UNIVERSITAT ROVIRA I VIRGILI
ESCOLA TÈCNICA SUPERIOR D'ENGINYERIA
DEPARTAMENT D'ENGINYERIA ELECTRÒNICA, ELÈCTRICA I AUTOMÀTICA

PROYECTO DE FINAL DE CARRERA

TELEMANIPULACIÓN DE UN BRAZO ROBOTIZADO ARTICULADO

Autor: Joan Mercadé Papiol

Director del P.F.C.: Enric Vidal Idearte

ÍNDICE

1.- MEMORIA DESCRIPTIVA

1.1.- Introducción

En menos de 30 años la robótica ha pasado de ser un mito, propio de la imaginación de algunos autores literarios, a una realidad imprescindible en el actual mercado productivo. Tras los primeros albores, tímidos y de incierto futuro, la robótica experimentó entre las décadas de los setenta y ochenta un notable auge, llegando en los noventa a lo que se considera su mayoría de edad, caracterizada por una estabilización de la demanda y una aceptación y reconocimiento pleno en la industria.

La robótica posee un reconocido carácter interdisciplinario, participando en ella diferentes disciplinas y tecnologías tales como la teoría de control, la mecánica, la electrónica, el álgebra y la informática, entre otras.

Actualmente, los robots están integrados plenamente en la industria participando en tareas que por ser monótonas, repetitivas o de riesgo para el hombre, las realizan de forma automática y repetitiva con un alto grado de precisión.

Hoy en día pueden encontrarse todo tipo de robots en cualquier lugar y en campos de lo más variado, des de robots industriales y militares, hasta robots agrícolas, médicos o de exploración submarina.

1.2.- Objetivo del proyecto

El objetivo del presente proyecto es diseñar un software para microordenador, capaz de controlar al robot educacional ESCORBOT ER-III. El programa deberá correr sobre sistema operativo MS-DOS.

Este programa deberá permitir el manejo del robot como si fuera un telemanipulador usando una palanca de mandos (joystick) y el aprendizaje por guiado activo. Además, deberá incluir las funciones típicas de las aplicaciones de programación de robots, como la edición de programas para tareas específicas. Este software deberá comunicarse con la unidad de control del robot por medio del puerto serie de comunicaciones asíncronas RS-232-C.

1.3.- Titular

El titular y destinatario del presente proyecto es el Departament d'Enginyeria Elèctrica, Electrònica i Automàtica de la ETSE (Escola Técnica Superior d'Enginyeria) de la Universitat Rovira i Virgili, ubicada en Tarragona, en el Campus Sescelades.

La documentación original quedará en propiedad del DEEEA. El representante legal en este caso es el Sr. Enric Vidal Idearte con DNI 40931578L.

1.4.- Antecedentes

El motivo de este proyecto es la ampliación del software original que el robot educacional ESCORBOT ER-III lleva consigo, ya que éste no permite el manejo mediante palanca de comandos ni la enseñanza por guiado activo.

Estos conceptos se consideran verdaderamente importantes para una mejor y de mayor calidad formación de los alumnos ya que los auténticos robots industriales implantados en las plantas de fabricación se sirven de ellos.

1.5.- Introducción a la robótica

1.5.1.- Antecedentes históricos y origen del robot

A lo largo de la historia, el hombre se ha sentido fascinado por las máquinas y dispositivos capaces de imitar las funciones y movimientos de los seres vivos. Los griegos tenían una palabra específica para denominar a estas máquinas: autómatas. De esta palabra deriva la actual autómata: máquina que imita la figura y movimientos de un ser animado.

A lo largo de la historia, se desarrollaron diversos autómatas. Estos artilugios mecánicos inicialmente tenían fines eminentemente lúdicos, pero no se tardó en darles una aplicación práctica, introduciéndolos en la vida de la realeza. Ejemplos de estos autómatas son el Hombre de hierro de Alberto Magno (1204-1282), la Cabeza parlante de Roger Bacon (1214-1294), el Gallo de Estrasburgo (1352) y el León mecánico de Leonardo Da Vinci (1452-1519).

A finales del siglo XVIII y principios del XIX se desarrollaron algunas ingeniosas invenciones mecánicas, utilizadas fundamentalmente en la industria textil, entre las que destacan la hiladora giratoria de Hargreaves (1770), la hiladora mecánica de Crompton (1779), el telar mecánico de Cartwright (1785) y el telar de Jacquard (1801). Este último utilizaba una cinta de papel perforada como un programa para las acciones de la máquina. Es a partir de este momento cuando se empiezan a utilizar dispositivos automáticos en la producción, dando paso a la automatización industrial.

La palabra robot es de origen eslavo. En ruso *robota* significa trabajo. En checo significa trabajo forzado. En 1921 el escritor checo Karol Capek escribió Rossums Universal Robots, una obra teatral que obtuvo un gran éxito en Broadway. En esta obra unos androides trabajaban más del doble que un ser

humano. El término *robotics* (robótica) se debe a Isaac Asimov, el famoso escritor de ciencia-ficción.

El robot como máquina lleva un desarrollo independiente del término robot. Tras los primeros autómatas descritos anteriormente, casi todos de aspecto humano, los progenitores más directos de los robots fueron los telemanipuladores. El primer telemanipulador fue desarrollado en 1948 por R.C. Goertz del Argonne National Laboratory, con el objetivo de manipular elementos radioactivos sin riesgo para el operador. Éste consistía en un dispositivo mecánico maestro-esclavo. El manipulador maestro, situado en la zona segura, era movido directamente por el operador, mientras que el esclavo, situado en contacto con los elementos radioactivos y unido mecánicamente al maestro, reproducía fielmente los movimientos de éste. Años más tarde, en 1954, Goertz hizo uso de la tecnología electrónica y del servocontrol sustituyendo la transmisión mecánica por la eléctrica y desarrollando así el primer telemanipulador con servocontrol bilateral.

La evolución de los telemanipuladores a lo largo de los últimos años no ha sido tan espectacular como la de los robots. Recluidos en un mercado selecto y limitado (industria nuclear, militar, espacial, etc.) son en general desconocidos y comparativamente poco atendidos por los investigadores y usuarios de robots. Por su propia concepción, un telemanipulador precisa del mando continuo de un operador, y salvo por las aportaciones incorporadas con el concepto de control supervisado y la mejora de la telepresencia promovida hoy en día por la realidad virtual, sus capacidades no han variado mucho respecto a la de sus orígenes,.

La sustitución del operador por un programa de ordenador que controlase los movimientos del manipulador dio paso al concepto de robot.

George Devol, ingeniero norteamericano, es considerado como el "padre del robot". Devol estableció las bases del robot industrial moderno. En 1946 invento un aparato mecánico que permitía repetir una secuencia de movimientos a una máquina herramienta. Ese mismo año Eckert y Mauchly constituyeron la primera computadora digital, ENAC. En 1952 se construyó en el MIT la primera máquina de control numérico. En ese mismo año, Devol patentó el primer brazo manipulador con memoria. Esta máquina era capaz de mover la herramienta de trabajo de punto a punto. En 1957 la corporación Planet produjo el primer robot comercial. En 1960 Devol vendió su patente a la corporación Condec que empezó a producir el robot Unimate es su subsidiaria Unimation. En 1962 General Motors instaló el primer Unimation en su planta de fundición por troquel.

A partir de 1965 varios centros importantes de investigación tales como MTI, SRI, etc. Empezaron la investigación en robótica y áreas asociadas. En 1970 se construyó en el SRI un manipulador con seis grados de libertad controlado por computador y con control de movimiento por bucle cerrado PID con servomotores de corriente continua. Hasta entonces todos los robots manipuladores tenían actuadores hidráulicos.

El primer robot controlado por microcomputador fue producido por Cincinnati Milacron en 1973. En 1978 Unimation desarrolló el PUMA (iniciales de Programmable Universal Machine for Assembling).

La evolución de los robots industriales desde sus primeros balbuceos ha sido vertiginosa. En poco más de 30 años las investigaciones y desarrollos sobre robótica industrial han permitido que los robots tomen posicione 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, adaptándose inmediatamente a los cambios de producción solicitados por la demanda variable.

Los futuros desarrollos de la robótica apuntan a aumentar su movilidad, destreza y autonomía de sus acciones. La mayor parte de los robots actuales son con base estática, y se utilizan en aplicaciones industriales tales como ensamblado, soldadura, alimentación de maquinas de herramientas, etc. Sin embargo, existen otro tipo de aplicaciones que han hecho evolucionar en gran medida tanto la concepción de los robots como su propia morfología. Entre estos robots dedicados aplicaciones no industriales destacan los robots espaciales (brazos para lanzamiento y recuperación de satélites, vehículos de exploración lunar, robots para la construcción y mantenimiento 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, inspección de sistemas de refrigeración de centrales nucleares); robots militares (desactivación de bombas, robots centinelas experimentales dedicados a patrullar áreas críticas); robots móviles industriales (robots bomberos para patrullar fábricas, robots bibliotecarios, robots andantes con piernas); aplicaciones médicas (prótesis robotizadas, sistemas de ayuda a discapacitados); aplicaciones agrícolas (sembrado y recogido de cosechas, robot para esquilar ovejas); y un largo etcétera.

1.5.2.- Definición y clasificación del robot

1.5.2.1.- Definición

Existen ciertas dificultades a la hora de establecer una definición formal de lo que es un robot industrial. La primera de ellas surge de la diferencia conceptual entre el mercado japonés y el euro-americano de lo que es un robot y lo que es un manipulador. En segundo lugar, aunque existe una idea común de lo que es un robot industrial, no es fácil ponerse de acuerdo a la hora de establecer una definición formal.

Distintas organizaciones (RIA, ISO, IFR, AFNOR...) han establecido su propia definición de lo que es un robot. De todas ellas, se puede establecer que un robot manipulador operando individualmente necesita como mínimo los siguientes componentes:

- El brazo (robot) consistente en un sistema de articulaciones mecánicas (eslabones, engranajes, transmisión por cadena o correa), actuadores (motores eléctricos o hidráulicos) y sensores de posición usados en el sistema de control de bucle cerrado.
- El controlador, generalmente basado en microcomputador, que recibe las señales de los sensores de posición y envía comandos a la fuente de potencia controlada (o unidad conversora).
- La unidad conversora de potencia que alimenta los motores que actúan las articulaciones.

Una de las definiciones más completas es la establecida por la Asociación Francesa de Normalización (AFNOR), ya que define primero el manipulador y, basándose en dicha definición, el robot:

- **Manipulador**: mecanismo formado generalmente por elementos en serie, articulados entre sí, destinado al agarre y desplazamiento de objetos. Es multifuncional y puede ser gobernado directamente por un operador humano o mediante dispositivo lógico.

- **Robot**: manipulador automático servocontrolado, reprogramable, polivalente, capaz de posicionar y orientar piezas, útiles o dispositivos especiales, siguiendo trayectorias variables reprogramables, para la ejecución de tareas variadas. Normalmente tiene la forma de uno o varios brazos terminados en su muñeca. Su unidad de control incluye un dispositivo de memoria y ocasionalmente la percepción del entorno. Normalmente su uso es el de realizar una tarea de manera cíclica, pudiéndose adaptar a otra sin cambios permanentes en su material.

Un sistema robotizado es un concepto más amplio. Engloba todos aquellos dispositivos que realizan tareas de forma automática en sustitución de un ser humano y que pueden incorporar o no a uno o varios robots, siendo esto último lo más frecuente.

1.5.2.2.- Clasificación

La Federación Internacional de Robótica (IFR) distingue entre cuatro tipos de robots:

- Robot secuencial.
- Robot de trayectoria controlable.
- Robot adaptativo.
- Robot telemanipulado.

En cuanto a los robots de servicio, se pueden definir como:

Dispositivos electromecánicos móviles o estacionarios, dotados normalmente de uno o varios brazos mecánicos independientes, controlados por un programa de ordenador y que realizan tareas no industriales de servicio.

En esta definición entrarían entre otros los robots dedicados a cuidados médicos, educación, domésticos, uso en oficinas, intervención en ambientes peligrosos, aplicaciones espaciales, aplicaciones submarinas y agricultura. Sin embargo, esta definición de robots de servicio excluye los telemanipuladores, pues éstos no se mueven mediante el control de un programa de ordenador, sino que están controlados directamente por el operador humano.

Los robots teleoperados son definidos por la NASA como:

Dispositivos robóticos con brazos manipuladores y sensores y cierto grado de movilidad, controlados remotamente por un operador humano de manera directa o a través de un ordenador.

1.5.3.- Morfología del robot

Un robot está formado por los siguientes elementos: estructura mecánica, transmisiones, sistema de accionamiento, sistema sensorial, sistema de control y elementos terminales.

1.5.3.1.- Estructura mecánica de un robot

Mecánicamente, un robot esta formado por una serie de elementos o eslabones unidos mediante articulaciones que permiten un movimiento relativo entre cada dos eslabones consecutivos. La constitución física de la mayor parte de los robots industriales guarda cierta similitud con la anatomía del cuerpo humano, por lo que en ocasiones, para hacer referencia a los distintos elementos que componen el robot, se usan términos como cuerpo, brazo, coso y muñeca.

El movimiento de cada articulación puede ser de desplazamiento, de giro, o de una combinación de ambos. De este modo son posibles los seis tipos diferentes de articulaciones:

- Esférica o rótula
- Planar
- Tornillo
- Prismática
- Rotación
- Cilíndrica

Cada uno de los movimientos independientes que puede realizar cada articulación con respecto a la anterior, se denomina grado de libertad (GDL). En la Figura 1.1 se indica el número de GDL de cada articulación. El número de

grados de libertad del robot viene dado por la suma de los grados de libertad de las articulaciones que lo componen. Puesto que las articulaciones empleadas actualmente en los robots industriales son únicamente las de rotación y prismática con un solo GDL cada una, el número de GDL del robot suele coincidir con el número de articulaciones de que se compone.

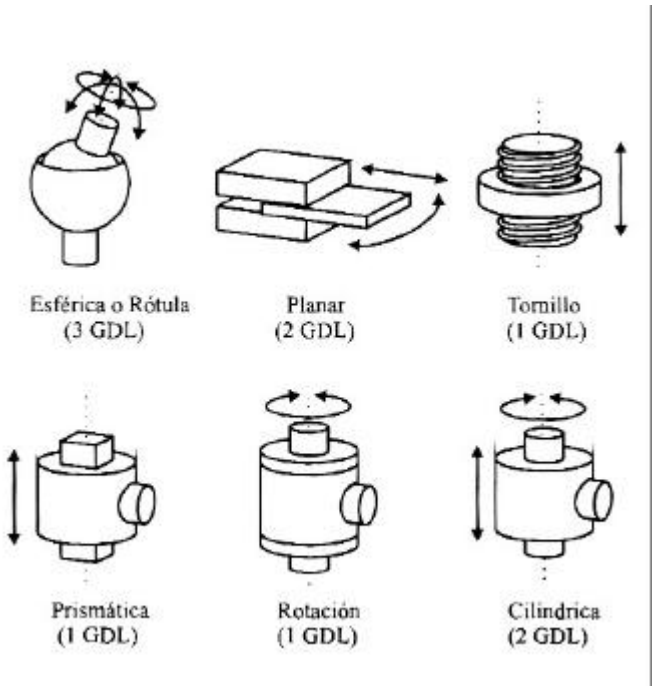

Figura 1.1. Tipos de articulaciones para robots.

El empleo de diferentes articulaciones en un robot, da lugar a diferentes configuraciones, con características a tener en cuenta tanto en el diseño y construcción del robot como en su aplicación. Las combinaciones más frecuentes son las representadas en la Figura 1.2 donde se atiende únicamente a las tres primeras articulaciones del robot, que son las más importantes a la hora de posicionar su extremo en un punto del espacio.

En la práctica, a pesar de ser necesarios seis GDL para tener total libertad en el posicionado y orientación del extremo del robot, muchos robots industriales cuentan con sólo cuatro o cinco GDL, por ser suficientes para llevar acabo las tareas que se les encomiendan.

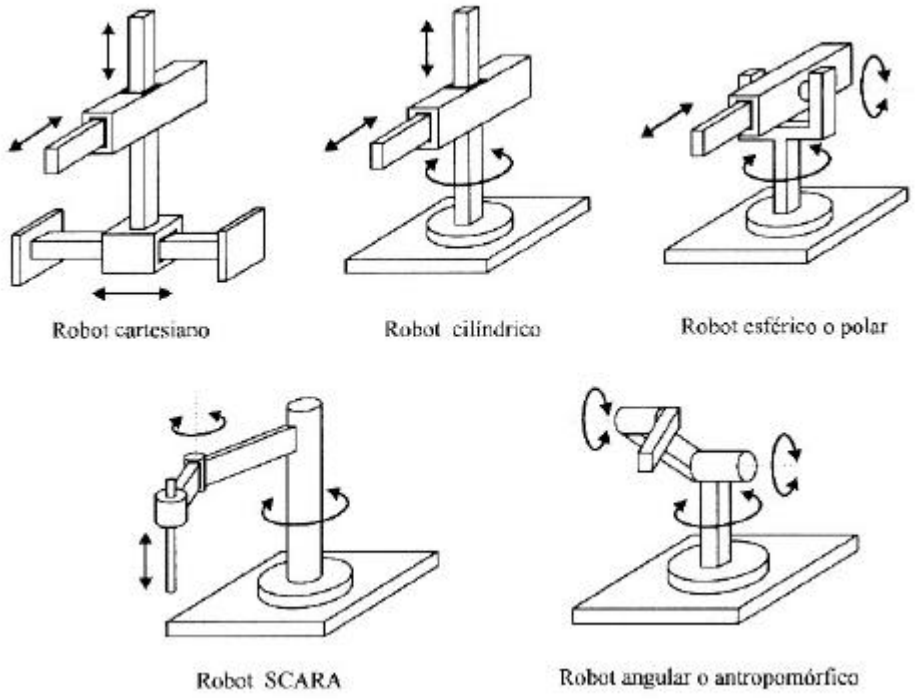

Figura 1.2. Configuraciones más frecuentes en robots industriales.

Existen también casos opuestos, en los que se precisan más de seis GDL para que el robot pueda tener acceso a todos los puntos de su entorno. Así, si se trabaja en un entorno con obstáculos, el dotar al robot con gados de libertad adicionales le permitirá acceder a posiciones y orientaciones de su extremo a las que, como consecuencia de los obstáculos, no hubiera llegado con seis GDL. Otra situación frecuente es la de dotar al robot de un GDL adicional que le permita desplazarse a lo largo de un carril, aumentando así el volumen del espacio al que puede acceder. Cuando el número de grados de libertad del robot es mayor que los necesarios para realizar una determinada tarea se dice que el robot es redundante.

La Figura 1.3 representa las dos situaciones comentadas para el caso de robots planares a los que les bastaría con dos GDL para posicionar su extremo en cualquier punto del plano.

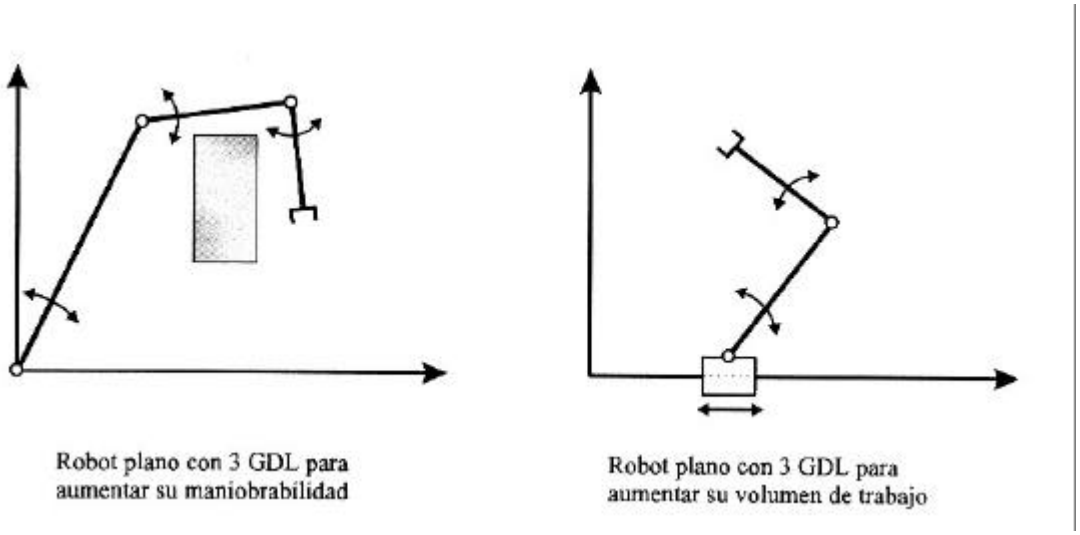

Figura 1.3. Robots planares redundantes.

1.5.3.2.- Transmisiones y reductores

Las transmisiones son los elementos encargados de transmitir el movimiento desde los actuadores hasta las articulaciones. Los reductores son los elementos encargados de adaptar el par y la velocidad de la salida del actuador a los valores adecuados para el movimiento de los elementos del motor.

1.5.3.2.1.- Transmisiones

Dado que un robot mueve su extremo con aceleraciones elevadas, es de gran importancia reducir al máximo su momento de inercia. Del mismo modo, los pares estáticos que deben vencer los actuadores dependen directamente de la distancia de las masas al actuador. Por estos motivos se procura que los actuadores, por lo general pesados, estén lo más cerca posible de la base del robot. Esta circunstancia obliga a utilizar sistemas de transmisión que trasladen el movimiento hasta las articulaciones, especialmente a las situadas en el

extremo del robot. Asimismo, las transmisiones pueden ser utilizadas para convertir movimiento circular en lineal o viceversa, lo que en ocasiones puede ser necesario.

Un buen sistema de transmisión debe cumplir una serie de características básicas: debe tener un tamaño y peso reducido, se ha de evitar que presente juegos u holguras considerables y se deben buscar transmisiones con un gran rendimiento.

Aunque no existe un sistema de transmisión específico para robots, sí existen algunos usados con mayor frecuencia y que se recogen clasificados en la Tabla 1.1. La clasificación se ha realizado en base al tipo de movimiento posible en la entrada y la salida: lineal o circular.

Tabla 1.1. Sistemas de transmisión para robots

Entrada-Salida	Denominación	Ventajas	Inconvenientes
Circular-Circular	Engranaje	Pares altos	Holguras
	Correa dentada	Distancia grande	-
	Cadena	Distancia grande	Ruido
	Paralelogramo	-	Giro limitado
	Cable	-	Deformabilidad
Circular-Lineal	Tornillo sin fin	Poca holgura	Rozamiento
	Cremallera	Holgura media	Rozamiento
Lineal-Circular	Paral. Articulado	-	Control difícil
	Cremallera	Holgura media	Rozamiento

Es muy importante que el sistema de transmisión a utilizar no afecte al movimiento que transmite, ya sea por el rozamiento inherente a su funcionamiento o por las holguras que su desgaste pueda introducir. También hay que tener en cuenta que el sistema de transmisión sea capaz de soportar un funcionamiento continuo a un par elevado, y a ser posible entre grandes distancias.

Las transmisiones más habituales son aquellas que cuentan con movimiento circular tanto en la entrada como a la salida. Incluidas en éstas se hallan los engranajes, las correas dentadas y las cadenas.

1.5.3.2.2.- Reductores

A los reductores utilizados en robótica se les exige unas condiciones de funcionamiento muy restrictivas. La exigencia de estas características viene motivada por las altas prestaciones que se le piden al robot en cuanto a precisión y velocidad de posicionamiento. La Tabla 1.2 muestra valores típicos de reductores para robótica actualmente empleados.

Tabla 1.2. Características de reductores para robótica.

Características	Valores típicos
Relación de reducción	50 ÷ 300
Peso y tamaño	0.1 ÷ 30 kg
Momento de inercia	10^{-4} kg· m^2
Velocidades de entrada máxima	6000 ÷ 7000 rpm
Par de salida nominal	5700 Nm
Par de salida máximo	7900 Nm
Juego angular	0 ÷ 2 “
Rigidez torsional	100 ÷ 2000 Nm/rad
Rendimiento	85 % ÷ 98 %

Se buscan reductores de bajo peso, reducido tamaño, bajo rozamiento y que al mismo tiempo sean capaces de realizar una reducción elevada de velocidad en un único paso. Se tiende también a minimizar su momento de inercia, de negativa influencia en el movimiento del motor, especialmente crítico en motores de baja inercia.

Los reductores, por motivos de diseño, tienen una velocidad máxima de entrada admisible, que como regla general aumenta a medida que disminuye el tamaño del motor. También existe una limitación en cuanto al par de salida nominal permisible (T_2) que depende del par de entrada (T_1) y de la relación de transmisión a través de la relación:

$$T_2 = \eta T_1 \frac{\omega_1}{\omega_2}$$

donde el rendimiento (η) puede llegar a ser cerca del 100% y la relación de reducción de velocidades (ω_1 = velocidad de entrada; ω_2 = velocidad de salida) varía entre 50 y 300.

Puesto que los robots trabajan en ciclos cortos que implican continuos arranques y paradas, es de gran importancia que el reductor sea capaz de soportar pares elevados puntuales. También se busca que el juego angular o backlash sea lo menor posible. Éste se define como el ángulo que gira el eje de salida cuando se cambia su sentido de giro sin que llegue a girar el eje de entrada. Por último, es importante que los reductores para robótica posean una alta rigidez torsional, definida como el par que hay que aplicar sobre el eje de salida para que, manteniendo bloqueado el de entrada, aquél gire un ángulo unitario.

1.5.3.2.3.- Accionamiento directo

Desde hace unos años existen en el mercado robots con accionamiento directo (Direct Drive DD), en el que el eje del actuador se conecta directamente a la carga o articulación, sin la utilización de un reductor intermedio. Este término suele utilizarse exclusivamente para robots con accionamiento eléctrico.

Este tipo de accionamiento aparece a raíz de la necesidad de utilizar robots en aplicaciones que exigen combinar gran precisión con alta velocidad. Los reductores introducen una serie de efectos negativos, como son juego angular, rozamiento o disminución de la rigidez del accionador, que pueden impedir alcanzar los valores de precisión y velocidad requeridos.

Las principales ventajas que se derivan de la utilización de accionamientos directos son las siguientes:

- Posicionamiento rápido y preciso, pues se evitan los rozamientos y juegos de las transmisiones y reductores.
- Aumento de las posibilidades de controlabilidad del sistema a costa de una mayor complejidad.
- Simplificación del sistema mecánico al eliminarse el reductor.

El principal problema que existe para la aplicación práctica de un accionamiento directo radica en el motor a emplear. Debe tratarse de motores que proporcionen un par elevado (unas 50-100 veces mayor que un reductor) a bajas revoluciones (las de movimiento de la articulación) manteniendo la máxima rigidez posible.

Otra cuestión importante a tener en cuenta en el empleo de accionamientos directos es la propia cinemática del robot. Colocar motores, generalmente pesados y voluminosos, junto a las articulaciones, no es factible para todas las configuraciones del robot debido a las inercias que se generan. El estudio de la cinemática con la que se diseña el robot ha de tener en cuenta estos parámetros, estando la estructura final elegida altamente condicionada por ellos. Por ese motivo, los robots de accionamiento directo son de tipo SCARA, cuyo diseño se corresponde bien con las necesidades que el accionamiento directo implica.

1.5.3.3.- Actuadores

Los actuadores tienen por misión generar el movimiento de los elementos del robot según las ordenes dadas por la unidad de control. Los actuadores utilizados en robótica pueden emplear energía neumática, hidráulica o eléctrica. Cada uno de estos sistemas presenta características diferentes, siendo preciso evaluarlas a la hora de seleccionar el tipo de actuador más conveniente. Las características a considerar son entre otras:

- Potencia
- Controlabilidad.
- Peso y volumen.
- Precisión.
- Velocidad.
- Mantenimiento.
- Coste.

1.5.3.3.1.- Actuadores neumáticos

En ellos la fuente de energía es aire a presión entre 5 y 10 bar. Existen dos tipos de actuadores neumáticos:

- Cilindros neumáticos.
- Motores neumáticos (de aletas rotativas o de pistones axiales).

En los primeros se consigue el desplazamiento de un émbolo encerrado en un cilindro, como consecuencia de la diferencia de presión en ambos lados de aquel (Figura 1.4). Los cilindros neumáticos pueden ser de simple o doble efecto. En los primeros, el émbolo se desplaza en un sentido como resultado del empuje ejercido por el aire a presión, mientras que en el otro sentido se

desplaza como consecuencia del efecto de un muelle (que recupera al émbolo a su posición de reposo). En los cilindros de doble efecto el aire a presión es el encargado de empujar al émbolo en las dos direcciones, al poder ser introducido de forma arbitraria en cualquiera de las dos cámaras.

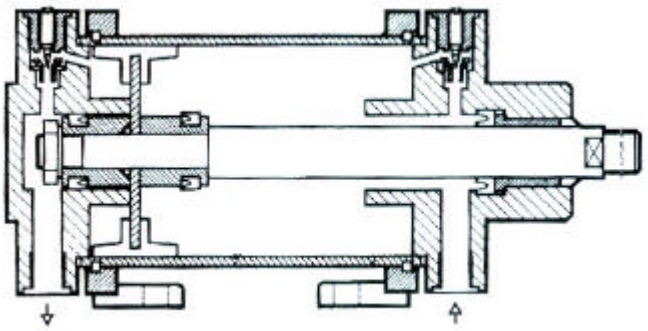

Figura 1.4. Esquema de cilindro neumático de doble efecto.

Normalmente, con los cilindros neumáticos solo se persigue un posicionamiento en los extremos del mismo y no un posicionamiento continuo. Esto último se puede conseguir con una válvula de distribución (generalmente de accionamiento eléctrico) que canaliza el aire a presión hacia una de las dos caras del émbolo alternativamente. Existen no obstante sistemas de posicionamiento continuo de accionamiento neumático, aunque debido a su coste y calidad todavía no resultan competitivos.

En los motores neumáticos se consigue el movimiento de rotación de un eje mediante aire a presión. Los dos tipos más usados son los motores de aletas rotativas y los motores de pistones axiales.

En general y debido a la compresibilidad del aire, los actuadores neumáticos no consiguen una buena precisión de posicionamiento. Sin embargo, su sencillez y robustez hacen adecuado su uso en aquellos casos en los que sea suficiente un posicionamiento en dos situaciones diferentes (todo o nada). Por ejemplo son utilizados en manipuladores sencillos, en apertura y

cierre de pinzas o en determinadas articulaciones de algún robot (como el movimiento vertical del tercer grado de libertad de algunos robots de tipo SCARA).

Debe tenerse en cuenta que para el uso de algún tipo de accionamiento neumático se deberá disponer de una instalación de aire comprimido, incluyendo: compresor, sistema de distribución (tuberías, electroválvulas), filtros, secadores, etc.

1.5.3.3.2.- Actuadores hidráulicos

Este tipo de actuadores no se diferencia funcionalmente en mucho de los neumáticos. En ellos, en vez de aire se utilizan aceites minerales a presión comprendida normalmente entre los 50 y 100 bar, llegándose en ocasiones a superar los 300 bar. Existen, como en el caso de los neumáticos, actuadores del tipo cilíndrico y del tipo motores de aletas y pistones.

Sin embargo, las características del fluido utilizado en los actuadores hidráulicos marcan ciertas diferencias con los neumáticos. En primer lugar, el grado de compresibilidad de los aceites usados es considerablemente inferior al del aire, por lo que la precisión obtenida en este caso es mayor. Por motivos similares, es más fácil en ellos realizar un control continuo, pudiendo posicionar su eje en todo un rango de valores (haciendo uso de servocontrol) con notable precisión. Además, las elevadas presiones de trabajo, diez veces superiores a las de los actuadores neumáticos, permiten desarrollar elevadas fuerzas y pares.

Por otra parte, este tipo de actuadores presenta estabilidad frente a cargas estáticas. Esto indica que el actuador es capaz de soportar cargas, como el peso o una presión ejercida sobre una superficie, sin aporte de energía (para mover el émbolo de un cilindro sería preciso vaciar éste de aceite). También es

destacable su elevada capacidad de carga y relación potencia-peso, así como sus características de autolubricación y robustez.

Frente a estas ventajas existen también ciertos inconvenientes. Por ejemplo, las elevadas presiones a las que se trabaja propician la existencia de fugas de aceite a lo largo de la instalación. Asimismo, esta instalación es más complicada que la necesaria para actuadores neumáticos y mucho más que para eléctricos, necesitando de equipos de filtrado de partículas, eliminación de aire, sistemas de refrigeración y unidades de control de distribución.

Los accionamientos hidráulicos se usan con frecuencia en aquellos robots que deben manejar grandes cargas.

1.5.3.3.3.- Actuadores eléctricos

Las características de control, sencillez y precisión de los accionamientos eléctricos han hecho que sean los más usados en los robots industriales actuales.

Dentro de los actuadores eléctricos pueden distinguirse tres tipos diferentes:

- Motores de corriente continua (DC):
    - Controlados por inducido.
    - Controlados por excitación.
- Motores de corriente alterna (AC):
    - Síncronos.
    - Asíncronos.

- Motores paso a paso.

1.5.3.3.3.1.- Motores de corriente continua (DC)

Son los más usados en la actualidad por su facilidad de control. En muchos casos, el propio motor incluye un codificador de posición (encoder) para poder realizar su control.

Los motores DC están constituidos por dos devanados internos, inductor e inducido, que se alimentan con corriente continua:

- El inductor, también denominado devanado de excitación, está situado en el estator y crea un campo magnético de dirección fija, denominado de excitación.
- El inducido, situado en el rotor, hace girar al mismo debido a la fuerza de Lorentz que aparece como combinación de la corriente circulante por él y del campo magnético de excitación. Recibe la corriente del exterior a través del colector de delgas, en el que se apoyan unas escobillas de grafito.

Para que se pueda realizar la conversión de energía eléctrica en energía mecánica de forma continua es necesario que los campos magnéticos del estator y del rotor permanezcan estáticos entre sí. Esta transformación es máxima cuando ambos campos se hallan en cuadratura.

El control de velocidad para estos motores se puede realizar de dos formas:

- Control por inducido. La intensidad del inductor se mantiene constante, mientras que la tensión del inducido se utiliza para controlar la velocidad de giro.
- Control por excitación. Se actúa al contrario que en la excitación por inducido.

Del estudio de ambos tipos de motores, y realizándose las simplificaciones correspondientes, se obtiene que la relación entre tensión de control y velocidad de giro (función de transferencia), responde a un sistema de primer orden en los controlados por inducido, mientras que en el caso de los motores controlados por excitación, esta relación es la de un segundo orden (Figura 1.5).

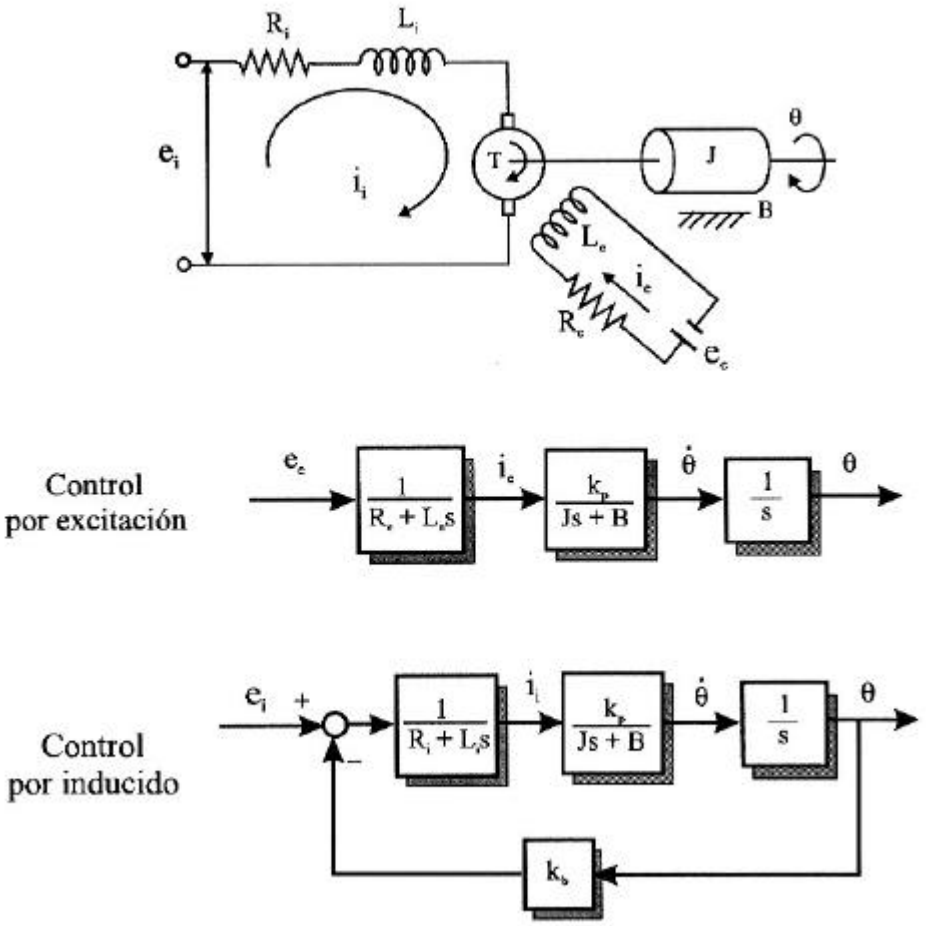

Figura 1.5. Motor DC. Esquema y funciones de transferencia.

Además, en los motores controlados por inducido se produce un efecto estabilizador de la velocidad de giro originado por la realimentación intrínseca que posee a través de la fuerza contraelectromotriz. Por estos motivos, de los motores DC es el controlado por inducido el que se usa en el accionamiento de robots.

Para mejorar el comportamiento de este tipo de motores, el campo de excitación se genera mediante imanes permanentes, con lo que se evitan fluctuaciones del mismo.

Las velocidades de rotación que se consiguen con estos motores son del orden de 1000 a 3000 r.p.m., con un comportamiento muy lineal y bajas constantes de tiempo. Las potencias que pueden manejar pueden llegar a los 10 kW.

El principal inconveniente de este tipo de motores es que apenas poseen masa térmica, lo que aumenta los problemas de calentamiento por sobrecarga.

1.5.3.3.3.2.- Motores paso a paso

En los últimos años se han desarrollado motores paso a paso con pares suficientemente grandes en pequeños pasos, lo que los ha hecho válidos para su uso en accionamientos industriales.

Existen tres tipos de motores paso a paso:

- De imanes permanentes.
- De reluctancia variable.
- Híbridos

En los primeros, de imanes permanentes (Figura 1.6), el rotor, que posee una polarización magnética constante, gira para orientar sus polos de acuerdo al campo magnético creado por las fases del estator. En los motores de reluctancia variable, el rotor es formado por un material ferromagnético que tiende a orientarse de modo que facilite el camino de las líneas de fuerza del

campo magnético generado por las bobinas del estator. Los motores híbridos combinan el modo de funcionamiento de los dos tipos anteriores.

En los motores paso a paso la señal de control son trenes de pulsos que van actuando rotativamente sobre una serie de electroimanes dispuestos en el estator. Por cada pulso recibido, el rotor del motor gira un determinado número discreto de grados.

Para conseguir el giro del rotor en un determinado número de grados, las bobinas del estator deben ser excitadas secuencialmente a una frecuencia que determina la velocidad de giro. Las inercias propias del arranque y parada impiden que el rotor alcance la velocidad nominal instantáneamente, por lo que ésta, y por tanto la frecuencia de los pulsos que la fija, debe ser aumentada progresivamente.

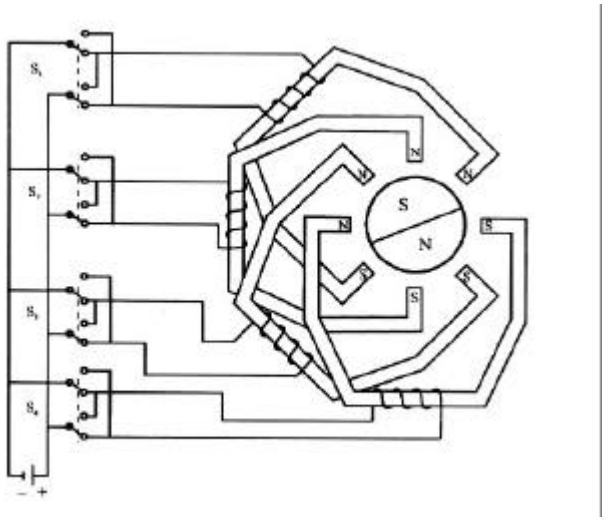

Figura 1.6. Esquema de un motor paso a paso de imanes permanentes con cuatro fases.

La principal ventaja respecto a los servomotores tradicionales es su capacidad para asegurar un posicionamiento simple y exacto. Pueden girar además de forma continua, con velocidad variable, como motores síncronos, ser sincronizados entre sí, obedecer a secuencias complejas de funcionamiento, etc. Se trata al mismo tiempo de motores muy ligeros, fiables y fáciles de controlar,

pues al ser cada estado de excitación del estator estable, el control se realiza en bucle abierto, sin la necesidad de sensores de realimentación.

Entre sus inconvenientes se puede citar que su funcionamiento a bajas velocidades no es suave, y que existe el peligro de pérdida de una posición por trabajar en bucle abierto. Tienden a sobrecalentarse trabajando a velocidades elevadas y presentan un límite en el tamaño que pueden alcanzar.

Su potencia nominal es baja y su precisión (mínimo ángulo girado) llega típicamente hasta 1,8º. Se emplean par el posicionamiento de ejes que no necesitan grandes potencias (giro de pinza) o para robots pequeños (educacionales); también son muy utilizados en dispositivos periféricos del robot, como mesas de coordenadas.

1.5.3.3.3.3.- Motores de corriente alterna (AC)

Estos motores no tenían aplicación en el campo de la robótica hasta hace unos años, debido fundamentalmente a la dificultad de su control. Sin embargo, las mejoras que se han introducido en las máquinas síncronas hacen que se presenten como un claro competidor a los motores de corriente continua. Esto se debe principalmente a tres factores:

- La construcción de rotores síncronos sin escobillas.
- Uso de convertidores estáticos que permiten variar la frecuencia (y así la velocidad de giro) con facilidad y precisión.
- Empleo de microelectrónica que permite una gran capacidad de control.

El inductor se sitúa en el rotor y está constituido por imanes permanentes, mientras que el inducido, situado en el estator, está formado por

tres devanados iguales decalados 120º eléctricos y se alimenta con un sistema trifásico de tensiones.

En los motores síncronos la velocidad de giro depende únicamente de la frecuencia de la tensión que alimenta al inducido. Para poder variar ésta con precisión, el control de velocidad se realiza mediante un convertidor de frecuencia. Para evitar el riesgo de pérdida de sincronismo se utiliza un sensor de posición continuo que detecta la posición del rotor y permite mantener en todo momento el ángulo que forman los campos del estator y el rotor. Este método de control se conoce como autosíncrono o autopilotado.

El motor síncrono autopilotado excitado con imán permanente, también llamado motor senoidal, no presenta problemas de mantenimiento debido a que no posee escobillas y tiene una gran capacidad de evacuación de calor, ya que los devanados están en contacto directo con la carcasa. El control de posición se puede realizar sin la utilización de un sensor externo adicional, aprovechando el detector de posición del rotor que posee el propio motor. Además permite desarrollar, a igualdad de peso, una potencia mayor que el motor de corriente continua. En la actualidad diversos robots industriales emplean este tipo de accionamientos con notables ventajas frente a los motores de corriente continua.

En el caso de los motores asíncronos, no se ha conseguido resolver satisfactoriamente los problemas de control que presentan. Esto ha hecho que hasta el momento no tengan aplicación en robótica.

1.5.3.4.- Sensores

Los sensores son dispositivos que permiten al robot tener conocimiento de su estado (sensores internos) y del estado de su entorno (sensores externos).

La información que la unidad de control puede obtener sobre el estado de su estructura mecánica es fundamentalmente la relativa a su posición y velocidad. En la Tabla 1.3 se resumen los sensores más comúnmente empleados para obtener información de presencia, posición y velocidad en robots industriales.

Tabla1.3. Tipos de sensores internos de robots.

Presencia	Inductivo Capacitivo Efecto hall Célula Reed Óptico Ultrasonido Contacto	
Posición	Analógicos	Potenciómetros Resolver Sincro Inductosyn LVDT
	Digitales	Encoders absolutos Encoders incrementales Regla óptica
Velocidad	Tacogeneratriz	

1.5.3.4.1.- Sensores de posición

1.5.3.4.1.1.- Sensores angulares

Para el control de posición angular se emplean fundamentalmente los denominados encoders y resolvers. Los potenciómetros dan bajas prestaciones por lo que no se emplean salvo en contadas ocasiones (robots educacionales, ejes de poca importancia).

Codificadores absolutos

Como puede verse en la Figura 1.7, un CA consta de **b** anillos, uno por bit. Cada anillo se divide en 2^b sectores, la mitad opacos y la otra mitad transparentes. Una fuente de luz (LED) y un fotodetector por bit se enfrentan, a cada lado del disco, radialmente. Cuando un sector opaco se encuentre entre un par, el fotodetector no recibe luz. Cuando un sector transparente lo hace, el fotodetector recibe luz. Estas dos situaciones corresponden a un '1' y un '0', respectivamente (o viceversa). Sólo un bit debe cambiar de estado a la vez para que la operación sea más fiable. Por esto motivo, estos discos son grabados con un código binario cíclico, normalmente código Gray. La resolución del decodificador es, en grados,

$$R = \frac{360^o}{2^b}$$

La principal desventaja de este tipo de codificador, es que para obtener una resolución relativamente pequeña, hace falta un gran número de bits, y en consecuencia de anillos concéntricos, con la dificultad que esta implementación física supone. La única ventaja de este tipo de codificador es que, al tener 'b' salidas en paralelo, no sería necesario una interfaz especial para entrar datos al

microprocesador. Sin embargo con los circuitos integrados actuales, esta ventaja prácticamente desaparece.

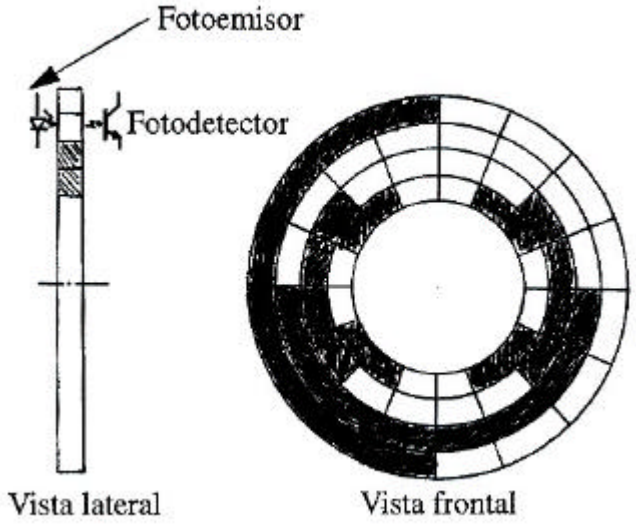

Figura 1.7. Codificador absoluto.

Codificadores incrementales

Un codificador de disco incremental tiene un solo anillo de 'ventanas' o 'ranuras', situado lo más cerca posible de la periferia del disco, ver Figura 1.8. Además, también tiene una ranura adicional situada en un radio menor, que se usa para determinar la posición de cero del disco. Para un ancho mínimo de ranura (determinado mediante condiciones mecánicas de construcción), el número de ranuras es directamente proporcional al radio del disco y, por supuesto, la resolución es proporcional al número de ranuras. Las ranuras y el espacio entre ellas tienen el mismo ancho. De este modo la onda cuadrada producida es simétrica. El disco está contenido en un módulo que contiene, además, el sensor consistente en un par emisor / detector de luz para el conjunto de ranuras y otro para la ranura de posición.

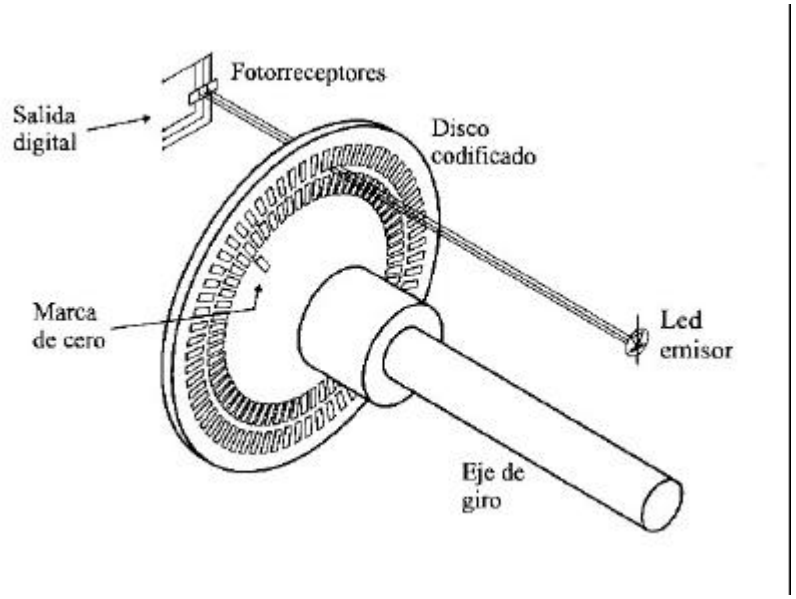

Figura 1.8. Codificador óptico (encoder) incremental.

Existen dos tipos de codificadores incrementales: lectura directa e interpolación electrónica. En los de lectura directa se detecta transiciones de '1' a '0' y de '0' a '1'. En los de interpolación electrónica, como el nombre indica, se interpola entre transiciones, basándose en una velocidad constante entre transición y transición.

