UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

TRABAJO FIN DE GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

DESARROLLO E IMPLEMENTACIÓN DE LA AUTOMATIZACIÓN PARA UN SISTEMA DE MECANIZADO DE PIEZAS CON ROBOT MANIPULADOR MEDIANTE COMUNICACIONES DISTRIBUIDAS CON SERVIDOR OPC Y SUPERVISIÓN SCADA

AUTOR: ABEL DOMÍNGUEZ GONZÁLEZ

TUTOR: RAÚL SIMARRO FERNÁNDEZ

Curso Académico 2016-17

Resumen

Existen numerosas empresas las cuales demandan sistemas automáticos, capaces de realizar multitud de tareas sin apenas intervención humana, siendo éstos el futuro tecnológico. Por lo tanto, este proyecto abarca el desarrollo e implementación de la automatización en un sistema de mecanizado de piezas con robot manipulador.

Para tal desarrollo, se ha necesitado de controladores lógicos programables (PLC), capaces de almacenar los datos recogidos por los sensores y, mediante *software*, gestionar dicha información para, posteriormente, enviar las señales a los actuadores, quienes ejecutan las tareas programadas.

Saber el estado del sistema y controlarlo son características indispensables en un sistema automatizado, de tal modo que se ha tenido que llevar a cabo el diseño e implementación de una supervisión, control y adquisición de datos (SCADA).

El hecho de automatizar dos procesos distintos (mecanizado y robot) ha hecho que se necesite de dos PLC, existiendo una comunicación entre ellos para lograr la coordinación entre ambos sistemas. Además, también se ha requerido utilizar una cámara de visión para detectar cuando ha llegado una pieza al puesto de trabajo.

Debido al uso de diferentes dispositivos y a la necesidad de la transferencia de datos entre ellos, se ha tenido que utilizar un servidor para el control de procesos de objetos (OPC) capaz de enlazar los datos obtenidos por la cámara con las variables de los PLC.

Como resultado, se ha obtenido un sistema capaz de gestionarse por sí mismo, como también diferentes interfaces de usuario (HMI) que permiten su supervisión y control.

Palabras clave: PLC, SCADA, servidor OPC, HMI, control automático.

Summary

Exist many companies that demand automatic systems, capable of performing multiple tasks without human intervention, these being the technological future. Therefore, this project covers the development and implementation of the automation in a system of machining parts with robot manipulator.

For such development, it has been necessary to use programmable logic controllers (PLC), capable of storing the data collected by the sensors and, by using a software, manage that information to send the signals to the actuators, who execute the scheduled tasks.

Knowing the state of the system and having control over it are indispensable characteristics in an automated system, so the design and implementation of supervisory, control and data acquisition (SCADA) have had to be carried out.

The fact about automation of two different processes (machining and robot) has made it necessary to have two PLCs, having a communication between them to achieve coordination between both systems. In addition, it has also been required to use a vision camera to detect when a piece of machine has arrived to the workplace.

Due to the use of different devices and the need for data transfer between them, it has been necessary to use a server object linking and embedding (OLE) for process control (OPC) capable of linking the data obtained by the camera with the variables of the PLCs.

As a result, a system capable of managing itself has been obtained, as well as different human machine interfaces (HMI) that allow its supervision and control.

Key words: PLC, SCADA, OPC server, HMI, automatic control.

Agradecimientos

Para empezar, me gustaría darle las gracias a Raúl Simarro Fernández por haberme guiado e instruido en el mundo de la automatización, ya que gracias a él y a este proyecto, he podido aprender nuevos conocimientos que seguro me serán útiles. Tengo que decir además, que Raúl siempre ha estado cuando he necesitado su ayuda, resolviéndome dudas y problemas, y eso es algo a tenerse en cuenta.

También quiero agradecer a esos compañeros que me han dado ánimos durante la realización de este proyecto y que siempre han estado ahí. En especial, me gustaría agradecerle a mi amigo Vicent Alborch por haberme escuchado siempre, sobre todo cuando le hablaba del proyecto, dándome consejos desde su experiencia en la realización del trabajo final de grado.

Y por finalizar, quiero darle las gracias a mis padres, unas personas maravillosas que me han dado su apoyo incondicional desde que tengo memoria, y que no sería quien soy si no fuera por ellos.

A todas las personas que me han acompañado a lo largo de mi vida, gracias.

CONTENIDO DEL PROYECTO

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

TRABAJO FIN DE GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

DESARROLLO E IMPLEMENTACIÓN DE LA AUTOMATIZACIÓN PARA UN SISTEMA DE MECANIZADO DE PIEZAS CON ROBOT MANIPULADOR MEDIANTE COMUNICACIONES DISTRIBUIDAS CON SERVIDOR OPC Y SUPERVISIÓN SCADA

DOCUMENTO Nº 1 MEMORIA

AUTOR: ABEL DOMÍNGUEZ GONZÁLEZ

TUTOR: RAÚL SIMARRO FERNÁNDEZ

Curso Académico 2016-17

Índice

Índice de figuras

Índice de tablas

1. Introducción

A día de hoy, vivimos en un mundo en el cual la tecnología toma un papel muy importante dentro de la sociedad, dando un gran apoyo en la realización de multitud de tareas y mejorando notoriamente la seguridad en muchos ámbitos.

A raíz de la tecnología, se ha obtenido la automatización, la cual reina en la gran mayoría de sectores a nivel mundial; el automovilístico, alimenticio, petrolero y textil son un claro ejemplo, entre muchos otros. Esto es debido a la gran necesidad de sistemas autónomos en las empresas, capaces de auto gestionarse sin apenas intervención humana, con el objetivo de reducir costes de fabricación, agilizar la producción en planta y mejorar la calidad de los productos y/o servicios; cuya finalidad es lograr una mejor calidad de vida en la sociedad.

Debido a la constante evolución tecnológica y a la gran demanda de sistemas automatizados, cada vez más sofisticados e interconectados entre ellos, se pretende desarrollar e implementar una automatización industrial, concretamente, en una planta de mecanizado de piezas y clasificación de éstas.

1.1 Objeto del proyecto

Este proyecto tiene como objeto la automatización y puesta en marcha de un sistema de mecanizado de piezas y clasificador de las mismas. Para ello, se requiere el uso de autómatas programables (comunicados entre ellos) y una supervisión SCADA, tanto para monitorizar el estado del sistema como para gestionarlo. Además, también se necesita de un servidor OPC para compartir la información de los autómatas con una cámara (y viceversa), la cual se utiliza para la detección e identificación de piezas antes de ser mecanizadas.

El objetivo del proyecto es lograr una gran autonomía en el sistema, siendo gestionado mayoritariamente por las máquinas y sistemas de control, como también ser supervisado y controlado (si la situación lo requiriese) por el operario en planta.

Por otra parte, también cabe resaltar que otra de las finalidades del proyecto es consolidar y aprender conocimientos de automatización, para dotar al alumno de una mejor formación como ingeniero.

1.2 Antecedentes

A día de hoy, existe una gran demanda, por parte de las empresas, de ingenieros totalmente cualificados y capaces de llevar a cabo diseños, implementaciones y mantenimientos de sistemas automatizados; sistemas capaces de llevar a cabo tareas (simples o complejas) con gran detalle y precisión, pero a la vez de manera rápida y eficaz. Así mismo, se requiere cada vez más, que dichos sistemas estén comunicados con las demás áreas de la empresa (logística, administración, etc.) para una mejor eficacia en la adquisición de datos por parte de estas áreas.

De esta manera, la realización de este proyecto permite poner en práctica los conocimientos adquiridos a lo largo de la carrera, alejando al alumno de las aulas y acercándolo más a la realidad, a las exigencias del mundo laboral.

1.3 Motivación

El hecho de querer ampliar los conocimientos de automatización ha sido lo que ha llevado a cabo la realización de este proyecto. No solo por querer consolidar las bases aprendidas en la asignatura de Automatización industrial de tercer curso, sino también por el hecho de querer aprender a trabajar con autómatas diferentes. Además, aprender conocimientos nuevos como comunicaciones entre autómatas, intercambio de datos mediante servidor OPC y desarrollo de interfaces de usuario, han sido clave para obtener una gran motivación y realizar el trabajo académico en este ámbito.

Otro punto a destacar, es la gran pasión y deseo de querer enfrentarse a un problema de automatización, el cual se acerque más a la realidad y, a la vez, que presente más complejidad que las prácticas realizadas en la asignatura de Automatización industrial.

Cabe mencionar que la realización de este proyecto, ha servido para hacerse a la idea de cómo trabajan los profesionales en este sector, algo muy importante en el desarrollo de un ingeniero recién graduado.

1.4 Factores a considerar

1.4.1 Normativa

Para la automatización y puesta en marcha del sistema de mecanizado y clasificador de piezas, es obligatorio cumplir con las siguientes normativas:

- **IEC 61131-3.** Estandarización completa de los lenguajes de programación de los controladores lógicos programables.
- **IEC 62541.** Especificaciones de la plataforma de comunicación universal OPC, para modelos estándar de información.
- **IEC 60870-5-101.** Definición el uso de una red TCP/IP.
- **UNE-EN 60848:2013.** Lenguaje de especificación GRAFCET para diagramas funcionales secuenciales.
- **UNE-EN 61000-6-2.** Compatibilidad electromagnética (CEM). Parte 6-2: Normas genéricas. Inmunidad en entornos industriales.
- **EN ISO 13849-1:2006.** Seguridad de las máquinas. Partes del sistema de mando relativas a seguridad. Parte 1: Principios generales para el diseño

1.4.2 Condiciones del encargo

Las condiciones del encargo provienen principalmente de:

- Las necesidades y exigencias del cliente.
- Las características técnicas del sistema a automatizar.
- La complejidad del proyecto

Prestaciones y funciones que debe cubrir el proyecto:

- Utilización de dos autómatas y comunicación entre ellos.
- Supervisión y control del sistema mediante SCADA, el cual debe ofrecer:
    - Modo de funcionamiento automático, para una producción continua.
    - Modos de funcionamiento manual y test, para mantenimiento y resolución de fallos.
    - Elección manual del tipo de pieza a mecanizar, recomendado para una producción por lotes.
    - Visualización del número de piezas fabricadas de cada modelo, como también del número de piezas sin mecanizar del que se dispone.
    - Control total del sistema, tanto de la línea indexada como del robot manipulador.
    - Monitorización del estado del sistema en todo momento.

- Incorporación de visualización web, para así poder monitorizar y controlar el sistema desde cualquier dispositivo electrónico con acceso a internet (ordenador, móvil, tablet, etc.)
- Implementación de un dispositivo capaz de detectar el color de la pieza antes de ser mecanizada (cámara).
- Implementación de una interfaz de usuario (HMI) para el control visual.
- Creación, configuración y puesta en marcha de un servidor OPC capaz de gestionar y compartir la información entre los autómatas y la cámara (incluido su HMI).
- Correcto funcionamiento del sistema, satisfaciendo las especificaciones dadas por el cliente.

2. Soluciones alternativas y justificación de la solución adoptada

Para el diseño y desarrollo de cualquier proyecto, es necesario el planteamiento y/o estudio de posibles soluciones a adoptar, para determinar cuál sería la más eficiente, tanto técnica como económicamente. Por lo tanto, en este proyecto se plantean diferentes soluciones, las cuales son las más comunes a día de hoy; pero no todas ellas son igual de válidas en el sector de la automatización.

2.1 Tipo de lógica

Dependiendo del tipo de tecnología a utilizar en un automatismo, se podría necesitar el uso de bastantes elementos físicos (cableado, componentes eléctricos, neumáticos, etc.), o bien la sustitución de éstos por un dispositivo programable, capaz de llevar a cabo las mismas funciones.

Para implementar un automatismo, existen dos opciones actualmente:

- **Lógica cableada:** Uso de elementos físicos como circuitos cableados, relés, contactores de potencia, válvulas neumáticas y/o hidráulicas, entre otros, para el diseño e implementación del automatismo; capaz de gestionar el sistema y , a la vez, ser gestionado desde el cuadro de mandos por el operario.

- **Lógica programada:** Utiliza, como elemento físico, un dispositivo lógico programable (PLC), capaz de llevar a cabo el control del sistema mediante el programa realizado por el ingeniero (programa el cual es capaz de reconfigurar los circuitos del PLC) por medio del *software* diseñado para dicho dispositivo lógico.

	Lógica programada	Lógica cableada
Ventajas	Gran flexibilidad para modificaciones, tanto simples como complejas Dispone de distintos lenguajes de programación Requiere poco espacio Permite una gran variedad de comunicaciones entre diferentes sistemas Conexión a diferentes dispositivos de manera sencilla Fácil de transportar Reduce notoriamente el coste de mano de obra al implementarse	Más fiable que la lógica programada en lugares críticos de una instalación, donde se requiere una seguridad extrema de las personas y máquinas. Permite pequeñas modificaciones
Inconvenientes	Se requiere de personal cualificado en configuración del *software* y del/los lenguaje/s de programación. Se debe tener en cuenta un elevado número de detalles en la programación para que el sistema funcione correctamente	Gran cantidad de elementos físicos a implementar Requiere bastante espacio La conexión entre elementos es compleja y costosa Requiere de un gran número de elementos de protección No permite modificaciones a gran escala, habría que cambiar el cableado por completo

Tabla 1. Ventajas y desventajas de los tipos de lógica.

Actualmente, las industrias demandan sistemas de gran alcance y bastante robustos, con una gran necesidad de tener todos los dispositivos conectados entre ellos; la cantidad de información que éstos transmiten entre unos y otros es tan elevada, que se requiere de una exhaustiva monitorización de dicha información (para prevenir/detectar errores). Cabe destacar, que las exigencias y gustos de los consumidores cambian constantemente, como también los productos de la competencia. Por lo tanto, las industrias deben tener una gran flexibilidad para realizar los cambios necesarios en planta y poder satisfacer las nuevas exigencias del mercado.

Como resultado al análisis realizado, se ha decidido utilizar la lógica programada debido a la gran cantidad de ventajas y ahorro de inconvenientes que proporciona a las industrias. En el caso de este proyecto, la lógica programada proporciona:

- Gran flexibilidad para realizar cambios en la programación y reestructurar o ampliar el sistema de mecanizado.
- Gran variedad de conexiones y comunicaciones entre dispositivos, perfecto para la transmisión de datos, al mismo tiempo que permite la ampliación de la planta de mecanizado (adición de sensores, cámaras, motores, etc.)
- Fácil y completa monitorización del sistema (tanto del mecanizado como del robot)

2.2 Tipos de controladores

Una vez se ha elegido la programación de dispositivos como solución para la automatización, se debe estudiar y decidir que elemento de control se va a utilizar, cuya finalidad es gestionar el sistema mediante el procesamiento y transmisión de datos. Los elementos de control más utilizados en las industrias son:

- **Tarjeta de adquisición de datos (DAQ):** Básicamente se trata de un dispositivo que sirve de intermediario entre la variable física deseada y su posterior tratamiento en un sistema digital. Dicha tarjeta recoge el valor físico leído por un sensor y lo transforma en un dato (por medio de tensiones eléctricas), para poder ser leído y almacenado en una computadora y ser, posteriormente, utilizado en un *software*.

- **Microcontrolador:** Se trata de un chip procesador utilizado para realizar determinadas tareas, especificadas por el programador a través de un *software* en específico (depende de la marca del microcontrolador). Está constituido por:

  - Una unidad central de procesamiento (CPU)
  - Canales de entrada/salida
  - Temporizadores

- **Controlador lógico programable (PLC):** Es un dispositivo electrónico diseñado específicamente para la automatización de sistemas con un gran número de entradas y salidas. El PLC es resistente a altas temperaturas, inmune al ruido eléctrico y a la vez, resistente a impactos, lo que lo hace un aliado perfecto para combatir los problemas de automatización en la industria. Cabe mencionar que el PLC tiene habilitadas entradas especiales para la incorporación de módulos externos, utilizados para ampliar sus funciones como controlador en función de las demandas exigidas y la complejidad del proyecto.

	Tarjeta de adquisición de datos (DAQ)	Microcontrolador	Controlador lógico programable (PLC)
Ventajas	Conversión de datos A/D y D/A Tratamiento de magnitudes físicas Gran almacenamiento de datos en la computadora Gran velocidad de procesamiento de datos al disponer de toda la memoria del ordenador	Portable, debido a su pequeño tamaño Fácil de implementar en circuitos Suele venir incorporado en circuitos integrados Contiene temporizadores y puertos E/S	Suelen ser portables y poco pesados (depende del fabricante) Gran número de E/S Habilitado para adición de módulos externos Distintos lenguajes de programación, estandarizados según norma IEC 61131-3 (**véase apartado 1.4.1**) Fácil programación (basada en lógica cableada y estructuras secuenciales) Fácil comunicación entre ellos, como también con otros dispositivos Permite la monitorización del sistema
Inconvenientes	Requiere ordenador Necesita espacio para el ordenador y su correspondiente cableado El sistema operativo y otras funciones del ordenador podrían ralentizar las tareas de la tarjeta de adquisición Coste elevado	Programación compleja Requiere de otros componentes electrónicos para controlar un sistema	Coste elevado (depende de marca y modelo) Puede requerir un espacio considerable Necesita alimentación de la red eléctrica (220 V) (depende del fabricante)

Tabla 2. Ventajas y desventajas de los tipos de controladores usados en la industria.

Se observa como el PLC brinda todas las características que las industrias demandan: capacidad para gestionar un gran número de entradas/salidas, programación pensada para procesos secuenciales y tareas específicas, monitorización de las variables del sistema, conexión a otros dispositivos, etc.

De tal modo que, para este proyecto, se ha decidido utilizar el PLC como elemento de control del sistema de mecanizado y clasificador de piezas.

2.3 Autómatas programables y *software* de programación

A día de hoy, la gran demanda de autómatas por parte de las industrias ha llevado a cabo la comercialización a gran escala de éstos, suponiendo un gran número de empresas dedicadas a la comercialización de estos dispositivos.

Estas empresas tienen su propia gama y modelos de PLC (como también sus respectivos *software* diseñados específicamente para cada tipo de autómata) con características y prestaciones diferentes, cuya finalidad es cubrir la demanda existente de autómatas en el mercado. Por lo tanto, la elección de un autómata podría llegar a ser una tarea compleja, pero al mismo tiempo, es fundamental para optimizar el gasto que suponen y la funcionalidad que pueden aportar.

Entre los distintos fabricantes que existen, se mencionan las compañías de las cuales se tienen sus respectivos autómatas en el departamento de ingeniería de sistemas y automática (DISA) de la universidad politécnica de valencia (UPV):

	Schneider Electric	***Omron***	***Siemens***
Modelo de autómata	Modicon M241 (TM241CE40R)	CJ2M	SIMATIC S7-1200
Software del autómata	SoMachine	CX-Programmer	TIA-Portal v13

Tabla 3. Autómatas y *software* disponibles en el DISA.

La elección del autómata se debe realizar en base a las prestaciones aportadas por cada uno de ellos. Como la finalidad del proyecto es aprender y profundizar en los conocimientos de automatización, se descarta el autómata de *Omron* debido a que se utilizó en la asignatura de Automatización industrial. De este modo, el alumno puede desarrollarse con otra gama de PLC y adquirir una mayor flexibilidad a la hora de automatizar una planta industrial.

A continuación, se resaltan las características más relevantes de los autómatas de *Schneider Electric* y *Siemens* que contiene el laboratorio, para una posterior elección:

Modicon M241	**SIMATIC S7-1200**
	
- **16** entradas normales y **8** entradas rápidas (digitales)	- **14** entradas digitales
- **12** salidas de relé y **4** salidas rápidas (digitales)	- **10** salidas normales y **2** salidas rápidas (digitales)
- **2** puertos de línea serie, **1** puerto de programación USB y **1** puerto *Ethernet*	- **1** puerto de comunicación *Ethernet*

Tabla 4. Características de los autómatas Modicon M241 y SIMATIC S7-1200.

En un principio, ambos autómatas cumplen con las prestaciones que el sistema demanda, ya que ambos contienen suficientes entradas y salidas, tanto para el sistema de mecanizado como para el robot manipulador, como también un puerto *Ethernet*.

Como el *software SoMachine* tiene la capacidad de monitorizar el sistema desde cualquier dispositivo conectado a la red (visualización web) y posee todos los lenguajes de programación estandarizados como SFC, diagrama de contactos, texto estructurado y diagrama de bloques, se ha decidido utilizar el **Modicon M241** de *Schneider Electric* para la realización del proyecto.

2.4 Servidores OPC

El servidor OPC (*OLE for Process Control*) es un estándar de comunicaciones en el campo de control y supervisión de procesos industriales, siendo un *software* que permite la comunicación entre dispositivos, utilizando el sistema cliente/servidor, en el cual un dispositivo hace de servidor (donde se gestionan todos los datos), mientras que los demás dispositivos hacen de clientes, demandando al servidor los datos cuando éstos lo necesitan.

En las industrias se utilizan bastantes tipos de servidores OPC, y hay muchos de ellos para las distintas marcas como *Omron, Schneider,Allen -Bradley, Siemens, ABB, etc.* Cabe mencionar que, muchos de estos servidores, solo pueden realizar comunicaciones con dispositivos de la misma marca (con el mismo *driver*), pero también, los hay que permiten una comunicación más flexible (entre dispositivos de diferentes marcas).

Para este proyecto, si bien es cierto que los dos autómatas a programar son de la misma marca, pero la cámara a utilizar y el *software* para ella, no lo son. Por lo tanto, se requiere de un servidor OPC que permita la comunicación entre diferentes marcas.

De entre los más utilizados, destacan:

- ***MatrikonOPC***
- ***KEPServerEX***

Ambos permiten la comunicación entre dispositivos con diferentes *drivers*. Para este proyecto, se ha utilizado el ***KEPServerEX*** debido a que, por una parte, es el que está instalado en el DISA y, por otra parte, la comunicación se establece de una forma rápida y muy sencilla.

2.5 Cámaras de visión artificial y *software* de programación

2.5.1 Cámaras

La visión artificial es muy utilizada en la industria con muchas finalidades: detección de objetos, inspección de la forma y/o color de un producto, como también la revisión de la calidad del mismo, entre muchas otras aplicaciones.

Existen bastantes tipos de cámaras, utilizadas en función de las necesidades del sistema. En este apartado se nombran algunas de las más utilizadas:

- **Cámaras matriciales**: Formadas por una matriz de píxeles, produciendo una imagen de un área.

- **Cámaras 3-D:** Permiten realizar medidas de formas en 3D. Para ello, no solo se utiliza la cámara, si no que en realidad, es un sistema formado por cámara y un láser de línea, además del *software* de triangulación, el cual permite obtener las medidas.

- **Cámaras infrarrojas/térmicas:** Pueden obtener la radiación infrarroja de hasta 1000 nm. Utilizadas en la industria para medir la temperatura de un cuerpo a partir de su radiación infrarroja.

- **Cámaras lineales:** Utiliza un sensor lineal, construyendo la imagen línea a línea, de tal modo que la cámara se desplaza con respecto al objeto a capturar (o viceversa). Muy útil y demandado en la industria, tanto para procesos de producción como de inspección.

  Debido a que la aplicación que se requiere de la cámara es simplemente la detección de colores, serviría casi cualquier tipo de cámara (siempre y cuando tuviesen implementado un sensor RGB).
  Para este proyecto, se ha utilizado el sensor Kinect de *Microsoft* debido a que es bastante económico, portable y ocupa poco espacio. Además, el sensor Kinect posee un sensor de profundidad 3-D (además del sensor RGB), por lo tanto, se podría decir que la cámara elegida ha sido una cámara 3-D (aunque no se vaya a usar tal función).

  Cabe resaltar que la cámara Kinect no es adecuada para el uso industrial, ya que es frágil a golpes y no soporta temperaturas elevadas (entre 5 - 35 ºC). Sin embargo, es bastante útil a la hora de realizar pruebas o investigaciones de automática con visión artificial en los laboratorios, ya que aporta bastantes prestaciones.

2.5.2 *Software*

Existen bastantes *software* para la programación de cámaras, tanto de la adquisición como del procesamiento de la imagen. Entre muchos de ellos, se hace mención de unos cuantos:

- ***Sherlock***
- ***HALCON 13***
- ***LabVIEW***
- ***OpenCV***
- ***Matlab***

Entre los mencionados, se ha elegido ***LabVIEW*** para la programación de la adquisición y procesamiento de la imagen captada por la Kinect, debido a diversos factores:

- Tanto *Matlab* como *Sherlock*, son programas utilizados a lo largo de la carrera, por lo tanto, no se iban a utilizar en este proyecto ya que se quería aprender a programar con otro *software.* (Cabe destacar que la programación con *Matlab* para este tipo de aplicaciones es bastante compleja y requiere amplios conocimientos de programación).

- *LabVIEW* es muy utilizado en las industrias debido a la gran calidad de los HMI y SCADA.

- Compatibilidad con todos los dispositivos de ***NATIONAL INSTRUMENTS*** (bastante utilizados en la industria).

- Fácil programación, muy intuitiva (lenguaje gráfico o lenguaje G).

- Permite el acceso a servidores OPC, como también el fácil acceso a las variables compartidas por otros dispositivos en dichos servidores.

3. Descripción técnica del equipo

En este apartado se describen las características técnicas de todos los elementos (tanto físicos como informáticos) que conforman todo el sistema de este proyecto.

3.1 Línea indexada de mecanizado *FischerTechnik*

El sistema de mecanizado a automatizar se trata de una maqueta de la marca *Fischertechnik* (debido a que no se posee una planta industrial real para la realización de este proyecto).

Está constituido por dos elementos, sensores y actuadores:

3.1.1 Sensores

- **Fototransistor:** Sensor electrónico capaz de activarse cuando recibe luz. Al recibir luz en la base, éste se activa y permite el paso de corriente entre su emisor y colector (igual a un transistor BJT). Por supuesto, para que funcione, requiere de un emisor (ya que el fototransistor actúa como receptor). El emisor utilizado es una lámpara con lente.

- **Final de carrera:** Sensor de contacto (en este caso, eléctrico) que al ser activado (por medio de presión o contacto con algún elemento, como su nombre indica), cierra el circuito, permitiendo el paso de corriente a través de él. También los hay neumáticos y mecánicos.

3.1.2 Actuadores

- **Motores de corriente continua (CC):** Elemento formado por un estator y un rotor, convirtiendo la energía eléctrica en mecánica para realizar un movimiento giratorio.
  En el sistema, los motores van conectados a engranajes, al igual que las cintas. De este modo, con el movimiento del motor, se mueven los engranajes, lo cual otorga movimiento a las cintas.

Elementos controlados por los actuadores:

- **Cinta**: Usada para el transporte de las piezas a lo largo de la línea de mecanizado.

- **Empujadores**: Funciona con un motor de CC, moviéndose linealmente por medio de una rueda dentada. Utilizado para el empuje de las piezas (para pasarlas de una cinta a otra).

- **Elementos de mecanizado**: Compuestos por una fresadora y una taladradora, utilizados para el arranque de virutas y perforación de las piezas.

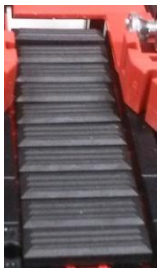
Cinta	Empujador	Fresadora	Taladradora
			

Tabla 5. Elementos físicos a controlar del sistema de mecanizado.

3.1.3 Características físicas

- Dimensiones: 450 x 410 x 190 mm

- 9 entradas digitales

- 10 salidas de 24V (6 motores de un sentido de rotación *counter-clockwise, clockwise rotation*)

- 2 Cables de 18 pines cada uno y codificado con colores con conector de 18 pines, sólo para 24V (PLC).

3.1.4 Vistas del sistema

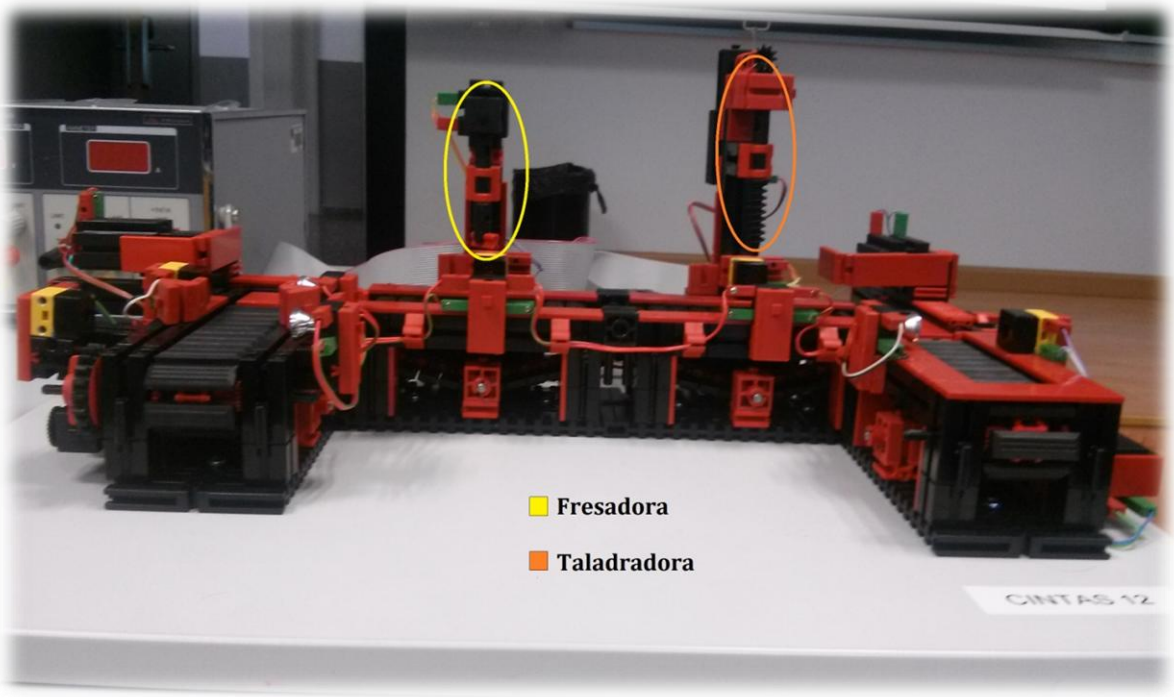


Figura 1. Vista desde alzado del sistema de mecanizado.

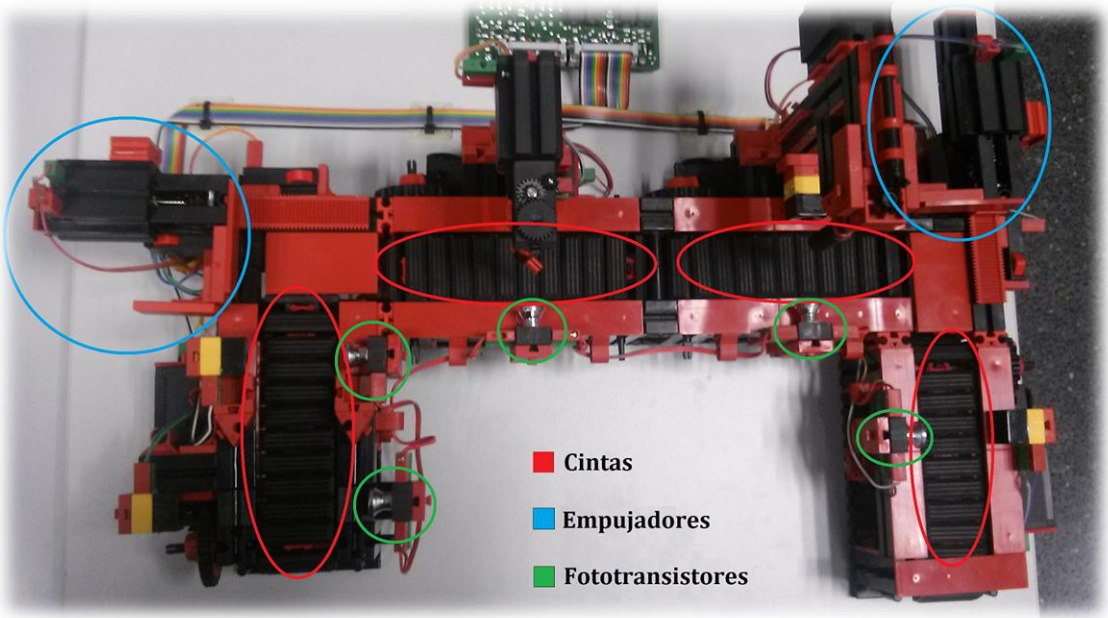


Figura 2. Vista desde planta del sistema de mecanizado.

3.2 Robot manipulador de vacío FischerTechnik

Para la puesta y clasificación de las piezas se utiliza una maqueta de un brazo robótico tridimensional (debido al uso de otra maqueta por la carencia de una planta industrial real), también de la marca FischerTechnik.

Posee dos elementos característicos:

3.2.1 Sensores

- **Final de carrera** (véase descripción en la página 13).

3.2.2 Actuadores

- **Motor CC** (véase descripción en la página 13).
- **Compresor:** Elemento constituido por un tubo y un émbolo. Dependiendo del movimiento del émbolo, expulsa o absorbe el fluido (en este caso, aire) para obtener un movimiento mecánico. Es utilizado para que la ventosa del robot pueda adherirse a la pieza por medio de un vacío entre ambos elementos, creado gracias al compresor.

Elementos controlados por los actuadores:

- **Ventosa:** Utilizada para el agarre de la pieza mediante aspiración por vacío.
- **Engranajes:** Gracias a los motores, los engranajes se mueven permitiendo el desplazamiento de los ejes del robot por medio de barras deslizadoras.

Ventosa	Engranajes
	

Tabla 6. Elementos físicos a controlar del robot manipulador.

3.2.3 Características físicas

- Eje X de 270° (750 pulsos de encoder)
- Eje Y (avance/retroceso) de 140 mm (≅1000 pulsos de encoder)
- Eje Z (arriba/abajo) de 120 mm (1000 pulsos de encoder)
- 3x *encoder*-motor (3 salidas)
- 3x finales de carrera (3 entradas)
- Ventosa de vacío
- Compresor
- 24 V de entrada

3.2.4 Vistas del sistema

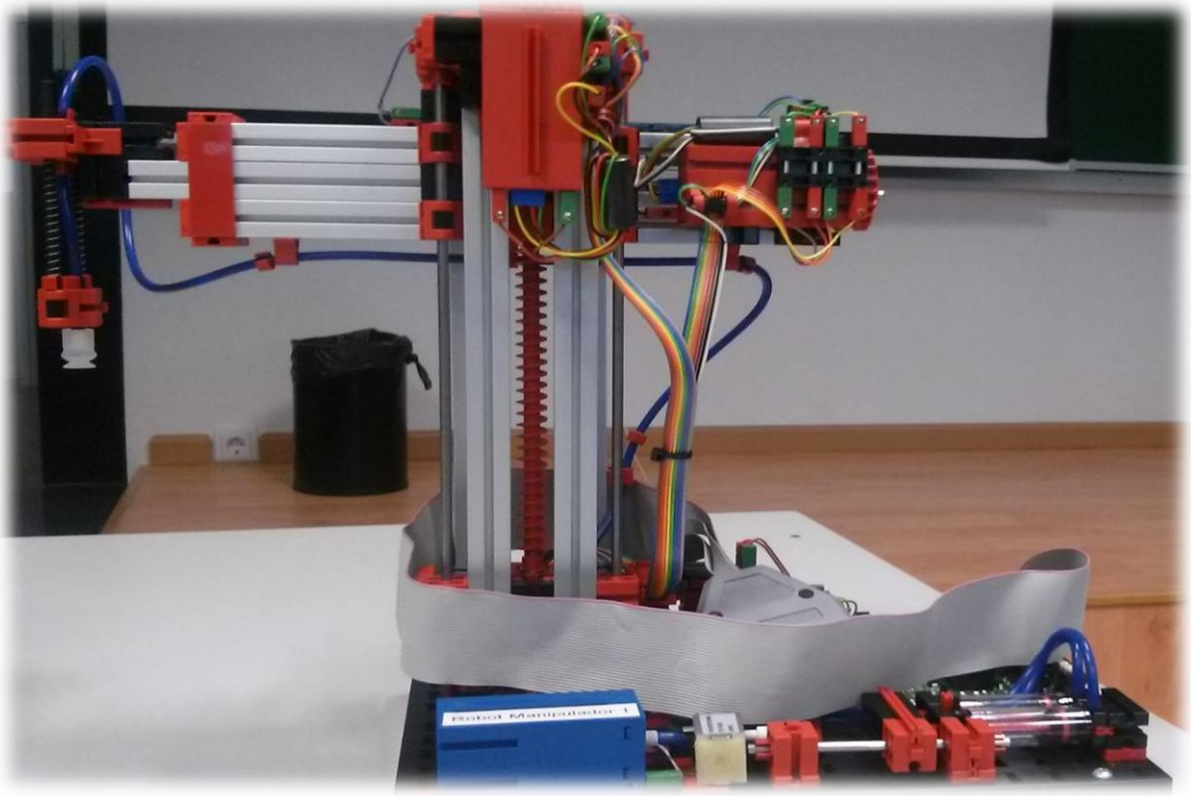


Figura 3. Vista desde alzado del robot.

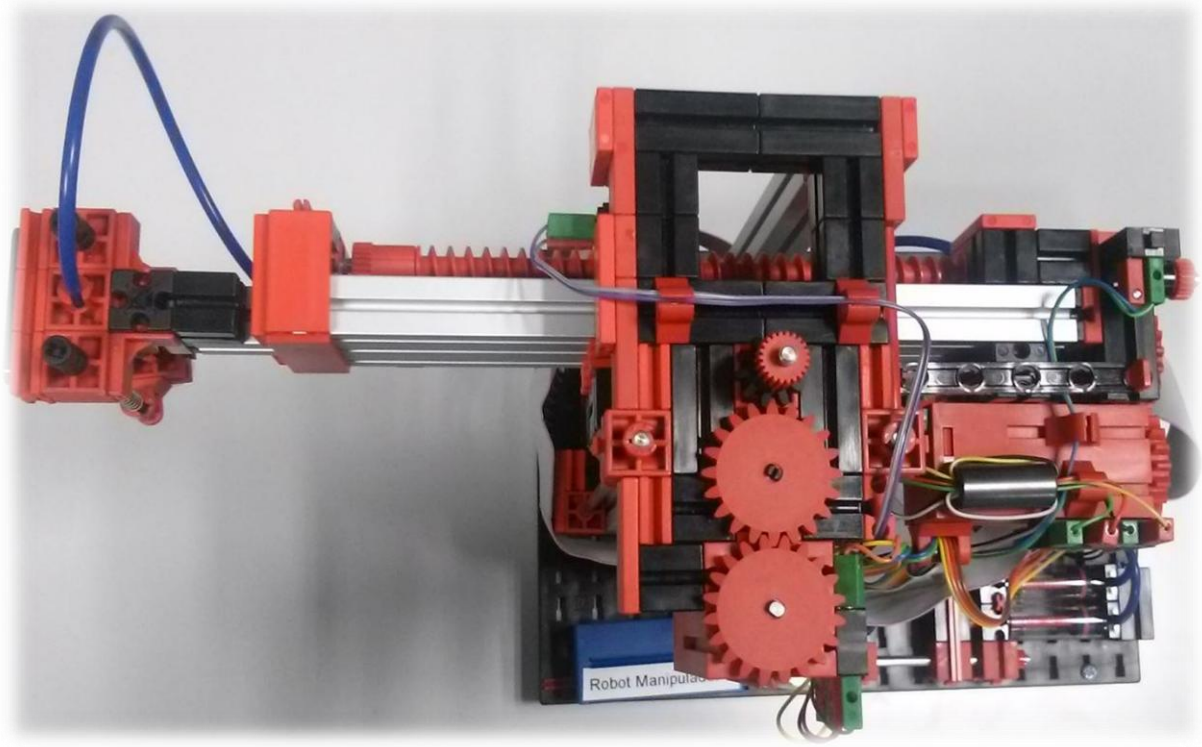


Figura 4. Vista desde planta del robot.

3.3 Controlador lógico Modicon M241

Figura 5. Autómata Modicon modelo M241 de *Schneider Electric*.

El controlador Modicon M241 contiene grandes prestaciones, ideal para máquinas modulares y aplicaciones donde las exigencias son elevadas y se requieren costes bajos, además, permite la reducción de los tiempos de instalación.

Características:

- 5 puertos: *Ethernet*, *CANopen*, 2 puertos serie y un puerto USB para programación.
- CPU con procesador de alto rendimiento y cartuchos de extensión.
- Servidor web y FTP.
- Salidas de tren de pulsos / contadores de alta velocidad.
- Tarjeta SD.
- Configuración rápida y fácil.
- Módulos de seguridad.
- Módulos para arrancadores de motor *Tesys SoLink.*
- Amplia gama de módulos de extensión de E/S analógicas y digitales.
- Comunicación *Ethernet* y *Profibus.*
- Programación intuitiva con *SoMachine .*
- Lógica estándar, control de movimiento y configuración HMI.
- Plantillas y librerías predefinidas.

3.4 Sensor *Kinect*

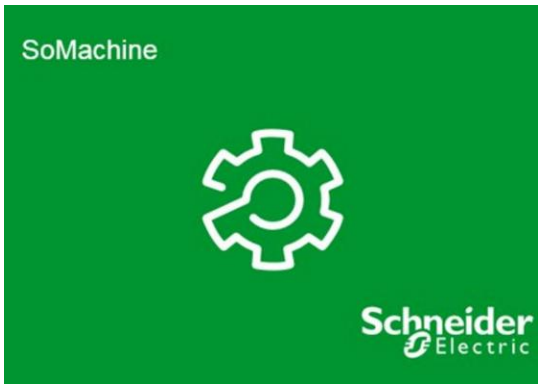
Figura 6. Sensor de visión *Kinect*.

El sensor *Kinect* es utilizado para distintas aplicaciones, entre ellas, pruebas de funcionamiento e investigación de visión artificial en robótica industrial.

Características que incluye el *Kinect*:

- Sensores de profunidad 3-D.
- Cámara RGB.
- Micrófono.
- Inclinación monitorizada mediante impulso mecánico en la base del sensor.
- Cable del sensor con puerto auxiliar (dicho puerto contiene dos cables, uno con el puerto USB y otro con el enchufe).
- Requiere alimentación a red eléctrica (220 V).
- Campo de visión horizontal: 57 grados
- Campo de visión vertical: 43 grados
- Rango de inclinación física: ± 27 grados
- Rango de profundidad del sensor: 1,2 – 3,5 metros
- Rango de temperatura: 5 - 35 ºC

3.5 *SoMachine (Schneider Electric)*



Software válido para los cuatro tipos de controladores de *Schneider* (Lógicos, Motion, HMI y Driver), además de transferir toda la aplicación a la máquina en una única descarga.

Al ser un *software,* se requiere por parte del ordenador, una serie de requisitos mínimos para poder trabajar adecuadamente con *SoMachine:*

3.5.1 Requisitos del sistema

Hardware

Descripción	Especificaciones Mínimas	Recomendado
Procesador	Intel® Core™ 2 Duo* o equivalente	Intel® Core™ I7 o equivalente
Memoria RAM	3 GB	8 GB
Espacio libre en el Disco Duro	8 GB	15 GB
Drive	Lector de DVD	Lector de DVD
Monitor	Resolución: 1024 x 1024 pixel	Resolución:1680 x 1050 pixel
Periféricos	USB interface	USB interface
Web	Para registrarse vía Web se necesita acceso a Internet	Para registrarse vía Web se necesita acceso a Internet

Tabla 7. Requisitos de *Hardware* para soportar *SoMachine*.

Software

SoMachine Componente	Versiones de Sistemas operativos
Sistema operativo	SoMachine software soporta los siguientes sistemas operativos: ➢ Microsoft Windows XP Professional ➢ Microsoft Windows 7 Professional Edition de 32 bits/64 bits

Tabla 8. Requisitos del sistema operativo para soportar *SoMachine*.

3.5.2 Características

Incluye los seis lenguajes que indica la normativa IEC 61131-3 (**véase apartado 1.4.1**):

- Diagrama de funciones (*Function Block Diagram*, FBD)
- Grafcet (*Sequential Functional Chart,* SFC)
- Texto estructurado (*Structured Text,*ST)
- Listado de instrucciones (*Instruction List*, IL)
- Diagrama de contactos (*Ladder,* LD)
- *Continuos Function Chart (*CFC)

Programación:

- Creación de Bloques de función (FBs) por el usuario.
- Creación de Funciones por el usuario.
- Creación de estructura de datos (DUT's).
- Cambios On-line.
- Ventanas de supervisión de variables.
- Monitorización Graficas de variables (trace).
- Puntos de interrupción, ejecución instrucción por instrucción.
- Simulación.
- Ventanas de visualización.

Interfaz de usuario (HMI):

- Librerías gráficas que contienen más de 4000 objetos 2D y 3D.
- Objetos de dibujo (puntos, líneas, rectángulos, elipses, etc.)
- Objetos pre configurados (botón, interruptor, barra gráfica, etc.)
- Tablas de Acciones
- Alarmas
- Imprimir
- Java scripts
- Gráficas.

