuales se generan elementos a partir de figuras sencillas introducidas por el programador. Para la mayoría de los casos los elementos se juntan en grupos para facilitar el acceso a sus propiedades y características, así como para asignar con mayor facilidad a una variable la característica de todos los elementos agrupados.

Tras su creación los elementos se organizan en una tabla situada en la parte de definiciones del objeto Visualización, llamada Lista de elementos donde se puede acceder a algunas características de los elementos presentes en la visualización. En la figura 1 aparece una parte de la tabla que proporciona el software, la cual muestra características de algunos elementos así como la pertenencia de esos elementos a varios grupos. Cabe destacar que las coordenadas X e Y de los elementos pertenecientes a grupos son relativas a la posición del grupo entero. Las tablas completas con todos los elementos de la visualización del simulador aparecen en el Anexo 3.

Tipo	X	Y	Ancho	Altura	ID	Nombre
#0 Group	354	306	216	239	675	Ca
#0 Group	14	149	100	90	357	ControlCa
#0 Rectángulo	9	0	112	124	356	ContornoControlCa
#1 Interruptor de e...	74	5	57	39	240	PulsadorMarchaCa
#2 Interruptor de e...	74	44	57	39	344	InterruptorParoCa
#3 Rotulación	0	6	82	30	345	Txt. Marcha Ca
#4 Rotulación	0	46	82	30	349	Txt. Paro Ca
#5 Interruptor de p...	74	84	47	39	353	Interruptor Carga Ca
#6 Rotulación	0	86	82	30	355	Txt. Carga Ca
#1 Group	16	6	200	190	674	CmpA
#0 Rectángulo	94	0	80	190	3	Chásis

Figura 1: Extracto de Lista de elementos.

7.3.3. Elementos de la visualización

Como es de esperar, aunque se trate de un modelo simplificado, la visualización está compuesta por decenas de elementos agrupados de la mejor forma para facilitar tanto su configuración como su comprensión posterior. Para fidelizar con la estructura y la colocación real de los diferentes procesos en una planta de producción cerámica, cada conjunto de elementos es posicionado y dimensionado de tal modo que genere en el usuario la impresión de continuidad entre los procesos y la verosimilitud de estos, además de proporcionar la información necesaria para un análisis apropiado.

En los subapartados siguientes se describen detalladamente algunos elementos y procesos presentes en la visualización haciendo una analogía con elementos y procesos genéricos presentes en líneas de producción reales.

7.3.3.1 Azulejos

Uno de los elementos básicos que ha de estar presente en la presente visualización son los azulejos, es decir, el producto que sufre transformaciones a lo largo de la linea de producción, y que es resultado final de esta.

Dada la sencillez de la pieza cerámica en sí, en la visualización esta es representada por un rectángulo simple, con bordes definidos, tal como se muestra en la figura 2. Las dimensiones de este elemento aparecen en la tabla 1, siendo estas relativamente exageradas para proporcionar una representación óptima dadas las propiedades de visualización de CodeSys y la resolución de visualización considerada adecuada y universal para cualquier usuario.

Figura 2: Representación del azulejo.

X (px)	Y (px)	Ancho (px)	Altura (px)
308	341	60	5

Tabla 1: Posición inicial y dimensiones del elemento azulejo.

En la misma tabla también se muestra la posición absoluta de las piezas antes de ser posicionadas en la primera ejecución; esta acción se explicará con más detalle en el apartado 7.3.5.2.

7.3.3.2 Compenser

El compenser es uno de los procesos más importantes de la línea de producción, no por afectar la calidad del producto final, sino por ser decisivo en cuanto a la productividad de la línea, dada su capacidad de almacenamiento temporal de piezas. De este modo, en caso de avería y necesidad de parada aguas abajo del compenser, este puede almacenar piezas procesadas aguas arriba, permitiendo no parar toda la línea, evitando posibles puestas en marcha costosas y duraderas.

Figura 3: Compenser cerámico.

Para este proyecto se decide implementar un compenser sencillo, cuyo control se realiza usando fotocélulas y motores de velocidad constante. En la figura 4 se muestra la representación que se le da al compenser en esta simulación. La tabla 2 contiene además las dimensiones y el posicionamiento de los elementos que forman el proceso.

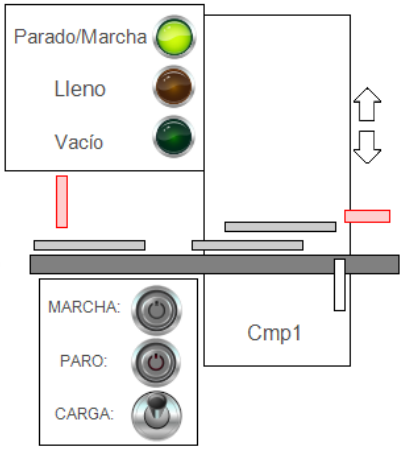

Figura 4: Representación del compenser.

Nombre	X (px)	Y (px)	Ancho (px)	Altura (px)
Compenser	354	306	100	90
Proceso	16	6	200	190
Chásis	94	0	80	190
CintaCa	14	87	6	28
Fot1	14	87	28	5
Fot2	165	132	6	28
SInd	171	132	6	28
Texto	113	157	41	30
Flch.Up Ca	174	39	20	20
Flch.Dow Ca	174	62	20	20
Control	14	149	100	90
Contorno	9	0	112	124
MarchaM	74	4	57	39
ParoM	74	44	57	39
Txt. Marcha	0	6	82	30
Txt. Paro	0	46	82	30
CargaM	74	84	47	39
Txt. Carga	0	86	82	30
Panel	0	0	110	90
Contorno	2	0	105	90
Lamp. M	83	6	25	25
Lamp. P	83	6	25	25
Txt. M/P	0	4	84	28
Lamp. Vacío	79	32	31	28
Txt. Vacío	0	32	84	28
Lamp. Lleno	79	61	31	28
Txt. Lleno	0	61	84	28

Tabla 2: Dimensiones del compenser.

Para mayor comprensión de la visualización y del funcionamiento del compenser se hace necesario enumerar algunas de las entradas y salidas que el controlador necesita para este proceso. Las tablas 7 y 8 muestran las entradas y salidas necesarias para el control del compenser.

Algunos de los elementos de la figura 4 cambian sus propiedades a lo largo de la simulación, cambiando su apariencia cuando el elemento que representan se activa. Para el compenser estos objetos son los relacionados con las señales: Fot1, Fot2, SInd, MCinta, MCSubir y MCBajar.

Los elementos relacionados con las fotocélulas cambian su color cuando, según el caso, la pieza alcanza una posición determinada en la visualización. Así, para las primeras dos señales (Fot1 y Fot2), cuando el extremo derecho del azulejo haya rebasado la posición de la esquina izquierda de las fotocélulas, se colorean dichos elementos, volviendo a su estado inicial cuando la pieza haya avanzado lo suficiente hasta no intersectar en la proyección horizontal del sensor. El caso del sensor inductivo SInd es idéntico pero con desplazamientos

verticales de las baldosas.

Cuando se activa la señal MCinta es porque el motor debe girar y transmitir movimiento a la cinta, haciendo por tanto que los azulejos posicionados en la parte superior a esta avancen horizontalmente. Además del movimiento de las piezas, la activación de la cinta se representa haciendo cambiar de color al elemento que simboliza a esta.

De forma análoga al caso anterior, las señales MCSubir y MCBajar hacen que las piezas situadas justo en la posición horizontal de la Fot2 asciendan o desciendan verticalmente. En este caso, dada la representación del proceso, no se puede asociar dicho movimiento a ningún elemento en concreto; por ello se añaden dos flechas indicadoras que se colorean cuando la acción correspondiente a su dirección es activada.

En cuanto a su funcionamiento en bucle cerrado, las piezas llegan al compenser por la parte izquierda y son transportadas a través de este gracias a la cinta con desplazamiento horizontal que es movida por el motor MCinta.

Si se tiene la orden de cargar, orden definida por la entrada CargaM o por la parada de los procesos siguientes, la pieza al llegar a la fotocélula 2 (Fot2) es subida por el motor MCSubir al compenser sólo si hay espacio, es decir, si el compenser no está lleno. El motor MCSubir sube la pieza hasta llegar al sensor inductivo, SInd, que es cuando para. En caso de que el compenser llegue al límite de almacenamiento este para la cinta y activa la salida Lleno.

Por el contrario, para casos de descarga, cuando CargaM está desactivada o funcionamiento normal, el compenser si tiene piezas almacenadas, ha de asegurarse que tiene espacio para bajar una pieza, para no provocar colisiones ni daños en el producto. Si existe dicho espacio, la pieza es bajada accionando el motor MCBajar hasta que se alcanza la altura de la cinta, detectada por Fot2. En caso de que se lleguen a descargar todas las piezas, se activa la salida Vacio.

Por último, en caso de que el compenser se pare, la salida Parado se activa.

Como se puede comprobar en la figura 4, las entradas Marcha, Paro y CargaM se pueden accionar mediante pulsadores e interruptores. Además, en la parte superior de la imagen, el panel muestra mediante luces el estado del compenser tanto en cuanto a funcionamiento (Parado con luz roja y no parado con luz verde) como en cuanto a capacidad (Lleno o Vacio).

La forma que se le da al compenser en esta visualización no está fija, por lo general, a ninguna relación en cuanto a dimensiones y posicionamientos se refiere. Así, el único factor

del que se parte para dimensionar el proceso es la longitud y el espesor de los azulejos (tabla 1).

Para el posicionamiento de estos elementos en cambio hay que prestar especial atención al funcionamiento de la instalación. Para el cálculo del espacio necesario para realizar una descarga, el controlador necesita relacionar las posiciones de las dos fotocélulas y el sensor inductivo y ser capaz en función de estos de calcular dicho espacio. Un posicionamiento erróneo de las fotocélulas puede conllevar una menor eficiencia, haciendo que el espacio necesario calculado por el controlador sea superior al real. Este punto se desarrolla con más detalle en la descripción del control automático del compenser (apartado 7.4.2).

7.3.3.3 Proceso 1

El Proceso 1, a diferencia de los casos anteriores, no tiene necesariamente un homólogo en una línea de producción real. El diseño de este proceso está enfocado en representar una multitud de procesos reales que tengan la misma dinámica, diseñando así para este caso una representación genérica.

En cuanto a su representación, el Proceso 1 aparece representado en la figura 5; las características de cada elemento vienen especificadas en la tabla 3.

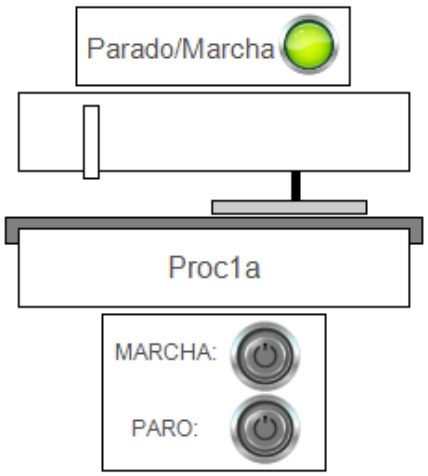

Figura 5: Representación del Proceso 1.

Elemento	X (px)	Y (px)	Ancho (px)	Altura (px)
Proceso 1	106	361	160	178
Proceso	0	33	160	82
Chorro	110	28	3	13
Cinta	0	48	160	10
Base1	5	52	150	30
Base2	5	0	150	30
Fotocélula	30	5	6	28
Control	30	118	100	60
Contorno	7	0	85	60
Pulsador M	56	3	44	28
Interruptor P	56	31	44	28
Txt. Marcha	0	4	63	21
Txt. Paro	0	33	63	21
Panel	25	0	110	30
Contorno	2	0	120	35
Lamp. M/P	90	3	36	30
Txt. M/P	0	3	96	30

Tabla 3: Dimensiones del Proceso 1.

Análogamente al caso anterior, para una correcta comprensión del funcionamiento es necesario tener en cuenta qué entradas y salidas se asocian a este proceso (tablas: 9 y 10 respectivamente).

Igual que para la representación del compenser, algunos elementos de la figura 5 cambian su apariencia en función de su estado; en este caso hay dos elementos que marcan su activación cambiando de color: la fotocélula 1 (Fot1) y la cinta (MCinta), y el elemento que representa el tratamiento en sí se vuelve visible cuando la señal Tratamiento está activa.

En bucle abierto, el funcionamiento de la cinta y la fotocélula es idéntico al descrito en el caso del compenser: en el primer caso Fot se activa coloreándose solo cuando la pieza intersecta con la proyección horizontal del elemento fotocélula; de forma análoga cuando se activa el motor con MCinta las piezas que se encuentran encima de la cinta perteneciente al Proceso 1 se mueven horizontalmente, además la apariencia del elemento que representa a la cinta cambia.

El elemento relacionado con la señal Tratamiento simplemente cambia de visibilidad, pa-

sando a ser visible cuando dicha señal está activada.

El funcionamiento en conjunto de este proceso es bastante intuitivo dada su sencillez: las piezas llegan al Proceso 1 por la izquierda transportadas por la cinta que debe su movimiento al motor accionado por MCinta. A continuación la pieza alcanza la posición de la fotocélula 1, y sigue avanzando por la cinta hasta recibir la aplicación del tratamiento correspondiente al Proceso 1. Este tratamiento es activado por el controlador gracias a la salida Tratamiento, pero además se activa un tiempo determinado después de que la baldosa cerámica haya alcanzado la fotocélula.

Las entradas MarchaM y Paro se pueden accionar mediante un pulsador y un interruptor respectivamente. Además, el panel muestra mediante luces el estado del Proceso 2 en cuanto a funcionamiento, de forma similar al caso anterior.

De nuevo las dimensiones y posiciones de los elementos son las necesarias para garantizar una correcta interpretación y cohesión espacial por parte del usuario. En este caso, el posicionamiento de la fotocélula depende de la posición de aplicación del tratamiento debido al control automático empleado para este proceso, descrito en el apartado 7.4.3.

7.3.3.4 Proceso 2

El siguiente proceso a describir es el Proceso 2, proceso cuyo funcionamiento es bastante más complejo en comparación con el proceso anterior. En este caso tampoco se ha representado un proceso real determinado, pero dada la dinámica del proceso la representación se puede asociar con procesos que realizan estampados sobre la pieza cerámica empleando un cilindro neumático. Esta última aclaración es muy importante, dado que conlleva la necesidad de parada de la cinta durante dicho estampado.

El Proceso 2 aparece representado en la figura 6 y las características de cada elemento vienen especificadas en la tabla 4.

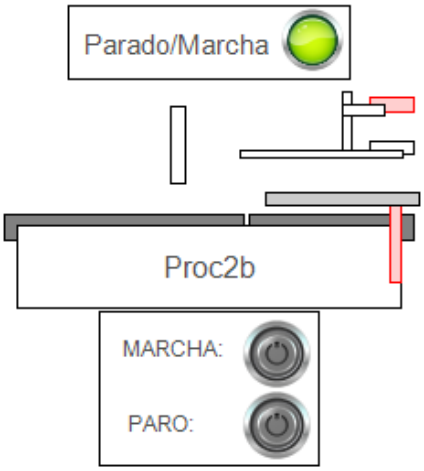

Figura 6: Representación del Proceso 2.

Nombre	X (px)	Y (px)	Ancho (px)	Altura (px)
Proceso 2	106	361	160	184
Proceso	0	34	160	86
Base	5	52	150	32
Cinta1	96	48	64	10
Cinta2	0	48	98	10
Base2	5	0	150	30
Fotocélula 1	65	6	6	30
Fotocélula 2	151	44	4	30
S. Inductivo 1	143	2	17	5
S. Inductivo 2	143	19	17	5
Base émbolo	92	27	64	3
Vástago	132	4	4	23
Metal	132	9	17	4
Control	30	123	107	61
Contorno	7	0	85	60
Pulsador M	63	3	44	28
Interruptor P	62	31	44	28
Txt. Marcha	4	8	63	21
Txt. Paro	0	33	63	21
Panel	25	2	114	30
Contorno	0	0	110	30
Lamp. M/P	83	3	31	26
Txt. M/P	0	1	85	30

Tabla 4: Dimensiones del Proceso 2.

Se requiere un análisis de las entradas y las salidas mostradas en las tablas 11 y 12 para un adecuado seguimiento de la explicación posterior.

Dada la mayor complejidad del proceso, en este caso hay múltiples elementos que sufren cambios durante la visualización, afectando estos cambios no solo a la apariencia de los elementos, sino que también en ciertos casos cambiando su posición, creando un desplazamiento dentro del mismo proceso. Igual que en los casos anteriores, las fotocélulas, las cintas y los sensores inductivos cambian de apariencia cuando se activan; pero además para el Proceso 2, se representa el movimiento vertical del émbolo del cilindro neumático mencionado en la descripción.

Las señales MCinta1L y MCinta1R controlan el desplazamiento de los azulejos existentes encima de la primera cinta, cinta izquierda, haciendo que estas piezas avancen de forma lenta o rápida según la señal activada; además al estar activada cualquiera de estas señales el elemento que representa la primera cinta cambia de apariencia. La segunda cinta tiene un funcionamiento similar siendo controlados los desplazamientos con las señales MCinta2L y MCinta2R.

El funcionamiento de las fotocélulas es igual que en los casos anteriores, activándose en presencia de azulejos. De forma similar, los sensores inductivos (SInd1 y SInd2) se activan cuando el desplazamiento del elemento Metal intersecta con la proyección lateral de dichos sensores.

En cuanto a las señales Asc y Desc, cuando estas son activadas el conjunto de elementos formados por Base émbolo, Vástago y Metal se desplaza verticalmente, de forma ascendente o descendente respectivamente.

En cuanto al funcionamiento, las piezas llegan al proceso por la parte izquierda transportadas por la cinta movida por el motor de velocidad variable controlado por las salidas del controlador MCinta1L y MCinta1R. Tras activar Fot1 la pieza es acelerada pasando a desplazarse a una velocidad superior a la inicial tal que garantice la posibilidad de realizar la parada necesaria para el estampado realizado por el émbolo, sin provocar colisiones posteriores. Por tanto el motor correspondiente ha de poder variar su velocidad pasando de un valor a otro según la señal activada por control.

Como se puede observar este proceso está diseñado con dos cintas distintas. La segunda de ellas controlada por el motor MCinta2L y MCinta2R ha de ser capaz de transportar la pieza cerámica a la velocidad alta marcada por la cinta anterior, para que tras alcanzar la posición de Fot2 se pare el tiempo necesario para la realización del estampado. Tras recibir el estampado la pieza ha de ser desplazada a velocidad lenta por la misma cinta hasta ser arrastrada por el siguiente tramo de la línea. Por tanto el motor ha de poder conmutar a tres estados distintos: lento (MCinta2L), rápido(MCinta2R) y parado.

El cálculo detallado del tiempo de parada mencionado en el párrafo anterior se detalla en el apartado 7.4.4.

En cuanto al desplazamiento del cilindro, este es controlado por las salidas Desc para descender y Asc para ascender del controlador. Cuando la posición de la pieza en la segunda cinta activa la Fot2 se hace descender el émbolo hasta activar el sensor SInd2, descenso justo para la aplicación del estampado considerado en este proyecto instantáneo. Luego se realiza el ascenso hasta activar el SInd1 que garantiza que el émbolo está a la altura suficiente para no afectar a la línea y preparado para el estampado de la pieza siguiente.

El diseño del Proceso 2, debido a la variabilidad de velocidades del proceso y la necesidad de no colisionar las piezas, ha de ser muy meticuloso en cuanto al posicionamiento de elementos. Asimismo para no alterar la disposición de las piezas tras el paso por este proceso, en cuanto a espaciado entre ellas se refiere, es necesario que la distancia entre la Fot1 y la Fot2 coincida exactamente con la longitud de una pieza más el espaciado por defecto con la siguiente.

Otro aspecto a destacar es el posicionamiento de los sensores SInd1 y SInd2, que marcan el recorrido del émbolo neumático, ya que hay que conseguir que el desplazamiento del émbolo sea el justo para realizar el estampado adecuadamente sin dañar la pieza. Estos posicionamientos se pueden comprobar en la tabla 4.

