INSTITUTO TECNOLÓGICO DE COSTA RICA

ESCUELA DE INGENIERÍA ELECTRÓNICA

Proyecto:

Sistema teleoperado de un brazo robótico Mitsubishi RV-2AJ por medio de una Interfaz Háptica con retroalimentación de fuerza

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

Institución

Instituto Tecnológico de Costa Rica

Edgar Fonseca Gentilini

200440541

3 de mayo del 2011

INSTITUTO TECNOLÓGICO DE COSTA RICA
ESCUELA DE INGENIERÍA ELECTRÓNICA
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

Dr. Alfonso Chacón Rodríguez
Profesor lector

Ing. Arys I. 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, 3 de mayo del 2011

Declaratoria de autenticidad

Declaro que el presente Proyecto de Graduación ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos propios.

En los casos en que he utilizado bibliografía, he procedido a indicar las fuentes mediante las respectivas citas bibliográficas.

En consecuencia, asumo la responsabilidad total por el trabajo de graduación realizado y por el contenido del correspondiente informe final.

Cartago, 3 de mayo del 2011

Gentilini

Edgar Fonseca Gentilini

7 - 0169 - 0407

Resumen

La tecnología háptica es aquella que a través de la retroalimentación táctil y por medio de sistemas electrónicos le permite al usuario percibir ciertas características en los objetos como los son el peso, forma, textura o vibración entre otras.

La ausencia de un control para manejar la estación de robot FESTO que contiene un brazo robótico de cinco grados de libertad y una pinza, hace de la interfaz háptica Novint Falcon un excelente dispositivo para realizar esta tarea.

El sistema como tal se pudo llevar a cabo gracias a un programa hecho en C++ que corre en la computadora y que se encarga de controlar la interfaz háptica, y por medio de la comunicación serial RS-232 puede intercambiar información con el brazo robótico cuyo lenguaje de programación es el Melfa Basic IV y que a través del software CIROS de la empresa FESTO permitió tener un ambiente de prueba y así simular los movimientos que realiza el brazo.

Se realizó el análisis de cinemática directa del brazo robótico con el fin de obtener la posición y orientación de la pinza de acuerdo a un sistema de referencia colocado en la base del mismo. Para esto se empleo el algoritmo de Denavit-Hartenberg.

Finalmente se logro accionar los cinco grados de libertad del brazo por medio de una interfaz háptica de tres, además se permitió abrir y cerrar la pinza del brazo de manera que se retroalimente una sensación de fuerza al mover algún objeto con la misma.

Palabras claves: Interfaz háptica, SDK, C++, brazo robótico, grados de libertad, FESTO, CIROS, Melfa Basic IV, RS-232, cinemática directa, matriz de transformación, Denavit-Hartenberg.

Summary

Haptic technology is one that through tactile feedback and by electronic systems allows the user to perceive certain features on objects as are the weight, shape, texture or vibration among others.

The absence of a control to handle the robot FESTO station containing a robotic arm of five degrees of freedom and a gripper, makes the Novint Falcon haptic interface an excellent device to do this task.

The system could be implemented thanks to a program made in C++ running on the computer and is responsible for controlling the haptic interface and through serial communication RS-232 can exchange information with the robotic arm whose language programming is the Melfa Basic IV and with FESTO'S software called CIROS allowed have a test environment and simulate the robotic arm movements.

The forward kinematics analysis was developed for a robotic arm with the aim to get the position and orientation of the gripper according to a reference system placed at the base of the robot. To do this the Denavit-Hartenberg algorithm was used.

Finally, was achieved handle the five degrees of freedom of the arm using a haptic interface of three and also was allowed open and close the robot's gripper so as to feed back a sense of strength at the moment to move an object with it.

Keywords: Haptic Interface, SDK, C++, robotic arm, degrees of freedom, FESTO, CIROS, Melfa Basic, RS-232, forward kinematics, transformation matrix, Denavit-Hartenberg**.**

Dedicatoria

A mi querida madre, quien siempre ha estado al lado mío incondicionalmente y ha sido el motor en todo momento de mi vida.

Agradecimiento

Agradezco a Dios por todo lo que me ha dado, por permitirme concluir de manera exitosa mis estudios, por darme el entendimiento y el conocimiento para la realización de este proyecto además de serenidad cuando más la necesitaba.

Agradezco inmensamente a mi madre, ya que a ella le debo lo que soy y lo que he logrado.

A mis profesores del TEC, desde el inicio de la carrera y especialmente a la profesora Ing. Arys I. Carrasquilla Batista quien confío en mí para llevar a cabo este proyecto.

Finalmente a mis amigos quienes se convirtieron en una segunda familia y estuvieron en las buenas y en las malas siendo un gran soporte durante todo este tiempo.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1: Introducción

El presente documento tiene como fin explicar el desarrollo y las etapas que hicieron posible implementar el proyecto llamado “Sistema teleoperado de un brazo robótico Mitsubishi RV-2AJ por medio de una Interfaz Háptica con retroalimentación de fuerza”.

1.1 Entorno del proyecto

La escuela de Ingeniera en Electrónica del Instituto Tecnológico de Costa Rica, a través de un adecuado balance teórico-práctico imparte cursos en los siguientes campos de aplicación:

- Sistemas digitales.
- Comunicaciones eléctricas.
- Electrónica de potencia.
- Control automático.
- Telemática. [1]

El Laboratorio de Investigación en Robótica y Automatización (LIRA) de la Escuela, es liderado por la Ing. Arys Carrasquilla Batista y cuenta con proyectos académicos que repercuten directamente en el desarrollo del perfil de formación de los estudiantes, para que estos puedan optar por el grado de licenciatura en Ingeniería Electrónica. Como parte de esos proyectos, el LIRA posee una interfaz Háptica capaz de moverse en 3 dimensiones (3D) y una estación de robot con cinco grados de libertad (5GD).

La tecnología háptica es aquella que a través de la retroalimentación táctil y por medio de sistemas electrónicos, le permite al usuario percibir ciertas características en los objetos como los son el peso, forma, textura o vibración entre otras. Actualmente la tecnología háptica se aplica en una variedad de campos tales como:

- La teleoperación, que consiste en manejar un dispositivo electrónico de forma remota, como por ejemplo pilotear un avión de manera que se le permita al operador percibir la fuerza de resistencia al mover la palanca de mando.
- En las tecnologías móviles, por ejemplo la vibración.
- En la medicina para simular ambientes para el entrenamiento de cirujanos y simulación de cirugías con retroalimentación de fuerzas.
- En la robótica que permite el diseño de robots con la capacidad del tacto, que puedan sentir la presión al sujetar algún objeto con su mano.
- En los videos juegos, como por ejemplo los controles que vibran en respuesta a diversas situaciones que ocurren durante el mismo.

La empresa Tecnologías Novint Inc, es pionera en la tecnología 3D táctil en el área de la informática. A partir de esto nace The Novint Falcon que es el primer dispositivo de juego en su clase capaz de permitir y experimentar en 3D la realidad y fuerza en juegos especializados al tacto [2].

La compañía FESTO es una empresa líder mundial en soluciones para la formación profesional y orientada a la industria, especializada en la automatización de fábricas y procesos. Está empresa ha desarrollado una estación de robot (calidad industrial), con las siguientes características [3]:

- Posee 5 grados de libertad.
- Retira piezas de un receptáculo.
- Deposita las piezas en el retenedor de montaje.
- Distingue las piezas por su color (negra/no negra).
- Pasa las piezas a la estación siguiente.

1.2 Aspectos generales del problema

Como se mencionó anteriormente, una interfaz háptica es aquella que es capaz de darle a conocer al ser humano ciertas características como textura, peso

y forma a través de un dispositivo electrónico. En este caso, se cuenta con la unidad del fabricante de Novint, llamada Novint Falcon, que se puede ver en la figura 3.16.

A pesar de que el Falcon es un dispositivo hecho para juegos de computadora, posee la cualidad de que su fabricante brinda a los usuarios o compradores un SDK (software development kit), además brinda importante documentación (materiales impresos, información en línea y documentación electrónica), así como también código fuente y código objeto, para que el programador pueda realizar modificaciones y adaptarlas a las necesidades de la aplicación requerida.

El Robot a operar es el RV-2AJ y se puede ver en la figura 3.4. Tiene un controlador CR1-571 (ver figura 3.5) que es el cerebro del brazo y que se encarga de enviar las señales de control a los codificadores de los servomotores para garantizar fiabilidad en los movimientos.

1.3 Requerimientos del proyecto

En este proyecto se desea que al maniobrar la interfaz háptica el brazo robótico responda al desplazamiento aplicado a la misma. El accionar de la estación se logra a través del robot angular de brazo articulado vertical, que posee cinco ejes que le permiten moverse en 3D.

Dado que el robot posee una pinza, también se quiere que al sujetar algún objeto con ésta se experimente una fuerza de retroalimentación que la haga más difícil de accionar, simulando de esta forma la carga que el robot esté moviendo.

Es de estas ideas que nace la interrogante del proyecto: ¿Es posible operar un brazo robótico por medio de una interfaz háptica y retroalimentar fuerza?

Esta pregunta marcó el inicio del proyecto, en el cual para su realización se necesitó investigar temas relacionados con programación, automatización, mecatrónica, teleoperación y robótica entre otros.

1.4 Solución seleccionada

El problema a resolver se separó en tres etapas, las cuales son las siguientes y en la figura 1.1 se ve un diagrama de bloques de la solución seleccionada:

- Programación de la interfaz háptica.
- Programación de la estación de robot.
- Acoplar el brazo robótico con la interfaz háptica.

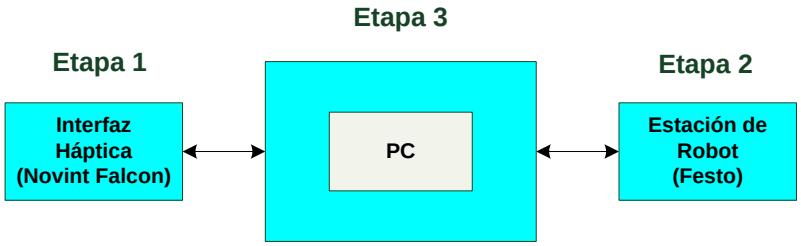


Figura 1.1. Diagrama de bloques de la solución.

Es importante destacar algunos aspectos que la solución como tal debe incluir:

- Retroalimentar fuerza.
- Comunicación serial entre la PC y el controlador del brazo robótico.
- Usar los 5 GD del brazo a pesar de que la interfaz solo posea 3 GD.
- Abrir y cerrar la pinza del brazo por medio de la interfaz.

Capítulo 2: Meta y objetivos

En este capítulo se presentan la meta y los objetivos que se desean alcanzar con la realización de este proyecto, cuyo cumplimiento será evaluado al final del mismo.

2.1 Meta

Programar y acoplar una interfaz háptica para interactuar con un brazo robótico capaz de moverse en 3D y retroalimentar la fuerza que el robot aplica para maniobrar un cuerpo.

2.2 Objetivo general

Controlar un robot por medio de una interfaz háptica para percibir la retroalimentación de la fuerza que emplea el brazo en mover un objeto en 3D.

2.3 Objetivos Específicos

- Programar las rutinas necesarias para la comunicación de la interfaz háptica a través de la computadora con la estación de robot Festo.

- Desarrollar las rutinas de programación para retroalimentar la fuerza que implica mover un objeto con el brazo robótico, a través de la activación del manubrio de la interfaz háptica.

- Programar el CPU de la estación de robot, de manera que éste se pueda mover en 3D a través de tres de sus cinco ejes, a partir de la información recibida por el puerto serial y enviar información a la PC para retroalimentar fuerza en la interfaz.

Capítulo 3: Marco teórico

El presente capitulo tiene como fin explicar las áreas que abarca y en que se fundamenta el proyecto. Entre los aspectos a mencionar se encuentran características técnicas del equipo, del software y conceptos entre otras cosas.

3.1 Robótica

La robótica es la ciencia y la tecnología de los robots, abarca desde el diseño, manufactura y aplicaciones de los robots. La robótica combina diversas disciplinas como son: la mecánica, la electrónica, la informática, la inteligencia artificial y la ingeniería de control. [23]

"A lo largo de la historia el hombre se ha sentido fascinado por máquinas y dispositivos capaces de imitar las funciones y los movimientos de los seres vivos. Los primeros mecanismos (los griegos le llamaban autómatos, de ahí la palabra autómata que significa máquina que imita la figura y movimientos de un ser animado) se movían a través de dispositivos hidráulicos, poleas y palancas; dichos mecanismo tenían fines meramente lúdicos". [22]

La palabra robot tiene su origen en la palabra eslava robota que se refiere al trabajo realizado de manera forzada. Por otro lado, en 1954, George Devol diseñó el primer robot programable. La evolución de los robots industriales desde sus inicios ha sido vertiginosa. Con el paso del tiempo y las investigaciones se ha logrado que los robots tomen posición en casi todas las áreas productivas y tipos de industria. En pequeñas y grandes fábricas, los robots pueden sustituir al hombre en aquellas tareas repetitivas y hostiles. Con el avance de la tecnología se apunta a aumentar la movilidad, destreza y autonomía de las decisiones de los robots. Gran cantidad de los robots se utilizan en aplicaciones industriales como ensamblado, soldadura, alimentación de máquinas herramientas, no obstante

existen otro tipo de aplicaciones que han provocado un cambio en la concepción y morfología de los robots. Algunas de esas aplicaciones no industriales son: [22]

- Robots espaciales: Brazos para lanzamientos y recuperación de satélites, vehículos de exploración lunar y robots para construcción y mantenimientos de hardware en el espacio.
- Robots para aplicaciones submarinas y subterráneas: Exploración submarina, instalación y mantenimiento de cables telefónicos submarinos, limpieza e inspección de tuberías y drenajes subterráneos.
- Robots militares: Desactivación de bombas.
- Aplicaciones médicas: Prótesis robotizas y ayudas a pacientes discapacitados.
- Aplicaciones agrícolas: Sembrada y recogida de cosechas.

Los robots se pueden clasificar de manera general, en los siguientes tipos: [24]

- Humanoides: Se busca la apariencia humana, trata de imitar su comportamiento, por ejemplo: ASIMO (ver figura 3.1 a), AIBO.
- Móviles: Se montan sobre una plataforma móvil.
- Industriales: Manipulador diseñado para mover materiales, herramientas, por ejemplo: SCARA, PUMA, STAÜBLI (ver figura 3.1 b.), RV-2AJ.
- Inteligentes: Capaces trabajar en entornos no estructurados y con eventos impredecibles. Utiliza sensores para conocer el entorno (capacidad de aprendizaje).
- Servicios: Trabajan con total o parcial autonomía para desarrollar servicios útiles (limpieza).

a) 

b) 

Figura 3.1 Robot ASIMO de Honda [28] y brazo robótico Stäubli [27].

En el presente proyecto se trabaja con un robot industrial. Según la AFNOR (Asociación Francesa de Normalización), el robot industrial se define como: "manipulador automático servo-controlado, reprogramable, capaz de posicionar y orientar piezas, útiles o dispositivos especiales, siguiendo trayectoria variables reprogramables, para la ejecución de tareas variadas. Normalmente tiene la forma de uno o varios brazos terminados en una muñeca. Su unidad de control incluye un dispositivo de memoria y ocasionalmente de percepción del entorno. Normalmente su uso es el de realizar una tarea de manera cíclica". [21]

En el caso de los robots industriales, una manera de clasificarlos es la que hace la AFRI (Asociación Francesa de Robótica industrial) en la que distingue cuatro tipos. [22]

- Tipo A: Manipulador con control manual o telemando.
- Tipo B: Manipulador automático con ciclos preajustados, regulación mediante fines de carrera o topes, control por PLC, accionamiento neumático, eléctrico o hidráulico.
- Tipo C: Robot programable con trayectoria continua o punto a punto. Carece de conocimiento de su entorno.

- Tipo D: Robot capas de adquirir datos de su entorno, readaptando su tarea en función de éstos.

3.2 Cinemática del Robot [22]

La cinemática del robot estudia el movimiento del mismo con respecto a un sistema de referencia. Se interesa por la descripción analítica del movimiento espacial del robot como una función del tiempo y en particular por las relaciones entre la posición y la orientación del extremo final del robot con los valores que toman sus coordenadas articulares.

En la cinemática del robot existen dos problemas fundamentales (ver figura 3.2). El primero es la cinemática directa que consiste en determinar cuál es la posición y orientación del extremo final del robot y el segundo es la cinemática inversa que resuelve la configuración que debe de adoptar el robot para una posición y orientación del extremo conocidas.

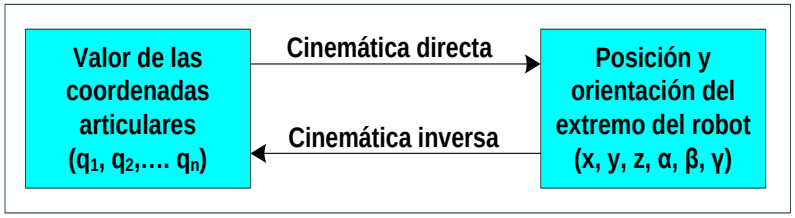


Figura 3.2 Relación entre cinemática directa e inversa.

En 1955 Denavit y Hartenberg propusieron un método sistemático para describir y representar la geometría espacial de los elementos de una cadena cinemática (de un robot) con respecto a un sistema de referencia fijo. Este método utiliza una matriz de transformación homogénea para describir la relación espacial entre dos elementos rígidos adyacentes, reduciendo el problema cinemática directo a encontrar una matriz de transformación homogénea 4x4 que relacione la localización espacial del extremo del robot, con respecto a un sistema de coordenadas en su base. Esta matriz será en función de las coordenadas articulares.

Para definir qué es una matriz de transformación homogénea es importante indicar que la representación mediante coordenadas homogéneas de la localización de sólidos en un espacio n-dimensional se realiza a través de coordenadas de un espacio (n+1)-dimensional. Es decir, un espacio n-dimensional se encuentra representado en coordenadas homogéneas por (n+1) dimensiones, de tal forma que un vector **p**(*x, y, z*) vendrá representado por **p**(w*x, wy, wz, w*), donde *w* tiene un valor arbitrario y representa un factor de escala, por ejemplo: El vector 2**i** +3**j** + 4**k** se puede representar en coordenadas homogéneas como [4, 6, 8, 2]^T. Ahora, la matriz de transformación homogénea T es una matriz de 4x4 que sirve para transformar un vector expresado en coordenadas homogéneas con respecto a un sistema O'UVW, a su expresión en las coordenadas del sistema de referencia OXYZ. También se puede utilizar para girar y rotar un vector referido a un sistema de referencia fijo, y en definitiva sirve para expresar la orientación y posición de un sistema de referencia O'UVW con respecto a otro fijo OXYZ.

La matriz **T** de transformación se suele escribir de la siguiente forma y más adelante se explica cómo se interpreta:

$$T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{n} & \boldsymbol{o} & \boldsymbol{o} & \boldsymbol{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

3.3 Estación de robot Festo

El brazo robótico Mitsubishi a utilizar en este proyecto viene incorporado en una mesa de trabajo que cuenta con otros accesorios (sensores, entradas y salidas digitales) que en este proyecto no se van a emplear, es por esto que no se mencionan, no obstante la estación de robot adquirida por el LIRA se puede ver en la figura 3.3.

Figura 3.3 Estación de robot del proyecto. [3]

3.3.1 Brazo robótico Mitsubishi RV-2AJ

El brazo robótico RV-2AJ de Mitsubishi se puede ver en la figura 3.4. Se trata de un robot angular de brazo articulado vertical con 5 GD que se utiliza para el transporte de piezas. El diseño del RV-2AJ lo hace ideal para aplicaciones donde no sobra el espacio y con movimiento de cargas de hasta 2 Kg de peso. Este robot tiene un alcance de 410mm, y combina una velocidad máxima de 2,100mm/s con

una precisión de ±0.02mm. Los servomotores de corriente alterna, unidos a codificadores de posición absolutos, garantizan fiabilidad y bajo mantenimiento. Estos codificadores permiten apagar el robot en cualquier momento y que al conectar de nuevo la alimentación, el robot podrá continuar trabajando desde la posición actual. El brazo tiene integradas en su interior una serie de conductos que permiten la conexión de pinzas y ventosas neumáticas. También tiene integrado un conector para cuatro sensores, y la posibilidad de utilizar pinzas de accionamiento eléctrico. En la tabla 3.1 se puede ver un resumen de las características del brazo robótico.

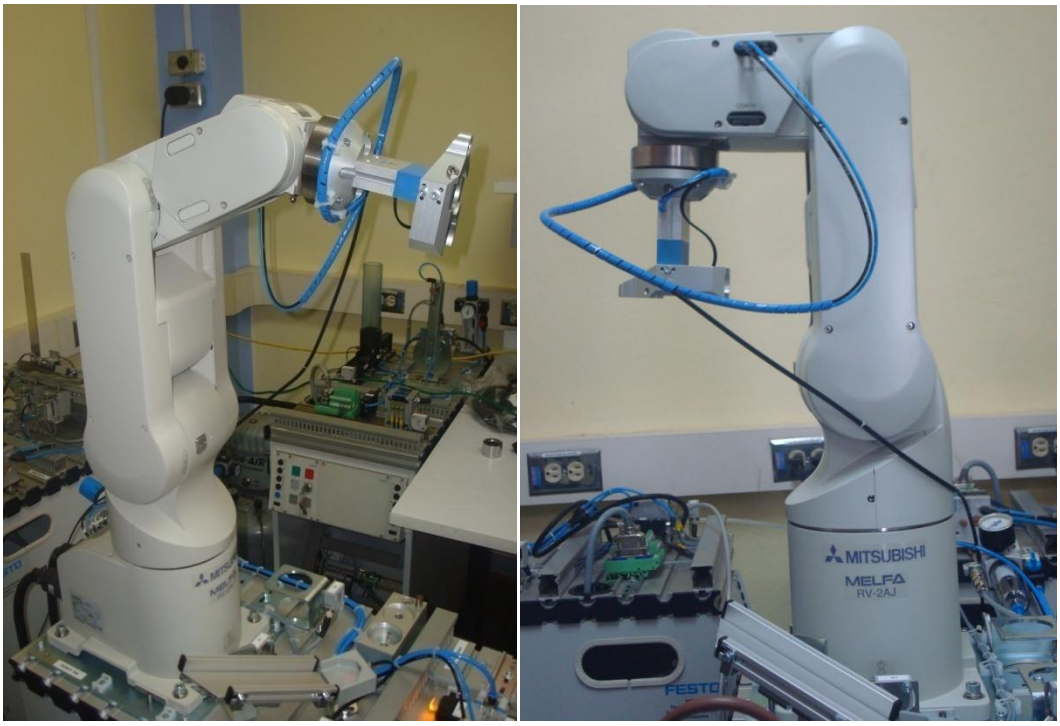


Figura 3.4 Robot Mitusbishi RV-2AJ.

Tabla 3.1 Características del brazo robótico RV-2AJ de Mitsubishi.

Grados de libertad		5
Motores		Servomotores AC (Ejes J1, J3 y J5 con freno)
Detección de posición		Codificadores absolutos
Máxima Carga (kg)		2
Longitud del brazo (mm)		250
Alcance radial máximo (mm)		410
Limite (grados)	J1	± 150
	J2	180 (-60 a +120)
	J3	230 (-110 a +120)
	J4	-
	J5	± 90
	J6	± 200
Velocidad máxima (grados/s)	J1	180
	J2	90
	J3	135
	J4	-
	J5	180
	J6	210
Velocidad máxima (mm/s)		2100
Repetitividad (mm)		±0,02
Peso (kg)		17
Instalación		Suelo, techo o base FESTO
Controlador del robot		CR1-571

Este brazo cuenta con un controlador CR1-571 que se puede ver en la figura 3.5. Este controlador es el cerebro del robot y es donde radica el sistema de control del Robot. El CR1-571 se basa en la misma arquitectura que utilizan los robots de mayores dimensiones de la marca, trabajando con las mismas posibilidades y lo más importante, el mismo lenguaje de programación. El corazón del controlador del robot es un CPU - RISC de 64 bits que permite la ejecución en paralelo de hasta 32 programas en modo multitarea. Un aspecto muy importante es el puerto RS-232 que se utiliza para descargar los programas hechos en la computadora y también se utiliza para comunicar el robot con dispositivos externos. Cabe destacar que por medio de un adaptador se puede comunicar el robot mediante red Ethernet (Protocolo TCP/IP), o bien con una red CC-Link de Mitsubishi que permite el intercambio rápido de datos, sobre todo entre el robot y

un PLC. Otro punto importante del controlador es que permite almacenar los programas en memoria, de esta forma se pueden guardar 88 programas de 5000 líneas cada uno y también se puede guardar 2500 posiciones. En la tabla 3.2 se puede ver un resumen de las características del controlador.

Figura 3.5 Controlador CR1-571.

Tabla 3.2 Características del controlador CR1-571.

Número de ejes controlables		6
Tipo de procesador		CPU principal: 64 bit RISC
Capacidad de memoria	# Posiciones	2500
	# Programas	88 de 5000 líneas máximo cada uno
Lenguaje de programación		MELFA-BASIC IV ó MOVEMASTER COMMAND6
Entradas/salidas		16/16, pero se puede ampliar a 240/240
Parada de emergencia		1
Conexión RS-232		1 para conexión a PC
Conexión RS-422		1 para Teaching Box
Interfaces de extensión		3
Pinza		4/0 (con opciones: 4/4)
Tensión de alimentación		Monofásica 90–132 V AC; 50/60 Hz; 0,7 kVA
		Monofásica 180–253 V AC; 50/60 Hz; 0,7 kVA
Temperatura ambiente		0 hasta 40 °C
Humedad ambiente		45–85%sin condensación
Montaje		En el suelo
Dimensiones		(212 x 166 x 290)mm
Peso		8 kg

Adicionalmente el brazo cuenta con una botonera que se puede ver en la figura 3.6. Cuenta con un display LC con 4 líneas x16 dígitos (con iluminación de fondo), se conecta al controlador via RS-422. La botonera o TeachBox se utiliza para determinar y grabar posiciones del brazo del robot para ayudar en la programación, desde borrar un programa o hacerlo nuevo y modificar uno existente. También permite mover manualmente cada articulación del brazo, abrir y cerrar las pinzas entre otras cosas por lo que realmente representa una gran herramienta para la persona que esté programando u operando el robot.

Figura 3.6 T/B botonera o TeachBox (R28TB).

3.3.2 Melfa Basic IV

Melfa Basic IV fue el lenguaje que se utilizó para la elaboración del programa que corre en el controlador para este proyecto, no obstante el CR1-571 también soporta el lenguaje Movemaster Command. El potente lenguaje de programación MELFA-BASIC IV sirve para controlar los robots y está basado en el lenguaje BASIC estándar. Este lenguaje de programación permite la creación de programas altamente complejos y además no sólo controla secuencias sencillas de movimientos, sino que ejecuta por sí mismo cálculos de extrema complejidad sin tener que estar conectado a una PC. Esto es posible gracias a una extensa

biblioteca de funciones, entre las que se cuentan también las funciones trigonométricas.

También, como cualquier otro lenguaje de programación, Melfa Basic IV permite hacer ciclos, condiciones, interrupciones, declarar variable de varios tipos, por ejemplo: de posición, de articulación, variables para hacer operaciones aritméticas o bien para manejar una cadena de caracteres.

3.3.3 CIROS

CIROS es un sistema general de simulación en 3D hecho en Alemania. Su flexibilidad lo hace idóneo para una gran variedad de áreas de aplicación además de poseer una gran variedad de opciones de equipamiento. Existen varias versiones de CIROS: CIROS® Robotics, CIROS® Mechatronics, CIROS® Production y CIROS® Studio entre otros. En el presento proyecto, se utilizo CIROS® Studio el cual representa una herramienta profesional para mediante una interfaz amigable (ver figura 3.7) llevar a cabo el modelado, la programación y la simulación de un entorno industrial. Algunas de sus características son:

- Posee todos los entornos de aprendizaje de mecatrónica, robótica y automatización industrial en una misma plataforma.
- Nueva interface de usuario orientada a la manipulación para tareas y proyectos con un nuevo asistente didáctico en línea.
- Bibliotecas con sistemas de robots industriales y numerosos componentes de automatización.
- Soporta los lenguajes Movemaster Command y Melfa Basic IV para los robots Mitsubishi.
- Interfaz de gran alcance para el control del robot de Mitsubishi a través de Ethernet TCP/IP o el puerto serie.
- Permite descargar los programas y archivos, visualizar en línea los datos del sistema del robot,
- Ejecución y depuración del programa línea por línea.

- Simulación en 3D en tiempo real: incluye simulación de transporte, de las conexiones de la tubería y las cadenas de la energía.
- Detección de colisiones a través del cambio de color o mensajes de advertencia con y sin acuse de recibo
- Simulación del sensor: desde el sensor inductivo para la cámara, casi todos los sensores y sus propiedades físicas pueden ser simuladas. De esta manera es posible analizar la interacción realista con el equipo periférico.
- Cliente OPC con el menú de configuración para la comunicación con cualquier servidor OPC.

