INSTITUTO TECNOLÓGICO DE COSTA RICA

Escuela de Ingeniería Electrónica

EL-5616 Proyecto de Graduación

Acople funcional de la interfaz háptica Novint Falcon a un proceso Teleoperado de Automatización.

Informe de Proyecto de Graduación para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura

Allan Navarro Garita
200307080

Cartago, Septiembre de 2010

INSTITUTO TECNOLOGICO DE COSTA RICA

ESCUELA DE INGENIERIA ELECTRONICA

PROYECTO DE GRADUACIÓN

TRIBUNAL EVALUADOR

Proyecto de Graduación defendido ante el presente Tribunal Evaluador como requisito para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura, del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal

Ing. Gabriela Ortiz León, M.Sc.

Profesora lectora

Lic. Ing. Marvin Hernández Cisneros

Profesor lector

Ing. Arys Carrasquilla Batista, M.Sc.

Profesora asesora

Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica

Cartago, 1 de Setiembre de 2010

Declaración de Autenticidad

Declaro que el presente documento ha sido realizado completamente por mi persona.

Se ha utilizado literatura pertinente, y se introduce además conocimientos propios. La bibliografía es indicada por medio de referencias en los casos que ha sido empleada para formular ideas.

Asumo la responsabilidad total por el trabajo presentado y su contenido.

Cartago, 01 de setiembre de 2010

Allan Navarro Garita
Cédula: 112570075

Resumen

Las interfaces hápticas permiten percibir sensaciones de fuerza, peso, presión, temperatura y demás, interactuando con un entorno virtual o real. En un video juego, por ejemplo, se pueden percibir golpes, choques, roces y todo tipo de eventos táctiles que concuerdan de forma sincronizada con las órdenes emitidas al programa, lo que se está también observando en la pantalla y con los sonidos escuchados.

La intención de este Proyecto fue acoplar la interfaz háptica Novint Falcon a un proceso de Automatización, cuyas acciones eran administradas por un PLC FESTO FEC Standard. Este último actuó como intermediario entre la PC, donde estaba acoplado el dispositivo háptico, y los actuadores.

Un programa creado en C++ resultó necesario para configurar las acciones de la interfaz. La programación de un cliente DDE conectado al servidor de datos leídos desde el PLC fue también útil para lograr el intercambio de información entre las aplicaciones.

Al final, resultó posible comandar el Brazo Giratorio de la Unidad MPS FESTO Distribución a la vez que se recibían sensaciones perceptibles generadas a partir de la interacción del equipo con su entorno.

Palabras clave: Interfaz Háptica, PLC, Automatización, Capa de Abstracción, DDE, HMI, MPS.

Summary

Haptic interfaces allow perceiving of sensations such as force, weight, pressure, temperature, from virtual or real environments. In a videogame, for instance, there can be perceived hits, crashes, friction and all sort of tactile events that agree with the orders issued by the program and what is being observed in the screen and with the sounds being listened.

The main goal of this Project was to connect the haptic interface Novint Falcon to a process of Automation, which actions were managed by a PLC FESTO FEC Standard. This controller acted like an intermediary between the PC, where the haptic device was connected, and the actuators.

A program created in C++ was necessary for configuring the actions of the haptic interface. The programming of a DDE client connected to a server with data from the PLC was also useful to achieve the information exchange between both applications.

At the end, it was possible to command the Rotary Device of the MPS FESTO Unit. At the same time, perceptible sensations generated from the interaction of the equipment and its environment were received.

Keywords: *Haptic Interface, PLC, Automation, Abstraction Layer, DDE, HMI, MPS.*

Dedicatoria

A mi familia y especialmente a mi madre, siempre mi inspiración y la razón para seguir adelante.

Agradecimiento

A Dios. Con Él, por Él y para Él todo esto ha sido posible.

A mis profesores, personal del TEC y a todos a quienes debo infinidad de apoyo.

A la profesora Arys, la mejor Asesora que pude haber tenido.

A Pilar, mi apoyo y compañía en todo momento durante la realización de este Proyecto.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1. INTRODUCCIÓN

1.1. Generalidades del Proyecto

En el Instituto Tecnológico de Costa Rica, específicamente en la Escuela de Ingeniería Electrónica, existe el Laboratorio de Investigación en Robótica y Automatización (LIRA). Ahí se dispone de equipo para aproximar a los estudiantes al área del control automático en el ámbito industrial a través de la interacción con sistemas de simulación y aprendizaje a pequeña escala.

El equipo disponible corresponde a un Sistema de Producción Modular (MPS por las siglas en inglés de *Modular Production System*) de la marca FESTO. Cada uno de estas unidades, con actuadores neumáticos, son administradas por medio de un Controlador Lógico Programable (PLC por las siglas en inglés de *Programmable Logic Controller*). En la imagen de la Figura 1 se muestra específicamente la llamada Unidad de Distribución (*Distributing Unit*).

Figura 1. Unidad de Distribución MPS FESTO. Tomado de [3].

La función básica de la Unidad de Distribución, tal y como se detalla en [3] es la de recoger objetos apilados y transferidos por el Módulo de la Recámara de Carga (*Stack Magazine Module* en inglés) para que el siguiente módulo en la secuencia, el Módulo de Cambio (*Changer Module*) los tome y transporte a otra posición. De ambos, el de mayor importancia para propósitos del Proyecto fue el Módulo de Cambio o Brazo Giratorio; una descripción de sus componentes funcionales se muestra en la Figura 2.

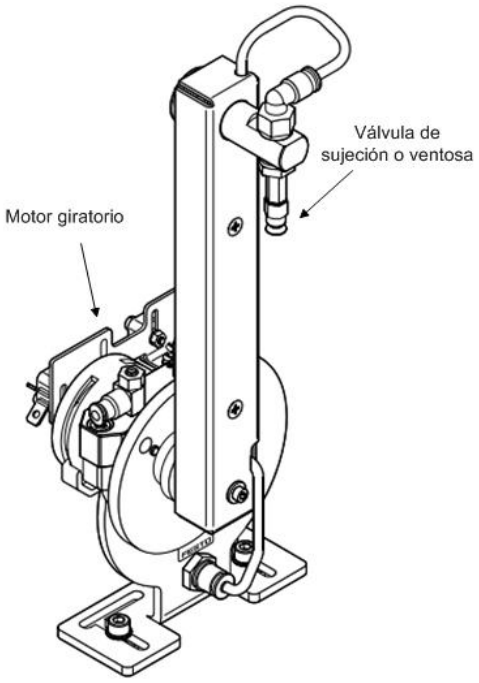

Figura 2. Brazo Giratorio. Tomado de [3].

El Motor Giratorio (*Swivel Drive* o *Rotary Drive*) y la Válvula de Succión o Ventosa (*Suction Cup*) son actuadores neumáticos, ambos. La rutina básica o estándar predispuesta para el Módulo de Cambio de la Unidad de Distribución es la que se presenta simplificada, a manera de ejemplo, en el diagrama de flujo de la Figura 3.

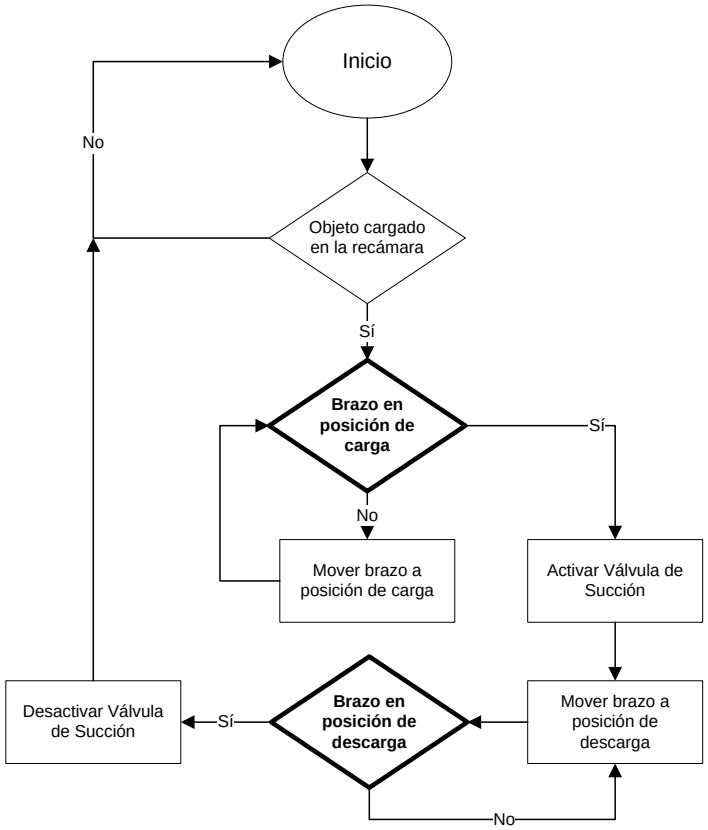

Figura 3. Diagrama de flujo para la secuencia simplificada del Módulo de Cambio.

Por otro lado, recientemente se ha equipado el LIRA con equipo para realizar investigación en el área de la Teleoperación asistida por interfaces hápticas. Específicamente, estos dispositivos son los controladores Falcon, diseñados por la compañía Novint originalmente para sus juegos de video. Entre las principales motivaciones para el Proyecto, estaba lograr un acople funcional de este dispositivo a un sistema de Automatización.

1.2. Requerimientos iniciales del Proyecto

En un principio, el reto fue lograr comandar el funcionamiento de la Unidad de Distribución MPS FESTO, controlada por un PLC Siemens S7-300 u otro PLC FESTO FEC Standard, a través de las órdenes recibidas por medio de la interfaz háptica Novint Falcon y así, obtener una retroalimentación de fuerza correspondiente a la interacción de los actuadores con el entorno.

La pregunta que dio origen al planteamiento del Proyecto se define a continuación:

¿Es posible lograr el acople funcional de la interfaz háptica Novint Falcon a un proceso Teleoperado de Automatización?

Se definió esta interrogante con el fin de tener descripción concreta acerca de lo que se buscaba. Esta pregunta es el fundamento de las etapas posteriores, y finalmente se logró darle una respuesta satisfactoria a través del Proyecto.

1.3. Aspectos influyentes en el Proyecto

Las capacidades dispuestas por el fabricante para trabajar con los MPS están limitadas a una serie de rutinas ya predefinidas, las cuales se pueden implementar de forma sencilla a través de las herramientas de software disponibles e incluidas con la compra del equipo. Ya se ha trabajado en la manera de abrir estas posibilidades a través de Proyectos de Graduación anteriores, sin embargo queda mucho por hacer en este aspecto.

El funcionamiento estándar para el Módulo de Cambio de la Unidad de Distribución MPS FESTO, descrito por el diagrama de flujo de la Figura 3 de forma simplificada, presenta una característica a considerar. En él, se encuentra resaltado el hecho de originalmente, sólo se admite un movimiento completo en todo el rango de giro desde la posición de carga hasta la posición de descarga y viceversa. Para lograr el acople funcional de la interfaz háptica al sistema era necesario romper este rígido esquema.

Adicionalmente, se presentaba el hecho de que la interfaz háptica Novint Falcon, disponible para la realización del Proyecto, es un dispositivo desarrollado para ser el mando de control en entornos virtuales en general. El soporte disponible de la compañía está orientado a programadores para juegos de video y programas de simulación en el mejor de los casos. Fue necesario idear la forma de establecer una programación compatible en estas condiciones como respuesta a los requerimientos planteados.

1.4. Acople funcional de la interfaz háptica

Se propuso ampliar el rango de acción del control predefinido de fábrica para el equipo, utilizando hasta donde hubiera posibilidad los recursos existentes en el LIRA. Se pensó manipular el funcionamiento del brazo de la Unidad de Distribución para lograr cambiar su rango de movimiento y hacer otro tipo de control sobre su mecanismo de acción.

Este actuador tiene por defecto dos posiciones estables y el movimiento se da entre

ambas, sin etapas intermedias. El control responde a órdenes discretas que involucran sensores de la posición en un extremo y variables para indicar el arranque eventual del movimiento hacia el otro. Este funcionamiento no involucra un sensor de la posición relativa a alguno de los extremos, únicamente se dispone de información acerca de si se está o no en uno de ellos.

Se pensó entonces crear un tipo de control continuo, que permitiera mover el mecanismo en un rango flexible y lograr así una respuesta a las órdenes del operador de manera continua. Esto fue para permitir el movimiento parcial del brazo en un sentido, detenerlo eventualmente en alguna posición intermedia y poder devolverlo aunque no hubiera completado el desplazamiento en todo su rango de acción. Para ello fue necesario alterar la programación predispuesta y "recomendada" por las herramientas de software, la documentación y en general, todo el mecanismo.

Por otro lado, en el caso de la interfaz háptica empleada, a partir del SDK de Novint se obtiene una base para la programación a través de HDAL (*Haptic Device Abstraction Layer,* por sus siglas en inglés). Con lo cual se dispone de un conjunto de funciones programables en lenguaje de alto nivel, específicamente C++. Estas se encargan de "hablar" con el dispositivo e interpretar señales eléctricas y cadenas de datos en una capa de abstracción desarrollada por el fabricante y que facilita el trabajo a alto nivel con el Falcon.

CAPÍTULO 2. META Y OBJETIVOS

2.1. Meta

Estimular la percepción háptica del operador en un proceso de Teleoperación.

2.2. Objetivo General

Diseñar un sistema de Teleoperación que ofrezca una percepción háptica del entorno sobre el cual se trabaja.

2.3. Objetivos Específicos

1) Diseñar el acople de interfaces hápticas en el proceso de Teleoperación.

2) Modelar el sistema de Teleoperación con realimentación de fuerza.

3) Desarrollar el control del sistema de Teleoperación con realimentación de fuerza.

CAPÍTULO 3. MARCO TEÓRICO

3.1. Interfaces Hápticas

La Háptica, como ciencia, ha tenido un desarrollo muy importante en épocas recientes. La tecnología háptica permite al ser humano interactuar con sistemas electrónicos por medio de las sensaciones táctiles como la fuerza, la percepción de textura o la vibración. Las interfaces hápticas permiten al usuario percibir, por medio de su percepción táctil, estímulos tales como la realimentación de fuerza y sensibilidad de movimiento y textura.

El desarrollo de interfaces hápticas se da en dos sentidos. Es posible el acople con entornos completamente virtuales, haciéndolos más interactivos con el usuario gracias a los estímulos continuos programados. Por otro lado, también se puede interactuar con entornos reales, e implementar la tecnología para el control de distintas actividades.

La Teleoperación es una poderosa herramienta, facilita la realización de actividades que por su naturaleza resultan peligrosas o incluso imposibles de llevar a cabo por el ser humano de manera presencial. El desarrollo de técnicas para operaciones no presenciales facilita mejoras en tiempo de ejecución, calidad de los resultados y mejor aprovechamiento de los equipos y materiales.

La tecnología háptica y la Teleoperación unidas tienen un gran potencial. Más específicamente, las interfaces hápticas permiten al operador tener una noción acerca de lo que está sucediendo en tiempo real al dictar una orden al mecanismo que es manipulado (sin importar el medio que se utilice para hacerlo). Esto va más allá y en complemento de lo que se percibe de forma visual o auditiva.

Los estímulos reproducidos van desde resistencia al movimiento (fuerza), proximidad, texturas, cambios de presión y temperatura. Para que la sensación de realismo quede manifiesta en las funciones del sistema, el intercambio de información debe ser continuo y bidireccional.

En el sentido hombre-máquina, las órdenes del operador deben ser reconocidas por el sistema de control a una frecuencia de muestreo determinada. Esta frecuencia debe ser establecida de forma tal que la reacción del sistema ante las instrucciones generadas se presente sin cortar la percepción de realidad para el usuario. Desde el actuador, en el sentido máquina-hombre, los sensores de posición y fuerza transmiten al control información y el muestreo de estos datos a una frecuencia adecuada es vital para que las sensaciones

generadas por la interfaz háptica correspondan a lo esperado y haya un nivel de similitud con la realidad. El operador debe experimentar las sensaciones que a una escala proporcional experimentaría al estar manipulando los objetos o materiales en un medio y características que le resulten conocidas y tolerables. Se ha llegado a descubrir que tal y como para encontrar realista una imagen se requieren de al menos 30 cuadros por segundo, en sensaciones hápticas son necesarias señales de estímulo de al menos 1kHz. [7]

La interacción no necesariamente tiene que ser exacta; las aplicaciones para esta tecnología muchas veces se dan en condiciones que no resultan asimilables para las sensaciones a las que el ser humano puede estar familiarizado. Un ejemplo sería en la manipulación de células en el campo de la genética. Por medio de las interfaces hápticas, es posible la programación de una adecuada interpretación de estas condiciones y el entorno donde se realiza la operación. Así, estos aspectos no sólo pueden ser reproducidos a una escala natural a través de los dispositivos hápticos disponibles, sino que también serán fácilmente reconocidos por el operador. Así se logra el fin último de la Teleoperación asistida por las tecnologías hápticas.

3.2. Interfaz háptica Novint Falcon

En la Figura 4 se introduce la interfaz háptica Novint Falcon empleada en el Proyecto.

Figura 4. Interfaz háptica Falcon de la casa fabricante Novint.

El dispositivo Novint Falcon es capaz de retribuir sensaciones vibrotáctiles y de fuerza que vienen a fortalecer la interacción entre el entorno virtual y el jugador. Según las especificaciones, la interfaz háptica en cuestión permite enviar órdenes para el movimiento

en tres dimensiones. El rango en los ejes tridimensionales para el desplazamiento del mando es de 4”×4”×4”, puede reproducir poco más de 2lb de fuerza y consume alrededor de 30W. [11]

A su vez, este dispositivo puede ser programado en sus diversas funciones a través de una capa de abstracción con funciones ya predispuestas para ser configuradas a gusto según las necesidades. Sin embargo, la concepción de uso para el Falcon está en la creación de entornos virtuales como juegos de video o programas de simulación. Por ello, este aspecto debe ser tomado en cuenta a la hora de pensar en incorporar la interfaz a un entorno real.

3.2.1. HDAL (Haptic Device Abstraction Layer)

La capa de abstracción HDAL ha sido diseñada para proporcionar un estándar en la programación del Falcon y otros dispositivos afines. Permite que el programador deje de preocuparse acerca del cómo el dispositivo es directamente inicializado, configurado, y mapeado, así como de las funciones de flujo de datos a bajo nivel. Así por ejemplo, no se debe entrar a programar la manera en que son interpretadas las condiciones en las que el dispositivo pasa a reproducir las sensaciones de fuerza directamente a nivel de hardware. Sí se debe pensar en un modo fluido y consistente para proveer a HDAL de la información que se genera a partir de la interacción con el entorno como variables de un lenguaje de programación de alto nivel. [4]

La estructura de esta capa de abstracción es mostrada en la Figura 5.

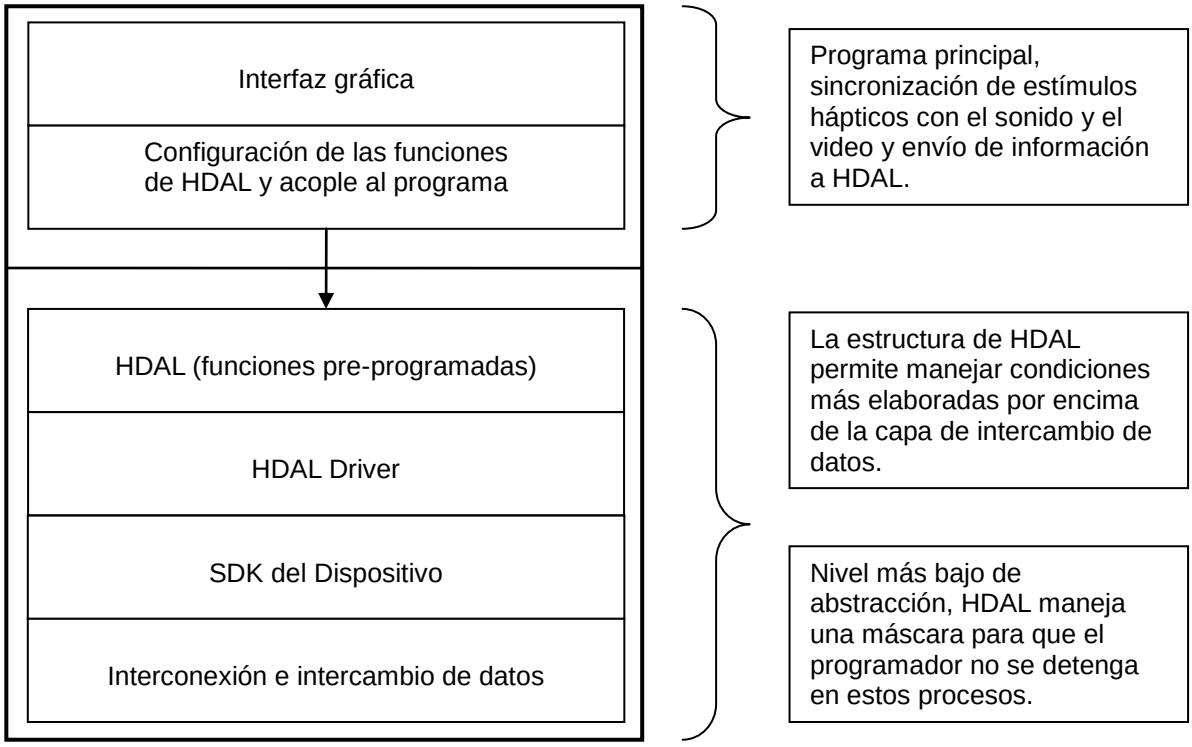

Figura 5. Estructura de un programa integrando HDAL. Tomado de [8]

Según se observa en la jerarquía de capas de abstracción mostrada en la Figura 5, por encima de todo está el programa principal, propio de cada aplicación. En él se tiene principalmente una interfaz gráfica (lo que observa el usuario), y en el caso del trabajo con interfaces hápticas además una etapa de configuración y programación de envío y recepción de información a partir de la interacción con el dispositivo. Esto estaría directamente relacionado con la capa de abstracción HDAL, la cual provee de las funciones, conexión de variables, hilos de ejecución y demás aspectos necesarios para la creación de un entorno con resultados satisfactorios. Por debajo de todo, esta la capa donde se tiene configurado el acople y la comunicación PC-Dispositivo (USB en este caso); este es el nivel de *driver*. Luego de instalar el controlador, este cumple su función gobernado en su ejecución por HDAL y no directamente por el programador/usuario.

