Trabajo de Fin de Grado

Grado en Ingeniería de Tecnologías Industriales

# Librería de automatización y simulación de una célula de fabricación flexible mediante Codesys

Autor: Jesús Martínez Fuentes

Tutor: Luis Fernando Castaño Castaño

**Dpto. de Ingeniería de Sistemas y Automática**

**Escuela Técnica Superior de Ingeniería**

**Universidad de Sevilla**

Sevilla, 2020

Trabajo de Fin de Grado
Grado en Ingeniería de Tecnologías Industriales

# Librería de automatización y simulación de una célula de fabricación flexible mediante Codesys

Autor:
Jesús Martínez Fuentes

Tutor:
Luis Fernando Castaño Castaño
Doctor Ingeniero Industrial

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2020

Trabajo de Fin de Grado: Librería de automatización y simulación de una célula de fabricación flexible mediante Codesys

Autor: Jesús Martínez Fuentes

Tutor: Luis Fernando Castaño Castaño

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2020

El Secretario del Tribunal

*A mi familia*

*A mis maestros*

# Agradecimientos

En estos humildes párrafos van mis esfuerzos por recompensar el tiempo, esfuerzo y dedicación que tanta gente ha depositado en mí para permitirme llegar hasta aquí.

A mi madre, mi padre y mis hermanos. Incondicionales, se merecen más que nadie este reconocimiento que, aunque discreto y siempre insuficiente, inmortaliza mi aprecio y admiración por lo que hicieron, hacen y no dejarán de hacer por mí.

A mis amigos, fruto de mis propias decisiones, que son las que me unen a ellos. Y no solo mías, pues la reciprocidad de estos lazos los hace especiales. Un acuerdo de cariño, apoyo y compañía que merece la pena cuidar tanto como pocas cosas más en esta vida.

A los profesores y profesoras, de vocación, que con sus más sinceras intenciones se han entregado a formarme a mí entre muchísmos otros, año tras año y curso tras curso, sin abandonar la cercanía y pasión con la que desarrollan la más honorable labor que pueda existir.

Gracias de corazón.

*Jesús Martínez Fuentes*

*Sevilla, 2020*

# Resumen

Este documento contiene la presentación de una librería de automatización, que cuenta con bloques funcionales desde un nivel de control básico, como sería generar un tren de pulsos para un motor paso a paso, hasta el nivel de estación completa, con sus sensores, actuadores, movimientos y rutinas de funcionamiento específicas.

Las estaciones tratadas están basadas en las máquinas autónomas montadas en la célula de fabricación flexible, dentro del laboratorio del departamento de Ingeniería de Sistemas y Automática, de la Escuela Técnica Superior de Ingeniería de Sevilla.

Su principal valor se fundamenta en su carácter ilustrativo y didáctico, siendo una ejemplificación magistral de las posibilidades de automatización en lenguaje LD a través de bloques funcionales dentro del entorno de programación de Codesys, siendo fácilmente extrapolable a otros entornos de programación gracias al estándar internacional IEC 61131-3, e incluso a otros propósitos de control, dentro de las posibilidades de reconfiguración que ofrece la propia librería.

Para este mismo propósito, los controladores de nivel de estación cuentan con pantallas de operador diseñadas en Codesys, con interfaces que permiten ilustrar el funcionamiento de los controladores e interactuar con estos de manera cómoda e intuitiva. Este HMI está pensado para que pueda estar operativo incluso en una simulación virtual, permitiendo ejecutar la demostración sin necesidad de conectarse a un PLC o a los dispositivos que conforman la máquina a controlar.

# Índice

# ÍNDICE DE FIGURAS

# Notación

| | |
|---|---|
| ARRAY | Variable vector |
| BOOL | Variable tipo booleana |
| DINT | Variable tipo *double* entero |
| Hz | Hercios |
| INT | Variable tipo entero |
| mm | Milímetros |
| PaP | Paso a paso |

# 1 INTRODUCCIÓN

La librería de automatización que se detallará a continuación consta de diversos bloques funcionales, tanto para el control de dispositivos y máquinas, como para el control de un simulador que los acompaña, permitiendo reproducir su desempeño de forma virtual. En esta librería, se pueden diferenciar dos partes:

- Bloques funcionales básicos.
- Bloques funcionales de estación.

Todos ellos conformarían los instrumentes suficientes para poder elaborar el sistema de control de una célula de fabricación flexible, basada en la célula de fabricación montada en los laboratorios del departamento de ingeniería de sistemas y automática. De esta se entrará en detalle a continuación.

## 1.1. Bloques funcionales básicos

Los bloques funcionales básicos de la librería componen la base del diseño de los bloques funcionales de control de las estaciones. Esta parte de la librería consta de los siguientes bloques: GenPulsos, MueveMotorCPU, MueveAcoplamientoCPU, MueveDispositivoCPU, OperaPinzaCPU.

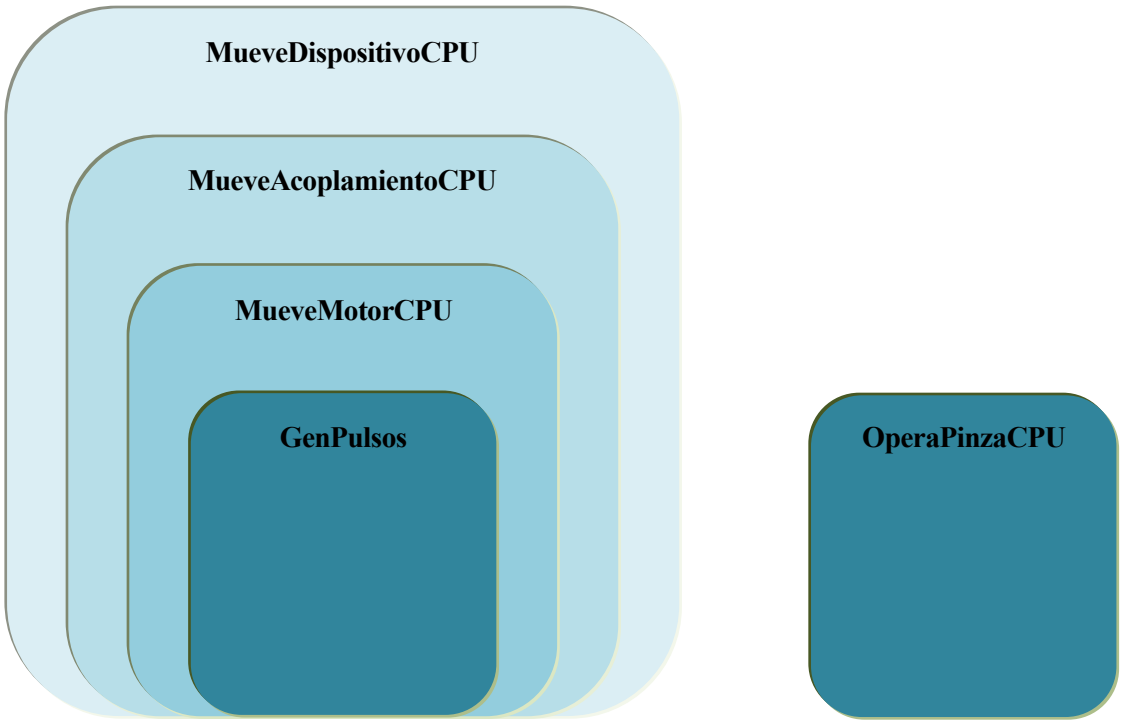


Figura 1-1. Estructura del diseño de los bloques funcionales básicos

Los primeros responden al control de un motor PaP universal a través de un driver compatible. Su programación, detallada en los capítulos posteriores, está planteada con una estructura escalonada por capas. Esto es, partiendo desde la función más básica (GenPulsos), que permite generar trenes de pulsos, se construye entorno a este nivel a nivel, para acabar obteniendo el bloque funcional completo (MueveDispositivoCPU), capaz de controlar un dispositivo de motor PaP plenamente funcional y capaz de realizar operaciones de calibración.

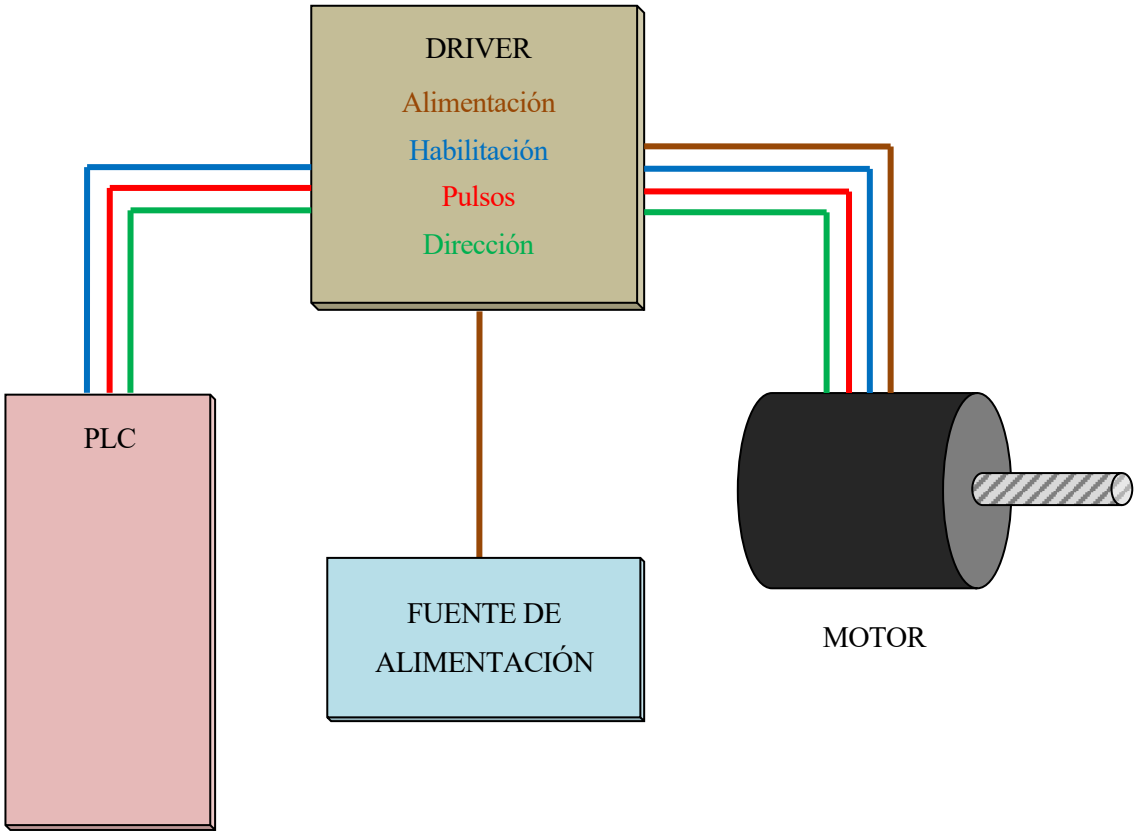


Figura 1-2. Configuración típica de motor PaP más driver

Por su parte, el bloque OperaPinzaCPU responde al control de la garra de una pinza estándar y es independiente de los bloques anteriores.

## 1.2. Bloques funcionales de estación

Los bloques funcionales de estación de la librería son los instrumentos de diseño de la automatización de una célula de fabricación flexible completa y funcional, partiendo nuevamente de las máquinas que hay disponibles en el laboratorio.

Esta célula de fabricación situada en el laboratorio funciona como una simulación de planta industrial. Consta de diversas máquinas dispuestas entorno a una red de cintas transportadoras. La función de esta célula de fabricación es puramente didáctica, como banco de pruebas para evaluar los controladores programados para las distintas máquinas y la sincronización de estas cuando trabajan conjuntamente. Adicionalmente sirve para ilustrar la posibilidad de reconfiguración de las estaciones, permitiendo diseñar células con procesos productivos distintos introduciendo nuevas estaciones o incluso cambiendo el orden de intervención de estos.

Es por eso que se puede hablar de célula de fabricación flexible. Para ese propósito es ideal la librería de automatización diseñada, permitiendo crear la célula de fabricación idónea para cada situación, dependiendo de qué estaciones y cómo se decidan implementar.

Los bloques disponibles en la librería son los siguientes: AlimentadorPiezasCPU, PorticoCPU, PuenteGruaCPU, ScaraCPU, ScorbotCPU, AlmacenPaletsCPU, AlimentadorBandejasCPU.

Cada una de estas máquinas, cuenta además con un simulador y una pantalla de operador, que permite visualizar en una interfaz gráfica el desempeño de las funciones de manera virtual.

A parte de las máquinas, otros elementos de importancia en la producción son las bandejas, los palets y las piezas.

Las bandejas son plataformas cuadradas empleadas para transportar el producto a lo largo del ciclo de producción. Sobre ellos se colocan los palets.

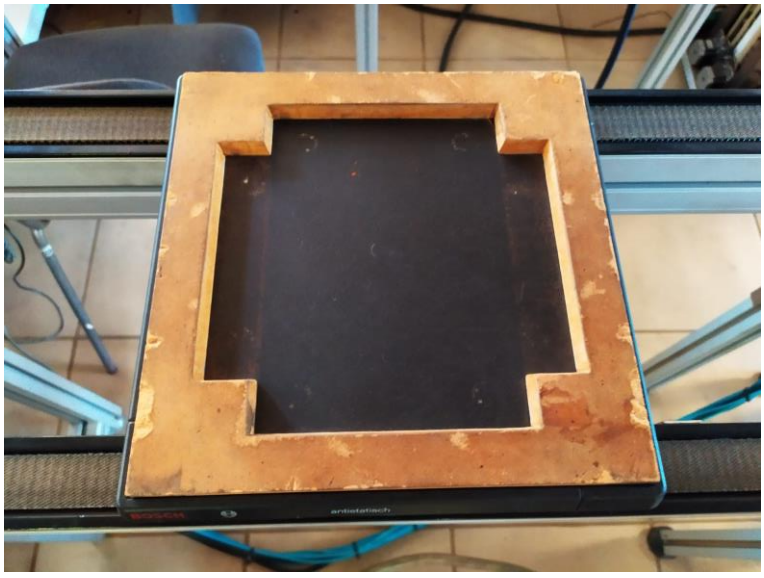

Figura 1-3. Bandeja – Fotografía del laboratorio

Los palets son estructuras rectangulares de plástico, que cuentan con ocho cilindros cónicos dispuestos a modo de soportes para dos piezas.

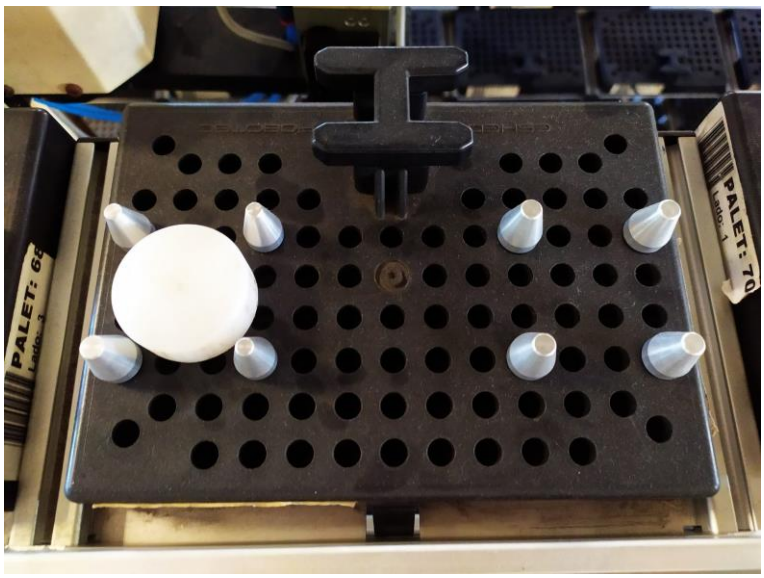


Figura 1-4. Palet con una pieza – Fotografía del laboratorio

Por último, las piezas son cilindros de reducido tamaño pensados para colocar en los palets. Hay cuatro variedades disponibles, de altura y materiales distintos, con los cuales, dependiendo de la selección escogida, conformarían junto al palet un producto determinado.

Figura 1-5. Variedad de piezas – Fotografía del laboratorio

# 2 BLOQUES FUNCIONALES BÁSICOS

El primer acercamiento al problema de obtener una librería de control de las distintas estaciones de una célula de fabricación flexible pasa por confeccionar los bloques funcionales básicos. La idea es buscar un elemento mínimo de control común. En este capítulo se hará un repaso de estos bloques funcionales, transversales en la automatización de las máquinas completas.

Se verá:

- GenPulsos.
- MueveMotorCPU.
- MueveAcoplamientoCPU.
- MueveDispositivoCPU.
- OperaPinzaCPU.

## 2.1. GenPulsos

### 2.1.1 Introducción

El bloque funcional GenPulsos es el más básico de los bloques funcionales que se detallarán, pero cumple un propósito imprescindible en el control de motores PaP. Para un periodo dado, este reproducirá en salida una onda cuadrada a la frecuencia establecida. Se trataría de un generador de pulsos sostenidos ideal para marcar la velocidad de giro de un motor PaP.

### 2.1.2 Variables del bloque funcional

**Argumentos de entrada**

**EN:** BOOL. Variable de habilitación del bloque funcional. A nivel alto, el bloque estará en funcionamiento.

**Periodo:** INT. Periodo que se desee especificar para la señal de salida, en milisegundos (ms).

**Argumentos de salida**

**ENO:** BOOL. Salida de habilitación, a nivel alto mientras el bloque funcional esté igualmente habilitado.

**OndaCuadrada:** BOOL. Señal de salida cuadrada y periódica.

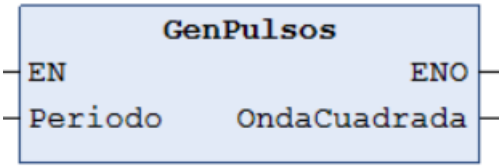


Figura 2-1. Bloque funcional GenPulsos

### 2.1.3 Funcionamiento

La idea de tener un bloque capaz de generar una onda cuadrada a la frecuencia deseada es poder reproducir un tren de pulsos que enviar al motor para marcar su giro. Esto se consigue a través de una pareja de temporizadores TON, que se van activando recíprocamente creando un pulso cada periodo. Para transformar esta señal de pulsos instantáneos see emplea un temporizador TOF, que manténgala señal a nivel alto durante medio periodo.

De esta forma, para un Periodo=1000, teniendo la habilitación activada EN=1, la salida OndaCuadrada describiría la siguiente trayectoria:

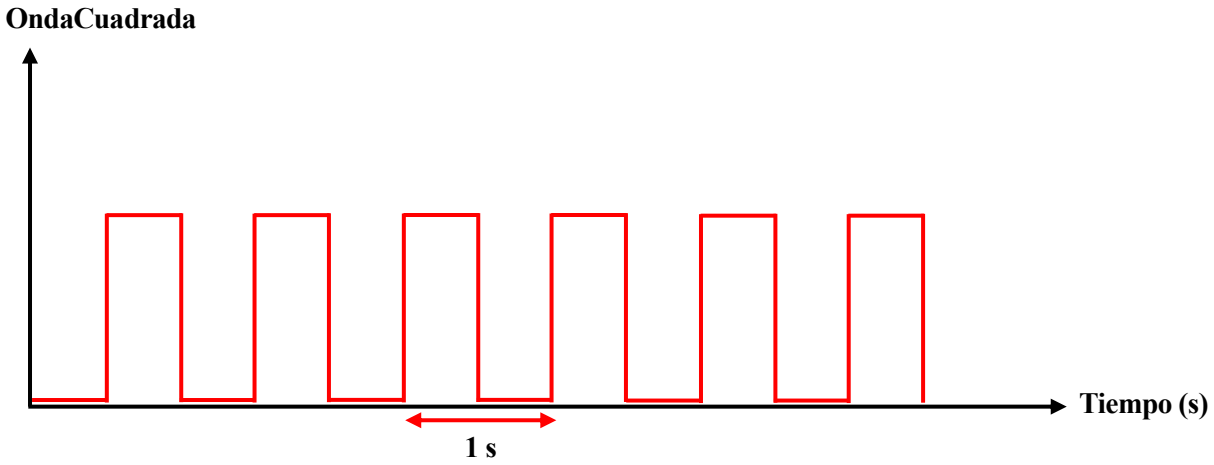


Figura 2-2. Cronograma GenPulsos

## 2.2. MueveMotorCPU

### 2.2.1. Introducción

El bloque funcional MueveMotorCPU constituye la unidad mínima e imprescindible para el control de un motor paso a paso universal. Para una referencia establecida y teniendo en cuenta la posición inicial, la función genera los pulsos para el motor a la frecuencia indicada. Entraremos más en detalle a continuación.

### 2.2.2. Variables del bloque funcional

**Argumentos de entrada:**

**RefPosPulsosVirtual:** DINT. Es la referencia que queremos que alcance el motor, referida en unidades de pasos.

**Hz:** INT. Frecuencia a la que queremos que funcione el motor en hercios (Hz). Funciona en un rango de [1,60].

**STOP:** BOOL. Parada de emergencia. A nivel alto, el motor se para de inmediato. Una vez vuelva a 0, el motor retomará su funcionamiento desde la posición en la que quedó.

**FrenoEnReposo:** BOOL. Variable digital que mantiene el motor habilitado incluso una vez alcanzada la referencia (como si de un freno se tratase) de modo que no se pierdan pasos por el efecto de la gravedad. Cuando esté a 1, el freno se activa, energizando la bobina a través de la variable de salida EN_Driver.

**Argumento de salida:**

**EN_Driver:** BOOL. Señal de habilitación del motor. A 1 está el motor habilitado (bobinas energizadas) y a 0 está deshabilitado.

**DIR_Driver:** BOOL. Señal que marca la dirección de giro del motor. A 0, el motor avanza a posiciones positivas (y a posiciones negativas cuando esté a 1).

**STEP _Driver:** BOOL. Señal de pulsos que recibe el motor a la frecuencia estipulada, marcando la velocidad de giro.

**ESTADO_ERROR:** INT. Variable de información para el usuario. Puede tomar los siguientes valores:

- 0 si el motor está en reposo.
- 1 si el motor está en movimiento.
- 2 motor en parada de emergencia.
- 3 frecuencia introducida fuera de rango.

**Argumentos de entrada/salida:**

**PosPulsosVirtual:** DINT. Contiene la posición actual en la que se encuentra el motor, referida en unidades de pasos. También es señal de entrada ya que nos permite sobrescribir una posición distinta si lo elegimos así.

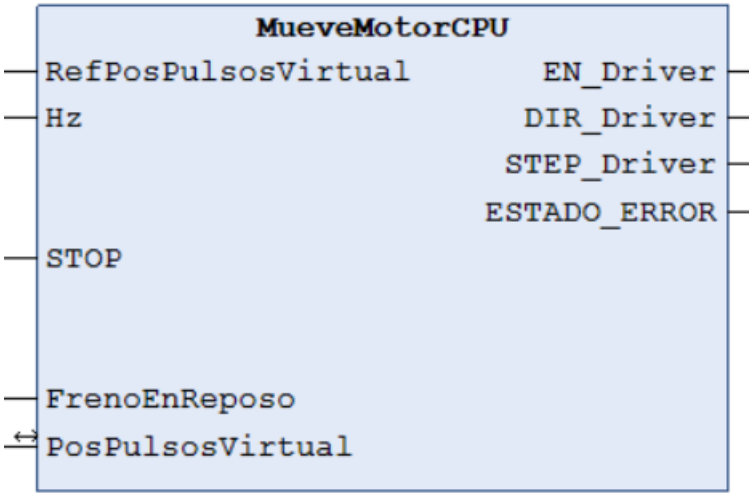


Figura 2-3. Bloque funcional MueveMotorCPU

### 2.2.3. Funcionamiento

Las salidas del bloque dan una pista del propósito del mismo. Este ha de gestionar los parámetros de entrada del driver, explicados previamente. Cuando se de una referencia diferente a la posición actual, el motor debe ponerse en marcha. Gracias al bloque GenPulsos, se puede obtener un tren de pulsos para la salida STEP_DRIVER. Ahora la entrada es en frecuencia, así que habrá que adaptarla antes de pasársela al bloque subordinado como parámetro. Comparando la referencia con la posición actual también se puede definir la dirección de giro. Para tener en todo momento una estimación de la posición actual, se lleva internamente la cuenta de los pulsos transmitidos. De ahí el sobrenombre de "virtual" para esta variable. Por su parte, la entrada STOP deja de transmitir pulsos al driver y el freno mantiene el motor habilitado incluso cuando no necesita moverse.

Para Hz=1 (nótese que es equivalente al anterior Periodo=1000), FrenoEnReposo=1 y partiendo desde PosPuslsoVirtual=0, se establece la referencia RePosPulsosVirtual=2 y al rato se reestablece a 0. Se tendría la siguiente respuesta en el tren de pulsos y la posición.

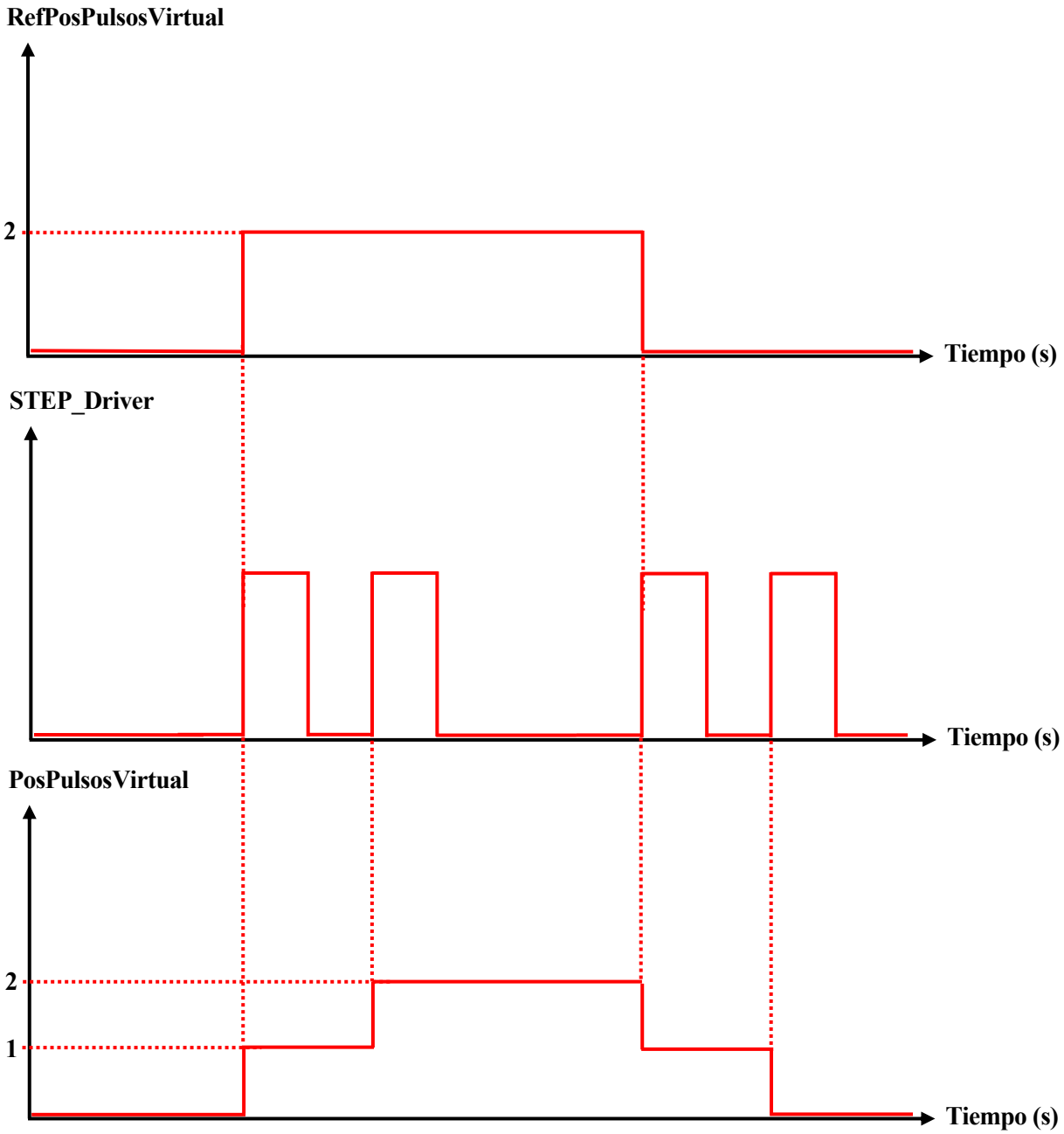


Figura 2-4. Cronograma MueveMotorCPU

## 2.3. MueveAcoplamientoCPU

### 2.3.1. Introducción

El bloque funcional MueveAcoplamientoCPU parte del bloque anterior MueveMotorCPU, incluyendo ciertas modificaciones para facilitar su uso en situaciones reales. Del mismo modo, dada una referencia en posición y una frecuencia se transmite un tren de pulsos dirigido al motor. Si bien el anterior empleaba posiciones referidas en pasos, esta lo hace en las unidades de ingeniería a las que sea preciso referenciarse.

### 2.3.2. Variables del bloque funcional

**Argumentos de entrada:**

**RefPosicionVirtual:** DINT. Es la referencia que queremos que alcance el motor, referida en unidades de ingeniería dadas.

**Avance:** DINT. Relación entre las unidades de ingeniería y los pasos del motor. Las unidades empleadas se escogerán libremente, atendiendo a la naturaleza del movimiento final efectuado. Por ejemplo, para un motor cuyo propósito sea levantar una pieza, se puede tomar como avance el desplazamiento que realiza dicha pieza por cada pulso de motor.

**Hz:** INT. Frecuencia a la que queremos que funcione el motor en hercios (Hz). Funciona en un rango de [1,60].

**STOP:** BOOL. Parada de emergencia. A nivel alto, el motor se para de inmediato. Una vez vuelva a 0, el motor retomará su funcionamiento desde la posición en la que quedó.

**FrenoEnReposo:** BOOL. Variable digital que mantiene el motor habilitado incluso una vez alcanzada la referencia (como si de un freno se tratase) de modo que no se pierdan pasos por el efecto de la gravedad. Cuando esté a 1, el freno se activa, energizando la bobina a través de la variable de salida EN_Driver.

**Argumento de salida:**

**EN_Driver:** BOOL. Señal de habilitación del motor. A 1 está el motor habilitado (bobinas energizadas) y a 0 está deshabilitado.

**DIR_Driver:** BOOL. Señal que marca la dirección de giro del motor. A 0, el motor avanza a posiciones positivas (y a posiciones negativas cuando esté a 1).