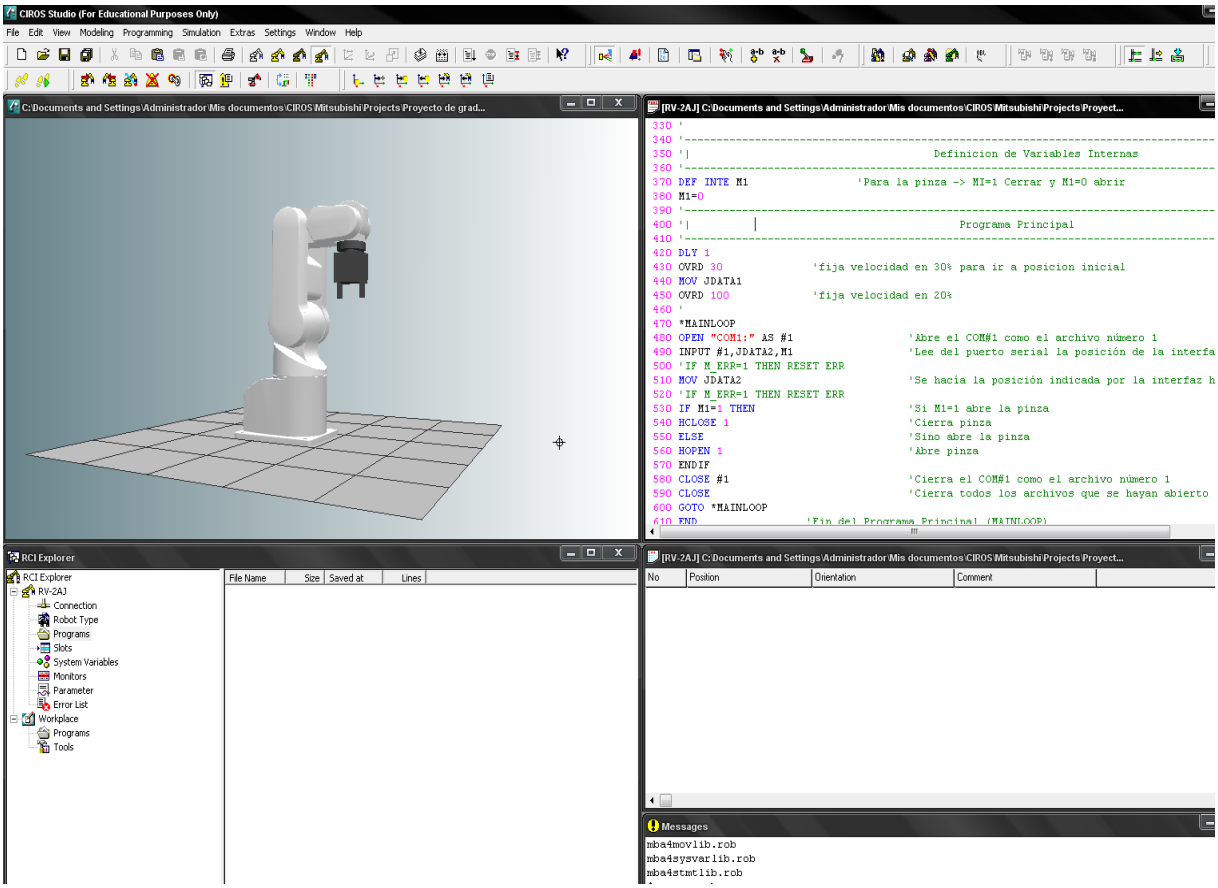


Figura 3.7 Ambiente de trabajo del software CIROS Studio.

Los requerimientos que debe de tener la PC en la que se va a instalar CIROS son los siguientes:

- Procesador de 1.0 GHz, aunque el recomendado es de 2.2 GHz.
- Sistema operativo: Windows XP Service Pack 2 ó Windows Vista.

- Memoria RAM de 512 MB, aunque se recomienda 2 GB.
- 5 GB de espacio en el disco duro, aunque se recomienda 20 GB.
- Windows XP o Vista, Microsoft Internet Explorer Versión 5.0 o superior.
- Tarjeta de video NVIDIA de 128MB, aunque se recomienda 7800GTX, 512 MB o superior.
- Monitor de 17” con una resolución de 1024 x 768 pixeles, aunque se recomienda uno de 19" con resolución de 1280 x 1024 píxeles.
- Un puerto USB para la llave (dongle) del software CIROS que hace de licencia.
- Unidad de DVD-ROM.
- 1 puerto serie.
- Adobe Acrobat Reader versión 6.0 o superior

3.4 Teleoperación y telerobótica

De acuerdo a tesis realizadas en diferentes universidades [5], [6], [7] y según Barrientos [22] “las investigaciones en torno a la telemanipulación de objetos nació en los laboratorios de la industria nuclear, esto por el alto riesgo que representa estar en contacto directo con los elementos radioactivos. En 1948 Raymond Goertz del Argonne National Laboratory desarrollo el primer telemanipulador que consistía en dispositivo mecánico maestro-esclavo. El manipulador maestro situado en la parte segura, era movido directamente por el operador, mientras que el esclavo estaba en contacto con los elementos radiactivos y unido mecánicamente al maestro, reproducía fielmente los movimientos de este. El operador sentía a través del dispositivo maestro, las fuerzas que el esclavo ejercía sobre el entorno. En 1954 Goertz empleó la tecnología electrónica y del servocontrol sustituyendo la transmisión mecánica por otra eléctrica y desarrollando así el primer telemanipulador con servocontrol bilateral. Cabe destacar que los progenitores más directos de los robots fueron los telemanipuladores (ver figura 3.8).

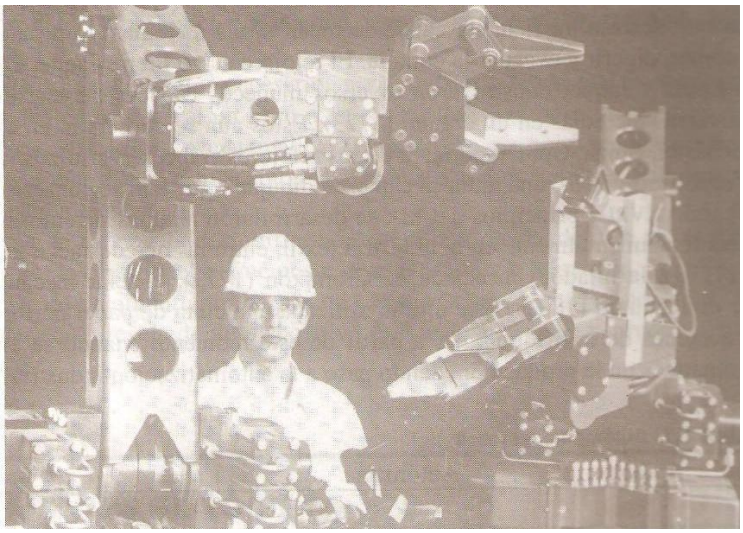


Figura 3.8 Sistema de telemanipulación bilateral. [22]

De ahí en adelante la tecnología ha ido evolucionando. En los sesenta se extendieron las investigaciones hasta el campo de aplicaciones submarinas (inclusión de cámaras y dispositivos para aumentar la telepresencia del operador). En los setenta, la tecnología de la teleoperación alcanzó su mayoría de edad con su utilización en aplicaciones espaciales. Hoy en día la teleoperación ha mejorado tecnológicamente gracias a los avances desarrollos en la informática y la robótica. [5]

La intervención del operador humano es imprescindible en un gran número de las aplicaciones de la robótica, especialmente en entornos no estructurados y dinámicos donde los problemas de percepción y planificación automática son muy complejos. En ocasiones, el operador se encuentra separado físicamente del robot, por lo que se necesita de un sistema de comunicación entre los dispositivos utilizados por el operador y el sistema de control del robot. La intervención del operador puede darse desde la teleoperación directa de los actuadores de las articulaciones, hasta la simple especificación de movimientos, o incluso de tareas, que se realizan automáticamente. [8]

Se puede definir teleoperación como la extensión de capacidades sensoriales y destreza humana a una localización remota [8], o bien; se puede

decir que es el conjunto de tecnologías que comprenden la operación o gobierno a distancia por un ser humano de un manipulador [5]. La telerobótica puede considerase como una forma evolucionada de teleoperación y se define como el conjunto de tecnologías que comprenden la monitorización y reprogramación a distancia de un robot por un ser humano. [9]

En la figura 3.9 se puede ver los bloques principales que conforman un sistema de telerobótica.

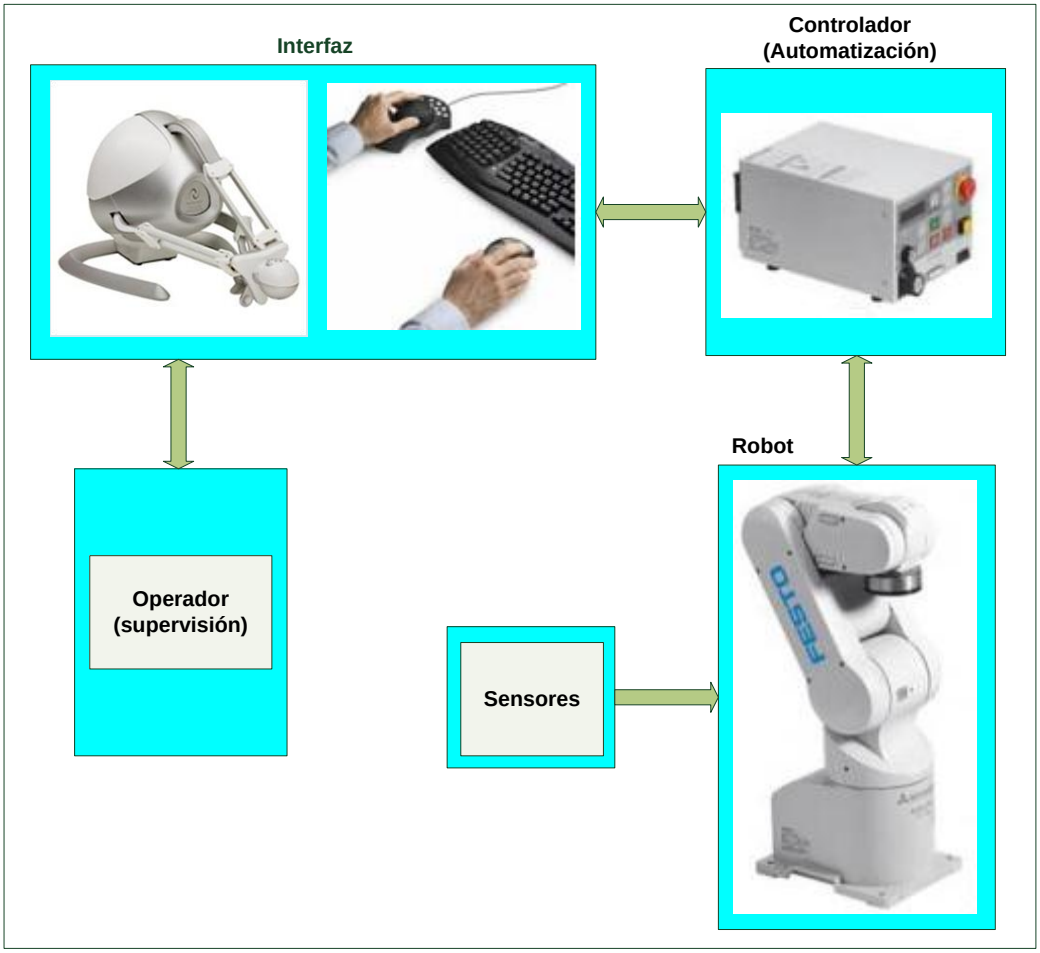


Figura 3.9 Diagrama de un sistema de telerobótica.

- Operador o teleoperador: Persona que realiza a distancia el manejo del robot. En algunos casos es realimentado con fuerzas producidas por el robot en su entorno de trabajo. [5] y [9]

- Interfaz: El operador se comunica con el robot a través de una interfaz la cual está compuesta generalmente por equipos y programas con prestaciones multimedia. [9]
- Controlador y canales de comunicación: Recibe y retorna información a la persona a través de la interfaz, se encarga de monitorear y controlar al robot. [9]
- Robot: Se encuentra en la zona remota y es manejado por el operador. [5]
- Sensores: Permiten captar información del entorno. [9]

En el caso de la teleoperación, las primeras aplicaciones fueron para el área nuclear, no obstante hoy en día la teleoperación y telerobótica tiene aplicación en una gran variedad de campos:

- Automatización: Manejo de robots en industrias.
- Medicina: Ayuda a personas con discapacidades y de la tercera edad, que por su situación física, falta de apoyo y accesibilidad sufren desventajas y exclusión social (ver figura 3.10).[9]

Figura 3.10 Usuarias de un sistema robótico ARM. [9]

- Aplicaciones en el espacio: Vehículos tipo Robert.[5]
- Aplicaciones submarinas.

- Aplicaciones en el área nuclear: La utilidad del sistema de teleoperación radica en poder tratar y manipular sustancias radiactivas, así como moverse por entornos contaminados, sin peligro para el ser humano. [5]
- Aplicaciones militares: Vehículo terrestre no tripulado, UGV por sus siglas en inglés (Unmanned Ground Vehicle); Un ejemplo de este tipo de vehículo es el SARGE (Surveillance And Reconnaissance Ground Equipment) creado por Sandia National Laboratories, el cual posee retroalimentación sensorial de video cámaras, así mismo tiene sistemas de comunicación que son muy rápidos gracias al uso de radio frecuencias (ver figura 3.11). [5]

Figura 3.11 SARGE vehículo tipo UGV. [5]

3.5 Tecnología e interfaces hápticas

La palabra háptica tiene su origen en griego (hapto, para sujetar o atar), y se refiere a la sensación del tacto. La háptica es el área que estudia e investiga cómo puede mezclarse el sentido del tacto con un mundo virtual que puede representar un entorno remoto [10], de modo que un dispositivo háptico es aquel capaz de generar estímulos (retroalimentar fuerza) a una persona que interactúa con entornos virtuales o remotos. Tales dispositivos trasladan una sensación de presencia al operador. [11]

Cabe mencionar que la diferencia entre un sistema háptico y un sistema teleoperado es el tipo de señal que genera la retroalimentación. Para el sistema háptico la zona remota no existe y se simula a través de software (mundo virtual), mientras que un sistema teleoperado la zona remota está formada por un robot manipulador y sensores. [12]

La retroalimentación de fuerza en las interfaces hápticas se lleva a cabo mediante motores, válvulas hidráulicas o neumáticas y frenos magnéticos entre otros. Existe una serie de variables que tienen que ver con el diseño, sensibilidad y estabilidad de las interfaces con retroalimentación táctil y son las siguientes: [11]

- Fuerzas de contacto: agarre fuerte o agarre de precisión.
- Inercia aparente: mínima masa percibida por el operador cuando mueve la interfaz háptica a través del espacio libre.
- Transparencia: no debe ejercerse ninguna fuerza sobre el usuario mientras no exista interacción física con el entorno virtual.
- Fricción aparente: las pérdidas por fricción en un interfaz háptica deben ser inferiores a la mínima fuerza que se pueda percibir mientras se interactúa con el entorno virtual, ya que sino la interfaz deja de ser transparente, impidiendo diferenciar si las fuerzas percibidas por el usuario provienen de la retroalimentación deseada o de las perdidas mecánicas del dispositivo.
- Ancho de banda de estímulos y control: referidos a la frecuencia de percepción y aplicación de estímulos táctiles.
- Espacio de trabajo y grados de libertad: definidos según la aplicación. [11]

Al momento de seleccionar una interfaz háptica, se debe de tener claro la aplicación para la que se quiere, ya que esto se tiene que ver reflejado en el tipo de interfaz elegida. Lo anterior es porque en cuanto a retroalimentación se refiere, las hay de tres tipos: [13] y [14]

- Con retroalimentación de fuerza: Retroalimenta la dureza, el peso y la inercia del objeto virtual.

- Con retroalimentación táctil: Retroalimenta la geometría de la superficie de contacto, su rugosidad, la suavidad, detecta contornos lisos ó materiales elásticos, además de la temperatura del objeto virtual.
- Retroalimentación propioceptiva (sentido que informa al organismo de la posición de los músculos): Retroalimenta información acerca de la posición del cuerpo del usuario o su postura. [14]

Las interfaces hápticas se pueden clasificar de acuerdo al tipo de actuadores utilizados, la escala en generación de fuerza, su portabilidad o soporte. La siguiente es la clasificación más común: [11]

- De escritorio: se refiere a los joystick e interfaces como la Falcon o la Force Feedback (ver figura 3.12).
- Fijas: Pueden ser exoesqueletos (dispositivo que se adapta a la fisiología del usuario, ver figura 3.13) fijos o brazos de robot.
- Portátiles: Pueden ser exoesqueletos sujetos al operador o guantes (ver figura 3.14 y 3.15 respectivamente). [11]

Figura 3.12 *Joysticks* esféricos de Microsoft "Force Feedback". [11]

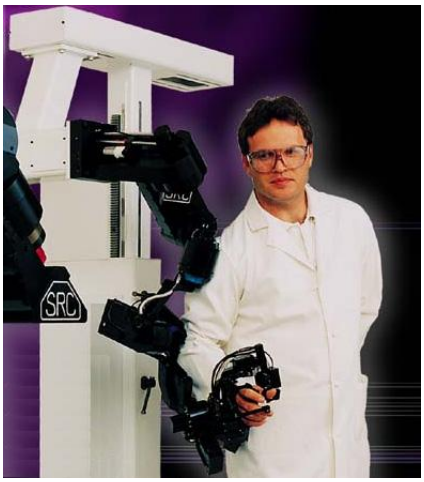


Figura 3.13 Exoesqueleto fijo (Dextrous Hand Master Exoskeleton). [11]

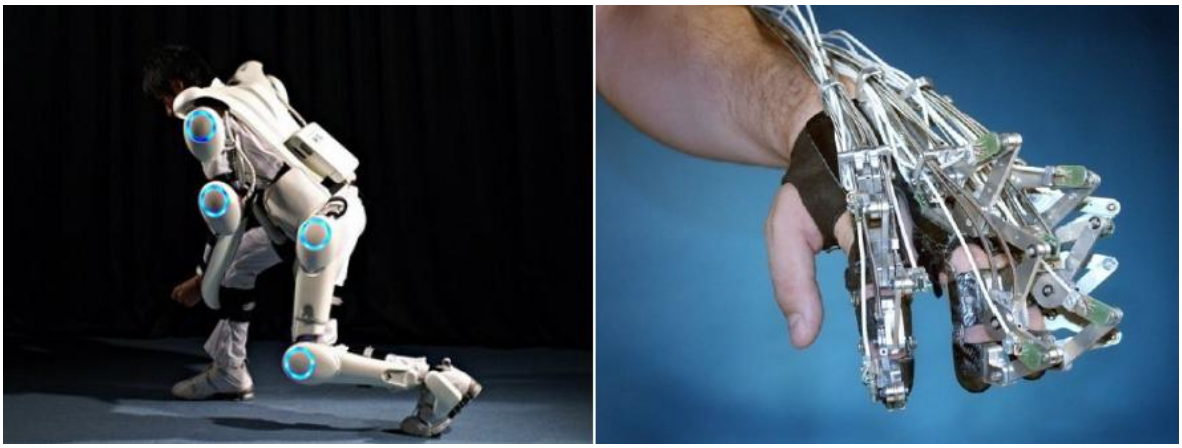


Figura 3.14 Exoesqueleto sujeto al operador. [15]

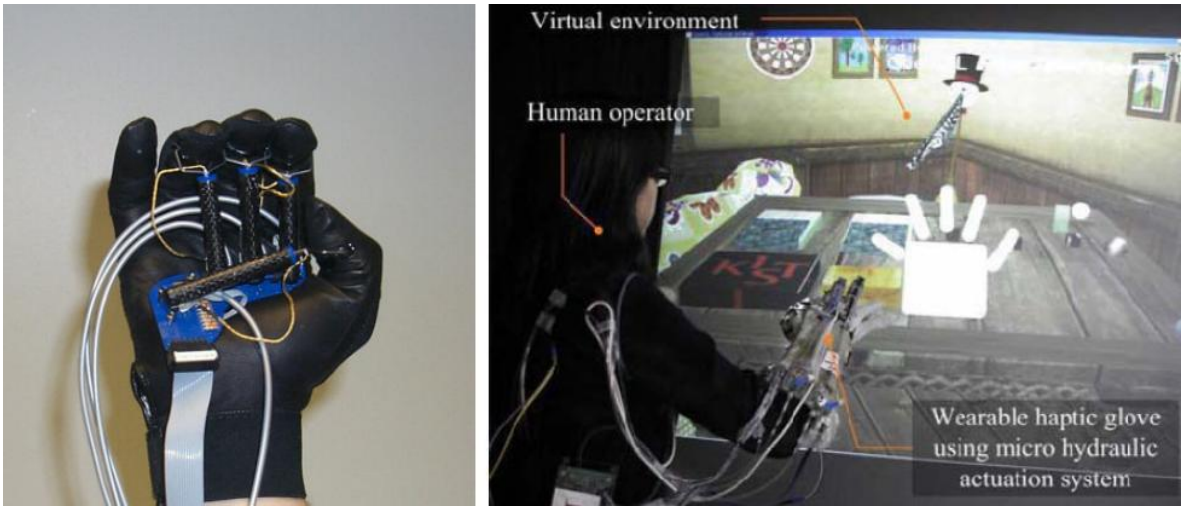


Figura 3.15 Guante háptico integrado con actuadores hidráulicos. [15]

Actualmente la tecnología háptica se aplica en una variedad de campos tales como:

- La teleoperación: Por ejemplo pilotear un avión de manera que se le permita al operador percibir la fuerza de resistencia al mover la palanca de mando.
- Las tecnologías móviles: la vibración.
- En la medicina: Desarrollando ambientes virtuales para el entrenamiento de cirujanos y simulación de cirugías. Puede ser para casos simples como la inserción de una aguja o micro robots para cirugías mínimamente invasivas. No obstante, se apunta hacia el modelado de órganos vitales (simular virtualmente y con fidelidad las propiedades mecánicas y físicas de los órganos) para poder llegar a realizar operaciones complejas. (Elizabeth Mesa Múnera) [15]. Otro ejemplo es TraumaVision, que es un simulador de entrenamiento ortopédico en un entorno informático seguro, con retroalimentación de fuerza creando un paciente virtual en él se puede practicar. TraumaVision es capaz de simular la sensación de raspado al mover el taladro junto con el hueso y también simula la sensación de un hueso denso en la capa externa y más suave en la capa interna del mismo. [16]
- Telerobótica: Donde a través de una interfaz el operador puede transmitir las acciones a un robot y por otra parte se excitan los sentidos humanos de acuerdo a la información recibida. A través de esta interfaz se pretende controlar el robot, entrenarlo para hacer ciertas tareas y en un futuro se plantea hacer teleoperación utilizando medios de comunicación como Internet. [17]
- En los videojuegos, como por ejemplo los controles que vibran en respuesta a diversas situaciones que ocurren en el mundo virtual, así como reconocer direcciones debido a que el dispositivo permite generar fuerzas en las tres dimensiones, entonces el jugador puede conocer la posición de un atacante debido a la dirección de la fuerza que recibe, entre otras aplicaciones. [18]

3.6 Interfaz Novint Falcon

En la figura 3.16 se puede ver la interfaz háptica utilizada en el proyecto que es de la empresa Novint y el modelo es el Falcon. Por otro lado, a continuación se mencionan las características técnicas de la interfaz y se explica en qué consiste el HDAL proporcionado por el fabricante.

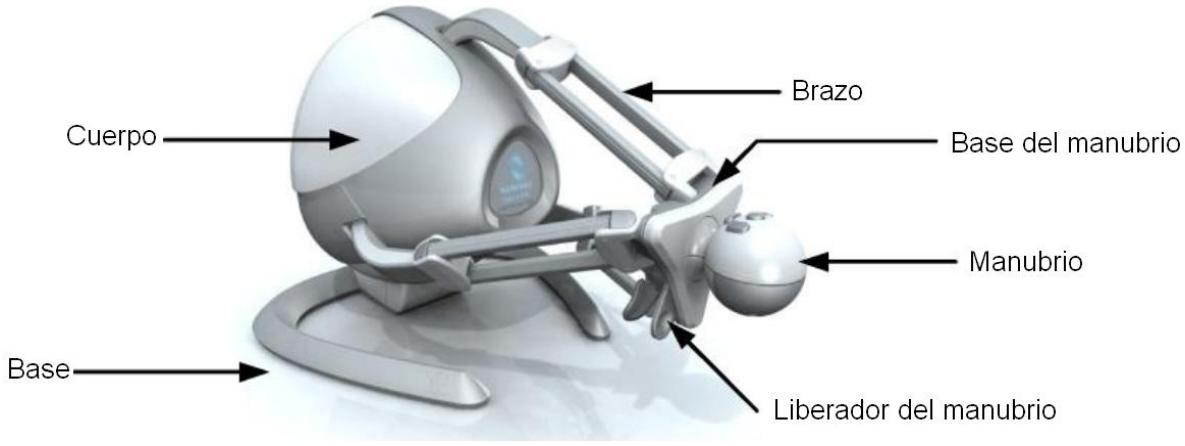


Figura 3.16 *Interfaz háptica Novint Falcon*. [4]

3.6.1 Detalles técnicos

Entre los aspectos técnicos de la interfaz háptica se encuentran los siguientes:

- 3 dimensiones con un área de trabajo de: 4 x 4 x 4 pulgadas.
- Retroalimentación de fuerza con un valor de hasta 2 lbs.
- Resolución de posición de 400 dpi.
- Desconexión rápida (1 segundo).
- Comunicación vía USB 2.0.
- Tamaño de la interfaz: 9 x 9 x 9 pulgadas.
- Peso: 6 lbs.
- Potencia: 30 watts.
- Alimentación: 100V-240V @ 50Hz-60Hz.

También la empresa Novint establece que la computadora en la que se va a instalar los drivers de la interfaz háptica tenga las siguientes características como mínimo:

- Procesador de 1.0 GHz, aunque el recomendado es de 2.4 GHz.
- Sistema operativo: Windows XP Service Pack 2 ó Windows Vista.
- Tarjeta de video de 128Mb, aunque la recomendada es de 256Mb.
- Versión DirectX: DirectX 9.0c
- Espacio en disco duro para la interfaz: 1.5 GB.
- Memoria RAM: 512 MB, aunque se recomienda 1 GB.
- Unidad de DVD.
- Puerto USB 2.0.
- Conexión a internet.

La información anterior fue tomada de la página del fabricante [4].

3.6.2 HDAL (Haptic Device Abstraction Layer)

En la siguiente parte se utiliza el término capa de abstracción como una forma de ocultar los detalles de implementación de ciertas funciones, donde el termino abstracción tiene como función hacer referencia al "¿qué hace?" más que en "¿cómo lo hace?" o bien, incluso se puede ver como el concepto de caja negra.

El objetivo principal de la capa de abstracción de dispositivos hápticos por sus siglas en ingles HDAL (Haptic Device Abstraction Layer) es proporcionar la comunicación entre la aplicación desarrollada y la interfaz háptica, es decir el programador no debe de preocuparse de cómo se inicializa el dispositivo, como se da el intercambio de datos (por ejemplo obtener la posición del manubrio, estado de botones), como se desconecta y apaga correctamente el Falcon, así como tampoco porqué medios (hardware) se da la retroalimentación de fuerza. El HDAL también es responsable de lograr fidelidad háptica (para percibir una sensación en

la realidad al tacto, se debe de trabajar con una frecuencia en el rango de 500 Hz y 1 kHz), por lo que el HDAL hace que la función de retroalimentar fuerza se ejecute a una velocidad de 1 KHz.

El HDAL como tal, se implementa a través de una serie de capas como se observa en la figura 3.17. A su vez las capas se agrupan en dos secciones: a nivel de aplicación y a nivel de HDAL. A nivel de aplicación se tiene un componente gráfico (videojuegos) y otro háptico (retroalimentación táctil), por lo que el objetivo principal del programador es asegurar sincronía en los dos componentes, esto con el fin de que el usuario pueda percibir una estrecha relación entre lo ve y lo que siente.

Por otro lado, a nivel de HDAL como se observa en la figura 3.17 si se continúa descendiendo se llega la capa de abstracción de datos, que posee funciones como el inicio de ejecución del hilo del programa (se explica en el capítulo 5) así como variables para lograr la aplicación deseada. La capa de abstracción maneja una familia de drivers, uno para cada tipo de dispositivo. En este caso sería el driver para la Novint Falcon. El driver del HDAL es la capa que sabe cómo comunicarse con el tipo de dispositivo asociado al SDK, de manera que este pueda proporcionar toda la funcionalidad a la aplicación, mientras que el SDK del dispositivo tiene la responsabilidad de comunicarse con el dispositivo real, donde dicha comunicación es la de la PC con la interfaz háptica y se realiza vía USB.

