UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

PROYECTO DE AUTOMATIZACIÓN INDUSTRIAL DE UNA LÍNEA FLEXIBLE DE FABRICACIÓN CON RECONOCIMIENTO DE PIEZAS POR VISIÓN ARTIFICIAL

AUTOR: Ivan Gregori Jorques

TUTOR: Javier Sanchís Saez

COTUTOR: Raúl Simarro Fernández

Curso Académico: 2016-17

RESUMEN

El trabajo ha consistido en el diseño y la implementación de la automatización de varios procesos industriales pertenecientes a un sistema de procesamiento y transporte de piezas, utilizando autómatas de distintas compañías (Siemens S7-1200, Schneider M241 y Omron CJ2M) y un sistema de distinción de las piezas a introducir en la cadena por percepción de color; permitiendo herramientas suficientes para futuras ampliaciones o evoluciones del proyecto.

El sistema de procesamiento y transporte de piezas se ha implementado sobre tres maquetas FischerTechnik (Estación Multiproceso, Línea de Mecanizado, Robot Manipulador), dos que formarían parte de una cadena industrial, y la tercera teniendo una función de distribución de los productos en la línea según su posición. Al proceso principal se le ha sumado un control del horno de la Estación Multiproceso, mediante el autómata Siemens que lo controla. Para la detección del color de las piezas introducidas se ha utilizado una cámara Kinect.

Se ha conseguido cumplir con los objetivos, diseñando una interfaz de usuario a la altura de la industria, con un amplio control, distintos modos de uso y un sistema de alarmas. Finalmente, se ha analizado el coste del proyecto en el documento de Presupuesto.

Palabras Clave: automatización, línea, industrial, detección, interfaz, Siemens, Schneider, Omron, FischerTechnik, LabView, Kinect, OPC, servidor.

RESUM

El treball ha consistit en el disseny i la implementació de la automatització de diversos processos industrials pertanyents a un sistema de processament i transport de peces, utilitzant autòmats de distintes companyies (Siemens S7-1200, Schneider M241 i Omron CJ2M) i un sistema de distinció de les peces a introduir en la cadena per percepció de color; permetent ferramentes suficients per a futures ampliacions o evolucions del projecte.

El sistema de processament i transport de peces se ha implementat sobre tres maquetes FischerTechnik (Estació Multiprocés, Línia de Mecanitzat, Robot Manipulador), dos que formarien part de una cadena industrial, i la tercera tenint una funció de distribució dels productes en la línia segons la seva posició. Al procés principal se li ha sumat un control del forn de la Estació Multiprocés, mitjançant el autòmat Siemens que ho controla. Per a la detecció del color de les peces introduïdes se ha utilitzat una càmera Kinect.

Se ha aconseguit complir amb els objectius, dissenyant una interfície de usuari a la altura de la industria, amb un ampli control, diversos modes d'us i un sistema de alarmes. Finalment, se ha analitzat el cost del projecte en el document de pressupost.

Paraules Clau: automatització, línia, industrial, detecció, interfície, Siemens, Schneider, Omron, FischerTechnik, LabView, Kinect, OPC, servidor.

ABSTRACT

The work has consisted in the design and implementation of the automation of several industrial processes that belong in a processing and transportation system, using controllers from different companies (Siemens S7-1200, Schneider M2421 and Omron CJ2M) and a system that distinguishes pieces to introduce by the perception of their color; allowing enough tools for future extensions or developments of the project.

The processing and transportation system has been implemented on three FischerTechnik models (Multiprocessing station, Machining Line, Gripper Robot), two of them being part of the industrial chain, and the last one having a distribution function depending on the position. The main process has been extended with a control for the oven in the Multiprocessing station, through the Siemens controller. A Kinect camera has been used for detecting the color of the introduced pieces.

The project has achieved the objectives, designing a user interface at the height of the industry, with a wide control, different modes and an alarm system. Finally, the cost of the project has been analyzed in the quotation document.

Keywords: automation, line, industrial, detection, interface, Siemens, Schneider, Omron, FischerTechnik, LabView, Kinect, OPC, server.

DOCUMENTOS

- MEMORIA
- PRESUPUESTO

ÍNDICE de la MEMORIA

ÍNDICE de PRESUPUESTO

MEMORIA

Ivan Gregori Jorques

1. OBJETO DEL PROYECTO

Se ha establecido como objeto en este Trabajo de Final de Grado el diseño y la implementación de la automatización de varios procesos industriales pertenecientes a un sistema de procesamiento y transporte de piezas mediante tres autómatas (Siemens S7-1200, Omron CJ2, Schneider M241) que controlarán distintas secciones del proceso industrial conjunto.

Esta línea cuenta con un proceso de tratamiento térmico y posteriormente 3 procesos físicos de fresado y taladrado. Se ha considerado y llevado a cabo la necesidad de una capacidad automática de cambios en las variables del proceso, concluyendo en un proceso industrial flexible y adaptable al tipo de pieza, y para ello se ha utilizado un sistema de detección basado en el color. También se ha programado un sistema de control PLC integrado en el autómata Siemens que ayuda a llevar el proceso térmico a la temperatura deseada según sea solicitado.

El proceso se ha simulado mediante 3 maquetas industriales FischerTechnik y una célula Peltier que simulará el control de temperatura.

Se ha pretendido una programación que unifique y asemeje los distintos mecanismos y herramientas que ofrecen los distintos autómatas, y facilitar así la comprensión del proyecto, enfocado en gran medida a las modificaciones de los programas que se puedan realizar en el futuro.

Al ser un proceso dividido entre los 3 autómatas, es necesaria una comunicación global entre estos y el operario, donde se ha utilizado un servidor OPC en la red Ethernet con este fin, y un sistema SCADA conectado también a este, que permite al operario la visualización de los distintos parámetros y modificar otros conforme a sus necesidades, de una forma sencilla, clara y moderna.

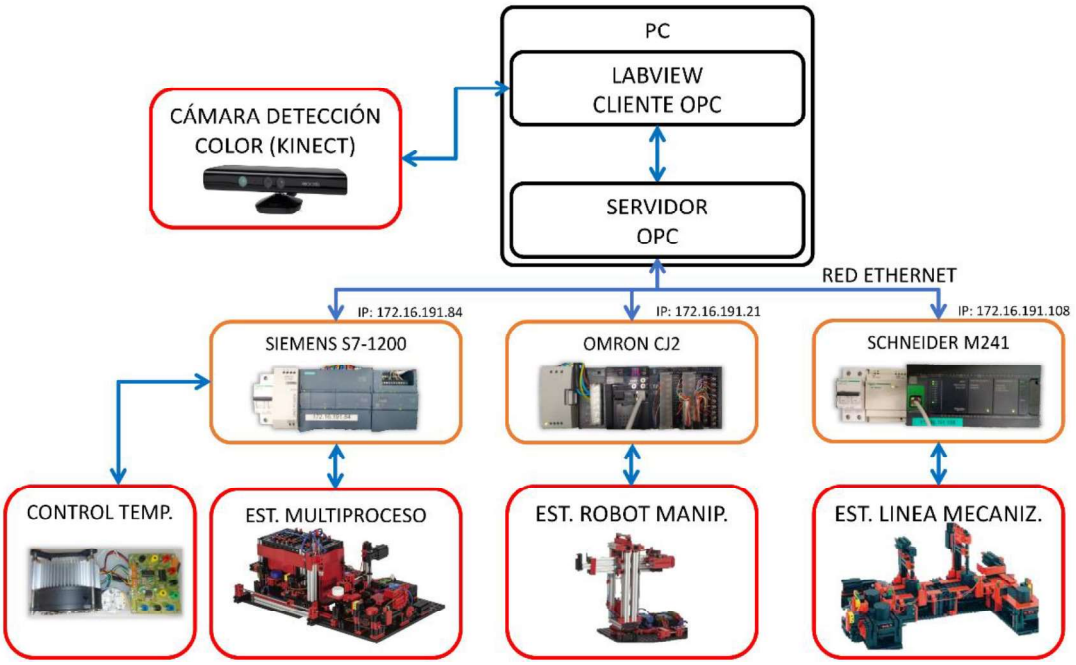


Figura 1. Esquema general del proyecto

2. FUNCIONAMIENTO DESEADO DEL SISTEMA

A continuación, se va a explicar el funcionamiento que se ha contemplado para el sistema. Cabe indicar que este apartado trata el comportamiento a nivel superficial de cada sección. Si se está buscando información más precisa sobre el funcionamiento de los programas, se encontrará en 3. Desarrollo de la solución.

Se trata de la programación de una línea con una fluencia de piezas relativamente pequeña, puesto que se incluyen procesos de carácter relativamente lento por los que sólo pasará una pieza a la vez. El proceso por lo tanto queda marcado por la linealidad, exceptuando quizás el caso de la *Estación Robot Manipulador*, que funciona como un brazo sustitutivo de un operario.

En este punto se encuentra el sistema explicado por secciones, siendo muy recomendable leer la primera sección, aunque se esté buscando información de otra sección, ya que esta trata el funcionamiento común que comparten las Estaciones. Las secciones quedan por lo tanto divididas de esta forma:

2.1. FUNCIONAMIENTO COMÚN EN LAS ESTACIONES Y LA INTERFAZ

Todas las estaciones comparten características importantes para una línea industrial. Estos son:

- **Modo Emergencia**: cuando se desee se puede activar este modo, que paralizará todo movimiento. Se habilitan también en este modo botones para activar algunas acciones de forma individual, para corregir posibles atascos que se hayan producido, comprobar roturas, etc.
- **Modo Automático/Paso A Paso**: se puede alternar entre ambos modos, siendo el automático el modo habitual. El Modo Paso A Paso permite comprobar la evolución del sistema y sirve también para detectar posibles problemas. Este último se ha programado de forma inteligente para evitar paradas innecesarias o que puedan causar pérdidas de calidad en el producto final.
- **Botón de Inicio**: Los programas están diseñados para que no arranquen el proceso nada más ser conectados, sino que precisen de la activación de la señal de inicio. Esto también funciona así para la salida del Modo Emergencia.
- **Inicialización**: Los programas están diseñados para que, tras el inicio, se coloquen primero en una posición inicial conocida, y a partir de ahí se desarrolle el programa con normalidad. Lo mismo ocurre tras el modo de Emergencia y el reinicio posterior. Esto se hace porque al arrancar un programa es imposible tener certeza de la posición de todos los elementos involucrados, y se les fuerza a colocarse en una posición concreta que se puede conocer para iniciar el desarrollo habitual del sistema.

2.2. CÁMARA (DETECCIÓN DE LA PIEZA)

Figura 2. Cámara y zona de grabación

El primer paso en la cadena consiste en la detección de la pieza para determinar el valor de las variables en el proceso según la pieza que se esté procesando, como podría ser temperatura de horno, o tiempo en fresado.

Se ha de tener en gran consideración que el sistema debe ser conocedor de qué variable utilizar en cada momento con respecto a la pieza que están procesando. Es decir, aunque en el sistema acabe de entrar una pieza con un color "A", y se registren las variables con las que debe ser procesada, la pieza de color "B" que ya está siendo tratada debe ser procesada con las variables correspondientes a esta, y no con las de la pieza de color "A".

En este proyecto se hará distinción entre 3 tipos de piezas, etiquetadas con los colores rojo, blanco y azul, mediante una cámara en la que se debe seleccionar en la interfaz de usuario la zona de la imagen de detección de color (donde habrá que colocar la pieza).

Figura 3. Piezas a procesar en la simulación por maquetas

Las variables implicadas en todo el proceso serán completamente modificables por el operario. Cuáles son y dónde se pueden observar y modificar se irá viendo en los siguientes apartados.

2.3. ESTACIÓN MULTIPROCESO

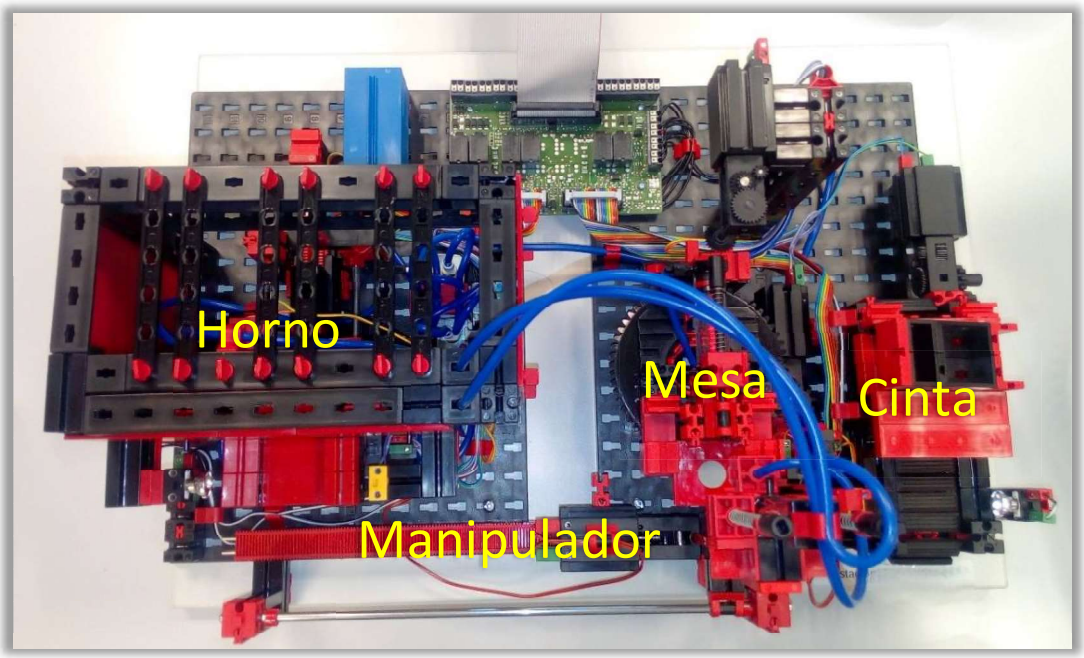


Figura 4. Maqueta de la Estación Multiproceso

2.3.1. Horno

Aunque la nomenclatura más adecuada sería *cámara térmica*, para indicar que se trata de una zona de aislamiento térmico, es necesario indicar que se ha llamado *Horno* a esta subsección del proceso debido a la similitud con la estructura industrial de un horno. En este proyecto particular, la cámara térmica ("horno" a partir de ahora) se ha utilizado como refrigerador.

La *Estación Multiproceso* debe recibir una pieza, que habrá pasado ya por la *Cámara* y se habrá registrado su color. Para ello necesita inicializarse sacando la plataforma hacia afuera (si no estaba ya en posición, y por supuesto abriendo la puerta en caso de no ser así, para evitar colisiones indeseadas). La pieza viene traída por el *Robot Manipulador*, y los fotorreceptores de la plataforma son los encargados de determinar que ya se ha realizado la de posición de la pieza.

La puerta se abre y se introduce la plataforma, siempre y cuando la temperatura dentro del *Horno* sea la adecuada según la pieza a introducir. Se avisa al *Manipulador* de que ya puede desplazarse hacia el horno para recoger la pieza, aunque esta no haya salido aún del horno. El control de la temperatura se realiza mediante un **control PI**. En el apartado 3.3.7. Control analógico de la temperatura, queda detallada la configuración del PID en el autómata Siemens.

Cuando ha pasado el tiempo necesario en el horno, este abrirá su puerta y la plataforma saldrá hacia afuera. Esperará a que el *Manipulador* haya llegado a la plataforma para llevarse la pieza (es posible que el *Manipulador* ya se encontrara allí, como se explica en el siguiente apartado. Tras ello, avisará a la *Estación Robot Manipulador*, indicándole que ya puede desplazarse para dejar una pieza sobre la plataforma, en el caso de que la tenga. Y así, el cido del *Horno* se ha completado.

2.3.2. Manipulador

Figura 5. Manipulador de la Estación Multiproceso

La posición de inicialización del *Manipulador* es en la posición de la *Mesa*, ya que así existe seguridad de que la *Estación Robot Manipulador* no va a producir colisión con el *Manipulador*.

Ahí espera a que el *Horno* le permita desplazarse para recoger la pieza que ha sido tratada térmicamente. En ese punto espera a que el horno haya sacado la plataforma (al no ser que la haya sacado ya). Entonces, baja la ventosa, activa la succión, vuelve a subirla, y se lleva la pieza hacia la *Mesa*.

En este punto, espera a que la *Mesa* esté en posición de recibir la pieza, y así poder bajar la ventosa, desactivar la succión para dejar la pieza, y subir la ventosa para esperar de nuevo la llamada del *Horno*. Con esto queda el ciclo del *Manipulador* cerrado.

2.3.3. Mesa

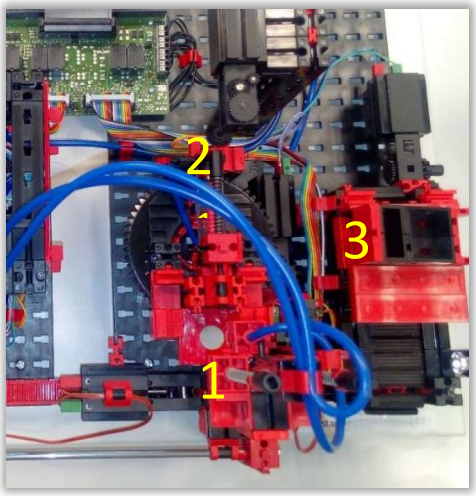


Figura 6. Mesa en la Estación Multiproceso

La *Mesa* tiene 3 posiciones a las que ir, marcadas con finales de carrera. En el inicio, ha de situarse en la primera de ellas, que corresponde a la posición donde recibirá la pieza por parte del *Manipulador*. Cuando este le indique que ya ha dejado la pieza, la *Mesa* girará hasta la segunda posición, donde se someterá a un procesado inicial durante unos segundos determinados por el usuario (y por el tipo de pieza que se esté procesando en el instante).

Finalmente, la *Mesa* continuará su giro horario hasta la tercera posición, donde, si la *Cinta* le comunica que el principio de esta está libre, podrá activar un pistón neumático que empujará la pieza hasta esta. En ese momento, la variable que indicaba que el principio de la *Cinta* está libre, se desactiva, y automáticamente se desplazará de nuevo a la posición inicial, cumpliendo así el ciclo.

2.3.4. Cinta

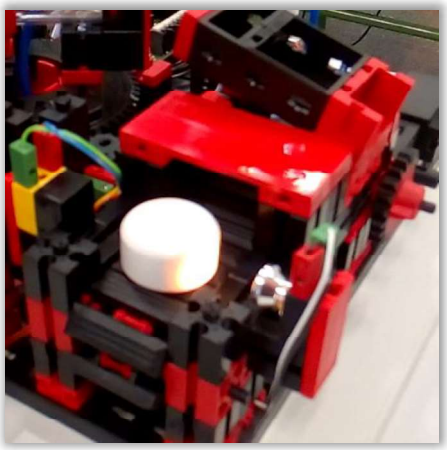


Figura 7. Cinta en la Estación Multiproceso

Esta subsección arranca activando la variable que permite a la *Mesa* depositar una pieza en su principio. Como se ha explicado en el apartado anterior, cuando la *Mesa* deposita la pieza, la variable se desactiva para evitar que la *Mesa* pueda echar una nueva pieza sin que la *Cinta* esté lista para ello.

Cuando recibe una pieza, siempre y cuando no exista una pieza en su parte final (posición de recogida), arrancará el motor de la cinta para llevar la pieza a la posición de recogida, y volverá a activar la variable que permite a la *Mesa* depositar una nueva pieza. El ciclo queda así cerrado.

Cabe remarcar que gracias a este mecanismo la *Cinta* consigue así mantener 2 piezas en su subsistema, a diferencia de las subsecciones anteriores citadas. Esto hace que el sistema global pueda soportar más piezas dentro de él.

2.3.5. Variables de control

La sección de la *Estación Multiproceso* tiene varias variables que podrán ser ajustadas por el operario para cada uno de los 3 tipos de piezas:

- **Tiempo de procesado térmico**: controla cuánto tiempo debe de pasar la pieza en el proceso térmico.
- **Tiempo de procesado**: controla cuánto tiempo debe pasar la pieza en el procesado situado en la *Mesa*.
- **Temperatura de proceso térmico**: controla los grados en los que debe situarse la cámara térmica del *Horno*.

Cómo ajustar estas variables quedará determinado en 2.6. Interfaz visual para el usuario.

2.4. ESTACIÓN LÍNEA MECANIZADO

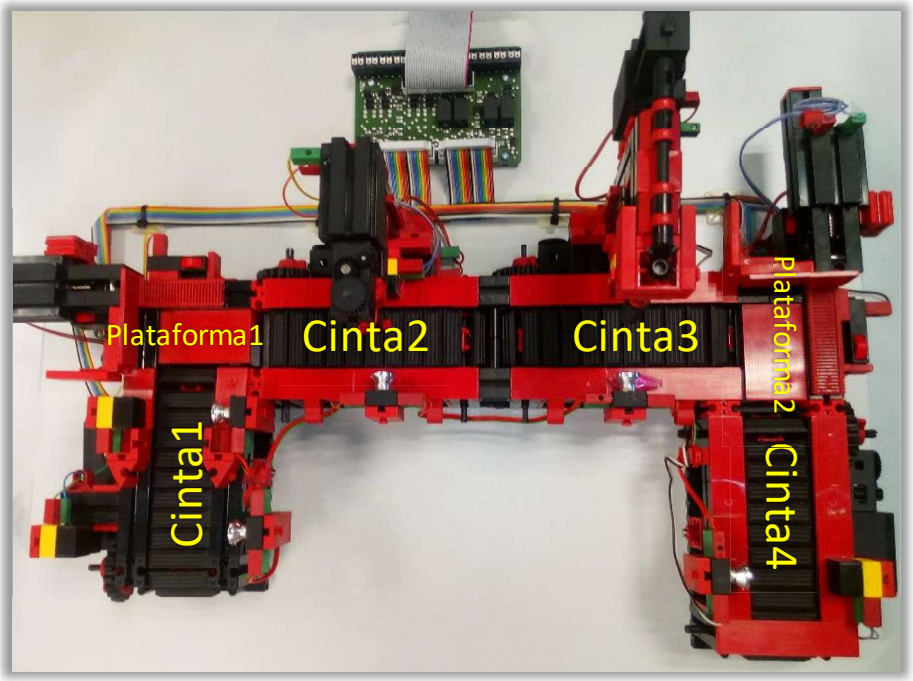


Figura 8. Maqueta de la Estación Línea Mecanizado

Esta estación se trata de una línea simple de cintas, con dos plataformas que redirigen el flujo de piezas, y un par de zonas de procesado en la *Cinta 2* y en la *Cinta 3*. Es muy importante que esta sección sea capaz de contener el mayor número de piezas posibles y que sea suficientemente robusto ante errores por acumulación.

Es necesario señalar, para evitar cualquier posible confusión, que no existe relación alguna entre las cintas aquí mencionadas y la *Cinta*, subsección de la *Estación Multiproceso*.

2.4.1. Cinta 1

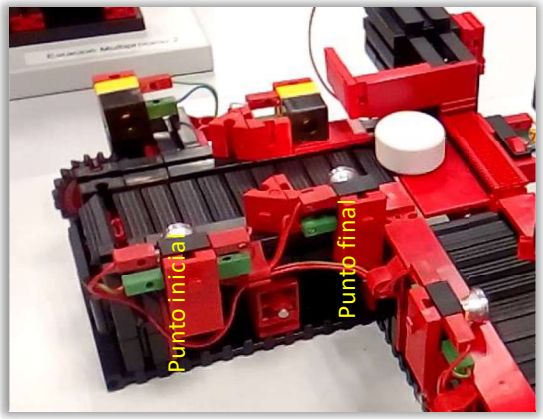


Figura 9. Cinta 1 en la Estación Línea Mecanizado

En primer lugar, tenemos la *Cinta 1*, que hace función de recepción de piezas. Recibirá las piezas que ya han sido procesadas por la *Estación Multiproceso* mediante la *Estación Robot Manipulador*, que dejará la pieza sobre el principio de la *Cinta 1* en cuanto esta lo permita (cuando no exista una pieza en ese punto o la cinta no se esté moviendo).

En cuanto reciba una pieza sobre ella, esperará un pequeño tiempo para permitir al Robot Manipulador terminar de dejar la pieza sin problemas, y comprobará si la posición final de la cinta está libre para activar el motor de la cinta y llevar la pieza hasta allí. En caso de existir una pieza en la posición final, la *Cinta 1* deberá esperar a que la *Plataforma 1* esté en posición recogida y no tenga ninguna pieza sobre ella, para poder ser cargada.

En cuanto la *Cinta 1* pueda cargar esta pieza en la *Plataforma 1*, deberá primero ser consciente de si existe una pieza en su punto inicial, puesto que al arrancar el motor para dejar la pieza en la posición final sobre la *Plataforma 1*, la pieza del punto inicial se desplazará también hacia la posición final, y debe controlar tanto que la pieza a cargar llegue a la *Plataforma 1* como que la posible pieza que iba detrás se pare precisamente en el punto final, ya que podría resultar grave que se quedara a mitad de camino y que no sea posible determinar si existe una pieza en la zona, o que se pasara del punto final y provocara un fallo al intentar también cargarse en la *Plataforma 1*.

