UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

DESARROLLO DEL SOFTWARE E INTERFAZ DE UN SISTEMA DE MANUFACTURA FLEXIBLE PARA UN PROCESO INDUSTRIAL DE PINTURA AUTOMATICO

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A N:

Beatriz González Ramírez
Dolores Leticia Maravilla Franco
Joel Villavicencio Cisneros

DIRECTOR Y ASESOR:

ING. IGNACIO JUAREZ CAMPOS

MEXICO, D. F. 1994

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

AGRADECEMOS PROFUNDAMENTE EL APOYO,
ORIENTACION Y AYUDA INCONDICIONALES BRINDADAS
POR NUESTRO DIRECTOR DE TESIS
MAESTRO EN INGENIERÍA
IGNACIO JUÁREZ CAMPOS

A NUESTROS PROFESORES, AMIGOS
Y COMPAÑEROS QUE TENDIERON
SU MANO PARA
LOGRAR NUESTRAS METAS

INDICE

INDICE

CAPITULO I

INTRODUCCIóN

INTRODUCCIÓN

La presente Tesis tiene como objetivo utilizar los conocimientos adquiridos durante la carrera en la implementación de un sistema de manufactura flexible para pintura.

A continuación se describe brevemente cada una de las unidades que conforman dicha Tesis mencionando antes dos etapas principales, una es el desarrollo de software que controla el sistema y la segunda es la implementación del hardware que será la interfaz entre el software y el sistema de manufactura flexible:

CAPÍTULO 2 REVISIÓN BIBLIOGRÁFICA DE SISTEMAS FLEXIBLES

En este capítulo se realizó una extensa investigación que nos permite conocer la historia de la robótica y definir todos los conceptos involucrados en los sistemas flexibles. Asimismo, se dan todas las características de éstos dividiéndolos en subsistemas como son: mecánico, decisión, percepción y comunicación.

El subsistema mecánico está constituido por la cadena cinemática asociada a la imagen del robot industrial, los actuadores que animan este mecanismo y todo el equipo complementario que permiten la correcta realización de las tareas definidas.

El subsistema de decisión, se encarga de procesar la información sensorial y controla la estructura mecánica para que la tarea programada sea realizada permitiendo el registro en memoria de dichas tareas.

El subsistema de percepción es aquel que está formado por los transductores y los circuitos asociados que permiten la generación de señales que informan al robot de su estado interno (posición y velocidad en cada articulación) o bien que le proporcionan información sobre el medio ambiente que le rodea.

El subsistema de comunicación permite al operador comunicarse con el robot, introducir las instrucciones que forman una tarea, modificarlas y, en general, activar las componentes del sistema del brazo robot.

CAPÍTULO 3 MODELADO MATEMÁTICO DEL SISTEMA

En este capítulo se realiza el análisis de los modelos matemáticos que nos permiten controlar la cinemática y la trayectoria del brazo robot.

Es importante comentar que la cinemática del brazo se refiere al estudio analítico de la geometría del movimiento de un robot con respecto a un sistema de coordenadas de referencia fijo como una función del tiempo sin considerar las fuerzas o los momentos que originan el movimiento.

La Generación de la trayectoria, en cambio, es la forma que va a describir el efector final (en este caso la herramienta de pintura), al desplazarse de un punto a otro (realizando la trayectoria recta o curva).

CAPÍTULO 4 DESARROLLO DEL SOFTWARE PARA EL DISEÑO DEL CONTROL

En este capítulo, se utiliza el modelo matemático seleccionado en el capítulo anterior para diseñar el software que nos proporcionará através del puerto paralelo las señales que serán alimentados al hardware del sistema de control. Además del modelo matemático, es importante contar con los conocimientos de Ingeniería de Software para realizar de una manera sistemática y organizada el desarrollo del software.

Algunos conceptos de Ingeniería de Software son el diagrama de flujo de datos y el diccionario de datos.

El diagrama de flujo de datos (DFD) describe en forma estructurada el funcionamiento del sistema. El diccionario de datos, es una parte integral de la especificación estructurada ya que el diagrama de flujo de datos por sí mismo podría proporcionar una idea errónea de lo que está sucediendo en el sistema.

El diccionario de datos se conforma principalmente por la descripción de definiciones como:

* Proceso
* Almacenamiento
* Entidad E/S
* Flujo de datos

CAPÍTULO 5 DESARROLLO DEL HARDWARE PARA EL DISEÑO DEL CONTROL

En este capítulo se llevó a cabo el análisis, diseño e implementación del hardware utilizado para controlar el brazo robot. Es importante tomar en cuenta que no todas las partes fueron diseñadas por nosotros ya que se trata de un proyecto multidisciplinario en la cual intervinieron Ingenieros Mecánicos del Instituto Tecnológico de Morelia.

Cada una de las partes que constituyen el Hardware se analizan por separado dando una breve introducción de sus características más importantes y una justificación de su empleo.

CAPÍTULO 6 PRUEBAS DE CAMPO

En este capítulo se dividieron las pruebas de campo en tres partes principales:

- Pruebas del Software
- Pruebas del Hardware para el diseño del control
- Acoplamiento de ambas etapas en el prototipo final

Las pruebas del software se realizaron en un principio con la ayuda de interfaces que nos permitían conocer las señales que eran enviadas desde la computadora al brazo robot pudiendo comprobar la buena operación del programa y de su manejo interactivo con el usuario final.

Las pruebas de hardware se realizaron mandando un tren de pulsos mediante un generador de funciones y un circuito diseñado por nosotros para generar la secuencia que permitiría el movimiento bidireccional de un motor de pasos.

Finalmente, se tomaron las señales que proporciona la computadora para manejar tres motores de pasos y la acción de una pistola de pintura dando por concluido el proyecto de Tesis.

Terminal
Computadora
Robot
Uso de Interface
Máquina de Interface
Software
Nivel Alto
Nivel Bajo

CAPITULO 2

REVISIÓN BIBLIOGRÁFICA DE SISTEMAS FLEXIBLES

II REVISIÓN BIBLIOGRÁFICA DE SISTEMAS FLEXIBLES

II.1 DEFINICIÓN DE SISTEMAS FLEXIBLES

El punto central es mostrar cómo y dónde las computadoras pueden ser usadas en el sector de la transformación de productos manufacturados. Nosotros hemos utilizado una sola computadora, la cual puede ser expandida para controlar una serie de procesos y máquinas de forma integral. El uso de computadoras en manufactura da origen al concepto de Sistema de Manufactura Flexible (SMF).

Un SMF es una extensión del control por computadora aplicado en la industria de la manufactura consistiendo en una serie de máquinas ligadas físicamente por un dispositivo de transporte de materiales (banda transportadora) y por una computadora maestra. La diferencia principal entre un proceso de manufactura y un SMF, es que este último puede usarse para manufacturar un amplio rango de productos reprogramando al sistema. Asimismo, un SMF es un sistema de máquinas automatizado controlado por computadora para convertir la materia prima en componente de conocida calidad.

Los beneficios generados con el uso de un SMF en comparación con el método manual o el método de manufactura no flexible son:

- Reducir costo por artículo
- Tiene una salida pronosticada
- Mejoramiento de la calidad

Comparándolo con el proceso manual se presentan varios problemas como son:

- Calidad variable
- Ritmo de salida impredecible
- Relativo alto costo por unidad producida

En la figura siguiente se observa que el proceso manual tiene generalmente un costo fijo y no varía con la cantidad. Los beneficios que pueden esperarse son:

- Una mínima inversión en equipo y
- Adaptabilidad de la persona al proceso con una calidad final aceptable.

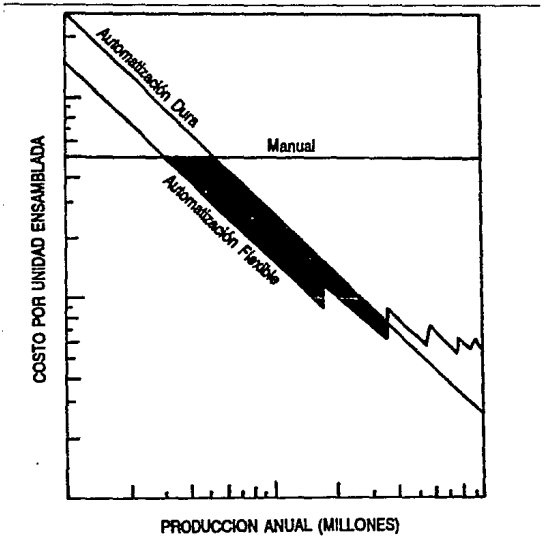


fig 1 Costo VS Demanda para tres diferentes métodos de producción

En la figura también se observa que el proceso manual tiene generalmente un costo fijo y no varía con la cantidad. Los beneficios que pueden esperarse son: una mínima inversión en equipo y la adaptabilidad de la persona al proceso con una calidad final aceptable.

La segunda opción de la figura es la de automatización no flexible, en la cual la máquina realiza específicamente una serie de tareas (generalmente una). El beneficio de esta opción es que

tiene ciclos cortos de tiempo, y alta velocidad de salida por turno; se utiliza en áreas donde es necesario un alto volumen de producción para hacer el costo eficaz.

La tercera opción es la automatización flexible, el principal ejemplo está en un brazo robot, esta opción necesita procesos precisos que realizar, pero no necesita de una alto volumen de producción para hacer su costo eficaz.

El volumen del producto que puede ser económicamente procesado por un SMF esta limitado por las curvas de los tres procesos, eligiendo los mas adecuados o dividiendo la inversión del capital en equipo mas el costo del material entre el numero de artículos manufacturados por unidad de tiempo.

El incremento del uso del SMF, es debido a la eficiencia en la integración jerárquica de objetivos y tareas, en el cual el resultado es alcanzado a través del concepto "divide y vencerás". El SMF puede procesar un articulo a través de un gran numero de máquinas con un mínimo retraso entre operaciones. El articulo es programado en el SMF por una computadora maestra, la cual carga los programas para cada máquina y genera la secuencia ordenada para que el articulo se transporte por las máquinas y realice las tareas necesarias.

La computadora central almacena grandes bases de datos con la información de como puede ser procesado un articulo. Si tenemos un nuevo articulo, podemos reprogramar y una vez probado, la información puede ser añadida a la base de datos. Si el nuevo articulo es una modificación de uno ya existente, la modificación será relativamente sencilla.

Los SMF son diseñados para una operación continua a través de 2 o 3 procesos, únicamente cambiando la parte terminal del robot, el cual es monitoreado a través de los sensores del brazo que le permitirán inclusive detectar un articulo defectuoso.

Las características básicas de los SMF son:

- Procesar por lotes pequeños y medianos
- Maximizar el uso de software para realizar cambios en el producto.
- Minimizar la necesidad de la intervención humana.
- Maximizar el uso de la computadora basado en tecnología para controlar y monitorear.
- Procesar una gran variedad de artículos.

Es importante mencionar que la intervención humana estará dedicada a:

- Mantenimiento.
- Programación para nuevos artículos.
- Configuración para nuevos lotes.
- Carga de materia prima.
- Rutinas administrativas de adquisición y despacho.

Las aplicaciones mas comunes de los SMF son:

- Preparación de muestras en análisis químicos.
- Preparación de muestras y análisis metalúrgicos.
- Análisis Biomédicos.
- Preparación de comida.
- Manufactura de ropa.

La filosofía de un SMF puede ser aplicada a sistemas grandes y pequeños, pero los niveles de productividad y costo son maximizados al incrementarse las aplicaciones.

II.2 BREVE HISTORIA DE LA ROBÓTICA

II.2.1 DEFINICIÓN DE ROBOT Y ROBÓTICA

Un robot es un manipulador mecánico con varios grados de libertad y capaz de ser reprogramado para llevar a cabo una variedad de tareas diferentes automáticamente.

A continuación se examinará cada parte de la definición.

1.-Un robot es un manipulador mecánico. Como una grúa o una pala mecánica, un robot es una maquina que es capaz de mover objetos de un lugar a otro.

2.-Un robot tiene varios grados de libertad, esto simplemente significa que un robot, debe ser capaz de moverse en varias direcciones diferentes. Esto también implica que debe estar equipado con alguna forma de potencia motriz.