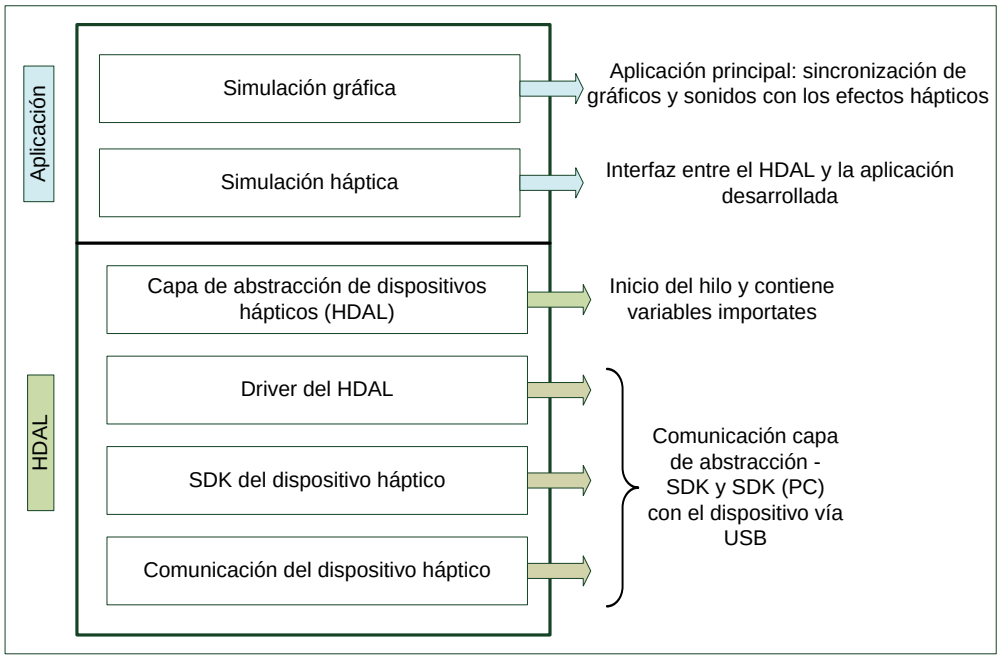


Figura 3.17 Estructura de capas del HDAL. [19]

Para poder utilizar el SDK se debe configurar un entorno de desarrollo (incluir la librería hdl.lib por ejemplo) en el programa que se vaya a usar, en este caso es el Visual C++ 6.0 como se indica en el documento "Haptic Device Abstraction Layer (HDAL), Programmer's Guide". El SDK de la Novint contiene la siguiente información: [20]

- Conceptos de tecnología háptica.
- Parte del modelado matemático que utilizó la empresa fabricante.
- Guía de instalación y de uso del SDK, así como de sus bibliotecas, rutinas de aplicación e inicialización del dispositivo.
- Interacción gráfica y personalización de un programa.
- Código de ejemplo, en estándar de Microsoft IDEs Visual C++ 6.0.

3.7 Comunicación serial, RS-232 [25 y 26]

La comunicación serial es un protocolo muy común para comunicación entre dispositivos que se incluye de manera estándar en prácticamente cualquier computadora. RS-232 (Recommended Standard 232) es una interfaz que designa una norma para el intercambio de una serie de datos binarios entre un DTE (Equipo terminal de datos) y un DCE (Equipo de Comunicación de datos), aunque existen otras en las que también se utiliza la interfaz RS-232. Cuando se transmite información a través de una línea serie es necesario utilizar un sistema de codificación que permita resolver los siguientes problemas:

- Sincronización de bits: El receptor necesita saber donde comienza y donde termina cada bit de la señal recibida para efectuar el muestreo de la misma en el centro del intervalo de cada símbolo.
- Sincronización del carácter: La información serie se transmite por definición bit a bit, pero la misma tiene sentido en bytes.
- Sincronización del mensaje: Es necesario conocer el inicio y fin de una cadena de caracteres por parte del receptor para, para poder detectar algún error en la comunicación de un mensaje por ejemplo,

En una comunicación en general, se pueden establecer canales para la comunicación de acuerdo a tres técnicas:

- Simplex: La comunicación serie usa una dirección y una línea de comunicación. Siempre existirá un transmisor y un receptor, no ambos.
- Half duplex: La comunicación serie se establece a través de una sola línea, pero en ambos sentidos. En un momento el transmisor enviará información y en otro recibirá, por lo que no se puede transferir información en ambos sentidos de forma simultánea.
- Full duplex: Se utilizan dos líneas (una transmisora y otra receptora) y se transfiere información en ambos sentidos. La ventaja de este método es que se puede transmitir y recibir información de manera simultánea.

En cuanto a modos de trasmisión existen dos modos básicos para realizar la transmisión de datos y son:

- Transmisión asíncrona.
- Transmisión síncrona.

3.7.1 Comunicación asíncrona

Cuando se opera en modo asíncrono no existe una línea de reloj común que establezca la duración de un bit y el carácter puede ser enviado en cualquier momento. Esto implica que cada dispositivo tenga su propio reloj y que previamente se haya acordado que ambos dispositivos transmitirán datos a la misma velocidad. En la transmisión asíncrona un carácter a transmitir es incluido con un indicador de inicio y fin de carácter, lo que se conoce como bit de inicio y un bit de parada. Durante el lapso en que no se transmite ningún byte, el canal debe estar en alto (1 lógico), de la misma forma al bit de parada se le asigna también un "1". Por otro lado, al bit de inicio del carácter a transmitir se le asigna un "0" lógico, por lo que un cambio de nivel de "1" a "0" lógico indicará al receptor que un nuevo carácter será transmitido.

La transmisión asíncrona definida por la norma RS-232, se basa en las siguientes reglas:

- Cuando no se envían datos por la línea, ésta se mantiene en estado alto.
- Cuando se desea transmitir un carácter, se envía primero un bit de inicio que pone la línea en bajo durante el tiempo de un bit.
- Luego se envían todos los bits del mensaje a transmitir con los intervalos que marca el reloj de transmisión. Por convenio se transmiten entre 5 y 8 bits.
- Se envía primero el bit menos significativo y de último el bit más significativo.
- Con el último bit del mensaje se envía el bit (o los bits) del final hace que la línea se ponga a 1 por lo menos durante el tiempo mínimo de un bit. Estos

bits pueden ser un bit de paridad para detectar errores y el bit o bits de stop, que indican el fin de la transmisión de un carácter.

Los datos codificados por esta regla, pueden ser recibidos de acuerdo a los siguientes pasos:

- Esperar la transición 1 a 0 en la señal recibida.
- Activar el reloj con una frecuencia igual a la del transmisor.
- Muestrear la señal recibida al ritmo de ese reloj para formar el mensaje.
- Leer un bit más de la línea y comprobar si es 1 para confirmar que no ha habido error en la sincronización.

3.7.2 Comunicación síncrona

Es un método más eficiente de comunicación en cuanto a velocidad de transmisión, esto porque no existe ningún tipo de información adicional entre los caracteres a ser transmitidos. Cuando se transmite de manera síncrona lo primero que se envía es un octeto de sincronismo, el cual realiza la misma función que el bit de inicio en la transmisión asíncrona, indicar al receptor que se va enviar un mensaje. Este carácter utiliza la señal local de reloj para determinar cuándo y con qué frecuencia será muestreada la señal, es decir, permite sincronizar los relojes de los dispositivos transmisor y receptor.

Capítulo 4: Pre diseño del sistema y disposición del equipo

Inicialmente se investigó el funcionamiento de la interfaz háptica Novint Falcon y se realizó un estudio completo del SDK basándose en el documento llamado HDAL. Seguidamente se procedió a seleccionar las partes de código a utilizar, se modificó y eliminó algunas otras, para finalmente crear las rutinas necesarias. Una vez que se tuvo el código fuente final, se procedió a programar el puerto serial para el circuito de prueba.

Una vez que se finalizó lo correspondiente a la interfaz háptica, se procedió a realizar el estudio y el análisis del manual de usuario del brazo robótico con el fin de entender su funcionamiento y para descargar uno de los programas de ejemplo en su CPU.

Seguidamente se procedió a investigar como comunicar la interfaz háptica con el brazo robótico. Inicialmente se pensó en protocolos como DDE (Dynamic Data Exchange) y OPC (OLE for Process Control) que son bajo Microsoft Windows y que son soportados por el software que se utiliza para simular y programar el brazo. COSIMIR es un software que se utiliza para programar el brazo y soporta DDE, mientras que CIROS que es la nueva versión de COSIMIR y que fue el software que finalmente adquirió el LIRA es capaz de soportar OPC. En estos protocolos se da una relación de cliente - servidor para compartir datos. No obstante, dado que el controlador del brazo robótico (ver figura 3.5) posee conexión RS-232 para comunicarse con la PC (para descargar los programas) se aprovecho de este puerto serial para enviar y recibir datos desde y hacia la computadora con lo que se resolvió un punto muy importante del proyecto. Cabe destacar que esto se logra gracias a la gama de instrucciones que soporta el robot Mitsubishi, cuyo lenguaje de programación es el Melfa Basic IV.

Una vez hecho el enlace entre la interfaz y el brazo robótico, se programó en el controlador del brazo robótico una instrucción capaz de leer, decodificar y ejecutar las órdenes que provengan de la Novint Falcon logrando así controlar el brazo y moverlo en 3D, además de enviar a la PC la información del valor de la fuerza a retroalimentar en la interfaz.

Finalmente se realizó el estudio cinemática directo para poder conocer la posición y la orientación de la pinza del brazo robótico respecto a un sistema de referencia colocado en la base del brazo. Para hacer esto se recurrió al estudio del algoritmo de Denavit-Hartenberg.

4.1 Disposición del equipo para el proyecto

Este proyecto en su inicio no contaba con todos los componentes, los cuales básicamente son 3: una interfaz háptica, una computadora y la estación de robot Festo. Las primeras dos si estaban desde un inicio del proyecto, no obstante la estación de robot no había llegado en la semana de arranque por lo que se recurrió a un circuito de prueba, que a la postre vendría siendo de gran ayuda para la comunicación de la interfaz con el brazo robótico.

En el caso de la interfaz se disponía de toda la información que trae el SDK que proporciona el fabricante además de los drivers, y en el caso del brazo robótico se procedió a buscar en internet la información directamente en las paginas oficiales de FESTO y MITSUBISHI para obtener la información e ir adelantando trabajo del robot aún cuando este no se encontraba en el LIRA.

4.2 Circuito de prueba de la interfaz háptica

El circuito de prueba se puede ver en la figura 4.1 y consistió en un PIC 18F2550 que recibía datos de la interfaz a través de la PC y se encargaba de

mover dos servo-motores. Para esto fue necesario programar el puerto serial tanto de la PC como del PIC.

En el caso de la interfaz háptica (se programa en C++) se abrió el puerto serial y cada 40ms se le enviaba al PIC las coordenadas de los ejes X, Y, Z y además se leía del PIC el valor de un potenciómetro que simulaba un sensor de peso o de presión, esto con el fin de retroalimentar una sensación de fuerza a la interfaz.

Por otro lado, para el PIC también se habilito la comunicación serial. La rutina que se programó en el PIC se puede ver en la figura 4.2, en la cual se configuro el puerto serial de la siguiente manera: velocidad de datos: 9600 baudios, paridad: ninguna, bits para el dato: 8 y 1 bit de parada. Seguidamente se establecían los dos puertos disponibles en el PIC para PWM y se habilitaba la interrupción del timer cero cada 8,1 ms para que el PIC leyera por medio de RS-232 las coordenadas de la interfaz (lee X, Y, Z pero solo utiliza X y Y) y traducía la coordenada en un valor de PWM que aplicaba a los servo-motores para lograr que estos se movieran. Finalmente digitalizaba el valor de tensión de un potenciómetro para enviarlo vía serial a la PC y de esta forma lograr retroalimentar un valor de fuerza a la interfaz háptica.

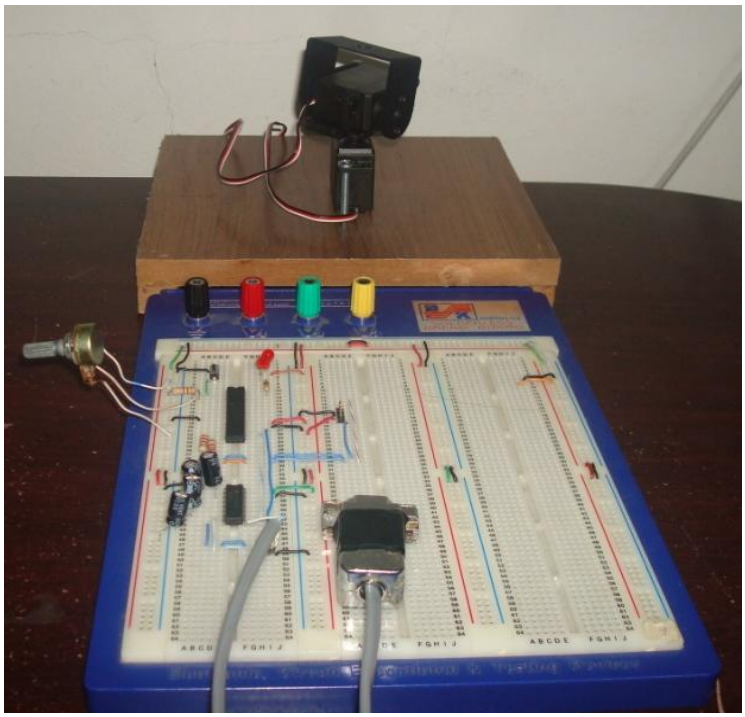


Figura 4.1 Circuito de prueba.

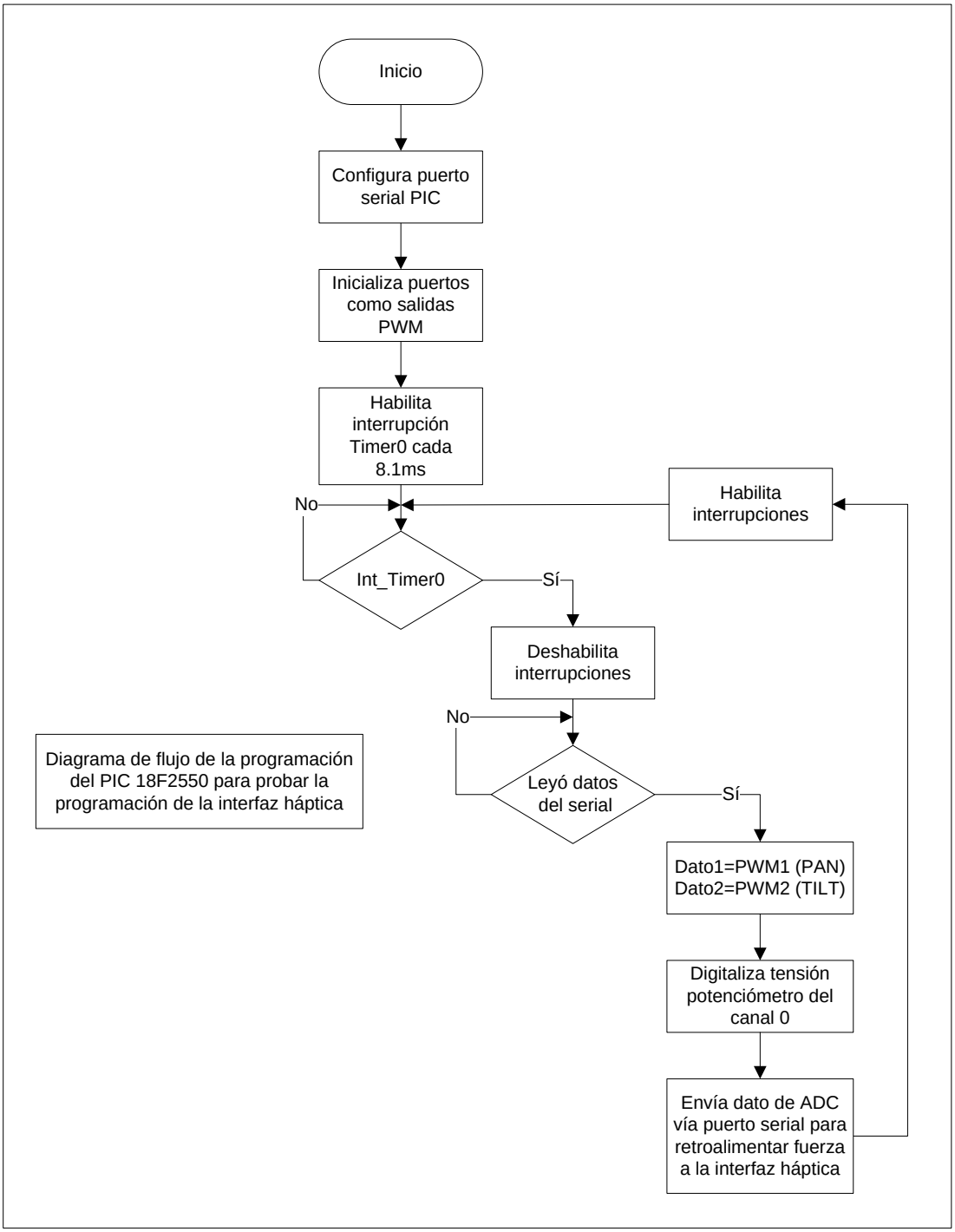


Figura 4.2 Diagrama para la comunicación del PIC con la interfaz.

Capítulo 5: Descripción de la solución del sistema teleoperado de un brazo robótico por medio de una interfaz háptica con retroalimentación de fuerza

El siguiente capítulo tiene como fin explicar detalladamente las rutinas desarrolladas tanto para la interfaz háptica como para el brazo robótico así como el medio que se utilizó para comunicar a ambos dispositivos. También se explica el desarrollo del modelo cinemática directo del brazo.

5.1 Diagrama general de la solución

En la figura 5.1 se muestran la solución general del proyecto. En esta figura se pueden ver tres etapas bien definidas: interfaz háptica, brazo robótico y el puente entre ellos para intercambiar datos.

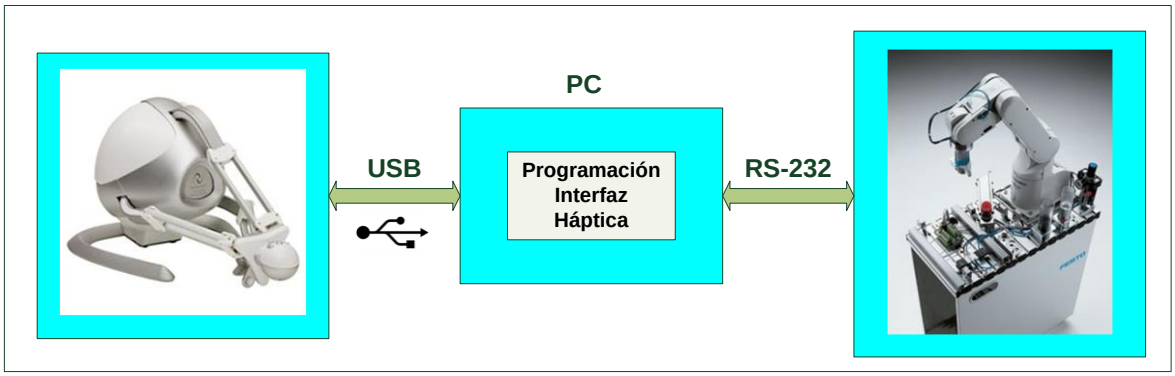


Figura 5.1 Diagrama para la comunicación del PIC con la interfaz.

Por medio de la interfaz háptica Novint Falcon se controla el brazo robótico. La comunicación entre la Interfaz y la PC se realiza a través del puerto USB, mientras que la comunicación entre la PC y el robot se realiza a través del puerto serial por el estándar RS-232. En ambas comunicaciones el intercambio de datos es bidireccional. La interfaz posee tres GD mientras que el brazo posee cinco por lo que no se puede mover todas las articulaciones del robot al mismo tiempo. Para esto la interfaz cuenta con cuatro botones y uno de ellos es el encargado de

alternar los ejes que se quieran mover (en un momento se mueven 3 y en otro 2). Otros dos botones se usan para abrir y cerrar la pinza, de modo que al cerrarla el brazo envía un valor de fuerza a la PC y esté se vea reflejado en un mayor esfuerzo para manipular la interfaz. Adicionalmente se implemento el algoritmo de Denavit-Hartenberg con el fin de conocer la posición y orientación final de la pinza y así evitar algún choque de la misma con el brazo o con la mesa de trabajo.

En la figura 5.2 se puede ver el diagrama general de la solución que corresponde a la programación en la PC que se realizo en Visual C++ 6.0, mientras que en la figura 5.9 se puede ver la solución de la programación del brazo robótico que se llevo a con el lenguaje Melfa Basic IV mediante el entorno de desarrollo que ofrece el programa CIROS. Todas estas funciones se explican en las secciones posteriores.

La programación en la PC se puede resumir de manera muy general en los siguientes pasos:

- Inicialización de la interfaz háptica (sección 5.2.1).
- Envió vía puerto serial la posición de inicio y la señal de abrir la pinza al brazo robótico el cual está esperando desde antes la información mencionada (sección 5.3.2).
- Se espera que el brazo conteste por el mismo medio con un valor de fuerza a retroalimentar en la interfaz (sección 5.3.3).
- Se pregunta si desea salir de la aplicación.
- En caso de que desee salir de la aplicación, se envía la posición de inicio, se des inicializa la interfaz (sección 5.2.4) y finalmente se cierra la aplicación.
- En caso de no salir de la aplicación, se procede a mover las articulaciones del brazo robótico por medio de la interfaz háptica (sección 5.6).
- Se aplica el algoritmo de Denavit-Hartenberg con el fin de analizar si la posición a enviar no provoca algún accidente (sección 5.4).

- Si el movimiento no es válido, se ajustan los ejes J2 y J3 con el fin de prevenir un accidente y se envía la posición. El valor de los ejes J1, J4, J5 y J6 no se modifican.

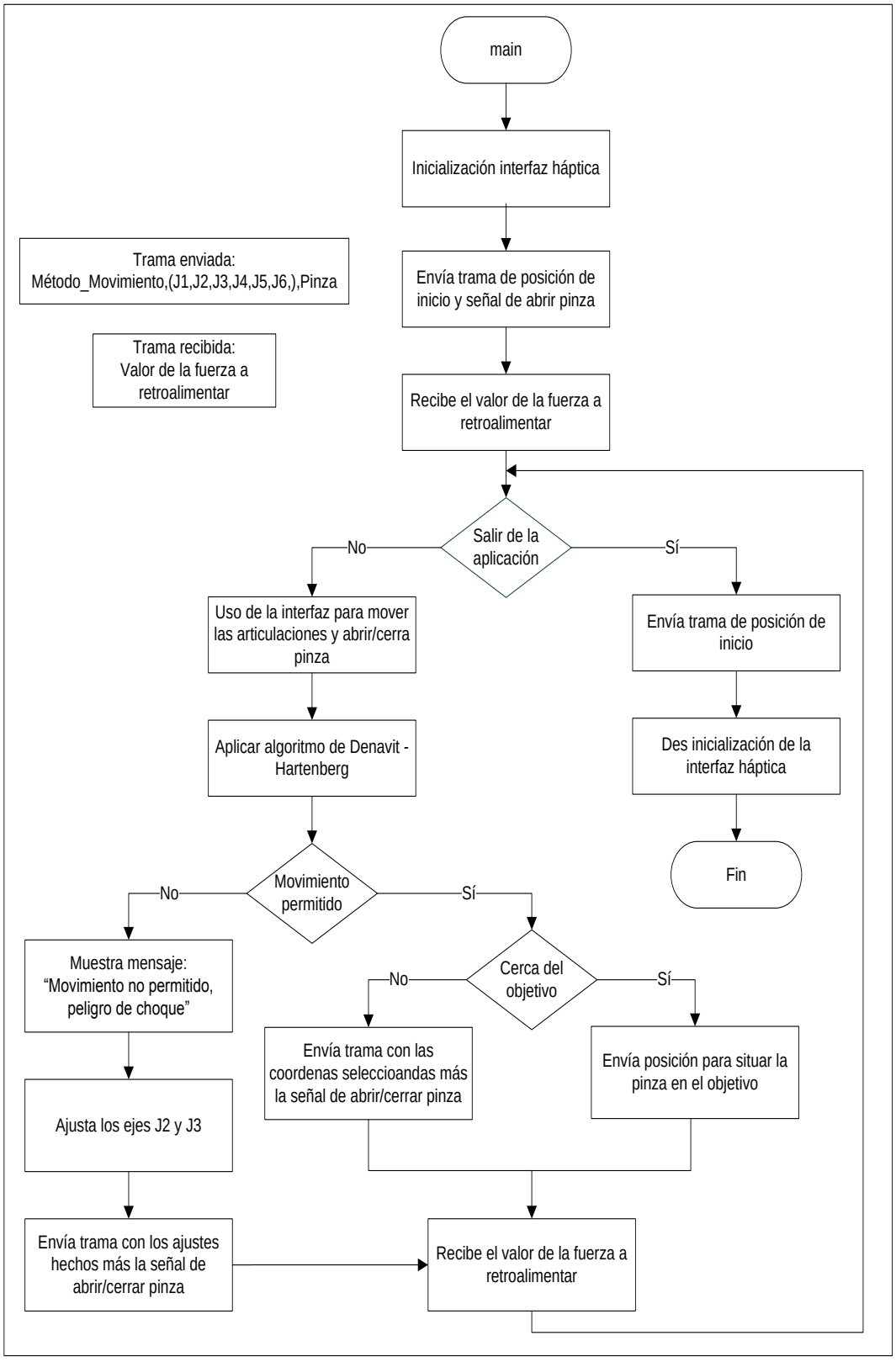


Figura 5.2 Diagrama de la solución general.

- Si el movimiento es válido, analiza si está cerca del objetivo (sección 5.4.1). En caso de estarlo se envía dicha posición y de no ser así envía la posición que indique la interfaz para luego recibir el valor de fuerza a retroalimentar.
- Finalmente se repite el ciclo hasta que se presione el botón de salir (sección 5.2.3) que cierra la aplicación.

5.2 Programación de la interfaz háptica

La programación de la Novint Falcon se realiza en Visual C++ 6.0 y dicho código debe de ser capaz de inicializar correctamente la interfaz, verificar errores de comunicación entre la interfaz y la PC, obtener la posición del manubrio del Falcon, conocer el estado de los botones, enviar la poción de movimiento y generar fuerzas de acuerdo la información recibida desde el brazo. En este punto es importante mencionar que lo anterior se logra gracias al ejecutar el hilo del HDAL que se explica más adelante.

5.2.1 Inicialización y manejo de la interfaz

En la figura 5.3 se puede observar el proceso de inicialización de la interfaz háptica, el cual se explica a continuación. Lo primero que se debe mencionar, es que para esto se utilizan funciones ya predefinidas que forman parte del HDAL que fue explicado en el capítulo 3. Para hacer uso de estas funciones basta con incluir las librerías <*hdl/hdl.h*> y la <*hdlu/hdlu.h*>. Una vez hecho esto, siguiendo el diagrama de la figura 5.3 el primer paso es identificar el dispositivo por medio de la función *hdlInitNamedDevice()*, está función se le asigna a un manejador de interfaz cuya función es verificar si se pudo establecer comunicación entre la PC y el Falcon (comunicación vía USB) para poder hacer uso de la misma en la aplicación que se va a desarrollar. En caso de que la identificación falla, la aplicación se cierra y se debe de volver a correr el programa. Seguidamente se inicia el hilo que va a ejecutar en paralelo tanto la parte del programa que se

comunica con la interfaz (ver figura 5.4) como de la aplicación hecha para manejar el brazo robótico (ver sección 5.3 y 5.5).

A continuación, se crea un manejador del servo que es el responsable de comunicarse con la interfaz para obtener cierta información (figura 5.4) y que se va a ejecutar cíclicamente durante la aplicación.