2.4.2. Plataforma 1

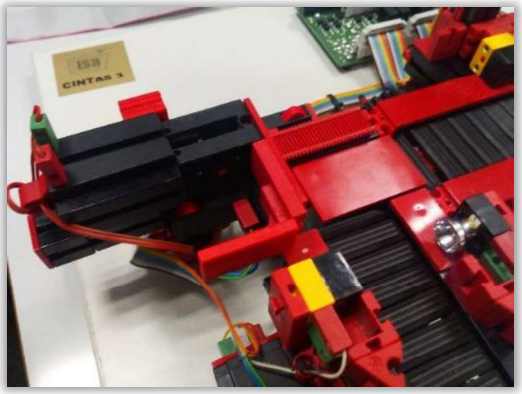


Figura 10. Plataforma 1 en la Estación Línea Mecanizado

Esta subsección tiene como posición de inicio la de recogida, permitiendo la carga de una pieza desde la *Cinta 1*. Cuando se haya producido esto, la cinta avisará a la plataforma de que ya está cargada, puesto que la *Plataforma 1* no tiene ninguna forma de conocerlo (como un sensor de peso, o un final de carrera) salvo esa.

La plataforma cargada esperará a que la *Cinta 2* esté libre para poder empujar la pieza hacia esta. Automáticamente, la plataforma se recogerá sobre sí misma, habilitando de nuevo a la *Cinta 1* a que pueda cargar una pieza en caso de que exista una, cerrando su ciclo.

2.4.3. Cinta 2

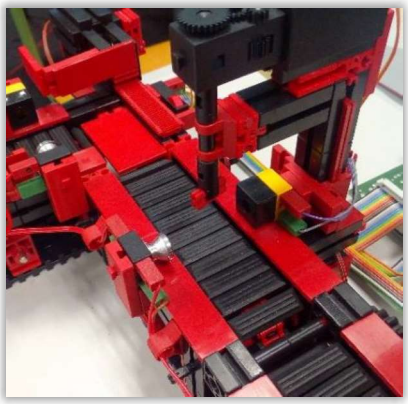


Figura 11. Cinta 2 en la Estación Línea Mecanizado

La *Cinta 2* contiene un punto de procesado de la pieza (simulado con un fresado). Esta cinta esperará a que la *Plataforma 2* se descargue para arrancar el motor de la cinta hasta que la pieza alcance el punto de fresado. Aquí, la pieza será sometida al tratamiento durante un tiempo determinado.

Después de ello, esperará a que la *Cinta 3* haya dejado de procesar una pieza (o simplemente no tuviera ninguna) para arrancar de nuevo el motor durante unos segundos, asegurando que la pieza llegue a la zona de la *Cinta 3*, y posteriormente volviendo a avisar de que está libre por si la *Plataforma 2* tiene una pieza que dejarle.

2.4.4. Cinta 3

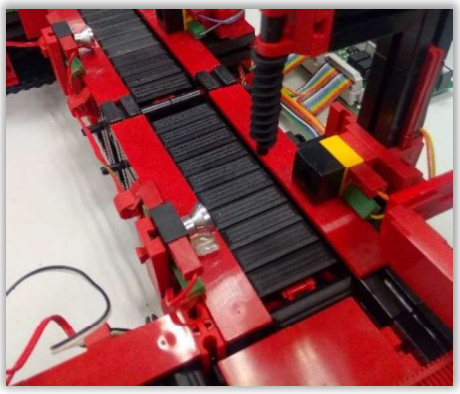


Figura 12. Cinta 3 en la Estación Línea Mcanizado

De forma similar a la *Cinta 2*, su punto de inicio es la espera de que la *Cinta 2* le envíe una pieza. En cuyo caso, arranca el motor de la cinta para favorecer la transición entre una cinta y la otra, y apagará este cuando detecte que la pieza haya llegado a su punto de procesado (simulado con un taladrado).

La pieza de nuevo es sometida a un tratamiento durante un tiempo determinado y espera a que la *Plataforma 2* esté recogida y no tenga ninguna pieza cargada. En ese caso, la *Cinta 3* arranca el motor de su cinta, permitiendo por un lado cargar la pieza en la *Plataforma 2*, y por otro lado, si se da el caso de que la *Cinta 2* ya tiene una pieza lista para enviar a la *Cinta 3*, avisa a esta para que pueda iniciar el transporte.

2.4.5. Plataforma 2

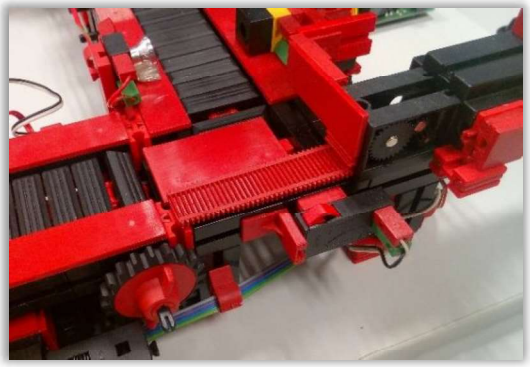


Figura 13. Plataforma 2 en la Estación Línea Mecanizado

Su funcionamiento es prácticamente idéntico al de la *Plataforma 1*. En su posición inicial de recogida y descargada, le dice a la *Cinta 3* que puede cargar una pieza en esta.

Cuando la pieza se ha cargado, y con la aceptación de la *Cinta 4* la plataforma empuja la pieza hacia esta cinta y automáticamente regresa a su posición inicial, cerrando su ciclo.

2.4.6. Cinta 4

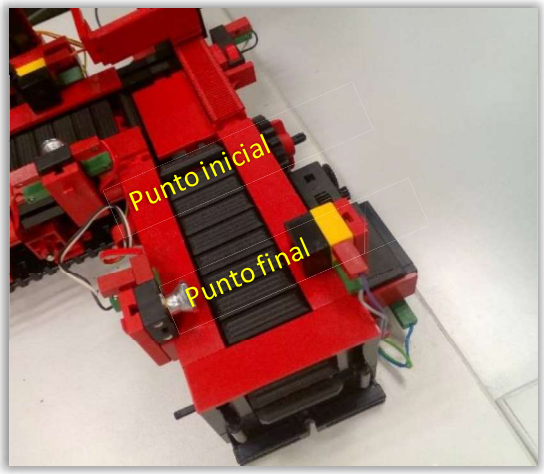


Figura 14. Plataforma 2 en la Estación Línea Mecanizado

Esta es la última parte de la cadena industrial programada. La *Cinta 4* va a ponerse a la espera de que la *Plataforma 2* le descargue una pieza en su punto inicial. Como ocurre con la *Cinta* en la *Estación Multiproceso*, cuando la *Plataforma 2* le deja la pieza, la cinta bloquea a la *plataforma* para que no pueda dejar más piezas puesto que su posición inicial está ahora ocupada.

Si el punto final de la cinta se encuentra libre, entonces arranca el motor de la cinta para llevar la pieza descargada de la *Plataforma 2* hacia el punto final o de extracción, donde el operario ya dispondrá de la pieza procesada. Cuando la pieza haya llegado a este punto, es cuando se permite de nuevo a la *Plataforma 2* descargar de nuevo. Y así se cierra el ciclo de la *Cinta 4*.

2.4.7. Variables de control

La sección de la *Estación Línea Mecanizado*, al igual que la *Estación Multiproceso*, tiene la capacidad de variar algunas variables en sus procesos:

- **Tiempo de fresado**: controla cuánto tiempo debe pasar la pieza en el procesado situado en la *Cinta 2*.
- **Tiempo de taladrado**: controla cuánto tiempo debe pasar la pieza en el procesado situado en la *Cinta 3*.

Cómo ajustar estas variables quedará determinado en 2.6. Interfaz visual para el usuario.

2.5. ESTACIÓN ROBOT MANIPULADOR

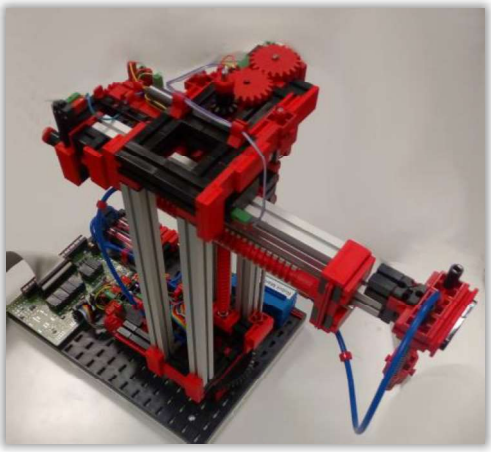


Figura 15. Maqueta de la Estación Robot Manipulador

Esta estación tiene un comportamiento diferente al de las otras. Su función es la de sustitución de un operario a la hora de desplazar las piezas de un sitio a otro. El uso de un brazo robótico tiene su justificación en la seguridad, ya que en un entorno industrial donde no sólo existe un riesgo ya intrínseco, sino que, además, se trabaja a temperaturas distintas a las que un ser humano puede llegar a soportar.

A diferencia de la *Estación Multiproceso* y la *Estación Línea Mecanizado*, no cuenta con un proceso lineal… sino que va a realizar dos tareas definidas, según ciertas condiciones. También a diferencia de las otras dos estaciones, la inicialización de esta estación se realiza principalmente para calibrar correctamente las coordenadas de su movimiento. Esto queda explicado con más detalle en 3.5. Diseño e implementación de los automatismos necesarios para la Estación Robot Manipulador. Como las tareas consisten en llevar piezas de un sitio a otro, las vamos a denominar *Camino 1 y Camino 2*, por orden de prioridad.

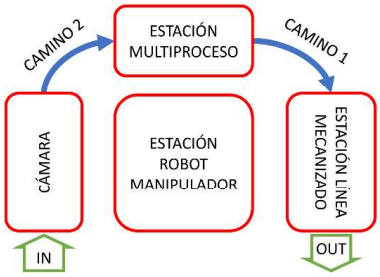


Figura 16. Diagrama de posición de las Estaciones y los caminos

2.5.1. Camino 1

Se ha llamado *Camino 1* al transporte de la pieza que se encuentra en el final de la *Estación Multiproceso* (final de la *Cinta*), que necesita ser llevada al principio de la *Estación Línea Mecanizado* (punto inicial de *Cinta 1*). Se ha escogido este camino como el prioritario, por la necesidad de terminar la producción y evitar estancamientos más rápidos.

Este camino se va a realizar cuando el *Robot Manipulador* no esté realizando ninguna tarea y exista una pieza al final de la *Estación Multiproceso*. Habiéndose inspirado en movimientos seguros que evitan posibles choques (manteniéndose en la posición más superior en todos los giros realizados), inicia su movimiento hasta la primera posición, el final de la *Cinta* de la *Estación Multiproceso*.

Allí el brazo baja para recoger la pieza, activa la succión con la ventosa, y regresa a la posición superior (la *Cinta* detecta automáticamente que ya no existe pieza al final) para dirigirse a la segunda posición, el punto inicial en la *Cinta 1* de la *Estación Línea Mecanizado*.

En esta segunda posición, el *Robot Manipulador* no bajará hasta que la *Cinta 1* lo permita (cuando no haya pieza ya en su punto inicial o no se esté moviendo la propia cinta). En cuanto pueda dejar la pieza, bajará y la dejará (desactivando la succión) de forma rápida para evitar retrasar el proceso.

Se le asigne o no al brazo un nuevo camino, el brazo ha de regresar primero a la posición inicial para evitar los errores que se han comentado antes.

2.5.2. Camino 2

Cómo se realizan los movimientos en este caso es prácticamente idéntico al *Camino 1*, salvo por algunas peculiaridades. En este caso, los dos puntos a los que viajará serán primero la *Cámara*, donde cogerá la pieza que habrá sido identificada, y la dejará en el principio de la *Estación Multiproceso*.

En este caso cabe destacar que el *Robot Manipulador* iniciará este camino siempre y cuando se detecte señal de existencia de pieza en la *Cámara* y no se detecte que existe una pieza al final de la *Estación Multiproceso*, ya que el *Camino 2* no es el prioritario.

Como peculiaridad de este camino, es importante destacar que el *Robot Manipulador*, después de recoger la pieza y subir a la posición superior, no girará hasta la segunda posición (como sí hacía en el *Camino 1*) hasta que la *Estación Multiproceso* se lo permita. Esto es así debido a que, a diferencia del *Camino 1*, la *Estación Multiproceso* tiene el elemento *Manipulador* interno que puede interferir en el camino del *Robot Manipulador*.

2.6. INTERFAZ VISUAL PARA EL USUARIO

La interfaz de usuario se ha diseñado para el uso y monitorización sencillo y práctico para el usuario. Para ello se han establecido una serie de pestañas, que evitan que exista demasiada información en una misma pantalla, mostrando un aspecto mucho más sencillo y organizado.

2.6.1. Controles e indicadores generales

El usuario puede ver como en la pantalla existirán controles y visualizaciones generales del *Modo Automático/Modo Paso A Paso*, señalización del *Robot Manipulador* según su situación, con un aviso visual de su aproximación a la zona de contacto humano más inminente, la *Cámara*, y debajo un indicador en forma de mensaje de cualquier alarma y situación excepcional que pueda estar sufriendo. También tiene tres indicadores visuales de alarma para las 3 Estaciones (existen también indicadores en forma de mensaje para las otras 2, como en el caso del *Robot Manipulador*, que se encuentran en sus respectivas pestañas, explicadas en el siguiente apartado), así como un botón de Emergencia.

Figura 17. Interfaz de usuario, pantalla Principal

2.6.2. Modo normal

En el *Modo Normal* (indicado con fondo azul) se puede gestionar, como bien dice el nombre, los aspectos rutinarios del funcionamiento del sistema completo.

En la pantalla ***Principal***, tenemos quizás los aspectos más importantes a la hora de monitorizar un proceso. Esto es el número de piezas introducidas (separando por su tipología) y un control de la zona probablemente más peligrosa, el tratamiento térmico, puesto que, aparte de poder ocasionar daños irreparables, puede suponer un gran coste en el consumo energético. Finalmente, también disponemos de indicadores que nos permiten saber si el sistema permite introducir una nueva pieza, o si existe una pieza que esté disponible para la recogida.

Esta pantalla también recoge el botón quizás más importante, *Inicio*, que arranca los sistemas. Con este botón se evita que al regresar de un *Modo Emergencia* o al encenderse los sistemas, el proceso pueda arrancar de inmediato (por ejemplo, con una inicialización del *Manipulador*) pudiendo provocar errores fatales o incluso daños humanos.

La pantalla ***Configuración*** es la respuesta a cómo se realiza la gestión de las distintas piezas. Aquí podemos ver directamente qué imágenes captura la *Cámara*, permitiéndonos configurar qué zona queremos que se utilice para la determinación del color, y a la derecha tenemos todos los parámetros a configurar para cada tipo de pieza.

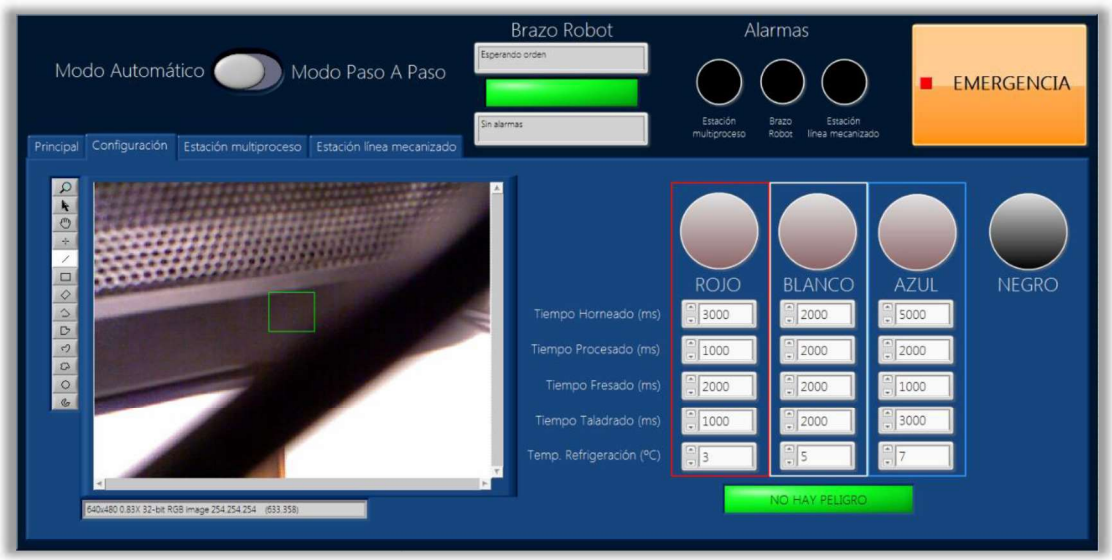


Figura 18. Interfaz de usuario, pantalla Configuración

En la pantalla ***Estación Multiproceso***, encontramos un control orientado particularmente a esta Estación. Encontramos información específica sobre cuántas piezas existen dentro de ésta, así como el mismo control de temperatura visto en la pantalla *Principal*. También encontramos unos indicadores de posición de piezas sobre una imagen real de la estación, para conocer aproximadamente la situación de estas. Por último, indicar el recuadro de Alarma que informará sobre qué tipo de anomalía está ocurriendo.

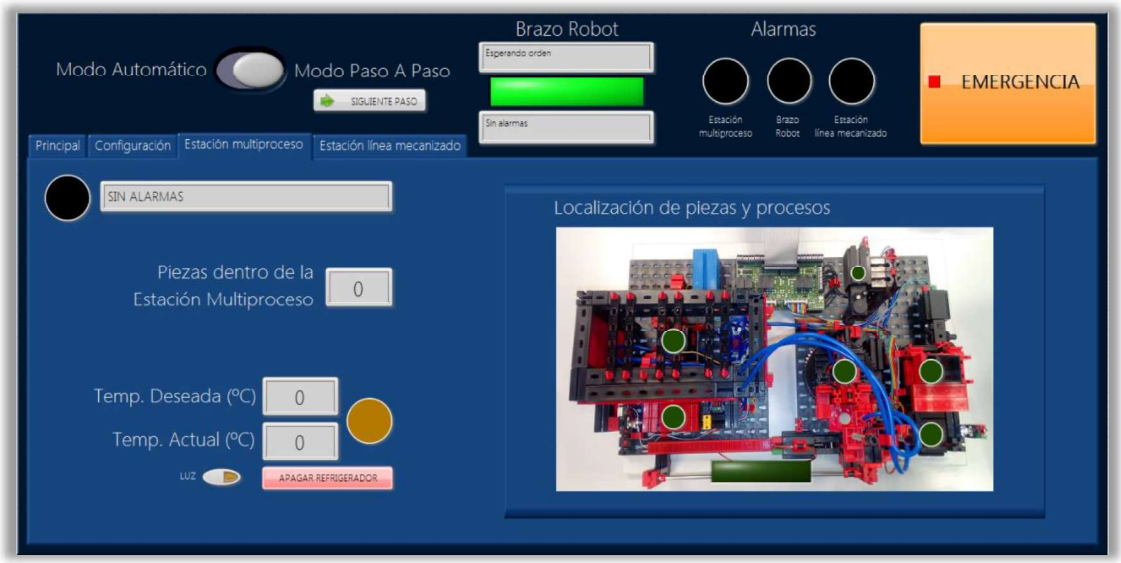


Figura 19. Interfaz de usuario, pantalla Estación Multiproceso

La pantalla ***Estación Línea Mecanizado*** es semejante a la anterior, aunque con menos complejidad debido a su composición. En este caso no incluye un control en la temperatura,

pero sí un indicador de que existe una pieza que ya puede ser recogida, además de todo lo anterior mencionado para la *Estación Multiproceso*.

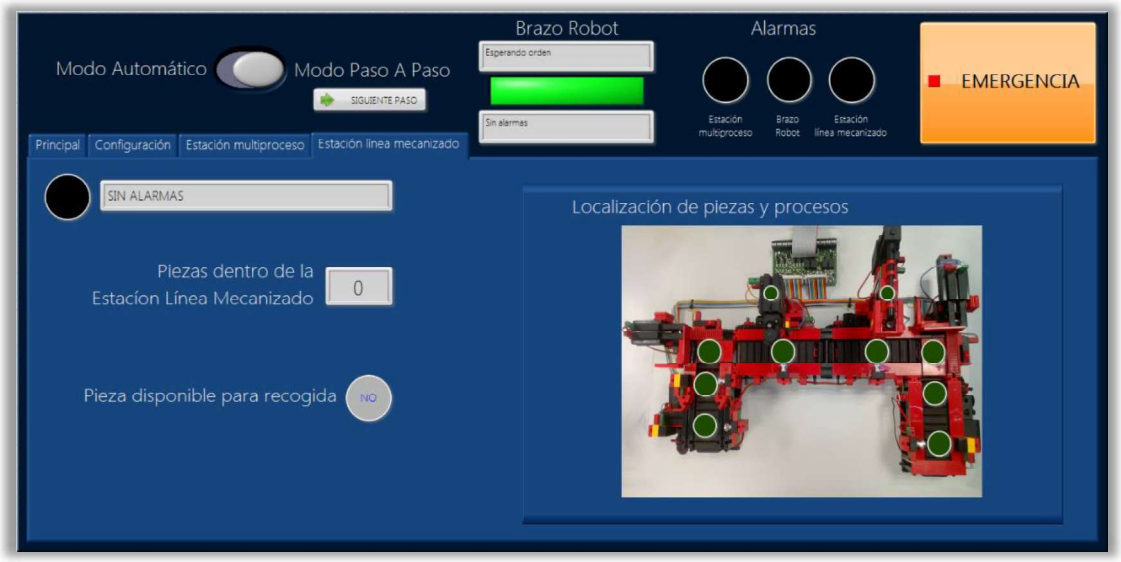


Figura 20. Interfaz de usuario, pantalla Estación Línea Mecanizado

2.6.3. Modo Emergencia

Al pulsar el botón de *Emergencia*, la pantalla cambia su estética y muestra un fondo rojizo, indicando que se encuentra en este modo, y presentando una nueva saga de pestañas para el control individual de ciertas acciones que se pueden realizar.

Obviamente al pulsar este modo, todo el sistema queda bloqueado, sin movimiento, y los botones que aparecen están diseñados para corregir ciertos problemas como pueden haber sido roturas, estancamientos, enganches… o cualquier problema por el que se requiera de la paralización de la planta.

Figura 21. Interfaz de usuario, pantalla Modo de Emergencia

Vemos también que podemos controlar la temperatura del tratamiento térmico, esto es porque el horno va a seguir funcionando en el *Modo Emergencia*, al no ser que indiquemos lo

contrario. Se ha escogido esta forma ya que habitualmente suele suponer un gran coste energético reactivar un horno o un refrigerador que ha estado un tiempo desactivado, y con ello se evita que, si el problema ha surgido en otro punto de la línea industrial, se apague el control de temperatura innecesariamente.

3. DESARROLLO DE LA SOLUCIÓN

A continuación, se van a desarrollar todos los puntos necesarios para completar el funcionamiento deseado del sistema. Como se podrá comprobar en los siguientes puntos, se ha procurado un método de desarrollo basado en la sencillez para la futura modificación y las mayores unificaciones posibles entre los distintos programas utilizados.

En este punto realizaremos un enfoque basado en la programación de cada autómata en particular, posteriormente veremos la conexión a la red (servidor OPC) y finalmente cómo se ha desarrollado el control del proceso global, que incluye la programación de la *Cámara*.

Las secciones quedarán, por lo tanto:

3.1. MATERIAL UTILIZADO

Estación Multiproceso	Autómata Siemens: SIMATIC S7-1200 CPU 1214C AC/DC/RLY	Módulo SB 1232 AQ 1x12 BIT +/- 10VDC / 0-20mA
		Módulo SM 1223 DC/RLY
	Software Siemens TIA Portal (Totally Integrated Automation Portal) Versión V13 SP1 Update 9 (BASIC)	
	Maqueta FischerTechnik 536632 Multi Processing Station with Oven 24V	
	Célula Peltier para simulación de procesos térmicos	Fuente de alimentación GRELCO VD-305 SF 0-30V / 0-5V
		Fuente de alimentación GRELCO GE0305DVIF 0-30Vcc/0-5A
Estación Línea Mecanizado	Autómata Schneider Modicon M241 TM241CE40R	TMC4AI2 Analog IN
		TMC4AQ2 Analog OUT 0-10V/4-20mA
	Software SoMachine V4.1 SP2	
	Maqueta FischerTechnik 536630 Vacuum Gripper Robot 24V	
Estación Robot Manipulador	Autómata Omron SYSMAC CJ2M CPU31	Módulo MAD42
		Módulo ID211
		Módulo OC211
	Software OMRON CX-Programmer Versión 9.50	
	Maqueta FischerTechnik 96790 Indexed Line with two Machining Stations 24V	

Servidor OPC	Software Kepware KEPServerEX V5.13.191.0
	Red Ethernet
Interfaz visual	Software National Instruments LabVIEW 2013 Service Pack 1 Version 13.0.1 (32-bit)
	Cámara Microsoft XBOX 360 Kinect Model 1414

Tabla 1. Material utilizado

3.2. CONSIDERACIONES GENERALES

El diseño y especificación de los automatismos se ha realizado utilizando la norma **UNE-EN 60848:2013**, con título: Lenguaje de especificación GRAFCET para diagramas funcionales secuenciales (Ratificada por AENOR en julio de 2013).