Buses de comunicación:

- *CANopen*
- *CANmotion*
- *Modbus Serial Line*
- *Profibus - DP*
- *Ethernet IP*
- *Modbus TCP*

En cuanto a servicios globales, ofrece seguridad, documentación de proyecto, comparaciones de proyectos y gestión de librerías, entre otros.

3.6 *LabVIEW* (*NATIONAL INSTRUMENTS*)

La principal característica a nivel funcional de *LabVIEW* es su fácil programación, válido tanto para profesionales como para personas con conocimientos básicos en programación; pudiendo realizar programas complejos de manera sencilla, los cuales serían difíciles de crear con lenguajes tradicionales.

En lo referente a características técnicas, presenta las siguientes:

- **Interfaces de comunicaciones:**

  - Puerto serie
  - Puerto paralelo
  - GPIB
  - PXI
  - VXI
  - TCP/IP, UDP, DataSocket
  - Bluetooth
  - USB
  - OPC (Utilizado en este proyecto)

- **Interacción con diferentes lenguajes de programación y *software, a destacar:***

  - ActiveX
  - Matlab/Simulink
  - AutoCAD, SolidWorks (programas CAD)

- **Herramientas gráficas y textuales para el procesado digital de señales.**
- **Visualización y manejo de gráficas con datos dinámicos.**
- **Adquisición y tratamiento de imágenes.**
- **Control de movimiento (combinado incluso con lo mencionado anteriormente).**
- **Tiempo Real.**
- **Sincronización entre dispositivos.**

3.7 *KEPServerEX* (*kepware*)

Este tipo de OPC, permite la comunicación entre casi cualquier dispositivo y sistema, de tal modo que se pueden agregar múltiples drivers de comunicación dentro de un único servidor OPC, sin necesidad de ir creando nuevos. Esto es a lo que se le denomina “proyectos escalables”.

Características más destacables:

- ***Tags* Avanzados** (variables creadas dentro del servidor OPC, directamente enlazadas con las variables deseadas del sistema).
- **Alarmas y eventos.**
- ***Datalogger*** (almacenamiento de datos del servidor en cualquier base de datos compatible con *Open DataBase Connectivity* (ODBC)).
- **Historiador local**, permitiendo la recolección, almacenamiento y uso de los datos de una manera cercana al origen de los mismos, previniendo pérdidas. Así mismo, permite la adición de nuevas variables sin realizar paradas en un sistema en marcha.
- **Políticas de seguridad**, para dar permisos a los usuarios (poder leer, escribir o simplemente acceder a las variables).
- **No requiere conexión *Virtual Private Network* (VPN)** (para el traspaso de datos de manera segura).

4. Descripción detallada de la solución adoptada

En este apartado se explica el funcionamiento del sistema en toda su totalidad, como también la programación realizada (a grandes rasgos).

4.1 Esquema funcional del sistema

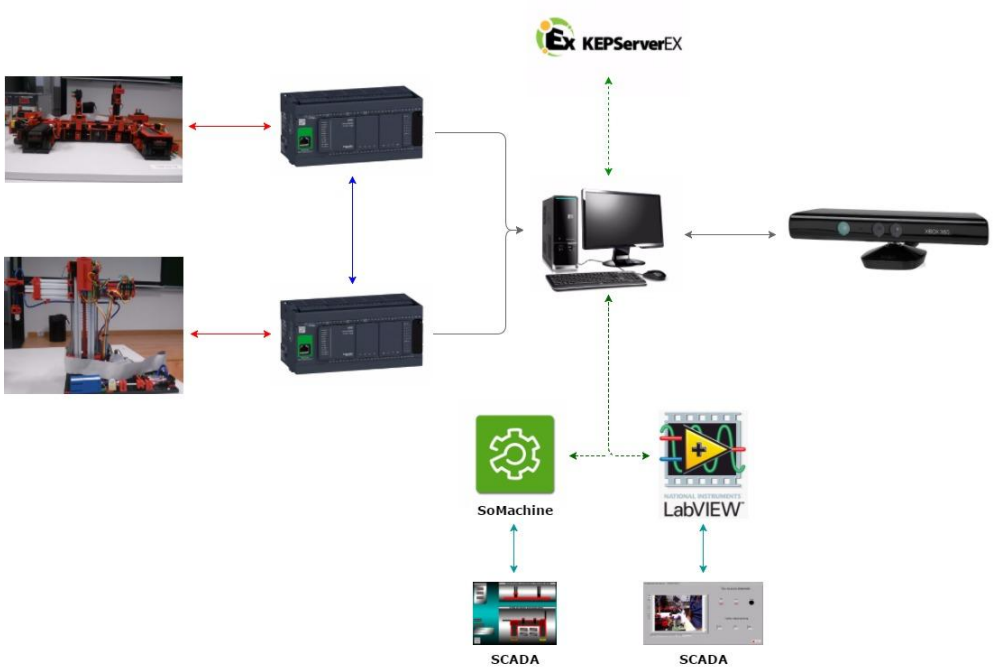


Figura 7. Esquema de la comunicación entre los elementos físicos e informáticos.

Como se visualiza en el esquema, cada maqueta está conectada a un autómata por medio de un conector, al mismo tiempo que cada autómata tiene establecida una conexión con la computadora por medio de *Ethernet*. Cabe destacar que los autómatas comparten una comunicación maestro/esclavo, establecida dentro del propio programa en *SoMachine* (**véase apartado 4.3.8**).

El ordenador es necesario para el uso de *SoMachine, LabVIEW* y el servidor OPC, haciendo de intermediario en el traspaso de datos entre los autómatas y el sensor *Kinect*. Además, *LabVIEW* puede establecer una comunicación directa con el servidor OPC desde su proyecto, creado por el programador.

Cabe mencionar que *LabVIEW* tiene librerías de la cámara *Kinect* (aunque no son de *NATIONAL INSTRUMENTS),* por lo tanto no es necesario realizar una preconfiguración para establecer una conexión entre el programa y la cámara.

4.2 Distribución en planta de los sistemas FischerTechnik

Para un correcto funcionamiento, se requiere una ubicación concreta entre ambos sistemas, como también de las piezas para su puesta en el mecanizado por el robot manipulador, tal y como se muestra en la siguiente figura:

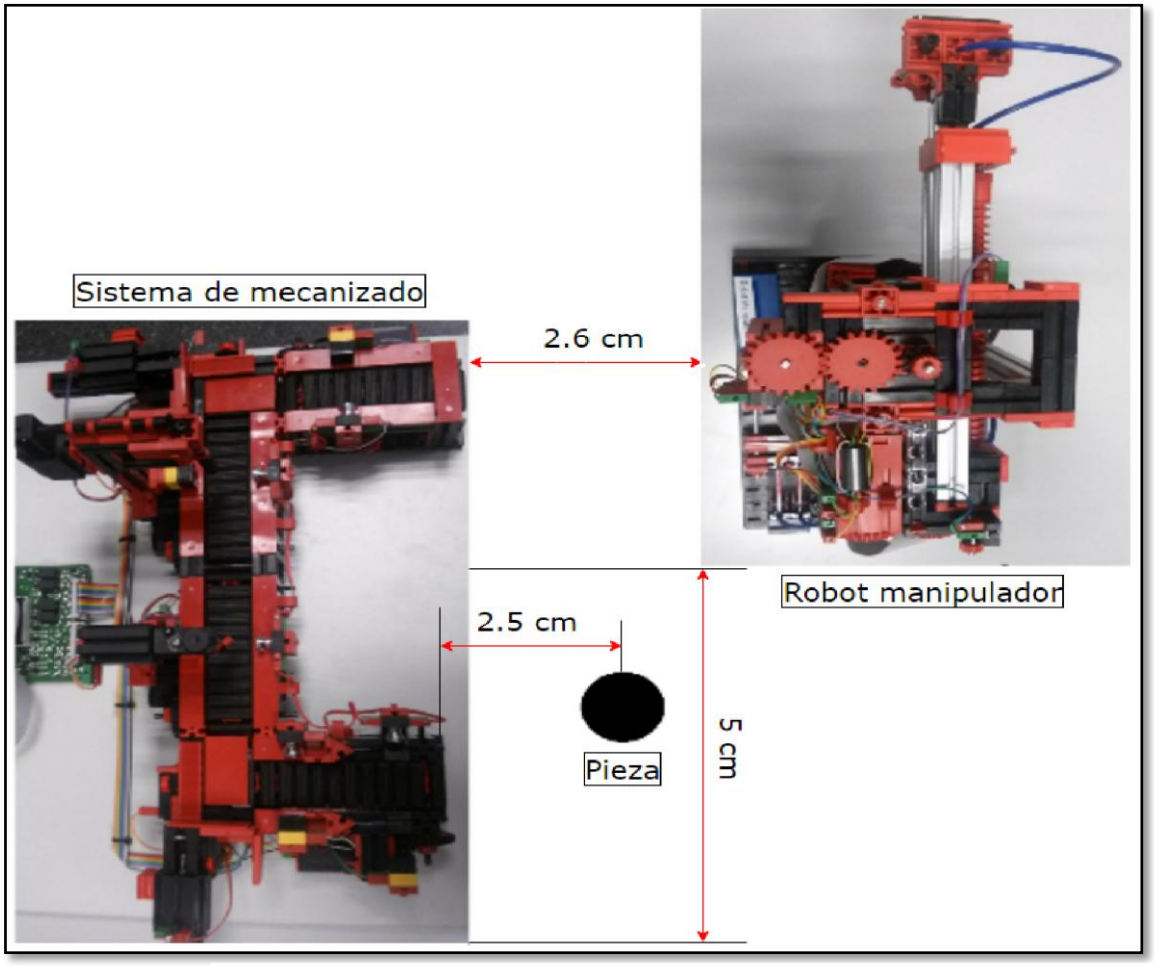


Figura 8. Distribución de los sistemas FischerTechnik (distancia entre ambos).

4.3 Sistema de mecanizado

4.3.1 Asignación de entradas y salidas (E/S)

A continuación se exponen las variables de entradas y salidas asociadas a los sensores y actuadores, respectivamente:

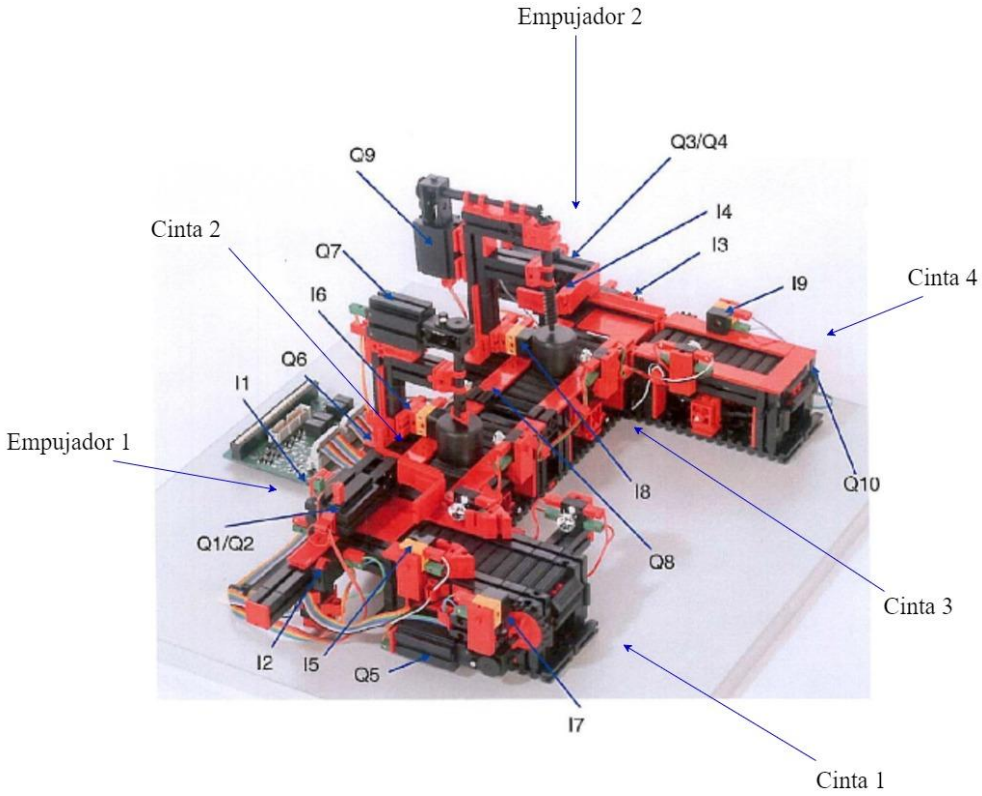


Figura 9. Asignación de E/S y referencia a actuadores.

Entrada	Descripción
I1	Final de carrera frontal del empujador 1
I2	Final de carrera trasera del empujador 1
I3	Final de carrera frontal del empujador 2
I4	Final de carrera trasera del empujador 2
I5	Fototransistor empujador 1
I6	Fototransistor fresadora
I7	Fototransistor estación de carga
I8	Fototransistor taladradora
I9	Fototransistor cinta transportadora salida

Salida	Descripción
Q1	Motor empujador 1 hacia adelante
Q2	Motor empujador 1 hacia atrás
Q3	Motor empujador 2 hacia adelante
Q4	Motor empujador 2 hacia atrás
Q5	Motor cinta transportadora de alimentación
Q6	Motor cinta transportadora fresadora
Q7	Motor fresadora
Q8	Motor cinta transportadora taladradora
Q9	Motor taladradora
Q10	Motor cinta transportadora salida
Q11	Habilitar sensores y empujadores

Tabla 9. Descripción de E/S del sistema de mecanizado..

4.3.2 Funcionamiento

La tarea del sistema es mecanizar tres tipos o modelos de piezas, distinguidas por su color (negro, azul o rojo). Dependiendo del tipo de pieza, se activará uno de los dos procesos de mecanizado, o bien ambos, tal y como se describe a continuación:

Color de pieza	Mecanizado
Negro	Fresadora Taladradora
Azul	Fresadora
Rojo	Taladradora

Tabla 10. Tipos de mecanizado.

El sistema empieza cuando se detecta una pieza en el sensor I7 (**véase apartado 4.3.1**), es decir, al inicio del sistema. La pieza es transportada por la cinta uno hasta llegar al primer émbolo o pistón, el cual empuja la pieza hacia la cinta dos. La cinta dos lleva la pieza hasta la fresadora. En este punto pueden darse dos situaciones:

1. Si la pieza es negra o azul, la fresadora se activa durante 5 segundos (el tiempo lo estipularía el cliente). Al terminar, la cinta dos y tres llevarían la pieza hasta la taladradora.
2. Si la pieza es roja, la fresadora **NO** se activará. En este caso, la cinta dos no se detiene cuando la pieza llega a la fresadora, sino que sigue activa (junto con la cinta tres) hasta llevar la pieza a la taladradora.

Cuando la pieza llega a la taladradora, se vuelve a tener dos casos:

1. Si la pieza es negra o roja, la taladradora se activa durante 5 segundos. Cuando termina, la cinta tres lleva la pieza hasta el émbolo o pistón dos.
2. Si la pieza es azul, la taladradora **NO** se activará. En esta situación, la cinta tres no se detiene cuando la pieza llega a la taladradora, sino que sigue activa hasta llevar la pieza al émbolo dos.

Una vez llegada la pieza al pistón dos, éste la empuja hasta llevarla a la cinta cuatro, la cual se encarga de llevar la pieza hasta el sensor I9 (salida del sistema).

Cabe mencionar que el sistema posee la capacidad de ponerse a cero (actuadores en el estado inicial) la primera vez que se ejecute, si lo requiriese (**véase apartado 4.3.3**). **Ejemplo:** Se inicia el sistema y el pistón 1 no está en la posición inicial (no está activado el final de carrera trasero I2).

IMPORTANTE tener en cuenta los siguientes puntos:

1. Si al poner una pieza en el inicio del sistema, hay una pieza en espera en el sensor I5, la cinta uno no se activará.

2. Si el pistón uno está empujando una pieza (1) y al mismo tiempo, hay otra pieza (2) en I5, la cinta uno no se moverá hasta que la pieza (1) haya pasado el proceso de fresado.

3. Si hay una pieza (1) en la taladradora y al mismo tiempo hay otra pieza (2) en la fresadora, la pieza (2) no será transportada hacia la taladradora hasta que la pieza (1) pase el proceso de taladrado.
   Nota: La pieza (2) se fresa (si es negra o azul) aunque haya otra en la taladradora.

4. La pieza que esté en el proceso de taladrado, no será transportada hacia la salida si en ésta ya hay una pieza (terminada y sin recoger).
   Nota: Aunque la pieza ubicada en el taladro no pueda ser transportada, la función de taladrado se efectuará (si la pieza es negra o roja).

Al tenerse en cuenta los puntos mencionados en la automatización del sistema, se evitan fallos en el mismo, mejorando la fluidez de fabricación y la seguridad tanto de las máquinas como de los operarios a cargo.

Para dar al sistema una mayor robustez y simplificar la programación del mismo, se ha dividido el funcionamiento en cuatro subprocesos, programados por separado en lenguaje SFC (basado en grafcet) pero teniendo en cuenta las condiciones necesarias para que se cumplan los puntos anteriormente mencionados. De tal modo que el funcionamiento se ha divido tal y como representa la siguiente imagen:

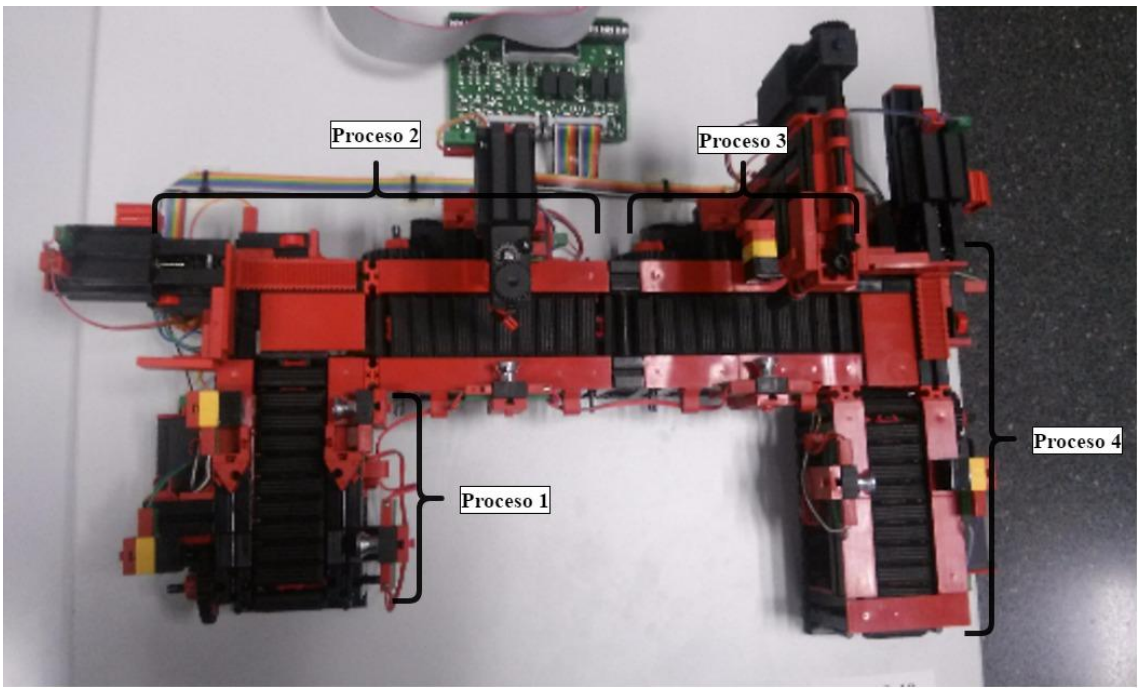


Figura 10. División de las tareas por procesos.

4.3.3 Inicio del sistema

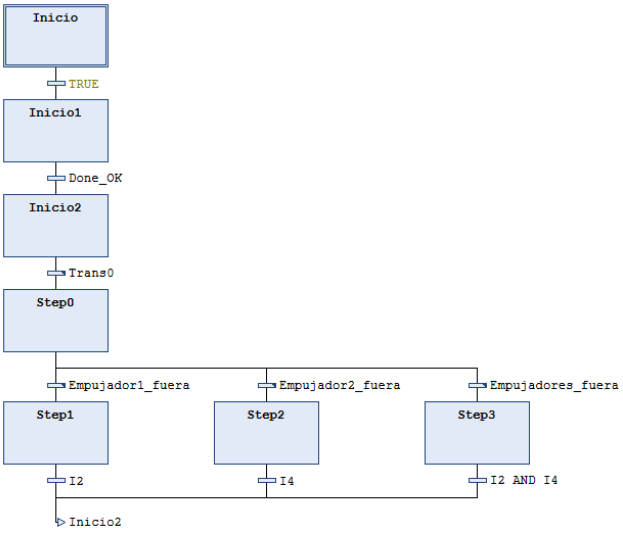


Figura 11. SFC del estado inicial del sistema.

Este programa SFC se encarga de meter los pistones hacia dentro en caso de que no lo estén cuando se inicia el sistema por primera vez (y cuando está en reposo). El bloque *Inicio1* se utiliza para activar mediante flanco de subida el habilitador para la comunicación entre los autómatas (**véase apartado 4.3.9**) y cuando el habilitador ha sido activado, pasa a *Inicio2*. En este bloque, simplemente se espera a ver si el sistema está en reposo y algún pistón no ha vuelto a su posición inicial. Si esto ocurre, pasa a *Step0*, donde se evalúa que pistón es el que ha fallado (o si han fallado los dos). *Step1* activa el motor para meter el pistón 1, *Step2* viene a ser lo mismo pero para el pistón 2 y *Step3* para ambos pistones sacados de su posición inicial. Estas tres etapas dejan de estar activas cuando el pistón vuelve a su posición inicial (I2 y/o I4 activo/s).

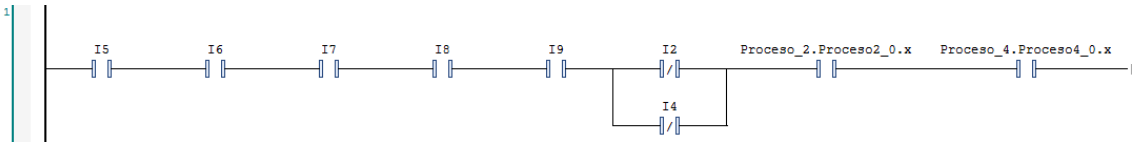


Figura 12. Transición *Trans0*, la cual se activa cuando el sistema está en reposo.

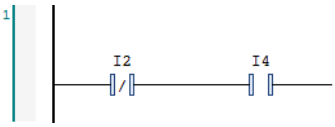


Figura 13. Transición *Empujador1_fuera.*

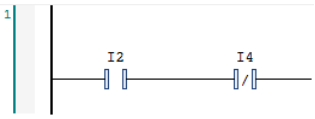


Figura 14. Transición *Empujador2_fuera.*

4.3.4 Proceso 1

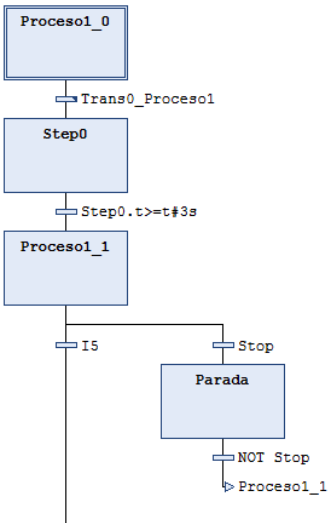
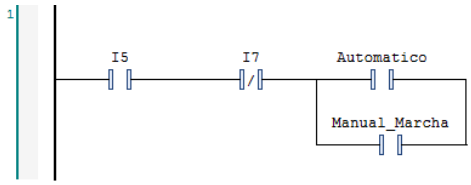


Figura 15. SFC del Proceso_1

El SFC mostrado se encarga de llevar la pieza desde el inicio del sistema hasta el sensor I5. El estado *Proceso1_0* espera hasta que I7 detecte una pieza y al mismo tiempo, no haya ninguna pieza en I5, permitiendo el paso hasta *Step0*. Este estado simplemente espera hasta que pasen 3 segundos, para dar tiempo al robot manipulador a retirarse del sistema de mecanizado después de depositar la pieza, antes de que la cinta 1 empiece a trasladar la pieza. El estado *Proceso1_1* activa la cinta 1 hasta que la pieza llega al sensor I5. Mientras la cinta1 está en funcionamiento (*Proceso1_1* activo), ésta puede ser parada mediante el botón "Parada de emergencia" del SCADA del sistema (**véase apartado 4.3.10,**

configuración y estado del proceso). Para reanudar la marcha solo requiere volver a pulsar dicho botón.



Cabe mencionar que debe estar activo el modo automático o manual del SCADA para que funcione el sistema (**véase apartado 4.3.10**).

Figura 16. Transición *Trans0_Proceso1.*

4.3.5 Proceso 2

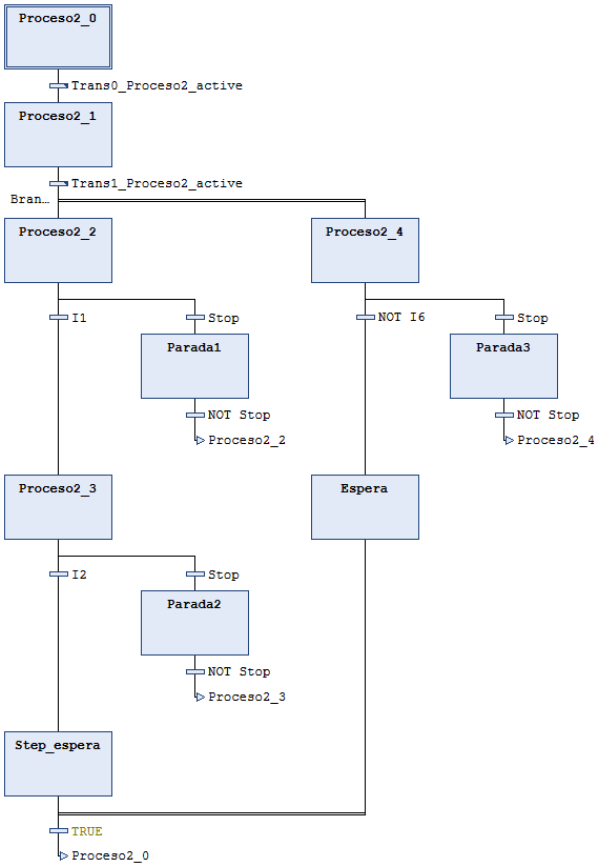


Figura 17. SFC de Proceso_2.

Este proceso trata de llevar la pieza desde I5 hasta la fresadora (contiene el sensor I6). El estado *Proceso2_0* espera hasta que haya una pieza en I5 y al mismo tiempo, que el pistón este dentro (I2 activo) y la fresadora esté libre, como también que el programa *Proceso1* esté en su estado inicial (para asegurarse que la cinta 1 está parada), además de tener activo el modo automático o manual. Al pasar al estado *Proceso2_1* se activa la cinta durante 0.9 segundos (para asegurarse que la pieza llega a la zona del pistón 1). Al franquearse esta transición (*Trans1_Proceso2_active*), se produce una bifurcación AND, realizándose dos tareas al mismo tiempo. Por una parte, el estado *Proceso2_2* activa el pistón 1 hasta que llega a su final de carrera delantero (I1 activo), posteriormente, *Proceso2_3* activa el pistón 1 hacia atrás hasta que llega a su final de carrera trasero (I2 activo). Por otro lado, *Proceso2_4* se encarga de activar la cinta 2 hasta que la pieza llega a la fresadora (I6 desactivado). Al terminar las dos tareas, esto es, *Step_espera* y *Espera* activos, se cierra la bifurcación AND y se vuelve al estado inicial. Cabe mencionar que se dispone de la opción "parada de emergencia" para las tareas de *Proceso 2*.

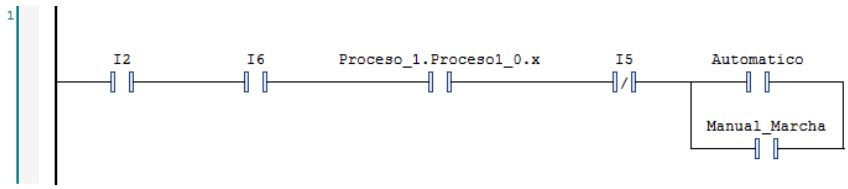


Figura 18. Transición *Trans0_Proceso2_active. Proceso_2* .

Figura 19. Transición *Trans1_Proceso2_active. Proceso_2.*

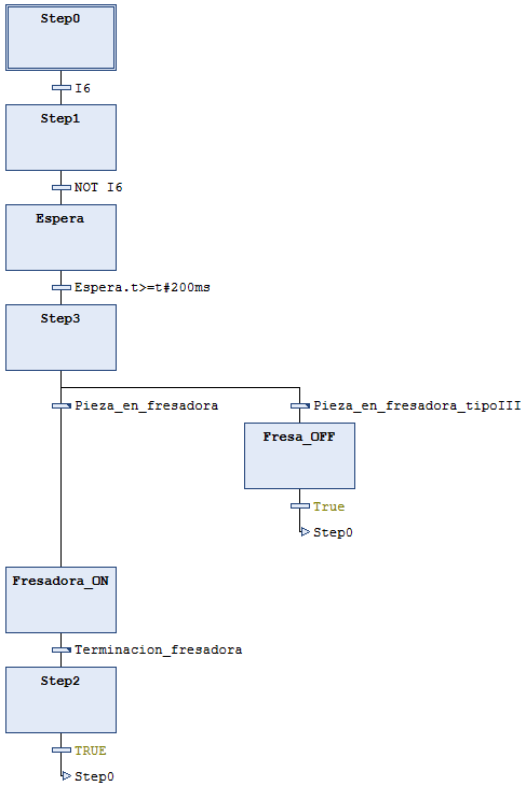


Figura 20. SFC de *Proceso_2_FRESADORA.*

Se encarga de activar la fresadora en función del tipo de pieza. Inicialmente, *Step0* está en reposo hasta que no haya pieza en la fresadora, esto es debido a que si no se pusiera tal condición, si hubiera una pieza en la taladradora, la pieza en la fresadora permanecería en el sitio, realizando otra vez la acción de fresar. *Step1* espera hasta que se detecte pieza en la fresadora. *Espera* es un estado que está activo durante 0.2 segundos, para darle tiempo a la función *First input First output* (FIFO) a desencolar (**véase apartado 4.3.8**). *Step3* espera hasta que se cumpla una de las dos transiciones. *Pieza_en_fresadora* se activa cuando hay pieza en la fresadora y el sistema ya sabe qué tipo de pieza tiene en la fresadora (ya sea mediante la cámara o por orden del operario). Si el tipo de pieza fuera el tres (rojo), se franquea la transición *Pieza_en_fresadora_tipoIII* y se vuelve al estado inicial. Si la pieza es negra o azul, se pasa al estado *Fresadora_ON,* la cual activa la fresadora durante 5 segundos, después vuelve al estado inicial (*Step0*).

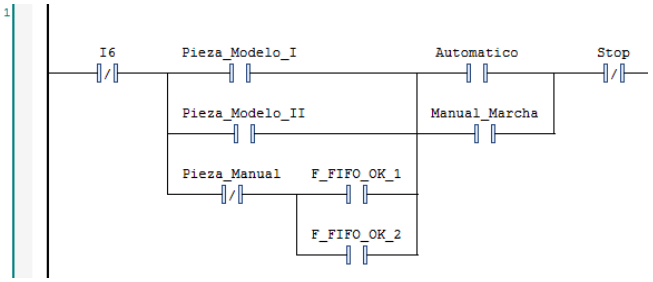


Figura 23. Transición *Pieza_en_fresadora. Proceso_2_FRESADORA.*

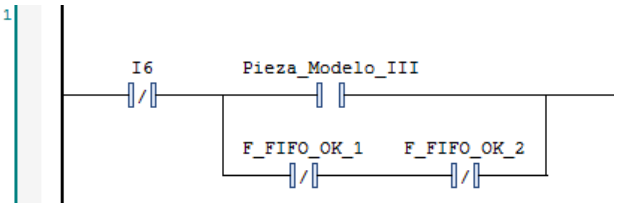


Figura 22. Transición *Pieza_en_fresadora_tipoIII. Proceso_2_FRESADORA.*

Figura 21. Transición *terminación fresadora. Proceso_2_FRESADORA.*

4.3.6 Proceso 3

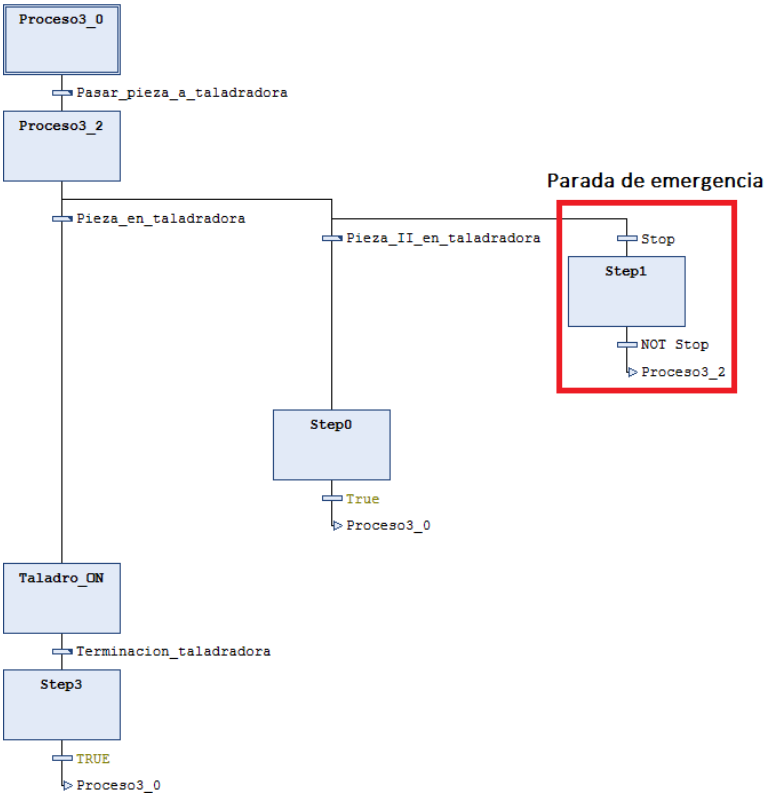


Figura 24. SFC de *Proceso_3.* Compone el Proceso 3.

El estado *Proceso3_0* (estado inicial) está activo hasta que la taladradora esté libre, en la fresadora haya una pieza y dicho mecanizado haya terminado (el SFC de *Proceso_2_FRESADORA* tenga activa su etapa inicial). Al cumplirse la primera transición, *Proceso3_2* se encarga de activar las cintas 3 y 4, para transportar la pieza hasta la taladradora. Una vez llegada la pieza al taladro, el sistema ya debe saber de qué pieza se trata (negra, azul o roja), y en función al tipo, franqueará una transición u otra. En el caso de *Pieza_II_en_taladradora,* significa que la pieza es azul, por lo tanto el Proceso 3 termina y vuelve a su estado inicial. En el caso de franquearse *Pieza_en_taladradora*, se activa la etapa *Taladro_ON*, donde se activa el taladro durante 5 segundos para, posteriormente, volver a su estado inicial, esto es, *Proceso3_0*.

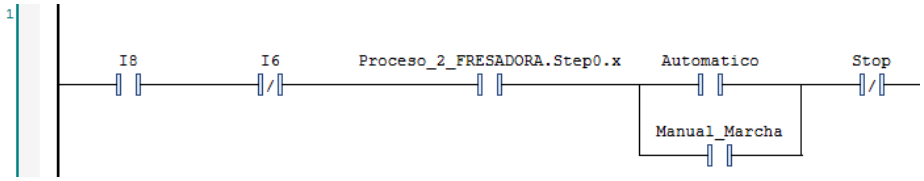


Figura 25. Transición *Pasar_pieza_a_taladradora.* Proceso_3.

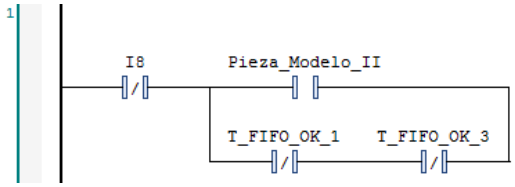


Figura 26. Transición *Pieza_II_en_taladradora.* Proceso_3.

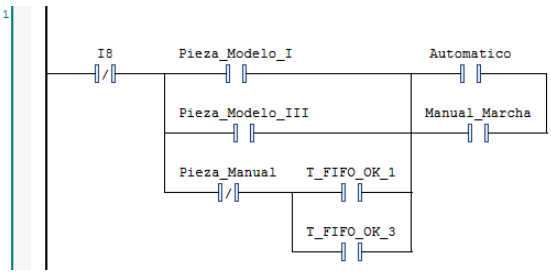


Figura 27. Transición *Pieza_en_taladradora.* Proceso_3.

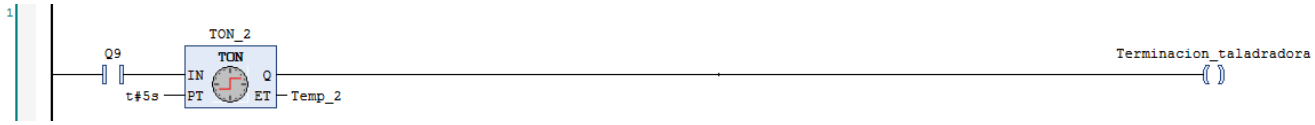


Figura 28. Transición *Terminacion_taladradora.* Proceso_3.

4.3.7 Proceso 4

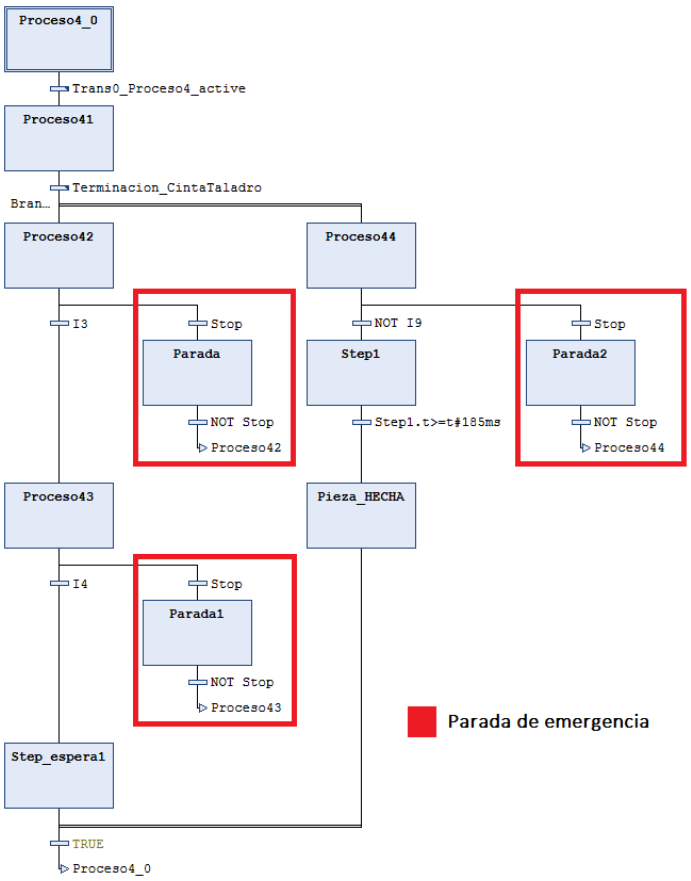


Figura 29. SFC de *Proceso_4. Compone el Proceso 4.*

El proceso empieza a avanzar cuando hay una pieza en la taladradora y a su vez, que dicho proceso haya terminado (*Proceso_3* en su etapa inicial), como también que no haya ninguna pieza en la salida del sistema (sensor I9 a 1 lógico o HIGH). La etapa *Proceso41* se encarga de activar la cinta 3 durante 1.2 segundos, tiempo necesario para llevar la pieza al pistón 2. Al franquear la transición *Terminacion_CintaTaladro*, se produce una bifurcación AND, ejecutándose varias tareas simultáneamente. Por una parte, *Proceso42* y *Proceso43* se encargan de activar hacia delante y hacia atrás el pistón 2, respectivamente. Por otro lado, *Proceso44* se encarga de activar la cinta 4 hasta que la pieza sea detectada por I9 (0 lógico o LOW). Como el robot manipulador no llega (por escasos milímetros) hasta la pieza, situada en I9, para su recogida, se ha añadido una etapa llamada *Step1* la cual activa la cinta 4 durante 0.185 segundos. De esta manera, la pieza no parará en el centro de I9, sino que parará un poco más en el borde (asegurandose que I9 sigue detectando la pieza). Al terminar ambas tareas, esto es, *Step_espera1* y *Pieza_HECHA* activas, el programa vuelve a su etapa inicial.

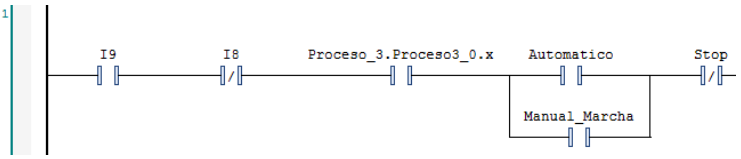


Figura 30. Transición *Trans0_Proceso4_active. Proceso_4.*

Figura 31. Transición *Terminacion_CintaTaladro. Proceso_4.*

4.3.8 Función *First Input First Output* (FIFO)

La función FIFO, integrada en *SoMachine* por los desarrolladores, sirve para guardar datos en un *array* que posteriormente se van a utilizar, según su orden de entrada.

Funciona igual que una lista, de manera que se guardan los datos (escritura) según su orden de llegada, es decir, el primer dato se guarda en la primera posición, el segundo dato en la segunda posición y así sucesivamente. A la acción de guardar el dato, se le ha llamado "encolar".

Cuando se quiere utilizar un dato de la FIFO (lectura), se saca dicho dato en el mismo orden en el que se ha encolado, es decir, el primer dato a extraer será el primero que se ha encolado, el segundo dato será el segundo que se ha encolado, y así sucesivamente. A esta acción se le ha denominado "desencolar".

La función FIFO en el proyecto sirve para que el sistema sepa de qué tipo es la pieza, atribuyendo desde *LabVIEW* (**véase apartado 4.6.2**) un valor numérico dependiendo del color (negro = 1, azul = 2, rojo = 3). De este modo, al detectar la cámara el color de la pieza, se manda por el servidor OPC hasta el programa en *SoMachine*, el número asociado a ese color detectado. Dicho número se almacena en la FIFO, de manera que, a medida que se van detectando piezas, la FIFO va almacenando sus valores. Así, se puede determinar qué tipo de pieza es la primera en llegar al mecanizado, de que tipo es la segunda, tercera, etc.

Cabe destacar que se han utilizado dos FIFO para el mecanizado, una asociada para la fresadora y otra asociada a la taladradora, siendo de este modo dos procesos independientes y facilitando el uso de los datos de las FIFO.

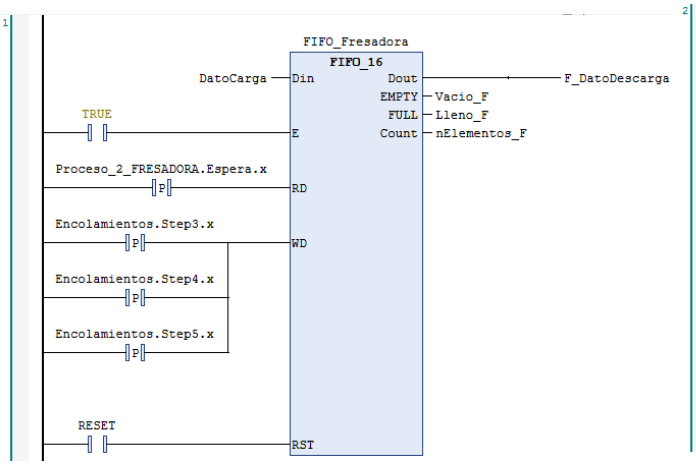


Figura 32. Función FIFO de la fresadora. *FIFO_Fresadora, diagrama de contactos.*

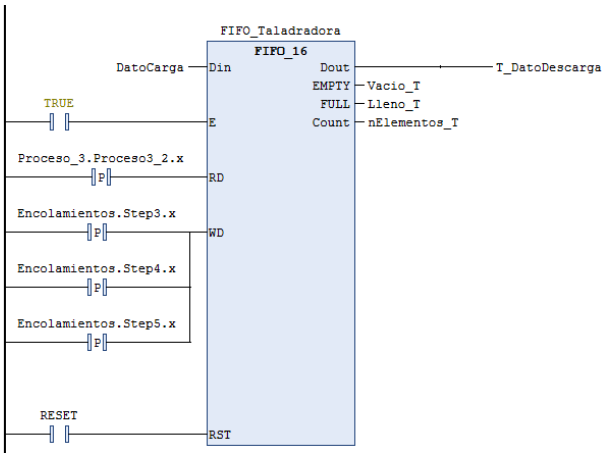


Figura 33. Función FIFO de la taladradora. *FIFO_Taladradora, diagrama de contactos.*

DatoCarga viene a ser la variable que recibe el número asociado al color de la pieza (variable de entrada), mientras que *F_DatoDescarga* y *T_DatoDescarga* son las variables de salida que leen los valores de la FIFO de la fresadora y taladradora, respectivamente.