3.-Un robot debe ser capaz de ser reprogramado para llevar a cabo diferentes tareas. Esta es, la parte principal de la definición; a diferencia de la mayoría de las maquinas construidas para un solo propósito, el robot es una maquina que puede ser

adaptada para realizar diferentes trabajos. La palabra reprogramar da a entender con claridad que es posible reprogramar el corazón

del robot que es una microcomputadora, cambiando totalmente la tarea que el realizaba a otra tarea muy diferente a la anterior.

4.-Un robot trabaja automáticamente. Una vez que al robot le ha sido programado o enseñado su trabajo, este debe ser capaz de llevarlo a cabo cualquier numero de veces sin intervención humana.

Son capaces de reaccionar a los estímulos externos para lo cual deben estar equipados con sensores. Para capacitar a los robots acerca de la información que les llega a través de los sensores deberá escribirse un software de análisis.

Sencillamente robótica es el estudio que abarca todo el conocimiento relativo a los robots.

II.2.2 HISTORIA DE LA ROBÓTICA

1 Manipuladores Mecánicos Maestro-Esclavo

Los primeros manipuladores mecánicos fueron diseñados para usarse en celdas radioactivas, donde el operador mueve un manipulador "maestro" y el manipulador "esclavo" dentro de la celda repite sus movimientos.

Al inicio, los manipuladores maestros fueron conectados a sus manipuladores esclavos mediante ligas mecánicas. Después fueron empleados cables de acero flexible.

Un desarrollo más reciente es el tipo de manipulador llamado "bilateral", ya que es el que provee fuerza de retroalimentación, es decir, el esclavo puede afectar al maestro. Por ejemplo, si el esclavo encuentra un obstáculo, el maestro se bloquea. Sin el uso de la retroalimentación ciertas operaciones serían extremadamente difíciles.

El siguiente desarrollo en los sistemas maestro-esclavo fue la introducción del control electrónico. Esto eliminó la necesidad de una liga mecánica entre maestro y esclavo. En su lugar, sensores de posición fueron montados en ambas unidades. Cuando el manipulador maestro es movido, señales eléctricas son mandadas al manipulador esclavo. Estas señales, en orden, causan que los motores en el

esclavo se desplacen hasta que los sensores indiquen que se ha llegado a la posición correcta. Los manipuladores mecánicos que no están mecánicamente acoplados se les puede adaptar la fuerza de retroalimentación, es decir, si el esclavo encuentra un obstáculo, éste no será capaz de seguir al maestro exactamente y la diferencia puede ser medida y convertida a señales eléctricas retroalimentando al operador.

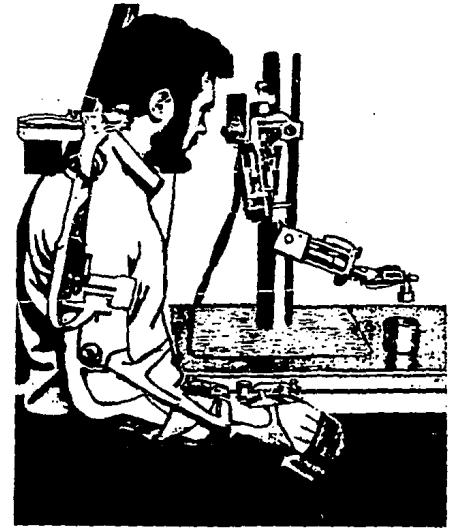


fig 2 Acoplamiento Eléctrico del "Rancho Arm"

2 Teleoperadores

Un sistema teleoperador es esencialmente el mismo que un sistema controlado electrónicamente, excepto que la liga de comunicación es inalámbrica y el maestro no necesita ser manipulado del todo; en su lugar se utiliza un panel con botones y perillas.

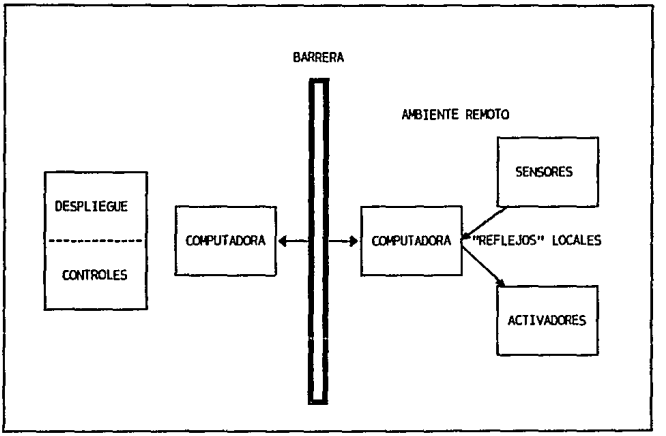


fig 3 Diagrama del Sistema de un Teleoperador

3 Teleoperadores que usan Computadoras

El desarrollo del teleoperador computarizado fue el resultado de la investigación patrocinada por la NASA. El elemento esencial en estos sistemas es la adición de inteligencia en forma de una o más computadoras locales (asociado a un maestro o esclavo).

Con una computadora remota y otra en el sitio de trabajo, las reacciones autónomas locales pueden ser programadas en el sistema (el término autónomo significa que en situaciones delicadas de

operación la computadora toma decisiones sin necesidad de que le indique el operador), realizando varias tareas regresando, al finalizar la tarea el control al operador.

4 El Robot con Visión

Se observa que bajo ciertas circunstancias un sistema completamente automático era factible. A la mitad de 1960, los investigadores de la Universidad de Stanford usaron técnicas de visión computarizada.

5 Robots Industriales

Una vez que los robots probaron su eficacia en los laboratorios, fueron puestos a trabajar en fábricas, realizando tareas repetitivas, tediosas y en muchos casos altamente peligrosas. Actualmente, miles de robots industriales son usados en fábricas y ayudan a manufacturar muchos productos que usamos cada día.

Por otro lado, en la figura siguiente se muestra el robot industrial T^3 (The Tomorrow Tool) creado a mitad de los 70's, el cual podía levantar hasta 100 libras a una velocidad de 50 in/seg, con un área de trabajo de 1000 ft^3, alcanzaba una altura de 13 ft y una precisión de 0.05 in. El T^3 era un robot de propósito general, completamente computarizado que tenía muchos sensores los cuales podían usarse para modificar su comportamiento.

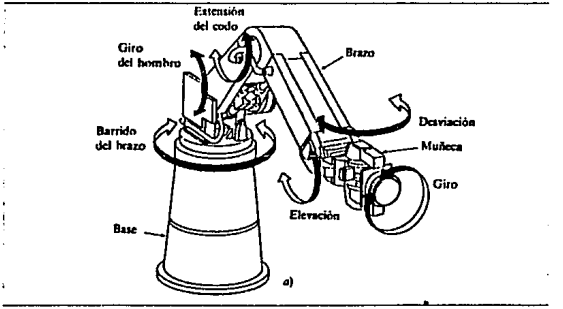


fig. 4 Robot Cincinnati Milacron T^3

Por otro lado, uno de los primeros robots industriales, el UNIMATE serie-4000 fué usada en líneas de ensamblado de automóviles, incrementaba la productividad y satisfacción de los trabajadores. El UNIMATE podía mover rápidamente cargas pesadas a su contraparte humana. Más aún, colocaba con prensas parte del automóvil reduciendo los accidentes en humanos.

6 Control de Enseñanza Industrial

Los primeros controladores tenían la forma en miniatura de un brazo. El usuario presionaba un botón de la unidad de control para mover el correspondiente miembro. Ala mitad de 1970, los controladores se sofisticaron para permitir especificar los movimientos del brazo en coordenadas.

Actualmente los controladores son más sofisticados. Algunas unidades permiten el movimiento del brazo almacenando sus posiciones, otras unidades están equipadas con sensores que simulan la vista, teclas programables multifunción. Los más avanzados permiten desarrollar programas desde el controlador, inclusive interactuar con lenguajes de alto nivel.

7 EL FUTURO DE LOS MANIPULADORES

Muchos laboratorios dirigen sus investigaciones en los campos de la robótica y de la Inteligencia Artificial. Actualmente las siguientes áreas están siendo exploradas: sensores de visión; sensores de fuerza y tacto; sensores de proximidad; cálculo de trayectorias; manipulación de lenguajes; planeación automática de sistemas.

II.3 CARACTERÍSTICAS DE SISTEMAS FLEXIBLES

II.3.1 SUBSISTEMA MECÁNICO

Constituida por la cadena cinemática asociada a la imagen de robot industrial, los actuadores que animan este mecanismo y todo el equipo complementario permiten la correcta realización de las tareas requeridas.

TIPOS DE CONFIGURACIONES EN ROBOTS

Los robots industriales están constituidos en una amplia gama

de tamaños, formas y configuraciones físicas. La gran mayoría de los robots comercialmente disponibles en la actualidad tienen una de estas cuatro configuraciones:

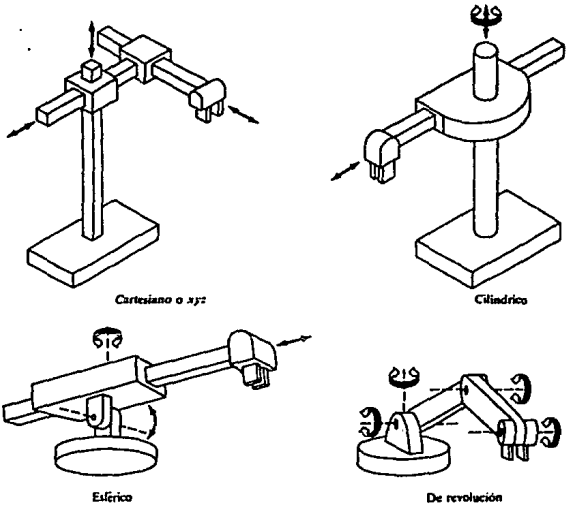


fig 5 Diversas categorías de robots

1 Configuración Polar ó Esférica

En la figura (a). Un brazo puede elevarse o bajar alrededor de un pivote horizontal. Este pivote está montado sobre una base giratoria. Estas diversas articulaciones proporcionan al robot la capacidad para desplazar su brazo dentro de un espacio esférico y de aquí la denominación de robot de coordenadas esféricas.

2 Configuración Cilíndrica

En la figura (b), se utiliza una columna vertical y un dispositivo de deslizamiento que puede moverse hacia arriba o hacia abajo a lo largo de una columna. El brazo del robot está unido al dispositivo deslizante de modo que puede moverse en sentido radial con respecto a la columna. Haciendo girar la columna, el robot es capaz de conseguir un espacio de trabajo que se aproxima a un cilindro.

3 Configuración de Coordenadas Cartesianas ó XYZ

En la figura (c), se utilizan tres dispositivos deslizantes perpendiculares para construir los ejes x,y,z. También llamado robot rectilíneo, xyz o de pórtico. Desplazando los tres dispositivos deslizantes entre sí, el robot es capaz de operar dentro de una envolvente rectangular de trabajo.

4 Configuración de Brazo Articulado ó de Revolución

En la figura (d), la configuración del brazo articulado es similar a la del brazo humano. Está constituido por dos componentes rectos que corresponden al brazo y antebrazo humanos montados sobre un pedestal vertical. Estos componentes están conectados por dos articulaciones giratorias que corresponden al hombro y al codo. Una muñeca está unida al extremo del antebrazo con lo que se le proporcionan varias articulaciones suplementarias. Existen en el mercado varios robots con este tipo de configuración.

Existen ventajas y desventajas en los cuatro tipos de configuraciones. En términos de repetibilidad de movimiento, el robot cartesiano es probable que tenga ventaja por su estructura rígida; en términos de alcance, las configuraciones polar y de brazo articulado resultan ventajosas; la configuración cilíndrica y el robot xyz pueden diseñarse para una alta rigidez y gran capacidad de carga; las configuraciones cilíndrica y polar, tienen ventaja en cuanto a la capacidad para penetrar a través de una pequeña abertura.

TIPOS DE ARTICULACIONES EN ROBOTS

La capacidad del robot para poder desplazar su cuerpo, brazo y muñeca se proporciona por el sistema de impulsión utilizado por el robot.

Los robots industriales, disponibles en el mercado, están accionados por uno de los tres tipos de sistemas de impulsión siguientes:

1 Impulsión hidráulica

Suele estar asociada con los robots más sofisticados y grandes. Su ventaja principal es la de proporcionar al robot una mayor velocidad y resistencia mecánica. Sus inconvenientes son que necesitan mayor cantidad de espacio además de estar propensos a fugas de aceite.