A su vez, la programación de estos se ha realizado en base a la norma **UNE-EN 61131-3:2013**, de título: Autómatas programables. Parte 3: Lenguajes de programación (Ratificada por AENOR en julio de 2013). Los lenguajes utilizados han sido:

- **SFC** (Sequential Function Chart): Permite organizar y programar secuencias.
- **LD** (Ladder Diagram): lenguaje gráfico que se asemeja a conexiones eléctricas (KOP en TIA Portal)
- **ST** (Structured Text): lenguaje textual ligero (SCL en TIA Portal)

Se ha hecho uso de estas herramientas en función de su disponibilidad en el software de programación del fabricante. La programación de los autómatas sigue, además, una doctrina general que cabe la pena indicar antes de entrar de lleno al desarrollo particular de cada uno.

Esta doctrina general incluye estos conceptos:

- **Programas principales**: contienen las distintas etapas del proceso, y están programados en forma de GRAFCET (lenguaje SFC), exceptuando el autómata Siemens (programado en SCL debido a la no existencia de SFC o similar). Cada proceso secuencial tendrá un programa que llevará su nombre. La elección de este tipo de lenguaje es obvia, debido al componente visual relacionado con los procesos secuenciales.
- **Acciones**: Las acciones, aunque estrechamente ligadas a las etapas del programa principal, se verán desarrolladas aparte utilizando Diagramas de Contactos, que harán referencia a las etapas del *Programa Principal*. Esto se ha hecho así debido a que es más sencillo ver y comprender qué activa o desactiva una Acción (por ejemplo, cierta etapa de cierto GRAFCET) y también es posible insertar mecanismos de seguridad para evitar fallos graves, como, por ejemplo, en el caso del *Modo Paso A Paso* y su capacidad de parar en ciertas etapas. Un ejemplo fundamental de esto es que todos los desplazamientos se programan para que dejen de estar activos cuando se active el final de carrera correspondiente.
- **Variables**: De forma similar a las *Acciones*, serán programadas en un lenguaje de Diagrama de Contactos, ya que se ve muy fácilmente qué condiciones las activan, desactivan, o modifican su valor. Aquí se encuentran todas las variables que sean vitales para la comunicación con otros autómatas o que no se puedan induir en los otros programas (*Testigos*, *Alarmas*).
- **Testigos**: Los testigos ayudan a conocer la posición y situación de las piezas y demás procesos de interés en el SCADA. Contendrá los mecanismos utilizados

para asegurar en el SCADA un conocimiento básico de posiciones y movimientos de las piezas, exceptuando algunos testigos que existan previamente, ya sea de forma directa (sensores) o por su necesaria aparición en otros bloques como el de *Variables*.

- **Alarmas**: Es importante en un proceso industrial saber cuándo algo no está desarrollándose como debería, y por ello se han incluído para los 3 autómatas una serie de alarmas basadas en temporizadores que detectan si alguna acción lleva demasiado tiempo activa, cuando normalmente debería haber terminado antes.
- **Modo Automático/Modo Paso A Paso**: Los 3 autómatas serán programados tal que exista un *Modo Automático* (el uso normal), un *Modo Paso A Paso*, donde se regula mediante un botón que las etapas del programa principal salten a la siguiente (esto se ha realizado de forma inteligente, es decir, no se aplica en todas las etapas, sino en aquellas en las que no vaya a causar errores o carezca de sentido). La variable booleana *NextStep* regula en todos los autómatas que se pueda pasar a la siguiente etapa. Valdrá *TRUE* siempre en el *Modo Automático*, mientras que en el *Modo Paso A Paso* será solamente *TRUE* cuando se pulse el botón de *SIGUIENTE PASO* en la interfaz visual. La gestión de esto se realiza en 3.7. Programación de la interfaz visual, comunicación entre autómatas y cámara, mientras que los autómatas tan sólo reciben la variable *NextStep*, que permite pasar o no a la siguiente etapa.
- **Modo Emergencia**: Modo que sencillamente bloquea los programas principales y permite ciertos controles excepcionales desde la interfaz visual del usuario para poder activar algunas acciones de forma individual e independiente. Al salir de este modo los programas se reinician. Cada software tiene distintas herramientas para gestionar este tipo de funcionamiento.
- **Excepciones**: La estructura y el funcionamiento de los distintos softwares hace que no se puedan programar todos los autómatas de la misma exacta forma, y las diferencias entre los procesos que regulan hace que también aparezcan otros programas necesarios, así como algunas configuraciones particulares, que se tratarán en los apartados de la programación individual de cada uno de ellos.

3.3. DISEÑO E IMPLEMENTACIÓN DE LOS AUTOMATISMOS NECESARIOS PARA LA ESTACIÓN MULTIPROCESO

Se ha escogido el autómata Siemens para programar la *Estación Multiproceso* debido principalmente a su gran capacidad en el control PID de procesos analógicos, y es esta la estación que incluye el tratamiento térmico, con la consecuente necesidad del control de la temperatura.

Para la programación del autómata Siemens se requiere del software de programación de autómatas TIA Portal.

Figura 22. Captura de los Bloques de programa que forman la programación del autómata Siemens

También cabe destacar que la automatización de esta estación solo se puede llevar a cabo gracias al módulo SM 1223, ya que la estación a programar requiere de un alto número de entradas y salidas que no se puede conseguir simplemente con el módulo principal.

3.3.1. Entradas, salidas y variables globales importantes

A continuación, se presentan las entradas y las salidas que existen para la *Estación Multiproceso*, junto a las variables globales más importantes que ayudan a tener un concepto global más claro de la programación realizada. Entre otras, se han obviado las variables que corresponderían con las "etapas" en el lenguaje SFC, y las variables auxiliares de detección de flancos, que en otros softwares de programación son generadas automáticamente, mientras que en este no.

Nombre	*Tipo*	*Dirección*	*Comentario*
I1	Bool	%I8.4	Final de carrera mesa en manipulador
I2	Bool	%I8.5	Final de carrera mesa en cinta
I3	Bool	%I8.6	Fototransistor pieza final cinta
I4	Bool	%I8.7	Final de carrera mesa en sierra
I5	Bool	%I8.0	Final de carrera manipulador en mesa
I6	Bool	%I9.2	Final de carrera alimentador horno dentro
I7	Bool	%I8.1	Final de carrera alimentador horno fuera
I8	Bool	%I9.3	Final de carrera manipulador en horno
I9	Bool	%I8.2	Fototransistor pieza en alimentador
Q1	Bool	%Q8.0	Motor giro mesa horario
Q2	Bool	%Q8.1	Motor giro mesa antihorario
Q3	Bool	%Q8.2	Motor movimiento avance cinta
Q4	Bool	%Q8.3	Motor movimiento sierra
Q5	Bool	%Q8.4	Motor alimentador horno adentro
Q6	Bool	%Q8.5	Motor alimentador horno afuera

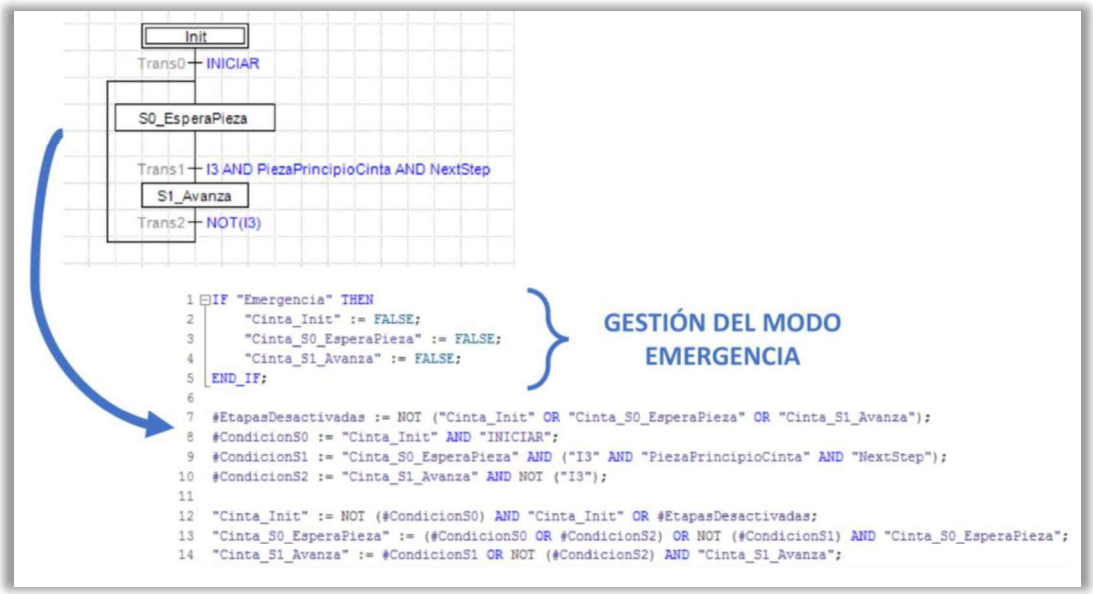
Q7	Bool	%Q8.6	Motor movimiento manipulador hacia el horno
Q8	Bool	%Q8.7	Motor movimiento manipulador hacia mesa giratoria
Q9	Bool	%Q9.0	Habilitar señales entrada del proceso
Q10	Bool	%Q9.1	Habilitar compresor
Q11	Bool	%Q9.2	Habilitar succión ventosa
Q12	Bool	%Q9.3	Movimiento manipulador abajo
Q13	Bool	%Q9.4	Movimiento puerta arriba
Q14	Bool	%Q9.5	Movimiento empujador mesa
Q15	Bool	%Q9.6	Habilitar luz horno
BotonQ1	Bool	%M0.0	Habilita Q1
BotonQ2	Bool	%M0.1	Habilita Q2
BotonQ3	Bool	%M0.2	Habilita Q3
BotonQ4	Bool	%M0.3	Habilita Q4
BotonQ5	Bool	%M0.4	Habilita Q5
BotonQ6	Bool	%M0.5	Habilita Q6
BotonQ7	Bool	%M0.6	Habilita Q7
BotonQ8	Bool	%M0.7	Habilita Q8
BotonQ10	Bool	%M1.1	Habilita Q10
BotonQ11	Bool	%M1.2	Desactiva la succión Q11
BotonQ12	Bool	%M1.3	Habilita Q12
BotonQ13	Bool	%M1.4	Habilita Q13
BotonQ14	Bool	%M1.5	Habilita Q14
BotonQ15	Bool	%M1.6	Habilita Q15
INICIAR	Bool	%M1.7	Inicia el sistema
NextStep	Bool	%M2.0	Habilita el siguiente paso (ya sea en modo automático o paso a paso)
PiezaEnMesa	Bool	%M6.1	Indica si existe una pieza en la mesa
PiezaPrincipioCinta	Bool	%M6.7	Existe una pieza al principio de la cinta
Emergencia	Bool	%M7.0	Activa el modo emergencia
BrazoPiezaDejada	Bool	%M7.1	Ya se ha dejado una pieza en la plataforma
ManipAccesoAdmitido	Bool	%M7.2	El manipulador puede desplazarse hacia el horno para recoger una futura pieza
MPAceptaPieza	Bool	%M7.3	Avisa a la Estación Robot Manipulador que puede dejar una pieza
TiempoHorno	Time	%MD8	Tiempo de procesado en el horno asignado
TiempoProcesado	Time	%MD12	Tiempo de procesado en la mesa asignado
PiezaFinalMultiproceso	Bool	%M20.5	Existe una pieza al final de la estación multiproceso
TiempoHornoRED	Time	%MD16	Tiempo en el tratamiento térmico para una pieza roja
RED	Bool	%M7.5	Existe una pieza roja en Cámara
WHITE	Bool	%M7.6	Existe una pieza blanca en Cámara
BLUE	Bool	%M7.7	Existe una pieza azul en Cámara
TiempoHornoBLUE	Time	%MD22	Tiempo en el tratamiento térmico para una pieza azul
TiempoProcesadoRED	Time	%MD26	Tiempo en el procesado para una pieza roja
TiempoProcesadoWHITE	Time	%MD30	Tiempo en el procesado para una pieza blanca
TiempoProcesadoBLUE	Time	%MD34	Tiempo en el procesado para una pieza azul

TiempoHornoWHITE	Time	%MD38	Tiempo en el tratamiento térmico para una pieza blanca
BrazoACamara	Bool	%M20.2	El brazo está bajando a por una pieza en la cámara (activa captura de color)
TempRED	Real	%MD42	Temperatura en el tratamiento térmico para una pieza roja
TempWHITE	Real	%MD46	Temperatura en el tratamiento térmico para una pieza blanca
TempBLUE	Real	%MD50	Temperatura en el tratamiento térmico para una pieza azul
TempAdecuada	Bool	%M20.0	El horno puede abrir porque se ha alcanzado la temperatura deseada
TempSensor	Word	%IW64	Entrada del PID, palabra digital
TempDeseada	Real	%MD54	Referencia del PID, en ºC
AccionPID	Word	%QW80	Salida del PID, palabra digital
TempHorno	Real	%MD66	Temperatura actual en el horno, conversión de la variable TempSensor
ApagarRefrigerador	Bool	%M20.6	Botón para poner a 0 V la salida del PID
PiezaEnManipulador	Bool	%M21.0	Existe una pieza en el Manipulador
PiezasIN	Int	%MW70	Número de piezas dentro
PiezasInRED	Int	%MW72	Número de piezas dentro rojas
PiezasInWHITE	Int	%MW74	Número de piezas dentro blancas
PiezasInBLUE	Int	%MW76	Número de piezas dentro azules
PiezasOutRED	Int	%MW78	Número de piezas procesadas rojas
PiezasOutWHITE	Int	%MW80	Número de piezas procesadas blancas
PiezasOutBLUE	Int	%MW82	Número de piezas procesadas azules
PiezasOUT	Int	%MW84	Número de piezas procesadas
PiezaFinalLineaMecanizado	Bool	%M21.2	Existe una pieza al final de toda la línea industrial
PiezasINMultiproceso	Int	%MW86	Número de piezas dentro
CodigoAlarma	Int	%MW88	Código de alarma para la interfaz visual
PiezaEnPlataforma	Bool	%M21.4	Indica que existe una pieza en la plataforma
ResetContadores	Bool	%M20.7	Botón para resetear todos los contadores

Tabla 2. Variables globales para la programación del autómata Siemens

3.3.2. Programas principales

Como se ha indicado en el apartado inmediatamente anterior, el software de programación para este autómata, TIA Portal no incluye un modo de programación que se asemeje al sistema GRAFCET, y se ha utilizado una "conversión" sencilla de los grafcets diseñados a formulación en programa de texto estructurado (lenguaje SCL).




```
IF "Emergencia" THEN
    "Cinta_Init" := FALSE;
    "Cinta_S0_EsperaPieza" := FALSE;
    "Cinta_S1_Avanza" := FALSE;
END_IF;

#EtapasDesactivadas := NOT ("Cinta_Init" OR "Cinta_S0_EsperaPieza" OR "Cinta_S1_Avanza");
#CondicionS0 := "Cinta_Init" AND "INICIAR";
#CondicionS1 := "Cinta_S0_EsperaPieza" AND ("I3" AND "PiezaPrincipioCinta" AND "NextStep");
#CondicionS2 := "Cinta_S1_Avanza" AND NOT ("I3");

"Cinta_Init" := NOT (#CondicionS0) AND "Cinta_Init" OR #EtapasDesactivadas;
"Cinta_S0_EsperaPieza" := (#CondicionS0 OR #CondicionS2) OR NOT (#CondicionS1) AND "Cinta_S0_EsperaPieza";
"Cinta_S1_Avanza" := #CondicionS1 OR NOT (#CondicionS2) AND "Cinta_S1_Avanza";
```

Figura 23. Ejemplo de cómo se ha pasado de un diagrama GRAFCET para el programa Cinta a lenguaje SCL, para el automata Siemens

Se puede comprobar que la división de la programación del proceso principal en los 4 distintos procesos individuales termina con diagramas GRAFCET lineales sin estructuras complejas, y por lo tanto fáciles de transcribir al lenguaje SCL. La contraposición es que serán necesarias variables que sirvan de "conexión" entre estos distintos "subprogramas".

Es de destacar que la programación del *Modo Paso A Paso* ha tenido en cuenta que el proceso térmico no puede permitirse esperar a que el operario active el siguiente paso. Es decir, aunque esté activado el *Modo Paso A Paso*, todas las etapas (apertura, entrada de pieza, cierre de la compuerta, tiempo de procesado, reapertura, salida de pieza y cierre de puerta) se realizarán de forma automática para evitar contaminación térmica y/o que la pieza no alcance las características demandadas.

Otro punto fundamental en el funcionamiento de la Estación es la entrada de la pieza en su subsistema. No existen sensores directos que detecten que existe pieza en la *Plataforma*, sino que existe un fotorreceptor que detecta si ha pasado un brazo a dejar la pieza. Es por ello que es necesario utilizar una variable auxiliar que se active cuando ocurra esta detección, y que se desactive cuando ya se haya gestionado la entrada de la pieza en el subsistema. Esta variable se ha llamado *BrazoPiezaDejada*. Con la *Estación Línea Mecanizado* ocurre algo parecido salvo que esta sí tiene sensores que detectan que existe una pieza en posición.

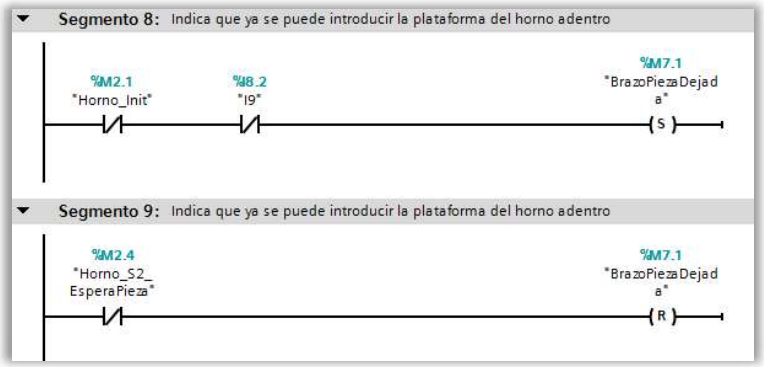


Figura 24. Captura del programa Variables para el autómata Siemens, donde se ve la gestión de BrazoPiezaDejada

Las variables que "conectan" los distintos GRAFCETs (*PiezaEnMesa*, *PiezaPrincipioCinta*, *ManipAccesoAdmitido*) así como las variables necesarias para la comunicación entre autómatas (*MPAceptaPieza*, *PiezaFinalMultiproceso*) son gestionadas en el programa *Variables*.

Un punto crucial que no debe pasar desapercibido es el hecho de que el *Horno* no permita introducir una pieza mientras que la temperatura no sea la correcta. Esto se ha gestionado mediante la variable *TempAdecuada*, en el programa de *Variables*. Para ello se ha creado una función sencilla que se ha llamado *RANGO*, que simplemente compara dos valores y comprueba que su valor no diverja entre ellas una *Diferencia* dada. La función saca una señal booleana indicando si se encuentran las señales dentro del rango o no.

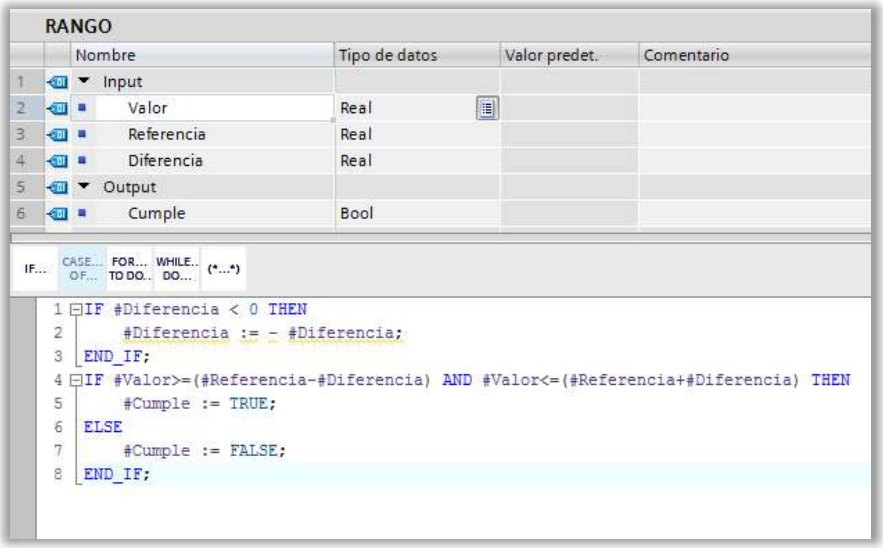


Figura 25. Variables locales y programación del programa RANGO para el autómata Siemens

Figura 26. Segmento 16 del programa Variables del autómata Siemens, mostrando el uso de RANGO

Para poder utilizar esta función se necesita comparar la temperatura deseada con la temperatura actual del horno. La temperatura del horno se recibe mediante *TempSensor*, pero esta es una palabra que está relacionada con la tensión que se recibe en la entrada que tiene asignada. Por lo tanto, necesitamos operar con ese valor para convertirlo a valor de temperatura real, y para ello se utilizan dos funciones nativas en TIA Portal, NORM_X y SCALE_X para realizar correctamente la conversión. La relación entre las tensiones y las temperaturas queda expuesta en el apartado 3.3.7. Control analógico de la temperatura.

En el Segmento 10 del programa *Acciones* vemos cómo se hace la gestión de la salida correspondiente al compresor de la *Estación Multiproceso*. Se ha tomado en consideración el ahorro energético que puede suponer utilizar el compresor solo en los casos necesarios. El compresor es una acción auxiliar, es decir, debemos activarlo solamente porque otra acción necesita de este, y por ello se ha programado para que se active cuando las acciones pertinentes se hayan activado.

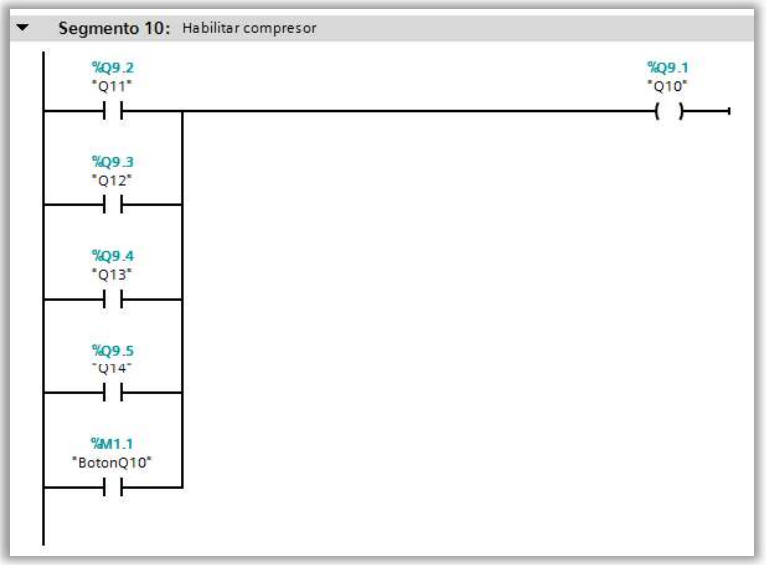


Figura 27. Segmento 10 del programa de Acciones para el autómata Siemens, mostrando la acción del compresor

Se ha utilizado un programa complementario *Temporizadores*, que sirve esencialmente para contener los temporizadores que se usan en los programas principales. Esto se ha hecho así ya que resulta más cómodo visualizar las funciones de temporización en el lenguaje KOP que en el lenguaje de texto SCL.

3.3.3. FIFOs

Al igual que ocurrirá con la *Estación Línea Mecanizado* con sus propias variables de control, necesitamos un sistema que permita guardar en orden qué tipo de pieza ha entrado al sistema, para poder cambiar el valor de las variables *TiempoHorno*, *TempHorno*, y *TiempoProcesado*, dependiendo de qué tipo sea. Para ello se ha realizado un programa en texto estructurado (SCL, que se puede reutilizar para los lenguajes ST en los otros autómatas) que se asemeja a las estructuras FIFO.

La señal de entrada *GET* sirve para leer qué pieza ha entrado en la cámara. Esto se realiza cuando la *Estación Robot Manipulador* active la señal *BrazoACamara*, activada cuando está apunto de recoger la pieza en la zona de la cámara, y con ello se asegura una lectura correcta en prácticamente la totalidad de los casos.