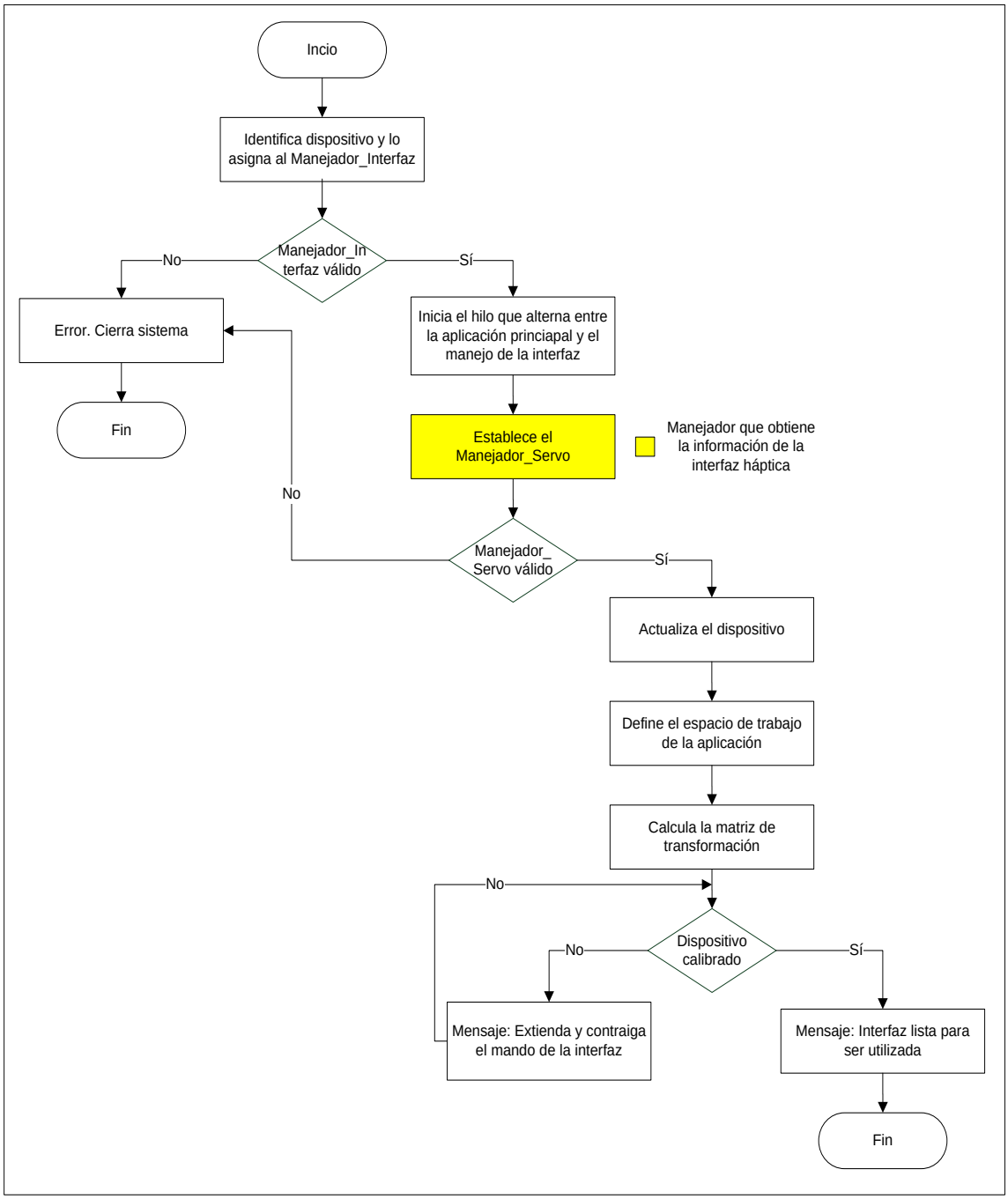


Figura 5.3 Diagrama general de la inicialización de la interfaz.

Inmediatamente se verifica que el manejador del servo se haya podido crear correctamente ya que es vital para el funcionamiento del sistema en general, de no ser así se procede a cerrar la aplicación. Luego de esto, se actualiza el dispositivo a través de la función *hdlMakeCurrent()*, que recibe de parámetro el manejador de la interfaz ya inicializado con el fin de reconocer el dispositivo que se va a programar y accionar en caso de tener más de una interfaz conectada.

Como se menciono antes, la interfaz tiene un área real de 4 x 4 x 4 pulgadas (de -2 a 2), no obstante se define un área de trabajo que corresponde a las medias con las que se desea trabajar, en este caso es el valor del ángulo de cada articulación, por lo que se eligió de -200 a 200 que corresponde al ángulo mayor de todas las articulaciones (J6, como se vio en la tabla 3.1). Para lograr extrapolar las medidas físicas de la interfaz con las de la aplicación se utiliza una matriz de transformación, que por medio de la función *hdluGenerateHapticToAppWorkspaceTransform()* logra hacer dicha operación. Finalmente se verifica si el dispositivo se encuentra calibrado o no mediante la función *hdlGetState().* En caso de que no se encuentre calibrado se debe de extender y contraer el manubrio de la interfaz hasta que se logre calibrar exitosamente y así concluye la inicialización de la interfaz.

En la figura 5.4 se puede ver la función que realiza el manejador del servo, que es el responsable de comunicarse con la interfaz. Cabe destacar que como se mencionó anteriormente este manejador se ejecuta de manera cíclica y corre de forma paralela con la aplicación principal por medio del hilo. De manera inicial el manejador servo por medio de la función *hdlToolPosition()* obtiene el estado del dispositivo, es decir, actualiza la posición real de la interfaz y con el dato de la matriz de transformación de la inicialización mapea la posición que debe de tener la aplicación en ese momento. Seguidamente se realiza el análisis de fuerza (ver sección 5.1.2), y finalmente, para poder aplica físicamente dicha fuerza al dispositivo se usa la función *hdlSetToolForce()*.

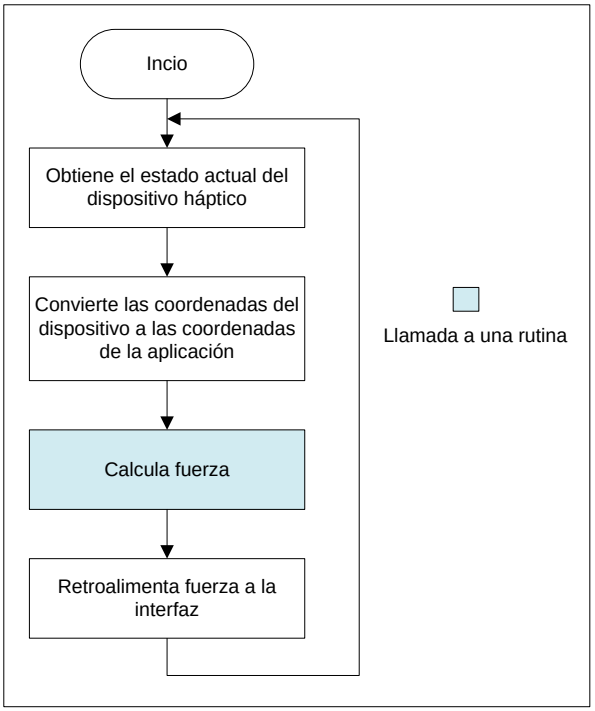


Figura 5.4 Diagrama general de la función del manejador servo.

5.2.2 Retroalimentación de fuerza

En el sistema que se desarrolló se retroalimentó fuerza de dos maneras. Esto se puede apreciar en la figura 5.5. En primer lugar, como cada articulación tiene un máximo de grados a moverse (por ejemplo, J1 va de -150º a 150º) se programó la interfaz de manera que cuando el brazo robótico se haya desplazado lo máximo posible en cualquiera de sus cinco GD y por medio de la interfaz se le ordene seguir moviendo alguno de ellos, el respectivo eje de la interfaz que se esté accionando se bloquee y no se permita mover más el manubrio en ese sentido (en la sección 5.5 se explica cómo manejar el brazo con la interfaz). Como se mencionó anteriormente la interfaz solo posee tres GD, por lo que alterna las articulaciones del brazo a mover, en un momento muevo J5 y J6 (cuando se presiona el botón 4 del Falcon) mientras que en otro momento mueve J1, J2 y J3. El chequeo de la coordenada que se le vaya a enviar al brazo se revisa constantemente con el fin de bloquear el eje de la interfaz en el momento preciso.

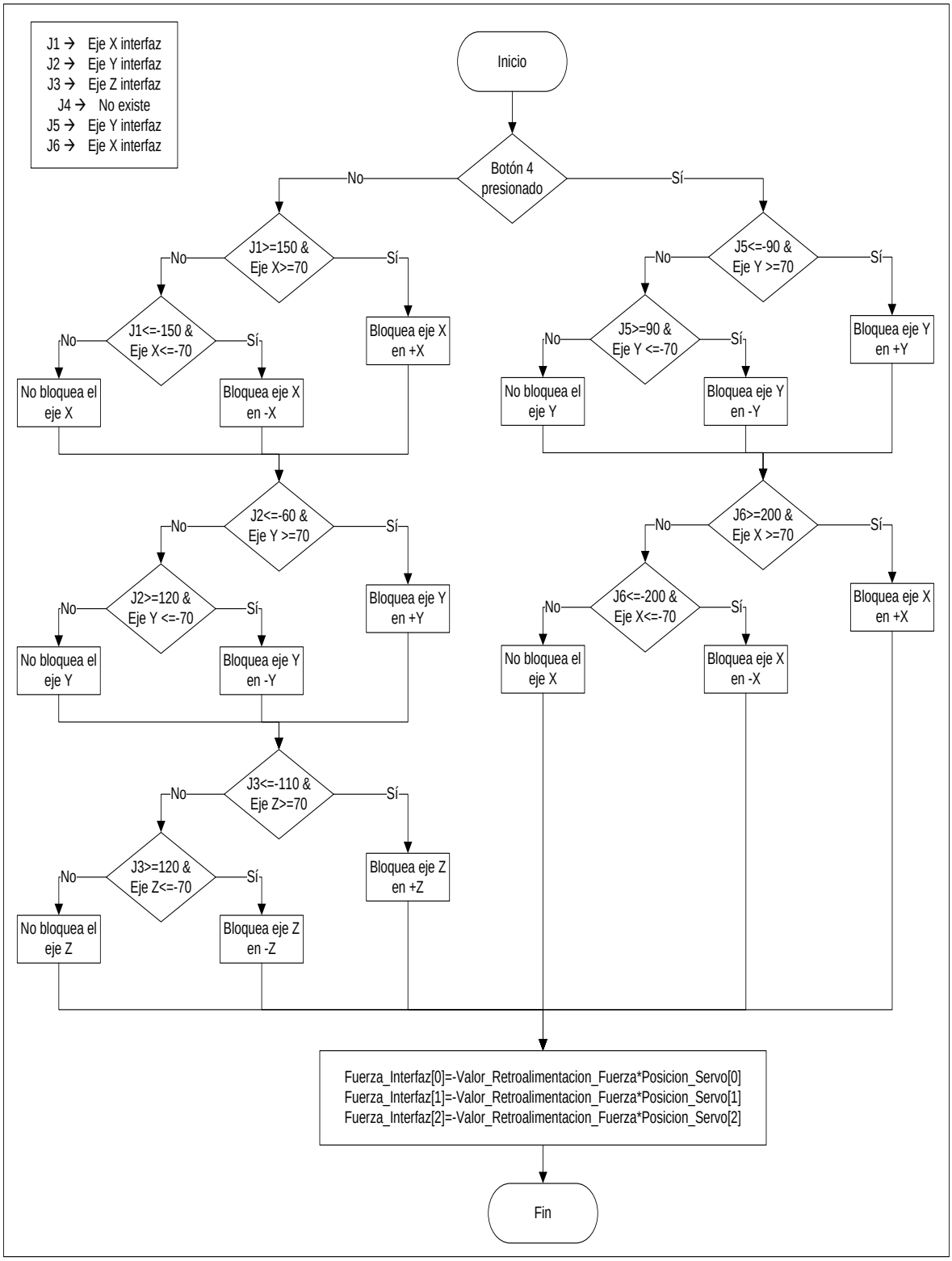


Figura 5.5 Diagrama general para la retroalimentación de fuerza.

Como se aprecia en la figura 5.5, las condiciones siempre evalúan si la posición de la interfaz es mayor a 70 y si el ángulo del brazo está en su extremo permitido, tanto positivo como negativo. En el caso del 70, se utilizo este número, porque se designó una zona neutra en los tres ejes de la interfaz que va desde -70º a 70º (ver sección 5.5).

La segunda forma de retroalimentar fuerza, es en el caso de que la pinza se haya serrado, esto porque el controlador del brazo robótico responde con un valor que representa la fuerza que se debe de aplicar. Este valor varía entre 0 y 160, que corresponde a cuando la pinza está abierta y cuando está cerrada respectivamente. El por qué del 160 radica en que con este valor el eje de la interfaz al que se le aplique la fuerza, experimenta una resistencia al movimiento razonable como para que el usuario perciba el efecto de mover algún objeto.

El uso del valor que se recibe del controlador, se explica con la figura 5.6, esto porque una vez que se representa el dispositivo háptico en el entorno virtual el cálculo y la aplicación de la fuerza se pueden dar en dependencia del movimiento, el tiempo o una combinación de ambos. En el caso de los vectores de fuerza dependientes del movimiento se requiere la retroalimentación de posición del manubrio del Falcon. De una manera más concreta, para la retroalimentación de fuerza se sigue el modelo resorte, que es utilizado por su versatilidad y simplicidad. Este modelo se basa en la Ley de Hooke

$$\mathbf{F} = -k * \mathbf{x} \qquad (2)$$

donde k es la constante de elasticidad y **x** un vector de desplazamiento (posición de la interfaz). Como se aprecia en la figura 5.6, al mover el manubrio de la interfaz a la derecha, esté va a experimentar una fuerza en sentido contrario (ecuación 2, por eso es el signo menos de la fórmula) que va a depender de la posición en la que se encuentre (letra “d” de la figura 5.6) y de la constante k (k=160 ó k=0). Está dependencia es lineal, entre mayor desplazamiento y mayor sea la contante, mayor el valor de fuerza a retroalimentar.

Finalmente, a nivel de programación, el valor de fuerza de cada eje se guarda en un arreglo, donde por ejemplo, en el caso del eje X, la función sería la siguiente:

Fuerza_Interfaz[0]=-Posicion_Aplicacion[0]*Valor_Retroalimentacion_Fuerza (3)

donde Valor_Retroalimentacion_Fuerza es igual a la constante k.

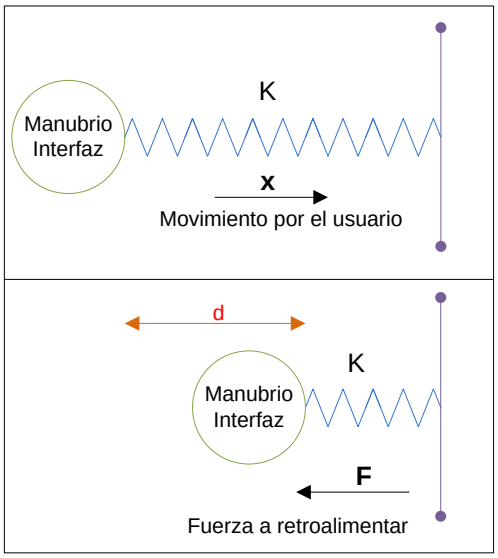


Figura 5.6 Diagrama del modelo para la reproducción de fuerza.

5.2.3 Designación y manejo de los botones

El manejo de los botones se realiza a través de la función *hdlToolButtons()*, con la que se revisa el estado de los mismos. Esta función es capaz de saber a través de una máscara cual de los cuatro botones se encuentra presionado y así poder tomar decisiones de acuerdo a la tarea que se le asigno a cada uno. En la tabla 5.1 se puede observar la función de cada botón y en la figura 5.7 se puede apreciar el número correspondiente.

Tabla 5.1 Función de los botones de la interfaz háptica.

Nombre del botón	Función
HDL_BUTTON_1	Salir del programa
HDL_BUTTON_2	Abrir pinza
HDL_BUTTON_3	Cerrar pinza
HDL_BUTTON_4	Alternar articulaciones del brazo (Si el botón no está presionado mueve J1, J2, J3 y si está presionado mueve J5, J6)

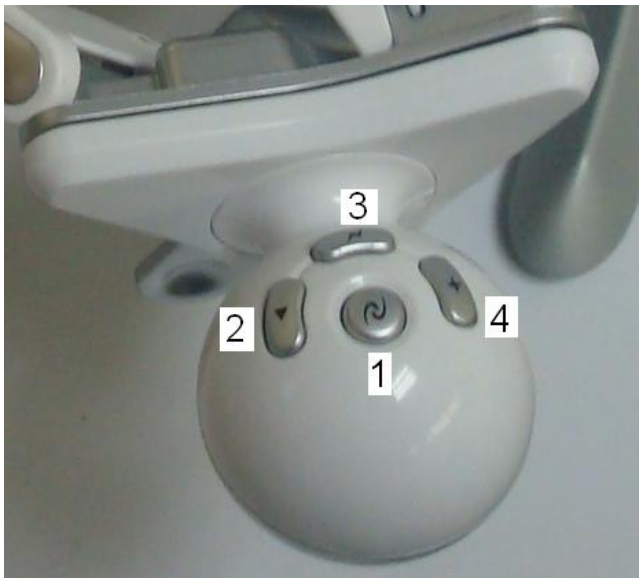


Figura 5.7 Número de cada botón de la interfaz.

5.2.4 Des inicialización de la interfaz

En el momento que se desea salir de la aplicación, se eliminan los manejadores creados tanto el de la interfaz (verifica si se pudo establecer comunicación entre la PC y el Falcon) como el del servo (adquirir datos de la interfaz) y se termina el hilo de ejecución. Todo esto es necesario para el posterior uso de la interfaz, ya sea corriendo de nuevo el programa hecho e incluso para jugar, que es el propósito principal por el cual fue hecho el FALCON.

5.3 Programación del puerto serial, RS-232 [29]

El protocolo para la conexión de la computadora con el controlador CR1-571 fue el serial bajo el estándar RS-232. En un principio se quiso usar USB, no obstante solo los controladores que le siguen al CR1-571 poseen ese tipo de conexión. Por otro lado, para el intercambio de datos inicialmente se pensó en protocolos industriales tales como DDE u OPC, pero debido a la facilidad que representa el uso del puerto serial en la computadora se prefirió usar el estándar RS-232. El control del puerto serie se realizó en C++ utilizando las funciones y estructuras que el WIN API32 tiene implementadas para realizar esta tarea. Para usar el puerto serial correctamente, se necesitó cuatro funciones: abrir (configuración), escribir, leer y finalmente cerrar el puerto.

5.3.1 Configuración del puerto serie

Para la configuración del puerto serie, lo primero que se hizo fue crear dos variables:

- Tipo DCB: Contiene la configuración del puerto.
- Tipo HANDLE: Es un manejador cuya función es manipular el puerto.

Al manejador que se creó se le asigna la función *CreateFile()* que abre un fichero o un puerto de comunicaciones. Está función recibe de parámetro lo siguiente:

- Puerto: Cadena de caracteres en la que se indica el nombre del puerto que se quiere abrir, por ejemplo “COM1”.
- GENERIC_READ | GENERIC_WRITE: Con esto se indica que se quiere leer y escribir.
- OPEN_EXISTING: Si el puerto no existe la función falla.

En el caso de que la asignación del manejador falle, por ejemplo si el puerto no existe, se devuelve el valor INVALID_HANDLE_VALUE y se cierra la

aplicación. El siguiente paso que se realizó fue mediante la función *GetCommState()* cargar la variable tipo DCB con la información de las características por defecto del puerto, luego con la función *BuildCommDCB*() se cambia por los valores almacenados en dicha variable, para esto se utilizó la cadena de caracteres cuyo formato es el siguiente “b, p, d, s” (bits por segundo: 9600, paridad: par, bit de datos: 8 y bits de parada: 2). Finalmente la función *SetCommState()* actualiza la configuración del puerto con la estructura DCB modificada.

Otro punto importante fueron los “ time out ” de las llamadas al puerto. Estos “time out” están almacenados en una estructura de tipo COMMTIMEOUTS que mediante la función *SetCommTimeouts()* se actualiza para el puerto. Los “time out” almacenan los tiempos que se quiso que el puerto estuviera a la espera de que llegue o salga un dato antes de indicar un error en la transmisión o en la recepción. El resultado final de la configuración del puerto serial es el siguiente método: AbrirPuertoSerie("COM1","9600,e,8,2") donde se aprecia las características antes mencionadas.

5.3.2 Envío de datos por el puerto serie

Una vez que el puerto se encuentra abierto se puede escribir o leer de él. Para poder escribir en el puerto serial se usa la función *WriteFile()* que recibe los siguientes parámetros: el manejador antes creado en la configuración para saber en cual puerto va escribir, un Buffer que corresponde al dato a enviar y una variable tipo int que por medio de la función *strlen()* obtiene el número de bytes a enviar.

En el caso del envió de datos es necesario armar la trama requerida. El controlador del brazo robótico recibe los datos en el formato ASCII y la variable que contiene la posición de la interfaz (la posición de la interfaz representa un valor de ángulo para su respectiva articulación del brazo) es de tipo double, por lo

que se emplea la función *sprintf(J1_Serial, "%4.2f", J1)* para hacer la conversión de datos, donde J1_Serial es un arreglo tipo char que contiene los caracteres ASCII a enviar y J1 es la variable tipo double que almacena el ángulo de la articulación respectiva, mientas que el %3.2f indica 3 dígitos antes del punto y 4 dígitos después.

Una vez hecha la conversión de datos se procede a enviar la trama que es la siguiente:

"PRN(J1_Serial,J2_Serial,J3_Serial,J4_Serial,J5_Serial,J6_Serial),M1_Pinza CR"

donde PRN es un encabezado que se debe de agregar para que el controlador del brazo reciba el dato correctamente. Seguidamente se envía el valor del ángulo de cada articulación, cabe destacar que a pesar de que el brazo RV-2AJ solo posee cinco GD se deben enviar seis posiciones, porque ese es el formato establecido y el controlador usado puede manejar un brazo robótico de seis GD. Las posiciones deben de ir encerradas en paréntesis, mientras que la coma "," que sigue, separa las variables de posición con las numéricas. La variable M1 es de tipo int y tiene un valor de cero en caso de que la pinza se encuentre abierta y un uno si la pinza está cerrada. Como M1 es de tipo int debe ser convertida a su correspondiente valor en ASCII y esto se hace con la función *itoa(M1,M1_Pinza,10)*, donde M1_Pinza es un arreglo de tipo char y el 10 indica que la variable int esta en decimal. Finalmente se debe de agregar un código para terminar la transmisión, en este caso es el CR (carriage return) que tiene un valor de **0D** en hexadecimal.

5.3.3 Leer datos del puerto serie

Para leer del puerto serial se usa la función ReadFile() que recibe como parámetro una variable tipo char que almacena el dato leído. Una vez que se envía los datos al brazo, este responde inmediatamente con un valor de fuerza a retroalimentar, este valor viene en formato ASCII por lo que debe ser pasado a entero para poder hacer operaciones con él. Para esto se emplea la función *Valor_Retroalimentacion_Fuerza = atoi(Dato_Fuerza)*, donde Dato_Fuerza es un

arreglo de char que contiene el dato leído del puerto y la variable Valor_Retroalimentacion_Fuerza es de tipo int con lo que se logra hacer la conversión del dato.

5.3.4 Cierre del puerto serie

En el momento que se presiona el botón de salir de la aplicación se procede a cerrar el puerto, esto se logra con la función *CloseHandle()* que lo que hace es eliminar el manipulador del puerto serie.

Con esto concluye todo lo que correspondió al manejo del canal que se utilizo para intercambiar datos entre la PC y el controlador del brazo robótico.

5.4 Denavit-Hartenberg [22], [30], [31], [32], [33], [34] y [35]

Como se menciono anteriormente, el algoritmo de Denavit-Hartenberg permite conocer la posición y orientación de la pinza del brazo robótico con respecto a un sistema de referencia colocado en la base del mismo. Según la representación D-H, escogiendo adecuadamente los sistemas de coordenadas asociados a cada eslabón es posible pasar de uno al siguiente mediante 4 transformaciones básicas que dependen exclusivamente de las características geométricas del eslabón.

Estas transformaciones básicas consisten en una sucesión de rotaciones y traslaciones que permiten relacionar el sistema de referencia del elemento *i* con el sistema del elemento *i-1*. Las transformaciones en cuestión son las siguientes:

a) Rotación alrededor del eje $\mathbf{z}_{i-1}$ un ángulo θ_i.
b) Traslación a lo largo de $\mathbf{z}_{i-1}$ una distancia d_i: un vector $\mathbf{d}_i(0,0, d_i)$.
c) Traslación a lo largo de $\mathbf{x}_{i-1}$ una distancia a_i: un vector $a_i(0,0, \mathbf{a}_i)$.
d) Rotación alrededor del eje $\mathbf{x}_{i-1}$ un ángulo α_i.

Dado que producto de las matrices no es conmutativo, las transformaciones se han de realizar en el orden indicado. De este modo se tiene que:

$$ {}^{i-1}A_i = R_{z,\theta_i}\, Trasl_{z,d_i}\; Trasl_{x,a_i}\, R_{x,\alpha_i} \tag{4} $$

$$ {}^{i-1}A_i = \begin{bmatrix} C\theta_i & -S\theta_i & 0 & 0 \\ S\theta_i & C\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\alpha_i & -S\alpha_i & 0 \\ 0 & S\alpha_i & C\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{i-1}A_i = \begin{bmatrix} C\theta_i & -C\alpha_i \cdot S\theta_i & S\alpha_i \cdot S\theta_i & a_i \cdot C\theta_i \\ S\theta_i & C\alpha_i \cdot C\theta_i & -S\alpha_i \cdot C\theta_i & a_i \cdot S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5} $$

donde θ_i, a_i, d_i, α_i son los parámetros D-H del eslabón *i* (ver figura 5.8). De este modo bastó con identificar los parámetros mencionados para obtener la Matriz ${}^{i-1}A_i$ y relacionar así todos y cada unos de los eslabones del robot RV-2AJ de Mitsubishi. Estos 4 parámetros D-H dependen únicamente de las características geométricas de cada eslabón y de las articulaciones que le unen con el anterior y siguiente. En concreto estos parámetros representan:

θ_i: Es el ángulo que forman $\mathbf{x}_{i-1}$ y $\mathbf{x}_i$ medido en un plano perpendicular al eje $\mathbf{z}_{i-1}$, utilizando la regla de la mano derecha. Se trata de un parámetro variable en articulaciones giratorias.

a_i: Es la distancia a lo largo del eje $\mathbf{x}_i$ que va desde la intersección del eje $\mathbf{z}_{i-1}$ con el eje $\mathbf{x}_i$ hasta el origen del sistema i-ésimo, en el caso de articulaciones giratorias. En el caso de articulaciones prismáticas, se calcula como la distancia más corta entre los ejes $\mathbf{z}_{i-1}$ y $\mathbf{z}_i$.

d_i: Es la distancia a lo largo del eje $\mathbf{z}_{i-1}$ desde el origen del sistema de coordenadas (i-1)-ésimo hasta la intersección del eje $\mathbf{z}_{i-1}$ con el eje $\mathbf{x}_i$. Se trata de un parámetro variable en articulaciones prismáticas.

α_i: Es el ángulo de separación del eje $\mathbf{z}_{i-1}$ y el eje $\mathbf{z}_i$, medido en un plano perpendicular al eje $\mathbf{x}_i$, utilizado la regla de la mano derecha.

Nota: Cuando se refiere a articulaciones, se refiere comúnmente a dos tipos:
- Articulaciones prismáticas: También conocida como junta deslizante, posibilita a un eslabón deslizarse en línea recta sobre otro. Permite realizar solo traslaciones lineales, de avance o retroceso.
- Articulaciones giratorias: Se considera el caso de un grado de libertad, toma la forma de una bisagra entre un eslabón y el próximo. Permite realizar solo un movimiento de giro.

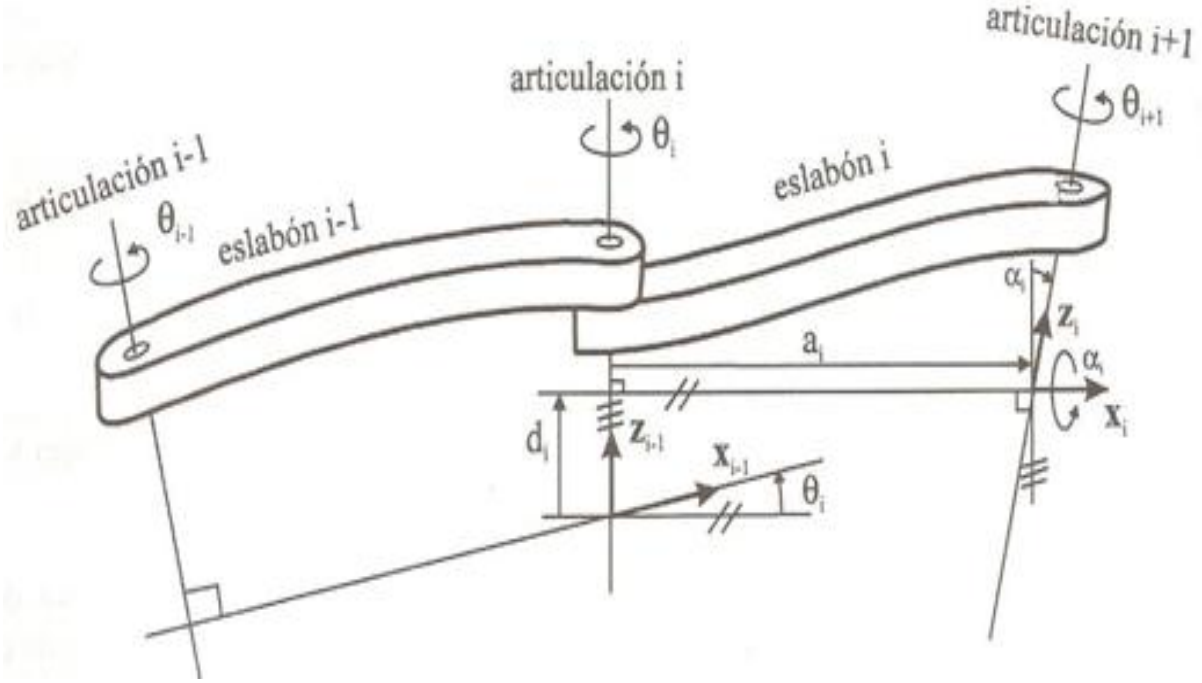


Figura 5.8 Parámetros D-H para un eslabón giratorio [22].