**STEP _Driver:** BOOL. Señal de pulsos que recibe el motor a la frecuencia estipulada, marcando la velocidad de giro.

**ESTADO_ERROR:** INT. Variable de información para el usuario. Puede tomar los siguientes valores:

- 0 si el motor está en reposo.
- 1 si el motor está en movimiento.
- 2 motor en parada de emergencia.
- 3 frecuencia introducida fuera de rango.
- 4 avance introducido de valor nulo.

**Argumentos de entrada/salida:**

**PosicionVirtual:** DINT. Contiene la posición actual en la que se encuentra el motor, referida en unidades de ingeniería. También es señal de entrada ya que nos permite sobrescribir una posición distinta si lo elegimos así.

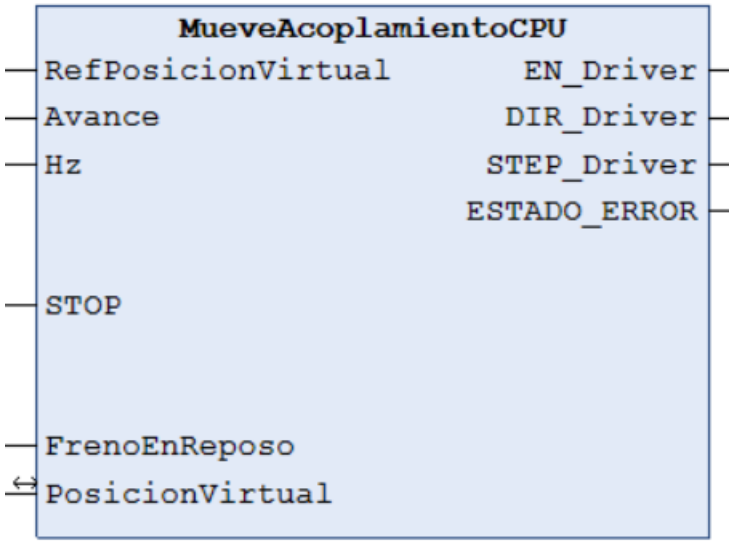


Figura 2-5. Bloque funcional MueveAcoplamientoCPU

### 2.3.3. Funcionamiento

Hechos los cambios de unidades dados por la variable Avance, el funcioniemiento es exactamente el mismo. Este dato debería obtenerse experimentalmente, ya que depende de motor PaP empleado y del contexto de funcionamiento del mismo. Si el motor PaP se va a emplear para el giro de una articulación de un brazo robótico, sería recomendable usar una relación que contenga una unidad angular. A continuación, empleando MueveMotorCPU, se comprobaría el ángulo que describe el brazo robótico por cada paso que da el motor PaP (ej. 2º/paso). Por su parte, si su finalidad fuese conseguir un desplazamiento rectilíneo, sería más conveniente emplear unidades de medida de longitud.

Empleando este bloque, se abandona la referencia de los pasos, viéndose representado la posición y el avance según las unidades de ingeniería elegidas. Por este motivo, se debe ser consistente con la relación de avance y la posición de referencia. Es decir, si se comprueba que el avance es de 2 mm/paso, las posiciones que recorrerá el motor serán múltiplos de este y la referencia dada deberá ser igualmente en milímetros, no en pasos. Para RefPosicionVirtual=20, se transmitirán 10 pulsos al driver del motor PaP, no al revés.

## 2.4. MueveDispositivoCPU

### 2.4.1. Introducción

El bloque funcional MueveDispositivoCPU está basado en el funcionamiento de un motor ya acoplado en una máquina con un área de trabajo delimitada y sensores que aportan información al conjunto.

Consta de dos finales de carrera y un sensor de posición para realizar maniobra de calibración de “homing” (búsqueda de la posición inicial Home).

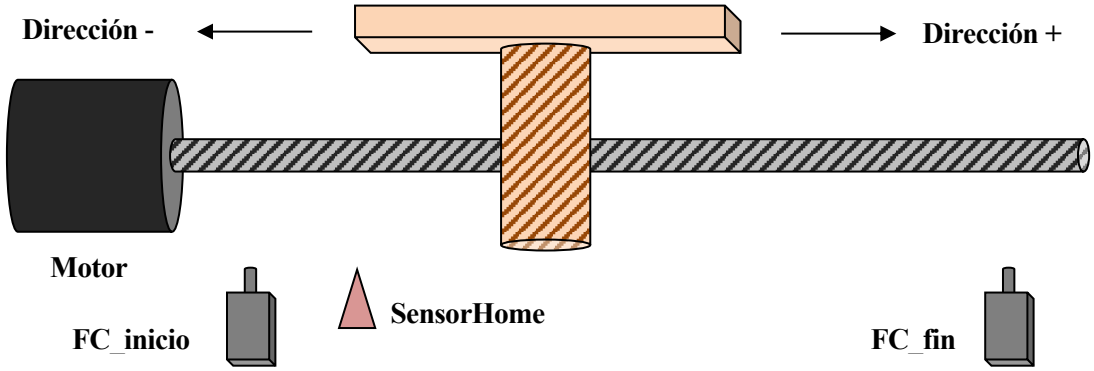


Figura 2-6. Esquema del dispositivo

### 2.4.2. Variables del bloque funcional

**Argumentos de entrada:**

**RefPosicionVirtual:** DINT. Es la referencia que queremos que alcance el motor, referida en unidades de ingeniería dadas.

**Avance:** DINT. Relación entre las unidades de ingeniería y los pasos del motor. Las unidades empleadas se escogerán libremente, atendiendo a la naturaleza del movimiento final efectuado. Por ejemplo, para un motor cuyo propósito sea levantar una pieza, se puede tomar como avance el desplazamiento que realiza dicha pieza por cada pulso de motor.

**Carrera:** DINT. Distancia en unidades de ingeniería (referidas al avance) que delimita el segmento de operación del motor.

**HzVelocidadAlta:** INT. Frecuencia a la que queremos que funcione el motor en las tareas de posicionamiento rápido, en hercios (Hz). Funciona en un rango de [1,60].

**HzVelocidadBaja:** INT. Frecuencia a la que queremos que funcione el motor en las tareas de acercamiento fino, en hercios (Hz). Funciona en un rango de [1,60].

**FC_inicio:** BOOL. Final de carrera de inicio. Está a 1 cuando se activa el sensor y 0 cuando no detecta nada.

**FC_fin:** BOOL. Final de carrera de final. Está a 1 cuando se activa el sensor y 0 cuando no detecta nada.

**SensorHome:** BOOL. Variable asociada al sensor de posición empleado para la operación de '*homing*'.

**STOP:** BOOL. Parada de emergencia. A nivel alto, el motor se para de inmediato. Una vez vuelva a 0, el motor retomará su funcionamiento desde la posición en la que quedó.

**FrenoEnReposo:** BOOL. Variable digital que mantiene el motor habilitado incluso una vez alcanzada la referencia (como si de un freno se tratase) de modo que no se pierdan pasos por el efecto de la gravedad. Cuando esté a 1, el freno se activa, energizando la bobina a través de la variable de salida EN_Driver.

**Argumento de salida:**

**EN_Driver:** BOOL. Señal de habilitación del motor. A 1 está el motor habilitado (bobinas energizadas) y a 0 está deshabilitado.

**DIR_Driver:** BOOL. Señal que marca la dirección de giro del motor. A 0, el motor avanza a posiciones positivas (y a posiciones negativas cuando esté a 1).

**STEP _Driver:** BOOL. Señal de pulsos que recibe el motor a la frecuencia estipulada, marcando la velocidad de giro.

**ESTADO_ERROR:** INT. Variable de información para el usuario. Puede tomar los siguientes valores:

- 0 si el motor está en reposo.
- 1 si el motor está en movimiento.
- 2 motor en parada de emergencia.
- 3 frecuencia introducida fuera de rango.
- 4 avance introducido de valor nulo.
- 10 dispositivo en posible fallo. Algún sensor activado (SensorHome o finales de carrera) mientras HomeOK esté activo.
- 11 dispositivo fuera del segmento de trabajo [Home, Carrera].

**Argumentos de entrada/salida:**

**PosicionVirtual:** DINT. Contiene la posición actual en la que se encuentra el motor, referida en unidades de ingeniería. También es señal de entrada ya que nos permite sobrescribir una posición distinta si lo elegimos así.

**HomeOK:** BOOL. Variable digital que indica si el dispositivo ha realizado anteriormente la maniobra de calibración. Se puede forzar a 0 para que el dispositivo ejecute la tarea de 'homing'.

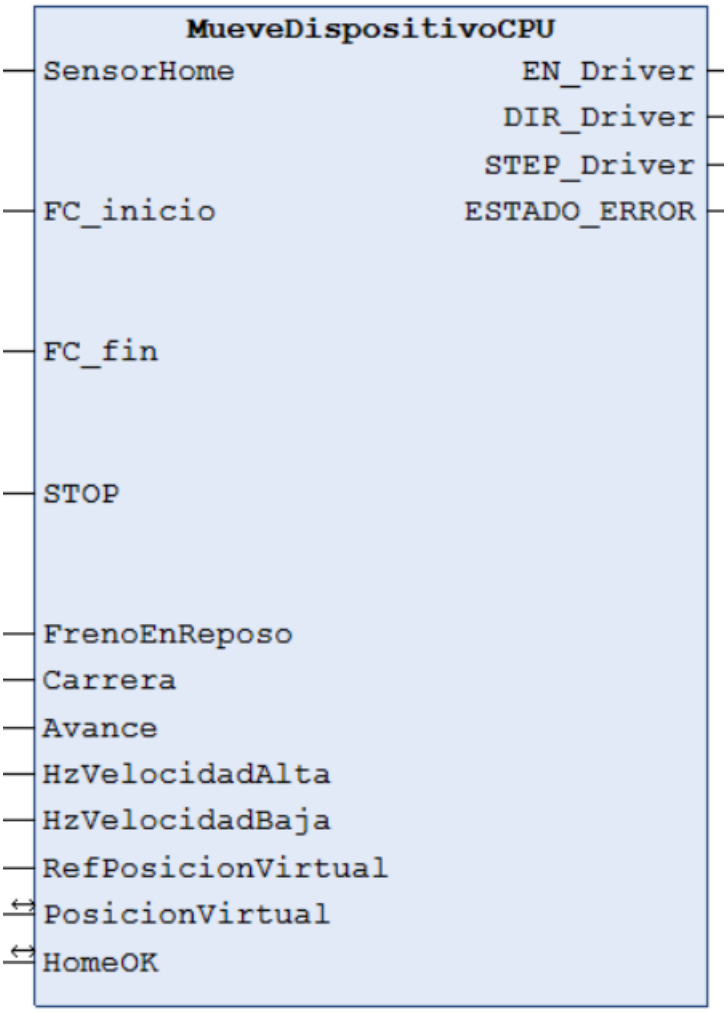


Figura 2-7. Bloque funcional MueveDispositivoCPU

### 2.4.3. Funcionamiento

Esta función tiene un comportamiento con estados y transiciones, en el que el estado de reposo permite un movimiento libre según las referencias dadas y los otros estados constituyen el proceso de calibración del dispositivo. Dicho proceso se inicia cuando la variable HomeOK se pone a 0. A continuación, el "homing" sucede en este orden.

- Desplazamiento a velocidad alta en sentido positivo hasta recorrer una décima parte de la carrera o bien, si sucediese antes, hasta la activación del final de carrera final.
- Desplazamiento a alta velocidad en sentido negativo hasta que se active el sensor SensorHome.
- Desplazamiento a baja velocidad en sentido positivo hasta que se desactive el sensor SensorHome.
- Se reinicia la posición actual como posición Home y se vuelve a establecer HomeOK a 1 para marcar el dispositivo como calibrado.

El diagrama de estados equivalente tendría este aspecto:

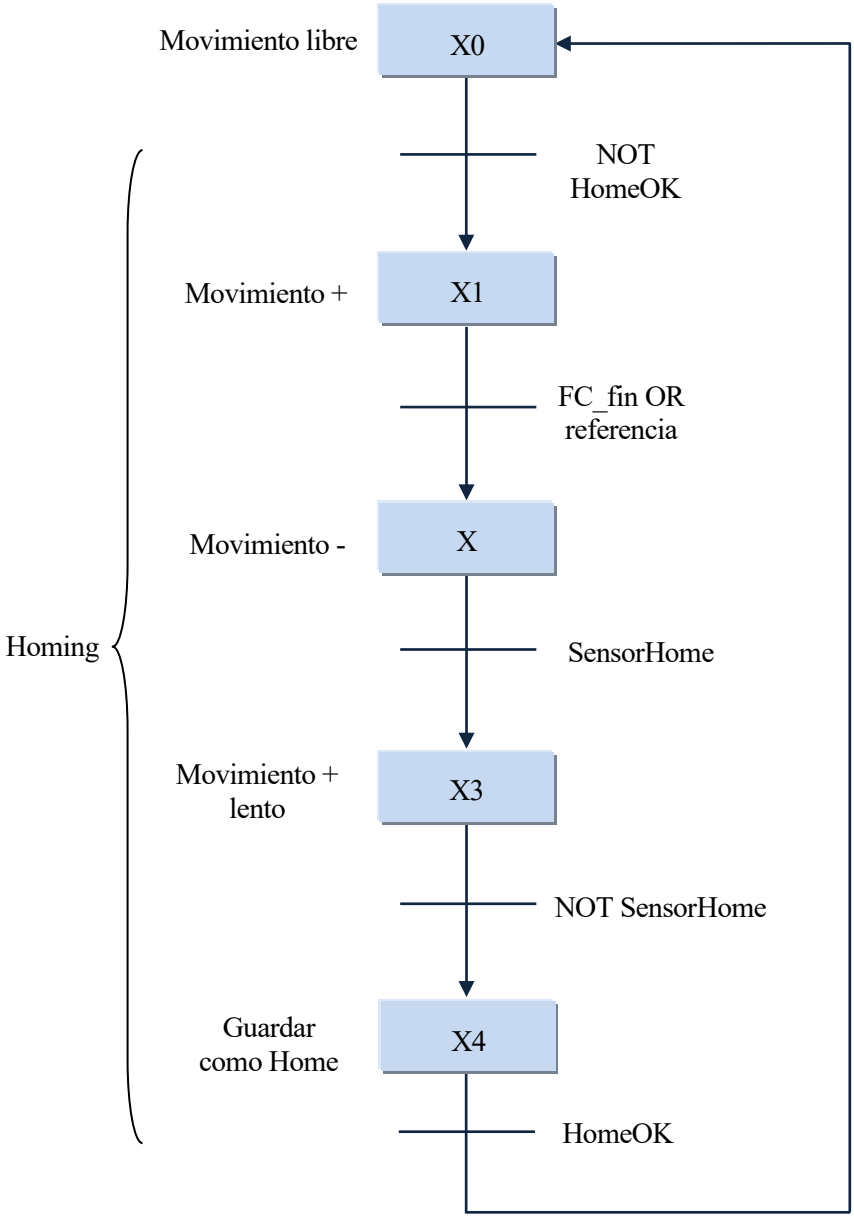


Figura 2-8. Diagrama de estados de MueveDispositivoCPU

El bloque está pensado para que la primera tarea que realice el dispositivo, tan pronto como se ejecute por primera vez, sea realizar la maniobra de calibración, sin tener que solicitarlo poniendo HomeOK a 0. De este modo, el dispositivo quedará calibrado y listo a la espera de recibir una referencia en posición. Si se representase en un diagrama temporal esta operación, para HzVelocidadAlta=10, HzVelocidadBaja=5, Avance=10, Carrera=100, FrenoEnReposo=1 y partiendo de la posición inicial PosVirtual=70, se tendría:

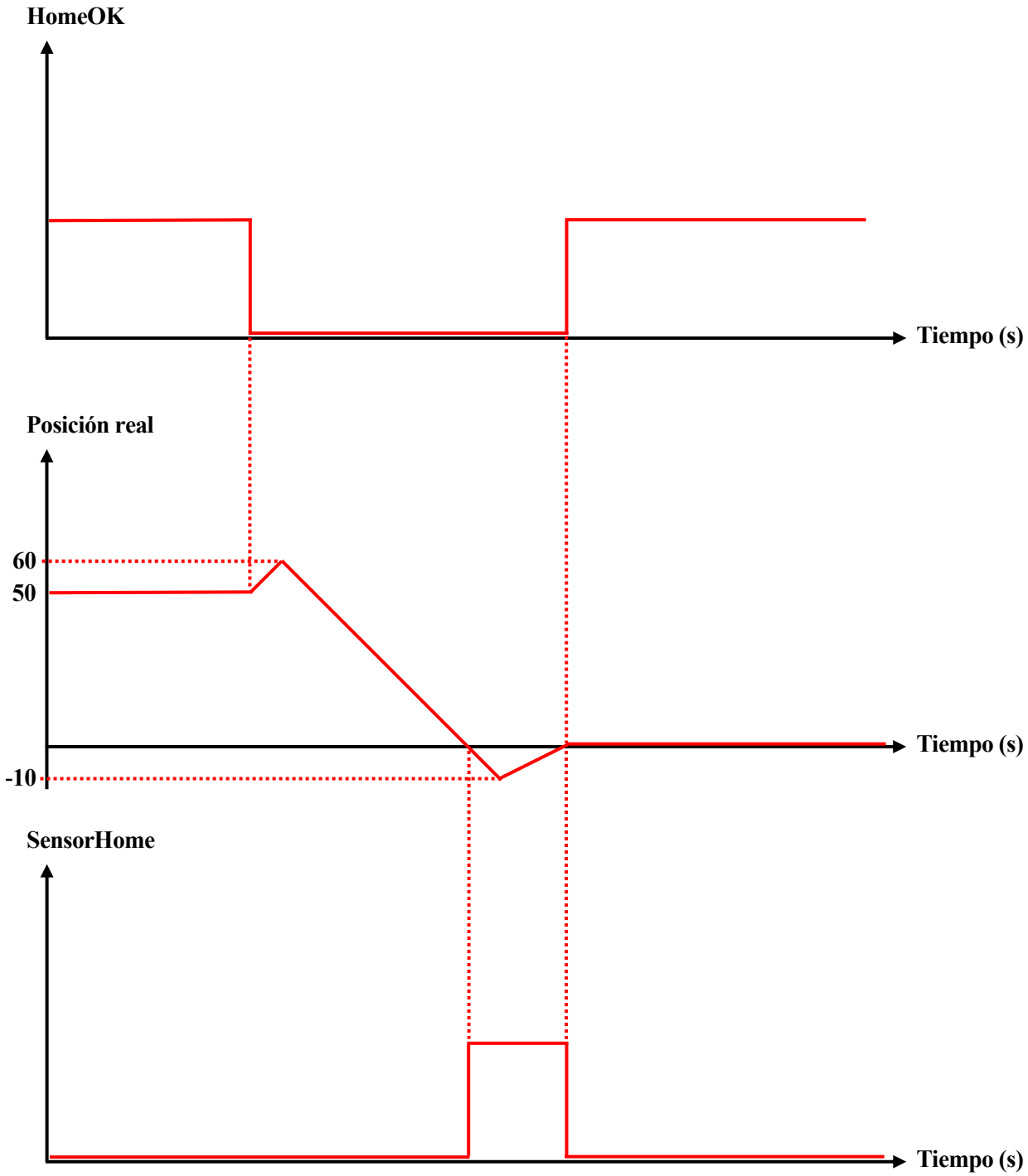


Figura 2-9. Cronograma MueveDispositivoCPU

## 2.5. OperaPinzaCPU

### 2.5.1. Introducción

Este bloque define las órdenes fundamentales de la operación de una pinza neumática genérica. Es el último bloque funcional básico de la lista. Es independiente a los bloques visto hasta ahora, luego su código de automatización está constituido partiendo de cero, con sus sensores y actuadores propios.

### 2.5.2. Variables del bloque funcional

**Argumentos de entrada**

**PresionAire:** BOOL. Variable asociada al sensor de presión, indicativo de que el aire a presión está en condiciones de operar la pinza.

**SensorPinzaAbierta:** BOOL. Sensor del estado de la pinza. A nivel alto cuando la pinza se encuentra completamente abierta.

**SensorPinzaCerrada:** BOOL. Sensor del estado de la pinza. A nivel alto cuando la pinza se encuentra completamente cerrada.

**STOP:** BOOL. Parada de emergencia. A nivel alto, la pinza se cerrará automáticamente.

**CerrarPinza:** BOOL. Orden externa para cerrar la pinza.

**Argumentos de salida**

**Electrovalvula:** BOOL. Salida encargada de controlar el actuador de la electrovávula que opera la pinza. A nivel alto, la pinza se cerrará.

**ESTADO:** INT. Variable de información para el usuario. Puede tomar los siguientes valores:

- -2 la pinza no puede cerrar
- -1 el aire a presión no está en las condiciones adecuadas.
- 1 pinza completamente abierta.
- 2 pinza completamente cerrada.
- 3 pinza en posición intermedia.

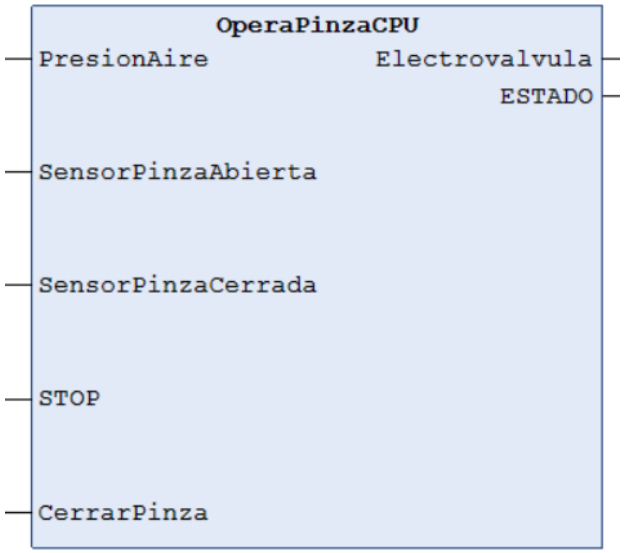


Figura 2-10. Bloque funcional OperaPinzaCPU

### 2.5.3. Funcionamiento

El funcionamiento del mecanismo de pinza se basa en la de un actuador todo o nada. Cuando se recibe una orden de cerrar la pinza a través de la entrada CerrarPinza, siempre y cuando PresionAire esté en valor alto, la salida Electrovalvula se activará. Esta salida será la que controle el propio actuador.

En cuanto a los sensores, como se ha descrito, se activan en los estados extremos, o bien completamente abierto o bien completamente cerrado. Dado que el movimiento de las garras tiene cierto recorrido, se conoce si las garras se encuentran en una posición intermedia. Teniendo esto en cuenta, cuando la pinza cierre para agarrar una pieza, lo habrá hecho correctamente si la pinza no cierra por completo. Del mismo modo es posible saber si no se ha podido agarrar la pieza o incluso si la pieza se desprende accidentalmente durante durante su transporte.

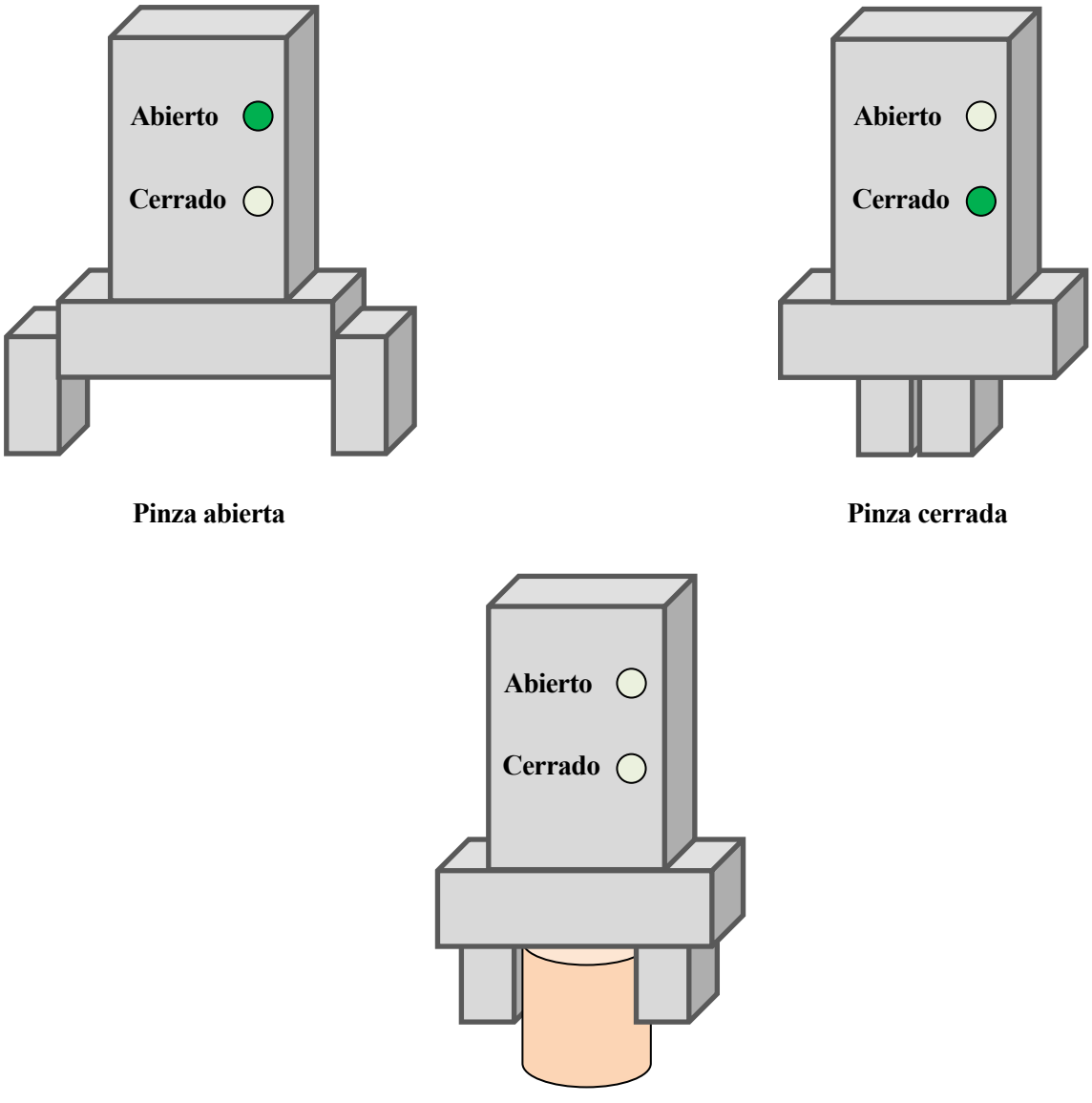


Figura 2-11. Esquema de los estados de la pinza

# 3 BLOQUES FUNCIONALES DE ESTACIÓN

En este capítulo se tratará la automatización de las distintas estaciones que incluidas en la librería. Para ello, se hará una introducción de la máquina y se explicarán las pautas seguidas para el control autónomo de cada una de ellas, además de las posibilidades que ofrecen.

Se verán los bloques de control:

- AlimentadorPiezasCPU.
- PorticoCPU.
- PuenteGruaCPU.
- ScaraCPU.
- ScorbotCPU.
- AlmacenPaletsCPU.
- AlimentadorBandejasCPU.

Adicionalmente, se tratarán los simuladores de estas estaciones, que incluyen sus pantallas de operador con su HMI correspondiente y sus bloques funcionales independientes para el control del simulador.

En el bloque de control se definen normas de funcionamiento, en los cuales se decide la respuesta de salida (en su mayoría actuadores) en función del valor de una serie de entradas (en su mayoría sensores). Sin embargo, a la hora de diseñar un simulador virtual, se requiere emular el comportamiento de muchas de estas entradas para asemejarlo al sistema real, a falta de poder tomar las señales de entrada de los sensores reales. Además, también se necesita reproducir las acciones de cada máquina. Es este el motivo por el que se han diseñado bloques funcionales para gestionar el simulador. Estos son:

- AlimentadorPiezasDEMO.
- PorticoDEMO.
- PuenteGruaDEMO.
- ScaraDEMO.
- ScorbotDEMO.
- AlmacenPaletsDEMO.
- AlimentadorBandejasDEMO.

## 3.1 AlimentadorPiezasCPU

### 3.1.1 Introducción

El bloque funcional AlimentadorPiezasCPU constituye el funcionamiento completo de una máquina con un motor paso a paso que almacena y extrae piezas en una pila LIFO. Partiendo del modelo de MueveDispositivoCPU, se requiere de un sensor adicional para detectar las piezas, situado justo en la entrada del alimentador. Funciona como sensor normalmente activo, señalizando la presencia de pieza a través de la interrupción de su señal.

La disposición se la máquina sería la de una plataforma en la base de la pila conectada al motor PaP capaz de describir una trayectoria vertical. Por petición se pueden almacenar o retirar piezas de la pila con el desplazamiento de la plataforma. Por otra parte, la tarea de depositar las piezas y retirarlas no está contemplada en este bloque, siendo responsabilidad de otra estación independiente.