Estos sistemas pueden diseñarse para actuar en articulaciones rotacionales (actuadores de paletas giratorias) o lineales (pistones hidráulicos).

2 Impulsión eléctrica

Suele estar asociada con los robots más sofisticados. Su ventaja principal es la exactitud y la repetibilidad en el proceso, por tanto, los robots eléctricos tienden a ser más pequeños, con menos exigencias de espacio y sus aplicaciones tienden hacia un trabajo más preciso.

Los motores de impulsión eléctrica son accionados por motores de pasos, servomotores de corriente contínua y motores de corriente alterna. Estos motores son idóneos para el movimiento de articulaciones rotacionales mediante sistemas de engranajes y trenes impulsores adecuados; aunque también puedan usarse para accionar articulaciones lineales por medio de sistemas de poleas u otros mecanismos de traslación.

La economía es un factor en la decisión adecuada para emplear la impulsión hidráulica en los robots grandes y la impulsión eléctrica en los pequeños. El costo de un motor eléctrico es proporcional a su tamaño, mientras el costo de uno hidráulico depende en menor medida de su tamaño.

3 Impulsores neumáticos

Suelen reservarse para los robots más pequeños, con menos grados de libertad (de 2 a 4 articulaciones). Estos robots suelen estar limitados a simples operaciones de "coger y situar" con ciclos rápidos. La potencia neumática puede adaptarse fácilmente a la actuación de dispositivos de pistón para proporcionar un movimiento de traslación de articulaciones deslizantes. Puede emplearse para accionar actuadores giratorios en articulaciones rotacionales.

EFECTORES FINALES

La versatilidad de un robot depende ampliamente del dispositivo unido al final del brazo del robot. Este dispositivo es conocido como efector final. Un efector final no necesita ser como una mano mecánica. Este podría ser como un taladro o una herramienta para pintura. Los efectores finales se dividen en dos grupos:

1 Grippers

Estos son dispositivos que pueden ser usados para sostener o agarrar un objeto. En ellos se incluyen las manos mecánicas y cualquier otra cosa como ganchos, magnetos o dispositivos de succión.

Por ejemplo, si se considera un gripper que toma un objeto para dejarlo en otro lugar. Este tipo de efector final es controlado por el método conocido como de "punto a punto" y debe ser diseñado para requerir una mínima cantidad de movimientos.

2 Herramientas

Estos son dispositivos que el robot usa para realizar operaciones sobre un objeto. Entre ellos están los taladros, soldadoras y muchos otros.

Por ejemplo, una pistola de pintura. No sólo la ruta sino también la velocidad del efector final tiene que ser cuidadosamente y continuamente controlada, ya que si el movimiento es muy rápido la línea de pintura será muy delgada; por el contrario, si el movimiento es muy lento la línea de pintura será muy gruesa y se podría correr. Si hay un alto volumen de producción el costo extra del uso de pintura podría ser considerable.

A continuación se presentan varios tipos de grippers y herramientas que se utilizan en el proceso de la industria:

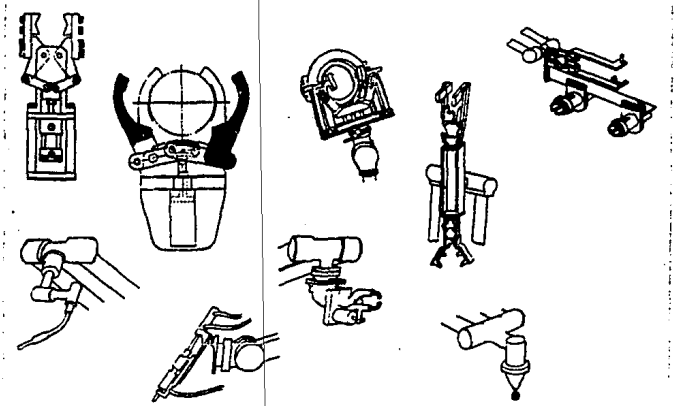


fig 6 Herramientas y Grippers

II.3.2 SUBSISTEMA DE PERCEPCIÓN

Subsistema formado por los transductores y los circuitos asociados que permiten la generación de señales que informan al robot de su estado interno (posición y velocidad en cada articulación) o bien que le proporcionan información sobre el medio ambiente que lo rodea.

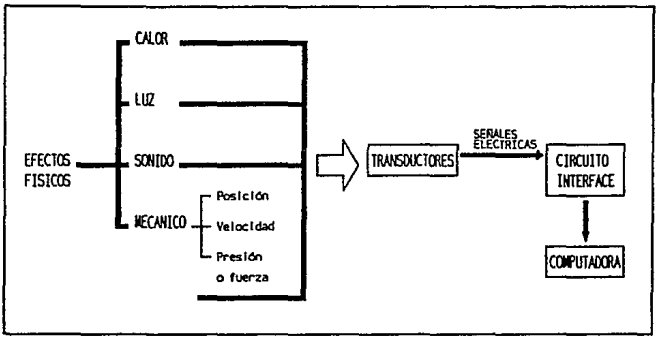


fig 7 Transductores convirtiendo efectos físicos en señales eléctricas

Un transductor es un dispositivo que convierte efectos físicos en señales eléctricas. Los efectos físicos pueden ser divididos en 4 categorías: calor, luz, sonido y mecánicos. Los efectos mecánicos pueden ser subdivididos en posición, velocidad, fuerza, presión y aceleración.

El empleo de mecanismos de detección exteriores permiten a un robot interaccionar con su ambiente de una manera flexible. Un robot que puede "ver" y "sentir" es mas fácil de entrenar en la ejecución de tareas complejas, mientras que exige mecanismos de control menos estrictos que las maquinas preprogramadas. Un sistema sensible y susceptible de entrenamiento es también adaptable a una gama mucho más amplia de tareas, con lo que se consigue un grado de universalidad que se traduce, a la larga, en bajos costos de producción y mantenimiento.

La función de los sensores de un robot puede dividirse en dos categorías principales: estado interno y estado externo. Los sensores de estado interno operan con la detección de variables, tales como la posición de la articulación del brazo.

Los sensores de estado externo operan con la detección de variables tales como el alcance, proximidad y contacto.

Los sensores de estado externo se utilizan por seguridad (evitan que el robot se dañe si encuentra algún obstáculo) y por dirección (sirve para el robot se oriente al tener que tomar, depositar o realizar una acción). Los sensores externos pueden dividirse en sensores de contacto o de no contacto. Como su nombre lo indica, la primera clase de sensores responde al contacto físico, tal como el tacto, deslizamiento y torsión. Los sensores de no contacto se basan en la respuesta de un detector a las variaciones en la radiación electromagnética, acústica y óptica.

SENSORES DE CONTACTO

Los sensores táctiles detectan el contacto con un objeto sensando la posible presión a aplicar debido a la forma y tamaño de un objeto. Por ejemplo, se tiene un sensor binario de contacto el cual está formado por un switch que detecta el contacto con un objeto en un punto. Otro sensor determina la presión con que toma un objeto produciendo un voltaje o corriente de acuerdo a la fuerza que debe ser aplicada. Entre otros sensores táctiles están los fotomecánicos en el cual al presionar un objeto varía un haz de luz que llega al receptor y esto hace que varíe voltaje o corriente. Los sensores piezo-resistivos, (contienen entre dos placas metálicas un material que cambia su resistencia cuando es presionado modificando la corriente que pasa a través de las placas) y los sensores de matriz (pueden detectar la forma de un objeto y su orientación por medio de una matriz de elementos piezo-eléctricos que cambian su voltaje al ser presionados).

El sensor neumático mostrado en la siguiente figura:

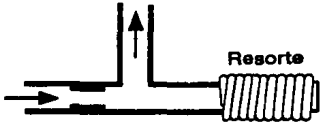


fig 8 Sensor neumático de Tacto

Es un sensor que cuando hace contacto el objeto con el resorte hermético se dobla permitiendo que el aire comprimido salga a la atmósfera y así sea detectado el objeto, como a continuación se muestra en la figura 9.

fig 9 Escapa el aire al doblarse el resorte con el objeto

SENSORES DE NO CONTACTO

Los sensores de proximidad ópticos (figura 10) detectan la distancia entre el sensor y un objeto, por su influencia sobre una onda propagadora que se desplaza desde un transmisor hasta un receptor.

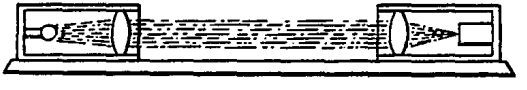


fig 10 Punto del Sensor Óptico

Los sensores acústicos convierten los sonidos en señales eléctricas que pueden ser discriminadas para que el robot entienda algunas palabras (reconocimiento de voz) e inclusive frases completas (reconocimiento de lenguaje).

Los sensores de gas se colocan al final de los brazos robots y permiten detectar la presencia de ciertos gases lo cual puede aplicarse a seguridad (medio ambiente peligroso) o detectar fugas.

II.3.3 SUBSISTEMA DE DECISIÓN

Este subsistema procesa la información sensorial y controla la estructura mecánica para que la tarea programada sea realizada permitiendo el registro en memoria de dichas tareas.

Para diseñar la solución de una aplicación, se necesita entender el problema; el entendimiento de que se necesita hacer, medir y controlar es llamado modelado del sistema. Algunas veces es muy sencillo modelar un sistema, pero la mayoría de las veces es difícil.

Una de las ventajas de usar una computadora como controlador es que las ecuaciones no tienen que ser construídas ya que pueden ser parte de un programa. Además, es posible hacer los parámetros de las ecuaciones variables y que estén bajo el control del programa, así la computadora es capaz de modificar sus acciones dependiendo de los resultados previos.

La computadora puede realizar varios procesos a la vez, pero realmente no es un multiproceso ya que a cada tarea se le dedica un intervalo de tiempo. Algunas veces hay tareas que se tienen que realizar inmediatamente ya que su ejecución es aleatoria. Dependiendo de los cambios de valor en los sensores, si la computadora detecta una falla en alguna operación deberá terminar esa tarea inmediatamente antes de producir algún daño.

Los brazos robots pueden tener varios ejes de movimiento conocidos como grados de libertad, cada eje se maneja como una subtarea en sí misma. Dependiendo de como este escrito el programa puede requerir como entrada alguna posición final además de su posición actual y el tiempo en el cual la posición será alcanzada. La salida será la información necesaria para controlar a los motores. Usando circuitos electrónicos el control podría ser muy complejo. Usando una computadora, cada tarea puede ser considerada independiente e incorporada dentro de un programa. Particionando las tareas de control de esta forma es posible hacer el problema manejable.

II.3.4 SUBSISTEMA DE COMUNICACIÓN

Este subsistema permite al operador comunicarse con el robot introduciendo las instrucciones que forman una tarea pudiendo modificarlas y, en general, activar las componentes del Sistema Robot.

Para proveer al robot con instrucciones de una manera textual y no a través de un panel de botones, utilizamos lenguajes de programación. Estos lenguajes son derivados, la mayoría de las veces, de lenguajes de programación como BASIC o PASCAL pero en algunos casos pueden ser desarrollados lenguajes nuevos.

Para usar estos lenguajes el operador está equipado con un teclado convencional y una pantalla, quizás con la adición de un numero de teclas de propósito especial. Además el usuario manda la información de sus requerimientos mediante un dialogo vía teclado y salida en pantalla. El programa acepta una entrada del teclado y entonces causa que el movimiento apropiado sea activado observando los efectos en pantalla.

CAPITULO III

MODELADO MATEMÁTICO DEL SISTEMA

MODELADO MATEMÁTICO DEL SISTEMA

Como ya se ha visto, un robot consta de varios subsistemas que hay que controlar para que al interactuar entre ellos realicen una tarea específica. El problema del control de movimientos, para este caso se puede dividir en dos partes:

- Cinemática del brazo
- Generación de la trayectoria.

La Cinemática del brazo de robot se refiere al estudio analítico de la geometría del movimiento de un robot con respecto a un sistema de coordenadas de referencia fijo como una función del tiempo sin considerar las fuerzas o los momentos que originan el movimiento; la Generación de la trayectoria, en cambio, es la forma que va a describir el efector final (en este caso la herramienta de pintura), al desplazarse de un punto a otro (realizando la trayectoria recta o circular).