FIFOTime

		Nombre	Tipo de datos	Valor predet.	Remanencia
1		Input			
2		SET	Bool	false	No remane...
3		RESET	Bool	false	No rem...
4		TiempoRED	Time	T#0ms	No remane...
5		TiempoWHITE	Time	T#0ms	No remane...
6		TiempoBLUE	Time	T#0ms	No remane...
7		GET	Bool	false	No remane...
8		Output			
9		Tiempo	Time	T#0ms	No remane...
10		InOut			
11		<Agregar>			
12		Static			
13		FreeSlot	Int	0	No remane...
14		QUEUE	Array[0..31] of Time		No remane...
15		SET_DONE	Bool	false	No remane...
16		GET_DONE	Bool	false	No remane...
17		Temp			
18		J	Int		
19		Constant			
20		<Agregar>			

Figura 28. Variables internas para la función FIFOTime, del autómata Siemens

```
IF NOT (#RESET) THEN
    IF #GET AND NOT(#GET_DONE) THEN
        IF "RED" AND NOT ("WHITE") AND NOT ("BLUE") THEN
            #QUEUE[#FreeSlot] := #TiempoRED;
            #FreeSlot := #FreeSlot + 1;
            #GET_DONE := TRUE;
        ELSIF "WHITE" AND NOT ("RED") AND NOT ("BLUE") THEN
            #QUEUE[#FreeSlot] := #TiempoWHITE;
            #FreeSlot := #FreeSlot + 1;
            #GET_DONE := TRUE;
        ELSIF "BLUE" AND NOT ("RED") AND NOT ("WHITE") THEN
            #QUEUE[#FreeSlot] := #TiempoBLUE;
            #FreeSlot := #FreeSlot + 1;
            #GET_DONE := TRUE;
        END_IF;
    ELSIF #GET_DONE AND NOT (#GET) THEN
        #GET_DONE := FALSE;
    END_IF;
    IF NOT (#SET_DONE) THEN
        IF #SET THEN
            #SET_DONE := TRUE;
            #Tiempo := #QUEUE[0];
            FOR #J := 0 TO 30 BY 1 DO
                #QUEUE[#J] := #QUEUE[#J + 1];
            END_FOR;
            #FreeSlot := #FreeSlot - 1;
        END_IF;
    ELSIF NOT (#SET) THEN
        #SET_DONE := FALSE;
    END_IF;
ELSE
    #FreeSlot := 0;
END_IF;
```

Figura 29. Programación de la función FIFOTime, del autómata Siemens

Existen otras variables de entrada, que se corresponden al valor que se quiere que se adopte en *TiempoHorno*, *TempHorno* y *TiempoProcesado* para cada uno de los tipos de pieza. Estas variables se pueden asignar en la interfaz de usuario. Cuando se activa *GET*, se guarda en el primer hueco libre de un vector estático el valor correspondiente a la pieza. El tamaño del vector se ha escogido suficientemente grande para poder almacenar el valor de todas las piezas que pueden encontrarse a la vez dentro del proceso industrial completo.

La señal *SET* coloca en la variable a la que haga referencia la salida del *FIFO* el primer valor del vector. Automáticamente, el resto de valores del vector se mueven "una posición hacia la izquierda", ya que el primer valor ha sido utilizado, y el resto de variables escalan una posición. Mediante la variable *FreeSlot* el programa es consciente de dónde se encuentra el primer hueco "libre" donde poder guardar información en el vector con la activación de la señal GET.

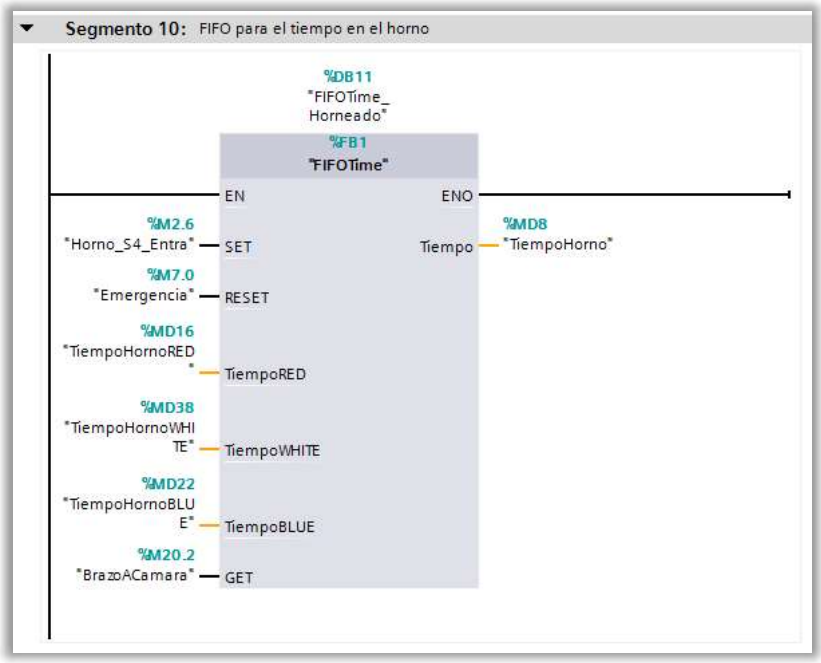


Figura 30. Segmento 10 del programa Variables, implementación del FIFOTime, para el autómata Siemens

Se puede comprobar que se ha diseñado el programa para que se detecte el flanco de subida de las señales de *GET* y de *SET*, sin tener que utilizar en el lenguaje KOP interruptores de flanco. También se ha programado para que la función reinicie el *FIFO* cuando se pulse el botón de *Emergencia*.

Debido a las necesidades diferentes de *TempHorno*, se ha realizado una modificación sobre el programa FIFO del que se ha hablado antes. Cabe destacar la importancia especial de que la variable *TempHorno*, que indica la temperatura a la que será procesada térmicamente la pieza, sea cambiada a la de la pieza siguiente inmediatamente después de procesar la anterior, ya que habitualmente cambiar la temperatura en una cámara térmica precisa de tiempo, y es importante que el proceso industrial sea lo más rápido posible. Esto se ha conseguido cambiando el concepto del SET anterior. Ahora cada vez que el programa se ejecuta, se pregunta si se puede cambiar ya la temperatura, y en cuanto esto sea cierto pasará a poner en la salida el valor de la siguiente temperatura siempre y cuando haya una siguiente pieza, en el caso de no haberla, el horno mantendrá la temperatura en la que estaba.

FIFOTemp

	Nombre	Tipo de datos	Valor predet.	Remanencia
1	Input			
2	RESET	Bool	false	No remane...
3	TempRED	Real	0.0	No remane...
4	TempWHITE	Real	0.0	No remane...
5	TempBLUE	Real	0.0	No remane...
6	GET	Bool	false	No remane...
7	SET_POSSIBLE	Bool	false	No remane...
8	Output			
9	Temp	Real	7.0	No remane...
10	InOut			
11	<Agregar>			
12	Static			
13	FreeSlot	Int	0	No remane...
14	QUEUE	Array[0..31] of Real		No remane...
15	SET_DONE	Bool	false	No remane...
16	GET_DONE	Bool	false	No remane...
17	AdmiteTempNueva	Bool	true	No remane...
18	Temp			
19	J	Int		
20	Constant			
21	<Agregar>			

Figura 31. Variables internas de la función FIFOTemp, para el autómata Siemens

```
IF NOT (#RESET) THEN
    IF #SET_POSSIBLE AND NOT (#AdmiteTempNueva) THEN
        #AdmiteTempNueva := TRUE;
    END_IF;

    IF #GET AND NOT (#GET_DONE) THEN
        IF "RED" AND NOT ("WHITE") AND NOT ("BLUE") THEN
            #QUEUE[#FreeSlot] := #TempRED;
            #FreeSlot := #FreeSlot + 1;
            #GET_DONE := TRUE;
        ELSIF "WHITE" AND NOT ("RED") AND NOT ("BLUE") THEN
            #QUEUE[#FreeSlot] := #TempWHITE;
            #FreeSlot := #FreeSlot + 1;
            #GET_DONE := TRUE;
        ELSIF "BLUE" AND NOT ("RED") AND NOT ("WHITE") THEN
            #QUEUE[#FreeSlot] := #TempBLUE;
            #FreeSlot := #FreeSlot + 1;
            #GET_DONE := TRUE;
        END_IF;
    ELSIF #GET_DONE AND NOT (#GET) THEN
        #GET_DONE := FALSE;
    END_IF;

    IF #FreeSlot>0 AND #AdmiteTempNueva THEN
        #AdmiteTempNueva:= FALSE;
        #Temp := #QUEUE[0];
        FOR #J := 0 TO 30 BY 1 DO
            #QUEUE[#J] := #QUEUE[#J + 1];
        END_FOR;
        #FreeSlot := #FreeSlot - 1;
    END_IF;
ELSE
    #AdmiteTempNueva := TRUE;
    #FreeSlot := 0;
END_IF;
```

Figura 32. Programación del FIFOTemp, para el autómata Siemens

A diferencia de otros casos donde se podría sencillamente guardar en estas "colas" de memoria el tipo de pieza que son; aquí se ha optado por directamente guardar el valor de la variable correspondiente, para así asegurar que si el operario cambia el valor de una variable, por ejemplo *TempHornoRED*, esto no afecte a las piezas rojas que ya se encuentren dentro del proceso, sino que afecte a las nuevas que se introduzcan a partir de ese momento.

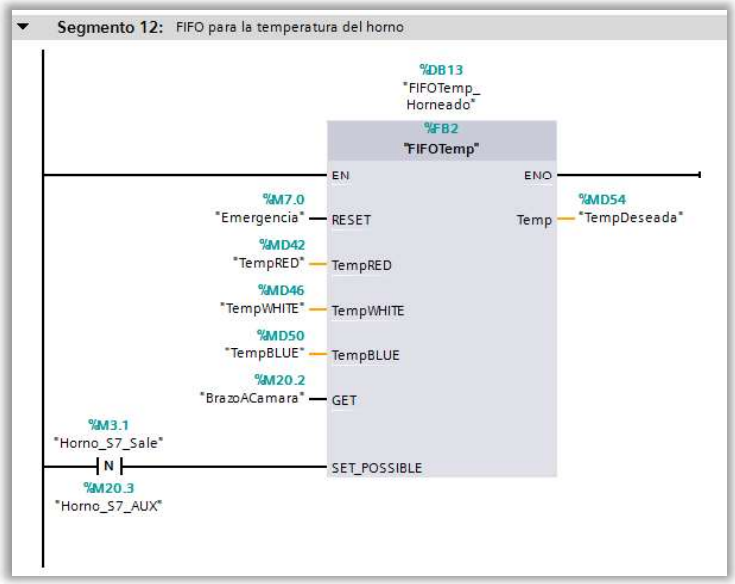


Figura 33. Implementación del FIFOTemp, en el programa Variables, para el autómata Siemens

3.3.4. Testigos

Se han programado *testigos* relativos a la posición de las piezas en la *Plataforma* inicial, en el *Horno*, para la localización en el *Manipulador*, en la *Mesa* (también se ha incluído uno que indica que el procesado en la *Mesa* está activo), y para la posición de una pieza en el inicio de la *Cinta* y en el final. Como ya se ha explicado en el apartado 3.2. Consideraciones generales, los testigos que cumplen otras funciones aparte de la simple señalización de posición de pieza en la interfaz no se encontrarán dentro del programa específico *Testigos*.

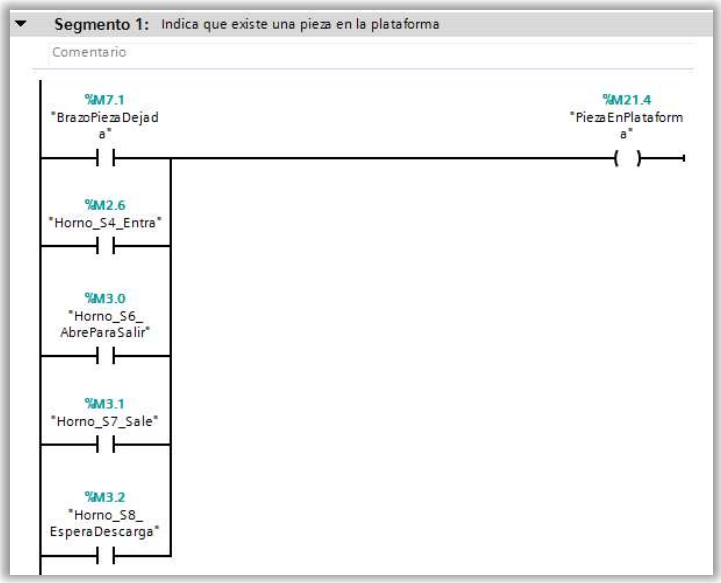


Figura 34. Segmento 1 del programa Testigos para el autómata Siemens

3.3.5. Alarmas

Para el programa de *Alarmas* se ha diseñado en texto estructurado un *Codificador* que es capaz de generar un entero dependiendo de qué alarma se ha activado. Como también se ha explicado en 3.2. Consideraciones generales, las alarmas están basadas en temporizadores de acciones que no deberían de activarse continuamente durante más de un tiempo determinado.

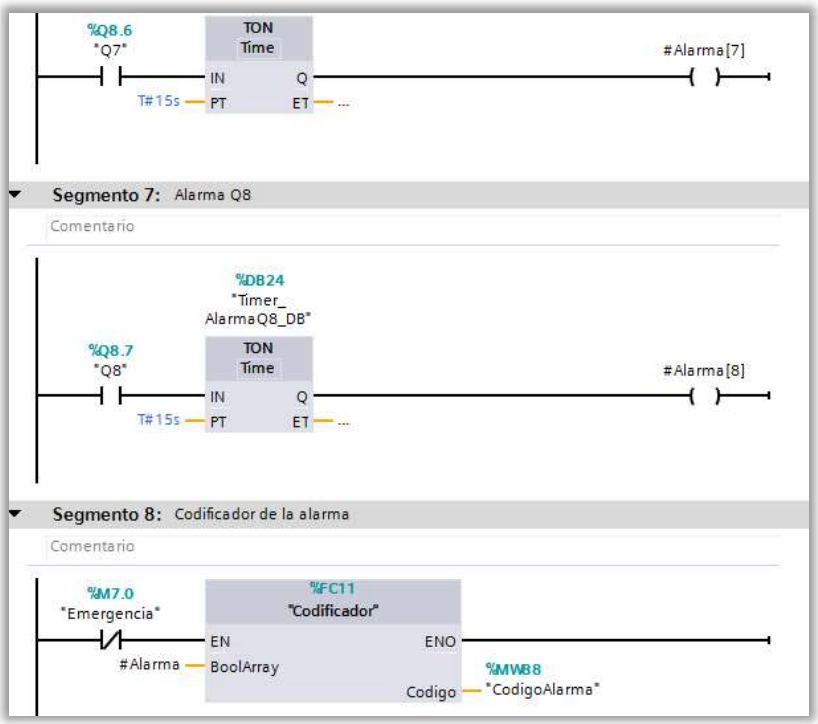


Figura 35. Parte del programa Alarmas, para el autómata Siemens

Las distintas alarmas se guardan en un vector booleano *Alarma[]* temporal del programa *Alarmas* que es usado por el *Codificador* para extraer la información, es decir, qué "casilla" del vector está activa. El número de *CodigoAlarma* corresponderá con el número de la acción que está activa demasiado tiempo.

Codificador

		Nombre	Tipo de datos	Valor predet.
1		Input		
2		BoolArray	Array[1..10] ...	
3		Output		
4		Codigo	Int	
5		InOut		
6		<Agregar>		
7		Temp		
8		I	Int	
9		HayActivo	Bool	
10		Constant		
11		<Agregar>		
12		Return		
13		Codificador	Void	

IF... CASE... OF... FOR... TO DO... WHILE... DO... (*...*)

```
1  #HayActivo := FALSE;
2  FOR #I := 1 TO 10 BY 1 DO
3      IF #BoolArray[#I] = TRUE THEN
4          #Codigo := #I;
5          #HayActivo :=TRUE;
6      END_IF;
7  END_FOR;
8  IF NOT(#HayActivo) THEN
9      #Codigo := 0;
10 END_IF;
```

Figura 36. Variables y programación de la función Codificador, en el autómata Siemens

3.3.6. Contadores

Se ha implementado un contador llamado *ContadorPiezas* que nos ayuda a conocer cuántas piezas existen dentro de nuestro proceso global, distinguiendo entre el tipo de pieza, así como también indicando cuántas piezas y de qué tipo se han producido ya. Como se trata del proceso global, y no solo de la *Estación Multiproceso*, el programa necesitará recibir la variable del servidor OPC *PiezaFinalLineaMecanizado*, que indica que una pieza ha terminado el proceso completo.

ContadorPiezas

		Nombre	Tipo de datos	Valor predet.	Remanencia
1		Input			
2		RESET	Bool	false	No rem...
3		IN	Bool	false	No remane...
4		OUT	Bool	false	No remane...
5		Output			
6		<Agregar>			
7		InOut			
8		<Agregar>			
9		Static			
10		QUEUE	Array[0..31] of Int		No remane...
11		IN_DONE	Bool	false	No remane...
12		OUT_DONE	Bool	false	No remane...
13		FreeSlot	Int	0	No remane...
14		J	Int	0	No remane...
15		RESET_DONE	Bool	false	No remane...
16		Temp			
17		<Agregar>			
18		Constant			
19		<Agregar>			

Figura 37. Variables locales del programa ContadorPiezas para el autómata Siemens

```
IF NOT (#RESET) AND NOT(#RESET_DONE) THEN
    IF #IN AND NOT (#IN_DONE) THEN
        IF "RED" AND NOT ("WHITE") AND NOT ("BLUE") THEN
            "PiezasInRED" := "PiezasInRED" + 1;
            #QUEUE[#FreeSlot] := 1;
        ELSIF "WHITE" AND NOT ("RED") AND NOT ("BLUE") THEN
            "PiezasInWHITE" := "PiezasInWHITE" + 1;
            #QUEUE[#FreeSlot] := 2;
        ELSIF "BLUE" AND NOT ("RED") AND NOT ("WHITE") THEN
            "PiezasInBLUE" := "PiezasInBLUE" + 1;
            #QUEUE[#FreeSlot] := 3;
        END_IF;
        #FreeSlot := #FreeSlot + 1;
        #IN_DONE := TRUE;
    ELSIF #IN_DONE AND NOT (#IN) THEN
        #IN_DONE := FALSE;
    END_IF;
    IF #OUT AND NOT (#OUT_DONE) THEN
        IF #QUEUE[0]=1 THEN
            "PiezasInRED" := "PiezasInRED" -1;
            "PiezasOutRED" := "PiezasOutRED" + 1;
        ELSIF #QUEUE[0]=2 THEN
            "PiezasInWHITE" := "PiezasInWHITE" - 1;
            "PiezasOutWHITE" := "PiezasOutWHITE" + 1;
        ELSIF #QUEUE[0]=3 THEN
            "PiezasInBLUE" := "PiezasInBLUE" - 1;
            "PiezasOutBLUE" := "PiezasOutBLUE" + 1;
        END_IF;
        FOR #J := 0 TO 30 BY 1 DO
            #QUEUE[#J] := #QUEUE[#J + 1];
        END_FOR;
        #FreeSlot := #FreeSlot - 1;
        #OUT_DONE := TRUE;
    ELSIF #OUT_DONE AND NOT (#IN) THEN
        #OUT_DONE := FALSE;
    END_IF;
    "PiezasIN" := "PiezasInRED" + "PiezasInWHITE" + "PiezasInBLUE";
    "PiezasOUT" := "PiezasOutRED" + "PiezasOutWHITE" + "PiezasOutBLUE";
ELSIF #RESET AND NOT #RESET_DONE THEN
    #RESET_DONE := TRUE;
ELSIF NOT(#RESET) AND #RESET_DONE THEN
    #RESET_DONE := FALSE;
    #FreeSlot := 0;
    "PiezasIN" := 0;
    "PiezasInRED" := 0;
    "PiezasInWHITE" := 0;
    "PiezasInBLUE" := 0;
    "PiezasOUT" := 0;
    "PiezasOutRED" := 0;
    "PiezasOutWHITE" := 0;
    "PiezasOutBLUE" := 0;
END_IF;
```

Figura 38. Programación del programa ContadorPiezas para el autómata Siemens

Se ha incluido un botón que podrá resetear las cuentas en cualquier momento. Esto puede ser útil ante posibles errores que hayan ocurrido con las cuentas, sin necesidad de entrar en el modo *Emergencia*, que también resetea estas estadísticas (junto con el resto del sistema).

También se ha utilizado un contador predefinido en el software, un *CTUD*, para contar las piezas que se encuentran dentro de la *Estación Multiproceso* de forma simple. Ambos contadores se pueden encontrar en los segmentos 13 y 14 del programa *Variables*.

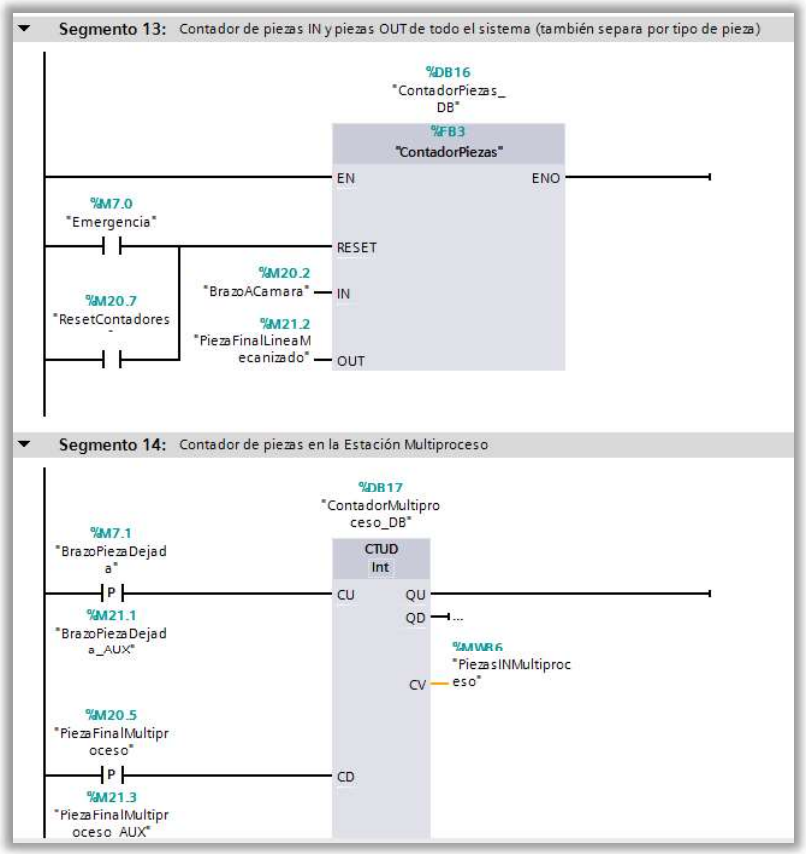


Figura 39. Captura del programa Variables para el autómata Siemens, mostrando los contadores

3.3.7. Control analógico de la temperatura

En esta Estación existe un proceso de tratamiento térmico, y se requiere de un control de su temperatura.

Se debe crear una función que se active por interrupción cada cierto tiempo determinado, *PID_Interrupt*, y se ha utilizado el programa predefinido de TIA Portal llamado *PID_Compact*. Se trata de un programa donde se permite configurar un PID o un PI para controlar una señal real (no booleana, obviamente). El software incluye una funcionalidad junto a su *Configuración* que analiza automáticamente el proceso a controlar y determina las variables de la función del PID, llamada *Puesta en servicio*.

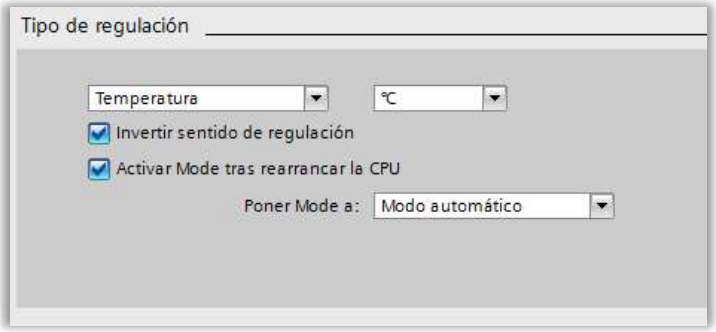


Figura 40. Configuración del Tipo de regulación del PID utilizado en el autóomata Siemens

En este caso, la simulación se ha realizado mediante el control de la cara fría de una célula Peltier, cuyo proceso interno escapa al alcance de este proyecto, pero que a grandes rasgos permite variar la temperatura de sus caras (fría y caliente) según la tensión aplicada. Se ha escogido el control de la cara fría debido a su mayor rapidez.