La compañía Novint proporciona esta herramienta por medio de un SDK, el cual se instala en el equipo donde se hace el desarrollo de la aplicación. Se requiere configurar el programa de desarrollo con los parámetros para que pueda encontrar las librerías respectivas. En la documentación de HDAL se indica cómo configurar MS Visual C++, aunque igualmente se puede encontrar el medio de hacerlo con otro tipo de herramienta que disponga de un compilador de C++.

3.3. Automatización a nivel industrial con PLCs

Los autómatas programables, o mejor llamados PLCs, son dispositivos de control industrial. Básicamente, tienen la capacidad de leer un cierto número de entradas, que pueden ser tanto analógicas como digitales, a partir de sus valores tomar decisiones acordes a la programación que se le haya cargado y con ello establecer el valor en las salidas. Por lo general las entradas corresponden a señales provenientes de sensores de todo tipo; las salidas son señales que alimentan a los actuadores controlados por un PLC. El componente fundamental de este tipo de dispositivos es la Unidad Central de Procesamiento (CPU por las siglas en inglés de *Central Processing Unit*).

Ya que pueden ser numerosas, tanto las entradas como las salidas se agrupan en módulos. Un módulo de entrada recoge información contenida en señales eléctricas para transformarla en señales lógicas. El módulo de salida recibe este tipo de información desde la CPU y la cambia en parámetros analógicos o digitales reconocidos por los dispositivos dependientes del control programado en el PLC.

Este tipo de control funcional vino a sustituir al antiguo método de conmutación de interruptores, contactores o relés (lógica cableada). Toda la lógica de control se generaba con el producto de una red de contactos, diseñada de acuerdo con las características deseadas. Un error de diseño o de instalación del sistema, generaba que el proceso de corrección fuera bastante complejo, ya que se tenía que deshacer la configuración. Añadir nuevas facultades al sistema igualmente implicaba desentrañar por completo el cableado, incluso aquellos módulos no alterados, dada la naturaleza del arreglo.

A través de las novedosas técnicas de control industrial moderno, es posible administrar los muy elaborados procesos de una planta de producción, por ejemplo. Estos procesos tienen un alto grado de complejidad y requieren por lo general de mucha precisión. Por ejemplo el llenado de botellas con un producto líquido, o cortar materiales sumamente costosos sin errores, a partir de la información recibida a través de sensores.

Seguidamente, se tratará en específico el tema de los PLC Siemens, delimitado por los aspectos relevantes para el Proyecto.

3.3.1. PLC S7 de Siemens

Los Autómatas Programables de la familia de S7 de Siemens han sido creados para el diseño modular de sistemas de control industrial. Tienen la característica de ser escalables y ampliables, esto es, permiten la interconexión de módulos adicionales de entrada/salida, así como la conexión con más PLCs. El S7-300, disponible junto con la Unidad MPS Distribución, está constituido por un CPU 313C con la fuente de poder y los módulos de entrada/salida consisten en bloques independientes, no integrados en el PLC pero acoplables a este.

La programación del control se realiza a través de lenguajes de programación específicos. En el caso del S7-300 se utiliza la herramienta de software STEP7 proveído por Siemens. A partir de este programa se disponen de numerosas funciones para interactuar con el Autómata.

La imagen de la Figura 6 muestra un PLC S7-300.

Figura 6. PLC Siemens S7-300.

3.3.2. Siemens SIMATIC STEP7

La herramienta STEP7 de Siemens es un entorno de desarrollo para los Autómatas de la serie S7. Permite configurar los módulos de Hardware, escribir programaciones para estos así como descargarlos para su puesta en marcha, configurar una red PROFIBUS o MPI, entre otras muchas cosas. STEP7 maneja cada programación de control como un proyecto, donde se administran diferentes bloques o módulos específicos que en conjunto pasan a formar el programa a ser ejecutado por el PLC.

En cuanto a los módulos en los cuales puede estar estructurada una programación en el entorno STEP7, básicamente existen los que se mencionan a continuación.

OB1: Es el bloque principal, donde se establecen los saltos a otros módulos. Su ejecución se da de forma cíclica.

FC (Módulos de Código): Se utilizan para programar funciones, que pueden ser llamadas por otros módulos y así tener mejor estructurado el código. Si un conjunto de instrucciones para realizar determinada tarea debe ser ejecutado de manera reiterada, es conveniente estructurarlo dentro de un bloque FC e invocarlo según sea conveniente.

FB (Módulos de Función): Al igual que un bloque FC, estos se utilizan para estructurar parte del programa. La diferencia es que los FB manejan memoria y pueden almacenar variables. Este tipo de bloques son invocados con unos parámetros de instancia, que en realidad representan valores no volátiles. Normalmente están asociados a uno o más Módulos de Datos de Instancia.

DB (Módulos de Datos): Representan secciones de la memoria para las variables del programa de usuario. Existen dos tipos: los Módulos de Datos Globales, que son accesibles por cualquier otro Módulo de Función en el programa; y los Módulos de Datos de Instancia, que están asociados a un Módulo de Función y actúan como parámetros de instancia cuando este es invocado. Una analogía para este último caso sería pensar en un puerto de entrada/salida para un dispositivo electrónico: el puerto sería el DB y el dispositivo el FB.

Existen además varios tipos de datos en el entorno STEP7. A continuación se mencionan los más comunes.

E (Entradas): Corresponden a las variables de entrada para la programación. Físicamente serían las conexiones a sensores, pulsadores y demás. Se pueden representar en forma de bit (E), Byte (EB), Palabra (EW) y Palabra Doble (ED).

A (Salidas): Son las señales que gobiernan el funcionamiento del equipo industrial a controlar. Igualmente se pueden representar a manera de bit (A), Byte (AB), Palabra (AW) y Palabra Doble (AD).

En el caso de Entradas y Salidas, como máximo un PLC podrá direccionar hasta 65536B. Existen de tipo Digital y Analógico.

M (Marcas de Memoria): Son localidades de memoria que pueden ser utilizadas para almacenar un resultado de forma temporal. A través de STEP7 se manejan 256B de memoria dedicada para este tipo de variables. También se disponen en marcas de un bit (M), Byte (MB), Palabra (MW) y Palabra Doble (MD).

Cada uno de estos tipos se puede direccionar específicamente, según sea requerido, como un bit, Byte, Palabra o Doble Palabra. La Tabla 1 muestra cómo hacerlo para cada caso.

Tabla 1. Direccionamiento de las variables de programa en STEP7.

Longitud	Descripción	Comentario
Bit	E 0.1	Bit 1 del Byte de Entrada 0
Byte	AB 4	Byte 4 de Salida
Palabra	MW 2	Palabra de Marcas 2
Palabra Doble	ED 1	Palabra Doble de Entrada 1

La programación de las funciones del control integrado en el PLC se realiza por medio de un lenguaje de programación especializado. Por medio del STEP7 se pueden emplear tres de ellos: KOP, AWL y FUP. Aquí se describen únicamente los dos primeros,

KOP

Este es un lenguaje gráfico que simboliza un Esquema de Contactos, conocido también como Diagrama de Escalera o *Ladder Diagram*. Se puede tener interconexión entre diversos contactos en condición de normalmente abiertos o normalmente cerrados, luego de recibir las señales provenientes de las entradas para asimilar y construir la información de la salida. El arreglo descrito entre la entrada y la salida comprende una función lógica, simple o compleja.

En la Figura 7 se presenta la descripción para una función lógica AND dispuesta por la conexión entre dos contactos normalmente abierto el primero (*E0.0*), y normalmente cerrado el segundo (*E0.1*). Estos representan variables en el sistema de control. La entrada es una señal de 24V y la salida está reflejada en la señal *A4.0*. De esta manera se tiene que sólo cuando el primer interruptor está cerrado y el segundo está abierto, la señal de 24V alimenta la salida.

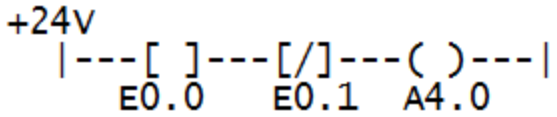

Figura 7. Diagrama básico de contactos que implementa una función AND.

En la programación KOP, se debe estructurar un arreglo de contactos para implementar cada una de las circunstancias que lleven a reproducir una determinada salida, contemplando los variados tipos de las mismas que se pueden generar.

AWL

Este lenguaje de programación no es gráfico. Es conocido también como STL (por las siglas en inglés de *Statement List*) o Lista de Instrucciones. Internamente, STEP7 traduce cualquiera de los otros dos lenguajes gráficos a AWL; este es el lenguaje empleado en el Proyecto ya que presenta una mayor capacidad. Cualquier programación hecha con KOP o FUP puede traducirse a AWL, pero en el caso contrario no todo el tiempo es posible. Existen funciones complejas que no pueden ser representadas gráficamente.

Existen algunas reglas básicas para emplear AWL. En la Tabla 2 se muestra la descripción de algunas funciones lógicas y demás de uso común.

Tabla 2. Instrucciones AWL básicas más comunes.

Operación	Descripción	Comentario
AND Lógica	U E0.2	AND entre el RLO y la señal E0.2
OR Lógica	O E0.3	OR entre el RLO y la señal E0.3
XOR Lógica	X E0.4	XOR entre el RLO y la señal E0.4
Set	S A4.2	Si RLO=1, A4.2=1
Reset	R A4.3	Si RLO=1, A4.3=0
Cargar	L 2	Guarda el valor 2 en el ACU1
Transferir	T MW10	Transfiere ACU1 a MW10
Suma	+I	ACU1+ACU2
Resta	-I	ACU2-ACU1
Multiplicación	*I	ACU1*ACU2
División	/I	ACU2/ACU1
Asignación	= A4.1	Asignar resultado a la señal A4.1

A continuación se mencionan algunos aspectos importantes a considerar cuando se programa con este lenguaje.

Tratamiento de los resultados: Dadas las capacidades de manipulación de datos propias de las arquitectura del CPU que constituye el PLC, el resultado de una sentencia se almacena temporalmente de dos maneras según el caso. Si se trata de una operación lógica a nivel de bit, se utiliza el RLO o bit de Resultado Lógico. Si se trata de una operación a nivel de Byte, Palabra o Palabra Doble, son empleados los registros ACU1 y ACU2. En todos los casos, estos datos son transparentes para el programador y están disponibles para ser utilizados inmediatamente luego de ser obtenidos. Cada instrucción del programa puede o no alterar sus valores de manera directa o indirecta.

Primera consulta: Cuando se comienza a ejecutar un bloque de código, el bit RLO está en un valor binario de 0, valor que no es asignado por alguna instrucción anterior. Esto es, en la primer sentencia del programa en general, o bien después de cada asignación. El estado

de primera consulta provoca que sin importar cuál sea la operación lógica inmediata, su operando será directamente introducido en el RLO ya que no existe un operando válido anterior. La Figura 8 presenta el ejemplo de una función lógica AND, tal y como se observó anteriormente para el lenguaje gráfico.

```
U    E0.0
UN   E0.1
=    A4.0
```

Figura 8. Lista básica de instrucciones que implementa una función AND.

Igualmente como en el caso del uso del lenguaje KOP, acá se tiene una asignación de una salida (*A4.0*) la cual, únicamente estará activa en el momento que esté en alto la entrada *E0.0* (introducida como primera consulta), así como *E0.1* en nivel bajo. Este tipo de asignación puede verse como una conexión combinacional: el resultado continuamente se actualiza y depende sólo del cambio en las entradas.

Conexión MPI

La conexión MPI (por las siglas en inglés de *Multi Point Interface*) se puede entender como una subred multipunto pequeña entre un número reducido de estaciones o dispositivos. Puede utilizarse como una conexión punto a punto para intercambiar datos con una PC, ya sea para la programación del PLC o bien para monitorear las señales involucradas en el sistema de control.

Esta conexión permite direccionar de 0 a 126 elementos, donde la direcciones 0 y 1 están reservadas para las estaciones de programación y monitorización (PCs). Habitualmente la dirección 2 se utiliza para conectar el primer CPU. En la familia S7-300 están configuradas de manera predeterminada las direcciones 0-32 y la velocidad de transmisión de datos es de 187,5kbps.

Este tipo de red permite una interconexión rápida y versátil entre sus nodos, en este caso PLCs y estaciones de monitorización, programación y control. Facilita la configuración de los CPUs e incluso su puesta en marcha junto con los módulos de entrada-salida, actuadores y señales de referencia. Además permite disponer de un grado de escalabilidad limitado pero aceptable, eficiente en sistemas no muy complejos.

A continuación se comenta acerca de los PLCs de la compañía FESTO. Se introduce información acerca de su metodología para la programación, y de las opciones de las que se dispone para el desarrollo de un sistema basado en su implementación en general.

3.3.3. PLC FESTO FEC Standard

Los PLC Festo FEC Standard son controladores de propósito general. Se han creado para romper con la enorme brecha que existe entre los PLC de uso doméstico, académico y en general, de uso un poco alejado al ámbito industrial, con los que son habitualmente buscados para construir las complejas estructuras de automatización que requieren las grandes fábricas y plantas de proceso.

El autómata programable FESTO FEC Standard es alimentado con una tensión CD de +24V. Tiene dispuestas cuatro entradas de un byte, y dos puertos con la misma longitud para salidas. Ambos tipos de señales pueden ser tanto analógicas como digitales. Para las conexiones se utiliza una tecnología llamada SAC (*Sensor-Actuator Connector System*), la cual permite la interconexión común de señales de una manera más discreta, preservando el uso más eficiente del espacio.

Para su programación, el controlador tiene disponible el acceso a una conexión RS-232 de fábrica preinstalada en el equipo. Esto le permite entablar el intercambio de datos con una PC, principalmente para su programación. Luego igualmente puede seguir con la comunicación por el puerto serie para la monitorización de información, actualización de los módulos programados y cargados por el usuario, y el intercambio de datos durante la ejecución de los procesos.

Una de las principales ventajas de este compacto pero versátil PLC, es el hecho de que permite además la conexión a una red Ethernet, mediante un *driver* disponible para ser cargado junto con la programación. Luego de la configuración de este módulo, es posible tener el dispositivo integrado en la red, acceder a la información de estado de las variables de entrada y salida, alterar la programación por esta vía de forma remota (de acuerdo con la estructura de red en la cual se trabaje), así como también crear un canal de datos para el intercambio de información entre aplicaciones.

En la Figura 9 se ofrece una imagen del PLC FESTO FEC Standard, visto desde el frente.

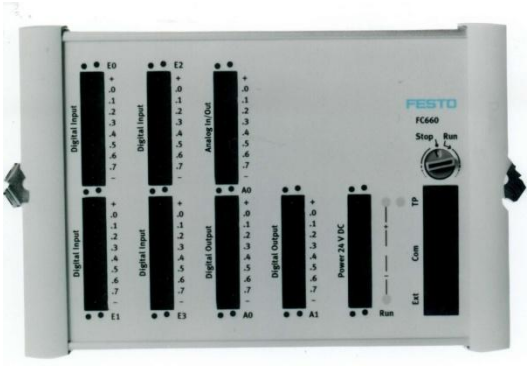

Figura 9. Vista frontal del PLC Festo FEC Standard.

3.3.4. FESTO FST

El programa FESTO FST proporciona una plataforma de administración para sistemas de automatización desarrollados a partir de un PLC FESTO en general. Partiendo desde la organización de los recursos para la programación y concepción de las rutinas posteriormente empleadas por el controlador, pasando por la creación y mantenimiento de un modo de comunicación entre la PC y el autómata, terminando en una estructura de análisis y monitorización de información; todo es fácilmente configurado, programado y puesto en marcha gracias a este paquete de software.

La Figura 10 muestra la pantalla principal del programa. Se observan las barras de herramientas, así como dos ventanas importantes a la hora de tener un proyecto abierto (caso de la imagen): un esquema de recursos tales como archivos de código, módulos disponibles y herramientas en general (ventana más pequeña en la parte superior-izquierda), y también un grupo de ventanas donde está superpuesta la lista de ubicación para las entradas-salidas del PLC.

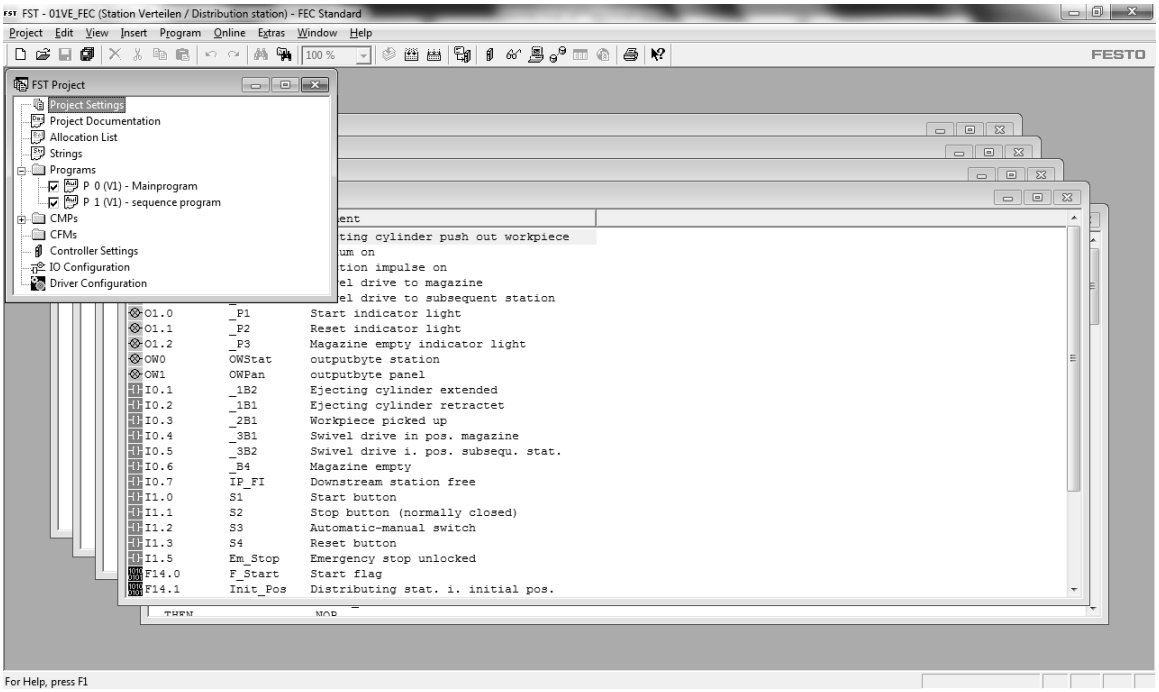

Figura 10. Ventana principal del programa FST con un proyecto abierto.

La manera de trabajar con FST es bastante intuitiva, sencilla y a la vez funcional. Adicionalmente al programa principal, existen otros programas que pueden ser ejecutados en conjunto y que brindan un mejor soporte y una mayor cobertura para las necesidades del trabajo que se esté realizando. Así por ejemplo, junto con FST se puede tener un servidor de datos llamado *IPC Data Server*, el cual actúa como un servidor para la comunicación DDE y el acceso a datos en general para diversas aplicaciones de monitorización y control. Todo esto se puede hacer tanto desde un enlace RS-232 así como por medio de la estructura de red IP.

Para programar en FST, se debe saber que el primer programa, el P0, es el programa principal y siempre es ejecutado de forma cíclica. Toda instrucción contenida en este será continuamente llevada a cabo. A partir de este código se puede invocar igualmente otros programas del mismo proyecto acorde con las condiciones preestablecidas y que responden a los requerimientos.

Igualmente, se pueden incluir distintos drivers en el programa, para ser descargados al PLC y complementar sus funciones. Por ejemplo puede cargarse el módulo de conexión Ethernet, o uno para configurar un control PID.

Adicionalmente, es deseable especificar una tabla de ubicación para las entradas-salidas (*Allocation List*). Un ejemplo de esta tabla se muestra en la Figura 11.

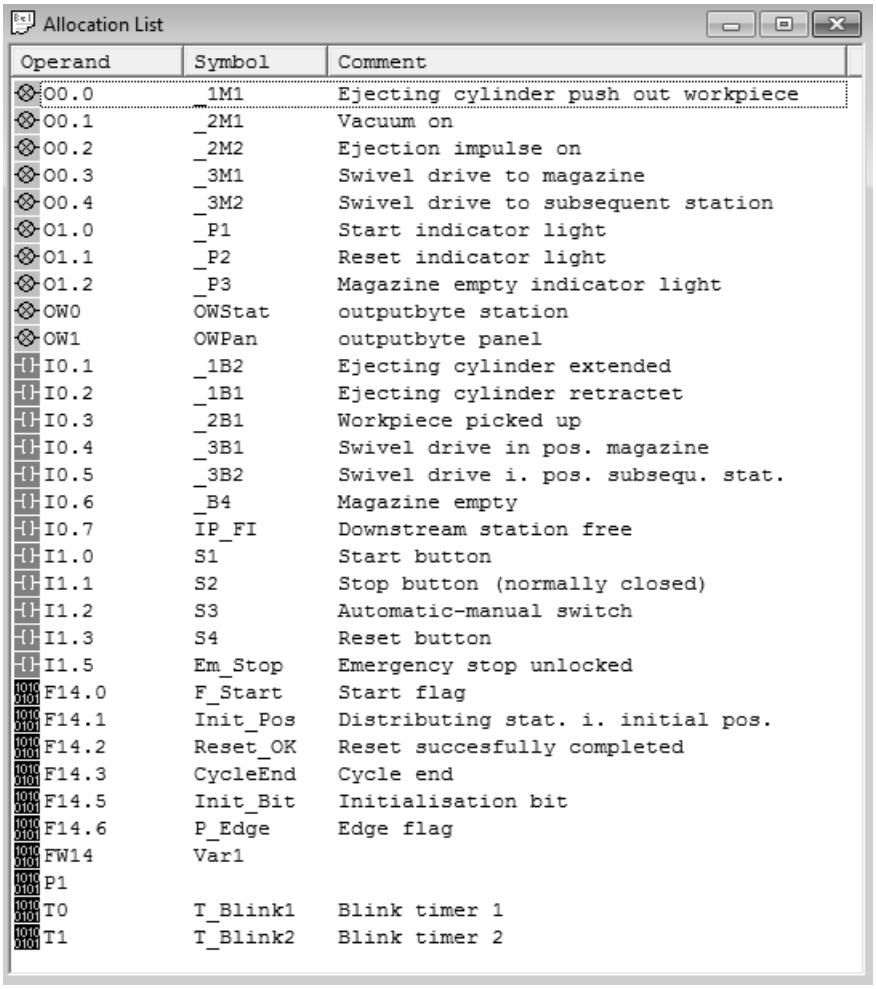

Figura 11. Ejemplo de una tabla de ubicación para las señales de entrada-salida.