Existen dos problemas en la cinemática de un robot que suelen conocerse como cinemática directa e inversa; en la cinemática directa, se conocen los ángulos de las articulaciones y con base a ellos se obtiene la posición y orientación del efector final (herramienta de pintura); en la cinemática inversa, se conoce la posición y orientación del efector final y con base a éstos se obtienen los ángulos de las articulaciones.

Para resolver los problemas de la cinemática directa se utilizó el método de Denavit y Hartenberg. Este método reduce el problema cinemático directo para encontrar una matriz de transformación homogénea de 4x4 que relacione el desplazamiento espacial del sistema de coordenadas de la herramienta de pintura al sistema de coordenadas de referencia. Esta matriz se puede dividir en 4 matrices pequeñas, en donde la primera es una matriz de rotación (que depende del eje o ejes alrededor de los cuales gire); la segunda, es un vector de posición desde la herramienta de pintura hasta el cero de las coordenadas fijas (coordenadas de origen), vector de ceros y una matriz de 1x1 con el valor de 1.

$$ {}^{i-1}_{i}T = \left| \begin{array}{c|c} {}^{i-1}_{i}R & {}^{i-1}Porg(i) \\ \hline 0 \quad 0 \quad 0 & 1 \end{array} \right| $$

Si hay movimiento de giro alrededor del eje X con ángulo de θ grados de una base generadora del espacio tridimensional {A} a una {B}, la matriz de rotación canónica será:

$$ {}^{A}_{B}R(x,\theta) = \begin{vmatrix} 1 & 0 & 0 \\ 0 & \text{Cos}\theta & -\text{Sen}\theta \\ 0 & \text{Sen}\theta & \text{Cos}\theta \end{vmatrix} $$

si hay movimiento de giro alrededor del eje Y la matriz de rotación canónica será:

$$ {}^{A}_{B}R(y,\theta) = \begin{vmatrix} \text{Cos}\theta & 0 & \text{Sen}\theta \\ 0 & 1 & 0 \\ -\text{Sen}\theta & 0 & \text{Cos}\theta \end{vmatrix} $$

si hay movimiento de giro alrededor del eje Z la matriz de rotación canónica será:

$$ {}^{A}_{B}R(z,\theta) = \begin{vmatrix} \text{Cos}\theta & -\text{Sen}\theta & 0 \\ \text{Sen}\theta & \text{Cos}\theta & 0 \\ 0 & 0 & 1 \end{vmatrix} $$

Por otro lado, para la cinemática inversa se utilizó un proceso iterativo, obteniéndose un algoritmo robusto que elimina errores de redondeo. El método numérico empleado para encontrar la solución al problema inverso fue Newton-Raphson, el cual tiene la siguiente forma:

$$ \theta_{k+1} = \theta_k + J(\theta_k)^{-1} \, [(O) - (F\theta_k)] $$

donde θ_{k+1} ángulos siguientes
θ_k ángulos actuales

O punto siguiente
$F(\theta_k)$ punto actual
$J(\theta_k)$ jacobiano de $F(\theta_k)$

III.1 CINEMÁTICA DIRECTA

Como se mencionó anteriormente, la notación a utilizar será la de Denavit-Hartenberg. Para ello se necesitan definir los siguientes parámetros de eslabonamiento (unión entre dos eslabones):

a_{i-1} Longitud de Eslabonamiento en cm.
(distancia más corta entre los ejes $_{i-1}$ e $_i$).
α_{i-1} Ángulo de Torsión en grados.
(ángulo que va del eje $_{i-1}$ al eje $_i$).
d_i Distancia que existe de a_{i-1} hasta a_i sobre el eje i en cm.
θ_{i-1} Ángulo que va de la prolongación de a_{i-1} hacia a_i tomando el sentido positivo de d_i en grados.

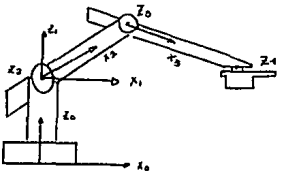


fig 1

De la figura anterior, se obtuvieron los valores de los parámetros de eslabonamiento con los cuales se completó la siguiente tabla. Nótese que si la articulación **i** es prismática (movimiento lineal), la distancia d_i sería variable y el ángulo θ_i

constante, en caso de que fuera rotacional (movimiento angular), la distancia d_i sería constante y el ángulo θ_i variable. Como en este caso, todas las articulaciones fueron rotacionales esto significó que todos los ángulos θ_i fueran variables, mientras que las distancias d_i tuvieran un valor fijo.

i	a_{i-1}	α_{i-1}	d_i	θ_i
1	0	0	l1	θ_1
2	0	90	0	θ_2
3	l2	0	0	θ_3
4	l3	0	0	θ_4

Es necesario, con estos datos formar las matrices de transformación sustituyendo los valores de la tabla en la siguiente matriz general:

$$ {}^{i-1}_{i}T = \begin{vmatrix} Cos\theta_i & -Sen\theta_i & 0 & a_{i-1} \\ Sen\theta_i Cos\alpha_{i-1} & Cos\theta_i Cos\alpha_{i-1} & -Sen\alpha_{i-1} & -d_i Sen\alpha_{i-1} \\ Sen\theta_i Sen\alpha_{i-1} & Cos\theta_i Sen\alpha_{i-1} & Cos\alpha_{i-1} & d_i Cos\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{vmatrix} $$

Fórmula 1

Pero, el origen de ésta matriz resultó de una serie de transformaciones sucesivas, es decir, la transformación ${}^{1}_{2}T$ es el resultado de una rotación de 90 grados en el eje X, una rotación de θ_2 grados en el eje Z y una traslación nula:

Rotación de 90 grados en el eje X

$$ {}^{A}_{B}R(x,90) = \begin{vmatrix} 1 & 0 & 0 \\ 0 & Cos90 & -Sen90 \\ 0 & Sen90 & Cos90 \end{vmatrix} $$

Se obtiene lo siguiente:

$$ {}^{A}_{B}R(x,90) = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{vmatrix} $$

multiplicado por la rotación sobre el eje Z, θ_2 grados

$$ {}^{A}_{B}R(z,\theta_2) = \begin{vmatrix} \text{Cos}\theta_2 & -\text{Sen}\theta_2 & 0 \\ \text{Sen}\theta_2 & \text{Cos}\theta_2 & 0 \\ 0 & 0 & 1 \end{vmatrix} $$

resulta

$$ {}^{A}_{B}R = \begin{vmatrix} \text{Cos}\theta_2 & -\text{Sen}\theta_2 & 0 \\ 0 & 0 & -1 \\ \text{Sen}\theta_2 & \text{Cos}\theta_2 & 0 \end{vmatrix} $$

uniendo esta rotación final con los demás componentes de la matriz de transformación queda finalmente:

$$ {}^{1}_{2}T = \begin{vmatrix} \text{Cos}\theta_2 & -\text{Sen}\theta_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \text{Sen}\theta_2 & \text{Cos}\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} $$

De la fórmula 1, se sustituyen los valores de la tabla quedando finalmente:

$$
{}^{0}_{1}T = \begin{vmatrix} \cos\theta_1 & -\operatorname{sen}\theta_1 & 0 & a_0 \\ \operatorname{sen}\theta_1\cos\alpha_0 & \cos\theta_1\cos\alpha_0 & -\operatorname{sen}\alpha_0 & -d_1\operatorname{sen}\alpha_0 \\ \operatorname{sen}\theta_1\operatorname{sen}\alpha_0 & \cos\theta_1\operatorname{sen}\alpha_0 & \cos\alpha_0 & d_1\cos\alpha_0 \\ 0 & 0 & 0 & 1 \end{vmatrix}
$$

$$
{}^{1}_{2}T = \begin{vmatrix} \cos\theta_2 & -\operatorname{sen}\theta_2 & 0 & a_1 \\ \operatorname{sen}\theta_2\cos\alpha_1 & \cos\theta_2\cos\alpha_1 & -\operatorname{sen}\alpha_1 & -d_2\operatorname{sen}\alpha_1 \\ \operatorname{sen}\theta_2\operatorname{sen}\alpha_1 & \cos\theta_2\operatorname{sen}\alpha_1 & \cos\alpha_1 & d_2\cos\alpha_1 \\ 0 & 0 & 0 & 1 \end{vmatrix}
$$

$$
{}^{2}_{3}T = \begin{vmatrix} \cos\theta_3 & -\operatorname{sen}\theta_3 & 0 & a_2 \\ \operatorname{sen}\theta_3\cos\alpha_2 & \cos\theta_3\cos\alpha_2 & -\operatorname{sen}\alpha_2 & -d_3\operatorname{sen}\alpha_2 \\ \operatorname{sen}\theta_3\operatorname{sen}\alpha_2 & \cos\theta_3\operatorname{sen}\alpha_2 & \cos\alpha_2 & d_3\cos\alpha_2 \\ 0 & 0 & 0 & 1 \end{vmatrix}
$$

$$
{}^{3}_{4}T = \begin{vmatrix} \cos\theta_4 & -\operatorname{sen}\theta_4 & 0 & a_3 \\ \operatorname{sen}\theta_4\cos\alpha_3 & \cos\theta_4\cos\alpha_3 & -\operatorname{sen}\alpha_3 & -d_4\operatorname{sen}\alpha_3 \\ \operatorname{sen}\theta_4\operatorname{sen}\alpha_3 & \cos\theta_4\operatorname{sen}\alpha_3 & \cos\alpha_3 & d_4\cos\alpha_3 \\ 0 & 0 & 0 & 1 \end{vmatrix}
$$

Evaluando las matrices anteriores sustituyendo los valores contenidos en la tabla inicial, obtenemos:

$$
{}^{0}_{1}T = \begin{vmatrix} \text{Cos}\Theta_1 & -\text{Sen}\Theta_1 & 0 & 0 \\ \text{Sen}\Theta_1 & \text{Cos}\Theta_1 & 0 & 0 \\ 0 & 0 & 1 & 11 \\ 0 & 0 & 0 & 1 \end{vmatrix}
$$

$$
{}^{1}_{2}T = \begin{vmatrix} \text{Cos}\Theta_2 & -\text{Sen}\Theta_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \text{Sen}\Theta_2 & \text{Cos}\Theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}
$$

$$
{}^{2}_{3}T = \begin{vmatrix} \text{Cos}\Theta_3 & -\text{Sen}\Theta_3 & 0 & 12 \\ \text{Sen}\Theta_3 & \text{Cos}\Theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}
$$

$$
{}^{3}_{4}T = \begin{vmatrix} \text{Cos}\Theta_4 & -\text{Sen}\Theta_4 & 0 & 13 \\ \text{Sen}\Theta_4 & \text{Cos}\Theta_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}
$$

Las matrices anteriores representan los movimientos angulares y lineales que se requieren para llegar de la articulación **i+1** a la articulación **i**. Como se necesitó para este brazo robot la transformación del eje 4 al eje 0 se multiplicó la matriz de transformación ${}^{0}_{1}T$ por la matriz ${}^{1}_{2}T$ para obtener la matriz ${}^{0}_{2}T$. Esta matriz de transformación, se multiplicó por la matriz ${}^{2}_{3}T$ para obtener la matriz ${}^{0}_{3}T$ y finalmente, ésta se multiplicó por la matriz ${}^{3}_{4}T$ para obtener la última matriz de transformación ${}^{0}_{4}T$, como sigue:

$$
{}^{0}_{4}T = \begin{vmatrix} [\ c\Theta_1 c\Theta_2\ \ [c\Theta_3 c\Theta_4 - s\Theta_3 s\Theta_4]\ -\ c\Theta_1 s\Theta_2\ \ [s\Theta_3 c\Theta_4 + c\Theta_3 s\Theta_4]\ ] \\ [\ s\Theta_1 c\Theta_2\ \ [c\Theta_3 c\Theta_4 - s\Theta_3 s\Theta_4]\ -\ s\Theta_1 s\Theta_2\ \ [s\Theta_3 c\Theta_4 + c\Theta_3 s\Theta_4]\ ] \\ [\ s\Theta_2\ \ [c\Theta_3 c\Theta_4 - s\Theta_3 s\Theta_4]\ +\ c\Theta_2\ \ [s\Theta_3 c\Theta_4 + c\Theta_3 s\Theta_4]\ ] \\ 0 \end{vmatrix}
$$

$$
\begin{bmatrix} [-c\Theta c\Theta_2\ \ [c\Theta_3 s\Theta_4 + s\Theta_3 c\Theta_4]\ -\ c\Theta s\Theta_2\ \ [-s\Theta_3 s\Theta_4 + c\Theta_3 c\Theta_4]\ ] \\ [-s\Theta c\Theta_2\ \ [c\Theta_3 s\Theta_4 + s\Theta_3 c\Theta_4]\ -\ s\Theta s\Theta_2\ \ [-s\Theta_3 s\Theta_4 + c\Theta_3 c\Theta_4]\ ] \\ [-s\Theta_2\ \ [c\Theta_3 s\Theta_4 + s\Theta_3 c\Theta_4]\ +\ c\Theta_2\ \ [-s\Theta_3 s\Theta_4 + c\Theta_3 c\Theta_4]\ ] \\ 0 \end{bmatrix}
$$

$$
\begin{vmatrix} s\Theta & [c\Theta c\Theta_2\ \ [L3 c\Theta_3 + L2]\ -\ c\Theta s\Theta_2\ \ [L3 s\Theta_3]\ ] \\ -c\Theta & [s\Theta c\Theta_2\ \ [L3 c\Theta_3 + L2]\ -\ s\Theta s\Theta_2\ \ [L3 s\Theta_3]\ ] \\ 0 & [\ s\Theta_2\ \ [L3 c\Theta_3 + L2]\ +\ c\Theta_2\ \ [L3 s\Theta_3]\ +\ L] \\ 0 & 1 \end{vmatrix}
$$