Para que la matriz $^{i-1}A_i$, definida en la ecuación 5 relacione los sistemas S_i y S_{i-1} es necesario que los sistemas se hayan escogido de acuerdo a unas determinadas normas que junto con la definición de los 4 parámetros de Denavit-Hartenberg conforman el siguiente algoritmo para la resolución del problema cinemático directo.

a) Numerar los eslabones de 1 a n. Se enumera como eslabón 0 a la base fija de la cadena.
b) Numerar cada articulación de 1 (empezando por el primer grado de libertad) a n.

c) Localizar el eje de cada articulación. Si esta es rotativa, el eje será su propio eje de giro. Si es prismática, será el eje a lo largo del cual se produce el desplazamiento.
d) Para i de 0 a n-1 situar el eje $\mathbf{z}_i$ sobre el eje de la articulación i+1.
e) Situar el origen del sistema de la base S_0 en cualquier punto del eje $\mathbf{z}_0$. Los ejes $\mathbf{x}_0$ e $\mathbf{y}_0$ se situarán de modo que formen un sistema dextrógiro con $\mathbf{z}_0$.
f) Para i de 1 a n-1, situar el sistema S_i (solidario al eslabón i) en la intersección del eje $\mathbf{z}_i$ con la línea normal común a $\mathbf{z}_{i-1}$ y $\mathbf{z}_i$. Si ambos ejes se cortasen se situaría S_i en el punto de corte. Si fuesen paralelos S_i se sitúa en la articulación i+1.
g) Situar $\mathbf{x}_i$ en la línea normal común a $\mathbf{z}_{i-1}$ y $\mathbf{z}_i$.
h) Situar $\mathbf{y}_i$ de modo que forme un sistema dextrógiro con $\mathbf{x}_i$ y $\mathbf{z}_i$.
i) Situar el sistema S_n en el extremo del robot de modo que $\mathbf{z}_0$ coincida con la dirección de $\mathbf{z}_{n-1}$ y $\mathbf{x}_n$ sea normal a $\mathbf{z}_{n-1}$ y $\mathbf{z}_n$.
j) Obtener θ_i como el ángulo que hay que girar en torno a $\mathbf{z}_{i-1}$ para que $\mathbf{x}_{i-1}$ y $\mathbf{x}_i$ queden paralelos.
k) Obtener d_i como la distancia, medida a lo largo de $\mathbf{z}_{i-1}$, que habría que desplazar S_{i-1} para que $\mathbf{x}_i$ y $\mathbf{x}_{i-1}$ quedasen alineados.
l) Obtener a_i como la distancia medida a lo largo de $\mathbf{x}_i$ (que ahora coincidiría con $\mathbf{x}_{i-1}$) que habría que desplazar el nuevo S_{i-1} para que su origen coincidiese con S_i.
m) Obtener α_i como el ángulo que habría que girar en torno a $\mathbf{x}_i$ (que ahora coincidiría con $\mathbf{x}_{i-1}$) para que el nuevo S_{i-1} coincidiese totalmente con S_i.
n) Obtener las matrices de transformación ${}^{i-1}A_i$ definidas en (5).
o) Obtener la matriz de transformación que relaciona el sistema de la base con el extremo del robot $T = {}^0A_1, {}^1A_2 \dots {}^{n-1}A_n$.
p) La matriz **T** define la orientación (submatriz de rotación) y posición (submatriz de traslación) del extremo referido a la base en función de las n coordenadas articulares.

En la figura 5.9 se aprecia las medidas de cada eslabón que forma el brazo mientras que en la figura 5.10 se puede ver la aplicación de cada paso del algoritmo de Denavit-Hartenberg al brazo robótico RV-2AJ. Con ambas figuras se construyó la tabla 5.2 que corresponde a la tabla de los parámetros D-H y cuyos datos son los que se emplean en la matriz (5) para poder obtener la matriz de transformación homogénea **T** final del brazo robótico.

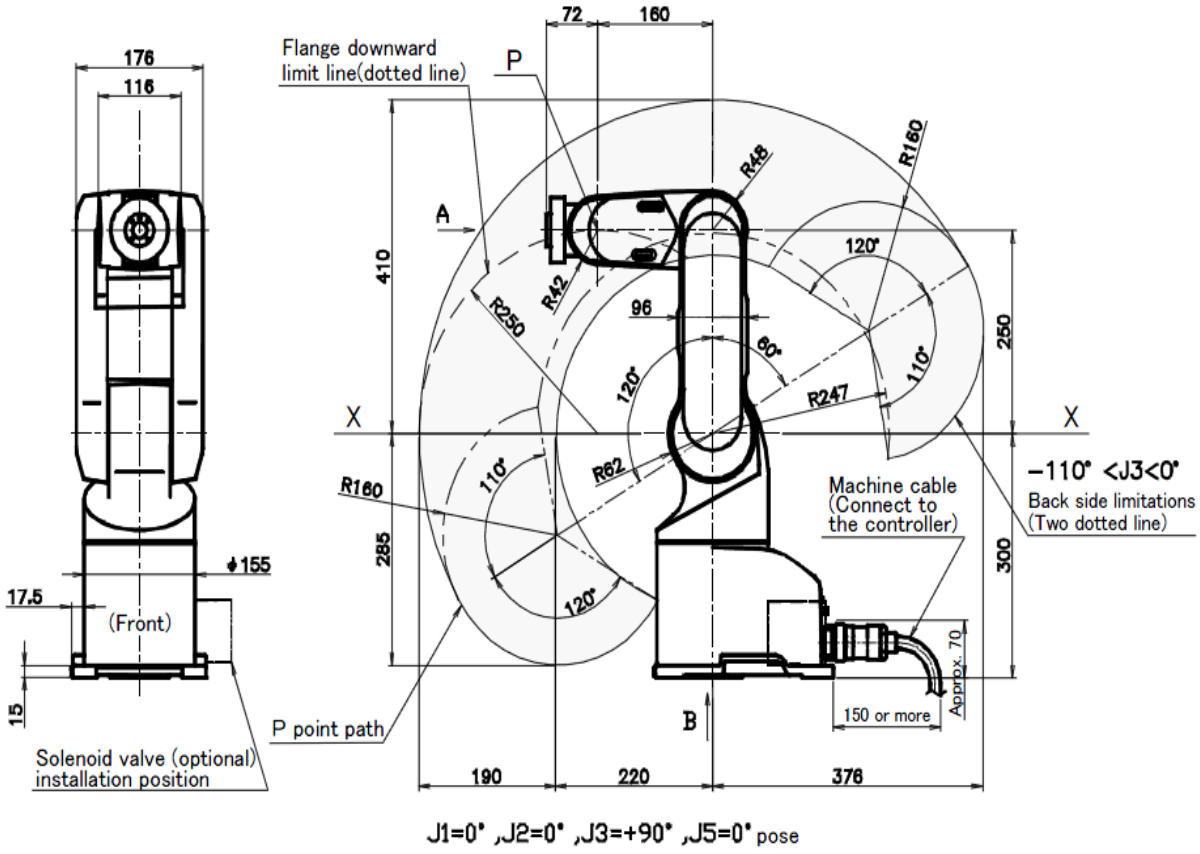


Figura 5.9 Aplicación del algoritmo de D-H al robot RV-2AJ de Mitsubishi.[34]

Cabe destacar que en esta figura 5.9 no aparece la pinza que sería el sistema S_n, es decir el último eslabón porque el único parámetro que es diferente de cero es el $\mathbf{d}_6$, que corresponde a la longitud de la pinza.

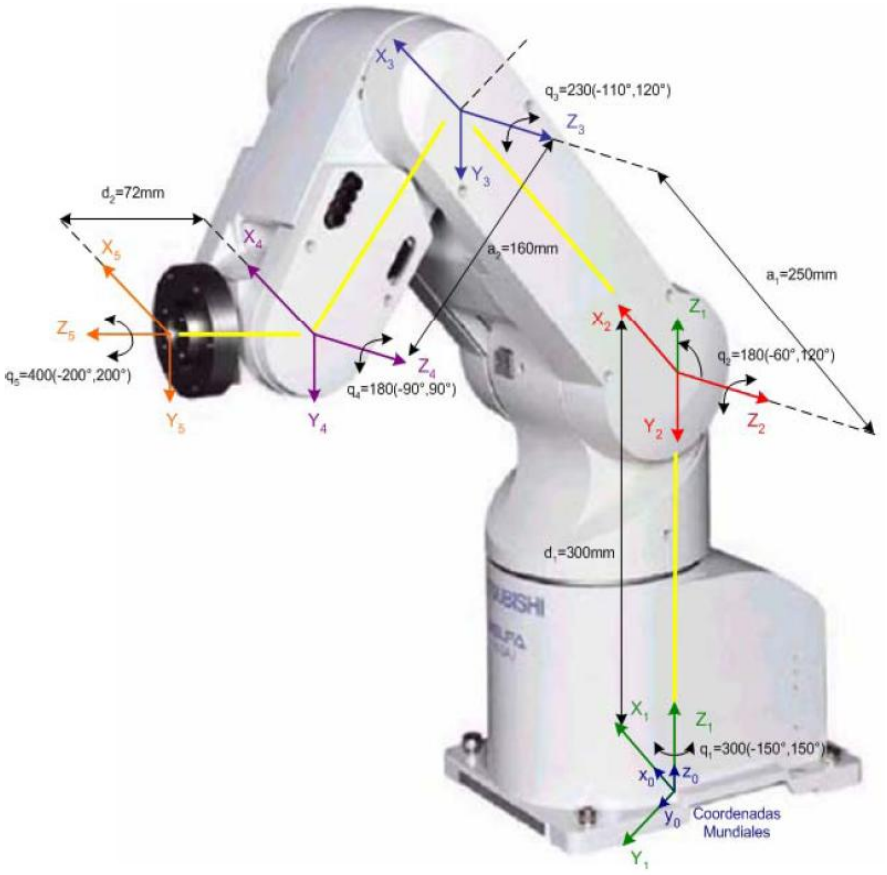


Figura 5.10 Aplicación del algoritmo de D-H al robot RV-2AJ de Mitsubishi.[30]

Tabla 5.2 Parámetros D-H para el brazo robótico RV-2AJ de la figura 5.8.

Articulación	θ_i (rad)	α_1 (rad)	a_i (mm)	d_i (mm)
1 (J1)	θ_1	-90	0	0,3
2 (J2)	θ2-90º	0	0,25	0
3 (J3)	θ3	0	0,16	0
4 (J5)	θ4+90º	90	0	0
5 (J6)	θ5	0	0	0,72
6 (Pinza)	0	0	0	0,123

Finalmente, si se aplica la formula 5 con los datos de la tabla 5.2 se obtienen los siguientes resultados:

$$ {}^{0}A_1 = \begin{bmatrix} C_1 & 0 & -S_1 & 0 \\ S_1 & 0 & C_1 & 0 \\ 0 & -1 & 0 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{1}A_2 = \begin{bmatrix} C_2 & -S_2 & 0 & a_2 \cdot C_2 \\ S_2 & C_2 & 0 & a_2 \cdot S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{2}A_3 = \begin{bmatrix} C_3 & -S_3 & 0 & a_3 \cdot C_3 \\ S_3 & C_3 & 0 & a_3 \cdot S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{3}A_4 = \begin{bmatrix} C_4 & 0 & S_4 & 0 \\ S_4 & 0 & -C_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{4}A_5 = \begin{bmatrix} C_5 & -S_5 & 0 & 0 \\ S_5 & C_5 & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{5}A_6 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

donde:

$$ C\alpha_i = \cos(\alpha_i) $$
$$ C\theta_i = \cos(\theta_i) = C_i $$
$$ S\alpha_i = \sin(\alpha_i) $$
$$ S\theta_i = \sin(\theta_i) = S_i $$

Para obtener la matriz de transformación homogénea **T** se aplicó la formula (6) que corresponde al paso o) del algoritmo de Denavit-Hartenberg:

$$ T = {}^{0}A_1 \cdot {}^{1}A_2 \cdot {}^{2}A_3 \cdot {}^{3}A_4 \cdot {}^{4}A_5 \cdot {}^{5}A_6 \qquad (6) $$

$$ T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

n_x = ((C1*C2*C3-C1*S2*S3)*C4+(-C1*C2*S3-C1*S2*C3)*S4)*C5-S1*S5
n_y = ((S1*C2*C3-S1*S2*S3)*C4+(-S1*C2*S3-S1*S2*C3)*S4)*C5+C1*S5
n_z = [((-S2*C3-C2*S3)*C4+(S2*S3-C2*C3)*S4)*C5

o_x = -((C1*C2*C3-C1*S2*S3)*C4+(-C1*C2*S3-C1*S2*C3)*S4)*S5-S1*C5
o_y = -((S1*C2*C3-S1*S2*S3)*C4+(-S1*C2*S3-S1*S2*C3)*S4)*S5+C1*C5

$\mathbf{o_z}$ = -((-S_2*C_3-C_2*S_3)*C_4+(S_2*S_3-C_2*C_3)*S_4)*S_5

$\mathbf{a_x}$ = (C_1*C_2*C_3-C_1*S_2*S_3)*S_4-(-C_1*C_2*S_3-C_1*S_2*C_3)*C_4
$\mathbf{a_y}$ = (S_1*C_2*C_3-S_1*S_2*S_3)*S4-(-S_1*C_2*S_3-S_1*S_2*C_3)*C_4
$\mathbf{a_z}$ = (-S_2*C_3-C_2*S_3)*S_4-(S_2*S_3-C_2*C_3)*C_4

$\mathbf{p_x}$=((C_1*C_2*C_3-C_1*S_2*S_3)*S_4-(-C_1*C_2*S_3-C_1*S_2*C_3)*C_4)*D_6+((C_1*C_2*C_3-C_1*S_2*S_3)*S_4-(-C_1*C_2*S_3-C_1*S_2*C_3)*C_4)*D_5+C_1*C_2*A_3*C_3-C_1*S_2*A_3*S_3+C_1*A_2*C_2
$\mathbf{p_y}$=((S_1*C_2*C_3-S_1*S_2*S_3)*S_4-(-S_1*C_2*S_3-S_1*S_2*C_3)*C_4)*D_6+((S_1*C_2*C_3-S_1*S_2*S_3)*S_4-(-S_1*C_2*S_3-S_1*S_2*C_3)*C_4)*D_5+S_1*C_2*A_3*C_3-S_1*S_2*A_3*S_3+S_1*A_2*C_2
$\mathbf{p_z}$=((-S_2*C_3-C_2*S_3)*S_4-(S_2*S_3-C_2*C_3)*C_4)*D_6+((-S_2*C_3-C_2*S_3)*S_4-(S_2*S_3-C_2*C_3)*C_4)*D_5-S_2*A_3*C_3-C_2*A_3*S_3-A_2*S_2+D_1

Como se aprecia en las ecuaciones anteriores, queda reflejado el valor de la posición (p_x, p_y, p_z) y orientación **(n, o, a)** del extremo del robot en función de las coordenadas articulares (θ_1, θ_2, θ_3, θ_4, θ_5, θ_6).

En cuanto a la orientación, fue necesario calcular el ángulo conocido como pitch (cabeceo) que corresponde al ángulo que posee la pinza de acuerdo al sistema de referencia establecido en la base del brazo. Este ángulo de cabeceo se obtuvo con la siguiente fórmula:

$$pitch = \theta = arcos(a_z) \tag{7}$$

Las demás componentes de los vectores (n, o, a), sirven para obtener otros datos que para la finalidad de este proyecto no fue necesario calcularlos, tal es el caso del los ángulos conocidos como (ver figura 5.11): roll (alabeo) y yaw (guiñada). Las ecuaciones de orientación **(n, o, a)** son utilizadas en caso de querer resolver el problema de la cinemática inversa del brazo robótico.

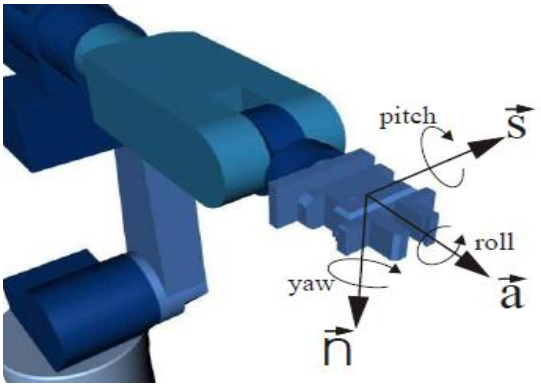


Figura 5.11 Orientación de la pinza de acuerdo al sistema de referencia en la base del brazo. [35]

5.4.1 Objetivos en la mesa y posiciones peligrosas

Una vez que se conocía la posición y orientación de la pinza se procedió a hacer restricciones de movimiento de acuerdo a su ubicación. Dichas restricciones consistieron en que la pinza no fuera a golpear ningún objeto de la mesa ni a sí mismo lo que corresponde a posiciones peligrosas (ver análisis de resultados).

Cabe destacar que el controlador del brazo robótico permite elegir entre tres métodos de posicionado para alcanzar el lugar deseado y son los siguientes:

- XYZ Jog: En este modo el brazo robótico se moverá respecto al eje de coordenadas situado en la base.
- Joint Jog: En este modo se puede mover individualmente cada eje del motor en formato polar (grados).
- XYZ Tool: En este modo el brazo robótico se moverá respeto al eje de coordenadas situado en el centro la herramienta (En este caso la pinza).

Para el presente proyecto el brazo opera en el modo Joint Jog, es decir mueve cada eje independientemente a través de la interfaz háptica alternando en un momento entre J1, J2, J3 y J5, J6. No obstante, con este tipo movimiento es casi imposible ubicar la pinza en la posición del módulo retenedor (objetivo 1), del módulo de montaje (objetivo 2) ya sea para recoger o dejar una pieza y depositarla en alguno de los 2 tubos si así se deseara. En la figura 5.12 se puede ver un diagrama de bloques de la vista superior de la ubicación de los objetivos y del brazo robótico.

La razón de la dificultad de colocar la pinza en alguno de los lugares antes mencionados radica en la precisión del operador y en la necesidad de realizar un movimiento totalmente vertical (moverse solamente en el eje Z debido a que en los objetivos 1 y 2 el margen que queda de la pieza respecto al borde cuando es sujetada con la pinza es aproximadamente 0,5 cm a cada lado) de la pinza para lo cual es necesario mover todos los ejes, lo cual resulta sumamente difícil de

realizar con la interfaz. Es por esto que se utilizo también el modo XYZ Jog. La idea en este caso fue programar condiciones de acuerdo a la posición y orientación de la pinza y si por medio de la interfaz se presionara el botón de abrir o cerrar pinza se envía la posición respectiva. Para hacer esto, además de enviar la trama antes mencionada (sección 5.3.2):

"PRN(J1_Serial,J2_Serial,J3_Serial,J4_Serial,J5_Serial,J6_Serial),M1_Pinza CR", fue necesario enviar un encabezado donde se le indica al controlador el método de posicionado, donde si el encabezado es: PRNXYZ las coordenadas que se le van a enviar corresponden a XYZ Jog y si el encabezado es PRNJOINT las coordenadas corresponden a Joint Jog.

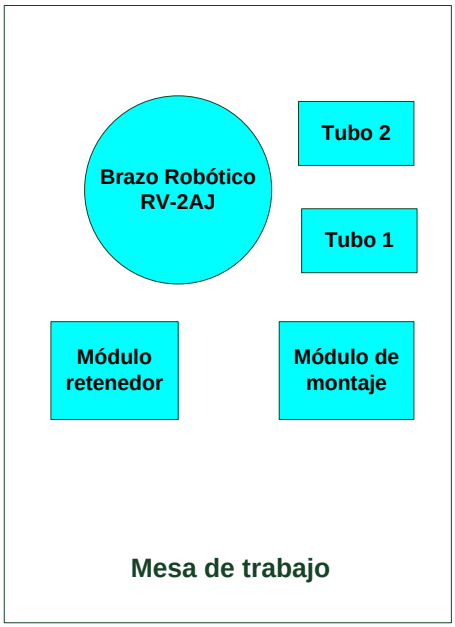


Figura 5.12 Vista superior de la estación de Robot Festo.

5.5 Programación del brazo robótico RV-2AJ

Como se menciono anteriormente el lenguaje de programación del brazo robótico de Mitsubishi RV-2AJ es el Melfa Basic IV y el programa CIROS de la empresa FESTO fue el que se utilizó tanto para compilar, simular y descargar el código que se programó.

En la figura 5.13 se puede ver el diagrama general del programa que corre en controlador del robot. Dicho programa es realmente sencillo y a continuación se explica cada paso:

- Primero se fija la velocidad en un 30% por medio de la instrucción “OVRD 30”.
- Luego se abre la pinza por medio de la instrucción “HOPEN 1”.
- Seguidamente se procede a posicionar las cinco articulaciones del brazo en la posición que se asigno como inicial: J1 = 0º, J2 = 0º, J3 = 90º; J4 = 0º, J5 = 90º y J6 = 0º. Para ello se usa la instrucción “MOV JDATA0”, donde JDATA0 es una variable de tipo Joint (articulaciones).
- Ahora se aumenta la velocidad a un 100% con el fin de que el robot se mueva lo más rápido posible ante las posiciones que se le indique producto del manejo de la interfaz.
- El siguiente paso es un ciclo donde el brazo del robot permanece esperando que le llegue un dato por el puerto serial, para esto antes el controlador CR1-571 abrió el puerto serial por medio de la instrucción “OPEN "COM1:" AS #1”.
- Se recibe la trama y se evalúa si el encabezado es igual a XYX. En caso de ser así, la trama a decodificar es la siguiente: (X, Y, Z, A, B),Pinza que corresponde al valor de la ubicación de la pinza (X, Y, Z afecta a la posición, A corresponde al ángulo de cabeceo y B corresponde al valor del eje J6) y a la señal de abrir ó cerrar la pinza. En caso de que el encabezado sea diferente de XYZ, la trama a decodificar es la siguiente: (J1, J2, J3, J4, J5, J6),Pinza que corresponde al valor de cada articulación y a la señal de

abrir ó cerrar la pinza. El procedimiento de leer del puerto serial se lleva a cabo a través de la instrucción “INPUT #1,C1”, donde C1 es un char que se va a comparar con “XYZ” para determinar el método de posicionamiento a emplear. Una vez hecho esto, se lee del puerto serial la ubicación y el estado de la pinza. Si es XYZ, la instrucción sería “INPUT #1,JPATA1,M1”, donde JPATA1 es una variable de tipo POS que contiene el valor de la posición y orientación que se estableció para alcanzar unos de los cuatro objetivos descritos anteriormente (sección 5.4.1) y M1 es un entero que en caso de ser uno provoca que la pinza se cierre y si es cero abre la pinza. Por otro lado, si C1 ≠ XYZ la instrucción sería: “INPUT #1,JDATA1,M1”, donde JDATA1 es una variable de tipo JOINT que contiene el valor del ángulo de cada articulación del brazo que se estableció a través de la interfaz y M1 es igual al caso anterior.

- El siguiente paso es enviar el valor de la fuerza a retroalimentar. En caso de que se haya recibido la orden de abrir la pinza el valor a retroalimentar tiene un valor de cero mientras que en caso de la orden haya sido de cerrar la pinza se retroalimenta un 160. Como se menciono anteriormente este valor se escogió porque representa un nivel de fuerza considerable para demostrar el objetivo del proyecto, retroalimentar fuerza. El controlador enviar la información a la PC por medio de la instrucción “PRINT #1, M2 “, donde M2 es un entero que según sea el caso va a valer cero ó 160.
- Finalmente este proceso se repite n veces, es decir hasta que se le dé stop al controlador.

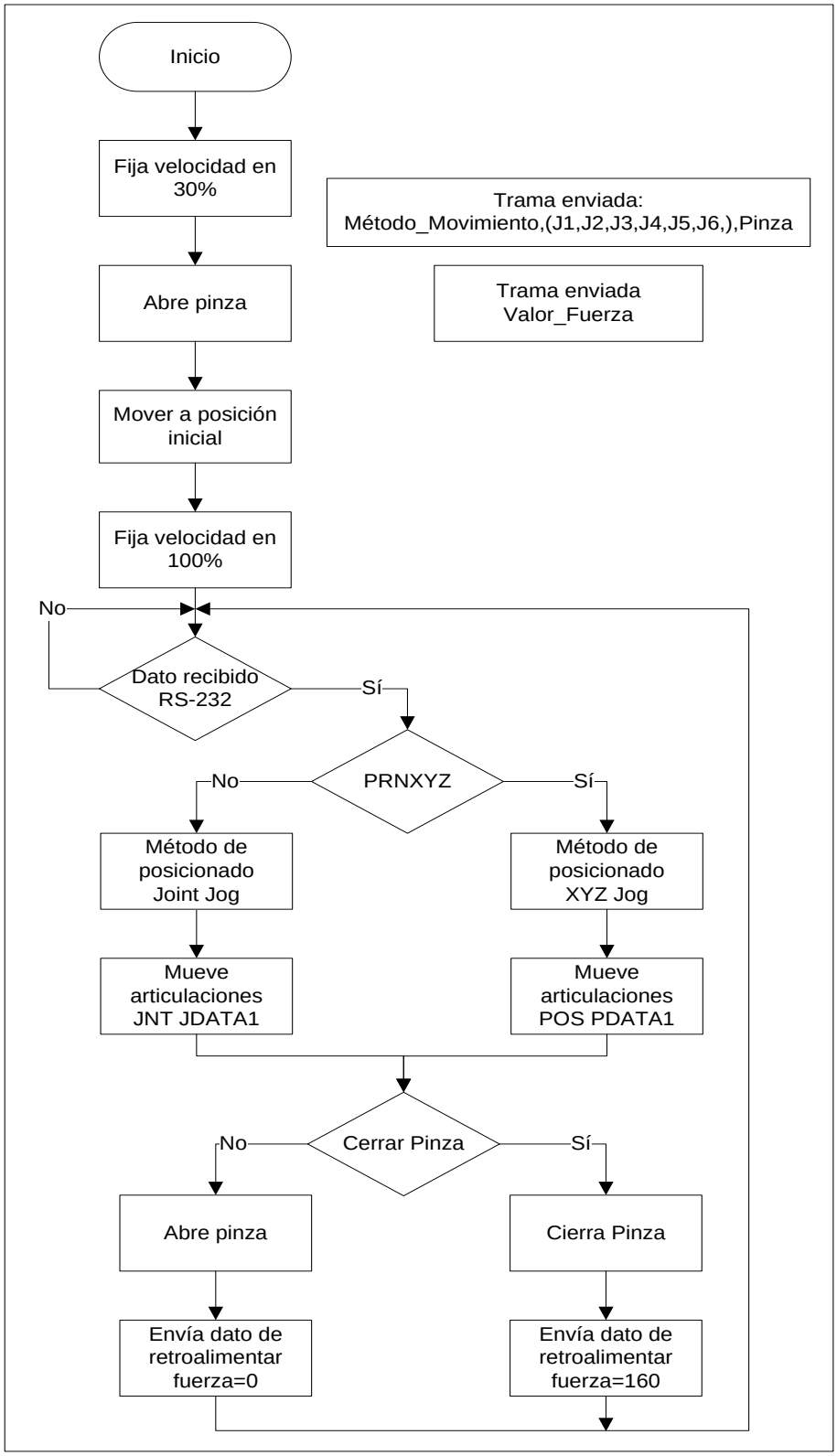


Figura 5.13 Diagrama general de la programación del brazo robótico.

5.6 Movimiento de articulaciones con la interfaz

En la figura 5.15 y 5.16 se observa el esquema para mover las articulaciones del robot. Como se ha mencionado anteriormente, por medio del hilo del HDAL de la programación se ejecuta en paralelo lo correspondiente al manejador del servo y la aplicación principal. Es en esta última donde se da el manejo de la interfaz para mover los ejes del brazo robótico. Si el botón cuatro de la interfaz no está presionado se pueden mover los ejes J1, J2 y J3 mientras que los ejes J5 y J6 permanecen inmóviles en la última posición establecida. En caso de presionar el botón cuatro ocurre todo lo contrario, los ejes J1, J2 y J3 son los que permanecen en la última posición que se les haya enviado y se procede a mover los ejes J5 y J6.

Los tres ejes de la interfaz (x, y, z) se definieron de -200 a +200 (espacio de trabajo de la aplicación). Igualmente se asigno una zona neutra que abarca los tres ejes desde -70 a +70, esto quiere decir que si el manubrio de la interfaz permanece en este rango, no se moverá ninguna articulación del brazo, como se observa en la figura 5.14.