En un disco de lectura directa, el emisor de luz consiste generalmente en un solo LED cuya luz es colimada por medio de una lente (contenido en el módulo). El circuito detector integrado consiste en un conjunto de fotodetectores, dos por canal. La distancia entre ranuras es tal que cuando una ranura se encuentra frente a un par de detectores, una "barra" se encuentra frente al par adyacente. El circuito condicionador, también incluido en el módulo, hace que las dos salidas se encuentren en cuadratura (90º eléctricos).

Como se ve, el codificador incremental es mucho más práctico que el absoluto. Sólo necesita dos sensores (más el de posición) independientemente del número de ranuras y permite lograr una resolución muchísimo más alta que el codificador absoluto. Además, permite determinar de forma sencilla el

sentido de rotación: cuando gira en un sentido, la salida A adelanta a la B en 90º, cuando gira en sentido opuesto, la B adelanta a la A en 90º.

El ángulo de rotación se determina contando el número de pulsos o el de transiciones de los canales A y B. Sea el número de ranuras N y el de sensores S (sin contar el de referencia). Hay N pulsos y 2N transiciones por revolución. Si sólo se cuentan los pulsos, la resolución es

$$R1 = \frac{360^o}{N \cdot S} \text{ grados}$$

Si se cuentan transiciones,

$$R2 = \frac{360^o}{2 \cdot N \cdot S} \text{ grados}$$

Normalmente los sensores de posición se acoplan al eje del motor. Considerando que en la mayor parte de los casos entre el eje del motor y el de la articulación se sitúa un reductor de relación N, cada movimiento de la articulación se vera multiplicado por N al ser medido por el sensor. Esto aumentará así su resolución multiplicándola por N.

Captadores angulares de posición (sincro-resolvers)

Se trata de captadores analógicos con resolución teóricamente infinita (aunque en la práctica viene depende de la electrónica asociada). El funcionamiento de los resolvers se basa en la utilización de una bobina solidaria al eje excitada por una portadora, generalmente de 400 Hz, y por dos bobinas fijas situadas a su alrededor (Figura 1.9).

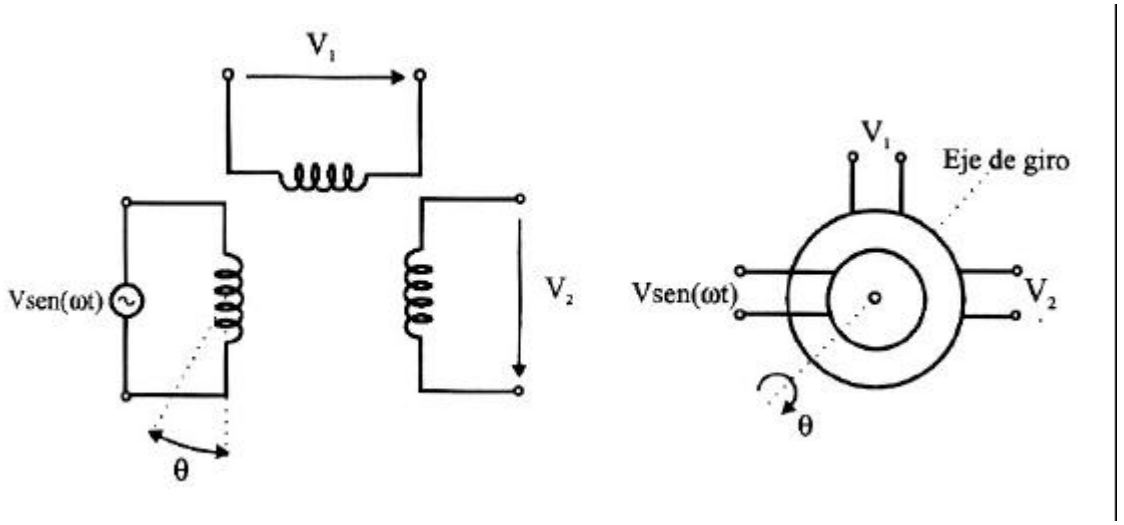

Figura 1.9. Esquema de funcionamiento de un resolver.

El giro de la bobina móvil hace que el acoplamiento con las bobinas fijas varíe, consiguiendo que la señal resultante es éstas dependa del seno del ángulo de giro. La bobina móvil excitada con tensión Vsen(ωt) y girada un ángulo θ induce en las bobinas fijas situadas en cuadratura las siguientes tensiones:

$$V_1 = V \cdot sen(\omega t) \cdot sen\theta$$
$$V_2 = V \cdot sen(\omega t) \cdot \cos\theta$$

que es la llamada representación de θ en formato resolver.

El funcionamiento de los sincros es análogo al de los resolvers, excepto que las bobinas fijas forman un sistema trifásico de estrella. Para un ángulo θ de la bobina móvil excitada con tensión Vsen(ωt), admitiendo que los acoplamientos y los desfases son los mismos para todos los devanados, se obtienen las siguientes tensiones entre las fases del estator:

$$V_{13} = \sqrt{3} V \cdot sen(\omega t) \cdot sen\theta$$
$$V_{32} = \sqrt{3} V \cdot sen(\omega t) \cdot sen(\theta + 120^o)$$
$$V_{21} = \sqrt{3} V \cdot sen(\omega t) \cdot sen(\theta + 240^o)$$

que es la llamada representación del ángulo θ en formato sincro.

El cambio de formato sincro a formato resolver o viceversa es inmediato, ya que se puede pasar de uno a otro a través de la llamada red de Scott, de funcionamiento bidireccional.

Para poder tratar en el sistema de control la información generada por los resolvers y los sincros es necesario convertir las señales analógicas en digitales. Para ello se utilizan los llamados convertidores resolver / digital (R / D), que tradicionalmente se basan en dos tipos de estructuras distintas: seguimiento (tracking) y muestreo (sampling).

Ambos captadores son de tipo absoluto en cada vuelta del eje acoplado a ellos. Entre sus ventajas destacan su buena robustez mecánica durante el funcionamiento y su inmunidad a contaminación, humedad, altas temperaturas y vibraciones. Debido a su reducido momento de inercia, imponen poca carga mecánica al funcionamiento del eje.

1.5.3.4.1.2.-Sensores lineales de posición

Codificadores incrementales lineales

También conocidos como reglas ópticas, un CIL se usa para medir desplazamientos lineales. Existe una gran cantidad de tipos y capacidad de alcance: desde unos pocos centímetros a más de dos metros de desplazamiento. Puede lograrse con ellos una gran precisión del orden de nanómetros.

El CIL suele usarse cuando en un robot se utilizan actuadores hidráulicos, ya que éstos tienen un movimiento inherentemente lineal. Un CIL consiste de dos partes principales: el emisor / detector y una *cinta-codificadora*

lineal. El principio de funcionamiento de los CIL es el mismo que el de los codificadores incrementales de disco.

Reglas magnéticas o Inductosyn

El funcionamiento es similar al del resolver con la diferencia que el rotor desliza linealmente sobre el estator, siendo la forma de los devanados la representada en la Figura 1.10. El estator se encuentra excitado por una tensión conocida que induce en el rotor dependiendo de su posición relativa una tensión V_S:

$$V_s = kV\cos\left(2\pi\frac{x}{P}\right)$$

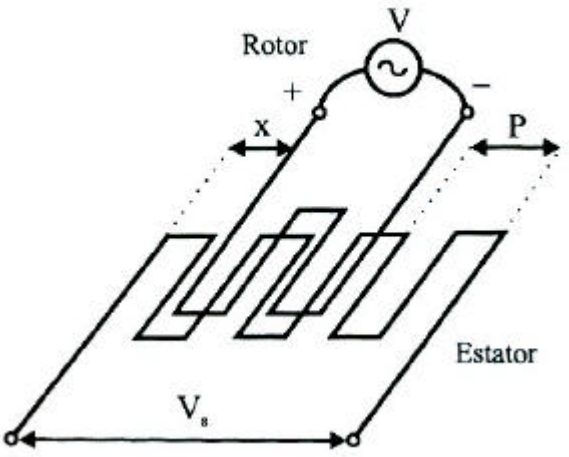

Figura 1.10. Esquema de funcionamiento de un Inductosyn.

LVDT (Transformador diferencial de variación lineal)

El transformador diferencial de variación lineal (LVDT) destaca debido a su casi infinita resolución, poco rozamiento y alta repetibilidad. Su funcionamiento se basa en la utilización de un núcleo de material ferromagnético unido al eje cuyo movimiento se quiere medir. Este núcleo se mueve linealmente entre un devanado primario y dos secundarios, haciendo con su movimiento que varíe la inductancia entre ellos. La Figura 1.11 presenta un breve esquema de su funcionamiento.

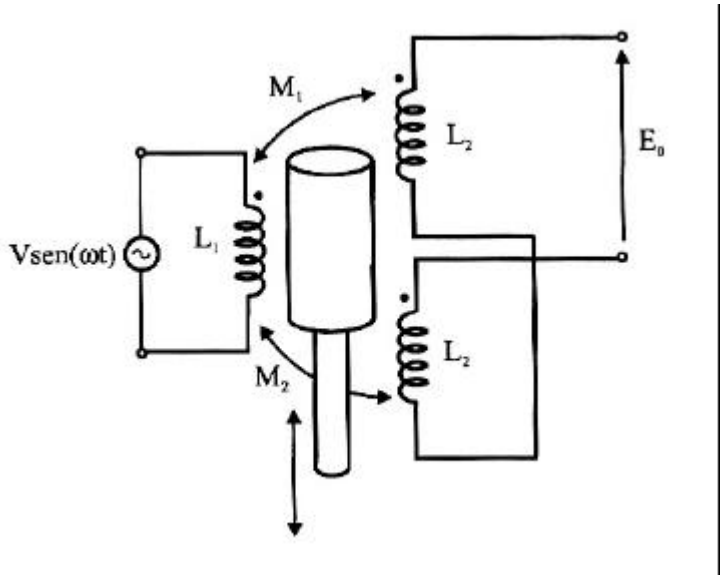

Figura 1.11. Esquema de funcionamiento de un LVDT.

Los dos devanados secundarios conectados en oposición serie ven como la inducción de la tensión alterna del primario, al variar la posición del núcleo, hace crecer la tensión en un devanado y disminuirla en el otro. Del estudio de la tensión E_0 se deduce que ésta es proporcional a la diferencia de inductancias mutuas entre el devanado primario con cada uno de los secundarios, y que por tanto depende linealmente del desplazamiento del vástago solidario al núcleo.

Además de las ventajas señaladas, el LVDT presenta una alta linealidad, gran sensibilidad y una respuesta dinámica elevada. Su uso está ampliamente

extendido, a pesar del inconveniente de poder ser aplicado únicamente en la medición de pequeños desplazamientos.

1.5.3.4.2.- Sensores de velocidad

La captación de la velocidad se hace necesaria para mejorar el comportamiento dinámico de los actuadores del robot.

Normalmente, y puesto que el bucle de control de velocidad es analógico, el captador usado es una tacogeneratriz que proporciona una tensión proporcional a la velocidad de giro de su eje (valores típicos pueden ser 10 milivoltios por rpm).

Otra posibilidad, usada para el caso en que la unidad de control del robot precise valorar la velocidad de giro de las articulaciones, consiste en derivar la información de posición que ésta posee.

1.5.3.4.3.- Sensores de presencia

Este tipo de sensor es capaz de detectar la presencia de un objeto dentro de un radio de acción determinado. Esta detección puede hacerse con o sin contacto con el objeto. En el segundo caso se utilizan diferentes principios físicos para detectar la presencia, dando lugar a los diferentes tipos de captadores (ver Tabla 1.3). En el caso de detección con contacto, se trata siempre de un interruptor, normalmente abierto o normalmente cerrado según interese, actuado mecánicamente a través de un vástago u otro dispositivo. Los detectores de presencia se utilizan en robótica principalmente como auxiliares

de los detectores de posición, para indicar los límites de movimiento de las articulaciones y permitir localizar la posición de referencia de cero de éstos en el caso que sean incrementales.

Además de esta aplicación, los sensores de presencia se usan como sensores externos, siendo muy sencillos de incorporar al robot por su carácter binario y su costo reducido. Los detectores inductivos permiten detectar la presencia o contar el número de objetos metálicos sin necesidad de contacto. Presentan el inconveniente de distinto comportamiento según el material de que se trate. El mismo tipo de aplicación tienen los detectores capacitivos, más voluminosos, aunque en este caso los objetos a detectar no precisan ser metálicos. En cambio presentan problemas de trabajo en condiciones húmedas y con puestas a tierra defectuosas.

Los sensores basados en el efecto may detectan la presencia de objetos ferromagnéticos por la deformación que éstos provocan sobre un campo magnético. Los captadores ópticos, sin embargo, pueden detectar la reflexión del rayo de luz procedente del emisor sobre el objeto.

1.5.3.5.- Sistema de control

El sistema de control de un robot se puede descomponer en dos subsistemas de control:

- Control cinemático
- Control dinámico

El objetivo fundamental de ambos subsistemas es el de poder establecer las adecuadas estrategias de control para obtener una mayor calidad de los

movimientos del robot en cuanto a posicionamiento, precisión y velocidad se refiere.

El control cinemático establece cuáles son las trayectorias que debe seguir cada articulación del robot a lo largo del tiempo para lograr los objetivos fijados por el usuario (punto de destino, trayectoria cartesiana del efector final, tiempo invertido, etc.) basándose en el modelo cinemático del robot. Estas trayectorias se seleccionarán atendiendo a las restricciones físicas propias de los acontecimientos y a ciertos criterios de calidad de trayectoria, como suavidad o precisión de la misma.

El control dinámico tiene por misión procurar que las trayectorias realmente seguidas por el robot sean lo más parecidas posibles a las propuestas por el control cinemático. Para ello hace uso del modelo dinámico del robot y de las herramientas de análisis y diseño aportadas por la teoría de servocontrol (representación interna, estado, estabilidad de Lyapunov, control PID, control adaptativo, etc.).

1.5.3.5.1.- Control cinemático

La Figura 1.12 muestra de manera esquemática el funcionamiento del control cinemático. Recibe como entrada los datos procedentes del programa del robot escrito por el usuario (puntos de destino, precisión, tipo de trayectoria deseada, velocidad o tiempo invertido, etc.) y apoyándose en el modelo cinemático del robot establece las trayectorias para cada articulación como funciones de tiempo. Estas trayectorias deben ser muestreadas con un período T a decidir, generándose en cada instante kT un vector de referencias articulares para los algoritmos de control dinámico.

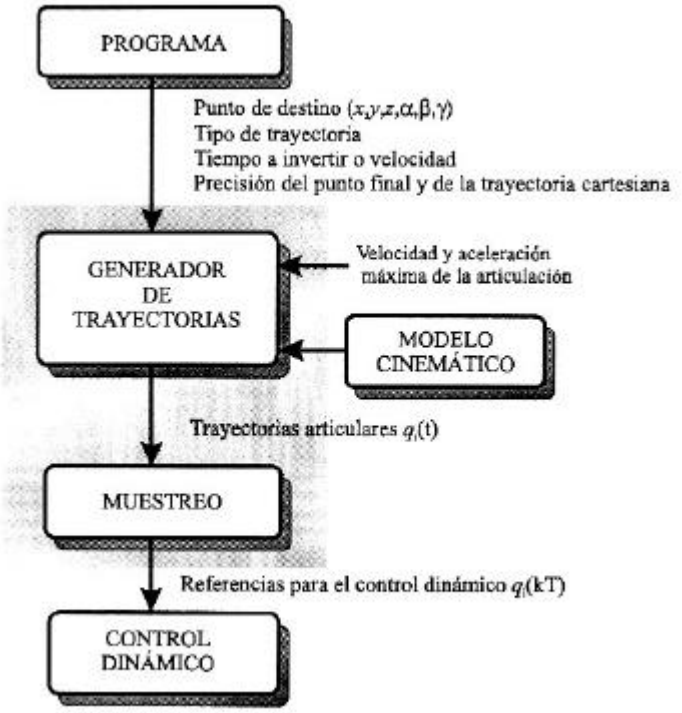

Figura 1.12. Diagrama de funcionamiento del control cinemático.

De manera general, el control cinemático realiza las siguientes funciones:

1. Convertir la especificación del movimiento dada en el programa en una trayectoria analítica en espacio cartesiano (evolución de cada coordenada cartesiana en función del tiempo).
2. Muestrear la trayectoria cartesiana obteniendo un número finito de puntos de dicha trayectoria. Cada uno de estos puntos vendrá dado por una 6-upla, típicamente (x, y , z, α, β, γ).
3. Utilizando la transformación homogénea inversa, convertir cada uno de estos puntos en sus correspondientes coordenadas articulares (q_1, q_2, q_3, q_4, q_5, q_6). Debe tenerse en cuenta aquí la posible solución múltiple de la transformación homogénea inversa, así como la posibilidad de ausencia de solución y puntos singulares, de modo que se asegure la continuidad de la trayectoria.
4. Interpolación de los puntos articulares obtenidos, generando para cada variable articular una expresión $q_i(t)$ que pase o se aproxime a ellos de

modo que, siendo una trayectoria realizable por los actuadores, se transforme en una trayectoria cartesiana lo más próxima a la especificada por el programa usuario (en cuanto a precisión, velocidad, etc.).

5. Muestreo de la trayectoria articular para genera referencias de control dinámico.

El principal inconveniente del procedimiento descrito para la generación de trayectorias radica en la necesidad de resolver repetidamente la transformada homogénea inversa, lo que conlleva un elevado coste conceptual. Como alternativa puede ser considerado un procedimiento basado en la utilización de la matriz Jacobiana.

La matriz Jacobiana establece las relaciones diferenciales entre variables articulares y cartesianas en términos de la ecuación:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_x}{\partial q_1} & \cdots & \frac{\partial f_x}{\partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_g}{\partial q_1} & \cdots & \frac{\partial f_g}{\partial q_n} \end{bmatrix} \cdot \begin{bmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_n \end{bmatrix} \qquad [1.1]$$

de manera compacta puede expresarse como:

$$\dot{j}(t) = J(q) \cdot \dot{q}(t) \qquad [1.2]$$

Esta relación es lineal y para una localización **q** determinada permite conocer las velocidades cartesianas a partir de las correspondientes articulares. Además suponiendo **J**(q) invertible, lo que ocurriría siempre que las dimensiones del espacio articular y de la tarea sean iguales y **q** no sea un punto singular, se podrá expresar:

$$\dot{q}(t) = J^{-1}(q) \cdot \dot{j}(t) \qquad [1.3]$$

De este modo, a partir de la trayectoria cartesiana deseada para el extremo del robot **j**(t) podrá obtenerse su derivada y, a través de la relación anterior, la velocidad que debe seguir cada una de las articulaciones. Debe considerarse que esta relación es lineal y, por lo tanto, más fácil de obtener que la que utiliza la transformación homogénea. No hay que olvidar, sin embargo, que **J**(q) varía para cada localización de **q** considerada, siendo por tanto necesario actualizar continuamente los valores de los elementos de la matriz **J**.

La implementación de este procedimiento en la unidad de control de un robot exigirá, una vez más, la discretización de la trayectoria **j**(t) y su derivada en un número finito de puntos que posteriormente, y tras ser convertidos a velocidades articulares deberán ser interpolados.

1.5.3.5.2.- Control dinámico

Como ya se mencionó, el control dinámico tiene por objetivo que las trayectorias reales seguidas por el robot sean lo más aproximadas posibles a las propuestas por el control cinemático. Existen múltiples métodos de control de robots, ya sea por bucle cerrado, PID, control adaptativo o redes neuronales. Algunos de ellos son muy complejos y se pueden encontrar en cualquier libro sobre control automático. Sin embargo, los más usados en robótica son el control de posición por bucle cerrado de cada articulación, y el control por momento calculado.

1.5.3.6.- Tipos de trayectoria

1.5.3.6.1.- Trayectorias punto a punto

En este tipo de trayectoria cada articulación evoluciona desde su posición inicial a la final sin realizar consideración alguna sobre el estado o evolución de las demás articulaciones. Normalmente, cada actuador trata de llevar a su articulación al punto de destino en el menor tiempo posible, pudiéndose distinguir dos casos:

- Movimiento eje a eje: Sólo se mueve un eje cada vez. Comenzará a moverse la primera articulación, y una vez ésta haya alcanzado su punto final lo hará la segunda, y así sucesivamente.

- Movimiento simultáneo de ejes: Todos los actuadores comienzan simultáneamente a mover las articulaciones del robot a una velocidad específica para cada una de ellas. Dado que la distancia a recorrer y las velocidades serán en general diferentes, cada una acabará su movimiento en un instante diferente.

Las trayectorias punto a punto están implementadas en robots muy simples (educacionales, etc.) o con unidades de control muy limitadas.

1.5.3.6.2.- Trayectorias coordinadas o isócronas

En este tipo de trayectorias, se trata de conseguir que el movimiento de todas las articulaciones empieze y finalice al mismo tiempo. Para ello, se realiza un cálculo previo para averiguar que articulación es la que más tiempo

invertirá. Entonces se ralentiza el movimiento de los otro ejes para que inviertan el mismo tiempo en su movimiento, acabando todos ellos simultáneamente.

El tiempo total invertido es el menor posible y no se piden aceleraciones y velocidades elevadas a los actuadores de manera inútil. Desde el punto de vista del usuario, la trayectoria que describe el extremo del robot no es significativa, siendo ésta impredecible.

1.5.3.6.3.- Trayectorias continuas

En este tipo de trayectorias, el extremo del robot sigue una trayectoria conocida por el usuario en el espacio cartesiano o de la tarea. Típicamente, las trayectorias que el usuario desea que el robot describa son trayectorias en línea recta o en arco de círculo. Para ello, es preciso calcular de manera continua las trayectorias articulares.

El resultado será que cada articulación sigue un movimiento aparentemente caótico con posibles cambios de dirección y velocidad y sin coordinación con el resto de las articulaciones. Sin embargo, el resultado del conjunto será que el extremo del robot describirá la trayectoria deseada.

Normalmente el usuario del robot indica el movimiento que éste debe realizar especificando las localizaciones espaciales inicial y final junto con otros datos, como velocidad y tipo de trayectoria. Puesto que estos puntos están, por lo general, excesivamente separados es preciso seleccionar puntos intermedios suficientemente cercanos como para que el control del robot consiga ajustar no sólo el punto final especificado, sino también la trayectoria seguida a la indicada en el programa.

Para ello, es preciso establecer un interpolador entre las localizaciones expresadas en el espacio de la tarea que dará como resultado una expresión analítica de la evolución de cada coordenada.

Para evitarse las discontinuidades de velocidad en el caso de paso por varios puntos, pueden utilizarse las técnicas de interpoladores a tramos o interpoladores cúbicos para el caso de variables articulares.

1.5.3.7.- Elementos terminales

Los elementos terminales, también llamados efectores finales (end effector) son los encargados de interaccionar directamente con el entorno del robot. Pueden ser tanto elementos de aprehensión como herramientas.

Si bien un mismo robot industrial es, dentro de unos límites lógicos, versátil y readaptable a una gran variedad de aplicaciones, no ocurre así con los elementos terminales, que son en muchos casos específicamente diseñados para cada tipo de trabajo.

Se puede establecer una clasificación de los elementos terminales atendiendo a si se trata de un elemento de sujeción o de una herramienta. Los primeros se pueden clasificar según el sistema de sujeción empleado. En la Tabla 1.4 se presentan estas opciones, así como los usos más frecuentes.

Los elementos de sujeción se utilizan para agarrar y sostener los objetos y se suelen denominar pinzas. Se distingue entre las que utilizan dispositivos de agarre mecánico, y las que utilizan algún otro tipo de dispositivo (ventosas, pinzas magnéticas, adhesivas, ganchos, etc.).

Tabla 1.4. Sistemas de sujeción para robots.

Tipos de sujeción	Accionamiento	Uso
Pinza de presión .desp. angular .desp. lineal	Neumático o eléctrico	Transporte y manipulación de piezas sobre las que no importe presionar
Pinza de enganche	Neumático o eléctrico	Piezas de grandes dimensiones o sobre las que no se puede ejercer presión
Ventosas de vacío	Neumático	Cuerpos con superficie lisa poco porosa (cristal, plástico, etc.)
Electroimán	Eléctrico	Piezas ferromagnéticas

En la elección o diseño de una pinza se han de tener en cuenta diversos factores. Entre los que afectan al tipo de objeto y de manipulación a realizar destacan el peso, la forma, el tamaño del objeto y la fuerza que es necesario ejercer y mantener para sujetarlo. Entre los parámetros de la pinza cabe destacar su peso (que afecta a las inercias del robot), el equipo de accionamiento y la capacidad de control.

El accionamiento neumático es el más utilizado por ofrecer mayores ventajas en simplicidad, precio y fiabilidad, aunque presenta dificultades de control de posiciones intermedias. En ocasiones se utilizan accionamientos de tipo eléctrico.

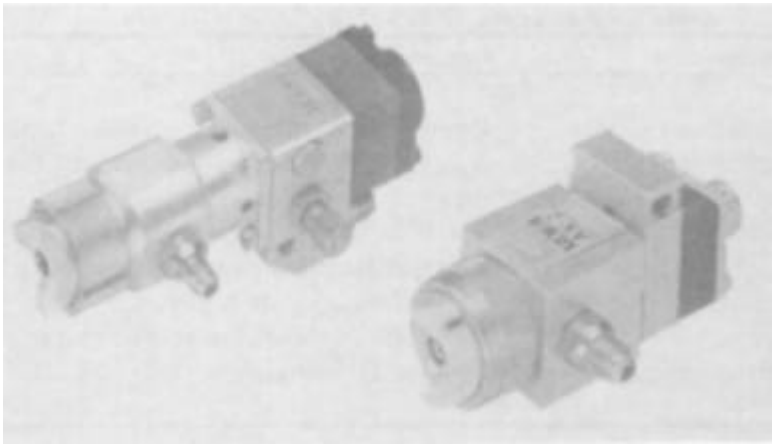

Figura 1.13. Pistolas neumáticas de pulverización de pintura.

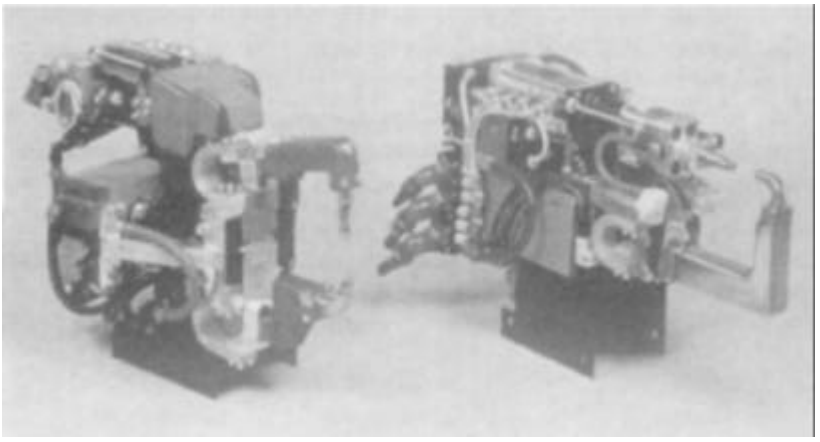

Figura 1.14. Pinzas de soldadura con transformador incorporado.
De accionamiento por tijera y rectilíneo.

En la pinza se suelen situar sensores para detectar el estado de la misma (abierto o cerrado). Se pueden incorporar a la pinza otro tipo de sensores para controlar el estado de la pieza, sistemas de visión que proporcionen datos geométricos de los objetos, detectores de proximidad, sensores fuerza-par, etc.

Tabla 1.5. Herramientas terminales para robots.

Tipo de herramienta	Comentarios
Pinza soldadura por puntos	Dos electrodos que se cierran sobre la pieza a soldar.
Soplete soldadura al arco	Aportan el flujo de electrodo que se funde.
Cucharón para colada	Para trabajos de fundición.
Atornillador	Suelen incluir la alimentación de tornillos.
Fresa-lija	Para perfilar, eliminar rebabas, pulir, etc.
Pistola pintura	Por pulverización de la pintura.
Cañón láser	Para corte de materiales, soldadura o inspección.
Cañón de agua a presión	Para corte de materiales.

En muchas aplicaciones el robot ha de realizar operaciones que no consisten en manipular objetos, sino que implican el uso de una herramienta. El tipo de herramientas con las que puede dotarse a un robot es muy amplio. La

Figura 1.13 y la Figura 1.14 muestran, respectivamente dos pistolas de pulverización de pintura y dos pinzas de soldadura por puntos. Normalmente la herramienta está fijada rígidamente al extremo del robot aunque en ocasiones se dota a éste de un dispositivo de cambio automático, que permita al robot usar herramientas diferentes durante su tarea. La Tabla 1.5 enumera algunas de las herramientas más frecuentes.

1.5.4.- Programación de robots

La programación es el proceso mediante el cual se le indica paso a paso al robot la secuencia de acciones que deberá llevar a cabo durante la realización de su tarea. Estas acciones consisten en su mayor parte en moverse a puntos predefinidos y manipular objetos del entorno.

En la actualidad no existe estandarización alguna en cuanto a procedimientos de programación de robots. Por el contrario, cada fabricante ha desarrollado su particular método, válido únicamente para sus propios robots. Sin embargo, muchos de estos métodos de programación son compartidos por diferentes tipos de robots (salvando pequeñas diferencias) o han servido de modelo para el desarrollo de otros.

Los métodos de programación más usados, clasificados según el sistema empleado para indicar la secuencia de acciones a realizar son:

- Programación por guiado
- Programación textual

1.5.4.1.- Métodos de programación por guiado

1.5.4.1.1.- Guiado pasivo directo

Con los actuadores del robot desconectados, el propio programador toma el extremo del robot y lo lleva hasta los puntos deseados siguiendo las trayectorias más adecuadas. La unidad de control del robot registrará, de manera automática, la señal de los sensores de posición de las articulaciones en todos los puntos recorridos. En este tipo de programación, el programador debe aportar directamente la energía para mover el robot.

1.5.4.1.2.- Guiado pasivo por maniquí (guiado por Maestro-Esclavo)

El sistema de programación es idéntico que en el guiado pasivo directo, pero por la dificultad física de mover toda la estructura del robot, se utiliza un doble del robot que permanece fuera de línea. Este doble (maniquí o maestro) tiene la misma configuración y dimensiones que el que esta en línea y se desea programar (esclavo), pero éste es más ligero y fácil de mover al no tener actuadores, solamente tiene sensores para registrar su posición.

1.5.4.1.3.- Guiado activo

En este tipo de programación se utiliza la botonera de enseñanza del propio robot o un joystick. De esta forma, es el propio robot el que mueve sus articulaciones y la unidad de control únicamente registra aquellas configuraciones del robot que el programador indica expresamente.

1.5.4.2.- Programación textual

El método de programación textual permite indicar la tarea del robot mediante el uso de un lenguaje de programación específico. El programa se corresponde con una serie de órdenes que son editadas y posteriormente ejecutadas.

La programación textual puede ser clasificada en tres niveles: robot, objeto y tarea, dependiendo de si las órdenes se refieren a los movimientos a realizar por el robot, el estado en que deben ir quedando los objetos manipulados o al objetivo a conseguir.

Actualmente, la programación textual se queda a nivel de robot, existiendo una gran cantidad de lenguajes de programación a este nivel. (AL, AML, LM, AL II, V+, RAPID, etc.).

1.5.5.- Otros parámetros importantes en robótica

Resolución: Mínimo incremento que puede aceptar la unidad de control del robot. Su valor esta limitado por la resolución de los captadores de posición y convertidores A/D y D/A, por el número de bits con los que se realizan las operaciones aritméticas en la CPU y por los elementos motrices, si estos son discretos (motores paso a paso, sistemas neumáticos todo o nada, etc.).

Precisión: Distancia entre el punto programado (normalmente de manera textual) y el valor medio de los puntos realmente alcanzados al repetir el movimiento varias veces con carga y temperatura nominales. Su origen se debe a errores en la calibración del robot (punto de sincronismo por ejemplo), deformaciones por origen térmico y dinámico, errores de redondeo en el cálculo de la transformación cinemática (especialmente en las carencias de puntos singulares), errores entre las dimensiones reales y teóricas del robot, etc.

Repetibilidad: Radio de la esfera que abarca los puntos alcanzados por el robot tras suficientes movimientos, al ordenarle ir al mismo punto de destino programado, con condiciones de carga, temperatura, etc., iguales. (Normalmente se considera la banda que abarca el 99% de los puntos respecto a la media). El error de repetibilidad es debido fundamentalmente a problemas en el sistema mecánico de transmisión como rozamientos, histéresis, zonas muertas (backlash).

1.6.- Descripción técnica

En este capítulo se detallarán las especificaciones técnicas tanto del robot usado, como de los distintos elementos que han intervenido en el desarrollo del proyecto.

1.6.1.- Robot ESCORBOT ER-III

Se trata de un robot educacional con 5 grados de libertad creado por la empresa Eshed Robotec, y distribuido en España por Creatividad y Tecnología S.A.

El sistema esta formado por el brazo articulado (robot) y la unidad de control. (Figura 1.15).

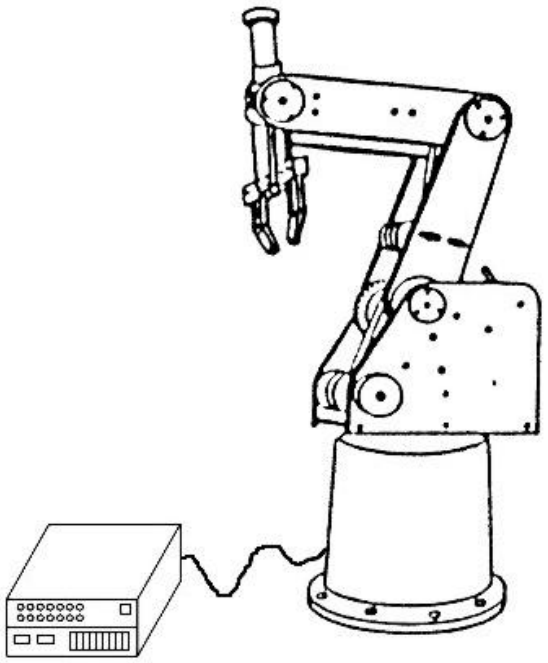

Figura 1.15. Robot ESCORBOT ER-III.

Las especificaciones técnicas dadas por el fabricante se pueden observar en la Tabla 1.6:

Tabla 1.6. Especificaciones técnicas del ESCORBOT ER-III

Alimentación	110 / 220 V. AC 60 / 50 Hz
Número de ejes	
Brazo	5 más pinza
Controlador	7 más pinza
Capacidad de carga	1 kg.
Repetibilidad	± 0.5 mm.
Velocidad máxima de giro	
Cuerpo (base)	21.4 grados / seg.
Hombro	34 grados / seg.
Codo	34 grados / seg.
Elevación	120 grados / seg.
Rotación	85 grados / seg.
Abertura máxima de la pinza	75 mm. Con almohadillas
	65 mm. Sin almohadillas
Envolvente de trabajo	
Articulación del cuerpo	340 grados
Articulación del hombro	170 grados
Articulación del codo	300 grados
Articulación de elevación	300 grados
Articulación de rotación	Ilimitada
Actuadores	6 servomotores DC
Realimentación	Codificadores ópticos incrementales en todos los ejes
Transmisión	Engranajes y correas dentadas
Interfaz de comunicaciones	RS-232-C
Hard Home	Referencias fijas de posición en todos los ejes
Dimensiones	
Altura máxima	930 mm.
Radio máximo de trabajo	610 mm.
Peso	
Brazo del robot	16 kg.
Controlador	3 kg.

La Figura 1.16 muestra la envolvente de trabajo para del brazo.

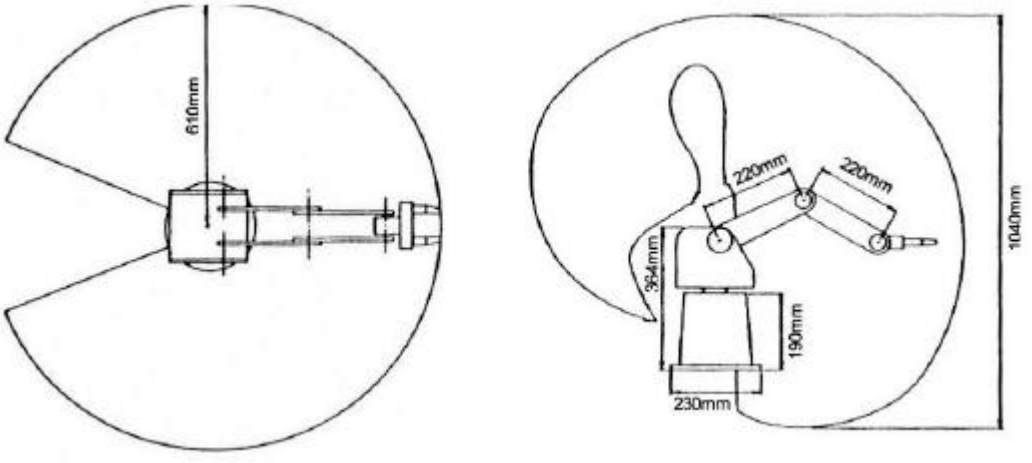

Figura 1.16. Envolvente de trabajo del ESCORBOT ER-III.

1.6.1.1.- Brazo mecánico

El brazo mecánico está constituido por eslabones y articulaciones, manteniendo cierto paralelismo con la estructura del brazo humano. Tiene 5 grados de libertad y una pinza de dos dedos como efector final (Figura 1.17).

Las articulaciones del brazo mecánico se describen en la Tabla 1.7.

Tabla 1.7. Articulaciones del ESCORBOT ER-III

Articulación Número	Nombre de la articulación	Tipo de movimiento de la articulación	Motor número
1	Base	Giratorio (enrollado)	1
2	Hombro	Giratorio (enrollado)	2
3	Codo	Giratorio (enrollado)	3
4	Muñeca (elevación)	Giratorio (enrollado)	4+5
5	Muñeca (giro)	Giratorio (enrollado)	4+5
6	Pinza	Lineal (prismático)	8

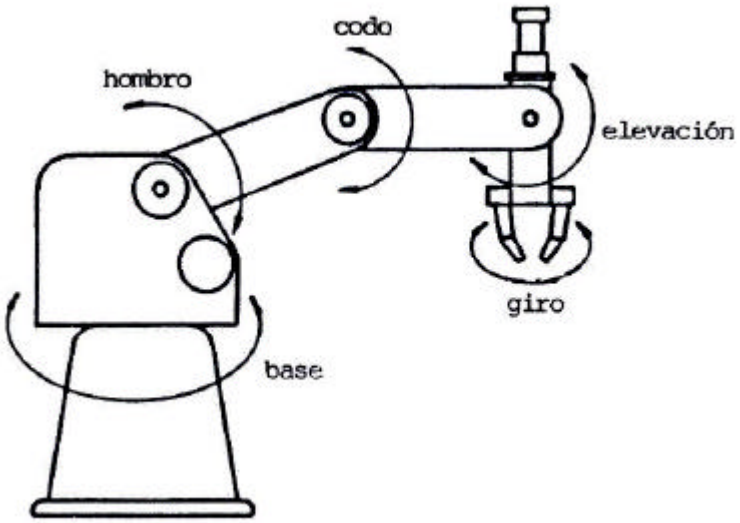

Figura 1.17. Robot ESCORBOT ER-III.

Las dimensiones de los eslabones del ESCORBOT ER-III, tomadas desde los ejes de las articulaciones, son las siguientes:

Tabla 1.8. Dimensiones de los eslabones.

Eslabón	Dimensión
Brazo superior	220 mm.
Antebrazo	220 mm.
Distancia desde la muñeca al extremo de la pinza	150 mm.
Altura del hombro (altura de la base + altura del cuerpo)	340 mm
Distancia horizontal entre el eje de la junta del hombro y el eje de la base del robot	16 mm.

Estos parámetros son necesarios a la hora de calcular el modelo cinemático del robot por el método de Denavit-Hartenberg, como se describe en la memoria de cálculo.

1.6.1.2.- Motores de accionamiento del robot

Todas las articulaciones del brazo son movidas por motores DC de excitación por imán permanente. El sentido de giro del motor se determina por la polaridad de la tensión de funcionamiento.

El ESCORBOT ER-III tiene montados 6 motores DC, de tres tipos diferentes. Todos ellos con tensión de alimentación de ± 12 V, y con un sistema de reductores para aumentar el par y reducir la velocidad.

Tabla 1.9. Tipos de motores del ESCORBOT ER-III.

Motores	Marca, modelo, velocidad y relación de transmisión
Base, hombro y codo	PITTMAN GM9413F208 ω=2412 rpm Tm=127'7:1
Elevación y giro de la muñeca	PITTMAN GM9413F207 ω=2412 rpm Tm=65'5:1
Pinza	PITTMAN GM8712E405 ω=2412 rpm Tm=19'5:1

Cada motor del robot tiene control mediante lazo cerrado. Esta realimentación es efectuada por los encoders ópticos, que facilitan información al controlador sobre la cantidad de movimiento y el sentido de giro.

Tanto los encoders como los reductores están ensamblados con el propio motor, formando un único elemento, como puede apreciarse en la Figura 1.18.

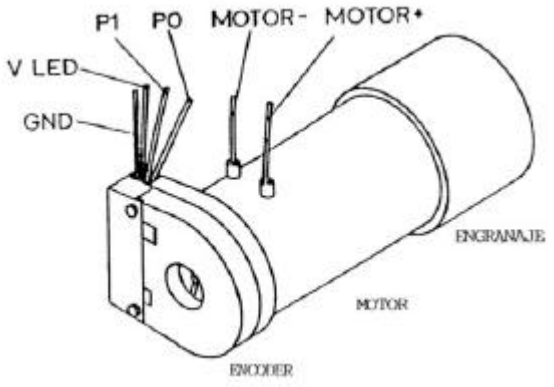

Figura 1.18. Sistema motor del ESCORBOT ER-III.

1.6.1.3.- Sistema de transmisión

La transmisión en el ESCORBOT ER-III es del tipo acoplamiento indirecto, ya que ésta se realiza mediante engranajes, correas dentadas y tornillo sin fin, en el caso de la pinza.

1.6.1.3.1.- Transmisión de la articulación de la base

La transmisión de la base se realiza mediante dos engranajes de distinto tamaño, tal como se muestra en la Figura 1.19.

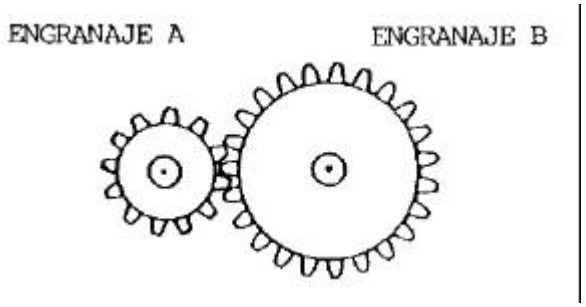

Figura 1.19. Transmisión indirecta de una etapa.

El engranaje ‘a’ está acoplado directamente al eje de salida del motor y gira con él. El engranaje ‘b’ está acoplado al cuerpo del rotor que gira con él.

La relación de transmisión para este sistema queda definida de la siguiente manera:

$$T_{ab} = \frac{n_b}{n_a}$$

Los datos de este sistema de transmisión son:

n_a = 24 dientes

n_b = 120 dientes

1.6.1.3.2.- Transmisión de la articulación del hombro

Esta transmisión es similar a la transmisión de la articulación de la base. La diferencia es que esta transmisión es doble, lo que significa que el hombro se mueve simultáneamente desde los dos lados del brazo mecánico. Esto mejora el movimiento del hombro así como su capacidad de soportar carga.

El engranaje 'a' está acoplado al eje de salida del motor; el engranaje 'b' está acoplado al hombro del robot y lo mueve.

Los datos de este sistema de transmisión son:

n_a = 18 dientes

n_b = 72 dientes

1.6.1.3.3.- Transmisión de la articulación del codo

La transmisión de la articulación del codo se muestra en la Figura 1.20.

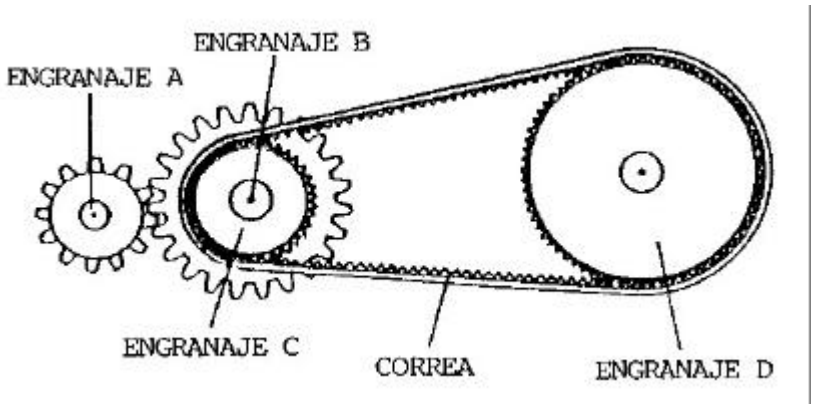

Figura 1.20. Transmisión indirecta de dos etapas.

Se trata de un sistema de transmisión indirecta de dos etapas. La transmisión del codo, como la del hombro, es una transmisión doble. El codo se

mueve simultáneamente desde ambos lados del brazo mecánico. Es decir, incluye dos sistemas motores del tipo que se ilustra en la Figura 1.20.

La combinación de dos transmisiones dobles (hombro y codo) previene que el brazo se retuerza e incrementa su estabilidad.

El engranaje 'a' está acoplado al eje de salida del motor, y mueve al engranaje 'b'. El engranaje 'c' está acoplado al engranaje 'b' y gira junto con él. El engranaje 'c' ataca al engranaje 'd' por medio de una correa.

La relación de transmisión para este sistema queda definida de la siguiente manera:

$$T_{ad} = T_{ab} \cdot T_{bc} \cdot T_{cd} = \frac{n_b}{n_a}\frac{n_d}{n_c} \quad \text{con} \quad T_{bc} = 1$$

Los datos de este sistema de transmisión son:

n_a = 18 dientes

n_b = 72 dientes

n_c = 17 dientes

n_d = 17 dientes

1.6.1.3.4.- Transmisión de la articulación de la muñeca

La articulación de la muñeca tiene dos grados de libertad, lo que le permite dos tipos de movimiento: movimiento de la muñeca arriba/abajo (elevación) y movimiento de la muñeca rotativo (giro).

La estructura de la articulación de la muñeca es la que muestra la Figura 1.21.

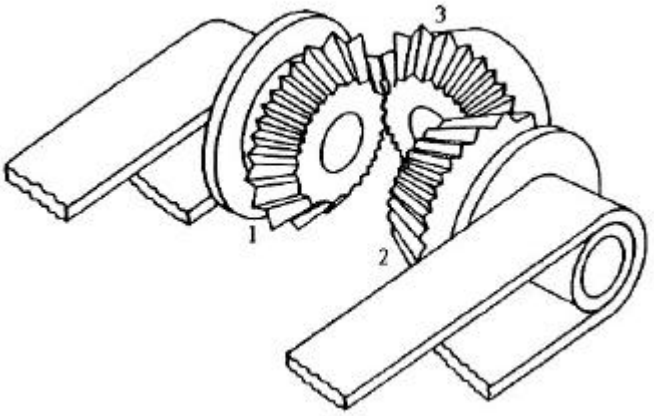

Figura 1.21. Engranaje diferencial.

Como se puede observar, la articulación de la muñeca tiene tres partes diferentes, que forman un conjunto conocido como **diferencial**. Las partes 1 y 2 están movidas por los motores 4 y 5 respectivamente. El movimiento relativo de las piezas 1 y 2 determina el movimiento de la pieza 3, a la cual está sujeta la pinza de la siguiente manera:

- Cuando las piezas 1 y 2 se mueven en la misma dirección, la pieza 3 se mueve hacia arriba o hacia abajo. Este movimiento se conoce generalmente como movimiento de elevación.
- Cuando las piezas 1 y 2 se mueven en sentidos opuestos, la pieza 3 tiene un movimiento giratorio. Este movimiento se conoce generalmente como movimiento de giro.

Tanto en el movimiento de elevación como en el de giro de la muñeca, los motores 4 y 5 giran la misma cantidad. (Sólo pueden diferir en su sentido).

Las transmisiones que van de los motores 4 y 5 a la articulación de la muñeca son idénticas, y corresponden a la transmisión que se ilustra en la Figura 1.22.

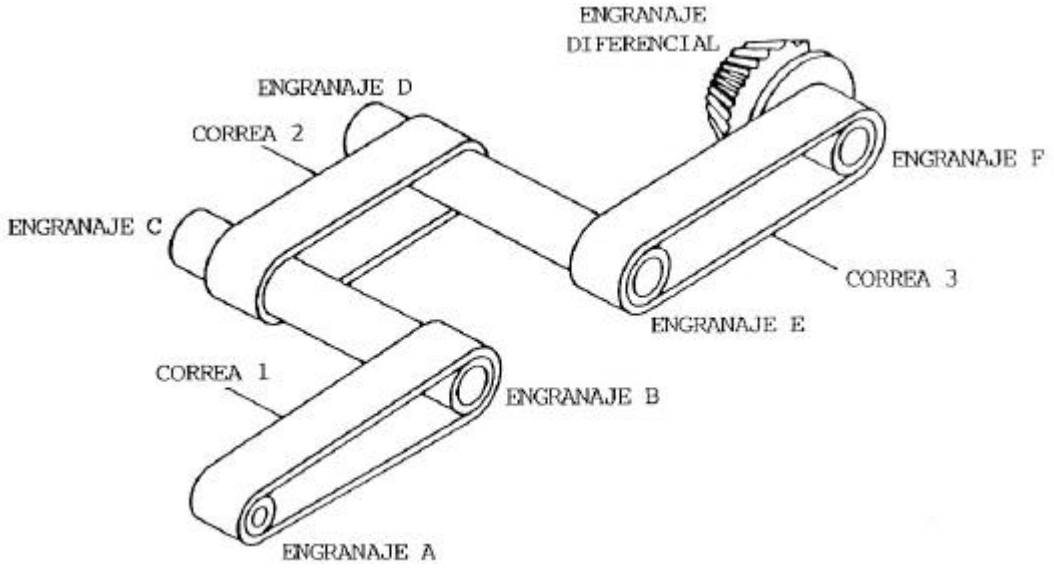

Figura 1.22. Transmisión indirecta de tres etapas.