Figura 3-1. Alimentador de piezas – Fotografía de laboratorio

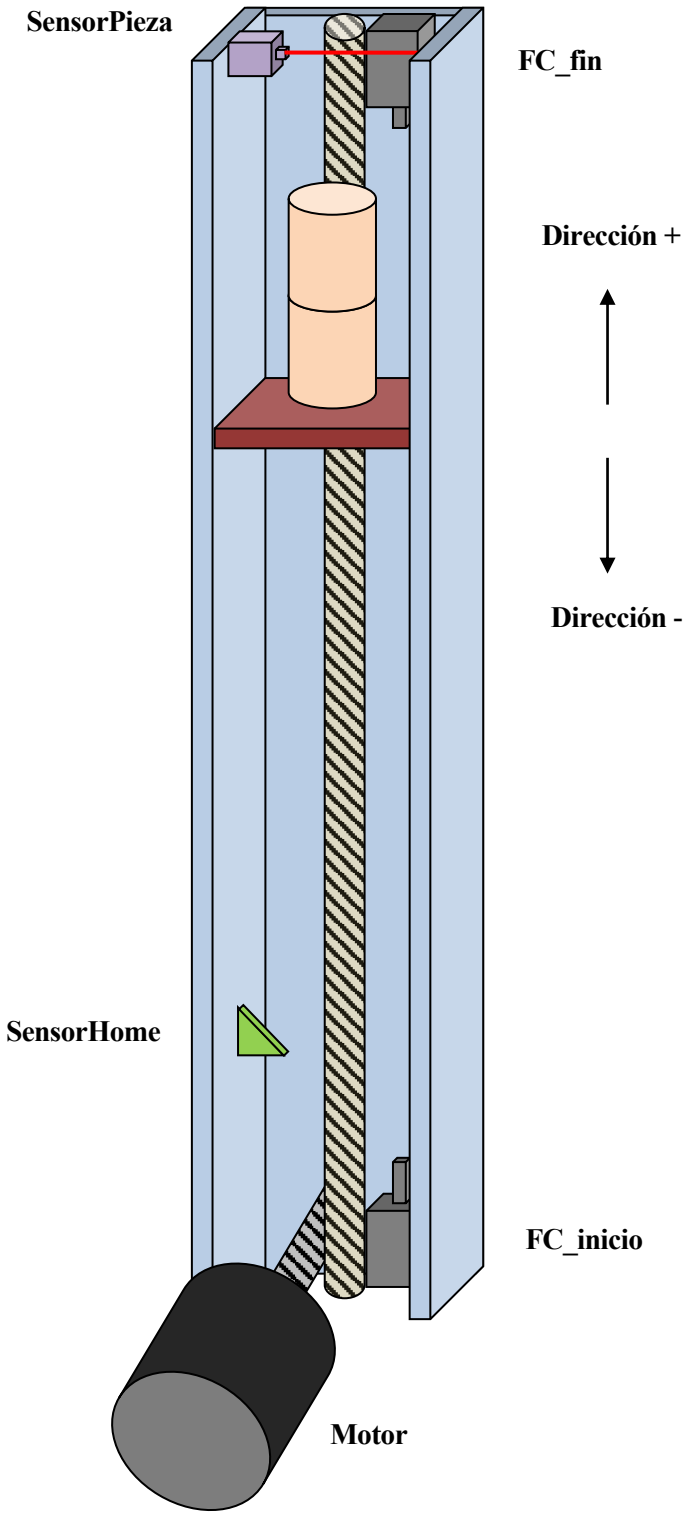


Figura 3-2. Esquema del alimentador de piezas

### 3.1.2 Variables del bloque funcional

**Argumentos de entrada:**

**Avance:** DINT. Relación entre las unidades de ingeniería y los pasos del motor. Las unidades empleadas se escogerán libremente, atendiendo a la naturaleza del movimiento final efectuado. Por ejemplo, para un motor cuyo propósito sea levantar una pieza, se puede tomar como avance el desplazamiento que realiza dicha pieza por cada pulso de motor.

**Carrera:** DINT. Distancia en unidades de ingeniería (referidas al avance) que delimita el segmento de operación del motor.

**HzVelocidadAlta:** INT. Frecuencia a la que queremos que funcione el motor en las tareas de posicionamiento rápido, en hercios (Hz). Funciona en un rango de [1,60].

**HzVelocidadBaja:** INT. Frecuencia a la que queremos que funcione el motor en las tareas de acercamiento fino, en hercios (Hz). Funciona en un rango de [1,60].

**FC_inicio:** BOOL. Final de carrera de inicio. Está a 1 cuando se activa el sensor y 0 cuando no detecta nada.

**FC_fin:** BOOL. Final de carrera de final. Está a 1 cuando se activa el sensor y 0 cuando no detecta nada.

**SensorHome:** BOOL. Variable asociada al sensor de posición empleado para la operación de '*homing*'.

**SensorPieza:** BOOL. Variable asociada al detector de piezas.

**AlturaPieza:** DINT. Variable que especifica la altura de las piezas.

**STOP:** BOOL. Parada de emergencia. Cuando se ponga a 1, se para el motor de forma inmediata y cuando vuelva a 0, se retomará en la posición que quedase, reanudando la operación.

**FrenoEnReposo:** BOOL. Variable digital que mantiene el motor en la posición que quede una vez alcanzada la referencia (como si de un freno se tratase) de modo que no se pierdan pasos por el efecto de la gravedad. Cuando esté a 1, el freno se activa, energizando la bobina a través de la variable de salida EN_Driver.

**Argumentos de salida:**

**EN_Driver:** BOOL. Señal de habilitación del motor. A 1 está el motor habilitado (bobinas energizadas) y a 0 está deshabilitado.

**DIR_Driver:** BOOL. Señal que marca la dirección de giro del motor. A 0, el motor avanza a posiciones positivas (y a posiciones negativas cuando esté a 1).

**STEP _Driver:** BOOL. Señal de pulsos que recibe el motor a la frecuencia estipulada, marcando la velocidad de giro.

**ESTADO_ERROR:** INT. Variable de información para el usuario. Puede tomar los siguientes valores:

- 0 si el motor está en reposo.
- 1 si el motor está en movimiento.
- 2 motor en parada de emergencia.
- 3 frecuencia introducida fuera de rango.
- 4 avance introducido de valor nulo.
- 5 almacén lleno.
- 6 almacén vacío.
- 7 orden de almacenar sin pieza colocada.

- 8 orden de extraer sin haber retirado la pieza superior.
- 10 dispositivo en posible fallo. Algún sensor activado (SensorHome o finales de carrera) mientras HomeOK esté activo.
- 11 dispositivo fuera del segmento de trabajo [Home, Carrera].

**Argumentos de entrada/salida:**

**ExtraePieza:** BOOL. Orden para extraer pieza.

**AlmacenaPieza:** BOOL. Orden para almacenar pieza.

**AutoTest:** BOOL. Orden para realizar una prueba de calibración y conteo de piezas en pila.

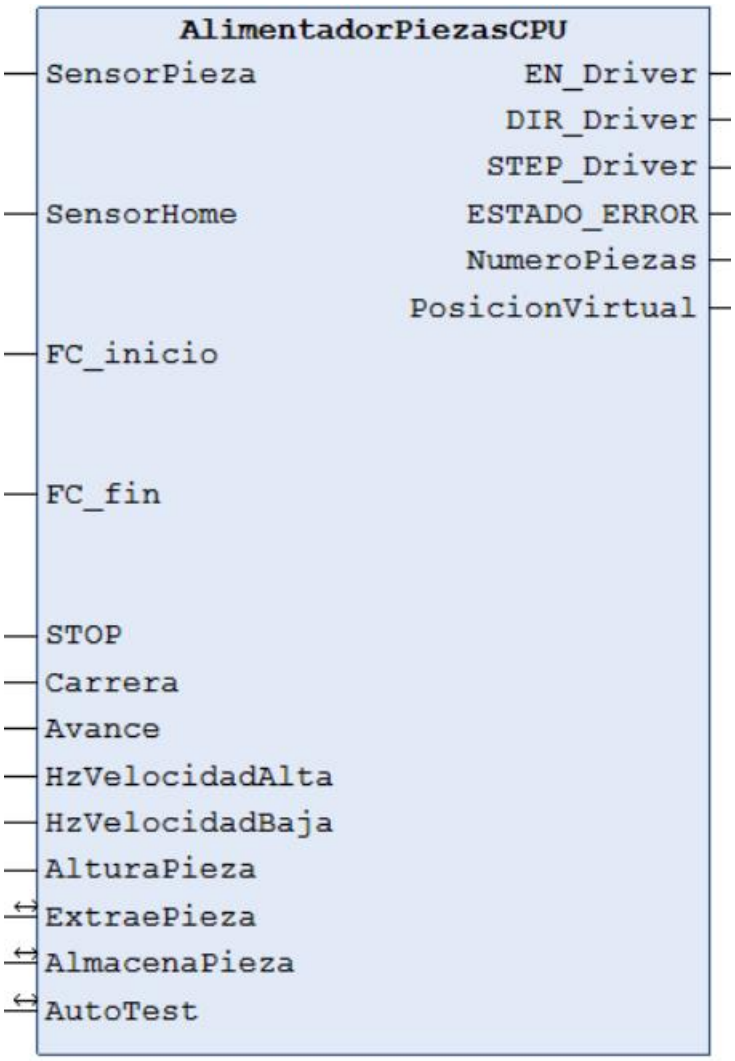


Figura 3-3. Bloque funcional AlimentadorPiezasCPU

### 3.1.3 Funcionamiento

La automatización de esta máquina está igualmente basada en estados y transiciones. Desde un estado de reposo, puede recibir tres tipos de órdenes, vistos anteriormente.

**AutoTest**

- La operación de AutoTest comienza cuando dicha variable es desactivada externamente. Primeramente, realiza una prueba de calibración del motor PaP.
- Una vez terminada, se eleva el soporte de la pila hasta opturar el sensor SensorPieza, previsiblemente por la pieza más alta en la pila.
- A continuación, baja lentamente hasta que la última pieza deje de interrumpir la señal de SensorPieza, dejando la pila de piezas justo al límite del alimentador.
- Finalmente, conociendo la posición actual en la que se encuentra el soporte y la altura de cada pieza, se puede inferir el número de piezas que hay en pila. Con esto se da por terminada la operación y se activa AutoTest antes de volver al estado inicial.

**AlmacenarPieza**

- La operación AlmacenarPieza solo comenzará bajo dos condiciones, que haya sitio para más piezas en el alimentador, y que se encuentre SensorPieza interrumpido por una pieza, lo que indicaría que está esperando ser almacenada.
- Cumplidas las condiciones, con la activación de AlmacenarPieza, el soporte desciende la longitud justa para almacenar la pieza, dada por su altura. Hecho esto se actualiza el conteo de piezas, se desactiva la orden y se vuelve al estado inicial.

**ExtraerPieza**

- La operación ExtraerPieza solo comenzará bajo dos condiciones, que haya piezas en el alimentador y que SensorPieza no esté interrumpido, lo que indicaría que no hay piezas fuera del alimentador estorbando, posiblemente esperando a ser retiradas por otra estación.
- Cumplidas las condiciones, con la activación de ExtraerPieza, el soporte asciende la longitud justa para extraer la pieza superior de la pila, dada por su altura. Hecho esto se actualiza el conteo de piezas, se desactiva la orden y se vuelve al estado inicial.

Al igual que sucedía con MueveDispositivoCPU, el AutoTest comienza automáticamente con la primera ejecución, con el fin de dejar la máquina calibrada y en posición para estar operativa.

A continuación, se muestra un esquema simplificado del diagrama de estados equivalente al bloque AlimentadorPiezasCPU, según lo recientemente expuesto.

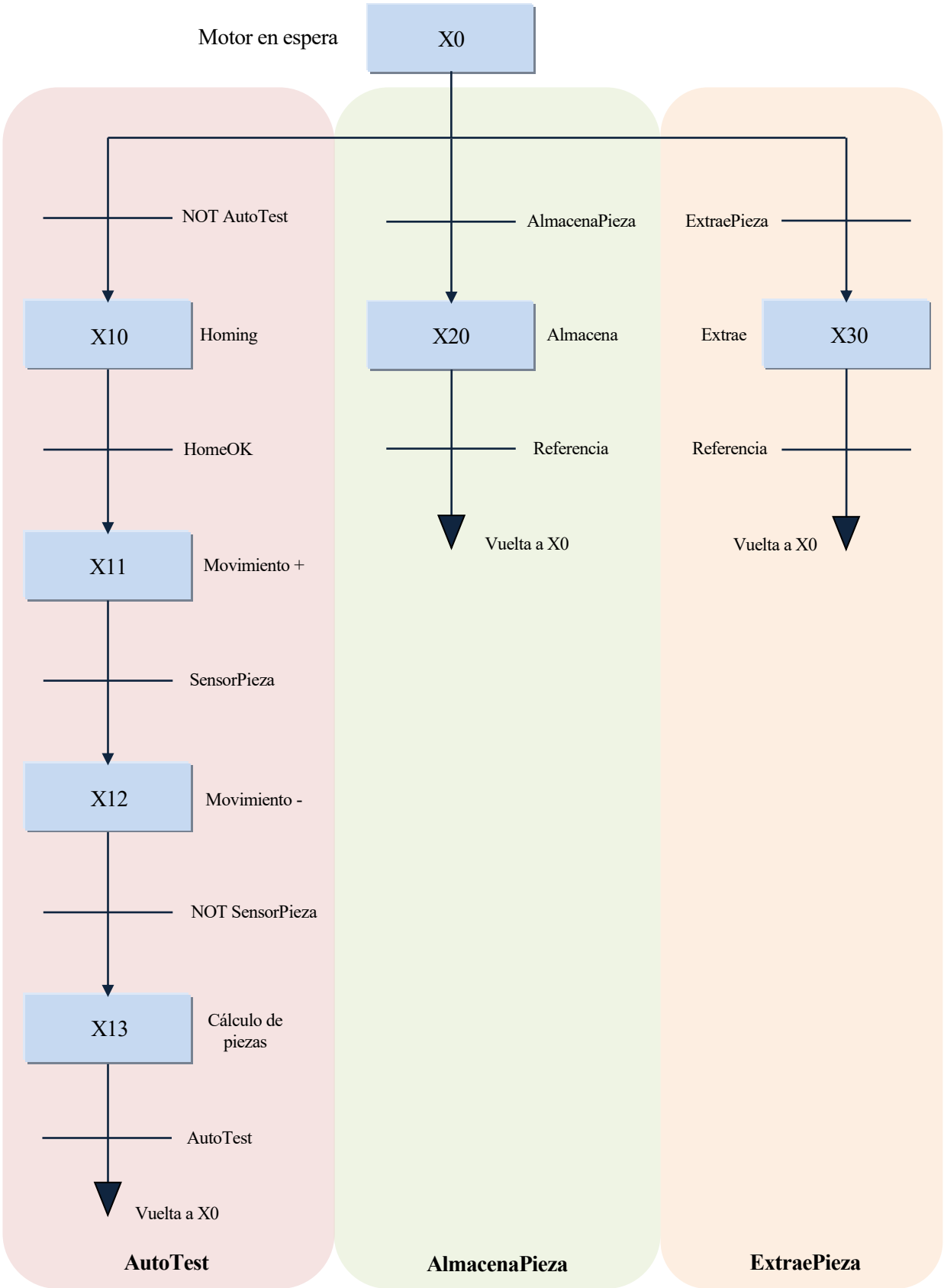


Figura 3-4. Diagrama de estados simplificado de AlimentadorPiezasCPU

### 3.1.4 Simulación

El bloque AlimentadoPiezasDEMO ha de encargarse de gestionar la visualización y activación de los sensores SensorHome y SensorPieza. También de la visualización del número de piezas en pantalla. Para ello empleará datos provenientes del controlador como la posición y algunos parámetros geométricos, y gestionará órdenes dadas desde la propia interfaz del simulador.

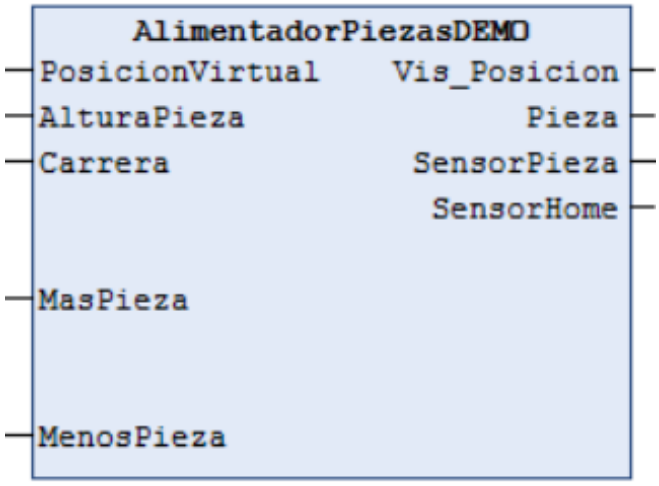


Figura 3-5. Bloque funcional AlimentadorPiezasDEMO

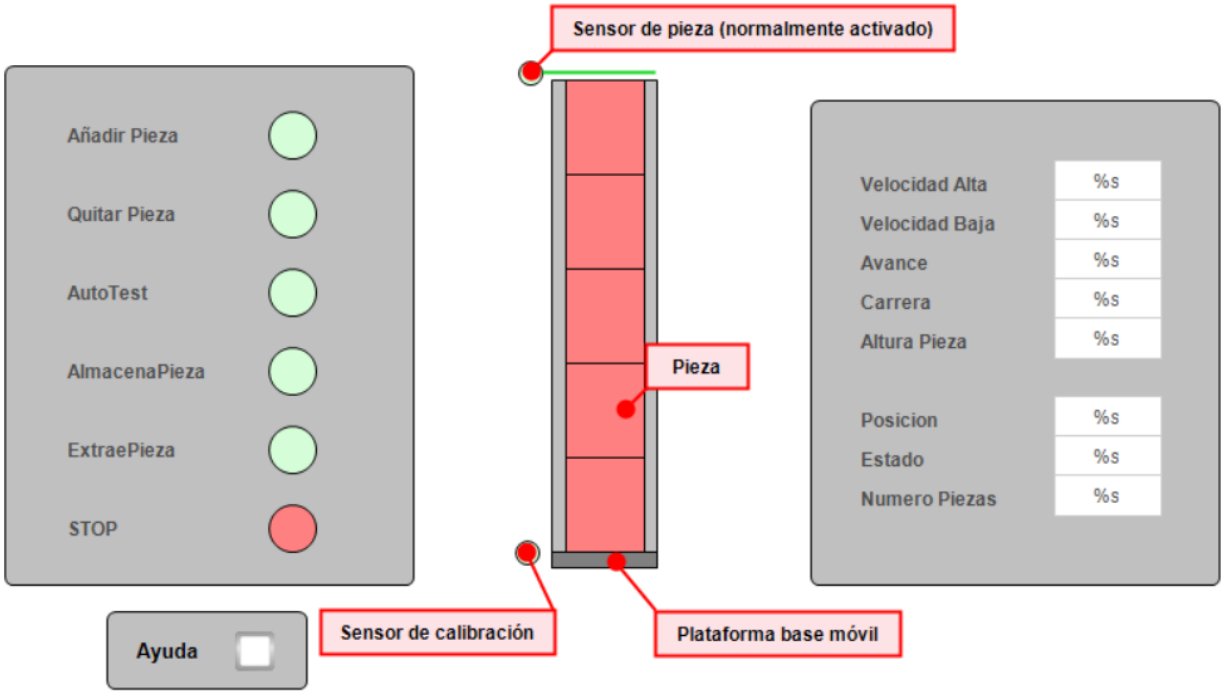


Figura 3-6. Interfaz del simulador del alimentado de piezas

## 3.2 PorticoCPU

### 3.2.1 Introducción

Este bloque funcional controla una estación constituida por una pinza instalada en una estructura de pórtico, cuyo fin es transportar piezas dentro de su entorno de trabajo. Para esta tarea, hará uso de los bloques funcionales OperaPinzaCPU para el control de la pinza y de dos bloques MueveDispositivoCPU; uno para el deplazamiento horizontal a través del pórtivo y otro para el deplazamiento vertical, permitiendo subir y bajar la pinza.

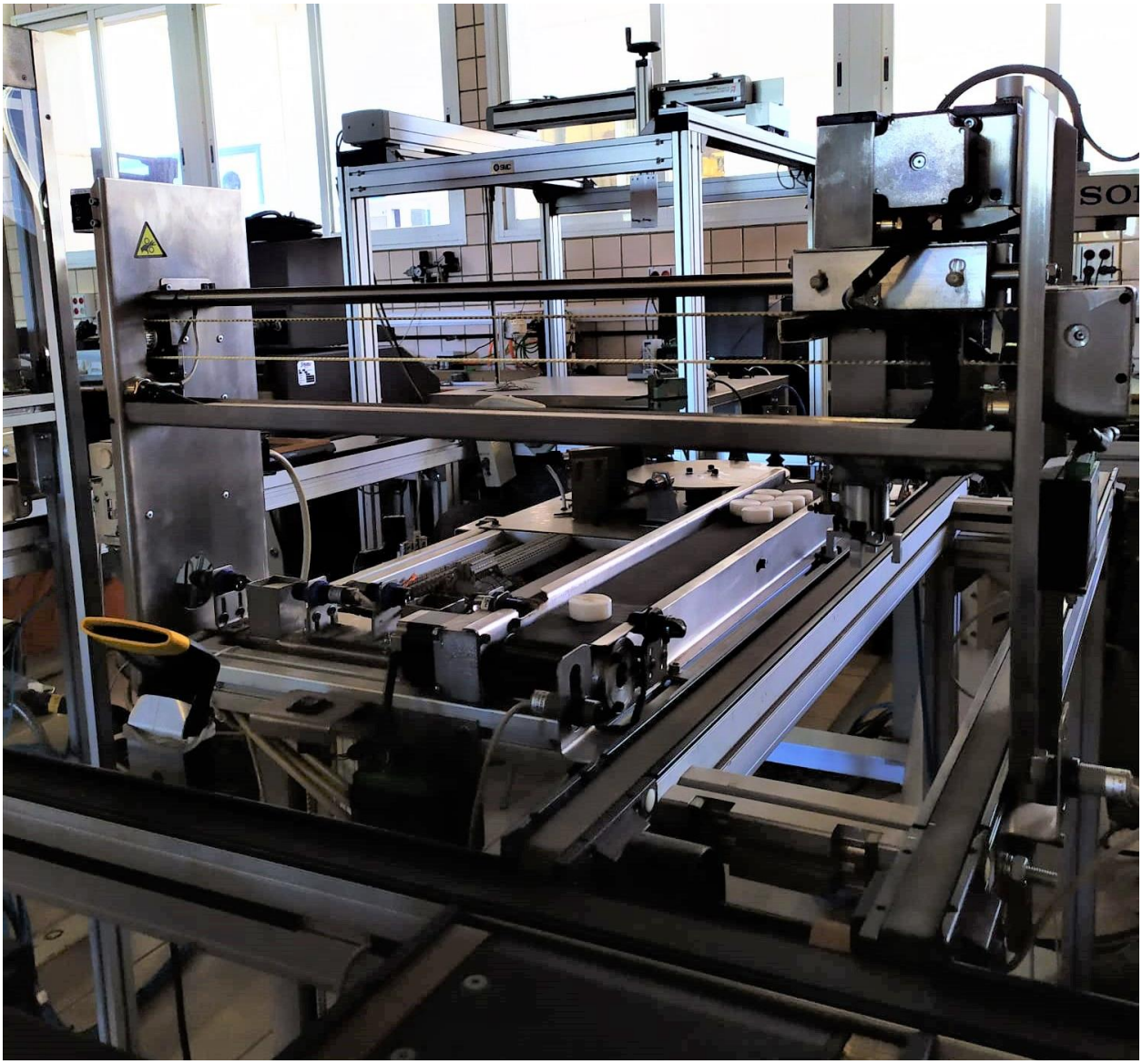

Figura 3-7. Pórtico - Fotgrafía de laboratorio

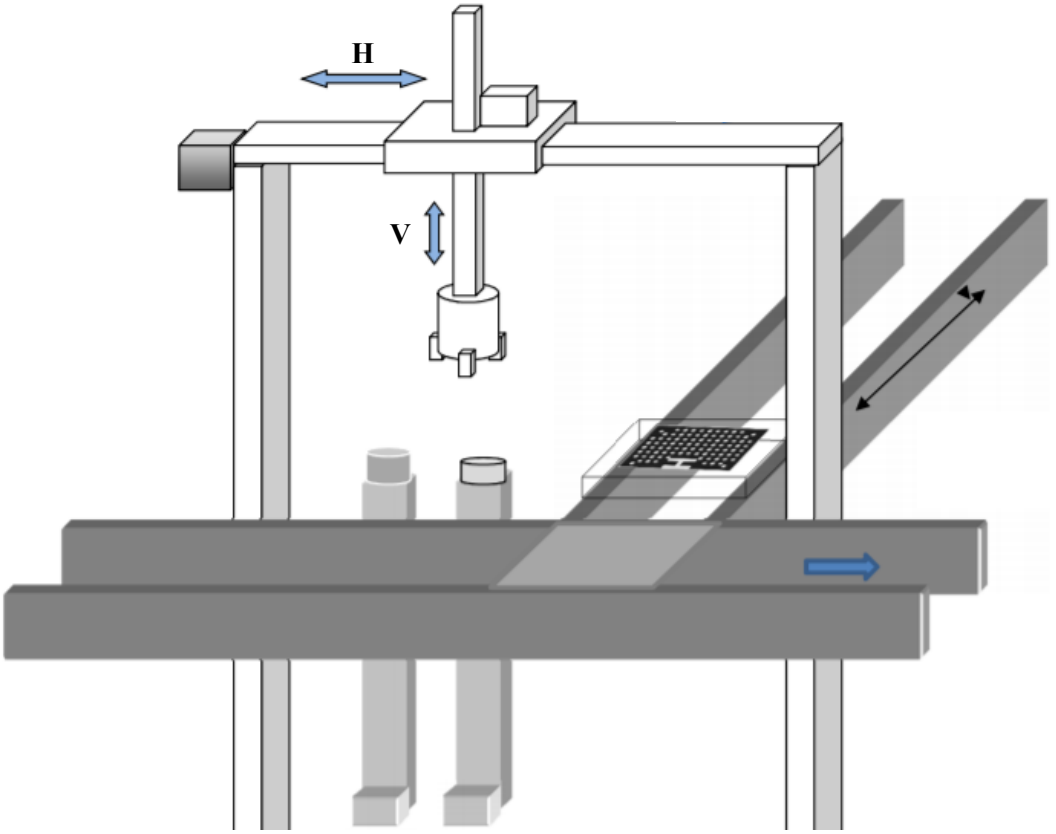


Figura 3-8. Esquema del pórtico

### 3.2.2 Variables del bloque funcional

**Argumentos de entrada:**

**SensorHome_H:** BOOL. Sensor Home del motor PaP encargado del desplazamiento horizontal.

**FC_inicio_H:** BOOL. Final de carrera de inicio del motor PaP encargado del desplazamiento horizontal.

**FC_fin_H:** BOOL. Final de carrera de final del motor PaP encargado del desplazamiento horizontal.

**SensorHome_V:** BOOL. Sensor Home del motor PaP encargado del desplazamiento vertical.

**FC_inicio_V:** BOOL. Final de carrera de inicio del motor PaP encargado del desplazamiento vertical.

**FC_fin_V:** BOOL. Final de carrera de final del motor PaP encargado del desplazamiento vertical.

**PresionAire:** BOOL. Variable asociada al sensor de presión, indicativo de que el aire a presión está en condiciones de operar la pinza.

**SensorPinzaAbierta:** BOOL. Sensor del estado de la pinza. A nivel alto cuando la pinza se encuentra completamente abierta.

**SensorPinzaCerrada:** BOOL. Sensor del estado de la pinza. A nivel alto cuando la pinza se encuentra completamente cerrada.

**Carrera_H:** DINT. Carrera del motor PaP encargado del desplazamiento horizontal.

**Avance_H:** DINT. Avance del motor PaP encargado del desplazamiento horizontal.

**Carrera_V:** DINT. Carrera del motor PaP encargado del desplazamiento vertical.

**Avance_V:** DINT. Avance del motor PaP encargado del desplazamiento vertical.

**HzVelocidadAlta:** INT. Frecuencia a la que queremos que funcione el motor en las tareas de posicionamiento rápido, en hercios (Hz). Funciona en un rango de [1,60].

**HzVelocidadBaja:** INT. Frecuencia a la que queremos que funcione el motor en las tareas de acercamiento fino, en hercios (Hz). Funciona en un rango de [1,60].

**STOP:** BOOL. Parada de emergencia. A nivel alto, los motores PaP se detendrán y la pinza se cerrará.

**Origen_X:** DINT. Posición de recogida de la pieza en la coordenada horizontal según la referencia dada por el motor PaP correspondiente.

**Origen _Y:** DINT. Posición de recogida de la pieza en la coordenada vertical según la referencia dada por el motor PaP correspondiente.

**Destino_X:** DINT. Posición de entrega de la pieza en la coordenada horizontal según la referencia dada por el motor PaP correspondiente.

**Destino _Y:** DINT. Posición de entrega de la pieza en la coordenada vertical según la referencia dada por el motor PaP correspondiente.

**Argumentos de salida:**

**EN_Driver_H:** BOOL. Habilitación del motor PaP encargado del deplazamiento horizontal.

**DIR_Driver_H:** BOOL. Dirección de giro del motor PaP encargado del deplazamiento horizontal.

**STEP_Driver_H:** BOOL. Señal de pulsos que recibe el motor PaP encargado del deplazamiento horizontal.

**HomeOK_H:** BOOL. Indica el estado de calibración del motor PaP encargado del deplazamiento horizontal.

**EN_Driver_V:** BOOL. Habilitación del motor PaP encargado del deplazamiento vertical.

**DIR_Driver_V:** BOOL. Dirección de giro del motor PaP encargado del deplazamiento vertical.

**STEP_Driver_V:** BOOL. Señal de pulsos que recibe el motor PaP encargado del deplazamiento vertical.

**HomeOK_V:** BOOL. Indica el estado de calibración del motor PaP encargado del deplazamiento vertical.

**Electrovalvula:** BOOL. Salida encargada de controlar el actuador de la electrovávula que opera la pinza. A nivel alto, la pinza se cerrará.

**PosicionPinza_X:** DINT. Posición actual de la pinza en la coordenada horizontal según la referencia dada por el motor PaP correspondiente.

**PosicionPinza_Y:** DINT. Posición actual de la pinza en la coordenada vertical según la referencia dada por el motor PaP correspondiente.

**ESTADO:** INT. Variable de información para el usuario. Puede tomar los siguientes valores:

- -1 máquina no calibrada. Realizando el AutoTest.
- 0 esperando orden.
- 1 máquina en funcionamiento.

**Argumentos de entrada/salida:**

**Ejecutar:** BOOL. Orden para realizar la operación de recogida y entrega de pieza conforme a las posiciones estipuladas.