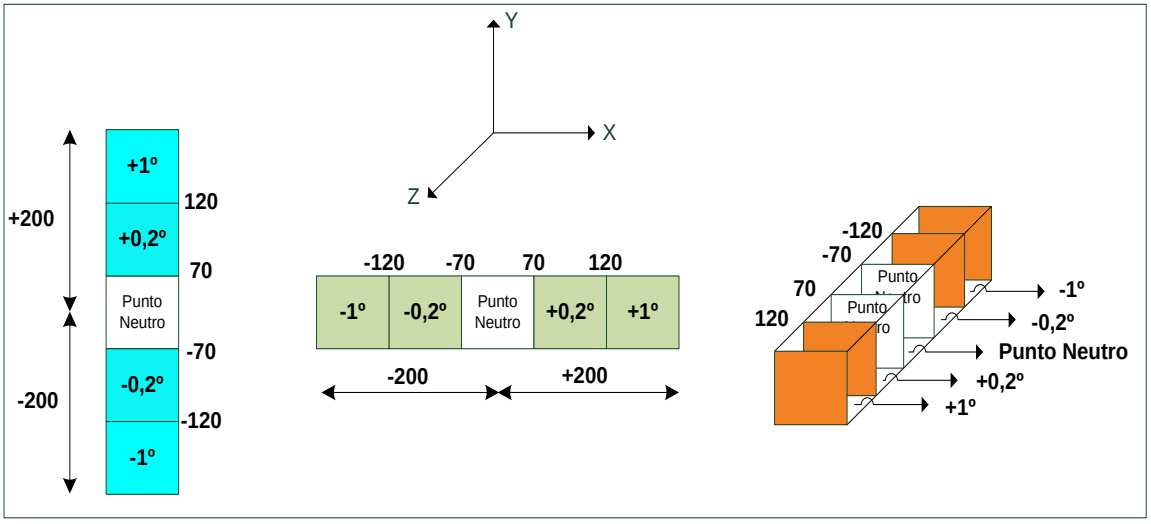


Figura 5.14 Manejo de la interfaz háptica.

En cuanto al movimiento del brazo, inicialmente se iba a enviar la posición que marcara la interfaz, por ejemplo, si en un momento el eje x de la interfaz indicaba un ángulo de -200, se enviaba y si se pasaba inmediatamente a +200, también, no obstante esto hacia difícil el manejo del brazo ya que este tipo de sistema no contaba con una zona neutra y el pulso del operador al manejar la interfaz hacía casi imposible colocar la pinza del brazo en una punto especifico. Este tipo de manejo presentaba otro problema y era que si se pasaba de -200 a +200, durante el movimiento del servo para colocar la articulación según la coordenada, la programación en la PC continuaba enviando datos y esto provocó que el buffer del controlador encargado de recibir los datos se llenara y generaba un error. Esto se pudo arreglar con un delay, no obstante el tiempo que permanecía congelado el programa provocaba que el sistema se volviera lento en cuanto a su manejo y perdiera fidelidad en los movimientos.

Es debido a lo anterior que se opto por el control del brazo que se observa en el diagrama de flujo de la figura 5.15 y 5.16, mientras que en la figura 5.14 se puede ver de otra forma el manejo de la interfaz para manipular el brazo, donde si cualquier eje de la interfaz se encuentra entre 70 y 120, la articulación correspondiente del brazo aumentara de 0,2º en 0,2º, mientras que si se encuentra entre -120 y -70 disminuirá de 0,2º en 0,2º. Por otro lado, si cualquier eje de la interfaz es mayor a 120º, se dará un aumento de 1º en 1º, mientras que si es menor a -120º disminuirá de 1º en 1º.

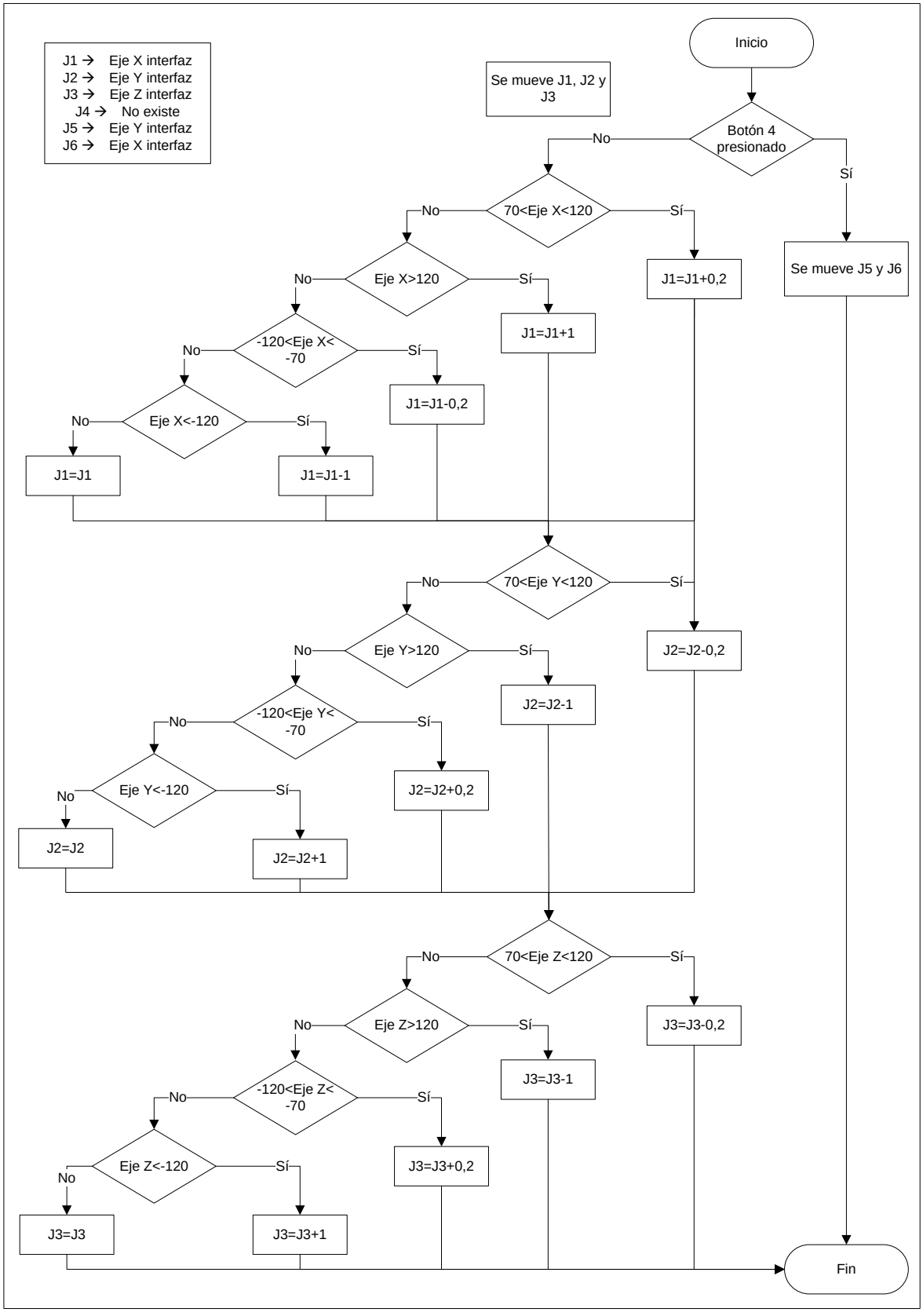


Figura 5.15 Diagrama para mover el brazo robótico con la interfaz háptica (parte No).

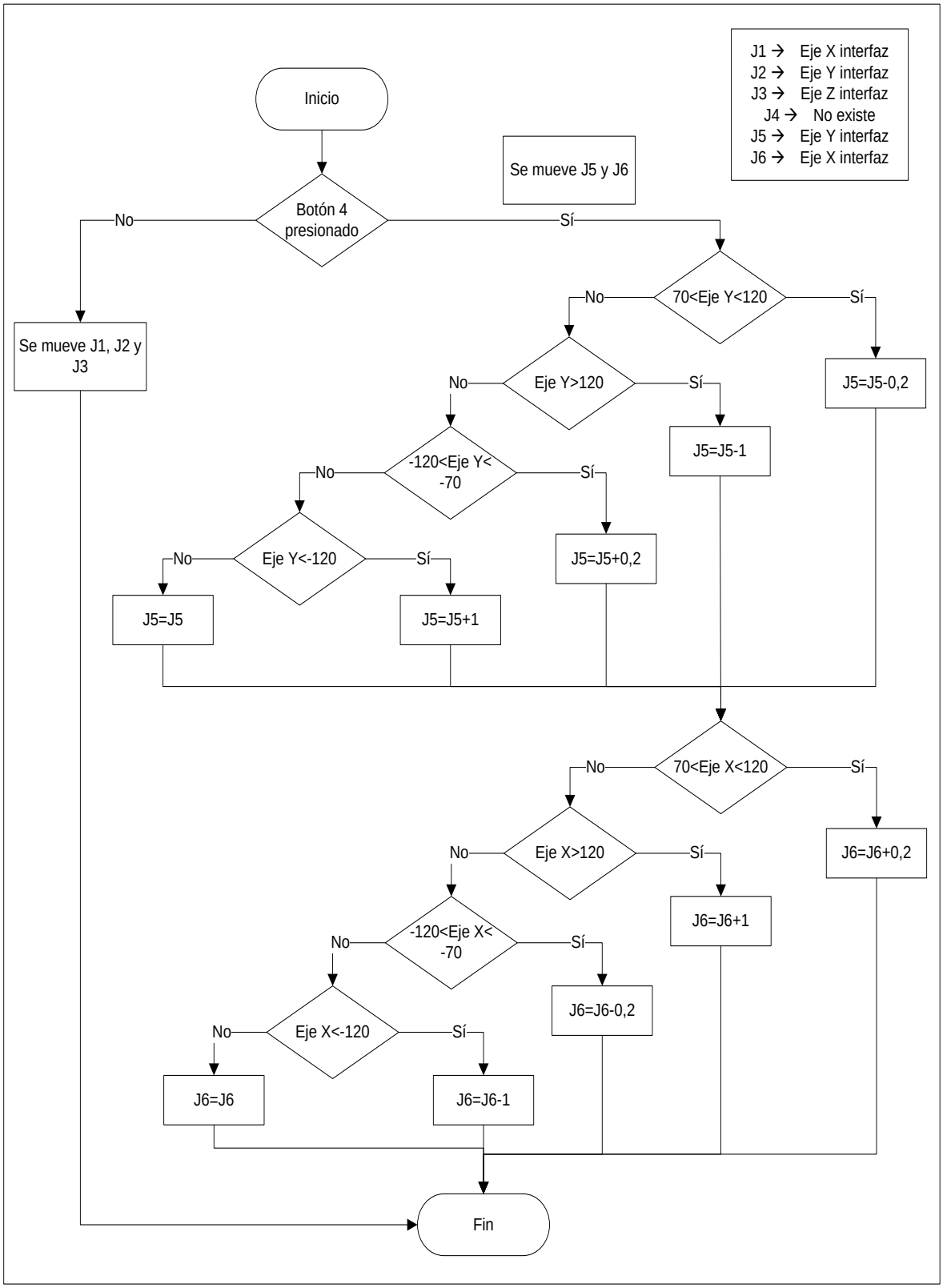


Figura 5.16 Diagrama para mover el brazo robótico con la interfaz háptica (parte Si).

Capítulo 6: Análisis de resultados

Con este capítulo se muestran los resultados producto del sistema descrito anteriormente con el que se logró implementar un sistema teleoperado con retroalimentación de fuerza a través de una interfaz háptica y un brazo robótico. Como se mencionó anteriormente, el programa principal fue hecho en C++ y corre en la computadora siendo el responsable de interactuar con la Novint Falcon tanto para procesos de inicialización, lectura de información como posición y estado de botones, retroalimentar fuerza a los motores de la interfaz para experimentar una sensación háptica, así como lo relacionado al algoritmo de Denavit-Hartenberg y al puente que representa la comunicación serial de la computadora con la estación de robot Festo.

Las rutinas implementadas para la comunicación entre la computadora y el controlador CR1-571 del robot de Mitsubishi fueron probadas inicialmente con un PIC 18F2550 dado la facilidad que brinda el microcontrolador al poseer un puerto serial y un par de puertos PWM con los que se simuló dos GD a través de dos servo-motores. Con esta prueba se concluyo que la única limitante para comunicar la interfaz con otro sistema que posea RS-232 es la velocidad con la que este procese los datos y pueda volver a leer la información proveniente de la PC. Lo anterior fue porque si desde la computadora se enviaban los datos a la velocidad a la que el ordenador ejecuta el programa hecho en C++, el PIC era incapaz de leer la información por lo que fue necesario agregar delays y ajustarse a la velocidad de procesamiento del microcontrolador.

Una vez que se establecido la comunicación bidireccional PC-PIC se procedió a probar con el brazo robótico. En este caso nuevamente fue necesario insertar delays ya que la velocidad con la que se envía la información de la PC al controlador del brazo debe de ser igual o mayor al tiempo que dure el programa en ejecutarse más un delay de 15 ms. De no incluirse este delay el buffer del controlador encargado de recibir los datos se llena y genera un error “6500” que

según el manual de troubleshooting del controlador indica un error de lectura debido a que no fue posible abrir el puerto serial.

Por otro lado, se realizaron cuatros pruebas finales de quince minutos cada donde cada una las instrucciones de movimiento que se hicieron con la interfaz háptica tuvieron un efecto de accionar las articulaciones del robot en un 100% de los casos, es decir siempre que cualquier eje de la interfaz se ubicara en un valor mayor a 70 o menor a -70 el brazo se movió ya fuera que se presionara el botón cuatro del Falcon (mueve J5 y J6) o que no se presionara (mueve los ejes J1, J2 y J3). De igual manera, durante las pruebas anteriores se probó la comunicación del controlador del brazo a la PC obteniendo un 100% de efectividad en la respuesta. Esto se logró cerrando y abriendo la pinza, ya que al hacer esto el controlador contesta con un valor de fuerza que se ve reflejado en un mayor esfuerzo para accionar la interfaz.

Otra prueba que se hizo, fue enviar posiciones específicas a través del programa en C++ para los dos métodos de posicionado utilizados en este proyecto (XYZ Jog y Joint Jog). En ambos casos la precisión fue de un 100% de exactitud. El valor enviado por el puerto serial de la PC se probó contra el valor medido por el software CIROS y la botonera (TeachBox). En el caso del método Joint Jog que puede mover individualmente cada eje del motor en formato polar (grados) cada articulación tomo el valor que se programó mientras que con el método XYZ Jog que mueve la pinza respecto al eje de coordenadas situado en la base se logró ubicarla en la posición y con la orientación indicada. Cabe destacar que era de esperar este tipo de exactitud al manejar al brazo robótico, dado que los manipuladores industriales tienen que tener la característica de gozar de una gran fidelidad al momento de un movimiento, en particular el RV-2AJ posee una repetitividad en la precisión de posicionado de ± 0.02 mm según se indica en la hoja de datos del fabricante.

En lo que respecta a la retroalimentación de fuerza se logró experimentar está sensación de dos maneras. Los resultados del primer caso se muestran en la tabla 6.1 donde se observa el efecto de manipular el brazo, es decir en caso de que se dé la señal de abrir la pinza ningún eje de la interfaz sufrirá efecto háptico al mover cualquiera de los cinco GD del brazo. Caso contrario sucede si se dá la señal de cerrar la pinza, ya que los tres ejes del Falcon presentarán un mayor esfuerzo al operador al tratar de accionar cualquier articulación del robot. La idea en este caso fue transmitir al usuario el esfuerzo que implica mover algún objeto con una masa asociada.

Tabla 6.1 Efecto de la fuerza al manipular la pinza del brazo robótico.

Acción	Nivel de fuerza	Comentario
Abrir pinza	Nulo	Al presionar el botón 2 de la interfaz
Cerrar pinza	Medio	Al presionar el botón 3 de la interfaz
Mover la interfaz	Bloqueado	Al situar alguna articulación en una posición final

El segundo caso en que se retroalimento fuerza, fue al tratar de mover cualquiera de los cinco GD del brazo robótico que ya estuvieran en una posición extrema, es decir que ya no podían girar más a través de su eje. Estos resultados se observan en la tabla 6.2 donde cada valor extremo de una articulación es diferente dado que no todos los ejes tienen el mismo alcance como se mencionó anteriormente en el marco teórico.

Tabla 6.2 Efecto de la fuerza al mover cualquier GD del brazo robótico.

Articulación	Posición Articulación	Movimiento	Efecto Fuerza
J1	150	J1 ≥ 150	Bloquear
J1	150	J1 < 150	Nulo
J1	-150	J1 > -150	Nulo
J1	-150	J1 ≤ -150	Bloquear
J2	120	J2 ≥ 120	Bloquear
J2	120	J2 < 120	Nulo
J2	-60	J2 > -60	Nulo
J2	-60	J2 ≤ -60	Bloquear
J3	120	J3 ≥ 120	Bloquear
J3	120	J3 < 120	Nulo
J3	-110	J3 > -110	Nulo
J3	-110	J3 ≤ -110	Bloquear
J5	90	J5 ≥ 90	Bloquear
J5	90	J5 < 90	Nulo
J5	-90	J5 > -90	Nulo
J5	-90	J5 ≤ -90	Bloquear
J6	200	J6 ≥ 200	Bloquear
J6	200	J6 < 200	Nulo
J6	-200	J6 > -200	Nulo
J6	-200	J6 ≤ -200	Bloquear

En la retroalimentación de fuerza el controlador del brazo robótico (CR1-571) es el responsable de que esto suceda para el caso del efecto de la pinza, ya que es el responsable decodificar la trama que recibe de la pinza y contestar con el valor de fuerza que se va a aplicar a los ejes de la interfaz.

En cuanto al algoritmo de Denavit-Hartenberg, se logró validar los datos calculados con la ecuación de la matriz número 5 en lo que respecta a posición (p_x, p_y, p_z) y a la orientación (ángulo de cabeceo) por medio del programa CIROS que también usa este algoritmo para modelar el brazo robótico. Los valores que se usaron para la simulación se aprecian en la tabla 6.1 mientras que en la tabla 6.2 se compara el valor obtenido con la matriz de transformación homogénea contra el valor que provee el software CIROS (se tomó este valor como teórico para calcular porcentajes de error).

Tabla 6.3 Valores de las articulaciones para probar el algoritmo de Denavit-Hartenberg.

Articulación	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5	Prueba 6
J1 (º)	0	0	0	125,1	-145,1	-110,2
J2 (º)	0	0	0	-32,3	110,3	-50,3
J3 (º)	0	90	90	119,9	-105,1	110,3
J4 (º)	0	0	0	0	0	0
J5 (º)	0	0	90	-85	90	-85,2
J6 (º)	0	0	0	-123,2	125,2	175,4

Tabla 6.4 Resultados de aplicar el algoritmo de Denavit-Hartenberg.

		x (mm)	**y (mm)**	**z (mm)**	**Cabeceo (º)**
Prueba 1	CIROS	0,00	0,00	905,00	0,00
	Matriz T	0,00	0,00	905,00	0,00
	Error (%)	0,00	0,00	0,00	0,00
Prueba 2	CIROS	355,00	0,00	550,00	90,00
	Matriz T	355,00	0,00	550,00	90,00
	Error	0,00	0,00	0,00	0,00
Prueba 3	CIROS	160,00	0,00	355,00	180,00
	Matriz T	160,00	0,00	355,00	180,00
	Error	0,00	0,00	0,00	0,00
Prueba 4	CIROS	-20,20	28,70	712,70	2,60
	Matriz T	-20,19	28,73	712,81	2,60
	Error	0,05	-0,10	-0,02	0,00
Prueba 5	CIROS	-363,50	-253,30	355,20	95,20
	Matriz T	-363,47	-253,56	354,93	95,20
	Error	0,01	-0,10	0,08	0,00
Prueba 6	CIROS	47,32	128,33	716,81	25,20
	Matriz T	47,24	128,40	716,13	25,20
	Error	0,17	-0,05	0,09	0,00

Como se puede en la tabla 6.2 el porcentaje de error es mínimo, confirmando así que el algoritmo de Denavit-Hartenberg y que la tabla 5.2 de los parámetros de D-H son correctos. El peor de los casos de acuerdo a la tabla 6.4 corresponde a un error del 0.17% en el caso de la prueba número seis para el valor de “x” que aún así se considera totalmente despreciable.

Una vez que se obtuvo la posición y la orientación de la pinza se procedió a determinar cuáles son posiciones peligrosas y cuáles son las condiciones para

acceder alguno de los objetivos. El brazo robótico se trato como un cilindro donde de acuerdo al valor del radio “r” se permitió el movimiento como se aprecia en la figura 6.1. El radio se calculo con la siguiente ecuación:

$$r = \sqrt{(x^2) + (y^2)} \tag{7}$$

donde el valor mínimo permitido del radio fue de 135 mm, ya que en caso de ser menor la pinza corría el riesgo de chochar con los eslabones y la base que forman el brazo robótico.

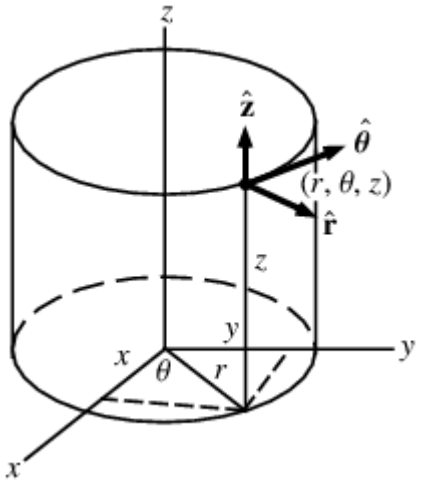


Figura 6.1 Diagrama para evitar que la pinza se golpee a sí mismo. [36]

Una vez hecho esto, se procedió a hacer un barrido la articulación de J1 desde -150º hasta +150 º con el fin de determinar la altura mínima a la que la pinza podía llegar sin golpear la mesa y para poder llegar a la posición de los objetivos antes mencionados. El resultado de este análisis se resume en la tabla 6.5.

Tabla 6.5 Condición para evitar choques de la pinza contra la mesa y consigo mismo.

J1 (º)	Radio (mm)	Z (mm)
-150 ≤ J1 < 33	r ≤ 135	Z ≤ 140
33 ≤ J1 < 136	r ≤ 135	Z ≤ 240
J1 ≥ 136	r ≤ 135	Z ≤ 140

De acuerdo con la tabla anterior si el J1 se encuentra entre -150º y 33º la pinza solo puede ubicarse a una altura que se mide desde la base hacía arriba de

140 mm mientras que si está entre 33º y 136º solo se permite bajar una altura de 240 mm debido a que es la parte de los tubos como se aprecia en la figura 6.2. Una vez que se pasa la sección de los tubos de descargue, la altura permitida es nuevamente 140 mm.

En el caso de los objetivos establecidos para este proyecto, las condiciones que se deben de dar para poder alcanzarlos se resumen en la tabla 6.4. Cabe destacar que entre todas las condiciones existe la operación AND, es decir, todas deben de cumplirse o de lo contrario no se puede alcanzar la posición establecida como objetivo.

Tabla 6.6 Condición para alcanzar alguno de los 4 objetivos de estación de robot Festo.

Objetivo	Cabeceo (º)	J1 (º)	Z (mm)	Requisito
Módulo retenedor	θ ≥ 160	-9 ≤ J1 < 8	Z ≤ 140	Presionar botón 2 ó 3 interfaz
Módulo montaje	θ ≥ 160	20 ≤ J1 < 33	Z ≤ 110	Presionar botón 2 ó 3 interfaz
Tubo 1	θ ≥ 160	54 ≤ J1 < 74	Z ≤ 260	Presionar botón 2 interfaz
Tubo 1	θ ≥ 160	96 ≤ J1 < 116	Z ≤ 260	Presionar botón 2 interfaz

Otro punto importante a destacar es que para el módulo de montaje se permitió que la pinza tuviera una altura menor porque este objetivo se encuentra en nivel más bajo de la mesa de trabajo.

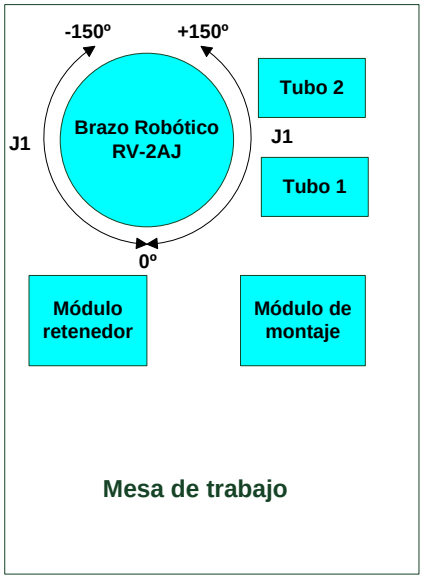


Figura 6.2 Vista superior de la estación de Robot Festo.

Capítulo 7: Conclusiones y recomendaciones

6.1 Conclusiones

- ✓ Se logró programar la interfaz háptica para controlar las cinco articulaciones del robot Mitsubishi, alternando entre mover las tres primeras articulaciones en un momento y las otras dos restantes al presionar el botón derecho de la interfaz háptica.

- ✓ El robot Mitsubishi es capaz de recibir y enviar datos mediante el puerto serial de programación e interactuar con otros dispositivos tales como una PC.

- ✓ Se logró retroalimentar una sensación de fuerza al mover un objeto mediante la interfaz háptica basando en el modelo del resorte.

- ✓ Se bloqueo la interfaz háptica en los ejes que correspondían a una articulación del brazo robótico que ya no podía girar más.

- ✓ Se alcanzó un éxito del 100% en la respuesta del brazo robótico ante movimientos de la interfaz si se envían datos cada 15 ms o en más tiempo.

- ✓ Es posible lograr una precisión menor de 0,5º en la colocación de la pinza del brazo robótico en una posición determinada.

- ✓ Se logró obtener la posición y orientación (cinemática directa) del brazo robótico RV-2AJ de Mitsubishi utilizando el resultado del algoritmo de Denavit-Hartenberg.

6.2 Recomendaciones

Es importante mencionar mejoras que se le puedan hacer al proyecto, o bien que forme parte de otro proyecto, de ahí surgen las siguientes observaciones y recomendaciones.

- ✓ Utilizar el sensor que trae el brazo robótico en la pinza que distingue piezas por el color (negra/no negra) para determinar si la pinza está manipulando algún objeto o si simplemente se encuentra cerrada pero sin ninguna pieza.

- ✓ Agregar un PLC al controlador CR1-571 a través de una tarjeta adicional conectada al controlador CR1-571 con el fin de tener una entrada analógica para poder incorporar un sensor de peso o de presión y así tener una mejor percepción de la realidad a través de la fuerza que se retroalimente a la interfaz.

- ✓ Sustituir la PC por algún tipo de sistema embebido capaz de realizar la tarea de comunicar la interfaz con el controlador del brazo.

7 Bibliografía

[1] Escuela de ingeniería electrónica. *Información de la carrera.* [En línea]. Última modificación: 2008. Última visita: 26 de mayo del 2010. URL: http://www.ie.itcr.ac.cr/

[2] Novint. Novint Falcon. [En línea]. Última modificación: 2010. Última visita: 26 de mayo del 2010. URL: http://home.novint.com/novint/company.php

[3] Festo. Estación de robot (Calidad Industrial). [En línea]. Última modificación: 2010. Última visita: 26 de mayo del 2010. URL: http://www.festo-didactic.com/int-es/learning-systems/mps-sistema-de-producci-n-modular/estaciones/estaci-n-de-robot-calidad-industrial.htm?fbid=aW50LmVzLjU1Ny4xNC4xOC42MDYuMzk1MA&basket=add&vid=5920

[4] Novint. *Novint Falcon Technical Specs*. [En línea]. Última modificación: 2010. Última visita: 25 de agosto del 2010.
URL: http://home.novint.com/products/technical_specs.php

[5] Nuño E. y Basañez L. Teleoperación: técnicas, aplicaciones, entorno sensorial y teleoperación inteligente. PhD thesis, Universidad Politécnica de Cataluña, Instituto de Organización y Control de Sistemas Industriales, Barcelona, España, Abril 2004. [En línea].
URL: http://upcommons.upc.edu/e-prints/bitstream/2117/570/1/IOC-DT-P-2004-05.pdf

[6] Padilla Oscar. Manipulador teleoperado inalámbricamente. Tesis profesional, Universidad de las Américas Puebla Escuela de Ingeniería y Ciencias Departamento de Computación, Electrónica y Mecatrónica. Cholula, Puebla, México a 15 de mayo de 2008. [En línea].
URL: http://catarina.udlap.mx/u_dl_a/tales/documentos/lmt/padilla_m_o/index.html

[7] Alvarado D, Butrón M, Díaz D, González L, Vázquez I, Viveros S. Teleoperación Háptica de Brazo Manipulador Laboratorio De Investigación En Energías Renovables, Universidad Nacional Autónoma De México, 30 De Agosto Al 3 De Septiembre 2010. [En línea].
URL: http://www.slideshare.net/newtonrules/ir-18

[8] Ollero A. Robótica: Manipuladores u robots móviles. Marcombo Alfaomega, Barcelona, España, 2001.

[9] González Andrés. Telerobótica de Asistencia - Aplicaciones en el Ámbito Doméstico para Personas Discapacitadas y Personas de la Tercera Edad. Doctorado en Control, Visión y Robótica Universidad Politécnica de Cataluña,

Curso de Monitorización, Inspección y Supervisión de Procesos 2006-2007. [En línea]. URL: http://bibliotecnica.upc.es/e-portals/tid/arxius/articles/article_6.pdf

[10] Ollero A. y García A. Gómez M. Teleoperación y Telerobótica. PEARSON Prentice Hall, CEA Comité Español de Automática, Madrid, España, 2006.