La transmisión que va desde el motor 4 (o motor 5) al diferencial de la articulación de la muñeca es como sigue:

- El engranaje ‘a’ está acoplado al eje de salida del motor.
- El engranaje ‘b’ está movido por el engranaje ‘a’ a través de la correa 1.
- El engranaje ‘c’ realmente es la continuación del engranaje ‘b’ (son una sola pieza).
- El engranaje ‘d’ es movido por el engranaje ‘c’ (igual que en el engranaje ‘b’) a través de la correa 2.
- El engranaje ‘e’ es la continuación del engranaje ‘d’ (una sola pieza).
- El engranaje ‘f’ es movido por el engranaje ‘e’ (igual que en el engranaje ‘d’) a través de la correa 3.
- El engranaje diferencial está acoplado al engranaje f y gira junto con él.

La relación de transmisión para este sistema queda definida de la siguiente manera:

$$T_{af} = T_{ab} \cdot T_{cd} \cdot T_{ef} = \frac{n_b}{n_a} \frac{n_d}{n_c} \frac{n_f}{n_e}$$

Los datos de este sistema de transmisión son:

$$n_a = 12 \text{ dientes}$$

$$n_b = n_c = 24 \text{ dientes}$$

$$n_d = n_e = 24 \text{ dientes}$$

$$n_f = 24 \text{ dientes}$$

Engranajes 1,2 y 3 del sistema diferencial: 32 dientes cada uno.

1.6.1.3.5.- Transmisión de la pinza

La pinza es movida por un motor más pequeño que los otros motores. Este motor está fijado permanentemente a la articulación de la muñeca. El giro del motor de la pinza hace girar un tornillo, que produce la apertura o cierre de la pinza. Este tipo de transmisión se conoce como transmisión de tornillo de avance y se ilustra en la Figura 1.23.

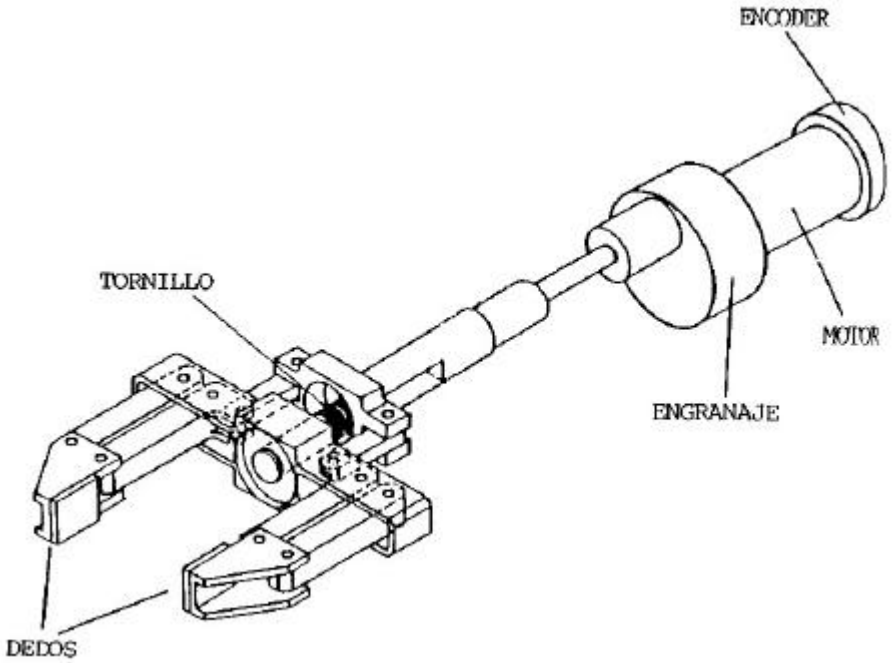

Figura 1.23. Articulación de la pinza.

El paso de tornillo se define como el número de milímetros entre dos hilos adyacentes del tornillo. El paso del tornillo en el sistema métrico es equivalente al movimiento lineal del tornillo. Ya que el tornillo de avance de la transmisión de la pinza está fijado al eje de salida del motor de la pinza, el paso del tornillo define el movimiento de la pinza (en mms.) producido por cada vuelta del eje salida.

A medida que se incrementa el paso del tornillo, lo hace la velocidad a la que la pinza se abrirá o cerrará. La resolución de los dedos de la pinza, sin embargo disminuye en proporción.

1.6.1.4.- Encoders ópticos

Los encoders del ESCORBOT ER-III son encoders incrementales y están usados como dispositivos de realimentación para el control de los motores.

Estos encoders están montados directamente sobre el eje de entrada de los motores, como puede verse en la Figura 1.18.

Cada encoder lleva dos pares de elementos emisor / receptor (LED y fototransistor), y un disco perforado (Figura 1.24).

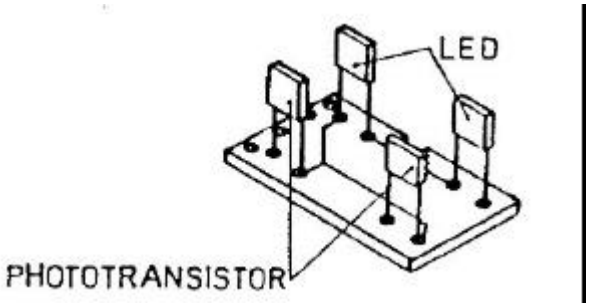

Figura 1.24. Pares emisor / fotoreceptor del encoder.

Hay dos tipos de encoders instalados en el ESCORBOT ER-III. De hecho, el tipo de encoder es el mismo, pero tienen montados discos con diferente número de perforaciones. En los motores 1,2,3,4 y 5 el disco tiene 6 perforaciones, mientras que el disco del encoder instalado en el motor número 8 (pinza) tiene solamente 3 perforaciones, como se muestra en la Figura 1.25.

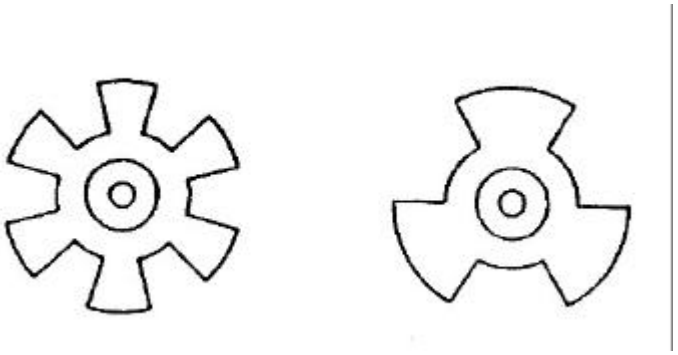

Figura 1.25. Discos utilizados en los encoders.

1.6.2.- Controlador

El controlador del ESCORBOT ER-III es la unidad encargada de recibir las órdenes del programa del computador y manejar los actuadores del robot acorde con éstas órdenes y la información recibida por los sensores (encoders y microswithces). En su interior dispone de una lógica electrónica capaz de transformar los comandos que le llegan procedentes del ordenador, en señales de potencia para mover cada una de las articulaciones.

Este controlador también es capaz de manejar hasta 8 entradas procedentes del exterior en forma de circuitos todo/nada (microswitches), y hasta 8 salidas del mismo tipo. Además, permite la conexión de hasta dos motores externos (motores 6 y 7) de elementos periféricos del robot, como pueden ser cintas transportadoras, etc.

Este controlador dispone de tres fuentes de alimentación en su interior para convertir la señal de la red en las tensiones que usará para alimentar a los motores y la electrónica de control:

- ±15 V, 4 A. No regulada, que suministra corriente a los motores.
- ±12 V, 0.4 A. Regulada, que suministra corriente a los circuitos de I/O y a las comunicaciones RS-232-C.
- +5 V, 1.5 A. Regulada, que suministra corriente al microprocesador, la memoria y la lógica del circuito de control.

En la parte trasera del controlador se hallan los conectores para la conexión de todo el sistema robot-controlador-ordenador, así como los fusibles de protección y la conexión a la red del controlador.

La comunicación entre el controlador y el robot, se realiza a través de un cable, con un conector del tipo D50, con 50 patillas distribuidas en 3 filas. Por este cable se conectan los motores, encoders y microswitches al controlador.

La comunicación entre el controlador y el ordenador, se realiza a través de un puerto serie RS-232-C de 25 pines estándar.

El controlador también admite la conexión de una botonera de enseñaza, que se realiza mediante otro puerto de comunicaciones RS-232-C de 25 pines.

El corazón de este controlador es el microprocesador de 8 bits Intel 8031AH

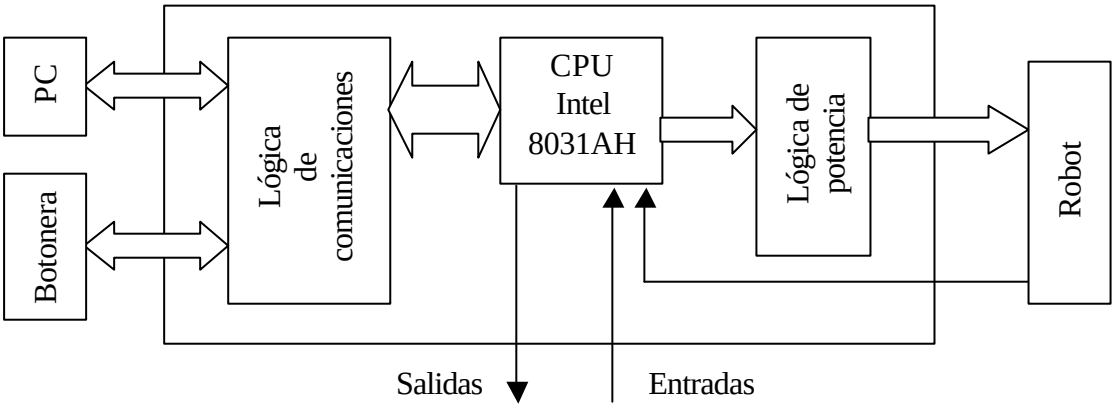

Figura 1.26. Esquema de conexión del sistema.

1.6.1.1.- Sistema operativo del controlador

El microprocesador de la unidad de control, tiene implementado un sistema operativo que le permite interactuar con el ordenador, recibiendo comandos desde éste y en caso necesario respondiendo a estos comandos, y en

algunas ocasiones iniciar él mismo la comunicación con el ordenador, como respuesta ha hechos específicos producidos en el robot.

Estos comandos se envían al controlador como una secuencia de códigos ASCII.

Los comandos que están implementados en el microprocesador son los siguientes:

Movimiento del motor

La secuencia de códigos ASCII a transmitir es la siguiente:

a M ± b c d e CR

donde,

a	Número del 1 al 8 que identifica al motor que se desea mover.
M	Designación del tipo de comando (Movimiento de Motor).
±	Sentido de giro del motor.
bcde	Cantidad de movimientos en impulsos del encoder. Esta cantidad no debe exceder de 8000. Cada uno de los caracteres se transmite independientemente.
CR	Retorno de carro.

El controlador almacena la cantidad de movimiento recibida en el registro de motor respectivo y acciona el motor. En el momento del accionamiento, el controlador lee la cantidad de movimiento mediante un encoder y actualiza continuamente el número en el registro del motor en funcionamiento, hasta llegar a 0.

Esta instrucción no recibe respuesta por parte del controlador.

Comprobar el resto de movimiento de motor

Los códigos ASCII que se envían al controlador tienen el formato siguiente:

a Q

donde,

a	número entre 1 y 8 que identifica al motor que se desea comprobar.
Q	define el tipo de comando. Para comprobar el resto de los impulsos necesarios para completar el movimiento.

En respuesta a este comando el controlador envía una respuesta binaria sobre el número de impulsos que el motor ha de realizar. Como el valor máximo puede ser ± 8000, la respuesta es de 14 bits, enviados en dos bytes de 8 bits cada uno.

El controlador divide el número de 14 bits en 2, suma a cada parte el bit '1' adicional como bit más significativo (MSB) (lo que no tiene consecuencia alguna) y devuelve al ordenador externo los dos bytes que comienzan por el byte más significativo y después el menos significativo. No hay ninguna señal adicional entre la transmisión de los dos bytes.

Activar las salidas

Los códigos ASCII que se envían al controlador tienen el formato siguiente:

aS

donde,

a	número entre el 1 y el 8 que identifica la salida.
S	define el tipo de comando. SET OUTPUT.

Este comando no recibe respuesta del controlador.

Desactivar las salidas

Los códigos ASCII que se envían al controlador tienen el formato siguiente:

aQ

donde,

a	número entre el 1 y el 8 que identifica la salida.
Q	define el tipo de comando. RESET OUTPUT.

Este comando no recibe respuesta del controlador.

Comprobar el estado de las entradas

Los códigos ASCII que se envían al controlador tienen el formato siguiente:

aI

donde,

a	número entre el 1 y el 8 que identifica la salida.
I	define el tipo de comando. INPUT.

El controlador responde con los siguientes códigos ASCII:

0	si la entrada está en OFF.
1	si la entrada está en ON.

Comprobar el final de carrera (microswitch)

Los códigos ASCII que se envían al controlador tienen el formato siguiente:

aL

donde,

a	número entre el 1 y el 8 que identifica la salida.
L	define el tipo de comando. LIMIT SWITCH.

El controlador responde con los siguientes códigos ASCII:

0	si el final de carrera está en OFF.
1	si el final de carrera está en ON.

Comprobar la terminación de una operación (fin de movimiento de todos los motores)

El código ASCII se envía al controlador en el formato siguiente:

T

El controlador responde con los siguientes códigos ASCII:

0	cuando el movimiento de por lo menos uno de los motores no esta completo.
1	cuando todos los motores han terminado su movimiento.

Comprobar si todos los motores están por debajo del desfase

El desfase por defecto es de 14 impulsos de encoder. EL código ASCII enviado al controlador tiene el formato siguiente:

E

El controlador responde con los siguientes códigos ASCII:

0	si hay uno o más motores que todavía tienen que completar más de 14 impulsos para terminar su movimiento.
1	si todos los motores tienen que completar menos de 14 impulsos para terminar su movimiento.

Este comando es útil cuando se necesite que el robot pase por varias posiciones continuamente sin pararse en cada punto.

Establecer un nuevo desfase

El código ASCII enviado al controlador tiene el formato siguiente:

Uabc*

donde,

U	designación del tipo de comando. CHARGE FOCET.
abc	el número del nuevo desfase.
*	denota el final del comando.

Este comando no genera respuesta por parte del controlador.

Parar un motor específico con pérdida de toda la información

El código ASCII enviado al controlador tiene el formato siguiente:

aP

donde,

a	número entre el 1 y el 8 que identifica al motor que se desea parar.
P	define el tipo de comando. Para poner a 0 el contador del motor.

Este comando no recibe respuesta del controlador.

Parada de todos los motores sin pérdida de información (Parada de Emergencia)

El código ASCII enviado al controlador tiene el formato siguiente:

B

El controlador parará todos los motores y almacenará en su memoria la cantidad de movimiento que todavía tienen que realizar éstos.

El controlador pone a cero los registros de todos los motores y traspasa la información que contienen a una sección diferente de la memoria llamada "memoria de soporte".

Este comando no recibe respuesta por parte del controlador.

Continuar el movimiento de los motores tras la Parada de Emergencia

El código ASCII enviado al controlador tiene el formato siguiente:

C

El controlador devuelve la información de la "memoria de soporte" a los registros de los motores y continua el movimiento de todos los motores que se pararon.

Este comando no recibe respuesta por parte del controlador.

Comprobar el resto de movimiento de los motores almacenado en la "memoria de soporte"

Los códigos ASCII se envían al controlador en el formato siguiente:

aZ

donde,

a	Número entre el 1 y 8 que identifica el motor que se desea comprobar.
Z	Define el tipo de comando.

La respuesta del controlador es idéntica a la del comando aQ. Se divide el número de 14 bits en 2, suma a cada parte el bit '1' adicional como bit más significativo (MSB) (lo que no tiene consecuencia alguna) y devuelve al ordenador externo los dos bytes que comienzan por el byte más significativo y después el menos significativo.

Comprobar el estado de todos los motores

El código ASCII enviado al controlador es el siguiente:

A

El controlador responde con los siguientes códigos ASCII:

0	Si no está en movimiento ninguno de los motores y no se pretende que se muevan.
-	Si todos los motores tienen una cantidad por debajo del desfase, para terminar su movimiento.
+	SI hay uno o más motores que tienen una cantidad por encima del desfase para terminar su movimiento.
1	Si se ha producido error del motor 1.

En el caso que el error se produjera en otro motor, el código ASCII recibido seria el correspondiente al número de motor (2,3,...8)

Establecer las velocidades de los motores

El código ASCII enviado al controlador tiene el formato siguiente:

aVb

donde,

a	Un número del 1 al 8 que identifica al motor cuya velocidad se desea ajustar. El número 0 denota ajuste de velocidad para todos los motores
V	Define el tipo de comando. VELOCIDAD
b	Un número entre 0 y 9 que identifica la velocidad que se desea fijar. La velocidad más lenta es 1, aumentando hasta 9 y 0 es la velocidad más rápida.

Este comando no recibe respuesta por parte del controlador.

Activar el modo interrupción

El código ASCII enviado al controlador tiene el formato siguiente:

W

Este comando permite al controlador efectuar interrupciones al ordenador cuando se producen hechos específicos. Los códigos ASCII que el controlador envía como interrupciones son los siguientes:

0	Cuando todos los motores pasan de un estado de movimiento a un estado de reposo
-	Cuando por lo menos un motor pasa de un estado en que la cantidad de movimiento es mayor que el desfase, a un estado en que la cantidad de movimiento es menor que el desfase
1	El cambio de estado de movimiento del motor1 a agarrotamiento del motor 1 (error del motor). En caso de tratarse de otro motor se envía el código ASCII correspondiente a ese número de motor (1 a 8).

L1	Cuando el microswitch 1 (1 a 8) pasa del estado OFF a ON
K1	Cuando el microswitch 1 (1 a 8) pasa del estado ON a OFF
I1	Cuando la entrada 1 (1 a 8) pasa del estado OFF a ON
J1	Cuando la entrada 1 (1 a 8) pasa del estado ON a OFF

Desactivar el modo interrupción

El código ASCII enviado al controlador tiene el siguiente formato:

X

Este comando asegura que el controlador no enviará interrupciones al ordenador principal (a través de la vía RS-232-C) cuando se produzcan los hechos específicos anteriormente mencionados.

1.6.3.- Comunicación controlador-ordenador

El controlador del robot contiene un microprocesador que funciona bajo el control de un ordenador externo. Para conectar las dos unidades se utiliza el estándar de comunicaciones asíncronas en serie norma RS-232-C.

Del gran número de funciones de la norma RS-232-C, sólo se emplean las siguientes:

- Transmisión de datos (TD)
- Recepción de datos (RD)
- Tierra (GND)

El controlador utiliza conectores RS-232-C de 25 patillas. Estas funciones se corresponden con las siguientes patillas del conector:

Tabla 1.10. Patillas de conexión para la transmisión.

Nº Patilla	Función
Patilla 2	Transmisión de datos
Patilla 3	Recepción de datos
Patilla 7	Tierra

Por ello, el cable que une el controlador y el ordenador tiene la configuración que se muestra en la Figura 1.27.

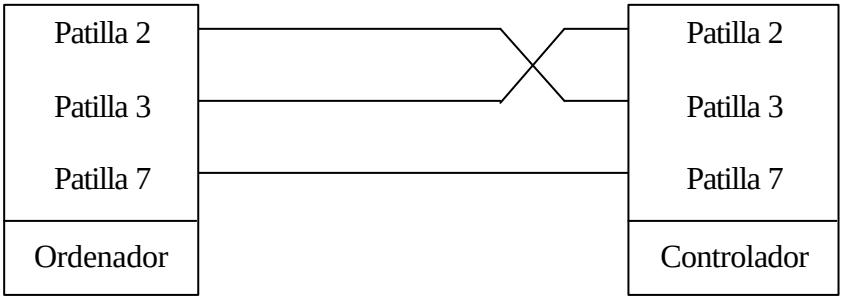

Figura 1.27. Conexión ordenador-controlador.

Las especificaciones técnicas para la comunicación entre el ordenador y el controlador son las que se muestran en la Tabla 1.11.

Tabla 1.11. Especificaciones de la comunicación serie

Parámetro	Valor
Régimen de baudios	9600
Bits de comienzo	1
Bits de datos	8
Bits de parada	2
Paridad	Sin paridad

Para la implementación de estas especificaciones, se debe reprogramar el gestor de comunicaciones interno del PC, conocido como UART (Universal Asynchronus Reciver / Transmiter).

1.6.3.1.- Gestor de comunicaciones del PC (UART 8250)

Este dispositivo se encarga de enviar los bits de datos uno por uno a través de los puertos de comunicaciones, teniendo en cuenta todos los tiempos necesarios para lograr una correcta comunicación y aliviar a la CPU de esta pesada tarea.

La línea que transmite los datos en serie esta inicialmente en estado alto. Al comenzar la transferencia, se envía un bit 0 o **bit de inicio**. Tras él irán los 8 **bits de datos** a transmitir (en ocasiones 7, 6 o 5): estos bits están espaciados con intervalo temporal fijo y preciso, ligado a la velocidad de transmisión que se esté empleando. Tras ello puede venir o no un **bit de paridad** generado automáticamente por la UART. Al final, aparecerá un bit (a veces un bit y medio o dos bits) a '1', que son los bits de parada o **bits de stop**. La presencia de

bits de inicio y de parada permite sincronizar la estación emisora y la receptora, haciendo que los relojes de ambas estén sincronizados.

La UART 8250 dispone de 11 registros (Tabla 1.12) pero sólo 3 líneas de dirección para seleccionarlos. Lo que permita distinguir unos de otros será, aparte de las líneas de direcciones, el sentido del acceso (lectura o escritura) y el valor de un bit de uno de los registros: el bit DLAB del registro LCR, que es el bit 7 de dicho registro. Realmente, DLAB se emplea sólo para acceder y programar los registros que almacenan el divisor de velocidad; el resto del tiempo , DLAB estará a '0' para poder acceder a los otros registro.

Tabla 1.12. Registros de la UART 8250.

A2	A1	A0	OFFSET	DLAB	MODO	NOMBRE	SIGNIFICADO
0	0	0	0	0	R	RBR	Receiver Buffer Register (Registro buffer de recepción)
0	0	0	0	1	R / W	DLL	Divisor Latch LSB (Divisor de velocidad, parte baja)
0	0	0	0	0	W	THR	Transmitter Holding Register (Registro de retención de transmisión)
0	0	1	1	0	R / W	IER	Interrupt Enable Register (Registro de habilitación de interrupciones)
0	0	1	1	1	R / W	DLM	Divisor Latch MSB (Divisor de velocidad, parte alta)
0	1	0	2	X	R	IIR	Interrupt Identification Register (Registro de identificación de Inter.)
0	1	1	3	X	R / W	LCR	Line Control Register (Registro de control de línea)
1	0	0	4	X	R / W	MCR	Modem Control Register (Registro de control del modem)
1	0	1	5	X	R / W	LSR	Line Status Register (Registro de estado de línea)
1	1	0	6	X	R / W	MSR	Modem Status Register (Registro de estado del modem)
1	1	1	7	X	R / W	SCR	Scratch Register (Registro residual)

Los ordenadores compatibles pueden tener conectados, de manera normal, hasta 4 puertos serie, nombrados COM1-COM4. El principal problema es que solo están previstas 2 interrupciones para los puertos serie. Ello implica que generalmente sólo 2 puertos podrán emplear interrupciones a un tiempo, debido a la arquitectura del bus ISA. Generalmente COM1 y COM 3 comparten la IRQ4, y COM2 y COM4 la IRQ3.

La Tabla 1.12 muestra los desplazamientos (offsets) que hay que sumar a la dirección E/S base del puerto serie para acceder a sus registros. COM1 suele estar en 3F8h, COM2 en 2F8h, COM3 en 3E8h y COM4 en 2E8h. Sin embargo, es mejor acceder a las variables de la BIOS para obtener la dirección.

La UART 8250 se programa a través de los registros de control LCR, IER, DLL, DLM y MCR. Aunque los registros de control pueden ser escritos en cualquier orden, IER debe ser escrito al final porque controla la habilitación de las interrupciones. Una vez que el 8250 ha sido programado, los registros pueden ser actualizados en cualquier momento en que el 82509 no se encuentre enviando o recibiendo datos.

A continuación, se describen los registros de la UART y el significado de sus bits.

1) LCR (Line Control Register). Controla el formato del carácter de datos.

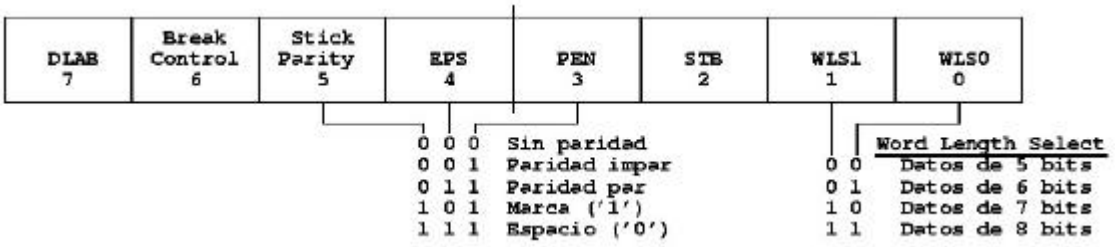

Figura 1.28. Descripción de los bits del LCR.

Los bits WLS seleccionan el tamaño de dato empleado. STB indica el número de bits de stop, que pueden ser 1 (STB=0) o 2 (STB=1), al trabajar con 5 bits STB=1 implica 1'5 bits de stop. PEN (Partity Enable) permite habilitar o no la generación de bit de paridad, EPS (Even Parity Select) selecciona paridad par si está a 1 (o impar en caso contrario). Stick Parity permite forzar el bit de paridad a un estado conocido según el valor de EPS.

El bit DLAB (Divisor Latch Access Bit) puesto a 1 permite acceder a los Latches divisores DLL y DLM del BRG en lectura y escritura. Para acceder al RBR, THR y al IER debe ser puesto a 0.

2) LSR (Line Status Register). Este suele ser el primer registro consultado tras una interrupción.

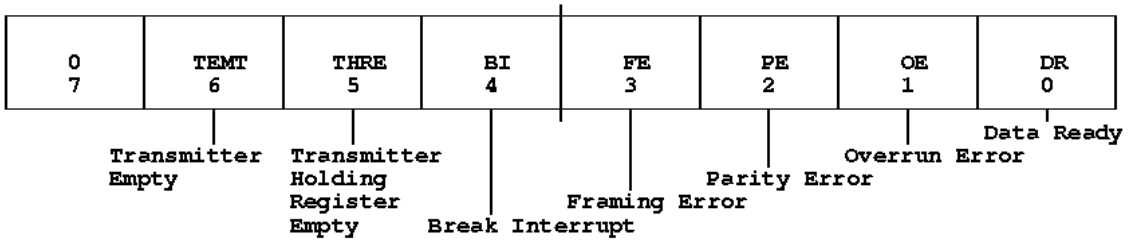

Figura 1.29. Descripción de los bits del LSR.

DR está activo cuando hay un carácter listo en el RBR y es puesto a 0 cuando se lee el RBR. Los bits 1 al 4 de este registro (OE, PE, FE y BI) son puestos a 0 al consultarlos (cuando se lee el LSR) y al activarse pueden generar una interrupción de prioridad 1 si ésta interrupción está habilitada. OE se activa para indicar que el dato en el RBR no ha sido leído por la CPU y acaba de llegar otro que lo ha sobreescrito. PE indica si hay un error de paridad. FE indica si el carácter recibido no tiene los bit de stop correctos. BI se activa cuando la entrada de datos es mantenida en espacio (a 0) durante un tiempo superior al de transmisión de un carácter (bit de inicio + bits de datos + bit de paridad + bit de parada).

THRE indica que el 8250 puede aceptar un nuevo carácter para la transmisión: este bit se activa cuando el THR queda libre y se desactiva escribiendo un nuevo carácter en el THR. Se puede producir, si está habilitada; la interrupción THRE (prioridad 3); INTRPT se borra leyendo el IIR. El 8250 emplea un registro interno para ir desplazando los bit y mandarles en serie (el Transmitter Shift Register), dicho registro se carga desde el THR. Cuando ambos registros (THR y el Transmitter Shift) están vacíos, TEMT se activa; volverá a desactivarse cuando se deje otro dato en el THR hasta que el último bit salga por SOUT.

3) MCR (Modem Control Register). Controla el interface con el modem.

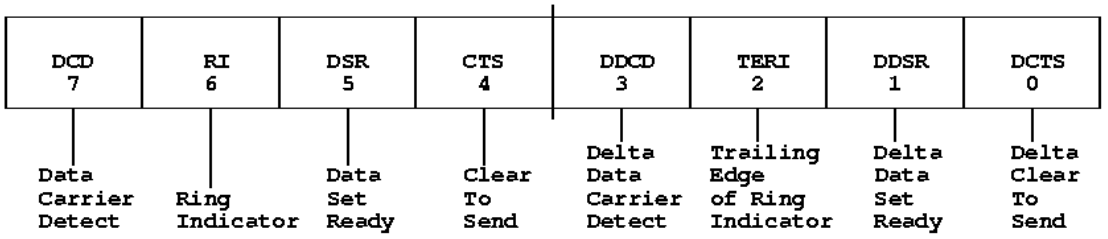

Figura 1.30. Descripción de los bits del MCR.

Las líneas de salida -DTR, -RTS, -OUT1 y -OUT2 están directamente controladas por estos bits; como se activan a nivel bajo, son puestas a 0 escribiendo un 1 en estos bits y viceversa. Estas líneas sirven para establecer diversos protocolos de comunicaciones.

El bit LOOP introduce el 8250 en un modo lazo (o bucle) de autodiagnóstico. Con LOOP activo, SOUT pasa a estado de marca (a 1) y la entrada SIN es desconectada. Los registros de desplazamiento empleados en la transmisión y la recepción son conectados entre sí. Las cuatro entradas de control del módem (-CTS, -DSR, DC y -RI) son desconectadas y en su lugar son internamente conectadas las cuatro salidas de control del módem (-DTR, -RTS, -OUT1 y -OUT2) cuyos pines son puestos en estado inactivo (alto). En esta

modalidad de operación (modo lazo o bucle), los datos transmitidos son inmediatamente recibidos, lo que permite comprobar el correcto funcionamiento del integrado. Las interrupciones son completamente operativas en este modo, pero la fuente de estas interrupciones son ahora los 4 bits bajos del MCR en lugar de las cuatro entradas de control. Estas interrupciones están aún controladas por el IER.

4) MSR (Modem Status Register).

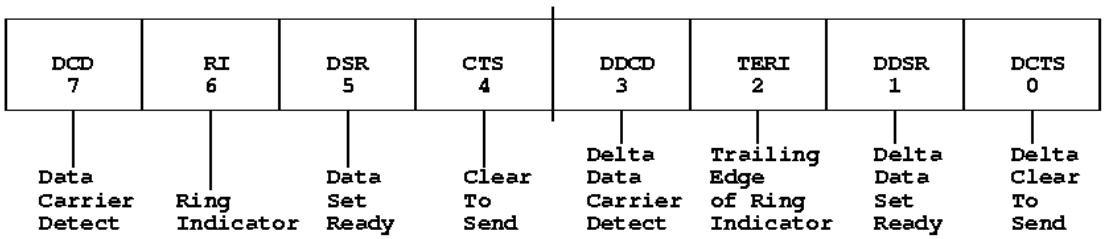

Figura 1.31. Descripción de los bits del MSR.

Además de la información de estado del modem, los 4 bits bajos (DDCD, TERI, DDSR, DCTS) indican si la línea correspondiente, en los 4 bits superiores, ha cambiado de estado desde la última lectura del MSR; en el caso de TERI sólo indica transiciones bajo-<alto en -RI (y no las de sentido contrario). La línea CTS del módem indica si está listo para recibir datos del 8250 a través de SOUT (en el modo lazo este bit equivale al bit RTS del MCR). La línea DSR del modem indica que está listo para dar datos al 8250 (en el modo lazo -o LOOP- equivale al bit DTR del MCR). RI y DCD indican el estado de ambas líneas (en el modo lazo se corresponden con OUT1 y OUT2 respectivamente). Al leer el MSR, **se borran** los 4 bits inferiores (que en una lectura posterior estarían a 0) pero no los bits de estado (los 4 más significativos).

Los bits de estado (DCD, RI, DSR y CTS) reflejan siempre la situación de los pines físicos respectivos (estado del módem). Si DDCD, TERI, DDSR ó DCTS están a 1 y se produce un cambio de estado durante la lectura, dicho cambio no

será reflejado en el MSR; pero si están a 0 el cambio será reflejado después de la lectura. Tanto en el LSR como en el MSR, la asignación de bits de estado está inhibida durante la lectura del registro: si se produce un cambio de estado durante la lectura, el bit correspondiente será activado después de la misma; pero si el bit ya estaba activado y la misma condición se produce, el bit será borrado tras la lectura en lugar de volver a ser activado.

5) y 6) BRSR (Baud Rate Select Register). Son los registros DLL (parte baja) y DLM (parte alta).

Estos dos registros de 8 bits constituyen un valor de 16 bits que será el divisor que se aplicará a la frecuencia base para seleccionar la velocidad a emplear. Dicha frecuencia base (por ejemplo, 1.8432 MHz) será dividida por 16 veces el valor almacenado aquí. Por ejemplo, para obtener 2400 baudios:

$$\frac{1843200}{16 \cdot 2400} = 48 \Rightarrow \text{DLL} = 48,\ \text{DLM} = 0$$

7) RBR (Receiver Buffer Register).

El circuito receptor del 8250 es programable para 5, 6, 7 u 8 bits de datos. En el caso de emplear menos de 8, los bits superiores de este registro quedan a 0. Los datos entran en serie por SIN (comenzando por el bit D0) en un registro de desplazamiento gobernado por el reloj de RCLK, sincronizado con el bit de inicio. Cuando un carácter completa el registro de desplazamiento de recepción, sus bits son volcados al RBR y el bit DR del LSR es activado para indicar a la CPU que puede leer el RBR. El diseño del 8250 permite la recepción continua de datos sin pérdidas: el RBR almacena siempre el último carácter recibido dando tiempo suficiente a la CPU para leerlo mientras simultáneamente está cargando el registro de desplazamiento con el siguiente; si la CPU tarda demasiado un

nuevo dato podría aparecer en el RBR antes de haber leído el anterior (condición de overrun, bit OE del LSR).

8) THR (Transmitter Holding Register).

El registro de retención de transmisión almacena el siguiente carácter que va a ser transmitido en serie mientras el registro de desplazamiento de transmisión está enviando el carácter actual. Cuando el registro de desplazamiento se vacíe, será cargado desde el THR para transmitir el nuevo carácter. Al quedar vacío THR, el bit THRE del LSR se activa. Cuando estén vacíos tanto el THR como el registro de desplazamiento de transmisión, el bit TEMT del LSR se activa.

9)SCR (Scratchpad Register).

Este registro no es empleado por el 8250, y de hecho no existía en las primeras versiones del integrado. Puede ser empleado por el programador como una celdilla de memoria.

10) IIR (Interrupt Identification Register).

Existen 4 niveles de prioridad en las interrupciones generables por el 8250, por este orden:

1) Estado de la línea de recepción.
2) Dato recibido disponible.
3) Registro de retención de transmisión vacío.
4) Estado del módem.

La información que indica que hay una interrupción pendiente y el tipo de la misma es almacenada en el IIR. El IIR indica la interrupción de mayor prioridad pendiente. No serán reconocidas otras interrupciones hasta que la CPU envíe la señal de reconocimiento apropiada. En el registro IIR, el bit 0 indica si hay una interrupción pendiente (bit 0=0) o si no la hay (bit 0=1), esto permite tratar las interrupciones en modo *polled* consultando este bit. Los bits 1 y 2 indican el tipo de interrupción. Los restantes están a 0 en el 8250, pero el 16550 utiliza alguno más.

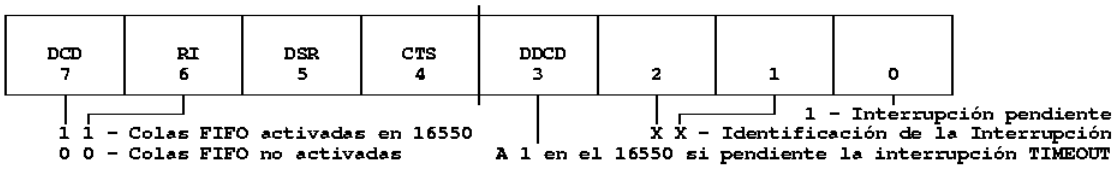

Figura 1.32. Descripción de los bits del IIR.

Tabla 1.13. Identificación y reconocimiento de las interrupciones.

Identificación de la interrupción				**Acitvación / Reconocimiento (Reset) de la interrupción**		
Bit 2	**Bit 1**	**Bit 0**	**Prioridad**	**Flag**	**Fuente**	**Reconocimiento**
X	X	1		Ninguno	Ninguna	
1	1	0	Primera	Línea de estado del receptor	OE, PE, FE o BI	Leer LSR
1	0	0	Segunda	Recibido dato disponible	Recibido dato disponible	Leer RBR
0	1	0	Tercera	THRE	THRE	Leer IIR si es la fuente de interrupción, o escribir THR
0	0	0	Cuarta	Estado del módem	-CTS, -DSR, -RI, -DCD	Leer MSR

11) IER (Interrupt Enable Register).

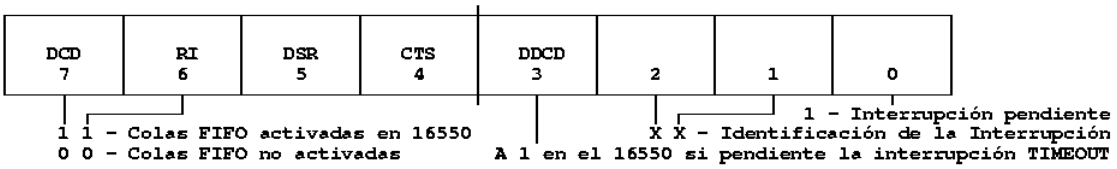

Figura 1.33. Descripción de los bits del IER.

Este registro de escritura se utiliza para seleccionar qué interrupciones activan INTRPT y, por consiguiente, van a ser solicitadas a la CPU. Deshabilitar el sistema de interrupciones inhibe el IIR y desactiva la salida INTRPT.

1.6.4.- Joystick standard

El joystick standard consiste en una palanca de mando y dos botones. La palanca de mando está mecánicamente unida a dos potenciómetros cuyos valores de resistencia varían entre 0 Ω y 100 kΩ (uno para el eje X y otro para el eje Y) con el movimiento de la palanca. Las resistencias de los potenciómetros toman el valor mínimo cuando el joystick está situado en la posición Arriba-Izquierda, y el valor máximo cuando se sitúa la palanca en la posición Abajo-Derecha.

Uno de los terminales finales del potenciómetro se conecta a la alimentación de +5V y el terminal central (resistencia variable) se conecta a la entrada analógica del joystick. El tercer terminal del potenciómetro no se conecta.

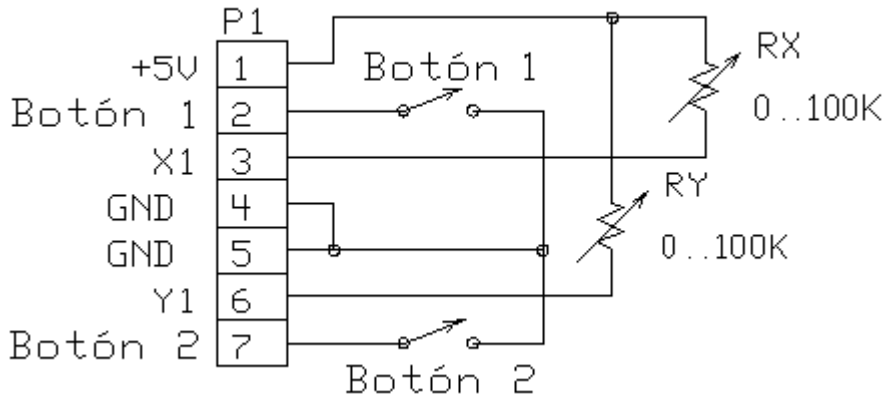

Figura 1.34. Diagrama de conexiones de un joystick standard.

1.6.4.1.- Conector del joystick

La Figura 1.35 muestra el conector macho DB15 situado en el cable de conexión del joystick.

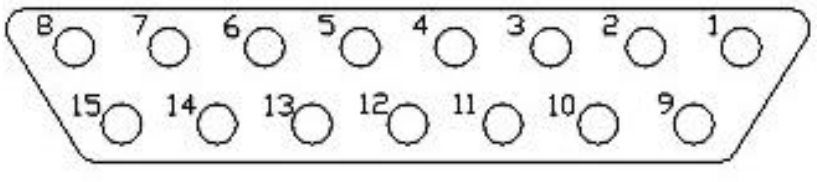

Figura 1.35. Conector macho del joystick. (Vista frontal).

Las funciones de cada pin del conector son las siguientes:

1. Alimentación para los ejes XY1 (+5V)
2. Botón 1
3. Coordenada del eje X1
4. Tierra (para el Botón 1)
5. Tierra (para el Botón 2)
6. Coordenada del eje Y1
7. Botón 2
8. No conectado
9. Alimentación para los ejes XY2 (+5V)
10. Botón 3
11. Coordenada del eje X2
12. Tierra (para el Botón 3 y 4)
13. Coordenada del eje Y2
14. Botón 4
15. No conectado

Los puertos de joystick implementados en los ordenadores, permiten la conexión de dos joysticks simultáneamente al mismo puerto, aunque es necesario el empleo de un conector de tipo Y.

1.6.4.2.- Puerto de joystick del PC

En los PC's actuales, el joystick se conecta a través de la tarjeta de sonido, ya que ésta utiliza el mismo conector para comunicarse con dispositivos MIDI.

El puerto de joystick es un puerto de 8 bits de entrada y salida mapeado a memoria en la dirección 201h. La CPU puede leer y escribir el puerto de joystick mediante la dirección 201h. La escritura en esta dirección inicializa el proceso de medida de la posición del joystick. La interficie del joystick solamente detecta que algo se ha escrito en el puerto de E/S para resetear el multivibrador que esta en la placa. El valor devuelto no queda registrado en ninguna parte, sino que se debe consultar de nuevo la dirección 201h, para comprobar la información de estado de la interficie del joystick.

De los 4 bits de más peso se obtiene información del estado de los botones. Los cuatro bits menos significativos revelan el estado de los multivibradores usados para la medida de la resistencia de los potenciómetros, y por tanto, de la posición del joystick. En la Tabla 1.14 se detalla la descripción del significado de estos bits.

Tabla 1.14. Significado de los bits del Puerto de Joystick.

7	6	5	4	3	2	1	0	Estado de los bits
X								Botón B2, 0 = presionado, 1 = no presionado (defecto)
	X							Botón B1, 0 = presionado, 1 = no presionado (defecto)
		X						Botón A2, 0 = presionado, 1 = no presionado (defecto)
			X					Botón A1, 0 = presionado, 1 = no presionado (defecto)
				X				Monoestable BY, 1 = contando, 0 = conteo finalizado
					X			Monoestable BX, 1 = contando, 0 = conteo finalizado
						X		Monoestable AY, 1 = contando, 0 = conteo finalizado
							X	Monoestable AX, 1 = contando, 0 = conteo finalizado

1.6.4.3.- Medida de la posición del joystick

Posición de la palanca

Las entradas de posición del joystick son simplemente entradas donde los potenciómetros de 100 kΩ del joystick están directamente conectados. Los potenciómetros están conectados entre el pin de alimentación de 5 V y el pin de entrada de los potenciómetros al controlador de joystick, tal como se muestra en la Figura 1.36.

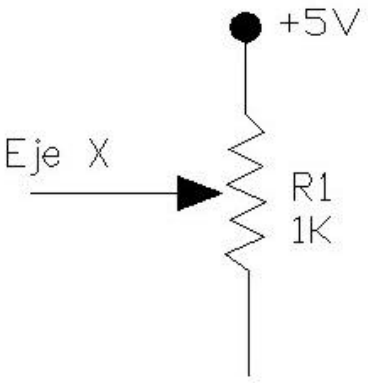

Figura 1.36. Conexión del potenciómetro del Eje X.

El valor de la resistencia de los potenciómetros se mide utilizando un circuito multivibrador monoestable, donde un pequeño condensador se carga a través del potenciómetro del joystick hasta un cierto nivel de tensión. La interficie del joystick posee cuatro de estos tipos de multivibradores monoestables. El circuito integrado usado es el 558 (4 circuitos integrados 555 en un solo chip).

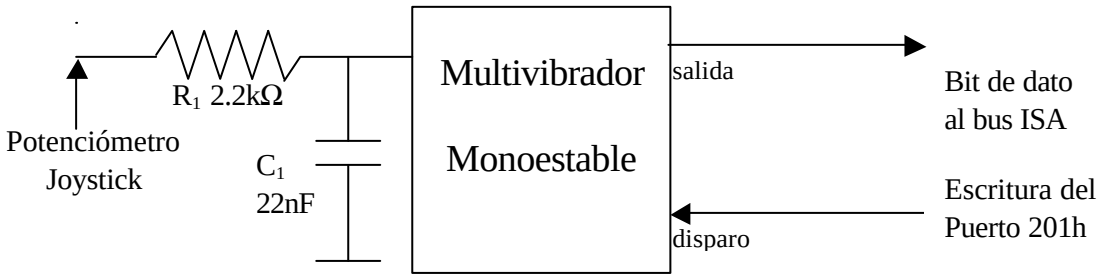

Figura 1.37. Circuito de medida con el multivibrador 558.

El multivibrador realiza la medida de la siguiente manera:

1. Normalmente, en condiciones iniciales el condensador C_1 estará completamente cargado (5V) y el multivibrador pondrá un '1' lógico a su salida.
2. La CPU realiza la escritura de un byte (no importa su valor) en la dirección 201h del espacio de E/S, para resetear el multivibrador. Entonces el multivibrador descarga el condensador C_1. Seguidamente se pone un '0' lógico a la salida del multivibrador.
3. El condensador empieza a cargarse con la corriente que circula a través de R1 y del potenciómetro del joystick.
4. Cuando la tensión del condensador alcanza cierto valor, el multivibrador pone un '1' lógico a su salida.

Cuanto mayor sea el valor de la resistencia del potenciómetro, mayor tiempo tardará el condensador en cargarse. El tiempo transcurrido entre el reset del multivibrador y el estado en que aparece el '1' lógico de nuevo, es medido por software. Con este tiempo se puede averiguar el valor de la resistencia del potenciómetro y, por lo tanto, de la posición del joystick.

Estado de los botones

Los botones del joystick son simples entradas todo/nada. El botón del joystick deja el pin sin conexión o lo conecta a tierra según si éste es oprimido o no.

El estado de los botones es directamente enviado a la interficie del joystick a través del bus ISA cuando se lee el puerto de E/S.

El cableado de los circuitos de control de los botones es muy sencillo, tal como muestra la Figura 1.38. Cada entrada tiene una resistencia R_1 = 1 kΩ. Esta resistencia esta conectada por un extremo a la tensión de alimentación (5V) y por el otro terminal no está conectada. Entonces la tensión en este segundo terminal es 5V, y se tiene un '1' lógico. Cuando el botón es presionado, se conecta eléctricamente la resistencia a tierra y la tensión en este punto entonces es 0, obteniéndose un '0' lógico.

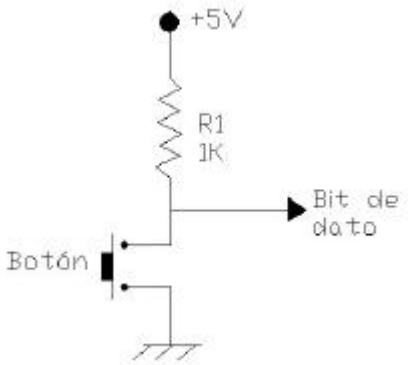

Figura 1.38. Circuito de los botones.

Interficie del joystick

La Figura 1.39 muestra la interficie de control standard para el puerto de joystick que se encuentra en los PC's.

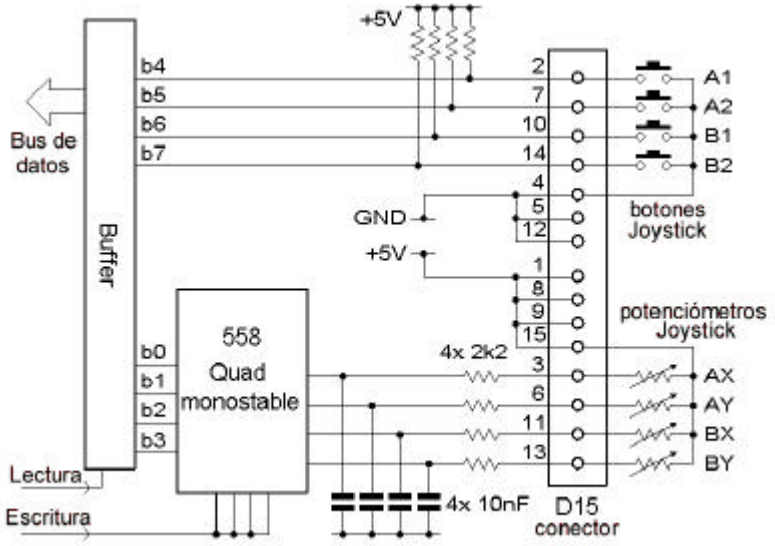

Figura 1.39. Interficie de control de puerto de joystick.