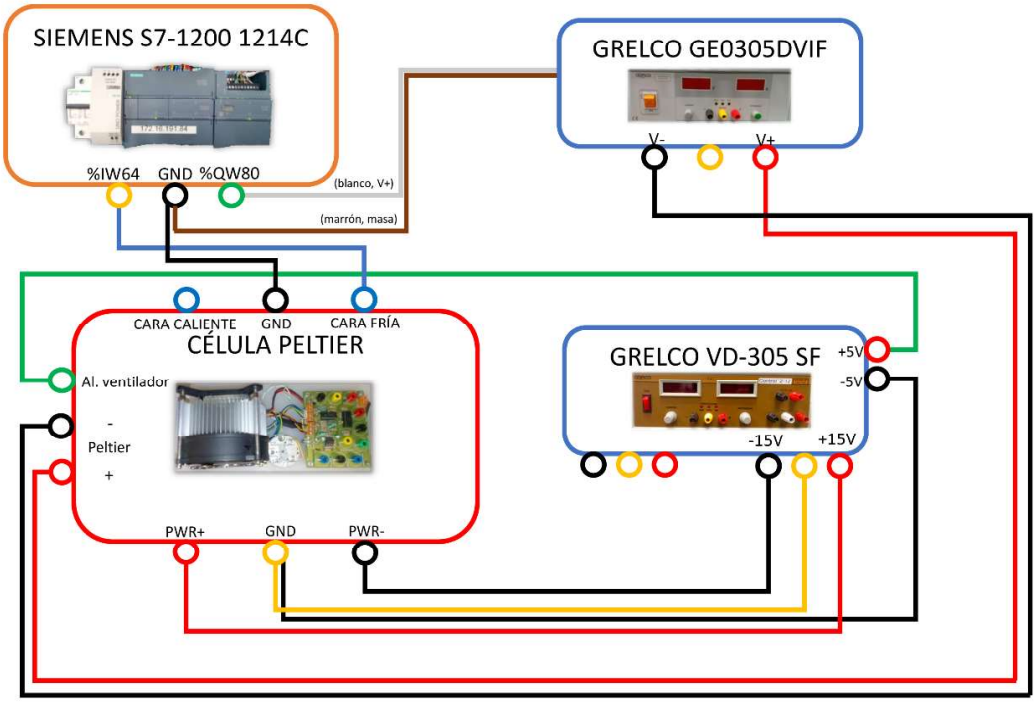


Figura 41. Esquema de conexiones para el control de temperatura simulado mediante la célula Peltier

En la configuración del PID del autómata, se ha escogido específicamente realizar un control PI, donde habrá que seleccionar la casilla que permite "*Invertir sentido de regulación*", ya que, al aumentar la tensión, disminuye la temperatura de la cara fría. En caso de controlar la cara caliente, habría que desactivar esta opción, puesto que, a más tensión, más temperatura.

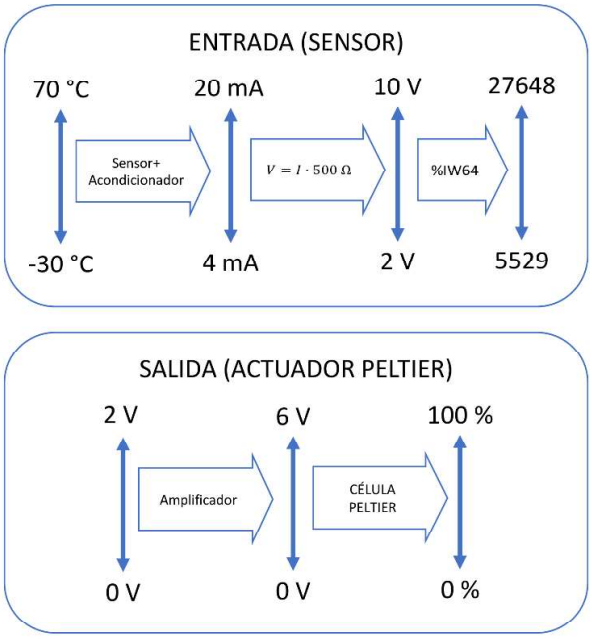


Figura 42. Equivalencias de temperatura y tensión para el control de la cara fría de la célula Peltier

Es importante también configurar cuál queremos que sea la *entrada (Input)*, cuál queremos que sea la *referencia (Setpoint)*, y cuál queremos que sea la *salida (Output)*. Las relaciones tensión/temperatura también se tienen que tener en cuenta, así como evitar que el PID pueda emitir una salida fuera de los rangos determinados por el proceso a controlar.

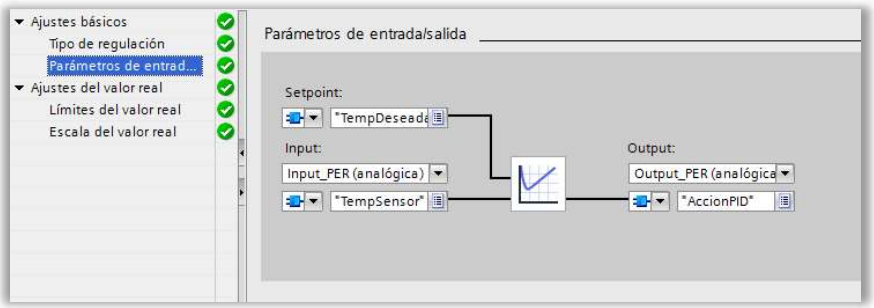


Figura 43. Parámetros seleccionados de entrada y salida para el PID del autómata Siemens

Se ha aprovechado la posibilidad de realizar un control manual de la señal de salida del PID para, mediante un botón, apagar el control de temperatura cambiando la salida a 0 V.

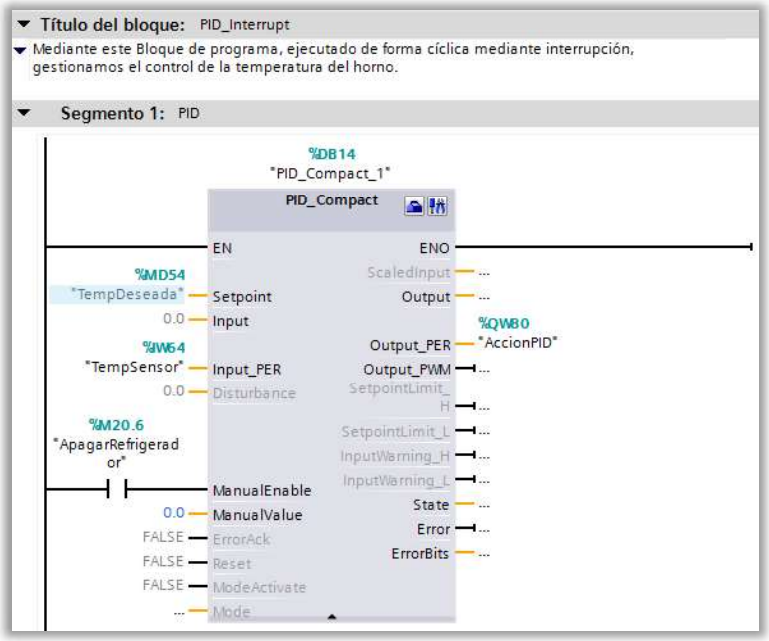


Figura 44. Captura del programa cíclico por interrupción, PID_Interrupt, para el autómata Siemens

3.3.8. Modo Emergencia

La gestión de este modo se ha realizado de forma manual, debido a la sencillez de aplicación del texto estructurado SCL. Como se puede ver en la Figura 23, existe un *IF* que básicamente desactiva todas las etapas. También se puede ver que estos programas automáticamente activan la primera etapa (**_Init*) en cuanto la variable *Emergencia* se ha desactivado.

Como particularidades interesantes que se pueden remarcar del programa de *Acciones*, el Segmento 11 y el Segmento 12, programados a conciencia para que la succión en la ventosa necesaria por el *Manipulador* de esta Estación no se desactive en caso de entrar al *Modo Emergencia*, y que sea el propio operario el que pueda desactivar la succión ya entrados en el

Modo Emergencia mediante el *BotonQ11*. Con esto conseguimos que, al entrar en Emergencia, no se produzcan posibles daños por el desprendimiento de la pieza que pudiera llevar el *Manipulador*. Esto también se ha realizado de una forma parecida para la *Estación Robot Manipulador*.

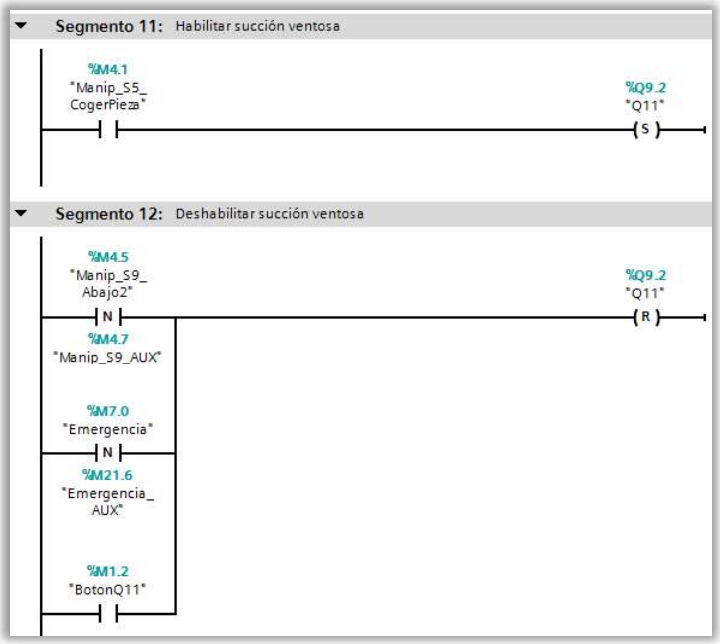


Figura 45. Segmentos 11 y 12 del programa Acciones para el autómata Siemens, mostrando la gestión de la Succión

Como detalle añadido, cabe decir que la programación de la acción del compresor también permite que en el *Modo Emergencia* no sea necesario activar el compresor de forma independiente para activar una acción que lo necesite, ya que automáticamente también arrancará cuando queramos activar la otra acción.

3.4. DISEÑO E IMPLEMENTACIÓN DE LOS AUTOMATISMOS NECESARIOS PARA LA ESTACIÓN LÍNEA MECANIZADO

Para la programación de esta estación se ha escogido el autómata Schneider, puesto que el software para programarlo, SoMachine, cuenta con una gran implementación de los diagramas secuenciales SFC, que se adaptan muy bien a las necesidades del proceso secuencial que se lleva a cabo en la *Estación Línea Mecanizado*.

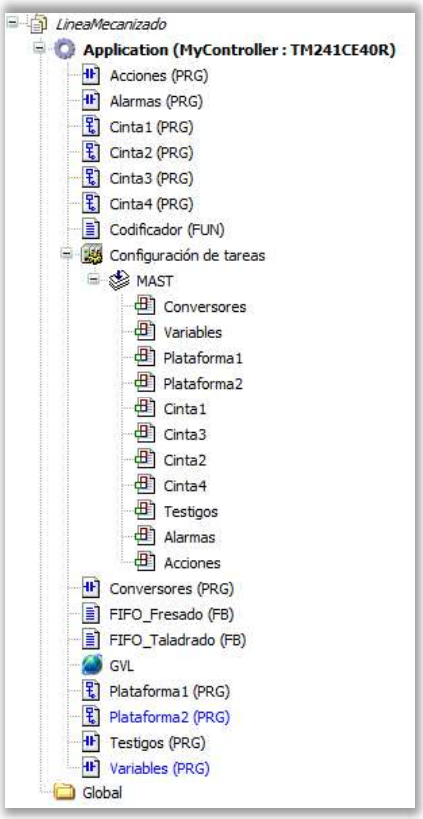


Figura 46. Captura de la gestión general de la Aplicación creada en SoMachine para el autómata Schneider

3.4.1. Entradas, salidas y variables globales importantes

A continuación se presentan las entradas y las salidas que existen para la *Estación Línea* Mecanizado, junto a las variables globales más importantes que ayudan a tener un concepto global más claro de la programación realizada.

Nombre	*Tipo*	*Dirección*	*Comentario*
Q1	BOOL	%QX0.4	Motor empujador 1 hacia adelante
Q2	BOOL	%QX0.5	Motor empujador 1 hacia atrás
Q3	BOOL	%QX0.6	Motor empujador 2 hacia adelante
Q4	BOOL	%QX0.7	Motor empujador 2 hacia atrás
Q5	BOOL	%QX1.0	Motor cinta transportadora de alimentación
Q6	BOOL	%QX1.1	Motor cinta transportadora fresadora
Q7	BOOL	%QX1.2	Motor fresadora
Q8	BOOL	%QX1.3	Motor cinta transportadora taladradora
Q9	BOOL	%QX1.4	Motor taladradora
Q10	BOOL	%QX1.5	Motor cinta transportadora salida
Q11	BOOL	%QX1.6	Habilitar sensores y empujadores
I1	BOOL	%IX0.0	Final de carrera frontal del empujador 1
I2	BOOL	%IX0.1	Final de carrera trasera del empujador 1
I3	BOOL	%IX0.2	Final de carrera frontal del empujador 2
I4	BOOL	%IX0.3	Final de carrera trasera del empujador 2
I5	BOOL	%IX0.4	Fototransistor empujador 1
I6	BOOL	%IX0.5	Fototransistor fresadora

I7	BOOL	%IX0.6	Fototransistor estación de carga
I8	BOOL	%IX0.7	Fototransistor taladradora
I9	BOOL	%IX1.0	Fototransistor cinta transportadora salida
BotonQ1	BOOL	%MX1.0	Activa Q1
BotonQ2	BOOL	%MX1.1	Activa Q2
BotonQ3	BOOL	%MX1.2	Activa Q3
BotonQ4	BOOL	%MX1.3	Activa Q4
BotonQ5	BOOL	%MX1.4	Activa Q5
BotonQ6	BOOL	%MX1.5	Activa Q6
BotonQ7	BOOL	%MX1.6	Activa Q7
BotonQ8	BOOL	%MX1.7	Activa Q8
BotonQ9	BOOL	%MX2.0	Activa Q9
BotonQ10	BOOL	%MX2.1	Activa Q10
INICIAR	BOOL	%MX0.0	Botón para arrancar el proceso
NextStep	BOOL	%MX0.3	Habilita el siguiente paso (según Paso a paso o Automático)
BrazoALineaMecanizado	BOOL	%MX0.5	El Robot Manipulador carga una pieza para depositar en la Linea de Mecanizado
Emergencia	BOOL	%MX0.4	Activa el Modo Emergencia
Plataforma1Cargada	BOOL	%MX0.7	La Plataforma 1 ha sido cargada
Plataforma2Cargada	BOOL	%MX2.2	La Plataforma 2 ha sido cargada
PiezaPrincipioCinta4	BOOL	%MX2.3	Existe una pieza al principio de la Cinta 4
PiezaPrincipioCinta2	BOOL		Existe una pieza al principio de la Cinta 2
LMAceptaPieza	BOOL	%MX0.6	El Robot Manipulador puede dejar su pieza
TiempoTaladrado	TIME		Tiempo de procesado en la cinta 2
TiempoFresado	TIME		Tiempo de procesado en la cinta 3
RED	BOOL	%MX3.0	Existe una pieza roja en Cámara
WHITE	BOOL	%MX3.1	Existe una pieza blanca en Cámara
BLUE	BOOL	%MX3.2	Existe una pieza azul en Cámara
TiempoFresadoREDOnline	DWORD	%MD10	Tiempo en el fresado para una pieza roja en formato DWORD
TiempoFresadoRED	TIME		Tiempo en el fresado para una pieza roja convertido a TIME
TiempoFresadoWHITEOnline	DWORD	%MD11	Tiempo en el fresado para una pieza blanca en formato DWORD
TiempoFresadoWHITE	TIME		Tiempo en el fresado para una pieza blanca convertido a TIME
TiempoFresadoBLUEOnline	DWORD	%MD12	Tiempo en el fresado para una pieza azul en formato DWORD
TiempoFresadoBLUE	TIME		Tiempo en el fresado para una pieza azul convertido a TIME
TiempoTaladradoREDOnline	DWORD	%MD13	Tiempo en el taladrado para una pieza roja en formato DWORD
TiempoTaladradoRED	TIME		Tiempo en el taladrado para una pieza roja convertido a TIME
TIempoTaladradoWHITEOnline	DWORD	%MD14	Tiempo en el taladrado para una pieza blanca en formato DWORD
TIempoTaladradoWHITE	TIME		Tiempo en el taladrado para una pieza blanca convertido a TIME
TIempoTaladradoBLUEOnline	DWORD	%MD15	Tiempo en el taladrado para una pieza azul en formato DWORD
TIempoTaladradoBLUE	TIME		Tiempo en el taladrado para una pieza azul convertido a TIME

BrazoACamara	BOOL	%MX0.2	El brazo está bajando a por una pieza en la cámara (activa captura de color)
PiezaFinalLineaMecanizado	BOOL	%MX4.7	Existe una pieza al final de toda la línea industrial
PiezasInLM	WORD	%MW7	Número de piezas dentro
PiezaPrincipioCinta1	BOOL	%MX4.0	Existe una pieza al principio de la Cinta 1
PiezaFinalCinta1	BOOL	%MX4.1	Existe una pieza al final de la Cinta 1
PiezaCinta2	BOOL	%MX4.2	Existe una pieza en la Cinta 2
PiezaCinta3	BOOL	%MX4.3	Existe una pieza en la Cinta 3
CodigoAlarma	INT	%MW5	Código de alarma para la interfaz visual
ResetContadores	BOOL	%MX4.4	Botón para resetear todos los contadores
PiezaDejada	BOOL		La pieza ha sido dejada correctamente
PiezaEnFresado	BOOL	%MX4.5	Testigo que indica que se está fresando una pieza en la Cinta 2
PiezaEnTaladrado	BOOL	%MX4.6	Testigo que indica que se está taladrando una pieza en la Cinta 3

Tabla 3. Variables globales utilizadas en la programación del autómata Schneider

3.4.2. Programas principales

La programación mediante SoMachine permite la realización de diagramas GRAFCET para los *Programas Principales*. En este caso quedan 6 programas principales individuales, que requerirán de variables para conectar unos con otros. Los programas son básicos comparados con los otros autómatas y con acciones básicas. La sencillez en el programa de *Acciones* es prueba de ello.

Cabe destacar el GRAFCET respectivo a la *Cinta 1*, ya que tiene una estructura no-lineal, y es que cuenta con 3 caminos distintos a recorrer. Esto es así debido a que nos interesa una etapa en la que la cinta se pregunte si el *Robot Manipulador* quiere y puede dejar una pieza en ella, y posteriormente realice ya un movimiento dependiendo de la posición de las piezas que pueda tener sobre la *Cinta 1*.

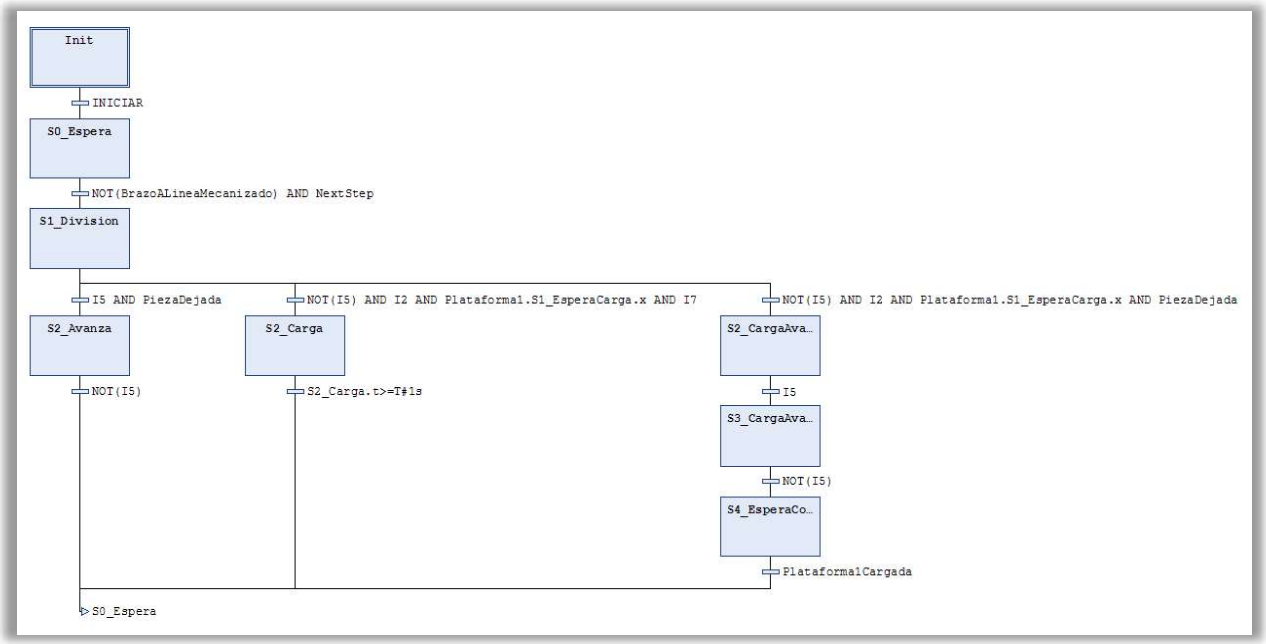


Figura 47. Captura del programa Cinta1 para el autómata Schneider

Este sistema también nos permite tener una única etapa en la que la transición está dominada por la variable *NextStep*, que controla el *Modo Automático/Modo Paso A Paso*. Necesitamos crear una etapa auxiliar en el tercer camino para evitar que exista una "realimentación" rápida y provoque que el sistema intente introducir dos piezas en la *Plataforma1*.

Esta realimentación es debida a que el flanco de bajada de la etapa activa la variable que hace que la *Plataforma1* ya no acepte pieza, *Plataforma1Cargada*, pero esto no surge efecto hasta que se ejecute el programa *Variables*, y por eso sin esta etapa el diagrama se "realimenta" (esta vez pasando por el segundo camino), porque detecta que la *Plataforma1* sigue esperando una carga.

Es importante puntualizar que con el desarrollo de esta *Cinta 1* y la *Cinta 4* se consigue soportar dos piezas en sus grafcets, provocando que el sistema global sea capaz de aguantar más piezas en su interior.

La detección de pieza en el inicio de una forma parecida a la *Estación Multiproceso,* aunque con claras diferencias. En este caso sí se incluye un fotorreceptor capaz de determinar si hay una pieza al principio, pero la variable que domina en *Cinta1* para que se inicie el arranque es *PiezaDejada*, gestionada en *Variables*, que está retrasada 2 segundos con respecto a la llegada de una pieza en el sensor. El retraso es debido a la necesidad de que la *Cinta 1* permita a la *Estación Robot Manipulador* dejar la pieza sin que la cinta inicie el movimiento nada más la detecte. También se ha tenido en cuenta el efecto del posible *Modo Paso A Paso* activo, ya que podría hacer inútil el temporizador si el *Robot Manipulador* no ha podido regresar de dejar la pieza a una posición segura.

Figura 48. Gestión de la variable PiezaDejada, dentro del programa Variables para el autómata Schneider

3.4.3. FIFOs

El funcionamiento del sistema de guardado del tipo de pieza es prácticamente idéntico al del apartado correspondiente para la *Estación Multiproceso*, 3.3.3. FIFOs, para las variables temporales, aunque el funcionamiento del programa ha requerido de realizar pequeñas variaciones para terminar con un funcionamiento igual al del autómata Siemens.

Al intentar utilizar una misma función dos veces distintas (como con el Siemens), el programa parecía no comprender que las funciones tienen cada una sus propias variables internas, y causaba errores que no permitían el correcto funcionamiento. Como una solución fácil y rápida al problema, se ha optado por "copiar" idénticamente el programa, y tener dos veces el mismo programa (*FIFO_Fresado* y *FIFO_Taladrado*) orientado cada uno a cada uno de los procesos.

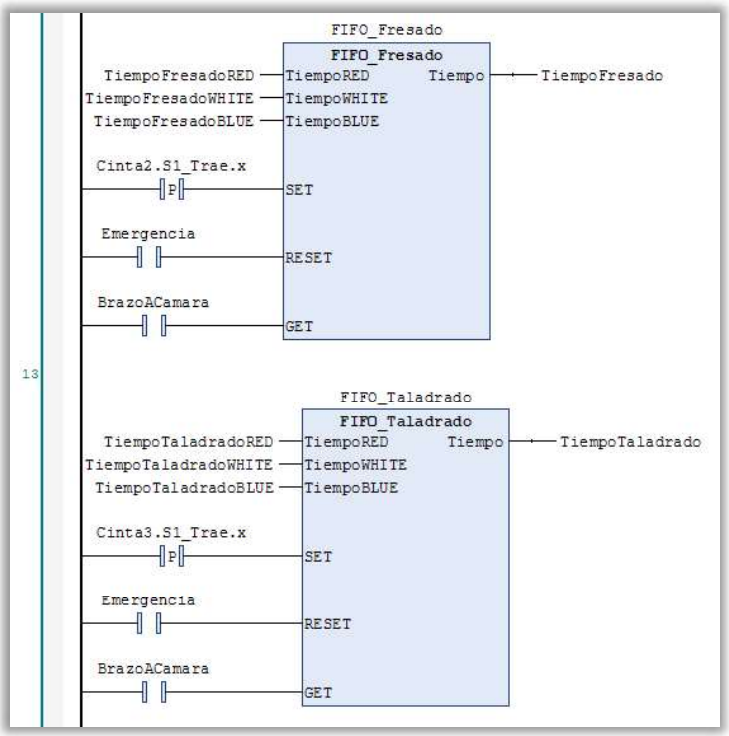


Figura 49. Captura del programa Variables para el autómata Schneider, donde se muestra la implementación de los FIFO

En esta Estación los 2 procesos que se dan son comparables al procesado de la *Estación Multiproceso*, y no requieren del tratamiento que se le da al caso especial de la temperatura. Si se requiere más información sobre el funcionamiento, se puede consultar en el apartado correspondiente indicado al inicio de este apartado.