[11] Pinto María. Análisis e implementación de una interfaz háptica en entornos virtuales. Magíster en ingeniería, Universidad Nacional de Colombia, Bogota, Facultad de Ingeniería, 2009. [En línea].
URL: http://www.bdigital.unal.edu.co/1761/1/280201.2009.pdf

[12] J.M. Sabater. Desarrollo de una Interfaz Kinestesica Paralela y Experimentación en Control de Sistemas Hápticos y Teleoperados. PhD thesis, Universidad Miguel Hernández, Elche, Alicante España, 30 Junio 2003. [En línea]. URL: ttp://isa.umh.es/personal/j.sabater/tesis/documentos/Phd_Sabater_ebook.pdf

[13] Rubén Diego. Interfaces Hápticas: Futuro de Interfaces Naturales, Tesis de Grado, Universidad Católica de Salta, Facultad de Ingeniería, martes 28 de octubre de 2008. [En línea]. URL: http://haptica.blogspot.com/2008/10/interfaces-hpticas.html

[14] Grigore Burdea. Haptic Feedback for Virtual Reality, Rutgers, The State University of New Jersey, FECHA. [En línea]. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.135.6358&rep=rep1&type=pdf

[15] Mesa E y Ramírez J. Planeación Neuroquirúrgica Y Navegación Estereotáxica, Trabajo Dirigido de Grado para optar título de Ingenieros Mecánicos, Universidad Nacional de Colombia, Facultad de Minas, Sede Medellín, 2009. [En línea]. URL: http://www.bdigital.unal.edu.co/923/1/1037577863_2009.pdf

[16] SensAble Technologies. Haptic Applications, Melerit Medical TraumaVision. Última modificación: 2010. Última visita: 20 de noviembre del 2010. [En línea]. URL: http://www.sensable.com/industries-haptic-gallery.htm#A

[17] Carrera I y Campos A. Robot Modular Asistente En Las Tareas De Cocina Controlado Por Una Interfaz Háptica, Universidad Politécnica de Madrid, Madrid España. [En línea]. URL:http://www.cea-ifac.es/actividades/jornadas/XXIX/pdf/268.pdf

[18] Sepúlveda Gabriel. Programación Háptica, el Futuro de los Videojuegos, CINVESTAV México. [En línea].
URL:http://www.google.co.cr/url?sa=t&source=web&cd=1&ved=0CBYQFjAA&url=http%3A%2F%2Fwww.creanimax.com%2Fcfp2008%2FPrograma cion_Videojuegos_articulo_creanimax.doc&rct=j&q=Programaci%C3%B3n%20H%C3%A1ptica%2C%20el%20Futuro%20de%20los%20Videojuegos&ei=1UoJTbbIHcKt8AaqleGhAQ&usg=AFQjCNE8fbXHOCvw5PdZJQwgko03NkwogA&cad=rja

[19] Novint. *Haptic Device Abstraction Layer (HDAL), Programmer's Guide*. Version 2.1.3. Novint Technologies Incorporated. August 14, 2008.

[20] Novint. Novint Falcon SDK License Agreement. [En Línea]. Última modificación: 2010. Última visita: 26 de mayo del 2010. URL: https://backup.filesanywhere.com/fs/v.aspx?v=8b72688a6162b5b0a699

[21] Quiroga José. Instalación de Sistemas de Automatización y Datos. Curso Orientación Instalaciones y Construcción, Universidad De Vigo, 2008. [En línea]. URL: http://tv2.uvigo.es/uploads/material/Video/1452/ISAD_Tema4.pdf

[22] Barrientos A, Peñin L, Balaguer C, Aracil R. "Fundamentos de Robótica", Editorial Mc Graw Hill, 1999

[23] Instituto Universitario de Robótica Física y Tecnología. *¿Qué es la Robótica?,* [En Línea]. Última modificación: 2010. Última visita: 29 de noviembre del 2010. URL: http://robothumano.galeon.com/productos774285.html

[24] Ramírez Jaime. Sistemas Mecánicos. Introducción a la Robótica, 2010. [En línea]. URL: http://sites.google.com/site/docenciajg/Home/sistemas-mecanicos-2009-2010/apuntes-de-clase-09-10

[25] Wanadoo. La comunicación serie. En Línea]. Última visita: 30 de noviembre del 2010. URL: En línea: http://perso.wanadoo.es/pictob/comserie.htm

[26] National Instruments. Comunicación Serial, conceptos generales. [En Línea]. Última visita: 30 de noviembre del 2010. URL: http://digital.ni.com/public.nsf/allkb/039001258CEF8FB686256E0F005888D1

[27] EXPO21XX. Robotics, Stäubli. [En Línea]. Última visita: 15 de enero del 2011. URL: http://www.expo21xx.com/automation21xx/13608_st3_robotics/default.htm

[28] howstuffworks. How ASIMO Works. [En Línea]. Última visita: 15 de enero del 2011. URL: http://www.howstuffworks.com/asimo.htm

[29] MCBtec. Conexión del PC a Microcontrolador por RS232. [En Línea]. Última visita: 16 de agosto del 2010. URL: http://www.mcbtec.com/ConexionRS232.pdf

[30] Rodríguez. Ricardo. Sistemas de Programación Off-Line para un Robot Industrial RV2AJ y sus Aplicaciones. Perú, 2010 [En línea]. URL:http://cgi-pe.com/descarga/Ing.%20Ricardo%20Rodriguez_Robotica%20CGI.PDF

[31] Sabater José. Robótica Industrial, Ingeniería de Sistemas y Autómatas. Universidad Miguel Hernández de Elche. España, 2010 [En línea]. URL:http://isa.umh.es/asignaturas/rvc/tema4.pdf

[32] Rojas, J. Anchayhua Nilton. Control de Trayectoria de Robot de 3 GDL, Universidad Nacional de Ingeniería, Perú. En línea]. URL: http://www.docstoc.com/docs/24434789/Control-de-Robot

[33] Baltes Jacky, FORWARD KINEMATICS: THE DENAVIT-HARTENBERG CONVENTION, November 17th, 2009 [En línea]. URL: http://www4.cs.umanitoba.ca/~jacky/Robotics/Papers/spong_kinematics.pdf

[34] Mitsubishi Electric Industrial Automation. Melfa, Industrial Robot, Standar Specifications Manual. Operating range diagram, Page 2-10.

[35] Romero Rafael, Plataforma de Experimentación de Controladores para Robots Industriales. Proyecto de Fin de Carrera, España, 25 de junio de 2001. [En línea]. URL: http://bibing.us.es/proyectos/abreproy/11296/fichero/Documentaci%F3n+Adicional%252FProyecto+Fin+de+Carrera+RM-10%252FPFC.PDF

[36] mathworld. Cylindrical Coordinates. [En Línea]. Última visita: 20 de febrero del 2011. URL: http://mathworld.wolfram.com/CylindricalCoordinates.html

[37] Garita Allan. Acople funcional de la interfaz háptica Novint Falcon a un proceso Teleoperado de Automatización. Instituto Tecnologico de Costa Rica, setiembre, 2010.

8 Apéndices

8.1 Glosario y abreviaturas

Algoritmo de Denavit-Hartenberg: Método sistemático para describir y representar la geometría espacial (posición y orientación) de los elementos de una cadena cinemática (de un robot) con respecto a un sistema de referencia fijo.

Automatización: Uso de sistemas o elementos computarizados y electromecánicos para controlar maquinarias y procesos industriales sustituyendo a operadores humanos.

Abstracción: Consiste en aislar un elemento de su contexto o del resto de los elementos que lo acompañan. En programación, el término se refiere al énfasis en el "¿qué hace?" más que en el "¿cómo lo hace?" o bien se puede ver como el concepto de caja negra.

Capa de abstracción: Forma de ocultar los detalles de implementación de ciertas funciones.

Campo de trabajo: Es el volumen espacial dentro del cual el robot puede situar el extremo de su muñeca para interactuar con objetos.

Cinemática del robot: Se interesa por la descripción analítica del movimiento espacial del robot como una función del tiempo y en particular por las relaciones entre la posición y la orientación del extremo final del robot con los valores que toman sus coordenadas articulares.

CIROS: Es un sistema general de simulación en 3D, representa una herramienta profesional para llevar a cabo el modelado, la programación y la simulación de un entorno industrial.

Grados de libertad: Esta definido por el numero de articulaciones en las que se puede mover el robot sin contar el movimiento propio de la pinza de sujeción.

Háptica: Se refiere a la sensación del tacto. La háptica es el área que estudia e investiga cómo puede mezclarse el sentido del tacto con un mundo virtual.

LIRA: Laboratorio de Investigación en Robótica y Automatización.

Matriz de transformación homogénea (T): Matriz de 4x4 que sirve para girar y rotar un vector referido a un sistema de referencia fijo ó para expresar la orientación y posición de un sistema de referencia O'UVW con respecto a otro fijo OXYZ.

Robot industrial: Manipulador automático servo-controlado, reprogramable, polivalente, capaz de posicionar y orientar piezas, útiles o dispositivos especiales, siguiendo trayectoria variables reprogramables, para la ejecución de tareas variadas.

SDK: Kit de desarrollo de software o por sus siglas en ingles (software development kit), hace referencia a un conjunto de herramientas de desarrollo que le permite a un programador crear aplicaciones para un sistema concreto.

Teleoperación: Extensión de capacidades sensoriales y destreza humana a una localización remota.

Telerobótica: Forma evolucionada de teleoperación y se define como el conjunto de tecnologías que comprenden la monitorización y reprogramación a distancia de un robot por un ser humano.

8.2 Hoja de información del proyecto

Información del estudiante:

Nombre: Edgar Fonseca Gentilini

Cédula: 7-0169-0407 **Carné ITCR:** 200440541

Dirección de su residencia en época lectiva: Catago, 200 m oeste de la entrada principal del TEC y 50 m sur

Dirección de su residencia en época no lectiva: Limón, Siquirres, Barrio San Martín, Urbanización Merayo, casa # 30.

Teléfono en época lectiva: 8825-2473

Teléfono época no lectiva: 2768-6134

Email: gentilini20@gmail.com

Información del proyecto:

Nombre del Proyecto: Sistema teleoperado de un brazo robótico Mitsubishi RV-2AJ por medio de una Interfaz Háptica con retroalimentación de fuerza

Área del Proyecto: Control automático, automatización, mecatrónica, robótica y programación.

Información de la empresa:

Nombre: Instituto Tecnológico de Costa Rica

Zona: Cartago

Dirección: De la esquina suroeste de los Tribunales de Justicia de Cartago 500 m sur y 500 m este.

Teléfono: 2550-2257 **Fax:** 2591-6629 **Apartado:** 159-7050 Cartagi, Costa Rica

Actividad Principal: Educación Universitaria

Información del encargado en la empresa:

Nombre: Ing. Arys I. Carrasquilla Batista, M.C.

Puesto que ocupa: Profesora e Investigadora

Departamento: Escuela de Ingeniería Electrónica

Profesión: Ingeniera en Electrónica y Máster en Computación.

Grado académico: Maestría en Computación y Licenciada en Ingeniería Electrónica

Teléfono: 2550-9184 **Ext.:** 9184

Email: acarrasquilla@itcr.ac.cr

9 Anexos

9.1 Hojas de información técnica

9.1.1 Microcontrolador PIC 18F2550

MICROCHIP **PIC18F2455/2550/4455/4550**

28/40/44-Pin High-Performance, Enhanced Flash USB Microcontrollers with nanoWatt Technology

Universal Serial Bus Features:

- USB V2.0 Compliant
- Low Speed (1.5 Mb/s) and Full Speed (12 Mb/s)
- Supports Control, Interrupt, Isochronous and Bulk Transfers
- Supports up to 32 endpoints (16 bidirectional)
- 1-Kbyte dual access RAM for USB
- On-chip USB transceiver with on-chip voltage regulator
- Interface for off-chip USB transceiver
- Streaming Parallel Port (SPP) for USB streaming transfers (40/44-pin devices only)

Power-Managed Modes:

- Run: CPU on, peripherals on
- Idle: CPU off, peripherals on
- Sleep: CPU off, peripherals off
- Idle mode currents down to 5.8 µA typical
- Sleep mode currents down to 0.1 µA typical
- Timer1 oscillator: 1.1 µA typical, 32 kHz, 2V
- Watchdog Timer: 2.1 µA typical
- Two-Speed Oscillator Start-up

Flexible Oscillator Structure:

- Four Crystal modes including High Precision PLL for USB
- Two External Clock modes, up to 48 MHz
- Internal oscillator block:
  - 8 user-selectable frequencies, from 31 kHz to 8 MHz
  - User-tunable to compensate for frequency drift
- Secondary oscillator using Timer1 @ 32 kHz
- Dual oscillator options allow microcontroller and USB module to run at different clock speeds
- Fail-Safe Clock Monitor
  - Allows for safe shutdown if any clock stops

Peripheral Highlights:

- High-current sink/source 25 mA/25 mA
- Three external interrupts
- Four Timer modules (Timer0 to Timer3)
- Up to 2 Capture/Compare/PWM (CCP) modules:
  - Capture is 16-bit, max. resolution 6.25 ns (TCY/16)
  - Compare is 16-bit, max. resolution 100 ns (TCY)
  - PWM output: PWM resolution is 1 to 10-bit
- Enhanced Capture/Compare/PWM (ECCP) module:
  - Multiple output modes
  - Selectable polarity
  - Programmable dead time
  - Auto-Shutdown and Auto-Restart
- Enhanced USART module:
  - LIN bus support
- Master Synchronous Serial Port (MSSP) module supporting 3-wire SPI™ (all 4 modes) and I²C™ Master and Slave modes
- 10-bit, up to 13-channels Analog-to-Digital Converter module (A/D) with programmable acquisition time
- Dual analog comparators with input multiplexing

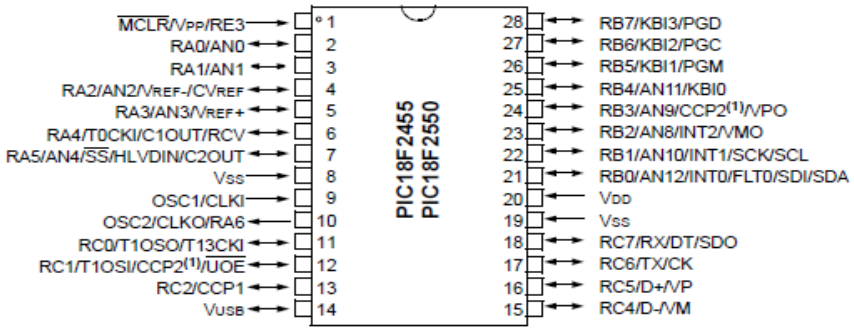
Special Microcontroller Features:

- C compiler optimized architecture with optional extended instruction set
- 100,000 erase/write cycle Enhanced Flash program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory typical
- Flash/Data EEPROM Retention: > 40 years
- Self-programmable under software control
- Priority levels for interrupts
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 41 ms to 131s
- Programmable Code Protection
- Single-Supply 5V In-Circuit Serial Programming™ (ICSP™) via two pins
- In-Circuit Debug (ICD) via two pins
- Optional dedicated ICD/ICSP port (44-pin devices only)
- Wide operating voltage range (2.0V to 5.5V)

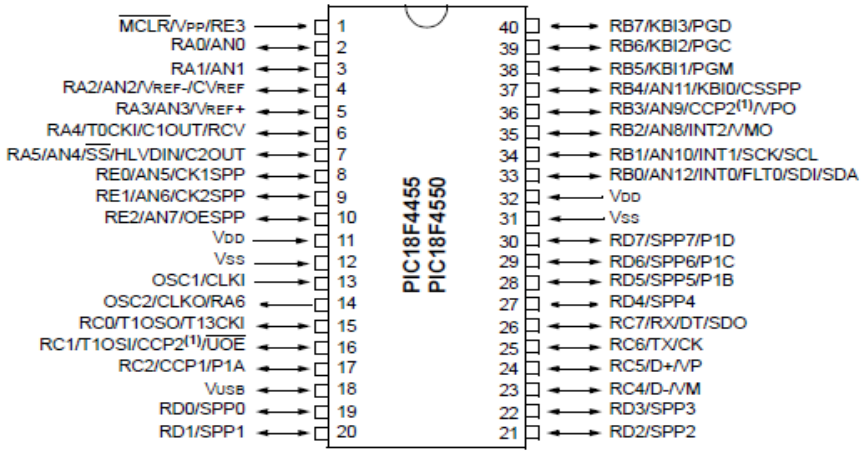
Device	Program Memory		Data Memory		I/O	10-bit A/D (ch)	CCP/ECCP (PWM)	SPP	MSSP		EAUSART	Comparators	Timers 8/16-bit
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)					SPI™	Master I²C™			
PIC18F2455	24K	12288	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F2550	32K	16384	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F4455	24K	12288	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3
PIC18F4550	32K	16384	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3

Pin Diagrams

28-Pin PDIP, SOIC



40-Pin PDIP



Note 1: RB3 is the alternate pin for CCP2 multiplexing.

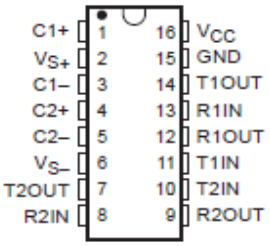
9.1.2 MAX232

MAX232, MAX232I
DUAL EIA-232 DRIVER/RECEIVER

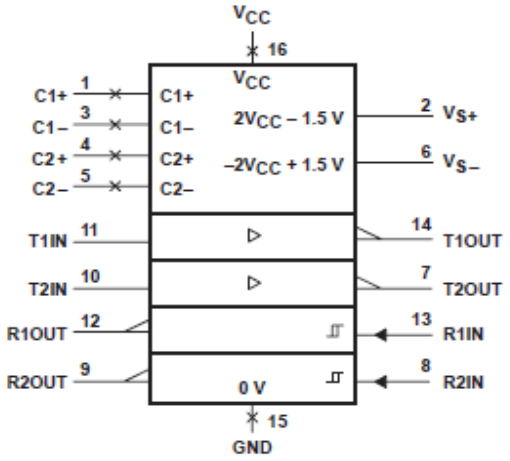
SLLS047G – FEBRUARY 1989 – REVISED AUGUST 1998

- Operates With Single 5-V Power Supply
- LinBiCMOS™ Process Technology
- Two Drivers and Two Receivers
- ±30-V Input Levels
- Low Supply Current . . . 8 mA Typical
- Meets or Exceeds TIA/EIA-232-F and ITU Recommendation V.28
- Designed to be Interchangeable With Maxim MAX232
- Applications
  - TIA/EIA-232-F
  - Battery-Powered Systems
  - Terminals
  - Modems
  - Computers
- ESD Protection Exceeds 2000 V Per MIL-STD-883, Method 3015
- Package Options Include Plastic Small-Outline (D, DW) Packages and Standard Plastic (N) DIPs

D, DW, OR N PACKAGE
(TOP VIEW)



logic symbol†



† This symbol is in accordance with ANSI/IEEE Std 91-1984 and IEC Publication 617-12.

description

The MAX232 device is a dual driver/receiver that includes a capacitive voltage generator to supply EIA-232 voltage levels from a single 5-V supply. Each receiver converts EIA-232 inputs to 5-V TTL/CMOS levels. These receivers have a typical threshold of 1.3 V and a typical hysteresis of 0.5 V, and can accept ±30-V inputs. Each driver converts TTL/CMOS input levels into EIA-232 levels. The driver, receiver, and voltage-generator functions are available as cells in the Texas Instruments LinASIC™ library.

The MAX232 is characterized for operation from 0°C to 70°C. The MAX232I is characterized for operation from –40°C to 85°C.

AVAILABLE OPTIONS

T_A	PACKAGED DEVICES		
	SMALL OUTLINE (D)	SMALL OUTLINE (DW)	PLASTIC DIP (N)
0°C to 70°C	MAX232D‡	MAX232DW‡	MAX232N
–40°C to 85°C	MAX232ID‡	MAX232IDW‡	MAX232IN

‡ This device is available taped and reeled by adding an R to the part number (i.e., MAX232DR).

Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

LinASIC and LinBiCMOS are trademarks of Texas Instruments Incorporated.

TEXAS INSTRUMENTS
POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

APPLICATION INFORMATION

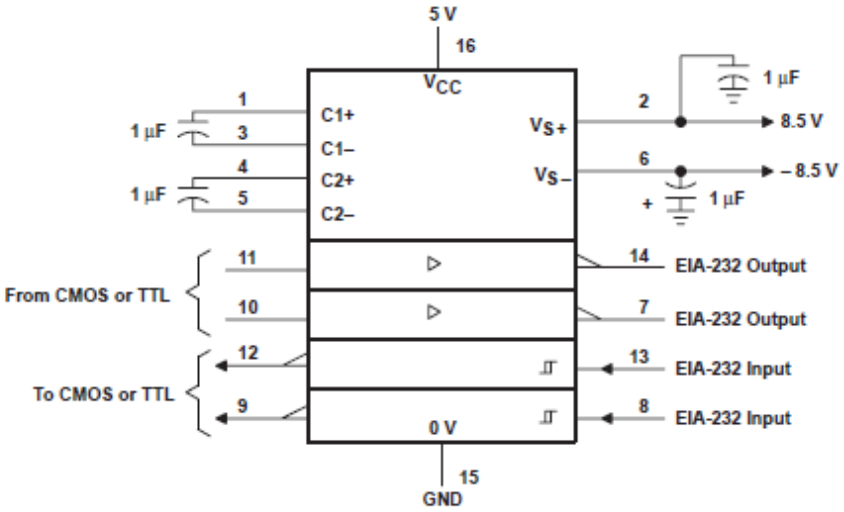


Figure 4. Typical Operating Circuit

9.1.3 Servo motor FUTABA

S3003 Standard

Volts	Torque	Speed
4.8V	44 oz-in (3.2 kg/cm)	0.23 sec/60°
6.0V	57 oz-in (4.1 kg/cm)	0.19 sec/60°
Dimensions		**Weight**
1-9/16 x 13/16 x 1-7/16 in (40 x 20 x 36 mm)		1.3 oz (37 g)

FUTM0031 — 3P

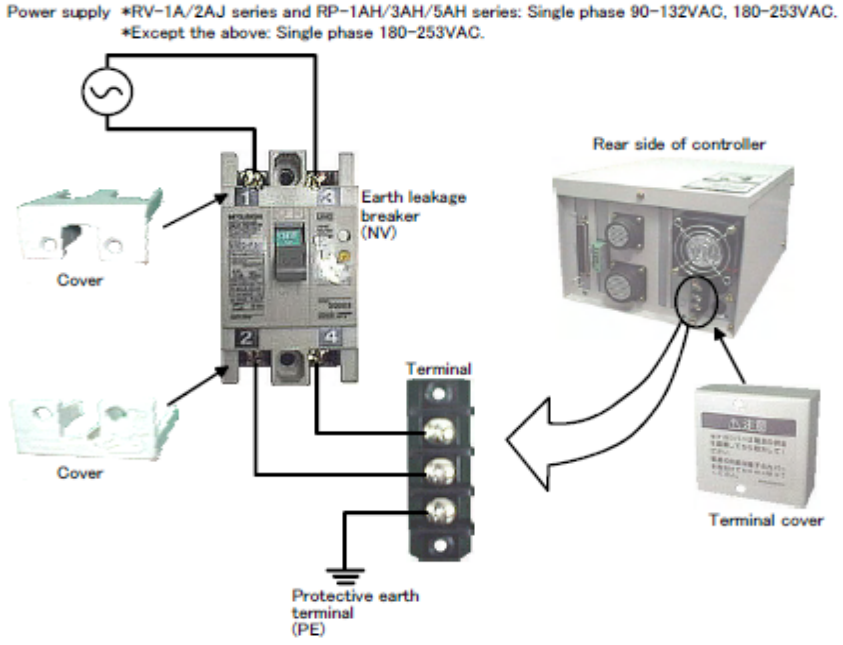
9.1.4 Estación de robot Festo

⚠CAUTION Do not stop the robot or apply emergency stop by turning the robot controller's main power OFF. If the robot controller main power is turned OFF during automatic operation, the robot accuracy could be adversely affected.Moreover, it may interfere with the peripheral device by drop or move by inertia of the arm.

⚠CAUTION Do not turn off the main power to the robot controller while rewriting the internal information of the robot controller such as the program or parameters.
If the main power to the robot controller is turned off while in automatic operation or rewriting the program or parameters, the internal information of the robot controller may be damaged.

Precautions for the basic configuration are shown below.(When CR1-571/CR1B-571 is used for the controller.)

⚠CAUTION Provide an earth leakage breaker that packed together on the primary power supply of the controller as protection against electric leakage. Confirm the setting connector of the input power supply voltage of the controller, if the type which more than one power supply voltage can be used. Then connect the power supply.
Failure to do so could lead to electric shock accidents.



⚠WARNING For using RH-5AH/10AH/15AH series or RH-6SH/12SH/18SH series.
While pressing the brake releasing switch on the robot arm, beware of the arm which may drop with its own weight.
Dropping of the hand could lead to a collision with the peripheral equipment or catch the hands or fingers.

2.2.5 Connecting with the controller

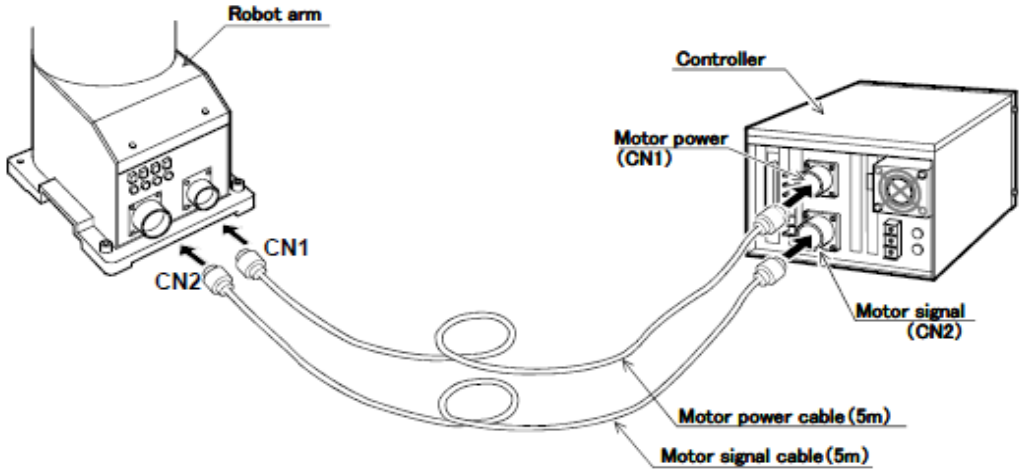


Fig.2-6 : Connecting the machine cables

Carry out the following procedure after installing the controller referring to the separate "Controller Setup, Basic Operation and Maintenance" manual.

1) Make sure that the power switch on the front of the controller is turned OFF.
2) Connect the machine cable to the robot arm and the corresponding connector on the controller

⚠ CAUTION The machine cable connectors are dedicated for the controller side and robot arm side, so take special care when connecting.
If connected incorrectly, the connector pins could bend or break. Thus, even if connected correctly, the robot will not operate correctly, creating a dangerous situation.

⚠ CAUTION Take special care to the leading of the connection cable. If the cable is pulled with force or bent excessively, wires could break or the connector could be damaged.

2 Robot arm

2.1 Standard specifications

2.1.1 Standard specifications

Table 2-1 : Tab Standard specifications of robot

Item		Unit	Specifications	
Type			RV-1A	RV-2AJ
Degree of freedom			6	5
Installation posture			On floor, hanging	
Structure			Vertical, multiple-joint type	
Drive system			ACservo motor	
Position detection method			J1 toJ3:50W with brake, J4,J6:15W no brake, J5:15Wwith brake	
Type			Absolute encoder	
Arm length	Shoulder shift	mm	0	
	Upper arm		250	
	Fore arm		160	
	Elbow shift		90	0
	Wrist length		72	
Operating range	J1	Degree	300(-150 to +150)	
	J2		180(-60 to +120)	
	J3		95(+60 to +155)	230(-110 to +120)
	J4		320(-160 to +160)	—
	J5		180(-90 to +90)	
	J6		400(-200 to +200)	
Speed of motion	J1	Degree/s	180	
	J2		90	
	J3		135	
	J4		180	—
	J5		180	
	J6		210	
Maximum resultant velocity		mm/s	Approx. 2200	Approx. 2100
Load	Maximum Note1)	kg	1.5	2
	Rating		1	1.5
Pose repeatability Note2)		mm	± 0.02	
Ambient temperature		℃	0 to 40	
mass		kg	Approx. 19	Approx. 17
Allowable moment load	J4	N・m	1.44	—
	J5		1.44	2.16
	J6		0.73	1.10
Allowable inertia	J4	$kg \cdot m^2$	2.16×10^{-2}	—
	J5		2.16×10^{-2}	3.24×10^{-2}
	J6		5.62×10^{-3}	8.43×10^{-3}
Arm reachable radius (front p-axis center point)		mm	418	410
Tool wiring Note3)			Four input signals (Hand section), Four output signals (Base section), Motorized hand output (Hand section)	
Tool pneumatic pipes			Φ4x4 (Base to hand section)	
Supply pressure		MPa	0.5 ± 10%	
Protection specification Note4)			IP30	

Note1)The maximum load capacity is the mass with the flange posture facing downword at the ± 10 degree limit.