Tal y como se aprecia en la tabla mostrada en la Figura 11, para cada señal se especifica el tipo (entrada o salida), la dirección absoluta o técnica, un nombre para la variable (indicador que puede ser utilizado en el programa de instancia) y un comentario que facilite un rápido entendimiento acerca de lo que significa la variable en cuestión. Una breve descripción de cómo se ubican las señales más comúnmente usadas, se muestra en la Tabla 3.

Tabla 3. Tipos de variables y ejemplos de direccionamiento absoluto en FST.

	Direccionamiento	
Tipo	**Bit**	**Palabra**
Entrada	I0.1	IW0
Salida	O4.3	OW4
Marca de memoria	F2.0	FW2

De esta manera, se facilita la programación al no tener que estar recordando la dirección absoluta para la variable. Una vez que se ha introducido un pseudónimo para la señal en la

tabla, éste puede ser empleado para indicar que se trata de la variable específica. El código en FST está estructurado de forma distinta a como lo está en Siemens STEP7. Aunque se dispone de un lenguaje en STL (*Statement List*), las instrucciones y la organización es bastante diferente. A continuación, la Tabla 4 ilustra por medio de un ejemplo las instrucciones lógicas combinacionales más comunes que se utilizan en STL para programar en FST.

Tabla 4. Ejemplo de construcción de un programa STL en FST.

Instrucción	Tipo	Ejemplo	Comentario
If	Condicional	`IF Reset_OK`	Se pregunta por el estado alto de la señal *Reset_OK*
And	Operador lógico	`AND No_Set`	Se revisa además el estado alto de *No_Set*
Or	Operador lógico	`OR ( Turn_Off`	Operación paralela a la anterior
N	Operador lógico	`AND N No_Set )`	Se pregunta por el estado bajo de *No_Set*
Then	Condicional	`THEN`	Se hace la asignación
Set	Asignación	`SET Off_Proc`	Se asigna a *Off_Proc* el valor '1'
Othrw	Condicional	`OTHRW`	Condición de asignación en cualquier otro caso
Reset	Asignación	`RESET Off_Proc`	Si no se cumple la condición, se hace reset de *Off_Proc*

Este tipo de programación, como se observa, es intuitivo y facilita la concreción de ideas en sentencias de código. No se requiere de una mayor explicación para comprender lo que hacen las líneas mostradas en la Tabla 4. Aspectos como este, al final influyeron en la decisión para determinar cuál herramienta utilizar.

3.3.5. Siemens WinCC

El programa WinCC, incluído en los paquetes de desarrollo de Siemens, permite crear interfaces Humano-Máquina (HMI, por las siglas en inglés de *Human-Machine Interface*). A través del mismo se puede acceder a diversos servidores de datos, por ejemplo información proveniente de STEP7 (variables directamente leídas desde el PLC), un servidor DDE o un canal de datos OPC. La idea es poder visualizar los procesos que se desarrollan en un ambiente de Automatización Industrial, e incluso lograr un control remoto a través del programa.

El aspecto quizá más importante, es que se dispone a través de WinCC de un medio para crear objetos gráficos que representan los equipos en monitorización. Además permite la creación de un servidor Web con el cual, se puede visualizar remotamente la actividad de interés.

3.4. Intercambio de datos entre aplicaciones

En el mundo de la Automatización Industrial, algunas de las características deseadas más importantes están en tener la capacidad de monitorear los procesos, generación automática de informes acerca del estado de los mecanismos, control de emergencia de eventos no deseados o imprevistos e incluso la facilidad para poder cambiar alguna característica de control sin tener que desmontar todo el sistema o pararlo por completo. Todo esto en conjunto permite un dinamismo mucho mayor y un ahorro económico considerable.

Para esto, a la par de las herramientas para la confección y puesta en marcha de los sistemas complejos de control de una planta industrial de manufactura, por ejemplo, también se disponen de recursos que permiten estructurar diversos canales de información cada uno con un propósito distinto. Todo esto conlleva a incorporar al proceso global las características antes mencionadas y facilitar así las labores de ingeniería sin la necesidad de tener que permanecer de forma presencial en el sitio de operación.

A continuación se ofrece una introducción a dos de los medios por los que se puede lograr este cometido. Primero se trata el estándar OPC, moderno, robusto, versátil y ampliamente difundido; luego se habla de DDE, herramienta sencilla pero funcional, aunque en los últimos tiempos haya perdido popularidad.

3.4.1. OPC: OLE For Process Control

El Protocolo OPC permite el intercambio de datos entre aplicaciones. Además, es posible a través de OPC crear estructuras complejas de monitorización y control remoto de equipos y sesiones de usuario entre otras muchas cosas.

Es un estándar mantenido por la *OPC Foundation*, y es una extensión de la tecnología OLE COM desarrollada por Microsoft como sucesor de DDE. OPC es empleado en mayor medida en el área de la Automatización Industrial.

Gracias a esta plataforma, es posible interrelacionar todo tipo de equipos, desde dispositivos de poca capacidad con software embebido, pasando por PLCs y estaciones de monitorización, programación y control de procesos, hasta llegar a los grandes servidores de una industria de la información.

3.4.2. DDE: Dynamic Data Exchange

El protocolo de intercambio dinámico de datos, conocido como DDE (*Dynamic Data Exchange*), es una herramienta disponible en el sistema operativo MS Windows. Permite básicamente crear un canal por medio del cual, dos aplicaciones (por lo general un cliente y un servidor) pueden compartir información y mutuamente enviarse comandos o acciones por ejecutar.

En este tipo de estructuras, el servidor es quien tiene el mayor peso de trabajo en las transacciones, ya que es quien ofrece una plataforma de comandos y un estándar para el nombre de las variables, así como los medios para organizar la conversación. El cliente por su parte, accede a la plataforma disponible a través del servidor, y utiliza este recurso para ejecutar acciones. No es un estándar la manera en que debe estar configurado el servidor, así que para que un cliente pueda tener éxito al querer entablar una conversación, debe de antemano conocer con detalle qué instrucciones o eventos están disponibles, así como también el medio por el que se puede tener acceso a la estructura específica del interlocutor.

Para la programación de un enlace DDE, se dispone DDEML (*DDE Management Library*). Esta librería provee de herramientas mejoradas y más robustas para la creación de clientes y servidores DDE. Proporciona una interfaz que simplifica el cómo añadir las funcionalidades de DDE a una aplicación específica, apenas en desarrollo. En lugar de enviar, publicar y procesar mensajes DDE directamente, las funciones proporcionadas por DDEML son utilizadas para administrar las conversaciones generadas por la interacción entre el cliente y el servidor de datos. La librería también proporciona un medio para la gestión de hilos de ejecución en paralelo con el programa principal, así como una manera más eficiente bajo la cual los programas pueden compartir información. [11]

Para codificar un canal de este tipo, básicamente es necesario siempre conocer cómo está definida la nomenclatura en el servidor, para acceder a los datos. Por lo general, la aplicación cliente debe especificar el nombre del servidor, el tema de las futuras transferencias y la dirección del dato concreto al que quiere tener acceso de acuerdo con la estructura de datos propia de la aplicación servidor. Por ejemplo, si el servidor reside en el programa IPC Data Server de FESTO, como es el caso del Proyecto, aquí el nombre del servidor es “*IPC_DATA”*, y este parámetro debe ser enviado como una cadena de caracteres empleando las funciones de configuración predispuestas en DDEML. Posteriormente, el tema de conversación sería en este caso *“IPC_1”*, dado que este es el nombre del PLC

administrado por FST al que se quiere enlazar. Posteriormente, el dato se especifica igualmente por una cadena de caracteres, para cada uno de los que sea necesario leer desde el origen de la información. Un ejemplo sería para la palabra de memoria 5 del PLC: "*MW5"*. En el Capítulo de Descripción Detallada de la Solución, se ofrecen ejemplos de cómo puede ser programado un enlace DDE, empleando DDEML para el caso concreto del desarrollo del Proyecto.

CAPÍTULO 4. PROCEDIMIENTO METODOLÓGICO

En esta sección se describe la forma en que fue enfrentado el Proyecto desde sus inicios, en lo que respecta a la manera de proceder para obtener la información pertinente, delimitar los alcances del mismo y ejecutar las acciones que llevaran a una satisfactoria culminación.

4.1. Definición de la problemática asociada al Proyecto

Primeramente, había que determinar lo realmente imprescindible para satisfacer las necesidades que motivaron la formulación del Proyecto. Sin importar qué circunstancias pudieran cambiar con el transcurso del tiempo destinado para finalizar el trabajo, era imperante terminar con un sistema a nivel de diseño e implementación, que ofreciera una solución real a lo que inicialmente fue solicitado.

Así se llegó a la conclusión que la problemática que motivó la concepción del Proyecto se basa en la pregunta ya antes introducida pero que se expone nuevamente:

¿Es posible lograr el acople funcional de la interfaz háptica Novint Falcon a un proceso Teleoperado de Automatización?

A partir de esto, y con la meta de dar una respuesta positiva a la pregunta, se realizó el trabajo.

4.2. Definición de la Metodología para el desarrollo del Proyecto

En este Proyecto se definieron tres consignas a cumplir durante su realización. Las mismas se enlistan a continuación:

1. Utilizar en la medida de lo posible el equipo y los recursos en general disponibles en el LIRA.
2. Investigar acerca de cada uno de los temas relacionados con el Proyecto, no sólo pensando en su desarrollo final sino también como una forma de obtener conocimientos de manera integral.

3. Trabajar siempre sobre el modelo o diseño más simple que funcione y lleve a cumplir satisfactoriamente los objetivos planteados.

El primer aspecto fue importante para minimizar costos y hacer un mejor aprovechamiento de los recursos disponibles para trabajar desde la fecha de arranque del Proyecto. Así se tenía la oportunidad de obtener avances más eficientemente sin caer en la necesidad de definir tiempos de espera para el trámite de compra adicional de componentes y equipos. Además, esto fomentó que primeramente se tuviera que realizar un análisis de las posibilidades reales acorde con los recursos disponibles: para construir algo funcional sin adquirir equipos y herramientas de software adicionales, era necesaria una verdadera comprensión y una medición de las capacidades concretas de los componentes a los que se tenía acceso en el LIRA. Una lectura a profundidad de manuales, hojas de datos y diversos recursos se volvió útil para cumplir esta consigna

Luego, para complementar con lo anterior, se tuvo siempre la disposición de leer, investigar, buscar y profundizar en cada aspecto relacionado con el Proyecto. Con esto, al final de cada etapa de investigación, se obtuvo una idea clara y concreta que pudiera llevar a culminar el trabajo planteado atendiendo también a los demás lineamientos paralelamente definidos, y favoreciendo en la medida de lo posible un proceso de aprendizaje integral.

Por último, se pensó que el diseño más simple, eficiente y funcional posible que cumpla con los objetivos siempre es mejor que uno más complejo y con mayores requerimientos de recursos, tanto en equipo, herramientas de software y horas de trabajo; todo traducido en costos económicos mayores. Al final, estos lineamientos favorecieron un mejor desempeño para construir la labor de ingeniería desarrollada a través del Proyecto.

4.3. Puesta en marcha y labores realizadas

Para iniciar con la ejecución del Proyecto sólo era necesario disponer de una computadora con acceso a internet: lo primero que se realizó fue una investigación acerca de los conceptos elementales implicados, para tener una base con la cual proceder luego. De antemano se conocía acerca de la disponibilidad de manuales, guías y documentación en general, así que este fue un buen punto de partida.

Con base en la información recopilada, con el paso de los días se creó una selección de temas más densos por estudiar y analizar. Todo ello en función de que al final de esta etapa se pudiera disponer de un criterio acerca de cómo enfocar la solución definitiva. Una vez se

consideró tener ya una idea lo suficientemente madura como para comenzar a ejecutar tareas de medición y programación, entonces se inició con la etapa de diseño y pruebas básicas.

A partir de lo anterior, se logró descartar variantes de la solución que no eran lo bastante convincentes, siempre pensando en observar los resultados y modificar todo aquello que pudiese mejorar la idea específica planteada. Acompañado siempre de un proceso continuo de investigación desarrollado en paralelo, así fue como transcurrió la realización del Proyecto. Al final, una idea madura y versátil desembocó en un producto terminado lo suficientemente satisfactorio para cumplir con los objetivos.

CAPÍTULO 5. DISEÑO DEL SISTEMA DE AUTOMATIZACIÓN CON INTERFACES HÁPTICAS

5.1. Diagrama de bloques general del sistema

El diagrama de bloques presentado en la Figura 12 muestra a nivel general la solución llevada a cabo en el Proyecto.

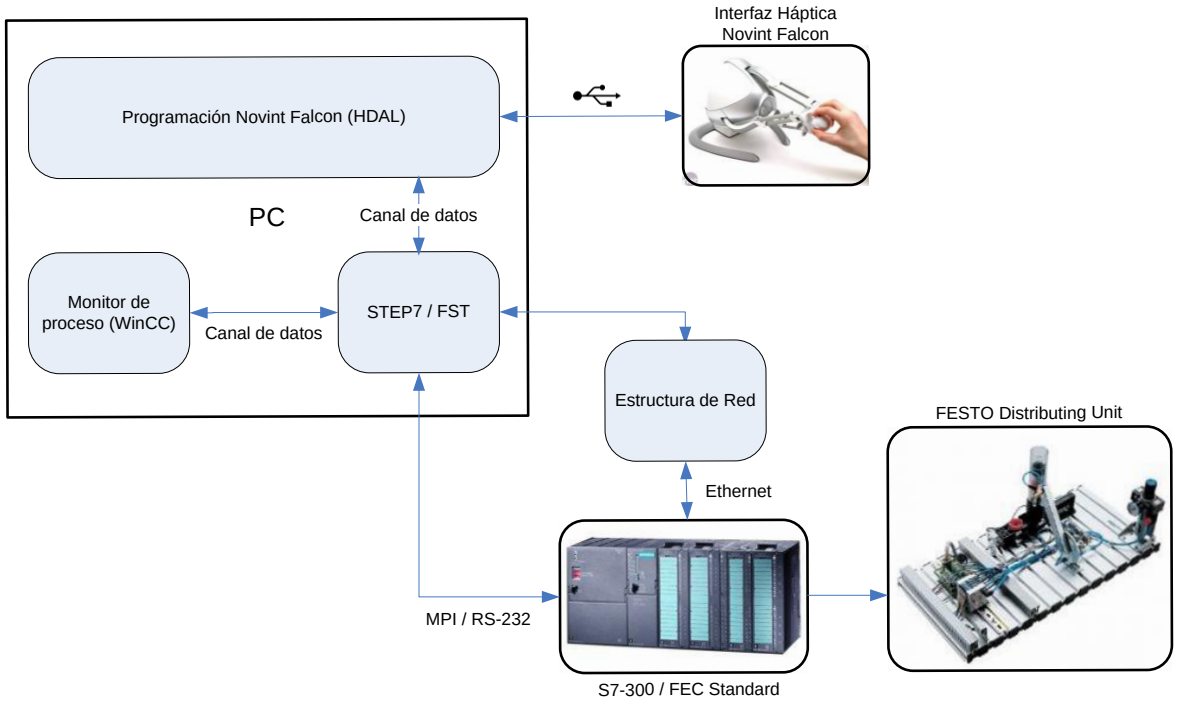

Figura 12. Diagrama de bloques general de la solución.

El sistema de la Figura 12 describe la solución implementada. Como se aprecia en el diagrama de bloques, es necesario un medio para enviar y recibir información desde y hacia la interfaz háptica. El operador interactúa con ella de manera que sus órdenes son transmitidas al sistema, y también puede experimentar realimentación por condiciones peso y esfuerzo, así como por la delimitación inherente al entorno. Las mismas son interpretadas por la interfaz gracias a la programación de las funciones de la capa de abstracción HDAL.

Cabe mencionar que originalmente todo recurso disponible para realizar una programación acertada de las condiciones de escucha y respuesta en las características propias de la interfaz, estaba dado para entornos virtuales como video juegos o programas de simulación. Para el proyecto, resultó necesaria la programación adaptable a un entorno real. Es esta la pieza clave de la solución: que la interfaz háptica Novint Falcon pudiera ser acoplada al sistema de Automatización.

Para ello, es necesario programar una máscara o árbitro de comunicación, que se encarga de interactuar directamente con el código HDAL (*Haptic Device Abstraction Layer*) y extraer la información proveniente de la interfaz, así como de suministrarle los datos extraídos desde el actuador para que pueda ofrecer estímulos hápticos acordes a la manipulación del entorno. Esto se realizó como un submódulo programado junto con el programa principal o *main* en el código generado. Lo que hace es administrar el flujo de información entre la programación de la interfaz y el canal de transmisión de datos compartido con la programación del PLC.

La programación a alto nivel para el *Falcon* se basa en las herramientas disponibles a través del SDK distribuido por el fabricante, hecho para el lenguaje C++. Es posible trabajar sobre HDAL sin preocuparse de explícitamente por aspectos de sincronización o el modo en el que se lleva a cabo el intercambio de datos, así como por la escritura directa de los registros de configuración o la lectura de las palabras de estado del dispositivo. Existen funciones y métodos en alto nivel para intervenir el funcionamiento del Falcon y volverlo útil para cada propósito específico.

Los comandos que se generan por medio de las órdenes del operador, deben ser transmitidos hacia el control del actuador. Este control se implementa por medio de un PLC. En primera instancia el trabajo se llevó a cabo con un Siemens S7-300, posteriormente se terminó programando un PLC FESTO FEC Standard debido a que la solución con este último tenía mucho mejores posibilidades de tener éxito en un menor plazo. Según se estableció con la metodología de trabajo, antes ya expuesta en el documento, lo importante era encontrar la manera más eficiente, versátil y al mismo tiempo robusta que permitiera implementar la solución planteada.

El actuador, que corresponde al Módulo de Cambio (Brazo Giratorio) y de manera extendida a los demás módulos en general de la Unidad de Distribución FESTO, es comandado desde la interfaz háptica a través del control dinámico programado con el PLC. La estación igualmente provee de información acerca de las variables de interés para monitorear, por medio de las características ensambladas en ella para este fin.

5.2. Programación de las funciones de la interfaz háptica

Al hacer el diseño del acople de la interfaz al sistema, se tenía que definir cómo interpretar la posición del mando del Falcon, en tres dimensiones, teniendo un actuador con

movimiento giratorio. Se pensó programar una limitación en los ejes *y* y *z*, para que el movimiento de la palanca de mando de la interfaz háptica sólo tuviera un desplazamiento en el eje *x*, y las órdenes se tomaran de la posición relativa en este. Si este se encontraba desplazado a la derecha, la orden transmitida sería de movimiento en sentido horario del brazo giratorio. En el caso contrario, el movimiento sería el opuesto. Ubicado en el origen, no se emitiría orden alguna y el módulo de cambio permanecería detenido. Estas condiciones se pudieron implementar de forma correcta a través del desarrollo del código, y su explicación se ofrece a continuación.

El aspecto más relevante a la hora de pensar en la manera que se iba a programar la interacción con la interfaz háptica, fue el reproducir fuerzas perceptibles por el operador. Para ello, lo mejor es encontrar un modelo de fuerzas que pueda ser fácilmente programado pero que a la vez produzca una sensación para el usuario lo más fiel posible a la realidad. Además, tuvo que considerarse que la información disponible para hacerlo se basa en un vector de posición del mando del Falcon en las tres dimensiones, y por ello, el modelo debía tener la característica de responder conforme estos valores cambiaban. Así es como se ideó tomar el modelo simple, práctico y versátil de un sistema masa-resorte. Este modelo se basa en el principio fundamental presentado por medio de la ecuación:

$$F = -kd \tag{5.1}$$

Donde *F* es la fuerza de oposición, *k* la constante del resorte y *d* la distancia relativa sobre uno de los ejes. Teniendo información de la posición, se podía tomar el dato negativo y multiplicarlo por un factor preferiblemente variable a conveniencia según los eventos presentados por la interacción del operador con el sistema, todo para obtener una sensación de fuerza reproducible por la interfaz háptica. Así, a mayor distancia del origen, mayor fuerza se experimenta, y el mando tiende siempre a ubicarse en la posición neutral.

El modelo en un formato gráfico, adaptado al Proyecto se muestra en la Figura 13.