**AutoTest:** BOOL. Orden para realizar una prueba de calibración de la máquina.

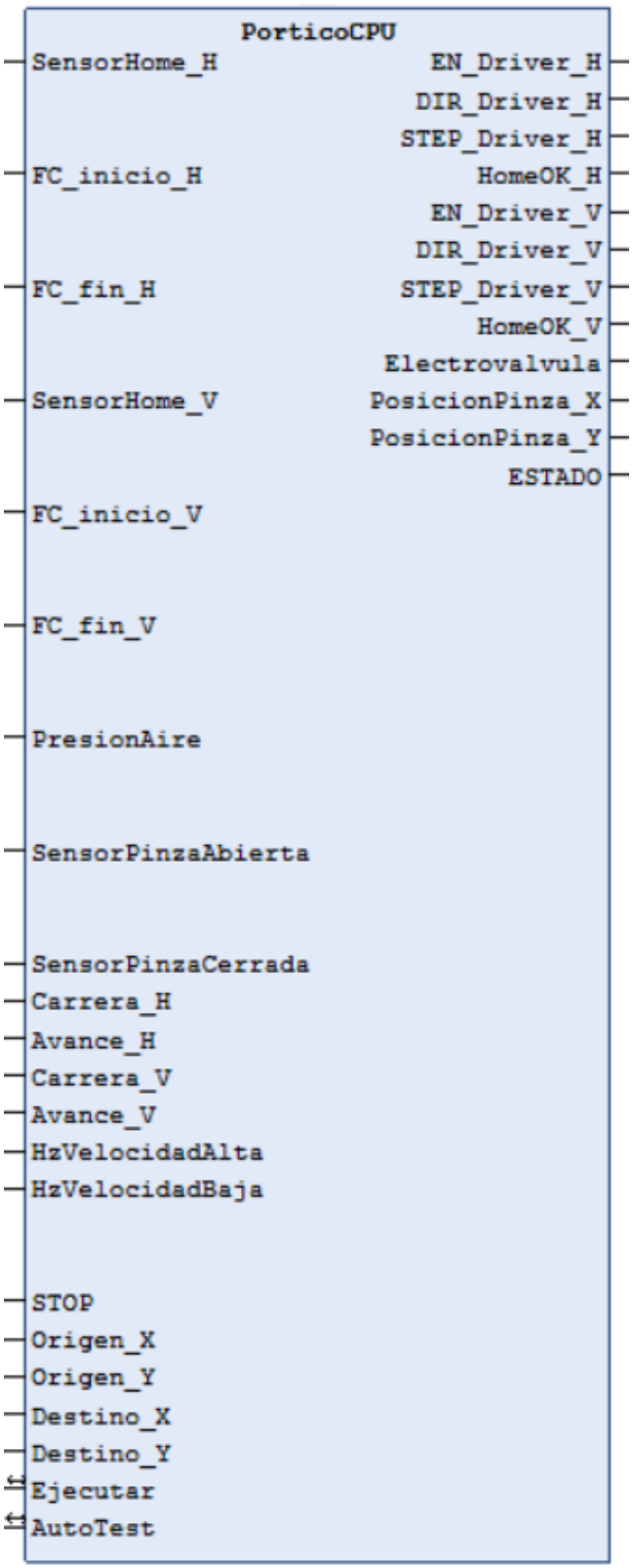


Figura 3-9. Bloque funcional PorticoCPU

### 3.2.3 Funcionamiento

La automatización de esta máquina está igualmente basada en estados y transiciones. Desde un estado de reposo, puede recibir dos tipos de órdenes, vistos anteriormente.

**AutoTest**

- La operación de AutoTest comienza cuando dicha variable es desactivada externamente. Primeramente, realiza una prueba de la pinza, cerrando y abriendo la garra por completo.
- A continuación, realiza una prueba de calibración del motor PaP encargado del desplazamiento horizontal.
- Seguidamente, una prueba de calibración del motor PaP encargado del desplazamiento vertical.
- Finalmente, se acciona el motor vertical para situar la pinza a media altura. Con esto se da por terminada la operación y se activa AutoTest antes de volver al estado inicial.

**Ejecutar**

- Con la activación de Ejecutar, se empieza por llevar a la pinza a la posición Origen. Esto se hace en orden, empezando por el desplazamiento horizontal, seguido del vertical.
- Llegado a la posición Origen, donde se espera que esté la pieza, se da la orden de cerrar la pinza y se eleva a una posición intermedia.
- A continuación, se transporta a la posición Destino, siguiendo el mismo orden en el desplazamiento.
- Llegado a la posición Destino, se ordena abrir la pinza para depositar la pieza en su lugar y se vuelve a elevar a una posición intermedia.
- Hecho esto, se desactiva la orden Ejecutar y se vuelve al estado inicial.

Como se ha visto en los bloques anteriores, el AutoTest comienza automáticamente con la primera ejecución, con el fin de dejar la máquina calibrada y en posición para estar operativa.

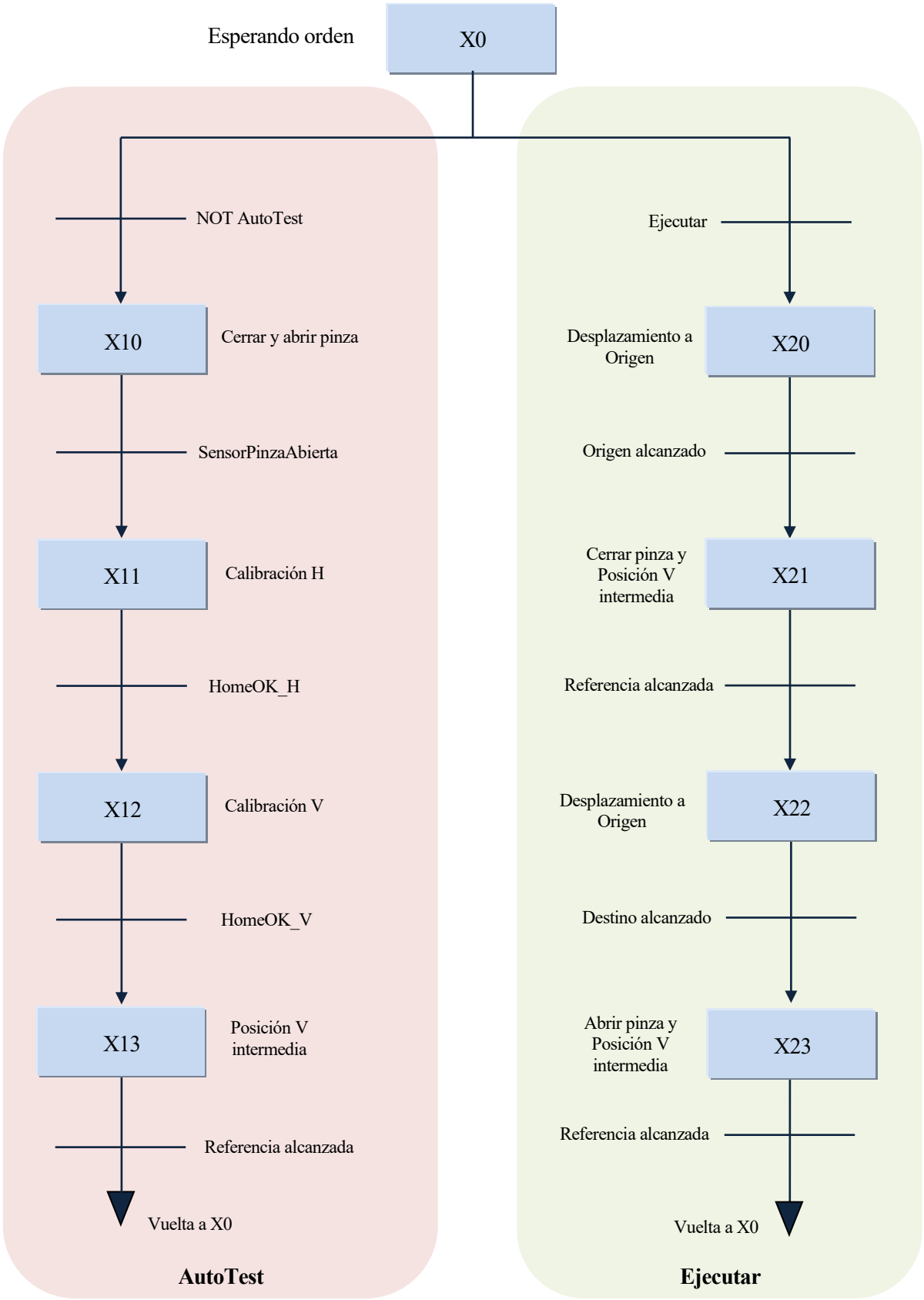


Figura 3-10. Diagrama de estados simplificado de PorticoCPU

### 3.2.4 Simulación

El bloque PorticoDEMO ha de encargarse de gestionar la visualización y activación de los sensores de calibración y la pinza. Para ello empleará datos provenientes del controlador como la posición de los motores PaP y el estado de la pinza.

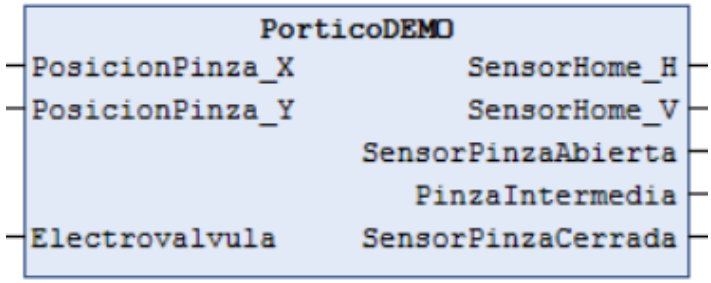


Figura 3-11. Bloque funcional PorticoDEMO

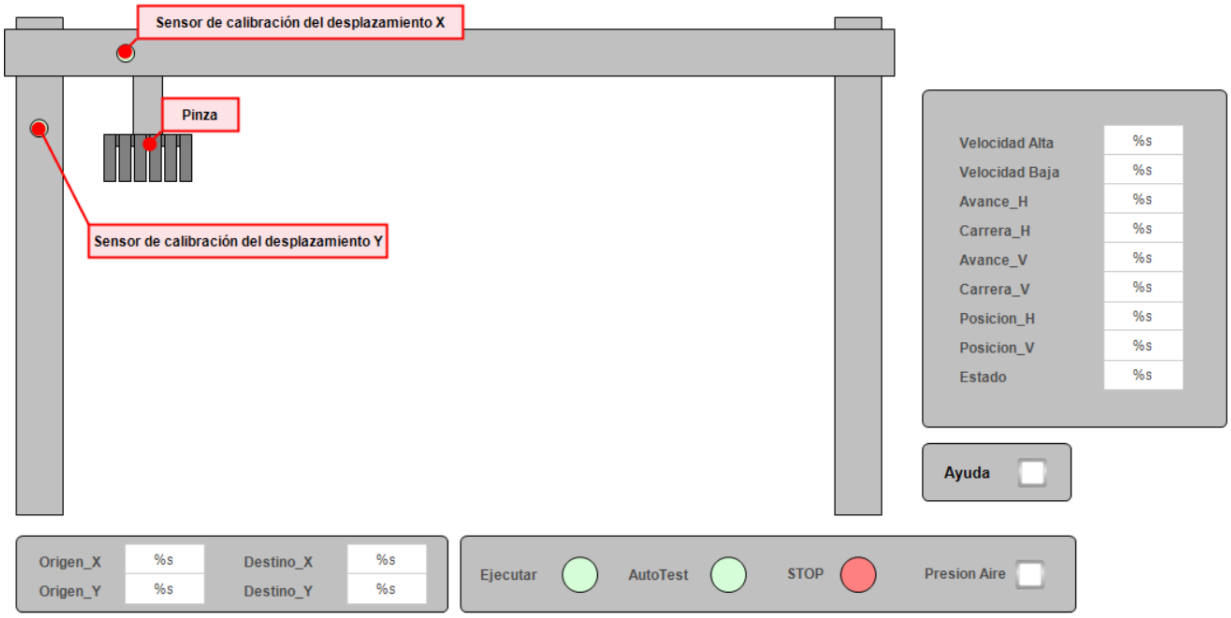


Figura 3-12. Interfaz del simulador del pórtico

## 3.3 PuenteGruaCPU

### 3.3.1 Introducción

Una vez visto el bloque PorticoCPU, el bloque funcional PuenteGruaCPU se puede concebir como su equivalente tridimensional. Si la estructura del pórtico permitía transportar piezas limitando su espació de trabajo a un plano, el puente grúa puede desarrollar la misma tarea en todo un volumen, solo delimitado por la propia estructura y alcance de la máquina. Para conseguir este grado de libertad extra se necesita implementar un motor PaP adicional, luego tendremos un bloque OperaPinza y tres bloques MueveDispositivoCPU.

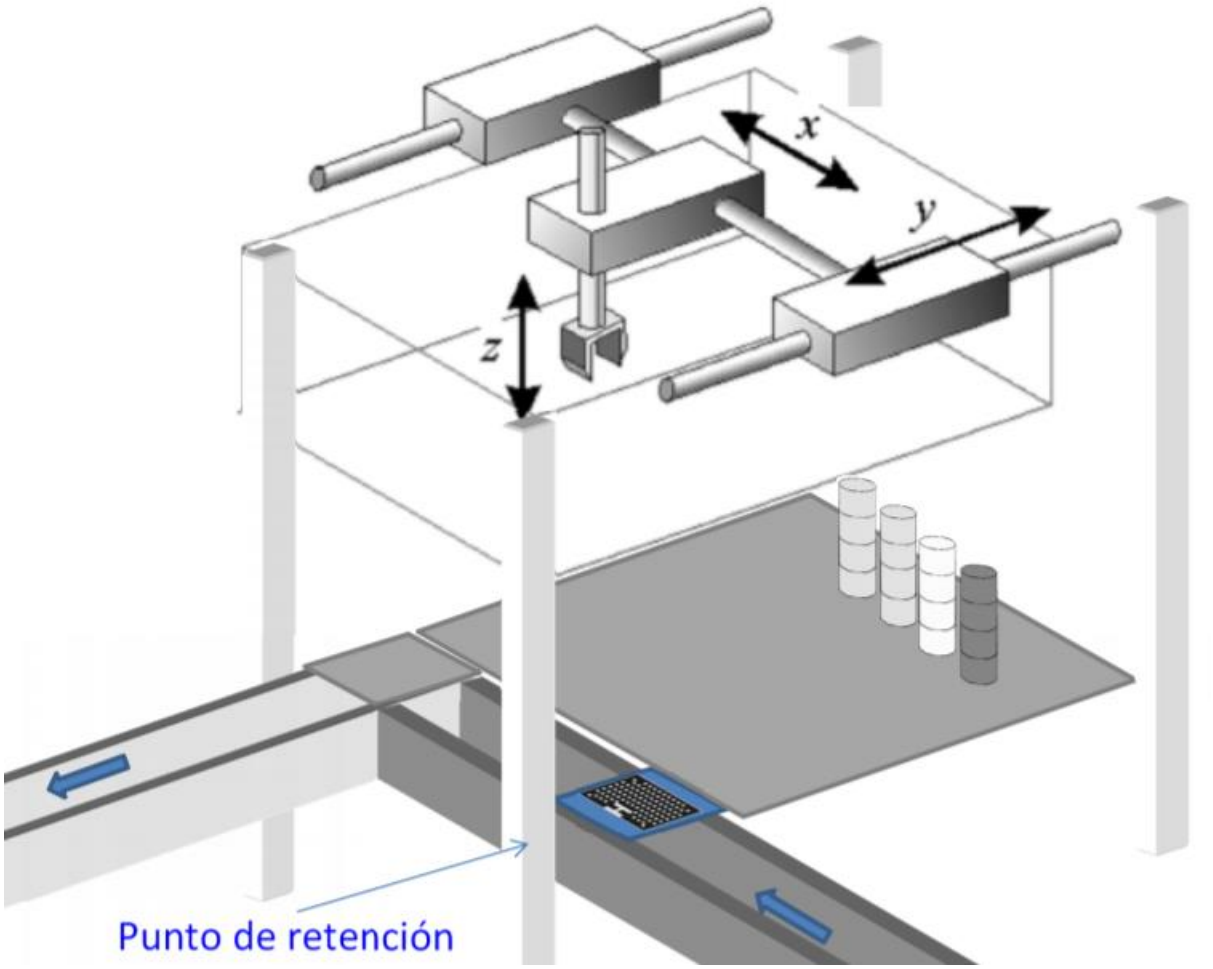


Figura 3-13. Esquema del puente grúa

### 3.3.2 Variables del bloque funcional

**Argumentos de entrada:**

**SensorHome_X:** BOOL. Sensor Home del motor PaP encargado del desplazamiento X.

**FC_inicio_X:** BOOL. Final de carrera de inicio del motor PaP encargado del desplazamiento X.

**FC_fin_X:** BOOL. Final de carrera de final del motor PaP encargado del desplazamiento X.

**SensorHome_Y:** BOOL. Sensor Home del motor PaP encargado del desplazamiento Y.

**FC_inicio_Y:** BOOL. Final de carrera de inicio del motor PaP encargado del desplazamiento Y.

**FC_fin_Y:** BOOL. Final de carrera de final del motor PaP encargado del desplazamiento Y.

**SensorHome_Z:** BOOL. Sensor Home del motor PaP encargado del desplazamiento Z.

**FC_inicio_Z:** BOOL. Final de carrera de inicio del motor PaP encargado del desplazamiento Z.

**FC_fin_Z:** BOOL. Final de carrera de final del motor PaP encargado del desplazamiento Z.

**PresionAire:** BOOL. Variable asociada al sensor de presión, indicativo de que el aire a presión está en condiciones de operar la pinza.

**SensorPinzaAbierta:** BOOL. Sensor del estado de la pinza. A nivel alto cuando la pinza se encuentra completamente abierta.

**SensorPinzaCerrada:** BOOL. Sensor del estado de la pinza. A nivel alto cuando la pinza se encuentra completamente cerrada.

**Carrera_X:** DINT. Carrera del motor PaP encargado del desplazamiento X.

**Avance_X:** DINT. Avance del motor PaP encargado del desplazamiento X.

**Carrera_Y:** DINT. Carrera del motor PaP encargado del desplazamiento Y.

**Avance_Y:** DINT. Avance del motor PaP encargado del desplazamiento Y.

**Carrera_Z:** DINT. Carrera del motor PaP encargado del desplazamiento Z.

**Avance_Z:** DINT. Avance del motor PaP encargado del desplazamiento Z.

**HzVelocidadAlta:** INT. Frecuencia a la que queremos que funcione el motor en las tareas de posicionamiento rápido, en hercios (Hz). Funciona en un rango de [1,60].

**HzVelocidadBaja:** INT. Frecuencia a la que queremos que funcione el motor en las tareas de acercamiento fino, en hercios (Hz). Funciona en un rango de [1,60].

**STOP:** BOOL. Parada de emergencia. A nivel alto, los motores PaP se detendrán y la pinza se cerrará.

**Origen_X:** DINT. Posición de recogida de la pieza en la coordenada X según la referencia dada por el motor PaP correspondiente.

**Origen_Y:** DINT. Posición de recogida de la pieza en la coordenada Y según la referencia dada por el motor PaP correspondiente.

**Origen_Z:** DINT. Posición de recogida de la pieza en la coordenada Z según la referencia dada por el motor PaP correspondiente.

**Destino_X:** DINT. Posición de entrega de la pieza en la coordenada X según la referencia dada por el motor PaP correspondiente.

**Destino_Y:** DINT. Posición de entrega de la pieza en la coordenada Y según la referencia dada por el motor PaP correspondiente.

**Destino_Z:** DINT. Posición de entrega de la pieza en la coordenada Z según la referencia dada por el motor PaP correspondiente.

**Argumentos de salida:**

**EN_Driver_X:** BOOL. Habilitación del motor PaP encargado del deplazamiento X.

**DIR_Driver_X:** BOOL. Dirección de giro del motor PaP encargado del deplazamiento X.

**STEP_Driver_X:** BOOL. Señal de pulsos que recibe el motor PaP encargado del deplazamiento X.

**HomeOK_X:** BOOL. Indica el estado de calibración del motor PaP encargado del deplazamiento X.

**EN_Driver_Y:** BOOL. Habilitación del motor PaP encargado del deplazamiento Y.

**DIR_Driver_Y:** BOOL. Dirección de giro del motor PaP encargado del deplazamiento Y.

**STEP_Driver_Y:** BOOL. Señal de pulsos que recibe el motor PaP encargado del deplazamiento Y.

**HomeOK_Y:** BOOL. Indica el estado de calibración del motor PaP encargado del deplazamiento Y.

**EN_Driver_Z:** BOOL. Habilitación del motor PaP encargado del deplazamiento Z.

**DIR_Driver_Z:** BOOL. Dirección de giro del motor PaP encargado del deplazamiento Z.

**STEP_Driver_Z:** BOOL. Señal de pulsos que recibe el motor PaP encargado del deplazamiento Z.

**HomeOK_Z:** BOOL. Indica el estado de calibración del motor PaP encargado del deplazamiento Z.

**Electrovalvula:** BOOL. Salida encargada de controlar el actuador de la electrovávula que opera la pinza. A nivel alto, la pinza se cerrará.

**PosicionPinza_X:** DINT. Posición actual de la pinza en la coordenada X según la referencia dada por el motor PaP correspondiente.

**PosicionPinza_Y:** DINT. Posición actual de la pinza en la coordenada Y según la referencia dada por el motor PaP correspondiente.

**PosicionPinza_Z:** DINT. Posición actual de la pinza en la coordenada Z según la referencia dada por el motor PaP correspondiente.

**ESTADO:** INT. Variable de información para el usuario. Puede tomar los siguientes valores:

- -1 máquina no calibrada. Realizando el AutoTest.
- 0 esperando orden.
- 1 máquina en funcionamiento.

**Argumentos de entrada/salida:**

**Ejecutar:** BOOL. Orden para realizar la operación de recogida y entrega de pieza conforme a las posiciones estipuladas.

**AutoTest:** BOOL. Orden para realizar una prueba de calibración de la máquina.

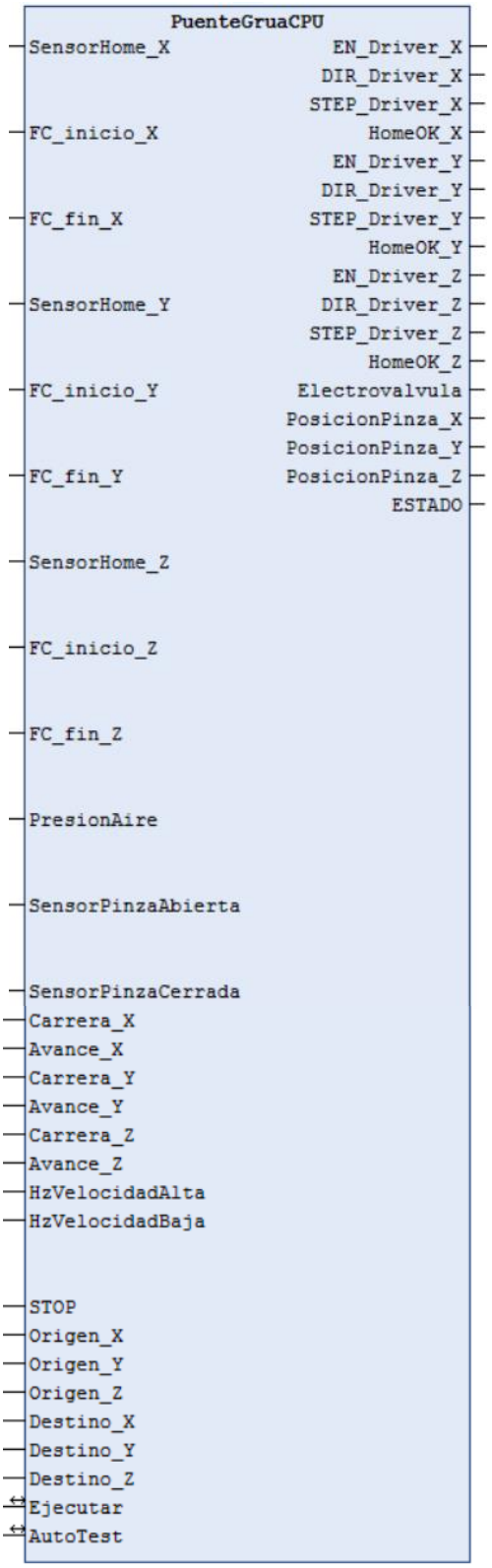


Figura 3-14. Bloque funcional PuenteGruaCPU

### 3.3.3 Funcionamiento

La automatización de esta máquina está igualmente basada en estados y transiciones. Desde un estado de reposo, puede recibir dos tipos de órdenes, vistos anteriormente.

**AutoTest**

- La operación de AutoTest comienza cuando dicha variable es desactivada externamente. Primeramente, realiza una prueba de la pinza, cerrando y abriendo la garra por completo.
- A continuación, realiza una prueba de calibración del motor PaP encargado del desplazamiento X.
- Seguidamente, una prueba de calibración del motor PaP encargado del desplazamiento Y.
- Por último, una prueba de calibración del motor PaP encargado del desplazamiento Z.
- Finalmente, se acciona el motor Z para situar la pinza a media altura. Con esto se da por terminada la operación y se activa AutoTest antes de volver al estado inicial.

**Ejecutar**

- Con la activación de Ejecutar, se empieza por llevar a la pinza a la posición Origen. Esto se hace en orden, empezando por el desplazamiento X, pasando por el Y y terminando con el Z.
- Llegado a la posición Origen, donde se espera que esté la pieza, se da la orden de cerrar la pinza y se eleva a una posición intermedia.
- A continuación, se transporta a la posición Destino, siguiendo el mismo orden en el desplazamiento.
- Llegado a la posición Destino, se ordena abrir la pinza para depositar la pieza en su lugar y se vuelve a elevar a una posición intermedia.
- Hecho esto, se desactiva la orden Ejecutar y se vuelve al estado inicial.

Como se ha visto en los bloques anteriores, el AutoTest comienza automáticamente con la primera ejecución, con el fin de dejar la máquina calibrada y en posición para estar operativa.

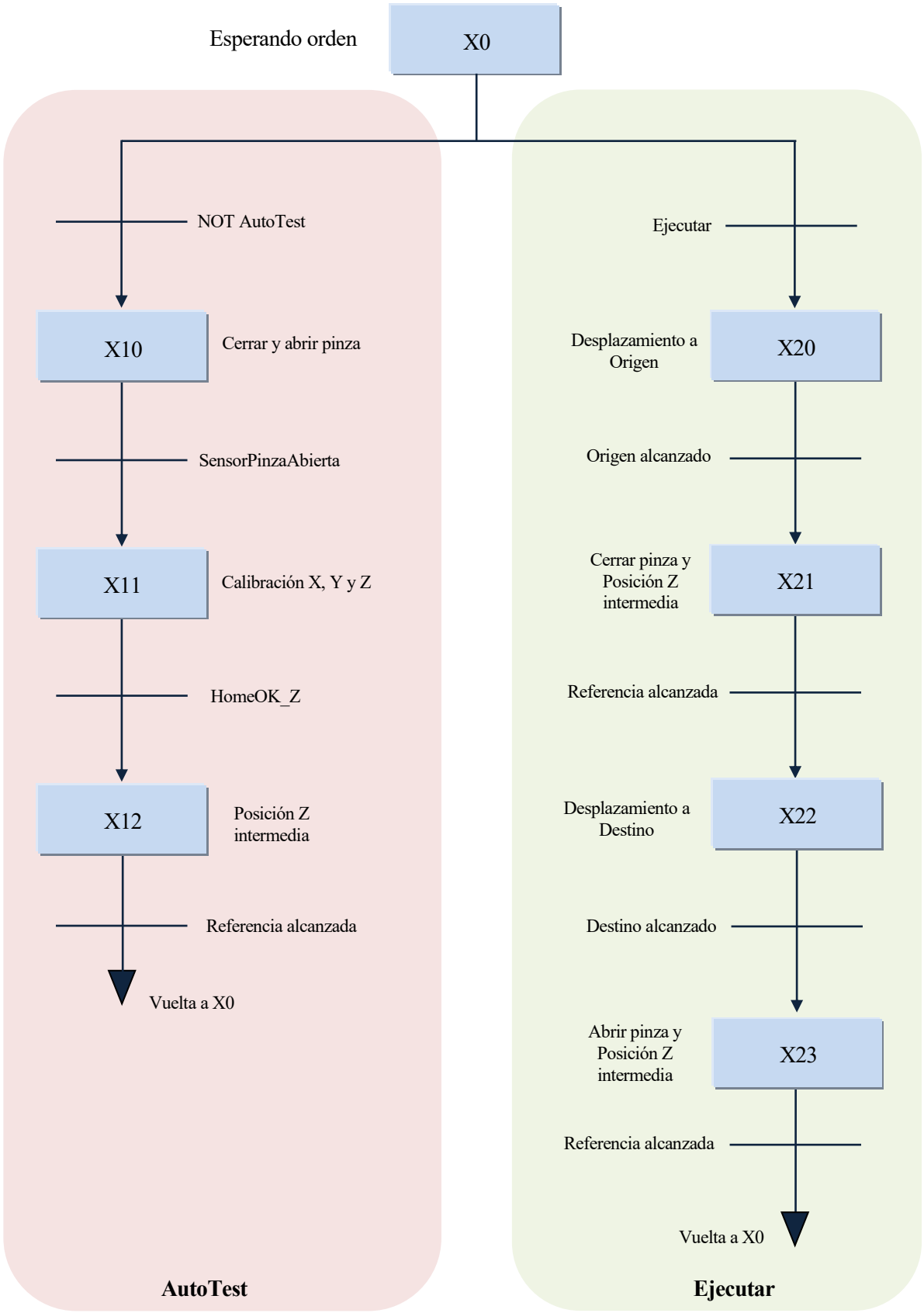


Figura 3-15. Diagrama de estados simplificado de PuenteGruaCPU

### 3.3.4 Simulación

El bloque PuenteGruaDEMO ha de encargarse de gestionar la visualización y activación de los sensores de calibración y la pinza. Para ello empleará datos provenientes del controlador como la posición de los motores PaP y el estado de la pinza.