3.4.4. Testigos

Los testigos programados utilizan considerablemente los sensores y acciones en la *Cinta1* y la *Cinta4*, y aprovechan las variables de *Plataforma1Cargada*, *Plataforma2Cargada* y *PiezaPrincipioCinta4* para reutilizarlas como testigos. En el caso de las cintas 2 y 3 se utilizan las etapas para determinar que existe pieza en ellas. Además, se utilizan testigos que señalan si los procesos de *Fresado* y *Taladrado* están activos o no.

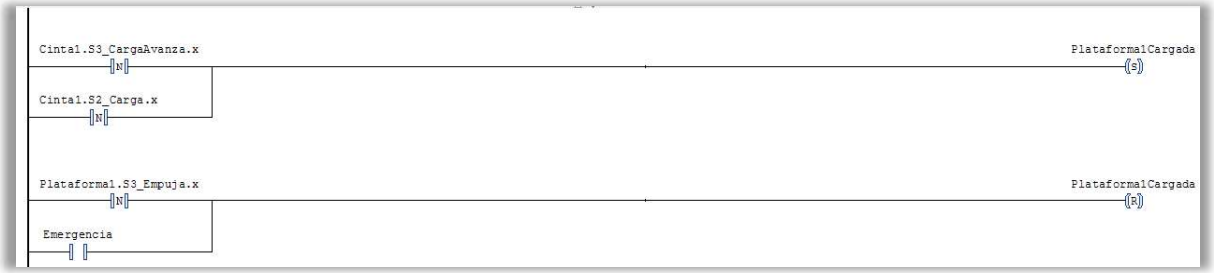


Figura 50. Captura de la gestión de Plataforma1Cargada, realizada en el programa Variables para el autómata Schneider

3.4.5. Alarmas

Como está explicado en 3.2 Consideraciones generales, y en el mismo apartado del autómata Siemens, se utilizan temporizadores para determinar que ciertas acciones están activas demasiado tiempo. De nuevo se utiliza un *Codificador* para enviar un código entero de alarma al Servidor OPC.

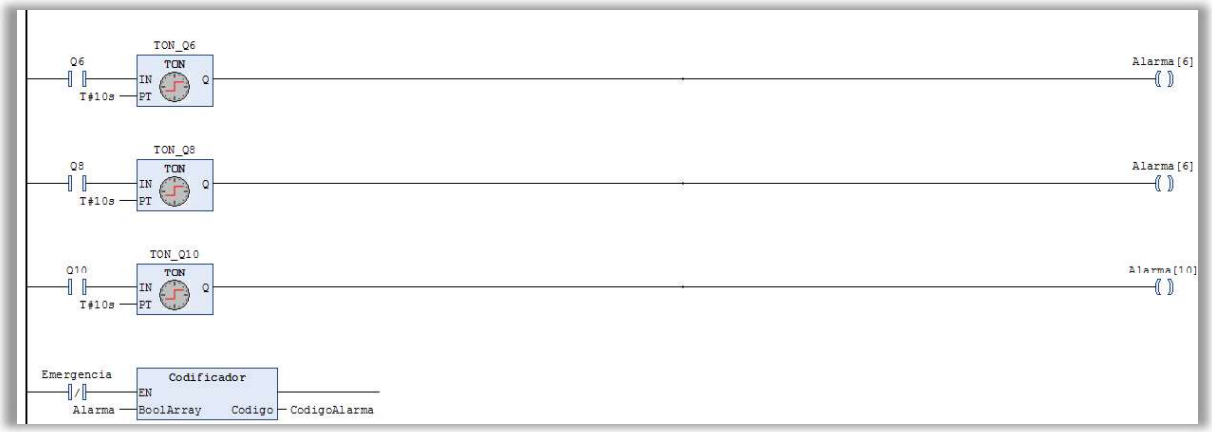


Figura 51. Captura del programa Alarmas para el autómata Schneider

El funcionamiento es idéntico al del autómata Siemens en el programa *Alarmas*. Se genera un vector booleano temporal del programa, donde se guarda qué alarmas están activas. Posteriormente el *Codificador* escribe en su salida qué "casilla" de este vector está activa. Este código será utilizado en la interfaz visual para identificarlo como una alarma concreta.

3.4.6. Contadores

En este caso, la programación del autómata Schneider únicamente incluirá un contador predefinido, *CTUD*, que contará cuantas piezas se encuentran dentro de la *Estación Línea Mecanizado*, ya que las cuentas a nivel global son llevadas por el autómata Siemens. Por supuesto también se reseteará con la *Emergencia* y el botón de *ResetContadores*, al igual que con el autómata Siemens.

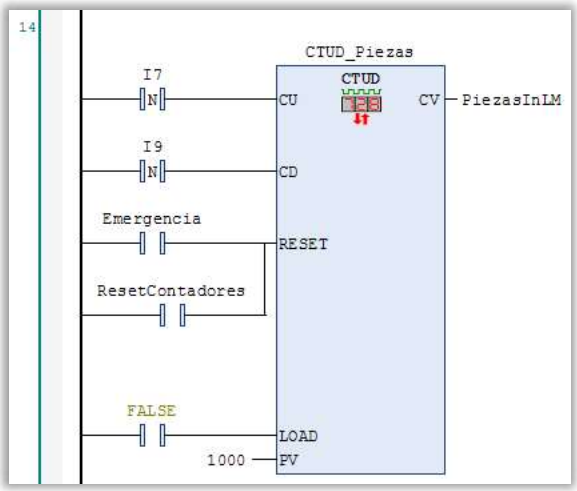


Figura 52. Captura del programa Variables para el autómata Schneider, mostrando la implementación del CTUD

3.4.7. Conversores

Debido a diversos problemas expuestos en el apartado 3.6. Configuración del Servidor OPC, la comprensión de las direcciones de memoria queda confusa a la hora de registrar las variables temporales en el servidor. Es por ello que se ha tomado la sencilla decisión de que los valores que se van a compartir en el servidor OPC sean de palabras dobles, que se encuentran fácilmente realizando una pequeña conversión en la configuración del OPC.

Esta solución implica la necesidad de unos conversores de palabras dobles a variables de tiempo en SoMachine, *DWORD_TO_TIME*, para poder utilizar la información en los programas FIFO, y, al tratarse de una cantidad considerable, se han concretado en un programa particular llamado *Conversores*.

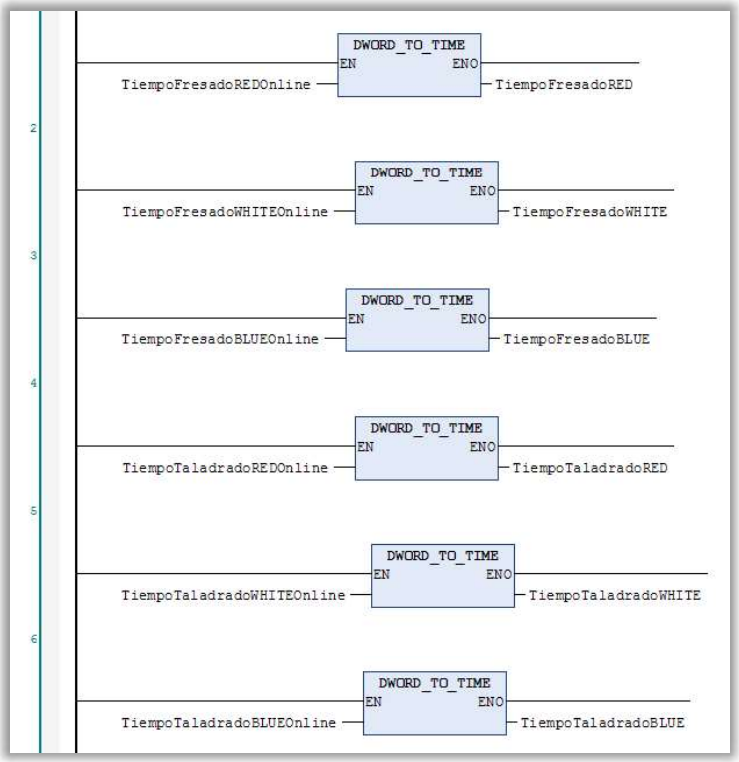


Figura 53. Captura del programa Conversores para el autómata Schneider

3.4.8. Modo Emergencia

El software SoMachine permite configurar los programas en SFC para activar variables especiales con las que se puede controlar distintos aspectos de los SFC. En nuestro caso, la variable que se adapta a nuestra necesidad es *SFCInit*, que reestablece todos los pasos y acciones cuando se activa, dejando como única etapa activa al paso inicial. Cuando esta variable es desactivada, el programa prosigue su curso normal desde esta etapa inicial.

Figura 54. Gestión de la Emergencia para los programas principales, en el programa Variables para el autómata Schneider

La variable *SFCInit* se tiene que configurar para cada programa, y además se ha de declarar como *VAR_INPUT* en las variables locales del programa, para poder ser utilizada en otros programas, como la activación del *Modo Emergencia* que se gestiona en el programa *Variables*.

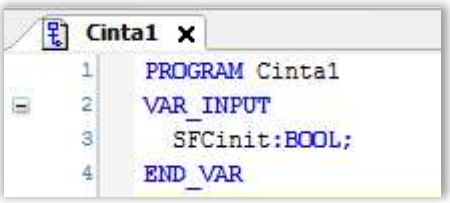


Figura 55. Captura de la declaración de la variable local SFCInit en la Cinta1 para el autómata Schneider

Cabe decir que esta función sirve debido a que las etapas iniciales en nuestros programas secuenciales son etapas de espera al botón inicial, y no tienen ninguna repercusión en alguna acción o variable que pudiera causar problemas en el *Modo Emergencia*.

3.5. DISEÑO E IMPLEMENTACIÓN DE LOS AUTOMATISMOS NECESARIOS PARA LA ESTACIÓN ROBOT MANIPULADOR

Se ha escogido el autómata Omron para la programación de la *Estación Robot Manipulador* debido a la accesibilidad y facilidad en la programación de entradas con lectura rápida. Esto es necesario puesto que el robot funciona con pseudo-coordenadas, basadas en los pulsos de los encoders instalados en los motores para los movimientos de *Giro*, *Vertical*, y *Horizontal*.

Es necesario contar estos pulsos para poder conocer de forma muy aproximada la posición del robot en cada uno de sus grados de libertad. Los pulsos de los encoders tienen frecuencias muy rápidas, y no configurar la lectura rápida haría el sistema de movimiento mucho más impreciso.

Para la programación del autómata Omron se ha requerido de usar el software CX-Programmer.

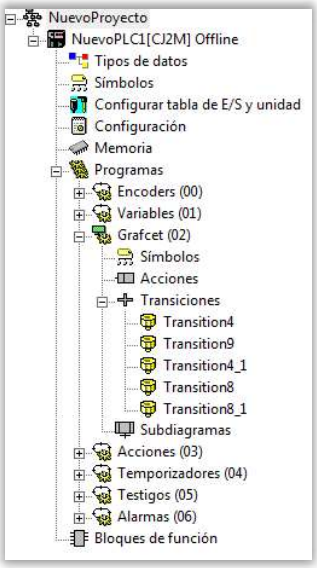


Figura 56. Captura de la vista general del proyecto para el autómata Omron

3.5.1. Entradas, salidas y variables globales importantes

A continuación se presentan las entradas y las salidas que existen para la *Estación Robot Manipulador*, junto a las variables globales más importantes que ayudan a tener un concepto global más claro de la programación realizada.

Variable	*Tipo*	*Dirección*	*Comentario*
B1	BOOL	0.00	Pulsos encoder movimiento vertical
B3	BOOL	0.01	Pulsos encoder movimiento horizontal
B5	BOOL	0.02	Pulsos encoder movimiento giratorio
I1	BOOL	0.04	Final de carrera referencia vertical
I2	BOOL	0.05	Final de carrera referencia horizontal
I3	BOOL	0.06	Final de carrera referencia giro
Q1	BOOL	1.00	Motor movimiento vertical arriba
Q2	BOOL	1.01	Motor movimiento vertical abajo
Q3	BOOL	1.02	Motor movimiento retroceso horizontal
Q4	BOOL	1.03	Motor movimiento avance horizontal
Q5	BOOL	1.04	Motor movimiento giro horario
Q6	BOOL	1.05	Motor movimiento giro antihorario
Q7	BOOL	1.06	Habilitar compresor
Q8	BOOL	1.07	Succión ventosa
Q9	BOOL	1.08	Habilitar señales de entrada del proceso
INICIAR	BOOL	40.00	Inicia el sistema
NextStep	BOOL	40.01	Controla el paso a paso en las transiciones
Emergencia	BOOL	40.02	Botón que para el sistema y espera a su reinicio
PeligroCamara	BOOL	45.00	Existe peligro de colisión con el humano en la zona Cámara
PiezaFinalMultiproceso	BOOL	45.01	Existe pieza al final de la estación multiproceso para ser recogida
LMAceptaPieza	BOOL	45.02	LineaMecanizado permite la llegada de una nueva pieza
MPAceptaPieza	BOOL	45.03	Estación Multiproceso acepta una pieza en la entrada
BrazoALineaMecanizado	BOOL	45.04	Llevando una pieza a la Estación Línea Mecanizado
BrazoACamara	BOOL	45.07	El brazo está bajando a por una pieza en la cámara (activa captura de color)
CoordenadaV	INT	50	Coordenadas verticales del encoder
CoordenadaH	INT	51	Coordenadas horizontales del encoder
CoordenadaG	INT	52	Coordenadas de giro del encoder
CoordenadaH1	INT	53	Coordenada del primer movimiento del brazo
CoordenadaV1	INT	54	Coordenada vertical primera que ha de visitar el brazo
CoordenadaH2	INT	55	Coordenada horizontal de la segunda posición
CoordenadaV2	INT	56	Coordenada vertical de la segunda posición
CoordenadaG1	INT	57	Coordenada del primer giro a realizar
CoordenadaG2	INT	58	Coordenada de la segunda posición de giro
Camino1	BOOL	60.00	Camino de Multiproceso a LineaMecanizado
Camino2	BOOL	60.01	Camino de Cámara a Multiproceso
TimeS5	BOOL	60.02	Activación del Temporizador 1
TimeS9	BOOL	60.03	Activación del Temporizador 20
RED	BOOL	65.00	Existe una pieza roja en Cámara

WHITE	BOOL	65.01	Existe una pieza blanca en Cámara
BLUE	BOOL	65.02	Existe una pieza azul en Cámara
BotonQ1	BOOL	70.00	Activa Q1
BotonQ2	BOOL	70.01	Activa Q2
BotonQ3	BOOL	70.02	Activa Q3
BotonQ4	BOOL	70.03	Activa Q4
BotonQ5	BOOL	70.04	Activa Q5
BotonQ6	BOOL	70.05	Activa Q6
BotonQ7	BOOL	70.06	Activa Q7
BotonQ8	BOOL	70.07	Boton que suelta la succión de la ventosa
MovimientoFinalMultiproceso	BOOL	80.00	Testigo que indica el movimiento del brazo hacia el final de la Estación Multiproceso para recoger una pieza
MovimientoCamara	BOOL	80.01	Testigo que indica el movimiento del brazo hacia la Cámara para recoger una pieza
SuccionandoFinalMultiproceso	BOOL	80.02	Testigo que indica la succión del brazo robot para coger una pieza al final de la Estación Multiproceso
SuccionandoCamara	BOOL	80.03	Testigo que indica la succión del brazo robot para coger una pieza en la Cámara
MovimientoLineaMecanizado	BOOL	80.04	Testigo que indica el movimiento del brazo hacia el principio de la Estación Línea Mecanizado
MovimientoMultiproceso	BOOL	80.05	Testigo que indica el movimiento del brazo hacia el principio de la Estación Multiproceso
Inicializacion	BOOL	80.06	Testigo que indica que el brazo está inicializándose
CodigoTestigos	INT	85	Código de testigos para la interfaz visual
CodigoAlarma	INT	90	Código de alarma para la interfaz visual

3.5.2. Programas Principales

El *Robot Manipulador* tiene una labor comunicativa muy importante, ha de indicar a la *Estación Línea Mecanizado* que tiene una pieza disponible para dejarla sobre su principio, y también ha de indicar a los autómatas que va a recoger una pieza en la Cámara y que deben "capturar" que tipo de pieza es (esto está explicado más detalladamente en 3.7.1. Programación de la Cámara).

No sólo tiene que avisar, sino que también ha de recibir si existe una pieza en la Cámara dispuesta a ser recogida, o si existe una pieza en el final de la *Estación Multiproceso*. También, por seguridad, debe asegurarse que la *Estación Multiproceso* y la *Estación Línea Mecanizado* aceptan que pueda dejar una pieza sobre su principio correspondiente.

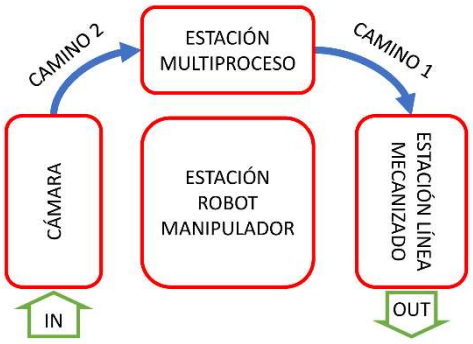


Figura 57. Esquema de posición de las estaciones y los caminos a realizar por la Estación Robot Manipulador

Esta estación ha de realizar dos caminos. Uno consta de llevar la pieza del final de la *Estación Multiproceso* al principio de la *Estación Línea Mecanizado*, y el otro llevar la pieza de la *Cámara* a la *Estación Multiproceso*. Pese a ello, el GRAFCET programado en SFC tan solo es la secuencia de un camino, y lo que se hace es cambiar las coordenadas de las posiciones a las que moverse según si se requiere de un camino u otro. Este aspecto se gestiona dentro del programa de *Variables*.

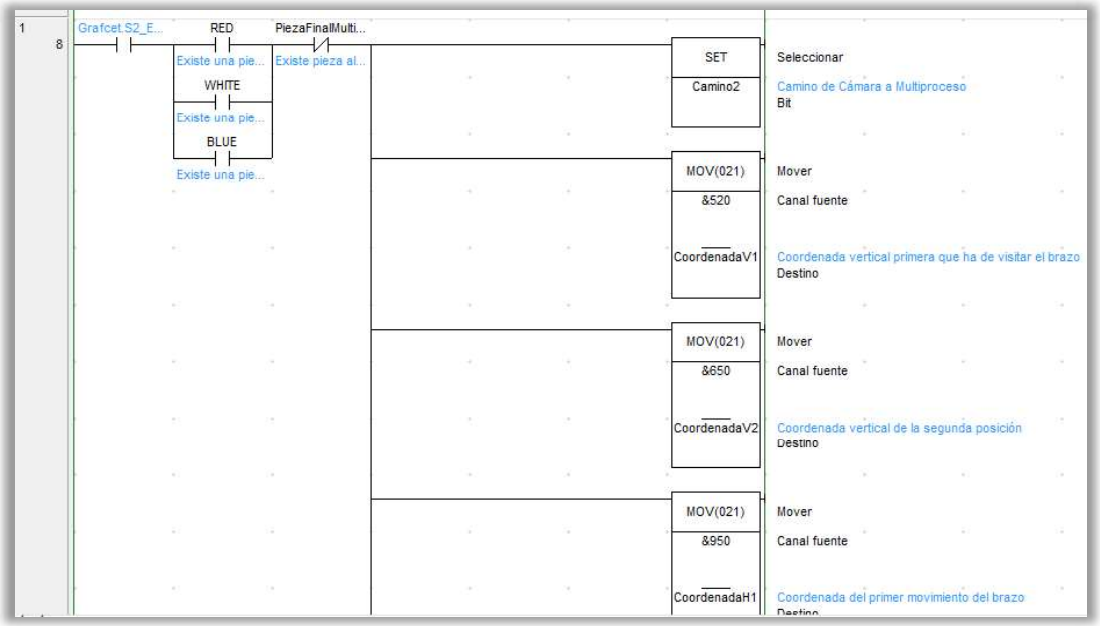


Figura 58. Parte de la programación de coordenadas para el Camino 2, en el programa Variables para el autómata Omron

Tanto el programa principal *Grafcet* como la posición física de la estación se ha escogido de forma que pueda regresar cada ciclo a las posiciones de los finales de carrera que tiene, que marcan sus puntos de inicio, y están programados para reinicializar los contadores evitando errores acumulados en la posición. Esto queda explicado en el siguiente apartado, referente a los encoders y su programación.

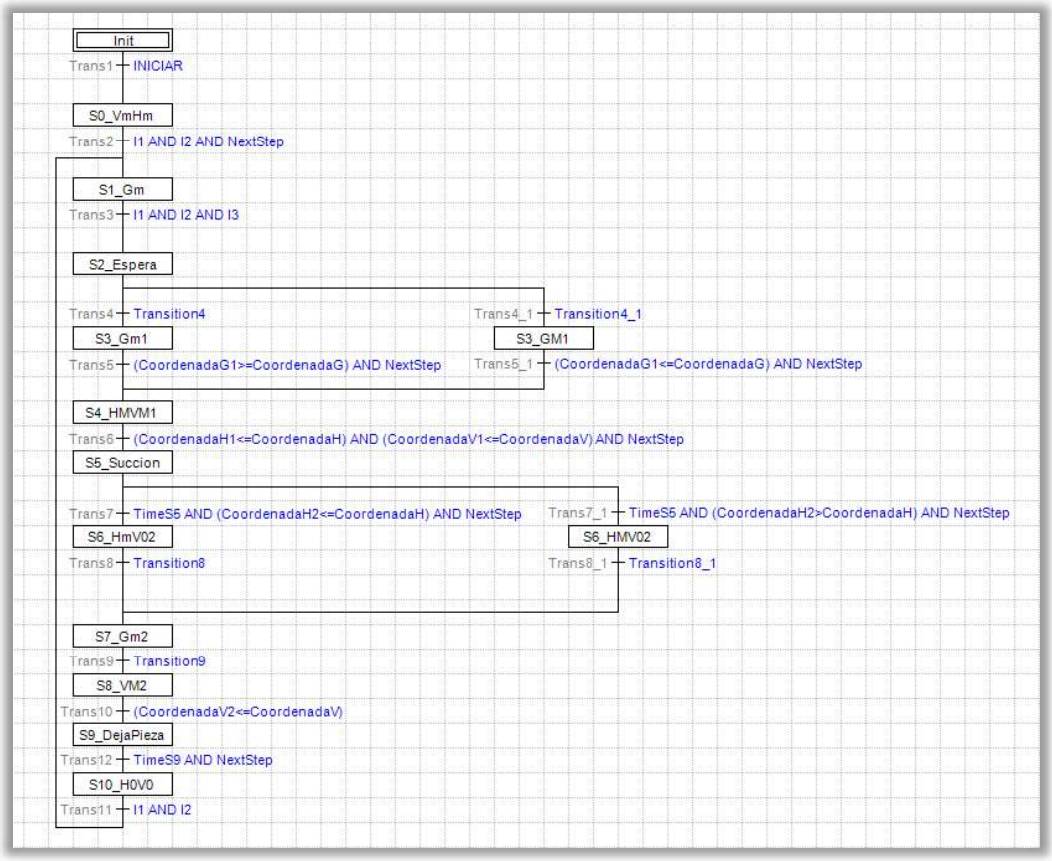


Figura 59. Captura del programa Grafcet para el autómata Omron

También cabe destacar el enfoque que se le ha dado al *Grafcet*. El relativamente gran tamaño del gráfico secuencial ha hecho utilizar un sistema en el cuál las etapas son, en cierta forma, simbólicas.

Esto es así debido a que hay varias etapas donde se realizan varios movimientos, que han de terminar individualmente. En una aplicación normal de un diagrama secuencial como el GRAFCET, lo habitual sería que si deseas activar acciones que terminen en momentos distintos respecto de las otras, establecieras una división del flujo del diagrama para "controlar" cada una de estas acciones individualmente.

Aunque un movimiento "A" haya llegado a la coordenada correspondiente a esa etapa, este debe parar aunque la etapa siga activa porque el movimiento "B" aún no ha llegado a su coordenada. Esto se consigue gracias a una programación del POU *Acciones* más compleja, que evita que las acciones sigan ocurriendo cuando hayan llegado al punto dado, de forma parecida a la programación de las acciones para evitar problemas con el *Modo Paso A Paso* (ver apartado 3.7.2. Programación relacionada con los autómatas, donde se remarca la importancia de la gestión del *Modo Paso A Paso*).