1.7.- Funcionamiento de la aplicación

La aplicación diseñada se divide en diversos menús de pantalla para hacer más fácil el uso de la misma. Al inicio del programa se parte de un menú principal desde el cual se puede acceder a las diferentes opciones del programa (configuraciones, home, enseñanza de posiciones, edición de programas, ejecución, etc.). El sistema de menús es asociativo, en forma de árbol, de tal manera que, una vez en un submenú, para volver hacia atrás hasta el menú principal, se ha de pasar por los menús por los que previamente se ha pasado hasta llegar al mencionado submenú.

Al iniciar la aplicación, y antes de acceder al menú principal, se pide que se defina el puerto de comunicaciones para establecer la comunicación con el robot, y seguidamente, se carga el fichero de configuraciones. En este fichero se encuentra la información relativa a la configuración de la pantalla que se muestra durante la ejecución de la aplicación. Estos parámetros pueden ser modificados por el usuario en cualquier momento durante la ejecución de la aplicación y al salir de ésta, quedan guardados en el fichero de configuración como los nuevos parámetros por defecto, que se cargarán la próxima vez que se ejecute la aplicación.

1.7.1.- Menú *Principal*

Este es el menú que se muestra al iniciar la aplicación. Desde él se puede acceder a las diferentes partes del programa, tal como se muestra en el diagrama siguiente:

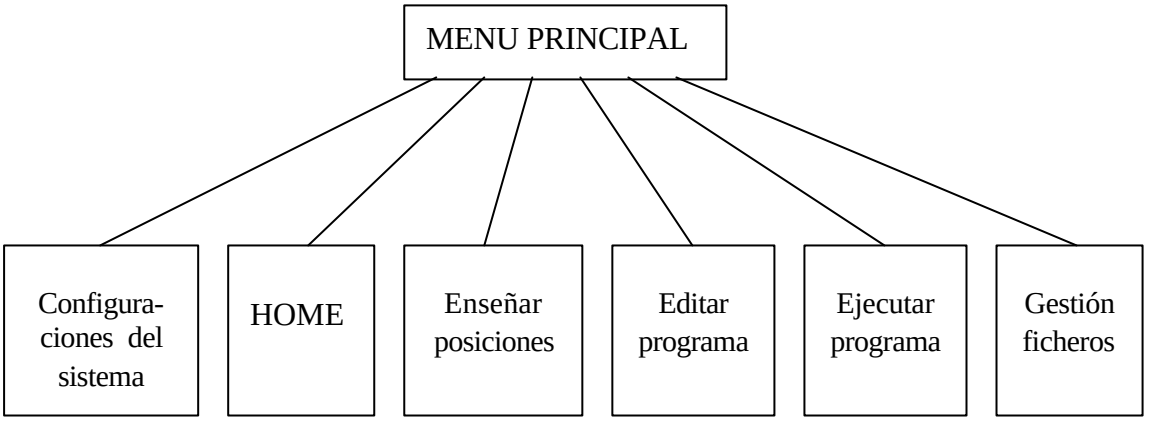

Figura 1.40. Diagrama del menú principal.

Además de éstas opciones, este menú nos permite abandonar la aplicación y salir al DOS. Cuando se elige esta opción aparece en pantalla una ventana que avisa que se perderán todos los datos no guardados en caso de seguir adelante y abandonar la aplicación.

1.7.2.- Menú *Configuraciones del sistema*

Desde este menú se pueden cambiar diferentes parámetros del sistema. Las opciones que permite este menú son las siguientes:

Puerto de comunicaciones

Esta opción permite cambiar el puerto de comunicaciones que se utiliza para comunicarse con el robot. Permite elegir entre COM1 y COM2.

Velocidad de los motores

Esta opción permite establecer la velocidad de cada uno de los motores que componen el brazo articulado.

Existen 10 velocidades diferentes que van des de la 1, la más lenta, hasta la 0, la más rápida (1,2,...,9,0). También pueden configurarse todos los motores a la misma velocidad escogiendo como número de motor el 0.

Pulsos del movimiento normal

Esta opción permite cambiar el número de pulsos de encoder que se envían al motor cada vez que éste tiene que moverse, es decir, cada vez que se pulsa una tecla o se mueve la palanca del joystick. Esto sólo afecta al movimiento cuando se esta operando por articulaciones individuales.

Cuanto mayor sea el número de pulsos, mayor será el movimiento realizado, y menor la precisión. Se permite elegir entre un valor mínimo de 10 y un valor máximo de 50.

Resolución en movimiento cartesiano

Esta opción afecta al movimiento del TCP sobre los ejes coordenados, y permite elegir el incremento de la posición en milímetros cada vez que se pulsa una tecla de movimiento o se acciona la palanca del joystick.

Cuanto mayor sea este incremento, menos 'recta' será la trayectoria entre dos puntos consecutivos. Se permite elegir entre un mínimo de 10 y un máximo de 50 milímetros. La resolución mínima del Pitch y el Roll será de 1º.

Calibrar joystick

Con esta opción se permite calibrar el joystick. Se pide que se coloque la palanca del joystick en una serie de posiciones y que se accionen los botones para poder determinar el rango de movimiento.

Una vez realizado esto, aparece una pantalla con los valores medidos y los valores actuales del joystick así como el estado de sus botones, donde se puede comprobar el correcto funcionamiento del joystick moviendo el cursor por la pantalla.

Configuraciones del usuario

A través de esta opción se accede a un submenú donde se permite elegir diferentes configuraciones de la pantalla. Concretamente se permite mostrar o no mostrar durante la ejecución de la aplicación lo siguiente:

- Información acerca de los parámetros articulares del robot (θ_1, θ_2 θ_3 θ_4 θ_5).
- Posición y orientación actual del brazo del robot.
- Estado de las transmisiones del controlador-ordenador.

Estas configuraciones quedan guardadas en el fichero “robot.cfg” al salir de la aplicación y son cargadas como configuraciones por defecto en la siguiente ejecución.

1.7.3.- Menú *Home*

Desde este menú se permite definir la posición de HOME del robot. Las opciones que se encuentran en este apartado son:

Introducir posición HOME manualmente

Con esta opción se introducen por teclado la posición y orientación del robot. Previamente se debe haber situado el robot en la posición y orientación adecuada.

Los valores a introducir son las tres coordenadas de posición en el espacio (x, y, z) del extremo de la herramienta de trabajo, las tres componentes del vector de acercamiento y las tres componentes del vector de orientación.

Buscar HARD HOME automáticamente

Con esta opción, el robot comienza a accionar cada una de sus articulaciones de forma secuencial, empezando por la base, hombro, codo, etc. hasta accionar todas las articulaciones. El movimiento empieza en un sentido, y en caso de llegar al final del recorrido y no encontrar la posición de HARD HOME de la articulación, se invierte el sentido del movimiento y se reemprende la búsqueda.

Una vez hallada la posición de HARD HOME de todas las articulaciones, el proceso queda detenido y queda establecida automáticamente la posición y orientación del robot.

Operar el brazo manualmente

Esta opción permite operar cada una de las articulaciones del robot independientemente, sin que esto tenga efecto sobre el control de la posición y orientación actual del brazo.

Esto permite situar el brazo en la posición deseada antes de introducir los valores de la posición HOME, o recalibrar la posición en el caso que, tras un largo período de ejecución de movimientos, el error de precisión fuera considerable.

1.7.4.- Menú *Enseñar posiciones*

Este es el menú más importante y de mayor uso en la aplicación, ya que desde aquí se puede operar el robot, memorizar posiciones, reproducir posiciones, trayectorias, etc.

Este menú consta de las siguientes opciones:

Operar por articulaciones

Esta opción permite mover el brazo articulado accionando cada una de sus articulaciones independientemente. Los valores de posición y orientación son actualizados a cada movimiento del brazo.

Operar mediante coordenadas cartesianas

Esta opción permite mover el brazo desplazando el extremo de la herramienta de trabajo en las tres direcciones del espacio (x, y, z). Los movimientos de la herramienta serán, por lo tanto, líneas rectas en las direcciones *x*, *y*, y *z*. La orientación de la herramienta se efectúa manteniendo el TCP (Punto del Centro de la Herramienta), que en este caso esta situado en el extremo de la pinza, inmóvil. Es decir, cuando se reorienta la mano, se mueve todo el brazo de tal forma que el TCP mantiene su posición en el espacio.

Los valores de posición y orientación son actualizados a cada movimiento del brazo.

Ir a un punto del espacio

Mediante esta opción, se introducen los valores de posición del TCP (coordenadas *x*, *y*, y *z*) y la orientación de la mano, y el robot mueve sus articulaciones hasta alcanzar dicha posición.

En este caso los valores de orientación son introducidos en forma de ángulos como valores de Pitch y Roll (expresados en grados) en lugar de introducir los vectores de acercamiento y orientación. Los valores de Pitch y Roll se darán en relación a la posición HOME, donde Pitch = 0º y Roll = 0º.

Enseñar trayectoria

Esta opción permite memorizar una trayectoria determinada. A la hora de enseñar la trayectoria se puede operar por articulaciones o mediante coordenadas articulares.

A medida que el brazo se va moviendo, las diferentes posiciones y orientaciones se van guardando automáticamente en la memoria. Para cada movimiento del robot se guarda una posición.

Memorizar posición

Mediante esta opción, se memoriza la posición y orientación actual, añadiéndola a la lista de posiciones en memoria.

Listado de posiciones

Este comando muestra una lista con todas las posiciones que están en la memoria. Mediante los cursores se puede recorrer la lista arriba y abajo, y con la tecla INTRO, se lleva el robot a la posición seleccionada de la lista.

Reproducir posiciones

Mediante este comando se lleva al robot a cada una de las posiciones almacenadas en la memoria por el mismo orden en que éstas han sido guardadas en la lista, empezando por la posición de HOME, que siempre es la primera posición de la lista.

En este tipo de ejecución no se tienen en cuenta pausas entre posiciones ni otras opciones de programación, simplemente es una herramienta para visualizar las posiciones memorizadas hasta el momento.

1.7.5.- Menú *Editar programa*

Desde este menú se pueden crear diferentes programas para la ejecución de tareas con las posiciones que están en memoria.

En la parte izquierda aparece una ventana con las líneas de programa que se editan. En la parte derecha superior hay una ventana con los comandos que se pueden usar. Estos comandos son:

- **Ir a posición**. Se especifica la posición que se desea alcanzar.
- **Velocidad**. Se especifica el motor cuya velocidad se desea modificar y se establece la velocidad de éste.
- **Pausa**. Introduce una pausa en la ejecución. El valor se expresa en milisegundos.
- **Abrir pinza**.
- **Cerrar pinza**.
- **Borrar línea**. Elimina una línea del programa.
- **Mover línea**. Permite desplazar una línea de código de una posición a otra del programa. Útil cuando se desea insertar un nuevo comando en el programa ya existente.
- **Trayectoria**. Introduce todas las posiciones que están en la memoria, en el mismo orden en que han sido guardadas. Este comando esta pensado para reproducir una trayectoria previamente creada.

En la parte derecha inferior está la ventana de comandos, donde aparece el comando seleccionado y donde deben introducirse los datos del programa (nº posición, línea a eliminar, etc.).

1.7.6.- Menú *Ejecutar programa*

Esta opción permite la ejecución de tres maneras distintas del programa que se ha creado. Los tipos de ejecución son:

Ejecutar una sola vez

El programa creado es ejecutado una sola vez en su totalidad, y cuando éste es finalizado se vuelve al menú de ejecución de programas.

Ejecutar de forma continua

La ejecución del programa se produce completamente de forma cíclica. Para detener la ejecución basta con pulsar cualquier tecla, y el programa se parará cuando termine el ciclo actual, es decir, cuando se ejecute la última instrucción de programa del ciclo que se está ejecutando.

Ejecutar paso a paso

Este es un tipo de ejecución muy útil para el análisis y depurado del programa. La ejecución del programa se realiza una sola vez, pero ejecutando una sola instrucción cada vez que se pulsa una tecla.

1.7.7.- Menú *Gestión de Ficheros*

Des de este menú se pueden guardar en el disco las posiciones y programas que están en memoria. Igualmente se pueden cargar en memoria las posiciones y programas guardados en el disco.

El nombre de los ficheros puede ser de un campo alfanumérico de hasta 8 caracteres, y no se pone extensión, ya que esta la añade automáticamente el programa.

En caso de que se trate de un fichero de posiciones se guardará como "*nombre_fichero*.pos", y en caso de que se trate de un fichero de programa se guardará como "*nombre_fichero*.pgm".

Los ficheros son guardados como ficheros de texto, de modo que se pueden leer con cualquier editor de texto, aunque no es aconsejable modificarlos, ya que pueden existir problemas a la hora de cargarlos de nuevo en memoria durante la ejecución de la aplicación.

1.7.8.- Otras consideraciones

Cada vez que se opera el robot desde cualquiera de sus opciones (articulares, cartesianas, en lazo abierto) aparece un submenú que permite elegir si se quiere usar el teclado o el joystick. Para cada una de estas dos opciones aparece una pantalla con las teclas que se pueden usar y el efecto que éstas producen o, en el caso del joystick, la secuencia de movimientos a realizar con la palanca y los botones que se han de pulsar para mover el robot.

La aplicación tiene una serie de protecciones para evitar el incorrecto uso del programa, y evitar posibles situaciones críticas. Por ello, al iniciar la aplicación, se muestra un mensaje que indica que no esta definido el HOME. Este mensaje permanece activo hasta que no se defina HOME. En esta situación, no se permite acceder al submenú *Enseñar posiciones*, ya que mover el robot sin estar definido el HOME provocaría problemas graves de posicionamiento. Del mismo modo, tampoco no se permite cargar ficheros de posición hasta que no se defina el HOME.

En esta misma línea, el submenú *Ejecutar programa*, tampoco podrá ser accedido hasta que no se haya creado un programa o se haya cargado un programa en memoria.

Mensajes de error

Los mensajes de error aparecen cuando se intenta realizar una operación no válida, ya sea por parte del usuario, o por parte del propio ordenador. Cuando se produce un error aparece en pantalla una ventana, acompañada de una alerta sonora, que indica el tipo de error que se ha producido.

Los errores contemplados por el programa son los siguientes:

- Error al abrir un archivo. No se puede cargar o guardar el archivo.
- Error de puerto de comunicaciones. Puerto inexistente para la BIOS.
- Error de acceso al menú *Enseñar posiciones*. HOME no definido.
- Error de acceso al menú *Ejecutar programa*. Programa no creado.
- Error al cargar fichero de posiciones. HOME no definido.
- Error de joystick. El joystick no está calibrado.

1.8.- Diseño y estructura del programa

El lenguaje de programación usado para implementar el programa ha sido el Lenguaje C, debido a que éste es un lenguaje de alto nivel que permite manejar instrucciones de bajo nivel con cierta facilidad, y es ideal para los programas que deben correr sobre MS-DOS.

El programa está estructurado en una serie de menús de pantalla asociativos en forma de árbol, de manera que, desde un menú se accede a sus submenús, y al volver hacia atrás se ha de pasar obligatoriamente por los menús previos.

Esto es debido a que para cada menú se ha creado una función, y desde ella, según el comando a ejecutar se van llamando las diferentes funciones de forma sucesiva, de manera que éstas se van anidando.

Para guardar las posiciones y programas en memoria, se han usado dos vectores de longitud finita. En el vector de posiciones se pueden almacenar hasta 300 posiciones con los valores de las componentes de posición y orientación (ésta última tanto en forma de vector como de ángulo). En el vector de programa se pueden almacenar hasta 500 instrucciones de programa.

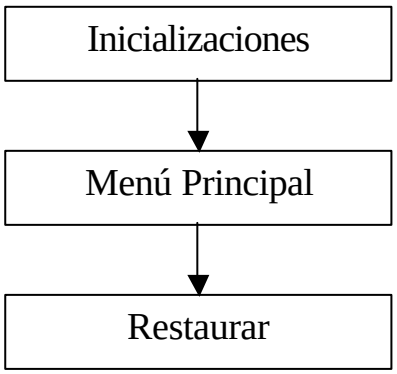

Figura 1.41. Diagrama del programa principal.

Debido a que las funciones se van anidando, el programa principal es muy simple, y su estructura es la mostrada en la Figura 1.41.

El centro del sistema es la función *Menú Principal* (ver página 208 del anexo), ya que se estará ejecutando continuamente hasta que se salga del programa. Desde esta función se llamará a las funciones de los submenús, y desde éstas, se ejecutarán las funciones de cálculo y posicionamiento del robot específicas para cada tipo de movimiento, según sea éste articular o por coordenadas cartesianas.

La función de *Inicializaciones* (ver página 190 del anexo), se encarga de guardar los antiguos vectores de interrupción, y de sustituirlos por los nuevos vectores, que llamarán a las rutinas de interrupción de reloj (IRQ0), y de comunicaciones por los puertos serie COM1 y COM2 (IRQ4 e IRQ3 respectivamente) propias.

Además de esto, esta función realiza una llamada a la rutina de *inicialización del puerto serie* (ver página 191 del anexo), que vacía los registros de comunicación y establece los parámetros de transmisión a los siguientes valores:

Tabla 1.15. Especificaciones de la comunicación serie

Parámetro	Valor
Régimen de baudios	9600
Bits de comienzo	1
Bits de datos	8
Bits de parada	2
Paridad	Sin paridad

La misma rutina, y después de la inicialización del puerto de comunicaciones, envía al controlador del robot el carácter ‘W’, que activa las

interrupciones al ordenador por parte del controlador, y seguidamente se envía la secuencia de caracteres 'U012*' que establece el nuevo desfase a 12 impulsos de encoder.

Esto se realiza en la subrutina de *inicialización del puerto serie*, ya que, en medio de la ejecución del programa se puede cambiar de puerto de comunicaciones (llamada a la rutina de inicialización del puerto serie), y si no se implementara aquí, y se hiciera sólo en la rutina de *inicializaciones*, se correría el riesgo que éstos parámetros nunca fueran configurados en el controlador.

La función *restaurar*, limpia la pantalla y restablece los colores originales y los vectores de interrupción iniciales, antes de salir de la aplicación.

Como ya se ha dicho, el núcleo del programa es la función *Menú Principal*, desde ésta se llamarán al resto de funciones que ejecutarán los diversos procedimientos de maniobra del robot, además de la ejecución de programas y gestión de ficheros.

En el programa se han implementado unos procedimientos básicos (movimiento por articulaciones, búsqueda de HARD-HOME, etc.) que son los que dotan de movilidad al robot. Éstos procedimientos tienen en común un par de funciones, que son sin duda de las más importantes de todo el programa:

Subrutina *Inversa* (ver página 200 del anexo)

Esta función implementa todos los cálculos descritos en la memoria de cálculo para determinar los parámetros articulares θ_1, θ_2, θ_3, θ_4, y θ_5. A partir de las coordenadas de posición del TCP, y de los vectores de acercamiento y orientación de la mano, se calculan los parámetros articulares.

Subrutina *Posición* (ver página 204 del anexo)

Esta función calcula la diferencia entre los parámetros articulares actuales y los parámetros articulares de la última posición conocida, además del sentido del movimiento a aplicar a los motores.

A partir de estos valores, se llama a las rutinas de accionamiento de cada motor pasando como parámetros el número de impulsos de encoder y el sentido del movimiento.

La rutina finaliza cuando se recibe por parte del controlador la interrupción que indica que todos los motores han finalizado su movimiento. De esta forma se consigue un control en lazo cerrado, al asegurarse que todos los motores han realizado su movimiento completo.

A continuación, se describen los procedimientos implementados en el programa.

1.8.1.- Movimiento por articulaciones

La Figura 1.42 muestra el diagrama de la secuencia que se ejecuta cuando el robot es operado por articulaciones independientes.

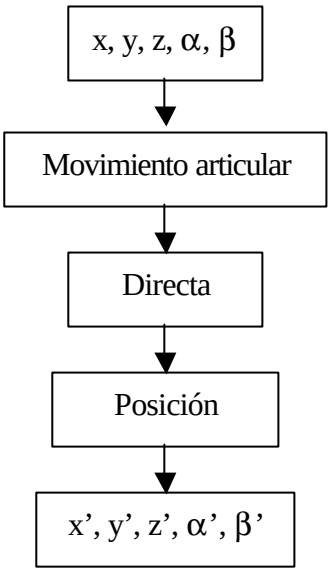

Figura 1.42. Diagrama de operación por articulaciones.

Se parte de una situación en que la posición y orientación del robot son conocidas, por tanto, se tienen unos valores determinados de p_x, p_y, p_z, a_x, a_y, a_z, o_x, o_y, o_z y de los parámetros articulares θ_1, θ_2, θ_3, θ_4, y θ_5.

Desde la función de *Movimiento articular* (ver página 213 del anexo), se incrementa o decrementa la variable articular correspondiente a la articulación que se desea mover (ya sea mediante teclado o mediante joystick).

Una vez se tienen las nuevas coordenadas articulares, que coincidirán con las anteriores excepto una de ellas que es la que se ha modificado, mediante la función *Directa* (ver página 202 del anexo), se recalcula la nueva posición teórica del brazo, es decir, los nuevos valores de posición del TCP, el vector acercamiento $\overline{a}$ y el vector orientación $\overline{o}$. Esto se consigue aplicando la

cinemática directa explicada en la memoria de cálculo, que es implementada por la función *Directa.*

A partir de los nuevos valores de la posición y de sus correspondientes coordenadas articulares se ejecuta la función *Posición* (ver página 204 del anexo), que realizará los movimientos adecuados actuando sobre el motor correspondiente, hasta llevar el brazo a la nueva posición y orientación, que dejará de ser teórica para ser real.

1.8.2.- Movimiento referido al TCP

En esta situación, el movimiento y orientación del brazo robotizado se realiza desde el punto de vista del TCP, que en este caso es el extremo de la herramienta de trabajo. Por ello, no se puede actuar directamente sobre los parámetros articulares, ya que, el movimiento del TCP puede producir la variación de diferentes parámetros articulares en diferentes cantidades.

a) <u>Desplazamiento en el espacio (TCP móvil)</u>

En este caso, el TCP se mueve por el espacio respecto uno de los ejes de coordenadas (*x*, *y* o *z*). De este modo, se pueden realizar trayectorias en línea recta sobre uno de los ejes, pero nunca fuera de éstos. Para ello sería necesario implementar la matriz Jacobiana que define la relación entre los movimientos y las velocidades de los motores, pero por limitaciones en el diseño del controlador del robot, esto no es posible. Además, debido a esto las líneas rectas que se realizaran en la dirección de uno de los ejes, se efectuaran a velocidad variable, dependiendo de los puntos por donde se pase, ya que para conseguir

esas posiciones se deberá mover más de un motor, pero en diferente cantidad de movimiento, y al no poder modificar la velocidad, los motores finalizarán su movimiento en diferentes tiempos.

La Figura 1.43 muestra el diagrama de la secuencia que se ejecuta cuando el robot es operado en este modo.

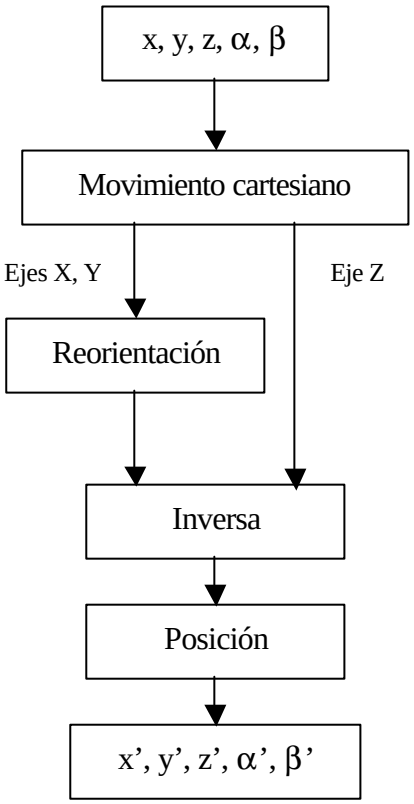

Figura 1.43. Diagrama de operación con TCP móvil.

Al igual que en caso anterior, se parte de una situación en que la posición y orientación del robot son conocidas, por tanto, se tienen unos valores determinados de p_x, p_y, p_z, a_x, a_y, a_z, o_x, o_y, o_z y de los parámetros articulares θ_1, θ_2, θ_3, θ_4, y θ_5.

La función *Movimiento cartesiano* (ver página 217 del anexo) realizará un incremento o decremento de 10 milímetros (resolución mínima permitida; valor

por defecto, modificable en el menú *Configuraciones del sistema*) sobre la posición de p_x, p_y o p_z, dependiendo de hacia donde se realice el movimiento.

En caso que el desplazamiento sea sobre los ejes X o Y, se debe efectuar una reorientación de los vectores $\overline{a}$ y $\overline{o}$. Esto es debido a que el robot no puede realizar el movimiento de *yaw* y por ello las proyecciones sobre el plano XY del vector de acercamiento $\overline{a}$ y del brazo del robot siempre tienen la misma dirección. Por lo tanto es imposible mantener la orientación de la muñeca en esta situación. Esto lo realiza la función *Reorientación* (ver página 227 del anexo, funciones Eje X y Eje Y).

En caso que el desplazamiento sea sobre el eje Z, no es necesario realizar ningún cálculo de orientación ya que los vectores $\overline{a}$ y $\overline{o}$ no se ven modificados por el movimiento en esta dirección.

Seguidamente se ejecuta la función *Inversa* para calcular los parámetros articulares de la nueva posición teórica y, una vez calculados estos parámetros, se ejecuta la función *Posición* para llevar el brazo hacia esta nueva situación.

b) Orientación de la mano (TCP fijo)

En esta situación, el TCP mantiene su posición en el espacio mientras se modifica la orientación de la mano, es decir, el brazo articulado mueve diferentes eslabones para conseguir la orientación deseada, pero el extremo de la mano conserva siempre la misma posición.

La Figura 1.44 muestra el diagrama de la secuencia que se ejecuta cuando el robot es operado de este modo.

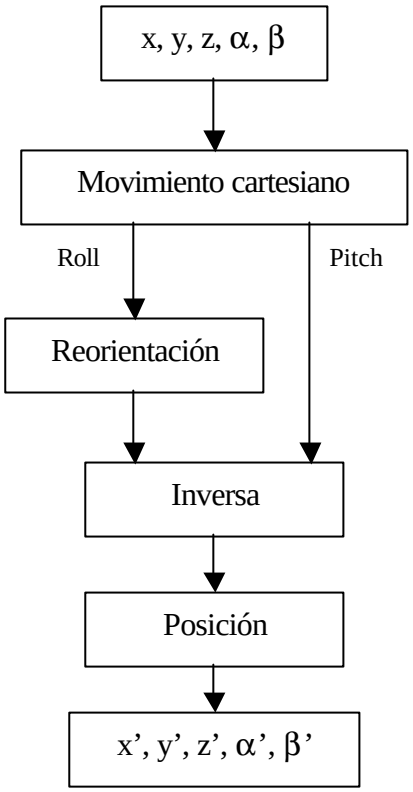

Figura 1.44. Diagrama de operación con TCP fijo.

Al igual que antes, se parte de una situación en que la posición y orientación del robot son conocidas. Seguidamente la función *Movimiento cartesiano* incrementará o decrementará en 1º (resolución mínima permitida por el sistema) el Roll o el Pitch.

En el caso que se realice una operación sobre el Roll se debe efectuar una reorientación del vector $\overline{o}$ (ver página 226 del anexo, función *Muñeca tcp*). Esta función calcula los nuevos valores de las componentes del vector orientación $\overline{o}$. Para ello, incrementa o decrementa en 1º el valor del parámetro articular θ_5, y aplicando la matriz de cinemática directa, obtiene el nuevo vector orientación $\overline{o}$.

Para el caso del incremento o decremento del valor de Pitch, no se efectúa ninguna operación adicional, ya que la función *Inversa* ha sido diseñada

teniendo en cuenta esta posibilidad de variación del Pitch con TCP fijo, a la hora de realizar los cálculos de la posición del extremo del brazo (sin tener en cuenta la muñeca). En este caso, la función *Inversa* incrementará o decrementará en 1º el ángulo β utilizado para calcular el vector de acercamiento $\overline{a}$ (ver apartado *2.1.4.5.- Determinación de las coordenadas de P_b (P_{bx}, P_{by}, P_{bz})*).

Finalmente, la función *Posición* llevará a cabo el movimiento de los motores para situar el brazo en la posición deseada.

1.8.3.- Ir a un punto del espacio

Este procedimiento permite situar el brazo en una posición del espacio con una orientación determinada, entrando los valores directamente por el teclado.

La Figura 1.45 muestra el diagrama de la secuencia que se ejecuta cuando se quiere llevar el robot a un punto determinado del espacio.

Al igual que en los casos anteriores, se parte de una situación en que la posición y orientación del robot son conocidas, por tanto, se tienen unos valores determinados de p_x, p_y, p_z, a_x, a_y, a_z, o_x, o_y, o_z y de los parámetros articulares θ_1, θ_2, θ_3, θ_4, y θ_5.

La función *Datos de la posición* (ver página 236 del anexo) se encarga de capturar los datos de la posición y orientación introducidos por teclado. Los valores de la orientación de la muñeca se introducirán como ángulos de Pitch y Roll.

Con estos valores de Pitch y Roll, la función *Orientación* (ver página 252 del anexo) calcula las componentes de los nuevos vectores acercamiento $\overline{a}$ y orientación $\overline{o}$, tal y como se detalla en la memoria de calculo.

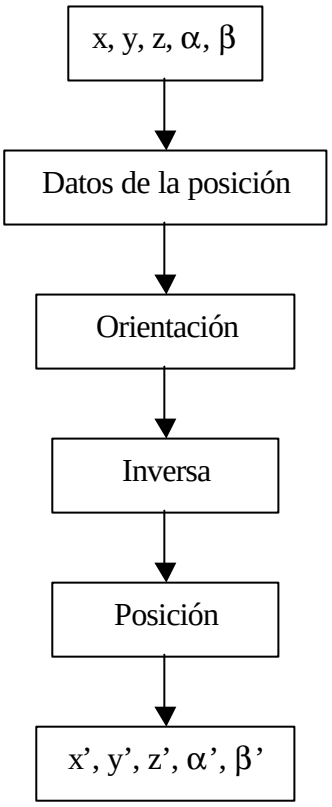

Figura 1.45. Diagrama de posicionamiento en un punto concreto.

Una vez conocidos los valores de las componentes de la posición y de los vectores de acercamiento $\overline{a}$ y de orientación $\overline{o}$, la función *Inversa* calcula los nuevos parámetros θ_1, θ_2, θ_3, θ_4, y θ_5. La función *Posición* se encarga, nuevamente, de llevar el brazo hasta la nueva situación.

1.8.4.- Determinación manual de HOME

Para la determinación de la posición HOME de forma manual, se debe llevar previamente al robot a dicha posición, ya sea manualmente con los motores del robot desconectados y situando el robot en la posición y orientación deseados, o mediante el accionamiento de los motores de las articulaciones desde el submenú de *Movimiento por articulaciones*, del menú *HOME*, ya que desde este submenú no se actualiza la posición y la orientación.

La Figura 1.46 muestra el diagrama de la secuencia que se ejecuta cuando se quiere determinar la posición HOME manualmente.

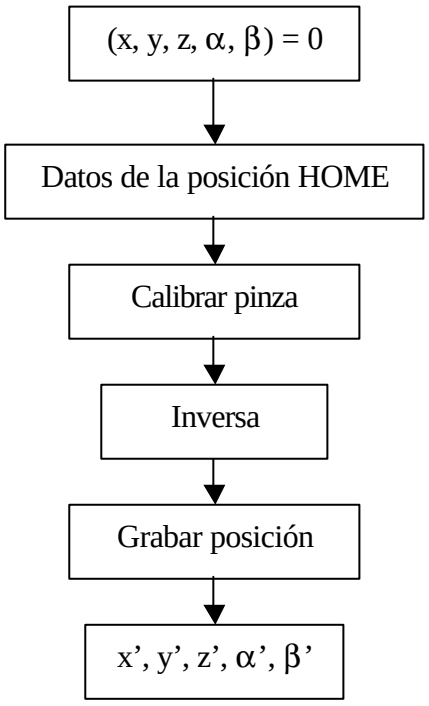

Figura 1.46. Diagrama de determinación de HOME.

Inicialmente, las variables de posición, los vectores de acercamiento $\overline{a}$ y de orientación $\overline{o}$ y los parámetros articulares θ_1, θ_2, θ_3, θ_4, y θ_5 tienen el valor cero.

Una vez con el robot situado en la posición deseada, la función *Datos de la posición HOME* (ver página 236 del anexo) se encarga de recoger los valores

de las componentes de la posición y de los vectores de acercamiento $\overline{a}$ y de orientación $\overline{o}$. En este caso, los valores de la orientación deben introducirse en forma de vectores y no en como ángulos, ya que todavía no existe una posición de referencia a la que referir estos ángulos.

Seguidamente, la función *Calibrar pinza* (ver página 197 del anexo, esta función está implementada dentro de la función HOME), cerrará y abrirá la pinza para obtener los valores extremos de apertura y cierre. Con estos valores se evitará que se sobrecargue el motor de accionamiento de la pinza por un uso indebido.

A continuación, la función *Inversa* calculará los parámetros articulares a partir de los valores de la posición y orientación introducidos.

Finalmente, la función *Grabar posición* (ver página 221 del anexo), se encargará de almacenar los datos de la posición y de la orientación (tanto en forma de vector, como de ángulos) en la primera posición del vector de posiciones.

1.8.5.- Determinación automática de HARD-HOME

La Figura 1.47 muestra el diagrama de la secuencia que se ejecuta cuando se quiere determinar la posición HARD-HOME automáticamente.

Al igual que en caso anterior, se parte de una situación en que tanto las variables de posición y de los vectores de acercamiento $\overline{a}$ y de orientación $\overline{o}$, como los parámetros articulares θ_1, θ_2, θ_3, θ_4, y θ_5 tienen el valor cero.

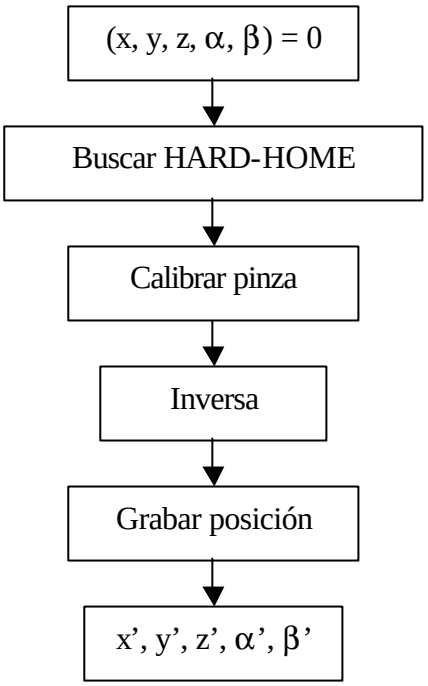

Figura 1.47. Diagrama de determinación de HARD-HOME.

Partiendo de este punto, se ejecuta la función *Buscar HARD-HOME* (ver página 198 del anexo). Esta función se encarga de mover cada una de las articulaciones de forma secuencial, empezando por la base y terminando por las articulaciones de la muñeca. El movimiento se inicia en un sentido, y en caso de llegar al final del recorrido y no encontrar la posición de HARD-HOME de la articulación, se inicia el movimiento en el sentido opuesto.

La posición de HARD-HOME la marcará una interrupción enviada desde el microcontrolador, indicando que el microswitch asociado a esa articulación ha sido pulsado o liberado. La pieza mecánica que actúa sobre el microswitch es la mostrada en la Figura 1.48. Esta pieza esta montada sobre el propio eje de giro de la articulación y gira con ella. Como se puede observar, la parte de la pieza que oprime el microswitch es bastante ancha, con lo que, dependiendo del sentido de giro, la posición de HARD HOME podría variar bastante.

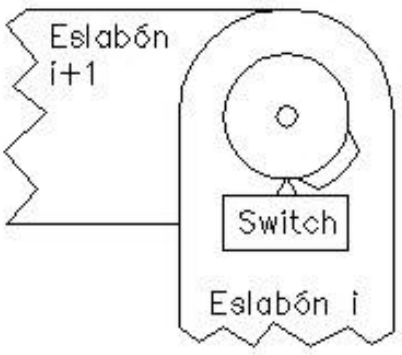

Figura 1.48. Pieza que actúa sobre el microswitch de HARD-HOME.

Es por ello que la función Buscar HARD-HOME tiene en cuenta el sentido del movimiento y el estado del microswitch, es decir, si este pasa del estado ON a OFF o viceversa, de modo que la posición de HARD-HOME para cada articulación sea siempre la misma.

Una vez fijadas todas las articulaciones a la posición de HARD-HOME, se establecen los siguientes valores de posición y orientación:

Tabla 1.16. Valores de posición HARD-HOME.

Coordenada	Valor
p_x	213.6
p_y	0.0
p_z	565.0
a_x	0.8
a_y	0.0
a_z	-0.51
o_x	0.0
o_y	1.0
o_z	0.0

Estos valores son el resultado de aplicar la cinemática directa al valor, medido empíricamente, de las coordenadas articulares.

En esta situación, el brazo del robot está situado justo encima del eje X. Los valores de la orientación están referidos a una posición de Pitch=0º y Roll=0º, cuando la muñeca esta horizontal y los dos dedos paralelos al plano de la base y a la misma altura.

A continuación se ejecutará la función *Calibrar pinza*, al igual que en el caso anterior, para evitar que se estropee el motor de accionamiento de la pinza por un exceso de apertura o cierre.

Seguidamente, la función *Inversa*, se encargará de obtener los parámetros articulares, y a continuación la función *Grabar posición*, guardará los valores de la posición y orientación en la primera posición del vector de posiciones.

1.8.6.- Trayectoria continua

El procedimiento seguido para guardar una trayectoria es el mismo que para el movimiento por articulaciones o por coordenadas cartesianas, pero se activa un flag de sistema que guarda automáticamente la posición y orientación en el vector de posiciones después de cada movimiento realizado.

El por qué se ha implementado de esta manera esta función en lugar de guardar los datos relativos a la posición cada cierto tiempo se explica a continuación.

Si se elige un intervalo de tiempo muy grande, se corre el riesgo de perder precisión en la trayectoria, ya que habría puntos por los que se pasaría

pero no quedarían guardados. Por otra parte, si se elige un intervalo de tiempo muy pequeño, se corre el riesgo de que se guarde la misma posición varias veces, ocupando un espacio innecesario en la memoria, que podría servir para guardar nuevas posiciones, ya que el tamaño del vector de posiciones es limitado.

Además, teniendo en cuenta que a la hora de editar un programa se pueden introducir pausas entre posiciones o variar la velocidad de los motores, se ha considerado que la solución óptima es la de guardar cada cambio de posición, para conseguir la máxima precisión con el mínimo espacio de memoria ocupado, pudiendo después ejecutarlas de forma continua o realizando pausas entre ellas.

1.8.7.- Gestión de ficheros

Existen tres tipos de ficheros que maneja la aplicación, todos ellos ficheros de texto que pueden ser leídos desde cualquier editor. Sin embargo no se aconseja que sean modificados fuera de la aplicación, ya que se pueden introducir erróneamente caracteres o marcas que aunque no se vean, pueden ocasionar problemas a la hora de volver a abrir los ficheros desde la aplicación.

Los ficheros que maneja la aplicación son los siguientes:

- Fichero de configuraciones del usuario. Este fichero tiene el nombre "robot.cfg". En el se guardan el valor de los flags que modificaran el aspecto de la pantalla que se mostrará. Las opciones que se permiten han sido explicadas en el aparatado 1.7, donde se explica el menú de *Configuraciones del sistema.* Estos flags de sistema, son consultados al inicio del programa para cargar en el sistema la última configuración que

se ejecutó. Cada vez que estos flags son modificados, el fichero de configuración es reescrito con los nuevos valores.

- Ficheros de posiciones. El nombre de estos ficheros es variable, elegido por el usuario en el momento de crearlo, y puede ser un campo alfanumérico de hasta 8 caracteres. La extensión de este tipo de ficheros es "*nombre_fichero*.pos". El fichero es de texto, y la información se guarda de la siguiente manera:

Pos *i* *x* *y* *z* A *x* *y* *z* AB *j* O *x* *y* *z* ED *k*

Donde *i*, es el número de la posición (0 en el caso de la posición HOME), los valores *x*, *y* y *z* son los valores de las componentes de la posición y de los vectores de acercamiento $\overline{a}$ y de orientación $\overline{o}$. El valor *j*, es valor del Pitch, en radianes, y *k* es el valor del Roll en radianes.

Para cada posición se escribe una línea de fichero .

- Ficheros de programas. El nombre de estos ficheros es variable, elegido por el usuario en el momento de crearlo, y puede ser un campo alfanumérico de hasta 8 caracteres. La extensión de este tipo de ficheros es "*nombre_fichero*.pgm". El fichero es de texto, y la información se guarda de la siguiente manera:

Lin *i* *comando* *j*

Donde *i* es el número de línea de programa, *comando* es un string que especifica el comando a ejecutar (ir a posición, pausa, abrir pinza, etc.) y *j* es la cantidad del comando a ejecutar (número de posición, duración de la pausa, etc.).Para cada instrucción de programa se escribe una línea de fichero.

1.8.8.- Control del Joystick

Inicialmente, la lectura de datos del joystick se iba a realizar empleando el procedimiento descrito en el apartado *1.6.4.3.- Medida de la posición del joystick*, midiendo los tiempos de carga del condensador para cada eje, y aplicando la ecuación de carga de un circuito RC, hallar el valor de las resistencias de los potenciómetros, y así determinar la posición de la palanca.

Pero este procedimiento es costoso, ya que se deben controlar unos intervalos de tiempo muy pequeños, y no es del todo preciso debido a la tolerancia de los valores de las resistencias y los condensadores.

Para obtener los valores del joystick, se ha decidido utilizar un procedimiento similar, igual de efectivo aunque mucho más simple de implementar. Se trata de disparar el conteo de los multivibradores realizando una escritura de un byte en la dirección del puerto de joystick (201h) y seguidamente empezar el conteo de una variable dentro de un bucle *while*.

Dentro del bucle, se incrementa la variable y se lee continuamente el puerto de joystick hasta que se detecta que los dos condensadores (eje X y eje Y) se han cargado (bit 0 y bit 1 con un '1' lógico). Para que este sistema de medida sea válido, es necesario inhibir las interrupciones antes de iniciar el bucle y desinhibirlas al acabar. De este modo, nos aseguramos que contaje sea siempre el mismo y que no varíe por la posible ejecución de alguna rutina de interrupción. Todo esto lo realiza la función *joystick* (ver página 230 del anexo).

Mediante un procedimiento de calibrado, situando el joystick en sus posiciones máximas (Arriba-Izquierda, Abajo-Derecha) y en la posición central, se obtiene el rango de valores que se podrán conseguir (uno para el eje X y otro para el eje Y). Escalando estos rangos, se puede determinar el movimiento del joystick, incluso establecer diferentes acciones según el grado de inclinación de la palanca (Figura 1.49).

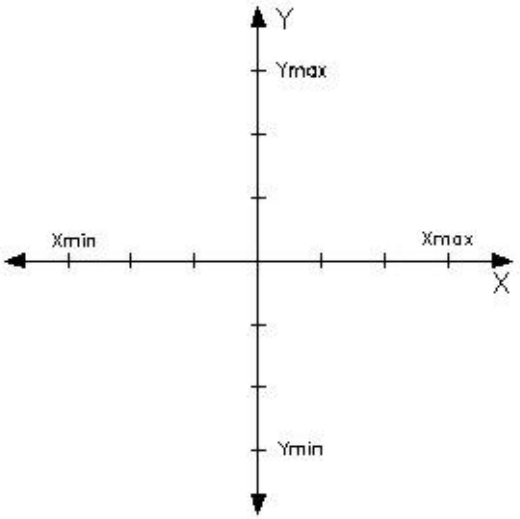

Figura 1.49. Escalado de los ejes del joystick.

El movimiento del robot mediante el joystick, seguirá la siguiente secuencia:

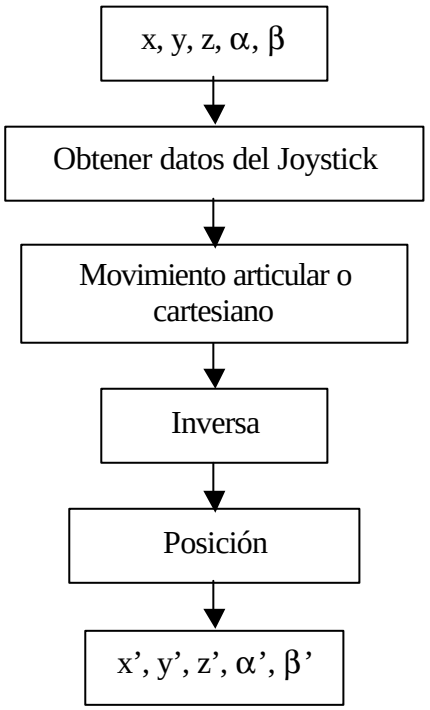

Figura 1.50. Procedimiento para mover el robot mediante joystick.

Se puede observar, que mientras se realiza el conteo para la determinación de la posición del joystick, no se ejecuta ninguna otra función (movimiento del brazo, cálculos de parámetros, etc.). Esto no tiene efecto

alguno sobre la manipulación continua del brazo ya que, el tiempo de carga de los condensadores del joystick es del orden de microsegundos, con lo cual el tiempo de ejecución de la función es de unos pocos microsegundos. Acto seguido se ejecutan las funciones de cálculo de la nueva posición y de accionamiento de los motores, que también consumen un tiempo de unos pocos microsegundos. Teniendo esto en cuenta, y que el inicio de un nuevo ciclo de movimiento se inicia antes que el motor haya terminado completamente su movimiento actual, (el controlador avisa mediante una interrupción de la proximidad del fin de movimiento), se consigue que el movimiento del robot sea suave y continuo.

1.8.9.- Comunicaciones

Las comunicaciones entre el ordenador y el controlador, son el punto más crítico de todo el programa. Éstas se realizan mediante la norma RS-232-C, con el protocolo de transmisión ya explicado en el apartado *1.6.3.-Comunicación controlador-ordenador*. La gestión de las comunicaciones se realiza mediante el uso de interrupciones, concretamente las correspondientes a las IRQ3 e IRQ4, que corresponden al puerto de comunicaciones Com2 y Com1 respectivamente.

Estas interrupciones serán generadas por la UART (Universal Asynchronus Reciver/Transmiter). Este dispositivo lógico es el que usan los PC's para gestionar las comunicaciones serie.

Primeramente, mediante la función *inicialización del puerto serie* (ver página 191 del anexo), se accede a los registros de control de la UART para indicarle que los parámetros del protocolo que se va a usar (velocidad de transmisión, bits de paridad, etc.).

Tabla 1.17. Valores de los registros de comunicación.

Registro	Valor
LCR	0x80
LSB	0X0c
MSB	0x00
LCR	0x07

Esto se consigue modificando los registros LCR, LSB y MSB, tal como se muestra en la Tabla 1.17. Seguidamente se le indica a la UART el tipo de interrupciones que ha de generar. En este caso se generarán interrupciones por el envío de un dato, la recepción de un dato y errores de la línea y módem. Esto se consigue escribiendo el valor hexadecimal 0x0f en el registro IER. A continuación, se habilita que la UART envíe interrupciones al controlador de interrupciones de PC, escribiendo el valor 0x08 en el registro MCR.

Una vez hecho esto, se hace una lectura de los registros MSR, LSR, IIR, RBR y THR para eliminar posibles interrupciones pendientes.

El diseño de la rutina que se ejecuta cuando se genera una interrupción por parte de la UART, requiere especial atención puesto que esta rutina debe ser lo más corta posible (ya que mientras se ejecuta ésta, la ejecución del programa queda interrumpida) y debe tener en cuenta todas las posibles causas de la interrupción y actuar en consecuencia.

Por ello, el primer paso una vez iniciada la rutina de servicio a la interrupción, es mirar por qué razón se ha generado ésta (dato recibido, dato enviado, etc.). Esto se realiza consultando los bits del registro IIR.

En el caso que la interrupción se haya generado por el envío satisfactorio de un dato a través de la línea serie, por un error en la línea de transmisión o por un error del módem, entonces el único tratamiento que se hace es el de

incrementar las variables asociadas a estos hechos específicos con el fin de tener un control de estado de las transmisiones, ya que estos hechos no repercuten en la comunicación en si misma.

El caso a tener en cuenta es cuando se recibe un dato por la línea serie. Entonces, se analiza el dato recibido, y según el valor de éste, y del modo de operación en que se encuentra el robot (movimiento del brazo, búsqueda de Hard-Home, etc.) se modifican unas variables u otras (hay que tener en cuenta que no se pueden ejecutar funciones desde una rutina de servicio a interrupción).

En el caso de búsqueda automática de HARD-HOME, los datos que se esperan recibir son los caracteres 'K' o 'L' seguidos del número de motor, que indican que se ha encontrado la posición de HARD-HOME para la articulación.

En el caso de movimiento del brazo, ya sea por articulaciones, o por coordenadas cartesianas, se analiza el dato recibido esperando que sea el carácter '0' que marca el fin de movimiento de los motores, o el carácter '-' que indica que se acerca el fin del movimiento de los motores.