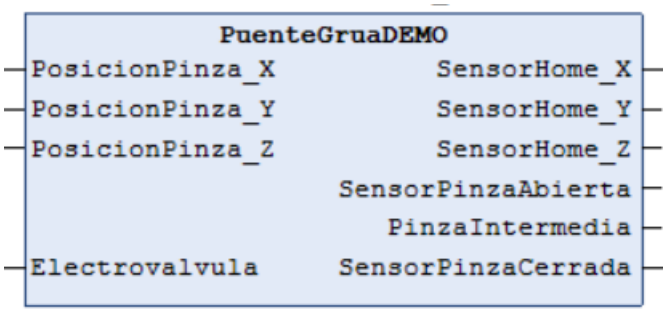


Figura 3-16. Bloque funcional PuenteGruaDEMO

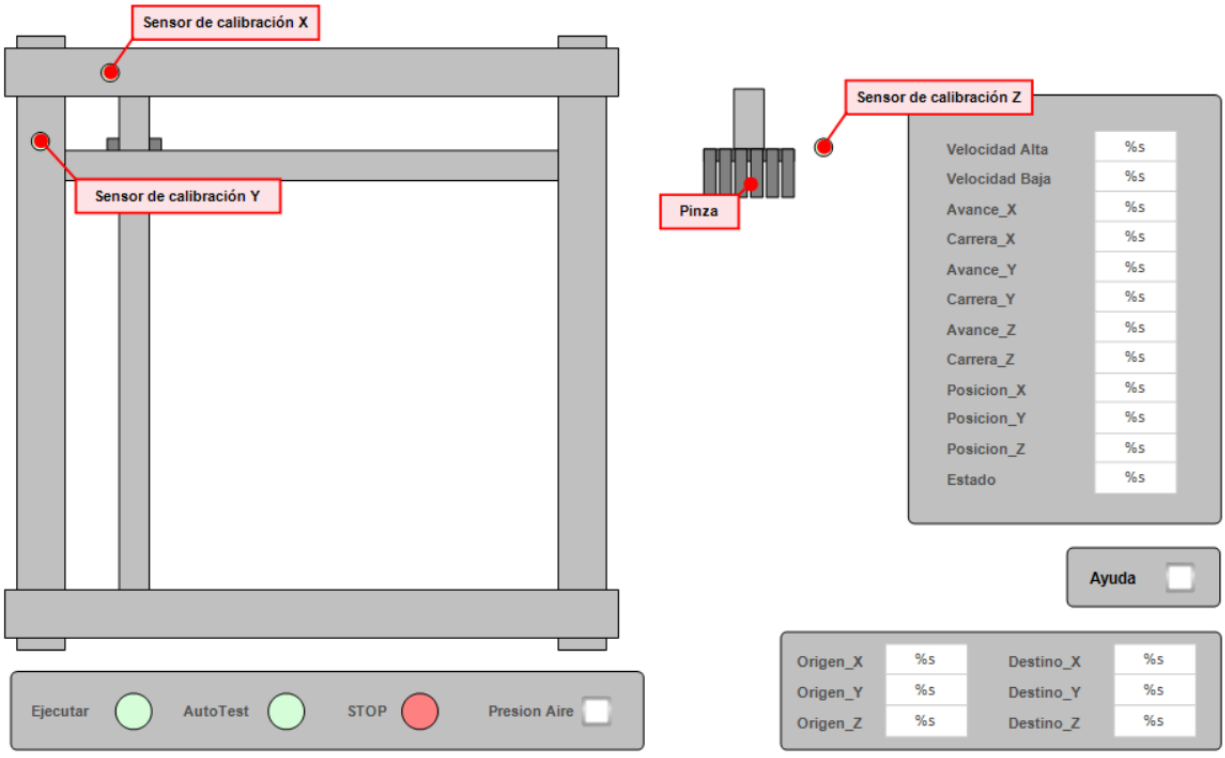


Figura 3-17. Interfaz del simulador del puente grúa

## 3.4 ScaraCPU

### 3.4.1 Introducción

Este bloque funcional se encarga de operar la estación destinada al control de calidad. Esta consta de un robot Scara que monta en su extremo una cámara encargada de tomar fotografías de los productos terminados, que serán procesadas por un programa de visión artificial para determinar si el producto cumple con las exigencias impuestas.

El robot Scara es una máquina de cuatro grados de libertad. Partiendo de la base, las tres primeras articulaciones son de rotación, y la última es prismática. Para ello se necesitarán cuatro motores PaP, luego se emplearán igualmente cuatro bloques MueveDispositivoCPU.

Por su parte, el control de la cámara es trivial, puesto que la ejecución del programa de visión artificial es externa a la automatización de la máquina. Bastará con destinar una salida para dar orden de comienzo a la inspección y una entrada para avisar de la finalización del programa de visión, una vez obtenidos los resultados.

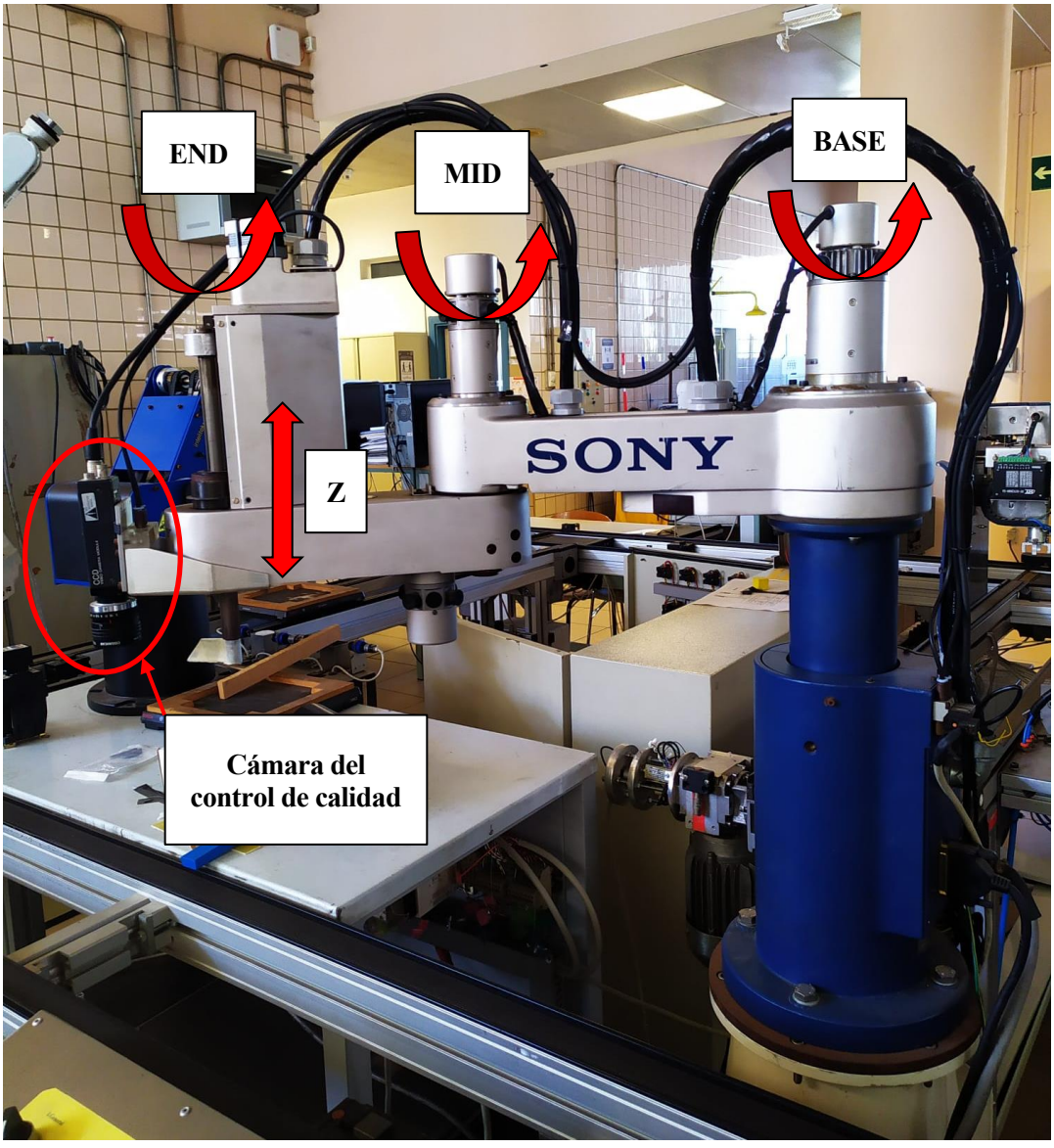


Figura 3-18. Robot Scara - Fotografía del laboratorio

### 3.4.2 Variables del bloque funcional

**Argumentos de entrada:**

**SensorHome_BASE:** BOOL. Sensor Home del motor PaP encargado del giro BASE.

**FC_inicio_BASE:** BOOL. Final de carrera de inicio del motor PaP encargado del giro BASE.

**FC_fin_BASE:** BOOL. Final de carrera de final del motor PaP encargado del giro BASE.

**SensorHome_MID:** BOOL. Sensor Home del motor PaP encargado del giro MID.

**FC_inicio_MID:** BOOL. Final de carrera de inicio del motor PaP encargado del giro MID.

**FC_fin_MID:** BOOL. Final de carrera de final del motor PaP encargado del giro MID.

**SensorHome_END:** BOOL. Sensor Home del motor PaP encargado del giro END.

**FC_inicio_END:** BOOL. Final de carrera de inicio del motor PaP encargado del giro END.

**FC_fin_END:** BOOL. Final de carrera de final del motor PaP encargado del giro END.

**SensorHome_Z:** BOOL. Sensor Home del motor PaP encargado del desplazamiento Z.

**FC_inicio_Z:** BOOL. Final de carrera de inicio del motor PaP encargado del desplazamiento Z.

**FC_fin_Z:** BOOL. Final de carrera de final del motor PaP encargado del desplazamiento Z.

**Carrera_BASE:** DINT. Carrera del motor PaP encargado del giro BASE.

**Avance_BASE:** DINT. Avance del motor PaP encargado del giro BASE.

**Carrera_MID:** DINT. Carrera del motor PaP encargado del giro MID.

**Avance_MID:** DINT. Avance del motor PaP encargado del giro MID.

**Carrera_END:** DINT. Carrera del motor PaP encargado del giro END.

**Avance_END:** DINT. Avance del motor PaP encargado del giro END.

**Carrera_Z:** DINT. Carrera del motor PaP encargado del desplazamiento Z.

**Avance_Z:** DINT. Avance del motor PaP encargado del desplazamiento Z.

**HzVelocidadAlta:** INT. Frecuencia a la que queremos que funcione el motor en las tareas de posicionamiento rápido, en hercios (Hz). Funciona en un rango de [1,60].

**HzVelocidadBaja:** INT. Frecuencia a la que queremos que funcione el motor en las tareas de acercamiento fino, en hercios (Hz). Funciona en un rango de [1,60].

**STOP:** BOOL. Parada de emergencia. A nivel alto, los motores PaP se detendrán.

**Inspeccion_BASE:** DINT. Posición angular del producto a examinar en la coordenada BASE según la referencia dada por el motor PaP correspondiente.

**Inspeccion_MID:** DINT. Posición angular del producto a examinar en la coordenada MID según la referencia dada por el motor PaP correspondiente.

**Inspeccion_END:** DINT. Posición angular del producto a examinar en la coordenada END según la referencia dada por el motor PaP correspondiente.

**Inspeccion_Z:** DINT. Posición del producto a examinar en la coordenada Z según la referencia dada por el motor PaP correspondiente.

**FinInspeccion:** BOOL. Confirmación externa proveniente del programa de visión artificial, que confirma que la inspección a finalizado.

**Argumentos de salida:**

**EN_Driver_BASE:** BOOL. Habilitación del motor PaP encargado del giro BASE.

**DIR_Driver_BASE:** BOOL. Dirección de giro del motor PaP encargado del giro BASE.

**STEP_Driver_BASE:** BOOL. Señal de pulsos que recibe el motor PaP encargado del giro BASE.

**HomeOK_BASE:** BOOL. Indica el estado de calibración del motor PaP encargado del giro BASE.

**EN_Driver_MID:** BOOL. Habilitación del motor PaP encargado del giro MID.

**DIR_Driver_MID:** BOOL. Dirección de giro del motor PaP encargado del giro MID.

**STEP_Driver_MID:** BOOL. Señal de pulsos que recibe el motor PaP encargado del giro MID.

**HomeOK_MID:** BOOL. Indica el estado de calibración del motor PaP encargado del giro MID.

**EN_Driver_END:** BOOL. Habilitación del motor PaP encargado del giro END.

**DIR_Driver_END:** BOOL. Dirección de giro del motor PaP encargado del giro END.

**STEP_Driver_END:** BOOL. Señal de pulsos que recibe el motor PaP encargado del giro END.

**HomeOK_END:** BOOL. Indica el estado de calibración del motor PaP encargado del giro END.

**EN_Driver_Z:** BOOL. Habilitación del motor PaP encargado del deplazamiento Z.

**DIR_Driver_Z:** BOOL. Dirección de giro del motor PaP encargado del deplazamiento Z.

**STEP_Driver_Z:** BOOL. Señal de pulsos que recibe el motor PaP encargado del deplazamiento Z.

**HomeOK_Z:** BOOL. Indica el estado de calibración del motor PaP encargado del deplazamiento Z.

**Electrovalvula:** BOOL. Salida encargada de controlar el actuador de la electrovávula que opera la pinza. A nivel alto, la pinza se cerrará.

**Posicion_BASE:** DINT. Posición angular actual de la pinza en la coordenada BASE según la referencia dada por el motor PaP correspondiente.

**Posicion_MID:** DINT. Posición angular actual de la pinza en la coordenada MID según la referencia dada por el motor PaP correspondiente.

**Posicion_END:** DINT. Posición angular actual de la pinza en la coordenada END según la referencia dada por el motor PaP correspondiente.

**ESTADO:** INT. Variable de información para el usuario. Puede tomar los siguientes valores:

- -1 máquina no calibrada. Realizando el AutoTest.
- 0 esperando orden.
- 1 máquina en funcionamiento.

**Argumentos de entrada/salida:**

**Ejecutar:** BOOL. Orden para realizar la operación de acercamiento e inspección conforme a las posiciones estipuladas.

**AutoTest:** BOOL. Orden para realizar una prueba de calibración de la máquina.

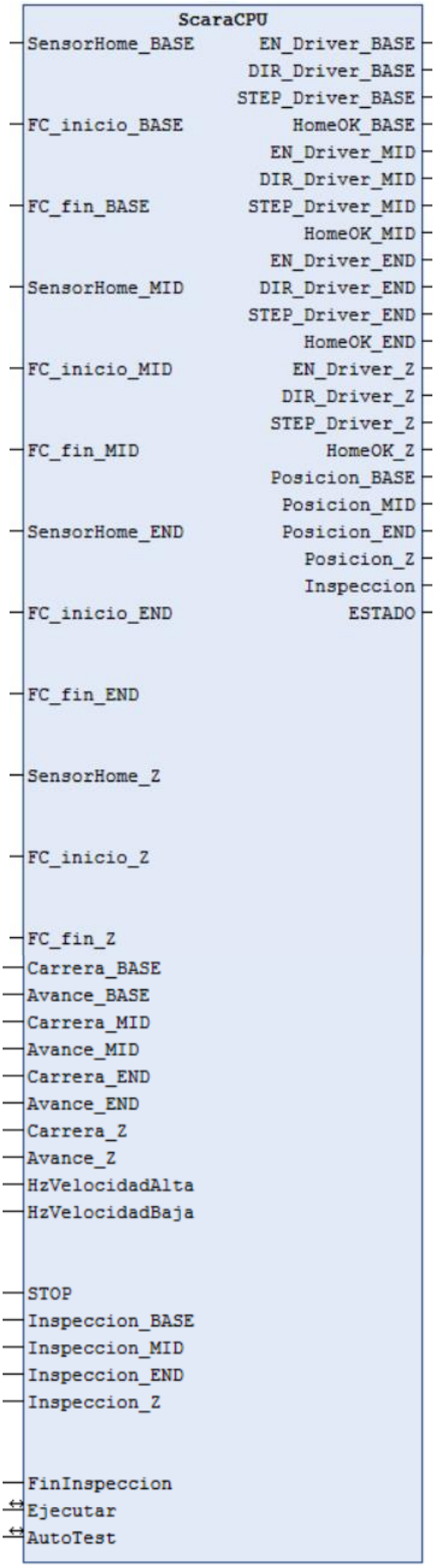


Figura 3-19. Bloque funcional ScaraCPU

### 3.4.3 Funcionamiento

La automatización de esta máquina está igualmente basada en estados y transiciones. Desde un estado de reposo, puede recibir dos tipos de órdenes, vistos anteriormente.

**AutoTest**

- La operación de AutoTest comienza cuando dicha variable es desactivada externamente.
- Primeramente, realiza una prueba de calibración del motor PaP encargado del del giro BASE, seguido de los giros MID y END, terminando con la calibración del desplazamiento Z.
- Finalmente, se acciona el motor Z para situar la pinza a media altura. Con esto se da por terminada la operación y se activa AutoTest antes de volver al estado inicial.

**Ejecutar**

- Con la activación de Ejecutar, se empieza por llevar a la pinza a la posición Inspección. Esto se hace en orden, empezando por el giro, seguido de los giros MID y END, terminando con el desplazamiento Z.
- Llegado a la posición Inspección, donde se espera que esté el producto a examinar, se da la orden de comenzar la inspección, lo que desencadenaría la toma de fotografías y la ejecución del programa de visión artificial.
- El robot espera a la confirmación de que el control de calidad se ha realizado a través de la variable FinInspeccion.
- Finalmente, el robot se desplaza a una posición donde no interfiera. Esta será la posición 0 o Home de los motores PaP del deplazamiento Z y los giros END y MID. El movimiento se realizará en ese mismo orden. El motor de giro BASE se mantendrá en la posición Inspección, dejando el robot más cerca para cuando se realice lel próximo control de calidad.
- Hecho esto, se desactiva la orden Ejecutar y se vuelve al estado inicial.

Como se ha visto en los bloques anteriores, el AutoTest comienza automáticamente con la primera ejecución, con el fin de dejar la máquina calibrada y en posición para estar operativa.

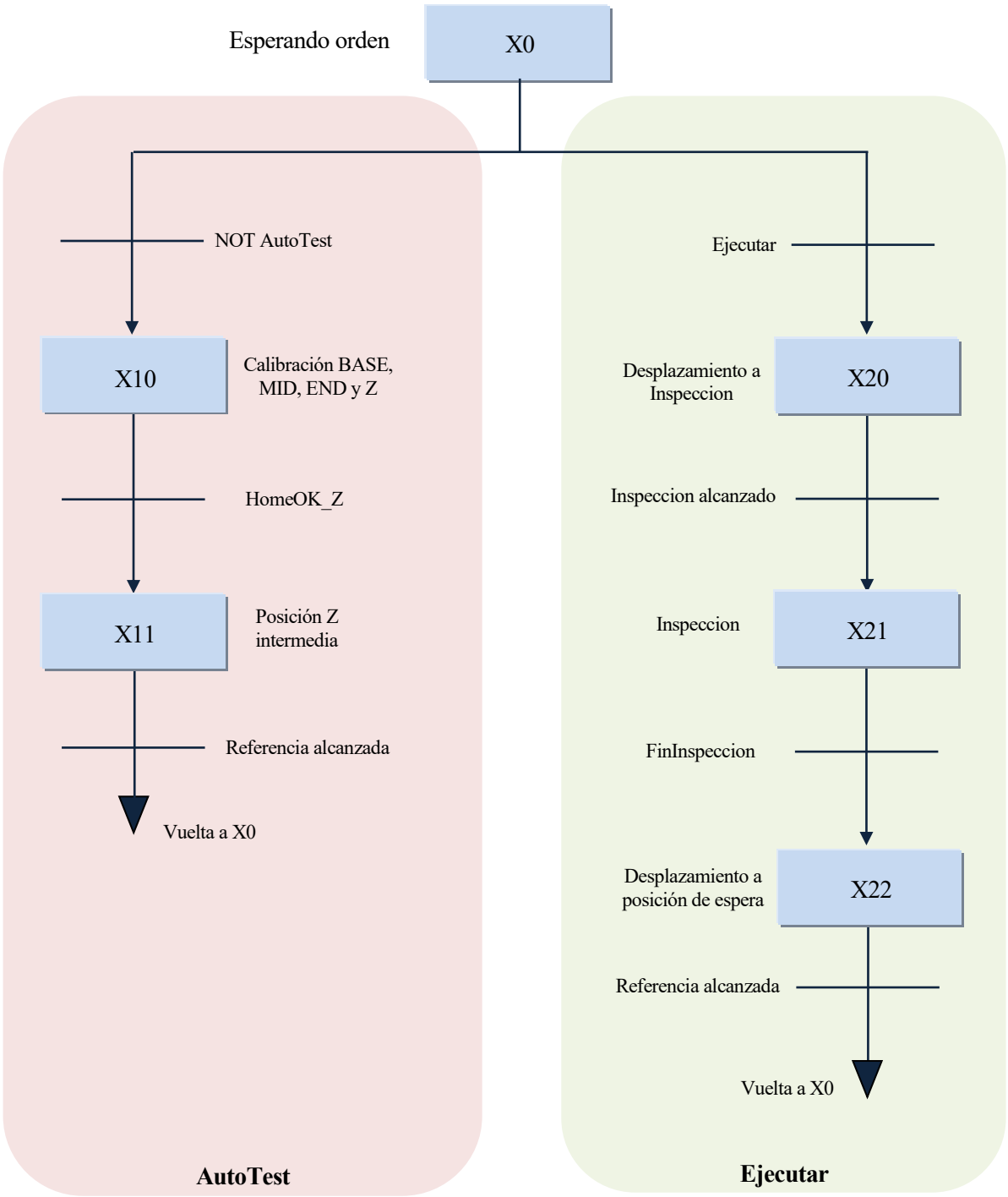


Figura 3-20. Diagrama de estados simplificado de ScaraCPU

### 3.4.4 Simulación

El bloque ScaraDEMO ha de encargarse de gestionar la visualización y activación de los sensores de calibración. Para ello empleará datos provenientes del controlador como la posición de los motores PaP.

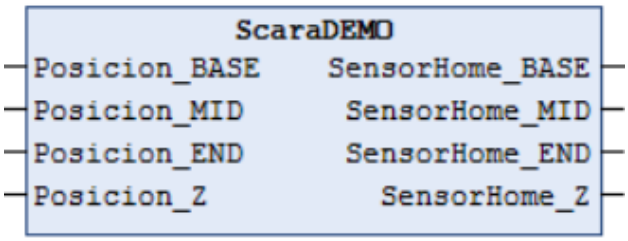


Figura 3-21. Bloque funcional ScaraDEMO

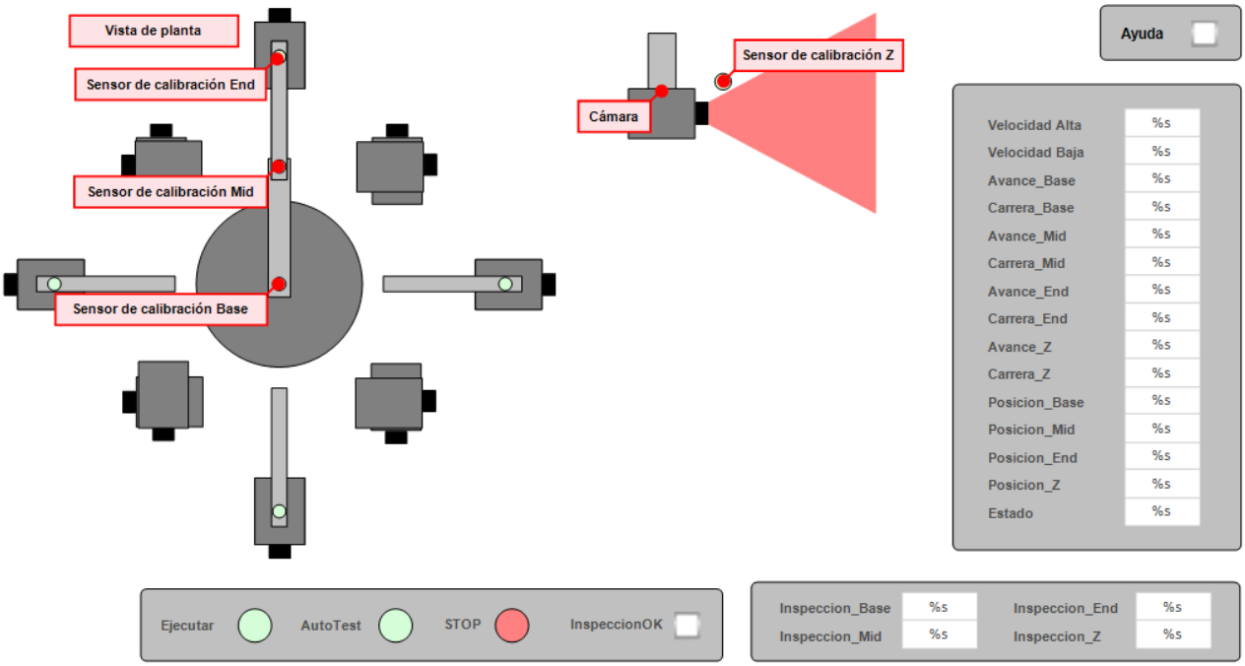


Figura 3-22. Interfaz del simulador del Scara

## 3.5 ScorbotCPU

### 3.5.1 Introducción

Este bloque funcional, como su propio nombre indica, se encarga del control de un brazo Scorbot. Éstos constan de cinco articulaciones y una pinza en el extremo. Por tanto, se emplearán cinco bloques MueveDispositivoCPU y un bloque OperaPinzaCPU.

Su propósito será, al igual que con el pórtico, transportar piezas en su entorno de trabajo.

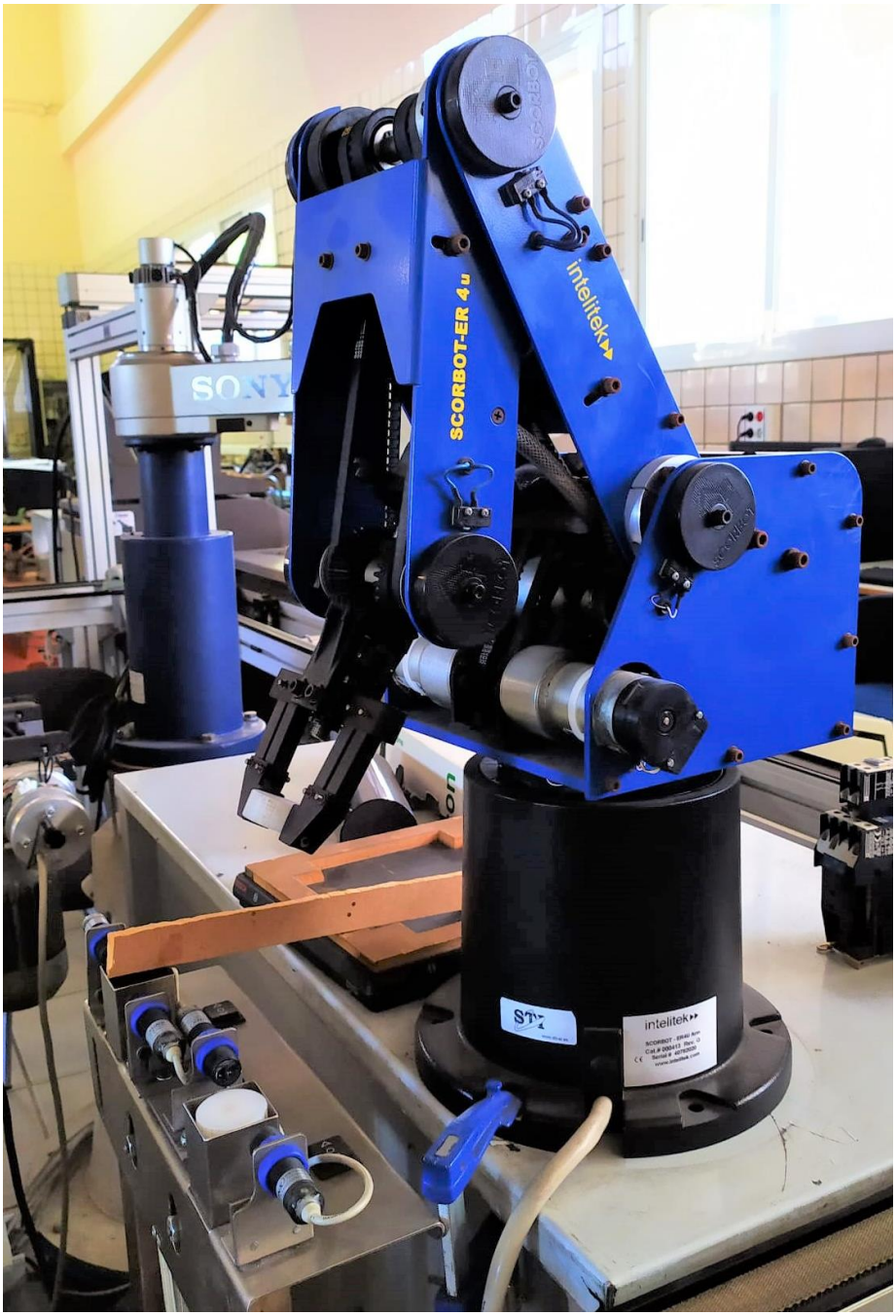


Figura 3-23. Robot Scorbot - Fotografía del laboratorio

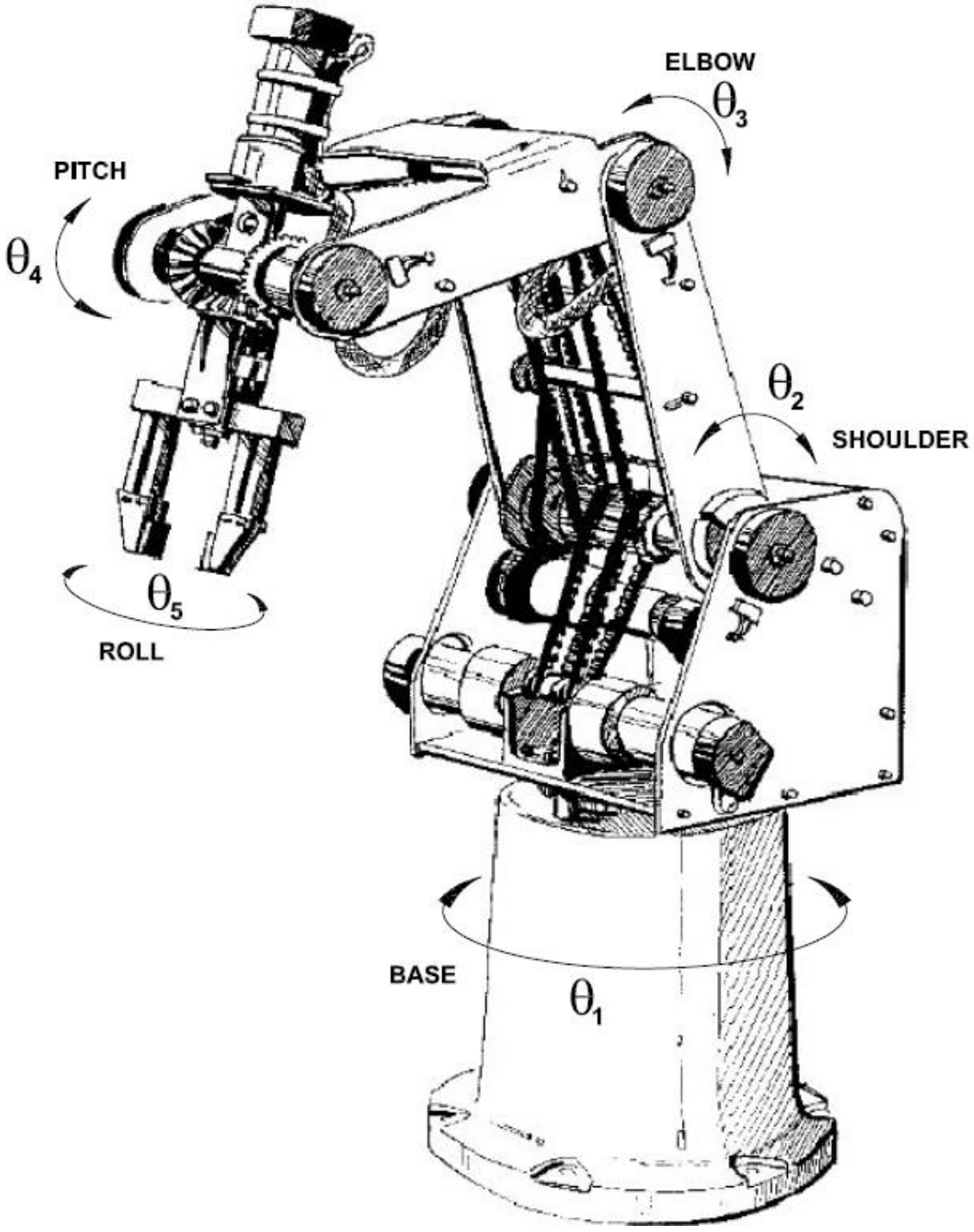


Figura 3-24. Esquema del Scorbot

### 3.5.2 Variables del bloque funcional

**Argumentos de entrada:**

**SensorHome_BASE:** BOOL. Sensor Home del motor PaP encargado del giro BASE.

**FC_inicio_BASE:** BOOL. Final de carrera de inicio del motor PaP encargado del giro BASE.

**FC_fin_BASE:** BOOL. Final de carrera de final del motor PaP encargado del giro BASE.

**SensorHome_SHOULDER:** BOOL. Sensor Home del motor PaP encargado del giro SHOULDER.

**FC_inicio_SHOULDER:** BOOL. Final de carrera de inicio del motor PaP encargado del giro SHOULDER.

**FC_fin_SHOULDER:** BOOL. Final de carrera de final del motor PaP encargado del giro SHOULDER.

**SensorHome_ELBOW:** BOOL. Sensor Home del motor PaP encargado del giro ELBOW.

**FC_inicio_ELBOW:** BOOL. Final de carrera de inicio del motor PaP encargado del giro ELBOW.

**FC_fin_ELBOW:** BOOL. Final de carrera de final del motor PaP encargado del giro ELBOW.

**SensorHome_PITCH:** BOOL. Sensor Home del motor PaP encargado del giro PITCH.

**FC_inicio_PITCH:** BOOL. Final de carrera de inicio del motor PaP encargado del giro PITCH.

**FC_fin_PITCH:** BOOL. Final de carrera de final del motor PaP encargado del giro PITCH.

**SensorHome_ROLL:** BOOL. Sensor Home del motor PaP encargado del giro ROLL.

**FC_inicio_ROLL:** BOOL. Final de carrera de inicio del motor PaP encargado del giro ROLL.

**FC_fin_ROLL:** BOOL. Final de carrera de final del motor PaP encargado del giro ROLL.

**PresionAire:** BOOL. Variable asociada al sensor de presión, indicativo de que el aire a presión está en condiciones de operar la pinza.