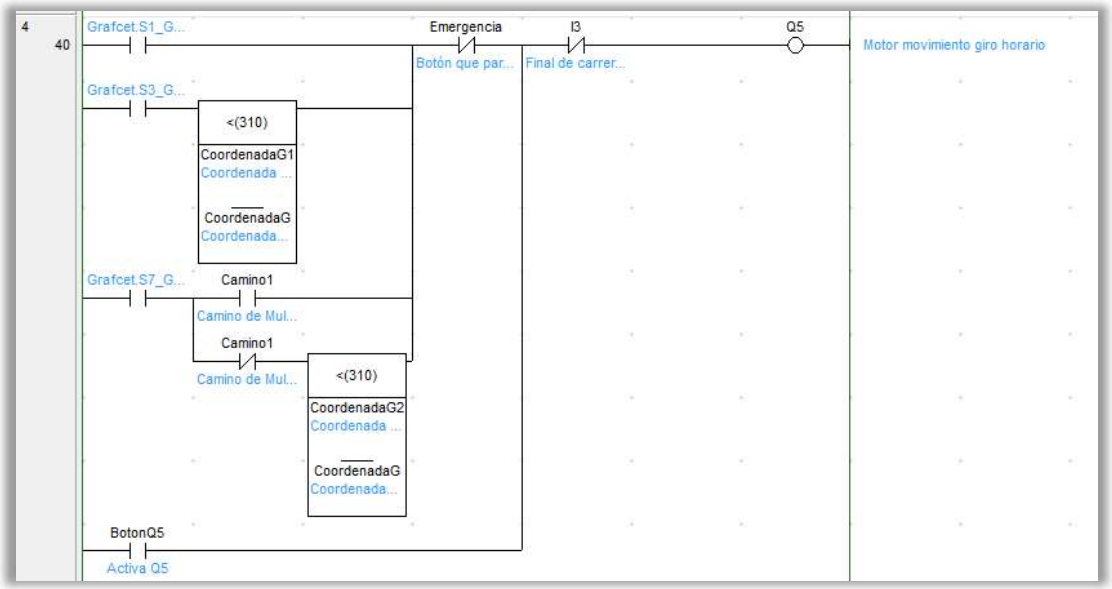


Figura 60. Captura de la acción Q5 en el programa Acciones para el autómata Omron

Si no se hubiera realizado así, habría que haber ampliado el diagrama Grafcet, y este hubiera complicado otras tareas tanto en la programación del momento como en las futuras modificaciones.

3.5.3. Encoders

Como ya se ha introducido antes, los encoders aportan una señal de pulsos muy rápida (cabe la necesidad de realizar la correcta configuración del autómata para que esa capaz de leerla con suficiente velocidad) con cierto movimiento realizado por los motores involucrados en cada uno de los desplazamientos del *Robot* Manipulador. Estos se pueden asemejar a los "pasos" humanos. Desde un punto inicial conocido, se puede guiar a una persona a ciertos puntos en el espacio de forma relativamente precisa si conocemos cuántos pasos necesita en determinadas direcciones. Este mismo enfoque se lleva a cabo, mediante unos contadores que cuentan los pulsos rápidos de los encoders hacia adelante o hacia detrás, dependiendo de si hemos arrancado la señal que activa el movimiento en una dirección o la contraria.

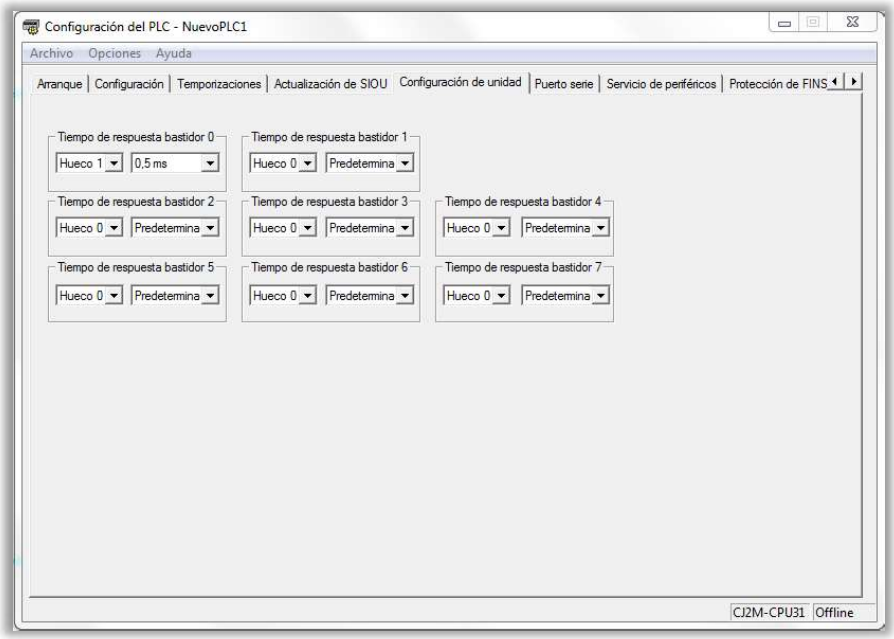


Figura 61. Configuración del Hueco 1 en el bastidor 0 para leer los pulsos rápidos de los encoders

Como ocurriría con los pasos humanos, en cuanto los movimientos se hacen demasiado largos, y con varias combinaciones, se empiezan a acumular errores que perjudican la posición a la que se termina llegando. Para ello, lo que se hace es, cada vez que es posible, llevar la posición del *Robot Manipulador* a sus coordenadas iniciales, marcadas por los finales de carrera. Por lo tanto, y como visto en 3.5.2. Programas Principales, se regresa a los puntos iniciales cada ciclo del *Grafcet*.

Estos finales de carrera se aprovechan para reinicializar los contadores de los encoders, y evitar ir acumulando los errores de precisión. De esta forma se consigue limpiar los errores y tener un proceso más preciso (aunque a costa de perder unos segundos en la inicialización en cada ciclo).

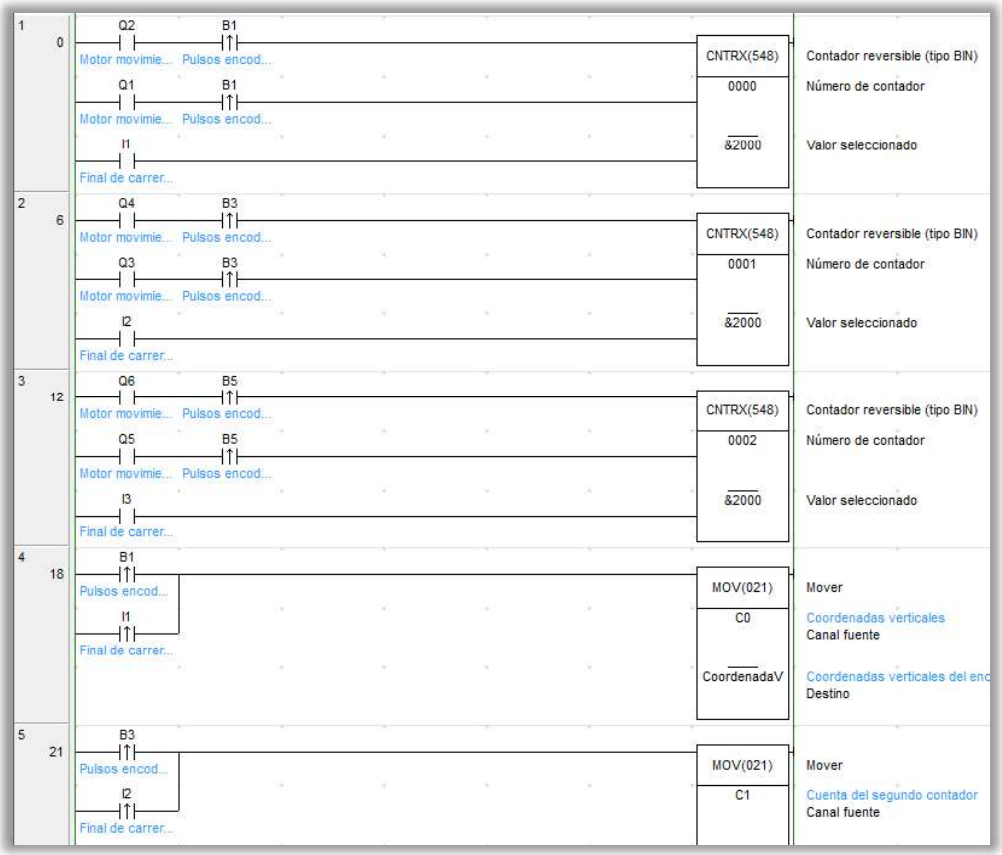


Figura 62. Captura de parte del programa Encoders para el autómata Omron

Cabe apuntar que en cierto punto del desarrollo del proyecto, se tuvo en consideración la posibilidad de que la estación *Robot Manipulador* sólo tuviera que regresar a la posición inicial del movimiento de giro en la realización del *Camino 1* (los movimientos vertical y horizontal regresan a la posición inicial en todos los ciclos de forma "natural", para evitar choques), ya que además se realizó la colocación de las estaciones de forma que coincidiera la posición de *Giro* inicial con la posición para dejar la pieza en la *Estación Línea Mecanizado*. Sin embargo, se tuvo que descartar esta opción debido a las imprecisiones que ocasionaba realizar cualquiera de los dos caminos sin empezar desde la posición inicial para todos sus movimientos.

3.5.4. Testigos

En este caso particular, se ha realizado un enfoque de los testigos distinto al de los otros autómatas, como se puede comprobar en el programa *Testigos*. Mientras que con los otros autómatas, los testigos servían esencialmente para determinar la posición de las piezas; en este caso, como se trata de un brazo robot, se ha creído más conveniente que desde la interfaz visual se avise al usuario de qué movimiento está haciendo, es decir, hacia dónde se dirige.

Para ello, se ha utilizado una variable entera, *CodigoTestigos*, que mediante la función *MOV* de CX-Programmer se le da cierto valor dependiendo de qué esté realizando. Posteriormente, y al igual que ocurre con las *Alarmas*, este código podrá ser utilizado en la interfaz visual para mostrar un mensaje sobre qué está ocurriendo.

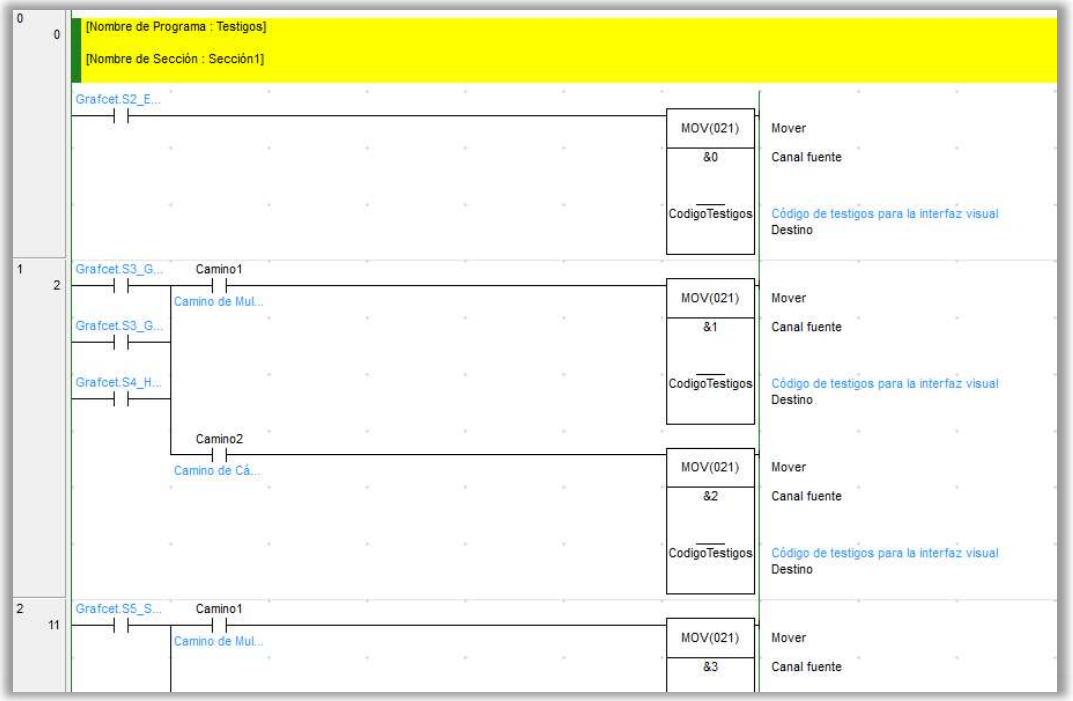


Figura 63. Captura de parte del programa Testigos para el autómata Omron

Como un testigo particular, se ha programado en la sección *Variables* el elemento *PeligroCamara*, que básicamente se utilizará en la interfaz como un "LED" que avisará de que la cámara puede producir colisión con el operario que pueda estar ocupando la zona de la *Cámara*.

3.5.5. Alarmas

Las alarmas, al igual que el resto de autómatas, se han basado en temporizadores enlazados a acciones, que permiten señalar si una acción está durando demasiado. En este caso hay dos programas implicados en las alarmas: *Alarmas* y *Temporizadores*.

Aunque *Temporizadores* incluye otros "timers" necesarios para el correcto funcionamiento del *Grafcet*, también incluye los temporizadores necesarios para *Alarmas*. El software CX-Programmer necesita por una parte crear los temporizadores, y por otra los interruptores enlazados a estos temporizadores, y es esta la razón por la cual se ha usado el programa *Temporizadores*.

El programa *Alarmas* aprovecha estos interruptores, y a diferencia de los otros autómatas, donde existía una función llamada *Codificador*, aquí directamente se utiliza la función nativa *MOV*, que coloca el valor de una variable A sobre una B, para escribir el código sobre la variable *CodigoAlarma*.

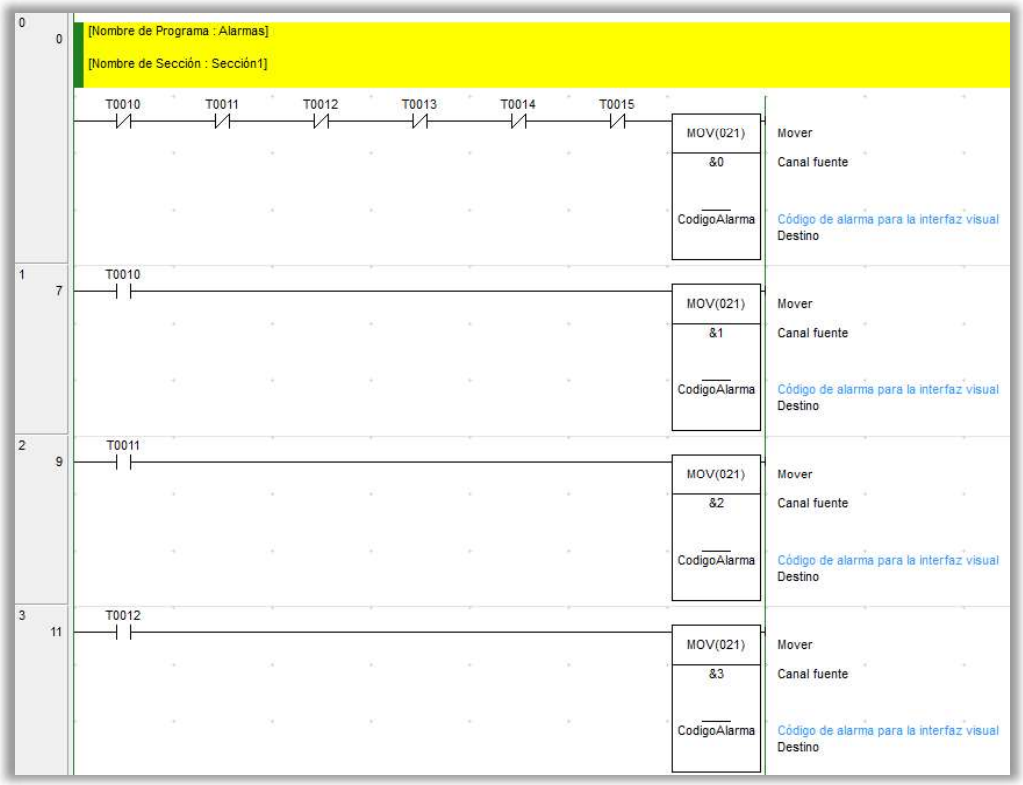


Figura 64. Captura de parte del programa Alarmas, para el autómata Omron

3.5.6. Modo Emergencia

La gestión del *Modo Emergencia* se realiza en este caso con la desactivación de las etapas del *Grafcet* mediante la función *SFCOFF*, que se activa con el flanco de subida de *Emergencia*, y después la reiniciación de este mediante la función *SFCON*.

Figura 65. Gestión del Modo Emergencia para el programa Grafcet, en el programa Variables para el autómata Omron

Por otro lado, y como se ha explicado anteriormente, este modo permite el manejo individual de ciertas acciones de la Estación, que por lo tanto habrá que proteger para evitar que se produzcan roturas por sobrepasar los límites del desplazamiento seguro de los elementos.

Como se ha visto con las otras estaciones comandadas por los otros autómatas, generalmente existen finales de carrera que nos sirven para determinar que la acción ya no debe poder producirse más para no ocasionar peligro como, por ejemplo, una plataforma llegando al final de su recorrido. Sin embargo, esta Estación solo tiene finales de carrera en sus puntos iniciales, y se ha requerido bloquear el movimiento de las acciones utilizando los propios contadores de coordenadas.

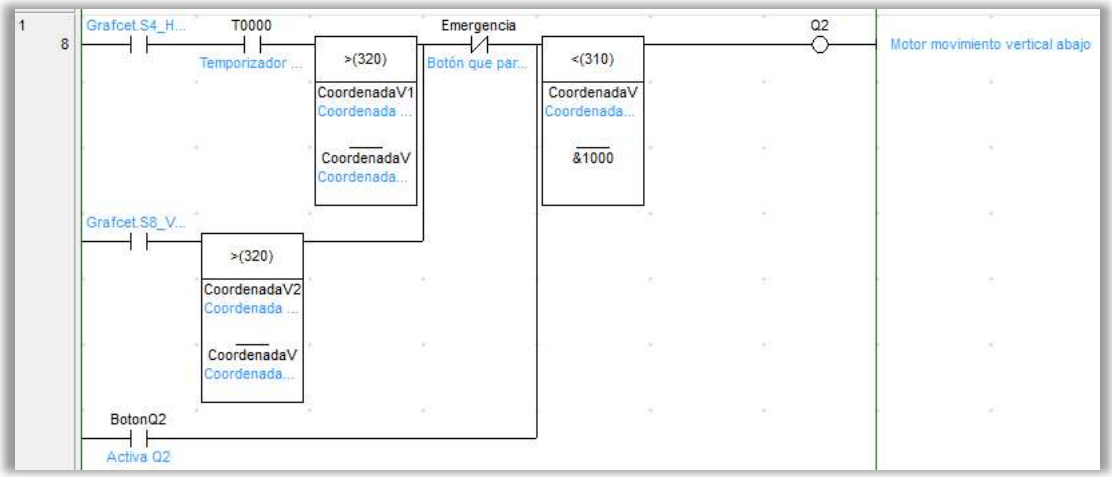


Figura 66. Captura del programa Acciones para el autómata Omron, mostrando la acción Q2

Al igual que ocurre con la *Estación Multiproceso*, existe un proceso de succión que se ha programado tal para que no se desactive en caso de entrar en el *Modo Emergencia*. El botón que maneja la acción de *Succión* en este modo, *BotonQ8*, funciona como desactivador de esta, para así asegurarnos que no existirá un desprendimiento inesperado de pieza al activar el *Modo Emergencia*.

3.6. CONFIGURACIÓN DEL SERVIDOR OPC

Existen diversas formas y sistemas para la comunicación entre autómatas conformando una red global de comunicación, aunque se ha escogido el sistema del servidor OPC por la facilidad, y la habitualidad de la industria a tener una red Ethernet local instalada, y la fácil aplicación de este sistema para situaciones como las de este proyecto, donde encontramos autómatas de distintas marcas con codificaciones diferentes y el uso de software de visualización como LabView, con gran adaptación a este tipo de comunicación en red.

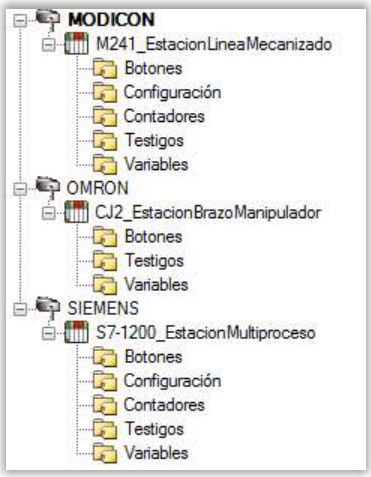


Figura 67. Captura de la organización general programada en el servidor OPC

El software KEPServerEX, de Kepware, es un software simple pero efectivo para la creación de servidores OPC, donde podemos organizar las variables en carpetas. Se debe tener

instalados los drivers correspondientes a cada autómata necesario para la conexión correcta con el servidor OPC.

IMPORTANTE: Revisar como permitir la conexión de lectura y escritura on-line para los distintos autómatas en sus respectivos manuales, así como asegurar una correcta configuración previa de estos.

3.6.1. SIEMENS

Para la correcta conexión del autómata Siemens al servidor OPC, se debe de utilizar el driver "Siemens TCP/IP Ethernet" para crear el canal, y se debe tener correctamente configurado el autómata, así como asignar correctamente la dirección IP tanto en el TIA Portal como en el KEPServerEX, y utilizar el modelo S7-1200.

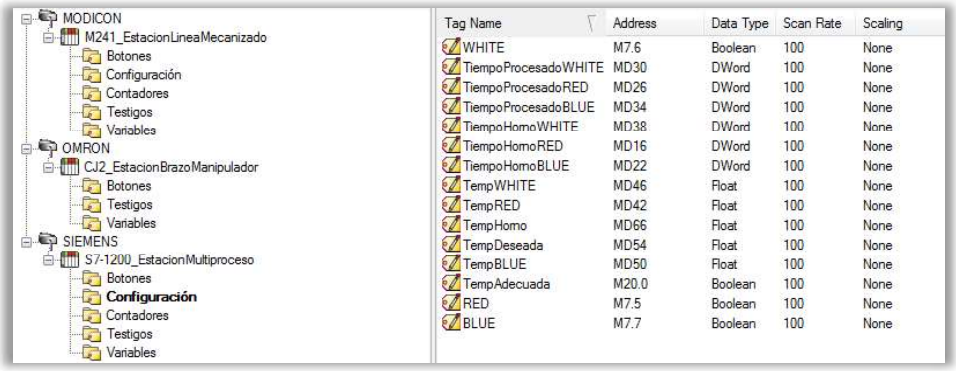


Figura 68. Ejemplo de "tags" creados para el autómata Siemens, agrupados en la carpeta Configuración

3.6.2. MODICON (SCHNEIDER)

El autómata Schneider fue el que más problemas ocasionó a la hora de programar el servidor OPC. Se creó el canal con el driver Modbus TCP/IP Ethernet, y se escogió el dispositivo con modelo Applicom y submodelo TSX Premium, que, pese a que el autómata no se corresponde con estas características, sí que resulta compatible con la configuración.

Se realizó de esta forma porque al parecer no se podía establecer una conexión correcta con la configuración teóricamente idónea para el modelo de autómata. Al elegir este modelo y submodelo distintos a los teóricamente idóneos, simplemente había que tener en cuenta que, a la hora de generar los Tags de las variables, la numeración iba a ser distinta.

La conversión a realizar se debe a que el autómata identifica direcciones cada 8 bits, y el servidor entiende que cada dirección tiene 16 bits. Es decir, lo que para el autómata es %MX1.0, para el servidor OPC es %MW0.8, y así para todos.

3.6.3. OMRON

En el caso del autómata Omron se requiere crear un canal con el driver Omron FINS Ethernet. El dispositivo insertado es un modelo CJ2, y hay que identificar correctamente mediante la IP del dispositivo. La estructura del software CX-Programmer es delicada con la configuración del autómata, y hay que tener especial cuidado en la asignación de la IP. Las variables están alojadas en el espacio de memoria conocido como CIO.