7.3.3.5 Otros elementos

Además de los elementos y procesos explicados en los párrafos anteriores en la visualización hay unos pocos elementos más que se explican en este punto.

En primer lugar, y aunque por su semejanza con los elementos de los procesos es de fácil deducción, mencionar las cintas que transportan las piezas entre procesos. Su función a nivel de visualización es más bien estética, dando la impresión de una línea de producción más alargada, y por tanto más real.

Estas cintas son movidas por motores exclusivos, aunque las señales de estos motores dependen del consentimiento de los procesos posteriores en la línea. Por ello se opta por no crear señales dedicadas a cada cinta extra, sino relacionarlas directamente con las señales de las cintas de los procesos posteriores. Así, la señal de la cinta anterior a un proceso en concreto es calculada en el mismo POU de control del proceso como MCintaAnt, y relacionada directamente con la cinta de entrada a dicho proceso. Dada

la reconfigurabilidad del simulador que se explica en el apartado 7.3.4, la presencia de las cintas se ve afectada por la configuración inicial, y por ello se decide simplificar la programación aplicando el método anterior.

En la tabla 5 se muestran todas las cintas que pueden aparecer en la simulación, a parte de las incluidas en los procesos descritos, además de las señales del controlador relacionadas con los POUs correspondientes según la configuración de partida.

Cinta	**X (px)**	**Y (px)**	**Ancho (px)**	Altura**(px)**	**Señal**
CintaA	4	442	364	10	Variables.MCintaAntCa
CintaA1	4	442	100	10	Variables.MCintaAntP1a o Variables.MCintaAntP2a
CintaA2	268	442	100	10	Cmp1a.MCintaAntCa
CintaB	572	442	364	10	Cmp1b.MCintaAntCb
CintaB1	572	442	100	10	Variables.MCintaAntP1b o Variables.MCintaAntP2b
CintaB2	836	442	100	10	Variables.MCintaAntCb

Tabla 5: Características de las cintas extra.

Por último, otro elemento presente en la visualización en el cual hay que hacer hincapié es el interruptor Operario. La función de este interruptor es la de iniciar la acción virtual de un operario humano que restablece la posición de las piezas tras una parada, con el fin de garantizar el correcto funcionamiento de los procesos aguas abajo en la línea.

La intervención del operario es necesaria, ya que en casos de parada en un proceso todos los procesos y cintas precedentes a este hasta el primer compenser son parados, mientras que el primer compenser recibe la orden de cargar mediante el no consentimiento. En dicho caso, dada la parada inmediata de las cintas posteriores al compenser, puede haber una pieza que haya superado la posición de carga definida para el compenser, y ser empujada sobre la cinta parada pudiendo impactar con piezas de su proximidad.

De forma análoga, dado que a lo largo de la línea pueden haber procesos en los cuales hayan acciones temporizadas o velocidades variables, en caso de parada dichos procesos entran en Parada, perdiendo el progreso de la acción o tratamiento realizado sobre la pieza cerámica. Por ello, la acción Operario extrae dichas piezas de estos procesos para garantizar la calidad adecuada de toda la producción.

El interruptor implementado sólo funciona cuando haya algún proceso parado (ya sea por paro manual o por saturación del compenser), dado que representando la acción de un operario humano es condición necesaria por cuestiones de seguridad.

Por tanto, si se acciona este pulsador y hay algún proceso parado se quitan las piezas que han impactado o que afectan negativamente a los procesos parados.

7.3.4. Reconfigurabilidad

La reconfigurabilidad es una de las características más importantes del simulador, dado que es la propiedad que consigue obtener a partir de un único código de simulador varios sistemas de complejidad variada según las preferencias del usuario. Como consecuencia de lo anterior el usuario puede, a parte de analizar el funcionamiento en sí de la configuración definida, diseñar y probar el funcionamiento de alternativas al control automático, sin necesidad de involucrar en el nuevo algoritmo todos los posibles elementos del simulador.

La visualización diseñada para este simulador es dividida en dos partes: la parte A y la parte B, tal como se muestra en la figura 7. La parte A está compuesta por, opcionalmente el Proceso 1 o el Proceso 2 (descritos en el apartado 7.3.3), y necesariamente por un compenser; es decir, la parte A, aunque se configure sin ninguno de los procesos genéricos, siempre contará con un compenser. A estos elementos se va a hacer referencia a partir de ahora como Compenser a, Proceso 1a y Proceso 2a.

La parte B, de forma análoga a la anterior, puede estar compuesta opcionalmente por el Proceso 1, el Proceso 2 y el compenser (pasándose a llamar a partir de ahora Compenser b, Proceso 1b y Proceso 2b); pero a diferencia del caso A, aquí se da la posibilidad de no escoger ningún elemento, anulando así la parte B por completo.

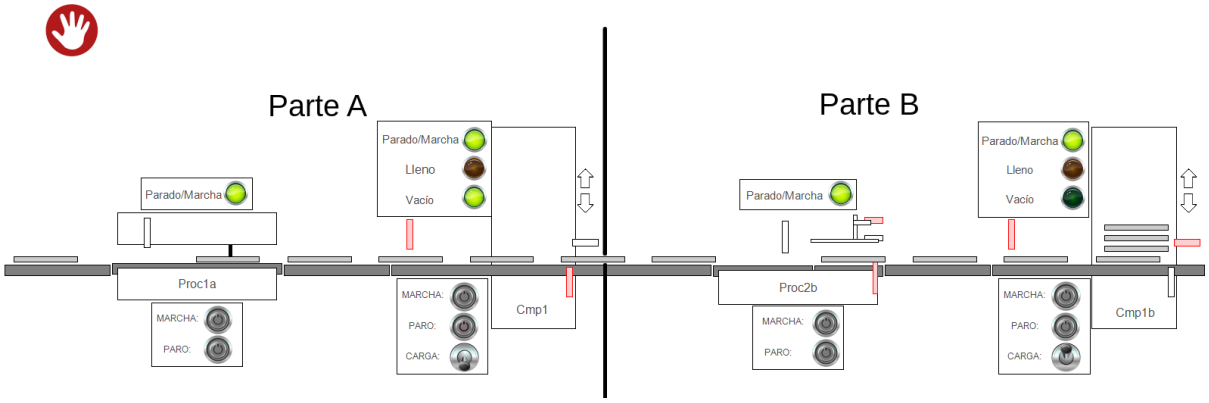

Figura 7: Partes *a* y *b* de la visualización.

El usuario define la configuración deseada accediendo a la parte de declaraciones del POU del simulador, el POU llamado en el software Sim (PRG). Así, se puede optar por varias opciones de visualización modificando los valores de un vector de tres enteros mostrado en el código 1.

```
1 (*RECONFIGURABILIDAD: *)
2   elem: ARRAY[0..3] OF INT := [1, 2, 2];
```

```
3     (*[a, b, c] ->  a = 0, no hay Compenser b
4             a = 1, hay Compenser b
5             b = 0, no hay Proceso en a
6             b = 1, hay Proc1 en a
7             b = 2, hay Proc2 en a
8             c = 0, no hay Proceso en b
9             c = 1, hay Proc1 en b
10            c = 2, hay Proc2 en b
11    *)
```

Código 1: Declaración de la reconfigurabilidad

En la línea 2 se define el vector, pero además este es inicializado con los valores correspondientes a la configuración preferida de acuerdo con la tabla 6. Para cualquier otra combinación, los valores del vector se considerarán [0, 0, 0].

Parte	Proceso configurado	Elemento del vector	Valor
A	-	b	0
A	*Proceso*1*a*	b	1
A	*Proceso*2*a*	b	2
B	-	c	0
B	*Proceso*1*b*	c	1
B	*Proceso*2*b*	c	2
B	-	a	0
B	*Compenserb*	a	1

Tabla 6: Posibilidades de configuración del simulador.

7.3.5. Código del simulador

Como ya se ha introducido en el apartado 7.2, el simulador como tal se diseña como otro POU más junto a los de control de los procesos. A lo largo de este punto se va a describir con detalle cada bloque perteneciente a este POU.

En el Anexo 1 aparece el código completo del simulador, dividido entre la parte correspondiente a las declaraciones y la parte de código.

7.3.5.1 Declaraciones

Reconfigurabilidad: El primer bloque, uno de los más importantes, es el de Reconfigurabilidad ya explicado en el apartado 7.3.4, y mostrado en el código 1.

Otros parámetros: Recorriendo el código, el siguiente bloque engloba las constantes que definen la visualización. Estas constantes son generalmente distancias respecto a la posición de partida de los azulejos, posición absoluta de la visualización, definida en la tabla 1. No se considera necesario mostrar este bloque en este apartado dada su extensión y su nomenclatura.

Una constante destacable del bloque anterior es tsim, mostrada en el código 2. Este valor, aunque sea definido en este punto como REAL, es el que marca el periodo de la visualización, es decir, cada cuantos milisegundos se actualizan los elementos (ya sea su posición, su estado, etc.). Es muy importante que para la ejecución correcta del simulador este tiempo sea siempre superior al tiempo establecido en el maintask.

```
1  (*Parametros de la visualizacion: *)
2    (*Ciclo de simulacion: *)
3      tsim: REAL := 15; (*REAL para realizar operaciones*)
```

Código 2: Configuración del periodo de simulación.

Las siguientes definiciones del código están relacionadas con la activación de los actuadores de los procesos. Como se ha mencionado en la descripción de los procesos, en caso de que algún actuador (generalmente motor) esté activo el elemento al que afecta (generalmente cinta) cambia de color. Por tanto se necesitan variables booleanos que definan la apariencia de dichos elementos.

La siguiente sección de código engloba los vectores de posición e invisibilidad asociados a los azulejos, así como un vector auxiliar empleado para la implementación de la reconfigurabilidad.

En penúltimo lugar aparecen algunas variables, tanto BOOL como REAL asociadas a los tratamientos y desplazamientos de los procesos; así como variables auxiliares para la inicialización y para la implementación de Operario.

Temporizador: Por último, imprescindible mostrar la declaración del temporizador empleado para la temporización de la simulación, es decir, el temporizador que marca

el periodo de actualización de la visualización. Tal como queda indicado en el código 3, se trata de un temporizador TP; aunque como se verá en la descripción del código del simulador, dada la forma de programación el tipo de temporizador es irrelevante.

```
1 (*Temporizador *)
2         TP_while: TP;
3         S_t_while: BOOL;
4         PT_t_while: TIME := REAL_TO_TIME(tsim);
5         Q_t_while: BOOL;
6         ET_t_while: TIME;
7         t_while: BOOL;
```

Código 3: Declaración del temporizador

Además del temporizador se definen las variables que se asociarán a este. El tiempo preestablecido, PT, del temporizador se define con la variable PT_t_while a la que se le asigna el valor de tsim tras la conversión correspondiente.

7.3.5.2 Código

En cuanto a la parte de código en sí, en este subapartado se explicará de forma similar, dividiendo por bloques, las comprobaciones, actualizaciones y operaciones realizadas.

Inicialización: Se considera como primer bloque la parte de inicialización dado que esta sólo se ha de ejecutar la primera vez y está fuera de cualquier temporización. Por ello, una forma sencilla de programar esto es condicionando todo el bloque a una variable BOOL llamada aquí Inicializado, comprobándose su valor al comienzo y poniéndose a TRUE al llegar al final del bloque.

```
1 (*Inicializacion de los vectores *)
2   IF NOT Inicializado THEN
3     FOR i := 1 TO b BY 1 DO
4       posx[i] := posx[0]-i*d;
5       posy[i] := 0;
6       inv[i] := 0;
7     END_FOR
```

Código 4: Inicialización de azulejos.

En este bloque, en primer lugar tal como aparece en el código 4 se posiciona el número total de azulejos (b) según las características de espaciado definidas para los procesos: en función de la distancia entre un mismo punto homólogo de todos los azulejos (d); también se asegura de que las piezas comienzan con cota cero. Además se hacen todas las piezas visibles poniendo a 0 el vector de invisibilidad.

A continuación se procesa el vector de enteros definido en el paso anterior, tal cual se muestra en el código 5, para conseguir las opciones descritas en la tabla 6. En la segunda parte del código se inicializa la variable limh, que es la variable que marca el desplazamiento máximo sufrido por las piezas antes de ser recolocadas (esto se explica con más detalle en la programación de la recolocación).

```
1  (*RECONFIGURABILIDAD: *)
2    (*Inicializa el vector de visibiliadad a 0: *)
3      FOR i := 0 TO 8 BY 1 DO visElem[i] := 0; END_FOR
4      IF elem[0] = 1 THEN visElem[0] := 1; ELSE visElem[0] := 0; END_IF
5      IF elem[1] = 0 THEN visElem[7] := 1;
6      ELSIF elem[1] = 1 THEN visElem[5] := 1; visElem[1] := 1;
7      ELSIF elem[1] = 2 THEN visElem[5] := 1; visElem[2] := 1;
8      ELSE ; END_IF
9      IF elem[2] = 0 AND elem[0] = 1 THEN visElem[8] := 1;
10     ELSIF elem[2] = 0 AND elem[0] = 0 THEN  visElem[8] := 0;
11     ELSIF elem[2] = 1 THEN visElem[6] := 1; visElem[3] := 1;
12     ELSIF elem[2] = 2 THEN visElem[6] := 1; visElem[4] := 1;
13     ELSE ; END_IF
14
15     (*Limite de retorno de las piezas: *)
16       IF NOT visElem[0] AND NOT visElem[6]  THEN limh := Castop;
17       ELSIF NOT visElem[0] THEN limh := 565;
18       ELSE limh := Cbstop; END_IF
```

Código 5: Realización de la reconfigurabilidad.

La línea 1 del código 6 manipula arbitrariamente la invisibilidad de algunas piezas, volviéndolas invisibles. Este paso es muy importante, ya que, como se detallará a lo largo de esta explicación, uno de los requisitos para que las piezas activen las fotocélulas y por tanto sean detectadas por los distintos procesos es la no invisibilidad; por el hecho de hacer algunas piezas invisibles se consigue proporcionar a la sucesión de azulejos espacios,

es decir, huecos donde el compenser puede descargar piezas.

Por último, en el bloque de inicialización se calculan las velocidades de transporte de toda la simulación; tal como se muestra en el código 6. Las variables que guardan estas velocidades están declaradas en un POU distinto desde donde son accedidas tanto por los POUs de control como por el de simulación, es decir, este mismo; dichas velocidades dependen de los incrementos de píxeles por desplazamiento definidos antes y del periodo de simulación tsim.

```
1  (*Hacer huecos: *) inv[2] := 1; inv[9] := 1; inv[16] := 1;
2
3    (*Calculo de velocidades en funcion de tsim: *)
4      (*Compenser a: *) Variables.vCintaCa := Ih/tsim; Variables.vSubirCa := IvC/tsim;
5      (*Compenser b: *) Variables.vCintaCb := Ih/tsim; Variables.vSubirCb := IvC/tsim;
6      (*Proceso 1a: *) Variables.vCintaP1a := Ih/tsim;
7      (*Proceso 1b: *) Variables.vCintaP1b := Ih/tsim;
8      (*Proceso 2a: *) Variables.vCintaLP2a := Ih/tsim; Variables.vCintaRP2a := IhP2/tsim;
9      Variables.vEmboloP2a := IvP2/tsim;
10     (*Proceso 2b: *) Variables.vCintaLP2b := Ih/tsim; Variables.vCintaRP2b := IhP2/tsim;
11     Variables.vEmboloP2b := IvP2/tsim;
12
13         S_t_while := 1;
14         Inicializado := 1;
```

Código 6: Salida del bloque de inicialización

Además en esta última parte de la inicialización es donde se pone en marcha el temporizador, poniendo la variable S_t_while a 1, y se da la inicialización por finalizada tras cambiar Inicializado a 1 también.

Periodo de simulación: El código 7 muestra la implementación del temporizador y la forma de control sobre la simulación realizada gracias a este. Tras ser activado en la inicialización, el temporizador se programa antes del condicional para asignar los valores correspondientes a los tiempos de la condición. Así, solo cuando el tiempo transcurrido (TP_while.ET) sea mayor o igual al preestablecido se realizarán los cambios en el simulador. En caso contrario se ejecutará únicamente la línea del temporizador con cada ejecución del maintask.

Las últimas líneas del código 7 reinician el conteo para la próxima comprobación del

condicional. Cabe destacar que tras la desactivación del booleano S_t_while es necesario volver a implementar el temporizador para que este realice los cambios en el mismo instante. En caso contrario, desde la desactivación de la variable hasta la desactivación real se pierde el tiempo de ejecución de todas las líneas intermedias. Lo mismo ocurre con la reactivación.

```
1 (*Reactivacion del temporizador para la proxima vez *)
2   TP_while(IN := S_t_while, PT := PT_t_while);
3 IF ( TP_while.ET >= TP_while.PT ) THEN
4   (*Timmer *)
5     S_t_while := 0;
6     TP_while(IN := S_t_while, PT := PT_t_while);
7     S_t_while := 1;
8     TP_while(IN := S_t_while, PT := PT_t_while);
```

Código 7: Cálculo y control del periodo de simulación

Operario: La programación realizada para obtener las acciones definidas para el elemento Operario aparece en el código 8. En caso de que Operario se pulse, se recorre, para cada una de las b baldosas, su desplazamiento horizontal, el vector auxiliar de configuración (visElem[], y el estado de los procesos configurados como activos. Dentro de este bucle se agrupan las posibles modificaciones del Operario en bloques que dependen de la posición de las piezas.

```
1  (*Intervencion del operario: *)
2  IF Operario THEN
3    FOR i := 0 TO b BY 1 DO
4      (*Para cuando hay piezas debajo de los procesos: *)
5        (*Proca: *)
6          IF (posx[i] > Pastart AND posx[i] < Pastop-wb) THEN
7            IF visElem[1] AND Variables.ParadoP1a THEN inv[i] := 1; END_IF
8            IF visElem[2] AND Variables.ParadoP2a THEN inv[i] := 1; END_IF
9          END_IF
10       (*Procb: *)
11         IF (posx[i] > Pbstart AND posx[i] < Pbstop-wb) THEN
12           IF visElem[3] AND Variables.ParadoP1b THEN inv[i] := 1; END_IF
13           IF visElem[4] AND Variables.ParadoP2b THEN inv[i] := 1; END_IF
```

```
14        END_IF
15    (*Para la salida del compenser a: *)
16      IF posx[i] > CpF2off AND posx[i] <= Castop THEN
17        IF NOT visElem[3] AND NOT visElem[4] AND visElem[0] AND Variables.ParadoCb
18          THEN inv[i] := 1;
19        ELSIF visElem[3] AND Variables.ParadoP1b THEN inv[i] := 1;
20        ELSIF visElem[4] AND Variables.ParadoP2b THEN inv[i] := 1;
21        END_IF
22      END_IF
23    (*Solucionar colisiones: *)
24      FOR k := 0 TO b BY 1 DO
25        IF ((posx[i] < Pastart) OR (posx[i] > Pastop AND posx[i] < Pbstart) OR
26          posx[i] > Pbstop) AND ((posx[k]< Pastart) OR (posx[k]> Pastop AND posx[k]< Pbstart)
27          OR posx[k] > Pbstop) AND i <> k AND inv[k] <> 1 AND inv[i] <> 1 AND posy[i] = 0
28          AND ((ABS(posx[i] - posx[k]) < (d)) ) AND posy[k] = 0
29            THEN inv[k] := 1; END_IF
30      END_FOR
31  END_FOR
32 END_IF
```

Código 8: Modificación realizada por el botón Operario

La acción de Operario sobre los procesos genéricos (Proceso 1 y Proceso 2) está definida a partir de la línea 5. Dado que en una situación real, por cuestiones de seguridad, el operario no debe alterar el proceso, la primera comprobación que se hace para realizar modificaciones es que el proceso activo (visElem[] no nulo) de la posición correspondiente esté Parado. Cumpliéndose con estas condiciones se opta por quitar la pieza de la cinta, haciéndola invisible. Se considera esta última una modificación adecuada para cualquier posición dentro de los procesos ya que generalmente son procesos temporizados o de velocidad variable, que tras una parada han de empezar desde cero.