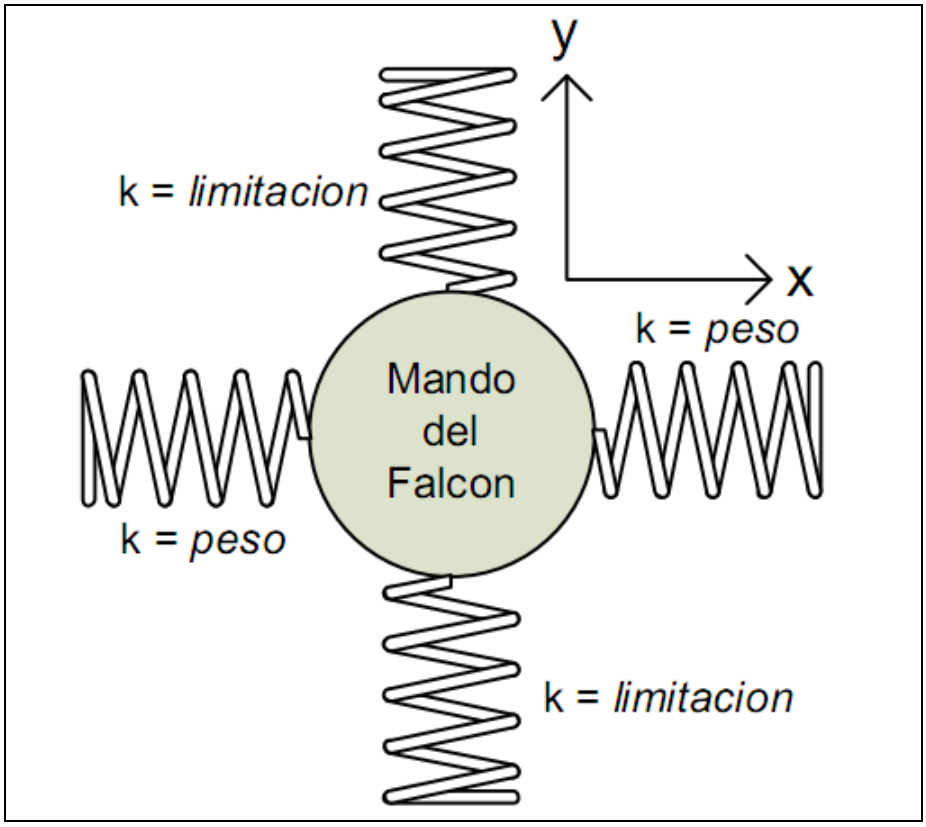

Figura 13. Modelo masa-resorte aplicado para el Falcon.

En el modelo presentado en la Figura 13 se asume que en ese momento el mando se ubica en el origen del sistema de coordenadas. Para facilitar su comprensión, se muestra únicamente el caso para los ejes *x* y *y*, aunque lo mismo aplica para el eje *z*. La idea es que conforme se quiera desplazar el mando a posiciones no neutrales, se experimente una fuerza en oposición que en este caso corresponde a la que sería tener un resorte con una constante dada.

El valor de la constante era distinto para cada caso. Así por ejemplo, dado que se quería restringir completamente el movimiento en el eje vertical (igualmente para el eje de proximidad), esta constante era máxima, de valor 100. Para el eje horizontal, la constante cambiaba de acuerdo con las condiciones dadas según se emitían las órdenes de movimiento al Módulo de Cambio de la Unidad de Distribución y este interactuaba con su entorno. Por ejemplo, el mando se podía desplazar normalmente a lo largo del eje para emitir las órdenes de movimiento y esta constante bajo estas condiciones tenía un valor de 8, lo que quiere decir que la fuerza de oposición era baja (nótese la relación entre este valor y el de limitación total o fuerza máxima aplicada). Pero cuando se levantaba un objeto, y se quería reproducir sensación de peso (mayor dificultad para mover el Brazo Giratorio), la

constante tomaba un valor de 15.

Para la programación de la interfaz háptica, en general, se echó mano de las ventajas que da utilizar los recursos disponibles por medio del SDK de Novint, específicamente el HDAL, ya antes introducido y comentado. Para lograr un funcionamiento satisfactorio del sistema, se requiere primeramente conocer qué estructura seguir, y como relacionar adecuadamente las funciones y variables involucradas. En el presente Proyecto, las etapas para la programación se describen en la Tabla 5.

Tabla 5. Etapas de programación para HDAL.

Etapa	Elemento	Características
1	Definición de clase	• Creación de una estructura de variables y funciones propia • Inclusión de las librerías HDAL en el código • Disposición de variables necesarias para las funciones HDAL
2	Programación de funciones propias de la clase	• Funciones de propósito específico a la aplicación que se está desarrollando • Métodos de soporte para la estructura del programa
3	Creación de funciones de enlace con HDAL	• Estas funciones no pertenecen a la clase pero interactúan con ella, son el enlace con HDAL • A partir de estas funciones serán invocadas las demás y se creará un hilo de ejecución en paralelo • Estas funciones administran a las de soporte antes programadas
4	Estructuración del programa principal	• Instancia de la clase y organización de las funciones ya programadas en la secuencia adecuada • Acople de las funciones del Falcon con los demás bloques del programa (interfaz gráfica, canal de datos, etc.)

La secuencia descrita en la Tabla 5 fue necesaria para estructurar la generación del código. Así por ejemplo, a continuación se ofrece en la misma secuencia, un ejemplo de creación de código funcional, específicamente para el caso de la variable que es utilizada para calcular la magnitud de fuerza y luego reproducirla en la capa de abstracción. Igualmente otras condiciones, como el accionamiento de un botón por ejemplo, siguen un proceso similar al mostrado para este caso específico.

```
//paso 1: incluir librerías
#include <hdl/hdl.h>
//(…)
//paso 2: declaración de la clase
class clase_hdal
{
      //(…)
      //paso 3: definir función friend
      friend HDLServoOpExitCode hdal_loop(void *punt);
      //(…)
      public:
      //(…)
      //paso 4: variables y métodos miembros de clase
      double fuerza_haptica[3];
      double peso;
      double limitacion;
      //(…)
      void obtener_posicion(double posic[3]);
      void calc_fuerza();
      //(…)
};
```

El bloque anterior provee una estructura para ejemplificar la primera etapa comentada en la Tabla 5. El paso 1 muestra la inclusión de la librería que permite utilizar en el código las funciones y tipos de datos HDAL. El paso 2 es la declaración de la clase de propósito específico creada en el programa. El paso 3 muestra, como dentro de la estructura de la clase se define una función *friend* o amiga pero que no pertenece a la misma. Esto es necesario para mantener coherencia con la estructura predispuesta para la capa de abstracción: la función no puede pertenecer a la clase ya que es de tipo *HDLServoOpExitCode*, un tipo de datos propio de HDAL. Con el paso 4 se definen las variables y métodos que componen la estructura del objeto.

En este caso, se muestran sólo lo pertinente para ejemplificar el proceso de la generación de estímulos de fuerza. El vector *fuerza_haptica[3]* contendrá el dato de fuerza a reproducir por el dispositivo, la variable *peso* provee de información acerca de la sensación de peso o dificultad para movilizar el mando de la interfaz en el eje horizontal y *limitacion* la proporción de fuerza que impide mover el mando hacia arriba-abajo y cerca-lejos. El método *obtener_posicion()*, copia el dato de posición instantánea, propio del momento, en la variable que se le pase como parámetro. Por último, la función *calc_fuerza()* es aquella donde se calcula, de acuerdo con las condiciones del entorno, la fuerza háptica que reproduce el Falcon.

Para explicar cómo se desarrolló la etapa 2 mencionada en la Tabla 5, se muestra el siguiente bloque de código.

```
void clase_hdal::calc_fuerza()
{
	//(…)
	//paso 1: definir variable que contiene inf. de posición
	double ubicacion_mando[3];
	//(…)
	//paso 2: leer la posición
	obtener_posicion(ubicacion_mando);
	//(…)
	//paso 3: ubicar valor de posición para el cálculo de fuerzas
	fuerza_haptica[0]=-ubicacion_mando[0];
	fuerza_haptica[1]=-ubicacion_mando[1];
	fuerza_haptica[2]=-ubicacion_mando[2]-1;
	//(…)
	//paso 4: cálculo de fuerzas
	fuerza_haptica[0]=fuerza_haptica[0]*peso;
	fuerza_haptica[1]=fuerza_haptica[1]*limitacion;
	fuerza_haptica[2]=fuerza_haptica[2]*limitacion;
	//(…)
{
```

En el paso 1 se define una variable donde se copiará la posición actual del mando del Falcon en un instante determinado, tal y como se realiza en el paso 2 gracias a la llamada de la función *obtener_posicion()*, que pertenece a la clase. El paso 3 copia en la variable *fuerza_haptica[]* cada uno de los valores de posición leídos pero negativos al querer reproducir fuerzas que se oponen al movimiento, según el modelo masa-resorte antes comentado. Al final, se incluye el otro componente del modelo de fuerza: A cada uno de los componentes del vector se le añade el efecto de la constante. Para el caso horizontal, este valor físico es variable y dependiente de las condiciones de operación. En los otros dos casos, se integra la limitación como un dato constante y máximo.

Para mostrar cómo se trabajó con la etapa 3, se presenta el siguiente código.

```
HDLServoOpExitCode hdal_loop(void* puntero)
{
	//paso 1: instancia de clase
	clase_hdal* clase=static_cast<clase_hdal*>(puntero);
	//(…)
	//paso 2: llamada a función que calcula fuerzas
	clase->calc_fuerza();
	//paso 3: asignación de variable de fuerza
	hdlSetToolForce(clase->fuerza_haptica);
	//(…)
	//paso 4: configuración de ejecución continua
	return HDL_SERVOOP_CONTINUE;
}
```

En este caso, se describe cómo se ve estructurada la función *friend* de la clase, y la secuencia de instrucciones que incluye. En el paso 1 se hace una conversión del puntero que es pasado como parámetro al método, que corresponde a una instancia de la clase creada

previamente. El paso 2 es un llamado a la función que realiza los cálculos y asignación para la fuerza háptica, ya comentado en el bloque de código anterior. Una vez se ha actualizado la variable *fuerza_haptica[]*, se invoca la función de HDAL *hdlSetToolForce()*, pasándole como parámetro ese dato en el paso 3. Lo que realiza el paso 4 es establecer la condición para indicar a HDAL que esta es la configuración para generar el hilo de ejecución propio de la capa de abstracción, y esto es porque lo que se incluye en este método se va a realizar de cíclica y continuamente. Al devolver el valor *HDL_SERVOOP_CONTINUE*, HDAL conocerá esta condición.

La etapa 4 fue desarrollada de manera que se tuviera una ejecución oportuna de las funcionalidades antes programadas. Lo importante fue mantener el orden correspondiente, partiendo de la inicialización y terminando con la puesta en marcha de la programación, iniciando el hilo HDAL que administra las actividades del Falcon.

5.3. Programación del PLC

Para las funciones del controlador lógico programable, fundamental en el Proyecto, se tuvo que analizar cuál tipo de programación era el más adecuado. Existían dos aspectos generales que se debían tener en consideración:

1. Los eventos comandados desde la interfaz háptica debían atenderse de forma continua, para lograr un efecto de realismo en la percepción del operador.
2. A partir de la información de los sensores predispuestos en la Unidad de Distribución, tenía que actualizarse la información que se suministraba a la programación del Falcon, a través del canal DDE.

Seguido de un estudio de la situación, se determinó que la mejor programación del control en el PLC, que cumplía con las tres consignas que figuran descritas en el capítulo 4 concerniente a la metodología del Proyecto, y que además llenaban las expectativas mencionadas antes, era una lógica combinacional con el uso de determinadas banderas o marcas de memoria para captar información. Esta programación, sería capaz de atender cualquier cambio en las condiciones, tanto desde la PC como desde el entorno Teleoperado (la Unidad de Distribución), a la vez de proceder a ejecutar las acciones pertinentes.

Se describe la programación dispuesta para el PLC FESTO FEC Standard, siendo este el

controlador empleado finalmente. En general, el trabajo realizado con él primeramente fue pensado para el PLC S7-300, e igualmente los procesos mostrados pueden ser análogamente implementados en esa familia de controladores.

La Tabla 6 introduce algunas definiciones importantes para describir la manera en que fue programado el PLC FESTO FEC Standard, acoplado a la Unidad FESTO MPS Distribución.

Tabla 6. Palabras clave empleadas en la descripción del programa del PLC.

Clave	Significado
Origen	Entorno donde se origina el valor de la variable
Propósito	Entorno en el que se usa el dato
HDAL	Programación de la capa de abstracción HDAL
WinCC	Datos para la HMI
D.U.	Sensores-actuadores de la Unidad de Distribución
Local	Programación del PLC

Una vez definidos estos términos, la Tabla 7 muestra la estructura de variables dentro de la programación del controlador empleando estas definiciones.

Tabla 7. Variables de programa del PLC FEC Standard.

Tipo	Nombre	Dirección	Origen	Propósito	Comentario
Bandera	*MOV_DER*	F0.00	HDAL	Local	Orden de movimiento horario
	MOV_IZQ	F0.01	HDAL	Local	Orden de movimiento antihorario
	ACCION	F0.02	HDAL	Local	Acción de un botón
	_botar	F5.02	Local	WinCC	El objeto puede ser liberado
	_recoger	F5.03	Local	Local	Pieza en bandeja
	_pieza_di	F8.00	Local	WinCC	Se puede recoger la pieza disponible
	_intermed	F8.01	Local	WinCC	Brazo giratorio en posición intermedia
	_peso	F10.00	Local	HDAL	Orden para reproducir peso
	_carga	F10.01	Local	HDAL	Brazo en posición de carga
	_descarga	F10.02	Local	HDAL	Brazo en posición de descarga
Entrada	*_espera*	I0.01	D.U.	Local	Cilindro en posición de espera
	_fincil	I0.02	D.U.	Local	Cilindro en el final del movimiento
	_pieza_to	I0.03	D.U.	Local	Pieza tomada de la bandeja
	_pos_carg	I0.04	D.U.	Local	Sensor en posición de carga activo
	_pos_desc	I0.05	D.U.	Local	Sensor en posición de descarga activo
Salida	*_activo*	O0.00	Local	D.U.	Cilindro transporta pieza a la bandeja
	_succion	O0.01	Local	D.U.	Orden para activar válvula de succión
	_soltar	O0.02	Local	D.U.	Orden para activar impulso expulsor
	_mov_izq	O0.03	Local	D.U.	Movimiento antihorario
	_mov_der	O0.04	Local	D.U.	Movimiento horario del brazo.

Así, la programación del PLC contempla estas y otras variables de apoyo, para generar el funcionamiento deseado. Aunque sería excesivo describir cómo se concretó esa programación para cada una de las señales de salida y marcas de memoria con propósito no

local en el controlador, se exhiben aquellas de mayor importancia para obtener el resultado final, las más descriptivas del proceso y a su vez, relevantes para comprender el resto de etapas en el programa.

Las órdenes emitidas por el operador a través de la interfaz háptica, eran recibidas en el controlador a través de marcas de memoria predispuestas para tal fin. En el caso de una voluntad para que el Brazo emprendiera el movimiento en sentido horario (o hacia la derecha), el código implementado es el siguiente:

```
""ordenar movimiento hacia la derecha
 IF                   MOV_DER        'movimiento derecha
       AND     N      _pos_desc      'brazo en posición de descarga
       AND     N      _mov_izq       'orden de movimiento a la izquierda
 THEN  SET            _mov_der       'orden de movimiento a la derecha
 OTHRW RESET          _mov_der       'orden de movimiento a la derecha
```

Lo que hace la primer línea de código, luego del comentario, es establecer la pregunta: ¿está activa la señal *MOV_DER*? De ser así, entonces pasa a la línea siguiente, donde se revisa además: ¿también se encuentra la señal *_pos_desc* en nivel bajo? Esto último también aplica para *_mov_izq*. Habiendo aprobado todas las condiciones, se asigna un nivel alto a *_mov_der* que es la orden interna al actuador de la Unidad de Distribución (el Brazo Giratorio) para que reproduzca un movimiento en sentido horario. Igualmente en el caso de que alguna de las condiciones no se cumpla, se procede a asignar a *_mov_der* un valor lógico de cero, gracias al *reset* que se presenta en la última línea, ejecutado gracias al condicional *othrw* que quiere decir “en cualquier otro caso”, cuando no se satisface la combinación de condiciones específica programada.

Así las cosas, un medio para representar el estado de la señal *_mov_der* en todo momento, está ilustrada en el circuito lógico de la Figura 14.

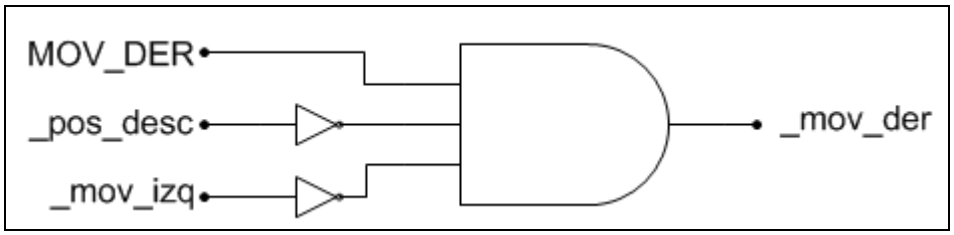

Figura 14. Descripción lógica para la señal de actuación *_mov_der*.

Tal y como se desprende de observar el circuito lógico de la Figura 14, no es difícil determinar que para ordenar el movimiento del actuador giratorio en sentido horario o hacia la derecha, debe darse una orden directa desde la interfaz háptica pero además, el mismo no

debe estar ya en posición de descarga (extremo máximo derecho de movimiento) y tampoco debe estar a la vez activa la señal de actuación opuesta (hacia la izquierda), *_mov_izq*.

Con lo anterior se introduce un criterio de diseño importante: en caso de ya estar en el límite derecho o izquierdo de movimiento para el brazo giratorio de la Unidad de Distribución, no se podía generar una orden de movimiento en ese sentido según el caso. Además, se podía generar una señal que pudiera ser empleada en la programación del Falcon para indicar la condición de posición extrema. De esta manera se ideó representar esta situación por medio de la interfaz para que fuera perceptible para el operador, impidiendo que el mando de la misma pudiera ser llevado en el sentido donde no pudiera ser movilizado también el Módulo de Cambio.

En el caso de la válvula que succiona objetos en el extremo del brazo giratorio, el código implementado que relaciona las señales apropiadas para su correcto funcionamiento se muestra a continuación:

```
""ordenar la succion del objeto
 IF                  ACCION         'acción de botón de la interfaz h
       AND           _pos_carg      'brazo en posición de carga
       AND     N     _pos_desc      'brazo en posición de descarga
       AND     N     _pieza_to      'la pieza ha sido tomada
 THEN  SET           _succion       'activar la válvula de succión
```

Haciendo una analogía con la anterior explicación, se puede concluir que la señal *_succion* sólo será activa en caso de que el operador pretenda hacerlo, pulsando un botón en el mando de la interfaz (*ACCION*). Además de ello, el Brazo debe estar en posición de carga (*_pos_carg*), o de otro modo la válvula de succión nunca se activará. Por otro lado, se revisa además que tampoco esté en posición de descarga (*_pos_desc*) ni que ya se haya tomado una pieza (*_pieza_to*) y se quiera volver a accionar la válvula.

Para desactivar la señal, una vez ha sido activada se deben dar igualmente otras condiciones. El código en STL que lo hace se presenta como sigue:

```
""ordenar soltar objeto
 IF                  ACCION         'acción de botón de la interfaz h
       AND           _pos_desc      'brazo en posición de descarga
       AND     N     _pos_carg      'brazo en posición de carga
       AND           _pieza_to      'la pieza ha sido tomada
       OR        (
             N     _pieza_to      'la pieza ha sido tomada
       AND     N     _pos_carg )    'brazo en posición de carga
 THEN  RESET         _succion       'activar la válvula de succión
```

De lo anterior, puede verse que sólo se libera el objeto cuando el brazo está en posición

de descarga y se indica por medio de la pulsación de un botón que se desea ejecutar esa orden. De esta forma, el circuito lógico que describe las instrucciones de *set* y *reset* antes mostradas, en conjunto para la señal *_succion*, se presenta en el diagrama de la Figura 15.

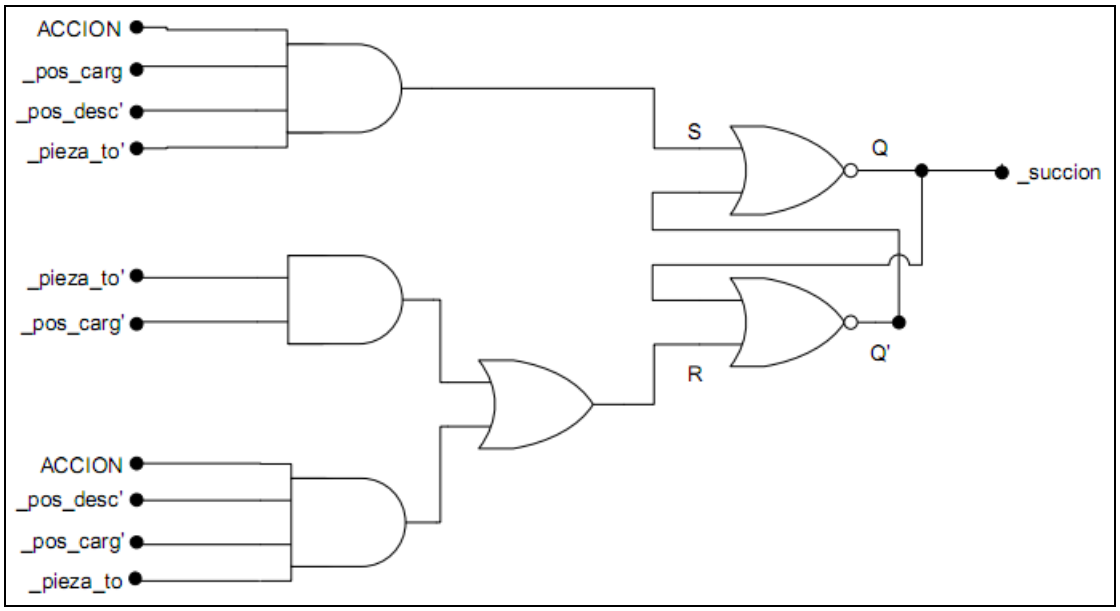

Figura 15. Diagrama lógico para la señal de actuación *_succion*.

El circuito lógico mostrado en la Figura 15, se confeccionó ahora indicando una señal negada por medio del símbolo de una comilla en la parte superior, en lugar de indicar explícitamente un inversor. Se puede apreciar que la señal *_succion* es generada por dos condiciones independientes, y que en conjunto ambas pueden ser sumidas en un *latch* RS o set-reset. De esta modo, sólo cuando la señal S del *latch* esté activa, conectada a la etapa anterior ya descrita, se succionará el objeto y así será hasta que se indique que se desea liberar (la señal se mantiene en nivel alto). Análogamente, sólo con la señal R activa, se soltará el objeto dado que la señal *_succion* estará en nivel bajo.