III.2 CINEMÁTICA INVERSA

Para el desarrollo de este punto, se utilizó el método de Newton-Raphson, el cuál tiene la siguiente forma matricial para 3 variables (X, Y, Z):

$$
\begin{vmatrix} \Theta_1 \\ \Theta_2 \\ \Theta_3 \end{vmatrix}_{k+1} = \begin{vmatrix} \Theta_1 \\ \Theta_2 \\ \Theta_3 \end{vmatrix}_{k} + \begin{vmatrix} \delta f_1/\delta\Theta_1 & \delta f_1/\delta\Theta_2 & \delta f_1/\delta\Theta_3 \\ \delta f_2/\delta\Theta_1 & \delta f_2/\delta\Theta_2 & \delta f_2/\delta\Theta_3 \\ \delta f_3/\delta\Theta_1 & \delta f_3/\delta\Theta_2 & \delta f_3/\delta\Theta_3 \end{vmatrix}^{-1} \begin{vmatrix} {}^{O_x}_{f_2({}^{k}\Theta_1, {}^{k}\Theta_2, {}^{k}\Theta_3)} \\ {}^{O_y}_{f_2({}^{k}\Theta_1, {}^{k}\Theta_2, {}^{k}\Theta_3)} \\ {}^{O_z}_{f_2({}^{k}\Theta_1, {}^{k}\Theta_2, {}^{k}\Theta_3)} \end{vmatrix}
$$

A continuación se presentan las ecuaciones que forman al vector de posición

$$f_1 = C\Theta C\Theta_2\ [L3C\Theta_3+L2] - C\Theta S\Theta_2\ [L3S\Theta_3]$$
$$f_2 = S\Theta C\Theta_2\ [L3C\Theta_3+L2] - S\Theta S\Theta_2\ [L3S\Theta_3]$$
$$f_3 = S\Theta_2\ [L3C\Theta_3+L2] + C\Theta_2\ [L3S\Theta_3] + L$$

donde

f1 es el desplazamiento en X
f2 es el desplazamiento en Y
f3 es el desplazamiento en Z

Por otro lado, las derivadas parciales del vector de posición son las siguientes:

$A = \delta f_1/\delta\Theta_1 = -S\Theta_1 C\Theta_2\ [L3C\Theta_3+L2] + S\Theta_1 S\Theta_2\ [L3S\Theta_3]$
$B = \delta f_1/\delta\Theta_2 = -C\Theta_1 S\Theta_2\ [L3C\Theta_3+L2] - C\Theta_1 C\Theta_2\ [L3S\Theta_3]$
$C = \delta f_1/\delta\Theta_3 = -C\Theta_1 C\Theta_2\ [L3S\Theta_3] - C\Theta_1 S\Theta_2\ [L3C\Theta_3]$
$D = \delta f_2/\delta\Theta_1 = C\Theta_1 C\Theta_2\ [L3C\Theta_3+L2] - C\Theta_1 S\Theta_2\ [L3S\Theta_3]$
$E = \delta f_2/\delta\Theta_2 = -S\Theta_1 S\Theta_2\ [L3C\Theta_3+L2] - S\Theta_1 C\Theta_2\ [L3S\Theta_3]$
$F = \delta f_2/\delta\Theta_3 = -S\Theta_1 C\Theta_2\ [L3S\Theta_3] - S\Theta_1 S\Theta_2\ [L3C\Theta_3]$
$G = \delta f_3/\delta\Theta_1 = 0$
$H = \delta f_3/\delta\Theta_2 = C\Theta_2\ [L3C\Theta_3+L2] - S\Theta_2\ [L3S\Theta_3]$
$I = \delta f_3/\delta\Theta_3 = -S\Theta_2\ [L3S\Theta_3] + C\Theta_2\ [L3C\Theta_3]$

Ahora, para obtener el inverso del jacobiano se tiene la siguiente expresión:

$$J^{-1} = (\quad /\text{Det } J)\ \text{Adj } J$$

En donde

$$\text{Det } J = \begin{vmatrix} A & B & C \\ D & E & F \\ G & H & I \end{vmatrix} = A(EI-FH) + B(FG-DI) + C(DH-EG)$$

y

$$ADJ = \begin{vmatrix} (EI-FH) & -(BI-CH) & (BF-EC) \\ -(DI-FG) & (AI-GC) & -(AF-CD) \\ (DH-EG) & -(AH-BG) & (AE-BD) \end{vmatrix}$$

finalmente, se obtienen los ángulos del siguiente punto:

$$\theta_{1\,K+1} = \theta_{1\,K} + 1/[A(EI-FH) + B(FG-DI) + C(DH-EG)] * [[(EI-FH)*(O_x-F_1)] + [(CH-BI)*(O_y-F_2)] + [(BF-EC)*(O_z-F_3)]]$$

$$\theta_{2\,K+1} = \theta_{2\,K} + 1/[A(EI-FH) + B(FG-DI) + C(DH-EG)] * [[(FG-DI)*(O_x-F_1)] + [(AI-GC)*(O_y-F_2)] + [(CD-AF)*(O_z-F_3)]]$$

$$\theta_{3\,K+1} = \theta_{3\,K} + 1/[A(EI-FH) + B(FG-DI) + C(DH-EG)] * [[(DH-EG)*(O_x-F_1)] + [(BG-AH)*(O_y-F_2)] + [(AE-BD)*(O_z-F_3)]]$$

III.3 GENERACIÓN DE LA TRAYECTORIA

Se puede observar que la forma en la que se generan las trayectorias en el programa de control es sencillo. Se tienen dos trayectorias que el brazo robot puede realizar, recta y curva.

La trayectoria recta se genera con las ecuaciones vectoriales de la recta, como son:

$$\begin{aligned} X &= X_i + t\,(X_f - X_i) \\ Y &= Y_i + t\,(Y_f - Y_i) \\ Z &= Z_i + t\,(Z_f - Z_i) \end{aligned}$$

donde si t=0, se obtiene el punto inicial:

$$\begin{aligned} X &= X_i \\ Y &= Y_i \\ Z &= Z_i \end{aligned}$$

y si t=1, se obtiene el punto final:

$$\begin{aligned} X &= X_i + (X_f - X_i) = X_f \\ Y &= Y_i + (Y_f - Y_i) = Y_f \\ Z &= Z_i + (Z_f - Z_i) = Z_f \end{aligned}$$

Por lo tanto, para generar una recta sólo basta con variar t entre 0 y 1. Así se llamó DIV al número de divisiones en que se fraccionó la recta y el valor de t se igualó a 1/DIV. Esto se hizo con el fin de que el error en los ángulos de movimiento en las articulaciones fuera lo más pequeño posible.

Por ejemplo, si el número de divisiones es 5 el valor de t es 0.2, por lo tanto para generar todos los puntos de la recta basta con que t tome los valores de 0, 0.2, 0.4, 0.6, 0.8, 1 y se obtenga la recta dividida en 5 partes iguales.

La trayectoria circular es el movimiento que el brazo robot realiza tomando sólo una parte del círculo para dibujar una curva; esta curva se genera con la interpolación de Lagrange. Para realizarla, se necesita tener una coordenada constante para que la curva se realice en un plano utilizando la siguiente ecuación:

$$Y = \sum_{i=0}^{n} \left[\pi_{j=0,\, j\neq i}^{1} \left[(X - X_j) / (X_i - X_j) \right] \right] * Y_i$$

Quedando finalmente la ecuación de la siguiente forma:

$$\begin{aligned} Y = {} & \left[\left[(X-X_2)*(X-X_3) \right) / \left((X_1-X_2)*(X_1-X_3) \right) \right] * Y1 + \\ & \left[(X-X_1)*(X-X_3) \right) / \left((X_2-X_1)*(X_2-X_3) \right) \right] * Y2 + \\ & \left[(X-X_1)*(X-X_2) \right) / \left((X_3-X_1)*(X_3-X_2) \right) \right] * Y3 \ ] \end{aligned}$$

Para este caso, la curva se realiza en un plano paralelo al XY. El valor de Z es constante.

En resumen, hemos aplicado los algoritmos que nos permitirán manejar la cinemática del brazo robot y generado las trayectorias más óptimas que descibirá la herramienta de pintura.

CAPITULO IV

DESARROLLO DEL SOFTWARE PARA EL DISEÑO DE CONTROL

DIAGRAMA DE FLUJO DE DATOS

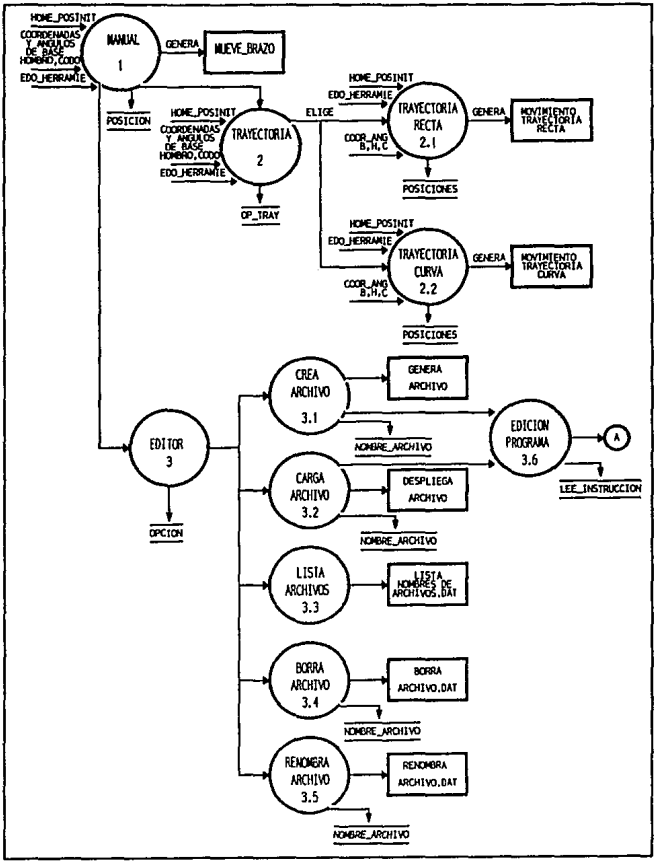
El diagrama de flujo de datos (DFD) describe en forma estructurada el funcionamiento del sistema. El DFD se estructura de componentes ya definidos para su creación como son:

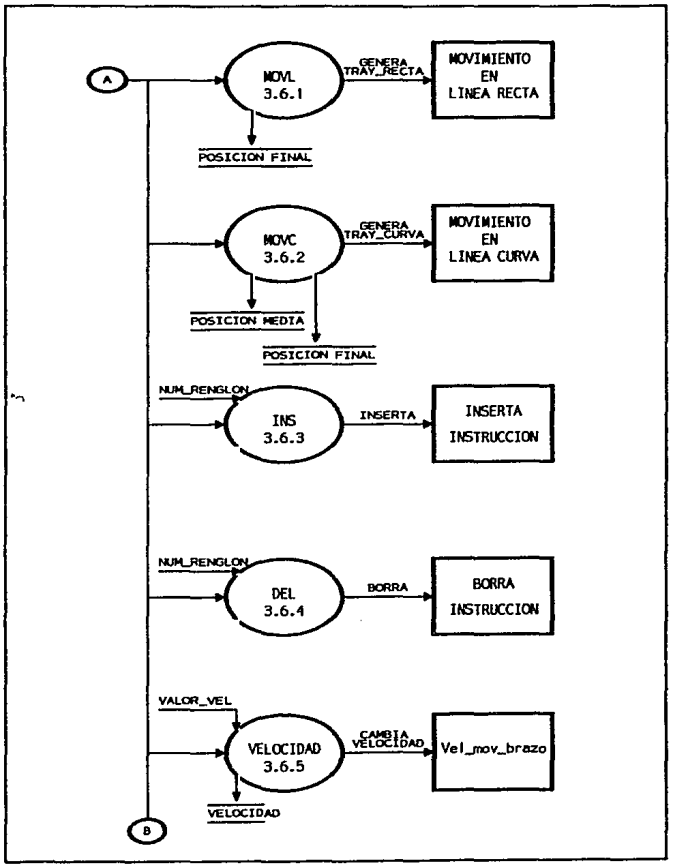
Al conjuntar éstos componentes para el análisis de un sistema de software en general es posible conocer el comportamiento de los procesos relacionados con las posibles entradas y salidas además del flujo de los datos dentro de los mismos.