Una modificación similar se realiza para el caso del Compenser a, a partir de la línea 15, dado que como se ha dejado indicado en apartados anteriores en caso de parada de algún proceso aguas abajo, los motores de las cintas correspondientes a dichos procesos se paran y el compenser empieza a cargar inmediatamente. Puede darse el caso de que con la cinta siguiente parada la pieza que es transportada por dentro del compenser rebase la posición de carga, cosa que inevitablemente lleva a una colisión con el azulejo siguiente parado en

la línea. La solución considerada apropiada para este caso es cambiar la visibilidad de la pieza que está parada en la cinta nada más salir del compenser. Dicha modificación es acertada en la mayoría de las ocasiones, menos en las que coincide que la pieza anterior a la parada en la posición problemática sí se consigue cargar, ya que ahí no se debería realizar acción alguna por parte del operario.

Cabe destacar que la operación necesaria, como se ha visto, para el Compenser a no lo es para el Compenser b, ya que la carga de este, en caso de haberse configurado su simulación, no afecta a procesos aguas abajo, dado que estos no existen a nivel de simulación.

Desplazamientos: El siguiente bloque de código que merece atención es el de los desplazamientos de azulejos, código que dada su extensión sólo se deja para el código 23 del Anexo 1, comprendido entre las líneas 91 y 183. Este bloque se divide inicialmente en dos partes: desplazamientos verticales y desplazamientos horizontales.

En cuanto a los desplazamientos verticales, estos se realizan sólo en los dos compensers (en caso de que los dos estén activos). Asimismo lo que se hace es recorrer para cada compenser el vector de desplazamientos horizontales (posx[i]), su vector de desplazamientos verticales (posy[i]), comprobando la cota y el vector de invisibilidad (inv[i]) para cada azulejo empleando un iterador i. Si se detecta la orden de MCSubir o MCBajar para alguno de los compensers, y cumpliéndose las comprobaciones anteriores, la pieza empieza a ascender o descender.

Los desplazamientos horizontales se programan de un modo análogo a los verticales, aunque dada la amplia reconfigurabilidad y la multitud de cintas existentes, el algoritmo de control está separado en varios bloques en función de la posición de los azulejos. A grandes rasgos las particiones se realizan para cada cinta configurada, desde que la pieza alcanza a entrar en dicha cinta hasta la posición de entrada de la siguiente. Entiéndase la posición de entrada en la cinta como el instante de simulación en el que, dado el movimiento de izquierda a derecha de las baldosas, el extremo derecho de la pieza rebasa el extremo izquierdo de la cinta.

Por consiguiente la pieza sufrirá un desplazamiento sobre la cinta correspondiente a su posición si el motor, o para este caso la señal que controla el motor de dicha cinta, generalmente $\mathsf{MCinta_X}$ está activo o no. Un caso especial es el Proceso 2, ya sea A o B, cuando hay que tener en cuenta la velocidad de giro del motor correspondiente a la cinta actual, no sólo el hecho de que la señal esté activa.

Una parte importante a aclarar del bloque de desplazamientos son las operaciones reali-

zadas para la recolocación de los azulejos tras alcanzar el valor definido como limh. Para mejor comprensión el algoritmo aparece en el código 9.

```
1 (*Caso de alcanzar el limite: *)
2   IF Sim.posx[i] >= limh THEN
3     IF posx[pos] < -4*d THEN  posx[i] := posx[pos]-d;
4     ELSE posx[i] := -5*d; END_IF
5     IF i <> 2 AND i <> 9 AND i <> 16 THEN inv[i] := 0; END_IF
6     pos := i;
7   END_IF
```

Código 9: Recolocación del azulejo una vez alcanzado limh.

La operación sólo se realiza si el desplazamiento horizontal de los azulejos alcanza el valor de limh; así, se comprueba la posición de la última pieza de la sucesión entera, para recolocar la pieza que ha llegado al límite a la distancia correspondiente d de la última, asegurando así la continuidad en la línea. Pero se puede llegar a la situación, por carga de compenser por ejemplo, de que la última pieza esté visible sobre alguna de las cintas; una recolocación directa conllevaría la aparición repentina de baldosas sobre las cintas, situación de simulación poco realista por tanto. Para evitarlo lo que se hace es comprobar que la última pieza de la sucesión está en la parte no visible de la visualización; dada la posición absoluta de los azulejos en el instante inicial mostrada en la tabla 1, se calcula esta posición. En caso de que no se cumpla el requisito de posición, lo que se hace es recolocar arbitrariamente la pieza actual en la parte no visible, evitando apariciones no deseadas, y marcando una referencia para la siguiente pieza que alcance el límite horizontal.

En el mismo código 9 se restablece la visibilidad de las piezas afectadas por la acción realizada por Operario; todas las piezas menos las definidas invisibles deliberadamente para la generación de huecos en la cinta.

Tratamientos: A continuación, hacia la parte final del código en sí del POU Sim (PRG), en el código 10 aparecen las variables que controlan los tratamientos que se les aplican a las baldosas en cada uno de los procesos. Esta parte está dividida en dos, separando una vez por proceso en sí (considerándose Proceso 1 o Proceso 2), y a su vez se vuelve a separar para definir si se trata de la parte A o B.

```
1 (*Tratamiento realizado en el Proceso 1: *)
2   (*Proc1: *)
3     IF Variables.TratamientoP1a THEN Trat1 := 1; ELSE Trat1 := 0; END_IF
```

```
4   (*Proc1b: *)
5     IF Variables.TratamientoP1b THEN Trat1b := 1; ELSE Trat1b := 0; END_IF
6 (*Tratamiento realizado en el Proceso 2: *)
7   (*a *)
8     IF Variables.DescP2a THEN posVa := posVa - IvP2;
9     ELSIF Variables.AscP2a THEN posVa := posVa + IvP2; END_IF
10  (*b *)
11    IF Variables.DescP2b THEN posVb := posVb - IvP2;
12    ELSIF Variables.AscP2b THEN posVb := posVb + IvP2; END_IF
```

Código 10: Operaciones realizadas para los tratamientos.

Para el tratamiento del Proceso 1, dado que, tal como se ha explicado en el apartado 7.3.3, solo se trata de la activación de la variable que controla la visibilidad del elemento correspondiente (Trat1 o Trat1b), solo se realiza la comprobación de la señal del controlador automático del proceso, señal ya descrita, Tratamiento, para activar directamente la variable de visibilidad correspondiente.

Para el caso del Proceso 2, la acción realizada es la de subir o bajar el émbolo neumático. Así, de forma parecida al caso anterior, si la señal de ascender o descender émbolo proveniente del controlador (Asc o Desc) es detectada, se realiza el desplazamiento del elemento que representa al émbolo en la visualización. Este desplazamiento está asociado a la variable posV, que se va incrementando o decrementando un valor IvP2 en cada ciclo de simulación.

Visualización de cintas: El bloque siguiente engloba el control de la apariencia de las cintas transportadoras mostradas en la tabla 5. Si el controlador automático de los procesos decide poner en funcionamiento el motor correspondiente a una cinta en concreto, dicha cinta pasa a colorearse simulando así, de forma simbólica, su movimiento. El bloque entero se muestra en el código 11.

```
1 (*Activacion de la visualizacion de actuadores: *)
2   (*Fuera de procesos: *)
3     CintaA := NOT (visElem[1] AND visElem[2]) AND Variables.MCintaAntCa;
4     CintaA1 := (visElem[1] AND Variables.MCintaAntP1a)
5       OR (visElem[2] AND Variables.MCintaAntP2a);
6     CintaA2  := Variables.MCintaAntCa;
7     CintaB := NOT (visElem[3] AND visElem[4]) AND Variables.MCintaAntCb;
```

```
8     CintaB1 := (visElem[3] AND Variables.MCintaAntP1b)
9       OR (visElem[4] AND Variables.MCintaAntP2b);
10    CintaB2  := Variables.MCintaAntCb OR (NOT visElem[0] AND visElem[6]);
11  (*Parte a: *)
12    (*Compenser: *) CintaCa := Variables.MCintaCa; SubirCa := Variables.MCSubirCa;
13                    BajarCa := Variables.MCBajarCa;
14    (*Proceso 1: *) CintaP1a := visElem[1] AND Variables.MCintaP1a;
15    (*Proceso 2:*) CintaP2a1:=visElem[2] AND (Variables.MCinta1LP2a OR Variables.MCinta1RP2a);
16                  CintaP2a2:=visElem[2] AND (Variables.MCinta2LP2a OR Variables.MCinta2RP2a);
17  (*Parte b: *)
18    (*Compenser: *) CintaCb := visElem[0] AND Variables.MCintaCb;
19                    SubirCb := Variables.MCSubirCb; BajarCb := Variables.MCBajarCb;
20    (*Proceso 1: *) CintaP1b := visElem[3] AND Variables.MCintaP1b;
21    (*Proceso 2: *) CintaP2b1 := visElem[4] AND NOT Variables.ParadoP2b;
22            CintaP2b2 := visElem[4] AND (Variables.MCinta2LP2b OR Variables.MCinta2RP2b);
```

Código 11: Cambios de la visualización de las cintas.

El efecto anterior se consigue cambiando el valor de las variables booleanas asociadas a cada cinta, tras haber relacionado el cambio de apariencia a dichas variables. El estado de cada cinta, aunque en una instalación real sólo va ligado al estado del motor que proporciona el movimiento de tracción, para este simulador depende además de la reconfigurabilidad ya descrita.

El algoritmo de este bloque empieza determinando el estado de las cintas externas a los procesos, divididas implícitamente entre la parte A y B, operando lógicamente entre variables de visibilidad de los procesos en sí (relacionadas con la reconfigurabilidad) y la señal correspondiente al motor de cada cinta.

De forma análoga, en la segunda parte del bloque, se calcula el estado de cada cinta interna de cada proceso, teniendo en cuenta otra vez variables de visibilidad y señales de motores, para los seis posibles procesos.

Activación de fotocélulas: El último bloque del código del simulador corresponde al cálculo del estado de las fotocélulas de cada proceso. Dada la extensión en cuanto a líneas de código pero la sencillez y repetitividad en cuanto a programación, este bloque se puede analizar en el código 23 del Anexo 1, comprendido entre la línea 225 y el final.

El propósito de este algoritmo es enviar al controlador de los procesos la señal correspon-

diente a cada fotocélula, marcando así el estado de estas. Por tanto, cada fotocélula del simulador, perteneciente a cualquier proceso, se activa o no a lo largo de este bloque.

El cálculo del estado consiste en recorrer, para cada fotocélula existente, el número total de azulejos y comprobar su posición, su visibilidad, además de la visibilidad del proceso al que pertenece la fotocélula correspondiente. Así, se realiza para cada caso un bucle FOR en cuyo interior se comprueban condiciones de posición y visibilidad. Para la mayoría de los casos se comprueba la visibilidad de la pieza definida con el vector inv[i], así como su posición horizontal definida con el vector posx[i], y el posicionamiento vertical del azulejo; pero no antes de comprobar que el proceso al que pertenece dicha fotocélula está visible en la simulación actual: valor no nulo para visElem[proc].

Un caso excepcional son los sensores inductivos relacionados con el émbolo neumático del Proceso2, ya que estos no dependen de las posiciones del azulejo, sino de la posición del vástago marcada por posVa y posVb. Dada la pertenencia al Proceso2 el estado de estos sensores también depende de la visibilidad de dicho proceso.

Es importante hacer referencia a la necesidad del comando EXIT tras la activación de una fotocélula, ya que basta con que solo una baldosa esté en la posición adecuada para cambiar el estado a true; y salir del bucle de comprobación. En caso de omitirse dicho comando, el estado de la fotocélula se activaría en condiciones favorables durante una iteración del bucle, y se desactivaría en la siguiente iteración (considerando un posicionamiento adecuado de los azulejos en la línea).

7.4. Desarrollo del control automático

7.4.1. Descripción general

El control automático es un complemento esencial en la realización de este proyecto dada la importancia de comprobar el funcionamiento del conjunto completo, además simular el funcionamiento normal de la línea como demostración de una resolución posible.

Para mejor organización durante el desarrollo del algoritmo de control, se opta por programar el control automático de cada proceso descrito en el apartado 7.3.3 en unidades de organización del programa diferentes, es decir POUs diferentes. Cabe destacar que como consecuencia de lo anterior se hace referencia en más de una ocasión a variables locales definidas en otros POUs.

A lo largo de este punto se explicará, de forma análoga al punto anterior, cada POU dividiéndose en la parte de declaraciones y código propiamente del control. Además se explicarán detalladamente los gráfcets empleados para el control de cada proceso.

7.4.2. Control automático del compenser

Para una comprensión correcta del control descrito a lo largo de este punto es necesario hacerse una idea de cuál es el comportamiento esperado del compenser, comportamiento descrito en el apartado 7.3.3.2. Asimismo se hace imprescindible contemplar las variables de entrada y salida consideradas para un funcionamiento apropiado, mostradas en las tablas 7 y 8.

Cabe destacar que en este apartado se describe el control del compenser de la parte A, es decir, se exponen partes del código programadas en el POU `Cmp1a`. Se opta por esta estrategia ya que los algoritmos son casi idénticos además de que los añadidos en el control realizado en la parte B son de fácil deducción.

Variables	Descripción
Fot1	Fotocélula 1
Fot2	Fotocélula 2
SInd	Sensor inductivo
Marcha	Pulsador para la marcha
Paro	Interruptor con enclavamiento para el paro
CargaM	Interruptor para empezar la carga del compenser
vCinta	Velocidad de transporte en la cinta
vSubir	Velocidad de carga y descarga de los azulejos

Tabla 7: Entradas del compenser.

Variables	Descripción
MCinta	Motor que mueve la cinta
MCintaAnt	Motor que mueve la cinta anterior en la línea
Parado	Señal que indica que el compenser está parado
Lleno	Señal que indica que el compenser está lleno
Vacío	Señal que indica que el compenser está vacío
MCSubir	Motor que hace ascender las piezas
MCBajar	Motor que hace descender las piezas

Tabla 8: Salidas del compenser.

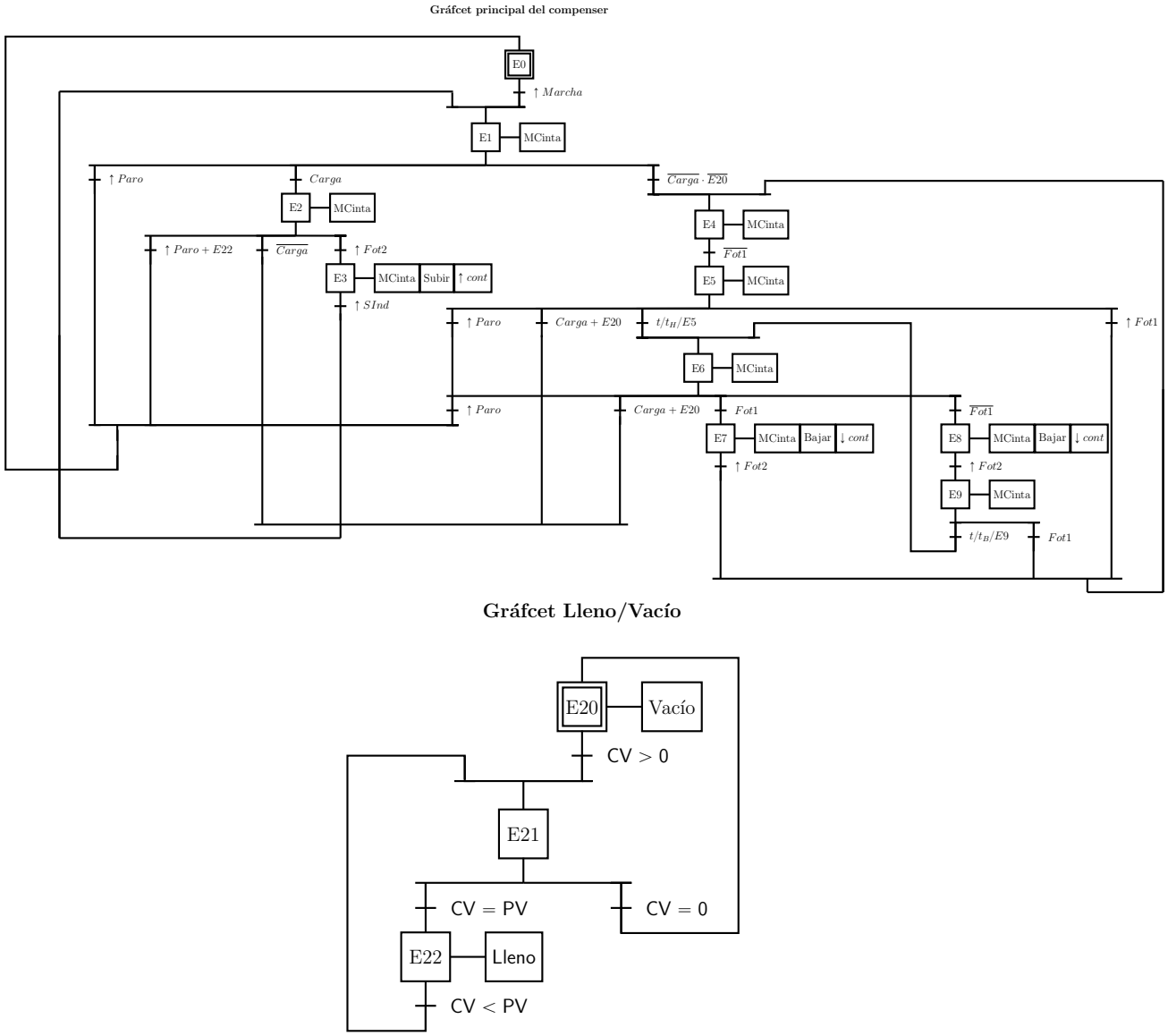

Figura 8: Gráfcets del control del compenser.

El control propuesto para el proceso compenser se organiza en dos gráfcets distintos, mostrados en la figura 8.

G0: Gráfcet principal del proceso El primero, el más complejo de ellos, es el gráfcet G0 encargado de controlar los actuadores del proceso en función de algunas entradas y de otras variables complejas determinadas por el otro diagrama. Este gráfcet comienza con su etapa inicial, E0, en la que sólo la salida Parado está activada. Cuando se detecte un flanco en el pulsador Marcha es cuando la cinta transportadora que atraviesa el compenser empieza a moverse. A continuación se puede pasar a tres estados distintos: volverse a la etapa inicial tras detectar un flanco de Paro, cambiar al estado de carga, etapa E2, y cargar las piezas que lleguen por la cinta, o en tercer lugar cambiar al estado de descarga, etapa E4, y descargar siguiendo el orden del proceso marcado por las etapas posteriores. Una cuarta opción es la de quedarse en la etapa E1 debido a que no se está en modo carga pero tampoco se tienen piezas para descargar; de este modo el compenser se limita a transportar los azulejos aguas abajo para los tratamientos de los procesos posteriores.

Para pasar a modo carga es necesario que la variable Carga descrita posteriormente esté activa, cumpliéndose así la transición anterior a la etapa E2. Durante la etapa E2 la cinta encargada del transporte de las piezas sigue en marcha dejando evolucionar el gráfcet hacia tres distintas opciones: tras detectar un pulso en Paro o Lleno volviendo a la etapa inicial igual que en el caso anterior; pudiendo volver a la etapa E1 en caso de cambiar el estado de la variable Carga; y por último, tras detectar un flanco de subida en la fotocélula 2, $\uparrow$ Fot2, pasar a la etapa E3 que además de hacer desplazar la cinta transportadora, hace elevar la estructura del compenser cargando así la pieza cerámica, mediante la salida MCSubir. Con la entrada en la etapa E3 además se incrementa el número de piezas existentes en el compenser, dejando constancia de este cambio en el gráfcet con la acción $\uparrow$ cont + +.