A la entrada **WD** (escritura) se le asocian las etapas donde se quiere que se encole un dato en la FIFO, de manera que hasta que alguna de sus etapas asociadas no se active mediante flanco ascendente, no se encolará en la FIFO el valor recibido en *DatoCarga*. Por otro lado, a la entrada **RD** (lectura) se le asocia la

etapa donde se quiere extraer y leer un dato de la FIFO, de tal modo que, hasta que no se active la etapa asociada a RD, no se desencolará el primer dato de la FIFO en la variable de salida.

Describiéndolo a grandes rasgos, se encola cuando la cámara detecta el color de la pieza, y se desencola antes de ejecutar la fresadora y la taladradora.

La variable *RESET* sirve para borrar los datos de la FIFO, lo cual no es necesario ya que la FIFO utilizada puede almacenar hasta 16 datos (existe FIFO para 32 datos), siendo más que suficiente ya que la línea indexada puede tener en su sistema hasta 5 piezas (una por cada fototransistor).

La entrada **E** debe estar a true, ya que es el *enable*, el cual debe estar activo para que funcione la FIFO.

Las entradas ***EMPTY, FULL*** y ***Count*** sirven para indicar cuando la FIFO está vacía, llena, y el número de datos que tiene almacenado, respectivamente.

4.3.9 Comunicación entre autómatas (Maestro / Esclavo)

Para la comunicación entre los dos Modicon M241, se ha utilizado dos funciones ya desarrolladas en *SoMachine*, las cuales guardan relación una con la otra. Dichas funciones son las representadas en la siguiente figura:

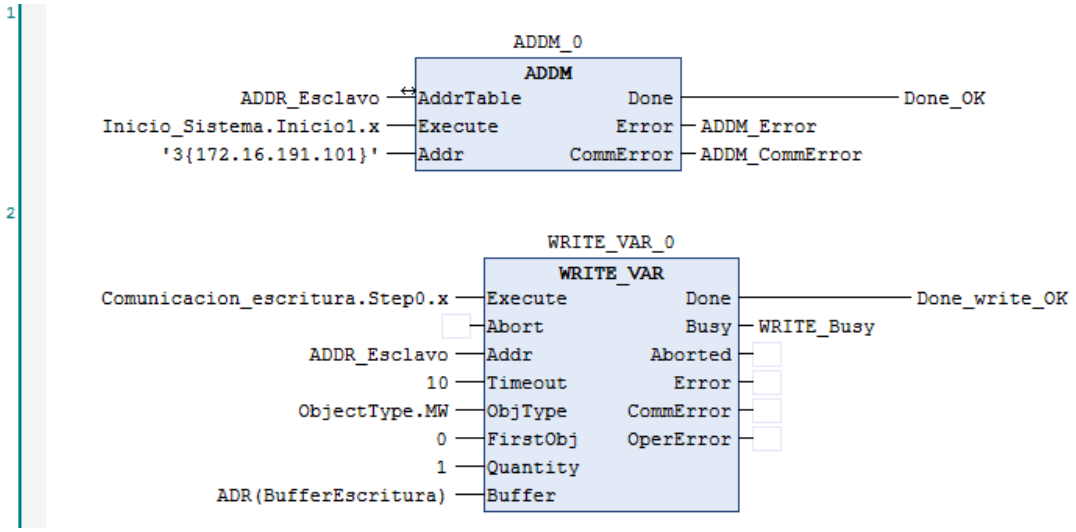


Figura 34. Comunicación entre autómatas mediante IP. Bloque *ADDM y WRITE_VAR.* ComunicacionesMaestro, Diagrama de bloques.

Bloque ADMM:

Sirve para configurar, desde un autómata a otro, la conexión mediante IP, como también para habilitar la lectura y/o escritura de datos.

La variable *ADDR_Esclavo*, asociada a la entrada *AddrTable*, sirve para nombrar al autómata esclavo desde el autómata donde se han implementado las funciones de comunicación (autómata maestro).

La entrada *Execute*, asociada al estado *Inicio1* del SFC *Inicio_Sistema*, sirve para habilitar la comunicación, una vez dicha etapa se active (solo requiere la activación una sola vez).

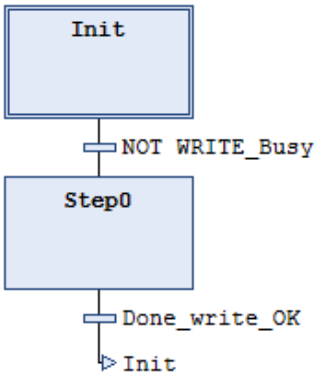
Nota: Si se hace un *reset* en frio o en caliente del sistema, durante el primer ciclo de ejecución es probable que no se habilite las comunicaciones.

En la entrada *Addr* se especifica la dirección IP del autómata al cual se va a establecer la conexión (esclavo), siguiendo la nomenclatura especificada " '3{Dirección IP}' ".

La salida *Done_OK* sirve para verificar que la habilitación de la comunicación se ha realizado correctamente. Es muy importante que esto ocurra, si no, podría dar problemas a posteriori. Por este motivo, la comprobación se realiza en el SFC *Inicio_Sistema*, para comprobar que la comunicación se ha habilitado antes de empezar el proceso de mecanizado.

Bloque *WRITE_VAR*

Sirve para escribir desde el autómata maestro, al autómata esclavo. Para realizar la escritura de forma continua y cíclica (sin dependencia de ningún evento, para evitarse retardos y problemas), se ha asignado a la entrada Execute (activa la escritura) la etapa *Step0* del SFC *Comunicación_escritura*. Dicho SFC se muestra en la siguiente figura:



Sirve para realizar la escritura al esclavo constantemente. La variable de salida *WRITE_Busy* indica cuando se está escribiendo en el esclavo (TRUE) y cuando se ha terminado con la escritura (FALSE). De esta manera, este SFC, ejecutándose de forma cíclica, activa la escritura cuando el bloque *WRITE_VAR* ha terminado de escribir (está en reposo), esto es, *WRITE_Busy = FALSE.* Se franquea la transición y se activa la etapa *Step0*, que como ya se comentó anteriormente, activa la escritura.

Figura 35. SFC de Comunicación_escritura.

Al igual que con el bloque ADDM, la salida *Done,* asociada a la variable *Done_write_OK*, sirve para advertir de cuando se ha terminado de realizar la escritura correctamente. De este modo se asegura la escritura en el esclavo de forma cíclica y sin problemas.

Para referirse a que dirección de memoria del esclavo se escribe el dato, se utilizan las entradas *ObjType, FirstObject* y *Quantity*. En la primera entrada, se describe el tipo de dirección (MW, Q), en la segunda, la primera dirección de memoria y en la tercera, el número de direcciones de memoria contando desde la primera especificada.

En este caso, se especifica el tipo de dirección MW, como primera dirección la 0, por lo tanto, tenemos la dirección de memoria MW0. Ahora bien, se especifica únicamente un 1 en la entrada *quantity*, de manera que solo se quiere escribir en la primera dirección de memoria contando desde la primera, es decir, la misma dirección (MW0).

Ejemplo: Si *quantity* tuviera el valor 3, se escribiría en las direcciones de memoria MW0, MW1 y MW2.

Ahora bien, el dato a transmitir al esclavo debe estar asociado a la entrada "*Buffer" del* WRITE_VAR. Dicha entrada debe tener asociada un *array* como variable (este *array* es, en este caso, de dos posiciones

Cabe mencionar que la dirección de memoria MW0 contiene, a nivel de *byte*, 2 direcciones y, a nivel de *bit*, 8 direcciones por cada *byte* (en total 16 *bits*), tal y como muestra la siguiente figura:

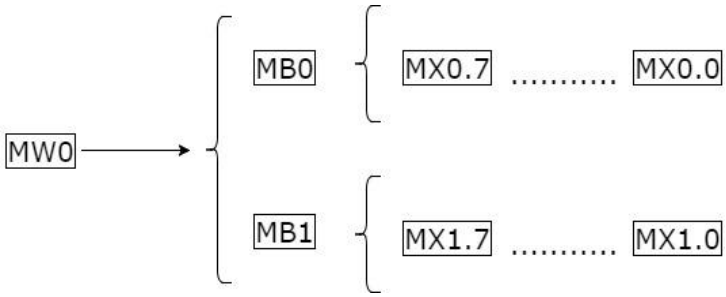


Figura 36. Direcciones de memoria por niveles (*byte* y *bit*).

Como se necesita que el robot manipulador sepa cuando ha terminado de mecanizarse una pieza para ir a por ella, el evento que producirá dicha acción por parte del robot, será el cambio en I9 al detectar una pieza mecanizada. Por lo tanto, el código a implementar sobre el *array BufferEscritura*, en texto estructurado (ST), es el siguiente:

```
36  //Intercambio datos plc1 a plc2 (mecanizado a brazo robot)
37
38  IF I9 = FALSE THEN
39      BufferEscritura[0] := 2; // Corresponde en binario al valor 00000010 , siendo en el plc2 MX0.7....MX0.0, siendo MX0.1 = 1
40  ELSIF I9 = TRUE THEN
41      BufferEscritura[0] := 0;
42
43  END_IF
```

Figura 37. Código en texto estructurado del dato a transmitir al robot manipulador (esclavo).

Lo que indica este código, es que, cuando I9 detecta pieza (FALSE), se escribe en la posición 0 del *buffer* un 2 (podría haber sido cualquier otro número). Dicho número, al ser una variable tipo WORD (es como un entero) se almacena en la dirección MW0 del autómata esclavo. Sin embargo, para facilitar la programación, se requiere trabajar con datos booleanos para ejecutar condiciones, por ello, hay que realizar una conversión de número decimal a binario, para trabajar a nivel de bit, y por tanto, con datos booleanos (1 o 0).

En el caso del 2, su número en binario es el 00000010, correspondiendo el bit más significativo (*Most Significant*, MS) al 0 de la izquierda y el bit menos significativo (*Less Significant*, LS), al 0 de la derecha. Y como bien indica la figura 36, el bit MS corresponde a la dirección de memoria MX0.7, y el LS al MX0.0.

De este modo, se deduce que cuando I9 detecte una pieza, se activará con un 1 lógico la dirección de memoria MX0.1 del autómata esclavo, y cuando el robot manipulador haya retirado la pieza de la cinta 4 (I9 a TRUE), el *buffer* adquirirá el valor 0 y por lo tanto, la dirección de memoria MX0.1 del esclavo se pondrá a 0.

4.3.10 Sistema SCADA con *SoMachine*

Para monitorizar el sistema se han diseñado varios SCADA, accesibles directamente desde el propio ordenador a través de *SoMachine,* como también desde otros dispositivos conectados a la misma red que los autómatas.

El SCADA está compuesto por tres HMI, un registro y acceso de usuario, el estado de fabricación y el estado del sistema.

Registro y acceso de usuario

En esta HMI, el usuario debe registrarse con un nombre y una contraseña, además de una clave de acceso (proporcionada por el fabricante) que tendrá que introducir para validar su usuario. Una vez registrado, ya podrá utilizar el resto de interfaces.

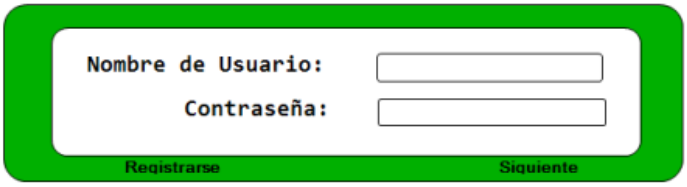


Figura 38. Acceso de usuario para la supervisión y control del sistema.

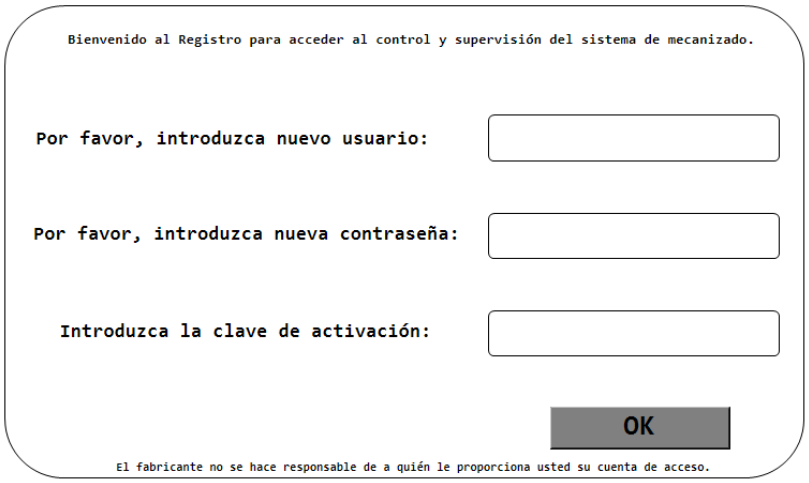


Figura 39. Registro de usuario con clave de activación.

En la figura 37 el usuario debe acceder al registro mediante el botón “Registrarse”, donde le aparecerá la imagen mostrada en la figura 38. Aquí se debe introducir el usuario y contraseña que la persona propietaria desee. También debe introducir la clave de activación proporcionada por el fabricante, siendo en este caso, A1B2C3 (implementada en el código).**Nota:** Sin la clave de activación, el registro no será efectuado correctamente.

La conmutación entre pantallas se realiza gracias a las variables *Registro* y *Registro_aux*. La pantalla de acceso a las demás interfaces es visible cuando la variable *Registro* está a *FALSE*, y la pantalla de registro cuando la variable *Registro_aux* está a *FALSE.* Siendo la variable *Registro* conmutada cuando se presiona el botón “Registrarse” o el botón “OK”.

Figura 40. Conmutación entre las variables *Registro* y *Registro_aux* para la visibilidad/invisibilidad de las pantallas de acceso y registro de usuario. Diagrama de contactos de *Registrar.*

Una vez finalizado el registro, el usuario ya puede introducir su cuenta y pasar a las interfaces de control pulsando “Siguiente”.

```
1
2   IF a = 0 THEN
3       a := 1;
4       Registro := TRUE;
5       No_Registro := FALSE;
6       clave := 'A1B2C3';
7       Usuario := '';
8       Registro_usuario := '  ';
9       Password := '';
10      Registro_password := '  ';
11
12  END_IF
```

Programa en texto estructurado para la inicialización de las variables de la interfaz de registro, donde la clave de activación dada por el fabricante viene implementada en dicho código. Este código solo se ejecuta una vez.

Figura 41. Programa en ST para la inicialización de las variables de registro y activación de SCADA.

Figura 42. Verificación de los datos introducidos en pantalla con el registro previo. Diagrama de contactos de *Registrar.*

En la figura 41 se muestra la comprobación con bloques de igualdad (válido tanto para números como cadena de caracteres) de los datos introducidos en pantalla (**Figura 37**) con los introducidos en el registro (**Figura 38**).

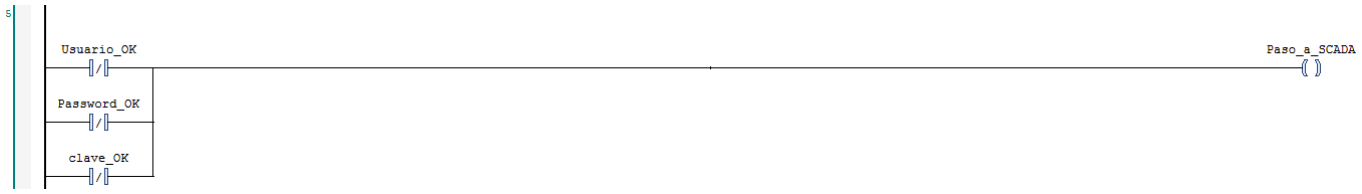


Figura 43. Permiso de uso del botón "Siguiente" cuando todos los datos son correctos. Diagrama de contactos de *Registrar.*

En la figura 43 se muestra la condición para que se pueda pulsar el botón "Siguiente" y acceder a las demás interfaces. El botón "Siguiente" está asociado a la variable *Paso_a_SCADA*, de manera que si dicha variable está a *TRUE*, el botón no podrá ser pulsado. Para que dicha variable esté desactivada (*FALSE*), las tres variables asociadas deben estar a *TRUE* (al ser negadas, serían *FALSE*), es decir, los datos introducidos en la pantalla de verificación y acceso deben ser iguales a los introducidos en el registro.

IMPORTANTE: El registro solo se puede realizar una vez, siendo imposible volver a realizar otro registro una vez ya se ha accedido a las HMI por primera vez. Esto se ha conseguido gracias al programa mostrado en la imagen siguiente:

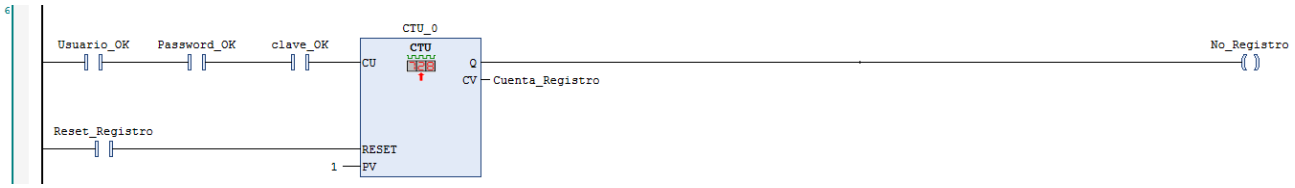


Figura 44. Cancelación de nuevos registros una vez se ha accedido a las demás HMI por primera vez. Diagrama de contactos de *Registrar.*

Configuración y estado de fabricación

Esta HMI le da la opción al operario de elegir manualmente el tipo de pieza a fabricar, ignorando por completo los datos que transmita la cámara al sistema (debe tenerse especial cuidado al elegir el fabricado manual para evitar problemas). Además, se visualiza el número fabricado de cada modelo de pieza, como también el número de piezas (material) disponibles en el almacén (para ser mecanizadas).

Nota: Solo se podrá seleccionar el modelo de pieza manualmente si el sistema está en reposo (para evitar fabricar piezas no deseadas a mitad de proceso por error).

Figura 45. Interfaz de usuario de la fabricación del sistema.

Para poder seleccionar el modelo de pieza manualmente, primero se ha de pulsar el botón "Manual", dentro de la sección "Fabricación de los modelos". También existen botones de *reset* para cada contador de los modelos de pieza. En caso de querer visualizar el HMI del proceso, basta con pulsar el botón "Proceso".

Para la programación de los contadores de los modelos y de las piezas disponibles, **véase anexo 1**.

Cabe mencionar que solo se puede tener pulsado un modelo de pieza a fabricar, si se pulsara otro estando otro modelo activo, simplemente conmutaría. Para que el sistema funcione de tal manera, ha sido necesario implementar el programa mostrado en la siguiente figura:

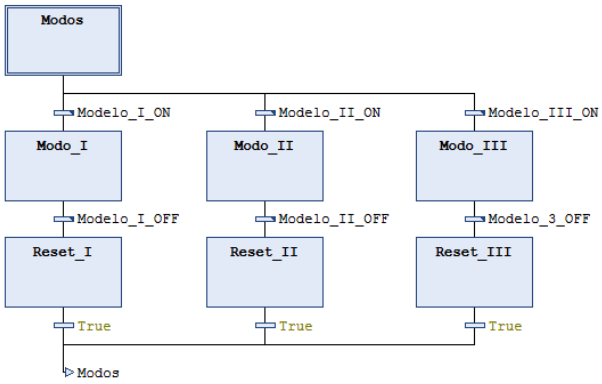


Figura 46. SFC de *Modos_Fabricación*, conmutación entre pulsadores para evitar tener pulsado más de uno a la vez por error.

Al activarse un modo de funcionamiento de los tres posibles, se activa uno de los tres estados posibles. Dichos estados son de reposo, hasta que se desactiva el modo previamente seleccionado o bien se cambia de modo sin desactivar el modo actual. Los estados de *reset* tienen por función desactivar su modo de fabricación correspondiente, produciéndose de este modo la conmutación entre pulsadores.

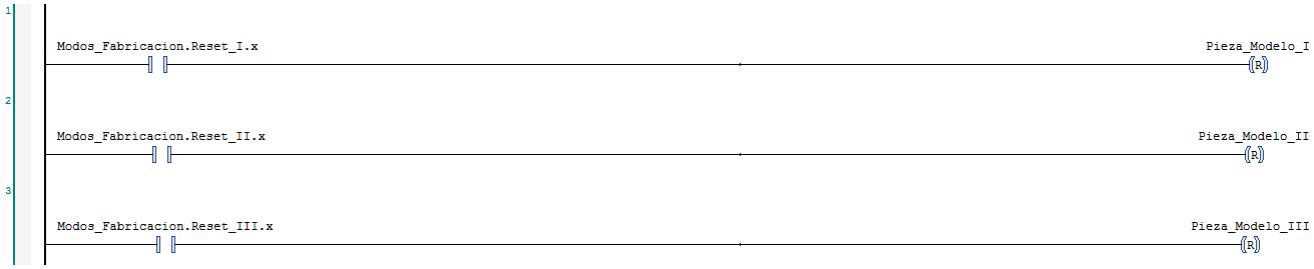


Figura 47. Diagrama de bloques de *Variables_Piezas*. Desactivación del modo correspondiente en los estados de *reset*.

Configuración y estado del proceso

Esta interfaz permite visualizar el estado actual del sistema de mecanizado, como también controlarlo en cualquier momento.

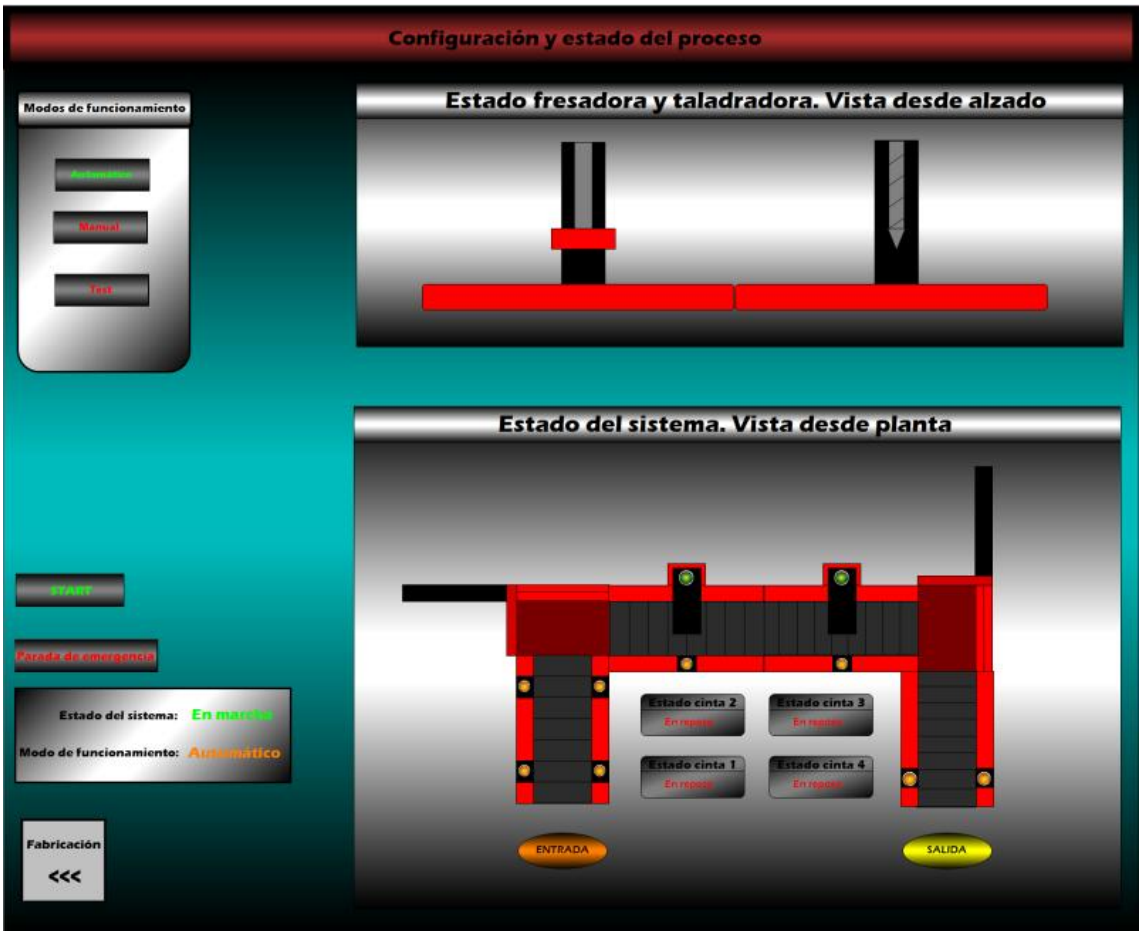


Figura 48. Interfaz de usuario de la configuración y estado del proceso. Modo automático.

Para que el sistema funcione, debe tener elegido un modo de funcionamiento y para ello, primero debe estar pulsado el botón *START.* Dependiendo del modo elegido, el sistema funcionará de una manera u otra:

- **Modo automático:** Permite al sistema ejecutarse constantemente sin ningún tipo de parada (excepto la de emergencia, pulsada por el operario).

- **Modo manual:** El sistema se ejecuta paso a paso, esperando la orden del operario (botón manual). Perfecto para detectar fallos (sobre todo de programación) y realizar pruebas sobre las piezas. **Nota:** Al activar el botón manual, aparece un pulsador. Dicho pulsador es el que debe ser pulsado para que el sistema funcione paso a paso. Al volver a pulsar el botón manual, el pulsador desaparece de la pantalla.

- **Modo test:** Permite al operario activar cualquier actuador del sistema (cintas, émbolos, fresadora o taladradora), o bien por separado o a la vez, según necesidades. Perfecto para revisiones de los actuadores y sensores. **Nota:** Al pulsar sobre el modo test, se abre una pestaña con los botones o pulsadores de cada actuador.

En las siguientes figuras se muestran algunos ejemplos para consolidar las funciones explicadas anteriormente:

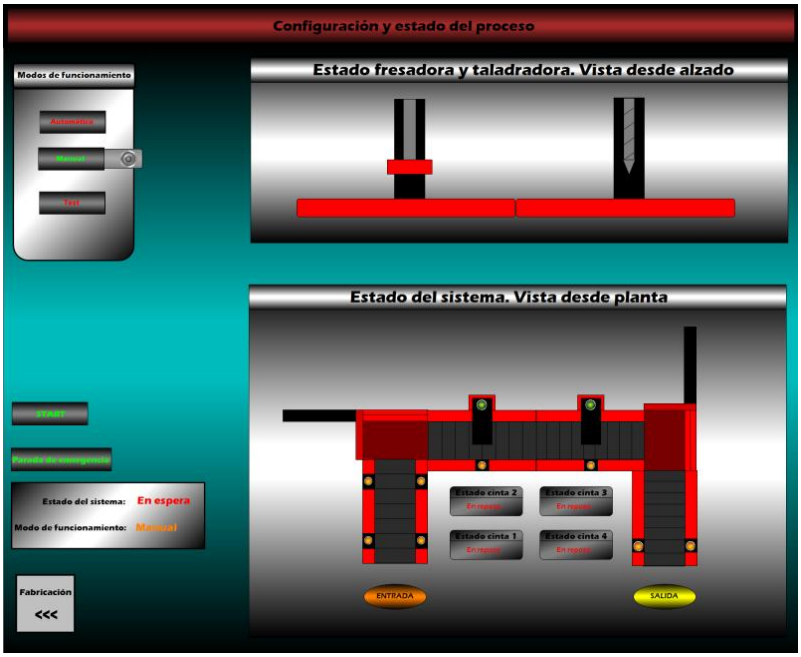


Figura 49. Interfaz de usuario de la configuración y estado del proceso. Modo manual.

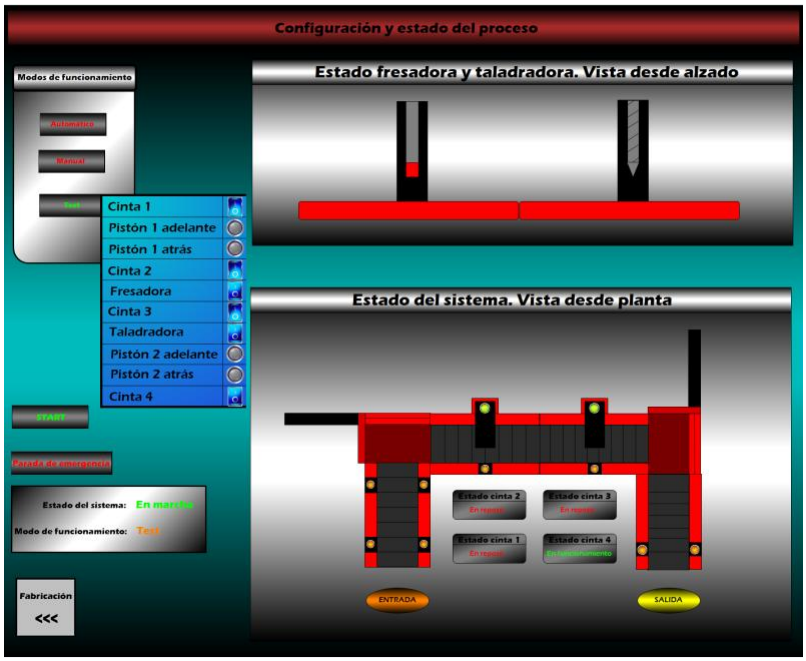


Figura 50. Interfaz de usuario de la configuración y estado del proceso. Modo test, fresadora, taladradora y cinta 4 en funcionamiento.

En las figuras mostradas anteriormente, se puede observar como debajo del botón de parada de emergencia hay una pequeña pantalla donde indica en qué estado se encuentra el sistema. Si el sistema está funcionando, se encuentra "En marcha", en caso de haber pulsado la parada de emergencia, el estado pasará a ser "En espera". Del mismo modo, también muestra el modo de funcionamiento elegido.

Cabe destacar que si se quiere volver a la interfaz de fabricación, existe en el borde inferior izquierdo de la visualización un botón llamado "Fabricación", el cual redirecciona directamente a la otra interfaz del sistema.

Para realizar esta interfaz se ha utilizado un gran número de variables (definidas y comentadas en el **anexo 2**) para la conmutación de letreros como también de imágenes para simular el movimiento de los actuadores. Se muestra, por tanto, algunas partes del programa realizado para dichas conmutaciones (para ver la programación completa, **véase el anexo 1**).

Figura 51. Diagrama de contactos de *Variables_Visu_proceso. Conmutación entre los bordes de la fresadora (cuadrado y rectángulo rojo).*

La variable *Fresadora_A* está asociada a una imagen la cual representa un cuadrado rojo y la variable *Fresadora_B* a un rectángulo rojo. Cuando *Fresadora_A* está desactivada, se muestra el cuadrado rojo y al mismo tiempo, el rectángulo rojo está invisible (*Fresadora_B* activa). Al activarse la fresadora (Q7), se activa un timer (TON_7) a los 0.5 segundos y las variables *Fresadora_A* y *Fresadora_B* conmutan sus valores, de este modo el rectángulo rojo se hace visible y el cuadrado rojo invisible.

El mismo procedimiento descrito ocurre pero a la inversa. De este modo, al ser las conmutaciones de 0.5 segundos, las imágenes cambian tan rápido y se consigue una simulación del movimiento de la fresadora (el procedimiento es el mismo para la taladradora, pero con diferentes variables).

Otro caso sería la conmutación del texto referente a los modos de funcionamiento. En este caso se muestra para el modo test:

Figura 52. Diagrama de contacto de *Variables_Visu_proceso. Conmutación entre visible e invisible del texto referente al modo test.*

Para que el letrero de "Test" aparezca en naranja en la pantalla donde se muestra el estado y modo de funcionamiento del sistema, deben darse tres casos:

- Modo automático activado
- Modo manual activado
- Ningún modo activado

Si se produce alguno de los casos mencionados, la variable *Texto_test* se activará (*TRUE*) y por lo tanto el texto "Test" será invisible. Si ninguno de los tres casos se produce, significará que el modo test estará activo, por lo tanto la variable *Texto_test* estará desactivada (*FALSE*) y el texto "Test" será visible.

El procedimiento para los demás textos ("Automático" y "Manual") es exactamente el mismo (**véase anexo 1**).

4.4 Robot manipulador (brazo robótico)

4.4.1 Asignación de entradas y salidas (E/S)

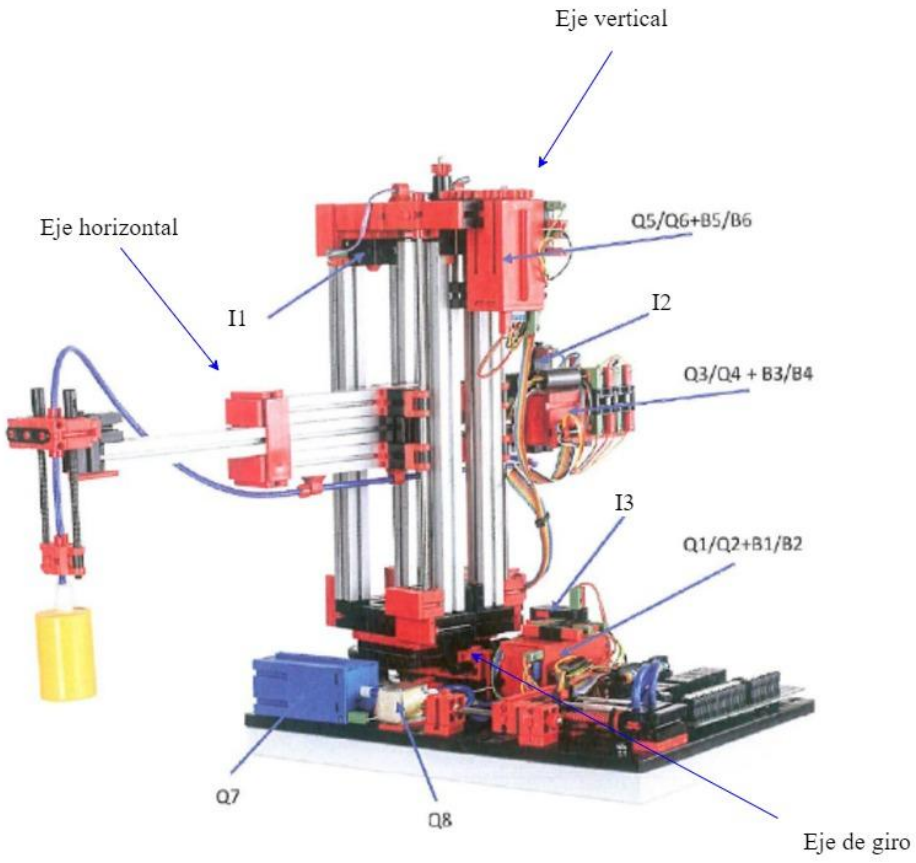


Figura 53. Asignación de las E/S y señalización de los ejes.

Entrada	Descripción
I1	Final de carrera referencia vertical
I2	Final de carrera referencia horizontal
I3	Final de carrera referencia giro
B1	Pulsos encoder movimiento vertical
B3	Pulsos encoder movimiento horizontal
B5	Pulsos encoder movimiento giratorio

Tabla 11. Descripción de las entradas del robot manipulador.

Salida	Descripción
Q1	Motor movimiento vertical arriba
Q2	Motor movimiento vertical abajo
Q3	Motor movimiento retroceso horizontal
Q4	Motor movimiento avance horizontal
Q5	Motor movimiento giro horario
Q6	Motor movimiento giro antihorario
Q7	Habilitar compresor
Q8	Succión ventosa
Q9	Habilitar señales de entrada del proceso

Tabla 12. Descripción de las salidas del robot manipulador.

4.4.2 Funcionamiento

El robot manipulador tiene dos tareas a realizar:

1. Cuando la cámara detecta pieza, el brazo robótico debe ir hasta la posición de dicha pieza, recogerla e ir hasta la cinta 1 del sistema de mecanizado y acto seguido, depositar la pieza exactamente en el centro del sensor I7.

2. Cuando una pieza ha sido mecanizada y llevada a la salida del sistema de mecanizado (sensor I9 en estado *FALSE*), el robot debe ir hasta dicha posición, recoger la pieza y llevarla al lugar especificado, además de clasificarla según su color. El lugar donde se deben dejar las piezas según su tipo se muestra en la siguiente figura:

Figura 54. Lugar de almacenamiento de las piezas según su color.

IMPORTANTE tener en cuenta los siguientes puntos:

- La tarea de recogida y clasificación de piezas tiene prioridad ante la tarea de recogida y puesta de piezas en el sistema de mecanizado (de esta manera se evitan acumulaciones de piezas).

- Cada vez que el robot realice alguna de las dos tareas descritas anteriormente, debe pasar por cero (estado inicial), esto es:

  - Eje vertical en su posición inicial (final de carrera vertical I1 activo).
  - Eje horizontal en su posición inicial (final de carrera horizontal I2 activo).
  - Eje de giro en su posición inicial (final de carrera de giro I3 activo).
  - En lo que pulsos de *encoder* se refiere, los tres *encoders* deben estar a cero.

  Nota: El motivo por el cual debe pasar por cero es debido a que cada vez que se toman lecturas de los *encoders,* se pierde información. Al moverse los ejes, se toman lecturas de los pulsos, las cuales se almacenan en variables (**véase apartado 4.4.3**), y aunque las entradas asignadas a los *encoders* sean entradas rápidas, se suelen perder lecturas durante el almacenamiento de los pulsos en las variables. Por lo tanto una calibración pasando por cero, solventa el problema, evitando que la pérdida de información sea sustancial.

- Al iniciar el sistema, si el robot no está en su posición inicial se ejecutará el programa *Punto_Inicial*, estructurado en SFC, el cual lleva el robot a dicho estado.

- Para facilitar la programación, se ha dividido las tareas en tres SFC.

  - SFC de *Alimentar_piezas*, asignado a la tarea de recoger y poner la pieza en el sistema de mecanizado.

  - SFC de *Recogida_piezas*, asignado a la tarea de recogida y clasificación de las piezas.

  - SFC de *Coger_pieza*, asignado a la acción de coger la pieza con la ventosa mediante aspiración (para evitar tener que repetir dicha acción en varios programas).

4.4.3 Lectura de los *encoders*

Para tomar lecturas de los pulsos, se han implementado los programas siguientes (uno para cada eje):

```
IF I3=TRUE THEN
    CuentaGiro:=0;
ELSE
    IF Q6=TRUE THEN
        CuentaGiro:=CuentaGiro+1;
    END_IF;

    IF Q5=TRUE THEN
        CuentaGiro:=CuentaGiro-1;
    END_IF;
END_IF;
```

Figura 57. Lectura del *encoder* de giro. ST de *ContadorGiro.*

```
IF I2=TRUE THEN
    CuentaHorizontal:=0;
ELSE
    IF Q4=TRUE THEN
        CuentaHorizontal:=CuentaHorizontal+1;
    END_IF;

    IF Q3=TRUE THEN
        CuentaHorizontal:=CuentaHorizontal-1;
    END_IF;
END_IF;
```

Figura 55. Lectura del *encoder* horizontal. ST de *ContadorHorizontal.*

```
IF I1=TRUE THEN
    CuentaVertical:=0;
ELSE
    IF Q2=TRUE THEN
        CuentaVertical:=CuentaVertical+1;
    END_IF;

    IF Q1=TRUE THEN
        CuentaVertical:=CuentaVertical-1;
    END_IF;
END_IF;
```

Figura 56. Lectura del *encoder* vertical. ST de *ContadorVertical.*

Cada uno de los programas está asignado a una tarea externa, ejecutadas mediante evento. El evento de cada una de las tres tareas son las entradas rápidas de los *encoders*, es decir:

- ST de *ContadorGiro* asignado a la tarea ContadorGiro. Dicha tarea se ejecuta cada vez que se activa *Input_B5*, entrada rápida asociada al *encoder* de giro.

- ST de *ContadorHorizontal* asignado a la tarea ContadorHorizontal. Dicha tarea se ejecuta cada vez que se activa *Input_B3*, entrada rápida asociada al *encoder* horizontal.

- ST de *ContadorVertical* asignado a la tarea ContadorVertical. Dicha tarea se ejecuta cada vez que se activa *Input_B1*, entrada rápida asociada al *encoder* vertical.

Se muestra un ejemplo en la siguiente imagen:

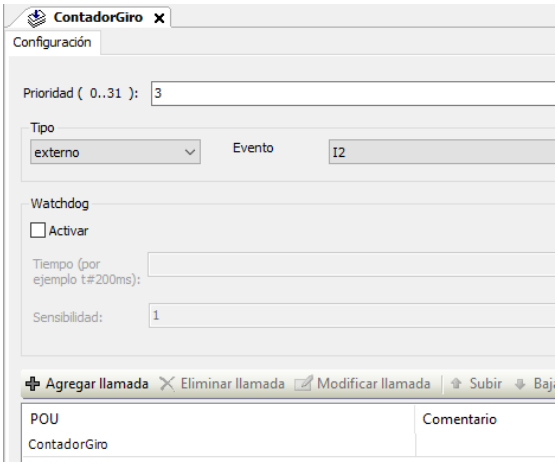


Figura 58. Configuración de la tarea ContadorGiro. Tipo: Externo. Evento: I2. POU agregado: ContadorGiro.

La lógica de los códigos implementados para las lecturas es bastante sencilla. Si se activa el final de carrera, el contador se pone a 0 (inicio). Dependiendo del actuador activado, el programa sumará o restará pulsos al contador.

Nota: El contador tiene un máximo de 1000 pulsos para el eje horizontal y vertical y de 750 pulsos para el eje de giro. Los contadores se ponen a 0 cuando los finales de carrera se activan.

4.4.4 Inicio del sistema

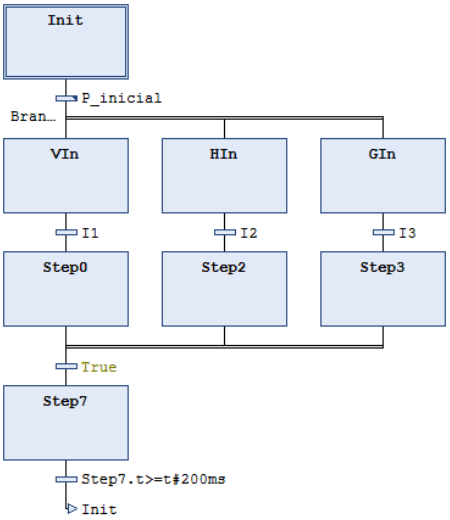


Figura 59. SFC de *Punto_Inicial*

Al franquearse la transición *P_inicial*, los tres ejes del robot vuelven a su posición inicial hasta que los finales de carrera se activan. Los estados *Step0, Step2* y *Step 3* son estados de espera, por si algún eje ha llegado a su punto inicial antes que los demás.

El estado *Step7* se utiliza para advertir al SFC *Alimentar_piezas* de que el robot ya ha pasado por cero, así dicho SFC termina su ciclo.

Nota: El *timer* de 0.2 segundos se utiliza como transición para asegurarse de que, al pasar al estado *Step7*, el programa tiene tiempo suficiente para avisar a *Alimentar_piezas* ya que sin el *timer*, puede que el cambio tan rápido de estados imposibilite el paso de información de un SFC a otro.

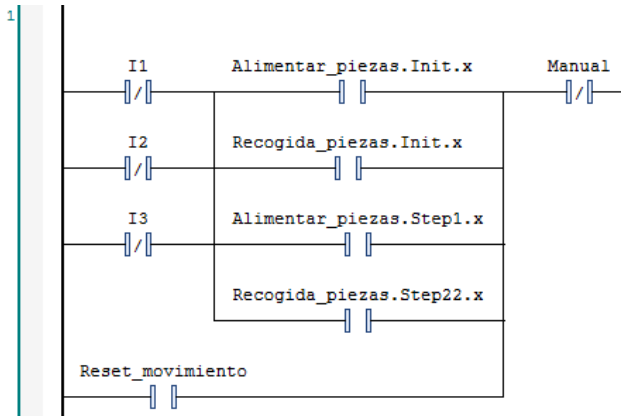


Figura 60. Transición *P_inicial*. Diagrama de contactos. *Punto_Inicial.*

Para ejecutar el paso por cero, algún final de carrera (o todos) deben estar en *FALSE*. Además, las tareas de puesta y clasificación de piezas deben de estar en su estado inicial o final. Todo esto ocurrirá el control es automático, en caso de que sea manual, el paso por cero del robot no se producirá. La variable *Reset_movimiento* sirve para que el robot pase por cero mientras está siendo usado manualmente (**véase apartado 4.4.9**).