En ambos casos, también se controla si se ha producido un error en la operación de alguno de los motores. Esto es indicado por la recepción de un carácter con el número del motor agarrotado.

El envío de datos a través de la línea serie, no representa ningún problema, ya que se envía la secuencia de caracteres a ser transmitidos a la UART y ella se encarga de enviarlos a través de la línea. Estos caracteres quedan almacenados en un buffer a la espera de poder ser enviados, y cada vez que un dato se transmite con éxito, el siguiente pasa a ser enviado.

Por parte del controlador del robot, hay que mencionar que se trata de un aparato de inicio de los años 80, con circuitería propia de su época, con lo que el sistema de control de comunicaciones y de gestión de interrupciones que lleva incorporado, está algo obsoleto. Esto ha provocado problemas en las comunicaciones por la pérdida de datos cuando se realizan transmisiones por parte del controlador del robot hacia el ordenador de forma continua y muy seguida como respuesta a un comando enviado por el ordenador.

Concretamente cuando se envía una orden de accionamiento de motor al controlador, y este movimiento del motor es de corta duración, la respuesta del controlador indicando que se ha realizado el movimiento con éxito del motor no llega nunca al ordenador, provocando una situación de fallo del sistema.

Esto limitaba bastante la resolución del sistema, ya que para no perder datos en las comunicaciones, se debían enviar un mínimo de 25 impulsos para estar seguros de no perder la respuesta que enviaría el controlador. De este modo la resolución del sistema era muy pequeña ya que los movimientos de los motores eran excesivamente grandes. Para solucionar esto se ha usado el comando del sistema operativo del controlador 'A'. Cuando se envía este comando, el controlador responde con un carácter '0' si todos los motores han finalizado el movimiento (NOTA: este comando también responde con otros caracteres según la información que vaya a enviar, pero en este caso, sólo interesa recibir el '0').

Se ha implementado un timer software de ½ segundo de duración. Este timer empieza a ejecutarse de forma cíclica mientras se ejecuta la rutina de *Posición*, y cada vez que expira envía el carácter 'A' al controlador. De este modo, en caso perderse la interrupción enviada por el controlador cuando el motor finaliza el movimiento, se asegura que al cabo de ½ segundo como mucho, se detectará que el robot ha finalizado el movimiento.

1.8.10.- Resolución del sistema

Como ya se ha mencionado, el sistema de comunicaciones no es todo lo eficaz que debería ser, y esto limita bastante las operaciones del robot. Debido a esto, cuanta mayor resolución se desee, menor suavidad en el movimiento continuo del brazo se conseguirá, ya que en muchas ocasiones habrá que esperar ½ segundo para detectar la finalización del movimiento del motor.

En el movimiento por articulaciones, el número de pulsos mínimo será el que definirá la resolución mínima del sistema. La Tabla 1.18 muestra las resoluciones mínimas de las articulaciones, calculadas en el apartado *2.4.6- Resolución mínima del sistema,* para 10 pulsos de encoder en cada articulación, que es el incremento mínimo de pulsos permitido. Para la articulación de la muñeca, el número de pulsos mínimo será siempre la mitad de la del resto de las articulaciones, en este caso 5 pulsos. Se ha establecido así debido a que la articulación de la muñeca tiene una resolución más baja que las otras, y el movimiento era excesivamente grande.

En el caso del movimiento por articulaciones, la resolución vendrá dada por el desplazamiento mínimo del TCP sobre cada uno de los ejes coordenados. Este parámetro es configurable por el usuario, y el valor mínimo es de 10 mm. En el caso de la mano o herramienta de trabajo, el ángulo de Pitch y Roll mínimo seleccionable por el usuario es de 1º.

Tabla 1.18. Resolución mínima de cada articulación.

Articulación	Resolución mínima
Base	0.94º
Hombro	1.17º
Codo	1.17º
Muñeca (Pitch)	2.29º
Muñeca (Roll)	2.29º

1.9.- Resumen del presupuesto

El precio de licitación resultante del presente proyecto, *Telemanipulación de un brazo robotizado articulado*, asciende a la cantidad de **2.122,51 euros** (DOS MIL CIENTO VEINTIDOS EUROS CON CINCUENTA Y UN CÉNTIMOS), teniendo en cuenta los gastos generales (13%), el beneficio industrial (6%) y el I.V.A. (16%), que se halla detallado en el apartado *3.- Presupuesto.*

Constantí, 24 de Enero de 2001

El ingeniero técnico industrial

Fdo. Joan Mercadé Papiol

1.10.- Bibliografía

La bibliografía consultada a lo largo de la elaboración del presente proyecto se enumera detalladamente a continuación:

[1] A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de robótica", Ed. McGraw-Hill

[2] Robert J. Shilling, "Fundamentals of robotics: analysis and control", Ed. Prentice Hall

[3] D. García, E. Guàrdia, "Elements de mecànica aplicada a la robòtica", Ed. UPC.

[4] M. Croquet, "PC y robótica. Técnicas de interfaz", Ed. Paraninfo.

[5] B. Gottfried, "Programación en C", Ed. McGraw-Hill.

[6] H. Schildt, "Turbo C, programación avanzada", Segunda edición, Ed. Borland-Osborne / McGraw-Hill

[7] "Manual del usuario del ESCORBOT ER-III", Eshed Robotec Inc.

[8] "Apuntes de Oficina Técnica". Universitat Rovira i Virgili (E.T.S.E.). Profesor: Carlos Turón. Año 1999.

[9] "Apuntes de Informática Industrial I". Universitat Rovira i Virgili (E.T.S.E.). Profesor: Enric Vidal. Año 1998.

[10] "Apuntes de Informática Industrial II". Universitat Rovira i Virgili (E.T.S.E.). Profesor: Esteban del Castillo. Año 2000.

[11] "Apuntes de Robótica". Universitat Rovira i Virgili (E.T.S.E.). Profesor: Rafael Iñigo. Año 1997.

[12] "Apuntes de Introducción a los Ordenadores". Universitat Politècnica de Catalunya (E.T.S.E.T.B.). Profesor: Juan Carlos Cruellas. Año 1996.

Páginas Web consultadas en la elaboración del proyecto:

[13] "EL Universo digital del IBM PC, AT y PS-2",
http://atc.ugr.es/docencia/udigital/

[15] "ePanorama. All about electronics", http://www.epanorama.net

2.- MEMORIA DE CÁLCULO

2.1.- Modelo cinemático del robot ESCORBOT ER-III

El modelo cinemático, permite describir la posición y orientación del robot. Esto se realiza por medio de una sola matriz 4x4 de traslación y rotación, denominada matriz total de transformación ${}^0T_n$,de tal forma que se puede pasar de un punto dado en el sistema de coordenadas de la herramienta a un punto expresado en el sistema de coordenadas de referencia o base.

La matriz de transformación tiene el siguiente aspecto:

$$ {}^0T_n = \begin{bmatrix} u_x & o_x & a_x & p_x \\ u_y & o_y & a_y & p_y \\ u_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} [R] & P \\ 0 & 1 \end{bmatrix} $$

Para calcular el modelo cinemático del ESCORBOT ER-III se ha decidido utilizar el método de Denavit-Hartenberg por ser éste el más utilizado en robótica, y establecer una descripción de la cinemática del robot con el menor número posible de parámetros para cada articulación.

2.1.1.- Método de Denavit-Hartenberg

En un robot manipulador una articulación ‘i’ conecta dos eslabones, el eslabón ‘i-1’ y el eslabón ‘i’, como se muestra en la Figura 2.1. El eslabón cero es la base estática del robot y se usa como referencia y los eslabones móviles se numeran del 1 al ‘n’. Con referencia a la Figura 2.1, el eslabón ‘i’ es controlado

por la articulación ‘i’ y su movimiento es referido al eslabón anterior, el ‘i-1’. Un robot con ‘n’ grados de libertad tiene ‘n+1’ eslabones, contando el número cero (de base), y ‘n+1’ sistemas de coordenadas, también contando el de referencia o base y el de herramienta o mano.

Una matriz transformación de eslabón será designada como $^{i-1}\mathbf{A}_i$, indicando que esta matriz transforma puntos dados en el sistema ‘i’ a puntos en el sistema ‘i-1’. De este modo, una vez conocidas todas las matrices de transformación de eslabón para cada uno de los eslabones que componen el robot, y multiplicando todas ellas, se podrá pasar de un punto expresado en el sistema ‘n’ (mano o herramienta) a un punto expresado en el sistema 0 (base o referencia). Para determinar una matriz de transformación de eslabón se deben determinar el número necesario y suficiente de parámetros que describen la relación entre dos eslabones.

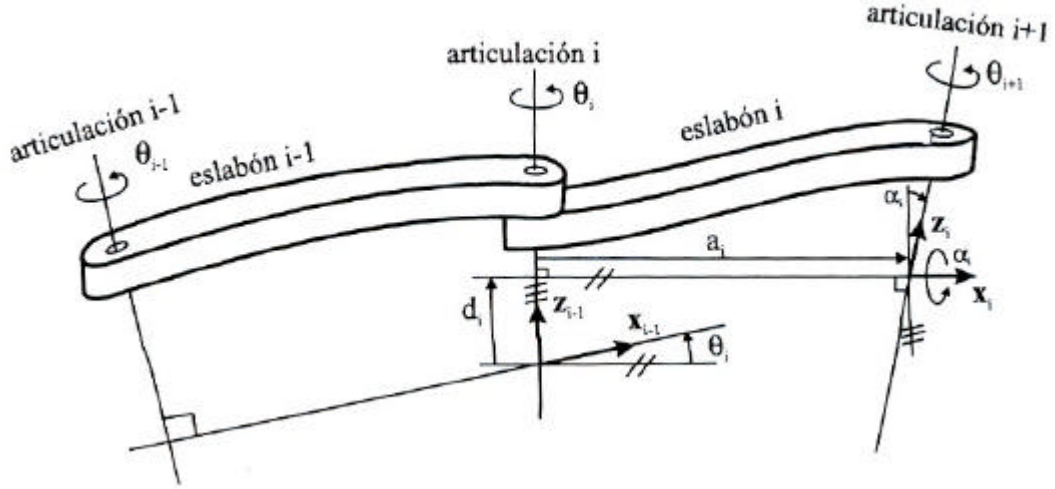

Figura 2.1. Parámetros D-H para un eslabóngiratorio.

2.1.1.1.- Parámetros de eslabón

En primer lugar se deben determinar cuantos parámetros son necesarios para definir una matriz de transformación. Si dos sistemas de coordenadas tienen el mismo origen pero uno está girado con respecto al otro, basta con efectuar dos rotaciones sobre dos de los ejes para que queden en coincidencia. Esto se debe a que, para cualquier orientación de un sistema con respecto al otro, una rotación basta para hacer que un plano (x-y, y-z ó z-x) coincida con el plano respectivo del otro sistema. Una segunda rotación sobre otro eje hará que un segundo plano coincida con el respectivo de otro sistema (sin afectar la coincidencia de los anteriores). El tercer plano queda automáticamente en coincidencia, ya que los tres planos son perpendiculares. Para que los dos orígenes coincidan si inicialmente están desplazados, basta con dos traslaciones. La distancia más corta entre dos líneas es la perpendicular a ambas. Si se traslada el origen de {1} a lo largo de la línea perpendicular (por ejemplo) a $\mathbf{z_1}$ y a $\mathbf{z_0}$ se consigue que el origen de {1} se encuentre sobre $\mathbf{z_0}$. Basta entonces una segunda traslación a lo largo de $\mathbf{z_0}$ para que los dos orígenes coincidan. Queda demostrado que dos traslaciones y dos rotaciones bastan para que dos sistemas queden en coincidencia. Es decir, sólo cuatro parámetros son necesarios para definir una matriz de transformación entre dos articulaciones.

Según el método de Denavit-Hartenberg se opera en el sistema 'i-1' hasta llevarlo a coincidir con el sistema 'i' para determinar ${}^{i-1}A_i$, aunque en la realidad es el sistema 'i' el que se mueve con respecto a 'i-1', empezando en la mano y terminando en la base. Las operaciones definidas por Denavit-Hartenberg son con respecto al sistema en movimiento y por lo tanto las matrices parciales deben posmultiplicarse, empezando por la primera operación posmultiplicada por la segunda, etc.

Para cada eslabón se establece un sistema de coordenadas. Con referencia a la Figura 2.1, el eje 'i' se coloca a lo largo del eje de la articulación 'i+1'. La

articulación puede ser rotatoria, en cuyo caso el eje será un eje de giro, o prismática, en cuyo caso el eje será un eje de deslizamiento. El sistema de coordenadas 'i' se obtiene eligiendo:

- eje $\mathbf{z_i}$ a lo largo del eje de la articulación 'i+1'
- origen de {i} en la intersección de $\mathbf{z_i}$ con la normal común a los ejes $\mathbf{z_i}$ y $\mathbf{z_{i-1}}$; O_i' es el punto de intersección de la normal común con $\mathbf{z_{i-1}}$ (ver Figura 2.1)
- eje xi a lo largo de la normal común a $\mathbf{z_i}$ y $\mathbf{z_{i-1}}$ en sentido articulación 'i' a articulación 'i+1'
- el eje $\mathbf{y_i}$ queda determinado automáticamente (sistema de mano derecha)

Observaciones:

- para el sistema {0} sólo se especifica la dirección de $\mathbf{z_0}$; O_0 y $\mathbf{x_0}$ pueden elegirse arbitrariamente, por supuesto con x0 perpendicular a $\mathbf{z_0}$; O_0 se elige generalmente al pie de la base
- en el sistema {n} (mano o herramienta) $\mathbf{x_n}$ debe ser normal al eje $\mathbf{z_{n-1}}$ y $\mathbf{z_n}$ (que es el vector de acercamiento **â**) se elige en la dirección de avance de la herramienta
- cuando dos ejes consecutivos son paralelos, la normal común, en lugar de un valor, tiene una infinidad de valores
- cuando dos ejes consecutivos se intersectan, la dirección de $\mathbf{x_i}$ es arbitraria

Una vez establecidos los sistemas de coordenadas, la posición y orientación de {i} con respecto a {i-1} quedan completamente definidas por medio de los cuatro parámetros mostrados en la siguiente tabla:

Tabla 2.1 Parámetros Denavit-Hartenberg.

Parámetro	Descripción
a_i	Distancia de O_i a O_i'
d_i	Distancia de O_i' a O_{i-1} a lo largo de z_{i-1}
α_i	Ángulo formado por los ejes z_{i-1} y z_i con giro sobre el eje xi en el sentido matemáticamente positivo
θ_i	Ángulo entre los ejes x_{i-1} y x_i girando positivamente sobre el eje z_{i-1}

Para una articulación rotatoria, θ es variable y los otro tres parámetros constantes; para una articulación prismática, *d* es variable y los otros tres parámetros constantes.

Nuevamente con referencia a la Figura 2.1, las siguientes operaciones colocarán a los dos sistemas en coincidencia:

- desplazar O_{i-1} a lo largo de $\mathbf{z_{i-1}}$ en una distancia **d** hasta que O_{i-1} coincida con O_i'
- rotar $\mathbf{x_{i-1}}$ sobre $\mathbf{z_{i-1}}$ en un angulo θ_i hasta que coincida con la dirección de $\mathbf{x_i}$
- mover O_{i-1} a lo largo de $\mathbf{x_{i-1}}$ (que ahora coincide en dirección $\mathbf{x_i}$) en una distancia **a**; esto hace que los orígenes de ambos sistemas coincidan
- finalmente, rotar el eje $\mathbf{z_{i-1}}$ en un ángulo α sobre $\mathbf{x_{i-1}}$

Las matrices parciales deben posmultiplicarse, empezando por la de traslación a lo largo de $\mathbf{z_{i-1}}$ en una distancia **d** y terminando con la de rotación sobre $\mathbf{x_{i-1}}$ en un ángulo α:

$$^{i-1}A_i = T_{z,d} R_{z,q} T_{x,a} R_{x,a} \qquad [2.1]$$

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

$$ {}^{i-1}A_i = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i sen\theta_i & sen\alpha_i sen\theta_i & a_i\cos\theta_i \\ sen\theta_i & -\cos\alpha_i \cos\theta_i & -sen\alpha_i \cos\theta_i & a_i sen\theta_i \\ 0 & sen\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad [2.2] $$

Usando la matriz ${}^{i-1}A_i$ se pueden referir puntos dados en el sistema {i} (eslabón ‘i’) al sistema {i-1},

$$ p_{i-1} = {}^{i-1}A_i\, p_i \qquad [2.3] $$

o, en general,

$$ p_0 = {}^{0}A_1\, {}^{1}A_2 \ldots\ldots {}^{n-1}A_n\, p_n \qquad [2.4] $$

2.1.2.- Parámetros de eslabón de Denavit-Hartenberg

En la Tabla 2.2 se muestran los parámetros calculados según el método mencionado:

Tabla 2.2. Parámetros Denavit-Hartenberg del ESCORBOT ER-III

Eslabón	d_i	q_I	a_i	a_I
1	d_1	θ_1	a_1	-90
2	0	θ_2	a_2	0
3	0	θ_3	a_3	0
4	0	θ_4	0	90
5	d_5	θ_5	0	0

Los ejes de coordenadas quedan establecidos como se muestra en la Figura 2.2.

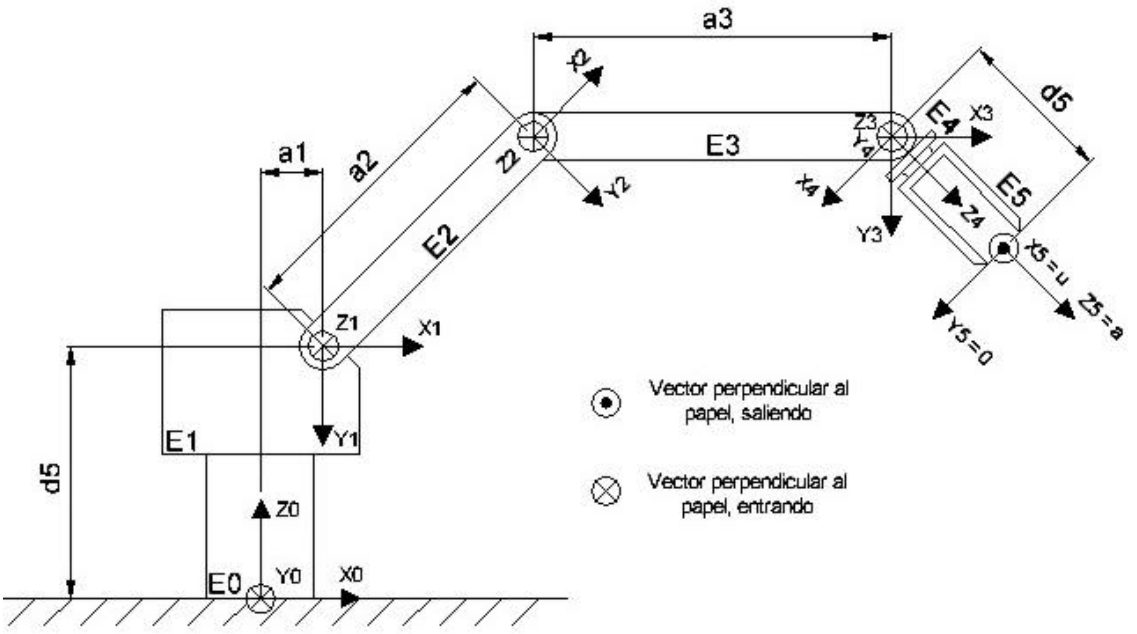

Figura 2.2. Ejes de coordenadas según Denavit-Hartenberg.

2.1.3.- Cinemática directa

La matriz de transformación para pasar del sistema de coordenadas de un eslabón al siguiente, viene expresada por la relación anteriormente expresada [2.2]:

$$ {}^{i-1}A_i = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i sen\theta_i & sen\alpha_i sen\theta_i & a_i\cos\theta_i \\ sen\theta_i & -\cos\alpha_i\cos\theta_i & -sen\alpha_i\cos\theta_i & a_i sen\theta_i \\ 0 & sen\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad [2.2] $$

Sustituyendo por los valores correspondientes en cada articulación, las matrices de transformación de eslabón quedan del siguiente modo:

$$ {}^{0}A_1 = \begin{bmatrix} \cos\theta_1 & 0 & -sen\theta_1 & a_1\cdot\cos\theta_1 \\ sen\theta_1 & 0 & \cos\theta_1 & a_1\cdot sen\theta_1 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad [2.5] $$

con a_1=16 y d_1=345

$$ {}^{1}A_2 = \begin{bmatrix} \cos\theta_2 & -sen\theta_2 & 0 & a_2\cdot\cos\theta_2 \\ sen\theta_2 & \cos\theta_2 & 0 & a_2\cdot sen\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad [2.6] $$

con a_2=220

$$ {}^{2}A_3 = \begin{bmatrix} \cos\theta_3 & -sen\theta_3 & 0 & a_3\cdot\cos\theta_3 \\ sen\theta_3 & \cos\theta_3 & 0 & a_3\cdot sen\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad [2.7] $$

con a_3=220

$$ {}^{3}A_4 = \begin{bmatrix} \cos q_4 & 0 & sen q_4 & 0 \\ sen q_4 & 0 & -\cos q_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad [2.8] $$

$$ {}^{4}A_5 = \begin{bmatrix} \cos q_5 & -sen q_5 & 0 & 0 \\ sen q_5 & \cos q_5 & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad [2.9] $$

con d_5=140

Una vez calculadas las matrices homogéneas de transformación de eslabón, la matriz homogénea de transformación del sistema 5 al sistema 0, vendrá expresada de la siguiente manera:

$$ {}^{0}T_5 = {}^{0}A_1 \cdot {}^{1}A_2 \cdot {}^{2}A_3 \cdot {}^{3}A_4 \cdot {}^{4}A_5 \qquad [2.10] $$

En adelante, y por razones de espacio, la notación de senos y cosenos quedará expresada de la siguiente forma:

$$ s_{i\ldots k} = \text{sen}(\theta_i + \ldots + \theta_k) $$
$$ c_{i\ldots k} = \cos(\theta_i + \ldots + \theta_k) $$

$$ {}^{0}T_5 = \begin{bmatrix} (c_1 c_{23} c_4 - c_1 s_{23} s_4) c_5 - s_1 s_5 & -(c_1 c_{23} c_4 - c_1 s_{23} s_4) s_5 - s_1 c_5 \\ (s_1 c_{23} c_4 - s_1 s_{23} s_4) c_5 - c_1 s_5 & -(s_1 c_{23} c_4 - s_1 s_{23} s_4) s_5 - c_1 c_5 \\ -s_{234} c_5 & s_{234} c_5 \\ 0 & 0 \end{bmatrix} $$

$$ \begin{bmatrix} c_1 c_{23} c_4 + c_1 s_{23} c_4 & (c_1 c_{23} s_4 + c_1 s_{23} c_4) d_5 + c_1 a_3 c_{23} + c_1 (a_2 c_2 + a_1) \\ s_1 c_{23} s_4 + s_1 s_{23} c_4 & (s_1 c_{23} s_4 + s_1 s_{23} c_4) d_5 + s_1 a_3 c_{23} + s_1 (a_2 c_2 + a_1) \\ c_{234} & c_{234} d_5 - a_3 s_{23} - a_2 s_2 + d_1 \\ 0 & 1 \end{bmatrix} $$

$$ {}^{0}T_{5} = \begin{bmatrix} c_1c_{234}c_5 - s_1s_5 & -c_1c_{234}s_5 - s_1c_5 & c_1s_{234} & d_5c_1s_{234} + c_1(a_3c_{23} + a_2c_2 + a_1) \\ s_1c_{234}c_5 + c_1s_5 & -s_1c_{234}s_5 + c_1c_5 & s_1s_{234} & d_5s_1s_{234} + s_1(a_3c_{23} + a_2c_2 + a_1) \\ -s_{234}c_5 & s_{234}s_5 & c_{234} & d_5c_{234} - a_3s_{23} - a_2s_2 + d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [2.11] $$

En adelante, la notación usada para las matrices de transformación homogéneas será la siguiente:

$$ {}^{i-1}A_i = A_i $$

La matriz de transformación homogénea se pude descomponer en el producto de dos matrices. Esta matrices serán las matrices de transformación homogéneas del brazo y de la mano, respectivamente.

$$ T = A_1A_2A_3A_4A_5 \quad [2.12] $$

$$ T_b = A_1A_2A_3 \quad [2.13] $$

$$ T_m = A_4A_5 \quad [2.14] $$

$$ T = T_bT_m \quad [2.15] $$

$$ T_b = \begin{bmatrix} c_1c_2c_3 - c_1s_2s_3 & -c_1c_2s_3 - c_1s_2c_3 & -s_1 & a_3c_1c_2c_3 - a_3c_1s_2s_3 + a_2c_1c_2 + a_1c_1 \\ s_1c_2c_3 - s_1s_2s_3 & -s_1c_2s_3 - s_1s_2c_3 & c_1 & a_3s_1c_2c_3 - a_3s_1s_2s_3 + a_2s_1c_2 + a_1s_1 \\ -s_2c_3 - c_2s_3 & s_2s_3 - c_2c_3 & 0 & -a_3s_2c_3 - a_3c_2s_3 - a_2s_2 + d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [2.16] $$

$$ T_m = \begin{bmatrix} c_4c_5 & -c_4s_5 & s_4 & s_4d_5 \\ s_4c_5 & -s_4s_5 & -c_4 & -c_4d_5 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [2.17] $$

2.1.4.- Cinemática inversa

La cinemática inversa permite calcular las coordenadas articulares (θ_1,θ_2,θ_3,θ_4,θ_5) a partir de la posición y orientación del brazo mecánico.

Al ser la matriz de tarea ${}^0T_n$ una función de productos de senos y cosenos de las variables θ_i, no existe una solución cerrada para obtener la posición y orientación. Existen diversos métodos para conseguir las ecuaciones cinemáticas inversas, siendo los más comunes el método geométrico y el analítico.

La inversa de una matriz de transformación homogénea tiene la siguiente forma:

$$\begin{bmatrix} u_x & o_x & a_x & p_x \\ u_y & o_y & a_y & p_y \\ u_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} u_x & u_y & u_z & -\overline{n}^T\overline{p} \\ o_x & o_y & o_z & -\overline{o}^T\overline{p} \\ a_x & a_y & a_z & -\overline{a}^T\overline{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad [2.18]$$

Por tanto, las inversas de las matrices de transformación homogéneas de eslabón serán:

$$A_1^{-1} = \begin{bmatrix} c_1 & 0 & -s_1 & a_1c_1 \\ s_1 & 0 & c_1 & a_1s_1 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} c_1 & s_1 & 0 & -a_1 \\ 0 & 0 & -1 & d_1 \\ -s_1 & c_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad [2.19]$$

$$A_2^{-1} = \begin{bmatrix} c_2 & -s_2 & 0 & a_2c_2 \\ s_2 & c_2 & 0 & a_2s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} c_2 & s_2 & 0 & -a_2 \\ -s_2 & c_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad [2.20]$$

$$A_3^{-1} = \begin{bmatrix} c_3 & -s_3 & 0 & a_3 c_3 \\ s_3 & c_3 & 0 & a_3 s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} c_3 & s_3 & 0 & -a_3 \\ -s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [2.21]$$

$$A_4^{-1} = \begin{bmatrix} c_4 & 0 & s_4 & 0 \\ s_4 & 0 & -c_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} c_4 & s_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ s_4 & -c_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [2.22]$$

$$A_5^{-1} = \begin{bmatrix} c5 & -s_5 & 0 & 0 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} c_5 & s_5 & 0 & 0 \\ -s_5 & c_5 & 0 & 0 \\ 0 & 0 & 1 & -d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [2.23]$$

2.1.4.1.- Determinación del parámetro q_1

Para obtener el parámetro θ_1 se utilizará el método analítico de partición de matrices. A partir de la relación [2.13] se tiene que:

$$A_1^{-1} T_b = A_2 A_3 \qquad [2.24]$$

$$\begin{bmatrix} c_1 & s_1 & 0 & -a_1 \\ 0 & 0 & -1 & d_1 \\ -s_1 & c_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x & o_x & a_x & p_{bx} \\ u_y & o_y & a_y & p_{by} \\ u_z & o_z & a_z & p_{bz} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_2 & -s_2 & 0 & a_2 c_2 \\ s_2 & c_2 & 0 & a_2 s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_3 & -s_3 & 0 & a_3 c_3 \\ s_3 & c_3 & 0 & a_3 s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \cdot & \cdot & \cdot & c_1 p_{bx} + s_1 p_{by} - a_1 \\ \cdot & \cdot & \cdot & -p_{bz} + d_1 \\ \cdot & \cdot & \cdot & -s_1 p_{bx} + c_1 p_{by} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cdot & \cdot & \cdot & a_2 c_2 + a_3 c_2 c_3 - a_3 s_2 s_3 \\ \cdot & \cdot & \cdot & a_2 s_2 + a_3 s_2 c_3 + a_3 c_2 s_3 \\ \cdot & \cdot & \cdot & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Fijándonos en el elemento (3,4) de ambas matrices, se obtiene la siguiente relación:

$$-sen\boldsymbol{q}_1 p_{bx} + \cos\boldsymbol{q}_1 p_{by} = 0$$

$$sen\boldsymbol{q}_1 p_{bx} = \cos\boldsymbol{q}_1 p_{by}$$

$$tg\boldsymbol{q}_1 = \frac{p_{by}}{p_{bx}}$$

$$\boxed{\boldsymbol{q}_1 = arctg\frac{p_{by}}{p_{bx}}} \qquad [2.25]$$

2.1.4.2.- Determinación del parámetro q_3

La determinación del parámetro θ_3 se realizará geométricamente a partir de la Figura 2.3.

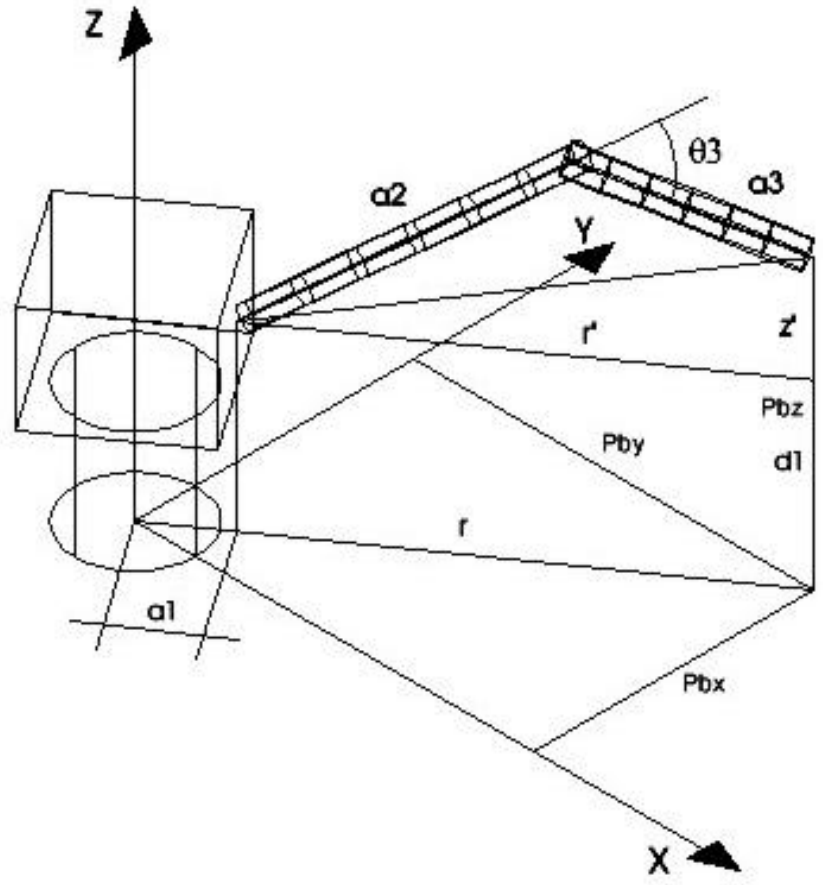

Figura 2.3.Componentes para la determinación de θ_3.

Observando la Figura 2.3 se deduce que:

$$r = a_1 + r' \quad \Rightarrow \quad r' = r - a_1$$

$$p_{bz} = d_1 + z' \quad \Rightarrow \quad z' = p_{bz} - d_1$$

$$r^2 = p_{bx}{}^2 + p_{by}{}^2 \quad \Rightarrow \quad r = \sqrt{p_{bx}{}^2 + p_{by}{}^2}$$

Aplicando el teorema del coseno (ángulo suplementario):

$$r'^2 + z'^2 = a_2{}^2 + a_3{}^2 + 2a_2 a_3 \cos\theta_3$$

$$\cos\theta_3 = \frac{r'^2 + z'^2 - a_2{}^2 - a_3{}^2}{2a_2 a_3}$$

$$\cos\theta_3 = \frac{(r - a_1)^2 + (p_{bz} - d_1)^2 - a_2{}^2 - a_3{}^2}{2a_2 a_3}$$

$$\cos\theta_3 = \frac{r^2 + a_1{}^2 - 2a_1 r + p_{bz}{}^2 + d_1{}^2 - 2p_{bz} d_1 - a_2{}^2 - a_3{}^2}{2a_2 a_3}$$

$$\cos\theta_3 = \frac{p_{bx}{}^2 + p_{by}{}^2 + a_1{}^2 - 2a_1\sqrt{p_{bx}{}^2 + p_{by}{}^2} + p_{bz}{}^2 + d_1{}^2 - 2p_{bz} d_1 - a_2{}^2 - a_3{}^2}{2a_2 a_3}$$

$$\boxed{\cos\theta_3 = \frac{p_{bx}{}^2 + p_{by}{}^2 + p_{bz}{}^2 + a_1{}^2 + d_1{}^2 - 2a_1\sqrt{p_{bx}{}^2 + p_{by}{}^2} - 2p_{bz} d_1 - a_2{}^2 - a_3{}^2}{2a_2 a_3}} \quad [2.26]$$

Por otra parte:

$$\sin^2\theta_3 + \cos^2\theta_3 = 1 \quad \Rightarrow \quad sen\theta_3 = \pm\sqrt{1 - \cos^2\theta_3}$$

$$\boxed{\theta_3 = arctg\frac{\pm\sqrt{1 - \cos^2\theta_3}}{\cos\theta_3}} \quad [2.27]$$

signo > 0 $\Rightarrow$ codo arriba

signo < 0 $\Rightarrow$ codo abajo

2.1.4.3.- Determinación del parámetro q_2

Antes de poder calcular este parámetro, ha de haberse calculado el parámetro θ_3. Para calcular θ_2 se utilizará el procedimiento analítico de partición de matrices.

En este caso, a partir de la ecuación [2.13] se obtiene:

$$A_2^{-1} A_1^{-1} T_b = A_3 \qquad [2.28]$$

$$\begin{bmatrix} c_2 & s_2 & 0 & -a_2 \\ -s_2 & c_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 & s_1 & 0 & -a_1 \\ 0 & 0 & -1 & d_1 \\ -s_1 & c_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} n_x & o_x & a_x & p_{bx} \\ n_y & o_y & a_y & p_{by} \\ n_z & o_z & a_z & p_{bz} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_3 & -s_3 & 0 & a_3 c_3 \\ s_3 & c_3 & 0 & a_3 s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} c_1 c_2 & s_1 c_2 & -s_2 & -a_1 c_2 + s_2 d_1 - a_2 \\ -c_1 s_2 & -s_1 s_2 & -c_2 & a_1 s_2 + c_2 d_1 \\ -s_1 & c_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} n_x & o_x & a_x & p_{bx} \\ n_y & o_y & a_y & p_{by} \\ n_z & o_z & a_z & p_{bz} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_3 & -s_3 & 0 & a_3 c_3 \\ s_3 & c_3 & 0 & a_3 s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \cdot & \cdot & \cdot & c_1 c_2 p_{bx} + s_1 c_2 p_{by} - s_2 p_{bz} - a_1 c_2 + s_2 d_1 - a_2 \\ \cdot & \cdot & \cdot & -c_1 s_2 p_{bx} - s_1 s_2 p_{by} - c_2 p_{bz} - a_1 s_2 + c_2 d_1 \\ \cdot & \cdot & \cdot & -s_1 p_{bx} + c_1 p_{by} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cdot & \cdot & \cdot & a_3 c_3 \\ \cdot & \cdot & \cdot & a_3 s_3 \\ \cdot & \cdot & \cdot & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Cogiendo los elementos (1,4), (2,4) y (3,4) de ambas matrices, se establecen las siguientes ecuaciones:

$$c_1 c_2 p_{bx} + s_1 c_2 p_{by} - s_2 p_{bz} - a_1 c_2 + s_2 d_1 - a_2 = a_3 c_3 \qquad [2.29]$$

$$-c_1 s_2 p_{bx} - s_1 s_2 p_{by} - c_2 p_{bz} + a_1 s_2 + c_2 d_1 = a_3 s_3 \qquad [2.30]$$

$$-s_1 p_{bx} + c_1 p_{by} = 0 \qquad [2.31]$$

Manipulando la ecuación [2.31] se obtiene que:

$$-s_1 p_{b_x} + c_1 p_{b_y} = 0$$

$$\left(c_1 p_{b_y} - s_1 p_{b_x}\right)^2 = c_1^{\,2} p_{b_y}^{\,2} + s_1^{\,2} p_{b_x}^{\,2} - 2 s_1 c_1 p_{b_x} p_{b_y} = 0$$

$$c_1^{\,2} p_{b_y}^{\,2} + s_1^{\,2} p_{b_x}^{\,2} = 2 s_1 c_1 p_{b_x} p_{b_y}$$

$$\left(1 - s_1^{\,2}\right) p_{b_y}^{\,2} + \left(1 - c_1^{\,2}\right) p_{b_x}^{\,2} = 2 s_1 c_1 p_{b_x} p_{b_y}$$

$$p_{b_y}^{\,2} - s_1^{\,2} p_{b_y}^{\,2} + p_{b_x}^{\,2} - c_1^{\,2} p_{b_x}^{\,2} = 2 s_1 c_1 p_{b_x} p_{b_y}$$

$$p_{b_x}^{\,2} + p_{b_y}^{\,2} = 2 s_1 c_1 p_{b_x} p_{b_y} + s_1^{\,2} p_{b_y}^{\,2} + c_1^{\,2} p_{b_x}^{\,2}$$

$$p_{b_x}^{\,2} + p_{b_y}^{\,2} = \left(p_{b_x} c_1 + p_{b_y} s_1\right)^2$$

$$\sqrt{p_{b_x}^{\,2} + p_{b_y}^{\,2}} = p_{b_x} c_1 + p_{b_y} s_1 \qquad [2.32]$$

Ahora se sustituye [2.31] en [2.29] y [2.30]:

$$\begin{cases} c_1 c_2 p_{b_x} + s_1 c_2 p_{b_y} - s_2 p_{b_z} - a_1 c_2 + s_2 d_1 - a_2 = a_3 c_3 \\ -c_1 s_2 p_{b_x} - s_1 s_2 p_{b_y} - c_2 p_{b_z} + a_1 s_2 + c_2 d_1 = a_3 s_3 \end{cases}$$

$$\begin{cases} c_2 \sqrt{p_{b_x}^{\,2} + p_{b_y}^{\,2}} - s_2 p_{b_z} - a_1 c_2 + s_2 d_1 - a_2 = a_3 c_3 \\ -s_2 \sqrt{p_{b_x}^{\,2} + p_{b_y}^{\,2}} - c_2 p_{b_z} + a_1 s_2 + c_2 d_1 = a_3 s_3 \end{cases}$$

$$\begin{cases} c_2\sqrt{{p_{bx}}^2+{p_{by}}^2}-s_2\left(p_{bz}-d_1\right)-a_1c_2-a_2=a_3c_3 \\ -s_2\sqrt{{p_{bx}}^2+{p_{by}}^2}-c_2\left(p_{bz}-d_1\right)+a_1s_2=a_3s_3 \end{cases}$$

$$\begin{cases} c_2\left(\sqrt{{p_{bx}}^2+{p_{by}}^2}-a_1\right)-s_2\left(p_{bz}-d_1\right)-a_2=a_3c_3 \\ -s_2\left(\sqrt{{p_{bx}}^2+{p_{by}}^2}-a_1\right)-c_2\left(p_{bz}-d_1\right)=a_3s_3 \end{cases}$$

Para mayor comodidad en el manejo de las ecuaciones, se efectúan los siguientes cambios de variables:

$$k_1=\sqrt{{p_{bx}}^2+{p_{by}}^2}-a_1$$

$$k_2=p_{bz}-d_1$$

Aplicando estos cambios de variable en las dos últimas ecuaciones:

$$k_1\cos\boldsymbol{q}_2-k_2 sen\boldsymbol{q}_2=a_3\cos\boldsymbol{q}_3+a_2 \qquad [2.33]$$

$$-k_1 sen\boldsymbol{q}_2-k_2\cos\boldsymbol{q}_2=a_3 sen\boldsymbol{q}_3 \qquad [2.34]$$

Manipulando adecuadamente estas dos ecuaciones, se puede obtener sen(θ_2) y cos(θ_2) en función de k1, k2, sen(θ_3) y cos(θ_3). Al estar θ_3 calculado previamente, todas estas variables tienen valores conocidos, con lo cual quedan determinados el sen(θ_2) y el cos(θ_2). Para obtener θ_2 sólo hay que aplicar el arcotangente.

Para obtener el sen(θ_2) se realizará la siguiente operación:

$$[2.33]\cdot k_2+[2.34]\cdot k_1$$

$$k_1 k_2 \cos\theta_2 - k_2^{\,2} sen\theta_2 = (a_3 \cos\theta_3 + a_2)k_2$$

$$-k_1^{\,2} sen\theta_2 - k_1 k_2 \cos\theta_2 = a_3 sen\theta_3 k_1$$

$$-(k_1^{\,2} + k_2^{\,2}) sen\theta_2 = k_1 a_3 sen\theta_3 + k_2 a_3 \cos\theta_3 + k_2 a_2$$

Despejando:

$$\boxed{sen\theta_2 = -\frac{k_1 a_3 sen\theta_3 + k_2 a_3 \cos\theta_3 + k_2 a_2}{k_1^{\,2} + k_2^{\,2}}} \qquad [2.35]$$

Análogamente, para obtener cos(θ_2) se realizará la siguiente operación:

$$[2.33]\cdot k_1 - [2.34]\cdot k_2$$

$$k_1^{\,2} \cos\theta_2 - k_1 k_2 sen\theta_2 = (a_3 \cos\theta_3 + a_2)k_1$$

$$k_1 k_2 sen\theta_2 + k_2^{\,2} \cos\theta_2 = -k_2 a_3 sen\theta_3$$

$$k_1^{\,2} \cos\theta_2 + k_2^{\,2} \cos\theta_2 = k_1 a_3 \cos\theta_3 + k_1 a_2 - k_2 a_3 sen\theta_3$$

Despejando:

$$\boxed{\cos\theta_2 = \frac{k_1 a_3 \cos\theta_3 + k_1 a_2 - k_2 a_3 sen\theta_3}{k_1^{\,2} + k_2^{\,2}}} \qquad [2.36]$$

Con [2.35] y [2.36] se obtiene θ_2.

$$tg\theta_2 = \frac{sen\theta_2}{\cos\theta_2}$$

$$\boxed{\theta_2 = arctg\frac{sen\theta_2}{\cos\theta_2}} \qquad [2.37]$$

2.1.4.4.- Determinación de los parámetros q_4 y q_5

Los parámetros θ_4 y θ_5 se determinarán con el método analítico. Para ello, se empleará la submatriz de rotación y la siguiente relación:

$$ {}^0R_5 = {}^0R_3 \cdot {}^3R_5 \qquad [2.38] $$

Y puesto que en una matriz de rotación se cumple que $R^{-1} = R^T$, se puede llegar fácilmente a la siguiente relación:

$$ \left({}^0R_3\right)^{-1} \cdot {}^0R_5 = {}^3R_5 \qquad [2.39] $$

La submatriz de rotación de ${}^3R_5$ se obtiene de la ecuación [2.14]:

$$ T_m = A_4 A_5 = \begin{bmatrix} \cos q_4 & 0 & sen\, q_4 & 0 \\ sen\, q_4 & 0 & -\cos q_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos q_5 & -sen\, q_5 & 0 & 0 \\ sen\, q_5 & \cos q_5 & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ T_m = \begin{bmatrix} c_4 c_5 & -c_4 s_5 & s_4 & s_4 d_5 \\ s_4 c_5 & -s_4 s_5 & -c_4 & -c_4 d_5 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Donde,

$$ {}^3R_5 = \begin{bmatrix} c_4 c_5 & -c_4 s_5 & s_4 \\ s_4 c_5 & -s_4 s_5 & -c_4 \\ s_5 & c_5 & 0 \end{bmatrix} \qquad [2.40] $$

Por otro lado, a partir de la ecuación [2.13] se obtiene ${}^0R_3$:

$$T_b = \begin{bmatrix} c_1c_2c_3 - c_1s_2s_3 & -c_1c_2s_3 - c_1s_2c_3 & -s_1 & a_3c_1c_2c_3 - a_3c_1s_2s_3 + a_2c_1c_2 + a_1c_1 \\ s_1c_2c_3 - s_1s_2s_3 & -s_1c_2s_3 - s_1s_2c_3 & c_1 & a_3s_1c_2c_3 - a_3s_1s_2s_3 + a_2s_1c_2 + a_1s_1 \\ -s_2c_3 - c_2s_3 & s_2s_3 - c_2c_3 & 0 & -a_3s_2c_3 - a_3c_2s_3 - a_2s_2 + d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^0R_3 = \begin{bmatrix} c_1c_2c_3 - c_1s_2s_3 & -c_1c_2s_3 - c_1s_2c_3 & -s_1 \\ s_1c_2c_3 - s_1s_2s_3 & -s_1c_2s_3 - s_1s_2c_3 & c_1 \\ -s_2c_3 - c_2s_3 & s_2s_3 - c_2c_3 & 0 \end{bmatrix} \qquad [2.41]$$

Invirtiendo esta matriz:

$$\left({}^0R_3\right)^{-1} = \begin{bmatrix} c_1c_2c_3 - c_1s_2s_3 & s_1c_2c_3 - s_1s_2s_3 & -s_2c_3 - c_2s_3 \\ -c_1c_2s_3 - c_1s_2c_3 & -s_1c_2s_3 - s_1s_2c_3 & s_2s_3 - c_2c_3 \\ -s_1 & c_1 & 0 \end{bmatrix} \qquad [2.42]$$

Sustituyendo en la ecuación [2.39]:

$$\left({}^0R_3\right)^{-1} \cdot {}^0R_5 = {}^3R_5$$

$$\begin{bmatrix} c_1c_2c_3 - c_1s_2s_3 & s_1c_2c_3 - s_1s_2s_3 & -s_2c_3 - c_2s_3 \\ -c_1c_2s_3 - c_1s_2c_3 & -s_1c_2s_3 - s_1s_2c_3 & s_2s_3 - c_2c_3 \\ -s_1 & c_1 & 0 \end{bmatrix} \begin{bmatrix} u_x & o_x & a_x \\ u_y & o_y & a_y \\ u_z & o_z & a_z \end{bmatrix} = \begin{bmatrix} c_4c_5 & -c_4s_5 & s_4 \\ s_4c_5 & -s_4s_5 & -c_4 \\ s_5 & c_5 & 0 \end{bmatrix}$$

Observando la matriz ${}^3R_5$ se puede ver que a partir de los elementos $R_{(1,3)}$ y $R_{(2,3)}$ se obtiene el parámetro θ_4:

$$tg\boldsymbol{q}_4 = -\frac{R_{(1,3)}}{R_{(2,3)}}$$

$$q_4 = arctg\left(-\frac{R_{(1,3)}}{R_{(2,3)}}\right) \qquad [2.43]$$

A partir de lado izquierdo de la igualdad de la ecuación [2.39], se puede deducir que los elementos $R_{(1,3)}$ y $R_{(2,3)}$ son equivalentes a los siguientes valores:

$$R_{(1,3)} = (c_1c_2c_3 - c_1s_2s_3)a_x + (s_1c_2c_3 - s_1s_2s_3)a_y + (-s_2c_3 - c_2s_3)a_z$$

$$R_{(2,3)} = (-c_1c_2s_3 - c_1s_2c_3)a_x + (-s_1c_2s_3 - s_1s_2c_3)a_y + (s_2s_3 - c_2c_3)a_z$$

y simplificando,

$$R_{(1,3)} = c_1c_{23}a_x + s_1c_{23}a_y - s_{23}a_z \qquad [2.44]$$

$$R_{(2,3)} = -c_1s_{23}a_x - s_1s_{23}a_y - c_{23}a_z \qquad [2.45]$$

Sustituyendo en [2.43],

$$\boxed{q_4 = arctg\left(-\frac{c_1c_{23}a_x + s_1c_{23}a_y - s_{23}a_z}{-c_1s_{23}a_x - s_1s_{23}a_y - c_{23}a_z}\right)} \qquad [2.46]$$

Operando análogamente con los elementos $R_{(3,1)}$ y $R_{(3,2)}$ se obtiene θ_5:

$$tgq_5 = \frac{R_{(3,1)}}{R_{(3,2)}}$$

$$q_5 = arctg\left(\frac{R_{(3,1)}}{R_{(3,2)}}\right) \qquad [2.47]$$

donde,

$$R_{(3,1)} = -s_1 u_x + c_1 u_y \qquad [2.48]$$

$$R_{(3,2)} = -s_1 o_x + c_1 o_y \qquad [2.49]$$

Sustituyendo en [2.47],

$$\boxed{q_5 = arctg\left(\frac{c_1 u_y - s_1 u_x}{c_1 o_y - s_1 o_x}\right)} \qquad [2.50]$$

2.1.4.5.- Determinación de las coordenadas de P_b (P_{bx}, P_{by}, P_{bz})

Para calcular las coordenadas de Pb, debe tenerse en cuenta que el vector de posición es el vector que se refiere al punto final del extremo de la herramienta de trabajo partiendo del sistema de coordenadas {0}, tal como muestra la Figura 2.4.