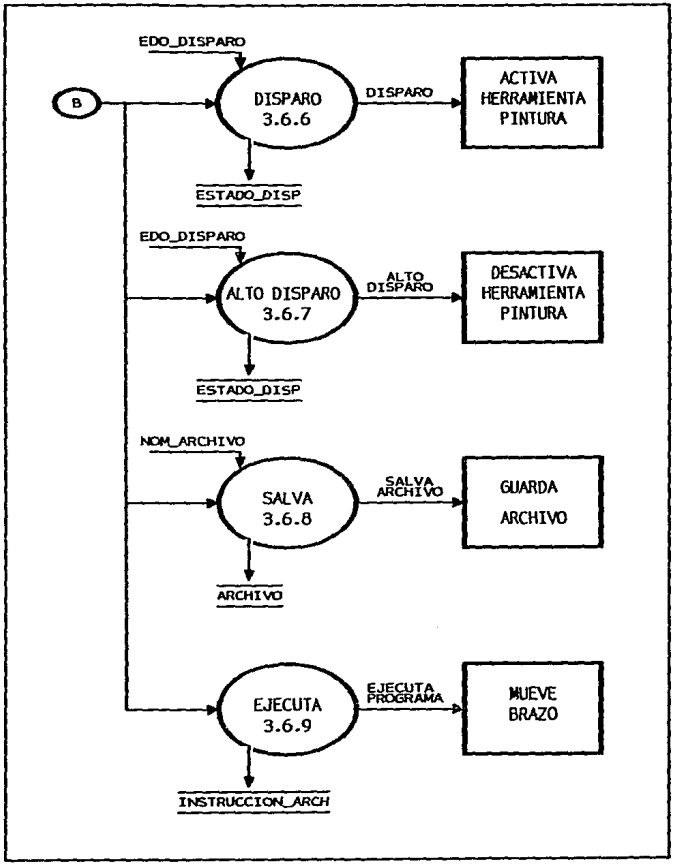
Es muy importante hacer éste tipo de análisis con éstas herramientas pués el programa no es lo suficientemente entendible en su lógica por la forma en que se define la aplicación y sus parámetros además de las reglas de programación. Por lo anterior es necesario tomar en cuenta la Ingeniería de software enfocandose principalmente al total entendimiento de los usuarios que accesan al sistema por primera vez.

A continuación, se presenta el diagrama de flujo de datos del Sistema de Control Flexible explicado bajo este concepto.

DIAGRAMA DE FLUJO DE DATOS


HOME_POSINIT
COORDENADAS Y ANGULOS DE BASE HOMBRO,CODO
EDO_HERRAMIE
MANUAL 1
GENERA
MUEVE_BRAZO
POSICION
HOME_POSINIT
COORDENADAS Y ANGULOS DE BASE HOMBRO,CODO
EDO_HERRAMIE
TRAYECTORIA 2
OP_TRAY
ELIGE
HOME_POSINIT
EDO_HERRAMIE
COOR_ANG B,H,C
TRAYECTORIA RECTA 2.1
GENERA
MOVIMIENTO TRAYECTORIA RECTA
POSICIONES
HOME_POSINIT
EDO_HERRAMIE
COOR_ANG B,H,C
TRAYECTORIA CURVA 2.2
GENERA
MOVIMIENTO TRAYECTORIA CURVA
POSICIONES
EDITOR 3
OPCION
CREA ARCHIVO 3.1
GENERA ARCHIVO
NOMBRE_ARCHIVO
CARGA ARCHIVO 3.2
DESPLIEGA ARCHIVO
NOMBRE_ARCHIVO
LISTA ARCHIVOS 3.3
LISTA NOMBRES DE ARCHIVOS.DAT
BORRA ARCHIVO 3.4
BORRA ARCHIVO.DAT
NOMBRE_ARCHIVO
RENOMBRA ARCHIVO 3.5
RENOMBRA ARCHIVO.DAT
NOMBRE_ARCHIVO
EDICION PROGRAMA 3.6
A
LEE_INSTRUCCION


A
MOVL
3.6.1
GENERA
TRAY_RECTA
MOVIMIENTO
EN
LINEA RECTA
POSICION FINAL
MOVC
3.6.2
GENERA
TRAY_CURVA
MOVIMIENTO
EN
LINEA CURVA
POSICION MEDIA
POSICION FINAL
NUM_RENGLON
INS
3.6.3
INSERTA
INSERTA
INSTRUCCION
NUM_RENGLON
DEL
3.6.4
BORRA
BORRA
INSTRUCCION
VALOR_VEL
VELOCIDAD
3.6.5
CAMBIA
VELOCIDAD
Vel_mov_brazo
VELOCIDAD
B


EDO_DISPARO
B
DISPARO
3.6.6
DISPARO
ACTIVA
HERRAMIENTA
PINTURA
ESTADO_DISP
EDO_DISPARO
ALTO DISPARO
3.6.7
ALTO
DISPARO
DESACTIVA
HERRAMIENTA
PINTURA
ESTADO_DISP
NOM_ARCHIVO
SALVA
3.6.8
SALVA
ARCHIVO
GUARDA
ARCHIVO
ARCHIVO
EJECUTA
3.6.9
EJECUTA
PROGRAMA
MUEVE
BRAZO
INSTRUCCION_ARCH

DICCIONARIO DE DATOS

El diccionario de datos es una parte integral de la especificación estructurada ya que el diagrama de flujo de datos por sí mismo podría proporcionar una idea errónea de lo que está sucediendo en el sistema.

El diccionario se conforma principalmente por la descripción de definiciones como:

* Proceso
* Almacenamiento
* Entidad E/S
* Flujo de datos

En cada una de estas partes se explica detallamente la función de cada uno de los elementos que conforman al sistema, especificando con ello la forma de su comportamiento.

A continuación se da una explicación detallada de cada proceso de ejecución del programa del Control del Brazo Robot.

Nombre Proceso: **MANUAL** No. Proceso: 1
Descripción: Se refiere al manejo del brazo robot en modo manual de las articulaciones principales que lo conforman como el hombro, el codo y la mano, además de controlar el disparo de la herramienta de pintura. Es necesario que antes de manejar el brazo robot, éste haya sido puesto en home (posición inicial) para que el sistema tenga un buen funcionamiento.
Datos de Entrada: home_posinit
coord_ang_hombro
coord_ang_codo
coord_ang_mano
estado_herramienta
Datos de salida: Coordenadas y ángulos de movimiento en hombro, codo, mano y estado de herramienta.
Resumen: El usuario va a mover el brazo manualmente con la ayuda de ciertas teclas específicas para cada articulación.
Estas teclas de movimiento de brazo están especificadas en el Manual de Usuario.

Almacenamiento: Posición

Descripción: Se almacenan las posiciones deseadas del movimiento del brazo y el estado de la herramienta de pintura.

Datos Almacenados: home_posinit
coord_ang_hombro
coord_ang_codo
coord_ang_mano

Entidad: Mueve brazo robot
Descripción: Se generan los movimientos en las articulaciones del brazo robot para ejecutar las opciones que se eligen por medio del teclado.
Datos que entrega: Coordenadas y ángulos de la última posición en las que el brazo robot se ha quedado.
Datos que recibe: Valor del punto siguiente. Si el punto está fuera del campo de trabajo éste movimiento no se realizará.

Flujo de Datos: Genera movimiento
Descripción: Contiene todos los datos que el robot necesita para realizar el movimiento de sus articulaciones.
Datos: Posiciones válidas.

Nombre Proceso: **TRAYECTORIAS** No. proceso: **2**
Descripción: Es un proceso en el cual están definidas las trayectorias recta y circular.
Datos de Entrada: coord_hombro
coord_codo
coord_mano
estado_herramienta
Datos de Salida: Ejecución de la trayectoria en línea recta.
Ejecución de la trayectoria en línea curva.
Resumen: Para ejecutar este proceso, el brazo robot debe tener definidas con anterioridad las posiciones a realizar.

Almacenamiento: Almacena la opción de la trayectoria.
Descripción: Estas opciones son ejecutadas independientemente, debe realizarse primero una trayectoria y volver a elegir la siguiente trayectoria para que ésta sea ejecutada.
Datos Almacenados: Opción trayectoria recta y opción trayectoria circular.

Flujo de Datos: Trayectoria a realizar.
Descripción: Se elige la trayectoria deseada para ir a los procesos de trayectoria recta y trayectoria circular.
Datos: Trayectorias válidas.

Nombre Proceso: **TRAYECTORIA RECTA** No. proceso: **2.1**

Descripción: Al elegir ésta opción es necesario especificar el punto final al que ha de llegar el brazo robot.
Datos de Entrada: Punto inicial en ángulos y coordenadas.

Datos de Salida: Valores de coordenadas y ángulos
Resumen: El funcionamiento principal de éste proceso es el de calcular la trayectoria recta que el robot debe ejecutar. Para ello, se deben haber grabado previamente las 2 posiciones deseadas, como el punto inicial y el punto final.

Almacenamiento: Posiciones.
Descripción: Se refiere a las posiciones inicial y final por donde tiene que pasar la trayectoria recta.

Datos Almacenados: posiciones inicial y final hombro
posiciones inicial y final codo
posiciones inicial y final mano
estado_herramienta

Entidad: Movimiento en línea recta
Descripción: El brazo robot ejecutará la trayectoria recta empezando por el punto inicial y terminando en el punto final.
Datos que entrega: Puntos calculados.
Datos que recibe: Puntos inicial y final.

Flujo de Datos: Trayectoria recta calculada.
Descripción: Contiene los datos de las coordenadas y ángulos d las articulaciones del brazo robot y el estado de la herramienta así como las posiciones inicial y final de la trayectoria recta.
Datos: Punto final válido.

Nombre Proceso: **TRAYECTORIA CIRCULAR** No. proceso: **2.2**
Descripción: Al elegir ésta opción es necesario especificar los puntos medio y final por los que el brazo robot ha de pasar.
Datos de Entrada: Punto inicial, medio y final en ángulos y coordenadas.
Datos de Salida: Valores de coordenadas y ángulos.
Resumen: El funcionamiento principal de éste proceso es el de calcular la trayectoria curva que el robot debe ejecutar. Para ello, se deben haber grabado previamente las 3

posiciones deseadas, como el punto inicial, punto medio y punto final.

Almacenamiento: Posiciones inicial, medio y final
Descripción: Se refiere a las posiciones por las que debe pasar el brazo robot para generar la trayectoria curva.
Datos Almacenados: posiciones inicial, medio y final hombro

posiciones inicial, medio y final codo
posiciones inicial, medio y final mano
edo_herramienta.

Entidad: Movimiento en línea curva.
Descripción: El brazo robot ejecutará la trayectoria curva pasando por los puntos medio y final comenzando por el punto inicial.
Datos que entrega: Puntos calculados.
Datos que recibe: Puntos inicial, medio y final.