4.4.5 Puesta de la pieza en el sistema de mecanizado

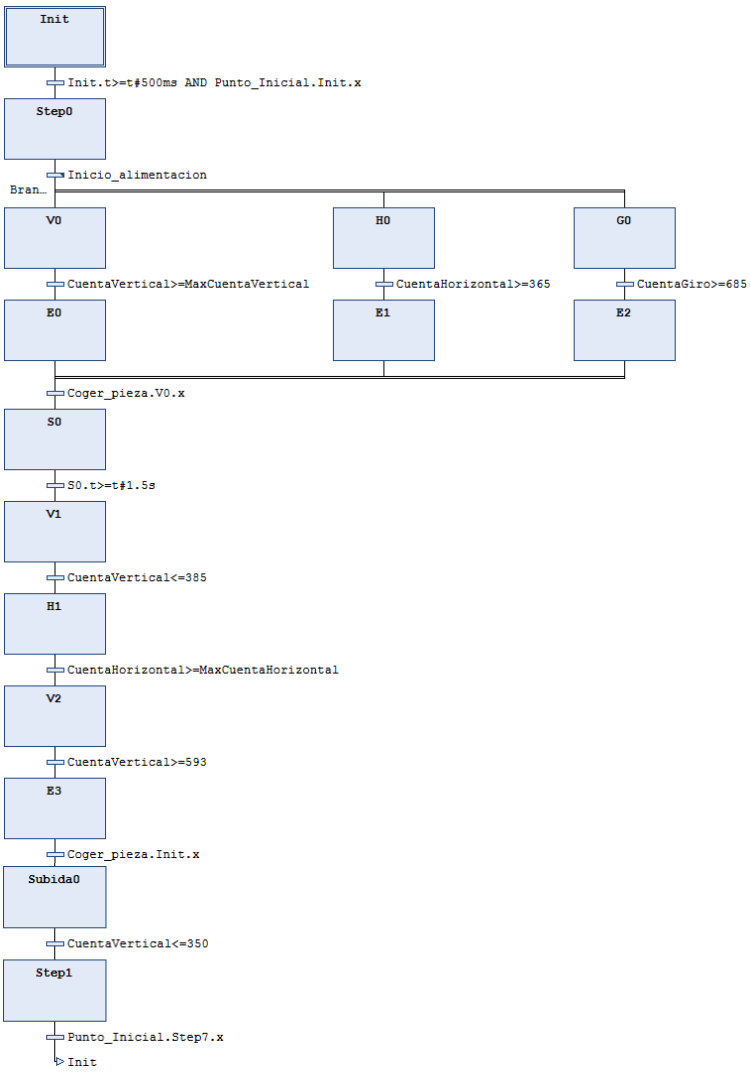


Figura 61. SFC de *Alimentar_piezas.*

Debe predominar el paso por cero del robot antes que la recogida de la pieza, en caso de que ambas situaciones ocurran al inicio del sistema (esto es lo que ocurre en el estado inicial *Init)*. Para empezar la alimentación de la pieza en el sistema de mecanizado (*Inicio_alimentacion*), o bien se inicializa manualmente (**véase apartado 4.4.10)** o se detecta pieza. En el caso de que se detecte pieza, la transición debe tener en cuenta si hay pieza mecanizada o no en el sensor I9 del sistema de mecanizado. Al ejecutarse la tarea, el robot se desplaza verticalmente hacia abajo, horizontalmente hacia delante y con giro antihorario al mismo tiempo hasta llegar al lugar de recogida de piezas.

Cuando los estados *E0, E1* y *E2* se activan, se produce la habilitación del compresor (**véase apartado 4.4.8**) y cuando la ventosa succiona la pieza, se franquea la transición *Coger_pieza.V0.x*. El estado *S0* sirve para dar 1.5 segundos a que la ventosa termine de succionar por completo la pieza. El resto de programa realiza los movimientos del robot necesarios para llevar la pieza a la cinta 1 del mecanizado, en el centro del sensor I7.

El estado *E3* es un estado de reposo, donde al mismo tiempo, envía la orden al SFC *Coger_pieza* (transición *Deshabilitar_todo*) para que deshabilite la succión y compresión. Cuando *Coger_pieza* finaliza su tarea, se pasa del estado *E3* al estado *Subida0*, donde el eje vertical del robot sube ligeramente antes de volver a su posición inicial (para evitar colisiones).

El estado *Step1* da permiso al SFC *Punto_Inicial* para iniciar el paso por cero, cuando éste termina, también lo hace *Alimentar_piezas*, volviendo a su estado inicial (*Init*).

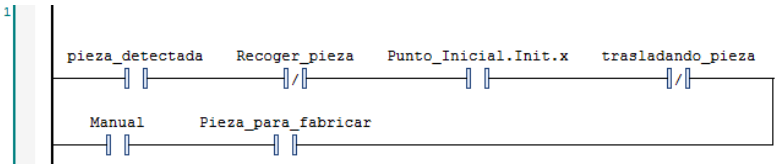


Figura 62. Transición *Inicio_alimentacion*, diagrama de contactos. *Alimentar_piezas.*

4.4.7 Recogida y clasificación de piezas

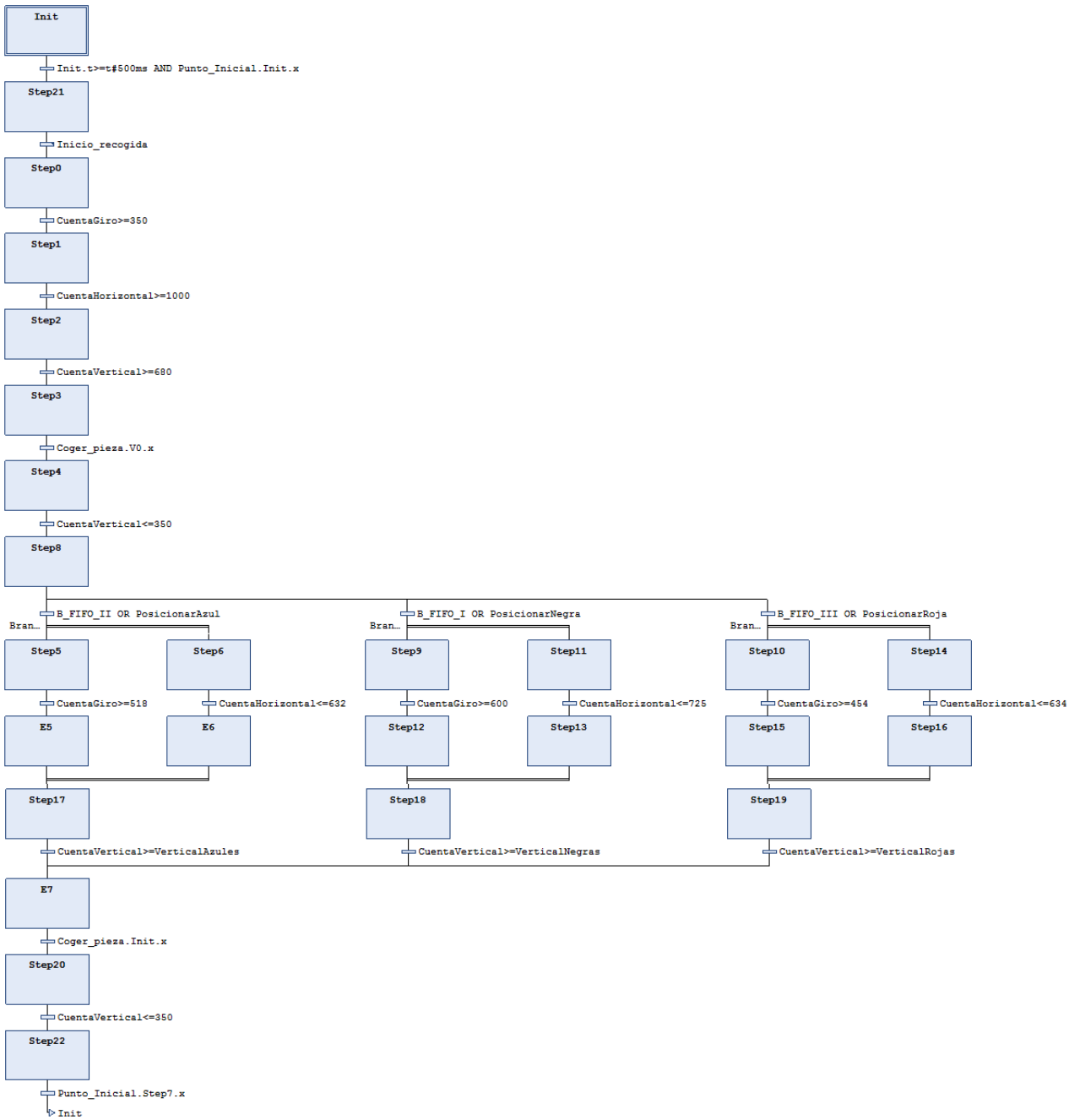


Figura 63. SFC de *Recogida_piezas.*

La metodología empleada para este programa es la misma que para *Alimentar_piezas*, pero con ligeros cambios. En este caso, la transición que permite ejecutar la tarea (*Inicio_recogida*) no tiene en cuenta si hay pieza o no para incorporar en el mecanizado, ya que esta tarea tiene preferencia, descrito anteriormente en el **apartado 4.4.2**.

Para determinar cuándo el robot debe ir a por la pieza (sensor I9 del mecanizado a *FALSE*), se utiliza el siguiente programa:

```
26  //Comunicaciones PLC
27
28  IF Escribeme = TRUE THEN
29      Recoger_pieza := TRUE;
30  ELSIF Escribeme = FALSE THEN
31      Recoger_pieza := FALSE;
32  END_IF
```

Figura 64. ST de la variable asociada a la comunicación maestro/esclavo junto con la condición de iniciar la recogida y clasificación de piezas.

La variable *Escribeme* está asociada a la dirección de memoria MX0.1 del robot manipulador, de manera que, gracias a la comunicación maestro/esclavo establecida, dicha variable se pone a *TRUE* cuando el sensor I9 se encuentra en *FALSE* (pieza detectada), y viceversa. De manera que cuando I9 detecta una pieza mecanizada, la variable *Recoger_pieza* se activa y se procede a la ejecución de la tarea.

Recoger_pieza es una variable que, además, está asociada al botón manual de la ejecución de recogida y clasificación de las piezas, por lo tanto no se ha querido asociar dicha variable a la dirección MX0.1, para evitar problemas. Por esta razón, se utiliza la variable *Escribeme* para la comunicación del valor de I9 de un autómata a otro.

Así mismo, la clasificación de piezas depende del color de éstas. Dependiendo del valor obtenido por la variable de salida de la función FIFO, se ejecutará un programa u otro. Además, existe la opción de clasificar las piezas manualmente (**véase apartado 4.4.10**).

Para saber cuánto tiene que bajar el brazo verticalmente para depositar la pieza (en función del color), se hace uso de este programa:

```
//******Depositar piezas*******//

//Negras

IF B_DatoDescarga = 1 THEN  //La pieza a recoger es negra

    Negras := Negras + 1;
    VerticalNegras := 980 - 205*(Negras - 1);

END_IF

//Azules

IF B_DatoDescarga = 2 THEN  //La pieza a recoger es azul

    Azules := Azules + 1;
    VerticalAzules := 980 - 205*(Azules - 1);

END_IF

//Rojas

IF B_DatoDescarga = 3 THEN  //La pieza a recoger es roja

    Rojas := Rojas + 1;
    VerticalRojas := 980 - 205*(Rojas - 1);

END_IF

IF Negras >= 4 THEN
    Negras := 0;
END_IF

IF Azules >= 4 THEN
    Azules := 0;
END_IF

IF Rojas >= 4 THEN
    Rojas := 0;
END_IF
```

Figura 65. ST de *Tipo_piezas* para el almacenaje de las piezas en una columna.

Mediante una serie de pruebas con el robot en modo manual, se obtuvo el número de pulsos necesarios en el eje vertical para dejar la primera pieza (980 pulsos). Para depositar las demás piezas (del mismo color) una encima de otra, se tuvo que comprobar el número de pulsos equivalentes a la altura de una pieza. Esto se consiguió depositando la segunda pieza encima de la primera y comprobando el resultado de la variable *CuentaVertical*. Ésta marcaba 775 pulsos, por lo tanto, realizando la resta (980 - 775) se obtuvo el número de pulsos equivalentes a la altura de una pieza, dando como resultado 205 pulsos.

El programa funciona de tal modo que, cuando se detecta una pieza, se le suma a la variable (*Negras, Azules o Rojas*) un 1, significando que es la primera pieza mecanizada de su tipo. Por lo tanto, con la ecuación descrita, se obtiene el número de pulsos a bajar verticalmente para depositar la primera pieza, y así sucesivamente.

Nota: Este programada en texto estructurado está asociado a una tarea del tipo evento, para que solo se ejecute una vez el robot haya cogido con la ventosa la pieza mecanizada. Además, hay que tener en cuenta la altura del robot, por ello el programa permite hasta un máximo de 4 piezas almacenadas una encima de otra. El operario debe estar al corriente, para que cuando llegue la quinta pieza, las otras cuatro se hayan retirado (procesos externos al proyecto o mediante mano de obra).

4.4.8 SFC de Coger_pieza (compresor y ventosa)

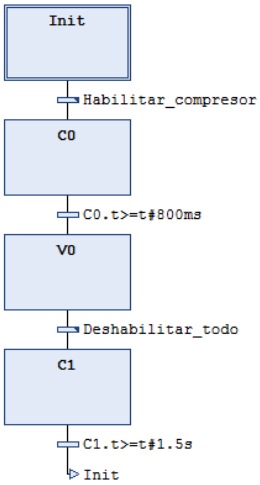


Figura 66 . SFC de *Coger_pieza.*

Para que se habilite el compresor, se debe franquear la transición *Habilitar_compresor*, la cual se activa mediante las etapas vistas anteriormente de *Alimentar_piezas* y *Recogida_piezas.* El estado *C0* pone en funcionamiento el compresor, dándole la siguiente transición 0.8 segundos para efectuar correctamente la compresión. El estado *V0* pone en marcha la succión de la ventosa, y mientras dicho estado esté activo, la ventosa seguirá succionando.

La transición *Deshabilitar_todo*, como bien indica su nombre, desactiva la succión y la compresión, en dicho orden. La última transición proporciona 1.5 segundos para asegurarse que la succión y compresión se han desactivado correctamente (para que el robot no ejecute otras tareas mientras dichas acciones están terminándose).

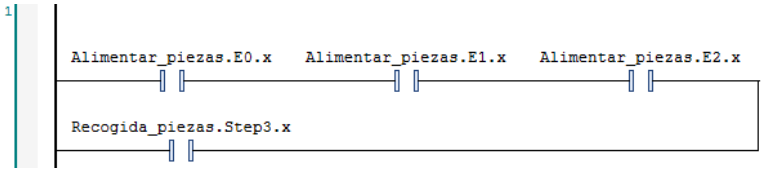


Figura 67. Transición *Habilitar_compresor*, diagrama de contactos. SFC *Coger_pieza.*

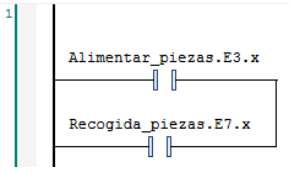


Figura 68. Transición *Deshabilitar_todo*, diagrama de contactos. SFC *Coger_pieza.*

4.4.9 Función *First Input First Output* (FIFO)

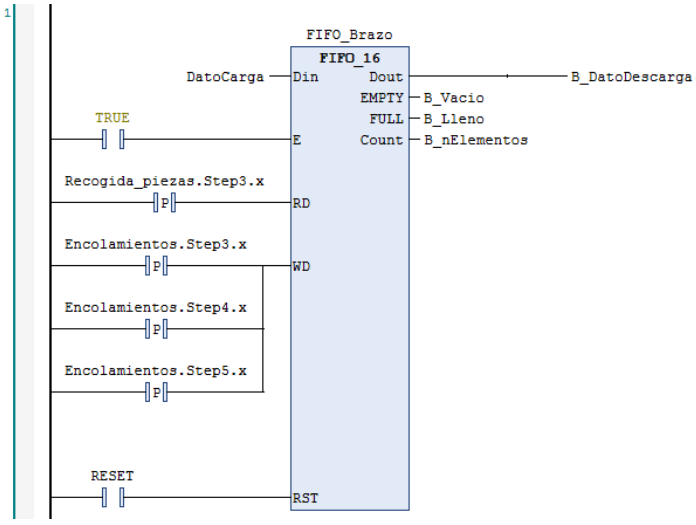


Figura 69. Diagrama de contactos de *ProgramaFIFO.*

Los encolamientos se inician cuando *DatoCarga* recibe un valor correspondiente a uno de los colores durante un cierto tiempo (explicación en el apartado **4.6.3** y la programación correspondiente, en **anexo 1**).

Se desencola de la lista mientras se ésta succionando la pieza con la ventosa.

Para determinar el tipo de pieza que se ha de clasificar una vez recogida, se utiliza el programa mostrado en la siguiente figura:

```
IF B_DatoDescarga = 1 THEN
    B_FIFO_I := TRUE;
    B_FIFO_II := FALSE;
    B_FIFO_III := FALSE;

ELSIF B_DatoDescarga = 2 THEN
    B_FIFO_I := FALSE;
    B_FIFO_II := TRUE;
    B_FIFO_III := FALSE;

ELSIF B_DatoDescarga = 3 THEN
    B_FIFO_I := FALSE;
    B_FIFO_II := FALSE;
    B_FIFO_III := TRUE;

ELSIF B_DatoDescarga = 0 OR B_DatoDescarga > 3 THEN
    B_FIFO_I := FALSE;
    B_FIFO_II := FALSE;
    B_FIFO_III := FALSE;

END_IF
```

En función del valor sacado y leído de la lista FIFO, se designan los valores (*TRUE o FALSE*) a las variables encargadas de administrar que tipo de pieza ha cogido el robot manipulador.

Se recuerda que se atribuye el valor 1 al tipo de pieza 1 (pieza negra), el valor 2 al tipo de pieza 2 (pieza azul) y el valor 3 al tipo de pieza 3 (pieza roja).

Figura 70. ST de *EstadoFIFO.*

4.4.10 Sistema SCADA con *SoMachine*

El SCADA del robot está compuesto únicamente por una interfaz de usuario, permitiendo al operador saber el estado de los ejes (número de pulsos) como también controlar el robot manualmente, tanto los ejes como la ejecución de las tareas.

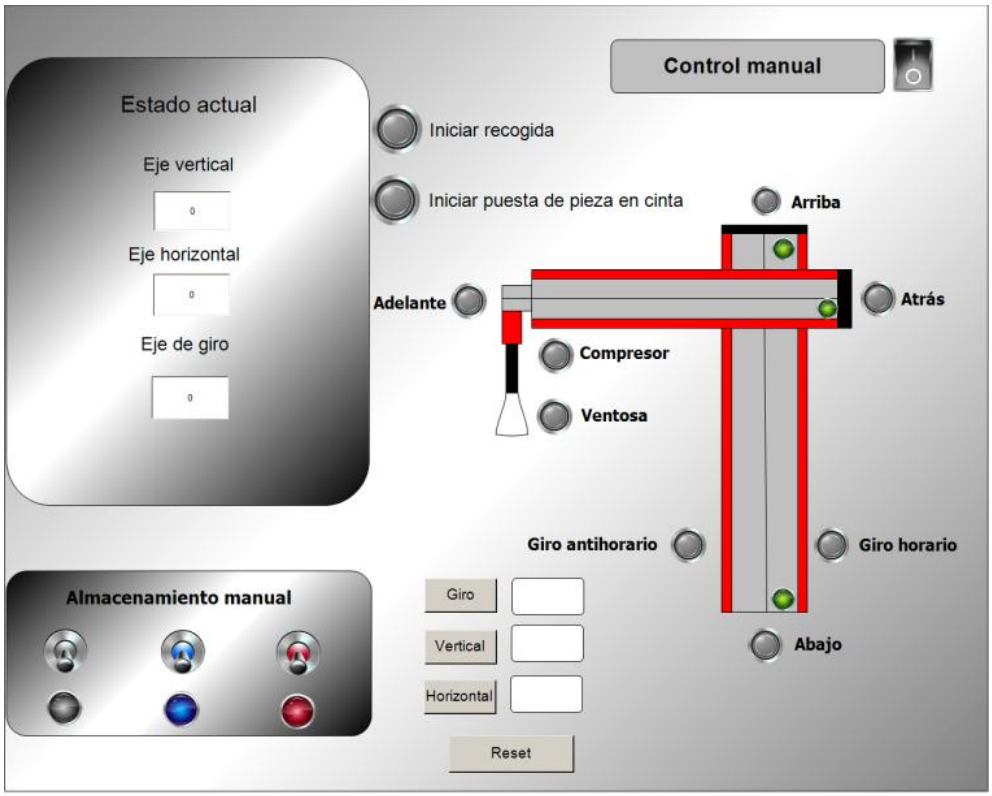


Figura 71 .Interfaz de usuario del robot manipulador. Control y supervisión.

Mediante la pantalla "Estado actual" se puede supervisar la configuración del robot en todo momento, mostrándose por cada eje el número de pulsos que se han registrado.

Por otro lado, si se presiona el botón que hay al lado del panel "Control manual", todos los controles manuales estarán operativos. Cabe mencionar que el panel se pondrá de color verde cuando se active el modo manual.

Existen dos botones para ejecutar las tareas de recogida y clasificación y puesta en el sistema de mecanizado. Por otra parte, si se quiere controlar los ejes manualmente, existen dos opciones:

1. Control por pulsadores: Mediante pulsadores, se pueden controlar los tres ejes del robot, tanto en un sentido como en otro, además del compresor y la ventosa.

2. Control numérico: Se introduce el valor deseado en las casillas (número de pulsos) y posteriormente, se presiona el botón deseado, al lado de las casillas. De este modo, el movimiento del eje seleccionado se efectuará hasta el número de pulsos indicado.
   Nota: Este tipo de botones actúan como pulsadores (ya que a los botones disponibles en *SoMachine*, no se les puede insertar nombre como los designados para el control numérico), por ello, deben ser pulsados dos veces, una para ejecutar la orden, y otra para poner el botón en su posición inicial y así, ser usado de nuevo.
   Cabe mencionar que este modo solo sirve para avanzar los ejes, no para retrocederlos. Para solventar el problema, existe el botón "*Reset*", el cual pone los ejes a cero.

Para el control numérico se han utilizado los siguientes programas SFC:

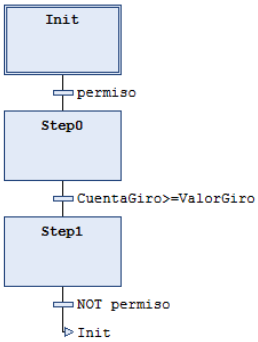


Figura 72. SFC de *Prueba_giro.*

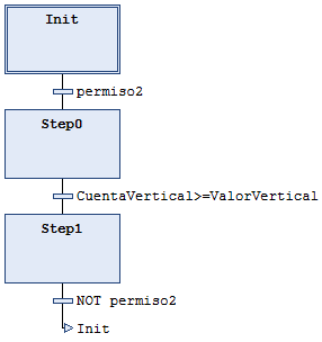


Figura 73. SFC de *Prueba Vertical.*

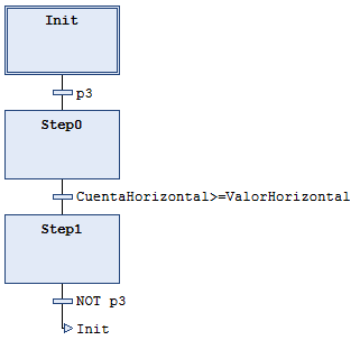


Figura 74. SFC de *Prueba_Horizontal.*

Al pulsar uno de los tres botones (Giro, Vertical u Horizontal) se activan las variables *permiso, permiso 2* y *p3*, respectivamente. Al entrar en el estado *Step0* de cada SFC, se activa el correspondiente motor asociado al correspondiente eje, siendo las variables *ValorGiro, ValorVertical* y *ValorHorizontal* las asociadas a cada casilla para recoger el valor introducido por el operario. Cuando el eje llega a la posición indicada, se pasa al estado *Step1*, el cual espera hasta que se vuelva a pulsar el respectivo botón para volver al inicio del programa.

El almacenamiento manual permite al operario elegir en qué posición clasificar la pieza (los lugares son los preestablecidos en el **apartado 4.4.2**). **IMPORTANTE:** Solo se debe utilizar el almacenamiento manual en caso de que no se vaya a utilizar la cámara, en caso contrario, se podrían producir contradicciones y se pararía el robot.

4.5 Servidor OPC

Para utilizar el servidor, se requiere una configuración y creación de un canal dentro del mismo antes de la puesta en marcha, como también la creación y configuración de los módulos o dispositivos del canal (cada módulo está asociado mediante IP a cada autómata). También se requiere de variables, llamadas "*tags*" dentro del dispositivo, para enlazarlas con las variables de los autómatas por medio de su dirección de memoria.

4.5.1 Configuración del servidor

Para configurar el servidor OPC se debe seguir una serie de pasos, mostrados por el asistente de configuración:

- ***Channel name*:** Se debe introducir el nombre del canal, a gusto del usuario. Para el proyecto, se utilizó el nombre "TFG_DG".

- ***Device:*** Dispositivo el cual se desea comunicar con el servidor (*Omron, Allen-Bradley,Siemens, etc.)*, siendo en este caso el *TSX Premium*, ya que es muy similar al Modicon, sobre todo en sus direcciones de memoria.

4.5.2 Configuración del dispositivo

Como se mencionó anteriormente, el módulo o dispositivo es básicamente el entorno dentro del canal del servidor el cual está enlazado al dispositivo físico mediante algún tipo de comunicación, en este caso, mediante *Modbus TCP/IP*.

Para enlazar el módulo con el *Modicon M241*, se debe realizar la configuración, siguiendo estos pasos:

- ***Channel name*:** Se debe introducir el nombre del dispositivo, a gusto del usuario. Para el proyecto, se utilizaron dos dispositivos, a los cuales se les llamaron "Brazo" y "Cintas".

- ***Device model:*** Se selecciona el modelo de *driver* que soporta el tipo de dispositivo seleccionado para el canal, siendo en este caso, el *driver Applicom* (utilizado por el *TSX Premium*).

- ***Device ID:*** Se requiere la dirección IP del dispositivo físico en cuestión, en este caso, al tener dos dispositivos dentro del canal, se precisa la IP de ambos autómatas, una para cada dispositivo a configurar.

4.5.3 Creación de *Tags*

Para crear los *tags* en el dispositivo creado dentro del canal del servidor, simplemente basta con un *click* derecho sobre el dispositivo en cuestión y pulsar donde pone "*Create new tag"*. Al crear un *tag,* se debe configurarlo, de tal modo que se requiere rellenar los siguientes campos:

- ***Tag name:*** Nombre del *tag*, a elección del usuario.

- ***Address:*** Se introduce la dirección de memoria deseada (correspondiente al autómata). Lo que se consigue con esto es enlazar el *tag* con la variable del autómata que está asociada a la misma dirección de memoria configurada en dicho *tag*.

- ***Data type:*** Suele venir por defecto al detectar una dirección de memoria válida (Si es una dirección MX por defecto vendrá como tipo de dato un booleano), de todos modos, es configurable (tipo *WORD,boolean, float*, etc.). Además, permite configurar el uso del *tag* (lectura y/o escritura).

En la siguiente figura se muestra los *tags* creados para ambos dispositivos:

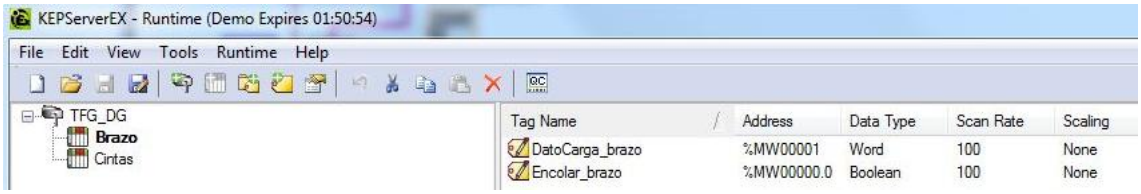


Figura 75. Servidor OPC. Canal TFG_DG. Dispositivo Brazo.

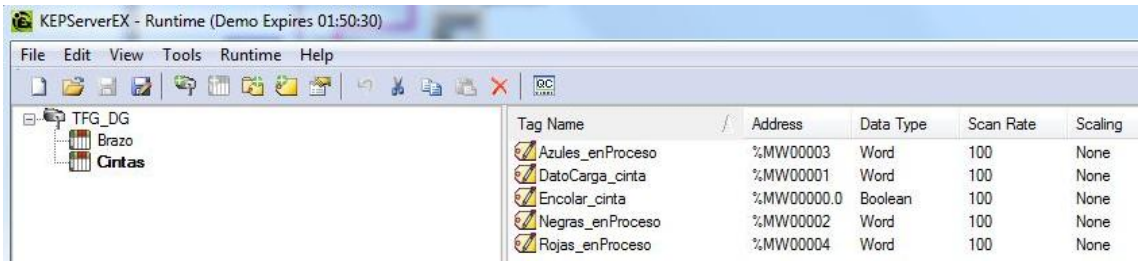


Figura 76. Servidor OPC. Canal TFG_DG. Dispositivo Cintas.

Cabe resaltar algunos aspectos en lo referente a las direcciones de memoria:

- Para direccionar un *tag* al tipo de memoria MW, se debe seguir la estructura "MW0000X", siendo X el número de memoria a nivel de *byte*. **Ejemplo:** MW1 = MW00001.

- Para direccionar un *tag* al tipo de memoria MX, se sigue la estrucutra "MW0000X.Y", siendo X el número de memoria a nivel de *byte* e Y el número de memoria a nivel de *bit.*
  Ejemplo: MX1.4 = MW00001.4.

4.5.4 Verificación de funcionamiento

Para comprobar que las variables de los autómatas enlazadas con sus respectivos *tags* se actualizan en tiempo real en el servidor, se debe ejecutar un cliente rápido (Quick client).

Para ejecutar dicho cliente, se debe hacer *click* sobre el icono mostrado en la siguiente imagen:

Figura 77. Barra de herramientas del servidor OPC. Ejecución del cliente rápido.

Una vez abierto el cliente, aparece una imagen como la que aparece a continuación:

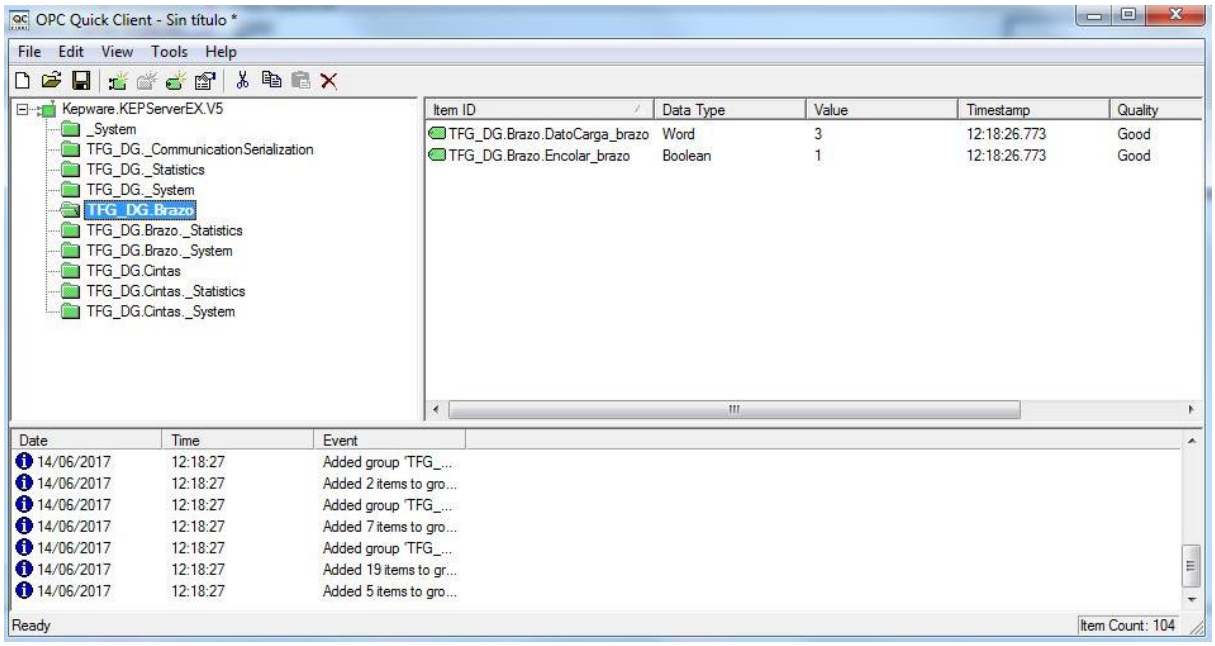


Figura 78. Cliente rápido del servidor OPC.

Como se observa, se puede visualizar los *tags* en cada uno de los dispositivos del canal. Además, también se muestran las características de dichos *tags* (*data type, value…*), pero para determinar si las variables del autómata transmiten la información al servidor OPC, esto es, a sus respectivos *tags*, se debe prestar suma atención a la ventana "*Quality*". Dicha ventana indica si la calidad de la conexión entre el *tag* y su respectiva variable del autómata es buena o mala.

En el caso de que haya algún error, la calidad será mala (*Quality = Bad)* y la transmisión de datos entre dicho *tag* y su respectiva variable del autómata no se producirá. En cambio, para que se produzca la transmisión de datos entre *tag* - variable del autómata, la ventana *Quality* debe marcar "*Good"*, tal y como se muestra en la figura anterior.

4.6 Monitorización de datos con *LabVIEW*

La función principal que tiene *LabVIEW* en este proyecto es la transmisión de los datos referentes a los colores de las piezas, captados por la cámara, a los autómatas. Por otra parte, también se ha introducido un pequeño SCADA para monitorizar en tiempo real lo que la cámara capta, como también el color de pieza detectado y, además, unos indicadores que muestran las piezas que están mecanizándose.

4.6.1 Creación y configuración del proyecto

Para poder usar las variables del servidor OPC en *LabVIEW*, ambos *software* deben estar conectados. Para ello, el nuevo proyecto a crearse desde *LabVIEW* debe configurarse para permitir dicha conexión. En las siguientes figuras se muestra los pasos a seguir:

1- Creación de un nuevo proyecto:

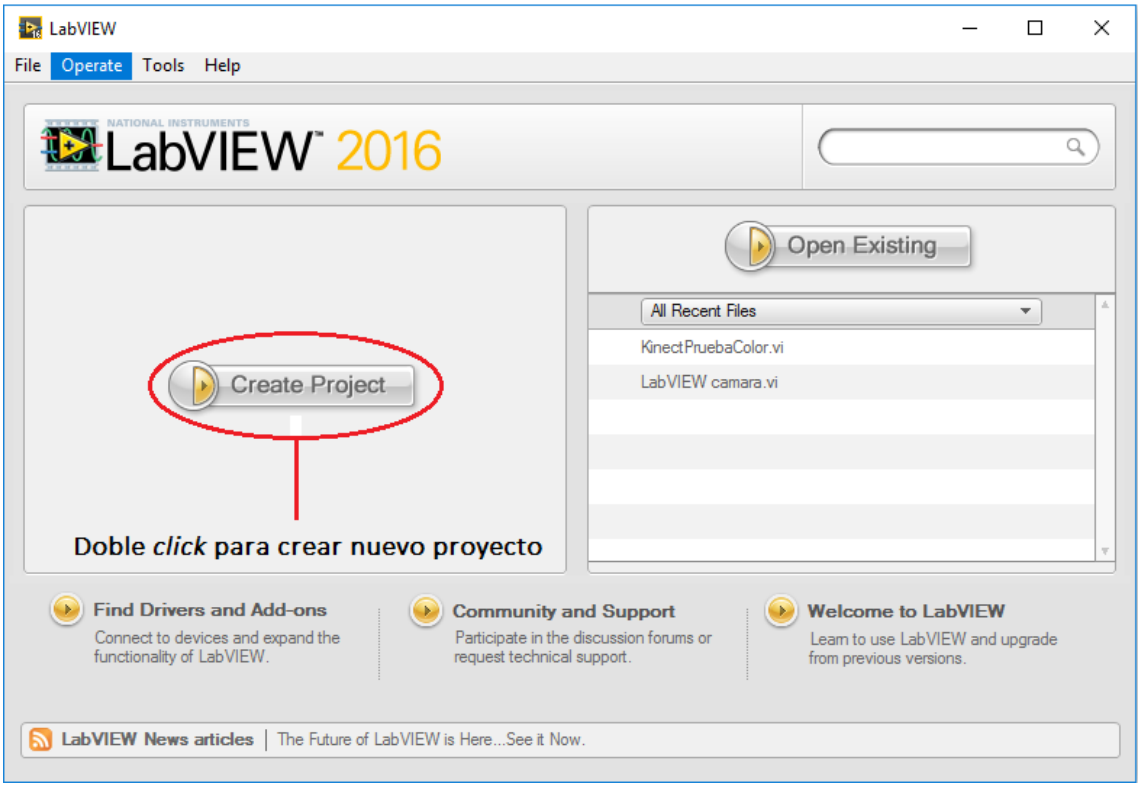


Figura 79. Pantalla inicial de *LabVIEW*. Creación de un nuevo proyecto

2- *Click* derecho en ***My Computer*** y seleccionar la opción ***New*** ≫ ***I/O Server***:

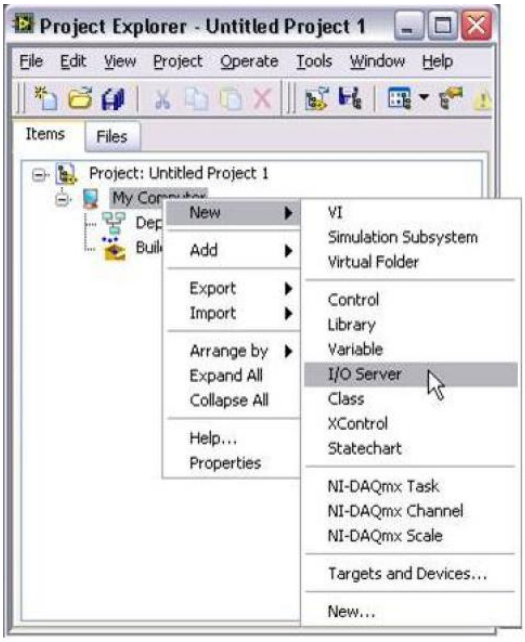


Figura 80. Creación de un nuevo servidor I/O con *LabVIEW*.

3- Una vez creado el servidor I/O, se selecciona **OPC *Client*** y, en la pantalla que aparecerá, se selecciona ***National Instruments.NIOPCServers***. Con ello se crea una librería comunicada con el servidor OPC.

Nota: Se debe cambiar el ***"Update rate"*** a 100 ms, de este modo las variables del OPC usadas en *LabVIEW* se actualizan cada 0.1 segundos.

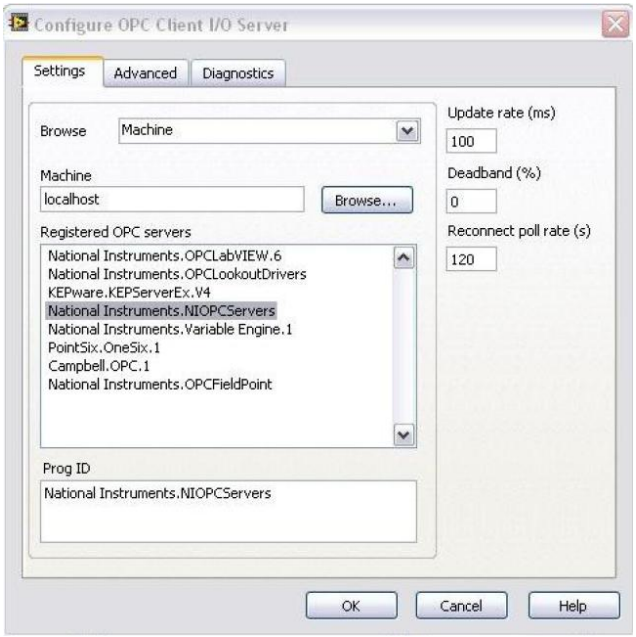


Figura 81. Configuración del cliente OPC en el servidor I/O de *LabVIEW*.

4- Para añadir las variables del servidor OPC al proyecto de *LabVIEW*, primero se debe crear las llamadas "***Bound Variables***":

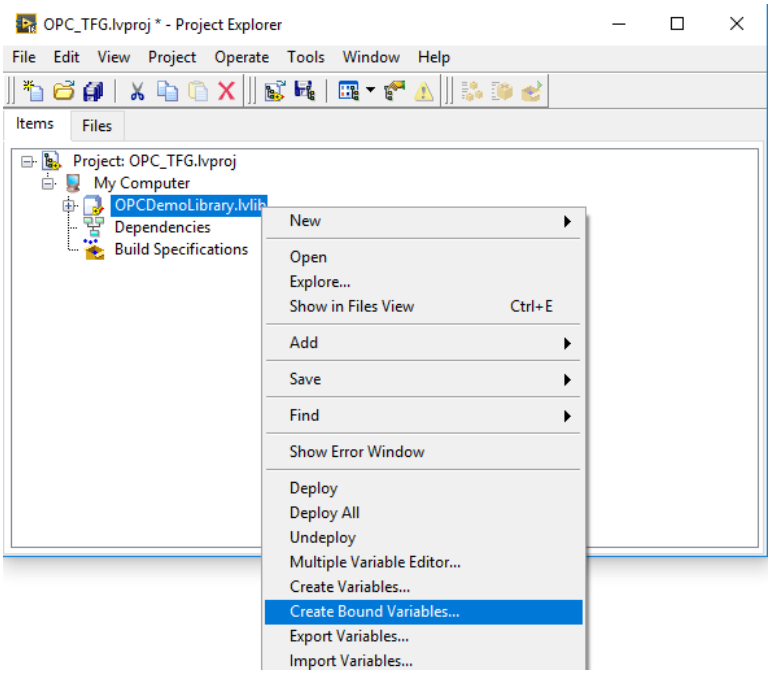


Figura 82. Creación de *Bound Variables*.

5- Una vez creadas, aparece una pantalla donde, accediendo a la librería anteriormente creada, se pueden seleccionar las variables existentes en el servidor OPC, pudiendo exportarlas al proyecto creado:

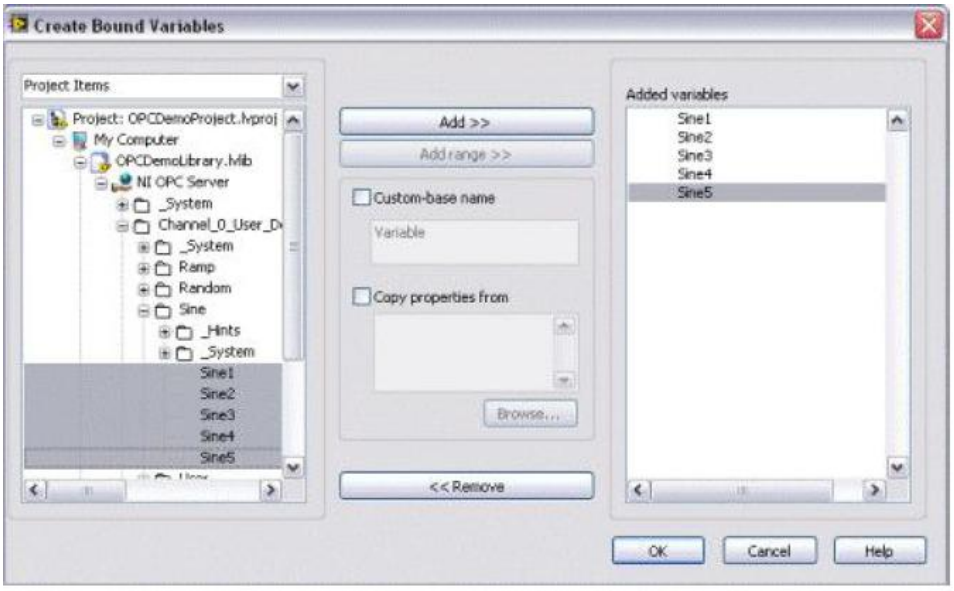


Figura 83. Exportación de variables desde el servidor OPC a *LabVIEW*.