De la misma manera, se pueden encontrar modelos similares para toda la programación del PLC. Resulta importante encontrar este tipo de representación ya que permite adaptar la estructura dispuesta en FST a otro tipo de entorno de desarrollo, como por ejemplo STEP7 para el caso de que se quiera usar controladores Siemens S7-300. Cabe resaltar que esta codificación resultó muy positiva para las intenciones y expectativas del Proyecto.

5.4. Programación del canal DDE

Las programaciones de la interfaz háptica y el PLC, anteriormente descritas, requerían de un medio para poder intercambiar información ya que sólo así la solución planteada tenía éxito. El bloque faltante consiste en un canal de datos para emitir las órdenes generadas por el operador a través de la interacción con el mando del Falcon, en un sentido, y en el otro la realimentación de información generada por la ejecución de las órdenes en el entorno de operación (Unidad de Distribución).

Para construir ese canal, se utilizó la librería DDEML para el lenguaje de programación C++, ya empleado para configurar las funciones de HDAL y crear el programa principal. La Tabla 8 muestra la secuencia de etapas que se siguieron para lograr entablar una comunicación entre el programa principal en C++ y el servidor de datos de FST.

Tabla 8. Etapas de desarrollo para el canal DDE.

Etapa	Elemento	Características
1	Declaración de variables, manejadores y cadenas de direccionamiento	• Inclusión de las librería DDEML en el código • Creación de una estructura de variables requerida por DDEML
2	Inicialización del canal	• Asignación de valores para los manejadores del canal
3	Programación para atender la llegada continua de datos	• Establecimiento de una función de llamada automática conforme se generan eventos en el canal
4	Manejar envío de información y acople de las funciones DDEML con el programa principal	• Creación de un ciclo para emitir datos (órdenes) al servidor conforme se actualizan • Integración de estas funciones al programa principal (interacción con el código HDAL)

A continuación se comenta cómo se realizó lo mencionado en la Tabla 8 con algunos ejemplos traídos a partir del código generado.

```
//(…)
//paso 1: incluir librerías
#include <ddeml.h>
//(…)
//paso 2: creación de espacio de memoria compartida
HDDEDATA hData;
//paso 3: definición de parámetros para enviar al servidor
char programa_dde[]="FPC_DATA";
char tema_dde[]="FPC_1";
//paso 4: declaración de manejadores
DWORD indicador=0;
HSZ hprograma_dde,htema_dde;
HCONV hconversacion;
//(…)
//paso 5: programa de inicialización
void inicializacion_dde()
{
    //paso 6: asignación de manejadores de canal
    hprograma_dde=DdeCreateStringHandle(indicador,programa_dde,0);
    htema_dde=DdeCreateStringHandle(indicador,tema_dde,0);
    //paso 7: manejador de conversación
    hconversacion=DdeConnect(indicador,hprograma_dde,htema_dde,NULL);
    //(…)
    return;
}
```

En el bloque de código anterior, se ilustran las dos primeras etapas del proceso, indicadas en la Tabla 8. El paso 1 es la inclusión de la librería para el manejo de las funciones DDE. El paso 2 es una definición de un tipo especial de datos propio de la librería DDEML y que permite apartar un bloque de memoria utilizado para las transacciones que se programarán. En el paso 3 se definen variables que contienen el nombre del programa y el tema de conversación que se tendrá durante la ejecución del programa, en este caso se refieren a la nomenclatura propia del servidor de datos IPC Data Server, programa complementario a FST. El paso 4 plantea la definición de manejadores para la conversación. En el paso 5 se introduce la función de inicialización, la cual contiene la asignación de valores para los manejadores (paso 6) y por último la apertura del canal a través de la función *DdeConnect()* en el paso 7 a la cual se le pasan como parámetro los manejadores ya asignados y que a su vez, devuelve un seudónimo del canal para ser utilizado posteriormente al querer efectuar una transacción a través de él.

Las funciones en la etapa 3 se observan en el siguiente código:

```
//(…)
//paso 1: método Callback
HDDEDATA CALLBACK DdeCallback(
    //paso 2: parámetros específicos de la función
    UINT uType,     // Transaction type.
    UINT uFmt,      // Clipboard data format.
    HCONV hconversacion,    // Handle to the conversation.
    HSZ hsz1,       // Handle to a string.
    HSZ hsz2,       // Handle to a string.
    HDDEDATA hData_, // Handle to a global memory object.
    DWORD dwData1,  // Transaction-specific data.
    DWORD dwData2)  // Transaction-specific data.
{
      //paso 3: discriminar etiquetas que llegan desde el servidor
      switch(uType)
      {
            //(…)
            //paso 4: etiqueta de llegada de datos
            case XTYP_ADVDATA:
                  //paso 5: copia de bloque de memoria en variable
                  dato_char=(char *)DdeAccessData(hData_,NULL);
                  //paso 6: liberación del bloque
                  DdeUnaccessData(hData_);
                  //paso 7: retorno de éxito en la transacción
                  return (DDERETURN) DDE_FACK;
                  break;
      }
      return (DDERETURN) 0;
}
```

Como se puede apreciar, se define una función especial, propia de DDEML. Esta es *DdeCallback(),* y se muestra en el paso 1. El paso 2 contiene definiciones propias de este método, siempre necesarias al ser programado. Lo más importante es la definición de un manejador nuevo, aquí llamado *hData_*; bloque de memoria asignado así para las transacciones referentes a la llegada de datos desde el servidor. El paso 3 es especial: se define una estructura que discrimina entre las etiquetas asignadas para cada transacción por el servidor. Existen variedad de estas, pero lo interesante en este caso es lo que se muestra en el paso 4. Cuando el servidor emite al cliente el valor *XTYP_ADVDATA*, lo hace porque se ha generado un envío de información y está disponible para ser captado por el cliente. Por ello, en el paso 5 por medio de la función DDEML *DdeAccessData()* se puede copiar el valor fresco en una variable, en este caso, un *string* o cadena de caracteres. En el paso 6 se indica que se ha terminado de obtener el dato y en el paso 7 se indica que se ha terminado el proceso.

Cabe mencionar que con esta estructura siempre que el servidor indique al cliente que hay un nuevo dato para ser recibido, este último siempre querrá refrescar su información de manera igualmente continua. La cuarta etapa mencionada en la Tabla 8 está descrita a través

de las siguientes líneas de código.

```
//(…)
//paso 1: configuración para solicitar datos
void DDERequest(DWORD indicador,HCONV hconversacion,char* szItem)
{
      HSZ hszItem=DdeCreateStringHandle(indicador,szItem,0);
      hData=DdeClientTransaction(NULL,0,hconversacionx,hszItem,CF_TEXT,
      XTYP_ADVSTART,TIMEOUT_ASYNC,NULL);
      return;
}
//(…)
//paso 2: configuración para el envío de datos
void DDEPoke(DWORD indicadorx,HCONV hconversacion,char* szItem,
char* szData)
{
      HSZ hszItem=DdeCreateStringHandle(indicador,szItem,0);
      DdeClientTransaction((LPBYTE)szData,(DWORD)(lstrlen(szData)+1),
      hconversacion,hszItem,CF_TEXT,XTYP_POKE,3000,NULL);
}
//(…)
//paso 3: ciclo principal
int main()
{
      //(…)
      //paso 4: solicitud de datos
      DDERequest(indicador,hconversacion,_palabra);
      //(…)
      //paso 5: condición para terminar el programa
      while(clase_h.cerrar==false)
      {
            //(…)
            //paso 6: envío de datos al servidor
            DDEPoke(indicador,hconversacion,_m_der,_tmp1);
            //(…)
      }
      //(…)
      return 0;
}
```

El paso 1 en el bloque anterior, define una función que contiene lo necesario para indicar a DDEML que se desea efectuar transacciones de recibo de información. Lo más destacable es el parámetro *XTYP_ADVSTART* pasado a la función *DdeClientTransaction()*, que indica que se desea recibir datos continuamente una vez que la variable en cuestión haya presentado cambios en las operaciones que maneja el servidor. Esto resultó conveniente para las intenciones del Proyecto: cada vez que surgía un suceso en la Unidad de Distribución con datos útiles para la programación HDAL y la reproducción de estímulos hápticos, la información era actualizada en el servidor y de este modo llegaba de manera fluida, sin un retardo apreciable. El canal programado era continuo y dinámico.

En el paso 2 se ilustra la forma en que es solicitada una transacción de envío de datos al

servidor desde el cliente. La misma función, *DdeClientTransaction()*, ahora es invocada con el parámetro *XTYP_POKE*. Esto indica que se desea emitir datos y que estos deben ser atendidos por el servidor.

El paso 3 muestra la creación del flujo principal del programa. En él, se indica que el paso 4, hace la llamada al método *DDERequest()* ya comentado. Se le pasan los manejadores de conversación adecuados y la variable donde se depositan los datos recién llegados. Una vez hecho esto, se crea un hilo de ejecución en paralelo porque siempre que el dato presente cambios, será actualizado de forma inmediata ahora sin la atención del resto del programa. Este proceso se realiza de forma automática dada la estructura programada con anterioridad. En el paso 5 se genera un ciclo hasta que se pretenda cerrar el programa por el usuario, a través de una indicación introducida por medio de la acción de uno de los botones del Falcon, específicamente programado para este propósito. Este evento es indicado por medio de la variable *cerrar*, perteneciente a la clase creada para configurar las funciones HDAL.

En el paso 6, se muestra un ejemplo de una solicitud para el envío de información al servidor, en este caso, el valor la variable *_m_der* que representa una solicitud para que el Brazo Giratorio de la estación MPS se mueva hacia la derecha. Ya que el ciclo *while* se ejecuta con cada ciclo de programa, se determinó que entonces los datos iban a ser enviados a una frecuencia muy alta de forma innecesaria. Se creó entonces una estructura (no mostrada en el ejemplo) para evitar esto: los datos son enviados sólo si se presenta un cambio. El mismo dato no es enviado más de una vez consecutivamente con el mismo valor.

Todo lo anterior permitió sostener una conversación eficiente durante la ejecución del programa. Los datos fueron emitidos y recibidos de forma continua y el sistema presentó el comportamiento esperado.

5.5. HMI creada en WinCC

A través del programa WinCC se creó una interfaz gráfica para la monitorización de los procesos y obtener una noción del estado del sistema remotamente. Se provee de información referente a la emisión de órdenes y la respuesta de los actuadores, por medio de una imagen representativa del equipo Teleoperado. Se han ubicado mensajes y etiquetas que aparecen conforme se suscitan los eventos mencionados e indican al operador en qué momento puede realizar una u otra acción. La obtención de datos se hace configurando un

enlace DDE con el servidor de datos, el programa IPC Data Server de FST.

Por ejemplo, cuando se encuentra disponible un objeto en la bandeja para ser recogido por el Brazo Giratorio de la Unidad de Distribución, y además este actuador se encuentra en posición de carga, se puede observar lo que muestra la Figura 16. Las etiquetas de texto sólo ocultan la información cuando no están activas, las mismas tienen fondo blanco sobre la imagen.

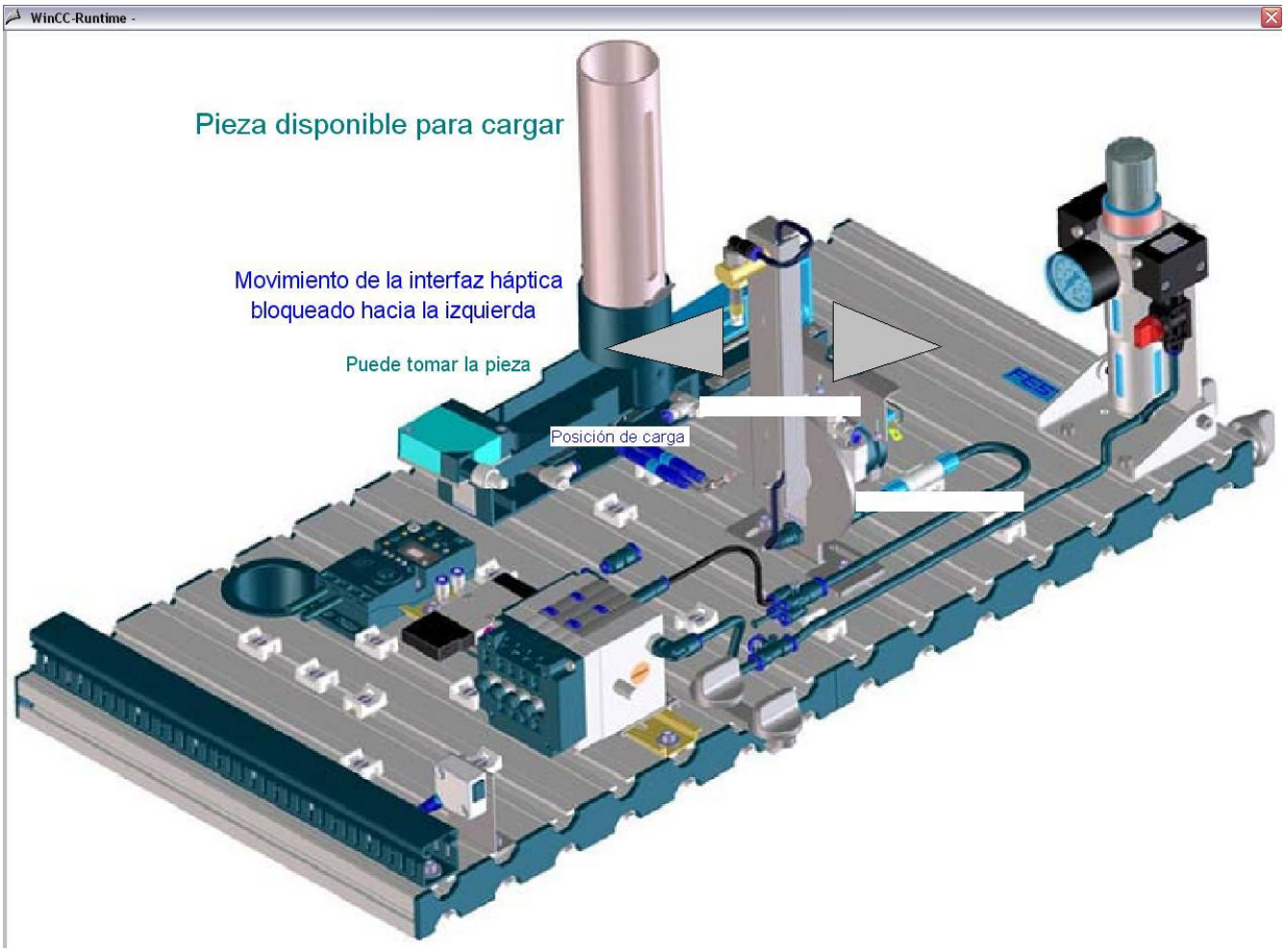

Figura 16. Visualización de estado del equipo manejado por la HMI.

Como se observa en la Figura 16, la HMI informa de que el Brazo Giratorio está en posición de carga, que un objeto está disponible y de que está bloqueado el movimiento hacia la izquierda. En otras condiciones, se brinda información cuando el actuador está en otra posición, sea esta intermedia o en el extremo de descarga, cuando el mismo se está moviendo en un sentido o en el otro (gracias a los indicadores dispuestos a los lados de la representación del Módulo de Cambio), y del momento en el que no hay objetos disponibles, por ejemplo.

CAPÍTULO 6. ANÁLISIS DE RESULTADOS

El diagrama presentado en la Figura 12, en el Capítulo de Descripción Detallada de la Solución, introduce de manera global el sistema desarrollado. Uno de los objetivos era obtener un modelo general para un sistema Teleoperado de Automatización integrando una interfaz háptica. Generalizando entonces los resultados obtenidos para el entorno y las condiciones específicas del Proyecto, se encuentra la estructura descrita a continuación.

1. Interfaz háptica: Emite órdenes y reproduce estímulos hápticos acordes a la interacción del equipo Teleoperado con su entorno. Se comunica con la PC por lo que debe tener incorporado un medio para intercambiar datos de forma continua. Las señales de realimentación, generadas desde el equipo remoto, son interpretadas aquí.
2. PC: Agrupa la programación de las funciones de la interfaz háptica. Proporciona los medios para entablar comunicación con el PLC a través de un canal de datos. A la vez, permite monitorear los procesos generados en el equipo Teleoperado.
3. Red: Permite el acceso a la información de las órdenes emitidas por el operador y administradas por el PLC. A la vez la programación residente en la PC, tiene disponible los datos que se generan conforme cambian las condiciones en el equipo operado de remotamente. Todo esto sucede gracias al direccionamiento propio de la estructura de red. El uso de la red para interconectar ambos dispositivos, sin la necesidad de que físicamente compartan el mismo lugar, permite decir que el sistema es Teleoperado.
4. PLC: Controlador de actividades del equipo a comandar. Es un intermediario entre la PC y el equipo remoto, aunque es mayormente necesario para ejercer control de los actuadores y recibir la información proveniente de los sensores, dispuestos para recibir la información del estado del equipo.
5. Equipo remoto: Es el conjunto de actuadores y sensores controlados y administrados por el PLC. Las órdenes del operador son ejecutadas.

Por otro lado, era necesario cumplir con el hecho de que la realimentación de fuerza debía tener un valor cercano a la fuerza máxima reproducible por el Falcon. Gracias al modelo masa-resorte empleado para la programación de estos estímulos, se pudo explotar todo el potencial del dispositivo y usarlo a conveniencia en el desarrollo del Proyecto. Se introducen algunas definiciones importantes con la Tabla 9.

Tabla 9. Definición de términos para evaluar la reproducción de fuerzas.

Eventos	Nivel de fuerza	Significado
Orden de movimiento		El mando del Falcon se pretende llevar hacia alguno de los extremos
Posición del brazo		Posición del Brazo Giratorio en el momento de emitir la orden
	Bajo	Movimiento libre del Brazo Giratorio
	Moderado	Dificultad de movimiento con un objeto sujetado por la Válvula
	Alto	Limitación total de movimiento (reproducción máxima de fuerza)

Luego de esto, la Tabla 10 muestra las condiciones que el operador puede percibir, gracias al Falcon, en cuanto a la interacción con el entorno y los estímulos de realimentación de fuerza programados.

Tabla 10. Combinación de eventos y reproducción de fuerzas en el eje *x*.

Eventos en la Unidad de Distribución					**Nivel de fuerza**
Orden de movimiento		Brazo en posición de		Objeto sujetado	
Derecha	Izquierda	Carga	Descarga		
Sí	No	No	No	No	Bajo
Sí	No	No	No	Sí	Moderado
Sí	No	No	Sí	Sí	Alto
Sí	No	No	Sí	No	Alto
Sí	No	Sí	No	Sí	Moderado
Sí	No	Sí	No	No	Bajo
No	Sí	No	No	No	Bajo
No	Sí	No	No	Sí	Moderado
No	Sí	Sí	No	Sí	Alto
No	Sí	Sí	No	No	Alto
No	Sí	No	Sí	Sí	Moderado
No	Sí	No	Sí	No	Bajo

Según la información suministrada por la Tabla 10, se puede determinar que el operador del sistema experimenta a través de la interfaz háptica, estímulos de fuerza consistentes con lo que puede al mismo tiempo observar que sucede en el entorno manipulado. Se destaca que en los casos donde el objeto se ubica en uno de los extremos (posición de carga, por ejemplo) y el mando se quiere llevar en el mismo sentido (movimiento hacia la izquierda, siguiendo con el ejemplo), se presenta una total limitación y la acción no se puede llevar a cabo. En cualquier otro caso, el sistema permite el desplazamiento condicionado a la condición de peso: objeto sujetado por la válvula o no. En el primer caso el movimiento será difícil pero posible, acorde con el hecho de que el Brazo Giratorio estaría efectuando un esfuerzo mayor para levantar el objeto. En el segundo caso, el movimiento del Módulo de Cambio se da libremente.

Un hecho interesante, es que el programa residente en la PC posee tres hilos de

ejecución. Si a esto se le suma las acciones propias del control dispuesto en el PLC y adicionalmente del monitor de proceso hecho con WinCC, en total se manejan cinco eventos en paralelo (creados directamente por la codificación realizada). A continuación se ofrece un análisis acerca de este hecho, por medio de diagramas de flujo que ilustran lo que sucedía en tiempo de ejecución con los tres hilos del programa de usuario. Por medio de la Figura 17 se muestra la manera en que se genera el hilo principal, del cual surgen los demás.