La tercera rama del gráfcet G0 es la encargada de realizar la descarga de las piezas almacenadas de manera que se asegure la distancia mínima entre azulejos, distancia imprescindible para los tratamientos posteriores en la línea. Asimismo, para empezar el proceso de descarga se necesita que la variable Carga no esté activada y, además, la variable interna Vacio no se encuentre a true, asegurando así que hay piezas almacenadas a descargar. Superada esta transición se pasa a la etapa E4, que en caso de no estar activa la señal de la fotocélula 1 ($\overline{\text{Fot1}}$), evoluciona G0 hacia E5. Al entrar en esta etapa se inicia el temporizador encargado de calcular si existe el espacio necesario en la cinta para evitar colisiones y asegurar el funcionamiento adecuado aguas abajo en la línea, temporizador llamado aquí t_H y explicado en el punto 7.4.2.2. Estando en la etapa E5 hay cuatro posibles opciones

posibles: la primera de ellas es un flanco en Paro, ya descrito anteriormente; la segunda es un cambio de Carga, que hace cambiar G0 a E1; una tercera opción es un flanco en Fot1 que reiniciaría la etapa E5 y por tanto el temporizador asociado a esta; y por último el final del conteo del temporizador, haciendo activar la etapa E6. De forma análoga a la quinta etapa, la etapa E6 contempla cuatro posibles cambios: las primeras dos opciones son idénticas al caso anterior (flanco en Paro o cambio en Carga); la siguientes opciones dependen del estado de la fotocélula 1: si esta está activada se pasa a la etapa E7 y en caso contrario a la E8. Si se pasa a la etapa E7 lo que se hace es realizar la descarga de una única pieza, mediante MCBajar, para retornar a la etapa E4; se realiza de este modo ya que el hecho de que Fot1 estaba activa al iniciar la descarga indica que hay una pieza entrando en el compenser que obliga a retomar los cálculos del espacio disponible tras finalizar la bajada. Por el contrario, si se evoluciona a E8 se vuelve a realizar una única descarga pero pasando esta vez a E9; la novena etapa inicia un segundo temporizador, llamado t_B y explicado también en el punto 7.4.2.2, que en caso de alcanzar su valor preestablecido evoluciona G0 hasta E6 que instantáneamente hace bajar otra pieza, ya sea por E7 o E8, siempre y cuando los valores de Carga y Paro lo permitan. De forma análoga a la carga, cuando se realiza la descarga de alguna pieza en las etapas E7 o E8 se decrementa el contador de piezas almacenadas mediante $\uparrow \mathsf{cont}--$. Por último, aunque a lo largo de todo el párrafo no se menciona, la cinta transportadora se mueve continuamente ya que la señal MCinta se mantiene activada en todas las etapas.

G1: Gráfcet de Lleno/Vacío El siguiente gráfcet a explicar es el G1 al que ya se ha hecho mención anteriormente. Este gráfcet se encarga de determinar el estado del compenser en cuanto a almacenamiento se refiere, conmutando el valor de las variables booleanas internas Lleno y Vacio según el número de piezas cargadas almacenadas. Este diagrama va ligado al contador CTUD empleado durante el control, contador que varía según las veces que se llega a las etapas E3, E7 y E8 del gráfcet G0.

El gráfcet G1 parte desde su estado inicial poniendo a true la variable Vacio, situación coherente con una línea real que se pone en marcha por primera vez. Una vez inicializado el proceso se comprueba el valor CV (Current Value) del contador, pudiendo cambiar el estado de Vacio a false cuando se incremente el valor actual. Tras varias cargas seguidas se puede alcanzar el valor PV (Preset Value), en cuyo caso se pondría la variable Lleno a true; indicando asimismo que se han cargado el máximo número de piezas que el compenser puede almacenar.

La variable Carga depende de varios factores, como consentimientos de procesos aguas

abajo en la línea, y por tanto visibilidad de dichos elementos según la reconfigurabilidad en ejecución, así como el interruptor de carga manual descrito en el apartado de simulador. Para mejor comprensión se extrae el algoritmo que determina el valor de dicha variable, mostrándolo en el código 12.

```
1 (*Calculo de transiciones complejas: *)
2         IF Sim.visElem[3] THEN Carga := NOT Varialbes.MCintaP1b OR CargaM;
3         ELSIF Sim.visElem[4] THEN Carga := Varialbes.ParadoP2b OR CargaM;
4         ELSIF Sim.visElem[0] THEN Carga := NOT Varialbes.MCintaCb OR CargaM;
5         ELSE Carga := CargaM; END_IF
```

Código 12: Cálculo del valor del booleano Carga.

Por último es interesante resaltar8 para este último gráfcet que para el compenser de la parte B, el compenser cuyo control se describe en el POU Cmp1b, la variable Carga sólo depende del valor introducido por el usuario mediante el interruptor CargaM durante la ejecución de la simulación. Por tanto, aunque el proceso sea idéntico en cuanto a interacción y secuencia de los elementos los algoritmos de control se diferencian por el cálculo de Carga.

7.4.2.1 Declaraciones del control automático del compenser

Aunque las declaraciones del control del compenser ya aparecen en el código 25 del Anexo 2, a lo largo de este punto se hará hincapié en algunos elementos empleados durante la realización del código de control, mostrando así trozos de código aislados.

En primer lugar y sin necesidad de profundizar más aparecen las variables booleanos de cada etapa descrita con los gráfcets anteriormente, así como las variables ya descritas. Es importante destacar que las variables de las etapas se inicializan a 0, menos las iniciales que son E0 y E20 que se inician a 1.

Como se ha visto, en varios puntos de los gráfcets se emplean variables tipo flanco, ya sea de bajada o subida. Estas variables tienen asignado un tipo aparte, en función del tipo de flanco que deben detectar. Así, las variables relacionadas con los flancos de subida se definen del tipo R_TRIG, mientras que las relacionadas con los flancos de bajada se definen como F_TRIG, tal como se muestra en el código 13.

```
1 (*Flancos:*)
2         FFot1: F_TRIG;
3         RFot2: R_TRIG;
```

```
4        RSInd: R_TRIG;
5        RMarcha: R_TRIG;
6        RParo: R_TRIG;
7        RMarchaM: R_TRIG;
8        RParoM: R_TRIG
```

Código 13: Declaración de flancos en el CodeSys.

Otro elemento encontrado a lo largo del código de declaraciones es el temporizador; tal como se ha visto, en este caso hay dos temporizadores. Para el control de este proceso se opta por emplear temporizadores tipo TON, que son los temporizadores con retardo a la conexión. La declaración de temporizadores en CodeSys necesita como mínimo el nombre del temporizador y su tipo, así como una variable tipo TIME para el valor de tiempo preestablecido PT. Por último, aunque de forma opcional se define además un booleano asociado al temporizador que marcará durante el cómputo si se ha alcanzado el tiempo preestablecido o no conmutando su valor. Como ejemplo el temporizador t_H aparece en el código 14.

```
1 (*Timmer Hueco: *)
2        TON_Hueco: TON;
3        PT_t_Hueco: TIME;
4        t_Hueco: BOOL;
```

Código 14: Declaración del temporizador t_H.

Y en último lugar aparece el contador empleado para el conteo de piezas almacenadas en el compenser. En este caso se hace necesario el uso de un contador ascendiente/descendiente, en CodeSys tipo CTUD, ya que se necesita poder incrementar o decrementar según si se carga o descarga un azulejo. Análogamente al caso anterior, se necesita declarar el nombre escogido para el contador así como su tipo, junto con una variable preestablecida tipo WORD para marcar el máximo a alcanzar, variable llamada a lo largo del gráfcet G2 PV. De nuevo, el código 15 es un extracto con la declaración del contador para el compenser de la parte A.

```
1 (*Counter *)
2        CTUDPiezas: CTUD;
3        PV_c_Piezas: WORD := 15;
```

Código 15: Declaración del contador CTUD del Cmp1a.

7.4.2.2 Código del control automático del compenser

El código que controla el compenser se puede dividir en varios bloques que aparecen en el código 26 del Anexo 2. A lo largo de este subapartado se extraen trozos de código cuando se considera necesario realizar una explicación más detallada.

Recorriendo el algoritmo el primer bloque escrito es el de inicialización. Tal como su nombre indica, este bloque se encarga de inicializar variables en función de la reconfigurabilidad establecida. Para los casos de control suele tratarse de variables de tipo tiempo, ya que estas varían en función del tiempo de simulación que es un parámetro configurable tal como se explica en el apartado 7.3.4. Para el compenser, dado que hay dos temporizadores independientes, hay dos variables de tiempo que se calculan en este bloque (código 16): el tiempo PT_t_Hueco y el PT_t_Bajada, los dos tiempos preestablecidos.

```
1 (*Inicializacion: *)
2   IF NOT Inicializado THEN
3     Inicializado := 1;
4     PT_t_Bajada := REAL_TO_TIME((Sim.d-Sim.dbaj+1)/Variables.vSubirCa);
5     PT_t_Hueco := REAL_TO_TIME((Sim.CpF2on-Sim.CpF1off+Sim.we-Sim.dbaj+1)
6                   /Variables.vSubirCa);
7   END_IF
```

Código 16: Inicialización de Cmp1a.

El valor de PT_t_Hueco es el tiempo que tarda un azulejo en recorrer la distancia entre la posición de desactivación de la fotocélula 1 y la posición de activación de la fotocélula 2, además de añadirle el tiempo necesario para recorrer el espaciado por defecto entre piezas quitándole la distancia de bajada predefinida en el simulador. De este modo el controlador puede asegurar que al descargar una pieza esta no colisiona con la última que acaba de pasar, ni con la próxima en llegar, asegurando además que el espaciado al rededor de la pieza descargada es el preestablecido. El tiempo correspondiente se calcula dividiendo la distancia anterior entre la velocidad de bajada de los azulejos, vSubir, obteniendo así el valor de tiempo para el temporizador t_H. Para este cálculo se usa la velocidad de bajada aunque se podría usar la de la cinta ya que sus valores son idénticos.

De forma parecida, el tiempo PT_t_Bajada es el tiempo que ta como booleanasrda la pieza recién bajada en recorrer su misma anchura más el espaciado por defecto, restando la distancia de bajada. De este modo se asegura que en la próxima bajada, que se inicia casi instantáneamente, la pieza no colisiona y tampoco invade el espacio reservado entre

baldosas. De forma análoga al caso anterior dicho tiempo se calcula dividiendo la distancia resultante entre la velocidad de descenso obteniéndose el tiempo preestablecido para el temporizador t_B.

Las siguientes líneas de código ya se han explicado junto con el gráfcet G2 y mostrado en el código 12 para el cálculo del valor de Carga.

El siguiente bloque programado se hace necesario para la correcta detección de los flancos. El cálculo de los flancos como ya se ha mencionado anteriormente se realiza usando las variables declaradas R_TRIG y F_TRIG, relacionando estas con la variable que se monitoriza para la detección del flanco. Algunos ejemplos aparecen en el código 17.

```
1 (*Deteccion de flancos*)
2   RMarcha(CLK := Marcha);
3   RMarchaM(CLK := Variables.MarchaMCa);
4   RParo(CLK := Paro);
```

Código 17: Programación de las variables flanco.

El siguiente bloque de código es la traducción de los gráfcets explicados anteriormente a texto estructurado (ST). Se trata de varios condicionales que comprueban los estados de las etapas y las transiciones siguientes garantizando la evolución secuencial esperada. Dada la extensión del bloque y la sencillez en cuanto al nivel de programación no se considera necesario profundizar más en esta parte.

La siguiente línea de código es la que hace referencia al contador CTUD declarado anteriormente. El contador, llamado en este programa CTUDPiezas, tiene cinco entradas y tres salidas. Para el funcionamiento correcto del contador es necesario asociar cuatro variables distintas como entrada: a la primera, CU (Count Up), tiene asociada la variable que hace incrementar el contador, que en este caso es la etapa E3; la siguiente entrada es CD (Count Down) a la que se asocian las variables que hacen descender el contador que en este caso son las epatas E7 y E8. Las dos entradas siguientes se ponen a false ya que en ningún momento es necesario poner forzadamente un valor determinado en el contador. Por último se asocia el máximo número de piezas a la entrada PV del contador. El uso de CTUDPiezas se muestra en el código 18.

```
1 (*Contadores *)
2    CTUDPiezas(CU := E4, CD := E8 OR E9, RESET := FALSE, LOAD :=  FALSE, PV := PV_c_Piezas);
```

Código 18: Programación del contador.

En el siguiente tramo de código se asocian las variables que representan las salidas, variables mostradas en la tabla 8, a las distintas etapas siguiendo las relaciones establecidas en el gráfcet G0.

Por último con el extracto mostrado en el código 19 se programan los dos temporizadores empleados para el control de este proceso: t_H y t_B. La programación de los temporizadores es semejante a la del contador, aunque aquí como entradas se tiene IN, que es el pin donde hay que conectar la variable que marca el inicio del temporizador, y en segundo lugar PT donde hay que introducir el tiempo preestablecido calculado anteriormente. Para estos dos casos las variables asociadas a IN son las etapas temporizadas E5 y E9 respectivamente.

```
1 (*Temporizadores *)
2         TON_Hueco(IN := E6, PT := PT_t_Hueco);
3         TON_Bajada(IN := E10, PT := PT_t_Bajada);
4         (*Booleano dependiente *)
5                 t_Hueco := TON_Hueco.Q;
6                 t_Bajada := TON_Bajada.Q;
```

Código 19: Programación de los temporizadores.

Las variables booleanas t_Hueco y t_Bajada son variables que se usan como transiciones de las etapas dependientes de temporizador. Como se puede observar se puede usar directamente la salida Q de los temporizadores, pero para mayor claridad se opta por emplear las variables mencionadas. La salida Q de los temporizadores TON cambia a true cuando se haya cumplido el tiempo preestablecido.

7.4.3. Control automático del Proceso 1

El control del Proceso 1 es considerablemente más sencillo que el del compenser. El funcionamiento de este proceso ya se ha descrito con detalle en el apartado 7.3.3.3. Las entradas y salidas del controlador de este proceso se muestran en las tablas 9 y 10 respectivamente.

Variables	Descripción
Fot1	Fotocélula 1
Marcha	Pulsador para la marcha
Paro	Interruptor con enclavamiento para el paro
vCinta	Velocidad de transporte en la cinta

Tabla 9: Entradas del Proceso 1.

Variables	Descripción
MCinta	Motor que mueve la cinta
MCintaAnt	Motor que mueve la cinta anterior en la línea
Tratamiento	Señal que indica la aplicación de tratamiento sobre la pieza
Parado	Señal que indica que el compenser está parado

Tabla 10: Salidas del Proceso 1.

Igual que en el caso anterior, a lo largo de este punto se exponen sólo partes del código de programación del POU Proc1a, correspondiente al Proceso 1 de la parte A de la configuración, las diferencias con su homólogo considerándose mínimas.

El gráfcet diseñado para realizar el control se muestra en la figura 9; el gráfcet G0 se encarga de gestionar todas las variables del proceso.

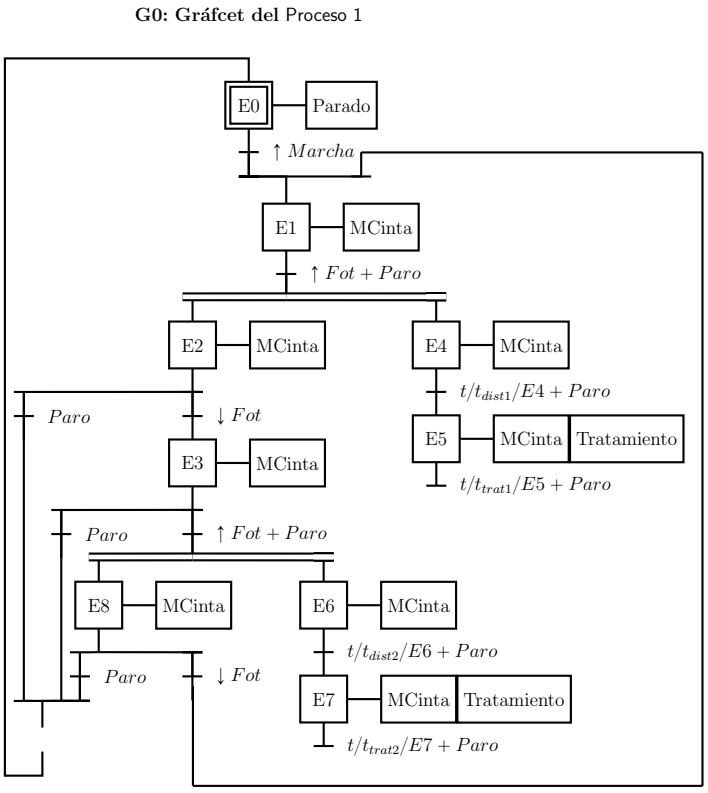

Figura 9: Gráfcet diseñados para el control del Proceso 1.

G0: Gráfcet del Proceso 1 La etapa inicial de G0, E0, es la etapa correspondiente al estado de parada, tal como indica la acción asociada. Al detectarse un pulso en Marcha se pasa a la etapa E1 que simplemente pone en marcha la cinta transportadora activando la señal MCinta, haciendo llegar piezas cerámicas al proceso. Cuando un azulejo es detectado

por la fotocélula se pasan a dos etapas distintas: E4 que lleva asociado el temporizador t_{dist1} para dar paso posterior a E5 que se encarga de realizar el tratamiento mediante la señal Tratamiento; y en segundo lugar se avanza en paralelo hacia E2, pudiendo salir de esta cuando se detecte un flanco de bajada en la fotocélula Fot. Estando en la etapa E3 se inicia una secuencia idéntica a la anterior con un flanco en la fotocélula; de este modo se garantiza que mientras la pieza ya entrada recibe el tratamiento correspondiente, un nuevo azulejo entrante por la cinta es detectado adecuadamente.

En caso de accionar el interruptor Paro varias de las etapas descritas anteriormente se vuelven inestables, asegurando de este modo un paro instantáneo y una vuelta a la etapa inicial E0.

7.4.3.1 Declaraciones del control automático del Proceso 1

Las líneas de declaraciones del Proceso 1 aparecen en el código 27 del Anexo 2. La estructura de estas es similar al caso del compenser, pudiéndose encontrar a lo largo de dicho código variables booleanas asociadas a las etapas del gráfcet explicado; además dada la necesidad de detección de flancos a lo largo del control también es imprescindible declarar variables tipo R_TRIG y F_TRIG para las correspondientes entradas.

Cabe destacar en este caso las declaraciones de los cuatro temporizadores, dos de las cuales aparecen en el código 20. Los cuatro temporizadores empleados para el control automático del Proceso 1 son del tipo TON ya que, tal como se muestra en el gráfcet, van asociados a etapas que por tanto necesitan retardo a la conexión.

```
1 (*Timmers: *)
2     (*Timmer a1: *)
3       TON_dist1: TON;
4       PT_t_dist1: TIME; (*Tiempo hasta que se active el Tratamiento*)
5       t_dist1: BOOL;
6     (*Timmer b1: *)
7       TON_trat1: TON;
8       PT_t_trat1: TIME;
9       t_trat1: BOOL;
```

Código 20: Declaraciones de los temporizadores del Proceso 1.

7.4.3.2 Código del control automático del Proceso 1

El código como tal del control del Proceso 1 aparece por completo en el código 28 del Anexo 2, sin embargo a lo largo de este subapartado se explicará con más detalle algunos de los bloques integrantes.

En primer lugar, de forma similar a los casos anteriores aparecen las líneas de inicialización, código 21. Uno de los cálculos programados en este bloque es el de los tiempos preestablecidos de los temporizadores implicados en el control del proceso. El primer tiempo obtenido es tiempo de tratamiento, PT_t_trat1, es decir, el tiempo que ha de mantenerse abierta la válvula del chorro a aplicar sobre la pieza; como es de esperar este tiempo es proporcional a la anchura del azulejo, y se obtiene dividiendo dicho valor, wb, entre la velocidad de la cinta transportadora. El segundo tiempo obtenido es el tiempo de espera desde la detección de la pieza por la fotocélula hasta el comienzo de la aplicación del tratamiento; PT_t_dist1 se calcula con la misma expresión que en el caso anterior, dividiendo la distancia correspondiente entre la velocidad de la misma cinta.

```
1 (*Inicializacion: *)
2   IF NOT Inicializado THEN
3     Inicializado := 1; PT_t_trat1 := REAL_TO_TIME(Sim.wb/Variables.vCintaP1a);
4     PT_t_dist1 := REAL_TO_TIME((Sim.P1pCh-Sim.P1pFon)/Variables.vCintaP1a);
5   END_IF
6   Marcha := Sim.visElem[1];
7   IF Sim.visElem[1] THEN Variables.MarchaMP1a := Variables.MCintaCa
8     OR Variables.MarchaMP1a; END_IF
```

Código 21: Inicialización del código del Proceso 1.