Al exportar las variables deseadas, se guardan dentro de la librería especificada. En el caso del proyecto, las variables exportadas son las que siguen:

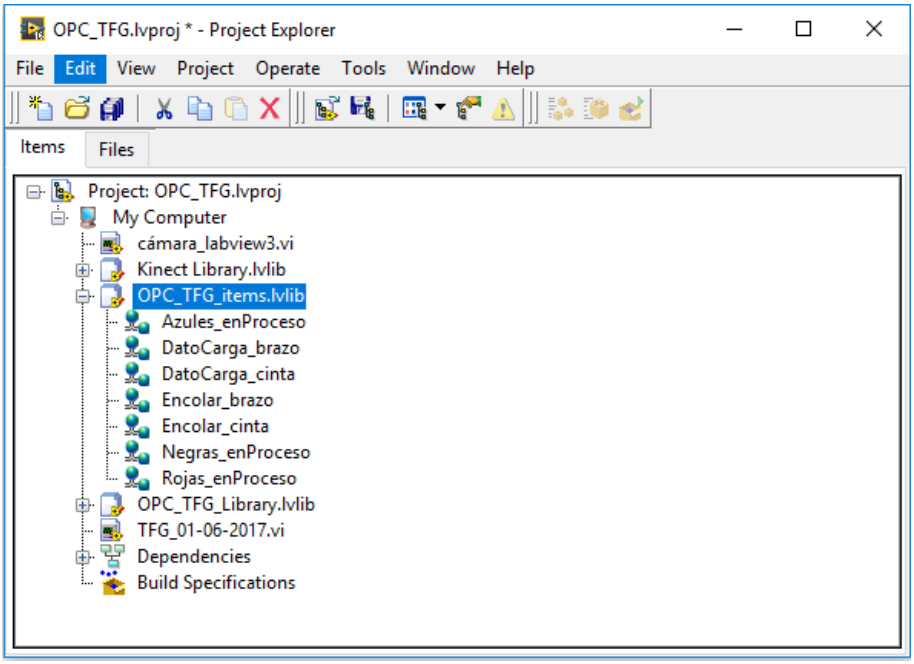


Figura 84. Variables del OPC exportadas al proyecto.

4.6.2 Sistema SCADA con *LabVIEW*

Con las variables exportadas del servidor OPC, es posible crear un enlace entre dichas variables y objetos de *LabVIEW*, de tal manera que se puede visualizar a tiempo real lo que sucede en el sistema.

El SCADA contiene dos interfaces, una para la configuración inicial de la cámara, y otra donde se visualiza el color de la pieza detecta y las piezas que están siendo mecanizadas.

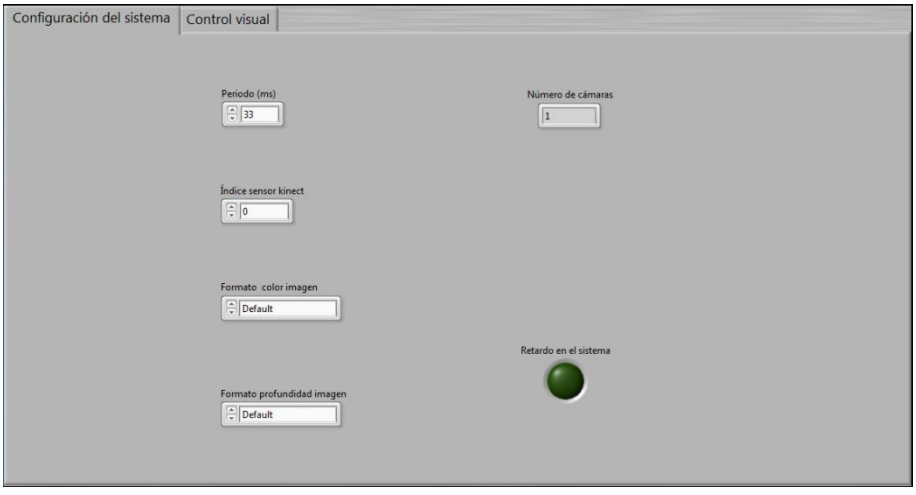


Figura 85. Configuración inicial de la cámara.

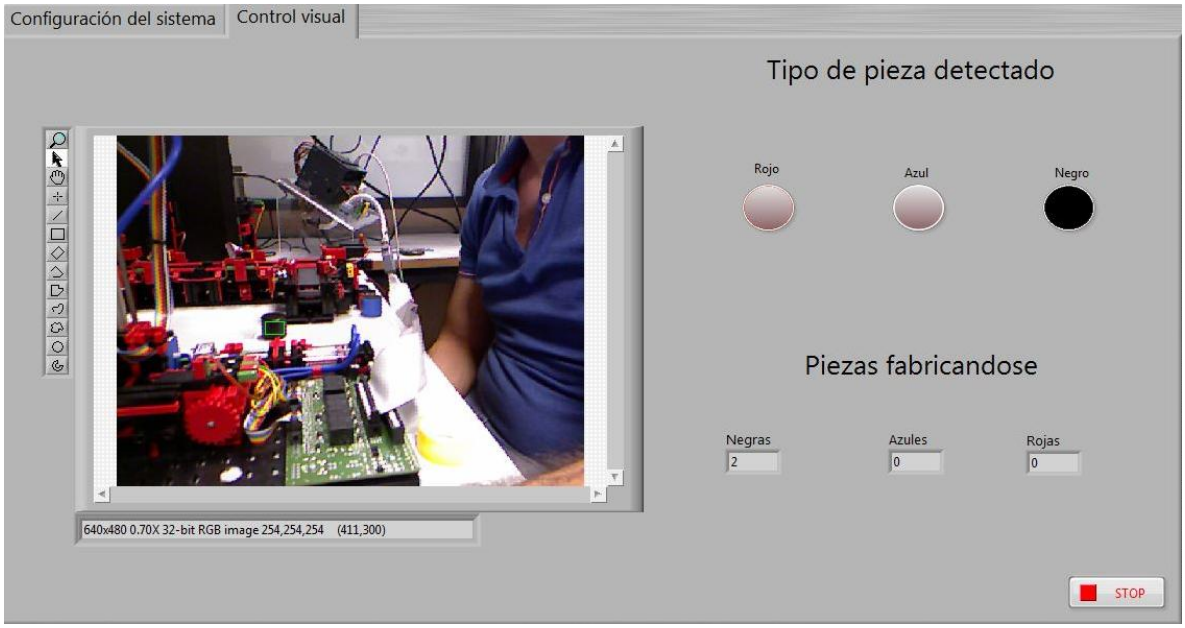


Figura 86. Control y supervisión de las piezas.

Como se observa, se transmite en tiempo real lo que la cámara capta. Por otra parte, existe un control de los tres colores de piezas, encendiéndose el respectivo LED al ser detectadas (mediante una región de interés preajustada). Además, el SCADA indica, en este caso, que hay en el sistema de mecanizado dos piezas negras (tipo o modelo 1) en plena fabricación. Cabe destacar que el sistema está dotado de un botón de parada, llamado "*Stop*", para detener el sistema y la grabación de la cámara.

En las siguientes figuras se observa distintos casos para la detección de piezas:

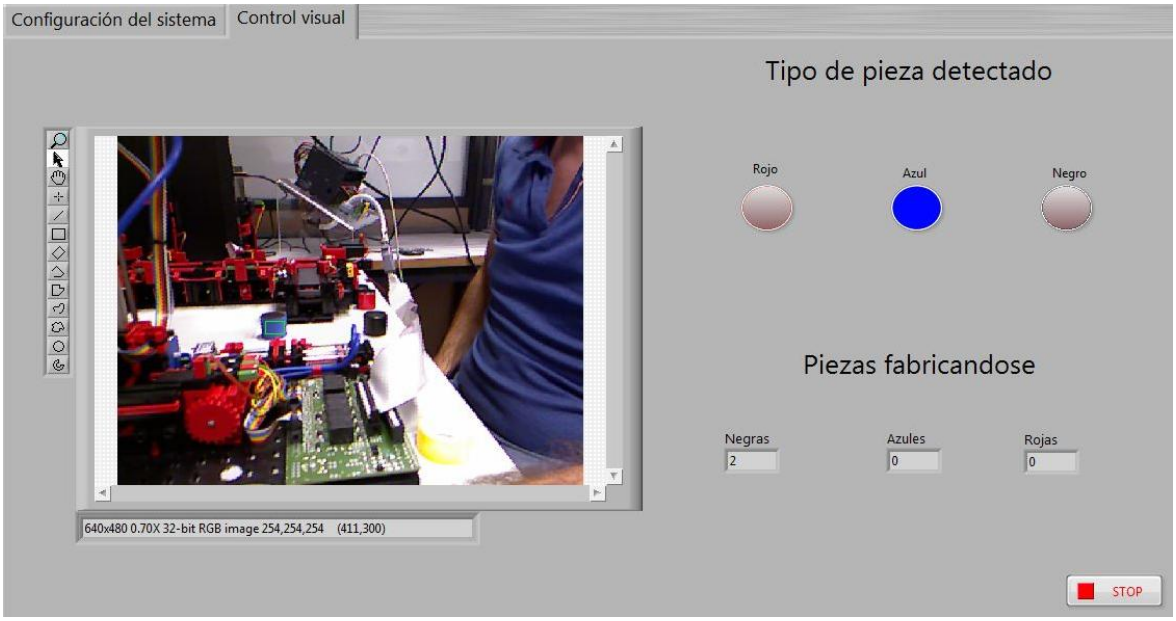


Figura 87. Detección de pieza azul (tipo 2).

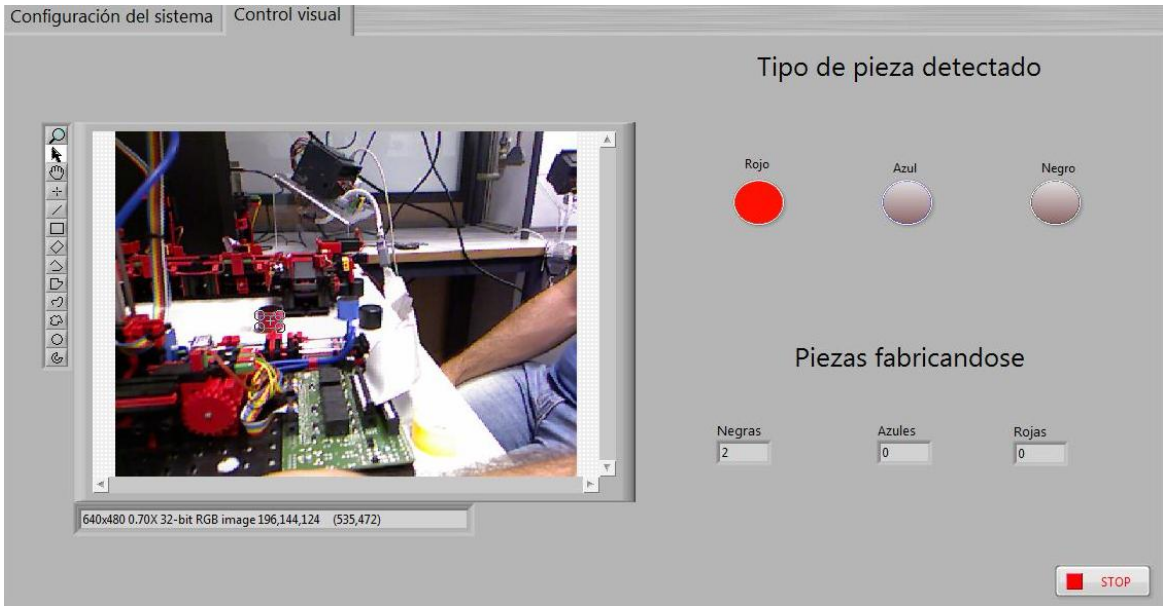


Figura 88. Detección de pieza roja (tipo 3).

Para la programación del SCADA usando las variables del OPC, se han utilizado librerías del sensor *Kinect*, como también librerías de vísion.

En la siguiente figura se muestra el programa de inicialización de la cámara:

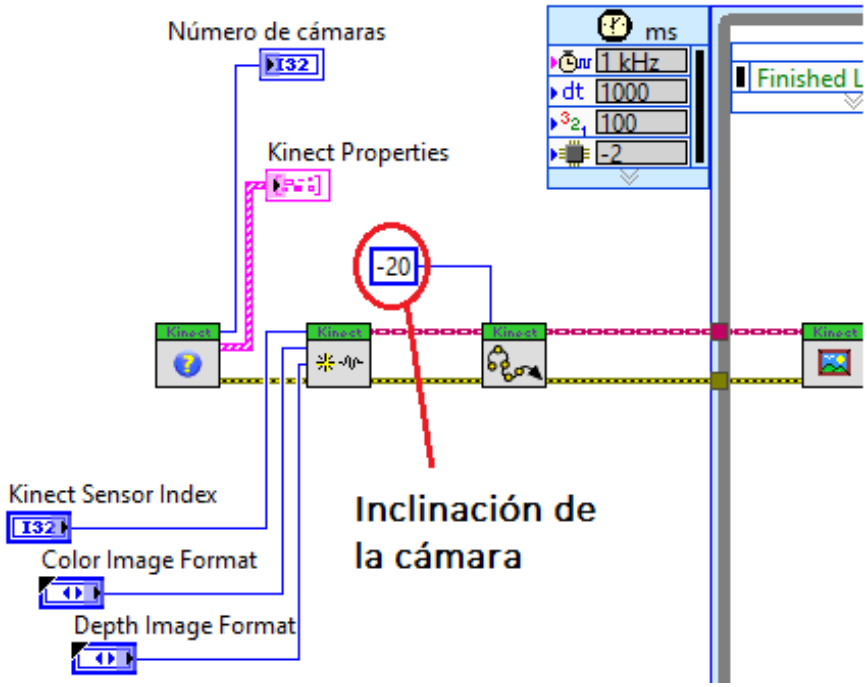


Figura 89. Programa en lenguaje G. Inicialización de la cámara.

Para detectar el color de la pieza, se utiliza un bloque de visión capaz de detectar el espectro de colores de una imagen (siendo en este caso un espectro de 16 *bits*).

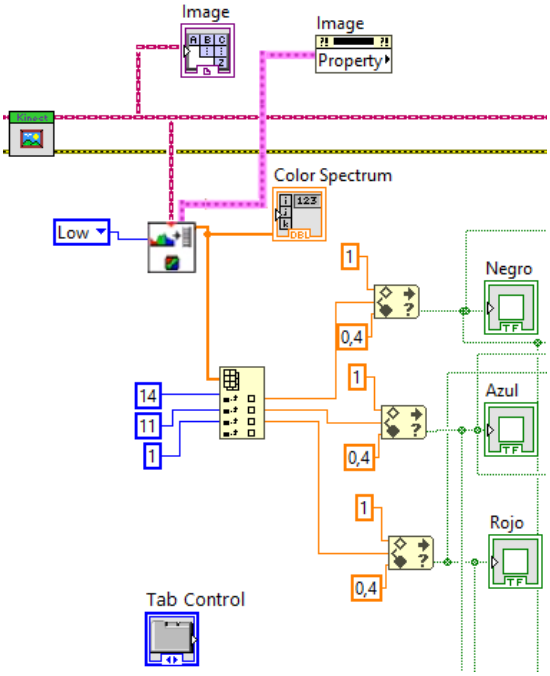


Figura 90. Programación en lenguaje G de la detección de colores.

Lo que se pretende con el programa mostrado en la figura 88 es, por una parte, enviar a la imagen que se muestra en el SCADA tanto la configuración de la cámara como las propiedades de color del bloque de visión RGB. Por otro lado, el resto del programa permite, mediante el espectro de colores, determinar cuándo se detecta negro, azul o rojo.

El espectro de colores mencionado es el mostrado a continuación:

Figura 91. Espectro de colores de 16 *bits*.

Cada uno de los *bits* (la primera casilla del espectro representa el *bit* 0 y la última, el *bit* 15) representa una tonalidad de color, de manera que para colores básicos como el blanco, negro o rojo, solo cambia el valor de un *bit*. En el caso de colores compuestos, como el naranja o el rosa, cambian los valores de varios *bits*.

Mediante pruebas con la cámara y el espectro de colores, se obtuvieron los siguientes resultados:

- Color negro asociado al *bit 14,* con un valor próximo a 1.
- Color azul asociado al *bit* 11, con un valor próximo a 1.
- Color rojo asociado al *bit* 1, con un valor próximo a 1.

Por lo tanto, en la figura 88 se muestra como, de los *bits* del espectro asociados al negro, azul y rojo, se realiza una comparación. De tal modo que, si el valor adquirido por dichos *bits* está comprendido entre 0,4 y 1, se activan los respectivos colores.

Para la asignación del respectivo color a la variable *DatoCarga* de las listas FIFO, se ha tenido que implementar el siguiente programa:

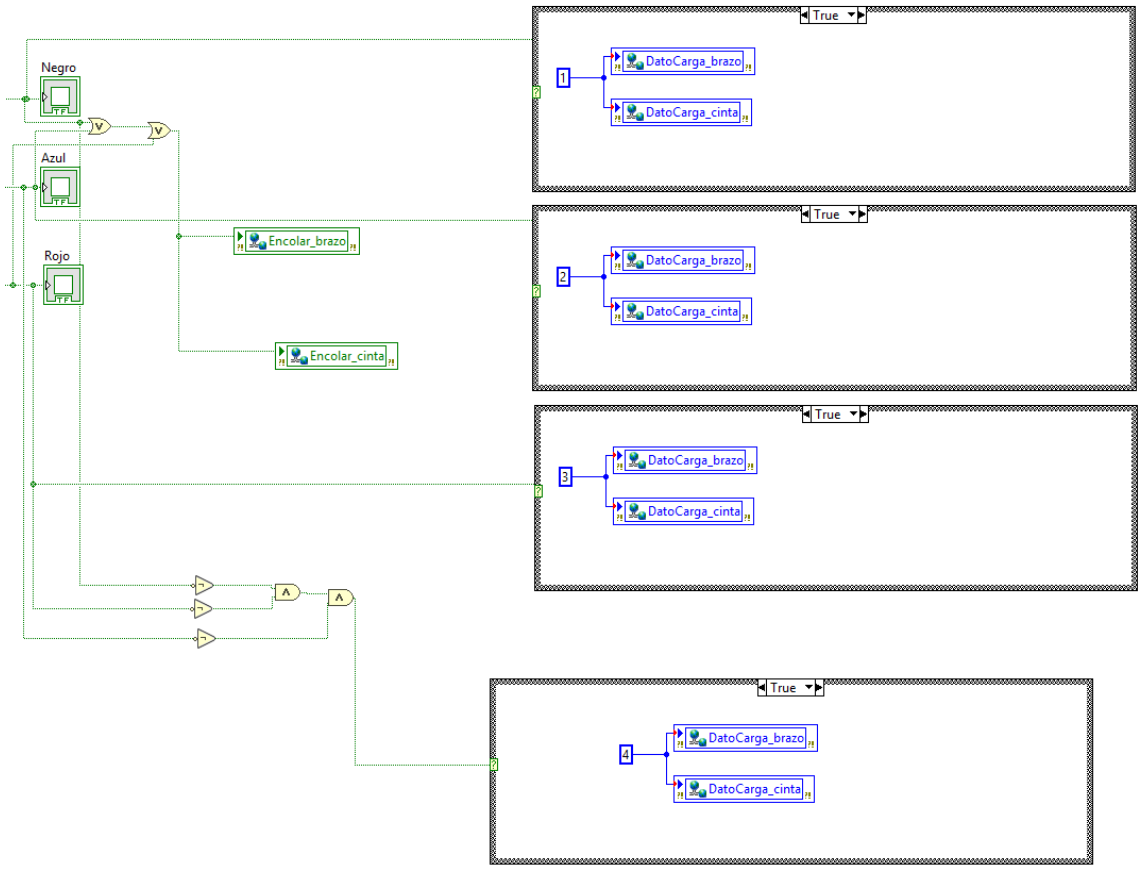


Figura 92. Asociación de cada color a la variable *DatoCarga* de las listas FIFO en ambos autómatas.

Como se observa en la figura anterior, cuando se detecta el color negro, se activa la sentencia ***if***, donde si se cumple la condición (Negro = *TRUE*), se escriben en las variables *DatoCarga_brazo* y *DatoCarga_cinta* un 1 (dato tipo *int*). Dichas variables están asociadas a las mismas direcciones de memoria que la variable *DatoCarga* (de cada autómata) de las listas FIFO.

Lo mismo ocurre para el azul y el rojo, siendo los números 2 y 3 escritos en las variables, respectivamente. En el caso de que no se detecte ninguno de los tres colores, se envía un 4 a las variables de las FIFO.

Cabe mencionar que en un principio, los encolamientos en las listas FIFO de los valores que representan a los colores, se producían en el momento en el que se detectaban dichos colores (como se puede apreciar en la figura 90). El problema era que muchas veces (sobre todo por temas de iluminación, algo muy importante en la industria), al detectarse un color, el programa hacía conmutaciones rápidas (en cuestión de milisegundos) de dicha detección, de tal modo que se encolaba más de una vez el valor asociado al color detectado.

Debido a dicho problema, las listas FIFO almacenaban, por ejemplo, 6 veces el mismo dato, cuando en realidad solo debía almacenarlo una sola vez. Para solventar el problema, se realizó en lenguaje SFC un programa para que encolara el dato si éste se había mantenido en el tiempo durante 1,5 segundos, evitando de este modo las conmutaciones rápidas.

Nota: Dicho programa puede encontrarse en el anexo **1**.

Para la visualización del número de piezas que están siendo mecanizadas, se han asignado las variables a unos indicadores:

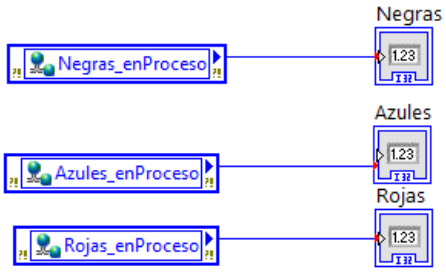


Figura 93. Asignación de las variables de piezas en proceso de mecanizado a indicadores.

Al terminar el programa mediante el botón *Stop*, se debe terminar también el uso de la *Kinect*, como también volver su inclinación al estado inicial, es decir, a cero:

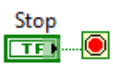


Figura 94. Botón de parada.

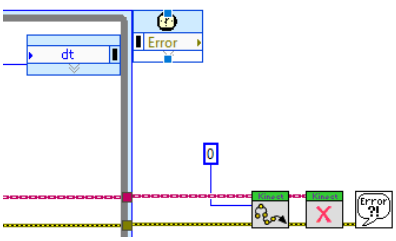


Figura 95. Terminación del programa. Salida del bucle temporal, condicionado a la parada.

5. Puesta en marcha del sistema

Para utilizar el sistema, primero se debe inicializar los programas del *SoMachine* y su previa conexión con los autómatas a utilizar antes de su ejecución, como también la puesta de las dos maquetas y sus conexiones a los PLCs. Cabe mencionar que la ubicación de la cámara es a convenio del personal a cargo, ya que la región de interés (ROI) es configurable desde el propio SCADA en *LabVIEW.* El seguimiento de los siguientes pasos es fundamental para un buen funcionamiento del sistema.

5.1 Preparación de los sistemas físicos

5.1.1 Autómatas *Modicon M241*

1. Conectar el enchufe de los autómatas a la red eléctrica, como también los cables de red *Ethernet*, los cuales deben ir conectados a la red global de la planta y al ordenador (dependerá del tipo de bus de comunicación utilizado).

2. Activar el diferencial de seguridad de los autómatas para ponerlos en funcionamiento.

3. Conectar cada autómata (mediante conectores industriales) a la respectiva maqueta de *FischerTechnik* o sistema real, en cuyo caso, tanto el funcionamiento del sistema como las características de los sensores, actuadores y elementos físicos deberán ser exactamente las mismas al de las maquetas (en caso contrario, se pueden producir daños graves e irreversibles).

5.1.2 Maquetas *FischerTechnik*

1. Colocar ambas maquetas exactamente en las posiciones detalladas en el apartado **4.2**(Distribución en planta de los sistemas FischerTechnik).
   ADVERTENCIA: Debe asegurarse que las maquetas están ubicadas en un lugar despejado, sin elementos que pudieran dificultar las tareas, especialmente los movimientos del robot manipulador.

2. La puesta de las piezas a mecanizar en el lugar de trabajo debe ser el indicado en el apartado **4.2**.

5.1.3 Sensor *Kinect*

1. Conectar el cable USB tanto al adaptador *Kinect* como al ordenador:

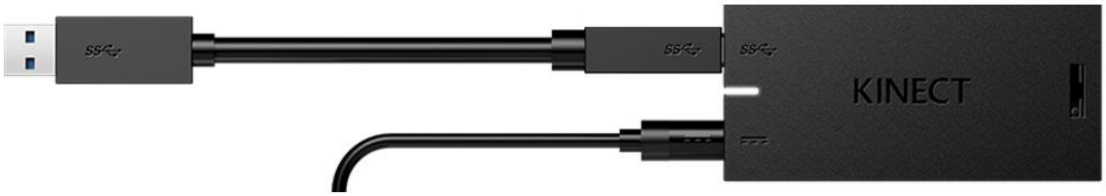


Figura 96. Conexión entre cable USB y adaptador *Kinect*.

2. De la misma manera que el paso anterior, conectar el enchufe tanto al adaptador *Kinect* como a la red eléctrica.

3. Para finalizar, conectar el adaptador *Kinect* al propio sensor *Kinect*:

Figura 97. Conexiones del sensor *Kinect.*

5.2 Preparación y ejecución del sistema de control con *SoMachine*

1. Ejecutar el fichero del programa creado con *SoMachine.*

2. En la ventana que se muestra al ejecutar el fichero, abrir el proyecto para visualizar la programación, SCADA y configuración:

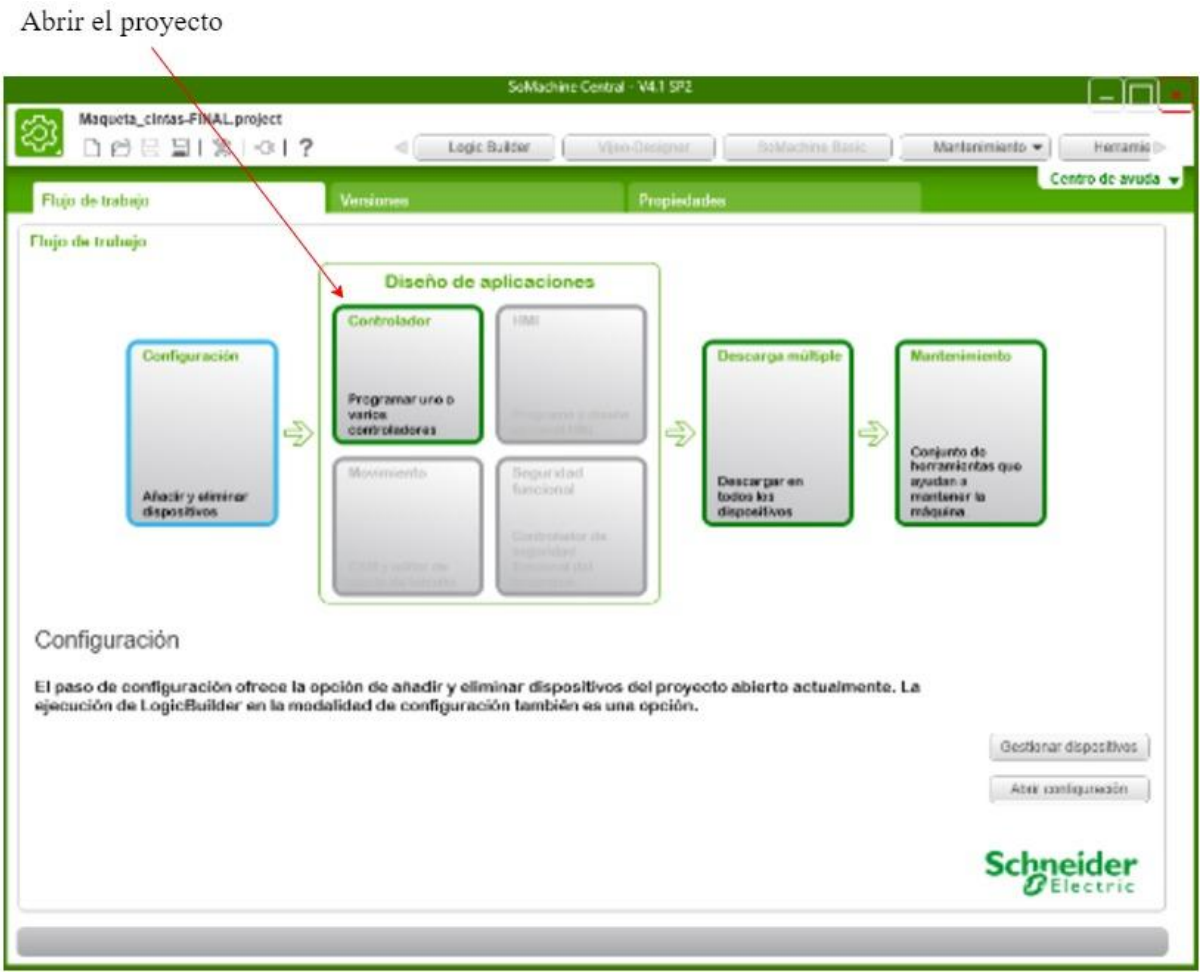


Figura 98. Abrir el proyecto del control del sistema con *SoMachine*

3. Establecer la conexión entre *SoMachine* y el autómata. Para ello, ir a la ventana **Dispositivos** y hacer doble *click* en ***MyController***:

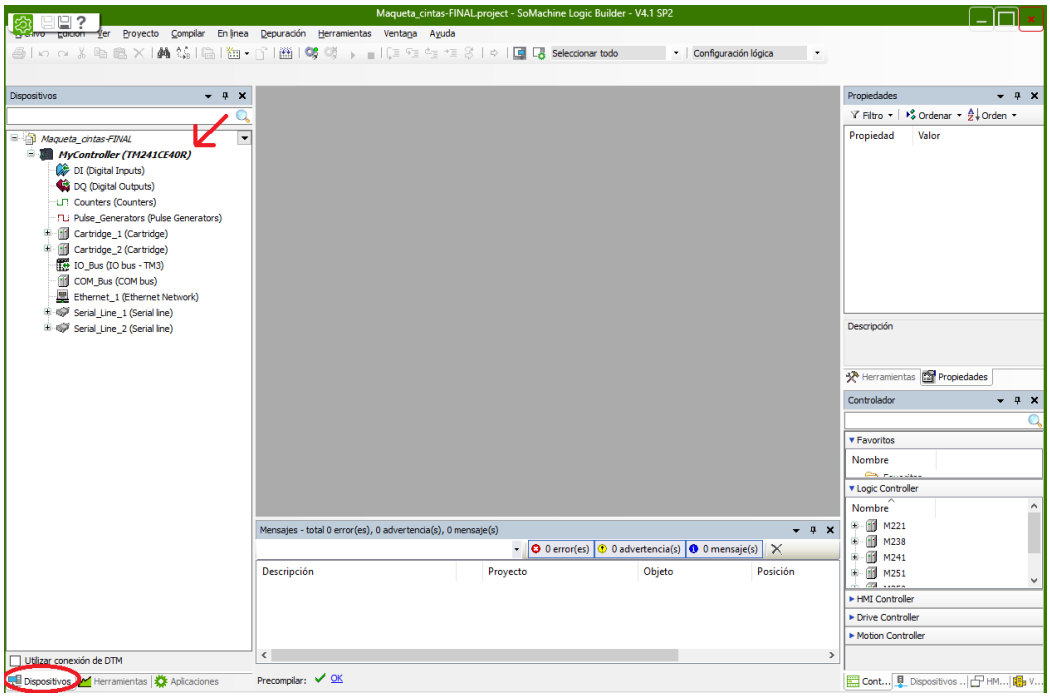


Figura 99. Acceso a la configuración de comunicación *software*-autómata.

4. Al abrirse la pantalla de ***MyController***, cambiar la modalidad de conexión a **Dirección IP** y hacer doble *click* sobre el controlador que se vaya a utilizar (fijarse en la IP para determinar cuál de todos los que aparecen es el que se va a utilizar):

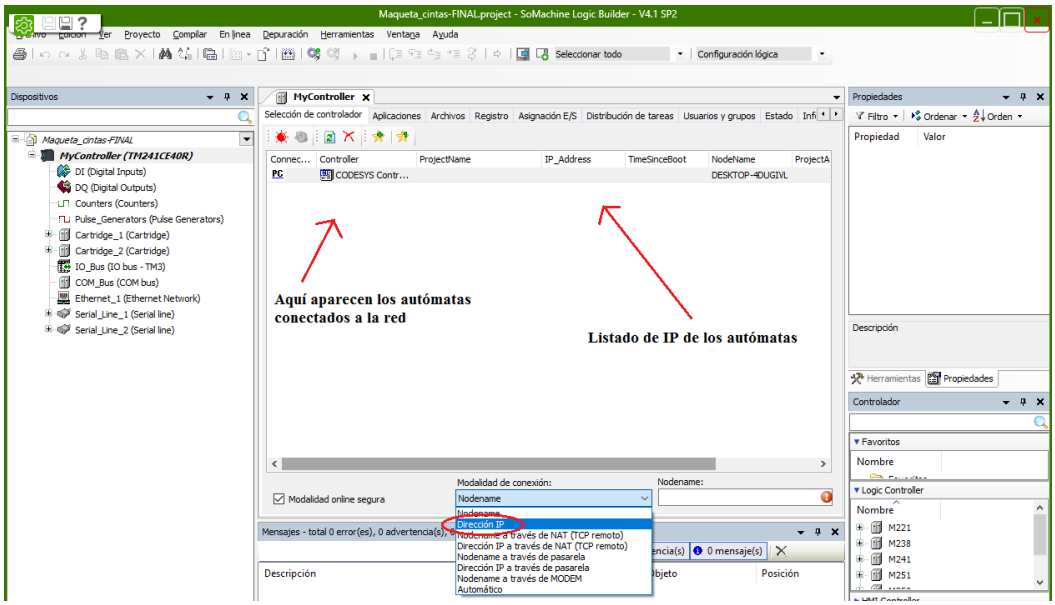


Figura 100. Conexión del *software* al

Nota: Si el autómata está mal conectado o no está conectado a la red, no aparecerá en pantalla, tal y como ocurre en la figura 100.

5. Una vez enlazado al autómata, se puede hacer parpadear los LEDs de éste haciendo *click* en **Óptica**, a modo de comprobación:

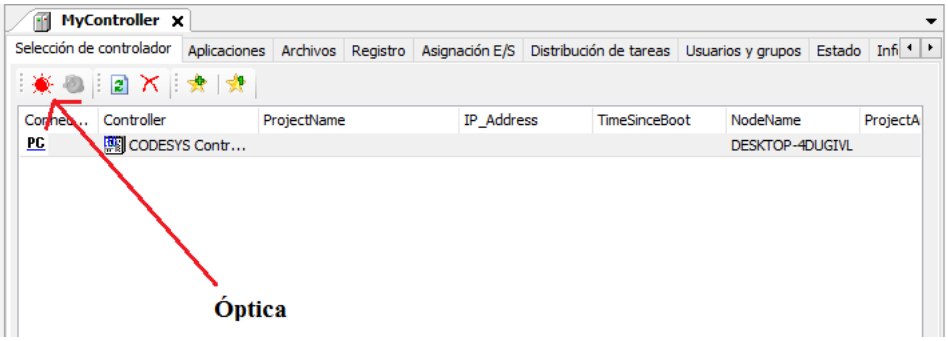


Figura 101. Comprobación de la conexión mediante parpadeo de LEDs.

6. Para ejecutar el programa, hacer *click* en **Iniciar la sesión** (Alt + F8):

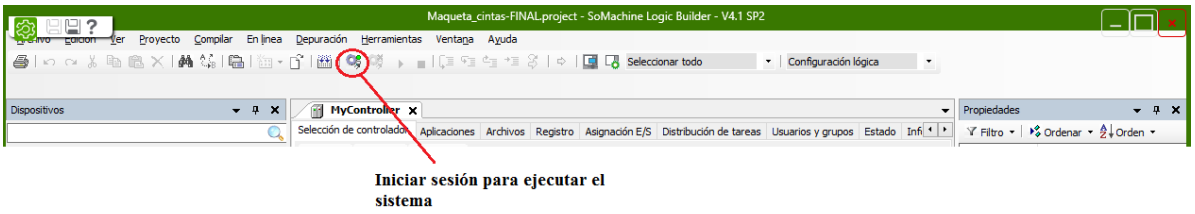


Figura 102. Ejecución del programa mediante inicio de sesión.

AVISO: Antes de realizar este paso, asegúrese de que el sistema físico está perfectamente instalado y configurado, como también de todas las conexiones (tanto físicas como a niveles de red) entre sistemas y PLCs. También se requiere que el sistema esté en reposo. Ejecutar un programa en una instalación en plena producción puede ocasionar desperfectos a las piezas, como también daños graves a la maquinaria y los operarios a cargo.

7. Para visualizar el SCADA se debe ir a la pantalla **Herramientas** y hacer doble *click* en cada una de las interfaces implementadas:

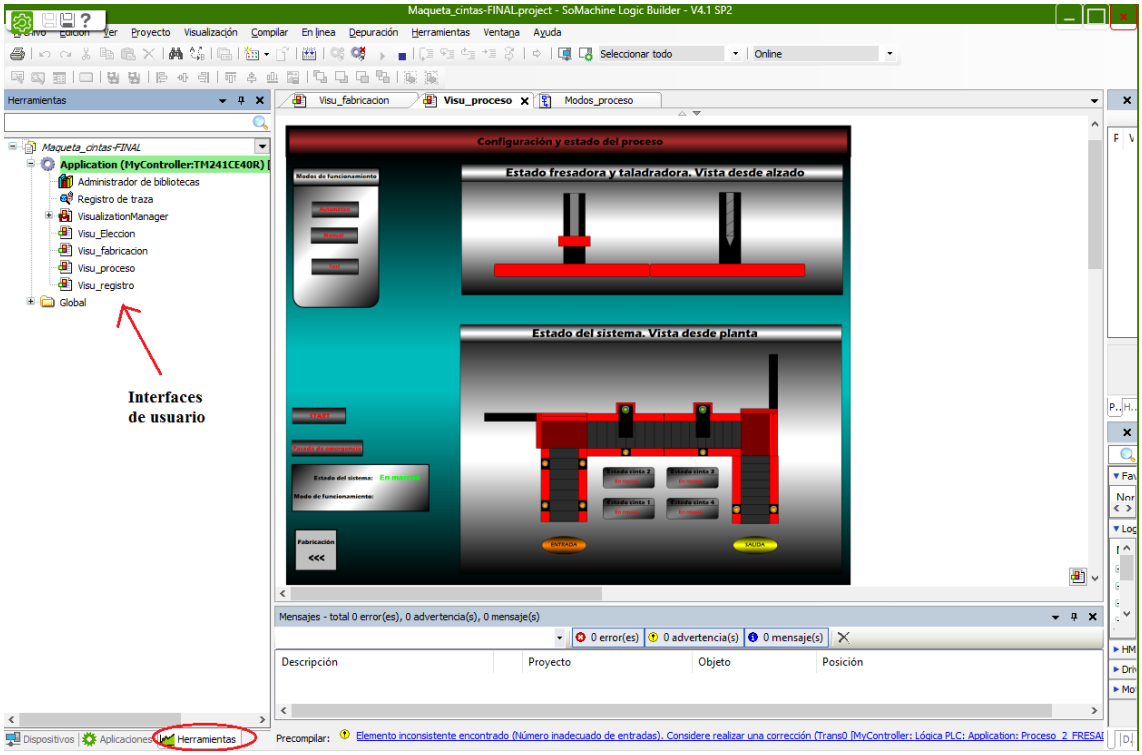


Figura 103. Visualización de las interfaces de usuario del SCADA.

Todo el procedimiento realizado se debe hacer para los dos autómatas, cada uno conectado al programa correspondiente, esto es, a cada proyecto realizado con *SoMachine* (un proyecto para el sistema de mecanizado y otro para el robot manipulador).

IMPORTANTE: Para el proyecto realizado con *SoMachine* del sistema de mecanizado es **necesario** verificar que la IP del autómata conectado al robot manipulador coincide con la IP escrita en el bloque **ADDM** de la comunicación entre ambos autómatas. De lo contrario, el sistema no funcionaria correctamente (la tarea de recogida y clasificación de piezas no se ejecutaría automáticamente).

Para comprobar la IP, se debe abrir el programa referente a la comunicación entre PLCs, para ello, se abre la ventana **Aplicaciones** y se realiza doble *click* en el programa **ComunicacionesMaestro**:

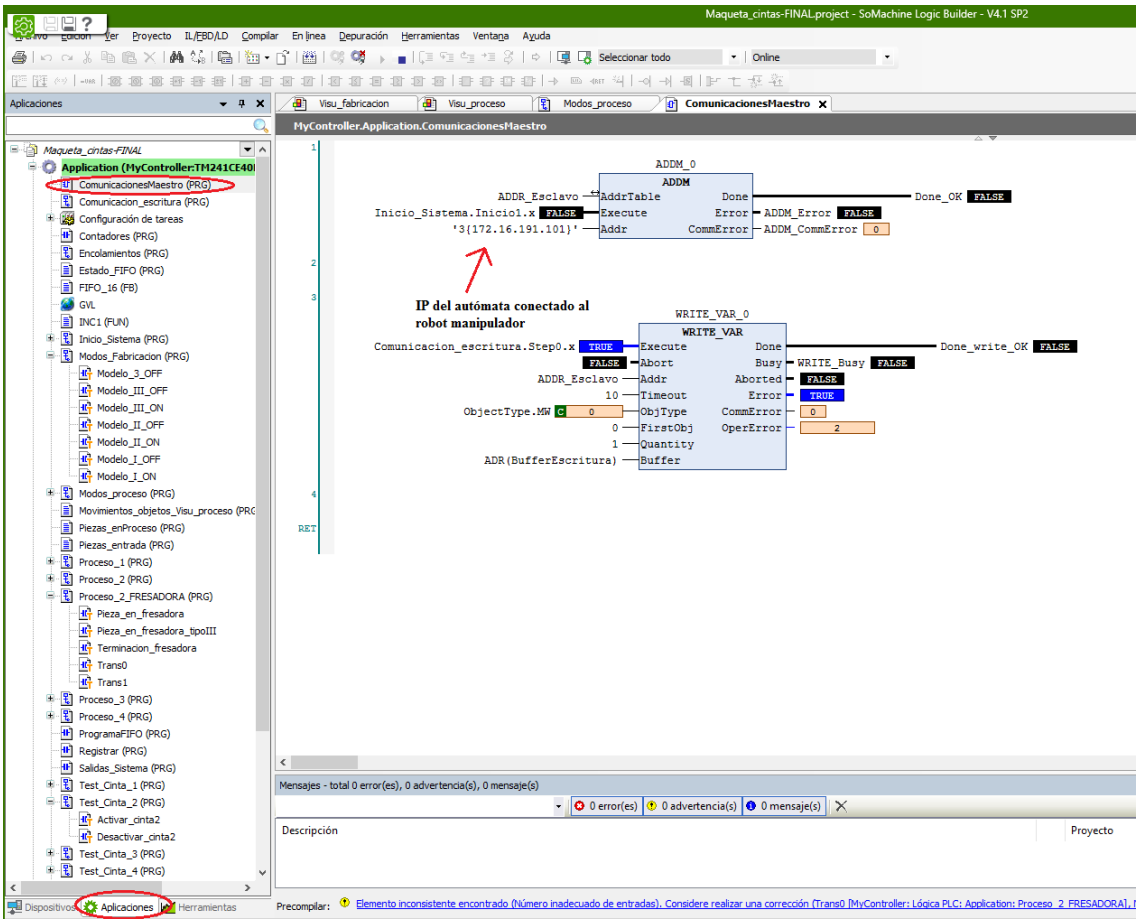


Figura 104. Comprobación de la IP en el bloque ADDM de la comunicación maestro / esclavo.

5.3 Ejecución del servidor OPC y *LabVIEW*

Para el funcionamiento de ambos *software*, simplemente se debe abrir los proyectos ya existentes para el proyecto, sin necesidad de configuraciones (excepto los dispositivos del OPC, ya que si se usan autómatas diferentes, las IP de estos deben cambiarse):

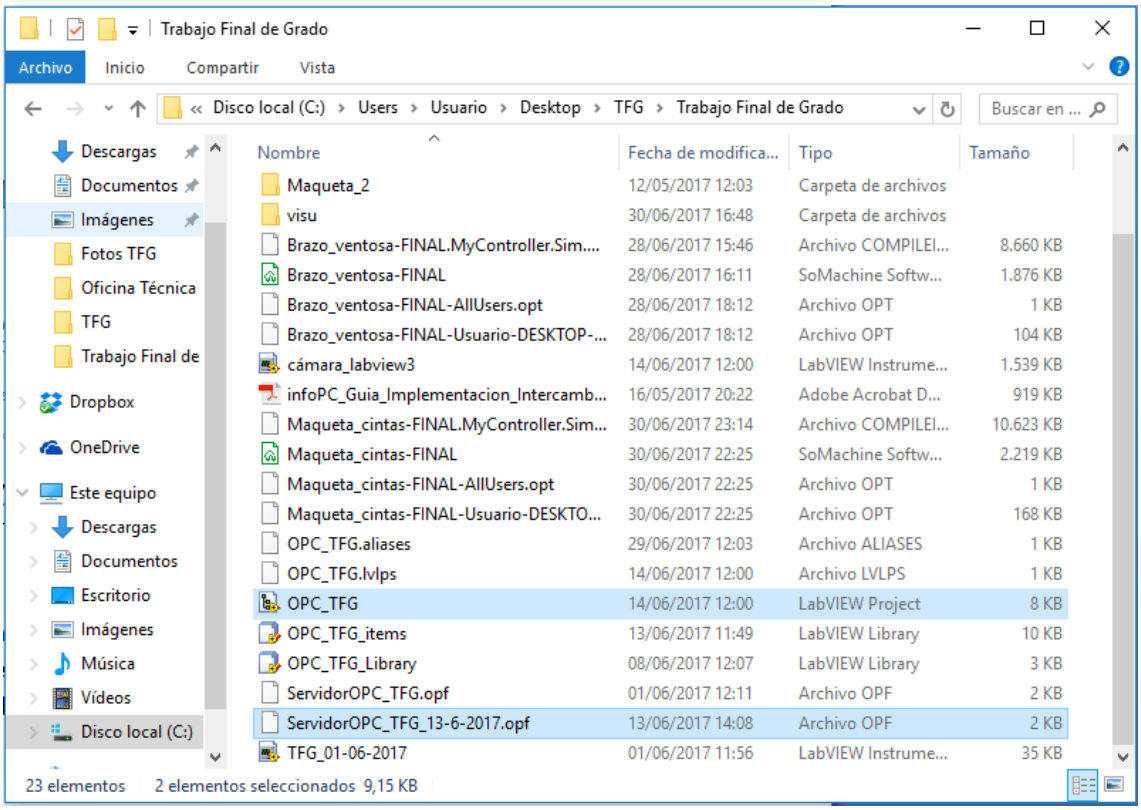


Figura 105. Archivos a ejecutarse para el funcionamiento del servidor OPC y del SCADA en *LabVIEW*.