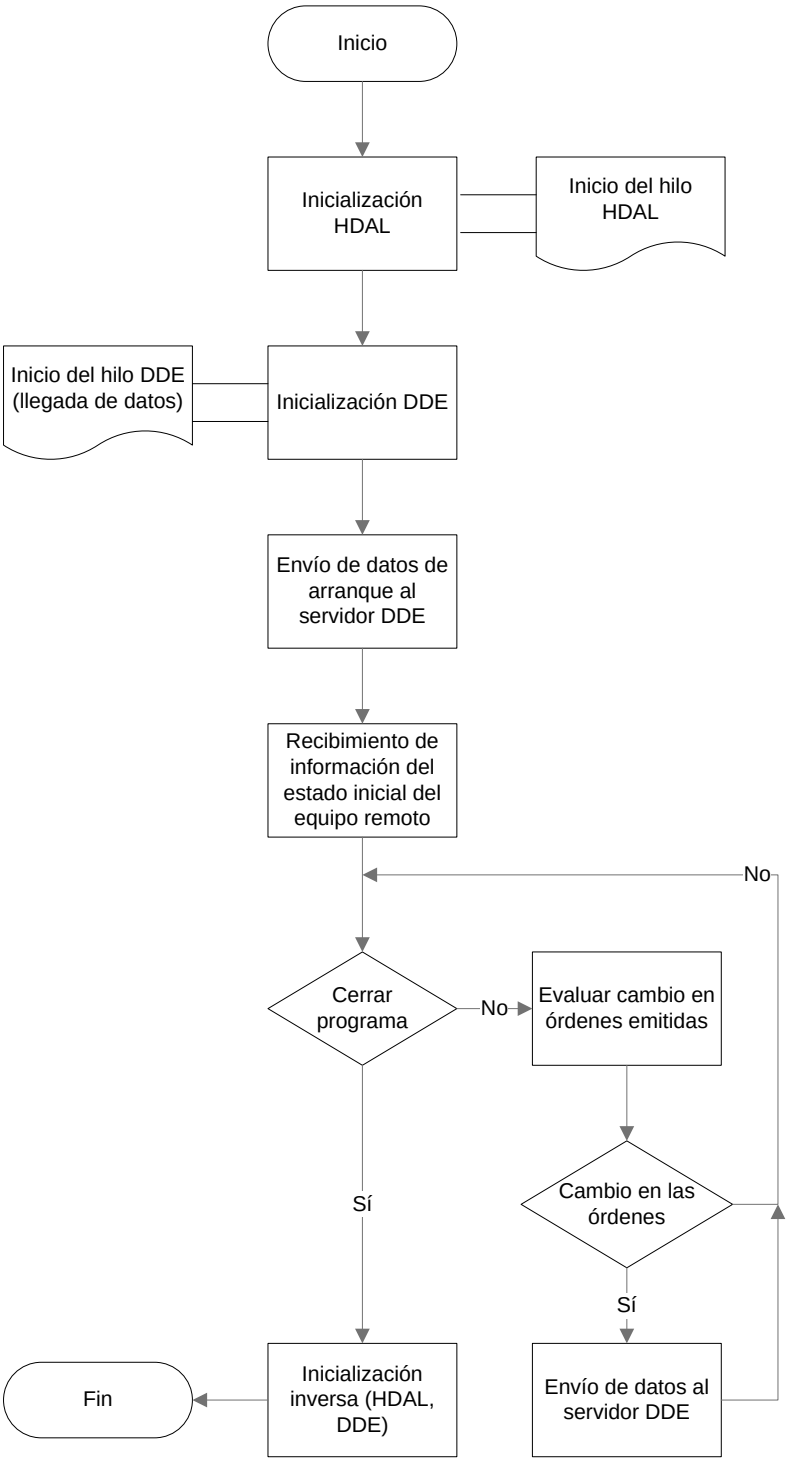

Figura 17. Diagrama de flujo del programa principal.

Como se puede observar en el diagrama de flujo principal de la Figura 17, una vez se inicializan las funciones HDAL y el canal DDE, se presenta el origen de los hilos correspondientes ejecutados paralelamente a este. Cabe mencionar que en el segundo caso, además de cualquier transacción de la llegada de datos administrada por el cliente

programado, se maneja un envío y arribo de información en particular, justo después de la inicialización. Esto es para obtener el estado del sistema al inicio; el Brazo Giratorio estará ubicado en alguna posición, sea un extremo o en el medio. Se solicita al servidor de datos el valor de las mediciones de los sensores desde el arranque para proceder de forma coherente conociendo este aspecto. Además, se envía antes un arreglo de órdenes neutrales, para estabilizar las funciones del PLC.

El hilo de ejecución de la capa de abstracción HDAL, se describe a partir de la información suministrada en el diagrama de flujo de la Figura 18.

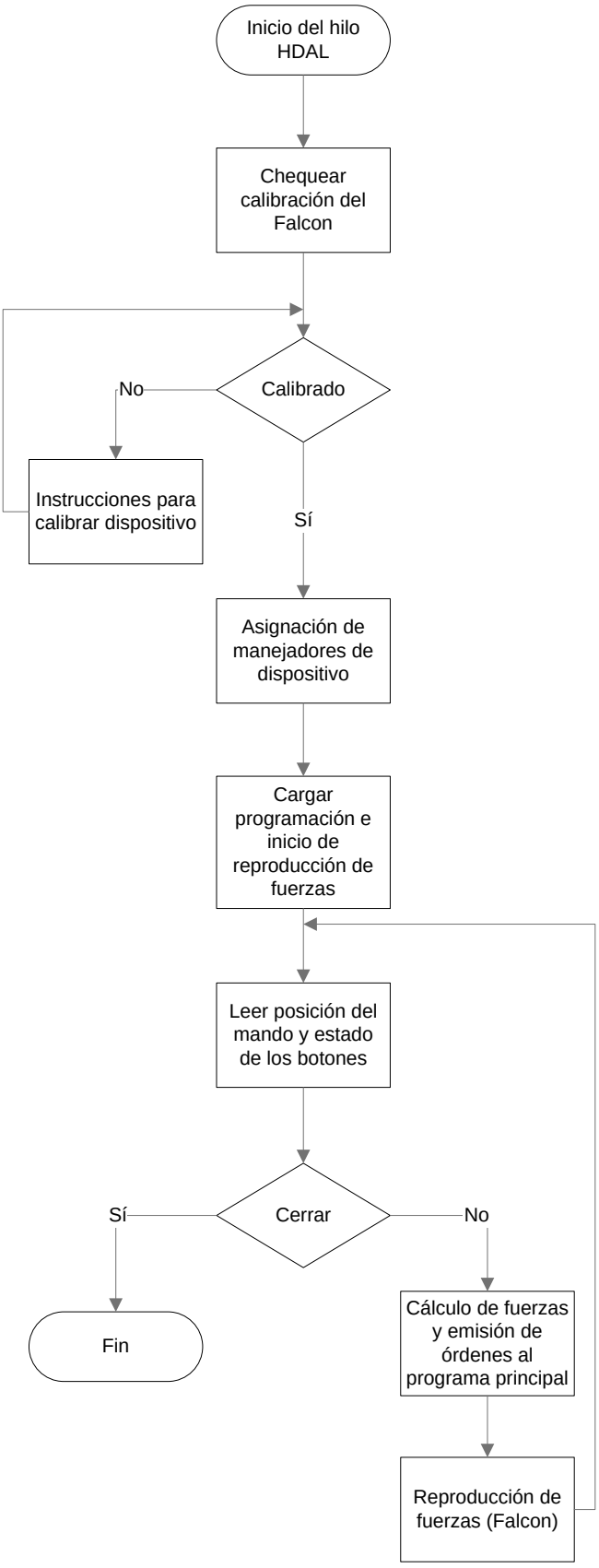

Figura 18. Diagrama de flujo del hilo HDAL.

Como se puede apreciar en el diagrama de flujo de la Figura 18 para las funciones hápticas desarrolladas a través de HDAL en el Proyecto, se crea un hilo de ejecución paralelo al programa principal. Esto sucede ya que la estructura y disposición de la capa de abstracción mantiene en determinado momento la reproducción de fuerzas en un *loop* que sólo se rompe en el momento que desde el hilo principal se emite la orden de cerrar con el programa. Primeramente la inicialización se generó en el flujo descrito en la Figura 17, acá se ejecutan las órdenes inmediatamente necesarias antes de comenzar con la reproducción de fuerzas. El arranque de este proceso se da con valores iniciales predispuestos (cero), posteriormente conforme comienza el ciclo, según la interacción del operador con el mando de la interfaz, estos valores serán alterados.

Por otro lado, para el canal DDE se muestra también su hilo de ejecución en el diagrama de flujo de la Figura 19.

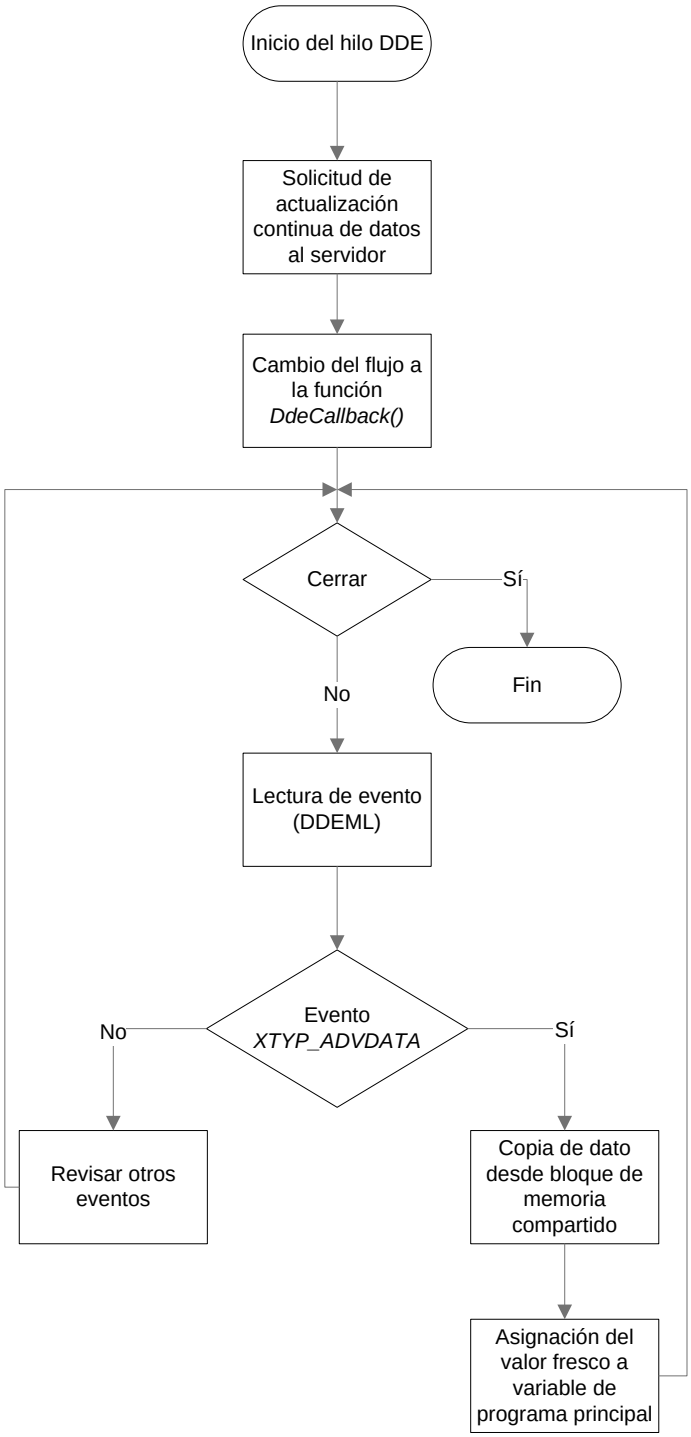

Figura 19. Diagrama de flujo para el hilo de recepción de datos DDE.

El diagrama de flujo de la Figura 19, introduce lo que sería la ejecución del proceso cíclico para la recepción y actualización de información, a través del servidor de datos, pasando por el canal DDE programado y que viene desde donde se originan los eventos en la Unidad de Distribución como respuesta a las órdenes emitidas por el operador. Primeramente se realiza la solicitud para obtener la información desde el servidor de manera

fluida, conforme se generaban actualizaciones en el estado de las variables requeridas. Luego el flujo se dirige a la función *DdeCallback()*, que espera que el servidor genere notificaciones de eventos. Aquí los mismos son leídos, procesados y analizados, como en el caso de la llegada de datos nuevos. El valor es copiado en las variables manejadas por el programa principal para pasárselo a su vez a la programación HDAL. Todo esto sucede de cíclica y automáticamente una vez generada esta configuración.

CAPÍTULO 7. CONCLUSIONES Y RECOMENDACIONES

7.1. Conclusiones

1. Es posible acoplar la interfaz háptica Novint Falcon a un sistema de Automatización.

2. El Falcon puede ser programado para su uso junto con entornos reales.

3. El modelo masa-resorte permitió la reproducción de fuerzas por medio de la interfaz háptica con la posición del mando como entrada.

4. La Unidad MPS FESTO Distribución puede realizar procesos programados de manera distinta a la configuración predispuesta de fábrica.

5. Una combinación de señales de los sensores de la Unidad de Distribución proporciona la información necesaria para conocer cualquier estado de operación del equipo.

6. DDE permite crear un canal de datos entre un programa de usuario y un sistema de Automatización.

7. La HMI creada con WinCC permitió conocer el estado del sistema al ser operado remotamente.

8. El sistema presentó una respuesta dinámica ante las órdenes emitidas de forma continua.

7.2. Recomendaciones

Siempre es bueno determinar hasta dónde se puede llegar con los recursos disponibles. Muchas veces se cree que es necesario adquirir nuevos equipos o herramientas de software adicionales cuando los medios que se tienen al alcance son suficientes para las condiciones dadas.

Por otro lado, es posible explorar nuevas ideas para una continuación del Proyecto. Una lista de “cosas por hacer” se ofrece a continuación.

- Mejorar la HMI desarrollada con WinCC: Puede integrarse un módulo de control adicional desde la pantalla de monitorización. Solamente se ha configurado un monitor de procesos pero el recurso da para mucho más.
- Programar un canal de datos utilizando OPC: Aunque el enlace DDE desarrollado para la aplicación resultó más que suficiente, es posible que en versiones del Sistema Operativo más recientes que MS Windows XP (utilizado para la programación y ejecución del Proyecto) como MS Windows Vista o MS Windows 7, no sea más soportado este medio y sea necesario migrar a una herramienta más moderna. Adicionalmente, en caso de querer escalar las funciones desarrolladas hasta este punto (totalmente posible), quizá el canal DDE no soporte un flujo muy grande de datos sin pérdida de información y una latencia difícil de manejar.
- Integrar información continua de la posición del Brazo Giratorio: En el Proyecto se empleó únicamente información discreta referente a la posición del actuador. Se conocían con certeza dos condiciones, cuando se encontraba en posición de cargar objetos y cuando se encontraba más bien en el sitio de descargarlos. Por analogía, se sabía cuando se encontraba además en una posición intermedia. Incluir el dato continuo relativo a los extremos de movimiento puede generar en la aplicación de eventos interesantes para reproducir fuerzas y otros estímulos en el Falcon.

BIBLIOGRAFÍA

[1] Bjelland, Hans y Tangeland, Kristian. “User-Centered Design Proposals for Prototyping Haptic User Interfaces” [en línea]. HAID 2007 - Haptic and Audio Interaction Design - Second International Workshop. 30 de noviembre de 2007. pp. 110-120.
<http://www.springerlink.com/content/4vu1127kg1g48569/fulltext.pdf> [Consulta: 20 de enero de 2010].

[2] Compañía FESTO. Controllers FEC, Standard [en línea].
<https://xdki.festo.com/xdki/data/doc_enus/PDF/US/FEC-STANDARD_ENUS.PDF> [Consulta: 22 de mayo de 2010]

[3] Compañía FESTO. Manual de la Estación de Distribución (*FESTO Distributing Station Manual*). FESTO Didactic GmbH & Co. KG - Software package FST – Version 4.1, 2004

[4] Compañía Novint. Haptic Device Abstraction Layer (HDAL) Programmer’s Guide. Version 2.1.3. Novint Technologies Incorporated.

[5] Compañía Siemens. Getting Started Step7. Manual de usuario del Software STEP7 v5.1. Siemens.

[6] Guô, Xing P., editor. Robotics Research Trends. 1ª Edición. Nova Science Publishers, Inc. New York, 2008.

[7] Huelin, Félix. Dispositivos hápticos y cirugía robótica [en línea].
<http://robolabo.etsit.upm.es/haptico/haptico-bio-1x2.pdf> [Consulta: 22 de enero de 2010].

[8] Ildar Farkhatdinov y Jee-Hwan Ryu. A Study on the Role of Force Feedback for Teleoperation of Industrial Overhead Crane [en línea]. <http://robot.kut.ac.kr/papers/50240796.pdf> [Consulta: 8 de diciembre de 2009].

[9] MacLean, K. E., Shaver, M. J., y Pai, D. K. Handheld Haptics: A USB Media Controller with Force Sensing [en línea].
<http://people.cs.ubc.ca/~maclean/publics/VR02-handheld.PDF> [Consulta: 8 de diciembre de 2009].

[10] Madrigal M, Randall. “Reconocimiento visual de objetos 2D en la estación de clasificación MPS Festo” [cd-rom] Biblioteca José Figueres Ferrer, ITCR. Agosto de 2009.

[11] Microsoft Developer Network. “About Dynamic Data Exchange” [en línea]
<http://msdn.microsoft.com/en-us/library/ms648774%28VS.85%29.aspx> [Consulta: 3 de abril de 2010]

[12] Murphy, Eric. “Comments: OPC advantages” [en línea]: OPC Foundation
<http://lists.opcfoundation.org/TACblog/Lists/Comments/ViewComment.aspx?ID=5> [Consulta: 24 de Mayo de 2010]

[13] Quinto Q., Carlos. Curso de Automatización I [en línea].
<http://galia.fc.uaslp.mx/~cantocar/automatas> [Consulta: 29 de abril de 2010]

[14] Villalobos A, Harold. “Sistema de control y monitorización de una estación de Distribución MPS” [cd-rom] Biblioteca José Figueres Ferrer, ITCR. Julio de 2008.

ANEXOS

Anexo A.1 Glosario y abreviaturas

- HDAL: *Haptic Device Abstraction Layer* o capa de abstracción del dispositivo háptico.
- DDE: *Dynamic Data Exchange* o intercambio dinámico de datos.
- MPS: *Modular Production Station* o estación modular de producción.
- PLC: *Programmable Logic Controller* o controlador lógico programable. Conocido también como autómata programable.
- FST: Entorno de desarrollo para los PLC del fabricante FESTO.
- HMI: *Human-Machine Interface* o interfaz humano-máquina.
- LIRA: Laboratorio Institucional de Robótica y Automatización.

Anexo B.1 Manual de usuario

Para utilizar el sistema de Automatización comandado con la interfaz háptica Novint Falcon se asume que:

- Se dispone de un equipo con MS Windows XP
- En la PC se ha instalado el driver del Falcon, siguiendo los pasos dados en la documentación del dispositivo háptico.
- Se ha conectado por USB la computadora y la interfaz.
- Se ha conectado el Falcon a la línea eléctrica.
- La fuente de alimentación de la Unidad MPS FESTO Distribución está encendida.
- Existe comunicación con el PLC FESTO FEC Standard por medio del protocolo RS-232 (igualmente configurable por la red, pero para propósitos de este manual se ilustra el uso del sistema empleando este medio) y este se encuentra en modo *RUN*.
- Se ha encendido el compresor de aire acoplado al equipo neumático.

Es necesario proveer de objetos que cargar al Brazo Giratorio, para ello hay que colocar en el cilindro las piezas plásticas hechas para tal fin. Es preferible si solamente se colocan en el cilindro y no en la bandeja; una vez puesto en marcha, el sistema colocará una pieza en ella automáticamente.

Primeramente, se debe cargar la programación para el PLC por medio del programa FST (si no se dispone de experiencia con este entorno de desarrollo, siempre se puede consultar la documentación de FESTO). El proyecto a cargar se llama *"PRIMERO"*. Una vez hecho esto, debe ponerse en marcha la programación por medio del Control Panel, dando la orden de arranque. Esto hará que la programación antes descargada al PLC comience a ejecutarse. El programa FST puede ahora cerrarse.

Debe ejecutarse el programa IPC Data Server, complementario a FST, y comprobar que se encuentra configurado para trabajar por el puerto serie. Además, el PLC direccionado debe ser el "*FPC_1"* en el mencionado programa.

Para utilizar el Novint Falcon, es importante conocer cómo debe ser empleado. A continuación una lista acerca de las consideraciones del caso.

1. El dispositivo háptico es un equipo de precisión. Debe ser utilizado moviendo el mando con una fuerza moderada, sin querer forzar las limitaciones programadas.
2. El mando del Falcon debe ser tomado de la manera correcta. La Figura B.1.1 presenta la forma indicada de manipular el dispositivo. Nótese la colocación de los dedos y la disposición de la mano.

Figura B.1.1. Forma correcta de manipular la interfaz háptica.

3. Los botones del mando cumplen una función vital. En la imagen de la Figura B.1.2 se observa cómo están clasificados estos componentes.

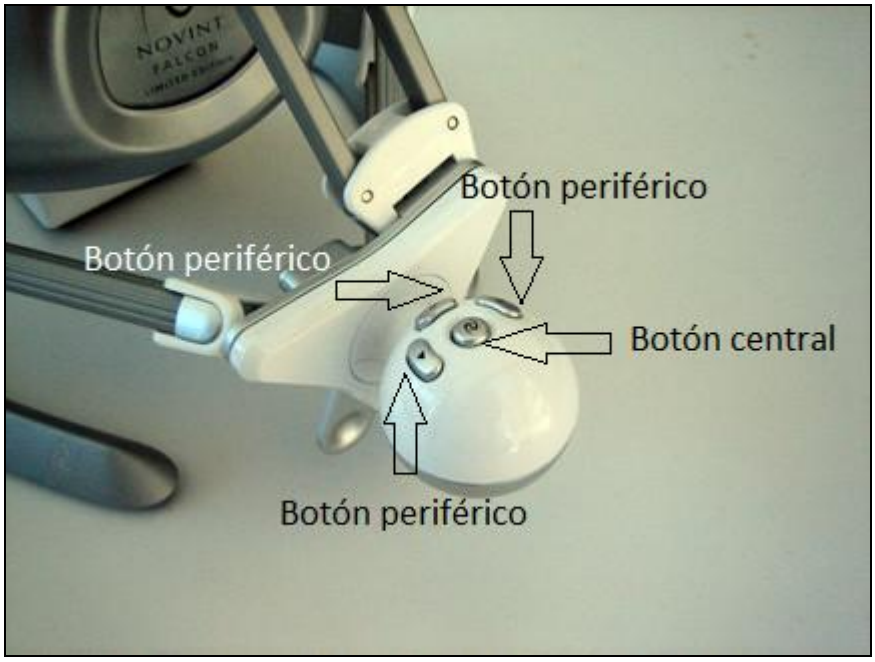

Figura B.1.2. Clasificación de botones.

4. A través de la Figura B.1.2 puede apreciarse que se dispone de un arreglo de tres botones periféricos y un botón central. Para los propósitos de la aplicación, todos los botones periféricos comparten la misma función de recoger y liberar objetos por la Válvula de Succión de la Unidad de Distribución; el botón central cierra el programa.