Flujo de Datos: Trayectoria curva calculada.
Descripción: Contiene los datos de las coordenadas y ángulos de las articulaciones del brazo robot y el estado de la herramienta así como las posiciones inicial, medio y final de la trayectoria curva.
Datos: Puntos medio y final válidos.

Nombre Proceso: **EDITOR** No. proceso: 3
Descripción: El proceso de editor nos permite fácilmente editar un programa con instrucciones específicas para que el brazo robot las ejecute. Es necesario antes dar el nombre del archivo en el cual se guardarán las instrucciones del programa a crear o cargar.
Datos de Entrada: home_posinit
coord_hombro
coord_codo
coord_mano
edo_herramienta.
Datos de salida: Coordenadas y ángulos de movimiento en hombro, codo, mano, estado de la herramienta y el valor de la velocidad de ejecución.
Resumen: La función de éste proceso es la de tener la facilidad de hacer operaciones con archivos e introducirnos a un editor amigable para crear ó editar un programa.

Almacenamiento: Opción de operación con archivos y edición del programa.
Descripción: Permite el manejo de las operaciones de los archivos como son crear, cargar, listar, borrar y renombrar.
Datos Almacenados: Guarda el valor de la opción elegida para introducirse a uno de los procesos de operaciones con los archivos y de ahí a la edición del programa.

Flujo de Datos: Opción de edición.

Descripción: Contiene el valor de la opción elegida para realizar la operación con el archivo.
Datos: Opción elegida válida.

Nombre Proceso: **CREAR ARCHIVO** No. proceso: **3.1**
Descripción: La función del proceso es crear un archivo tipo texto con el nombre deseado. Los nombres no deben de exceder de 8 caracteres sin extensión.
Datos de Entrada: Nombre archivo
Datos de Salida: Archivo creado
Resumen: La opción de crea archivo no permite que se edite un programa sin antes haber dado un nombre al archivo nuevo.

Almacenamiento: Archivos
Descripción: Se almacena el nuevo nombre del archivo.
Datos Almacenados: Nombre del archivo.

Entidad: Genera archivo
Descripción: Se crea un nuevo archivo que permite almacenar instrucciones de una cierta tarea para el brazo robot.
Datos que entrega: Archivo creado.
Datos que recibe: Nombre del archivo.

Flujo de Datos: Orden para generar archivo.
Descripción: Esta orden contiene el nombre del nuevo archivo a crear para entrar posteriormente al editor.
Datos: Nombre válido

Nombre Proceso: **CARGA ARCHIVO** No. proceso: **3.2**
Descripción: Dar el nombre de un archivo que haya sido previamente creado. Al teclearlo, automáticamente se desplegará la pantalla del editor con el contenido del archivo para su reedición.

Datos de Entrada: Nombre del archivo.
Datos de Salida: Instrucciones que el archivo contenga.
Resumen: Observar siempre que el archivo que se desee cargar ya esté previamente creado, si no lo está, se desplegará un mensaje de error y presentará nuevamente el menú de editor para elegir una nueva opción.

Almacenamiento: Nombre del archivo.
Descripción: Guarda el nombre del archivo que se haya tecleado y que por default ya estaba creado.
Datos Almacenados: Nombre del archivo deseado.

Entidad: Despliega contenido del archivo deseado.
Descripción: Despliega lo que contiene el archivo deseado en la pantalla del editor.
Datos que entrega: Instrucciones almacenadas anteriormente.
Datos que recibe: Nombre del archivo deseado.

Flujo de Datos: Orden de cargar archivo.
Descripción: Esta orden contiene el nombre del archivo a cargar en pantalla del editor.
Datos: Nombre válido.

Nombre Proceso: **LISTAR ARCHIVOS** No. proceso: **3.3**
Descripción: Al elegir ésta opción se desplegarán en pantalla todos los nombres de los archivos que han sido creados.
Datos de Entrada: Opción de lista archivo.
Datos de Salida: Nombres de los archivos.
Resumen: Es una de las opciones importantes para observar los archivos que se han creado.

Entidad: Desplegar nombres de archivos.
Descripción: En pantalla se presentan los nombres de los archivos creados.
Datos que entrega: Nombres de archivos.
Datos que recibe: Nombres de archivos.

Flujo de Datos: Orden de listar archivos.
Descripción: Esta orden contiene los nombres de los archivos a desplegar en la pantalla.
Datos: Opción elegida válida.

Nombre Proceso: **BORRAR ARCHIVO** No. proceso: **3.4**
Descripción: La función del proceso es borrar el nombre de un

archivo así como su contenido. Se debe dar un nombre de no más de 8 caracteres sin extensión.
Datos de Entrada: Nombre archivo.
Datos de Salida: Archivo borrado.
Resumen: La opción de borra archivo permite que un archivo no deseado sea eliminado.

Almacenamiento: Nombre del archivo a borrar.
Descripción: Se almacena el nombre del archivo a borrar.
Datos Almacenados: Nombre del archivo.

Entidad: Borra archivo.
Descripción: Instrucción que quita de memoria el archivo que

ya no se desea.
Datos que entrega: Archivo borrado.
Datos que recibe: Nombre del Archivo.

Flujo de Datos: Orden para borrar archivo.
Descripción: Esta orden contiene el nombre del archivo y su extensión para ser eliminado.
Datos Nombre válido.

Nombre Proceso: **RENOMBRAR ARCHIVO** No. proceso: **3.5**
Descripción: Renombra o dá un nuevo nombre a un archivo que haya sido previamente creado.
Datos de Entrada: Nombre del archivo anterior.
Nombre del archivo nuevo.
Datos de Salida: Archivo con nuevo nombre.
Resumen: Sólo se debe teclear el nombre anterior del archivo y después se debe dar el nuevo nombre del archivo a renombrar.

Almacenamiento: Nombre del archivo anterior.
Nuevo nombre del archivo.
Descripción: Guarda el nombre anterior y el nombre nuevo con el cual va a ser renombrado el archivo.
Datos Almacenados: Nombre archivo anterior.
Nuevo nombre.

Entidad: Renombra archivo anterior
Descripción: Sólo se le dará un nuevo nombre al archivo que se haya querido renombrar.
Datos que entrega: Nuevo nombre de archivo.
Datos que recibe: Nombres anterior y nuevo del archivo a

renombrar.

Flujo de Datos: Orden de renombrar archivo.
Descripción: Esta orden contiene el nuevo nombre del archivo.
Datos: Nombres válidos.

Nombre Proceso: **EDICIÓN PROGRAMA** No. proceso: **3.6**
Descripción: Desplegará una pantalla en la que se pueda editar un programa con instrucciones válidas para el brazo robot.
Datos de Entrada: Archivo creado en memoria.
Datos de Salida: Instrucciones que se guardan en el archivo.
Resumen: El editor permite que se seleccione cualquiera de las instrucciones creando así un programa con una tarea

específica para que el brazo robot la ejecute.

Almacenamiento: Instrucción de editor elegida.
Descripción: Guarda el valor de la instrucción del editor para ser llamada a pantalla.
Datos Almacenados: Clave de la instrucción.

Flujo de Datos: Instrucción
Descripción: Contiene esencialmente la clave de la instrucción que se invocó en el editor.
Datos: Instrucción válida.

Nombre Proceso: **MOVL** No. proceso: **3.6.1**
Descripción: Instrucción base para realizar la trayectoria lineal.
Datos de Entrada: Punto inicial
punto final.
Datos de Salida: Movimiento en línea recta.
Resumen: Para éste proceso es necesario dar el final de la trayectoria recta. Esto por medio de las teclas de flechas (izquierda y derecha). Es importante que el punto dado haya sido almacenado anteriormente.

Almacenamiento: Posición inicial y posición final.
Descripción: Al tener estos puntos el brazo robot dibujará una trayectoria recta.
Datos Almacenados: Coordenadas y ángulos iniciales y finales.
Entidad: Movimiento en línea recta.
Descripción: El brazo robot realizará la trayectoria recta desde el punto inicial al punto final.
Datos que entrega: Ultima posición en la que se quedó.

Datos que recibe: Punto final.

Flujo de Datos: Realiza trayectoria recta.
Descripción: Contiene los puntos inicial y final para que el brazo robot realice la trayectoria recta.
Datos: Punto final y trayectoria válidos.

Nombre Proceso: **MOVC** No. proceso: **3.6.2**
Descripción: Instrucción base para realizar la trayectoria circular.
Datos de Entrada: Punto inicial
punto medio
punto final.
Datos de Salida: Movimiento en línea curva.
Resumen: Para este proceso es necesario dar los puntos medio y final de la trayectoria curva. Esto por medio de las teclas de flechas (izquierda y derecha). Es importante que los puntos hayan sido grabados con anterioridad.

Almacenamiento: Posición inicial
posición media
posición final.
Datos Almacenados: Coordenadas y ángulos iniciales, medios y finales.

Entidad: Movimiento en línea curva.
Descripción: El brazo robot realizará la trayectoria curva desde el punto inicial pasando por el punto medio y llegando al punto final.
Datos que entrega: Ultima posición en la que se quedó.
Datos que recibe: Punto medio y punto final.

Flujo de Datos: Realiza trayectoria curva.
Descripción: Contiene los puntos iniciales, medios y finales para que el brazo robot realice la trayectoria curva.
Datos: Puntos medio, final y trayectoria válidos.

Nombre Proceso: **INS** No. proceso: **3.6.3**
Descripción: La función principal de éste proceso es el de insertar una nueva instrucción en el programa que se está editando.
Datos de Entrada: Renglón a ser insertado.
Datos de Salida: No. renglón a insertar instrucción.
Resumen: Se posiciona el cursor en la línea en donde se desea

insertar la nueva instrucción y se invoca INS para que abra el espacio para ésa instrucción.

Entidad: Inserta renglón en blanco
Descripción: Recorre las instrucciones posteriores a la posición del cursor un renglón hacia abajo e inserta un renglón en blanco para introducir la nueva instrucción.
Datos que entrega: Localidad del renglón en blanco.
Datos que recibe: Número de renglón.

Flujo de Datos: Abre espacio en blanco.
Descripción: Contiene el número de renglón para insertar el blanco.
Datos: Espacio en blanco válido.

Nombre Proceso: **DEL** No. proceso: **3.6.4**
Descripción: La función principal de este proceso es el de

borrar una instrucción del programa que se está editando.
Datos de Entrada: Renglón a ser borrado.
Datos de Salida: Renglón borrado.
Resumen: Se posiciona el cursor en la línea en donde se desea borrar la instrucción y se invoca DEL para que la borre.

Entidad: Borra instrucción.
Descripción: Borra instrucción y recorre las instrucciones posteriores a la posición del cursor un renglón hacia arriba.
Datos que entrega: Localidad de la instrucción borrada.

Flujo de Datos: Borra instrucción.
Descripción: Contiene el número de renglón a ser borrado.
Datos: Renglón válido.

Nombre Proceso: **VELOCIDAD** No. proceso: **3.6.5**
Descripción: Esta instrucción cambia el valor de la velocidad a la que se debe mover el brazo robot.
Datos de Entrada: Valor de velocidad.
Datos de Salida: Movimiento del brazo robot a la velocidad especificada.
Resumen: La selección de la velocidad está dada por medio de las teclas de las flechas horizontales, variando ésta de 1 a 3 (lento, medio y rápido).

Almacenamiento: Velocidad
Descripción: Valor de la velocidad deseada.

Datos Almacenados: Valor de la velocidad.

Entidad: Velocidad del brazo.
Descripción: Esta instrucción cambiará el tiempo de ejecución d
la trayectoria del brazo robot.
Datos que entrega: Velocidad especificada anteriormente.
Datos que recibe: Velocidad especificada.

Nombre Proceso: **DISPARO** No. proceso: **3.6.6**
Descripción: Esta instrucción realiza el encendido de la herramienta de pintura.
Datos de Entrada: Estado del disparo.
Datos de Salida: Estado del disparo.
Resumen: Se realizará el disparo de la herramienta de pintura con instrucciones específicas para el modo manual y en la edición de programas.

Almacenamiento: Estado del disparo.
Descripción: Guarda el estado de encendido de la herramienta de pintura.
Datos Almacenados: Estado de encendido de disparo.

Entidad: Disparo de la herramienta de pintura.
Descripción: Activa el estado de disparo de la herramienta de pintura.
Datos que entrega: Estado de encendido de la herramienta.
Datos que recibe: Estado de la herramienta.