Para crear un proyecto nuevo, tanto para el servidor OPC como para *LabVIEW*, **véase apartados 4.5 y 4.6.1**, respectivamente).

6. Conclusiones

Tras la finalización de todo el sistema automático y sus respectivos documentos, se han obtenido unas valiosas conclusiones. Dichas conclusiones no solo validan este trabajo, sino que además, reflejan las competencias, destrezas y conocimientos adquiridos por el alumno a través de este trabajo académico de final de grado.

A nivel técnico, se han concluido los siguientes aspectos:

- Tanto la elección del autómata como del *software* es de vital importancia para conseguir una optimización de los recursos disponibles.
- Se debe conocer previamente los elementos físicos a automatizar, incluidas sus conexiones, para realizar correctamente las asignaciones de E/S.
- La cantidad de variables a manejar en un sistema automático es muy elevada, de manera que realizar pequeñas anotaciones de cada una de ellas facilita mucho la comprensión del sistema.
- La comunicación entre dispositivos es algo muy utilizado en la industria, ya que se requiere que todos los elementos estén conectados para lograr una optimización en la transferencia de datos y una perfecta coordinación.
- Es muy importante controlar el tiempo de ejecución de las tareas para que no se produzcan errores en la ejecución de las mismas o de tareas dependientes de otras.
- Se debe incidir de manera notoria en la seguridad de máquinas y operarios a la hora de realizar un sistema automatizado.
- Es de suma importancia conocer las leyes y normativas vigentes antes de empezar un proyecto técnico.
- La redacción técnica del proyecto debe ser clara y detallada para evitar problemas a la hora de la implementación y uso del sistema.
- En la utilización de cámaras industriales, se debe tratar la iluminación tanto a nivel interno (filtros) como a nivel externo (luces, focos…).

A nivel personal, se ha llegado a las siguientes conclusiones:

- Se requiere un amplio conocimiento de autómatas y sus respectivos *software* dado que hay una infinitud de ellos por todo el mercado.
- Realizar una puesta en marcha es complejo dado que se requiere haber verificado el funcionamiento del sistema programado previamente.
- Al automatizar un sistema, se debe ser cauto, dado que un simple error en la programación o una condición mal estructurada podría ocasionar grandes daños a la maquinaria y al personal a cargo.
- Automatizar un sistema no requiere únicamente de conocimientos de programación y diseño de visualizaciones, sino también de conocimientos de sensores y actuadores utilizados en la industria, como también de comunicaciones de redes.
- Este trabajo ha ayudado considerablemente al alumno, no solo a consolidar sus conocimientos adquiridos previamente, más aun, a aprender cómo diseñan, programan e implementan los profesionales un sistema automático industrial.

7. Glosario

PLC: Controlador lógico programable (PLC) es un ordenador o microordenador muy utilizado en ingeniería automática o automatización industrial, para automatizar procesos electromecánicos, tales como el control de la maquinaria de la fábrica en líneas de montaje o atracciones mecánicas, ya que está diseñado para múltiples señales de entrada y salida.

SCADA: Acrónimo de *Supervisory Control And Data Acquisition* (Supervisión, Control y Adquisición de Datos). Se trata de un *software* para ordenadores que permite controlar y supervisar procesos industriales a distancia. Facilita la retroalimentación en tiempo real con los dispositivos de campo (sensores y actuadores), y controla el proceso automáticamente. Provee de toda la información que se genera en el proceso productivo (supervisión, control calidad, control de producción, almacenamiento de datos, etc.) y permite su gestión e intervención.

Servidor OPC: Se trata de un estándar de comunicación en el campo del control y supervisión de procesos industriales, basado en una tecnología Microsoft, que ofrece una interfaz común para la comunicación que permite que componentes *software* individuales interactúen y compartan datos. La comunicación OPC se realiza a través de una arquitectura Cliente-servidor. Cualquier aplicación basada en OPC puede acceder a dicho servidor para leer/escribir cualquier variable que ofrezca el servidor.

HMI: La interfaz de usuario (*Human Machine Interface*, HMI) es un software visualizado en una pantalla donde se producen las interacciones entre seres humanos y máquinas. El objetivo de esta interacción es permitir el funcionamiento y control más efectivo de la máquina desde la interacción con el humano.

Control automático: Sistema de control que funciona por sí solo, sin ninguna o con escasa intervención humana, siendo previamente programado por el ser humano mediante el uso de dispositivos de control como PLC o microcontroladores.

8. Bibliografía

[1]
AENOR, «AENOR: Norma UNE-EN 60848:2013», *http://www.aenor.es/*. [En línea]. Disponible en: http://www.aenor.es/aenor/normas/normas/fichanorma.asp?tipo=N&codigo=N0051395. [Accedido: 19-jun-2017].

[2]
AENOR, «AENOR: Norma UNE-EN 61000-6-2:2006», *http://www.aenor.es/*. [En línea]. Disponible en: http://www.aenor.es/aenor/normas/normas/fichanorma.asp?tipo=N&codigo=N0035282. [Accedido: 19-jun-2017].

[3]
«Cámaras de Visión Artificial | Sistemas de visión Artificial para INDUSTRIA | INFAIMON». [En línea]. Disponible en: http://www.infaimon.com/es/camaras-industria. [Accedido: 22-jun-2017].

[4]
«Comparacion_SoMachine_3.pdf». .

[5]
«Componentes de Kinect | Partes de Kinect | Partes de Kinect de Xbox». [En línea]. Disponible en: http://support.xbox.com/es-PY/xbox-360/kinect/kinect-sensor-components. [Accedido: 22-jun-2017].

[6]
«Controlador lógico - Modicon M221 | Schneider Electric España». [En línea]. Disponible en: /es/product-range/62128-controlador-logico---modicon-m221/. [Accedido: 17-jun-2017].

[7]
«IEC 60870-5-101», *Wikipedia, la enciclopedia libre*. 24-oct-2016.

[8]
«infoPLC_net_Intro_estandar_IEC_61131-3.pdf». .

[9]
«infoPLC_net_SoMachine_Manual_de_formacion.pdf». .

[10]
«Lógica cableada y lógica programada. - Disseny Producte.» .

[11]
«microcontrolador | Real Academia de Ingeniería». [En línea]. Disponible en: http://diccionario.raing.es/es/lema/microcontrolador-0. [Accedido: 20-jun-2017].

[12]
«Modicon M241 | Schneider Electric». [En línea]. Disponible en: /es/product-range/62129-modicon-m241/. [Accedido: 21-jun-2017].

[13]
«OPC», *Wikipedia, la enciclopedia libre*. 29-jun-2017.

[14]
«plds.pdf». .

[15]
«referencia_normas_iso_iec_reg_tecnica.pdf». .

[16]
«RO-BOTICA: Linea indexada 24V PLC Fischertechnik Industry». [En línea]. Disponible en: https://www.ro-botica.com/Producto/Linea-indexada-Fischertechnik-Industry/. [Accedido: 22-jun-2017].

[17]
«RO-BOTICA, tu tienda de robótica para educación secundaria y superior con fischertechnik education». [En línea]. Disponible en: https://ro-botica.com/producto/Manipulador-de-aspiracion-al-vacio-fischertechnik-9V. [Accedido: 25-jun-2017].

[18]
«S71200-MANUAL DEL SISTEMA.pdf». .

[19]
«SCADA», *Wikipedia, la enciclopedia libre*. 16-may-2017.

[20]

«Search results for “IEC 62541” | IEC Webstore». [En línea]. Disponible en: https://webstore.iec.ch/searchform&q=IEC%2062541. [Accedido: 19-jun-2017].

[21] «SoMachine | Schneider Electric España». [En línea]. Disponible en: /es/product-range/2226-somachine/. [Accedido: 17-jun-2017].

[22] «Tarjetas de adquisición de datos». [En línea]. Disponible en: https://www.jmi.com.mx/tarjetas-de-adquisicion-de-datos. [Accedido: 17-jun-2017].

[23] «Todo sobre el Kinect de Xbox 360 – Características y Precio», *ComuSOFT.com*. .

[24] «Ventajas y Desventajas de los PLC (Controlador Lógico Programable) - Noticias fabricantes maquinaria industrial, automatizacion y maquinas», *Blog de los fabricantes de maquinaria Industrial*, 01-feb-2014. .

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

TRABAJO FIN DE GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

DESARROLLO E IMPLEMENTACIÓN DE LA AUTOMATIZACIÓN PARA UN SISTEMA DE MECANIZADO DE PIEZAS CON ROBOT MANIPULADOR MEDIANTE COMUNICACIONES DISTRIBUIDAS CON SERVIDOR OPC Y SUPERVISIÓN SCADA

DOCUMENTO Nº 2 PLIEGO DE CONDICIONES

AUTOR: ABEL DOMÍNGUEZ GONZÁLEZ

TUTOR: RAÚL SIMARRO FERNÁNDEZ

Curso Académico 2016-17

Índice

1. Definición y alcance del pliego

El objeto de este documento es fijar las condiciones técnicas mínimas que debe cumplir tanto la automatización como la puesta en marcha del sistema de mecanizado con robot manipulador (dispositivos, conexiones, monitorización…).

El ámbito de aplicación de este documento se extiende únicamente a los sistemas de control diseñados e implementados en el sistema de mecanizado y clasificador de piezas, tanto a nivel de *hardware* (autómatas y cámara) como a nivel de *software* (SCADA y servidor OPC).

En determinados supuestos se podrá adoptar diferentes soluciones a las exigidas en este documento, siempre que se cumplan las normativas y exigencias del mismo y no reduzca la seguridad y calidad de los propios sistemas.

2. Condiciones y normas de carácter general

Para la ejecución de éste proyecto es obligatorio el cumplimiento de las siguientes normativas:

- **IEC 61131-3.** Estandarización completa de los lenguajes de programación de los controladores lógicos programables.
- **IEC 62541.** Especificaciones de la plataforma de comunicación universal OPC, para modelos estándar de información.
- **IEC 60870-5-101.** Definición el uso de una red TCP/IP.
- **UNE-EN 60848:2013.** Lenguaje de especificación GRAFCET para diagramas funcionales secuenciales.
- **UNE-EN 61000-6-2.** Compatibilidad electromagnética (CEM). Parte 6-2: Normas genéricas. Inmunidad en entornos industriales.
- **EN ISO 13849-1:2006.** Seguridad de las máquinas. Partes del sistema de mando relativas a seguridad. Parte 1: Principios generales para el diseño.

3. Condiciones técnicas

3.1 Condiciones de los materiales

3.1.1 Controlador lógico programable

Descripción

Autómata programable de 10 entradas o más (3 de lectura rápida), siendo el número de salidas superior a 11. Capacidad tanto de conexión a red mediante clave *Ethernet* y puerto serie como vía *Modbus TCP/IP*, como también la visualización de las interfaces por medio de gestor web. Alimentación entre 100-240 V de corriente alterna.

Como modelo, se recomienda el ***Modicon M241*** *(TM241CE40R)* de *Schneider Electric*.

Control de calidad

Se conectará el autómata a la red eléctrica y se encenderá el diferencial de seguridad para verificar que el autómata se enciende correctamente (parpadeo de varios LEDs). Para verificar su conexión a la red, se conectará el cable *Ethernet* al puerto serie y mediante *SoMachine* se verificará si se detecta o no el autómata en la red.

3.1.2 Cámara de visión

Descripción

Cámara dotada con sensor RGB e inclinación vertical de la propia base mediante motor. Compatibilidad con el *software LabVIEW*.

Se recomienda el uso de la cámara o sensor *Kinect*, debido a que es el utilizado en el proyecto y cumple con los requisitos mencionados.

Control de calidad

Se conectará la cámara tanto a la red eléctrica como al ordenador para verificar que se enciende (parpadeo del LED) y que el propio ordenador la reconoce (instalación y actualización de *drivers* de la *Kinect*).

3.2 Condiciones de ejecución

3.2.1 Responsabilidad del ejecutor del proyecto

Es responsabilidad del ejecutor del proyecto los siguientes puntos:

- Desarrollo e implementación de todos los sistemas de control.
- La aportación de dos autómatas *Modicon M241* y un sensor *Kinect.*
- Conexión de los dispositivos con los sistemas industriales.
- Puesta en marcha del sistema
- Verificaciones de funcionamiento, tanto de monitorización de variables como de modos de funcionamiento.

3.2.2 Conexiones del sistema

Descripción

Se debe conectar cada autómata a cada uno de los sistemas físicos, es decir, el sistema de mecanizado y el robot manipulador. Asegurarse que las conexiones entre autómatas y sistemas quedan bien sujetas, para ello, asegúrese de que el conector industrial queda atornillado. Los autómatas deben quedar guardados en un lugar seco y seguro.

Controles de calidad

Se deberá revisar con regularidad que las conexiones siguen bien sujetas, como también de que los autómatas están en buen estado. Se recomienda que se conecten los autómatas a una bancada para mayor seguridad.

3.2.3 Ejecución de los sistemas de control

Descripción

Se debe inicializar los proyectos de *SoMachine* y *LabVIEW* para ejecutar tanto los SCADA de ambos sistemas (mecanizado y robot) como el SCADA de visión (cámara). Cabe mencionar que los SCADA de *SoMachine* pueden ejecutarse desde el propio ordenador o desde cualquier dispositivo conectado a la red (móvil, tablet…). En el caso del SCADA de la cámara, éste debe ser ejecutado desde el propio ordenador del puesto de trabajo obligatoriamente.

Nota: Si se quisiera realizar nuevos sistemas de control excluyendo los ya creados, los pasos a seguir figuran en el la memoria de este proyecto.

Controles de calidad

Para verificar que los sistemas de control del mecanizado y robot manipulador funcionan, se deberá activar el modo manual para realizar pruebas de test y comprobar que los actuadores seleccionados mediante la interfaz de usuario responden adecuadamente. Es muy recomendable además, utilizar el modo automático con un par de piezas defectuosas o inservibles a modo de prueba, para comprobar que funciona y, en el peor de los casos, no obtener pérdidas.

En el caso del control mediante visión, se puede realizar test con colores, para verificar que la cámara y el programa reconocen los colores de manera adecuada. También se recomienda encarecidamente utilizar filtros en la iluminación y que esta sea siempre la misma a ser posible, para evitar problemas en el reconocimiento de colores.

3.2.4 Ejecución del servidor OPC

Descripción

Simplemente se debe inicializar el servidor con el *software KEPServerEX* mediante el archivo creado para este proyecto. Las variables del sistema deberán visualizarse en pantalla.

Nota: Si se quisiera crear un nuevo servidor OPC excluyendo el ya creado, los pasos a seguir para la configuración y asignación de variables se encuentra en la memoria de este proyecto.

Controles de calidad

Para verificar que las variables deseadas se encuentran conectadas en el servidor OPC, se debe ejecutar el modo "*Quick Client".* De este modo se visualizará por pantalla el estado de las variables, sabiendo si están conectadas al servidor y compartiendo la información.

3.3 Condiciones de garantía

1. El sistema de control de todo el proceso tiene una garantía de 2 años, cubriendo únicamente fallos debidos al propio sistema de control (tanto *hardware* como *software*).

2. El proyectista no se hace responsable de posibles daños ocasionados por los sistemas físicos (empujadores, motores…) si éstos están en mal estado o por parte de un mal uso del sistema de control (caso omiso de las instrucciones y precauciones).

3. Si el sistema de control es manipulado por terceros (cambios en el *software*), se anula la garantía de 2 años.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

TRABAJO FIN DE GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

DESARROLLO E IMPLEMENTACIÓN DE LA AUTOMATIZACIÓN PARA UN SISTEMA DE MECANIZADO DE PIEZAS CON ROBOT MANIPULADOR MEDIANTE COMUNICACIONES DISTRIBUIDAS CON SERVIDOR OPC Y SUPERVISIÓN SCADA

DOCUMENTO Nº 3 PRESUPUESTO

AUTOR: ABEL DOMÍNGUEZ GONZÁLEZ

TUTOR: RAÚL SIMARRO FERNÁNDEZ

Curso Académico 2016-17

Índice

Índice de Tablas

1. Introducción

En este documento se expone el presupuesto del proyecto, tanto el coste del diseño e implementación de los sistemas de control y SCADA como la redacción del mismo y su posterior puesta en marcha.

Cabe mencionar que los sistemas físicos (sistema de mecanizado y robot manipulador) son elementos ya adquiridos por el cliente, por ello no han sido tenidos en cuenta en este documento.

2. Cuadro de precios elementales

En este apartado se procede a la determinación de los costes del proyecto según su naturaleza.

2.1 Cuadro de precios nº 1: Mano de obra

Los salarios atribuidos al ingeniero electrónico industrial y automático (con atribuciones de ingeniero técnico industrial) han sido obtenidos de la UGT FICA Industria y construcción del País Valenciano.

- Ingeniero electrónico industrial y automático (categorizado como personal técnico):

Concepto	Importe (€)
Salario base (mensual)	**1.824,83**
Pagas extra (Julio y navidad)	**147,64**
Plus salarial (carencia incentivos, movilidad, asistencia)	81,86 + 15,35 + 15,35 = **112,56**
Cotización a la Seguridad Social:	
- Contingencias comunes (28,3%)	**590,06**
- I.R.P.F (15%)	**312,75**
- Desempleo (7,05%)	**147,00**
- Formación (0.7%)	**14,60**
- Accidentes de trabajo y enfermedad (1%)	**20,85**
- FOGASA (0,2%)	**4,17**
TOTAL MENSUAL:	**3.174,46**
TOTAL ANUAL:	**38.093,52**
POR JORNADA MENSUAL (22 días):	**144,30**
POR HORA:	**18,03**

Tabla 1. Cuadro de precios de mano de obra. Coste de contratación de un ingeniero técnico industrial.

Cabe resaltar que las pagas extra y plus salarial se han expresado como pagas mensuales, de manera que los porcentajes aplicados de la seguridad social hacen referencia a cada mes de trabajo.

Se ha necesitado de aproximadamente dos meses (37 días hábiles) para la realización del proyecto, estimando unas horas de trabajo por parte del ingeniero electrónico industrial y automático de 296 h.

De tal modo que el precio de mano de obra asciende a:

Ref	Ud	Descripción	Precio (€/h)	Tiempo (h)	Importe (€)
h1	h	Ingeniero Electrónico Industrial y automático	18,03	296	5.336,88

Tabla 2. Coste de la mano de obra del proyecto.

2.2 Cuadro de precios nº 2: Materiales

Primero se debe determinar cómo calcular la amortización de los materiales. Para ello, se debe saber la base amortizable, esto es, la diferencia entre el precio del material y su valor residual (a estimar). Por otra parte, se debe saber o bien el porcentaje de amortización lineal máximo o bien los años de amortización máximos, proporcionados por la agencia tributaria.

$$Base\ amortizable = Precio\ material - Valor\ residual$$

	Porcentaje de amortización lineal máximo (%)	Periodo de años amortizables máximo
Dispositivos electrónicos	20	10
Sistemas y programas informáticos	33	6

Tabla 3. Valores de amortización extraídos de la agencia tributaria.

Para calcular la amortización existen dos maneras:

1. $Amortización = Base\ amortizable \cdot coeficiente\ de\ amortización$ (€/año)
2. $Amortización = \frac{Base\ amortizable}{Periodo\ de\ años}$

Para este proyecto, se ha utilizado para los dispositivos electrónicos un coeficiente de amortización del 15 % y un periodo de amortización de 5 años. En el caso de los programas informáticos, un coeficiente del 20 % y un periodo de 1 año (debido a las licencias). **Nota:** En el periodo por años, según la agencia tributaria, cada año se invierten alrededor de 1800 h para proyectos (no el año completo).

Hardware

Dispositivo	Precio Unidad (€)	Valor residual (€)	Amortización por coeficiente (€/h)	Amortización por años (€/h)
Portátil *Toshiba*	500	40	0,0078	0,0511
Modicon M241	141,60	14	0,0022	0,0141
Sensor *Kinect*	99,99	9	0,0016	0,0101

Tabla 4. Coste a amortizar del *Hardware*.

Software

Aplicación *software*	Precio Unidad (€)	Valor residual (€)	Amortización por coeficiente (€/h)	Amortización por años (€/h)
SoMachine	0	0	0	0
LabVIEW	3.605	300	0,082	1,8360
KEPServerEX	381,38	38	0,0087	0,1907

Tabla 5. Coste a amortizar del *software*.

En base de las amortizaciones obtenidas, se elige la amortización por años ya que presenta un valor mayor (la licencia de *SoMachine* es gratuita). Por lo tanto, el coste del material del proyecto asciende a:

Ref	Ud	Descripción	Precio (€/h)	Tiempo de uso (h)	Importe (€)
m1	ud	Portátil *Toshiba*	0,0511	296	15,1256
m2	ud	*Modicon M241*	0,0141	180	2,5380
m3	ud	Sensor *Kinect*	0,0101	40	0,4040
m4	ud	*SoMachine*	0	280	0
m5	ud	*LabVIEW*	1,8360	48	88,128
m6	ud	*KEPServerEX*	0,1907	26	4,9582
				TOTAL:	**111,1538**

Tabla 6. Costes de los materiales del proyecto.

3. Cuadro de precios descompuestos

Este apartado indica los precios de cada unidad de obra, teniendo en cuenta los materiales y mano de obra utilizada.

(d1) Diseño e implementación del sistema de control para el sistema de mecanizado					
Ref	**Ud**	**Descripción**	**Precio (€/h)**	**Cantidad**	**Importe (€)**
m1	ud	Portátil *Toshiba*	0,0511	0,340	0,0173
m2	*ud*	*Modicon M241*	0,0141	1	0,0141
h1	h	Ingeniero	18,03	0,340	6,1302
				Medios auxiliares sobre costes directos (10 %)	0,6161
				TOTAL:	**6,7777**

Tabla 7. Cuadro de precios descompuestos de la unidad de obra 1 (d1).

(d2) Diseño e implementación del sistema de control del robot manipulador					
Ref	**Ud**	**Descripción**	**Precio (€/h)**	**Cantidad**	**Importe (€)**
m1	ud	Portátil *Toshiba*	0,0511	0,310	0,0158
m2	*ud*	*Modicon M241*	0,0141	1	0,0141
h1	h	Ingeniero	18,03	0,310	5,5893
				Medios auxiliares sobre costes directos (10 %)	0,5619
				TOTAL:	**6,1811**

Tabla 8. Cuadro de precios descompuestos de la unidad de obra 2 (d2).

(d3) Diseño e implementación del servidor OPC					
Ref	**Ud**	**Descripción**	**Precio (€/h)**	**Cantidad**	**Importe (€)**
m1	ud	Portátil *Toshiba*	0,0511	0,120	0,006
m6	*ud*	*KEPServerEX*	0,1907	1	0,1907
h1	h	Ingeniero	18,03	0,120	2,1636
				Medios auxiliares sobre costes directos (10 %)	0,2360
				TOTAL:	**2,5966**

Tabla 9. Cuadro de precios descompuestos de la unidad de obra 3 (d3).

(d4) Diseño e implementación de la detección y clasificación de piezas (*Kinect +* SCADA *LabVIEW)*					
Ref	**Ud**	**Descripción**	**Precio (€/h)**	**Cantidad**	**Importe (€)**
m1	ud	Portátil *Toshiba*	0,0511	0,240	0,0122
m3	*ud*	Sensor *Kinect*	0,0101	1	0,1907
m5	*ud*	*LabVIEW*	1,8360	1	1,8360
h1	h	Ingeniero	18,03	0,240	4,3272
				Medios auxiliares sobre costes directos (10 %)	0,6366
				TOTAL:	**7,0027**

Tabla 10. Cuadro de precios descompuestos de la unidad de obra 4 (d4).

4. Cuadro de precios unitarios

Ref	Ud	Descripción	Precio (€/h)	Cantidad (h)	Importe (€)
d1	ud	Diseño e implementación del sistema de control para el sistema de mecanizado	6,7777	130	880,10
d2	ud	Diseño e implementación del sistema de control del robot manipulador	6,1811	95	587,20
d3	ud	Diseño e implementación del servidor OPC	2,5966	5	12,98
d4	ud	Diseño e implementación de la detección y clasificación de piezas (*Kinect* + SCADA *LabVIEW)*	7,0027	66	462,18

Tabla 11. Cuadro de precios unitarios donde se indica el precio por unidad de obra.

5. Valoración del presupuesto

En la siguiente tabla se muestra el presupuesto final de la ejecución del material, es decir, el importe que refleja el coste del trabajo:

Ref	Ud	Descripción	Precio (€/h)	Cantidad (h)	Importe (€)
d1	ud	Diseño e implementación del sistema de control para el sistema de mecanizado	6,7777	130	880,10
d2	ud	Diseño e implementación del sistema de control del robot manipulador	6,1811	95	587,20
d3	ud	Diseño e implementación del servidor OPC	2,5966	5	12,98
d4	ud	Diseño e implementación de la detección y clasificación de piezas (*Kinect* + SCADA *LabVIEW)*	7,0027	66	462,18
				TOTAL PRESUPUESTO EJECUCIÓN MATERIAL:	**1.942,46**

Tabla 12. Total presupuesto de ejecución material.

El presupuesto total de ejecución material asciende a un total de:

MIL NOVECIENTOS CUARENTA Y DOS EUROS CON CUARENTA Y SEIS

6. Resumen del presupuesto

PRESUPUESTO	IMPORTE (€)
PRESUPUESTO EJECUCIÓN MATERIAL	**1.942,46**
13% GASTOS GENERALES	252,52
6% BENEFICIO INDUSTRIAL	116,55
PRESUPUESTO EJECUCIÓN POR CONTRATA	**2311,53**
21% IVA	485,42
PRESUPUESTO BASE LICITACIÓN	**2796,95**

Tabla 13. Presupuesto ejecución por contrata y base licitación.

El presupuesto del proyecto es de un total de:

DOS MIL SETECIENTOS NOVENTA Y SEIS EUROS CON NOVENTA Y CINCO

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

TRABAJO FIN DE GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

DESARROLLO E IMPLEMENTACIÓN DE LA AUTOMATIZACIÓN PARA UN SISTEMA DE MECANIZADO DE PIEZAS CON ROBOT MANIPULADOR MEDIANTE COMUNICACIONES DISTRIBUIDAS CON SERVIDOR OPC Y SUPERVISIÓN SCADA

DOCUMENTO Nº 4 ANEXO I. PROGRAMACIÓN

AUTOR: ABEL DOMÍNGUEZ GONZÁLEZ

TUTOR: RAÚL SIMARRO FERNÁNDEZ

Curso Académico 2016-17

Índice

Tabla de figuras

1. Introducción

En este documento se adjunta toda la programación realizada en cada autómata mediante *SoMachine* para la automatización del sistema de mecanizado y brazo manipulador.

2. Línea indexada de mecanizado *FischerTechnik*

2.1 Programación del sistema físico

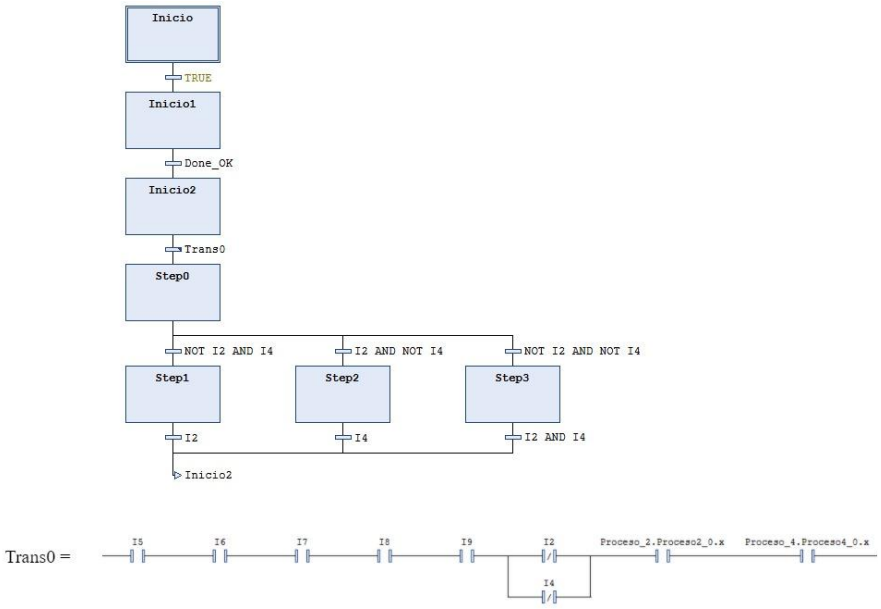


Figura 1. SFC y transición de Inicio_Sistema.

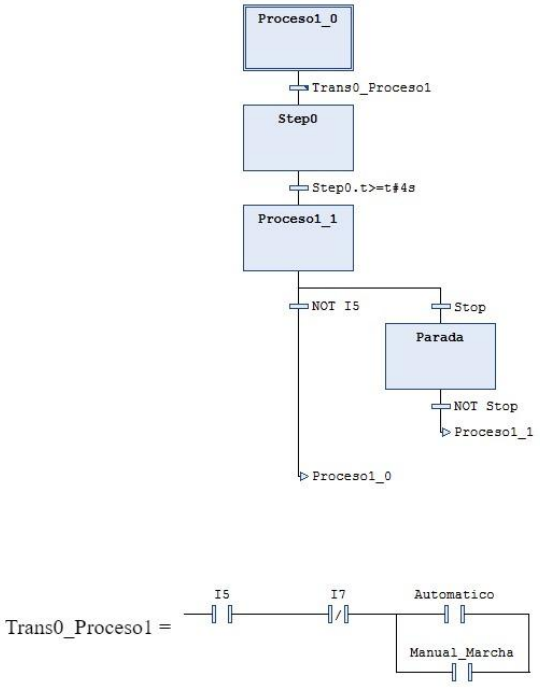


Figura 2. SFC y transición de Proceso_1.

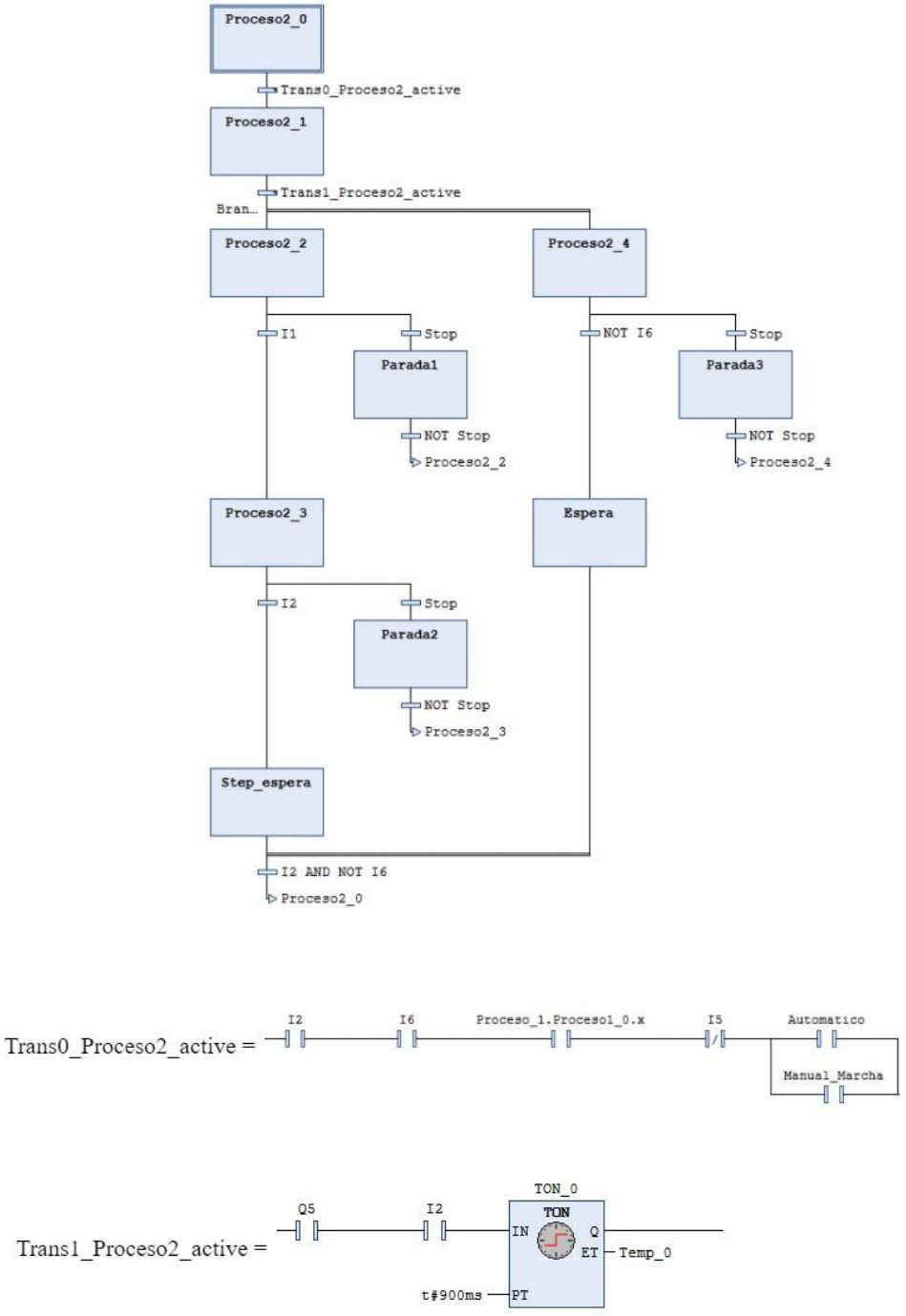


Figura 3. SFC y transiciones de Proceso_2.

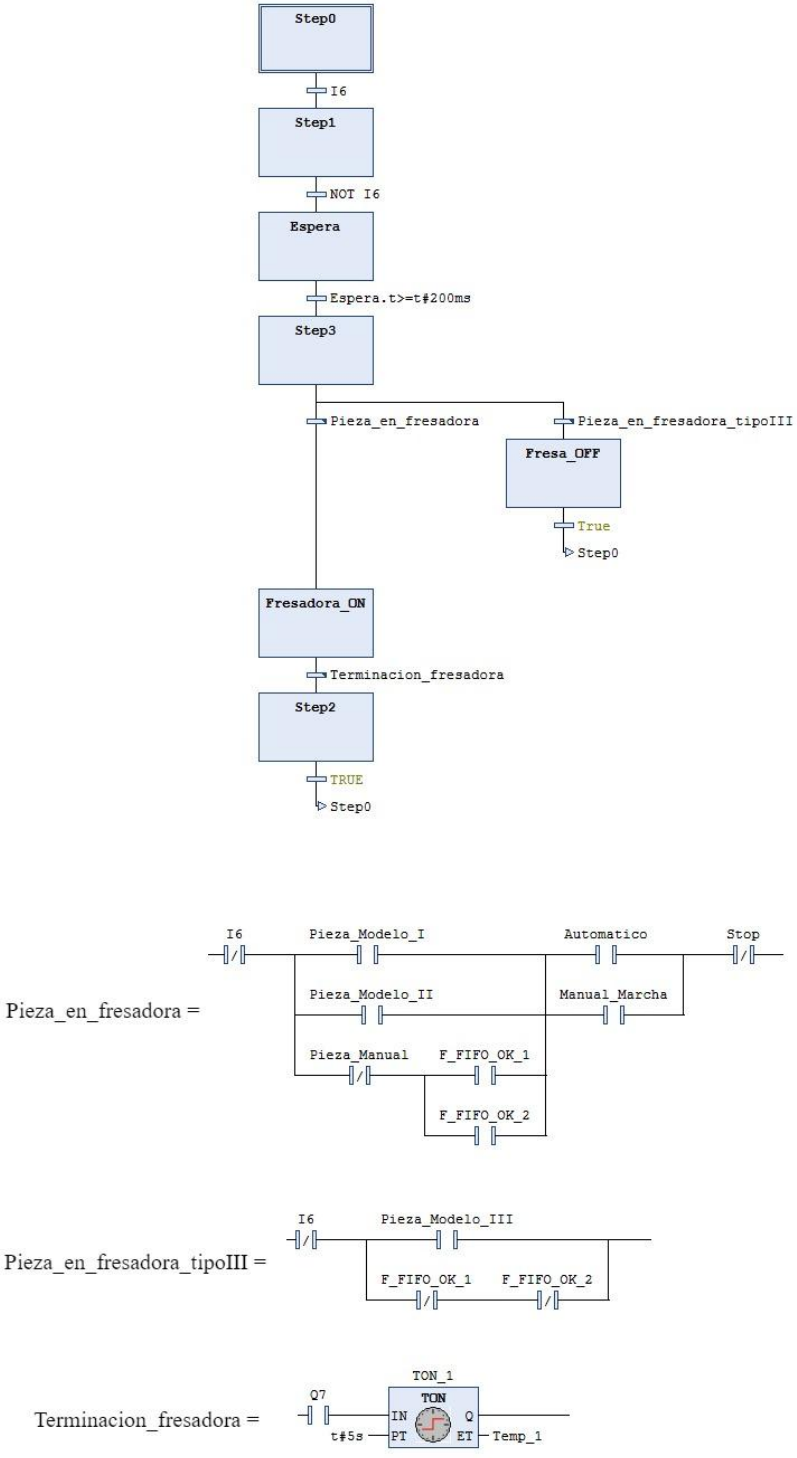


Figura 4. SFC y transiciones de Proceso_2_FRESADORA.

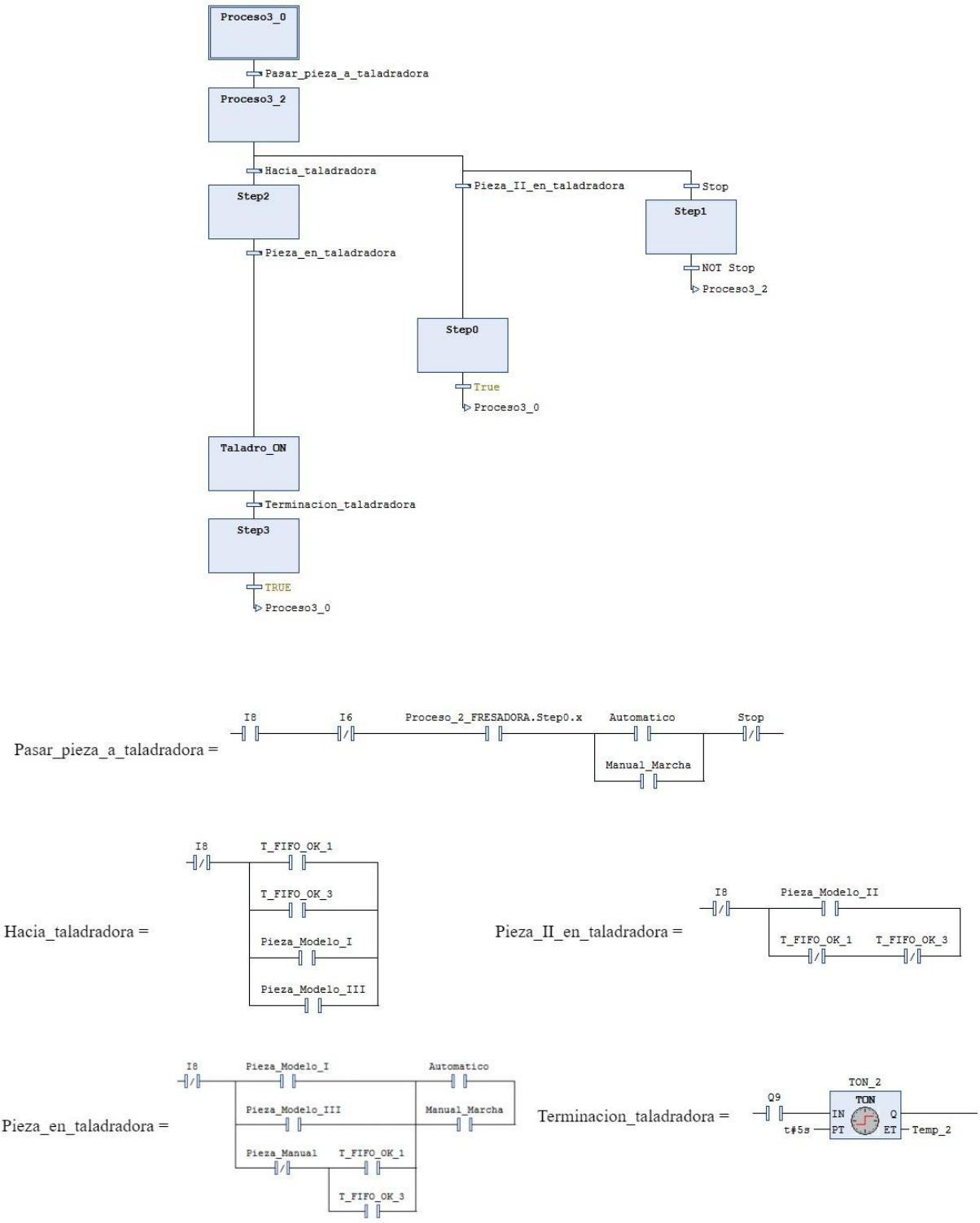


Figura 5. SFC y transición de Proceso_3.

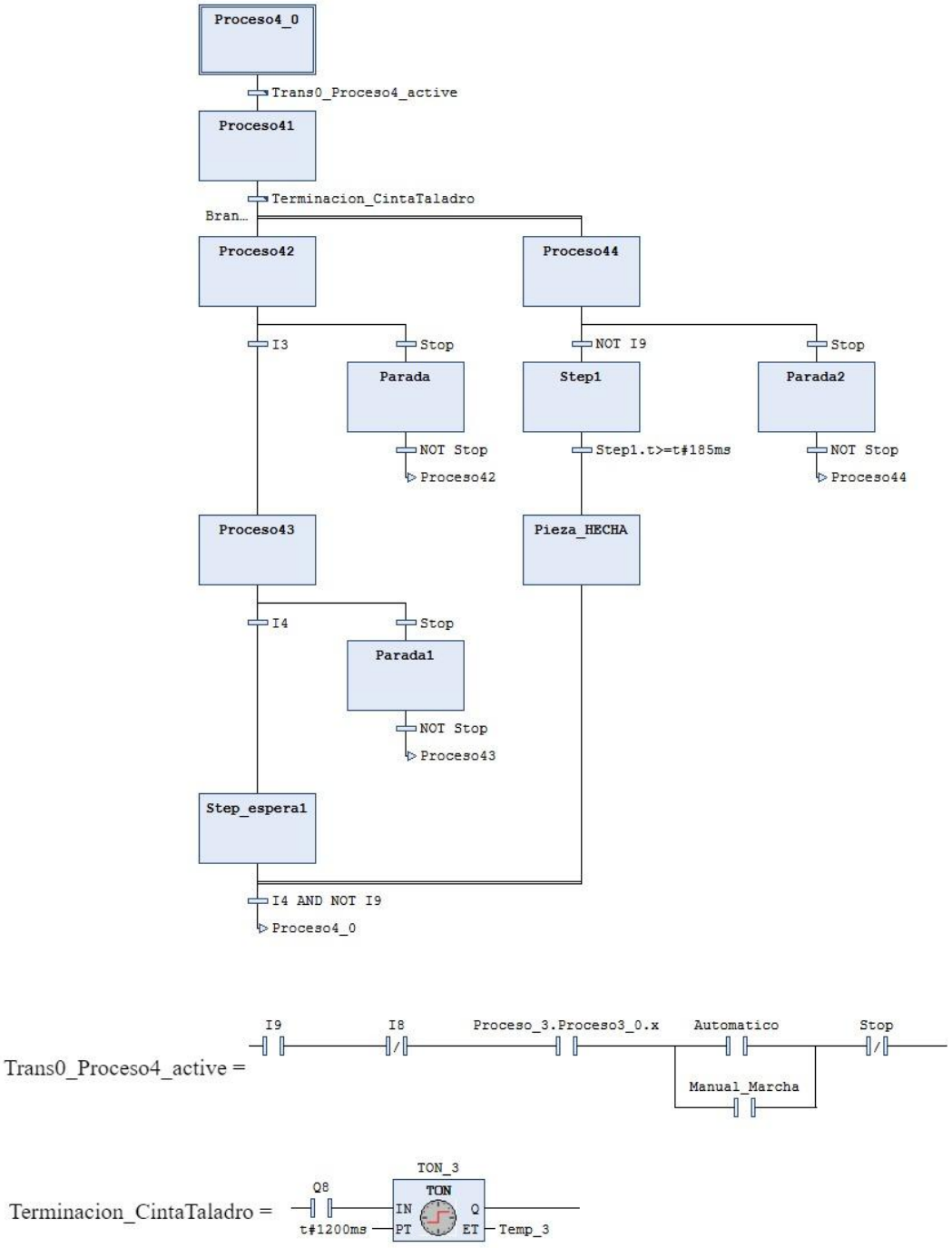


Figura 6. SFC y transiciones de Proceso_4.