Ahora debe ejecutarse el programa que contiene la configuración para las funciones de la interfaz háptica. El mismo se encuentra con el nombre *"SEGUNDO.exe"* y está disponible junto con la documentación de este Proyecto. Una vez ejecutado, en la pantalla de la PC aparecerá la etapa previa a la utilización del sistema. La secuencia de posibles eventos se muestra a continuación.

La Figura B.1.3 presenta lo que ocurre al iniciar el programa del Falcon.

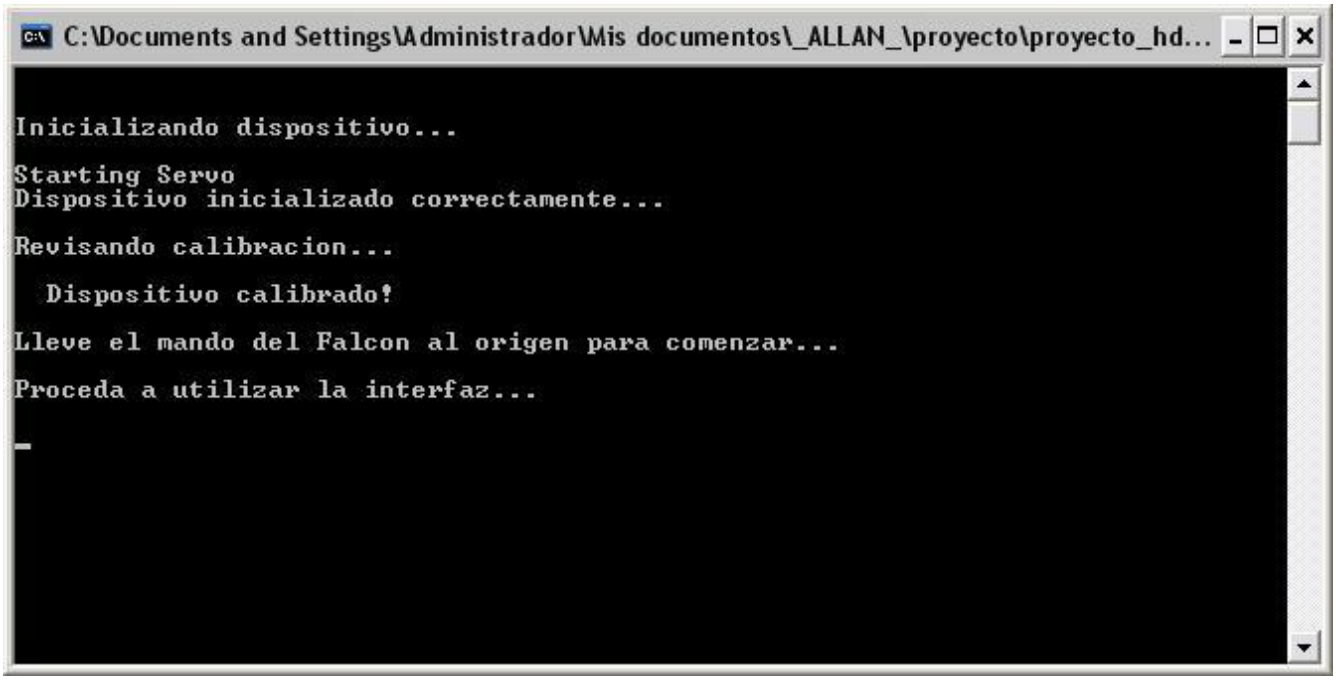

Figura B.1.3. Primer imagen del programa.

Al ejecutar el programa, aparecerá una ventana DOS con la información mostrada en la Figura B.1.3. Se ofrece información acerca de la inicialización del dispositivo háptico y se indica que luego de este proceso se puede utilizar la interfaz. Para comenzar con la reproducción de fuerzas es necesario llevar el mando a la posición neutral, extendiéndolo desde la posición home y acercándolo hacia el usuario, hasta el momento en el que sobre la línea del eje próximo se experimenten las sensaciones de fuerza.

Si por el contrario se observa la información visualizada como en la Figura B.1.4, existe un error que se debe corregir antes.

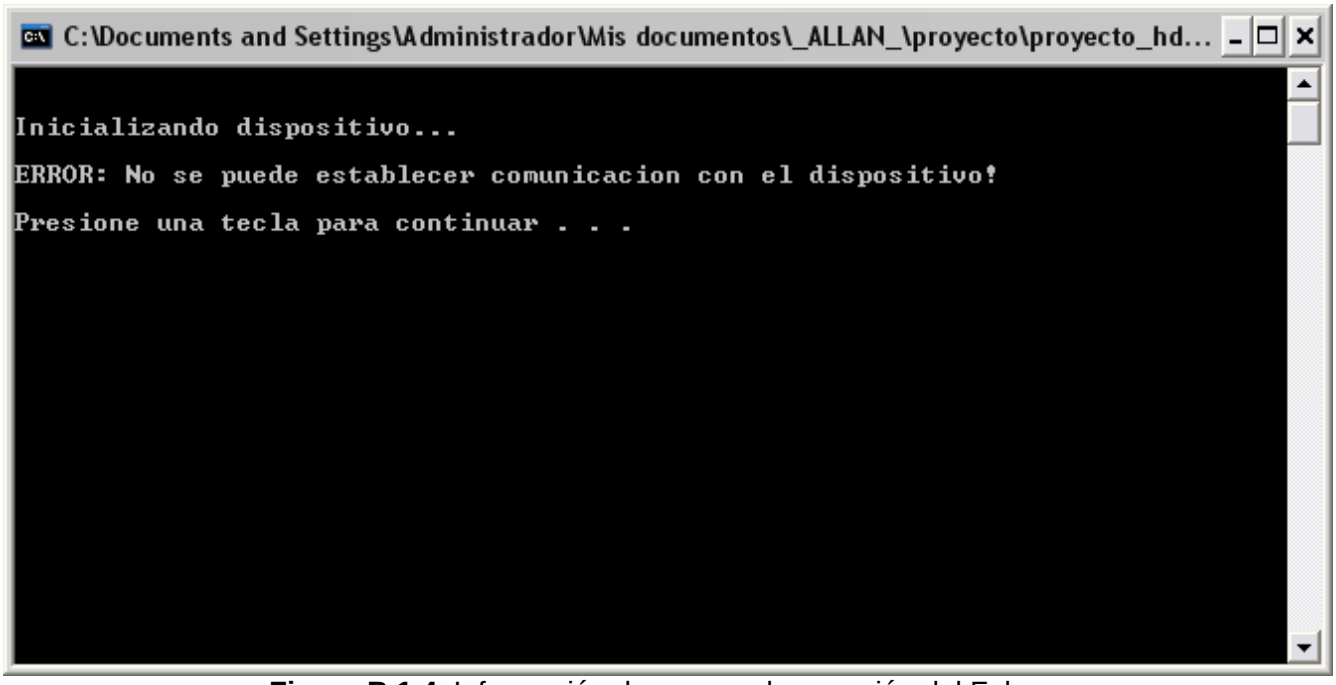

Figura B.1.4. Información de error en la conexión del Falcon.

Lo que muestra la Figura B.1.4 es debido muy probablemente a que no se puede encontrar un dispositivo Novint Falcon conectado a la PC. Es conveniente entonces revisar si el mismo se encuentra bien conectado al puerto USB, y si el driver de Novint está correctamente instalado en el equipo.

Otra situación que se puede presentar en este punto, es la que se muestra en la Figura B.1.5.

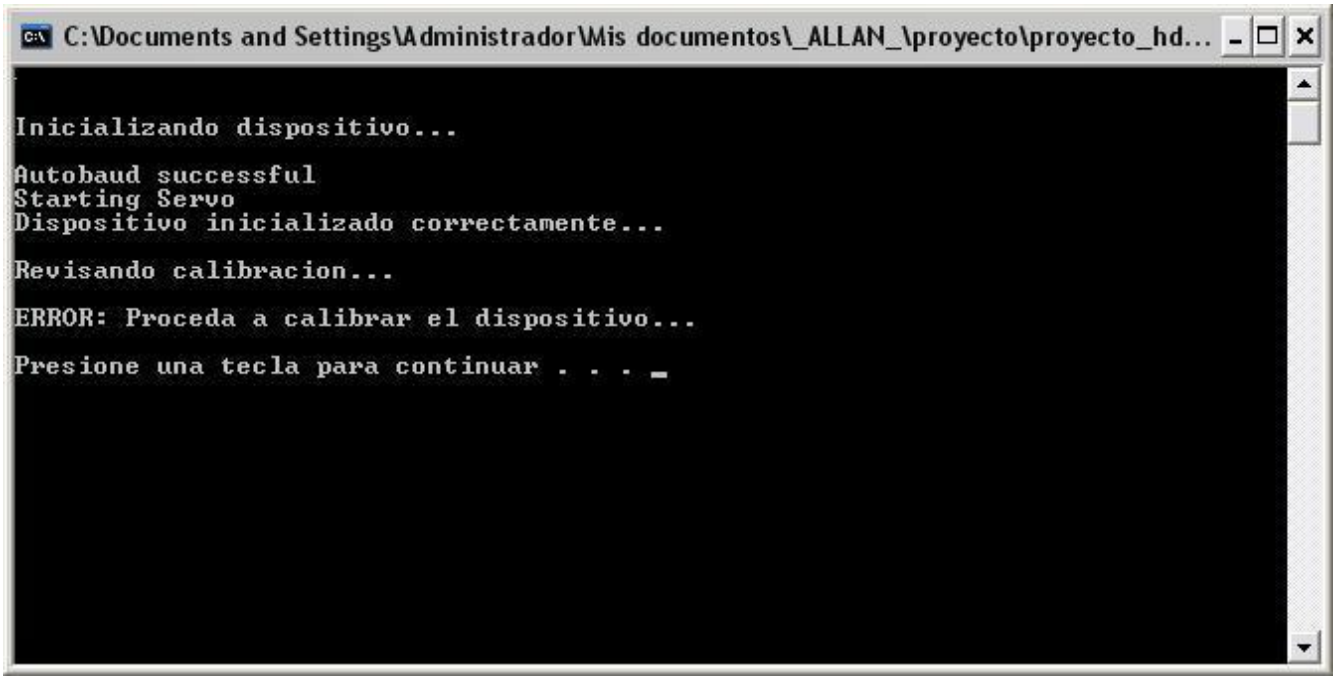

Figura B.1.5. Dispositivo no calibrado.

La Figura B.1.5 muestra cómo se genera una instrucción para que la interfaz háptica sea calibrada. La razón de esto se explica bien en la documentación del Falcon, y comúnmente debe realizarse cada vez que el dispositivo es nuevamente conectado a la PC, o cuando la programación interna lo considere necesario. Hay que extender el mando de la interfaz completamente en todo su rango de acción, acercándolo en dirección del usuario. Luego hay que llevar el mando a la posición home. Una vez hecho esto, aparecerá la información que se muestra en la Figura B.1.6.

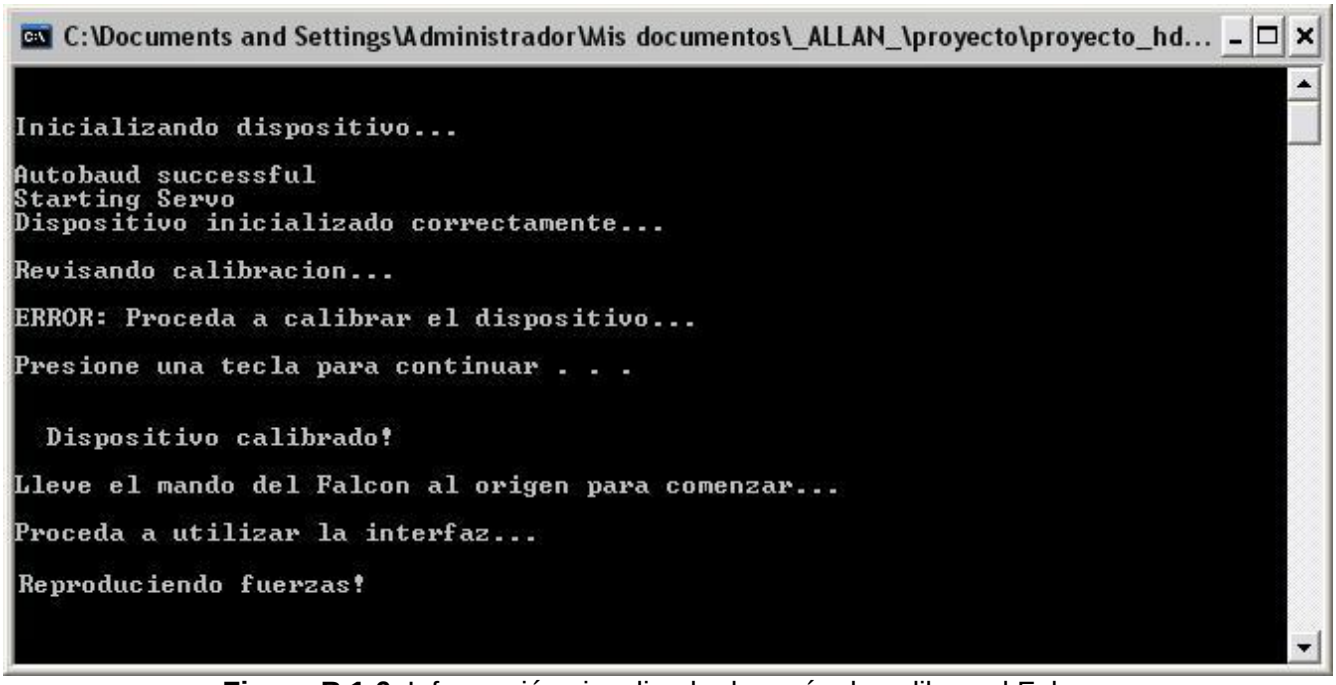

Figura B.1.6. Información visualizada después de calibrar el Falcon.

Una vez completado este proceso, comienza la reproducción de fuerzas, tal y como se observa en la Figura B.1.6. Se podrá percibir una limitación total para desplazar el mando de la interfaz en los sentidos arriba-abajo y cerca-lejos. En el eje horizontal, se experimenta por el contrario libertad de movimiento. Ya para entonces, un objeto estará disponible en la bandeja para ser recogido. El brazo giratorio ahora va a responder a las órdenes emitidas desde la interfaz háptica. Las órdenes posibles se enumeran a continuación.

- Posicionar el mando hacia la derecha, generará un movimiento en sentido horario del Brazo Giratorio.
- Llevar el mando hacia la izquierda, indicará una voluntad de movimiento para el actuador en el sentido anti-horario.
- Mover el mando hacia la posición neutral en pleno movimiento del Brazo Giratorio, emitirá una orden de parada inmediata.
- **Luego de comandar una orden de movimiento, el mando del dispositivo debe ser llevado a la posición neutral.** Las órdenes deben ser emitidas una a una, cuidadosamente en su transición.
- Presionar uno de los botones periféricos efectuará un evento de interacción con los objetos disponibles. Si el Brazo está en posición de carga, y un objeto está disponible, entonces se activa la Válvula de Succión y se transportará el mismo. En la posición de

descarga, se desactiva la Válvula y el objeto es liberado. Entonces otro será llevado a la bandeja para ser recogido, siempre y cuando haya estado antes disponible en el cilindro de carga. En cualquier otro caso, una orden desde el botón no tiene efecto sobre el funcionamiento del sistema.

- Efectuar una presión sobre el botón central del mando del Falcon, cerrará el programa de configuración del dispositivo en cualquier momento.

En este momento, si es requerido, puede ahora recurrir a la HMI, programada en WinCC. Para ello, hay que abrir el proyecto TERCERO, darle orden de inicio y esperar que se estabilice la pantalla de monitorización, alrededor de unos 5 segundos a partir del momento en que la misma aparece. La Figura B.1.7 presenta el caso en el que el estado inicial del actuador es la posición de descarga. Las etiquetas de información muestran texto referente a la condición representada, cuando no están activas igualmente se visualizan pero sin contenido y con su fondo en blanco.

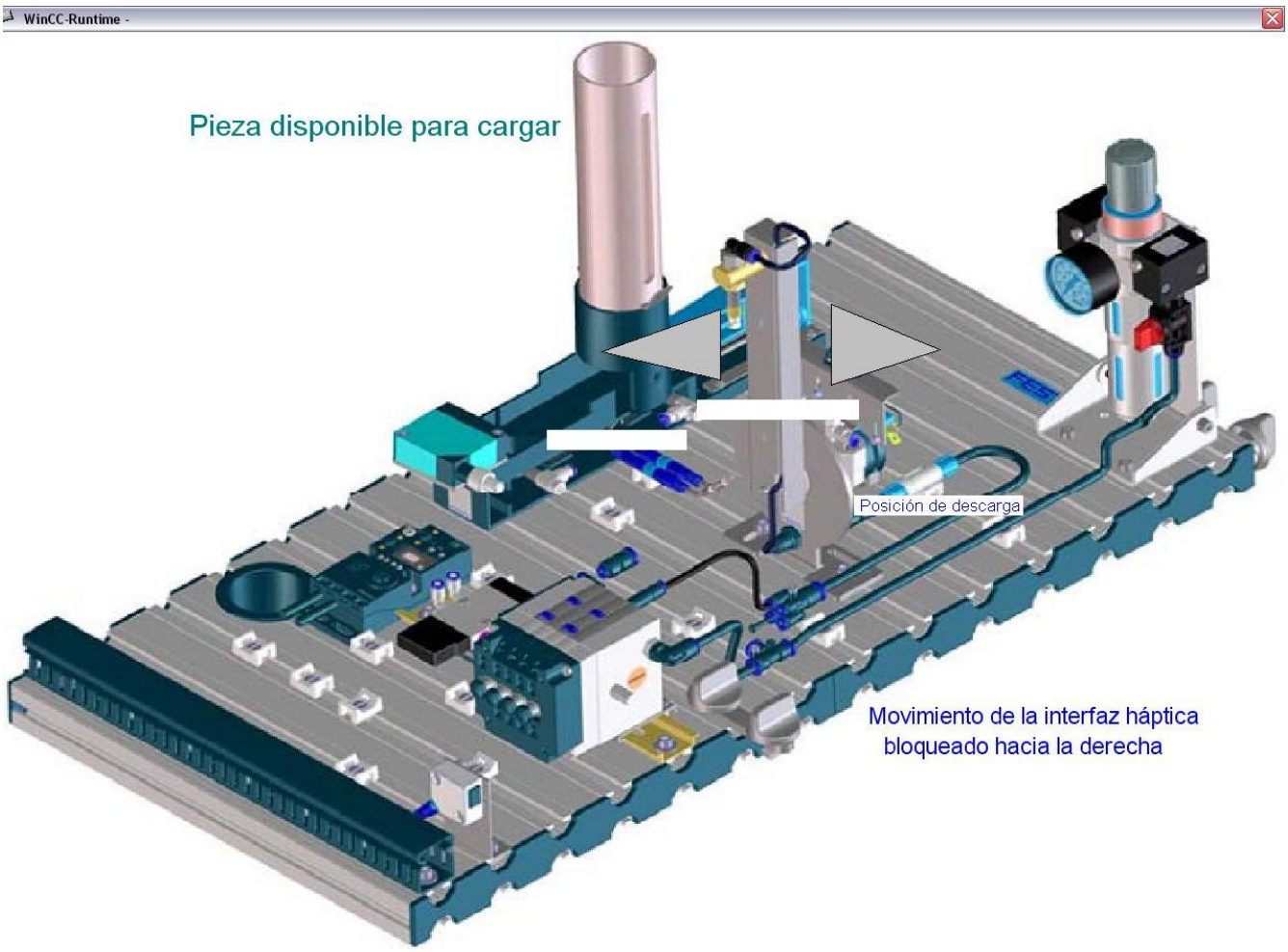

Figura B.1.7. Ejemplo del primer evento visualizado.

Una vez se ha abierto la HMI, se produce la visualización del estado del equipo comandado desde la interfaz háptica. La Figura B.1.7 muestra cómo se proporciona información acerca de la posición del Brazo Giratorio y otros datos complementarios. Suponiendo que se desea desplazar el actuador ahora a la izquierda, luego de emitir la orden se observaría lo que presenta la Figura B.1.8. Nótese el indicador de emisión de orden de movimiento, dispuesto al lado del Módulo de Cambio, resaltado con un color verde que indica que se encuentra activo.