Las últimas dos líneas del código anterior, aunque no estén incluidas dentro del condicional de inicialización también se consideran parte de esta. En primer lugar se tiene la variable booleana del proceso Marcha que se relaciona con los parámetros de reconfigurabilidad; así, este proceso se pone en marcha automáticamente al ejecutar la simulación sólo si se ha configurado su existencia dentro de dicha simulación. Esta línea de código no aparece en la inicialización del compenser, código 16, porque para la explicación de este proceso se toma el código del compenser Cmp1a, que es el que aparece por defecto sea cual sea la configuración escogida.

Por último en cuanto a la inicialización, la entrada MarchaM que es la relacionada con el pulsador de marcha del panel del control del proceso, está configurada para que además

de ser controlada por dicho pulsador también se inicie con el pulsador de marcha del proceso siguiente, en este caso el Compenser 1a. Esto se hace para que, tras una parada del proceso siguiente y por consiguiente del actual también, baste con accionar el pulsador aguas abajo para activar todas las máquinas paradas anteriores.

Hay otra línea tras la inicialización, que tal como se describe en el comentario de código, define el consentimiento del proceso siguiente en la línea. A lo largo de estos algoritmos se considera que se tiene consentimiento cuando el motor de la cinta posterior está en marcha: es decir la variable MCintaAnt del proceso siguiente está a true. En este caso, como se está describiendo el control del Proceso 1a, el proceso venidero es el Compenser 1a.

En los demás bloques de este código se sigue la misma dinámica que en los casos anteriores: siguiendo con la programación de los flancos empleados, la implementación de texto estructurado para programar los gráfcets descritos, la activación de las salidas relacionándolas con las etapas de evolución, y por último la programación de los temporizadores. No se considera necesario volver a explicar cada bloque dada la semejanza con el caso descrito para el compenser.

7.4.4. Control automático del Proceso 2

De forma análoga a los casos anteriores el control del proceso restante, Proceso 2, se explica a lo largo de este apartado. La explicación detallada de su funcionamiento queda descrita en el apartado 7.3.3.4; las entradas y salidas necesarias para el control aparecen en las tablas 11 y 12 respectivamente.

Cabe destacar de nuevo que a lo largo de este punto se desglosará el código correspondiente al control del Proceso 2 situado en la parte A de la simulación, por tanto código que se puede encontrar en el POU Proc2a.

Variables	Descripción
Fot1	Fotocélula 1
Fot2	Fotocélula 2
SInd1	Sensor inductivo 1
SInd2	Sensor inductivo 2
vCintaL	Velocidad lenta de transporte en las cintas
vCintaR	Velocidad rápida de transporte en las cintas
vEmbolo	Velocidad de movimiento del cilindro
Marcha	Pulsador para la marcha
Paro	Interruptor con enclavamiento para el paro

Tabla 11: Entradas del Proceso 2.

Variables	Descripción
MCinta1L	Motor que mueve la cinta 1 a velocidad lenta
MCinta1R	Motor que mueve la cinta 1 a velocidad rápida
MCinta2L	Motor que mueve la cinta 2 a velocidad lenta
MCinta2R	Motor que mueve la cinta 2 a velocidad rápida
MCintaAnt	Motor que mueve la cinta anterior en la línea
Asc	Señal que hace subir el émbolo
Desc	Señal que hace bajar el émbolo
Parado	Señal que indica que el compenser está parado

Tabla 12: Salidas del Proceso 2.

La lógica empleada para el control del Proceso 2 queda reflejada en los dos gráfcets de la figura 10, con un primer gráfcet G0 diseñado para describir el control de parte de la tracción inicial del proceso; y un segundo gráfcet que además de interactuar con la segunda mitad de la tracción controla el desplazamiento del émbolo y el tiempo de parada de la segunda cinta.

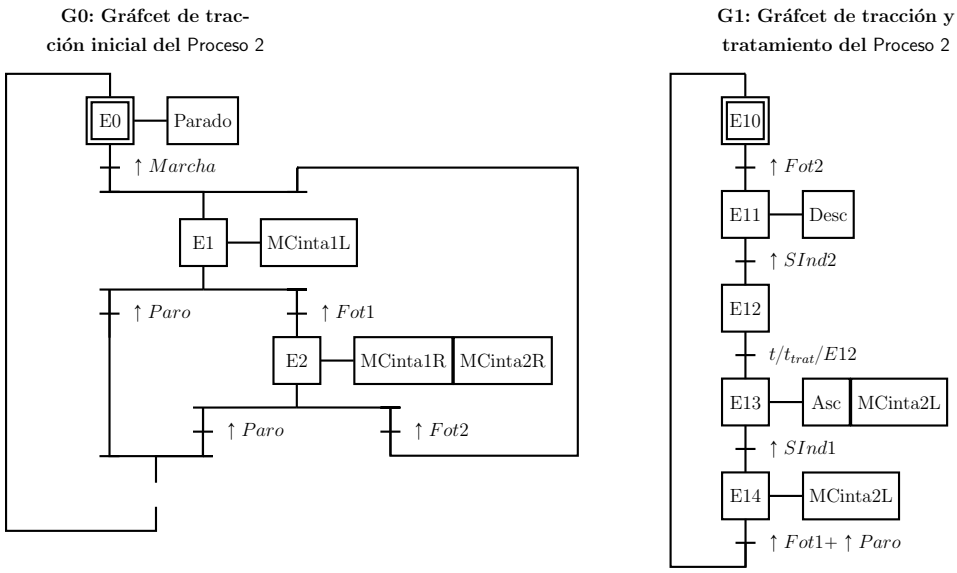

Figura 10: Gráfcets diseñados para el control del Proceso 2.

G0: Gráfcet de tracción inicial del Proceso 2 El gráfcet G0 comienza con su estado inicial, E0, cuya acción asociada es únicamente la salida Parado. Tras detectar un pulso en la señal Marcha se cambia de etapa a E1 donde la primera cinta del proceso se pone en marcha con velocidad lenta, es decir, la velocidad común determinada para todos los procesos. Al detectarse la activación de la primera fotocélula se cambia la velocidad de la primera cinta, pasando a moverse a velocidad rápida activando la saldia MCinta1R, y se pone en marcha a velocidad elevada también la segunda cinta mediante la señal MCinta2R; todo esto se realiza en la etapa E2. Cuando el azulejo llega a la fotocélula 2, Fot2, el gráfcet G0 se vuelve al estado anterior E1 preparado para recibir la siguiente pieza. Análogamente a las situaciones descritas en los procesos anteriores, estando en cualquiera de las dos etapas, E1 o E2, si se detecta un cambio en la variable Paro el controlador para todas las acciones relacionadas a este gráfcet devolviéndolo a su etapa inicial.

G1: Gráfcet de tracción y tratamiento del Proceso 2 El gráfcet G1 es un gráfcet lineal que parte de su estado inicial con la activación de la fotocélula 2. En dicho instante se para la tracción de la segunda cinta transportadora y se hace descender el émbolo activando la salida Desc. Cuando el émbolo llega al final de su recorrido, posición indicada por el sensor inductivo SInd2, se pasa a la etapa E12 y se realiza el estampado; además se mantiene por unos instantes la pieza quieta para evitar desperfectos en el tratamiento. El tiempo de parada queda determinado por el temporizador asociado a la etapa E12,

cuya descripción se realizará con la explicación del código del Proceso 2. Tras cumplirse el tiempo de espera se pasa a la etapa E13 en la que se asegura el ascenso del émbolo, acción marcada por la señal Asc, y también se hace avanzar la pieza hacia la salida del proceso a velocidad lenta, señal MCinta2L, para evitar deslizamientos con la cinta siguiente ajena al proceso actual. Aunque el ascenso del émbolo se realiza hasta activarse el sensor inductivo SInd1, la tracción en la segunda cinta es asegurada hasta activarse de nuevo la fotocélula de entrada en el proceso.

Como se puede observar, hay una alta correlación entre los dos gráfcets anteriores ya que el comienzo de G1 viene marcado por el final de G0, pero también el final de G1 ha de coincidir con la reactivación de la etapa E2 de G0. Esta correlación hace imprescindible un cálculo exacto del tiempo de avance y parada de la pieza sobre las cintas, tiempo que a la vez depende de la rapidez del descenso y ascenso del émbolo así como de la diferencia entre la velocidad considerada rápida y lenta.

7.4.4.1 Declaraciones del control automático del Proceso 2

Todas las variables declaradas para el control del Proceso 2 aparecen en los códigos 24 y 29 del Anexo 2. Como se puede observar la estructura es idéntica a la usada para los procesos anteriores, partiendo las variables según sean: entradas o salidas en el POU Variables y variables de proceso, flancos, etapas y temporizadores en el mismo POU de control.

Es conveniente mencionar que el temporizador que condiciona la permanencia en la etapa E12 del gráfcet G1 es un temporizador con retardo a la conexión, por tanto se declara TON_trat con tipo TON.

7.4.4.2 Código del control automático del Proceso 2

El código en texto estructurado escrito para la realización del control descrito anteriormente aparece por completo como código 30 del Anexo 2. Para esta parte del POU Proc2a se sigue una metodología parecida a las descritas anteriormente.

Aunque la mayor parte del código sea de fácil deducción dada la explicación realizada para los casos anteriores, se considera necesario profundizar en la parte de inicialización y consentimiento posterior.

En cuanto al código de inicialización, en este se calcula el tiempo preestablecido para el temporizador TON_trat, que coincide con el tiempo que el gráfcet G1 pasa en la etapa E12, siendo esta etapa la correspondiente al estado estacionario de la baldosa sobre la

segunda cinta del Proceso 2. Como ya se pone de manifiesto en puntos anteriores, la parada realizada por la cinta para la realización del estampado no ha de influir de forma alguna en el espaciado posterior de los azulejos. Por ello, dado que antes y después del proceso la velocidad de la línea es la misma, hay que conseguir que el tiempo que tarda la pieza en recorrer la distancia entre las fotocélulas Fot1 y Fot2 teniendo en cuenta el cambio de velocidad y la parada, sea el mismo que recorriendo la misma distancia a velocidad normal (durante el ensamblado de la simulación del Proceso 2 las dos fotocélulas anteriores se posicionan de tal modo que la distancia entre ellas sea la misma que la distancia entre el mismo punto de dos azulejos consecutivos, distancia llamada en el apartados anteriores d).

Hay que tener en cuenta también el tiempo de descenso y ascenso del émbolo, ya que con el gráfcet diseñado estas acciones no se realizan en paralelo con ningún otro desplazamiento; dicho tiempo es calculado teniendo en cuenta la distancia de descenso ($\mathsf{d_{desc}}$) y la velocidad de descenso del émbolo (vEmbolo). Del mismo modo hay que asegurar el tiempo necesario para que el azulejo sea transportado fuera del alcance del vástago, es decir, asegurar que la pieza sea desplazada su misma anchura (wb).

Por tanto se ha cumplir la ecuación:

$$\frac{d}{wMCintaP2L} = \frac{d}{wMCintaP2R} + \frac{d_{desc}}{wEB} + \frac{d_{desc}}{wES} + t_{trat} + \frac{wb}{wMCintaP2L}$$

Cambiando algunos valores y simplificando se llega a la expresión programada en el bloque de inicialización:

$$t_{trat} = \frac{wb}{wMCintaP2L} - 2\frac{d_{desc}}{wEB} - \frac{d}{wMCintaP2R}$$

En cuanto a las siguientes dos líneas programadas en el código, estas son idénticas a las escritas para el control automático del Proceso 1: en el instante inicial de la simulación se pone en Marcha este proceso si se configura como tal; y en segundo lugar, el no consentimiento viene del Compenser 1a, dado que se trata del Proceso 2 situado en la primera parte.

8. GUÍA PARA EL USO DEL SIMULADOR

A lo largo de este punto se describe de forma detallada la metodología a emplear para conseguir ejecutar el simulador desarrollado en este proyecto, pudiéndole sacar el máximo provecho en cuanto a los propósitos académicos descritos se refiere.

8.1. Inicialización en CodeSys

Uno de los complementos de partida al iniciar el simulador es el POU Sim (PRG), POU principal que gestiona la evolución de la visualización durante la ejecución de la simulación, según las salidas del control automático a implementar. Es en esta parte del programa donde se configura la línea de producción cerámica a simular en la visualización así como el periodo de actualización de los elementos simulados.

La línea simulada está dividida en dos partes: la parte A ocupando los dos procesos de la primera mitad de la visualización y la parte B que engloba los otros dos posibles procesos situados en la segunda mitad. La configuración de la línea simulada se realiza modificando el vector elem=[a,b,c] en las declaraciones del POU Sim (PRG) según la tabla 13.

Parte	Proceso configurado	Elemento del vector	Valor
A	-	b	0
A	$Proceso1A$	b	1
A	$Proceso2A$	b	2
B	-	c	0
B	$Proceso1B$	c	1
B	$Proceso2B$	c	2
B	-	a	0
B	$CompenserB$	a	1

Tabla 13: Posibilidades de configuración del simulador.

Otro parámetro a configurar de este POU es el periodo de simulación t_{sim}, cuyo valor es recomendable no bajar de 20ms. Cabe destacar en este punto que el tiempo definido en maintask ha de ser siempre inferior a t_{sim}; un valor aceptable dado el tiempo anterior son 10ms.

En segundo lugar, junto con el POU de simulación, aparecen las declaraciones de todas las variables de entrada y salida de los posibles procesos a simular en un POU exclusivo para variables llamado del mismo modo: Variables. Por consiguiente, el maintask del CodeSys

inicial ha de contener por lo menos estas dos partes del programa, recorriendo en primer lugar el Sim (PRG) y luego el POU Variables.

8.2. Programación del control automático

El control automático a desarrollar para la línea simulada varía en función de la configuración ejecutada, y de los procesos involucrados en la simulación. Dada la cantidad de procesos a controlar se recomienda repartir el algoritmo de control en distintos POUs, uno por cada proceso, haciendo así la tarea de programar más modular y sistemática.

Para garantizar una gestión completa del simulador, el POU Sim (PRG) necesita acceder a las variables relacionadas con las entradas y las salidas de cada proceso, y sea visualizado en la simulación configurada o no. Como consecuencia, el simulador accederá a las variables declaradas en el POU Variables para hacer las comprobaciones y los controles de visualización.

Además, las decisiones tomadas por el control automático diseñado por el usuario también han de modificar los valores declarados en el POU Variables, para que así el controlador de la simulación acceda a estas y evolucione la simulación en consecuencia. Del mismo modo, los cambios en el sistema se escriben en las variables de entrada declaradas en el mismo POU; por ello, el control automático hecho por el usuario ha de basarse en los valores de dichas entradas, valores modificados por el controlador del simulador.

Cabe destacar que el usuario puede programar el control automático en tantos POUs como crea conveniente, y además escribiendo en cualquiera de los lenguajes que CodeSys ofrece, ya sea Texto Estructurado, Diagramas Ladder, Lista de Instrucciones, Diagramas de Bloques de Funciones o Bloques de Función Secuenciales.

En los subapartados siguientes se proporciona la información necesaria para que el usuario con intención de programar un posible control automático pueda hacerlo de forma correcta.

8.2.1. Azulejo

En este subapartado se describen propiedades geométricas del azulejo, centrándose en las dimensiones de este elemento siendo estas imprescindibles para realizar un control adecuado. Además, otro parámetro imprescindible es la distancia (we) por defecto a dejar entre piezas para conseguir la compatibilidad con el simulador.

Ancho wb (px)	**Altura (px)**	**we (px)**
60	5	26

Tabla 14: Dimensiones y espaciado del elemento azulejo.

8.2.2. Compenser

El compenser es uno de los procesos más complejos no tanto por el funcionamiento sino en cuanto a control se refiere. En la figura 8 se muestra la representación que se le da en este simulador.

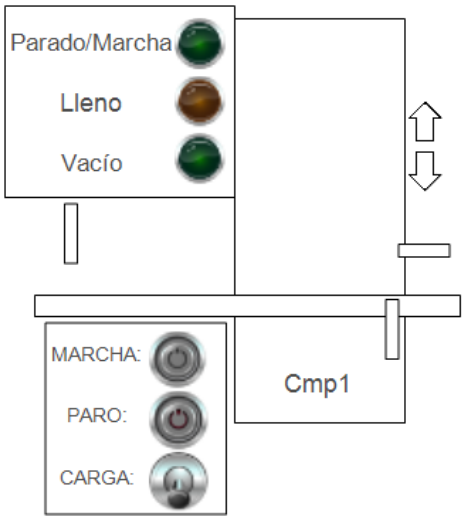

Figura 11: Representación del compenser.

Su funcionamiento es intuitivo: las piezas llegan a este proceso movidas por la cinta controlada por la señal MCinta de izquierda a derecha. Cuando llegan a la altura de la fotocélula 2 (fotocélula situada en la parte posterior del proceso a nivel de cinta), en caso de que, ya sea por no consentimiento posterior o por orden manual, se tenga la orden de cargar, los azulejos son elevados hasta que el sensor inductivo (SInd) detecte la presencia de la pieza. Por el contrario, si no se tiene dicha orden, el compenser se queda a la espera de detectar un hueco en la línea para soltar las piezas que tiene almacenadas (siempre y cuando haya piezas que descargar) activando la señal MCBajar. Para calcular si hay espacio disponible para realizar una descarga adecuadamente, es decir, sin provocar colisiones y respetando siempre la distancia entre piezas, el controlador del compenser debe usar la fotocélula de entrada al proceso: Fot1.

Es necesario hacer hincapié en la salida MCintaAnt ya que esta va ligada al funcionamiento

de la cinta anterior al compenser. Como se puede deducir, esta cinta ha de transportar piezas siempre y cuando la cinta que atraviesa el compenser también lo haga, es decir, siempre y cuando el proceso compenser no esté parado.

Las variables de entrada y salida definidas en el POU Variables para este proceso aparecen en las tablas 15 y 16 respectivamente.

Variables	**Descripción**
Fot1Ca	Fotocélula 1
Fot2Ca	Fotocélula 2
SIndCa	Sensor inductivo
MarchaMCa	Pulsador para la marcha
ParoMCa	Interruptor con enclavamiento para el paro
CargaMCa	Interruptor para empezar la carga del compenser
vCintaCa	Velocidad de transporte en la cinta
vSubirCa	Velocidad de carga y descarga de los azulejos

Tabla 15: Entradas del compenser de la parte A.

Variables	**Descripción**
MCintaCa	Motor que mueve la cinta
MCintaAntCa	Motor que mueve la cinta anterior en la línea
ParadoCa	Señal que indica que el compenser está parado
LlenoCa	Señal que indica que el compenser está lleno
VacíoCa	Señal que indica que el compenser está vacío
MCSubirCa	Motor que hace ascender las piezas
MCBajarCa	Motor que hace descender las piezas

Tabla 16: Salidas del compenser de la parte A.

Cabe destacar las señales homólogas a las anteriores para el compenser de la parte B cambian la última letra de su nombre, pasando a acabar todas con la terminación Cb.

Para poder implementar un control adecuado son necesarios parámetros geométricos del diseño, además de algunas velocidades con las que el azulejo es movido. En la tabla 17 aparecen estas velocidades, además de los desplazamientos que sufren las piezas entre algunos elementos del proceso.

Compenser	
v_{CINTA} (px/t_{sim})	1
v_{SUBIR} (px/t_{sim})	1
d_{F1-F2} (px)	142
$d_{F2-SInd}$ (px)	10

Tabla 17: Parámetros de diseño del compenser.