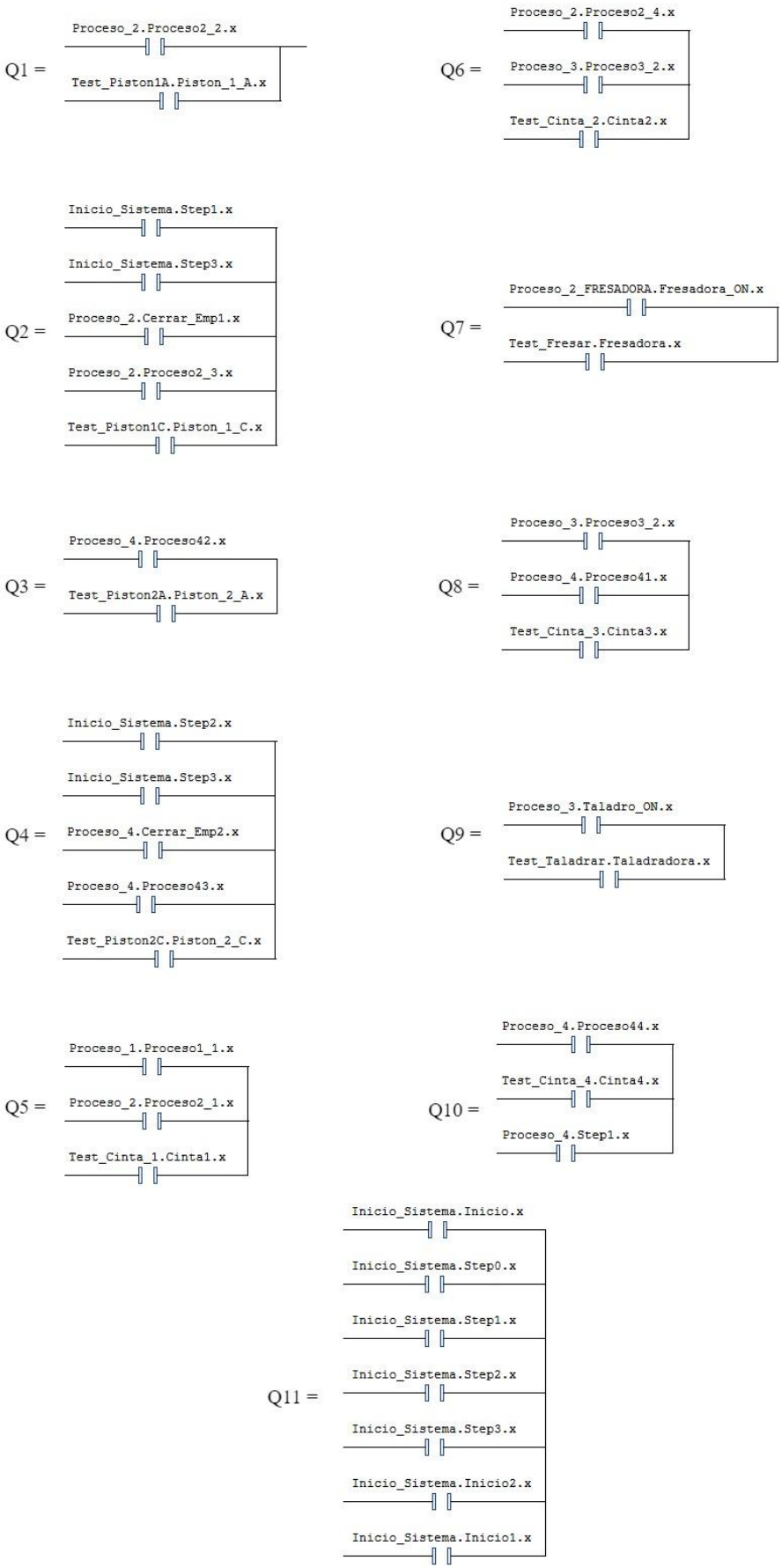


Figura 7. Diagrama de bloques de las etapas que activan las salidas.

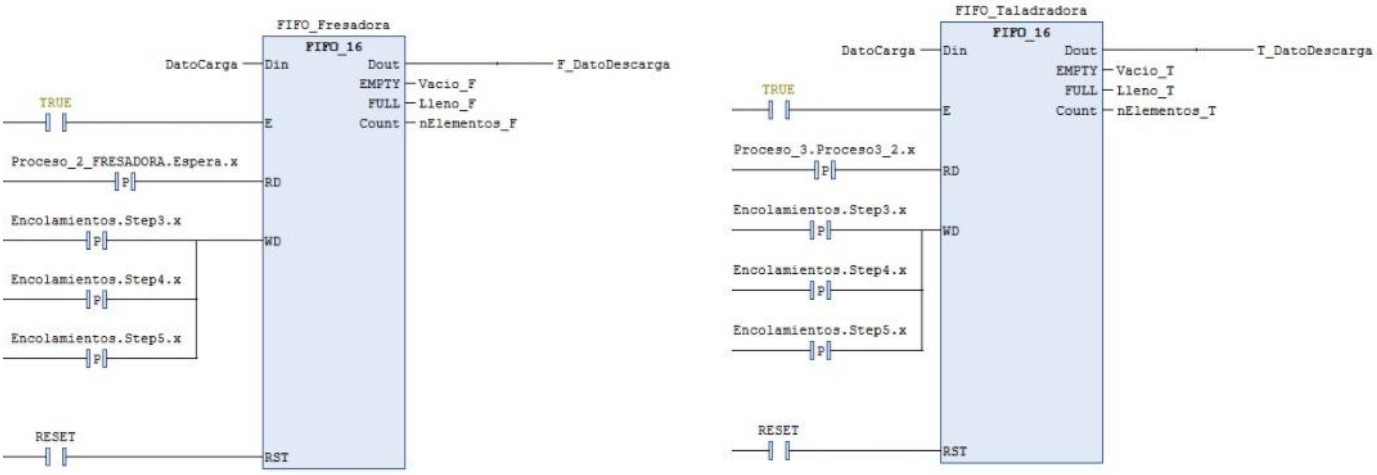


Figura 8. Diagrama de bloques de las funciones FIFO de la fresadora y taladradora.

```
//****** FRESADORA **********//
IF F_DatoDescarga = 1 THEN
    F_FIFO_OK_1 := TRUE;
    F_FIFO_OK_2 := FALSE;

ELSIF F_DatoDescarga = 2 THEN

    F_FIFO_OK_2 := TRUE;
    F_FIFO_OK_1 := FALSE;

ELSIF F_DatoDescarga = 0 OR  F_DatoDescarga = 3 THEN
    F_FIFO_OK_1 := FALSE;
    F_FIFO_OK_2 := FALSE;
END_IF


//****** TALADRADORA *********//

IF T_DatoDescarga = 1 THEN
    T_FIFO_OK_1 := TRUE;
    T_FIFO_OK_3 := FALSE;
ELSIF T_DatoDescarga = 3 THEN

    T_FIFO_OK_3 := TRUE;
    T_FIFO_OK_1 := FALSE;
ELSIF T_DatoDescarga = 0 OR  T_DatoDescarga = 2 OR T_DatoDescarga > 3 THEN
    T_FIFO_OK_1 := FALSE;
    T_FIFO_OK_3 := FALSE;
END_IF
```

Figura 9. Texto estructurado de *EstadoFIFO*. Determinación del tipo de pieza a fabricar.

```
IF RST THEN
    pw := pr;
    FULL := FALSE;
    EMPTY := TRUE;
    Dout := 0;
ELSIF E THEN
    IF NOT EMPTY AND RD THEN
        Dout := fifo[pr];
        pr := INC1(pr,n);
        EMPTY := pr = pw;
        FULL := FALSE;
    END_IF;
    IF NOT FULL AND WD THEN
        fifo[pw] := Din;
        pw := INC1(pw,n);
        FULL := pw = pr;
        EMPTY := FALSE;
    END_IF;
END_IF;

IF pw>=pr THEN
    Count :=pw-pr;
    IF FULL THEN
        Count := n;
    END_IF;
ELSE
    Count :=pw+n-pr;
END_IF;


IF X >= N - 1 THEN
    INC1 := 0;
ELSE
    INC1 := X + 1;
END_IF
```

Figura 10. Funciones internas del FIFO. Necesario tener dicho programa para que las FIFO funcionen.

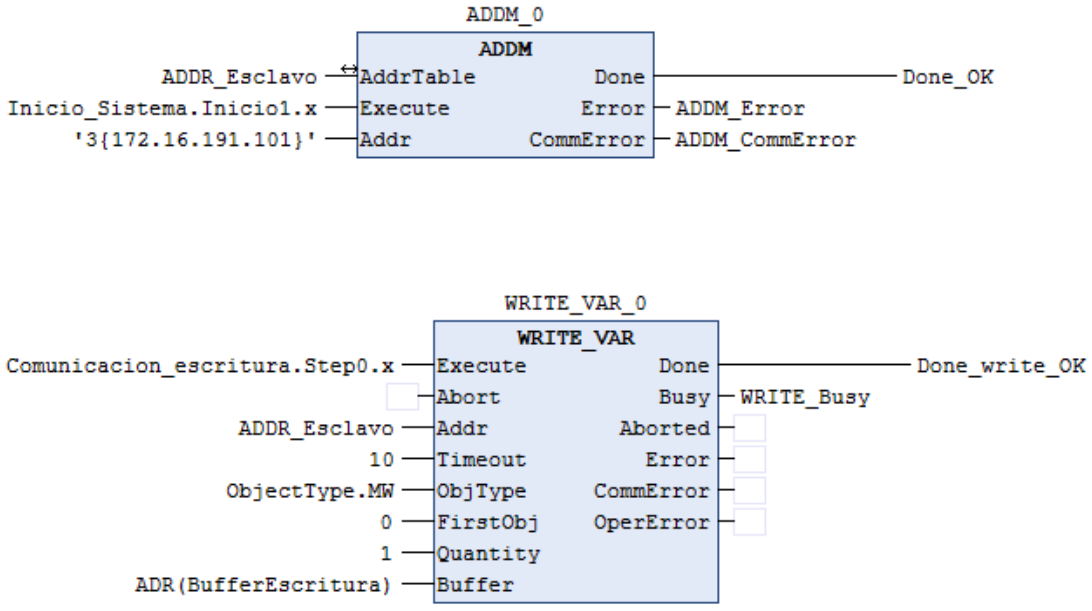


Figura 11. Diagrama de bloques de ComunicacionesMaestro.

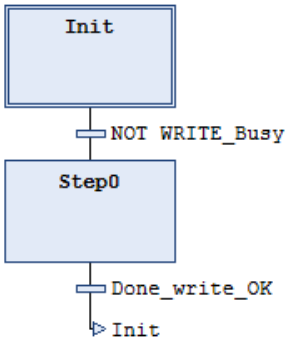


Figura 12. SFC de Comunicación_escritura.

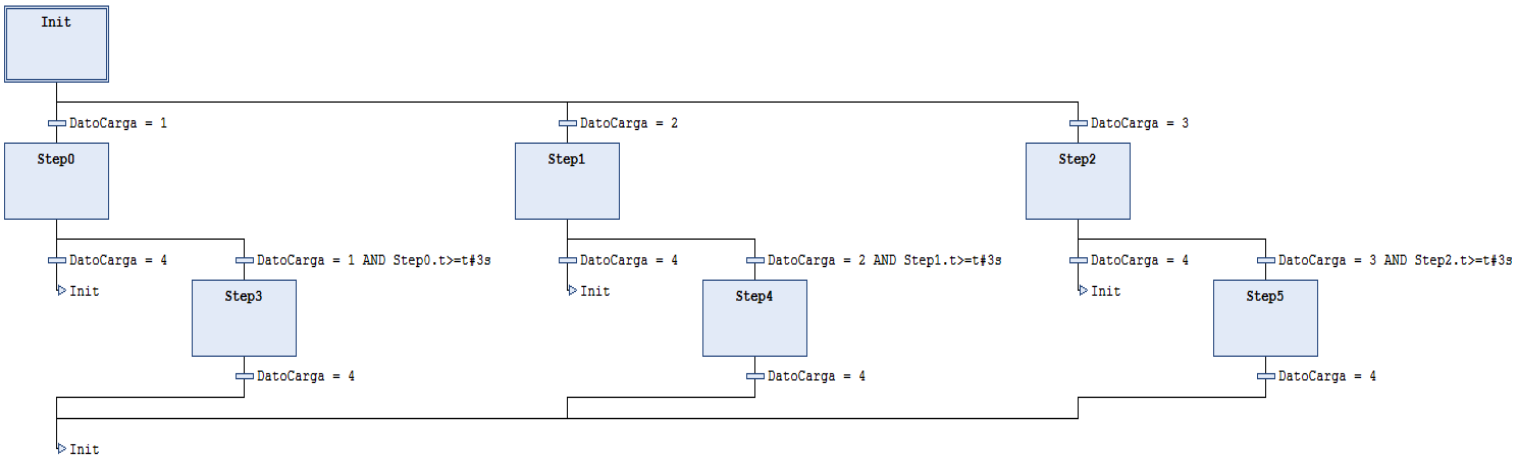


Figura 13. SFC de Encolamientos. Determinación de cuando encolar el valor de DatoCarga.

```
IF DatoCarga = 1 THEN     //Pieza negra
    Negras := Negras + 1;
END_IF

IF DatoCarga = 2 THEN     //Pieza azul
    Azules := Azules + 1;
END_IF

IF DatoCarga = 3 THEN     //Pieza roja
    Rojas := Rojas + 1;
END_IF
```

Figura 14. ST de Piezas_entrada. Determinación del número de piezas que se han insertado en el sistema.

```
//Piezas negras

Negras_enProceso := Negras - Piezas_I_terminada;

//Piezas azules

Azules_enProceso := Azules - Piezas_II_terminada;

//Piezas rojas

Rojas_enProceso := Rojas - Piezas_III_terminada;
```

Figura 15. ST de Piezas_enProceso. Determinación del las piezas en proceso.

2.2 Programación de la interfaz de registro y acceso de usuario

```
IF a = 0 THEN
    a := 1;
    Registro := TRUE;
    No_Registro := FALSE;
    clave := 'A1B2C3';
    Usuario := '';
    Registro_usuario := '  ';
    Password := '';
    Registro_password := '  ';

END_IF
```

Figura 16. ST de *Variables*_VisuRegistro. Inicialización de las variables.

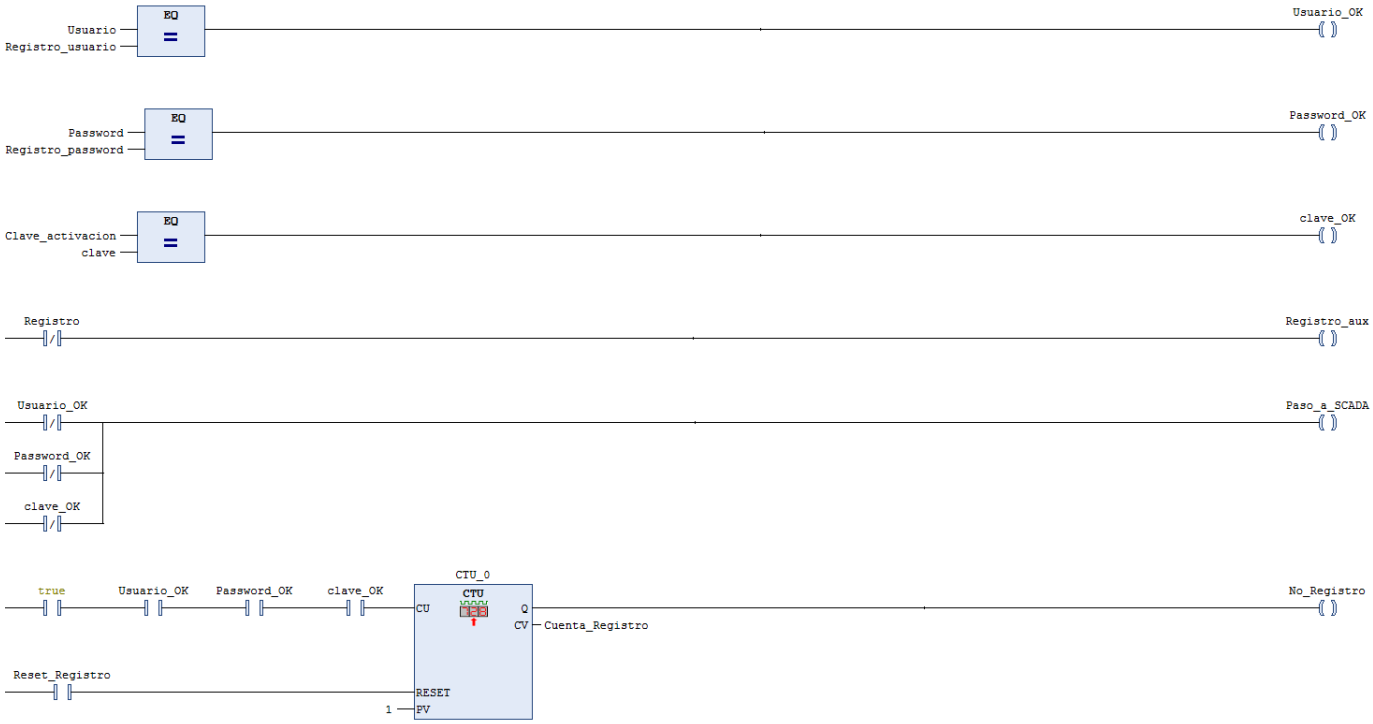


Figura 17. Diagrama de contactos de *Registrar*.

2.3 Programación de la interfaz de fabricación

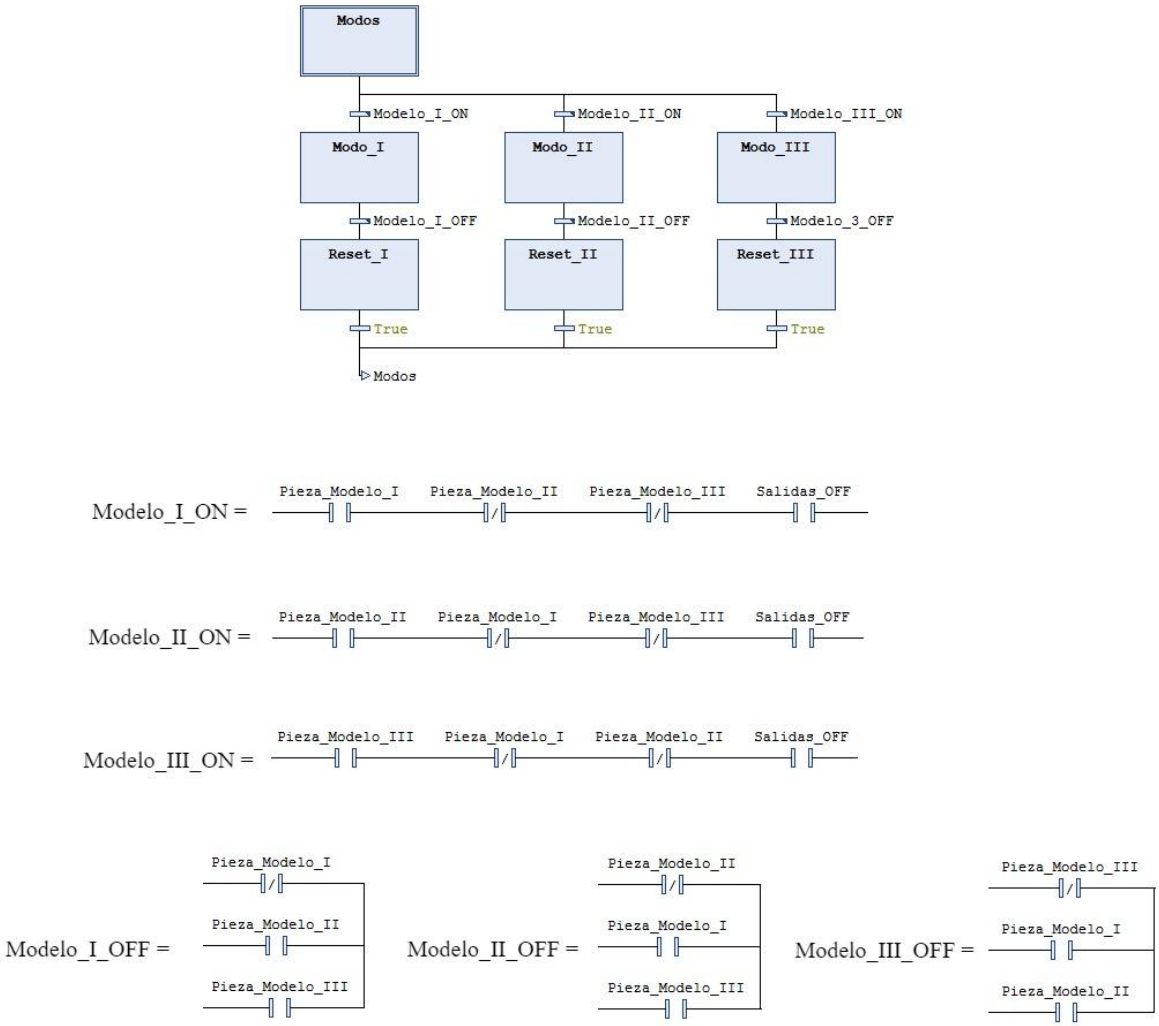


Figura 18. SFC y transiciones de Modos_Fabricación.

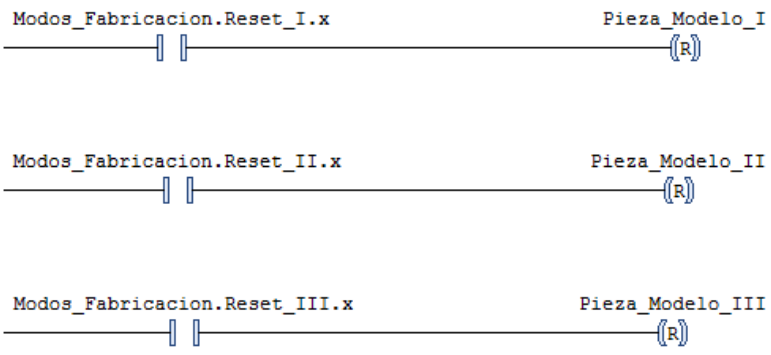


Figura 19. Diagrama de contacto de *Variables_piezas*.

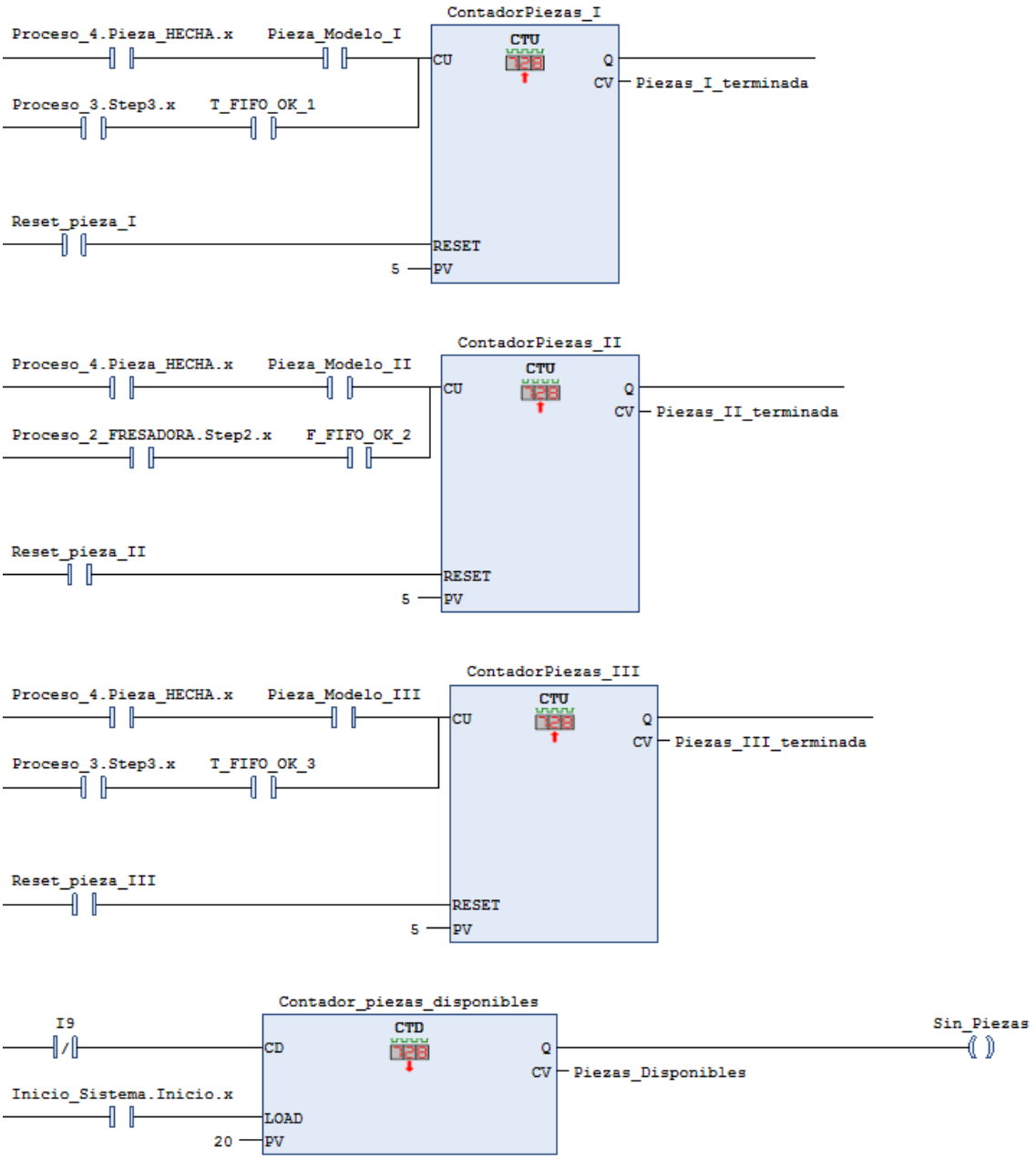


Figura 21. Diagrama de contactos de *Contadores*.

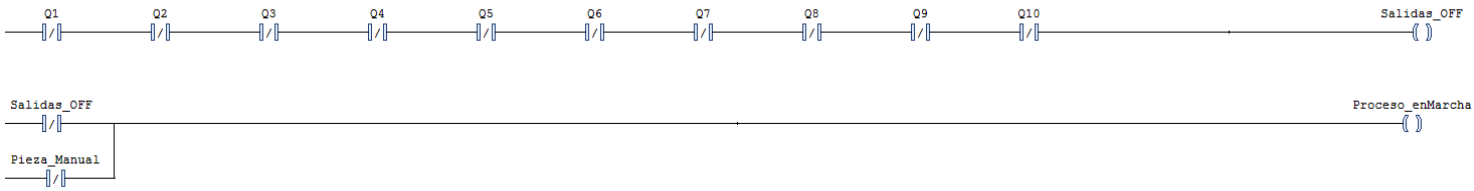


Figura 20. Diagrama de bloques de *Salidas_OFF* y *Proceso_enMarcha* ubicado en Salidas_sistema. Determina si el sistema está en marcha o en reposo.

2.4 Programación de la interfaz del proceso

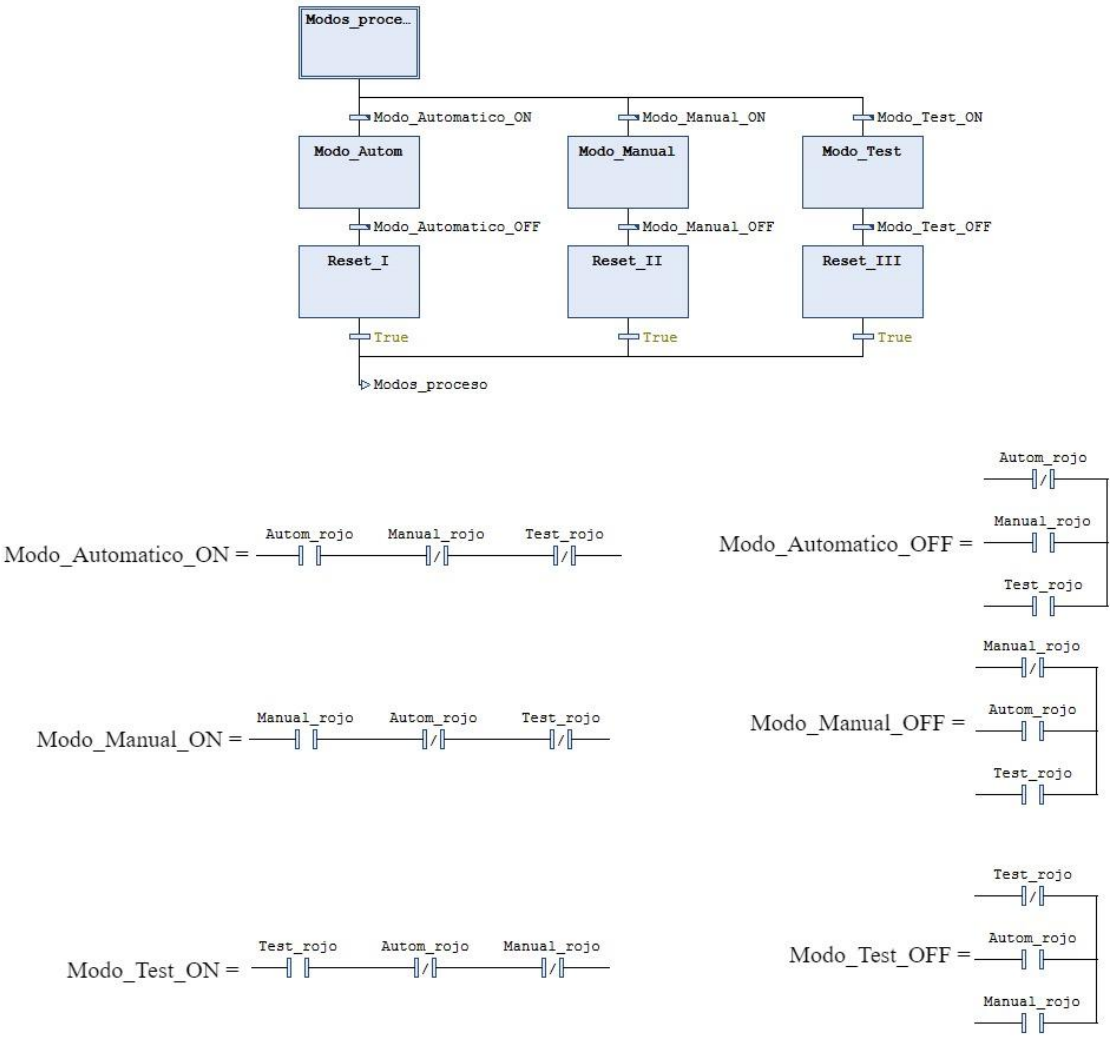


Figura 22. SFC y transiciones de *Modos_proceso.*

```
// Mover el empujador 1 hacia adelante y hacia detrás
IF I1 = TRUE THEN

    Mover_piston1 := 146;

END_IF

IF I2 = TRUE THEN

    Mover_piston1 := 0;

END_IF

// Mover el empujador 2 hacia adelante y hacia detrás

IF I3 = TRUE THEN

    Mover_piston2 := 135;

END_IF

IF I4 = TRUE THEN

    Mover_piston2 := 0;

END_IF
```

Figura 23. ST de *Movimientos_empujadores.* Permiten que los empujadores de la visualización se muevan.

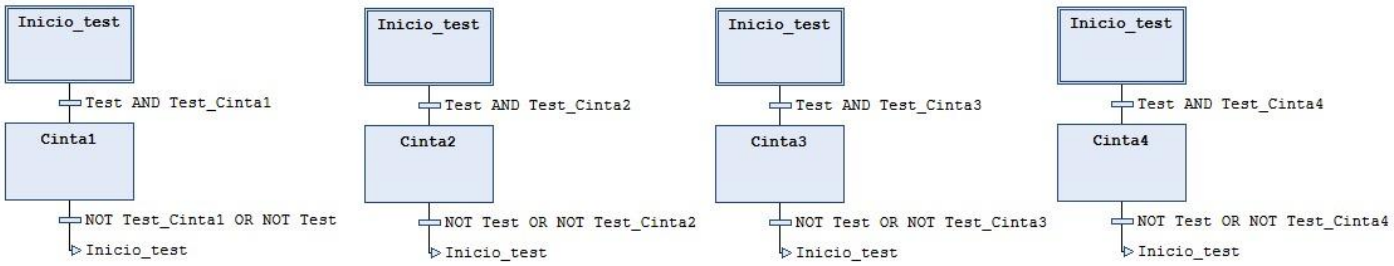


Figura 24. SFCs de los test de las cintas.

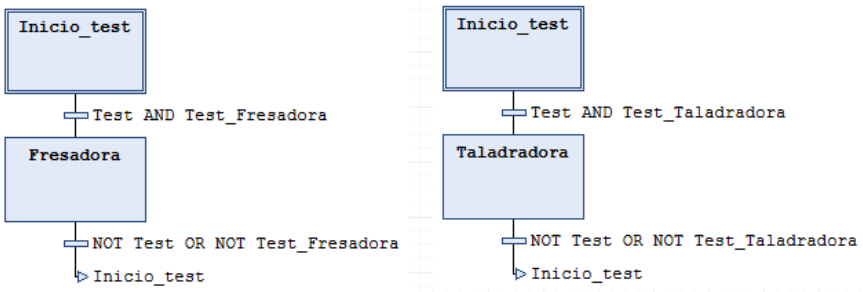


Figura 25. SFCs de los test de los puestos de mecanizado.

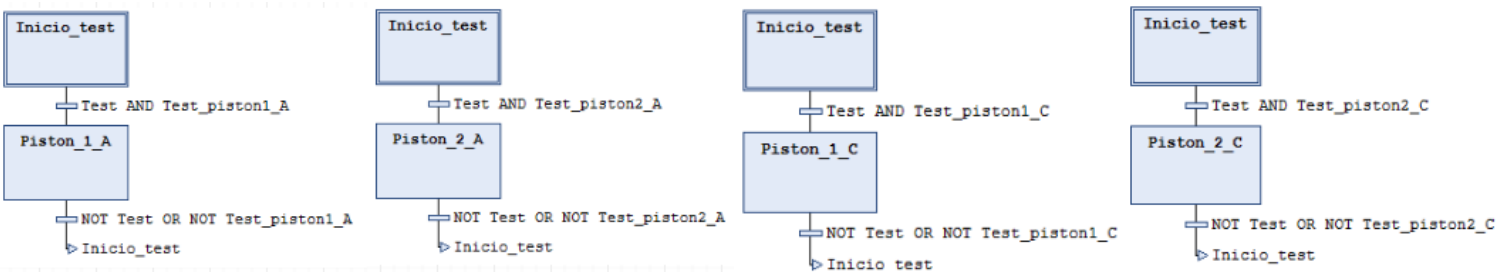


Figura 26. SFCs de los test de ambos empujadores (pistones).

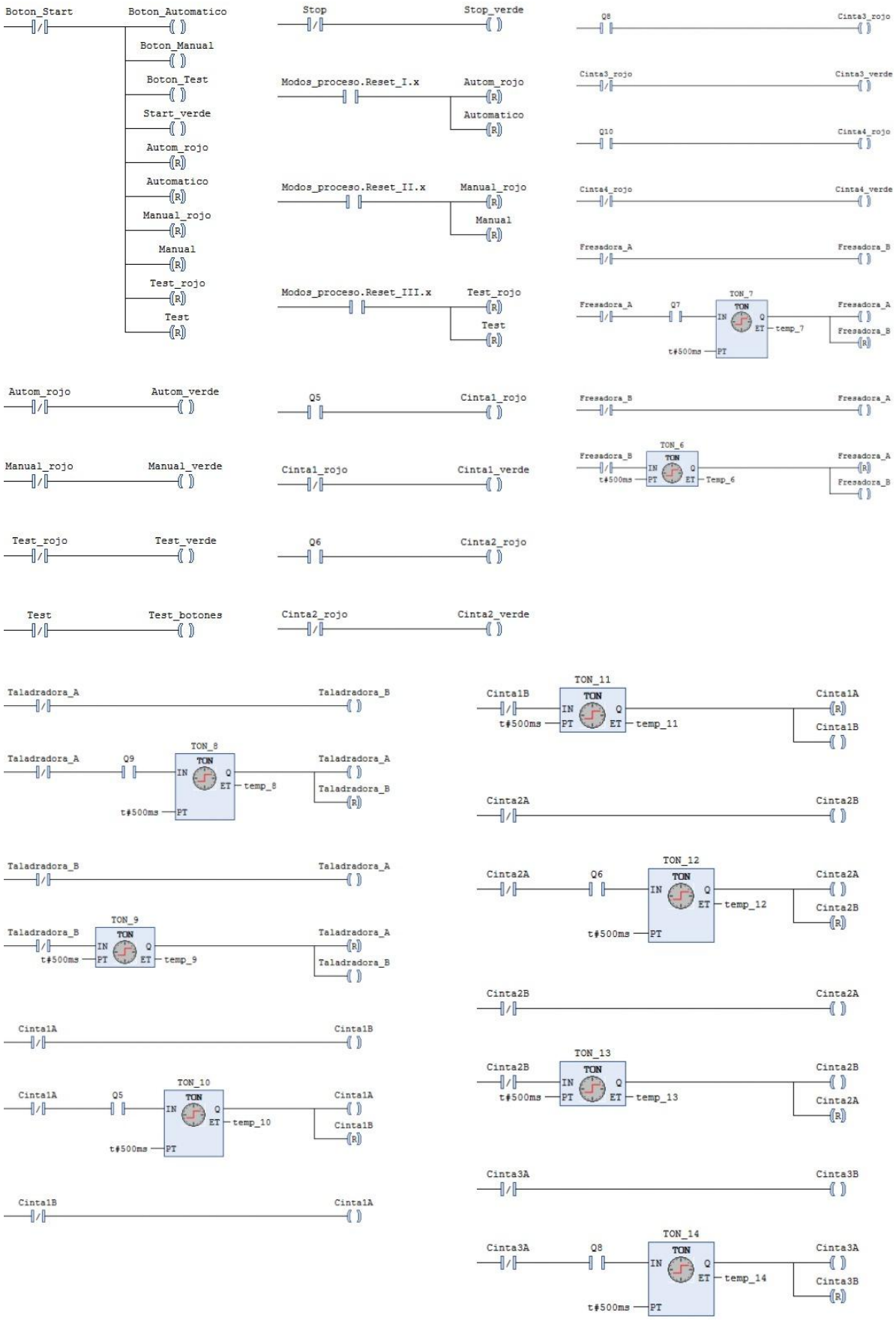


Figura 27. Diagrama de contacto de *Variables_Visu_proceso.* 1/2.

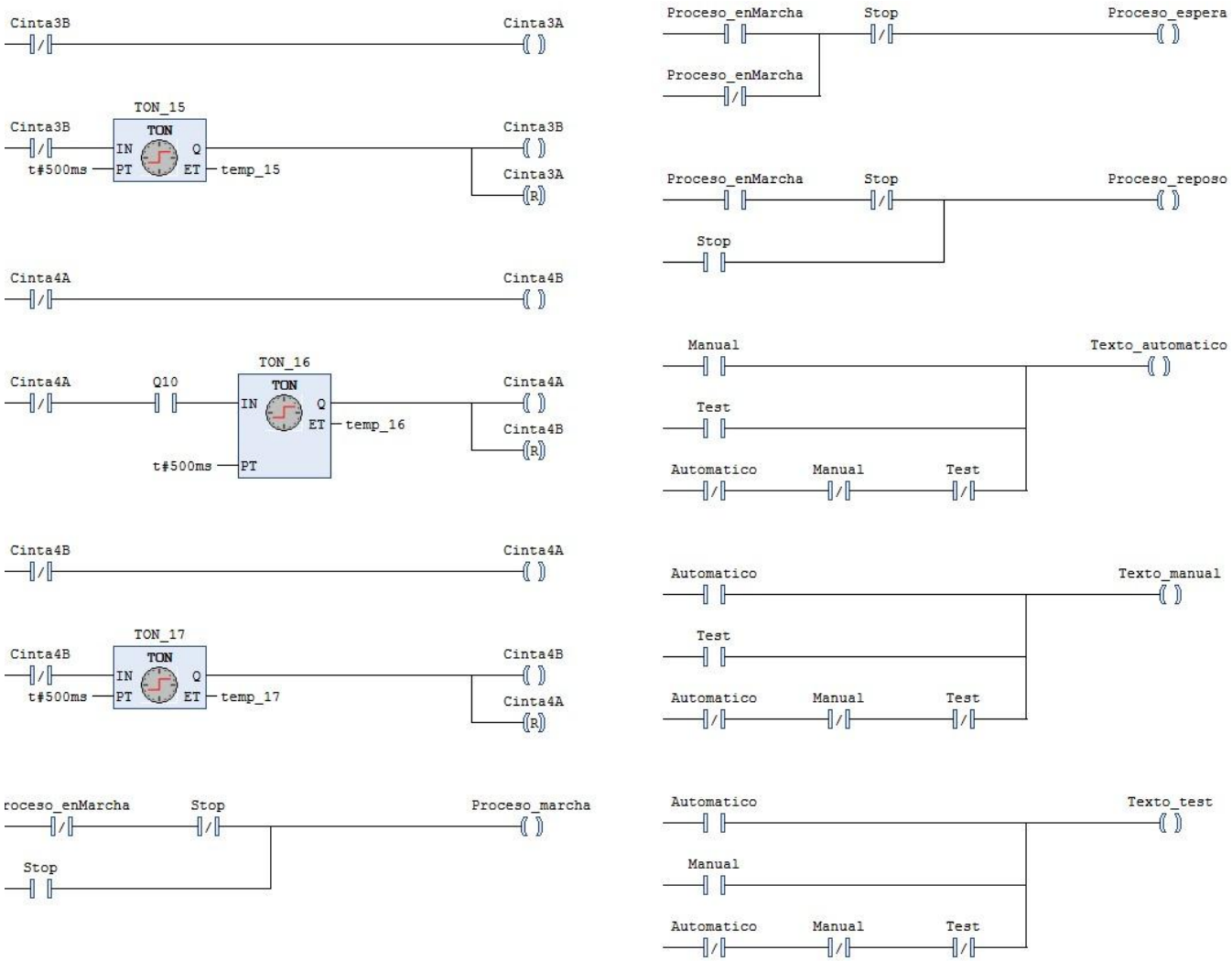
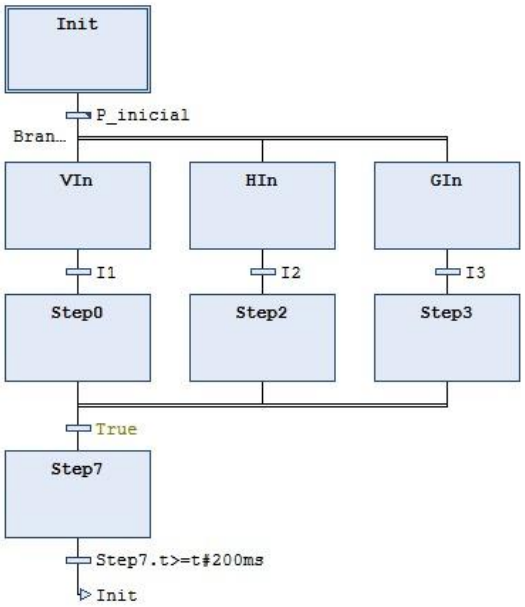


Figura 28. Diagramas de contacto *Variables_Visu_proceso.* 2/2

3. Robot manipulador de vacío *FischerTechnik*

3.1 Programación del sistema físico



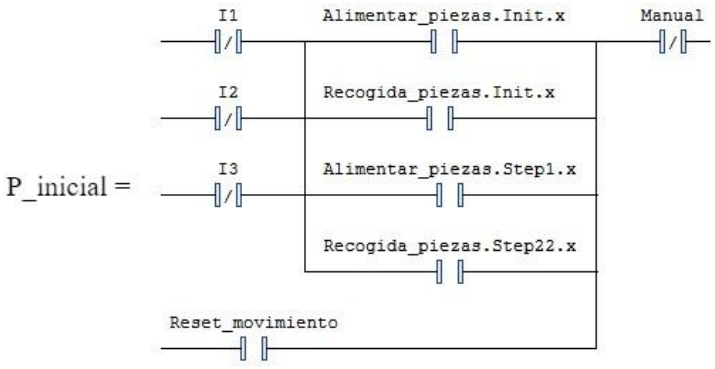


Figura 29. SFC y transición de *Punto_Inicial.*

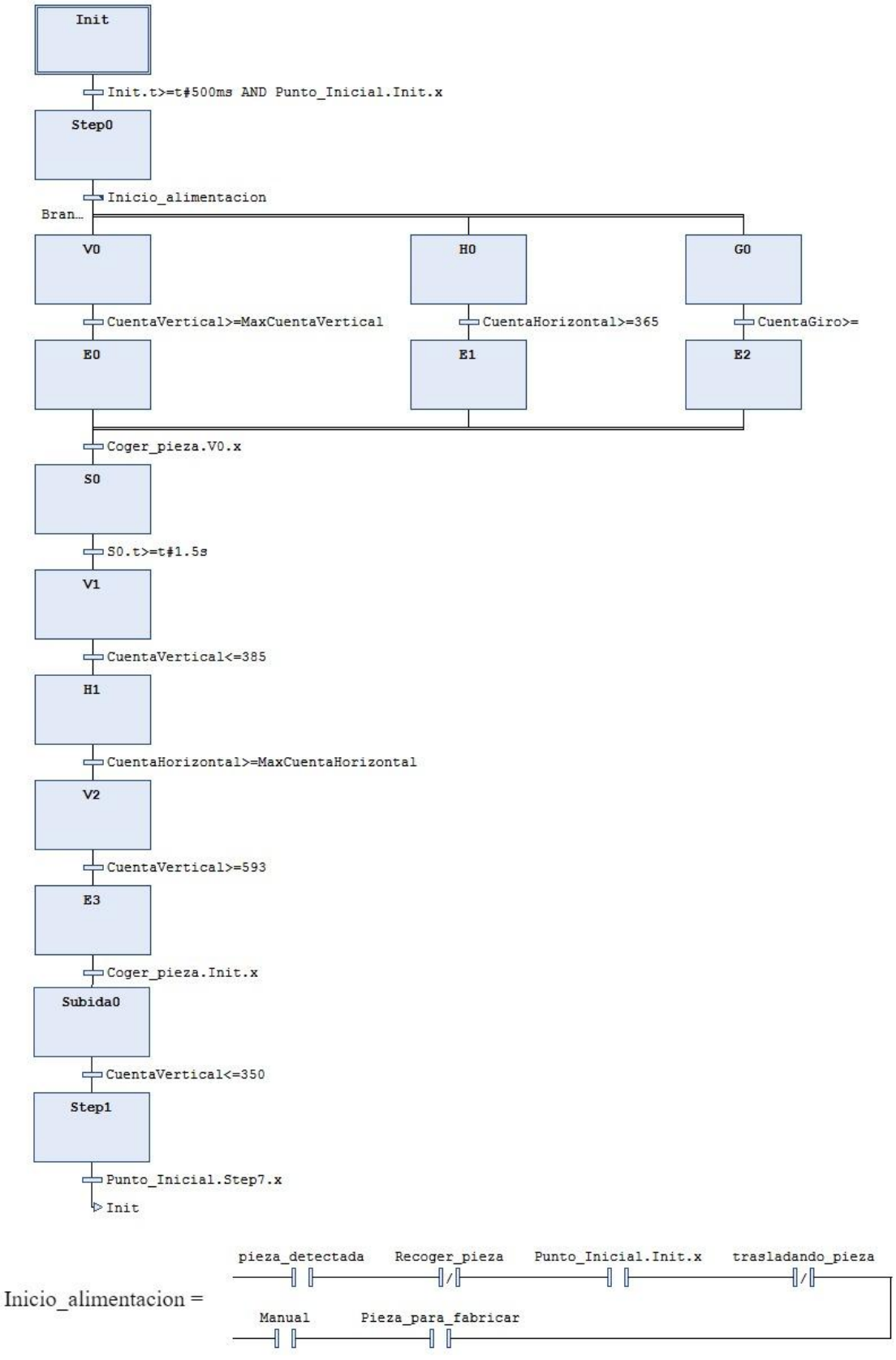


Figura 30. SFC y transición de *Inicio_alimentacion*.