**SensorPinzaAbierta:** BOOL. Sensor del estado de la pinza. A nivel alto cuando la pinza se encuentra completamente abierta.

**SensorPinzaCerrada:** BOOL. Sensor del estado de la pinza. A nivel alto cuando la pinza se encuentra completamente cerrada.

**Carrera_BASE:** DINT. Carrera del motor PaP encargado del giro BASE.

**Avance_BASE:** DINT. Avance del motor PaP encargado del giro BASE.

**Carrera_SHOULDER:** DINT. Carrera del motor PaP encargado del giro SHOULDER.

**Avance_SHOULDER:** DINT. Avance del motor PaP encargado del giro SHOULDER.

**Carrera_ELBOW:** DINT. Carrera del motor PaP encargado del giro ELBOW.

**Avance_ELBOW:** DINT. Avance del motor PaP encargado del giro ELBOW.

**Carrera_PITCH:** DINT. Carrera del motor PaP encargado del giro PITCH.

**Avance_PITCH:** DINT. Avance del motor PaP encargado del giro PITCH.

**Carrera_ROLL:** DINT. Carrera del motor PaP encargado del giro ROLL.

**Avance_ROLL:** DINT. Avance del motor PaP encargado del giro ROLL.

**HzVelocidadAlta:** INT. Frecuencia a la que queremos que funcione el motor en las tareas de posicionamiento rápido, en hercios (Hz). Funciona en un rango de [1,60].

**HzVelocidadBaja:** INT. Frecuencia a la que queremos que funcione el motor en las tareas de acercamiento fino, en hercios (Hz). Funciona en un rango de [1,60].

**STOP:** BOOL. Parada de emergencia. A nivel alto, los motores PaP se detendrán y la pinza se cerrará.

**Origen_BASE:** DINT. Posición angular de recogida de la pieza en la coordenada BASE según la referencia dada por el motor PaP correspondiente.

**Origen_SHOULDER:** DINT. Posición angular de recogida de la pieza en la coordenada SHOULDER según la referencia dada por el motor PaP correspondiente.

**Origen_ELBOW:** DINT. Posición angular de recogida de la pieza en la coordenada ELBOW según la referencia dada por el motor PaP correspondiente.

**Origen_PITCH:** DINT. Posición angular de recogida de la pieza en la coordenada PITCH según la referencia dada por el motor PaP correspondiente.

**Origen_ROLL:** DINT. Posición angular de recogida de la pieza en la coordenada ROLL según la referencia dada por el motor PaP correspondiente.

**Destino_BASE:** DINT. Posición angular de entrega de la pieza en la coordenada BASE según la referencia dada por el motor PaP correspondiente.

**Destino_SHOULDER:** DINT. Posición angular de entrega de la pieza en la coordenada SHOULDER según la referencia dada por el motor PaP correspondiente.

**Destino_ELBOW:** DINT. Posición angular de entrega de la pieza en la coordenada ELBOW según la referencia dada por el motor PaP correspondiente.

**Destino_PITCH:** DINT. Posición angular de entrega de la pieza en la coordenada PITCH según la referencia dada por el motor PaP correspondiente.

**Destino_ROLL:** DINT. Posición angular de entrega de la pieza en la coordenada ROLL según la referencia dada por el motor PaP correspondiente.

**Argumentos de salida:**

**EN_Driver_BASE:** BOOL. Habilitación del motor PaP encargado del giro BASE.

**DIR_Driver_BASE:** BOOL. Dirección de giro del motor PaP encargado del giro BASE.

**STEP_Driver_BASE:** BOOL. Señal de pulsos que recibe el motor PaP encargado del giro BASE.

**HomeOK_BASE:** BOOL. Indica el estado de calibración del motor PaP encargado del giro BASE.

**EN_Driver_SHOULDER:** BOOL. Habilitación del motor PaP encargado del giro SHOULDER.

**DIR_Driver_SHOULDER:** BOOL. Dirección de giro del motor PaP encargado del giro SHOULDER.

**STEP_Driver_SHOULDER:** BOOL. Señal de pulsos que recibe el motor PaP encargado del giro SHOULDER.

**HomeOK_SHOULDER:** BOOL. Indica el estado de calibración del motor PaP encargado del giro SHOULDER.

**EN_Driver_ELBOW:** BOOL. Habilitación del motor PaP encargado del giro ELBOW.

**DIR_Driver_ELBOW:** BOOL. Dirección de giro del motor PaP encargado del giro ELBOW.

**STEP_Driver_ELBOW:** BOOL. Señal de pulsos que recibe el motor PaP encargado del giro ELBOW.

**HomeOK_ELBOW:** BOOL. Indica el estado de calibración del motor PaP encargado del giro ELBOW.

**EN_Driver_PITCH:** BOOL. Habilitación del motor PaP encargado del giro PITCH.

**DIR_Driver_PITCH:** BOOL. Dirección de giro del motor PaP encargado del giro PITCH.

**STEP_Driver_PITCH:** BOOL. Señal de pulsos que recibe el motor PaP encargado del giro PITCH.

**HomeOK_PITCH:** BOOL. Indica el estado de calibración del motor PaP encargado del giro PITCH.

**EN_Driver_ROLL:** BOOL. Habilitación del motor PaP encargado del giro ROLL.

**DIR_Driver_ROLL:** BOOL. Dirección de giro del motor PaP encargado del giro ROLL.

**STEP_Driver_ROLL:** BOOL. Señal de pulsos que recibe el motor PaP encargado del giro ROLL.

**HomeOK_ROLL:** BOOL. Indica el estado de calibración del motor PaP encargado del giro ROLL.

**PosicionPinza_BASE:** DINT. Posición angular actual de la pinza en la coordenada BASE según la referencia dada por el motor PaP correspondiente.

**PosicionPinza_SHOULDER:** DINT. Posición angular actual de la pinza en la coordenada SHOULDER según la referencia dada por el motor PaP correspondiente.

**PosicionPinza_ELBOW:** DINT. Posición angular actual de la pinza en la coordenada ELBOW según la referencia dada por el motor PaP correspondiente.

**PosicionPinza_PITCH:** DINT. Posición angular actual de la pinza en la coordenada PITCH según la referencia dada por el motor PaP correspondiente.

**PosicionPinza_ROLL:** DINT. Posición angular actual de la pinza en la coordenada ROLL según la referencia dada por el motor PaP correspondiente.

**ESTADO:** INT. Variable de información para el usuario. Puede tomar los siguientes valores:

- -1 máquina no calibrada. Realizando el AutoTest.
- 0 esperando orden.
- 1 máquina en funcionamiento.

**Argumentos de entrada/salida:**

**Ejecutar:** BOOL. Orden para realizar la operación de recogida y entrega de pieza conforme a las posiciones estipuladas.

**AutoTest:** BOOL. Orden para realizar una prueba de calibración de la máquina.

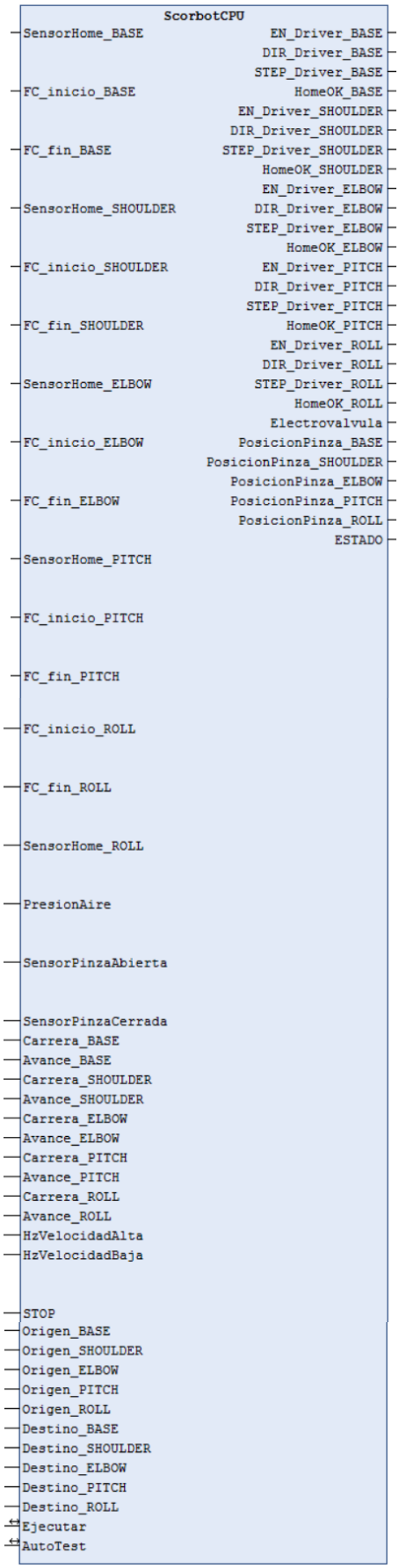


Figura 3-25. Bloque funcional ScorbotCPU

### 3.5.3 Funcionamiento

La automatización de esta máquina está igualmente basada en estados y transiciones. Desde un estado de reposo, puede recibir dos tipos de órdenes, vistos anteriormente.

**AutoTest**

- La operación de AutoTest comienza cuando dicha variable es desactivada externamente. Primeramente, realiza una prueba de la pinza, cerrando y abriendo la garra por completo.
- A continuación, realiza una prueba de calibración del motor PaP encargado del del giro BASE, seguido de los giros intermedios SHOULDER, ELBOW y los de muñeca PITCH y ROLL.

**Ejecutar**

- Con la activación de Ejecutar, se empieza por llevar a la pinza a la posición Origen. Esto se hace articulación a articulación siguiendo un orden específco. Este es, primero BASE, para orientar el robot a la zona de trabajo, luego ELBOW, PITCH y ROLL para posicionar el brazo en una posición natural de agarre, y finalmente SHOULDER para bajar el brazo a la posición donde se encuentra la pieza. De esta forma se evitan colisiones.
- Llegado a la posición Origen, donde se espera que esté la pieza, se da la orden de cerrar la pinza y se eleva a una posición intermedia.
- A continuación, se transporta a la posición Destino, siguiendo el mismo orden en el desplazamiento.
- Llegado a la posición Destino, se ordena abrir la pinza para depositar la pieza en su lugar.
- Finalmente, el robot se desplaza a una posición donde no interfiera. Esta será la posición 0 o Home de los motores PaP. El movimiento se realizará deshaciendo la trayectoria recientemente descrita, es decir, empezando por el giro SHOULDER y terminando por el de BASE.
- Hecho esto, se desactiva la orden Ejecutar y se vuelve al estado inicial.

Como se ha visto en los bloques anteriores, el AutoTest comienza automáticamente con la primera ejecución, con el fin de dejar la máquina calibrada y en posición para estar operativa.

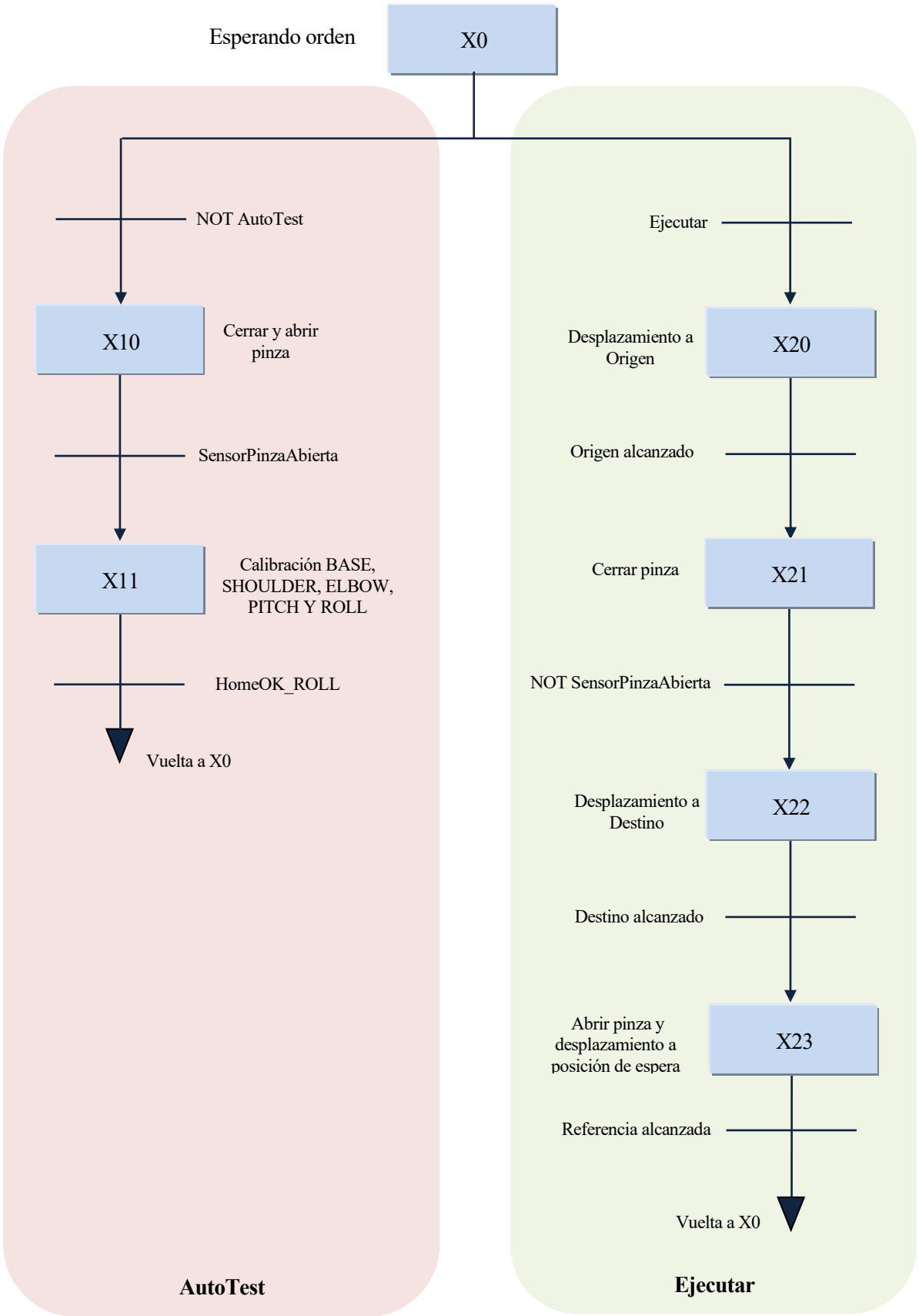


Figura 3-26. Diagrama de estados simplificado de ScorbotCPU

### 3.5.4 Simulación

El bloque ScorbotDEMO ha de encargarse de gestionar la visualización y activación de los sensores de calibración y la pinza. Para ello empleará datos provenientes del controlador como la posición de los motores PaP y el estado de la pinza.

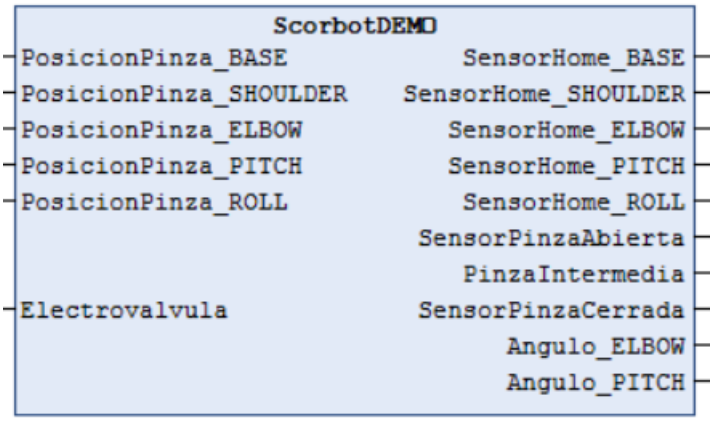


Figura 3-27. Bloque funcional ScorbotDEMO

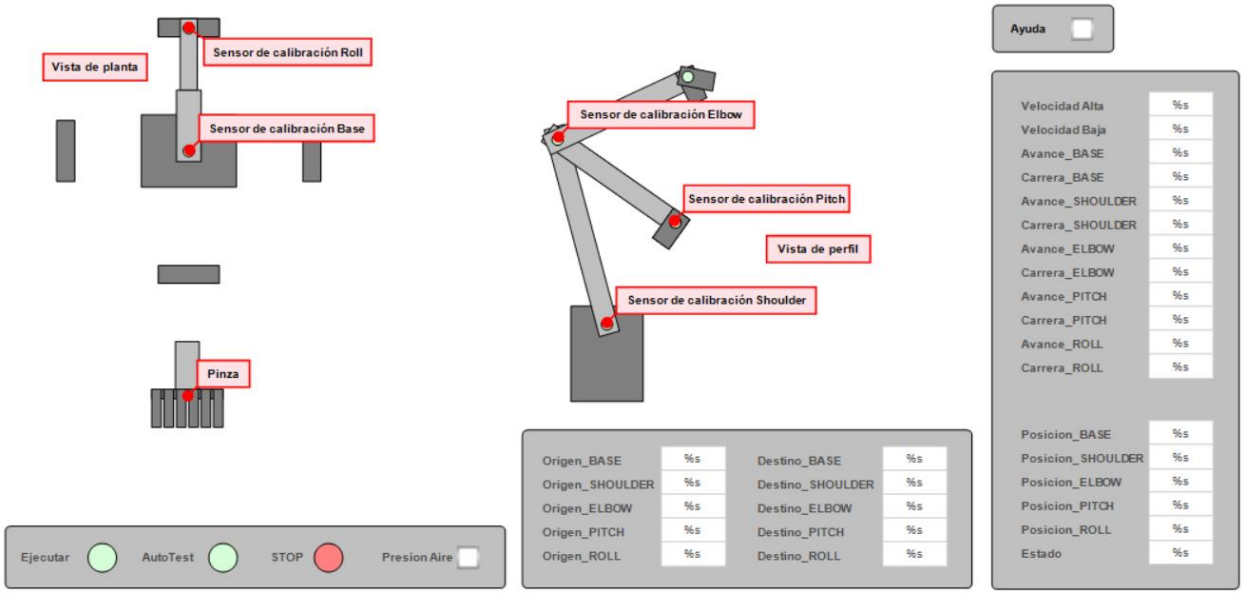


Figura 3-28. Interfaz del simulador del Scorbot

## 3.6 AlmacenPaletsCPU

### 3.6.1 Introducción

El bloque funcional AlmacenPaletsCPU maneja la estación encargada de gestionar un almacén de 72 plazas para palets junto a la máquina que transporta dichos palets, comunicando el almacén con el exterior. Esta máquina consta de cuatro grados de libertad (tres prismáticos y un giro). En su extremo, posee una pinza neumática para agarrar los palets, luego se emplearán cuatro bloques MueveDispositivoCPU y un bloque OperaPinzaCPU.


Figura 3-29. Vista superior del almacén de palets - Fotografía del laboratorio


Figura 3-30. Vista lateral del almacén de palets - Fotografía del laboratorio

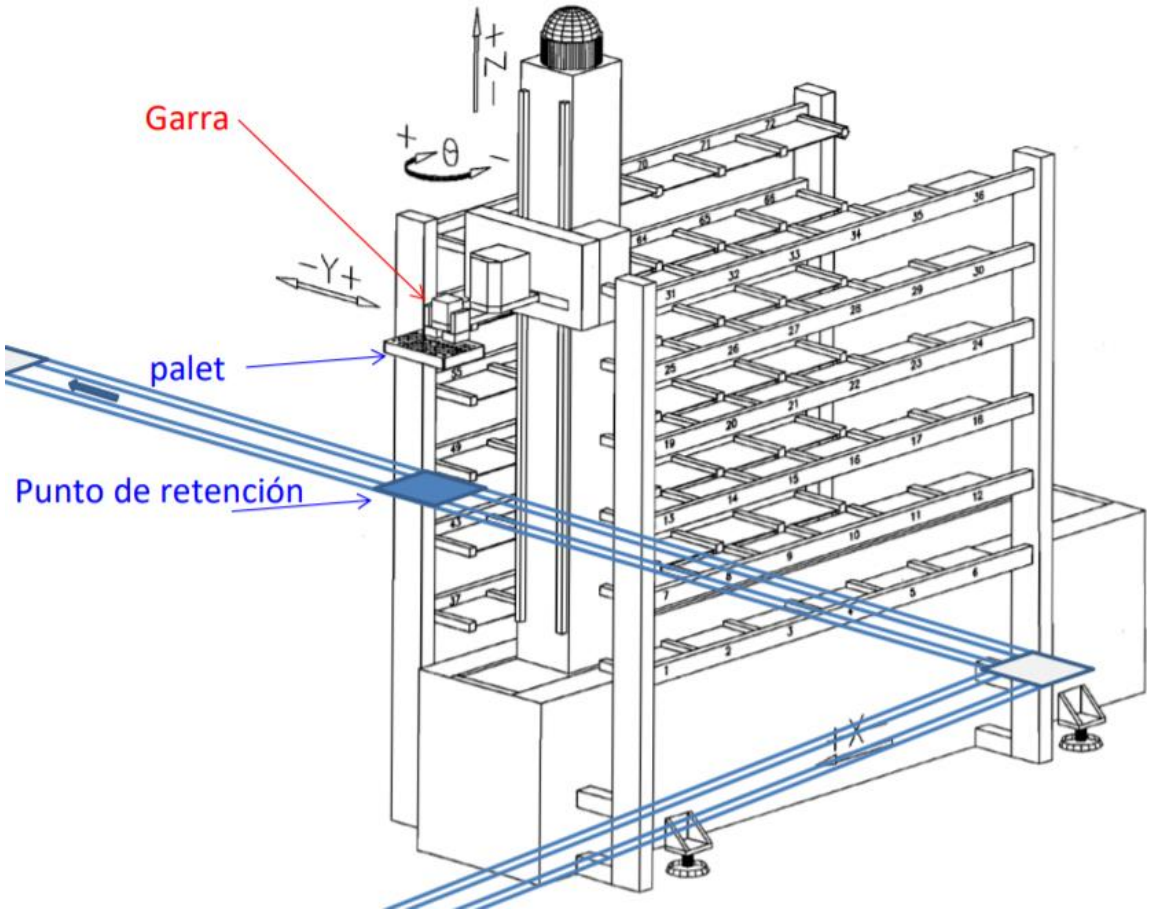


Figura 3-31. Esquema del almacén de palets

### 3.6.2 Variables del bloque funcional

**Argumentos de entrada:**

**SensorHome_X:** BOOL. Sensor Home del motor PaP encargado del desplazamiento X.

**FC_inicio_X:** BOOL. Final de carrera de inicio del motor PaP encargado del desplazamiento X.

**FC_fin_X:** BOOL. Final de carrera de final del motor PaP encargado del desplazamiento X.

**SensorHome_Y:** BOOL. Sensor Home del motor PaP encargado del desplazamiento Y.

**FC_inicio_Y:** BOOL. Final de carrera de inicio del motor PaP encargado del desplazamiento Y.

**FC_fin_Y:** BOOL. Final de carrera de final del motor PaP encargado del desplazamiento Y.

**SensorHome_Z:** BOOL. Sensor Home del motor PaP encargado del desplazamiento Z.

**FC_inicio_Z:** BOOL. Final de carrera de inicio del motor PaP encargado del desplazamiento Z.

**FC_fin_Z:** BOOL. Final de carrera de final del motor PaP encargado del desplazamiento Z.

**SensorHome_THETA:** BOOL. Sensor Home del motor PaP encargado del giro THETA.

**FC_inicio_THETA:** BOOL. Final de carrera de inicio del motor PaP encargado del giro THETA.

**FC_fin_THETA:** BOOL. Final de carrera de final del motor PaP encargado del giro THETA.

**PresionAire:** BOOL. Variable asociada al sensor de presión, indicativo de que el aire a presión está en condiciones de operar la pinza.

**SensorPinzaAbierta:** BOOL. Sensor del estado de la pinza. A nivel alto cuando la pinza se encuentra completamente abierta.

**SensorPinzaCerrada:** BOOL. Sensor del estado de la pinza. A nivel alto cuando la pinza se encuentra completamente cerrada.

**Carrera_X:** DINT. Carrera del motor PaP encargado del desplazamiento X.

**Avance_X:** DINT. Avance del motor PaP encargado del desplazamiento X.

**Carrera_Y:** DINT. Carrera del motor PaP encargado del desplazamiento Y.

**Avance_Y:** DINT. Avance del motor PaP encargado del desplazamiento Y.

**Carrera_Z:** DINT. Carrera del motor PaP encargado del desplazamiento Z.

**Avance_Z:** DINT. Avance del motor PaP encargado del desplazamiento Z.

**Carrera_THETA:** DINT. Carrera del motor PaP encargado del giro THETA.

**Avance_THETA:** DINT. Avance del motor PaP encargado del giro THETA.

**HzVelocidadAlta:** INT. Frecuencia a la que queremos que funcione el motor en las tareas de posicionamiento rápido, en hercios (Hz). Funciona en un rango de [1,60].

**HzVelocidadBaja:** INT. Frecuencia a la que queremos que funcione el motor en las tareas de acercamiento fino, en hercios (Hz). Funciona en un rango de [1,60].

**STOP:** BOOL. Parada de emergencia. A nivel alto, los motores PaP se detendrán y la pinza se cerrará.

**Retencion_X:** DINT. Posición del punto de retención de la estación en la coordenada X según la referencia dada por el motor PaP correspondiente.

**Retencion_Y:** DINT. Posición del punto de retención de la estación en la coordenada Y según la referencia dada por el motor PaP correspondiente.

**Retencion_Z:** DINT. Posición del punto de retención de la estación en la coordenada Z según la referencia dada por el motor PaP correspondiente.

**Retencion_THETA:** DINT. Posición angular del punto de retención de la estación en la coordenada THETA según la referencia dada por el motor PaP correspondiente.

**Argumentos de salida:**

**EN_Driver_X:** BOOL. Habilitación del motor PaP encargado del deplazamiento X.

**DIR_Driver_X:** BOOL. Dirección de giro del motor PaP encargado del deplazamiento X.

**STEP_Driver_X:** BOOL. Señal de pulsos que recibe el motor PaP encargado del deplazamiento X.

**HomeOK_X:** BOOL. Indica el estado de calibración del motor PaP encargado del deplazamiento X.

**EN_Driver_Y:** BOOL. Habilitación del motor PaP encargado del deplazamiento Y.

**DIR_Driver_Y:** BOOL. Dirección de giro del motor PaP encargado del deplazamiento Y.

**STEP_Driver_Y:** BOOL. Señal de pulsos que recibe el motor PaP encargado del deplazamiento Y.

**HomeOK_Y:** BOOL. Indica el estado de calibración del motor PaP encargado del deplazamiento Y.

**EN_Driver_Z:** BOOL. Habilitación del motor PaP encargado del deplazamiento Z.

**DIR_Driver_Z:** BOOL. Dirección de giro del motor PaP encargado del deplazamiento Z.

**STEP_Driver_Z:** BOOL. Señal de pulsos que recibe el motor PaP encargado del deplazamiento Z.

**HomeOK_Z:** BOOL. Indica el estado de calibración del motor PaP encargado del deplazamiento Z.

**EN_Driver_THETA:** BOOL. Habilitación del motor PaP encargado del giro THETA.

**DIR_Driver_THETA:** BOOL. Dirección de giro del motor PaP encargado del giro THETA.

**STEP_Driver_THETA:** BOOL. Señal de pulsos que recibe el motor PaP encargado del giro THETA.

**HomeOK_THETA:** BOOL. Indica el estado de calibración del motor PaP encargado del giro THETA.

**Electrovalvula:** BOOL. Salida encargada de controlar el actuador de la electrovávula que opera la pinza. A nivel alto, la pinza se cerrará.

**Posicion_X:** DINT. Posición actual de la pinza en la coordenada X según la referencia dada por el motor PaP correspondiente.

**Posicion_Y:** DINT. Posición actual de la pinza en la coordenada Y según la referencia dada por el motor PaP correspondiente.