Note2)The pose repeatability details are given in Page 6, "2.2.1 Pose repeatability and distance accuracy"

Note3)When using the 4-point hand output, the pneumatic hand interface (option) is required.

Note4)The protection specification details are given in Page 8, "2.2.3 Protection specifications and working environment" .

2.3 Names of each part of the robot

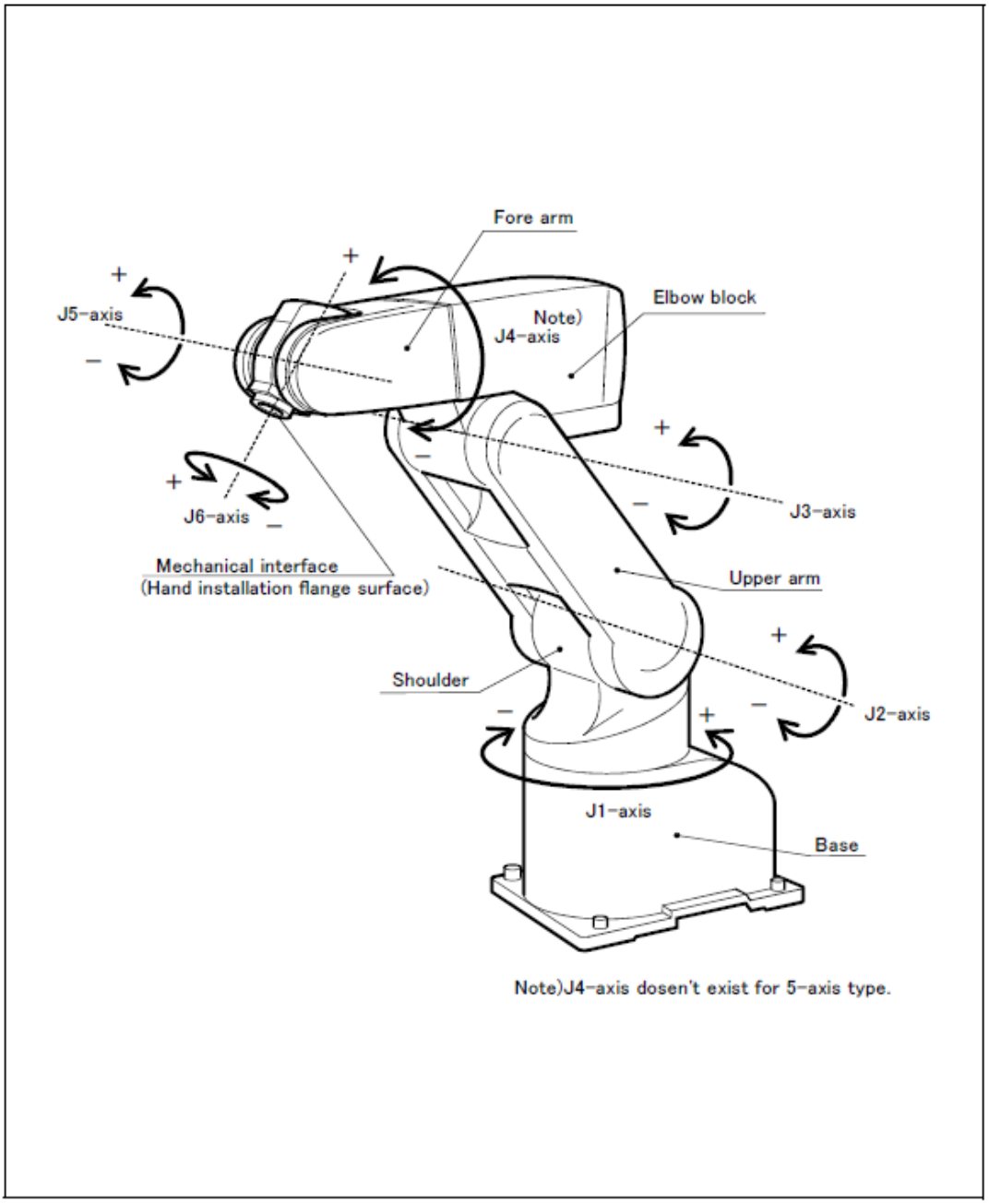
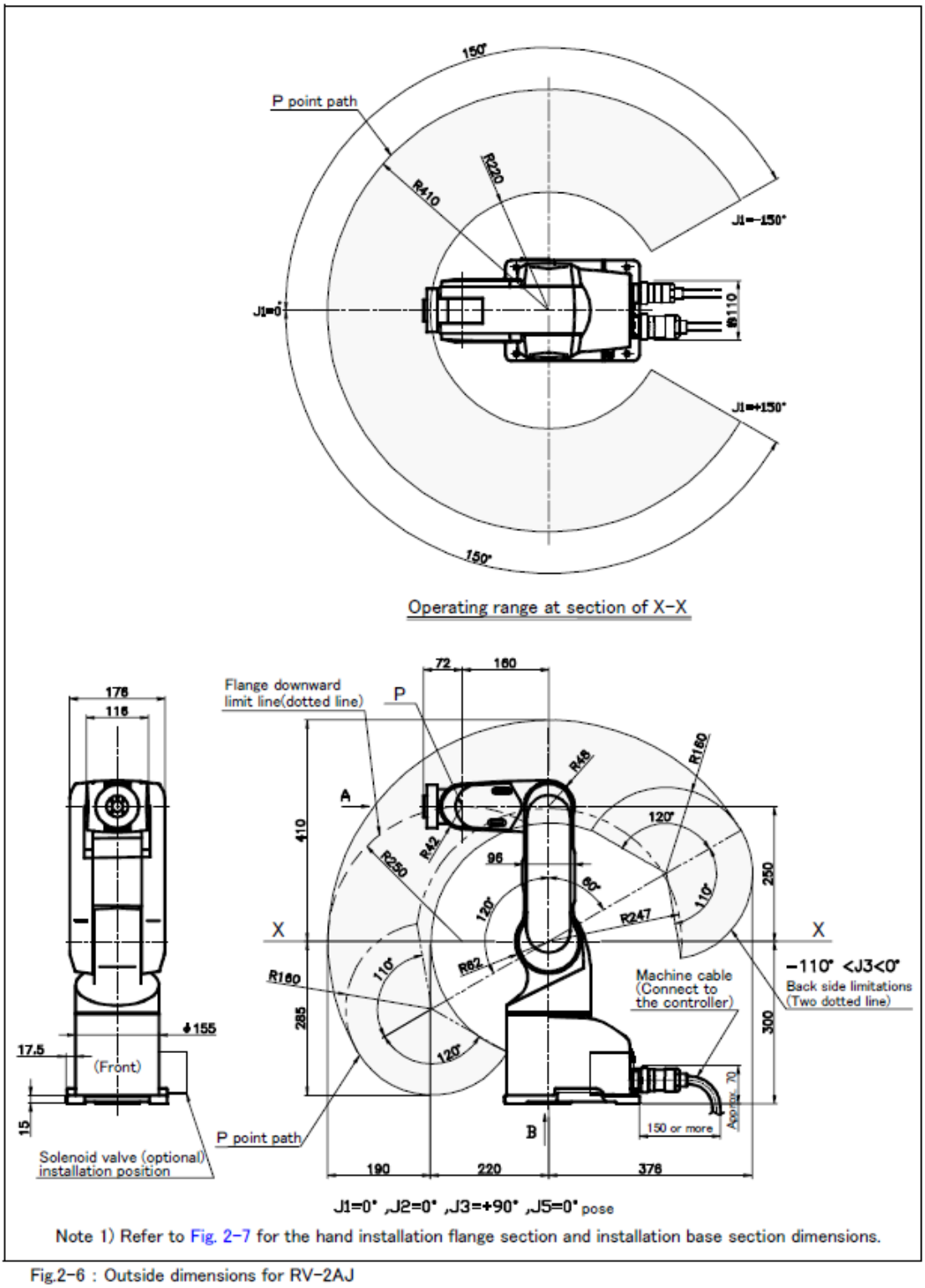


Fig.2-4 : Names of each part of the robot

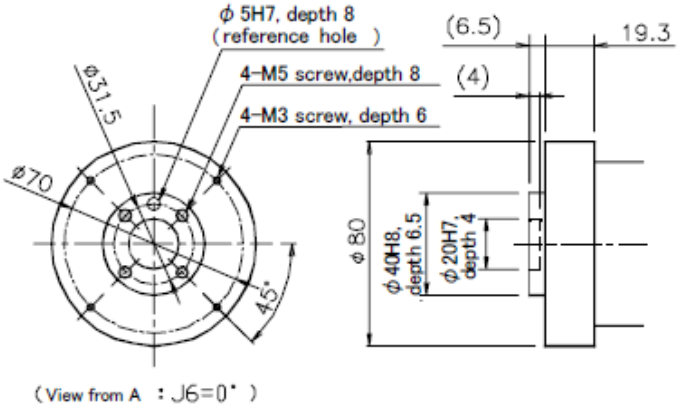
(2) RV-2AJ



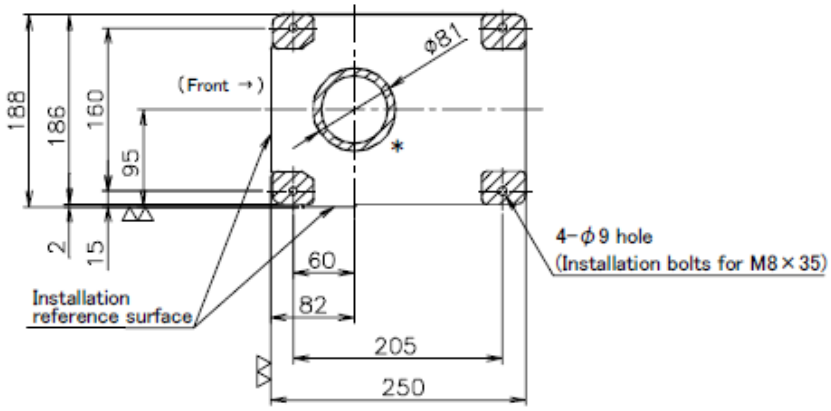
Note 1) Refer to Fig. 2-7 for the hand installation flange section and installation base section dimensions.

Fig.2-6 : Outside dimensions for RV-2AJ

(3) Mechanical interface and Installation surface of RV-1A/2AJ



Detail of Mechanical interface
(ISO9409-1)



View from B : Installation dimension details

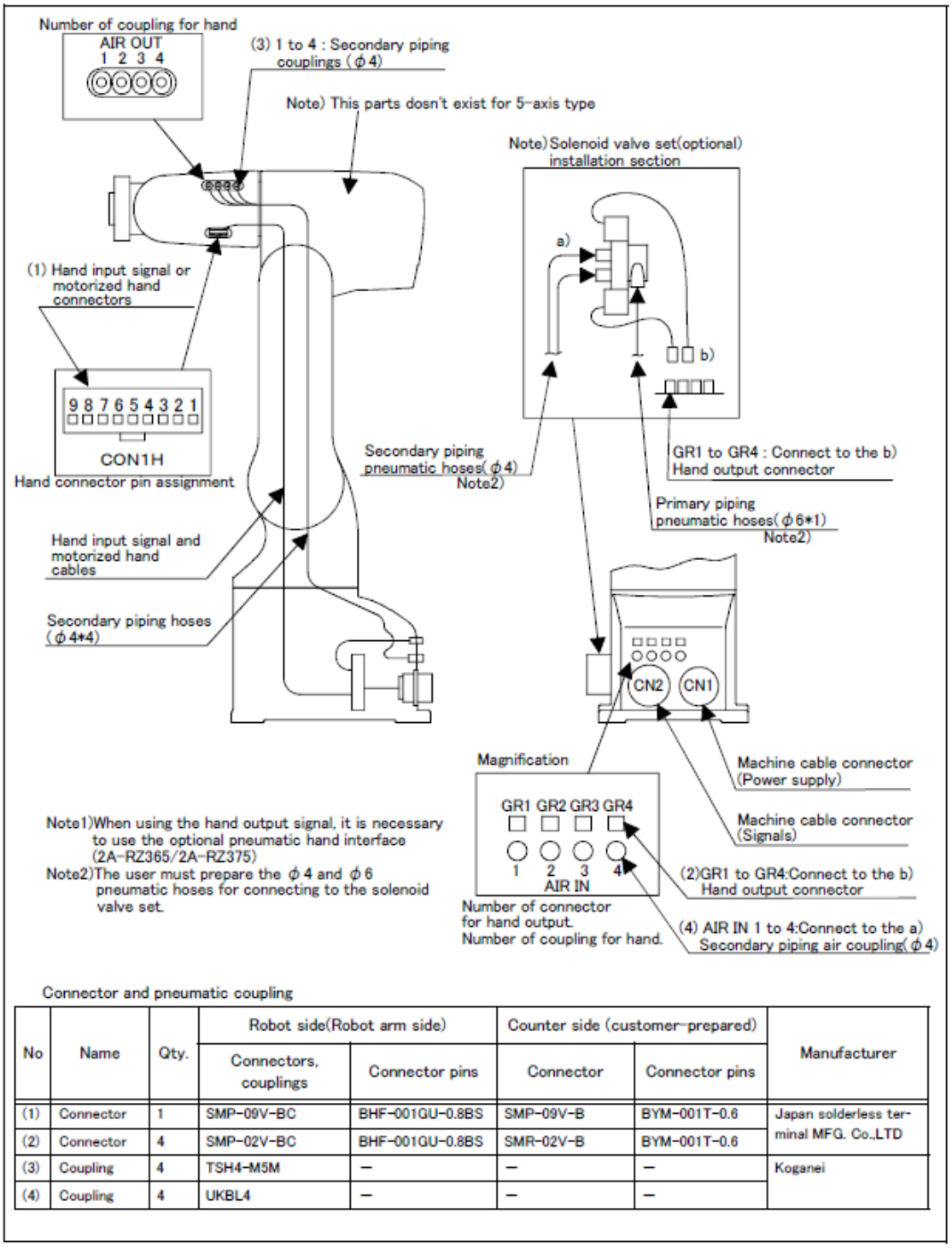
・The contact section shown with shading on the robot installation surface must be finished to ▽▽.
・The section marked with * must be contacted against the installation surface to ensure the robot rigidity.

Fig.2-7 : Mechanical interface and Installation surface of RV-1A/2AJ

2.5 Tooling

2.5.1 Wiring and piping for hand

Shows the wiring and piping configuration for a standard-equipped hand.



Connector and pneumatic coupling

No	Name	Qty.	Robot side(Robot arm side)		Counter side (customer-prepared)		Manufacturer
			Connectors, couplings	Connector pins	Connector	Connector pins	
(1)	Connector	1	SMP-09V-BC	BHF-001GU-0.8BS	SMP-09V-B	BYM-001T-0.6	Japan solderless terminal MFG. Co.,LTD
(2)	Connector	4	SMP-02V-BC	BHF-001GU-0.8BS	SMR-02V-B	BYM-001T-0.6	
(3)	Coupling	4	TSH4-M5M	—	—	—	Koganei
(4)	Coupling	4	UKBL4	—	—	—	

Fig.2-8 : Wiring and piping for hand

4.1 Handling the controller

4.1.1 Functions of each key

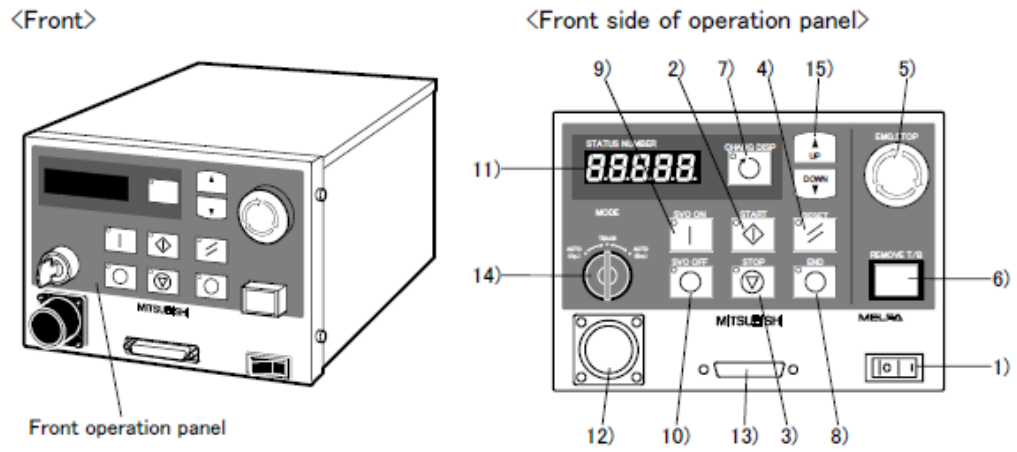


Fig.4-1 : Names of controller parts

1) POWER switch ..... This turns the control power ON/OFF.
2) START button ..... This executes the program and operates the robot. The program is run continuously.
3) STOP button ..... This stops the robot immediately. The servo does not turn OFF.
4) RESET button ..... This resets the error. This also resets the program's halted state and resets the program.
5) Emergency stop switch ..... This stops the robot in an emergency state. The servo turns OFF.
6) T/B remove switch ..... This is used to connect/disconnect the T/B without turning OFF the controller's control power.
7) CHNGDISP button ..... This changes the details displayed on the display panel in the order of "Override" → "Program No." → "Line No.".
8) END button ..... This stops the program being executed at the last line or END statement.
9) SVO.ON button ..... This turns ON the servo power. (The servo turns ON.)
10) SVO.OFF button ..... This turns OFF the servo power. (The servo turns OFF.)
11) STATUS NUMBER (display panel) ..... The alarm No., program No., override value (%), etc., are displayed.
12) T/B connection connector ..... This is a dedicated connector for connecting the T/B.
13) Personal computer connection connector ..... This is an RS-232C specification connector for connecting the personal computer.
14) MODE changeover switch ..... This changes the robot's operation mode.
AUTO (Op.) ..... Only operations from the controller are valid. Operations for which the operation mode must be at the external device or T/B are not possible.
TEACH ..... When the T/B is valid, only operations from the T/B are valid. Operations for which the operation mode must be at the external device or controller are not possible.
AUTO (Ext.) ..... Only operations from the external device are valid. Operations for which the operation mode must be at the T/B or controller are not possible.
15) UP/DOWN button ..... This scrolls up or down the details displayed on the "STATUS. NUMBER" display panel.

4.2.2 Functions of each key

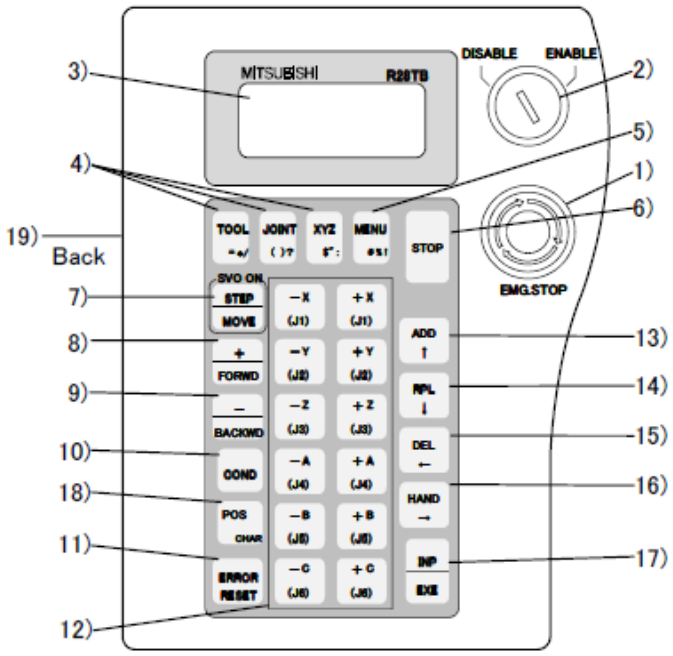


Fig.4-2 : Teaching pendant (Front side of R28TB)

1) [EMG. STOP] switch
This is a push-button switch with lock function for emergency stop. When this switch is pressed, the servo will turn OFF and the robot will stop immediately regardless of the T/B enable/disable state. To cancel this state, turn the switch clockwise.

2) [ENABLE/DISABLE] switch
This changeover switch is used to enable or disable the T/B key operations. To carry out operations using the T/B, always set this switch to "ENABLE" (valid). Operations with the T/B will be enabled, and operations from the controller and external sources will be disabled. The T/B will have the operation rights. To operate with the controller or external source, set this switch to "DISABLE" (invalid).

3) Display LCD
The program contents and robot state are displayed with the T/B key operations.

4) [TOOL] key
This selects the TOOL JOG mode.

4) [JOINT] key
This selects the JOINT JOG mode.

4) [XYZ] key
This selects the XYZ JOG, 3-AXIS XYZ or CYLINDER JOG mode.

5) [MENU] key
This returns the display screen to the "menu screen"

6) [STOP] key
This stops the program and decelerates the robot to a stop. This is the same function as the [STOP] switch on the front of the controller, and can be used even when the T/B [ENABLE/DISABLE] switch is set to DISABLE.

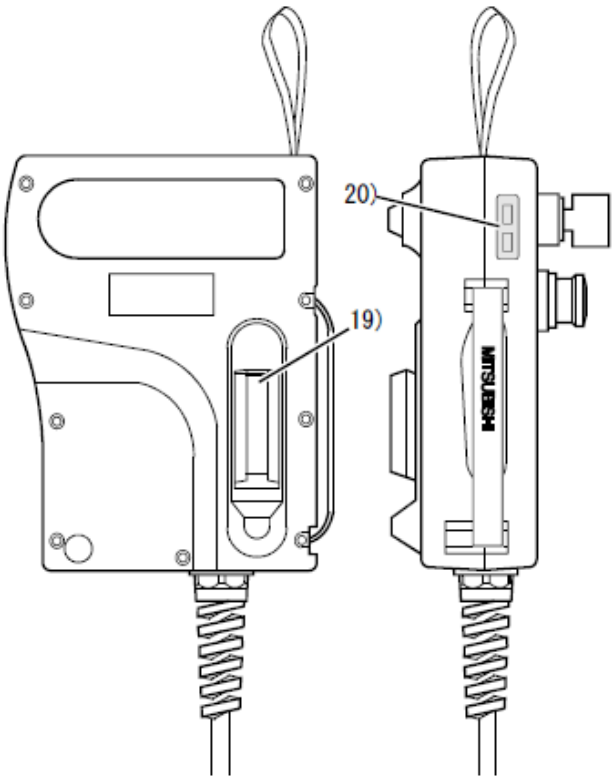
7) [STEP/MOVE] key
Jog operations are possible when this key is pressed simultaneously with the 12) jog operation key. Step jump is carried out when pressed simultaneously with the [INP/EXE] key. This also turns the servo ON.

8) [+/FORWD] key
Step feed is carried out when this key is pressed simultaneously with the [INP/EXE] key. On the edit screen, the next program line is displayed. When pressed simultaneously with the [STEP/MOVE] key, the override will increase.

9) [-/BACKWD] key
On the edit screen, the previous program line is displayed. When pressed simultaneously with the [INP/EXE] key, the axis will return along the robot's operation path. When pressed simultaneously with the [STEP/MOVE] key, the override (speed) will decrease.

10) [COND] key
This is used to edit the program.

11) [ERROR RESET] key
This key resets an error state that has occurred. When pressed simultaneously with the [INP/EXE] key, the program will be reset.



12) [Jog operation] key (12 keys from [−X (J1)] to [+C (J6)]
In this manual, these keys are generically called the "jog operation" keys. When JOINT JOG is selected, each axis will rotate, and when XYZ JOG is selected, the robot will move along each coordinate system. These keys are also used to input numeric values such as when selecting a menu or inputting a step No.

13) [ADD/ ↑] key
This additionally registers the position data. It also moves the cursor upward.

14) [RPL/ ↓] key
This corrects the position data. It also moves the cursor downward .

15) [DEL/ ←] key
This deletes the position data. It also moves the cursor to the left .

16) [HAND/ →] key
When pressed simultaneously with the [+C (J6)] or [−C (J6)] key, hand 1 will open or close. In the same manner, hand 2 will open/close when pressed simultaneously with the [+B (J5)] or [−B (J5)] key, hand 3 with the [+A (J4)] or [−A (J4)] key, and hand 4 with the [+Z (J3)] or [−Z (J3)] key. This key also moves the cursor to the right .

Fig.4-3 : Teaching pendant (Rear and side of R28TB)

17) [INP/EXE] key
This inputs the program, and carries out step feed/return.

18) [POS CHAR] key This changes between numbers and alphabetic characters when editing the position data, etc.

19) Deadman switch
When the [ENABLE/DISABLE] switch 2) is set to "ENABLE", and this key is released or pressed with force, the servo will turn OFF. Press this switch lightly when carrying out functions with the servo ON, such as jog operations. If emergency stop or servo OFF operation have been applied, and the servo is OFF, the servo will not turn ON even when this switch is pressed. In this case, carry out the servo ON operation again.

20) Contrast setting switch (Top: Dark, bottom: light)
This sets the display LCD brightness.

5.5 Communication parameter

These parameters set the items pertaining to communications

Table 5-5:List Communication parameter

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Communication setting Refer to "5.15About the communication setting"			Communication environment is set for RS-232C in the front of the robot controller. However, since this is used by the PC support software, this normally does not need to be changed. When connecting vision sensors, etc., use of optional expansion serial interface is recommended.	
	COMDEV	Character string 8	This configures which lines will be assigned to COM1 and COM2 when using communication lines in the OPEN instruction in MELFA BASIC IV. This parameter must be set if data link (used by the OPEN instruction) is to be performed. This parameter specifies the device that corresponds to COMn specified in the OPEN statement in the program (n is between 1 and 8). Parameters are starting from the left COM1, COM2, ... , COM8 in that order. To use expansion serial interface as COM, if CH1 is installed in option slot 1, "OPT11" should be entered as the value for the parameter. If CH2 (RS232) is installed in slot 1, "OPT12" should be entered as the value for the parameter. If CH2 (RS422) is installed in slot 1, "OPT13" should be entered as the value for the parameter. If CH1 is installed in slot 2, "OPT21" should be entered as the value for the parameter. If CH2 (RS232) is installed in slot 2, "OPT22" should be entered as the value for the parameter. If CH2 (RS422) is installed in slot 2, "OPT23" should be entered as the value for the parameter. For instance, if CH1 is used in slot 1 and is to be assigned to COM2, use '"RS232", "OPT11", , , , ,'. If CH2 is used in slot 2 and is to be assigned to COM3, use '"RS232", , "OPT22", , , , ,'. If Ethernet interface is to be used as COM and is installed in option slot 1 with NETPORT1, enter "OPT11" for the parameter value. NETPORT2, enter "OPT12" for the parameter value. NETPORT3, enter "OPT13" for the parameter value. NETPORT4, enter "OPT14" for the parameter value. NETPORT5, enter "OPT15" for the parameter value. NETPORT6, enter "OPT16" for the parameter value. NETPORT7, enter "OPT17" for the parameter value. NETPORT8, enter "OPT18" for the parameter value. NETPORT9, enter "OPT19" for the parameter value. COM1 is already assigned the standard RS232C in the front of the controller. The following restrictions apply when optional cards are installed. Option slot 1: Additional axis, expansion serial interface, Ethernet Option slot 2: Additional axis, expansion serial interface, CC-Link Option slot 3: Additional axis Additional axis cards are for CR1-only cases. For optional expansion serial interface, refer to "Expansion Serial Interface Instruction Manual". For optional expansion serial interface, refer to "Ethernet Interface Instruction Manual ".	"RS232", , , , , , ,
	CBAU232	Integer 1	Baud rate(9600,19200) For optional expansion serial interface, refer to "Expansion Serial Interface Instruction Manual".	9600
	CPRTY232	Integer 1	Parity bit(0: None, 1: Odd, 2: Even) For optional expansion serial interface, refer to "Expansion Serial Interface Instruction Manual".	2 (Even)
	CSTOP232	Integer 1	Stop bit(1,2) For optional expansion serial interface, refer to "Expansion Serial Interface Instruction Manual".	2 (Stop bit)

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
	CTERM232	Integer 1	End code(0:CR 1:CR+LF) For optional expansion serial interface, refer to "Expansion Serial Interface Instruction Manual".	0 (CR)
	CPRC232	Integer 1	Communication method(protocol) 0: For support software (Non-procedure) If data link is to be performed (OPEN, PRINT and INPUT instructions are executed from the program) under this setting, the external device must attach three characters "PRN" at the beginning when transmitting data. 1: For support software (With procedure) PC side must also be changed. 2: For data link with the robot program (used when communicating with vision sensors, etc.) Be advised that under this setting, connection with the support software cannot be made. Use the optional expansion serial interface. For optional expansion serial interface, refer to "Expansion Serial Interface Instruction Manual".	0