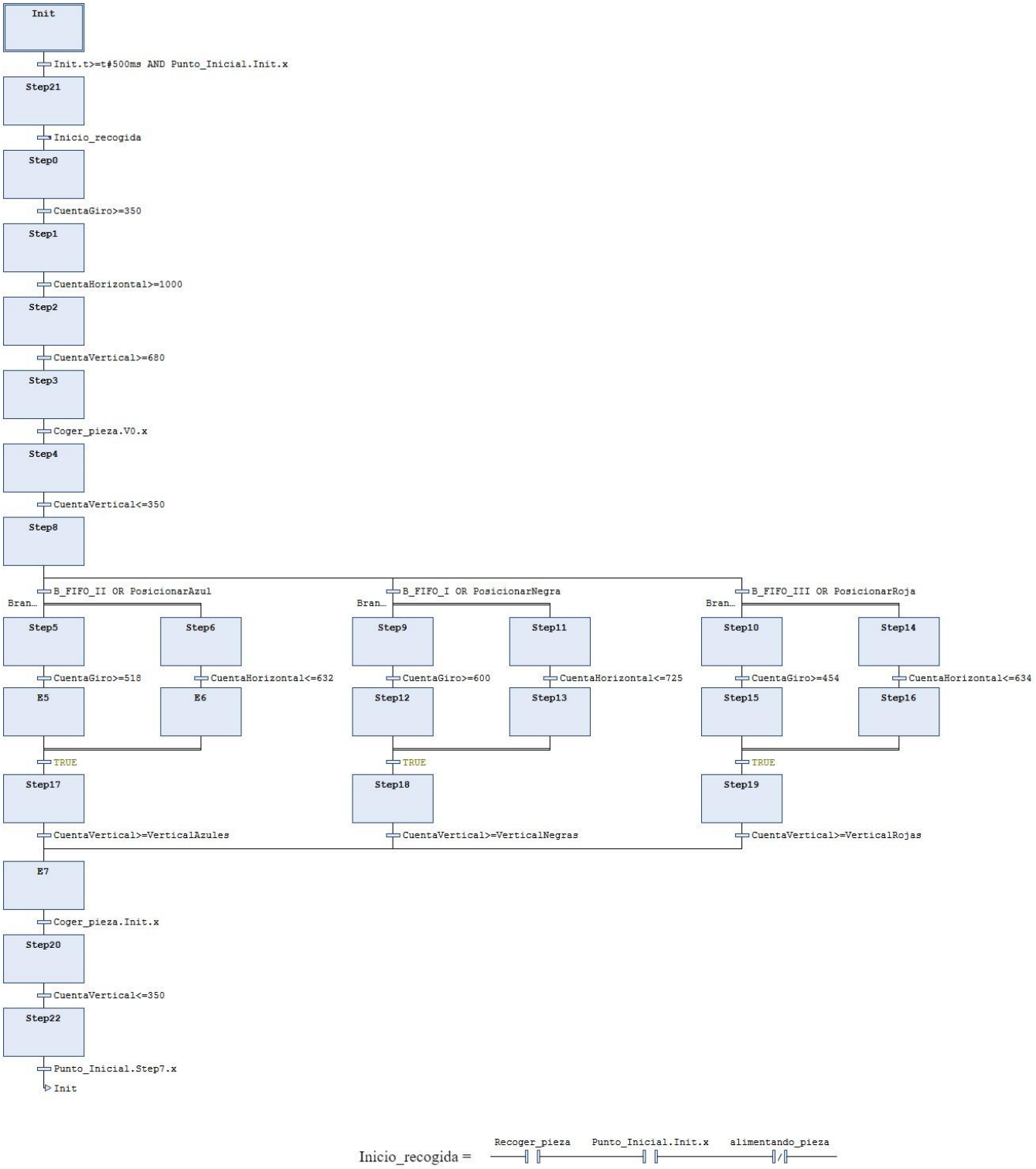


Figura 31. SFC y transición de *Inicio_recogida*.

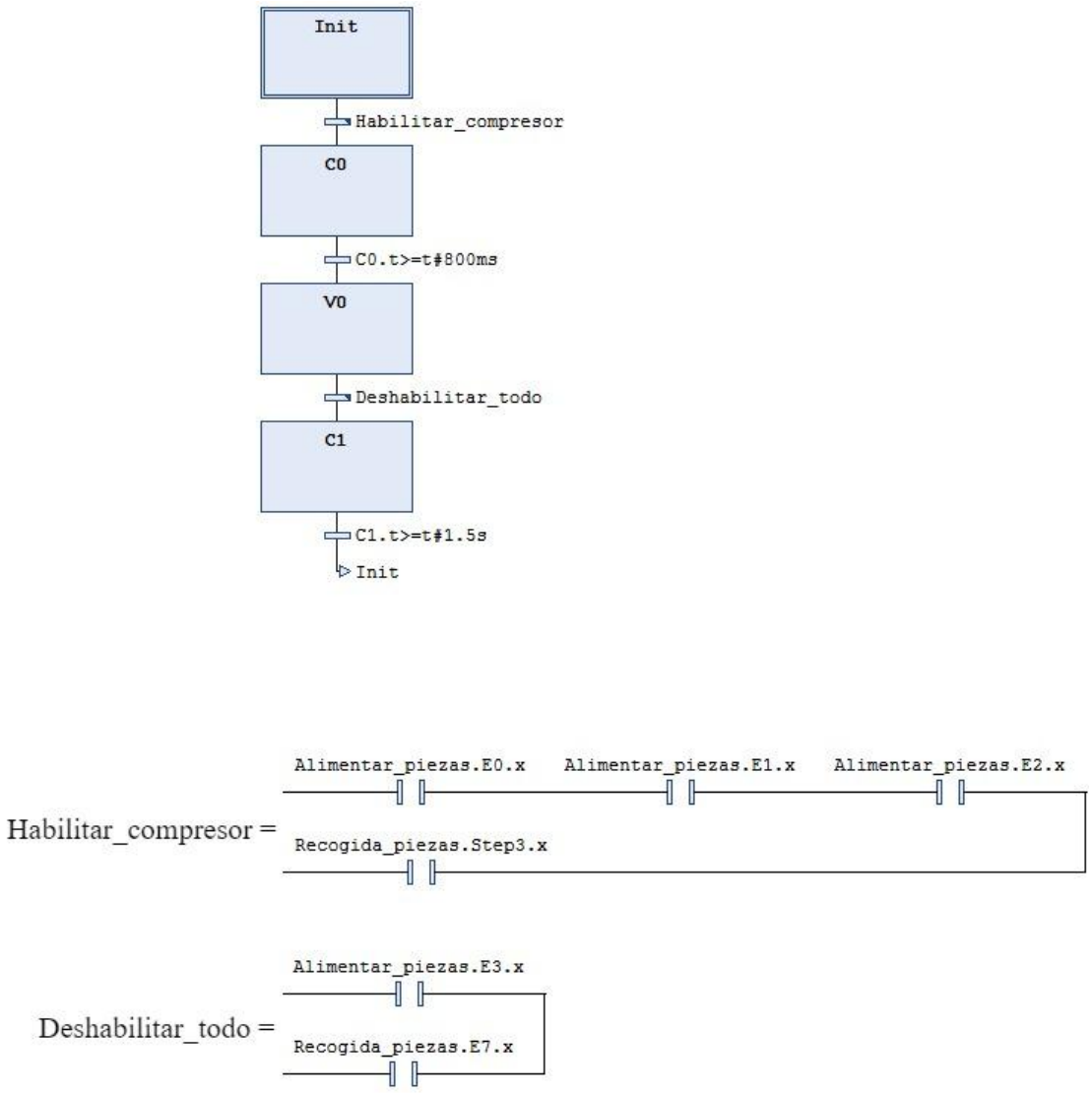


Figura 32. SFC y transiciones de *Coger_pieza*.

```
IF I3=TRUE THEN
    CuentaGiro:=0;
ELSE
    IF Q6=TRUE THEN
        CuentaGiro:=CuentaGiro+1;
    END_IF;

    IF Q5=TRUE THEN
        CuentaGiro:=CuentaGiro-1;
    END_IF;
END_IF;
```

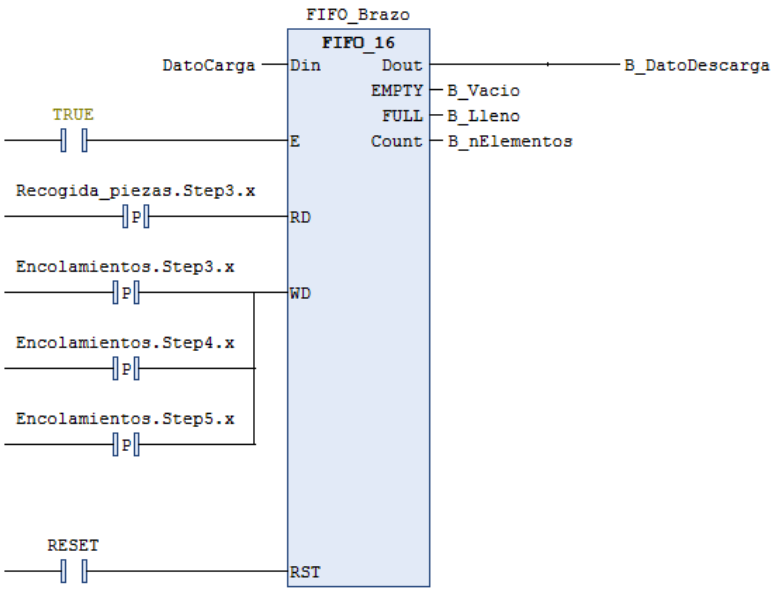
```
IF I2=TRUE THEN
    CuentaHorizontal:=0;
ELSE
    IF Q4=TRUE THEN
        CuentaHorizontal:=CuentaHorizontal+1;
    END_IF;

    IF Q3=TRUE THEN
        CuentaHorizontal:=CuentaHorizontal-1;
    END_IF;
END_IF;
```

```
IF I1=TRUE THEN
    CuentaVertical:=0;
ELSE
    IF Q2=TRUE THEN
        CuentaVertical:=CuentaVertical+1;
    END_IF;

    IF Q1=TRUE THEN
        CuentaVertical:=CuentaVertical-1;
    END_IF;
END_IF;
```

Figura 33. ST de las cuentas de los pulsos de los *encoders*. *ContadorGiro,ContadorHorizontal y ContadorVertical*.




```
IF B_DatoDescarga = 1 THEN
    B_FIFO_I := TRUE;
    B_FIFO_II := FALSE;
    B_FIFO_III := FALSE;

ELSIF B_DatoDescarga = 2 THEN
    B_FIFO_I := FALSE;
    B_FIFO_II := TRUE;
    B_FIFO_III := FALSE;

ELSIF B_DatoDescarga = 3 THEN
    B_FIFO_I := FALSE;
    B_FIFO_II := FALSE;
    B_FIFO_III := TRUE;

ELSIF B_DatoDescarga = 0 OR B_DatoDescarga > 3 THEN
    B_FIFO_I := FALSE;
    B_FIFO_II := FALSE;
    B_FIFO_III := FALSE;

END_IF
```

Figura 34. Diagrama de contacto de la función FIFO y ST del estado del tipo de pieza, dependiente del valor de la FIFO.

```
IF RST THEN
    pw := pr;
    FULL := FALSE;
    EMPTY := TRUE;
    Dout := 0;
ELSIF E THEN
    IF NOT EMPTY AND RD THEN
        Dout := fifo[pr];
        pr := INC1(pr,n);
        EMPTY := pr = pw;
        FULL := FALSE;
    END_IF;
    IF NOT FULL AND WD THEN
        fifo[pw] := Din;
        pw := INC1(pw,n);
        FULL := pw = pr;
        EMPTY := FALSE;
    END_IF;
END_IF;

IF pw>=pr THEN
    Count :=pw-pr;
    IF FULL THEN
        Count := n;
    END_IF;
ELSE
    Count :=pw+n-pr;
END_IF;

IF X >= N - 1 THEN
    INC1 := 0;
ELSE
    INC1 := X + 1;
END_IF;
```

Figura 35. Función desarrollada para poder utilizar la función FIFO.

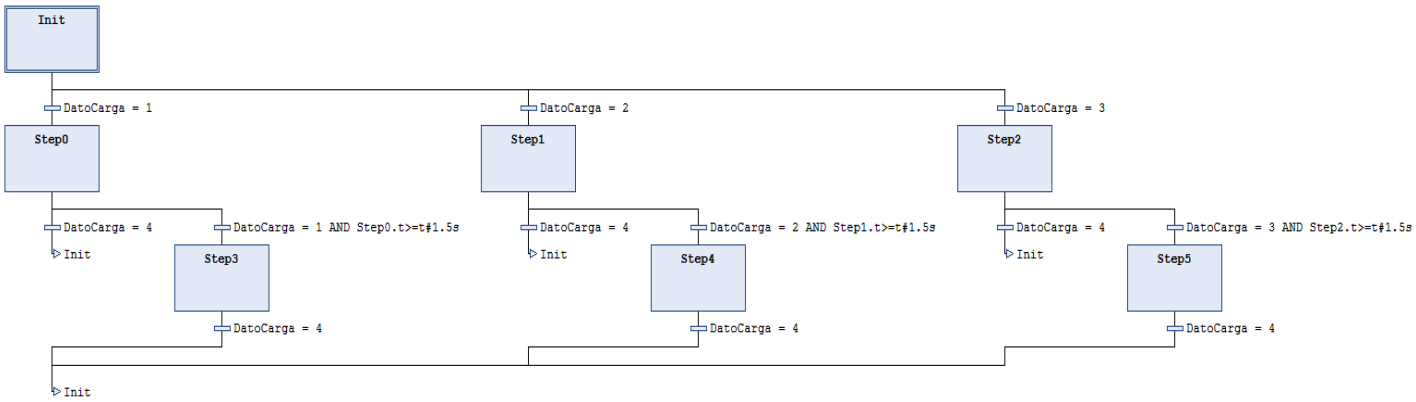


Figura 36. SFC de *Encolamientos*. Se encola el valor de *DatoCarga* pasado 1.5 segundos.

```
IF B_DatoDescarga = 1 THEN  //La pieza a recoger es negra

    Negras := Negras + 1;
    VerticalNegras := 980 - 205*(Negras - 1);

END_IF

//Azules

IF B_DatoDescarga = 2 THEN  //La pieza a recoger es azul

    Azules := Azules + 1;
    VerticalAzules := 980 - 205*(Azules - 1);

END_IF

//Rojas

IF B_DatoDescarga = 3 THEN  //La pieza a recoger es roja

    Rojas := Rojas + 1;
    VerticalRojas := 980 - 205*(Rojas - 1);

END_IF

IF Negras >= 4 THEN
    Negras := 0;
END_IF

IF Azules >= 4 THEN
    Azules := 0;
END_IF

IF Rojas >= 4 THEN
    Rojas := 0;
END_IF
```

Figura 37. ST de *Tipo_Piezas.* Sirve para depositar las piezas en una columna (en función del tipo).

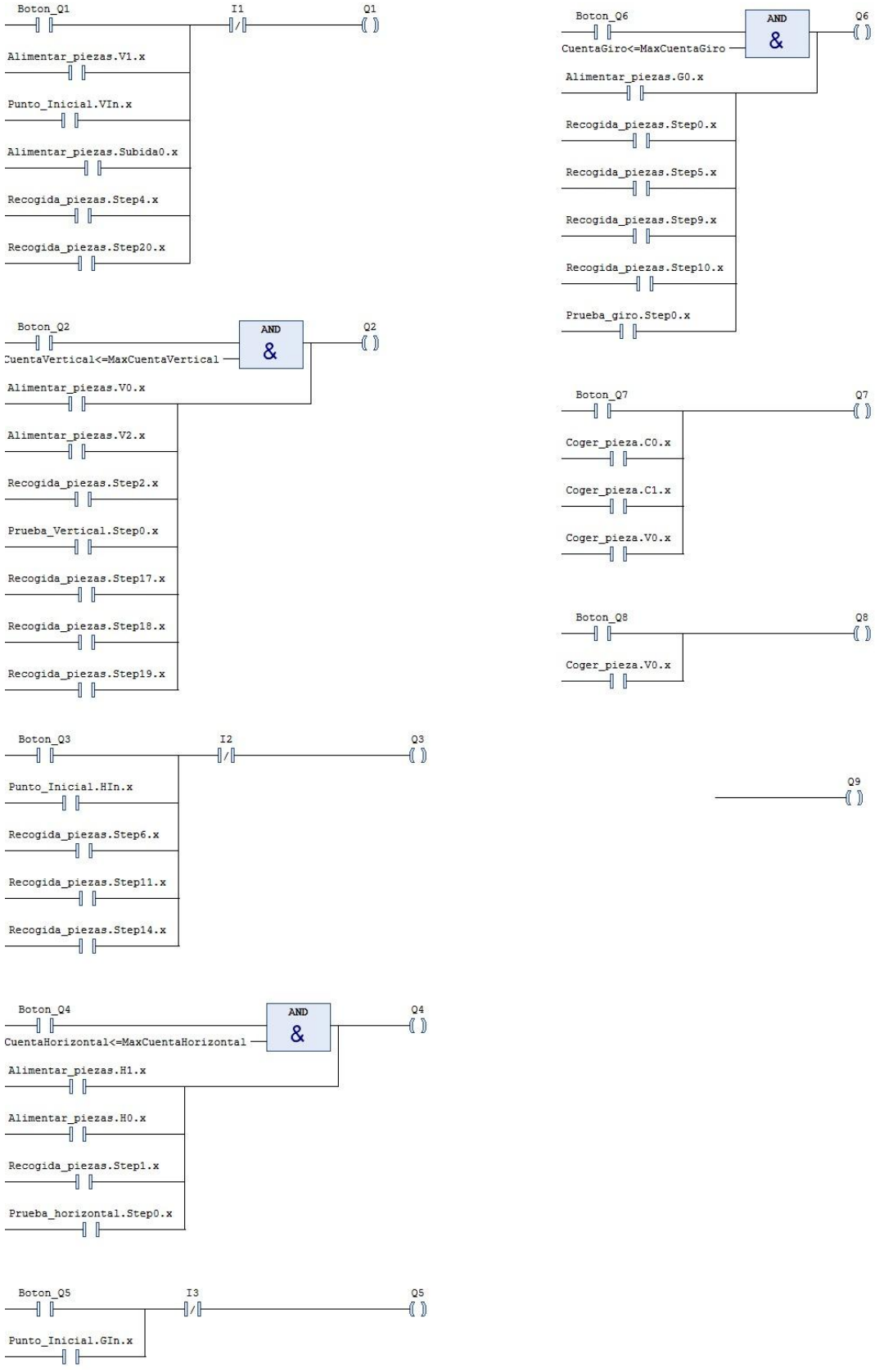


Figura 38. Diagrama de contacto de las salidas del sistema. Q9 siempre activa.

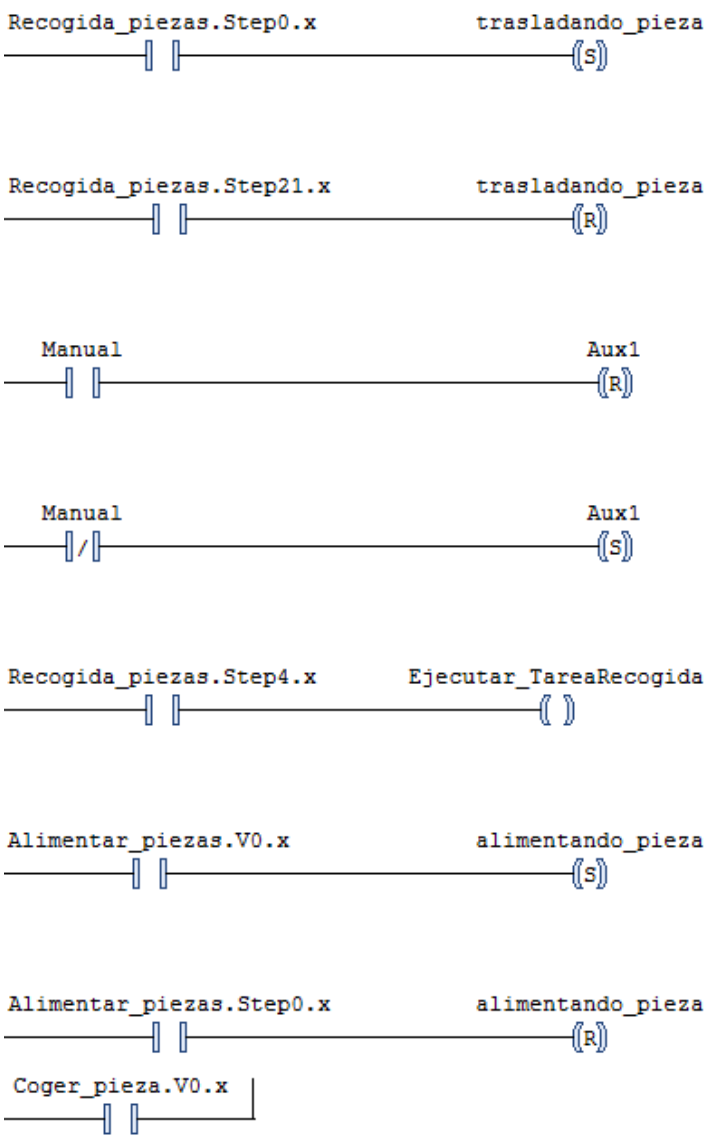


Figura 39. Activación de variables auxiliares.

3.2 Programación del SCADA

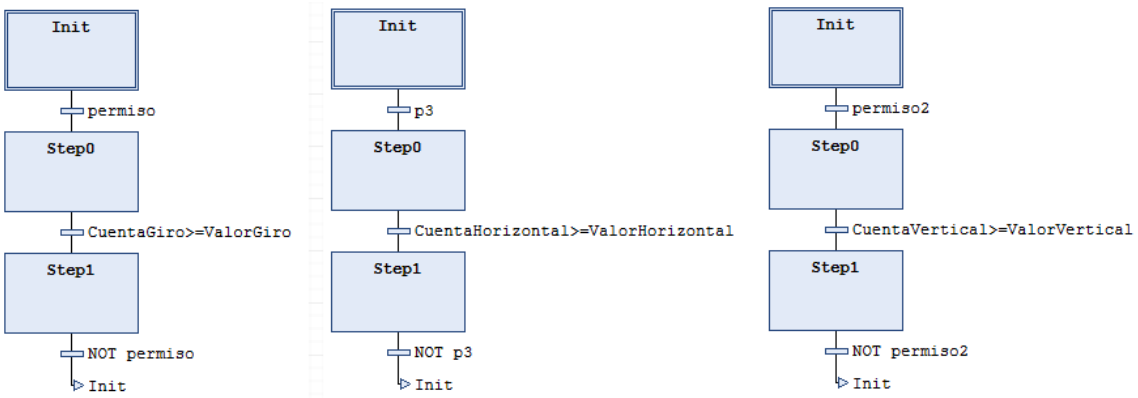


Figura 40. SFCs del controlador numérico. *PruebaGiro, PruebaHorizontal y PruebaVertical,* respectivamente.

TRABAJO FIN DE GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

DESARROLLO E IMPLEMENTACIÓN DE LA AUTOMATIZACIÓN PARA UN SISTEMA DE MECANIZADO DE PIEZAS CON ROBOT MANIPULADOR MEDIANTE COMUNICACIONES DISTRIBUIDAS CON SERVIDOR OPC Y SUPERVISIÓN SCADA

DOCUMENTO Nº 5 ANEXO II. VARIABLES DEL SISTEMA

AUTOR: ABEL DOMÍNGUEZ GONZÁLEZ

TUTOR: RAÚL SIMARRO FERNÁNDEZ

Curso Académico 2016-17

Índice

Índice de figuras

1. Introducción

En este documento se adjunta todas las variables utilizadas para la programación del sistema, tanto de su control físico como de sus interfaces de usuario. Además, se adjuntan anotaciones para una mejor comprensión de las mismas.

2. Línea de mecanizado *FischerTechnik*

```
I1 AT %IX0.0: BOOL;     //Final de carrera frontal del empujador 1
I2 AT %IX0.1: BOOL;     //Final de carrera trasero del empujador 1
I3 AT %IX0.2: BOOL;     //Final de carrera frontal del empujador 2
I4 AT %IX0.3: BOOL;     //Final de carrera trasero del empujador 2
I5 AT %IX0.4: BOOL;     //Fototransistor del empujador 1
I6 AT %IX0.5: BOOL;     //Fototransistor de la fresadora
I7 AT %IX0.6: BOOL;     //Fototransistor de la cinta transportadora de alimentación
I8 AT %IX0.7: BOOL;     //Fototransistor de la taladradora
I9 AT %IX1.0: BOOL;     //Fototransistor de la cinta transportadora de salida
```

Figura 1. Variables de entrada.

```
Q1 AT %QX0.4: BOOL;     //Motor empujador 1 hacia adelante
Q2 AT %QX0.5: BOOL;     //Motor empujador 1 hacia atrás
Q3 AT %QX0.6: BOOL;     //Motor empujador 2 hacia adelante
Q4 AT %QX0.7: BOOL;     //Motor empujador 2 hacia atrás
Q5 AT %QX1.0: BOOL;     //Motor cinta transportadora de alimentación
Q6 AT %QX1.1: BOOL;     //Motor cinta transportadora de la fresadora
Q7 AT %QX1.2: BOOL;     //Motor fresadora
Q8 AT %QX1.3: BOOL;     //Motor cinta transportadora de la taladradora
Q9 AT %QX1.4: BOOL;     //Motor taladradora
Q10 AT %QX1.5: BOOL;    //Motor cinta transportadora de salida
Q11 AT %QX1.6: BOOL;    //Alimentación fototransistores y actuadores
```

Figura 2. Variables de salida.

```
// Elección del modo de fabricación

Pieza_Modelo_I: BOOL;    // Pieza fresada y taladrada
Pieza_Modelo_II: BOOL;   // Pieza solo fresada
Pieza_Modelo_III: BOOL;  // Pieza solo taladrada
Pieza_Manual: BOOL;      // Variable asociada al botón el cual es necesario pulsar para poder cambiar
                        // manualmente el tipo de pieza a fabricar, ignorando la cámara industrial.


//Pueden ser reseteadas por los pulsadores de reset

Piezas_I_terminada: INT;     // Cuenta las piezas del tipo 1 producidas
Piezas_II_terminada: INT;    // Cuenta las piezas del tipo 2 producidas
Piezas_III_terminada: INT;   // Cuenta las piezas del tipo 3 producidas


//Variables asignadas directamente a los pulsadores de reset y contadores

Reset_pieza_I: BOOL;     // Resetea el contador de las piezas del tipo 1
Reset_pieza_II: BOOL;    // Resetea el contador de las piezas del tipo 2
Reset_pieza_III: BOOL;   // Resetea el contador de las piezas del tipo 3
Piezas_Disponibles: INT; //Inidica el número de piezas disponibles en stock para poder fabricar
Sin_Piezas: BOOL;        //Indica que no hay más piezas para poder producir
```

Figura 3. Variables de las piezas a mecanizar.

```
//Variables auxiliares
    Salidas_OFF: bool;      //Salidas físicas (Q1.....Q10) OFF
    Proceso_enMarcha: BOOL; //El proceso esta en marcha (alguna/s de la/s salida/s están activas)

//Variables de escritura

    Usuario,Registro_usuario, Password, Registro_password, Clave_activacion: STRING;  // Guarda las cadenas de caracteres del nombre de usuario y contraseña


// Variables auxiliares de las visualizaciones

    Registro: BOOL;         //Si está activa, la ventana de registro de usuario es invisible.
                            //Se desactiva pulsando el botón "registrarse".
    No_Registro: BOOL;      //Si está desactivada, permite al usuario registrarse.
                            //Cuando el usuario se registra y se logea (sin darle a siguiente), se activa e impide otro registro de usuario.
    Reset_Registro: BOOL;   //Reset del contador, el cual es usado para permitir un único registro de usuario
    Cuenta_Registro: INT;   //Cuenta del contador, el cual es usado para permitir un único registro de usuario
    Registro_aux: BOOL;     //Tendrá el valor opuesto del valor actual de la variable "Registro".
                            //Sirve para conmutar la visibilidad entre panel de registro y de login de usuario.
    clave: STRING;          //Contiene el 'key' de activación del software proporcionado por la empresa en cuestión
    Usuario_OK: BOOL;       //Verifica que el nombre de usuario registrado y el de login coincidan.
    Password_OK: BOOL;      //Verifica que la contraseña de usuario registrado y el de login coincidan.
    clave_OK: BOOL;         //Verifica que la 'key' proporcionada por el fabricante coincida con la introducida.
    Paso_a_SCADA: BOOL;     //Permite, una vez logeado, pinchar en el botón "siguiente" e ir al SCADA del sistema
                            // (para que esto ocurra, la variable debe estar en FALSE).
```

Figura 5. Variables auxiliares y de la interfaz de registro y acceso de usuario.

```
    Boton_Start: BOOL;      //Se activa al pulsar el botón "start" (inicio del sistema).
                            //También muestra si el botón está presionado (TRUE) o no (FALSE). También muestra el texto "Start" en rojo si está en FALSE.

    Boton_Automatico: BOOL; //Se desactiva al pulsar el botón "start". De esta forma, permite usar el botón del modo Automático
    Boton_Manual: BOOL;     //Se desactiva al pulsar el botón "start". De esta forma, permite usar el botón del modo Manual
    Boton_Test: BOOL;       //Se desactiva al pulsar el botón "start". De esta forma, permite usar el botón del modo Test
    Start_verde: BOOL;      //Muestra el texto "Start" en verde si está en FALSE.
    Autom_rojo: BOOL;       //Muestra si el botón está presionado (TRUE) o no (FALSE).
                            //También muestra el texto "Automático" en rojo si está en FALSE.
    Autom_verde: BOOL;      //Muestra el texto "Automático" en verde si está en FALSE.
    Automatico: BOOL;       //Variable que permite ejecutar los grafcets de los procesos si esta en TRUE (El proceso se ejecuta de forma automática).
    Manual_rojo: BOOL;      //Muestra si el botón está presionado (TRUE) o no (FALSE).
                            //También muestra el texto "Manual" en rojo si está en FALSE.
    Manual_verde: BOOL;     //Muestra el texto "Manual" en verde si está en FALSE.
    Manual: BOOL;           //Variable que permite ejecutar los grafcets de los procesos si esta en TRUE
                            //(Será un pulsador, para que el proceso avance cuando el usuario lo indique).
    Manual_marcha: BOOL;    //Cuando se active (mediante un pulsador), le proporciona permiso al sistema para realizar la siguiente operación.
    Test_rojo: BOOL;        //Muestra si el botón está presionado (TRUE) o no (FALSE).
                            //También muestra el texto "Test" en rojo si está en FALSE.
    Test_verde: BOOL;       //Muestra el texto "Test" en verde si está en FALSE.
    Test: BOOL;             //Variable que permite utilizar los botones de todos los accionadores.
    Test_botones: BOOL;     //Permite ver los botones y pulsadores de los accionadores cuando esta variable está en false (variable test en true).
    Test_Cinta1: BOOL;      //Botón que activa/desactiva la cinta 1 en el modo test (manualmente).
    Test_Cinta2: BOOL;      //Botón que activa/desactiva la cinta 2 en el modo test (manualmente).
    Test_Cinta3: BOOL;      //Botón que activa/desactiva la cinta 3 en el modo test (manualmente).
    Test_Cinta4: BOOL;      //Botón que activa/desactiva la cinta 4 en el modo test (manualmente).
    Test_piston1_A: BOOL;   //Pulsador que activa el pistón 1 hacia adelante en modo test (manualmente).
    Test_piston1_C: BOOL;   //Pulsador que activa el pistón 1 hacia atrás en modo test (manualmente).
    Test_piston2_A: BOOL;   //Pulsador que activa el pistón 1 hacia adelante en modo test (manualmente).
    Test_piston2_C: BOOL;   //Pulsador que activa el pistón 1 hacia atrás en modo test (manualmente).
    Test_Fresadora: BOOL;   //Botón que activa/desactiva la fresadora en el modo test (manualmente).
    Test_Taladradora: BOOL; //Botón que activa/desactiva la taladradora en el modo test (manualmente).
    Stop: BOOL;             //Parada de emergencia
    Stop_rojo: BOOL;
    Stop_verde: BOOL;
```

Figura 4. Variables enlazadas con los botones de la interfaz de proceso. *Visu_proceso.*

```
//Variables para el cambio de texto rojo/verde
Cinta1_rojo: BOOL;
Cinta1_verde: BOOL;
Cinta2_rojo: BOOL;
Cinta2_verde: BOOL;
Cinta3_rojo: BOOL;
Cinta3_verde: BOOL;
Cinta4_rojo: BOOL;
Cinta4_verde: BOOL;
Proceso_reposo: BOOL;
Proceso_marcha: BOOL;
Proceso_espera: BOOL;
Texto_automatico: BOOL;
Texto_manual: BOOL;
Texto_test: BOOL;



//Variables para el movimiento de objetos animados en visu_proceso

Mover_piston1: INT;
Mover_piston2: INT;
Fresadora_A: BOOL;
Fresadora_B: BOOL;
Taladradora_A: BOOL;
Taladradora_B: BOOL;
Cinta1A: BOOL;
Cinta1B: BOOL;
Cinta2A: BOOL;
Cinta2B: BOOL;
Cinta3A: BOOL;
Cinta3B: BOOL;
Cinta4A: BOOL;
Cinta4B: BOOL;
Fresado_hecho: INT;
```

Figura 6. Variables para la conmutación de colores de texto y objetos en la interfaz de proceso.

```
Vacio_F: BOOL;
Vacio_T: BOOL;
Lleno_F: BOOL;
Lleno_T: BOOL;
nElementos_F: INT;
nElementos_T: INT;
RESET: BOOL;
F_Desencolar: BOOL;  //Quita de la cola del fifo para fresadora
T_Desencolar: BOOL;  //Quita de la cola del fifo para taladradora
DatoCarga AT %MW1: WORD; //Dato que recoge la lista fifo
F_DatoDescarga: DWORD;   //Dato que elimina de la lista fifo una vez leído (lista fresadora)
T_DatoDescarga: DWORD;   //Dato que elimina de la lista fifo una vez leído (lista taladradora)
F_FIFO_OK_1: BOOL;   // Se activa cuando la fresadora tiene que ponerse en marcha para fabricar piezas tipo I
F_FIFO_OK_2: BOOL;   // Se activa cuando la fresadora tiene que ponerse en marcha para fabricar piezas tipo II
T_FIFO_OK_1: BOOL;       // Se activa cuando la taladradora tiene que ponerse en marcha para fabricar piezas tipo I
T_FIFO_OK_3: BOOL;       // Se activa cuando la taladradora tiene que ponerse en marcha para fabricar piezas tipo III
```

Figura 7. Variables de las funciones FIFO.

```
ADDM_0: ADDM;
ADDR_Esclavo: ADDRESS;
exe_ADDM: BOOL;
empezar_comunicacion: BOOL;
ADDM_Error: BOOL;
ADDM_CommError: BYTE;
MW: ObjectType;
READ_Busy: BOOL;
READ_Aborted: BOOL;
READ_Error: BOOL;
READ_CommError: BYTE;
READ_OperError: DWORD;
WRITE_Busy: BOOL;
WRITE_Aborted: BOOL;
WRITE_Error: BOOL;
WRITE_CommError: BYTE;
WRITE_OperError: DWORD;
READ_VAR_0: READ_VAR;
WRITE_VAR_0: WRITE_VAR;
BufferEscritura: ARRAY [0..1] OF WORD;
BufferLectura: ARRAY [0..1] OF WORD;
Inicio_escritura: BOOL;
Done_OK: BOOL; //ADDM OK
Done_write_OK: BOOL; //Write OK
Encolar: BOOL;
```

Figura 8. Variables de la comunicación Maestro / esclavo.

```
// Nº de piezas detectadas por la cámara
Negras: INT;
Azules: INT;
Rojas: INT;
//Piezas que siguen en el proceso de mecanizado
Negras_enProceso AT %MW2: INT;
Azules_enProceso AT %MW3: INT;
Rojas_enProceso AT %MW4: INT;
pieza_detectada AT %MX0.0: BOOL;
```

Figura 9. Variables usadas para el conteo de las piezas detectadas y en proceso.

3. Robot manipulador de vacío *FischerTechnik*

```
Input_B1 AT %IX0.0: BOOL;   //Pulsos encoder vertical
Input_B3 AT %IX0.1: BOOL;   //Pulsos encoder horizontal
Input_B5 AT %IX0.2: BOOL;   //Pulsos encoder giro
I1 AT %IX0.4: BOOL;         //Final de carrera  vertical
I2 AT %IX0.5: BOOL;         //Final de carrera  horizontal
I3 AT %IX0.6: BOOL;         //Final de carrera  giro
```

Figura 10. Variables de entrada.

```
Q1 AT %QX0.4: BOOL;         //Motor movimiento vertical arriba
Q2 AT %QX0.5: BOOL;         //Motor movimiento vertical abajo
Q3 AT %QX0.6: BOOL;         //Motor movimiento retroceso horizontal
Q4 AT %QX0.7: BOOL;         //Motor movimiento avance horizontal
Q5 AT %QX1.0: BOOL;         //Motor movimiento giro horario
Q6 AT %QX1.1: BOOL;         //Motor movimiento giro antihorario
Q7 AT %QX1.2: BOOL;         //Habilitar compresor
Q8 AT %QX1.3: BOOL;         //Succión ventosa
Q9 AT %QX1.4: BOOL;         //Habilitar señales entrada proceso
```

Figura 11. Variables de salida.

```
CuentaVertical: INT;        //Cuenta número de pulsos del estado actual en eje vertical
CuentaHorizontal: INT;      //Cuenta número de pulsos del estado actual en eje horizontal
CuentaGiro: INT;            //Cuenta número de pulsos del estado actual en eje de giro
MaxCuentaVertical: INT := 1000;     //Cuenta máxima vertical
MaxCuentaHorizontal: INT := 1000;   //Cuenta máxima horizontal
MaxCuentaGiro: INT := 750;          //Cuenta máxima giro
```

Figura 12. Variables para las cuentas de pulsos.

```
Boton_Q1 : BOOL;
Boton_Q2 : BOOL;
Boton_Q3 : BOOL;
Boton_Q4 : BOOL;
Boton_Q5 : BOOL;
Boton_Q6 : BOOL;
Boton_Q7: BOOL;
Boton_Q8: BOOL;
```

Figura 13. Variables asociadas a los botones del SCADA.

```
Pieza_para_fabricar: BOOL;
alimentando_pieza: BOOL;
Manual: BOOL;
Aux1: BOOL;
Recoger_pieza: BOOL;    //TRUE, recoge la pieza terminada para llevarla a su lugar correspondiente
trasladando_pieza, pieza_en_almacen: BOOL;

permiso,permiso2,p3,p4,p5,p6: BOOL; //Variables para probar configuraciones con el robot en el modo de control numérico
```

Figura 14. Variables auxiliares.

```
DatoCarga AT %MW1: WORD;
B_DatoDescarga: DWORD;
B_Vacio: BOOL;
B_Lleno: BOOL;
B_nElementos: INT;
RESET: BOOL;
pieza_detectada AT %MX0.0: BOOL;
B_FIFO_I: BOOL;
B_FIFO_II: BOOL;
B_FIFO_III: BOOL;
```

Figura 15. Variables asociadas a la función FIFO.

```
Escribeme  AT %MX0.1: BOOL;
```

Figura 16. Variable utilizada para la comunicación con el sistema de mecanizado.

```
ValorGiro: INT;
ValorVertical: INT;
ValorHorizontal: INT;
Reset_movimiento: BOOL;
```

Figura 17. Variables asociadas al número insertado en el control numérico. Además de variable de reset del sistema.

```
//Cuentan el nº de piezas que el brazo a recogido a lo largo del proceso. Ej: Ha recogido hasta ahora 2 rojas, 3 negras y 0 azules
Negras: INT;
Azules: INT;
Rojas: INT;
//Distancia a la cual el robot debe bajar verticalmente para depositar las piezas (teniendo en cuenta las que ya ha puesto anteriormente)
VerticalNegras: INT;
VerticalAzules: INT;
VerticalRojas: INT;
//Variable para activar tarea de evento "recogida" (para ejecutar unos IF una sola vez)
Ejecutar_TareaRecogida: BOOL;

//Variables para almacenar las piezas manualmente: que el robot ponga las piezas donde el usuario quiera (dentro de las programadas)

PosicionarNegra: BOOL;
PosicionarAzul: BOOL;
PosicionarRoja: BOOL;
```

Figura 18. Variables relacionadas con la recogida y la clasificación de las piezas.