8.2.3. Proceso 1

El Proceso 1 consta de una cinta movida por un motor cuya señal de activación es MCinta. Al alcanzar la fotocélula el controlador detecta la entrada del azulejo en el proceso y espera a que este llegue a la posición del tratamiento, para aplicarlo a la pieza mediante la señal Tratamiento. Hay que tener en cuenta que dadas las dimensiones del proceso la aplicación del tratamiento a una pieza puede coincidir con la entrada de otro azulejo por la cinta. En la figura 12 se muestra este proceso simulado:

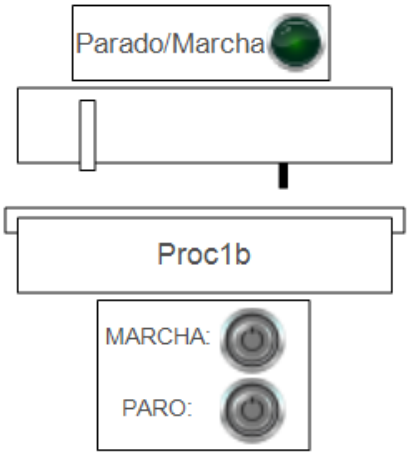

Figura 12: Representación del Proceso 1.

De forma análoga al caso anterior, la salida MCintaAnt hace la misma función, y es necesario que esté activa siempre que se quiera el transporte anterior, es decir, siempre que la cinta del proceso también lo esté.

Para realizar un control adecuado son necesarias el nombre concreto de las variables declaradas en el POU Variables; dichas entradas y salidas aparecen en las tablas 18 y 19.

Variables	Descripción
MarchaMP1a	Pulsador para la marcha
ParoMP1a	Interruptor con enclavamiento para el paro
FotP1a	Fotocélula
vCintaP1a	Velocidad de transporte en la cinta

Tabla 18: Entradas del Proceso 1 de la parte A.

Variables	Descripción
MCintaP1a	Motor que mueve la cinta
MCintaAntP1a	Motor que mueve la cinta anterior en la línea
TratamientoP1a	Señal que indica la aplicación de tratamiento sobre la pieza
ParadoP1a	Señal que indica que el Proceso 1 está parado

Tabla 19: Salidas del Proceso 1 de la parte A.

De forma análoga al caso anterior, para el Proceso 1 de la parte B solo cambia la terminación de cada señal, acabando todas estas en P1b.

Para este caso, el algoritmo de control se puede conseguir con los parámetros de la tabla 20.

Proceso 1	
v_{CINTA} (px/t_{sim})	1
$d_{F1-Trat}$ (px)	80

Tabla 20: Parámetros de diseño del Proceso 1.

8.2.4. Proceso 2

El Proceso 2 aporta más complejidad a la hora de controlar dado que incorpora dos cintas diferentes, cada una de la cual puede funcionar a dos velocidades diferentes, y una parada total del azulejo durante el proceso. Además, también incorpora un cilindro con un émbolo neumático para realizar un estampado en la baldosa, de ahí la necesidad de parar la pieza cerámica. Su representación en la visualización aparece en la figura 13.

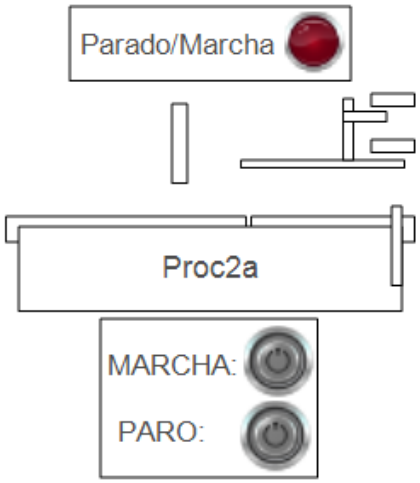

Figura 13: Representación del Proceso 2.

En cuanto a funcionamiento, las piezas entran en el proceso a velocidad lenta movidas por la primera cinta, señal MCinta1L; al ser detectadas por la fotocélula 1 (Fot1, situada más a la izquierda), dichas piezas son transportadas a una velocidad superior por la misma cinta 1 pasando debido a su desplazamiento sobre la cinta 2, que en consecuencia ha de transportar a la misma velocidad para evitar deslizamiento. Las velocidades altas en las cintas 1 y 2 se consiguen con las señales MCinta1R y MCinta2R respectivamente. Cuando la pieza en cuestión llega a activar la fotocélula 2 (Fot2), la segunda cinta para, junto con la pieza, para dejar paso a la bajada del vástago del cilindro neumático, y a que este aplique el correspondiente estampado. Tras volver a elevarse el cilindro la pieza vuelve a ponerse en marcha para ser llevada aguas abajo en la línea; esto último se ha de realizar a velocidad lenta ya que es esa la velocidad de los demás procesos, y se consigue activando la señal MCinta2L. Cabe destacar que para parar la segunda cinta basta con desactivar las dos señales asociados al motor. El movimiento del cilindro es asegurado por las señales Asc y Desc, haciéndolo ascender y descender según el caso. El cilindro se diseña con unos finales de carrera incorporados, SInd1 para subir y SInd2 para bajar, para controlar su movimiento. La señal MCintaAnt funciona de forma idéntica a los casos anteriores.

Las señales mencionadas en el párrafo anterior entre otras necesarias para realizar el control automático del Proceso 2, declaradas todas en el POU Variables, se muestran en las tablas 21 y 22.

Variables	**Descripción**
MarchaMP2a	Pulsador para la marcha
ParoMP2a	Interruptor con enclavamiento para el paro
Fot1P2a	Fotocélula 1
Fot2P2a	Fotocélula 2
SInd1P2a	Sensor inductivo 1
SInd2P2a	Sensor inductivo 2
vCintaLP2a	Velocidad lenta de transporte en las cintas
vCintaRP2a	Velocidad rápida de transporte en las cintas
vEmboloP2a	Velocidad de movimiento del cilindro

Tabla 21: Entradas del Proceso 2 de la parte A.

Variables	Descripción
MCinta1LP2a	Motor que mueve la cinta 1 a velocidad lenta
MCinta1RP2a	Motor que mueve la cinta 1 a velocidad rápida
MCinta2LP2a	Motor que mueve la cinta 2 a velocidad lenta
MCinta2RP2a	Motor que mueve la cinta 2 a velocidad rápida
MCintaAntP2a	Motor que mueve la cinta anterior en la línea
AscP2a	Señal que hace subir el émbolo
DescP2a	Señal que hace bajar el émbolo
ParadoP2a	Señal que indica que el Proceso 2 está parado

Tabla 22: Salidas del Proceso 2 de la parte A.

Las mismas variables pero relacionadas con el Proceso 2 de la parte B se diferencian de estas por la última letra, acabando todas en P2b.

El control automático de este proceso depende además de los parámetros de la tabla 23.

Compenser	
$v_{CINTA1L}$ (px/t_{sim})	1
$v_{CINTA1R}$ (px/t_{sim})	8
$v_{CINTA2L}$ (px/t_{sim})	1
$v_{CINTA2R}$ (px/t_{sim})	8
v_{Embolo} (px/t_{sim})	4
$desp_{F1-F2}$ (px)	86
$desp_{Embolo}$ (px)	16

Tabla 23: Parámetros de diseño del Proceso 2.

9. Referencias

- Murria De Las Heras, J.; Sanchis Llopis, R. (Coord). (2000).*Desarrollo de un sistema de control distribuido reconfigurable para líneas de esmaltado.* Castellón de la Plana: Universidad Jaume I.

- Sanchis Llopis, R.; Romero Pérez, J.A.; Ariño Latorre, C.V.. (2010). *Automatización industrial.* Castellón de la Plana: Universidad Jaume I.

- *User manual for PLC programming with CodeSys 2.3.* (2003). Versión 2.0. 3S - Smart Software Solutions GmbH https://www.parkermotion.com/manuals/Hauser/Compax3/CoDeSys_Manual_V2p3.pdf 06/09/2017

- *CodeSys Control V3 Manual.* Versión 18.0. 3S - Smart Software Solutions GmbH

  http://support.crosscontrol.com/sites/default/files/documentation/Software/CoDeSys/Documentation/CODESYSControl_V3_Manual.pdf

  06/09/2017

- *The CoDeSys Visualization. Supplement to the User Manual for PLC Programming with CoDeSys 2.3.* (2003) Versión 1.0. 3S - Smart Software Solutions GmbH http://www.dimoulas.com.gr/PDF/WAGO%20PLC/CoDeSys_Visu_E.pdf

  06/09/2017

ANEXO I: Código completo del simulador

```
PROGRAM Sim
VAR_INPUT
  (*RECONFIGURABILIDAD: *)
    elem: ARRAY[0..3] OF INT := [1, 2, 2]; (*
      [a, b, c] -> a = 0, no hay Compenser b
              a = 1, hay Compenser b
              b = 0, no hay Proceso en a
              b = 1, hay Proc1 en a
              b = 2, hay Proc2 en a
              c = 0, no hay Proceso en b
              c = 1, hay Proc1 en b
              c = 2, hay Proc2 en b
      *)

  (*Parametros de la visualizacion: *)
    (*Ciclo de simulacion: *)
      tsim: REAL := 15; (*REAL para realizar operaciones*)
    (*Generales: *)
      b: INT := 19; limh: REAL;
    (*Baldosa: *)
      d: REAL := 86; wb: REAL := 60; we: REAL := 26; hb: REAL := 5;
    (*Incrementos/Velocidades: *)
      Ih: REAL := 1; IhP2: REAL := 8; IvC: REAL := 1; IvP2: REAL := 4;
    (*Compenser: *)
      (*Generales: *)
        dbaj: REAL := 10; Pastart: REAL := -202; Pastop: REAL := -42; Pbstart: REAL := 366;
        Pbstop: REAL := 566; Castart: REAL := 62; Castop: REAL := 262; Cbstart: REAL := 630;
        Cbstop: REAL := 830; PC2P2aStart: REAL := -106; PC2P2bStart: REAL := 462;

      (*Cmp1*)
        CpF1on: REAL := 86; CpF1off: REAL := 92; CpF2on: REAL := 228; CpF2off: REAL := 233;
        CpF3on: REAL := 10; CpF3off: REAL := 16;
      (*Cmp2*)
        CbpF1on: REAL := 654; CbpF1off: REAL := 660; CbpF2on: REAL := 796;
```

```
35          CbpF2off: REAL := 802; CbpF3on: REAL := 10; CbpF3off: REAL := 16;
36      (*Procesos: *)
37        (*Prc1: *)
38          P1pFon: REAL := -172; P1pFoff: REAL := -166; P1pCh: REAL := -92;
39        (*Prc1b: *)
40          P1bpFon: REAL := 396; P1bpFoff: REAL := 402; P1bpCh: REAL := 476;
41        (*Prc2: *)
42          P2pF1on: REAL := -137; P2pF1off: REAL := -131; P2pF2on: REAL := -51;
43          P2pF2off: REAL := -45; P2pF3on: REAL := 14;
44          ddesc: REAL := 4;
45        (*Prc2: *)
46          P2bpF1on: REAL := 431; P2bpF1off: REAL := 437; P2bpF2on: REAL := 517;
47          P2bpF2off: REAL := 523; P2bpF3on: REAL := 14;
48
49    (*Variables de activacion de Cintas: *)
50      (*Fuera de procesos: *)
51        CintaA: BOOL := 0;
52        CintaB: BOOL := 0;
53        CintaA1: BOOL := 0;
54        CintaB1: BOOL := 0;
55        CintaA2: BOOL := 0;
56        CintaB2: BOOL := 0;
57      (*Parte a: *)
58        (*Compenser: *)  CintaCa: BOOL := 0; SubirCa: BOOL := 0; BajarCa: BOOL := 0;
59        (*Proceso 1: *)  CintaP1a: BOOL := 0;
60        (*Proceso 2: *)  CintaP2a1: BOOL := 0; CintaP2a2: BOOL := 0;
61      (*Parte b: *)
62        (*Compenser: *)  CintaCb: BOOL := 0; SubirCb: BOOL := 0; BajarCb: BOOL := 0;
63        (*Proceso 1: *)  CintaP1b: BOOL := 0; CintaP2b1: BOOL := 0;
64        (*Proceso 2: *) CintaP2b2: BOOL := 0;
65
66    (*Vectores *)
67      visElem: ARRAY[0..8] OF BOOL; //[Cmpb, Proc1a, Proc2a, Proc1b, Proc2b,
68                                      barras_a, barras_b, Barra_a, Barra_b ]
69      posx: ARRAY[0..19] OF REAL;
70      posy: ARRAY[0..19] OF REAL;
```

```
71      inv: ARRAY[0..19] OF BOOL;
72      i: INT; k: INT;
73
74    (*Tratamiento proceso 1: *) Trat1: BOOL := 0; Trat1b: BOOL := 0;
75    (*Tratamiento proceso 2: *)  posVa: REAL := 0; posVb: REAL := 0;
76
77    (*Auxiliares *)  Inicializado: BOOL := 0; pos: INT := b; Operario: BOOL := 0;
78
79    (*Temporizador *)
80      TP_while: TP;
81      S_t_while: BOOL;
82      PT_t_while: TIME := REAL_TO_TIME(tsim); (*Tiempo entre desplazamientos *)
83      Q_t_while: BOOL;
84      ET_t_while: TIME;
85      t_while: BOOL;
86  END_VAR
```