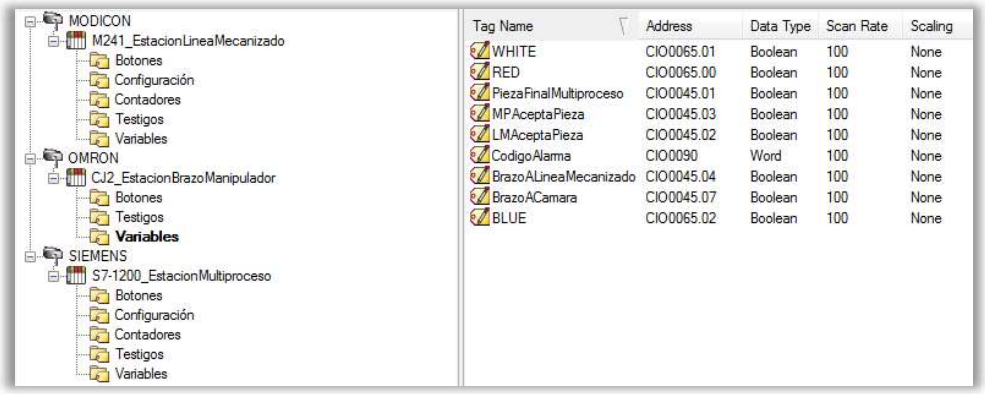


Figura 69. Ejemplo de "tags" creados para el autómata Omron, agrupados en la carpeta Variables

3.7. PROGRAMACIÓN DE LA INTERFAZ VISUAL, COMUNICACIÓN ENTRE AUTÓMATAS Y CÁMARA

Se ha elegido la herramienta LabView para diversas tareas necesarias en la programación global del sistema, englobando estas en dos grandes apartados:

- Programación de la Cámara
- Programación relacionada con los autómatas

Se ha utilizado este software por su gran versatilidad, personalización, manejo fácil con drivers de terceros y funcionamiento idóneo con servidores OPC; incluyendo una parte visual y un diagrama de bloques para la programación.

3.7.1. Programación de la Cámara

La programación consiste en un bucle "while" donde vamos capturando imágenes recibidas por la *Cámara*, y mediante LabView podemos seleccionar qué zona de la imagen vamos a utilizar para leer el espectro de color. Lo que se hace a continuación es comparar este espectro con el rango de los colores que nosotros queremos capturar.

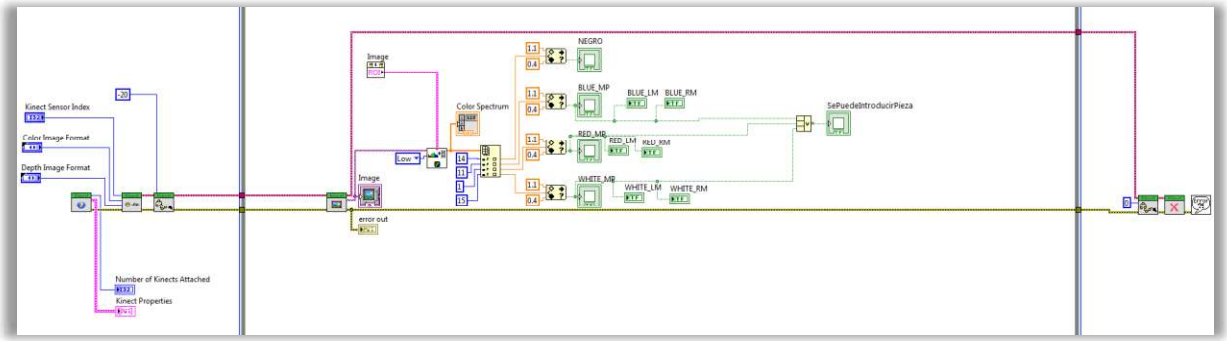


Figura 70. Programación de la detección del color para la cámara Kinect en el programa LabView

Si este espectro de color coincide con un rango determinado, activamos la señal correspondiente, que activará 3 "LEDs" que están conectados a la variable correspondiente al color determinado de cada uno de los 3 autómatas. Obviamente, de estos 3 LEDs solo se muestra uno de ellos en la interfaz visual.

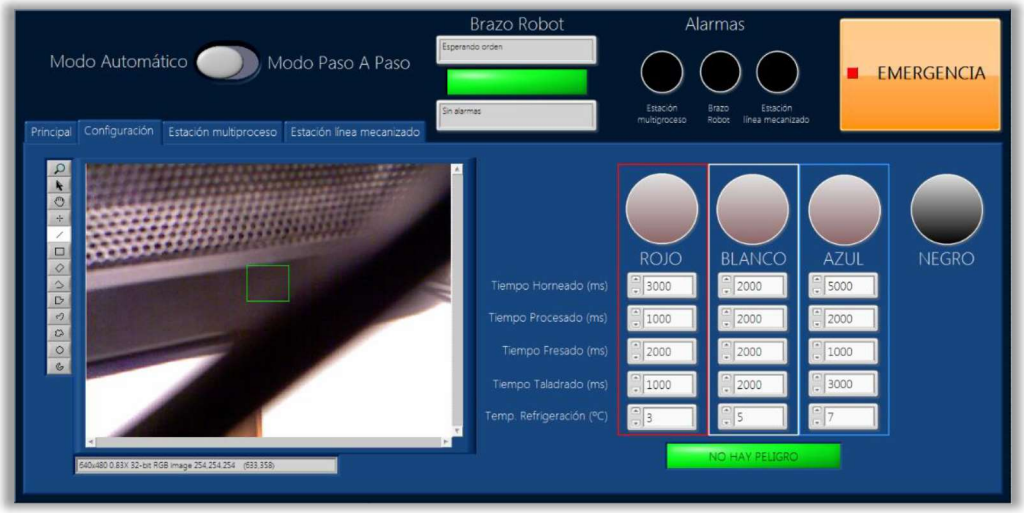


Figura 71. Interfaz de usuario, pestaña de Configuración

Puede ocurrir que en extrañas circunstancias se detecten dos colores a la vez, pero como se puede comprobar en la programación de los autómatas, se ha tenido en cuenta a la hora de el momento en el que reconocer el color de la pieza.

Estos "errores" de detección de más de un color solo ocurren (normalmente) en el momento de colocación de la pieza en la zona de detección de color, ya que puede "contaminarse" la imagen con algún otro color del operario o la máquina que deje la pieza en esa posición; por ello no se "captura" en los *FIFOs* el valor del color hasta que el *Robot Manipulador* esté a punto de recoger la pieza, donde se supone que ya no existirá esa "contaminación", mediante la variable *BrazoACamara* (visto en los apartados correspondientes de la programación de los autómatas Siemens y Schneider).

3.7.2. Programación relacionada con los autómatas

El funcionamiento de LabView obliga a identificar cada variable del OPC con un elemento en su diagrama de bloques (no se puede, por ejemplo, enlazar un control en LabView con dos variables del servidor OPC mediante la propiedad de Data Binding). Por lo tanto, en muchos casos se ha necesitado de utilizar la opción de "ocultar indicador", ya que, por ejemplo, sólo es necesario un botón de emergencia en la interfaz visual, aunque tenemos que crear varios de estos "botones" para enlazarlos a las variables de *Emergencia* para cada uno de los autómatas y, por lo tanto, ocultarlos de la interfaz visual.

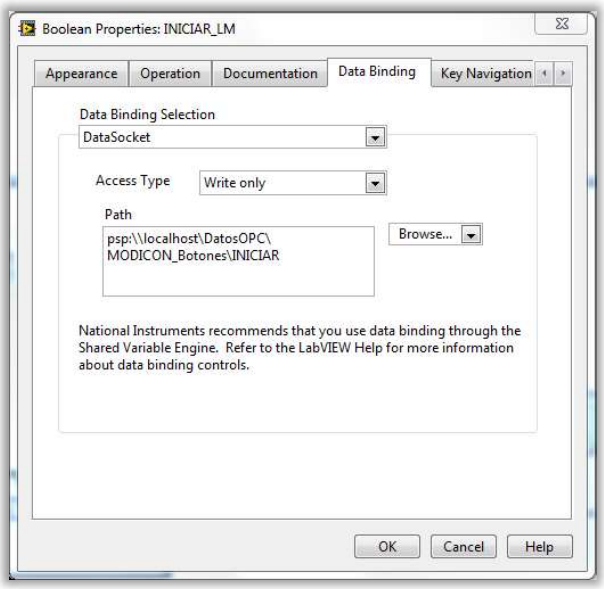


Figura 72. Configuración de Data Binding para un elemento en LabView

Por una parte, tendríamos la **gestión visual**, variables que necesitamos controlar o modificar desde la interfaz visual, y que sufre en muchas ocasiones el problema comentado anteriormente, como que necesitamos crear 3 botones de emergencia (uno para cada autómata), aunque sólo mostraremos uno, y el resto lo subordinaremos al primero. En algunos casos de lectura de variables en los que se precisa de la lectura de estos en varios puntos del programa, se ha podido aprovechar la herramienta de *"Local Variable"* que tienen los elementos de LabView, para que solo un elemento esté conectado a la variable en el Servidor OPC, y el resto simplemente lea o escriba sobre este.

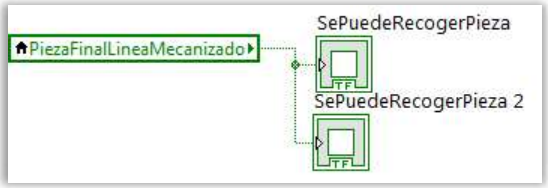


Figura 73. Ejemplo de el uso de la herramienta "Local Variable" para modificar el valor de otros elementos

Es importante destacar el uso de elementos *Ring* para establecer mensajes concretos según el valor de los enteros recibidos mediante el servidor OPC, *CodigoAlarma* (de cada autómata) y *CodigoTestigos* (del autómata Omron). También cabe destacar que el código de alarma valdrá 0 cuando no exista ninguna alarma, y esto se ha aprovechado para la programación de unos "LEDs" de alarma.

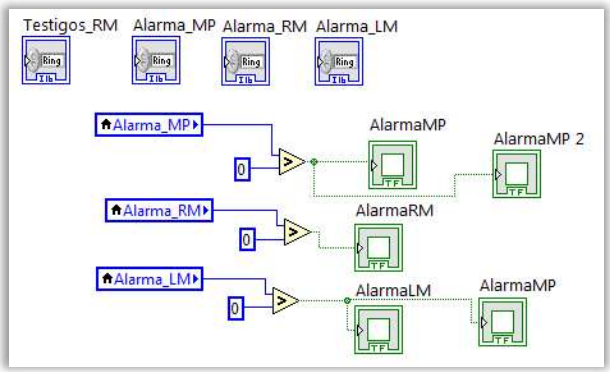


Figura 74. Captura de la parte de programación de alarmas, mostrando los elementos Ring y cómo se han utilizado

Se ha utilizado la herramienta *Tab Control* para poder utilizar varias "pantallas" mediante pestañas y no vernos obligados a sobrecargar toda la información en una única pantalla. La programación ha sido tal para que, al arrancar el programa por primera vez, la aplicación se inicie en la pestaña principal del *Modo Normal* (los distintos modos de la interfaz visual quedan identificados en 2.6. Interfaz visual para el usuario).

Cuando se clica en el botón de Emergencia, aparte de activar la variable *Emergencia* en los 3 autómatas, se ha programado para que oculte el *Tab Control* del *Modo Normal*, mostrando el recuadro del *Modo Emergencia*, que se encuentra debajo, que incluye un nuevo *Tab Control*. Además, al salir del *Modo Emergencia*, hay que asegurarse que el botón que abre la puerta de la *Estación Multiproceso*, no se quede pulsado (es el único botón con enclavamiento en el *Modo Emergencia*).

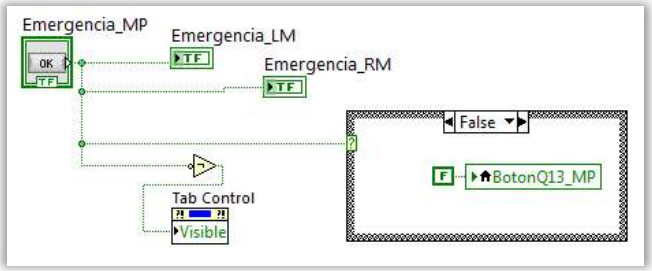


Figura 75. Programación de la emergencia para la interfaz visual, en LabView

Por otro lado, es importante destacar la gestión del ***Modo Automático/Modo Paso A Paso***, donde la variable *ActivaPAP* determina si está o no activo el *Paso A Paso*, y pone visible el botón que se corresponde con la variable *next*, que marca el *Siguiente Paso*. Como se puede comprobar, la variable *NextStep*, que se envía a los 3 autómatas, estará siempre activa cuando *ActivaPAP* esté desactivada, permitiendo que se pueda saltar a la siguiente etapa en los programas cuando se cumplan las condiciones (*Modo Automático*), y solo se activará en caso de que *ActivaPAP* esté activa cuando se pulse en el botón de *Siguiente Paso* (*Modo Paso A Paso*).

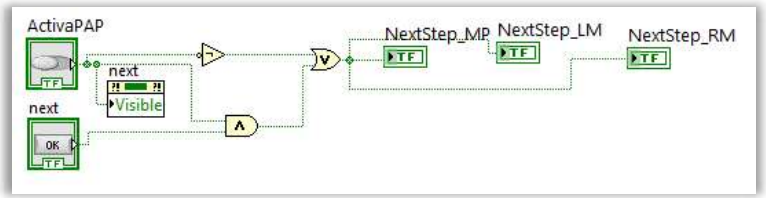


Figura 76. Gestión del Modo Automático/Modo Paso A Paso en el LabView

IMPORTANTE: La programación de las etapas que tienen *NextStep* en la transición requerirán de medidas para evitar que las acciones activadas por la etapa no provoquen daños por la prolongación de esta debido a no saltar automáticamente a la siguiente. Estas programaciones necesarias se pueden ver en los programas de *Acciones* dentro de la programación de cada autómata.

Para terminar la programación del LabView, hay que recordar que también existen variables de comunicación entre autómatas. Estas son variables booleanas simples que cada autómata necesita enviar para que otro reciba, y viceversa. Este envío de información, o "copia" de una variable de un autómata a otro, se ha realizado también en LabView, unificando y simplificando así el software necesario.

Figura 77. Gestión de transmisión de variables entre autómatas en LabView

4. CONCLUSIONES

El proyecto de automatización industrial se ha cumplido con gran éxito, llegándose a obtener un sistema global robusto, con una comunicación efectiva entre autómatas distintos, capaz de soportar la carga de piezas en la cadena, y con los modos de funcionamiento idóneos para el uso normal en la industria.

Se ha equilibrado el modo de Paso A Paso para que produzca la menor pérdida posible en la calidad del producto final, teniendo especial cuidado en el proceso térmico.

Se ha llevado a cabo con éxito el enfoque eficiente en cuanto a ahorro energético se refiere, consiguiendo tanto que el proceso térmico se encuentre lo más aislado posible para evitar fugas de calor (o frío) y la activación por el menor tiempo posible de los motores y compresores de las estaciones involucradas.

Una interfaz de usuario atractiva, sencilla y directa ha sido llevada a cabo, permitiendo aprovechar prácticamente toda la información del proceso industrial, así como controlar este proceso con eficacia y poder tener un sistema de detección de problemas y de solución mediante el Modo Emergencia.

El enfoque dado en la programación permite una fácil comprensión del sistema y una alta capacidad de modificación y mejora para proyectos futuros.

4.1. TRABAJO FUTURO

Como proyectos interesantes para ampliar y mejorar el funcionamiento del sistema se podrían incluir los siguientes:

- Generación automática de archivos de información estadística.
- Control con diferentes perfiles temporales de temperatura en el tratamiento térmico.
- Modificación de coordenadas de la *Estación Robot Manipulador* desde la propia interfaz.
- Adición de contraseñas para operarios con permisos.
- Posibilidad de configuración del PID desde la interfaz visual.

PRESUPUESTO

Ivan Gregori Jorques

1. INTRODUCCIÓN

Este documento de presupuesto incluye los elementos de los que se ha requerido para la realización de este proyecto. Se toma en cuenta que los únicos gastos en material necesarios han ocurrido para estos:

Estación Multiproceso	Autómata Siemens: SIMATIC S7-1200 CPU 1214C AC/DC/RLY 6ES7214-1BG40-0XB0	Módulo SB 1232 AQ 1x12 BIT +/- 10VDC / 0-20mA
		Módulo SM 1223 DC/RLY
	Software Siemens TIA Portal (Totally Integrated Automation Portal) Versión V13 SP1 Update 9 (BASIC)	
Estación Línea Mecanizado	Autómata Schneider Modicon M241 TM241CE40R	TMC4AI2 Analog IN
		TMC4AQ2 Analog OUT 0-10V/4-20mA
	Software SoMachine V4.1 SP2	
Estación Robot Manipulador	Autómata Omron SYSMAC CJ2M CPU31	Módulo MAD42
		Módulo ID211
		Módulo OC211
	Software OMRON CX-Programmer Versión 9.50	
Servidor OPC	Software Kepware KEPServerEX V5.13.191.0	
Interfaz visual	Software National Instruments LabVIEW 2013 Service Pack 1 Version 13.0.1 (32-bit)	

Tabla 4. Material utilizado a presupuestar

Mientras que el gasto realizado en mano de obra ha sido el de un único trabajador, que ha desarrollado las actividades de:

- Diseño y especificación de los sistemas a automatizar (100 h)
- Programación y configuración de los 3 autómatas involucrados (100 h)
- Configuración del Servidor OPC (5 h)
- Programación de la Interfaz visual de usuario (65 h)
- Desarrollo de la documentación del proyecto (30 h)

Por otro lado, las licencias de software se les va a considerar con 3 años de amortización. Y mientras que LabView y KEPServerEX serán necesarios para el contínuo funcionamiento, los software de programación de autómatas solo se requieren durante el periodo de programación.

2. JORNALES

Se han establecido en el apartado anterior las distintas actividades realizadas por el trabajador. Estas actividades se pueden agrupar en dos niveles de trabajo:

- Nivel 1: Diseño, dirección y documentación (130 h)
- Nivel 2: Programación y configuración (170 h)

Estas actividades exigen diferentes herramientas y conocimientos, es por ello que se han expuesto distintas gratificaciones, que permiten ajustar de forma más precisa el salario del trabajador.

En la siguiente tabla se muestra el gasto producido en salario, distinguiendo en los dos niveles de trabajo propuestos anteriormente, con respecto al tiempo empleado:

Nº	Descripción de la mano de obra	Tiempo	€/h	Importe (€)
1	Nivel 1: Diseño, dirección y documentación	130	25,00	3250,00
2	Nivel 2: Programación y configuración	170	23,00	3910,00
	TOTAL (€)			**7160,00**

Tabla 5. Cuadro de precios de jornales

3. COSTE DE LOS MATERIALES

Para el coste de los autómatas y el software, se ha recurrido a distintas fuentes para la comparación de precios. Caben destacar las siguientes:

- https://relepro.com/ para todos los productos de Siemens, siendo este además un partner aprobado por Siemens para la distribución de sus productos.
- http://es.rs-online.com/ para los productos de Schneider y Omron.
- https://www.kepware.com para el software de Kepware, siendo esta su web oficial.
- http://www.ni.com/ para el software LabView, siendo esta su web oficial.

Uso en el proyecto	Nº	Unidades	Descripción del material empleado	Rendimiento	€/unidad	Importe (€)
Estación Multiproceso	1	n	Siemens SIMATIC S7-1200 CPU 1214C AC/DC/RLY 6ES7214-1BG40-0XB0	1	357,66	357,66
	2	n	Siemens SB 1232 AQ 1x12 BIT +/- 10VDC / 0-20mA	1	75,62	75,62
	3	n	Siemens SM 1223 DC/RLY	1	244,27	244,27
	4	h	Software Siemens TIA Portal (Totally Integrated Automation Portal) Versión V13 SP1 Update 9 (BASIC)	40	0,37	14,67
Estación Línea Mecanizado	5	n	Autómata Schneider Modicon M241 TM241CE40R	1	430,59	430,59
	6	n	TMC4AI2 Analog IN	1	116,52	116,52
	7	n	TMC4AQ2 Analog OUT 0-10V/4-20mA	1	135,04	135,04
	8	h	Software SoMachine V4.1 SP2	35	0,00	0,00
Estación Robot Manipulador	9	n	Autómata Omron SYSMAC CJ2M CPU31	1	996,93	996,93
	10	n	Módulo MAD42	1	734,76	734,76
	11	n	Módulo ID211	1	204,05	204,05
	12	n	Módulo OC211	1	274,00	274,00
	13	h	Software OMRON CX-Programmer Versión 9.50	30	0,33	9,90
Servidor OPC	14	d	Software Kepware KEPServerEX V5.13.191.0	747,208	1,88	1.401,28
Interfaz visual	15	d	Software National Instruments LabVIEW 2013 Service Pack 1 Version 13.0.1 (32-bit)	749,71	8,04	6.026,77
			TOTAL			**11.022,06**

Tabla 6. Cuadro de precios de coste de los materiales

4. PRECIOS UNITARIOS

A continuación se muestran los costes por la realización de cada uno de los puntos clave del proyecto:

Concepto	Medición	€/unidad	Importe (€)
Estación Multiproceso	1	2.612,22	2612,22
Estación Línea Mecanizado	1	2.247,15	2247,15
Estación Robot Manipulador	1	3.659,64	3659,64
Servidor OPC	1	1.641,28	1641,28
Interfaz visual	1	7.771,77	7771,77
Presupuesto	1	250,00	250,00
TOTAL			**18182,06**

Tabla 7. Cuadro de precios de precios unitarios

5. PRECIOS DESCOMPUESTOS

Unidad de proyecto	Nº	Unidades	Concepto	Rendimiento	€/unidad	Importe (€)
Estación Multiproceso	1	n	Siemens SIMATIC S7-1200 CPU 1214C AC/DC/RLY 6ES7214-1BG40-0XB0	1	357,66	357,66
	2	n	Siemens SB 1232 AQ 1x12 BIT +/- 10VDC / 0-20mA	1	75,62	75,62
	3	n	Siemens SM 1223 DC/RLY	1	244,27	244,27
	4	h	Software Siemens TIA Portal (Totally Integrated Automation Portal) Versión V13 SP1 Update 9	40	0,37	14,67
		h	Diseño, dirección y documentación	40	25,00	1.000,00
		h	Programación y configuración	40	23,00	920,00
Estación Línea Mecanizado		n	Autómata Schneider Modicon M241 TM241CE40R	1	430,59	430,59
		n	TMC4AI2 Analog IN	1	116,52	116,52
		n	TMC4AQ2 Analog OUT 0-10V/4-20mA	1	135,04	135,04
		h	Software SoMachine V4.1 SP2	35	0,00	0,00
		h	Diseño, dirección y documentación	35	25,00	875,00
		h	Programación y configuración	30	23,00	690,00
Estación Robot Manipulador		n	Autómata Omron SYSMAC CJ2M CPU31	1	996,93	996,93
		n	Módulo MAD42	1	734,76	734,76
		n	Módulo ID211	1	204,05	204,05
		n	Módulo OC211	1	274,00	274,00
		h	Software OMRON CX-Programmer Versión 9.50	30	0,33	9,90
		h	Diseño, dirección y documentación	30	25,00	750,00
		h	Programación y configuración	30	23,00	690,00
Servidor OPC		d	Software Kepware KEPServerEX V5.13.191.0	747,208	1,88	1.401,28
		h	Diseño, dirección y documentación	5	25,00	125,00
		h	Programación y configuración	5	23,00	115,00
Interfaz visual		d	Software National Instruments LabVIEW 2013 Service Pack 1 Version 13.0.1 (32-bit)	749,71	8,04	6.026,77
		h	Diseño, dirección y documentación	10	25,00	250,00
		h	Programación y configuración	65	23,00	1.495,00
Presupuesto		h	Diseño, dirección y documentación	10	25,00	250,00
			TOTAL			**18.182,06**

Tabla 8. Cuadro de precios descompuestos

6. PRESUPUESTOS DE EJECUCIÓN

6.1. PRESUPUESTO DE EJECUCIÓN MATERIAL

Concepto	Importe (€)
Estación Multiproceso	2612,22
Estación Línea Mecanizado	2247,15
Estación Robot Manipulador	3659,64
Servidor OPC	1641,28
Interfaz visual	7771,77
Presupuesto	250,00
TOTAL	**18182,06**

Tabla 9. Presupuesto de ejecución material

El presupuesto de ejecución material asciende a: **DIECIOCHO MIL CIENTO OCHENTA Y DOS EUROS CON SEIS CÉNTIMOS DE EURO**.

6.2. PRESUPUESTO DE EJECUCIÓN POR CONTRATA

Concepto	Importe (€)
Presupuesto de ejecución material	18182,06
Gastos generales	2727,31
Beneficio industrial	1090,92
PRESUPUESTO DE EJECUCIÓN POR CONTRATA	**22000,29**

Tabla 10. Presupuesto de ejecución por contrata

El presupuesto de ejecución por contrata asciende a: **VEINTIDOS MIL EUROS CON VEINTINUEVE CÉNTIMOS DE EURO**.

7. PRESUPUESTO BASE DE LICITACIÓN

Concepto	Importe (€)
Presupuesto de ejecución por contrata	22000,29
IVA (21%)	3300,04
PRESUPUESTO BASE DE LICITACIÓN	**25300,34**

Tabla 11. Presupuesto base de licitación

El presupuesto base de licitación asciende a: **VEINTICINCO MIL TRESCIENTOS EUROS CON TREINTA Y CUATRO CÉNTIMOS**.