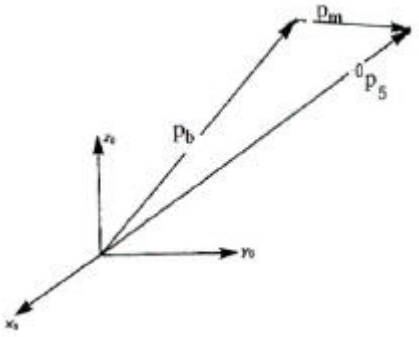

Figura 2.4. Componentes del vector ${}^0\overline{p}_5$.

Este vector de posición ${}^0\overline{p}_5$ tiene dos componentes, ${}^0\overline{p}_3 = \overline{p}_b$ (brazo) y ${}^3\overline{p}_5 = \overline{p}_m$ (mano o herramienta de trabajo), donde ${}^0\overline{p}_3$ es el vector desde el origen de la base al extremo del tercer eslabón (sin incluir la muñeca) y ${}^3\overline{p}_5$ es el vector desde ese punto al extremo de la muñeca , de modo que

$$ {}^0\overline{p}_5 = {}^0\overline{p}_3 + {}^3\overline{p}_5 \qquad [2.51]$$

Como puede apreciarse en la Figura 2.4, las coordenadas de Pb vienen determinadas por la posición del extremo de la herramienta y la orientación de ésta en el espacio.

De estos tres vectores, sólo ${}^0\overline{p}_3 = \overline{p}_b$ es desconocido, ya que, observando la Figura 2.5, el vector ${}^3\overline{p}_5 = \overline{p}_m$ puede expresarse en función de su módulo y de los ángulos β y ϕ.

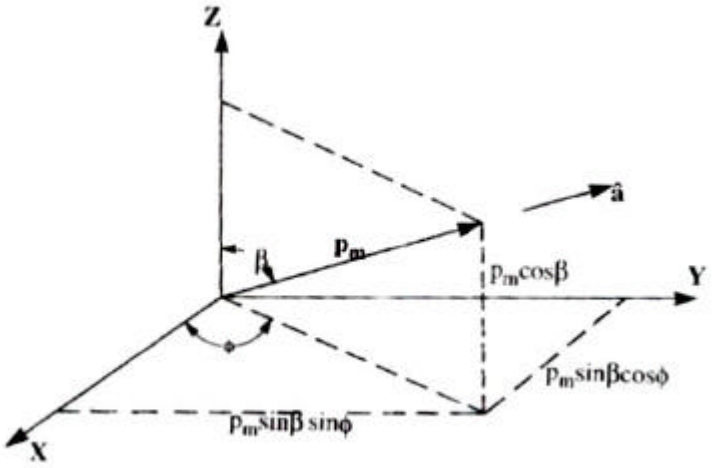

Figura 2.5. Vector ${}^3\overline{p}_5 = \overline{p}_m$ y sus componentes x, y, z en función de $\overline{a}$.

Nótese que el vector unitario $\overline{a}$ a, por definición, está siempre en la dirección de $\overline{p}_m$, de manera que

$$ \boxed{f = arctg\left(\frac{a_y}{a_x}\right)} \qquad [2.52]$$

$$sen\boldsymbol{b} = \frac{\sqrt{a_x{}^2 + a_y{}^2}}{a}$$

$$\cos \boldsymbol{b} = \frac{a_z}{a}$$

$$\boxed{\boldsymbol{b} = arctg\left(\frac{\sqrt{a_x{}^2 + a_y{}^2}}{a_z}\right)} \qquad [2.53]$$

Donde a_x, a_y y a_z son ya conocidas por la matriz de tarea, lo mismo que el vector ${}^0\overline{p}_5$,

$${}^0T_5 = \begin{bmatrix} u_x & o_x & a_x & p_x \\ u_y & o_y & a_y & p_y \\ u_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad [2.54]$$

El vector ${}^3\overline{p}_5 = \overline{p}_m$ referido al sistema de la base se obtiene de

$${}^3p_5 = \begin{bmatrix} p_m \cdot sen\boldsymbol{b} \cdot \cos \boldsymbol{f} \\ p_m \cdot sen\boldsymbol{b} \cdot sen\boldsymbol{f} \\ p_m \cdot \cos \boldsymbol{b} \end{bmatrix} \qquad [2.55]$$

A partir de la ecuación [2.51] se obtienen las componentes de ${}^0\overline{p}_3 = \overline{p}_b$ en las direcciones x,y,z:

$$\boxed{p_{bx} = p_x - p_m \cdot sen\boldsymbol{b} \cdot \cos \boldsymbol{f}} \qquad [2.56]$$

$$\boxed{p_{by} = p_y - p_m \cdot sen\boldsymbol{b} \cdot sen\boldsymbol{f}} \qquad [2.57]$$

$$\boxed{p_{bz} = p_z - p_m \cdot \cos \boldsymbol{b}} \qquad [2.58]$$

2.2.- Matriz de rotación sobre el eje Z

En el desarrollo del programa, es necesario realizar rotaciones de los antiguos vectores de acercamiento $\overline{a}$ y orientación $\overline{o}$ alrededor del eje Z para poder obtener los nuevos vectores de acercamiento $\overline{a}'$ y orientación $\overline{o}'$.

La matriz de rotación que permite girar un ángulo α sobre el eje Z es la siguiente:

$$R(z,\alpha)=\begin{bmatrix}\cos\alpha & -sen\alpha & 0\\ sen\alpha & \cos\alpha & 0\\ 0 & 0 & 1\end{bmatrix} \qquad [2.59]$$

2.3.- Matriz de orientación

Para la representación de la orientación, se ha decidido utilizar los ángulos de Euler, ya que permiten representar la orientación de un sistema de coordenadas respecto a otro sistema fijo mediante tres ángulos: ϕ, θ, ψ. Además de conocer estos ángulos, es necesario conocer el orden en que se efectúan los giros para llevar un sistema hasta el otro. Existen diversas posibilidades según este orden. La más utilizada en robótica son los ángulos de yaw, pitch y roll (guiñada, cabeceo y alabeo). Estos ángulos quedan definidos como se muestra en la Figura 2.6.

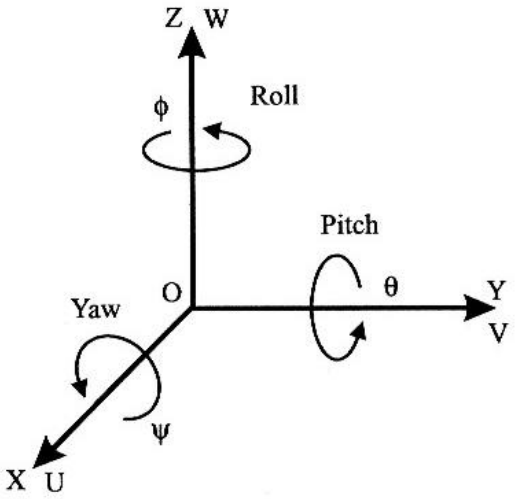

Figura 2.6. Ángulos de Euler: yaw, pitch y roll.

Teniendo en cuanta que la multiplicación de matrices no es conmutativa, la secuencia a seguir para obtener la coincidencia de los dos sistemas de coordenadas, debe realizarse en el siguiente orden:

1. Girar el sistema OUVW un ángulo ψ con respecto al eje OX. Es el denominado Yaw o guiñada.
2. Girar el sistema OUVW un ángulo θ con respecto al eje OY. Es el denominado Pitch o cabeceo.

3. Girar el sistema OUVW un ángulo ϕ con respecto al eje OZ. Es el denominado Roll o alabeo.

Siguiendo este orden, la matriz de rotación que expresa la aplicación continua de estas tres rotaciones es:

$$R = R(z,\varphi)R(y,\theta)R(x,\psi) = \begin{bmatrix} \cos\varphi & -sen\varphi & 0 \\ sen\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & sen\theta \\ 0 & 1 & 0 \\ -sen\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -sen\psi \\ 0 & sen\psi & \cos\psi \end{bmatrix}$$

$$R = \begin{bmatrix} \cos\varphi\cdot\cos\theta & -sen\varphi\cdot\cos\psi + \cos\varphi\cdot sen\theta\cdot sen\psi & sen\varphi\cdot sen\psi + \cos\varphi\cdot sen\theta\cdot\cos\psi \\ sen\varphi\cdot\cos\theta & \cos\varphi\cos\psi + sen\varphi sen\theta sen\psi & -\cos\varphi\cdot sen\psi + sen\varphi\cdot sen\theta\cdot\cos\psi \\ -sen\theta & \cos\theta\cdot sen\psi & \cos\theta\cdot\cos\psi \end{bmatrix}$$

[2.60]

Para conseguir la nueva orientación de la herramienta (nuevos vectores acercamiento $\bar{a}$ y orientación $\bar{o}$) basta con multiplicar los vectores de orientación $\bar{a}$ y $\bar{o}$, por la matriz de rotación R, habiendo sustituido previamente los ángulos de yaw, pitch y roll por el valor deseado.

El ESCORBOT ER-III sólo tiene 5 grados de libertad, ya que la mano únicamente permite el movimiento de pitch y roll. Por ello, en los cálculos el ángulo ψ siempre tomará el valor 0.

2.4.- Resolución de las transmisiones

Para calcular la resolución del movimiento de las articulaciones se deben tener en cuenta los siguientes aspectos:

- Tipos de transmisiones usadas.
- Relación de transmisión del motor. Es decir, relación de transmisión entre el eje del motor y el eje de salida, debida a los propios engranajes y reductores del motor.
- Sensibilidad del motor. Esta depende de la precisión del encoder.

Los encoders están montados sobre el propio eje del motor, tal como se puede ver en la Figura 2.7.

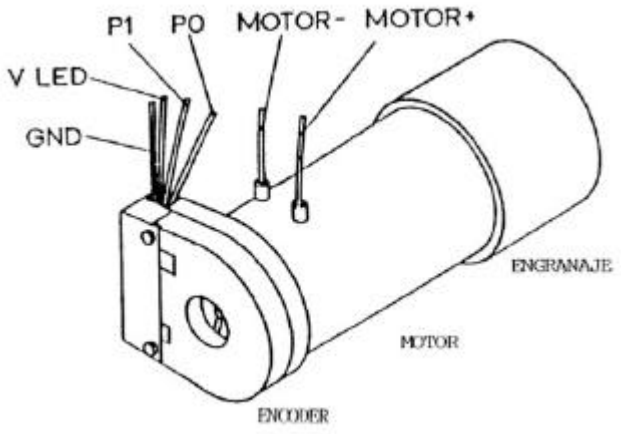

Figura 2.7. Imagen del motor con el encoder ensamblado.

Para todos los motores, excepto el de la pinza, los discos de los encoders tienen 6 pares de agujeros, con lo que la resolución es:

$$Sm_{1,2,3,4,5} = \frac{360^o}{6} = 60^o$$

Para el motor de la pinza, el disco del encoder tiene sólo 3 pares de agujeros, con lo que la resolución del motor es:

$$\boxed{Sm_8 = \frac{360^o}{3} = 120^o}$$

2.4.1.- Articulación de la base

La transmisión de la base del robot es un sistema de transmisión indirecto de una etapa.

Los datos de la transmisión son:

n_a = 24 dientes

n_b = 120 dientes

Los datos del motor son:

Tm_1 = 127.7 : 1

$Sm_1 = 60^o$

Con estos datos, se tiene que la sensibilidad en el eje de salida del motor es:

$$So_1 = Sm_1 \frac{1}{Tm_1} = 60^o \frac{1}{127.7} = 0.469851^o$$

La relación de transmisión del engranaje es:

$$T_{ab} = \frac{n_b}{n_a} = \frac{120}{24} = 5$$

Con lo que la resolución de la articulación de la base queda:

$$S_{base} = So_1 \frac{1}{T_{ab}} = 0.469851^o \frac{1}{5} = 0.093970^o$$

Resumiendo, se tiene que para 1 pulso del encoder la base gira 0.09397º. Entonces, para que la base gire en 1º serán necesarios

$$\frac{1}{0.09397} = 10.641694 \text{ pulsos del encoder}$$

2.4.2.- Articulación del hombro

La transmisión del hombro, al igual que la base, tiene un sistema de transmisión indirecto de una etapa.

Los datos de la transmisión son:

n_a = 18 dientes

n_b = 72 dientes

Los datos del motor son:

Tm_2 = 127.7 : 1

Sm_2 = 60º

Con estos datos, se tiene que la sensibilidad en el eje de salida del motor es:

$$So_2 = Sm_2 \frac{1}{Tm_2} = 60^o \frac{1}{127.7} = 0.469851^o$$

La relación de transmisión del engranaje es:

$$T_{ab} = \frac{n_b}{n_a} = \frac{72}{18} = 4$$

Con lo que la resolución de la articulación del hombro queda:

$$S_{\text{hombro}} = So_2 \frac{1}{T_{ab}} = 0.469851^o \frac{1}{4} = 0.117462^o$$

Resumiendo, se tiene que para 1 pulso del encoder el hombro gira 0.117462º. Entonces, para que el hombro gire en 1º serán necesarios

$$\frac{1}{0.117462} = 8.513337 \text{ pulsos del encoder}$$

2.4.3.- Articulación del codo

La transmisión del codo tiene un sistema de transmisión indirecto de dos etapas.

Los datos de la transmisión son:

n_a = 18 dientes

n_b = 72 dientes

n_c = 17 dientes

n_d = 17 dientes

Los datos del motor son:

$$Tm_3 = 127.7 : 1$$

$$Sm_3 = 60^o$$

Con estos datos, se tiene que la sensibilidad en el eje de salida del motor es:

$$So_3 = Sm_3 \frac{1}{Tm_3} = 60^o \frac{1}{127.7} = 0.469851^o$$

La relación de transmisión del engranaje es:

$$T_{ad} = \frac{n_b}{n_a}\frac{n_d}{n_c} = \frac{72}{18}\frac{17}{17} = 4$$

Con lo que la resolución de la articulación del codo queda:

$$\boxed{S_{codo} = So_3 \frac{1}{T_{ad}} = 0.469851^o \frac{1}{4} = 0.117462^o}$$

Resumiendo, se tiene que para 1 pulso del encoder el codo gira 0.117462º. Entonces, para que el codo gire en 1º serán necesarios

$$\frac{1}{0.117462} = 8.513337 \text{ pulsos del encoder}$$

2.4.4.- Articulación de la muñeca

La transmisión de la muñeca tiene un sistema de transmisión indirecto de tres etapas.

Los datos de la transmisión son:

n_a = 12 dientes

$n_b = n_c$ = 24 dientes

$n_d = n_e$ = 24 dientes

n_f = 24 dientes

Los datos del motor son:

Tm_4 = 65.5 : 1

$Sm_4 = 60^o$

Con estos datos, se tiene que la sensibilidad en el eje de salida del motor es:

$$So_4 = Sm_4 \frac{1}{Tm_4} = 60^o \frac{1}{65.5} = 0.916031^o$$

La relación de transmisión del engranaje es:

$$T_{af} = \frac{n_b}{n_a}\frac{n_d}{n_c}\frac{n_f}{n_e} = \frac{24}{12}\frac{24}{24}\frac{24}{24} = 2$$

Con lo que la resolución de la articulación de la muñeca queda:

$$\boxed{S_{muñeca} = So_4 \frac{1}{T_{af}} = 0.916031^o \frac{1}{2} = 0.458016^o}$$

Resumiendo, se tiene que para 1 pulso del encoder la muñeca gira 0.458016º. Entonces, para que la muñeca gire en 1º serán necesarios

$$\frac{1}{0.45816} = 2.183332 \text{ pulsos del encoder}$$

2.4.5.- Articulación de la pinza

La transmisión de la pinza es un sistema de transmisión por tornillo de avance.

Los datos del motor son:

$$Tm_8 = 19.5 : 1$$

$$Sm_8 = 120º$$

Con estos datos, se tiene que la sensibilidad en el eje de salida del motor es:

$$So_8 = Sm_8 \frac{1}{Tm_8} = 120º \frac{1}{19.5} = 6.153846º$$

Resumiendo, se tiene que para 1 pulso del encoder el motor de la pinza gira 6.153846º. Entonces, para que el motor gire en 1º serán necesarios

$$\frac{1}{6.153846} = 0.1625 \text{ pulsos del encoder}$$

2.4.6.- Resolución mínima del sistema

La resolución en el caso del movimiento por coordenadas articulares vendrá determinada por el número de pulsos enviados a los encoders de los motores. Puesto que el número mínimo de pulsos es de 10, las resoluciones quedan de la siguiente manera:

Articulación de la base

$$1\,pulso = 0.093970^{o} \quad \Rightarrow \quad \boxed{\text{Re}\, s_{\min} = 10\,pulsos \cdot \frac{0.093970^{o}}{1\,pulso} = 0.939700^{o}}$$

Articulación del hombro y del codo

$$1\,pulso = 0.117462^{o} \quad \Rightarrow \quad \boxed{\text{Re}\, s_{\min} = 10\,pulsos \cdot \frac{0.117462^{o}}{1\,pulso} = 1.174620^{o}}$$

Articulación de la muñeca (Pitch y Roll)

$$1\,pulso = 0.458016^{o} \quad \Rightarrow \quad \boxed{\text{Re}\, s_{\min} = \frac{10\,pulsos}{2} \cdot \frac{0.458016^{o}}{1\,pulso} = 2.290080^{o}}$$

Constantí, 24 de Enero de 2001

El ingeniero técnico industrial

Fdo. Joan Mercadé Papiol

3.- PRESUPUESTO

3.1.- Mediciones

3.1.1.- Material

Nº	Uds.	Descripción	Cantidad
01	ud.	Joystick standard para PC de una palanca y dos botones con conector tipo D15.	1
02	ud.	Conector hembra de 25 pines para puerto RS-232-C.	1
03	ud.	Conector macho de 25 pines para puerto RS-232-C.	1
04	m	Cable UTP categoría 5 para comunicaciones.	2

3.1.2.- Mano de obra

Nº	Uds.	Descripción	Cantidad
05	hora	Tiempo de análisis	15
06	hora	Tiempo de programación	40

3.2.- Cuadro de precios

3.2.1.- Material

Nº	Uds	Descripción	Precio	
01	ud.	Joystick standard para PC de una palanca y dos botones con conector tipo D15.	6,00	Seis euros
02	ud.	Conector hembra de 25 pines para puerto RS-232-C.	1,50	Un euro con cincuenta céntimos
03	ud.	Conector macho de 25 pines para puerto RS-232-C.	1,50	Un euro con cincuenta céntimos
04	m	Cable UTP categoría 5 para comunicaciones.	1,80	Un euro con ochenta céntimos

3.2.2.- Mano de obra

Nº	Uds.	Descripción	Precio	
05	hora	Tiempo de análisis	35,00	Treinta y cinco euros
06	hora	Tiempo de programación	25,00	Veinticinco euros

3.3.- Aplicación de precios

3.2.1.- Material

Nº	Uds	Descripción	Precio unitario	Total medidas	Total euros
01	Ud.	Joystick standard para PC de una palanca y dos botones con conector tipo D15.	6,00	1	6,00
02	Ud.	Conector hembra de 25 pines para puerto RS-232-C.	1,50	1	1,50
03	Ud.	Conector macho de 25 pines para puerto RS-232-C.	1,50	1	1,50
04	M	Cable UTP categoría 5 para comunicaciones.	1,80	2	3,60
TOTAL MATERIAL: 12,60 euros					

3.2.2.- Mano de obra

Nº	Uds.	Descripción	Precio unitario	Total medidas	Total euros
05	hora	Tiempo de análisis	35,00	15	525,00
06	hora	Tiempo de programación	25,00	40	1000,00
TOTAL MANO DE OBRA: 1525,00 euros					

3.4.- Resumen del presupuesto

Material.....12,60 euros

Mano de obra.....1525,00 euros

_____

Presupuesto de Ejecución de Material (P.E.M.).....1537,60 euros

Gastos generales (13% del P.E.M.).....199,89 euros

Beneficio industrial (6% del P.E.M.).....92.26 euros

Presupuesto de Ejecución por Contrato (P.E.C.).....1829,75 euros

I.V.A. (16%).....292,76 euros

Presupuesto de Licitación.....2122'51 euros

El presupuesto de licitación asciende a la cantidad de **DOS MIL CIENTO VEINTIDOS EUROS CON CINCUENTA Y UN CÉNTIMOS**.