Código 22: Declaraciones del POU Sim (PRG)

```
1   (*Inicializacion de los vectores *)
2     IF NOT Inicializado THEN
3       FOR i := 1 TO b BY 1 DO
4         posx[i] := posx[0]-i*d;
5         posy[i] := 0;
6         inv[i] := 0;
7       END_FOR
8     (*RECONFIGURABILIDAD: *)
9       (*Inicializa el vector de visibiliadad a 0: *)
10        FOR i := 0 TO 8 BY 1 DO visElem[i] := 0; END_FOR
11
12        IF elem[0] = 1 THEN visElem[0] := 1; ELSE visElem[0] := 0; END_IF
13
14        IF elem[1] = 0 THEN visElem[7] := 1;
15        ELSIF elem[1] = 1 THEN visElem[5] := 1; visElem[1] := 1;
16        ELSIF elem[1] = 2 THEN visElem[5] := 1; visElem[2] := 1;
17        ELSE ; END_IF
18
```

```
19      IF elem[2] = 0 AND elem[0] = 1 THEN visElem[8] := 1;
20      ELSIF elem[2] = 0 AND elem[0] = 0 THEN  visElem[8] := 0;
21      ELSIF elem[2] = 1 THEN visElem[6] := 1; visElem[3] := 1;
22      ELSIF elem[2] = 2 THEN visElem[6] := 1; visElem[4] := 1;
23      ELSE ; END_IF
24
25      (*Limite de retorno de las piezas: *)
26        IF NOT visElem[0] AND NOT visElem[6]  THEN limh := Castop;
27        ELSIF NOT visElem[0] THEN limh := 565;
28        ELSE limh := Cbstop; END_IF
29
30    (*Pone la baldosas como invisibles: *) inv[2] := 1; inv[9] := 1; inv[16] := 1;
31
32  (*Calculo de velocidades en funcion de tsim: *)
33    (*Compenser a: *) Variables.vCintaCa := Ih/tsim; Variables.vSubirCa := IvC/tsim;
34    (*Compenser b: *) Variables.vCintaCb := Ih/tsim; Variables.vSubirCb := IvC/tsim;
35    (*Proceso 1a: *) Variables.vCintaP1a := Ih/tsim;
36    (*Proceso 1b: *) Variables.vCintaP1b := Ih/tsim;
37    (*Proceso 2a: *) Variables.vCintaLP2a := Ih/tsim; Variables.vCintaRP2a := IhP2/tsim;
38    Variables.vEmboloP2a := IvP2/tsim;
39    (*Proceso 2b: *) Variables.vCintaLP2b := Ih/tsim; Variables.vCintaRP2b := IhP2/tsim;
40    Variables.vEmboloP2b := IvP2/tsim;
41
42  S_t_while := 1;
43  Inicializado := 1;
44  END_IF
45
46 //(*Debug mode: *) FOR i := 1 TO b BY 2 DO inv[i] := 1; END_FOR
47
48 (*Reactivacion del temporizador para la proxima vez *)
49  TP_while(IN := S_t_while, PT := PT_t_while);
50
51 IF ( TP_while.ET >= TP_while.PT ) THEN
52  (*Timmer *)
53    S_t_while := 0;
54    TP_while(IN := S_t_while, PT := PT_t_while);
```

```
55      S_t_while := 1;
56      TP_while(IN := S_t_while, PT := PT_t_while);
57
58  (*Intervencion del operario: *)
59  IF Operario THEN
60    FOR i := 0 TO b BY 1 DO
61      (*Para cuando hay piezas debajo de los procesos: *)
62        (*Proca: *)
63          IF (posx[i] > Pastart AND posx[i]< Pastop-wb) THEN
64            IF visElem[1] AND Variables.ParadoP1a THEN inv[i] := 1; END_IF
65            IF visElem[2] AND Variables.ParadoP2a THEN inv[i] := 1; END_IF
66          END_IF
67        (*Procb: *)
68          IF (posx[i] > Pbstart AND posx[i] < Pbstop-wb) THEN
69            IF visElem[3] AND Variables.ParadoP1b THEN inv[i] := 1; END_IF
70            IF visElem[4] AND Variables.ParadoP2b THEN inv[i] := 1; END_IF
71          END_IF
72      (*Para la salida del compenser a: *)
73        IF posx[i] > CpF2off AND posx[i] <= Castop THEN
74          IF NOT visElem[3] AND NOT visElem[4] AND visElem[0] AND Variables.ParadoCb
75            THEN inv[i] := 1;
76          ELSIF visElem[3] AND Variables.ParadoP1b THEN inv[i] := 1;
77          ELSIF visElem[4] AND Variables.ParadoP2b THEN inv[i] := 1;
78          END_IF
79        END_IF
80      (*Solucionar colisiones: *)
81        FOR k := 0 TO b BY 1 DO
82          IF ((posx[i] < Pastart) OR (posx[i] > Pastop AND posx[i] < Pbstart) OR
83            posx[i] > Pbstop) AND ((posx[k]< Pastart) OR (posx[k]> Pastop AND posx[k]< Pbstart)
84            OR posx[k] > Pbstop) AND i <> k AND inv[k] <> 1 AND inv[i] <> 1 AND posy[i] = 0
85            AND ((ABS(posx[i] - posx[k]) < (d)) ) AND posy[k] = 0
86              THEN inv[k] := 1; END_IF
87        END_FOR
88    END_FOR
89  END_IF
90
```

```
91  (*Desplazamientos: *)
92    (*Desplazamiento vertical: *)
93      (*Desplazamientos verticales del compenser a: *)
94        FOR i := 0 TO b BY 1 DO
95          IF NOT inv[i] AND Variables.MCSubirCa AND Sim.posx[i] = CpF2on-wb
96            THEN Sim.posy[i] := Sim.posy[i] + IvC; END_IF
97          IF NOT inv[i] AND Variables.MCBajarCa AND Sim.posx[i] = CpF2on-wb AND Sim.posy[i] > 0
98            THEN Sim.posy[i] := Sim.posy[i] - IvC; END_IF
99        END_FOR
100     (*Desplazamientos verticales del compenser b: *)
101       IF visElem[0] THEN
102         FOR i := 0 TO b BY 1 DO
103           IF NOT inv[i] AND Variables.MCSubirCb AND Sim.posx[i] = CbpF2on-wb
104             THEN Sim.posy[i] := Sim.posy[i] + IvC; END_IF
105           IF NOT inv[i] AND Variables.MCBajarCb AND Sim.posx[i] = CbpF2on-wb
106           AND Sim.posy[i] > 0 THEN Sim.posy[i] := Sim.posy[i] - IvC; END_IF
107         END_FOR
108       END_IF
109
110   (*Desplazamiento horizontal: *)
111       FOR i := 0 TO b BY 1 DO
112         IF posy[i] = 0 THEN
113         (*Antes del Proceso a: *)
114           IF posx[i] < Pastart-wb THEN
115             IF Variables.MCintaAntP1a OR Variables.MCintaAntP2a
116               THEN Sim.posx[i] := Sim.posx[i] + Ih; END_IF
117             IF (NOT visElem[1] AND NOT visElem[2]) AND Variables.MCintaAntCa
118               THEN Sim.posx[i] := Sim.posx[i] + Ih; END_IF
119         (*Cinta 1 del proceso 2a *)
120           ELSIF Sim.posx[i] < PC2P2aStart-wb THEN
121             IF visElem[2] THEN
122               IF Variables.MCinta1LP2a THEN Sim.posx[i] := Sim.posx[i] + Ih; END_IF
123               IF  Variables.MCinta1RP2a THEN
124                 Sim.posx[i] := Sim.posx[i] + IhP2; END_IF END_IF
125             IF visElem[1] AND Variables.MCintaP1a THEN
126               Sim.posx[i] := Sim.posx[i] + Ih; END_IF
```

```
127             IF (NOT visElem[1] AND NOT visElem[2]) AND Variables.MCintaAntCa
128               THEN Sim.posx[i] := Sim.posx[i] + Ih; END_IF
129         (*Cinta 2 del proceso 2a *)
130           ELSIF Sim.posx[i] < Pastop-wb THEN
131             IF visElem[2] THEN
132               IF Variables.MCinta2LP2a THEN Sim.posx[i] := Sim.posx[i] + Ih; END_IF
133               IF  Variables.MCinta2RP2a THEN Sim.posx[i] := Sim.posx[i] + IhP2; END_IF END_IF
134             IF visElem[1] AND Variables.MCintaP1a THEN Sim.posx[i] :=Sim.posx[i] +Ih; END_IF
135             IF (NOT visElem[1] AND NOT visElem[2]) AND Variables.MCintaAntCa
136               THEN Sim.posx[i] := Sim.posx[i] + Ih; END_IF
137         (*Antes el Cmp1a: *)
138           ELSIF posx[i] < Castop THEN
139             IF Variables.MCintaAntCa THEN Sim.posx[i] := Sim.posx[i] + Ih; END_IF
140         (*Durante el Cmp1a: *)
141           ELSIF posx[i] < Castop THEN
142             IF Variables.MCintaCa THEN Sim.posx[i] := Sim.posx[i] + Ih; END_IF
143         (*Antes del proceso b: *)
144           ELSIF posx[i] < Pbstart-wb THEN
145             IF Variables.MCintaAntP1b OR Variables.MCintaAntP2b THEN
146               Sim.posx[i] := Sim.posx[i] + Ih; END_IF
147             IF (NOT visElem[3] AND NOT visElem[4]) AND Variables.MCintaCb
148               THEN Sim.posx[i] := Sim.posx[i] + Ih; END_IF
149         (*Cinta 1 del proceso 2b *)
150           ELSIF Sim.posx[i] < PC2P2bStart-wb THEN
151             IF visElem[4] THEN
152               IF Variables.MCinta1LP2b THEN Sim.posx[i] := Sim.posx[i] + Ih; END_IF
153               IF  Variables.MCinta1RP2b THEN Sim.posx[i] := Sim.posx[i] + IhP2; END_IF
154             END_IF
155             IF visElem[3] AND Variables.MCintaP1b THEN Sim.posx[i]:= Sim.posx[i] + Ih; END_IF
156             IF (NOT visElem[3] AND NOT visElem[4]) AND Variables.MCintaAntCb
157               THEN Sim.posx[i] := Sim.posx[i] + Ih; END_IF
158         (*Cinta 2 del proceso 2b *)
159           ELSIF Sim.posx[i] < Pbstop-wb+3 THEN
160             IF visElem[4] THEN
161               IF Variables.MCinta2LP2b THEN Sim.posx[i] := Sim.posx[i] + Ih; END_IF
162               IF  Variables.MCinta2RP2b THEN Sim.posx[i] := Sim.posx[i] + IhP2; END_IF END_IF
```

```
163             IF visElem[3] AND Variables.MCintaP1b THEN Sim.posx[i]:= Sim.posx[i] + Ih; END_IF
164             IF (NOT visElem[3] AND NOT visElem[4]) AND Variables.MCintaAntCb THEN
165              Sim.posx[i] := Sim.posx[i] + Ih; END_IF
166         (*Antes el Cmp1b: *)
167           ELSIF posx[i] < Cbstop THEN
168             IF Variables.MCintaAntCb THEN Sim.posx[i] := Sim.posx[i] + Ih;
169             ELSIF NOT visElem[0] AND visElem[6] THEN Sim.posx[i] := Sim.posx[i] + Ih; END_IF
170         (*Durante el Cmp1b: *)
171           ELSIF posx[i] < (*Cbstop*)limh THEN
172             IF Variables.MCintaCb THEN Sim.posx[i] := Sim.posx[i] + Ih; END_IF
173           END_IF
174     (*Caso de alcanzar el limite: *)
175           IF Sim.posx[i] >= limh THEN
176             IF posx[pos] < -4*d THEN  posx[i] := posx[pos]-d;
177             ELSE posx[i] := -5*d;
178             END_IF
179             IF i <> 2 AND i <> 9 AND i <> 16 THEN inv[i] := 0; END_IF
180             pos := i;
181           END_IF
182         END_IF
183       END_FOR
184
185 (*Tratamiento realizado en el Proceso 1: *)
186   (*Proc1: *)
187     IF Variables.TratamientoP1a THEN Trat1 := 1; ELSE Trat1 := 0; END_IF
188
189   (*Proc1b: *)
190     IF Variables.TratamientoP1b THEN Trat1b := 1; ELSE Trat1b := 0; END_IF
191
192 (*Tratamiento realizado en el Proceso 2: *)
193   (*a *)
194     IF Variables.DescP2a THEN posVa := posVa - IvP2;
195     ELSIF Variables.AscP2a THEN posVa := posVa + IvP2;
196     END_IF
197   (*b *)
198     IF Variables.DescP2b THEN posVb := posVb - IvP2;
```

```
199       ELSIF Variables.AscP2b THEN posVb := posVb + IvP2;
200       END_IF
201
202   (*Activacion de la visualizacion de actuadores: *)
203     (*Fuera de procesos: *)
204       CintaA := NOT (visElem[1] AND visElem[2]) AND Variables.MCintaAntCa;
205       CintaA1 := (visElem[1] AND Variables.MCintaAntP1a)
206         OR (visElem[2] AND Variables.MCintaAntP2a);
207       CintaA2  := Variables.MCintaAntCa;
208       CintaB := NOT (visElem[3] AND visElem[4]) AND Variables.MCintaAntCb;
209       CintaB1 := (visElem[3] AND Variables.MCintaAntP1b)
210         OR (visElem[4] AND Variables.MCintaAntP2b);
211       CintaB2  := Variables.MCintaAntCb OR (NOT visElem[0] AND visElem[6]);
212     (*Parte a: *)
213       (*Compenser: *) CintaCa := Variables.MCintaCa; SubirCa := Variables.MCSubirCa;
214                        BajarCa := Variables.MCBajarCa;
215       (*Proceso 1: *) CintaP1a := visElem[1] AND Variables.MCintaP1a;
216       (*Proceso 2:*) CintaP2a1:=visElem[2] AND (Variables.MCinta1LP2a OR Variables.MCinta1RP2a);
217                      CintaP2a2:=visElem[2] AND (Variables.MCinta2LP2a OR Variables.MCinta2RP2a);
218     (*Parte b: *)
219       (*Compenser: *) CintaCb := visElem[0] AND Variables.MCintaCb;
220                        SubirCb := Variables.MCSubirCb; BajarCb := Variables.MCBajarCb;
221       (*Proceso 1: *) CintaP1b := visElem[3] AND Variables.MCintaP1b;
222       (*Proceso 2: *) CintaP2b1 := visElem[4] AND NOT Variables.ParadoP2b;
223               CintaP2b2 := visElem[4] AND (Variables.MCinta2LP2b OR Variables.MCinta2RP2b);
224
225   (*Actualizacion de fotocelulas: *)
226     (*Compenser: *)
227       (*a *)
228         FOR i := 0 TO b BY 1 DO
229           IF NOT inv[i] AND posx[i] >= CpF1on-wb AND posx[i] < CpF1off AND posy[i] = 0
230             THEN Variables.Fot1Ca := 1; EXIT; ELSE Variables.Fot1Ca := 0; END_IF END_FOR
231         FOR i := 0 TO b BY 1 DO
232           IF NOT inv[i] AND posx[i] >= CpF2on-wb AND posx[i] < CpF2off AND posy[i] = 0
233             THEN Variables.Fot2Ca := 1; EXIT; ELSE Variables.Fot2Ca := 0; END_IF END_FOR
234         FOR i := 0 TO b BY 1 DO
```

```
235          IF NOT inv[i] AND posy[i] >= CpF3on AND posy[i] < CpF3off AND posx[i] = CpF2on-wb
236            THEN Variables.SIndCa := 1; EXIT; ELSE Variables.SIndCa := 0; END_IF END_FOR
237        (*b *)
238          FOR i := 0 TO b BY 1 DO
239          IF NOT inv[i] AND visElem[0] AND posx[i] >= CbpF1on-wb AND posx[i] < CbpF1off
240            AND posy[i] =0 THEN Variables.Fot1Cb := 1; EXIT; ELSE Variables.Fot1Cb := 0; END_IF
241          END_FOR
242          FOR i := 0 TO b BY 1 DO
243          IF NOT inv[i] AND visElem[0] AND posx[i] >= CbpF2on-wb AND posx[i] < CbpF2off
244            AND posy[i] = 0 THEN Variables.Fot2Cb := 1; EXIT; ELSE Variables.Fot2Cb:=0; END_IF
245          END_FOR
246          FOR i := 0 TO b BY 1 DO
247          IF NOT inv[i] AND visElem[0] AND posy[i] >= CbpF3on AND posy[i] < CbpF3off
248            AND posx[i] = CbpF2on-wb THEN Variables.SIndCb:=1; EXIT; ELSE Variables.SIndCb:=0;
249            END_IF END_FOR
250      (*Proceso1: *)
251        (*a *)
252          FOR i := 0 TO b BY 1 DO
253          IF NOT inv[i] AND visElem[1] AND posx[i] >= P1pFon-wb AND posx[i] < P1pFoff
254            AND posy[i]=0 THEN Variables.FotP1a:=1; EXIT; ELSE Variables.FotP1a := 0; END_IF
255          END_FOR
256        (*b *)
257          FOR i := 0 TO b BY 1 DO
258          IF NOT inv[i] AND visElem[3] AND posx[i] >= P1bpFon-wb AND posx[i] < P1bpFoff
259            AND posy[i]=0 THEN Variables.FotP1b:=1; EXIT; ELSE Variables.FotP1b := 0; END_IF
260          END_FOR
261      (*Proceso2: *)
262        (*a *)
263          FOR i := 0 TO b BY 1 DO
264          IF NOT inv[i] AND visElem[2] AND posx[i] >= P2pF1on-wb AND posx[i] < P2pF1off
265            AND posy[i]=0 THEN Variables.Fot1P2a:=1; EXIT; ELSE Variables.Fot1P2a := 0; END_IF
266          END_FOR
267          FOR i := 0 TO b BY 1 DO
268          IF NOT inv[i] AND visElem[2] AND posx[i] >= P2pF2on-wb AND posx[i] < P2pF2off
269            AND posy[i]=0 THEN Variables.Fot2P2a := 1; EXIT; ELSE Variables.Fot2P2a:=0; END_IF
270          END_FOR
```

```
271         IF NOT inv[i] AND visElem[2] AND posVa > 0
272           THEN Variables.SInd1P2a := 1; ELSE Variables.SInd1P2a := 0; END_IF
273         IF NOT inv[i] AND visElem[2] AND posVa < -ddesc
274           THEN Variables.SInd2P2a := 1; ELSE Variables.SInd2P2a := 0; END_IF
275       (*b *)
276         FOR i := 0 TO b BY 1 DO
277           IF NOT inv[i] AND visElem[4] AND posx[i] >= P2bpF1on-wb AND posx[i] < P2bpF1off
278             AND posy[i]=0 THEN Variables.Fot1P2b:=1; EXIT; ELSE Variables.Fot1P2b:=0; END_IF
279         END_FOR
280         FOR i := 0 TO b BY 1 DO
281           IF NOT inv[i] AND visElem[4] AND posx[i] >= P2bpF2on-wb AND posx[i] < P2bpF2off-6
282             AND posy[i] = 0 THEN Variables.Fot2P2b:=1; EXIT; ELSE Variables.Fot2P2b:=0; END_IF
283         END_FOR
284         IF NOT inv[i] AND visElem[4] AND posVb > 0
285           THEN Variables.SInd1P2b := 1; ELSE Variables.SInd1P2b := 0; END_IF
286         IF NOT inv[i] AND visElem[4] AND posVb < -ddesc
287           THEN Variables.SInd2P2b := 1; ELSE Variables.SInd2P2b := 0; END_IF
288 END_IF
```

Código 23: Código del POU Sim (PRG)

ANEXO II: Código completo del control automático

Declaraciones del POU Variables

```
PROGRAM Variables
VAR_INPUT
  (*Compensers: *)
    (*Compenser A: *)
      (*Entradas: *)
        MarchaMCa: BOOL := 0;
        ParoMCa: BOOL := 0;
        CargaMCa: BOOL := 0; //Representando la orden de Carga Manual
        Fot1Ca: BOOL := 0;
        Fot2Ca: BOOL := 0;
        SIndCa: BOOL := 0;
        vCintaCa: REAL;
        vSubirCa: REAL;
      (*Salidas *)
        MCSubirCa: BOOL;
        MCBajarCa: BOOL;
        MCintaCa: BOOL;
        MCintaAntCa: BOOL;
        LlenoCa: BOOL;
        VacioCa: BOOL;
        ParadoCa: BOOL;
    (*Compenser B: *)
      (*Entradas: *)
        MarchaMCb: BOOL := 0;
        ParoMCb: BOOL := 0;
        CargaMCb: BOOL := 0; //Representando la orden de Carga Manual
        Fot1Cb: BOOL := 0;
        Fot2Cb: BOOL := 0;
        SIndCb: BOOL := 0;
        vCintaCb: REAL;
        vSubirCb: REAL;
```

```
32      (*Salidas *)
33        MCSubirCb: BOOL;
34        MCBajarCb: BOOL;
35        MCintaCb: BOOL;
36        MCintaAntCb: BOOL;
37        LlenoCb: BOOL;
38        VacioCb: BOOL;
39        ParadoCb: BOOL;
40
41  (*Proceso 1: *)
42    (* Proceso 1A: *)
43      (*Entradas: *)
44        MarchaMP1a: BOOL := 0;
45        ParoMP1a: BOOL := 0;
46        FotP1a: BOOL := 0;
47        vCintaP1a: REAL;
48      (*Salidas: *)
49        TratamientoP1a: BOOL := 0;
50        MCintaP1a: BOOL := 0;
51        MCintaAntP1a: BOOL;
52        ParadoP1a: BOOL;
53    (* Proceso 1B: *)
54      (*Entradas: *)
55        MarchaMP1b: BOOL := 0;
56        ParoMP1b: BOOL := 0;
57        FotP1b: BOOL := 0;
58        vCintaP1b: REAL;
59      (*Salidas: *)
60        TratamientoP1b: BOOL := 0;
61        MCintaP1b: BOOL := 0;
62        MCintaAntP1b: BOOL;
63        ParadoP1b: BOOL;
64
65  (*Proceso 2:*)
66    (*Proceso 2A: *)
67      (*Entradas: *)
```

```
68          MarchaMP2a : BOOL := 0;
69          ParoMP2a: BOOL := 0;
70          Fot1P2a: BOOL := 0;
71          Fot2P2a: BOOL := 0;
72          SInd1P2a: BOOL := 0;
73          SInd2P2a: BOOL := 0;
74          vCintaLP2a: REAL;
75          vCintaRP2a: REAL;
76          vEmboloP2a: REAL;
77        (*Salidas: *)
78          AscP2a: BOOL := 0;
79          DescP2a: BOOL := 0;
80          MCinta1RP2a: BOOL := 0;
81          MCinta1LP2a: BOOL := 0;
82          MCinta2RP2a: BOOL := 0;
83          MCinta2LP2a: BOOL := 0;
84          MCintaAntP2a: BOOL := 0;
85          ParadoP2a: BOOL;
86      (*Proceso 2B: *)
87        (*Entradas: *)
88          MarchaMP2b : BOOL := 0;
89          ParoMP2b: BOOL := 0;
90          Fot1P2b: BOOL := 0;
91          Fot2P2b: BOOL := 0;
92          SInd1P2b: BOOL := 0;
93          SInd2P2b: BOOL := 0;
94          vCintaLP2b: REAL;
95          vCintaRP2b: REAL;
96          vEmboloP2b: REAL;
97        (*Salidas: *)
98          AscP2b: BOOL := 0;
99          DescP2b: BOOL := 0;
100         MCinta1RP2b: BOOL := 0;
101         MCinta1LP2b: BOOL := 0;
102         MCinta2RP2b: BOOL := 0;
103         MCinta2LP2b: BOOL := 0;
```

```
104         MCintaAntP2b: BOOL := 0;
105         ParadoP2b: BOOL;
106 END_VAR
```

Código 24: Declaraciones del POU Variables.