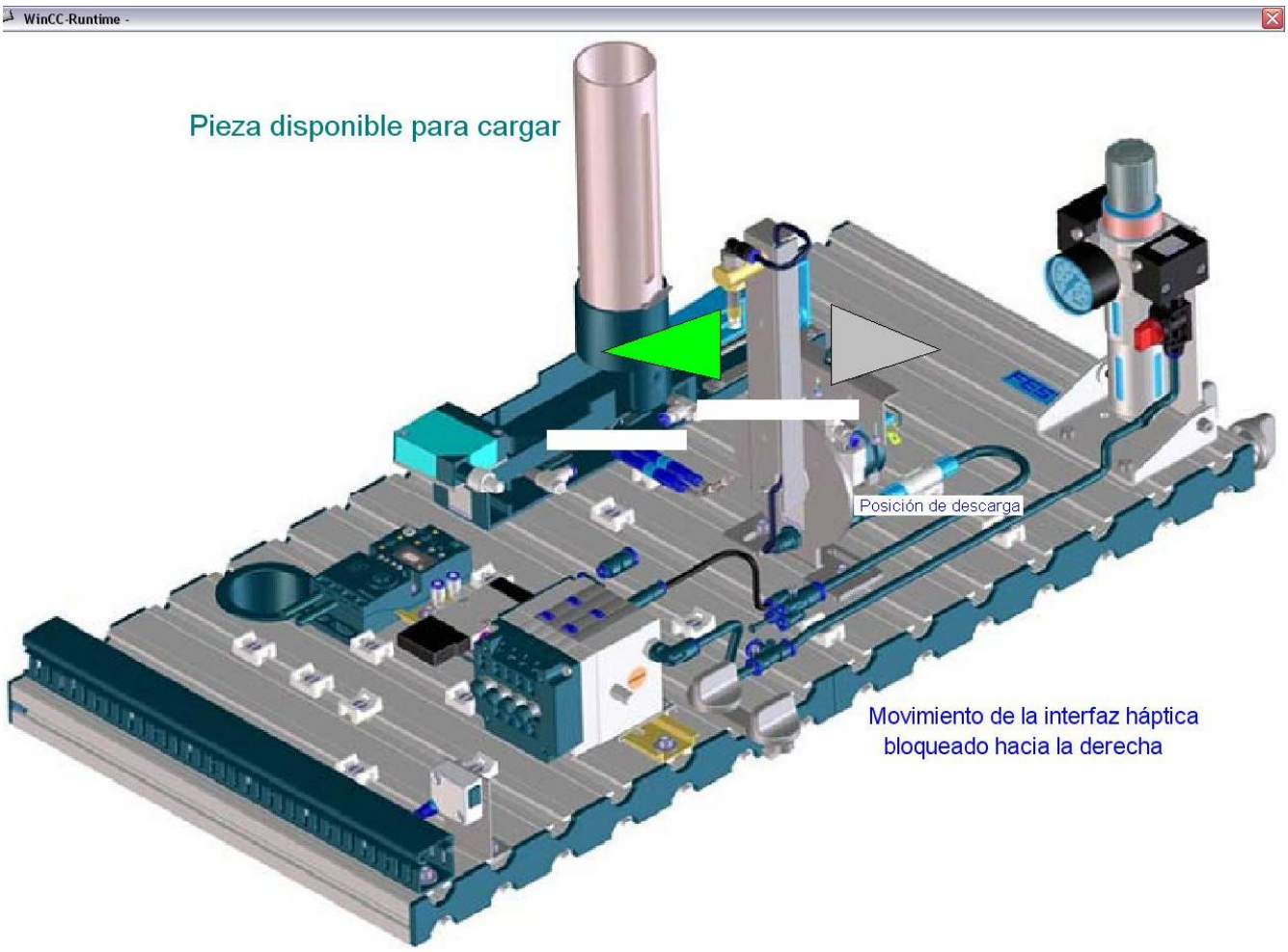

Figura B.1.8. Indicación de orden de movimiento hacia la izquierda.

Una vez se ha desplazado el Brazo Giratorio desde una posición extrema (carga o descarga), y este queda en una posición intermedia al emitir una orden de parada (mando en posición neutral), se observará lo que presenta la Figura B.1.9.

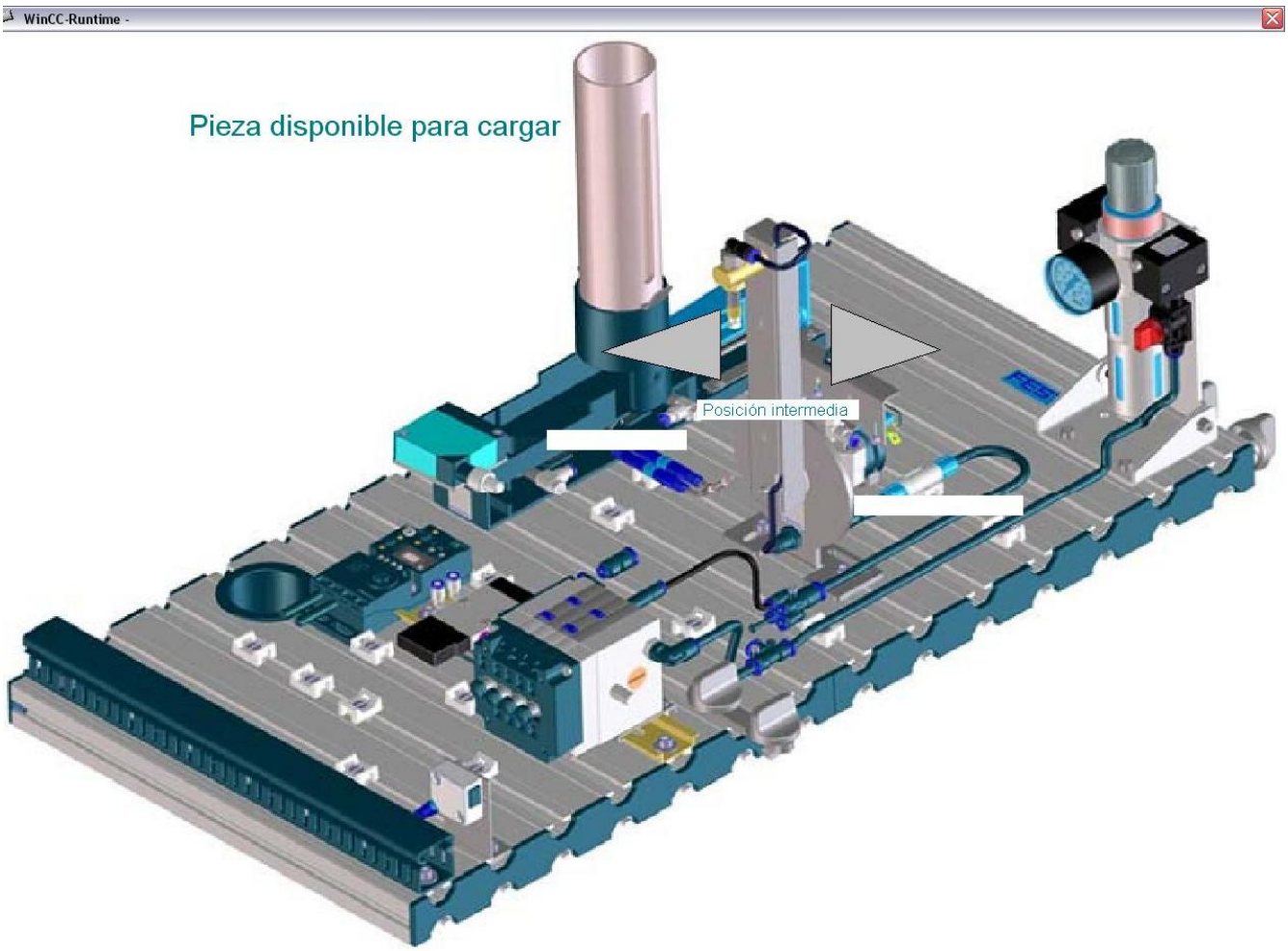

Figura B.1.9. Indicación del actuador en una posición intermedia.

Ahora, si se lleva el Brazo Giratorio hasta la posición de carga, la información suministrada al operador se muestra en la Figura B.1.10.

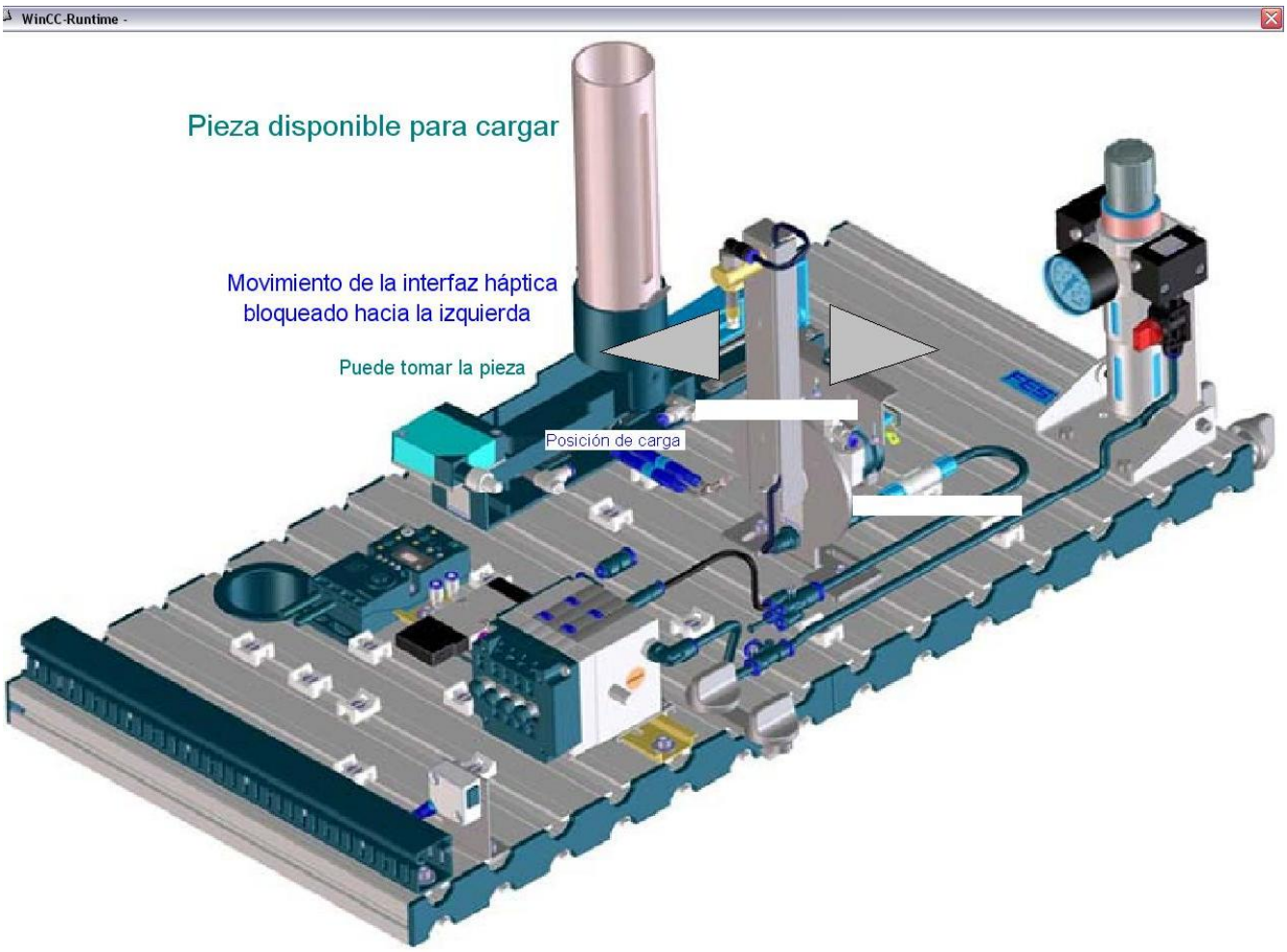

Figura B.1.10. Visualización con el Brazo Giratorio en posición de cargar un objeto.

Ahora se puede cargar un objeto. Esto se hace presionando uno de los botones periféricos del mando de la interfaz. Hecho esto se observará lo mostrado en la Figura B.1.11.

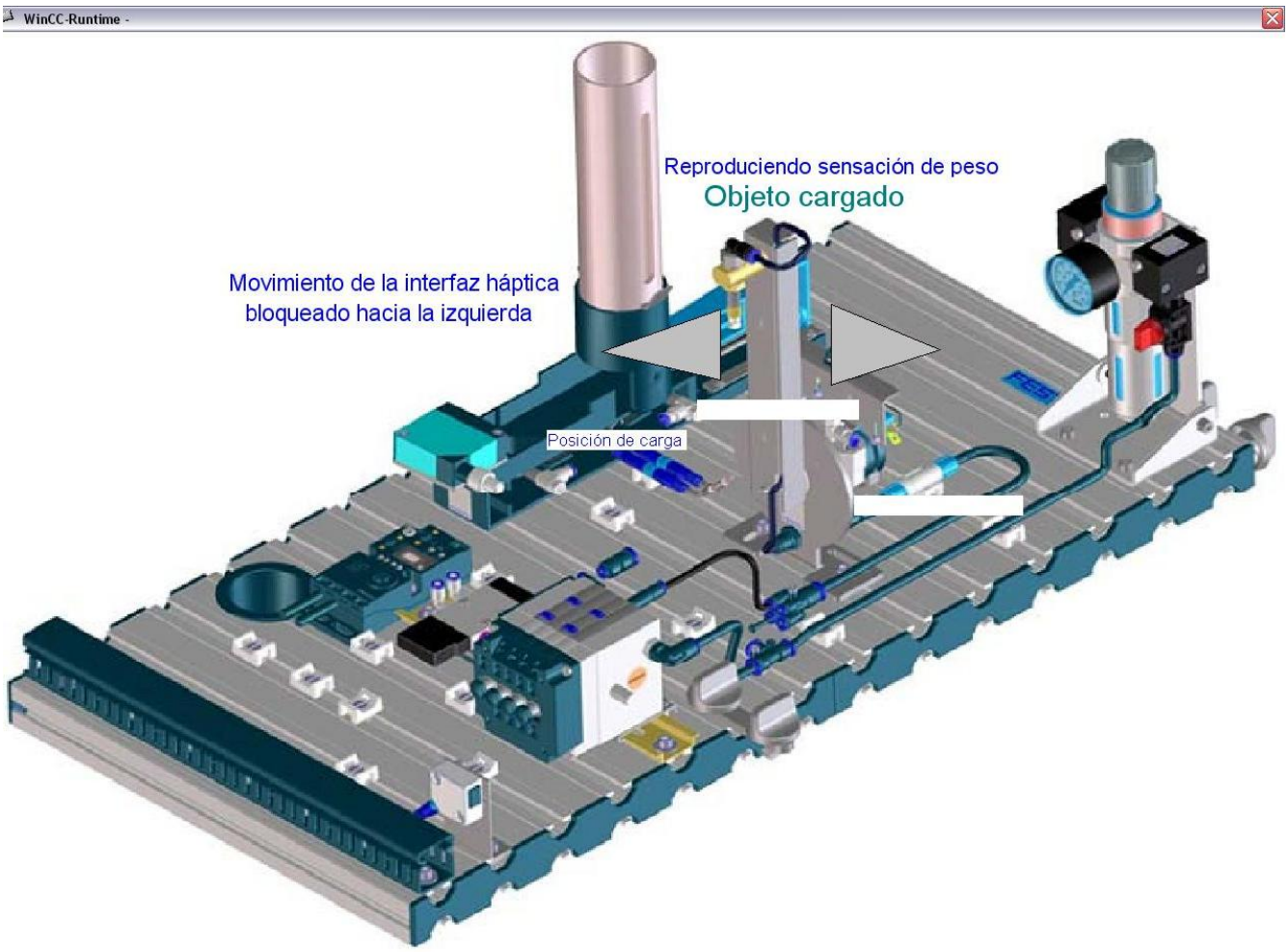

Figura B.1.11. Visualización con un objeto cargado.

Llevar el Brazo Giratorio a una posición diferente ahora, será más difícil puesto que el peso del objeto cargado genera un efecto perceptible en la interfaz háptica. Si se lleva ahora el actuador a la posición de descarga, se observa lo mostrado en la Figura B.1.12.

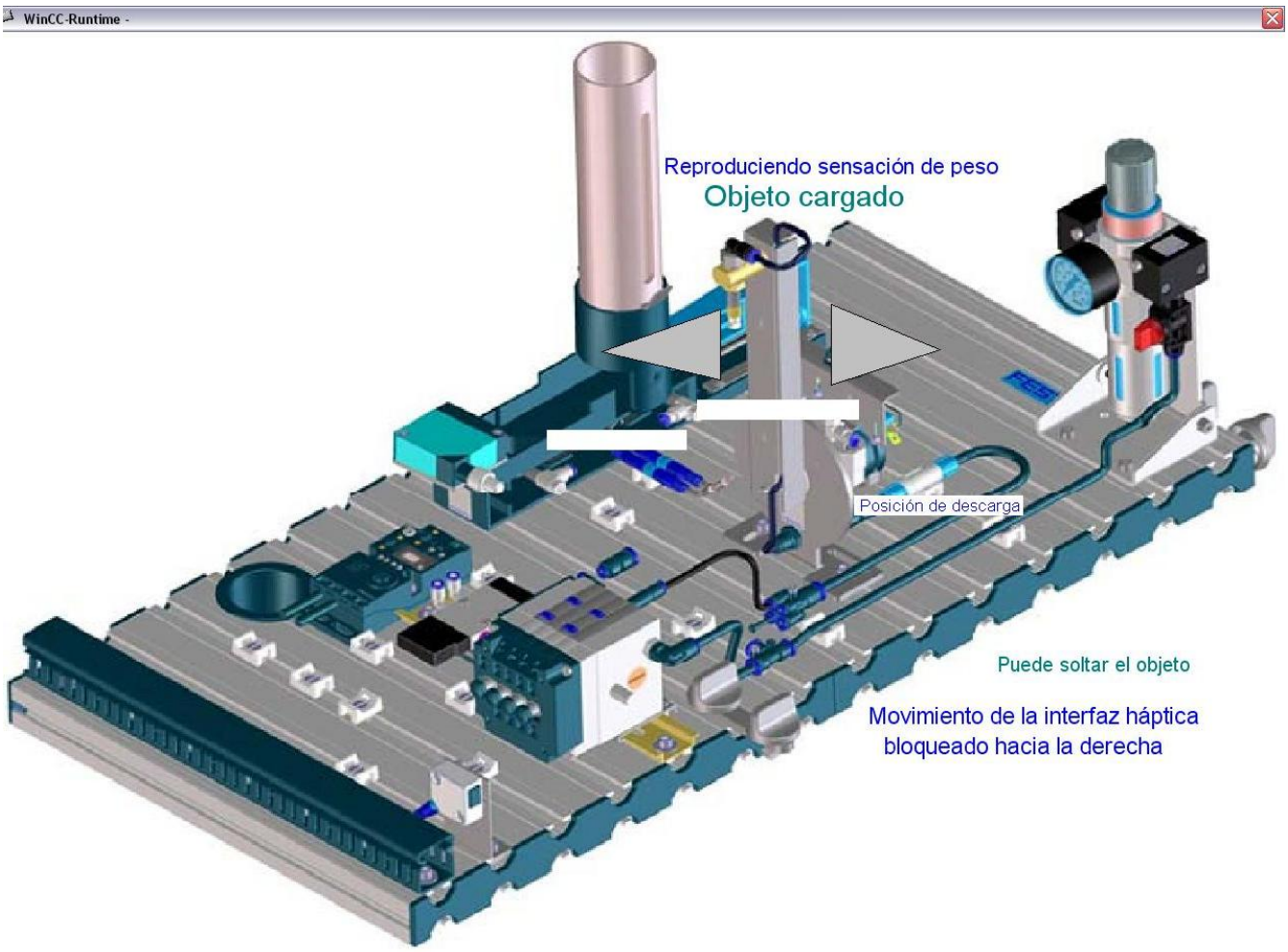

Figura B.1.12. Estado del sistema con posibilidad de soltar el objeto.

Si ahora se desea liberar el objeto, en el estado presentado en la Figura B.1.12 el sistema lo permite. Uno de los botones periféricos debe ser presionado para lograrlo. Una vez hecho esto, se volverá exactamente al mismo punto de arranque (para el caso de esta guia), introducido antes por medio de la Figura B.1.7.

Luego de manipular el sistema, debe cerrarse el programa que administra las funciones del Falcon para que este quede entonces correctamente desinicializado. Esto se hace pulsando el botón central del mando de la interfaz háptica. Una vez hecho esto, en la ventana de información de este programa, aparecerá lo que se muestra en la Figura B.1.13.

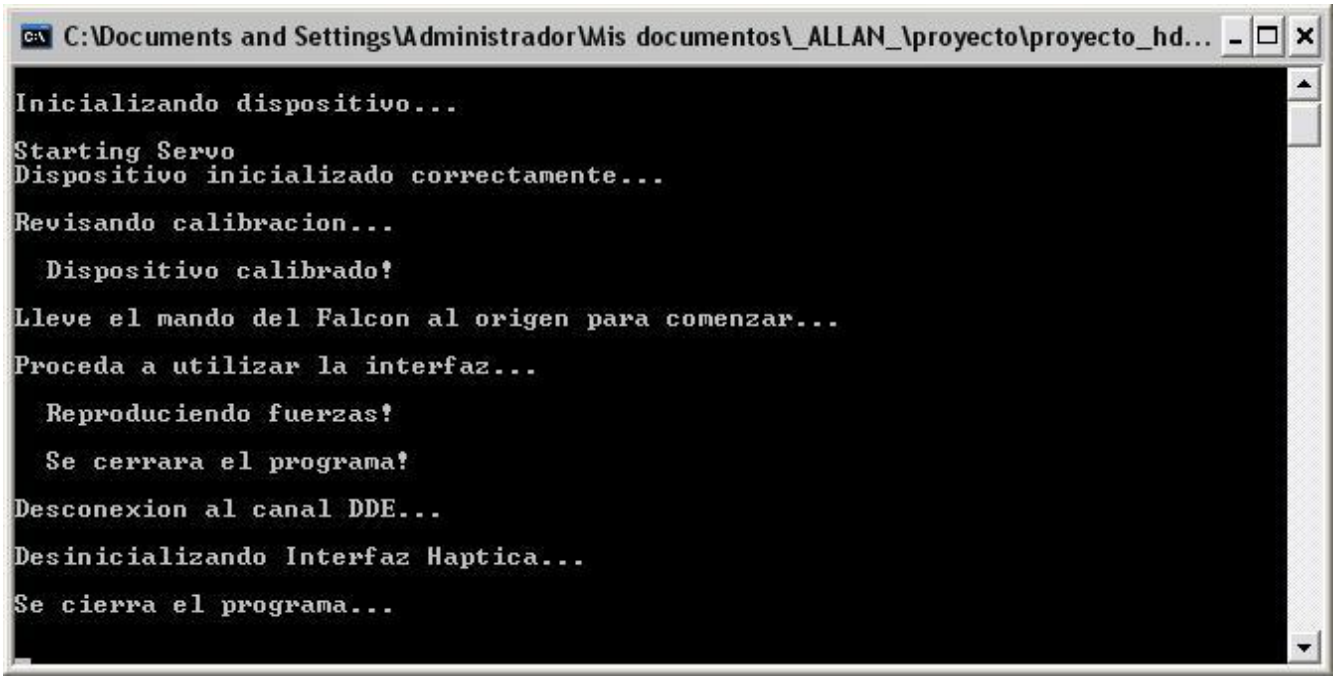

Figura B.1.13. Cierre del programa.

Luego de que el programa de configuración del dispositivo háptico y el canal de intercambio de datos se ha cerrado, debe igualmente detenerse el proceso de la HMI, desde la ventana principal de WinCC. Además, debe terminarse la ejecución del programa servidor de datos, IPC Data Server.