Constantí, 24 de Enero de 2001

El ingeniero técnico industrial

Fdo. Joan Mercadé Papiol

4.- ANEXO.
CÓDIGO FUENTE

```
#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <graphics.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>

#ifdef __cplusplus
      #define __cppargs ...
#else
      #define __cppargs
#endif

/*************************** CONSTANTS ****************************/
/********* Parametres de la linea serie ***********/
#define LCR (base+3)    /*Line Control Register*/
#define LSB (base+0)    /*LSB del Baud Rate Generator*/
#define MSB (base+1)    /*MSB del Baud Rate Generetor*/
#define IIR (base+2)    /*Interrupt Identification Register*/
#define IER (base+1)    /*Interrupt Enable Register*/
#define RBR (base+0)    /*Reciver Buffer Register*/
#define THR (base+0)    /*Transmitter Holding Register*/
#define MSR (base+6)    /*Modem Status Register*/
#define MCR (base+4)    /*Modem Control Register*/
#define LSR (base+5)    /*Line Status Register*/
#define MASK_IRQ4 0xef  /*mascara per habilitar int. al COM1*/
#define MASK_IRQ3 0xf7  /*mascara per habilitar int. al COM2*/


/******** Paramentres del joystick ***************/
#define JPORT 0x201        /*adressa del port del joystick*/
#define CMAX 19999          /*valor maxim del contatge*/


#define MASK_IRQ0 0xFE   /*mascara per habilitar les interrupcions de
                                                        IRQ0*/
#define KTICS 3            /*temps de refresc*/
#define KTICS2 10    /*mig segon*/

#define BASE_8259 0x20     /*adressa base del controlador 8259*/
#define EOI 0x20           /*fi d'interrupcio*/
#define IRQ3 0x0b    /*adressa de IRQ3*/
#define IRQ4 0x0c    /*adressa de IRQ4*/
#define IRQ0 0x08    /*adressa de IRQ0*/


#define M 'M'     /*comanda per accionar un motor*/
#define V 'V'     /*comanda per ajustar la velocitat*/

#define MOT1 '1'        /*numero de motor*/
#define MOT2 '2'
#define MOT3 '3'
#define MOT4 '4'
#define MOT5 '5'
#define MOT8 '8'
#define CR '\r'

#define A '0'
#define B '0'
#define C '0'
```

```
#define D '1'

#define SI 1
#define NO 0

/****** Parametres Denavit-Hatenberg *** mides en milimetres *******/
#define D1 345.0
#define D5 151.0
#define A1 16.0
#define A2 220.0
#define A3 220.0

#define PI 3.14159265359
#define GRAU 0.017453292
#define RES_ARTICULAR 0.5   /* resolucio per coordenades articulars */

/*********** Polsos del encoder perque giri 1§ l'articulacio *******/
#define P1 10.64166667
#define P2 8.513333333
#define P3 8.513333333
#define P4 2.183333333
#define P5 2.183333333
#define P8 0.1625


/*********** DECLARACIO DE TIPUS DE VARIABLES ******************/
typedef struct {
      unsigned int valor;
      unsigned int final;
      void (*punter_a_funcio) (char);
} softtimer;

typedef struct {
      double x;
      double y;
      double z;
} coordenades;

typedef struct {
      coordenades p;
      coordenades a;
      coordenades o;
      double ab;
      double ed;
} estructura_posicions;

typedef struct {
      char comanda[10];
      int quant;
} estructura_pgm;

typedef enum {BIOS, HOME, HOME2, FITXER, JOYSTICK, EXEC} Terror;



/***************** DECLARACIO DE FUNCIONS **********************/
void interrupt (*old_IRQ0_attn)(__cppargs);
void interrupt IRQ0_attn (__cppargs);
void interrupt (*old_IRQ3_attn)(__cppargs);
void interrupt IRQ3_attn (__cppargs);
void interrupt (*old_IRQ4_attn)(__cppargs);
```

```
void interrupt IRQ4_attn (__cppargs);
void inicialitzacions (void);
void ini_8259(void);
void restaurar (void);
void ini_serial_port (void);
void enviar_car (char c);
void enviar_trama (char a, char b, char c, char d, char e, char f,
char g, char h);
void moure_base (char sentit);
void moure_hombru (char sentit);
void moure_colze (char sentit);
void moure_canell (char sentit);
void rotar_canell(char sentit);
void velocitat (void);
void home_manual (void);
void home_auto (void);
void inversa (void);
void directa (char mot, char sentit);
void orientacio (void);
void anar_punt (void);
void posicio (void);
void desglosa (double polsos, char *a, char *b, char *c, char *d, int
*mov);
char conv_car (int a);
void parada_emergencia (void);
void llistat_posicions (void);
void grabar_posicio (void);
void reproduir_posicions (void);
void menu_principal (void);
void menu_config (void);
void menu_home (void);
void menu_ensenyar_pos (void);
void menu_articulacions (void);
void menu_tcp (void);
void menu_executar (void);
void menu_fitxers (void);
void dades_menu_principal (void);
void dades_menu_config (void);
void dades_menu_home (void);
void dades_menu_ensenyar_pos (void);
void dades_menu_articulacions (void);
void dades_menu_tcp (void);
void dades_menu_executar (void);
void dades_menu_fitxers (void);
void dades_articulacions_tecla (void);
void dades_articulacions_joy (void);
void dades_tcp_joy (void);
void dades_tcp_tecla (void);
void dades_posicio_home (void);
void dades_trajectoria (void);
void dades_config_usuari (void);
void articulacions_tecla (void);
void articulacions_tecla_home (void);
void articulacions_joy (void);
void tcp_joy (void);
void tcp_tecla (void);
void anar_posicio (int n_pos);
void increment_tcp (void);
void increment_polsos (void);
void eix_x (char c);
void eix_y (char c);
```

```
void canell_tcp (char c);
void trajectoria (void);
void config_usuari (void);
void llegir_config_usuari (void);
double modul (coordenades m);
void pos_actual (void);
void angle_actual (void);
void comunicacions (void);
void dades_posicio (void);
void editar_pgm (void);
void executa_un_cop (void);
void executa_cont (void);
void executa_pas (void);
void editar_pos (void);
void editar_trajectoria(void);
void editar_vel (void);
void editar_pausa (void);
void editar_obrir_pinsa (void);
void editar_tancar_pinsa (void);
void editar_borrar_linea (void);
void editar_moure_linea (void);
void ajustar_vel (int n);
void obrir_tancar_pinsa (char s);
void operar_comanda (char comanda[10], int quant);
void carrega_fitxer_pos (void);
void carrega_fitxer_pgm (void);
void guarda_fitxer_pos (void);
void guarda_fitxer_pgm (void);
void copyright (void);
void msg_error (Terror error);
char confirmar_exit (void);
void finestra (int x1, int y1, int x2, int y2, int fondo, int text);
void finestra2 (int x1, int y1, int x2, int y2, int fondo, int text);
void ini_timer (void);

void joystick (void);
void dades_joy(void);
void calibrar_joy (void);
void dades_calibrar_joy (void);
void posicio_cursor (void);
void joy_articulacions (void);
void joy_articulacions_home (void);
void joy_tcp (void);



/****************** VARIABLES GLOBALS ************************/
long int inte=0, enviat=0, rebut=0, error=0, modem=0;
unsigned int base, lsr;

int byte1=0, byte2=0, car_rebut=0, num_pos=0, num_lin=0;
char sentit_mov='+';
int pinsa=0, pinsa_max=0, pinsa_min=0;

double z1, z2, z3, z4, z5, z3_rel, z4_rel, beta=0.0, fi=0.0, roll=0.0,
pitch=0.0, yaw=0.0;
double ant_z1=0.0, ant_z2=0.0, ant_z3=0.0, ant_z4=0.0, ant_z5=0.0,
ant_z3_rel=0.0, ant_z4_rel=0.0;
double inc_tcp=25.0;
double polsos=25.0;
double num_grau=5.0;
```

```
double angle_AB=0.0, angle_ED=0.0;

coordenades P, Pm, Pb, a, o, a_ref, o_ref;
estructura_posicions pos[300];
estructura_pgm pgm[500];
char vel_mot[5];
softtimer timer;

/************ Flags del sistema ********************/
int fi_mov=SI, carte=NO, tcp=NO, tcp2=NO, home_ok=NO, articulars=NO;
int executa=NO, execucio=NO, error_motor=NO, menu_home_ok=NO;
int traject=NO, pinsa_ok=NO;
int parametres=SI, p_actual=SI, ccions=NO;
int conta_temps=NO;
int refresc_joy=NO;
int mantenir_orientacio=NO;


/************** Variables glogals Joystick **********************/
int tics=0, x,y;
int refresc=SI, inici=SI, joy_calibrat=NO;
int eix[2], calibra[6];
char bot[2];
float inc_x_esq, inc_x_dre, inc_y_dalt, inc_y_baix;



/*********************************************************************/
/********************** PROGRAMA PRINCIPAL ************************/
/*********************************************************************/
void main (void)
{
      textbackground(BLUE);
      textcolor(WHITE);
      clrscr();
      inicialitzacions();
      menu_principal();
      restaurar();
}


/*********************************************************************/
void inicialitzacions (void)
{
      int i;

      _setcursortype(_NOCURSOR);
      copyright();
      ini_timer();
      ini_8259();
      ini_serial_port();
      llegir_config_usuari();
      for (i=0; i<5; i++)
            vel_mot[i]='0';
      clrscr();
}
```

```
/***** Inicialitzacio del controlador d'interrupcions 8259 **********/
void ini_8259(void)
{
      disable();
      old_IRQ0_attn=getvect(IRQ0);     /* salvem el vell vector */
      setvect(IRQ0,IRQ0_attn);        /* posem el nou vector */
      old_IRQ3_attn=getvect(IRQ3);    /* salvem el vell vector */
      setvect(IRQ3,IRQ3_attn);        /* posem el nou vector */
      old_IRQ4_attn=getvect(IRQ4);    /* salvem el vell vector */
      setvect(IRQ4,IRQ4_attn);        /* posem el nou vector */
      outportb(BASE_8259+1,(inportb(BASE_8259+1))&MASK_IRQ0);
                                        /*habilita interrupcions*/
      outportb(BASE_8259+1,(inportb(BASE_8259+1))&MASK_IRQ3);
      outportb(BASE_8259+1,(inportb(BASE_8259+1))&MASK_IRQ4);
      enable();
}


/***************** Restaurar vectors d'interrupcio **************/
void restaurar (void)
{
      enviar_car('X');  /* desactivem inter. al controlador */
      disable();
      setvect(IRQ0,old_IRQ0_attn);   /*reposem els antics vectors
                                          d'interrupcio*/
      setvect(IRQ3,old_IRQ3_attn);
      setvect(IRQ4,old_IRQ4_attn);
      enable();
      textbackground(BLACK);
      textcolor(LIGHTGRAY);
      clrscr();
}


/*********************************************************************/
void ini_serial_port (void)
{
      char m, c, port;
      int num_port;

      clrscr();
      textbackground(GREEN);
      textcolor(BLACK);
      gotoxy(25,2);
      cprintf(" ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ» ");
      gotoxy(25,3);
      cprintf(" º   PORT DE COMUNICACIONS   º ");
      gotoxy(25,4);
      cprintf(" ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼ ");

      finestra2(17,11,62,13,BROWN,WHITE);
      do {
            gotoxy(21,12);
            printf("Quin es el port de comunicacions? (1/2)");
            port=getch();
      } while (port!='1' && port!='2');

      if (port=='1') num_port=1;
      else num_port=2;

      base=peek(0x40,(num_port-1)*2);    /*port de comunicacions per
```

```
                                                        la BIOS*/
      if (base==0) {
            msg_error(BIOS);
            exit (1);
      } else {
            outportb(LCR,0x80);     /*Acces a DLLSB i DLMSB*/
            outportb(LSB,0x0c);     /*baudis 9600*/
            outportb(MSB,0x00);     /*baudis 9600*/
            outportb(LCR,0x07);     /*8 bits caracter, no paritat, no
                                    break control acces resgistres
                                    trans/rep dades, 2 bits stop*/
            outportb(IER,0x0f);     /*activacio de les interrupcions:
                              transmissio, recepcio, errors i modem*/
            outportb(MCR,0x08);     /*permis d'interrupcions
                                                  controlador*/

            disable();
            m=inportb(BASE_8259+1); /*registre mascara CI8259*/
            if (port=='1')       m=m&MASK_IRQ4;    /*int. pel COM1 i
                                                     COM3*/
            else if (port=='2')  m=m&MASK_IRQ3;    /*int. pel COM2 i
                                                     COM4*/

            outportb(BASE_8259+1,m); /*activacio IRQ(linea serie
                                                  COM1/2)*/
            enable();

            c=inportb(MSR);   /*eliminar interrupciuons pendents*/
            c=inportb(LSR);   /*eliminar interrupciuons pendents*/
            c=inportb(IIR);   /*eliminar interrupciuons pendents*/
            c=inportb(RBR);   /*eliminar interrupciuons pendents*/
            c=inportb(THR);   /*eliminar interrupciuons pendents*/
      }
      enviar_car('W');    /* activar inte. en el controlador */
      enviar_car('U');    /* nou desfasament */
      enviar_car('0');
      enviar_car('1');
      enviar_car('2');
      enviar_car('*');

      textbackground(BLUE);
      textcolor(WHITE);
      clrscr();
}


/*********** Rutina de servei a l'interrupcio IRQ0 *****************/
void interrupt IRQ0_attn (__cppargs)
{
      if (refresc_joy) {
            tics++;
            if (tics>=KTICS){
                  tics=0;
                  refresc=SI;    /*refresc de les dades de la
                                              pantalla*/
            }
      }

      if (conta_temps) {         /*inici moviment de motor */
            if (timer.valor==timer.final) {
                  timer.punter_a_funcio('A');
```

```
                timer.valor=KTICS2;
            }
            timer.valor--;
      }

      outportb(BASE_8259,EOI);
}


/************ Rutina de servei a l'interrupcio IRQ3 ****************/
void interrupt IRQ3_attn (__cppargs)
{
      char c;

      inte++;
      c=inportb(IIR);
      if (c&0x04) {  /*caracter rebut*/
            rebut++;
            car_rebut=inportb(RBR);
            byte1=byte2;
            byte2=car_rebut;
            if ((!fi_mov && car_rebut==48 || car_rebut=='-'))
                  fi_mov=SI;
            else {
                  if (!(byte1=='K' || byte1=='L') && (int(byte2)>=49 &&
int(byte2)<=56))
                        error_motor=SI;
                  if (byte2=='8')
                        error_motor=SI;
            }
      }
      else if (c&0x06) {   /*canvi estat de la linia*/
            error++;
            c=inportb(LSR);
      }
      else if (c&0x00) {   /*error de modem*/
            modem++;
            c=inportb(MSR);
      }
      outportb(BASE_8259,EOI);      /*fi d'interrupcio*/
}


/************ Rutina de servei a l'interrupcio IRQ4 ****************/
void interrupt IRQ4_attn (__cppargs)
{
      char c;

      inte++;
      c=inportb(IIR);
      if (c&0x04) {  /*caracter rebut*/
            rebut++;
            car_rebut=inportb(RBR);
            byte1=byte2;
            byte2=car_rebut;
            if ((!fi_mov && car_rebut==48 || car_rebut=='-'))
                  fi_mov=SI;
            else {
                  if (!(byte1=='K' || byte1=='L') && (int(byte2)>=49 &&
int(byte2)<=56))
                        error_motor=SI;
```

```
                if (byte2=='8')
                      error_motor=SI;
          }
     }
     else if (c&0x06) {   /*canvi estat de la linia*/
          error++;
          c=inportb(LSR);
     }
     else if (c&0x00) {   /*error de modem*/
          modem++;
          c=inportb(MSR);
     }
     outportb(BASE_8259,EOI);      /*fi d'interrupcio*/
}


/*****************************************************************/
void enviar_car (char c)
{
     do {
          lsr=inportb(LSR);
     } while(!(lsr&0x20)); /*esperem a poder enviar*/

     outportb(THR,c);     /*coloquem el caracter al THR*/
}


/*****************************************************************/
void enviar_trama (char a, char b, char c, char d, char e, char f,
char g, char h)
{
     enviar_car(a);
     enviar_car(b);
     enviar_car(c);
     enviar_car(d);
     enviar_car(e);
     enviar_car(f);
     enviar_car(g);
     enviar_car(h);
}


/*****************************************************************/
void moure_base (char sentit)
{
     char mil,cent,dec,unit;
     int mov;

     desglosa (polsos,&mil,&cent,&dec,&unit,&mov);

     if (mov) {
          if (sentit=='+')
               enviar_trama(MOT1,M,'+',mil,cent,dec,unit,CR);
          else if (sentit=='-')
               enviar_trama(MOT1,M,'-',mil,cent,dec,unit,CR);
     }
}
```

```
/*************************************************************/
void moure_hombru (char sentit)
{
      char mil,cent,dec,unit;
      int mov;

      desglosa (polsos,&mil,&cent,&dec,&unit,&mov);

      if (mov) {
            if (sentit=='+')
                  enviar_trama(MOT2,M,'+',mil,cent,dec,unit,CR);
            else if (sentit=='-')
                  enviar_trama(MOT2,M,'-',mil,cent,dec,unit,CR);
      }
}


/*************************************************************/
void moure_colze (char sentit)
{
      char mil,cent,dec,unit;
      int mov;

      desglosa (polsos,&mil,&cent,&dec,&unit,&mov);

      if (mov) {
            if (sentit=='+')
                  enviar_trama(MOT3,M,'+',mil,cent,dec,unit,CR);
            else if (sentit=='-')
                  enviar_trama(MOT3,M,'-',mil,cent,dec,unit,CR);
      }
}


/*************************************************************/
void moure_canell (char sentit)
{
      char mil,cent,dec,unit;
      int mov;

      desglosa (polsos,&mil,&cent,&dec,&unit,&mov);

      if (mov) {
            if (sentit=='+') {
                  enviar_trama(MOT4,M,'+',mil,cent,dec,unit,CR);
                  enviar_trama(MOT5,M,'-',mil,cent,dec,unit,CR);
            } else if (sentit=='-') {
                  enviar_trama(MOT4,M,'-',mil,cent,dec,unit,CR);
                  enviar_trama(MOT5,M,'+',mil,cent,dec,unit,CR);
            }
      }
}


/*************************************************************/
void rotar_canell (char sentit)
{
      char mil,cent,dec,unit;
      int mov;

      desglosa (polsos,&mil,&cent,&dec,&unit,&mov);
```

```
      if (mov) {
            enviar_trama(MOT4,M,sentit,mil,cent,dec,unit,CR);
            enviar_trama(MOT5,M,sentit,mil,cent,dec,unit,CR);
      }
}


/*****************************************************************/
void velocitat (void)
{
      char vel, motor;

      clrscr();
      textbackground(GREEN);
      textcolor(BLACK);
      gotoxy(25,2);
      cprintf(" ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ» ");
      gotoxy(25,3);
      cprintf(" º   VELOCITAT DELS MOTORS   º ");
      gotoxy(25,4);
      cprintf(" ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼ ");

      textbackground(WHITE);
      textcolor(BLACK);
      gotoxy(23,7);
      cprintf(" ÉÍÍÍÍÍÍÍÍÍÍÍËÍÍÍËÍÍÍËÍÍÍËÍÍÍËÍÍÍ» ");
      gotoxy(23,8);
      cprintf(" º   MOTOR   º 1 º 2 º 3 º 4 º 5 º ");
      gotoxy(23,9);
      cprintf(" ÌÍÍÍÍÍÍÍÍÍÍÍÎÍÍÍÎÍÍÍÎÍÍÍÎÍÍÍÎÍÍÍ¹ ");
      gotoxy(23,10);
      cprintf(" º VELOCITAT º %c º %c º %c º %c º %c º ",
            vel_mot[0],vel_mot[1],vel_mot[2],vel_mot[3],vel_mot[4]);
      gotoxy(23,11);
      cprintf(" ÈÍÍÍÍÍÍÍÍÍÍÍÊÍÍÍÊÍÍÍÊÍÍÍÊÍÍÍÊÍÍÍ¼ ");


      finestra2(21,14,59,16,BROWN,WHITE);
      do {
            gotoxy(25,15);
            printf("Quin motor vols ajustar? (1...5)");
            motor=getch();
      } while (motor<'1' || motor>'5');
      finestra2(21,18,59,20,BROWN,WHITE);
      do {
            gotoxy(27,19);
            printf("Quina velocitat? (1,2,...9,0)");
            vel=getch();
      } while (vel<'0' || vel>'9');

      clrscr();
      vel_mot[int(motor)-49]=vel;

      enviar_car(motor);
      enviar_car(V);
      enviar_car(vel);
      textbackground(BLUE);
      textcolor(WHITE);
      clrscr();
}
```

```
/********* Sequencia de busqueda de HOME ********************/
void home_manual (void)
{
      int i=0;
      home_ok=NO;

      clrscr();

      textbackground(GREEN);
      textcolor(BLACK);
      gotoxy(25,2);
      cprintf(" ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ» ");
      gotoxy(25,3);
      cprintf(" º   LOCALITZACIO DE HOME   º ");
      gotoxy(25,4);
      cprintf(" ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼ ");

      gotoxy(19,6);
      printf("Coloca el robot en una posicio coneguda i");
      gotoxy(19,7);
      printf("introdueix els seguents parametres:");
      dades_posicio_home();

      a_ref.x=a.x;    /*vectors d'orientacio de referencia*/
      a_ref.y=a.y;
      a_ref.z=a.z;
      o_ref.x=o.x;
      o_ref.y=o.y;
      o_ref.z=o.z;

      sentit_mov='-';
      while (i<3) {                  /* calibrar la pinsa*/
            if (error_motor) {
                  if (sentit_mov=='+') sentit_mov='-';
                  else sentit_mov='+';
                  error_motor=NO;
                  i++;
            }
            if (i==2) pinsa++;
            enviar_trama(MOT8,M,sentit_mov,A,B,C,D,CR);
      }
      for (i=0; i<49; i++)
            enviar_trama(MOT8,M,sentit_mov,A,B,C,D,CR);
      pinsa_max=pinsa-25;
      pinsa_min=85;
      pinsa_ok=SI;
      angle_AB=0;
      angle_ED=0;

      inversa();
      grabar_posicio();
      home_ok=SI;
      clrscr();
}
```

```
/*****************************************************************/
void home_auto (void)
{
      int i=0;
      home_ok=NO;

      clrscr();
      textbackground(GREEN);
      textcolor(BLACK);
      gotoxy(25,2);
      cprintf(" ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ» ");
      gotoxy(25,3);
      cprintf(" º   BUSQUEDA DE HARD HOME   º ");
      gotoxy(25,4);
      cprintf(" ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼ ");

      sentit_mov='-';
      gotoxy(31,9);
      textattr(159);
      cprintf("La base esta fent HOMING");
      while (!(byte1=='K' && byte2=='1' && sentit_mov=='-') &&
                  !(byte1=='L' && byte2=='1' && sentit_mov=='+')) {
            if (error_motor) {
                  if (sentit_mov=='+') sentit_mov='-';
                  else sentit_mov='+';
                  error_motor=NO;
            }
            enviar_trama(MOT1,M,sentit_mov,'0','0','0','1',CR); }
      gotoxy(31,9);
      textattr(31);
      cprintf("La base es a HOME        ");

      sentit_mov='-';
      gotoxy(31,11);
      textattr(159);
      cprintf("L'hombru esta fent HOMING");
      while (!(byte1=='L' && byte2=='2' && sentit_mov=='-') &&
                  !(byte1=='K' && byte2=='2' && sentit_mov=='+')) {
            if (error_motor) {
                  if (sentit_mov=='+') sentit_mov='-';
                  else sentit_mov='+';
                  error_motor=NO;
            }
            enviar_trama(MOT2,M,sentit_mov,'0','0','0','1',CR); }
      gotoxy(31,11);
      textattr(31);
      cprintf("L'hombru es a HOME          ");

      sentit_mov='+';
      gotoxy(31,13);
      textattr(159);
      cprintf("El colze esta fent HOMING");
      while (!(byte1=='K' && byte2=='3' && sentit_mov=='-') &&
                  !(byte1=='L' && byte2=='3' && sentit_mov=='+')) {
            if (error_motor) {
                  if (sentit_mov=='+') sentit_mov='-';
                  else sentit_mov='+';
                  error_motor=NO;
            }
            enviar_trama(MOT3,M,sentit_mov,'0','0','0','1',CR); }
      gotoxy(31,13);
```

```
textattr(31);
cprintf("El colze es a HOME         ");

sentit_mov='-';
gotoxy(31,15);
textattr(159);
cprintf("Pitch esta fent HOMING");
while (!(byte1=='L' && byte2=='4' && sentit_mov=='-') &&
            !(byte1=='K' && byte2=='4' && sentit_mov=='+')) {
      if (error_motor) {
            if (sentit_mov=='+') sentit_mov='-';
            else sentit_mov='+';
            error_motor=NO;
      }
      moure_canell(sentit_mov);  }
gotoxy(31,15);
textattr(31);
cprintf("Pitch es a HOME        ");

sentit_mov='+';
gotoxy(31,17);
textattr(159);
cprintf("Roll esta fent HOMING");
while (!(byte1=='L' && byte2=='5' && sentit_mov=='-') &&
            !(byte1=='K' && byte2=='5' && sentit_mov=='+')) {
      if (error_motor) {
            if (sentit_mov=='+') sentit_mov='-';
            else sentit_mov='+';
            error_motor=NO;
      }
      rotar_canell(sentit_mov);  }
gotoxy(31,17);
textattr(31);
cprintf("Roll es a HOME         ");

sentit_mov='-';         /*calibrar pinsa*/
gotoxy(31,19);
textattr(159);
cprintf("La Pinsa esta fent HOMING");
while (i<3) {
      if (error_motor) {
            if (sentit_mov=='+') sentit_mov='-';
            else sentit_mov='+';
            error_motor=NO;
            i++;
      }
      if (i==2) pinsa++;
      enviar_trama(MOT8,M,sentit_mov,A,B,C,D,CR);
}
for (i=0; i<49; i++)
      enviar_trama(MOT8,M,sentit_mov,A,B,C,D,CR);
pinsa_max=pinsa-50;
pinsa_min=100;
pinsa_ok=SI;
gotoxy(31,19);
textattr(31);
cprintf("Pinsa a HOME              ");

P.x=213.6;  /* Posicio de HARD HOME */
P.y=0.0;
P.z=565.0;
```

```
      a.x=0.8;
      a.y=0.0;
      a.z=-0.51;
      o.x=0.0;
      o.y=1.0;
      o.z=0.0;

      a_ref.x=1.0;    /*vectors d'orientacio de referencia*/
      a_ref.y=0.0;
      a_ref.z=0.0;
      o_ref.x=0.0;
      o_ref.y=1.0;
      o_ref.z=0.0;


      inversa();
      grabar_posicio();
      home_ok=SI;

      textbackground(GREEN);
      textcolor(BLACK);
      gotoxy(25,2);
      cprintf(" ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ» ");
      gotoxy(25,3);
      cprintf(" º   BUSQUEDA DE HARD HOME   º ");
      gotoxy(25,4);
      cprintf(" ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼ ");

      finestra2(28,9,52,11,BROWN,WHITE);
      gotoxy(32,10);
      textattr(239);
      cprintf("El robot es a HOME");
      gotoxy(25,19);
      textattr(31);
      printf("Prem qualsevol tecla per continuar");
      while (!kbhit());
      clrscr();
}




/********* INVERSA Calcul dels angles a partir de la posicio *******/
void inversa (void)
{
      double k1,k2,s2,c2,s3,c3,r13,r23,r31,r32;
      double modul_a, modul_Pm;
      coordenades u;

      /********* guardem els angles de la posicio anterior ******/
      ant_z1=z1;
      ant_z2=z2;
      ant_z3=z3;
      ant_z4=z4;
      ant_z5=z5;
      ant_z3_rel=z3_rel;
      ant_z4_rel=z4_rel;
```

```
/*********** DETERMINACIO DE LES COORDENADES DE Pb *********/
if (!tcp2) {
      if ((a.x==0.0) && (a.y==0.0)) fi=fi;
      else if (a.y==0.0) fi=0.0;
      else if (a.x==0.0) {
            if (a.y>0.0) fi=PI/2;
            else fi=-PI/2;
      } else fi=atan2(a.y,a.x);
      beta=atan2(sqrt((a.x*a.x)+(a.y*a.y)),a.z);
}
/******* coordenades de Pm *********/
Pm.x=D5*sin(beta)*cos(fi);
Pm.y=D5*sin(beta)*sin(fi);
Pm.z=D5*cos(beta);

if (tcp2) {
      modul_Pm=sqrt((Pm.x*Pm.x)+(Pm.y*Pm.y)+(Pm.z*Pm.z));
      a.x=Pm.x/modul_Pm;
      a.y=Pm.y/modul_Pm;
      a.z=Pm.z/modul_Pm;
      tcp2=NO;
}
/******* coordenades de Pb *************/
Pb.x=P.x-Pm.x;
Pb.y=P.y-Pm.y;
Pb.z=P.z-Pm.z;

/*********** DETERMINACIO DE LA POSICIO **************/
/********* determinacio de z1 ****************/
if (Pb.x==0.0) {
      if (Pb.y>0.0) z1=PI/2;
      else if (Pb.y<0.0) z1=-PI/2;
      else z1=z1;
}
else z1=atan2(Pb.y,Pb.x);

/********** determinacio de z3 ***************/
c3=((Pb.x*Pb.x)+(Pb.y*Pb.y)+(Pb.z*Pb.z)+(A1*A1)+(D1*D1)-
(2*A1*sqrt((Pb.x*Pb.x)+(Pb.y*Pb.y)))-(2*Pb.z*D1)-(A2*A2)-
                                          (A3*A3))/(2*A2*A3);
s3=sqrt(1-(c3*c3));
if (c3==0.0) {
      if (s3>0.0) z3=PI/2;
      else if (s3<0.0) z3=-PI/2;
      else z3=z3;
}
else z3=atan2(s3,c3);

/********** determinacio de z2 **************/
k1=(sqrt((Pb.x*Pb.x)+(Pb.y*Pb.y)))-A1;
k2=Pb.z-D1;
s2=-((k1*A3*s3)+(k2*A3*c3)+(k2*A2))/((k1*k1)+(k2*k2));
c2=((k1*A3*c3)-(k2*A3*s3)+(k1*A2))/((k1*k1)+(k2*k2));
if (c2==0.0) {
      if (s2>0.0) z2=PI/2;
      else if (s2<0.0) z2=-PI/2;
      else z2=z2;
}
else z2=atan2(s2,c2);
```

```
      /************** DETERMINACIO DE LA ORIENTACIO ***************/
      /****** determinacio del vector U ******************/
      u.x=(o.y*a.z)-(o.z*a.y);
      u.y=(o.z*a.x)-(o.x*a.z);
      u.z=(o.x*a.y)-(o.y*a.x);

      /********* determinacio de z4 i z5 ***************/
      r13=(cos(z2+z3)*((cos(z1)*a.x)+(sin(z1)*a.y)))-(sin(z2+z3)*a.z);
      r23=-(sin(z2+z3)*((cos(z1)*a.x)+(sin(z1)*a.y)))-
                                                  (cos(z2+z3)*a.z);
      r31=(cos(z1)*u.y)-(sin(z1)*u.x);
      r32=(cos(z1)*o.y)-(sin(z1)*o.x);

      if (r23==0.0) {          /***** determincaio de z4 ******/
            if (r13>0.0) z4=PI/2;
            else if (r13<0.0) z4=-PI/2;
            else z4=z4;
      }
      else z4=atan2(r13,-r23);
      if ((z4<-(PI/2)) && (z4>(-PI)))
            z4=z4+(2*PI);

      if (r32==0.0) {         /******* determinacio de z5 ******/
            if (r31>0) z5=PI/2;
            else if (r31<0) z5=-PI/2;
            else z5=z5;
      }
      else z5=atan2(r31,r32);
      if (tcp) {              /* gir de 360º en el mateix sentit */
            if (z5!=angle_ED)
                  z5=angle_ED;
      }


      z3_rel=z2+z3;                    /*compensacio de z3 i z4*/
      z4_rel=z2+z3+z4-(PI/2);
}


/********************* Cinematica Dirtecta ************************/
void directa (char mot, char sentit)
{
      double m_directa[4][4];

      ant_z1=z1;
      ant_z2=z2;
      ant_z3=z3;
      ant_z4=z4;
      ant_z5=z5;
      ant_z3_rel=z3_rel;
      ant_z4_rel=z4_rel;

      if (mot==MOT1) {     /*moure BASE*/
            if (sentit=='+') z1=ant_z1+((PI*polsos)/(180*P1));
            else z1=ant_z1-((PI*polsos)/(180*P1));
      }
      else if (mot==MOT2) {     /*moure hombru*/
            if (sentit=='+') {
                  z2=ant_z2+((PI*polsos)/(180*P2));
                  z3=ant_z3-((PI*polsos)/(180*P2));
            } else {
```

```
                z2=ant_z2-((PI*polsos)/(180*P2));
                z3=ant_z3+((PI*polsos)/(180*P2));
          }
     }
     else if (mot==MOT3) {    /*moure colze*/
          if (sentit=='+') {
                z3=ant_z3-((PI*polsos)/(180*P3));
                z4=ant_z4+((PI*polsos)/(180*P3));
          } else {
                z3=ant_z3+((PI*polsos)/(180*P3));
                z4=ant_z4-((PI*polsos)/(180*P3));
          }
     }
     else if (mot==MOT4) {      /*moure canell Amunt/aBaix*/
          if (sentit=='+') {
                z4=ant_z4-((PI*polsos*RES_ARTICULAR)/(180*P4));

     angle_AB=angle_AB+((PI*polsos*RES_ARTICULAR)/(180*P4));
          } else {
                z4=ant_z4+((PI*polsos*RES_ARTICULAR)/(180*P4));
                angle_AB=angle_AB-
((PI*polsos*RES_ARTICULAR)/(180*P4));
          }
     }
     else if (mot==MOT5) {      /*girar canell Esquerra/Dreta*/
          if (sentit=='-') {
                z5=ant_z5+((PI*polsos*RES_ARTICULAR)/(180*P5));

     angle_ED=angle_ED+((PI*polsos*RES_ARTICULAR)/(180*P5));
          } else {
                z5=ant_z5-((PI*polsos*RES_ARTICULAR)/(180*P5));
                angle_ED=angle_ED-
((PI*polsos*RES_ARTICULAR)/(180*P5));
          }
     }

     z3_rel=z2+z3;                    /*compensacio de z3 i z4*/
     z4_rel=z2+z3+z4-(PI/2);

     m_directa[0][0]=((cos(z1)*cos(z2+z3+z4)*cos(z5))-
                                          (sin(z1)*sin(z5)));
     m_directa[0][1]=(-(cos(z1)*cos(z2+z3+z4)*sin(z5))-
                                          (sin(z1)*cos(z5)));
     m_directa[0][2]=cos(z1)*sin(z2+z3+z4);
     m_directa[0][3]=(D5*cos(z1)*sin(z2+z3+z4))+(cos(z1)*((A3*cos(z2+
                                          z3))+(A2*cos(z2))+A1));
     m_directa[1][0]=((sin(z1)*cos(z2+z3+z4)*cos(z5))+(cos(z1)*sin(z5
                                                          )));
     m_directa[1][1]=(-(sin(z1)*cos(z2+z3+z4)*sin(z5))+
                                          (cos(z1)*cos(z5)));
     m_directa[1][2]=sin(z1)*sin(z2+z3+z4);
     m_directa[1][3]=(D5*sin(z1)*sin(z2+z3+z4))+(sin(z1)*((A3*cos(z2+
                                     z3))+(A2*cos(z2))+(A1)));
     m_directa[2][0]=-sin(z2+z3+z4)*cos(z5);
     m_directa[2][1]=sin(z2+z3+z4)*sin(z5);
     m_directa[2][2]=cos(z2+z3+z4);
     m_directa[2][3]=((D5*cos(z2+z3+z4))-(A3*sin(z2+z3))-
                                               (A2*sin(z2))+D1);
     m_directa[3][0]=0;
     m_directa[3][1]=0;
     m_directa[3][2]=0;
```

```
      m_directa[3][3]=1;

      P.x=m_directa[0][3];
      P.y=m_directa[1][3];
      P.z=m_directa[2][3];
      a.x=m_directa[0][2];
      a.y=m_directa[1][2];
      a.z=m_directa[2][2];
      o.x=m_directa[0][1];
      o.y=m_directa[1][1];
      o.z=m_directa[2][1];

      posicio();
      if (traject) grabar_posicio();
}



/****************** MOVIMENT DELS MOTORS **********************/
void posicio (void)
{
      double pols1,pols2,pols3,pols4,pols5;
      int mov=NO;
      char mil,cent,dec,unit;
      char c;

      pols1=P1*(fabsl(z1-ant_z1))*180/PI;     /*polsos a enviar als
                                                      motors*/
      pols2=P2*(fabsl(z2-ant_z2))*180/PI;
      pols3=P3*(fabsl(z3_rel-ant_z3_rel))*180/PI;
      pols4=P4*(fabsl(z4_rel-ant_z4_rel))*180/PI;
      pols5=P5*(fabsl(z5-ant_z5))*180/PI;

      conta_temps=SI;


      /********** moure la base ************/
      desglosa (pols1,&mil,&cent,&dec,&unit,&mov);

      if (mov) {
            if ((z1-ant_z1)>0) {
                  fi_mov=NO;
                  enviar_trama(MOT1,M,'+',mil,cent,dec,unit,CR);
            } else if ((z1-ant_z1)<0) {
                  fi_mov=NO;
                  enviar_trama(MOT1,M,'-',mil,cent,dec,unit,CR);
            }
      }

      /********** moure l'hombru ************/
      desglosa (pols2,&mil,&cent,&dec,&unit,&mov);

      if (mov) {
            if ((z2-ant_z2)>0) {
                  fi_mov=NO;
                  enviar_trama(MOT2,M,'+',mil,cent,dec,unit,CR);
            } else if ((z2-ant_z2)<0) {
                  fi_mov=NO;
                  enviar_trama(MOT2,M,'-',mil,cent,dec,unit,CR);
            }
      }
```

```
/********** moure el colze ************/
desglosa (pols3,&mil,&cent,&dec,&unit,&mov);

if (mov) {
      if ((z3_rel-ant_z3_rel)>0) {
            fi_mov=NO;
            enviar_trama(MOT3,M,'-',mil,cent,dec,unit,CR);
      } else if ((z3_rel-ant_z3_rel)<0) {
            fi_mov=NO;
            enviar_trama(MOT3,M,'+',mil,cent,dec,unit,CR);
      }
}

/********** moure el canell ************/
desglosa (pols4,&mil,&cent,&dec,&unit,&mov);

if (mov) {
      if ((z4_rel-ant_z4_rel)>0) {
            fi_mov=NO;
            enviar_trama(MOT4,M,'-',mil,cent,dec,unit,CR);
            enviar_trama(MOT5,M,'+',mil,cent,dec,unit,CR);
      } else if ((z4_rel-ant_z4_rel)<0) {
            fi_mov=NO;
            enviar_trama(MOT4,M,'+',mil,cent,dec,unit,CR);
            enviar_trama(MOT5,M,'-',mil,cent,dec,unit,CR);
      }
}

/********** rotar el canell ************/
desglosa (pols5,&mil,&cent,&dec,&unit,&mov);

if (mov) {
      if ((z5-ant_z5)>0) {
            fi_mov=NO;
            enviar_trama(MOT4,M,'-',mil,cent,dec,unit,CR);
            enviar_trama(MOT5,M,'-',mil,cent,dec,unit,CR);
            if (mantenir_orientacio) {
                  angle_ED=angle_ED+(z5-ant_z5);
                  mantenir_orientacio=NO;
            }
      } else if ((z5-ant_z5)<0) {
            fi_mov=NO;
            enviar_trama(MOT4,M,'+',mil,cent,dec,unit,CR);
            enviar_trama(MOT5,M,'+',mil,cent,dec,unit,CR);
            if (mantenir_orientacio) {
                  angle_ED=angle_ED+(z5-ant_z5);
                  mantenir_orientacio=NO;
            }
      }
}

while (!fi_mov) {
      if (error_motor)
            parada_emergencia();
      if (kbhit() && !execucio)
            c=getch();
}
conta_temps=NO;
timer.valor=KTICS2;

angle_actual();
```

```
      pos_actual();
}


/*****************************************************************/
void anar_punt (void)
{
      clrscr();
      textbackground(GREEN);
      textcolor(BLACK);
      gotoxy(16,2);
      cprintf(" ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ» ");
      gotoxy(16,3);
      cprintf(" º   UBICACIO DEL ROBOT EN UN PUNT CONCRET   º ");
      gotoxy(16,4);
      cprintf(" ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼ ");
      gotoxy(18,7);
      printf("Introdueix les seg•ents dades per portar el");
      gotoxy(22,8);
      printf("robot al punt desitjat de l'espai:");
      dades_posicio();
      inversa();
      posicio();

      window(1,1,80,25);
      textbackground(BLUE);
      textcolor(WHITE);
      clrscr();
}


/*****************************************************************/
void desglosa (double polsos, char *a, char *b, char *c, char *d, int
*mov)
{
      int resto, pols;

      if (polsos<0.5)
            *mov=NO;
      else {
            *mov=SI;
            pols=int(polsos);       /*arrodonim a l'enter mes proper*/
            if ((polsos-pols)>=0.5)
                  pols++;

            *a=conv_car(pols/1000);
            resto=pols%1000;
            *b=conv_car(resto/100);
            resto=resto%100;
            *c=conv_car(resto/10);
            *d=conv_car(resto%10);
      }
}
```

```
/*****************************************************************/
char conv_car (int a)
{
      char c;

      switch (a) {
            case 1: c='1'; break;
            case 2: c='2'; break;
            case 3: c='3'; break;
            case 4: c='4'; break;
            case 5: c='5'; break;
            case 6: c='6'; break;
            case 7: c='7'; break;
            case 8: c='8'; break;
            case 9: c='9'; break;
            case 0: c='0'; break;
      }
      return c;
}


/*****************************************************************
**/
void parada_emergencia (void)
{
      int i;
      char c, car;

      car=car_rebut;

      if (car=='8' && pinsa_ok) {
            enviar_car(car);
            enviar_car('P');
            fi_mov=SI;
      } else {
            enviar_car('B');
            do {
                  finestra(22,8,62,16,BLACK,BLACK);
                  finestra(20,7,60,15,RED,BLACK);
                  gotoxy(30,10);
                  printf("ERROR de motor n§ %c",car);
                  gotoxy(30,12);
                  printf("Continuar o Parar?");

                  while(!kbhit());
                  c=getch();
            } while (c!='c' && c!='C'&& c!='p' && c!='P');

            if (c=='p' || c=='P') {     /* Parar */
                  enviar_car(car);
                  enviar_car('P');
                  fi_mov=SI;
            } else  enviar_car('C');      /* Seguir */
      }
      error_motor=NO;
      window(1,1,80,25);
      textbackground(BLUE);
      textcolor(WHITE);
      clrscr();
      textcolor(BLUE);
      gotoxy(75,1);
```

```
      putch('z');
      textcolor(WHITE);
}


/********************** FUNCIO: menu principal **********************/
void dades_menu_principal (void)
{
      finestra2(28,2,52,4,GREEN,BLACK);
      gotoxy(34,3);
      printf("MENU PRINCIPAL");

      finestra2(22,7,58,21,BROWN,WHITE);
      gotoxy(27,8);
      printf("1) Configuracions del sistema");
      gotoxy(27,10);
      printf("2) Definir HOME");
      gotoxy(27,12);
      printf("3) Ensenyar posicions");
      gotoxy(27,14);
      printf("4) Editar programa");
      gotoxy(27,16);
      printf("5) Executar programa");
      gotoxy(27,18);
      printf("6) Carregar/Guardar fitxers");
      textbackground(RED);
      textcolor(BLACK);
      gotoxy(35,20);
      cprintf("0) SORTIR");
      angle_actual();
      pos_actual();
      comunicacions();

      if (!home_ok) {
            gotoxy(15,23);
            textattr(192);
            cprintf("           El robot NO es a HOME !!!!!        ");
            textbackground(BLUE);
            textcolor(WHITE);
      }
}


/*********************************************************************
/
void menu_principal (void)
{
      char opcio;

      clrscr();
      do {
            dades_menu_principal();
            while (!kbhit());
            opcio=getch();

            switch (opcio) {
                  case '1': menu_config();
                        break;
                  case '2': menu_home();
                        break;
                  case '3': menu_ensenyar_pos();
```

```
                        break;
                  case '4': editar_pgm();
                        break;
                  case '5': menu_executar();
                        break;
                  case '6': menu_fitxers();
                        break;
                  case '0': opcio=confirmar_exit();
                        break;
            }
      } while (opcio!='0');
}


/*****************************************************************/
void dades_menu_config (void)
{
      textbackground(GREEN);
      textcolor(BLACK);
      gotoxy(22,2);
      cprintf(" ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ» ");
      gotoxy(22,3);
      cprintf(" º   CONFIGURACIONS DEL SISTEMA   º ");
      gotoxy(22,4);
      cprintf(" ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼ ");
      finestra2(21,7,57,21,BROWN,WHITE);
      gotoxy(24,8);
      printf("1) Port de comunicacions");
      gotoxy(24,10);
      printf("2) Velocitat dels motors");
      gotoxy(24,12);
      printf("3) Polsos del moviment articular");
      gotoxy(24,14);
      printf("4) Resolucio del moviment de tcp");
      gotoxy(24,16);
      printf("5) Calibrar Joystick");
      gotoxy(24,18);
      printf("6) Configuracions de l'usuari");
      gotoxy(30,20);
      printf("ESC) Menu anterior");

      if (!home_ok) {
            gotoxy(15,23);
            textattr(192);
            cprintf("           El robot NO es a HOME !!!!!         ");
            textbackground(BLUE);
            textcolor(WHITE);
      }
      angle_actual();
      pos_actual();
      comunicacions();
}


/*****************************************************************/
void menu_config (void)
{
      char opcio;

      do {
            clrscr();
```

```
            dades_menu_config();
            while (!kbhit());
            opcio=getch();

            switch (opcio) {
                  case '1': ini_serial_port();
                        break;
                  case '2': velocitat();
                        break;
                  case '3': increment_polsos();
                        break;
                  case '4': increment_tcp();
                        break;
                  case '5': calibrar_joy();
                        break;
                  case '6': config_usuari();
                        break;
            }
      } while (int(opcio)!=27);    /*tecla ESCAPE*/
      clrscr();
}


/*********************************************************************/
void dades_menu_home (void)
{
      textbackground(GREEN);
      textcolor(BLACK);
      gotoxy(29,2);
      cprintf(" ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ» ");
      gotoxy(29,3);
      cprintf(" º     MENU HOME     º ");
      gotoxy(29,4);
      cprintf(" ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼ ");
      finestra2(19,8,61,17,BROWN,WHITE);
      gotoxy(23,9);
      printf("1) Introduir posicio HOME manualment");
      gotoxy(23,11);
      printf("2) Trobar HARD HOME automaticament");
      gotoxy(23,13);
      printf("3) Operar el bra‡ manualment");
      gotoxy(29,16);
      printf("ESC) Menu anterior");

      if (!home_ok) {
            gotoxy(15,23);
            textattr(192);
            cprintf("           El robot NO es a HOME !!!!!         ");
            textbackground(BLUE);
            textcolor(WHITE);
      }

      angle_actual();
      pos_actual();
      comunicacions();
}
```

```
/*********************************************************************/
void menu_home (void)
{
      char opcio;
      float ant_polsos;

      clrscr();
      ant_polsos=polsos;
      polsos=1.0;
      menu_home_ok=SI;
      do {
            dades_menu_home();
            while (!kbhit());
            opcio=getch();

            switch (opcio) {
                  case '1': home_manual();
                        break;
                  case '2': home_auto();
                        break;
                  case '3': menu_articulacions();
                        break;
            }
      } while (int(opcio)!=27);    /*tecla ESCAPE*/
      menu_home_ok=NO;
      polsos=ant_polsos;
      clrscr();
}


/*********************************************************************/
void dades_menu_ensenyar_pos (void)
{
      textbackground(GREEN);
      textcolor(BLACK);
      gotoxy(27,2);
      cprintf(" ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ» ");
      gotoxy(27,3);
      cprintf(" º   ENSENYAR POSICIONS   º ");
      gotoxy(27,4);
      cprintf(" ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼ ");
      finestra2(20,6,60,22,BROWN,WHITE);
      gotoxy(25,7);
      printf("1) Moviment per articulacions");
      gotoxy(25,9);
      printf("2) Moviment del TCP");
      gotoxy(25,11);
      printf("3) Anar a un punt de l'espai");
      gotoxy(25,13);
      printf("4) Ensenyar trajectoria");
      gotoxy(25,15);
      printf("5) Memoritzar posicio");
      gotoxy(25,17);
      printf("6) Llistat de posicions");
      gotoxy(25,19);
      printf("7) Reproduir pos. memoritzades");
      gotoxy(29,21);
      printf("ESC) Menu anterior");
      angle_actual();
      pos_actual();
      comunicacions();
```

```
/*********************************************************************/
void menu_ensenyar_pos (void)
{
      char opcio;

      if (!home_ok)
            msg_error(HOME);
      else {
            clrscr();
            do {
                  dades_menu_ensenyar_pos();
                  while (!kbhit());
                  opcio=getch();

                  switch (opcio) {
                        case '1': menu_articulacions();
                              break;
                        case '2': menu_tcp();
                              break;
                        case '3': anar_punt();
                              break;
                        case '4': trajectoria();
                              break;
                        case '5': grabar_posicio();
                              break;
                        case '6': llistat_posicions();
                              break;
                        case '7': reproduir_posicions();
                              break;
                  }
            } while (int(opcio)!=27);    /*tecla ESCAPE*/
      }
      clrscr();
}


/*********************************************************************/
void dades_menu_articulacions (void)
{
      textbackground(GREEN);
      textcolor(BLACK);
      gotoxy(25,2);
      cprintf(" ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ» ");
      gotoxy(25,3);
      cprintf(" º   OPERAR PER ARTICULACIONS   º ");
      gotoxy(25,4);
      cprintf(" ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼ ");
      finestra2(26,9,56,16,BROWN,WHITE);
      gotoxy(30,10);
      printf("1) Utilitzar el JOYSTICK");
      gotoxy(30,12);
      printf("2) Utilitzar les tecles");
      gotoxy(34,15);
      printf("ESC) Menu anterior");
      angle_actual();
      pos_actual();
      comunicacions();
}
```

```
/*****************************************************************/
void menu_articulacions (void)
{
      char opcio;

      clrscr();
      articulars=SI;
      do {
            dades_menu_articulacions();
            while (!kbhit());
            opcio=getch();

            switch (opcio) {
                  case '1': articulacions_joy();
                        break;
                  case '2': if (menu_home_ok)
                                    articulacions_tecla_home();
                              else articulacions_tecla();
                        break;
            }
      } while (int(opcio)!=27);    /*tecla ESCAPE*/
      articulars=NO;
      clrscr();
}


/*****************************************************************/
void dades_articulacions_joy (void)
{
      textbackground(GREEN);
      textcolor(BLACK);
      gotoxy(25,2);
      cprintf(" ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ» ");
      gotoxy(25,3);
      cprintf(" º   OPERAR PER ARTICULACIONS   º ");
      gotoxy(25,4);
      cprintf(" ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼ ");

      if (menu_home_ok) finestra2(21,6,61,20,BROWN,WHITE);
      else finestra2(21,6,61,22,BROWN,WHITE);
      gotoxy(25,7);
      printf("® ¯) Moure la base");
      gotoxy(25,9);
      printf("_ _) Moure l'hombru");
      gotoxy(25,11);
      printf("_ _ + B0) Moure el colze");
      gotoxy(25,13);
      printf("_ _ + B1) Moure el canell (pitch)");
      gotoxy(25,15);
      printf("® ¯ + B1) Rotar el canell (roll)");
      gotoxy(25,17);
      printf("® ¯ + B0+B1) Obrir/Tancar la pinsa");
      if (menu_home_ok) {
            gotoxy(34,19);
            printf("ESC) Menu anterior");
      } else {
            gotoxy(30,19);
            printf("INTRO) Memortizar posicio");
            gotoxy(34,21);
            printf("ESC) Menu anterior");
      }
```

```
      angle_actual();
      pos_actual();
      comunicacions();
}


/*********************************************************************/
void articulacions_joy (void)
{
      char opcio;

      if (!joy_calibrat)
            msg_error(JOYSTICK);
      else {
            clrscr();
            do {
                  dades_articulacions_joy();
                  while (!kbhit()) {
                        if (menu_home_ok) joy_articulacions_home();
                        else joy_articulacions();
                  }
                  opcio=getch();
                  if (opcio==(char(13)) && !menu_home_ok)
                        grabar_posicio();
            } while (int(opcio)!=27);    /*tecla ESCAPE*/
      }
      clrscr();
}


/*********************************************************************/
void dades_articulacions_tecla (void)
{
      textbackground(GREEN);
      textcolor(BLACK);
      gotoxy(25,2);
      cprintf(" ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ» ");
      gotoxy(25,3);
      cprintf(" º   OPERAR PER ARTICULACIONS   º ");
      gotoxy(25,4);
      cprintf(" ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼ ");

      if (menu_home_ok) finestra2(19,7,61,18,BROWN,WHITE);
      else finestra2(19,7,61,20,BROWN,WHITE);
      gotoxy(23,8);
      printf("q) Base dreta");
      gotoxy(23,9);
      printf("a) Base esquerra");
      gotoxy(23,11);
      printf("w) Hombro amunt");
      gotoxy(23,12);
      printf("s) Hombro avall");
      gotoxy(23,14);
      printf("e) Colze amunt");
      gotoxy(23,15);
      printf("d) Colze avall");
      gotoxy(42,8);
      printf("r) Canell amunt");
      gotoxy(42,9);
      printf("f) Canell avall");
      gotoxy(42,11);
```

```
      printf("t) Canell dreta");
      gotoxy(42,12);
      printf("g) Canell esquerra");
      gotoxy(42,14);
      printf("y) Obrir pinsa");
      gotoxy(42,15);
      printf("h) Tancar pinsa");
      if (menu_home_ok) {
            gotoxy(30,17);
            printf("ESC) Menu anterior");
      } else {
            gotoxy(28,17);
            printf("INTRO) Memoritzar posicio");
            gotoxy(32,19);
            printf("ESC) Menu anterior");
      }

      angle_actual();
      pos_actual();
      comunicacions();
}


/*****************************************************************/
void articulacions_tecla_home (void)
{
      char opcio;

      clrscr();
      do {
            dades_articulacions_tecla();
            while (!kbhit());
            opcio=getch();

            switch (opcio) {
                  case 'q': moure_base ('+');
                        break;
                  case 'a': moure_base ('-');
                        break;
                  case 'w': moure_hombru ('-');
                        break;
                  case 's': moure_hombru ('+');
                        break;
                  case 'e': moure_colze ('+');
                        break;
                  case 'd': moure_colze ('-');
                        break;
                  case 'r': moure_canell('+');
                        break;
                  case 'f': moure_canell('-');
                        break;
                  case 't': rotar_canell('-');
                        break;
                  case 'g': rotar_canell('+');
                        break;
                  case 'y': obrir_tancar_pinsa('-');
                        break;
                  case 'h': obrir_tancar_pinsa('+');
                        break;
            }
      } while (int(opcio)!=27);    /*tecla ESCAPE*/
```

```
      clrscr();
}


/*****************************************************************/
void articulacions_tecla (void)
{
      char opcio;

      clrscr();
      do {
            dades_articulacions_tecla();
            while (!kbhit());
            opcio=getch();

            switch (opcio) {
                  case 'q': directa (MOT1,'+');
                        break;
                  case 'a': directa (MOT1,'-');
                        break;
                  case 'w': directa (MOT2,'-');
                        break;
                  case 's': directa (MOT2,'+');
                        break;
                  case 'e': directa (MOT3,'+');
                        break;
                  case 'd': directa (MOT3,'-');
                        break;
                  case 'r': directa (MOT4,'+');
                        break;
                  case 'f': directa (MOT4,'-');
                        break;
                  case 't': directa (MOT5,'-');
                        break;
                  case 'g': directa (MOT5,'+');
                        break;
                  case 'y': obrir_tancar_pinsa('-');
                        break;
                  case 'h': obrir_tancar_pinsa('+');
                        break;
                  case char(13): grabar_posicio();
                        break;
            }
      } while (int(opcio)!=27);    /*tecla ESCAPE*/
      clrscr();
}


/*****************************************************************/
void dades_menu_tcp (void)
{
      textbackground(GREEN);
      textcolor(BLACK);
      gotoxy(26,2);
      cprintf(" ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ» ");
      gotoxy(26,3);
      cprintf(" º      MOVIMENT DEL TCP      º ");
      gotoxy(26,4);
      cprintf(" ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼ ");
      finestra2(26,9,56,16,BROWN,WHITE);
      gotoxy(30,10);
```

```
      printf("1) Utilitzar el JOYSTICK");
      gotoxy(30,12);
      printf("2) Utilitzar les tecles");
      gotoxy(34,15);
      printf("ESC) Menu anterior");
      angle_actual();
      pos_actual();
      comunicacions();
}



/********************* MOVIMENT CARTESIA *****************/
void menu_tcp (void)
{
      char opcio;

      clrscr();
      tcp=SI;

      do {
            dades_menu_tcp();
            while (!kbhit());
            opcio=getch();

            switch (opcio) {
                  case '1': tcp_joy();
                        break;
                  case '2': tcp_tecla();
                        break;
            }
      } while (int(opcio)!=27);    /*tecla ESCAPE*/
      tcp=NO;
      clrscr();
}


/*********************************************************************/
void dades_tcp_joy (void)
{
      textbackground(GREEN);
      textcolor(BLACK);
      gotoxy(26,2);
      cprintf(" ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ» ");
      gotoxy(26,3);
      cprintf(" º      MOVIMENT DEL TCP      º ");
      gotoxy(26,4);
      cprintf(" ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼ ");
      finestra2(19,6,62,22,BROWN,WHITE);
      gotoxy(23,7);
      printf("® ¯) Per despla‡ar-se sobre l'eix X");
      gotoxy(23,9);
      printf("_ _) Per despla‡ar-se sobre l'eix Y");
      gotoxy(23,11);
      printf("_ _ + B0) Per despla‡ar-se sobre eix Z");
      gotoxy(23,13);
      printf("_ _ + B1) Per moure el canell (pitch)");
      gotoxy(23,15);
      printf("® ¯ + B1) Per rotar el canell (roll)");
      gotoxy(23,17);
      printf("® ¯ + B0+B1) Obrir/Tancar la pinsa");
```

```
      gotoxy(29,19);
      printf("INTRO) Memoritzar posicio");
      gotoxy(32,21);
      printf("ESC) Menu anterior");
      angle_actual();
      pos_actual();
      comunicacions();
}


/*********************************************************************/
void tcp_joy (void)
{
      char opcio;

      if (!joy_calibrat)
            msg_error(JOYSTICK);
      else {
            clrscr();
            do {
                  dades_tcp_joy();
                  while (!kbhit())
                        joy_tcp();
                  opcio=getch();
                  if (opcio==(char(13))) grabar_posicio();
            } while (int(opcio)!=27);    /*tecla ESCAPE*/
      }
      clrscr();
}


/*********************************************************************/
void dades_tcp_tecla (void)
{
      textbackground(GREEN);
      textcolor(BLACK);
      gotoxy(26,2);
      cprintf(" ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ» ");
      gotoxy(26,3);
      cprintf(" º      MOVIMENT DEL TCP      º ");
      gotoxy(26,4);
      cprintf(" ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼ ");
      finestra2(19,6,61,22,BROWN,WHITE);
      gotoxy(23,7);
      printf("q/a) Per despla‡ar-se sobre l'eix X");
      gotoxy(23,9);
      printf("o/p) Per despla‡ar-se sobre l'eix Y");
      gotoxy(23,11);
      printf("e/d) Per despla‡ar-se sobre l'eix Z");
      gotoxy(23,13);
      printf("t/g) Per moure el canell (pitch)");
      gotoxy(23,15);
      printf("u/j) Per rotar el canell (roll)");
      gotoxy(23,17);
      printf("y/h) Obrir/Tancar la pinsa");
      gotoxy(28,19);
      printf("INTRO) Memoritzar posicio");
      gotoxy(32,21);
      printf("ESC) Menu anterior");
      angle_actual();
      pos_actual();
```

```
      comunicacions();
}


/*****************************************************************/
void tcp_tecla (void)
{
      char opcio;

      clrscr();
      do {
            dades_tcp_tecla();
            while (!kbhit());
            opcio=getch();
            switch (opcio) {
                  case 'q': eix_x('+');
                        break;
                  case 'a': eix_x('-');
                        break;
                  case 'o': eix_y('+');
                        break;
                  case 'p': eix_y('-');
                        break;
                  case 'e': P.z=P.z+inc_tcp;
                        break;
                  case 'd': P.z=P.z-inc_tcp;
                        break;
                  case 't': tcp2=SI;
                          beta=beta+(GRAU*num_grau);
                          angle_AB=angle_AB-(GRAU*num_grau);
                        break;
                  case 'g': tcp2=SI;
                          beta=beta-(GRAU*num_grau);
                          angle_AB=angle_AB+(GRAU*num_grau);
                        break;
                  case 'u': canell_tcp ('-');
                        break;
                  case 'j': canell_tcp ('+');
                        break;
                  case 'y': obrir_tancar_pinsa('-');
                        break;
                  case 'h': obrir_tancar_pinsa('+');
                        break;
                  case char(13): grabar_posicio();
                        break;
            }
            if (opcio!='m' && int(opcio)!=27) {
                  inversa();
                  posicio();
                  if (traject) grabar_posicio();
            }
      } while (int(opcio)!=27);    /*tecla ESCAPE*/

      clrscr();
}
```

```
/*****************************************************************/
void llistat_posicions (void)
{
      char opcio;
      int linea=0, i=0, j=0, cursor=0, max=NO;

      textbackground(BLUE);
      textcolor(WHITE);

      parametres=NO;

      clrscr();
      textbackground(GREEN);
      textcolor(BLACK);
      gotoxy(26,2);
      cprintf(" ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ» ");
      gotoxy(26,3);
      cprintf(" º   POSICIONS MEMORITZADES   º ");
      gotoxy(26,4);
      cprintf(" ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼ ");
      textbackground(CYAN);
      textcolor(BLACK);
      gotoxy(1,6);
      cprintf(" _) Pujar   _) Baixar    ESC->Sortir   INTRO->Anar a
                                        posicio                  ");
      gotoxy(64,6);
      cprintf("n§ pos.: %d",num_pos);
      textbackground(CYAN);
      textcolor(RED);
      gotoxy(2,6);
      cprintf("_");
      gotoxy(13,6);
      cprintf("_");
      gotoxy(26,6);
      cprintf("ESC");
      gotoxy(40,6);
      cprintf("INTRO");

      pos_actual();

      finestra(1,7,80,24,BLACK,WHITE);
      gotoxy(36,7);
      printf(" POSICIONS ");
      textbackground(RED);
      textcolor(BLACK);
      gotoxy(2,8);
      cprintf("Posicio        X          Y          Z
                                 Angle AB        Angle ED   ");

      textcolor(WHITE);
      textbackground(BLACK);
      do {
            linea=0;
            j=i;
            while (j<num_pos && linea<15) {
                  if (linea==cursor-i) {
                        textcolor(BLACK);
                        textbackground(WHITE);
                        gotoxy(2,9+linea);
                        cprintf("                                    ");
                  } else {
```

```
                        textcolor(WHITE);
                        textbackground(BLACK);
                        gotoxy(2,9+linea);
                        cprintf("                                        ");
                  }
                  gotoxy(2,9+linea);
                  cprintf("Pos %d:   ",j);
                  gotoxy(14,9+linea);
                  cprintf("%7.2lf",pos[j].p.x);
                  gotoxy(25,9+linea);
                  cprintf("%7.2lf",pos[j].p.y);
                  gotoxy(36,9+linea);
                  cprintf("%7.2lf",pos[j].p.z);
                  gotoxy(54,9+linea);
                  cprintf("%4.1lf§   ",pos[j].ab*180/PI);
                  gotoxy(70,9+linea);
                  cprintf("%4.1lf§   ",pos[j].ed*180/PI);
                  j++;
                  linea++;
            }
            if (j==num_pos) max=SI;
            else max=NO;
            while (!kbhit());
            opcio=getch();
            switch (opcio) {
                  case char(72): if (cursor>0) {
                                    cursor--;
                                    if (cursor-i<0 && i>0) i--;
                                    }

                                 break;
                  case char(80): if (cursor<num_pos-1) {
                                       cursor++;
                                       if (cursor>=15 && !max)