Control del proceso compenser

```
1  PROGRAM Cmp1a
2  VAR_INPUT
3  (*Flancos y copias*)
4    FFot1: F_TRIG;
5    RFot2: R_TRIG;
6    RSInd: R_TRIG;
7    RMarcha: R_TRIG;
8    RParo: R_TRIG;
9    RMarchaM: R_TRIG;
10   RParoM: R_TRIG;
11 (*Etapas *)
12   (*Grafcet principal: *)
13     E0: BOOL := 1; E1: BOOL := 0; E2: BOOL := 0; E3: BOOL := 0; E4: BOOL := 0;
14     E5: BOOL := 0; E51: BOOL := 0; E6: BOOL := 0; E61: BOOL := 0; E7: BOOL := 0;
15     E8: BOOL := 0; E9: BOOL := 0; E10: BOOL := 0;
16   (*Grafcet Lleno/Vacio: *)
17     E20: BOOL := 1; E21: BOOL := 0; E22: BOOL := 0;
18 (*Variables del proceso *)
19   Paro: BOOL := 0;
20   Marcha: BOOL := 1;
21   Carga: BOOL;
22   (*Timmer Hueco*)
23     TON_Hueco: TON;
24     PT_t_Hueco: TIME; (*Tiempo desde bajada Fot1Ca para soltar pieza *)
25     t_Hueco: BOOL;
26   (*Timmer Bajada *)
27     TON_Bajada: TON;
28     PT_t_Bajada: TIME;
```

```
    t_Bajada: BOOL;
  (*Counter: *)
    CTUDPiezas: CTUD;
    PV_c_Piezas: WORD := 15;
  (*Otros *)
    str: STRING := ' ';
    dbug: BOOL := 1;
  (*Auxiliares: *)
    Inicializado: BOOL := 0;
END_VAR
```

Código 25: Declaraciones del control automático del compenser.

```
(*Inicializacion: *)
  IF NOT Inicializado THEN
    Inicializado := 1;
    PT_t_Bajada := REAL_TO_TIME((Sim.d-Sim.dbaj+1)/Variables.vSubirCa);
    PT_t_Hueco := REAL_TO_TIME((Sim.CpF2on-Sim.CpF1off+Sim.we-Sim.dbaj+1)
                  /Variables.vSubirCa);
  END_IF
(*Calculo de transiciones complejas: *)
  IF Sim.visElem[3] THEN Carga := NOT Variables.MCintaP1b OR Variables.CargaMCa;
  ELSIF Sim.visElem[4] THEN Carga := Proc2b.E0 OR Variables.CargaMCa;
  ELSIF Sim.visElem[0] THEN Carga := NOT Variables.MCintaCb OR Variables.CargaMCa;
  ELSE Carga := Variables.CargaMCa; END_IF
(*Deteccion de flancos*)
  RMarcha(CLK := Marcha);
  RMarchaM(CLK := Variables.MarchaMCa);
  RParo(CLK := Paro);
  RParoM(CLK := Variables.ParoMCa);
  FFot1(CLK := Variables.Fot1Ca);
  RFot2(CLK := Variables.Fot2Ca);
  RSInd(CLK := Variables.SIndCa);
(*Desactivacion y activacion de etapas *)
  (*Grafcet principal: *)
    IF E0 AND (RMarcha.Q OR RMarchaM.Q) THEN E0 := 0; E1 := 1; str := 'E1'; END_IF
    IF E1 AND Paro THEN E1 := 0; E0 := 1; str := 'E0'; END_IF
```

```
25      (*Carga: *)
26      IF E1 AND E31 THEN E1 := 0; E2 := 1; str := 'E2'; END_IF
27      IF E2 AND (Variables.LlenoCa OR Paro) THEN E2 := 0; E0 := 1; str := 'E0'; END_IF
28      IF E2 AND NOT Variables.LlenoCa AND NOT Paro THEN E2 := 0; E3 := 1; str := 'E3'; END_IF
29      IF E3 AND (Paro OR Variables.LlenoCa) THEN E3 := 0; E0 := 1; str := 'E0'; END_IF
30      IF E3 AND E30 THEN E3 := 0; E1 := 1; str := 'E1'; END_IF
31      IF E3 AND RFot2.Q THEN E3 := 0; E4 := 1; str := 'E4'; END_IF
32      IF E4 AND RSInd.Q THEN E4 := 0; E1 := 1; str := 'E1'; END_IF
33      (*Descarga: *)
34      IF E1 AND E30 AND (NOT Variables.VacioCa) THEN E1 := 0; E5 := 1; str := 'E5'; END_IF
35      IF E5 AND NOT Variables.Fot1Ca THEN E5 := 0; E6 := 1; str := 'E6'; END_IF
36      IF E6 AND Paro THEN E6 := 0; E0 := 1; str := 'E0'; END_IF
37      IF E6 AND (E31 OR Variables.VacioCa) THEN E6 := 0; E1 := 1; str := 'E1'; END_IF
38      IF E6 AND FFot1.Q THEN E6 := 0; E5 := 1; str := 'E5'; END_IF
39      IF E6 AND t_Hueco THEN E6 := 0; E7 := 1; str := 'E7'; END_IF
40      IF E7 AND Paro THEN E7 := 0; E0 := 1; str := 'E0'; END_IF
41      IF E7 AND (E31 OR Variables.VacioCa) THEN E7 := 0; E1 := 1; str := 'E1'; END_IF
42      IF E7 AND Variables.Fot1Ca THEN E7 := 0; E8 := 1; str := 'E8'; END_IF
43      IF E7 AND NOT Variables.Fot1Ca THEN E7 := 0; E9 := 1; str := 'E9'; END_IF
44      IF E8 AND RFot2.Q THEN E8 := 0; E5 := 1; str := 'E5'; END_IF
45      IF E9 AND RFot2.Q THEN E9 := 0; E10 := 1; str := 'E10'; END_IF
46      IF E10 AND Variables.Fot1Ca AND NOT Variables.Fot1Ca
47        THEN E10 := 0; E5 := 1; str := 'E5'; END_IF
48      IF E10 AND t_Bajada THEN E10 := 0; E7 := 1; str := 'E7'; END_IF
49    (*Grafcet Lleno/Vacio: *)
50      IF E20 AND CTUDPiezas.CV > 0 THEN E20 := 0; E21 := 1; END_IF
51      IF E21 AND CTUDPiezas.CV = CTUDPiezas.PV THEN E21 := 0; E22 := 1; END_IF
52      IF E21 AND CTUDPiezas.CV = 0 THEN E21 := 0; E20 := 1; END_IF
53      IF E22 AND CTUDPiezas.CV < CTUDPiezas.PV THEN E22 := 0; E21 := 1; END_IF
54  (*Contadores *)
55    CTUDPiezas(CU := E4, CD := E8 OR E9, RESET := FALSE, LOAD :=  FALSE, PV := PV_c_Piezas);
56  (*Activacion de salidas *)
57    Variables.MCSubirCa := E4;
58    Variables.MCBajarCa := E8 OR E9;
59    Variables.MCintaCa := E1 OR E2 OR E3 OR E4 OR E5 OR E6 OR E7 OR E8 OR E9 OR E10;
60    Variables.MCintaAntCa := E1 OR E2 OR E3 OR E4 OR E5 OR E6 OR E7 OR E8 OR E9 OR E10;
```

```
61    Variables.LlenoCa := E22;
62    Variables.VacioCa := E20;
63    Variables.ParadoCa := E0;
64  (*Temporizadores *)
65    TON_Hueco(IN := E6, PT := PT_t_Hueco);
66    TON_Bajada(IN := E10, PT := PT_t_Bajada);
67    (*Booleano dependiente *)
68      t_Hueco := TON_Hueco.Q;
69      t_Bajada := TON_Bajada.Q;
```

Código 26: Código del control automático del compenser.

Control del Proceso 1

```
1  PROGRAM Proc1a
2  VAR_INPUT
3    (*Variables del proceso:  *)
4      Paro: BOOL;
5      Marcha: BOOL;
6    (*Flancos y copias *)
7      RMarcha: R_TRIG;
8      RMarchaM: R_TRIG;
9      RParo: R_TRIG;
10     RFot: R_TRIG;
11     FFot: F_TRIG;
12   (*Etapas *)
13     E0: BOOL := 1; E1: BOOL := 0; E2: BOOL := 0; E3: BOOL := 0; E4: BOOL := 0;
14     E5: BOOL := 0; E6: BOOL := 0; E7: BOOL := 0; E8: BOOL := 0;
15   (*Timmers: *)
16     (*Timmer a1: *)
17       TON_dist1: TON;
18       PT_t_dist1: TIME; (*Tiempo hasta que se active el Tratamiento*)
19       t_dist1: BOOL;
20     (*Timmer b1: *)
21       TON_trat1: TON;
22       PT_t_trat1: TIME;
```

```
23      t_trat1: BOOL;
24    (*Timmer a2: *)
25      TON_dist2: TON;
26      t_dist2: BOOL;
27    (*Timmer b2: *)
28      TON_trat2: TON;
29      t_trat2: BOOL;
30  (*Auxiliares: *)
31    Inicializado: BOOL := 0;
32 END_VAR
```

Código 27: Declaraciones del control automático del Proceso 1.

```
1  (*Inicializacion: *)
2    IF NOT Inicializado THEN
3      Inicializado := 1; PT_t_trat1 := REAL_TO_TIME(Sim.wb/Variables.vCintaP1a);
4      PT_t_dist1 := REAL_TO_TIME((Sim.P1pCh-Sim.P1pFon)/Variables.vCintaP1a);
5    END_IF
6    Marcha := Sim.visElem[1];
7    IF Sim.visElem[1] THEN Variables.MarchaMP1a := Variables.MCintaCa
8      OR Variables.MarchaMP1a; END_IF
9  (*Comprobacion de Consentimiento: *)
10   Paro := Variables.ParoMP1a OR NOT Variables.MCintaCa;
11 (*Deteccion de flancos *)
12   RMarcha(CLK := Marcha);
13   RMarchaM(CLK := Variables.MarchaMP1a);
14   RParo(CLK := Paro);
15   RFot(CLK := Variables.FotP1a);
16   FFot(CLK := Variables.FotP1a);
17 (*Desactivacion y Activacion de etapas *)
18   IF E0 AND (RMarcha.Q OR RMarchaM.Q) THEN E0 := 0; E1 := 1; Variables.MarchaMP1a := 0;
19   END_IF
20   IF E1 AND (RFot.Q OR Paro) THEN E1 := 0; E2 := 1; E4 := 1; END_IF
21   IF E4 AND (t_dist1 OR Paro) THEN E4 := 0; E5 := 1; END_IF
22   IF E5 AND (t_trat1 OR Paro) THEN E5 := 0; END_IF
23   IF E2 AND Paro THEN E2 := 0; E0 := 1; END_IF
24   IF E2 AND FFot.Q THEN E2 := 0; E3 := 1; END_IF
```

```
25    IF E3 AND (RFot.Q OR Paro) THEN E3 := 0; E6 := 1; E8 := 1; END_IF
26    IF E6 AND (t_dist2 OR Paro) THEN E6 := 0; E7 := 1; END_IF
27    IF E7 AND (t_trat2 OR Paro) THEN E7 := 0; END_IF
28    IF E8 AND Paro THEN E8 := 0; E0 := 1; END_IF
29    IF E8 AND FFot.Q THEN E8 := 0; E1 := 1; END_IF
30  (*Activacion de salidas: *)
31    Variables.MCintaP1a := E1 OR E2 OR E3 OR E4 OR E5 OR E6 OR E7 OR E8;
32    Variables.MCintaAntP1a := E1 OR E2 OR E3 OR E4 OR E5 OR E6 OR E7 OR E8;
33    Variables.TratamientoP1a := E5 OR E7;
34    Variables.ParadoP1a := E0;
35  (*Temporizadores 1: *)
36    TON_dist1(IN := E4, PT := PT_t_dist1);
37    TON_trat1(IN := E5, PT := PT_t_trat1);
38    (*Booleano asociado: *)
39    t_dist1 := TON_dist1.Q;
40    t_trat1 := TON_trat1.Q;
41  (*Temporizadores 2: *)
42    TON_dist2(IN := E6, PT := PT_t_dist1);
43    TON_trat2(IN := E7, PT := PT_t_trat1);
44    (*Booleano asociado: *)
45    t_dist2 := TON_dist2.Q;
46    t_trat2 := TON_trat2.Q;
```

Código 28: Código del control automático del Proceso 1.

Control del Proceso 2

```
1  VAR_INPUT
2    (*Variables del Proceso: *)
3      Paro: BOOL;
4      Marcha: BOOL;
5    (*Flancos y copias: *)
6      RMarcha: R_TRIG;
7      RMarchaM: R_TRIG;
8      RParo: R_TRIG;
9      RFot1: R_TRIG;
```

```
    RFot2: R_TRIG;
    RSInd1: R_TRIG;
    RSInd2: R_TRIG;
  (*Etapas: *)
    E0: BOOL := 1; E1: BOOL := 0; E2: BOOL := 0; E3: BOOL := 0; E4: BOOL := 0;
    E00: BOOL := 1; E10: BOOL := 0; E20: BOOL := 0; E30: BOOL := 0; E40: BOOL := 0;
  (*Timmers: *)
    (*Timmer 1: *)
      TP_trat: TP;
      PT_t_trat: TIME;
      t_trat: BOOL;
  (*Otros *)
    str: STRING := 'E0';
    str2: STRING := 'E00';
  (*Auxiliares: *)
    Inicializado: BOOL := 0;
END_VAR
```

Código 29: Declaraciones del control automático del Proceso 2.

```
(*Inicializacion: *)
  IF NOT Inicializado THEN
    Inicializado := 1;
    PT_t_trat := REAL_TO_TIME( Sim.we/Variables.vCintaLP2a —
      — 2*Sim.ddesc/Variables.vEmboloP2a — Sim.d/Variables.vCintaRP2a);
  END_IF
  Marcha := Sim.visElem[2];
  IF Sim.visElem[2] THEN Variables.MarchaMP2a := Variables.MarchaMCa OR
    Variables.MarchaMP2a; END_IF
(*Comprobacion de Consentimiento: *)
  Paro := Variables.ParoMP2a OR NOT Variables.MCintaCa;
(*Deteccion de flancos: *)
  RMarcha(CLK := Marcha);
  RMarchaM(CLK := Variables.MarchaMP2a);
  RParo(CLK := Paro);
  RFot1(CLK := Variables.Fot1P2a);
  RFot2(CLK := Variables.Fot2P2a);
```

```
18   RSInd1(CLK := Variables.SInd1P2a);
19   RSInd2(CLK := Variables.SInd2P2a);
20 (*Desactivacion y Activacion de etapas: *)
21   IF E0 AND (RMarcha.Q OR RMarchaM.Q) THEN E0 := 0; E1 := 1;
22     Variables.MarchaMP2a := 0; str := 'E1'; END_IF
23   IF E1 AND Paro THEN E1 := 0; E0 := 1; str := 'E0'; END_IF
24   IF E1 AND RFot1.Q THEN E1 := 0; E2 := 1; str := 'E2'; END_IF
25   IF E2 AND Paro THEN E2 := 0; E0 := 1; str := 'E0'; END_IF
26   IF E2 AND RFot2.Q THEN E2 := 0; E1 := 1; str := 'E1'; END_IF
27   IF E00 AND RFot2.Q THEN E00 := 0; E10 := 1; str2 := 'E10'; END_IF
28   IF E10 AND RSInd2.Q THEN E10 := 0; E20 := 1;  str2 := 'E20';END_IF
29   IF E20 AND t_trat THEN E20 := 0; E30 :=1; str2 := '30'; END_IF
30   IF E30 AND RSInd1.Q THEN E30 := 0; E40 := 1; str2 := 'E40'; END_IF
31   IF E40 AND (RFot1.Q OR Paro) THEN E40 := 0; E00 := 1; str2 := 'E00'; END_IF
32 (*Activacion de salidas: *)
33   Variables.MCinta1LP2a := E1;
34   Variables.MCinta1rP2a := E2;
35   Variables.MCinta2RP2a := E2;
36   Variables.MCinta2LP2a := E30 OR E40;
37   Variables.MCintaAntP2a := E1 OR E2;
38   Variables.DescP2a := E10;
39   Variables.AscP2a := E30;
40   Variables.ParadoP2a := E0;
41 (*Temporizador: *)
42   TP_trat(IN := E20, PT := PT_t_trat);
43   (*Booleano asociado: *)
44   t_trat := TP_trat.Q;
```

Código 30: Código del control automático del Proceso 2.

ANEXO III: Propiedades de los elementos de visualización

Nombre	X (px)	Y (px)	Ancho (px)	Alto (px)
Compenser A	354	306	216	239
Proceso	16	6	200	190
Chásis	94	0	80	190
CintaCa	14	87	6	28
Fot1	14	87	28	5
Fot2	165	132	6	28
SInd	171	132	6	28
Texto	113	157	41	30
Flch.Up Ca	174	39	20	20
Flch.Dow Ca	174	62	20	20
Control	14	149	100	90
Contorno	9	0	112	124
MarchaM	74	4	57	39
ParoM	74	44	57	39
Txt. Marcha	0	6	82	30
Txt. Paro	0	46	82	30
CargaM	74	84	47	39
Txt. Carga	0	86	82	30
Panel	0	0	110	90
Contorno	2	0	105	90
Lamp. M	80	6	25	25
Lamp. P	80	6	25	25
Txt. M/P	0	4	84	28
Lamp. Vacío	80	33	25	25
Txt. Vacío	0	32	84	28
Lamp. Lleno	80	60	25	25
Txt. Lleno	0	61	84	28
Compenser B	922	307	216	238
Proceso	16	6	200	190
Chásis	94	0	80	190
CintaCa	14	87	6	28
Fot1	14	87	28	5
Fot2	165	132	6	28
SInd	171	132	6	28
Texto	113	157	41	30
Flch.Up Ca	174	39	20	20
Flch.Dow Ca	174	62	20	20
Control	14	149	100	90
Contorno	9	0	112	124
MarchaM	74	4	57	39
ParoM	74	44	57	39
Txt. Marcha	0	6	82	30
Txt. Paro	0	46	82	30
CargaM	74	84	47	39
Txt. Carga	0	86	82	30
Panel	0	0	110	90
Contorno	2	0	105	90
Lamp. M	80	6	25	25
Lamp. P	80	6	25	25
Txt. M/P	0	4	84	28
Lamp. Vacío	70	32	25	25
Txt. Vacío	0	32	84	28
Lamp. Lleno	70	61	25	25
Txt. Lleno	0	61	84	28

Tabla 24: Propiedades de los elementos de visualización correspondientes al proceso compenser a y b.

Nombre	X (px)	Y (px)	Ancho (px)	Alto (px)
Proceso 1A	106	361	160	178
Proceso	0	33	160	82
Chorro	110	28	3	13
Cinta	0	48	160	10
Base1	5	52	150	30
Base2	5	0	150	30
Fotocélula	30	5	6	28
Control	30	118	100	60
Contorno	7	0	85	60
Pulsador M	56	3	44	28
Interruptor P	56	31	44	28
Txt. Marcha	0	4	63	21
Txt. Paro	0	33	63	21
Panel	25	0	110	30
Contorno	2	0	120	35
Lamp. M	90	3	25	25
Lamp. P	90	3	25	25
Txt. M/P	0	3	96	30
Proceso 1B	674	361	160	178
Proceso	0	33	160	82
Chorro	110	28	3	13
Cinta	0	48	160	10
Base1	5	52	150	30
Base2	5	0	150	30
Fotocélula	30	5	6	28
Control	30	118	100	60
Contorno	7	0	85	60
Pulsador M	56	3	44	28
Interruptor P	56	31	44	28
Txt. Marcha	0	4	63	21
Txt. Paro	0	33	63	21
Panel	25	0	110	30
Contorno	2	0	120	35
Lamp. M	90	3	25	25
Lamp. P	90	3	25	25
Txt. M/P	0	3	96	30

Tabla 25: Propiedades de los elementos de visualización correspondientes al Proceso 1 a y b.

Nombre	X (px)	Y (px)	Ancho (px)	Alto (px)
Proceso 2A	106	361	160	184
Proceso	0	34	160	86
Base	5	52	150	32
Cinta1	96	48	64	10
Cinta2	0	48	98	10
Base2	5	0	150	30
Fotocélula 1	65	6	6	30
Fotocélula 2	151	44	4	30
S. Inductivo 1	143	2	17	5
S. Inductivo 2	143	19	17	5
Base émbolo	92	27	64	3
Vástago	132	4	4	23
Metal	132	9	17	4
Control	30	123	107	61
Contorno	7	0	85	60
Pulsador M	63	3	44	28
Interruptor P	62	31	44	28
Txt. Marcha	4	8	63	21
Txt. Paro	0	33	63	21
Panel	25	2	114	30
Contorno	0	0	110	30
Lamp. M	83	3	25	25
Lamp. P	83	3	25	25
Txt. M/P	0	1	85	30
Proceso 2B	674	361	160	180
Proceso	0	34	160	86
Base	5	52	150	32
Cinta1	96	48	64	10
Cinta2	0	48	98	10
Base2	5	0	150	30
Fotocélula 1	65	6	6	30
Fotocélula 2	151	44	4	30
S. Inductivo 1	143	2	17	5
S. Inductivo 2	143	19	17	5
Base émbolo	92	27	64	3
Vástago	132	4	4	23
Metal	132	9	17	4
Control	30	123	107	61
Contorno	7	0	85	60
Pulsador M	63	3	44	28
Interruptor P	62	31	44	28
Txt. Marcha	4	8	63	21
Txt. Paro	0	33	63	21
Panel	25	2	114	30
Contorno	0	0	110	30
Lamp. M/P	83	3	31	26
Txt. M/P	0	1	85	30

Tabla 26: Propiedades de los elementos de visualización correspondientes al Proceso 2 a y b.

PLIEGO DE CONDICIONES

El presente documento incluye condiciones exclusivamente técnicas, dada la índole del proyecto al que se refiere. Los requisitos mínimos están relacionados con el hardware y el software a usar: los primeros son definidos por el software empleado, y los segundos asociados a la versión de dicho software para asegurar un funcionamiento correcto.

Hardware	
Memoria en disco duro	500 MB
Memoria RAM	512 MB
Software	
Sistema operativo	Windows XP/7/8/10
Versión de CodeSys	CODESYS V3.5 SP10 Patch 4

Tabla 1: Requisitos mínimos de hardware y software.

PRESUPUESTO

El presupuesto de este proyecto es consecuencia exclusivamente de las horas de ingeniería, ya que no se ven involucrados de forma alguna elementos físicos dedicados, y el software base de este trabajo es CodeSys cuya licencia es gratuita a nivel de programador.

Trabajo ingeniería	**Coste por hora (€/h)**	**Horas (h)**	**Coste (€)**
Concepción gráfica	25	40	1000
Programación del simulador	50	150	7500
Programación del control	40	50	2000
Total			10500

Tabla 1: Desglose del coste relacionado con las actividades de ingeniería.