**Posicion_Z:** DINT. Posición actual de la pinza en la coordenada Z según la referencia dada por el motor PaP correspondiente.

**Posicion_THETA:** DINT. Posición angular actual de la pinza en la coordenada THETA según la referencia dada por el motor PaP correspondiente.

**ESTADO:** INT. Variable de información para el usuario. Puede tomar los siguientes valores:

- -1 máquina no calibrada. Realizando el AutoTest.
- 0 esperando orden.
- 1 máquina almacenando palet.
- 2 máquina extrayendo palet.

- 3 almacén vacío.
- 4 almacén lleno.

**Argumentos de entrada/salida:**

**AutoTest:** BOOL. Orden para realizar una prueba de calibración de la máquina.

**AlmacenaPalet:** BOOL. Orden para realizar la operación de guardar un palet en el almacén

**ExtraePalet:** BOOL. Orden para realizar la operación de retirar un palet del almacén.

**Plazas:** ARRAY [1..72] OF INT. Vector de gestión del estado de las plazas del almacén de palets:

- 0 plaza libre.
- 1 plaza ocupada por palet vacío.
- 2 plaza ocupadad por producto terminado.

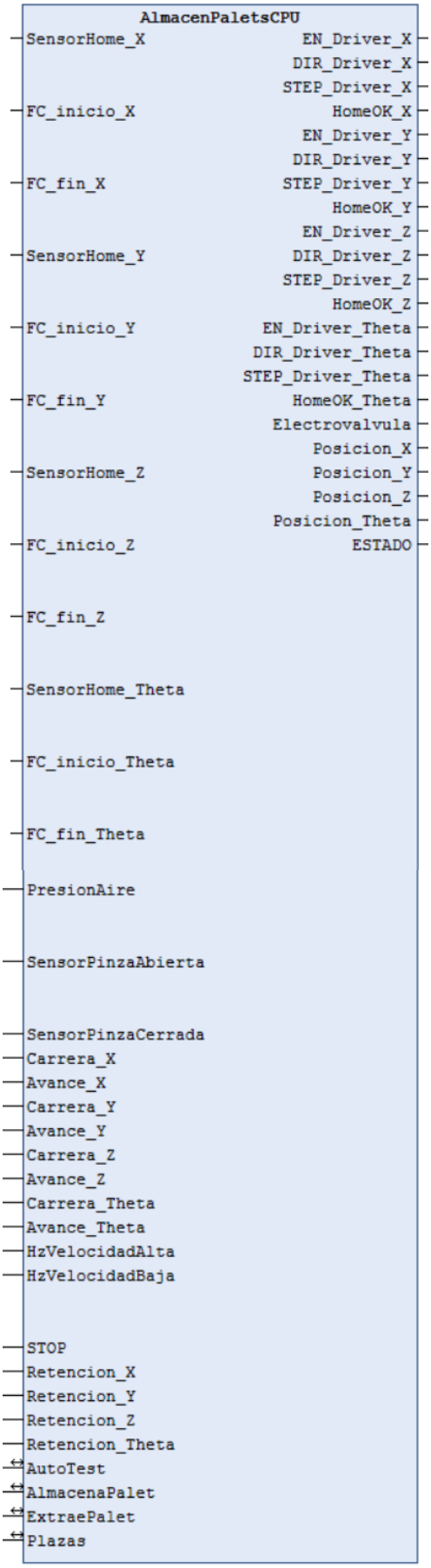


Figura 3-32. Bloque funcional AlmacenPaletsCPU

### 3.6.3 Funcionamiento

La automatización de esta máquina está igualmente basada en estados y transiciones. Desde un estado de reposo, puede recibir tres tipos de órdenes, vistos anteriormente.

**AutoTest**

- La operación de AutoTest comienza cuando dicha variable es desactivada externamente. Primeramente, realiza una prueba de la pinza, cerrando y abriendo la garra por completo.
- A continuación, realiza una prueba de calibración del motor PaP encargado del del giro THETA, seguido de los desplazamientos Y y Z, terminando con la calibración del desplazamiento X. La razón de este orden es evitar colisiones.
- Finalmente, se sitúa la garra en una posición y orientación centrada de reposo, accionando el motor Y seguido del THETA, quedando el robot situado en la posición de espera.

**AlmacenaPalet**

- Con la activación de AlmacenaPalet, siempre y cuando haya espacion libre, se analiza la memoria en busca de una plaza libre y se fijan las posiciones de cada articulación para alcanzar dicha plaza.
- Hecho esto, se lleva la pinza a la posición Retencion. Esto se hace en orden de los motores X, THETA, Y y Z.
- Llegado a la posición Retencion, donde se espera que esté el palet, se da la orden de cerrar la pinza y se eleva Z a una posición de seguridad.
- A continuación, se lleva la garra a su posición centrada, primero Y y luego THETA.
- Comienza el deplazamiento a la posición de la plaza libre. El desplazamiento se hace en el orden X, Z, THETA, Y y nuevamente Z. Estos corresponden respectivamente al desplazamiento horizontal hasta la columna de plazas, desplazamiento vertical para situarse por encima de la plaza libre, orientación de la pinza, acercamiento de la pinza para situarse sobre la plaza, descenso final para situar el palet en la plaza.
- Se abre la pinza para dejar el palet y se actualiza el estado de la plaza a ocupado por producto terminado.
- Finalmente, se lleva al robot a la posición de espera. El orden a seguir será Y, THETA, Z y X. Estos consisten en retirar la pinza, centrar la orientación, elevar el brazo y desplazarlo horizontalmente fuera de la estantería.
- Hecho esto, se desactiva la orden AlmacenaPalet y se vuelve al estado inicial.

**ExtraePalet**

- Con la activación de ExtraePalet, siempre y cuando haya palets vacíos, se analiza la memoria en busca de su ubicación en el almacén y se fijan las posiciones de cada articulación para alcanzar dicha plaza.
- Comienza el deplazamiento a la posición de la plaza libre. El desplazamiento se hace en el orden X, Z, THETA e Y. Estos corresponden respectivamente al desplazamiento horizontal hasta la columna de plazas, desplazamiento vertical para situarse frente al palet, orientación de la pinza y acercamiento de la pinza hacia el palet.
- Se da la orden de cerrar la pinza y se actualiza el estado de la plaza a vacío.
- Se eleva Z a una posición de seguridad, donde pueda maniobrar entre estanterías y se lleva la garra a su posición centrada, primero Y y luego THETA.
- Comienza el deplazamiento a la posición de Retencion. El desplazamiento se hace en el orden Z, X, THETA, Y y nuevamente Z. Estos corresponden respectivamente al desplazamiento vertical para situarse

por encima del punto de retención, horizontal hasta abandonar las estanterías, orientación de la pinza, acercamiento de la pinza para situarse sobre el punto de retención y descenso final para situar el palet en la cinta.

- Se abre la pinza para dejar el palet.
- Finalmente, se lleva al robot a la posición de espera. El orden a seguir será Z, Y, THETA, X.
- Hecho esto, se desactiva la orden ExtraePalet y se vuelve al estado inicial.

Como se ha visto en los bloques anteriores, el AutoTest comienza automáticamente con la primera ejecución, con el fin de dejar la máquina calibrada y en posición para estar operativa.

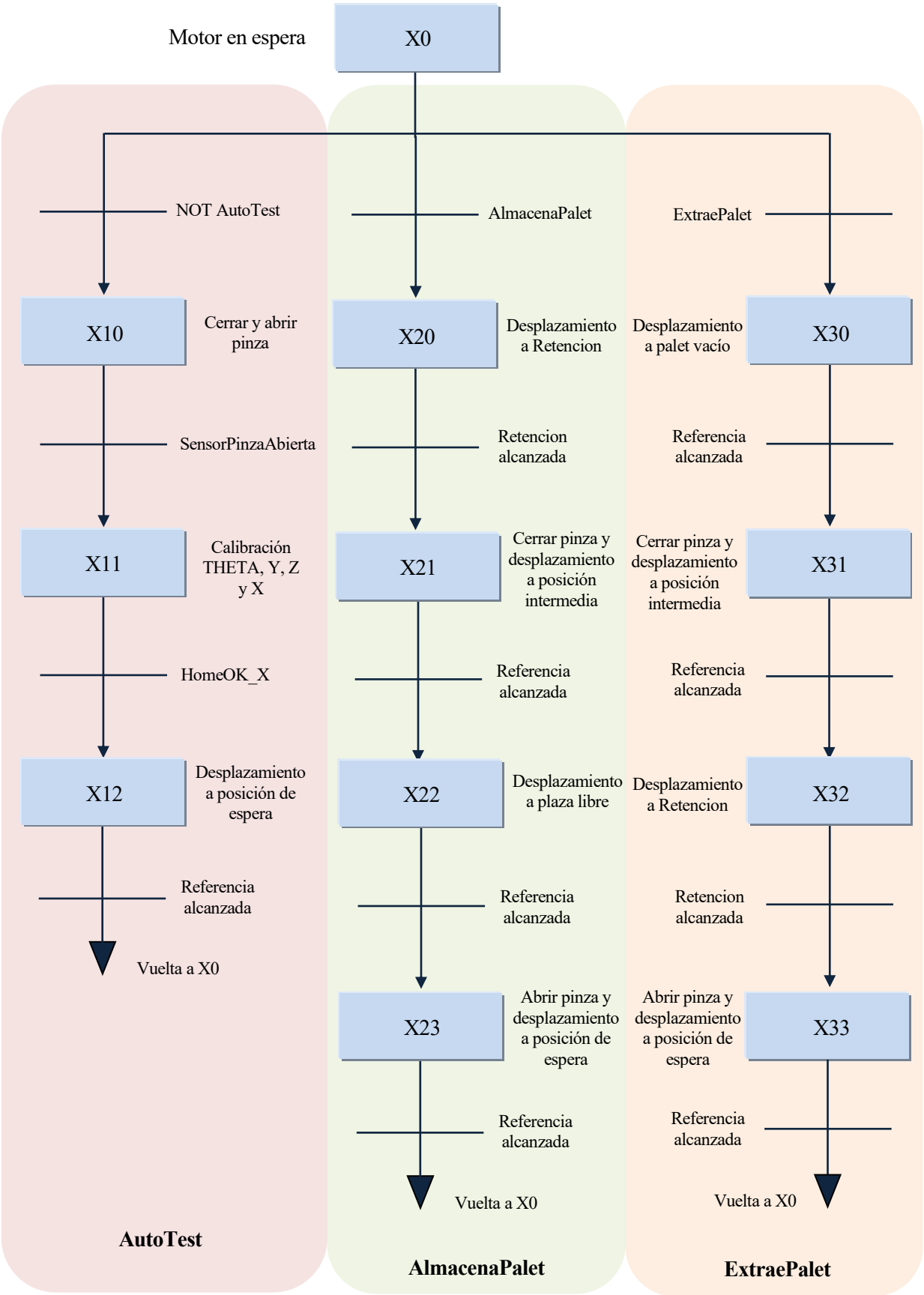


Figura 3-33. Diagrama de estados simplificado de AlmacenPaletsCPU

### 3.6.4 Simulación

El bloque AlmacenPaletsDEMO ha de encargarse de gestionar la visualización y activación de los sensores de calibración y la pinza. También de la visualización del estado de las plazas del almacén. Para ello empleará datos provenientes del controlador como la posición de los motores PaP, el estado de la pinza y el vector de estado de las plazas.

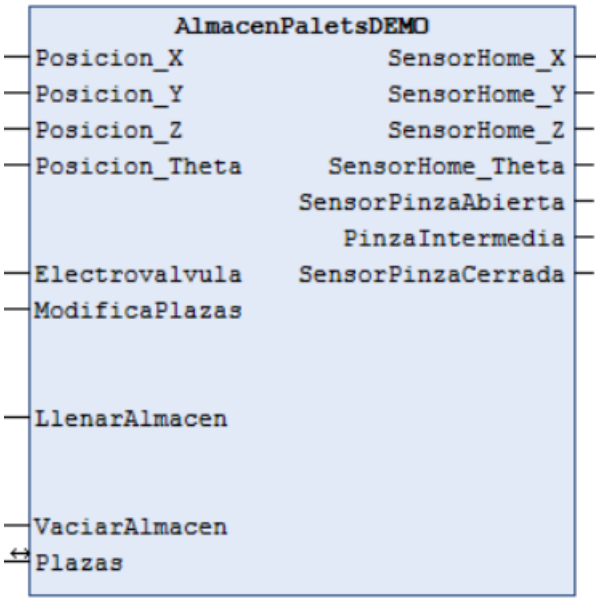


Figura 3-34. Bloque funcional AlmacenPaletsDEMO

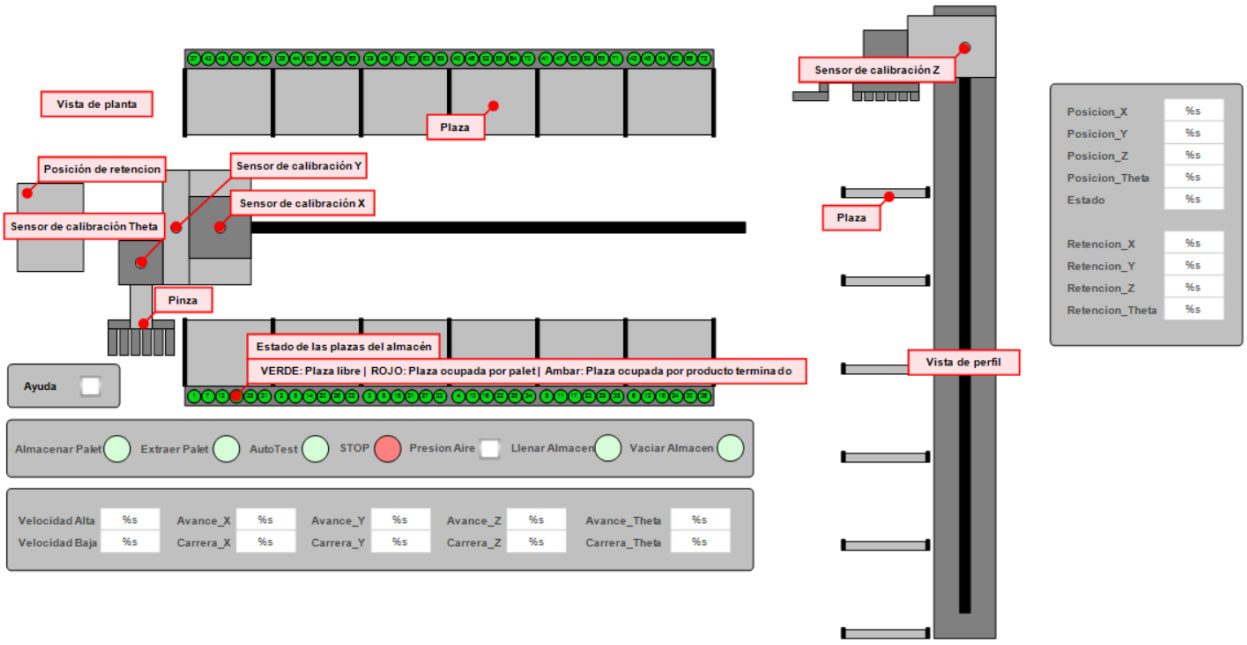


Figura 3-35. Interfaz del simulador del almacén de palets

## 3.7 AlimentadoBandejasCPU

### 3.7.1 Introducción

El bloque AlimentadorBandejasCPU controla la estación encargada de gestionar el suministro de bandejas que entra en el proceso productivo. Para dicha tarea, consta multitud de sensores y actuadores propios, pero no emplea motores PaP ni pinzas, luego su automatización es independiente de los bloques funcionales empleados hasta ahora.

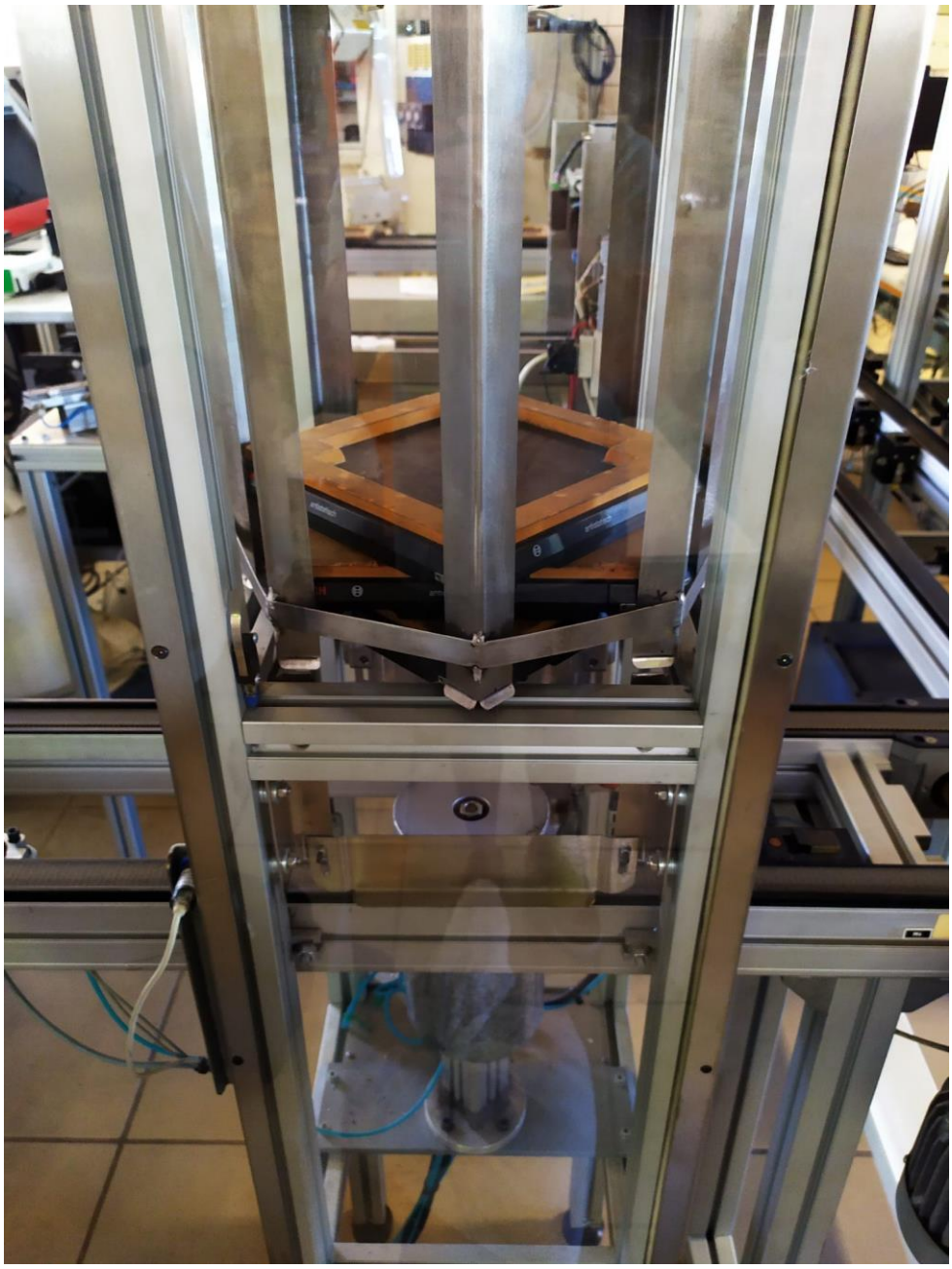

Figura 3-36. Alimentador de bandejas - Fotografía del laboratorio

En el siguiente esquema quedan representados los sensores y actuadores que intervienen en esta estación. Adicionalmente, se puede observar dos posiciones significativas, dadas por la ubicación de los retenedores. Estas son la posición previa y la posición de elevación. La primera es la posición donde se retienen bandejas esperando a pasar por la estación, mientras que la de elevación en la posición de tránsito de las bandejas que son almacenadas o extraídas del alimentador.

| | | |
|---|---|---|
| 1 | Sensor de Bandeja Previa | **S1** |
| 2 | Sensor de Bandeja Elevación | **S2** |
| 3 | Sensor Alto Retenedor Previo | **S3** |
| 4 | Sensor Bajo Retenedor Previo | **S4** |
| 5 | Sensor Alto Retenedor Elevación | **S5** |
| 6 | Sensor Bajo Retenedor Elevación | **S6** |
| 7 | Sensor Superior Cilindro Elevador | **S7** |
| 8 | Sensor Inferior Cilindro Elevador | **S8** |
| 9 | Sensor Superior Cilindro Giro | **S9** |
| 10 | Sensor Inferior Cilindro Giro | **S10** |
| 11 | Sensor de Barrido | **S11** |

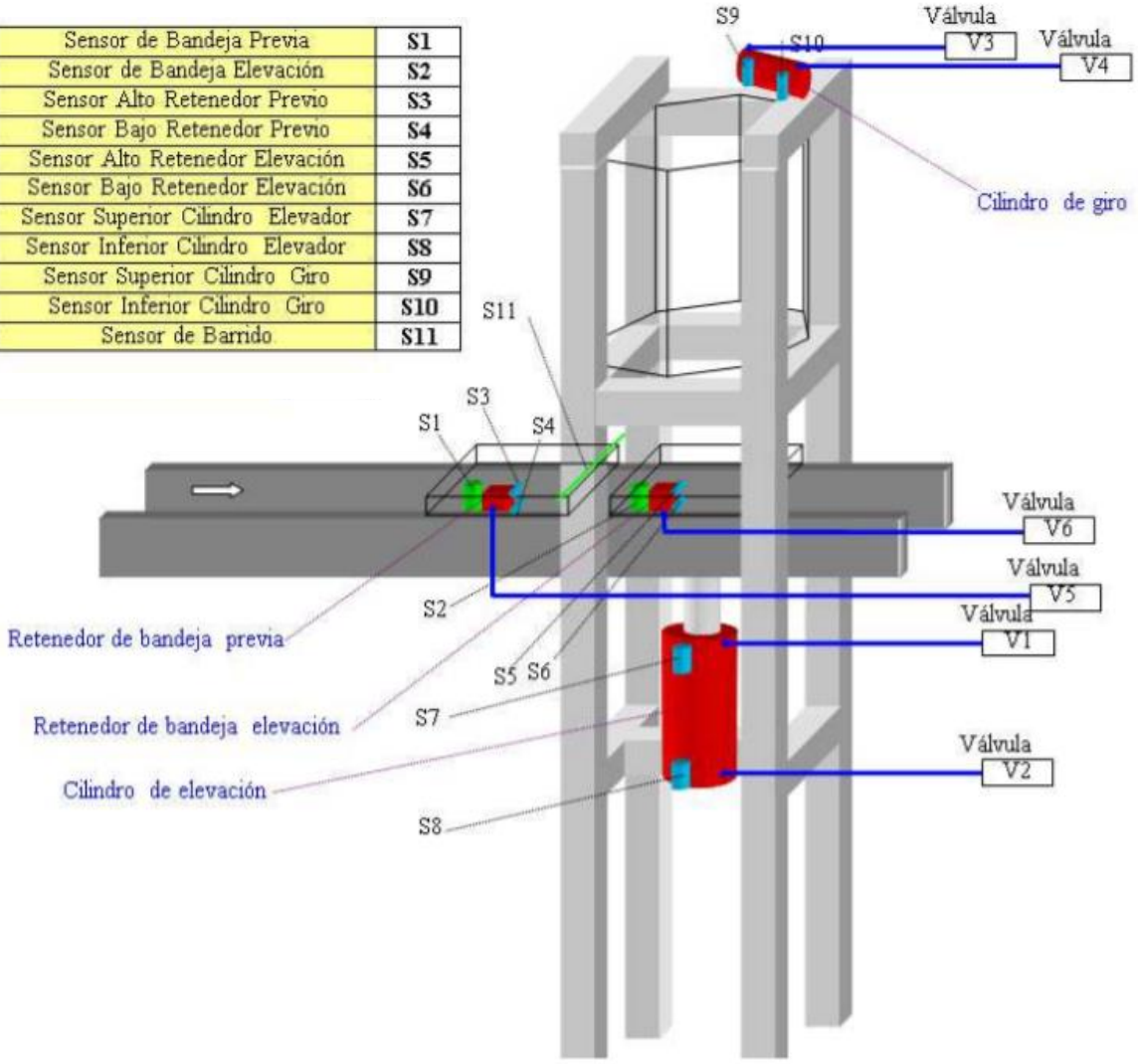


Figura 3-37. Esquema del alimentador de bandejas

### 3.7.2 Variables del bloque funcional

**Argumentos de entrada:**

**SensorBandejaPrevia:** BOOL. Sensor de presencia de bandeja en zona previa.

**SensorBandejaElevación:** BOOL. Sensor de presencia de bandeja en zona de elevación.

**SensorPalet:** BOOL. Sensor de presencia de palet sobre bandeja. Normalmente activo, marca el paso del palet cuando se interrumpe la señal.

**SensorAlto_RetenedorPrevio:** BOOL. Sensor de estado del retenedor previo. A nivel alto cuando el retenedor se encuentra extendido.

**SensorBajo_RetenedorPrevio:** BOOL. Sensor de estado del retenedor previo. A nivel alto cuando el retenedor se encuentra retraido.

**SensorAlto_RetenedorElevacion:** BOOL. Sensor de estado del retenedor de elevación. A nivel alto cuando el retenedor se encuentra extendido.

**SensorBajo_RetenedorElevacion:** BOOL. Sensor de estado del retenedor de elevación. A nivel alto cuando el retenedor se encuentra retraido.

**SensorAlto_Piston:** BOOL. Sensor de estado del pistón. A nivel alto cuando el pistón se encuentra extendido.

**SensorBajo_Piston:** BOOL. Sensor de estado del pistón. A nivel alto cuando el pistón se encuentra retraido.

**SensorAlto_Giro:** BOOL. Sensor de estado del actuador de giro de bandejas. A nivel alto cuando el giro alcanza el límite superior.

**SensorBajo_Giro:** BOOL. Sensor de estado del actuador de giro de bandejas. A nivel alto cuando el giro alcanza el límite inferior.

**CintaEnMovimiento:** BOOL. Indicación del estado de la cinta. A nivel alto si está en movimiento.

**Argumentos de salida:**

**SubirPiston:** BOOL. Controla el actuador que extiende el pistón.

**BajarPiston:** BOOL. Controla el actuador que retrae el pistón.

**GiroPositivo:** BOOL. Controla el actuador de giro de bandejas. A nivel alto, gira hacia el límite superior.

**GiroNegativo:** BOOL. Controla el actuador de giro de bandejas. A nivel alto, gira hacia el límite inferior.

**RetenedorPrevio:** BOOL. Control del retenedor previo. Se encontrará extendido a nivel alto.

**RetenedorElevación:** BOOL. Control del retenedor de elevacion. Se encontrará extendido a nivel alto.

**EspacioOcupado:** BOOL. Variable de información para el usuario. A nivel alto indica que la zona de elevación está ocupada por una bandeja.

**ESTADO:** INT. Variable de información para el usuario. Puede tomar los siguientes valores:

- 0 esperando orden.
- 1 máquina almacenando bandeja.
- 2 máquina extrayendo bandeja.

**Argumentos de entrada/salida:**

**AlmacenarBandeja:** BOOL. Orden para realizar la operación de guardar una bandeja en el alimentador

**ExtraerPalet:** BOOL. Orden para realizar la operación de retirar una bandeja del alimentador.

**NumeroBandejas:** INT. Conteo del número de bandejas disponibles en el alimentador.

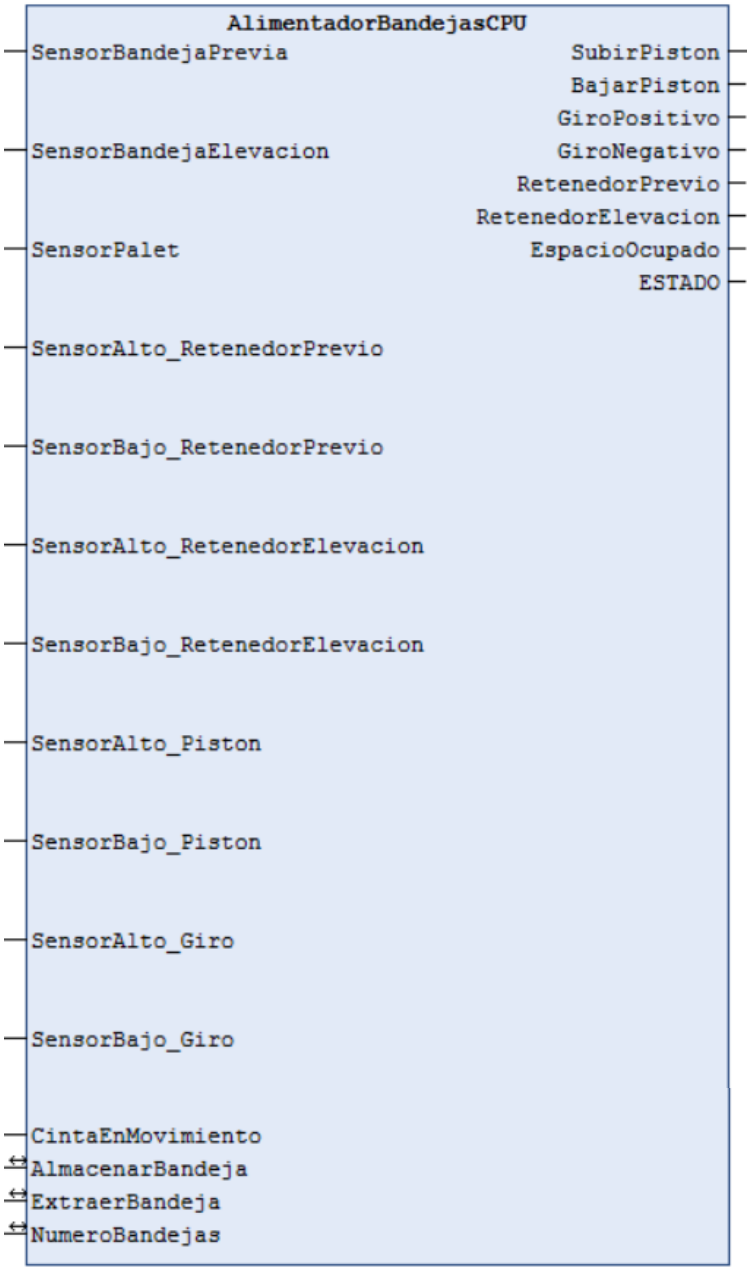


Figura 3-38. Bloque funcional AlimentadorBandejasCPU

### 3.7.3 Funcionamiento

A diferencia de las demás estacines, el alimentador de bandejas tiene su propio sistema de retenedores y sensores, necesarios para sincronizar la salida y entrada de bandejas. Por este motivo, el control de estos es local e independiente del resto de dispositivos de sincronización a lo largo del sistema de transporte. Por parte del alimentador de bandejas propiamente dicho, su automatización está igualmente basada en estados y transiciones. Desde un estado de reposo, puede recibir tres tipos de órdenes, vistos anteriormente.