                                             i++;
                                    }
                                 break;
                  case char(13): anar_posicio(cursor);
                                 break;
            }
      } while (int(opcio)!=27);

      parametres=SI;
      textbackground(BLUE);
      textcolor(WHITE);
      clrscr();
}


/*****************************************************************/
void grabar_posicio (void)
{
      if (!home_ok) {
            pos[0].p.x=P.x;
            pos[0].p.y=P.y;
            pos[0].p.z=P.z;
            pos[0].a.x=a.x;
            pos[0].a.y=a.y;
            pos[0].a.z=a.z;
            pos[0].o.x=o.x;
            pos[0].o.y=o.y;
```

```
            pos[0].o.z=o.z;
            pos[0].ab=angle_AB;
            pos[0].ed=angle_ED;
            gotoxy(10,23);
            textattr(64);
            cprintf("     ²²²²² Posicio 'Pos 0' memoritzada ²²²²   ");
            delay(1000);
            if (num_pos==0) num_pos++;
            window(1,1,80,25);
            textbackground(BLUE);
            textcolor(WHITE);
            clrscr();
      } else {
            pos[num_pos].p.x=P.x;
            pos[num_pos].p.y=P.y;
            pos[num_pos].p.z=P.z;
            pos[num_pos].a.x=a.x;
            pos[num_pos].a.y=a.y;
            pos[num_pos].a.z=a.z;
            pos[num_pos].o.x=o.x;
            pos[num_pos].o.y=o.y;
            pos[num_pos].o.z=o.z;
            pos[num_pos].ab=angle_AB;
            pos[num_pos].ed=angle_ED;
            num_pos++;
            if (!traject) {
                  gotoxy(10,23);
                  textattr(64);
                  cprintf("²² Posicio 'Pos %d' memoritzada ²",num_pos);
                  delay(1000);
                  window(1,1,80,25);
                  textbackground(BLUE);
                  textcolor(WHITE);
                  clrscr();
            }
      }
}


/*****************************************************************/
void reproduir_posicions (void)
{
      int j;
      char c;

      clrscr();
      textbackground(GREEN);
      textcolor(BLACK);
      gotoxy(16,2);
      cprintf(" ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ» ");
      gotoxy(16,3);
      cprintf(" º   REPRODUCCIO DE LES POSICIONS MEMORITZADES   º ");
      gotoxy(16,4);
      cprintf(" ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼ ");
      pos_actual();
      angle_actual();
      comunicacions();

      for (j=0; j<num_pos; j++)    /*recorrem totes les posicions
                                          memoritzades*/
            anar_posicio(j);
```

```
      finestra2(23,11,57,13,BROWN,WHITE);
      gotoxy(27,12);
      printf("Prem una tecla per continuar");
      while(!kbhit());
      c=getch();
      window(1,1,80,25);
      textbackground(BLUE);
      textcolor(WHITE);
      clrscr();
}

/*****************************************************************/
void dades_menu_executar (void)
{
      textbackground(GREEN);
      textcolor(BLACK);
      gotoxy(30,2);
      cprintf(" ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ» ");
      gotoxy(30,3);
      cprintf(" º   MENU EXECUTAR   º ");
      gotoxy(30,4);
      cprintf(" ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼ ");
      finestra2(23,8,58,17,BROWN,WHITE);
      gotoxy(27,9);
      printf("1) Executar 1 sol cop");
      gotoxy(27,11);
      printf("2) Executar de forma continua");
      gotoxy(27,13);
      printf("3) Executar pas a pas");
      gotoxy(32,16);
      printf("ESC) Menu anterior");
      angle_actual();
      pos_actual();
      comunicacions();
}

/*****************************************************************/
void menu_executar (void)
{
      char opcio;

      if (!executa)
            msg_error(EXEC);
      else {
            execucio=SI;
            clrscr();
            do {
                  dades_menu_executar();
                  while (!kbhit());
                  opcio=getch();

                  switch (opcio) {
                        case '1': executa_un_cop();
                              break;
                        case '2': executa_cont();
                              break;
                        case '3': executa_pas();
```

```
                              break;
                        }
                  } while (int(opcio)!=27);    /*tecla ESCAPE*/
                  execucio=NO;
            }
            clrscr();
}


/*****************************************************************/
void dades_menu_fitxers (void)
{
      textbackground(GREEN);
      textcolor(BLACK);
      gotoxy(30,2);
      cprintf(" ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ» ");
      gotoxy(30,3);
      cprintf(" º   MENU FITXERS   º ");
      gotoxy(30,4);
      cprintf(" ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼ ");
      finestra2(20,8,61,19,BROWN,WHITE);
      gotoxy(24,9);
      printf("1) Carregar un fitxer de posicions");
      gotoxy(24,11);
      printf("2) Carregar un fitxer de programa");
      gotoxy(24,13);
      printf("3) Guardar un fitxer de posicions");
      gotoxy(24,15);
      printf("4) Guardar un fitxer de programa");
      gotoxy(30,18);
      printf("ESC) Menu anterior");

      if (!home_ok) {
            gotoxy(15,23);
            textattr(192);
            cprintf("          El robot NO es a HOME !!!!!          ");
            textbackground(BLUE);
            textcolor(WHITE);
      }

      angle_actual();
      pos_actual();
      comunicacions();
}


/*****************************************************************/
void menu_fitxers (void)
{
      char opcio;

      clrscr();
      do {
            dades_menu_fitxers();
            while (!kbhit());
            opcio=getch();

            switch (opcio) {
                  case '1': carrega_fitxer_pos();
                        break;
                  case '2': carrega_fitxer_pgm();
```

```
                        break;
                  case '3': guarda_fitxer_pos();
                        break;
                  case '4': guarda_fitxer_pgm();
                        break;
            }
      } while (int(opcio)!=27);    /*tecla ESCAPE*/
      clrscr();
}


/*************************************************************/
void increment_tcp (void)
{
      float increment, inc_grau;

      clrscr();
      textbackground(GREEN);
      textcolor(BLACK);
      gotoxy(23,2);
      cprintf(" ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ» ");
      gotoxy(23,3);
      cprintf(" º      INCREMENT DEL TCP      º ");
      gotoxy(23,4);
      cprintf(" ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼ ");
      finestra2(14,8,63,14,BROWN,WHITE);
      gotoxy(17,9);
      printf("Indica la quantitat d'increment en la posici¢");
      gotoxy(17,10);
      printf("del TCP cada cop que es premi una tecla de");
      gotoxy(17,11);
      printf("despla‡ament (10-50 mm).");

      _setcursortype(_NORMALCURSOR);
      do {
            gotoxy(20,13);
            printf("Valor actual: %0.2f    Quantitat?:     ",inc_tcp);
            gotoxy(55,13);
            scanf("%f",&increment);
      } while (increment<10.0 || increment>50.0);
      inc_tcp=increment;

      finestra2(14,16,63,21,BROWN,WHITE);
      gotoxy(17,17);
      printf("Indica els graus de resolucio del movinent");
      gotoxy(17,18);
      printf("de Pitch i Roll  (1-10 §)");
      do {
            gotoxy(20,20);
            printf("Valor actual: %0.2f     Quantitat?:    ",num_grau);
            gotoxy(55,20);
            scanf("%f",&inc_grau);
      } while (inc_grau<1.0 || inc_grau>10.0);
      num_grau=inc_grau;

      _setcursortype(_NOCURSOR);
      textbackground(BLUE);
      textcolor(WHITE);
      clrscr();
}
```

```
/*************************************************************/
void increment_polsos (void)
{
      int pol;

      clrscr();
      textbackground(GREEN);
      textcolor(BLACK);
      gotoxy(23,2);
      cprintf(" ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ» ");
      gotoxy(23,3);
      cprintf(" º   POLSOS DEL MOVIMENT CARTESIA   º ");
      gotoxy(23,4);
      cprintf(" ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼ ");

      finestra2(13,10,67,15,BROWN,WHITE);
      gotoxy(16,11);
      printf("Indica la quantitat de polsos que s'enviaran cada");
      gotoxy(16,12);
      printf("cop que es premi una tecla de despla‡ament (10-50)");

      _setcursortype(_NORMALCURSOR);
      do {
            gotoxy(23,14);
            printf("Valor actual: %0.0f      Quantitat?:     ",polsos);
            gotoxy(56,14);
            scanf("%d",&pol);
      } while (polsos<10 || pol>50);
      polsos=pol;
      _setcursortype(_NOCURSOR);
      textbackground(BLUE);
      textcolor(WHITE);
      clrscr();
}



/*****************************************************************/
void canell_tcp (char increment)
{
      double nou_z5;

           /*increment o decremet de z5*/
      if (increment=='+') {
            nou_z5=z5+(GRAU*num_grau);
            angle_ED=angle_ED+(GRAU*num_grau);
      } else {
            nou_z5=z5-(GRAU*num_grau);
            angle_ED=angle_ED-(GRAU*num_grau);
      }
            /*noves coordenades del vecotr orientacio*/
      o.x=(-(cos(z1)*cos(z2+z3+z4)*sin(nou_z5))-
(sin(z1)*cos(nou_z5)));
      o.y=(-
(sin(z1)*cos(z2+z3+z4)*sin(nou_z5))+(cos(z1)*cos(nou_z5)));
      o.z=sin(z2+z3+z4)*sin(nou_z5);
}
```

```
/*****************************************************************/
void eix_x (char increment)
{
      coordenades p1, p2, a1, o1;
      double modul_p1, modul_p2, alfa, numerador, denominador;
      double m_rot_z [3][3];

      a1.x=a.x;       /* copia del vector apropament */
      a1.y=a.y;
      a1.z=a.z;

      o1.x=o.x;       /*copia del vector orientacio */
      o1.y=o.y;
      o1.z=o.z;

      p1.x=P.x;       /*antic vector posicio */
      p1.y=P.y;
      p1.z=P.z;

      if (increment=='+')
            P.x=P.x+inc_tcp;
      else
            P.x=P.x-inc_tcp;

      p2.x=P.x;   /* nou vector posicio, nomes varia la component x */
      p2.y=P.y;
      p2.z=P.z;

      numerador=(p1.x*p2.x)+(p1.y*p2.y);     /* producte escalar */
      denominador=(modul(p1))*(modul(p2));

      alfa=acos(numerador/denominador);      /* angle entre els 2
                                                      vectors */

      if (increment=='+')
            alfa=-alfa;

      m_rot_z[0][0]=cos(alfa); /* matriu de rotacio alfa graus sobre
                                                      l'eix Z */
      m_rot_z[0][1]=-sin(alfa);
      m_rot_z[0][2]=0;
      m_rot_z[1][0]=sin(alfa);
      m_rot_z[1][1]=cos(alfa);
      m_rot_z[1][2]=0;
      m_rot_z[2][0]=0;
      m_rot_z[2][1]=0;
      m_rot_z[2][2]=1;

      /* noves components del vector apropament */
      a.x=(m_rot_z[0][0]*a1.x)+(m_rot_z[0][1]*a1.y)+
                                          (m_rot_z[0][2]*a1.z);
      a.y=(m_rot_z[1][0]*a1.x)+(m_rot_z[1][1]*a1.y)+
                                          (m_rot_z[1][2]*a1.z);
      a.z=(m_rot_z[2][0]*a1.x)+(m_rot_z[2][1]*a1.y)+
                                          (m_rot_z[2][2]*a1.z);

      if (a.x<0.000000001 && a.x>-0.000000001 && a.y<0.000000001 &&
                                          a.y>-0.000000001) {
            mantenir_orientacio=SI;
      } else {
```

```
      /* noves components del vector orientacio */
      o.x=(m_rot_z[0][0]*o1.x)+(m_rot_z[0][1]*o1.y)+
                                                (m_rot_z[0][2]*o1.z);
      o.y=(m_rot_z[1][0]*o1.x)+(m_rot_z[1][1]*o1.y)+
                                                (m_rot_z[1][2]*o1.z);
      o.z=(m_rot_z[2][0]*o1.x)+(m_rot_z[2][1]*o1.y)+
                                                (m_rot_z[2][2]*o1.z);
      }
}


/*****************************************************************/
void eix_y (char increment)
{
      coordenades p1, p2, a1, o1;
      double modul_p1, modul_p2, alfa, numerador, denominador;
      double m_rot_z [3][3];


      a1.x=a.x;       /* copia del vector apropament */
      a1.y=a.y;
      a1.z=a.z;

      o1.x=o.x;       /*copia del vector orientacio */
      o1.y=o.y;
      o1.z=o.z;

      p1.x=P.x;       /*antic vector posicio */
      p1.y=P.y;
      p1.z=P.z;

      if (increment=='+')
            P.y=P.y+inc_tcp;
      else
            P.y=P.y-inc_tcp;

      p2.x=P.x;   /* nou vector posicio, nomes varia la component y */
      p2.y=P.y;
      p2.z=P.z;

      numerador=(p1.x*p2.x)+(p1.y*p2.y);     /* producte escalar */
      denominador=(modul(p1))*(modul(p2));

      alfa=acos(numerador/denominador);      /* angle entre els 2
                                                      vectors */

      if (increment=='-')
            alfa=-alfa;

      m_rot_z[0][0]=cos(alfa); /* matriu de rotacio alfa graus sobre
                                                        l'eix Z */
      m_rot_z[0][1]=-sin(alfa);
      m_rot_z[0][2]=0;
      m_rot_z[1][0]=sin(alfa);
      m_rot_z[1][1]=cos(alfa);
      m_rot_z[1][2]=0;
      m_rot_z[2][0]=0;
      m_rot_z[2][1]=0;
      m_rot_z[2][2]=1;

      /* noves components del vector apropament */
```

```
      a.x=(m_rot_z[0][0]*a1.x)+(m_rot_z[0][1]*a1.y)+
                                                (m_rot_z[0][2]*a1.z);
      a.y=(m_rot_z[1][0]*a1.x)+(m_rot_z[1][1]*a1.y)+
                                                (m_rot_z[1][2]*a1.z);
      a.z=(m_rot_z[2][0]*a1.x)+(m_rot_z[2][1]*a1.y)+
                                                (m_rot_z[2][2]*a1.z);

      if (a.x<0.000000001 && a.x>-0.000000001 && a.y<0.000000001 &&
                                                a.y>-0.000000001) {
            mantenir_orientacio=SI;
      } else {
      /* noves components del vector orientacio */
      o.x=(m_rot_z[0][0]*o1.x)+(m_rot_z[0][1]*o1.y)+
                                                (m_rot_z[0][2]*o1.z);
      o.y=(m_rot_z[1][0]*o1.x)+(m_rot_z[1][1]*o1.y)+
                                                (m_rot_z[1][2]*o1.z);
      o.z=(m_rot_z[2][0]*o1.x)+(m_rot_z[2][1]*o1.y)+
                                                (m_rot_z[2][2]*o1.z);
      }
}


/*****************************************************************/
double modul (coordenades m)
{
      double mod;

      mod=sqrt((m.x*m.x)+(m.y*m.y));
      return mod;
}


/********************** Calibrar Joystick ************************/
void calibrar_joy (void)
{
      clrscr();
      dades_calibrar_joy();
      dades_joy();
      joy_calibrat=SI;
      clrscr();
}


/*****************************************************************/
void dades_joy (void)
{
      char c;

      clrscr();
      refresc_joy=SI;

      while (!kbhit()) {
            joystick();
            posicio_cursor();
            if (refresc) {
                  finestra(8,14,22,20,WHITE,BLACK);
                  refresc=NO;
                  gotoxy(10,15);
                  printf("Eix X: %04d",eix[0]);
                  gotoxy(10,16);
                  printf("Eix Y: %04d",eix[1]);
```

```
                gotoxy(10,18);
                printf("Boto 0: %d",bot[0]);
                gotoxy(10,19);
                printf("Boto 1: %d",bot[1]);
                gotoxy(1,1);
                printf("%04d",calibra[0]);
                gotoxy(1,2);
                printf("%04d",calibra[1]);
                gotoxy(39,11);
                printf("%04d",calibra[2]);
                gotoxy(39,12);
                printf("%04d",calibra[3]);
                gotoxy(76,24);
                printf("%04d",calibra[4]);
                gotoxy(76,25);
                printf("%04d",calibra[5]);
                gotoxy(5,22);
                printf("Prem qualsevol tecla per sortir");
          }
      }
      refresc_joy=NO;
      c=getch();
      textbackground(BLUE);
      textcolor(WHITE);

}


/******** Funcio que obte les dades del joystick ***************/
void joystick (void)
{
      int i, j;
      int mask=15;

      outportb(JPORT,0);      /* disparem el CI 558*/
      disable();
      for (i=1; mask && i<CMAX; i++) {      /*contem el temps*/
            j=inportb(JPORT)^mask;
            if (j & 1) {eix[0]=i; mask ^=1;}
            if (j & 2) {eix[1]=i; mask ^=2;}
      }
      enable();
      j=inportb(JPORT);
      bot[0]=!(j&0x10);           /*llegim l'estat dels botons*/
      bot[1]=!(j&0x20);
}


/****************** Calibracio del joystick ******************/
/****** els valors es guarden en un vector calibra[6] *************/
void dades_calibrar_joy (void)
{
      int ok=NO, ok1=NO;

      textbackground(GREEN);
      textcolor(BLACK);
      gotoxy(24,2);
      cprintf(" ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ» ");
      gotoxy(24,3);
      cprintf(" º   CALIBRACIO DEL JOYSTICK   º ");
      gotoxy(24,4);
```

```
      cprintf(" È͹͹͹͹͹͹͹͹͹͹͹͹͹͹͹͹͹͹͹͹͹¼ ");

      ok=NO;
      finestra2(4,9,76,11,BROWN,WHITE);
      gotoxy(8,10);
      printf("Coloca el joystick en la posicio ADALT - ESQUERRA i prem
                                                      el boto 0");
      while (!ok) {
            if (bot[0]==1) ok1=SI;
            if (bot[0]==0 && ok1) {
                  ok=SI;
                  ok1=NO;
            }
            joystick();
      }
      ok=NO;
      calibra[0]=eix[0];
      calibra[1]=eix[1];
      finestra2(4,14,76,16,BROWN,WHITE);
      gotoxy(13,15);
      printf("Coloca el joystick en la posicio CENTRAL i prem el
                                                         boto 0");
      while (!ok) {
            if (bot[0]==1) ok1=SI;
            if (bot[0]==0 && ok1) {
                  ok=SI;
                  ok1=NO;
            }
            joystick();
      }
      ok=NO;
      calibra[2]=eix[0];
      calibra[3]=eix[1];
      finestra2(4,19,76,21,BROWN,WHITE);
      gotoxy(10,20);
      printf("Coloca el joystick en la posicio ABAIX - DRETA i prem el
                                                         boto 0");
      while (!ok) {
            if (bot[0]==1) ok1=SI;
            if (bot[0]==0 && ok1) {
                  ok=SI;
                  ok1=NO;
            }
            joystick();
      }
      calibra[4]=eix[0];
      calibra[5]=eix[1];

      inc_x_esq=(calibra[2]-calibra[0])/4;     /*escalat dels eixos*/
      inc_x_dre=(calibra[4]-calibra[2])/4;
      inc_y_dalt=(calibra[3]-calibra[1])/4;
      inc_y_baix=(calibra[5]-calibra[3])/4;
      delay(100);
      textbackground(BLUE);
      textcolor(WHITE);
      clrscr();
}
```

```
/************** Posicio del cursor en la pantalla *****************/
void posicio_cursor (void)
{
      if (inici) {
            gotoxy(35,12);
            printf("Û");
            inici=NO;
            x=wherex()-1;
            y=wherey();
      }

      /******* moviment esquerra *****************/
      if (eix[0]<(calibra[2]-(3*inc_x_esq))) {
            if (x>1) {
                  gotoxy(x,y);
                  printf(" ");
                  x=x-1;
            }
      }

      /****************** moviment dreta ******************/
      else if (eix[0]>(calibra[2]+(3*inc_x_dre))) {
            if (x<79) {
                  gotoxy(x,y);
                  printf(" ");
                  x=x+1;
            }
      }

      /*************** moviment amunt ******************/
      else if (eix[1]<(calibra[3]-(3*inc_y_dalt))) {
            if (y>1) {
                  gotoxy(x,y);
                  printf(" ");
                  y=y-1;
            }
      }

      /*************** moviment avall **************/
      else if (eix[1]>(calibra[3]+(3*inc_y_baix))) {
            if (y<25) {
                  gotoxy(x,y);
                  printf(" ");
                  y=y+1;
            }
      }
      gotoxy(x,y);
      printf("Û");
      delay(30);
}



/*****************************************************************/
void joy_articulacions_home (void)
{
      joystick();

      /*********** Moviment de la base (eix X) ********************/
      if ((eix[0]<(calibra[2]-(3*inc_x_esq))) && bot[0]==0 &&
                                                      bot[1]==0)
```

```
            moure_base('+');
      if ((eix[0]>(calibra[2]+(3*inc_x_dre))) && bot[0]==0 &&
                                                      bot[1]==0)
            moure_base('-');

      /*********** Moviment de l'hombru (eix Y) ********************/
      if ((eix[1]<(calibra[3]-(3*inc_y_dalt))) && bot[0]==0 &&
                                                      bot[1]==0)
            moure_hombru('-');
      if ((eix[1]>(calibra[3]+(3*inc_y_baix))) && bot[0]==0 &&
                                                      bot[1]==0)
            moure_hombru('+');

      /*********** Moviment del colze (Eix Y + Boto 0) ************/
      if ((eix[1]<(calibra[3]-(3*inc_y_dalt))) && bot[0]==1 &&
                                                      bot[1]==0)
            moure_colze('+');
      if ((eix[1]>(calibra[3]+(3*inc_y_baix))) && bot[0]==1 &&
                                                      bot[1]==0)
            moure_colze('-');

      /************* Girar canell (Eix Y + Boto 1) ***************/
      if ((eix[1]<(calibra[3]-(3*inc_y_dalt))) && bot[0]==0 &&
                                                      bot[1]==1)
            moure_canell('+');
      if ((eix[1]>(calibra[3]+(3*inc_y_baix))) && bot[0]==0 &&
                                                      bot[1]==1)
            moure_canell('-');

      /************** Rotacio del canell  (Eix X + Boto 1)**********/
      if ((eix[0]<(calibra[2]-(3*inc_x_esq))) && bot[0]==0 &&
                                                      bot[1]==1)
            rotar_canell('-');
      if ((eix[0]>(calibra[2]+(3*inc_x_dre))) && bot[0]==0 &&
                                                      bot[1]==1)
            rotar_canell('+');

      /****** Obrir i tancar pinsa  (Eix X + Boto 0 + Boto 1) ******/
      if ((eix[0]<(calibra[2]-(3*inc_x_esq))) && bot[0]==1 &&
                                                      bot[1]==1)
            obrir_tancar_pinsa('+');
      if ((eix[0]>(calibra[2]+(3*inc_x_dre))) && bot[0]==1 &&
                                                      bot[1]==1)
            obrir_tancar_pinsa('-');
}


/*****************************************************************/
void joy_articulacions (void)
{
      joystick();

      /*********** Moviment de la base (eix X) ********************/
      if ((eix[0]<(calibra[2]-(3*inc_x_esq))) && bot[0]==0 &&
                                                      bot[1]==0)
            directa (MOT1,'+');
      if ((eix[0]>(calibra[2]+(3*inc_x_dre))) && bot[0]==0 &&
                                                      bot[1]==0)
            directa (MOT1,'-');
```

```
      /*********** Moviment de l'hombru (eix Y) ********************/
      if ((eix[1]<(calibra[3]-(3*inc_y_dalt))) && bot[0]==0 &&
                                                        bot[1]==0)
            directa (MOT2,'-');
      if ((eix[1]>(calibra[3]+(3*inc_y_baix))) && bot[0]==0 &&
                                                        bot[1]==0)
            directa (MOT2,'+');

      /********** Moviment del colze (Eix Y + Boto 0) *************/
      if ((eix[1]<(calibra[3]-(3*inc_y_dalt))) && bot[0]==1 &&
                                                        bot[1]==0)
            directa (MOT3,'+');
      if ((eix[1]>(calibra[3]+(3*inc_y_baix))) && bot[0]==1 &&
                                                        bot[1]==0)
            directa (MOT3,'-');

      /************ Girar canell (Eix Y + Boto 1) ***************/
      if ((eix[1]<(calibra[3]-(3*inc_y_dalt))) && bot[0]==0 &&
                                                        bot[1]==1)
            directa (MOT4,'+');
      if ((eix[1]>(calibra[3]+(3*inc_y_baix))) && bot[0]==0 &&
                                                        bot[1]==1)
            directa (MOT4,'-');

      /*********** Rotacio del canell  (Eix X + Boto 1)*************/
      if ((eix[0]<(calibra[2]-(3*inc_x_esq))) && bot[0]==0 &&
                                                        bot[1]==1)
            directa (MOT5,'-');
      if ((eix[0]>(calibra[2]+(3*inc_x_dre))) && bot[0]==0 &&
                                                        bot[1]==1)
            directa (MOT5,'+');

      /********** Obrir i tancar pinsa  (Eix X + Boto 0 + Boto 1) ***/
      if ((eix[0]<(calibra[2]-(3*inc_x_esq))) && bot[0]==1 &&
                                                        bot[1]==1)
            obrir_tancar_pinsa('+');
      if ((eix[0]>(calibra[2]+(3*inc_x_dre))) && bot[0]==1 &&
                                                        bot[1]==1)
            obrir_tancar_pinsa('-');
}



/*************************************************************/
void joy_tcp (void)
{
      joystick();

      /*********** Despla‡ament per l'eix X ********************/
      if ((eix[0]<(calibra[2]-(3*inc_x_esq))) && bot[0]==0 &&
                                                  bot[1]==0) {
            eix_x('-');
            inversa();
            posicio();
            if (traject) grabar_posicio();
      }
      if ((eix[0]>(calibra[2]+(3*inc_x_dre))) && bot[0]==0 &&
                                                  bot[1]==0) {
            eix_x('+');
            inversa();
            posicio();
```

```
      if (traject) grabar_posicio();
}

/*********** Despla‡ament per l'eix Y ********************/
if ((eix[1]<(calibra[3]-(3*inc_y_dalt))) && bot[0]==0 &&
                                                bot[1]==0) {
      eix_y('+');
      inversa();
      posicio();
      if (traject) grabar_posicio();
}
if ((eix[1]>(calibra[3]+(3*inc_y_baix))) && bot[0]==0 &&
                                                bot[1]==0) {
      eix_y('-');
      inversa();
      posicio();
      if (traject) grabar_posicio();
}

/*********** Despla‡ament per l'eix Z ********************/
if ((eix[1]<(calibra[3]-(3*inc_y_dalt))) && bot[0]==1 &&
                                                bot[1]==0) {
      P.z=P.z+inc_tcp;
      inversa();
      posicio();
      if (traject) grabar_posicio();
}
if ((eix[1]>(calibra[3]+(3*inc_y_baix))) && bot[0]==1 &&
                                                bot[1]==0) {
      P.z=P.z-inc_tcp;
      inversa();
      posicio();
      if (traject) grabar_posicio();
}

/************** Girar canell ******************************/
if ((eix[1]<(calibra[3]-(3*inc_y_dalt))) && bot[0]==0 &&
                                                bot[1]==1) {
      tcp2=SI;
      beta=beta+(GRAU*num_grau);
      angle_AB=angle_AB-(GRAU*num_grau);
      inversa();
      posicio();
      if (traject) grabar_posicio();
}
if ((eix[1]>(calibra[3]+(3*inc_y_baix))) && bot[0]==0 &&
                                                bot[1]==1) {
      tcp2=SI;
      beta=beta-(GRAU*num_grau);
      angle_AB=angle_AB+(GRAU*num_grau);
      inversa();
      posicio();
      if (traject) grabar_posicio();
}

/***************** Rotacio del canell *********************/
if ((eix[0]<(calibra[2]-(3*inc_x_esq))) && bot[0]==0 &&
                                                bot[1]==1) {
      canell_tcp('+');
      inversa();
      posicio();
```

```
            if (traject) grabar_posicio();
      }
      if ((eix[0]>(calibra[2]+(3*inc_x_dre))) && bot[0]==0 &&
                                                  bot[1]==1) {
            canell_tcp('-');
            inversa();
            posicio();
            if (traject) grabar_posicio();
      }

      /****************** Obrir i tancar pinsa *********************/
      if ((eix[0]<(calibra[2]-(3*inc_x_esq))) && bot[0]==1 &&
                                                    bot[1]==1)
            obrir_tancar_pinsa('+');
      if ((eix[0]>(calibra[2]+(3*inc_x_dre))) && bot[0]==1 &&
                                                    bot[1]==1)
            obrir_tancar_pinsa('-');
}


/*****************************************************************/
void dades_posicio (void)
{
      finestra2(26,10,50,18,BROWN,WHITE);
      _setcursortype(_NORMALCURSOR);
      gotoxy(29,11);
      printf("Valor de Px: ");
      scanf("%lf",&P.x);
      gotoxy(29,12);
      printf("Valor de Py: ");
      scanf("%lf",&P.y);
      gotoxy(29,13);
      printf("Valor de Pz: ");
      scanf("%lf",&P.z);
      gotoxy(29,15);
      printf("Valor del Pitch: ");
      scanf("%lf",&angle_AB);
      angle_AB=angle_AB*PI/180;
      gotoxy(29,17);
      printf("Valor del Roll: ");
      scanf("%lf",&angle_ED);
      angle_ED=angle_ED*PI/180;
      _setcursortype(_NOCURSOR);
      orientacio();
}


/*****************************************************************/
void dades_posicio_home (void)
{
      finestra2(20,9,58,21,BROWN,WHITE);
      _setcursortype(_NORMALCURSOR);
      gotoxy(30,10);
      printf("Valor de Px: ");
      scanf("%lf",&P.x);
      gotoxy(30,11);
      printf("Valor de Py: ");
      scanf("%lf",&P.y);
      gotoxy(30,12);
      printf("Valor de Pz: ");
      scanf("%lf",&P.z);
```

```
      gotoxy(24,14);
      printf("Valor de la component Ax: ");
      scanf("%lf",&a.x);
      gotoxy(24,15);
      printf("Valor de la component Ay: ");
      scanf("%lf",&a.y);
      gotoxy(24,16);
      printf("Valor de la component Az: ");
      scanf("%lf",&a.z);

      gotoxy(24,18);
      printf("Valor de la component Ox: ");
      scanf("%lf",&o.x);
      gotoxy(24,19);
      printf("Valor de la component Oy: ");
      scanf("%lf",&o.y);
      gotoxy(24,20);
      printf("Valor de la component Oz: ");
      scanf("%lf",&o.z);
      _setcursortype(_NOCURSOR);
}


/*****************************************************************/
void editar_pgm (void)
{
      int linea=0, i=0, inc_lin=0;
      char opcio;

      clrscr();
      do {
            textbackground(GREEN);
            textcolor(BLACK);
            gotoxy(28,2);
            cprintf(" ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ» ");
            gotoxy(28,3);
            cprintf(" º   EDITAR PROGRAMA   º ");
            gotoxy(28,4);
            cprintf(" ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼ ");

            textcolor(BLACK);
            textbackground(WHITE);          /*comandes de edicio*/
            gotoxy(53,6);
            cprintf("ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ»");
            gotoxy(53,7);
            cprintf("º Anar a posicio  º");
            gotoxy(53,8);
            cprintf("º Velocitat       º");
            gotoxy(53,9);
            cprintf("º Pausa           º");
            gotoxy(53,10);
            cprintf("º Obrir Pinsa     º");
            gotoxy(53,11);
            cprintf("º Tancar Pinsa    º");
            gotoxy(53,12);
            cprintf("º Borrar linea    º");
            gotoxy(53,13);
            cprintf("º Moure linea     º");
            gotoxy(53,14);
            cprintf("º Trajectoria     º");
```

```
gotoxy(53,15);
cprintf("ÌÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¹");
gotoxy(53,16);
cprintf("º _ pujar prog.   º");
gotoxy(53,17);
cprintf("º _ baixar prog.  º");
gotoxy(53,18);
cprintf("º ESC -> Sortir   º");
gotoxy(53,19);
cprintf("ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼");

textcolor(RED);
gotoxy(55,7);
cprintf("A");
gotoxy(55,8);
cprintf("V");
gotoxy(55,9);
cprintf("P");
gotoxy(55,10);
cprintf("O");
gotoxy(55,11);
cprintf("T");
gotoxy(55,12);
cprintf("B");
gotoxy(55,13);
cprintf("M");
gotoxy(58,14);
cprintf("j");
gotoxy(55,16);
cprintf("_");
gotoxy(55,17);
cprintf("_");
gotoxy(55,18);
cprintf("ESC");

finestra(10,6,42,23,BLACK,WHITE);
gotoxy(22,6);
printf(" PROGRAMA ");

finestra(50,21,74,23,BLACK,WHITE);
gotoxy(58,21);
printf(" COMANDA ");

linea=0;
i=num_lin-16+inc_lin;
if (i<=0) i=0;

while (i<num_lin && linea<16) {
      gotoxy(11,7+linea);
      printf("Lin %d",i);
      gotoxy(21,7+linea);
      printf("%s %d",pgm[i].comanda,pgm[i].quant);
      i++;
      linea++;
}
gotoxy(52,22);

while (!kbhit());
opcio=getch();
switch (opcio) {
      case 'a': editar_pos();
```

```
                        inc_lin=0;
                        break;
                case 'v': editar_vel();
                        inc_lin=0;
                        break;
                case 'p': editar_pausa();
                        inc_lin=0;
                        break;
                case 'o': editar_obrir_pinsa();
                        inc_lin=0;
                        break;
                case 't': editar_tancar_pinsa();
                        inc_lin=0;
                        break;
                case 'b': editar_borrar_linea();
                        inc_lin=0;
                        break;
                case 'm': editar_moure_linea();
                        inc_lin=0;
                        break;
                case 'j': editar_trajectoria();
                        break;
                case char(72): if (num_lin-16+inc_lin>0) inc_lin--;
                        break;
                case char(80): if (i<num_lin) inc_lin++;
                        break;
            }
      } while (int(opcio)!=27);    /*tecla ESCAPE*/
      if (num_lin>0) executa=SI;
      else executa=NO;

      textbackground(BLUE);
      textcolor(WHITE);
      clrscr();
}


/*********************************************************************
/
void executa_un_cop (void)
{
      char comanda[10];
      int quant, i;

      clrscr();
      textbackground(GREEN);
      textcolor(BLACK);
      gotoxy(26,2);
      cprintf(" ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ» ");
      gotoxy(26,3);
      cprintf(" º   EXECUTAR UN SOL COP   º ");
      gotoxy(26,4);
      cprintf(" ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼ ");

      finestra2(22,11,56,13,BROWN,WHITE);
      gotoxy(26,12);
      printf("El programa s'esta executant");

      for (i=0;i<num_lin;i++) {
            strcpy(comanda,pgm[i].comanda);
            quant=pgm[i].quant;
```

```
            operar_comanda(comanda,quant);
      }
      window(1,1,80,25);
      textbackground(BLUE);
      textcolor(WHITE);
      clrscr();
}


/*********************************************************************
/
void executa_cont (void)
{
      char comanda[10];
      int i,quant;

      clrscr();
      textbackground(GREEN);
      textcolor(BLACK);
      gotoxy(24,2);
      cprintf(" ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ» ");
      gotoxy(24,3);
      cprintf(" º   EXECUTAR EN MODE CONTINU   º ");
      gotoxy(24,4);
      cprintf(" ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼ ");


      finestra2(19,10,61,16,BROWN,WHITE);
      gotoxy(29,12);
      printf("Prem una tecla per parar");
      gotoxy(22,14);
      printf("El robot s'aturar… quan acabi el cicle");

      do {
            for (i=0;i<num_lin;i++) {
                  strcpy(comanda,pgm[i].comanda);
                  quant=pgm[i].quant;
                  operar_comanda(comanda,quant);
            }
      } while (!kbhit());

      window(1,1,80,25);
      textbackground(BLUE);
      textcolor(WHITE);
      clrscr();
}


/*********************************************************************
/
void executa_pas (void)
{
      int quant,i=0;
      char comanda[10],c;

      clrscr();
      textbackground(GREEN);
      textcolor(BLACK);
      gotoxy(26,2);
      cprintf(" ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ» ");
      gotoxy(26,3);
```

```
      cprintf(" º   EXECUTAR PAS A PAS   º ");
      gotoxy(26,4);
      cprintf(" ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼ ");


      while (i<num_lin) {
            finestra2(23,9,56,13,BROWN,WHITE);
            gotoxy(31,10);
            printf("Prem una tecla per");
            gotoxy(27,12);
            printf("executar la comanda seg•ent");
            while(!kbhit());
            c=getch();
            gotoxy(31,10);
            printf("                  ");
            gotoxy(27,12);
            printf("                              ");
            strcpy(comanda,pgm[i].comanda);
            quant=pgm[i].quant;
            operar_comanda(comanda,quant);
            i++;
      }

      window(1,1,80,25);
      textbackground(BLUE);
      textcolor(WHITE);
      clrscr();
}


/*****************************************************************/
void editar_pos (void)
{
      int n;
      char comanda[]="Posicio";

      _setcursortype(_NORMALCURSOR);
      printf("Posicio ");
      scanf("%d",&n);
      strcpy(pgm[num_lin].comanda,comanda);
      pgm[num_lin].quant=n;
      num_lin++;
      _setcursortype(_NOCURSOR);
}


/*****************************************************************/
void editar_trajectoria (void)
{
      int i;
      char comanda[]="Posicio";

      for (i=0; i<num_pos; i++) {
            strcpy(pgm[num_lin].comanda,comanda);
            pgm[num_lin].quant=i;
            num_lin++;
      }
}
```

```
/*****************************************************************/
void editar_vel (void)
{
      int n=0;
      char comanda[]="Velocitat";

      _setcursortype(_NORMALCURSOR);
      printf("Velocitat ");
      scanf("%d",&n);
      strcpy(pgm[num_lin].comanda,comanda);
      pgm[num_lin].quant=n;
      num_lin++;
      _setcursortype(_NOCURSOR);
}


/*****************************************************************/
void editar_pausa (void)
{
      int n=0;
      char comanda[]="Pausa";

      _setcursortype(_NORMALCURSOR);
      printf("Pausa ");
      scanf("%d",&n);
      strcpy(pgm[num_lin].comanda,comanda);
      pgm[num_lin].quant=n;
      num_lin++;
      _setcursortype(_NOCURSOR);
}


/*****************************************************************/
void editar_obrir_pinsa (void)
{
      char comanda[]="Obrir";

      _setcursortype(_NORMALCURSOR);
      printf("Obrir ");
      strcpy(pgm[num_lin].comanda,comanda);
      pgm[num_lin].quant=0;
      num_lin++;
      _setcursortype(_NOCURSOR);
}


/*****************************************************************/
void editar_tancar_pinsa (void)
{
      char comanda[]="Tancar";

      _setcursortype(_NORMALCURSOR);
      printf("Tancar ");
      strcpy(pgm[num_lin].comanda,comanda);
      pgm[num_lin].quant=0;
      num_lin++;
      _setcursortype(_NOCURSOR);
}
```

```
/*****************************************************************/
void editar_borrar_linea (void)
{
      int i, pos;

      _setcursortype(_NORMALCURSOR);
      printf("Borrar linea ");
      scanf("%d",&pos);
      if (pos>=0 && pos<num_lin) {
            for (i=pos; i<num_lin; i++) {
                  strcpy(pgm[i].comanda,pgm[i+1].comanda);
                  pgm[i].quant=pgm[i+1].quant;
            }
            num_lin--;
      }
      _setcursortype(_NOCURSOR);
}


/*****************************************************************/
void editar_moure_linea (void)
{
      int i=0, lin, nova_lin;
      estructura_pgm insertar;

      _setcursortype(_NORMALCURSOR);
      do {
            gotoxy(52,22);
            printf("Moure linea      ");
            gotoxy(64,22);
            scanf("%d",&lin);
      } while (lin<0 || lin>=num_lin);
      do {
            gotoxy(52,22);
            printf("a nova linea     ");
            gotoxy(65,22);
            scanf("%d",&nova_lin);
      } while (nova_lin<0 || nova_lin>=num_lin);

      strcpy(insertar.comanda,pgm[lin].comanda);
      insertar.quant=pgm[lin].quant;

      if (lin!=nova_lin) {
            if (nova_lin<lin) {
                  for (i=lin; (i>0 && i>nova_lin); i--) {
                        strcpy(pgm[i].comanda,pgm[i-1].comanda);
                        pgm[i].quant=pgm[i-1].quant;
                  }
            } else {
                  for (i=lin; (i<num_lin && i<nova_lin); i++) {
                        strcpy(pgm[i].comanda,pgm[i+1].comanda);
                        pgm[i].quant=pgm[i+1].quant;
                  }
            }
            strcpy(pgm[i].comanda,insertar.comanda);
            pgm[i].quant=insertar.quant;
      }
      _setcursortype(_NOCURSOR);
}
```

```
/*****************************************************************/
void ajustar_vel (int n)
{
      int i;

      enviar_car(conv_car(0));   /* tots els motors */
      enviar_car(V);             /* comanda velocitat */
      enviar_car(conv_car(n));   /* velocitat seleccionada */
}


/*****************************************************************/
void obrir_tancar_pinsa (char s)
{
      if (s=='+') {       /*tancar pinsa*/
            while (pinsa>pinsa_min && car_rebut!='8') {
                  enviar_trama(MOT8,M,'+',A,B,C,D,CR);
                  pinsa--;
            }
      } else {            /*obrir pinsa*/
            while (pinsa<pinsa_max && car_rebut!='8') {
                  enviar_trama(MOT8,M,'-',A,B,C,D,CR);
                  pinsa++;
            }
      }
      if (car_rebut=='8') {
            enviar_car('8');
            enviar_car('P');
            car_rebut='0';
      }
}


/*****************************************************************/
void operar_comanda (char comanda[10], int quant)
{
      if (!strcmp("Posicio",comanda))
            anar_posicio(quant);
      if (!strcmp("Velocitat",comanda))
            ajustar_vel(quant);
      if (!strcmp("Pausa",comanda))
            delay(quant);
      if (!strcmp("Obrir",comanda))
            obrir_tancar_pinsa('-');
      if (!strcmp("Tancar",comanda))
            obrir_tancar_pinsa('+');
}


/*****************************************************************/
void carrega_fitxer_pos (void)
{
      FILE *fitxer;
      char nom_fitxer[13], ext[5]=".pos";
      int i=0;
      float px,py,pz,ax,ay,az,ox,oy,oz,ab,ed;

      if (!home_ok)
            msg_error(HOME2);
      else {
```

```
            clrscr();
            _setcursortype(_NORMALCURSOR);
            finestra2(7,9,72,11,BROWN,WHITE);
            gotoxy(10,10);
            printf("Quin fitxer de POSICIO vols carregar a la
                                                  memoria?:");
            scanf("%s",nom_fitxer);
            strcat(nom_fitxer,ext);
            _setcursortype(_NOCURSOR);

            fitxer=fopen(nom_fitxer,"r");
            if (fitxer==NULL)
                  msg_error(FITXER);
            else {
                  rewind(fitxer);
                  while (!feof(fitxer)) {
                        fscanf(fitxer,"Pos %d    P %f  %f  %f     A %f
                        %f  %f     AB %f     O %f  %f  %f    ED %f\n",
                        &i,&px,&py,&pz,&ax,&ay,&az,&ab,&ox,&oy,&oz,&ed)
                        pos[i].p.x=px;
                        pos[i].p.y=py;
                        pos[i].p.z=pz;
                        pos[i].a.x=ax;
                        pos[i].a.y=ay;
                        pos[i].a.z=az;
                        pos[i].ab=ab;
                        pos[i].o.x=ox;
                        pos[i].o.y=oy;
                        pos[i].o.z=oz;
                        pos[i].ed=ed;
                  }
                  fclose(fitxer);
                  num_pos=i+1;
            }
      window(1,1,80,25);
      textcolor(WHITE);
      textbackground(BLUE);
      clrscr();
      }
}


/*****************************************************************/
void carrega_fitxer_pgm (void)
{
      FILE *fitxer;
      char nom_fitxer[13], comanda[8], ext[5]=".pgm";
      int quant, lin=0;

      clrscr();
      _setcursortype(_NORMALCURSOR);
      finestra2(7,9,72,11,BROWN,WHITE);
      gotoxy(10,10);
      printf("Quin fitxer de PROGRAMA vols carregar a la memoria?: ");
      scanf("%s",nom_fitxer);
      strcat(nom_fitxer,ext);
      _setcursortype(_NOCURSOR);

      fitxer=fopen(nom_fitxer,"r");
      if (fitxer==NULL)
            msg_error(FITXER);
```

```
      else {
            rewind(fitxer);
            while (!feof(fitxer)) {
                  fscanf(fitxer,"Lin %d  %s %d\n",&lin,comanda,&quant);
                  strcpy(pgm[lin].comanda,comanda);
                  pgm[lin].quant=quant;
            }
            fclose(fitxer);
            executa=SI;
            num_lin=lin+1;
      }
      window(1,1,80,25);
      textcolor(WHITE);
      textbackground(BLUE);
      clrscr();
}


/*****************************************************************/
void guarda_fitxer_pos (void)
{
      FILE *fitxer;
      char nom_fitxer[13], ext[5]=".pos";
      int i=0;

      clrscr();
      _setcursortype(_NORMALCURSOR);
      finestra2(7,9,72,11,BROWN,WHITE);
      gotoxy(10,10);
      printf("Quin fitxer de POSICIO vols guardar al disc?: ");
      scanf("%s",nom_fitxer);
      strcat(nom_fitxer,ext);
      _setcursortype(_NOCURSOR);

      fitxer=fopen(nom_fitxer,"w");
      if (fitxer==NULL)
            msg_error(FITXER);
      else {
            for (i=0;i<num_pos;i++)
                  fprintf(fitxer,"Pos %d    P %f  %f  %f     A %f  %f
%f     AB %f     O %f  %f  %f     ED
%f\n",i,pos[i].p.x,pos[i].p.y,pos[i].p.z,pos[i].a.x,pos[i].a.y,pos[i].
a.z,pos[i].ab,pos[i].o.x,pos[i].o.y,pos[i].o.z,pos[i].ed);
            fclose(fitxer);
      }
      window(1,1,80,25);
      textcolor(WHITE);
      textbackground(BLUE);
      clrscr();
}


/*****************************************************************/
void guarda_fitxer_pgm (void)
{
      FILE *fitxer;
      char nom_fitxer[13], ext[5]=".pgm";
      int i=0;

      clrscr();
      _setcursortype(_NORMALCURSOR);
```

```
      finestra2(7,9,72,11,BROWN,WHITE);
      gotoxy(10,10);
      printf("Quin fitxer de PROGRAMA vols guardar al disc?: ");
      scanf("%s",nom_fitxer);
      strcat(nom_fitxer,ext);
      _setcursortype(_NOCURSOR);

      fitxer=fopen(nom_fitxer,"w");
      if (fitxer==NULL)
            msg_error(FITXER);
      else {
            for (i=0;i<num_lin;i++)
                  fprintf(fitxer,"Lin %d  %s
                              %d\n",i,pgm[i].comanda,pgm[i].quant);
            fclose(fitxer);
      }
      window(1,1,80,25);
      textcolor(WHITE);
      textbackground(BLUE);
      clrscr();
}


/*****************************************************************/
void pos_actual (void)
{
      if (p_actual) {
            textbackground(CYAN);
            textcolor(BLACK);
            gotoxy(1,25);
            cprintf("                                                ");
            textbackground(MAGENTA);
            textcolor(BLACK);
            gotoxy(1,25);
            cprintf("Pos:");
            gotoxy(39,25);
            cprintf("Pitch:");
            gotoxy(64,25);
            cprintf("Roll:");
            textbackground(CYAN);
            textcolor(BLACK);
            gotoxy(7,25);
            cprintf("%7.2lf",P.x);
            gotoxy(16,25);
            cprintf("%7.2lf",P.y);
            gotoxy(24,25);
            cprintf("%7.2lf",P.z);
            gotoxy(49,25);
            cprintf("%4.1lf§",angle_AB*180/PI);
            gotoxy(72,25);
            cprintf("%4.1lf§",angle_ED*180/PI);

            window(1,1,80,25);
            textcolor(WHITE);
            textbackground(BLUE);
      }
}
```

```
/*****************************************************************/
void angle_actual (void)
{
      int x=3,y=9;
      if (parametres) {
            textcolor(BLACK);
            textbackground(WHITE);

            gotoxy(x,y);
            cprintf("ÉÍÍÍÍÍÍÍÍÍÍÍÍ»");
            gotoxy(x,y+1);
            cprintf("º z1:        º");
            gotoxy(x,y+2);
            cprintf("º z2:        º");
            gotoxy(x,y+3);
            cprintf("º z3:        º");
            gotoxy(x,y+4);
            cprintf("º z4:        º");
            gotoxy(x,y+5);
            cprintf("º z5:        º");
            gotoxy(x,y+6);
            cprintf("ÈÍÍÍÍÍÍÍÍÍÍÍÍ¼");

            gotoxy(x+6,y+1);
            cprintf("%6.1f",z1*180/PI);
            gotoxy(x+6,y+2);
            cprintf("%6.1f",z2*180/PI);
            gotoxy(x+6,y+3);
            cprintf("%6.1f",z3*180/PI);
            gotoxy(x+6,y+4);
            cprintf("%6.1f",z4*180/PI);
            gotoxy(x+6,y+5);
            cprintf("%6.1f",z5*180/PI);
      }
      window(1,1,80,25);
      textcolor(WHITE);
      textbackground(BLUE);
}


/*****************************************************************/
void comunicacions (void)
{
      int x=1,y=1;
      if (ccions) {
            textcolor(BLACK);
            textbackground(WHITE);

            gotoxy(x,y);
            cprintf("ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ»");
            gotoxy(x,y+1);
            cprintf("º car. rebut:        º");
            gotoxy(x,y+2);
            cprintf("º n§ rebuts:         º");
            gotoxy(x,y+3);
            cprintf("º err. linea:        º");
            gotoxy(x,y+4);
            cprintf("º err. modem:        º");
            gotoxy(x,y+5);
            cprintf("ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼");
```

```
            gotoxy(x+14,y+1);
            cprintf("%c",car_rebut);
            gotoxy(x+14,y+2);
            cprintf("%d",rebut);
            gotoxy(x+14,y+3);
            cprintf("%d",error);
            gotoxy(x+14,y+4);
            cprintf("%d",modem);
      }
      window(1,1,80,25);
      textcolor(WHITE);
      textbackground(BLUE);
}


/*************************************************************/
void msg_error (Terror err)
{
      char c;

      char msg_a[55];
      char msg_b[55];
      char msg0[]="         Prem qualsevol tecla per continuar      ";
      char msg1[]="              S'ha produit un error!!!           ";
      char msg2[]="    El COM escollit no existeix per a la BIOS!!! ";
      char msg3[]="              Impossible d'executar!!!!          ";
      char msg4[]="           No s'ha creat cap programa!!!         ";
      char msg5[]="          Impossible de moure el bra‡!!!!        ";
      char msg6[]="             No s'ha definit el HOME!!!          ";
      char msg7[]="          No s'ha calibrat el JOYSTICK!!!        ";
      char msg8[]="         No s'ha pogut obrir el fitxer!!!!       ";
      char msg9[]="   Impossible carregar un fitxer de posicions!!! ";


      switch (err) {
            case BIOS: strcpy(msg_a,msg1);
                       strcpy(msg_b,msg2);
                  break;
            case EXEC: strcpy(msg_a,msg3);
                       strcpy(msg_b,msg4);
                  break;
            case HOME: strcpy(msg_a,msg5);
                       strcpy(msg_b,msg6);
                  break;
            case HOME2: strcpy(msg_a,msg9);
                        strcpy(msg_b,msg6);
                  break;
            case JOYSTICK: strcpy(msg_a,msg5);
                           strcpy(msg_b,msg7);
                  break;
            case FITXER: strcpy(msg_a,msg1);
                         strcpy(msg_b,msg8);
                  break;
      }

      finestra2(15,8,68,17,BLACK,BLACK);
      finestra2(13,7,66,16,RED,BLACK);
      gotoxy(15,9);
      printf("%s",msg_a);
      gotoxy(15,11);
      printf("%s",msg_b);
```

```
	gotoxy(15,14);
	printf("%s",msg0);

	sound(70);
	delay(300);
	nosound();

	while (!kbhit());
	c=getch();
	window(1,1,80,25);
	textbackground(BLUE);
	textcolor(WHITE);
	clrscr();
}


/*****************************************************************/
void copyright (void)
{
	char c;

	window(7,6,74,19);
	textbackground(GREEN);
	textcolor(BLACK);
	clrscr();
	cprintf("
ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ»\r\n");
	cprintf(" º                                         º\r\n");
	cprintf(" MANIPULACIà D'UN ROBOT INDUSTRIAL D'ENSENYAN€º\r\n");
	cprintf(" º                                         º\r\n");
	cprintf(" Projecte de Final de Carrera de Joan Mercade º\r\n");
	cprintf(" º                                         º\r\n");
	cprintf(" º Departament d'Enginyeria Electr•nica, ElŠctrica i
	                                          Autom…tica  º\r\n");
	cprintf(" º    Escola TŠcnica Superior d'Enginyeria    º\r\n");
	cprintf(" º      Universitat Rovira i Virgili          º\r\n");
	cprintf(" º                                         º\r\n");
	cprintf(" º                                         º\r\n");
	cprintf(" º  Amb la colaboraci¢ de l'IES Comte de Rius º\r\n");
	cprintf(" º                                         º\r\n");
	cprintf(" ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼");

	while(!kbhit());
	c=getch();
	window(1,1,80,25);
	textbackground(BLUE);
	textcolor(WHITE);
	clrscr();
}


/****************************************************************/
char confirmar_exit (void)
{
	char c;

	sound(70);
	delay(200);
	nosound();
	do {
		finestra2(12,9,73,17,BLACK,BLACK);
```

```
            finestra2(10,8,71,16,RED,BLACK);
            gotoxy(26,11);
            printf("Segur que vols SORTIR? (");
            textcolor(YELLOW);
            cprintf("S");
            textcolor(BLACK);
            cprintf("i/");
            textcolor(YELLOW);
            cprintf("N");
            textcolor(BLACK);
            cprintf("o)");
            textattr(192);
            gotoxy(16,13);
            cprintf("Es perdr… tota la informaci¢ que no s'hagi
                                                  g•ardat");
            while(!kbhit());
            c=getch();
      } while (c!='s' && c!='n'&& c!='S' && c!='N');

      if (c=='s' || c=='S') {
            window(1,1,80,25);
            textbackground(BLACK);
            textcolor(LIGHTGRAY);
            clrscr();
            return '0';
      } else {
            window(1,1,80,25);
            textbackground(BLUE);
            textcolor(WHITE);
            clrscr();
            return '9';
      }
}

/*****************************************************************/
void finestra (int x1, int y1, int x2, int y2, int fondo, int text)
{
      int i=0, j=0;

      textbackground(fondo);
      textcolor(text);
      gotoxy(x1,y1);
      cprintf("É");
      for (i=x1+1; i<x2 ; i++) {
            gotoxy(i,y1);
            cprintf("Í");
      }
      gotoxy(x2,y1);
      cprintf("»");
      gotoxy(x1,y2);
      cprintf("È");
      for (i=x1+1; i<x2 ; i++) {
            gotoxy(i,y2);
            cprintf("Í");
      }
      gotoxy(x2,y2);
      cprintf("¼");
      for (i=y1+1; i<y2; i++) {
            gotoxy(x1,i);
            cprintf("º");
            for (j=x1+1; j<x2 ; j++) {
```

```
                gotoxy(j,i);
                cprintf(" ");
            }
            gotoxy(x2,i);
            cprintf("º");
      }
}


/*****************************************************************/
void finestra2 (int x1, int y1, int x2, int y2, int fondo, int text)
{
      int i=0, j=0;

      textbackground(fondo);
      textcolor(text);
      gotoxy(x1,y1);
      cprintf(" É");
      for (i=x1+2; i<x2 ; i++) {
            gotoxy(i,y1);
            cprintf("Í");
      }
      gotoxy(x2,y1);
      cprintf("» ");
      gotoxy(x1,y2);
      cprintf(" È");
      for (i=x1+2; i<x2 ; i++) {
            gotoxy(i,y2);
            cprintf("Í");
      }
      gotoxy(x2,y2);
      cprintf("¼ ");
      for (i=y1+1; i<y2; i++) {
            gotoxy(x1,i);
            cprintf(" º");
            for (j=x1+2; j<x2 ; j++) {
                  gotoxy(j,i);
                  cprintf(" ");
            }
            gotoxy(x2,i);
            cprintf("º ");
      }
}


/*****************************************************************/
void orientacio (void)
{
      double m_orientacio[4][4];
      coordenades i,j;
      double mod, alfa=0.0;

      if (P.x==0.0) {
            if (P.y>0.0) alfa=PI/2;
            else if (P.y<0.0) alfa=-PI/2;
            else alfa=alfa;
      }
      else alfa=atan2(P.y,P.x);

      roll=alfa;         /*angles d'Euler*/
      pitch=-angle_AB;
```

```
      yaw=angle_ED;

      m_orientacio[0][0]=cos(roll)*cos(pitch);
      m_orientacio[0][1]=(cos(roll)*sin(pitch)*sin(yaw))-
                                          (sin(roll)*cos(yaw));
      m_orientacio[0][2]=(cos(roll)*sin(pitch)*cos(yaw))+
                                          (sin(roll)*sin(yaw));
      m_orientacio[0][3]=P.x;
      m_orientacio[1][0]=sin(roll)*cos(pitch);
      m_orientacio[1][1]=(sin(roll)*sin(pitch)*sin(yaw))+
                                          (cos(roll)*cos(yaw));
      m_orientacio[1][2]=(sin(roll)*sin(pitch)*cos(yaw))-
                                          (cos(roll)*sin(yaw));
      m_orientacio[1][3]=P.y;
      m_orientacio[2][0]=-sin(pitch);
      m_orientacio[2][1]=cos(pitch)*sin(yaw);
      m_orientacio[2][2]=cos(pitch)*cos(yaw);
      m_orientacio[2][3]=P.z;
      m_orientacio[3][0]=0;
      m_orientacio[3][1]=0;
      m_orientacio[3][2]=0;
      m_orientacio[3][3]=1;

      /*noves coordenades del vector apropament*/
      a.x=((m_orientacio[0][0])*a_ref.x)+((m_orientacio[0][1])*
                           a_ref.y)+((m_orientacio[0][2])*a_ref.z);
      a.y=((m_orientacio[1][0])*a_ref.x)+((m_orientacio[1][1])*
                           a_ref.y)+((m_orientacio[1][2])*a_ref.z);
      a.z=((m_orientacio[2][0])*a_ref.x)+((m_orientacio[2][1])*
                           a_ref.y)+((m_orientacio[2][2])*a_ref.z);

      /*noves coordenades del vector orientacio*/
      o.x=((m_orientacio[0][0])*o_ref.x)+((m_orientacio[0][1])*
                           o_ref.y)+((m_orientacio[0][2])*o_ref.z);
      o.y=((m_orientacio[1][0])*o_ref.x)+((m_orientacio[1][1])*
                           o_ref.y)+((m_orientacio[1][2])*o_ref.z);
      o.z=((m_orientacio[2][0])*o_ref.x)+((m_orientacio[2][1])*
                           o_ref.y)+((m_orientacio[2][2])*o_ref.z);

      /*normalitzem els vectors*/
      mod=sqrt((a.x*a.x)+(a.y*a.y)+(a.z*a.z));
      a.x=a.x/mod;
      a.y=a.y/mod;
      a.z=a.z/mod;

      mod=sqrt((o.x*o.x)+(o.y*o.y)+(o.z*o.z));
      o.x=o.x/mod;
      o.y=o.y/mod;
      o.z=o.z/mod;
}


/*********************************************************************/
void dades_trajectoria (void)
{
      textbackground(GREEN);
      textcolor(BLACK);
      gotoxy(25,2);
      cprintf(" ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ» ");
      gotoxy(25,3);
      cprintf(" º     ENSENYAR TRAJECTORIA     º ");
```

```
      gotoxy(25,4);
      cprintf(" ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼ ");
      finestra2(23,9,59,16,BROWN,WHITE);
      gotoxy(26,10);
      printf("1) Moviment per articulacions");
      gotoxy(26,12);
      printf("2) Moviment del TCP");
      gotoxy(31,15);
      printf("ESC) Menu anterior");
      angle_actual();
      pos_actual();
      comunicacions();
}


/*********************************************************************/
void trajectoria (void)
{
      char opcio;

      traject=SI;
      clrscr();
      do {
            dades_trajectoria();
            while (!kbhit());
            opcio=getch();

            switch (opcio) {
                  case '1': menu_articulacions();
                        break;
                  case '2': menu_tcp();
                        break;
            }
      } while (int(opcio)!=27);    /*tecla ESCAPE*/
      clrscr();
      traject=NO;
}



/*********************************************************************/
void anar_posicio (int n_pos)
{
      P.x=pos[n_pos].p.x;
      P.y=pos[n_pos].p.y;
      P.z=pos[n_pos].p.z;
      a.x=pos[n_pos].a.x;
      a.y=pos[n_pos].a.y;
      a.z=pos[n_pos].a.z;
      o.x=pos[n_pos].o.x;
      o.y=pos[n_pos].o.y;
      o.z=pos[n_pos].o.z;
      angle_AB=pos[n_pos].ab;
      angle_ED=pos[n_pos].ed;

      inversa();
      posicio();
}
```

```
/*****************************************************************/
void dades_config_usuari (void)
{
      textbackground(GREEN);
      textcolor(BLACK);
      gotoxy(22,2);
      cprintf(" ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ» ");
      gotoxy(22,3);
      cprintf(" º    CONFIGURACIONS DE USUARI    º ");
      gotoxy(22,4);
      cprintf(" ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼ ");
      finestra2(24,8,55,17,BROWN,WHITE);
      gotoxy(28,9);
      printf("1) Parametres articulars");
      gotoxy(28,11);
      printf("2) Posicio actual");
      gotoxy(28,13);
      printf("3) Caracters rebuts");
      gotoxy(32,16);
      printf("ESC) Menu anterior");

      angle_actual();
      pos_actual();
      comunicacions();
}


/*****************************************************************/
void config_usuari (void)
{
      char opcio;
      FILE *fitxer;

      do {
            clrscr();
            dades_config_usuari();
            while (!kbhit());
            opcio=getch();

            switch (opcio) {
                  case '1': if (parametres) parametres=NO;
                            else parametres=SI;
                           break;
                  case '2': if (p_actual) p_actual=NO;
                            else p_actual=SI;
                           break;
                  case '3': if (ccions) ccions=NO;
                            else ccions=SI;
                           break;
            }
      } while (int(opcio)!=27);    /*tecla ESCAPE*/

      fitxer=fopen("robot.cfg","w");
      if (fitxer==NULL)
            msg_error(FITXER);
      else {
            fprintf(fitxer,"parametres articulars = %d\n",parametres);
            fprintf(fitxer,"posicio actual = %d\n",p_actual);
            fprintf(fitxer,"comunicacions = %d\n",ccions);
            fclose(fitxer);
      }
```

```
      clrscr();
}


/*****************************************************************/
void llegir_config_usuari (void)
{
      FILE *fitxer;
      fitxer=fopen("robot.cfg","r");
      if (fitxer==NULL)
            msg_error(FITXER);
      else {
            rewind(fitxer);
            while (!feof(fitxer)) {
                  fscanf(fitxer,"parametres articulars =
                                          d\n",&parametres);
                  fscanf(fitxer,"posicio actual = %d\n",&p_actual);
                  fscanf(fitxer,"comunicacions = %d\n",&ccions);
            }
            fclose(fitxer);
      }
}



/*****************************************************************/
void ini_timer (void)           /*timer decremental*/
{
      timer.valor=KTICS2;
      timer.final=0;
      timer.punter_a_funcio=enviar_car;
}
```