**AlmacenarBandeja**

- Con la activación de AlmacenaPalet, siempre y cuando se cumplan las condiciones de que haya una bandeja en la posición de elevación y que no transporte un palet; se da orden al pistón de que suba y eleve la bandeja.
- Hecho esto, el motor de giro de bandejas, dependiendo de en qué extremo se encuentre, gira en un sentido o en otro. Con esto, rota la pila de bandejas un octavo de vuelta, dejando la bandeja recién añadida sujeta por el soporte.
- Terminado el giro, el pistón baja a su posición inicial, se desactiva la orden y se vuelve al estado inicial.

**ExtraerBandeja**

- Con la activación de ExtraePalet, siempre y cuando se cumplan la condición de uqe el espacio de trabajo no esté ocupado por bandejas pasantes; se da orden al pistón de que suba.
- Hecho esto, el motor de giro de bandejas, dependiendo de en qué extremo se encuentre, gira en un sentido o en otro. Con esto, rota la pila de bandejas un octavo de vuelta, dejando la bandeja inferior libre.
- Terminado el giro, el pistón desciende con la bandeja a su posición inicial, se desactiva la orden y se vuelve al estado inicial.

Mientras tanto, se va gestionando la llegada y salidade bandejas a través de los sensores y retenedores en la posición previa y de elevación. Estas operaciones pasan por activar el retenedor previo si el espacio de trabajo está ocupado, dar salida a las bandejas que acaban de ser extraídas del alimentador, dejar pasar a las bandejas que llegan transportando un palet, etc.

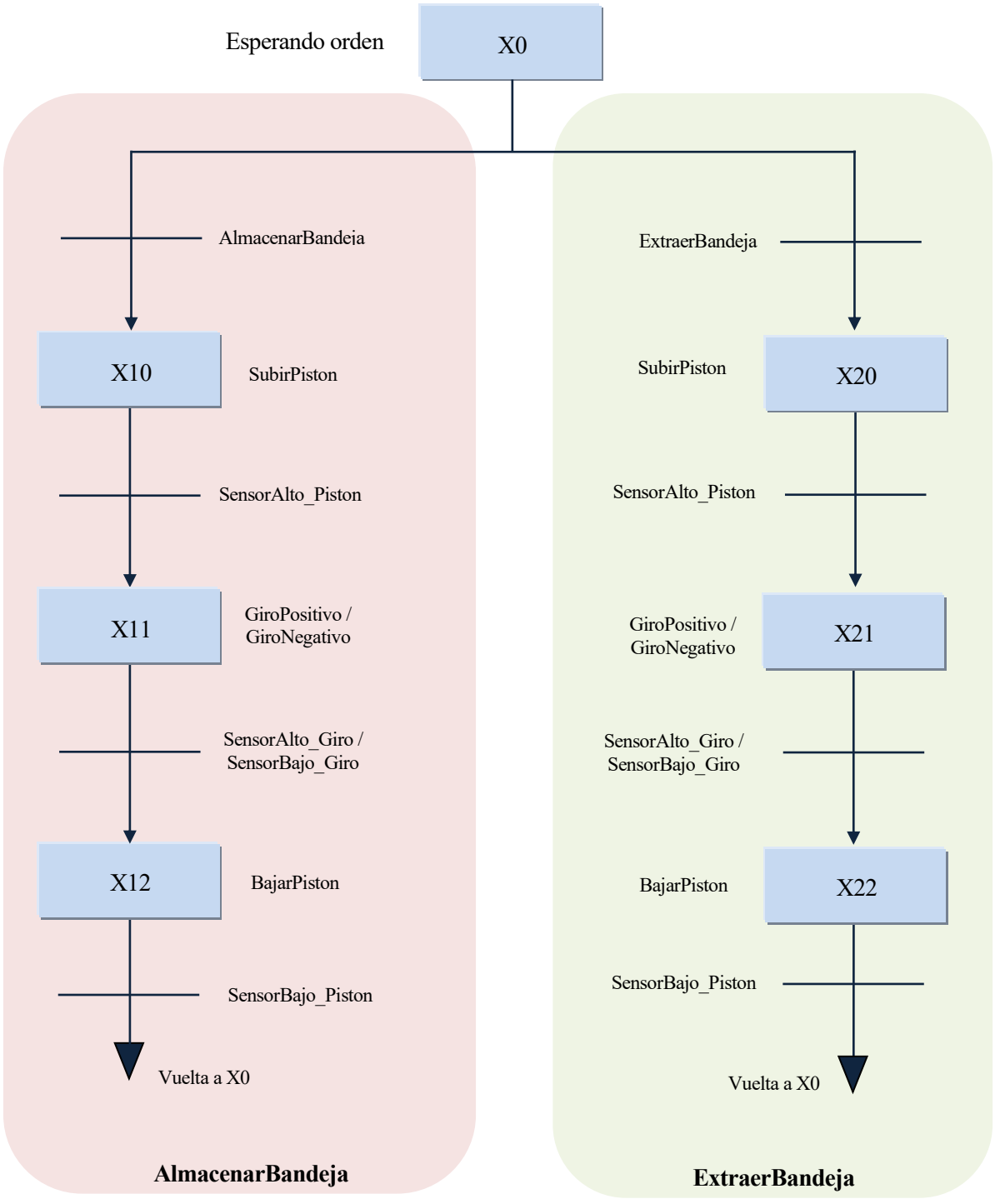


Figura 3-39. Diagrama de estados simplificado de AlimentadorBandejasCPU

### 3.7.4 Simulación

El bloque AlimentadorBandejasDEMO ha de encargarse de gestionar la visualización y activación de todos los sensores de la estación. Para ello empleará datos provenientes del controlador como el estado de los actuadores y las órdenes dadas a través de la propia interfaz del simulador.

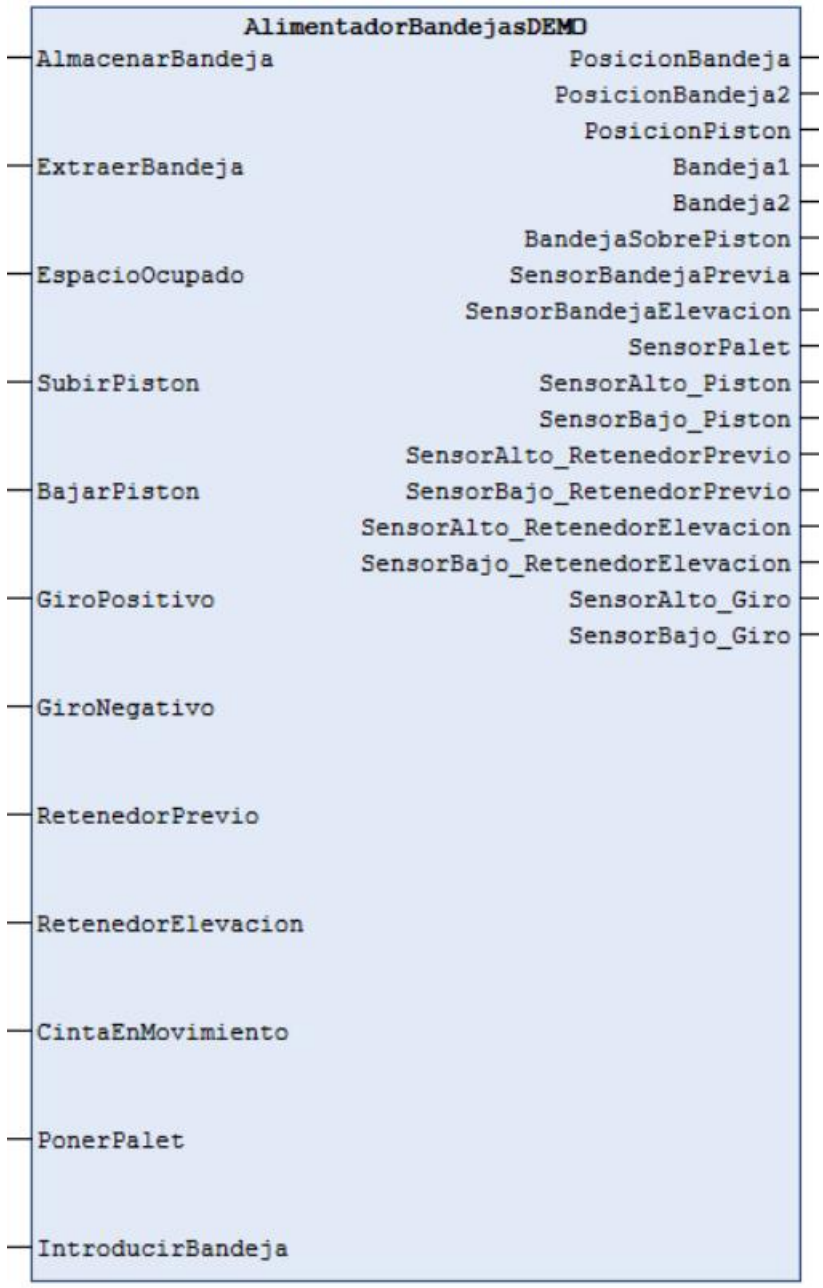


Figura 3-40. Bloque funcional AlimentadorBandejasDEMO

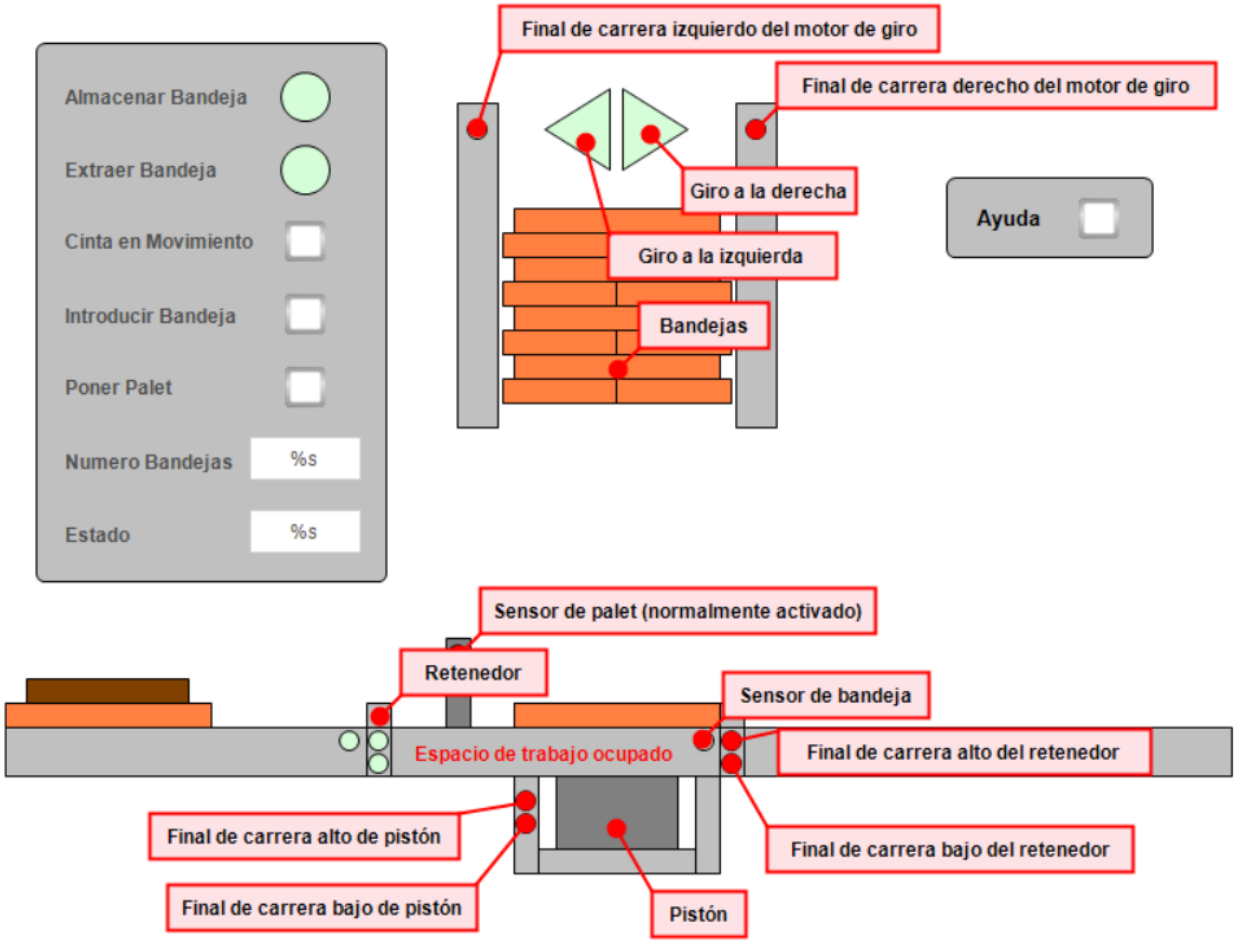


Figura 3-41. Interfaz del simulador del alimentador de bandejas

# 4 CÉLULA DE FABRICACIÓN COMPLETA

En este capítulo se verá un ejemplo de una célula de fabricación completa, empleando los bloques funcionales vistos hasta ahora. El fin es dar una idea de las posibilidades que ofrece la librería, empleando las máquinas vistas hasta ahora.

Acompañado de la automatización de la célula completa, se ha diseñado una interfaz del simulador virtual, encargada de simular internamente el comportamiento cada una de las estaciones que intervienen en el proceso de producción.

## 4.1 Ejemplo de célula

La célula de fabricación flexible diseñada incorpora todas las estaciones de la librería. Estas están dispuestas entorno a cuatro cintas de tranporte, que cierran el ciclo de producción. Los elementos de producción empleados son los vistos anteriormente, procedentes del laboratorio: las piezas, los palets y las bandejas.

Entrando en el proceso de producción, se ha planteado de la siguiente forma:

- El alimentador de bandejas situa una bandeja en la cinta 1.
- La bandeja vacía pasa por la cinta 1 hasta la cinta 2.
- En la cinta 2, llega a la estación del almacén de palets. Este extrae un palet y lo coloca sobre la bandeja vacía.
- A continuación en la misma cinta, el plalet vacío llega a la estación del pórtico. Este está acompañado de dos máquinas alimentadoras de piezas. Estas se sincronizan para que la garra del pórtico situe una de las piezas en el primer soporte del palet.
- Del portico sale el producto semielaborado y llega a la cinta 3.
- En la cinta 3 se encuentra la estación del Scorbot. Al igual que en la estación enterior, el Scorbot se sincroniza con otros dos alimentadores de piezas y se encargará de rellenar el segundo soporte del palet con otra pieza.
- Del Scorbot sale el producto terminado y llega a la cinta 4.
- En la cinta 4, el product pasa primeramente por la estación Scara. Aquí será objeto de un control de calidad, donde se comprobará por inspección si las piezas son las correctas y se han ensamblado correctamente.
- Por último, el producto llega a la estación del puente grúa. Esta cuenta con piezas de los 4 tipos apiladas y su propósito será realizar las correcciones que procedan, en caso de que se hubiese detectado algún error en la estación anterior.
- Terminado el ciclo, el producto Vuelve a pasar por la cinta 1 y 2 hasta llegar al almacén de palets, que almacenará el product terminado.

- La bandeja que queda vacía en la cinta de transporte recibirá otro palet para continuar la producción, o será almacenada en el almacén de bandejas cuando no se necesite más.

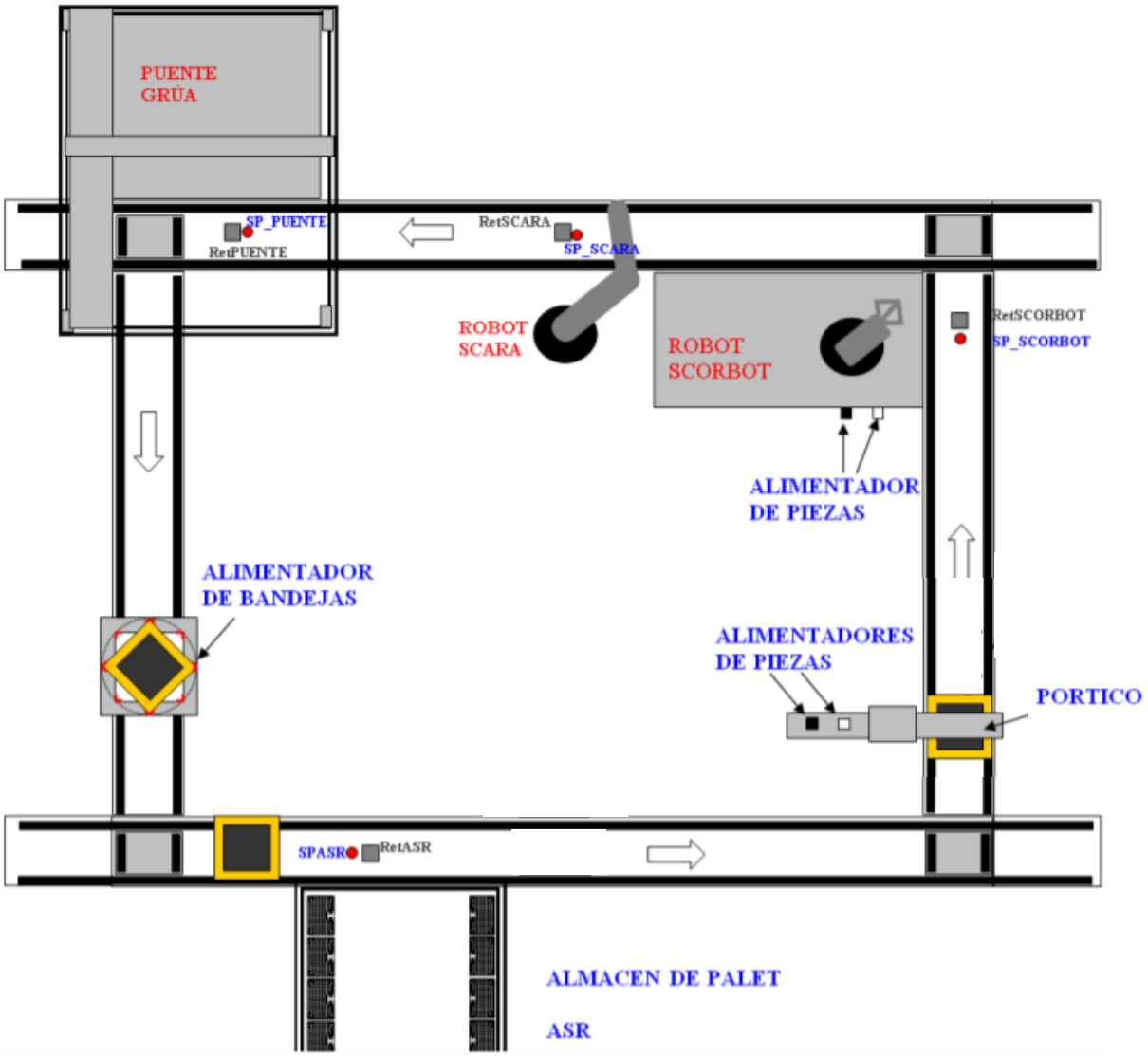


Figura 4-1. Esquema de la célula de fabricación completa

## 4.2 Funcionamiento

La automatización de la célula completa se basa principalmente en tres aspectos, dar órdenes a las distintas estaciones, gestionar la información recibida de estas y controlar las cintas de transporte junto a los sensores y retenedores para sincronizar el movimiento de bandejas con las distintas operaciones del proceso productivo.

Los retenedores están situados frente a cada una de las estaciones, junto a un sensor capacitivos de presencia de bandeja, para saber cuando ha de activarse. Esto retenedores permiten fijar la bandeja en una posición para que la estación pueda realizar operaciones sobre esta incluso con la cinta activa. La cinta deslizaría bajo la bandeja sin moverla de la posición de retención.

Figura 4-2. Retenedores - Fotografía del laboratorio

Adicionalmente, se emplean sensores de presencia normalmente activos para la detección de palets. Esta información es significativa, puesto que la condición de operación de algunas estaciones depende de la presencia o no de un palet. Por ejemplo, el almacén de palets almacenará el palet si la bandeja en retención lleva un palet encima, o bien extraerá uno del almacén si la bandeja está vacía.

Cabe destacar que el alimentador de bandejas gestiona localmente sus sensores y retenedores, luego no se necesita incluir su automatición a nivel de célula completa.

Volviendo a las tareas realizadas por las estaciones, en este ejemplo se ha decidido que el proceso productivo se realice para un único tipo de producto y sin errores.

Figura 4-3. Célula de fabricación completa - Fotografía del laboratorio

## 4.3 Simulación

Como ya se adelantó, la simulación de la célula completa necesita de los bloques de simulación de cada una de las estaciones. Aunque la interfaz sea distinta, se necesitan de estos bloques DEMO para gestionar la activación de los sensores de cada estación en este entorno virtual.

Por su parte, tanto para el control de la animación de la interfaz como para los sensores incluidos a nivel de célula completa, se ha diseñado el bloque CelulaCompletaDEMO. Para ello, empleará datos provenientes del controlador como el estado de los actuadores y las órdenes recibidas por las estaciones.

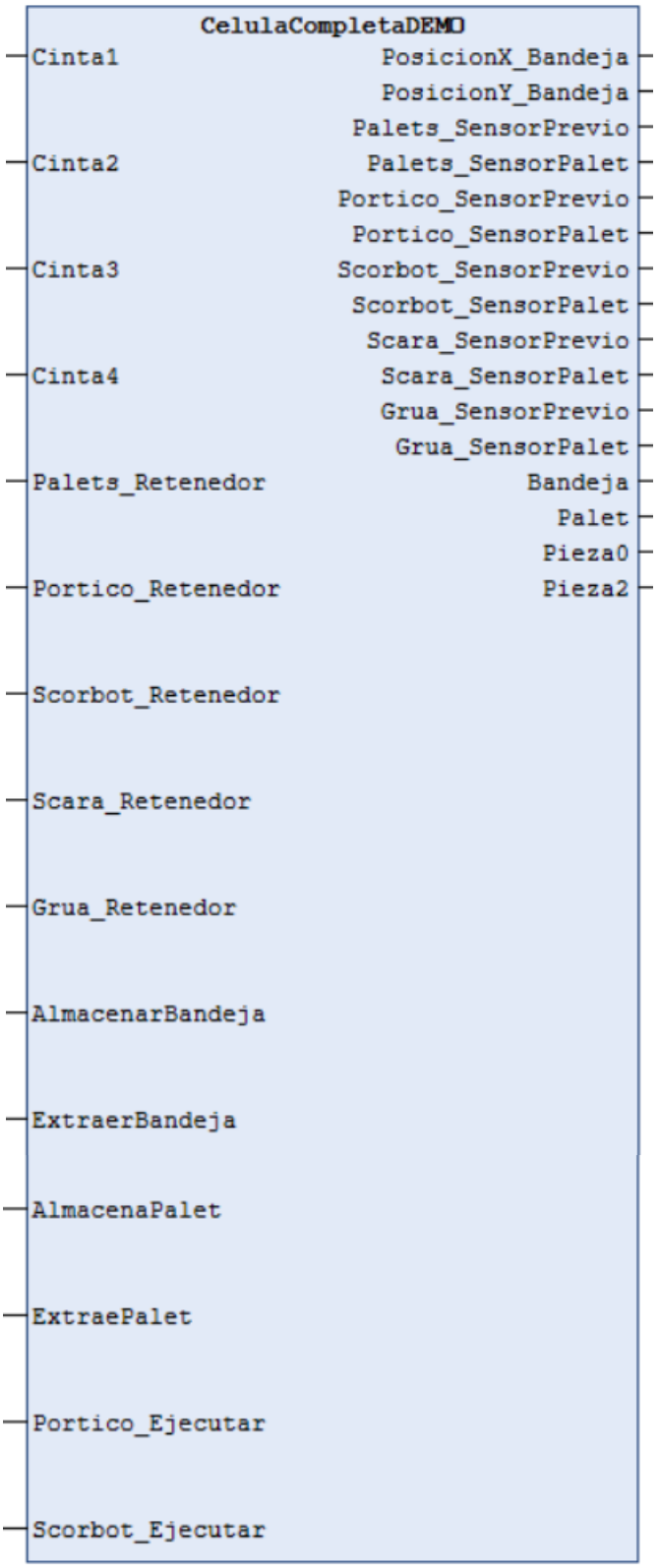


Figura 4-4. Bloque funcional CelulaCompletaDEMO

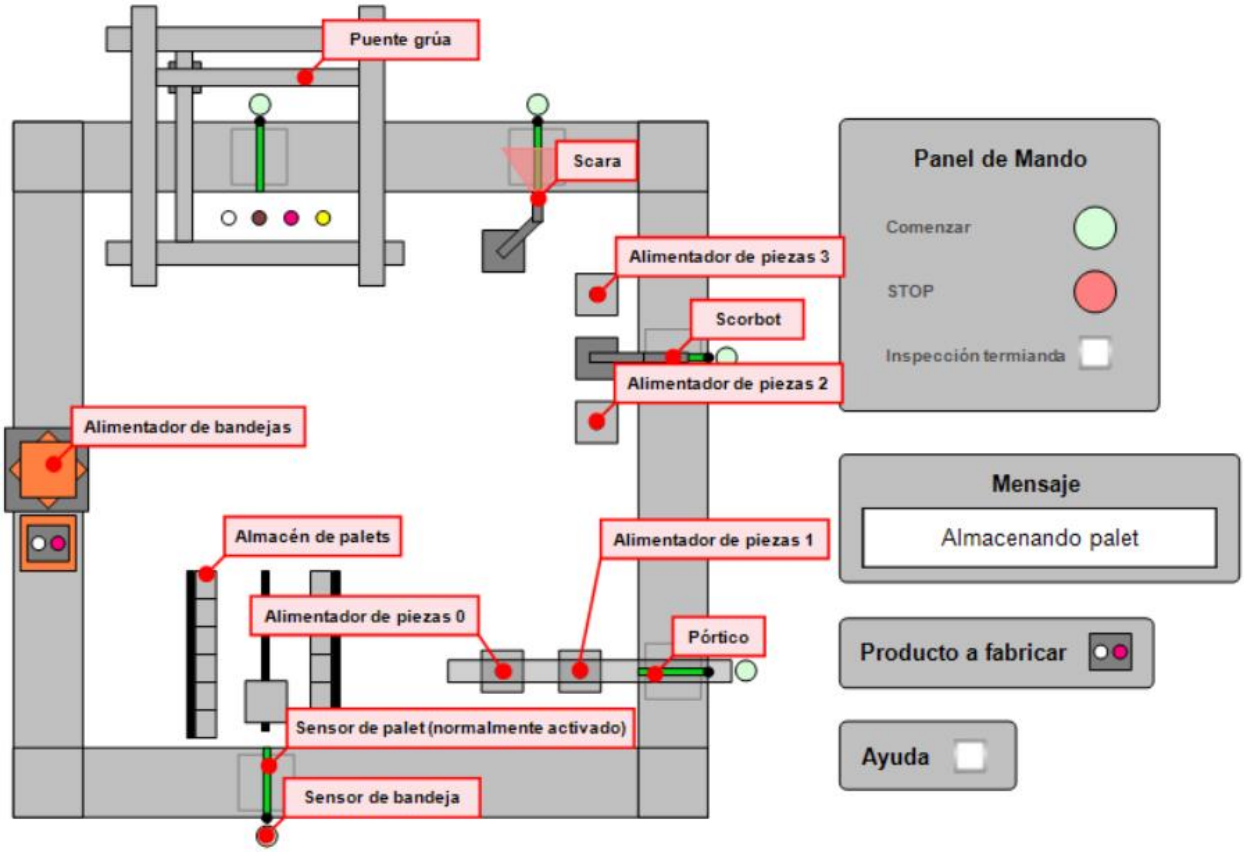


Figura 4-5. Interfaz del simulador de la célula completa

# 5 CONCLUSIONES

Una vez repasado todo lo que da de sí el proyecto, se pueden hacer algunas valoraciones finales. Empezando por el planteamiento del mismo, sería interesante comprobar la adaptabilidad de la librería a una situación real. Es decir, evaluar las modificaciones requeridas para poder implementar la automatización de una célula de fabricación flexible en un PLC real, con sus entradas y salidas conectadas a los equipos reales y partiendo del trabajo realizado en este proyecto.

Si pensamos en esa puesta en práctica en una situación industrial real, teniendo en cuenta el proceso de producción que se quiere diseñar, se pueden configurar distintas células de fabricación. Las posibilidades son incontables, añadiendo modificaciones en los bloques de estaciones, creando bloques para estaciones nuevas a partir de los bloques funcionales básicos, reconfigurando el orden y sincronización de las estaciones empleadas, duplicando estaciones existentes con el fin de tener redundancia operativa y mejorar la disponibilidad de la planta.

Por ejemplo, dependiendo de las pretensiones del diseñador de la célula de fabricación flexible, podría ser interesante sustituir la estación Scorbot por una estación de pórtico o puente grúa adicional y viceversa. También se podrían incluir más alimentadores de pieza para aumentar la variedad de piezas disponible.

También se podrían incluir nuevas cintas para procesos de producción más largos, o bien para llevar varios procesos en paralelo. Estos podrían tener sus estaciones independientes o bien compartir algunas, lo cual supondría un interesante reto de sincronización de procesos, intentando no descuidar ninguno de los procesos llevados en paralelo.

Por su parte, la estación de inspección que emplea el robot Scara se podría reconvertir en una máquina en una estación de transporte de piezas instalando una pinza en lugar de la cámar e incluyendo un bloque OperaPinzaCPU.

Refinar la elección de estas variables supondría una optimización del tiempo de producción, de la disponibilidad de la planta ante posibles fallos o del gasto de recursos entre otros, repercutiendo favorablemente en el aspecto económico.

Desde el punto de vista académico, se pueden explorar también otras modificaciones interesantes que permitan experimentar con las posibilidades de la célula, gracias a la flexibilidad que ofrece la librería. Se pueden construir estaciones desde cero que empleen motores PaP y aprovechar los bloques básicos. Dichas estaciones podrían estar alejadas de la perspectiva de célula de fabricación, pudiendo ser interesantes por sí solas.

Los simuladores también son realmente valiosos desde este punto de vista, gracias a la posibilidad de ejecutarlos virtualmente, podiendo prescindir de los equipos reales. En caso de que sí se decidiesen emplear, el simulador permite depurar fallos de funcionamiento, poniendo de manifiesto el estado de los equipos y los errores de instalación que pudiesen presentarse.

# GLOSARIO