Flujo de Datos: Disparo.
Descripción: Contiene el estado de encendido de la herramienta de pintura.
Datos: Estado de encendido.

Nombre Proceso: **ALTO DISPARO** No. proceso: **3.6.7**
Descripción: Esta instrucción realiza el apagado de la herramienta de pintura.
Datos de Entrada: Estado del disparo.
Datos de Salida: Estado del disparo.
Resumen: Se realiza el alto del disparo de la herramienta de pintura con instrucciones específicas para el modo manual y en la edición de programas.

Almacenamiento: Estado de alto de disparo.

Descripción: Guarda el estado de apagado de la herramienta d e pintura.
Datos Almacenados: Estado de apagado de disparo.

Entidad: Alto de disparo de la herramienta de pintura.
Descripción: Activa el estado de alto de disparo de la herramienta de pintura.
Datos que entrega: Estado de apagado de la herramienta.
Datos que recibe: Estado de la herramienta.

Flujo de Datos: Alto disparo.
Descripción: Contiene el estado de apagado de la herramienta de pintura.
Datos: Estado de apagado.

Nombre Proceso: **SALVA PROGRAMA** No. proceso: **3.6.8**
Descripción: Su función es la de almacenar el conjunto de instrucciones que se encuentran editadas en un archivo tipo texto.
Datos de Entrada: Nombre del Archivo.
Datos de Salida: Archivo almacenado.

Resumen: Se realiza el almacenamiento de las instrucciones en un archivo predefinido.

Almacenamiento: Archivo tipo texto.
Descripción: Guarda el conjunto de instrucciones que debe de realizar el brazo robot.
Datos Almacenados: Instrucciones.

Entidad: Salva archivo tipo texto.
Descripción: Almacena la serie de instrucciones deseadas.
Datos que entrega: Archivo tipo texto con las instrucciones deseadas.
Datos que recibe: Serie de instrucciones en el orden deseado y el nombre del archivo.

Flujo de Datos: Almacena archivo.
Descripción: Contiene el nombre del archivo.
Datos: Nombre de archivo correcto.

Nombre Proceso: **EJECUTA PROGRAMA** No. proceso: **3.6.9**
Descripción: Su función es la de ejecutar las instrucciones almacenados en archivos.

Datos de Entrada: Instrucciones.
Datos de Salida: Ángulos y coordenadas para el movimiento del brazo robot, estado de la herramienta de pintura y valor de la velocidad de movimiento.
Resumen: Se realiza la ejecución de las instrucciones almacenadas en el archivo.

Entidad: Mueve Brazo.
Descripción: Realiza el movimiento del brazo robot.
Datos que recibe: Serie de instrucciones.

Flujo de Datos: Ejecuta programa.
Descripción: Contiene todos los datos necesarios de las instrucciones de un programa para que el brazo robot realice una tarea específica.
Datos: Instrucción válida.

CAPITULO V

DESARROLLO DEL HARDWARE PARA EL DISEÑO DE CONTROL

DESARROLLO DEL HARDWARE PARA EL DISEÑO DE CONTROL

V ANALISIS DEL HARDWARE

En este capítulo se hablará de los dispositivos que se utilizaron para implementar el brazo robot, estos son:

V.1 Una computadora personal
V.2 Una etapa digital de señal pequeña (tarjeta controladora de motores de pasos y driver)
V.3 Una etapa de potencia (Digital-Analógica)

V.1 COMPUTADORA PERSONAL

En una computadora personal se desarrolló el programa que manda los pulsos de control a la etapa digital de señal pequeña. Este programa toma en cuenta el área de trabajo del robot además de las trayectorias de desplazamiento en las que se moverá el brazo robot mediante diferentes velocidades de movimiento. El efector final utilizado será una herramienta de pintura.

V.2 ETAPA DE SEÑAL PEQUEÑA

La etapa digital de señal pequeña se transforma en patrones de pulsos que se reciben de la computadora transmitiéndose a los motores de pasos. Es importante mencionar que la tarjeta contiene memoria RAM que permite almacenar la información liberando al microprocesador de la computadora para que realice otros procesos.

V.2.1 TARJETA CONTROLADORA DE MOTOR DE PASOS

Asociado al desarrollo de los motores de pasos, surgieron circuitos integrados y tarjetas especiales justamente proyectados para controlar estos nuevos elementos mecánicos íntimamente ligados a la "robótica".

La tarjeta controladora de motor de pasos que se utilizó fue la PCL-838, la cual controla hasta 3 motores de pasos simultáneamente y carga las operaciones de datos de la computadora para generar pulsos en cada uno de sus tres canales.

La tarjeta PCL-838 tiene las siguientes características principales:

- Control independiente y simultáneo de hasta tres motores
- Interpretador de comandos de alto nivel independiente del lenguaje
- Velocidad programable de 1 a 10,000 pps (pulsos por segundo)
- Velocidad inicial y final programadas para obtener una aceleración y desaceleración lineal
- Soporta un reloj (pulso y dirección y dos relojes) cuya salida es compatible con TTL.
- Cuenta con 24 entradas y salidas digitales compatibles con TTL.
- La tarjeta se encuentra optoacoplada.

La figura de la tarjeta controladora de motor de pasos PCL-838 es el siguiente:

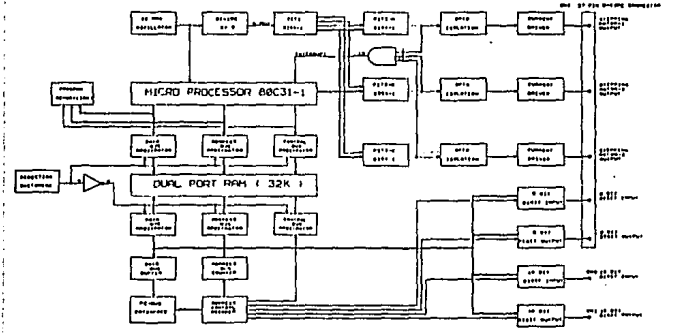


fig 1 Diagrama de Conexiones de la Tarjeta PCL-838

La tarjeta de control de pasos PCL-838 convierte cualquier PC en una estación multiejes y multicontrol. Se puede instalar más de una tarjeta PCL-838 en una PC.

V.2.1.1 JUSTIFICACION DE LA TARJETA

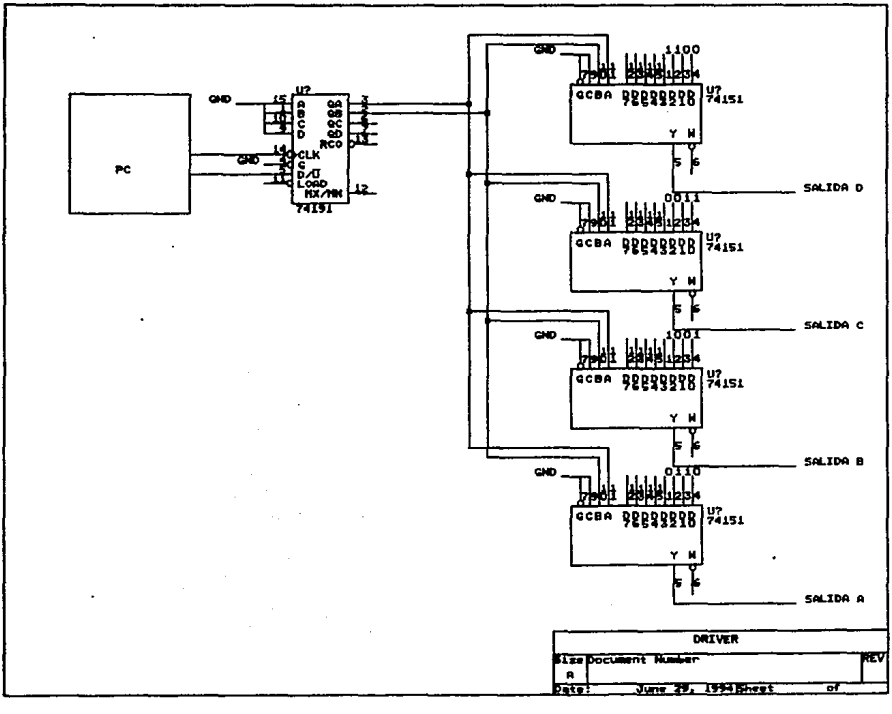
Se podría pensar que el tiempo que tarda la PC en mandar pulsos a los motores de pasos es relativamente poco importante, sin embargo, dado que el robot podría estar en una línea de producción el tiempo es vital y la optimización de éste se traduce en mayor producción y eficiencia, es por esto que se decidió utilizar una tarjeta para hacer que el sistema trabaje en tiempo real (proceso que se realiza y calcula simultáneamente).

En cuanto a la operación del robot mediante el sistema de software diseñado, el código de programa disminuye notablemente y permite hacer mantenimiento del programa únicamente cambiando los parámetros de los comandos que se integran para usar la tarjeta. Es importante comentar que en los prototipos no se utilizó la tarjeta controladora de motores de pasos y después de éstos se decidió optimizar el desempeño del robot industrial y de la PC.

El uso de la tarjeta lleva la ventaja adicional de contar con un reloj interno para llevar a cabo sus procesos. Al utilizar una PC, la generación de pulsos siempre está en función de la velocidad del procesador de la computadora por lo que se tienen que sincronizar cada vez que se cambiara de máquina pués afecta la operación total del robot. La tarjeta calcula automáticamente la aceleración y desaceleración del robot durante su operación evitando al máximo las vibraciones producidas debido a la inercia.

V.2.2 DRIVER

El driver, es el circuito que genera la secuencia para controlar el movimiento de un motor de pasos para lo cual se necesita que la computadora mande dos señales por cada motor de pasos. La primera señal consiste en la dirección en la cual deberá moverse el motor de pasos y la segunda es el tren de pulsos que indicará la velocidad y trayectoria en la cual se moverá el brazo robot. Es importante comentar que por cada motor de pasos existe un driver y el movimiento de cada articulación está sincronizado mediante el programa que genera los pulsos trabajando cada uno simultáneamente.


PC
GND
U2
74191
CLK
G
D/U
LOAD
MAX/MIN
RCO
1100
0011
1001
0110
U?
74151
SALIDA D
SALIDA C
SALIDA B
SALIDA A
DRIVER
Size
A
Document Number
REV
Date:
June 29, 1994
Sheet
of

V.3 ETAPA DE POTENCIA

Para la etapa de potencia, se diseñó una fuente de voltaje y se utilizó una tarjeta que contiene relevadores proporcionando la corriente necesaria a los motores para mover el brazo robot.

La fuente de voltaje tiene 3 salidas para cada motor de pasos a controlar. Como principales características, en cada una de ellas, se tiene un voltaje de CD de salida y una corriente máxima de 6 A.

V.3.1 MOTORES DE PASOS

Cada articulación del brazo robot es controlada por motores de pasos. Los motores de pasos usan 4 bobinas las cuales son manejadas por una etapa de potencia. El manejo es digital mediante cambios lógicos de nivel (1,0) en los pulsos para obtener el patrón deseado de corrientes en el motor. Al cambiar el patrón de corrientes, un campo magnético rotatorio es obtenido dentro del motor lo que causa que el motor gire en pequeños incrementos o pasos.

Los motores de pasos, consisten en elementos especiales de la familia de los motores de corriente continua, siendo dotados de diversos embobinados como se muestra en la figura.

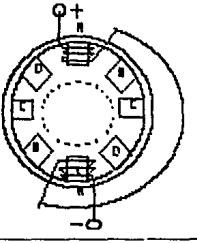


fig 2 Vista (arriba) de un motor de pasos

Para girar el eje de este tipo de motor se aplicaron pulsos, cada pulso hace que el eje se desplace un ángulo preciso que varía normalmente entre 1.8 y 7.5 grados. Con una secuencia apropiada de pulsos es posible hacer que el eje gire en cualquier dirección y además, se ubique en lugares precisos, múltiplos del valor del paso.

La figura anterior muestra un motor de 4 fases. Se observa que el rotor está formado por una especie de rueda dentada que se ubica siempre en relación a la bobina que es energizada.

Para calcular el valor del paso del motor es necesario aplicar la siguiente fórmula:

$$X = \frac{360}{(f * n)}$$

donde X es el valor del paso en grados
f es el número de fases
n es el número de dientes del rotor

Las razones principales para utilizar motores de pasos son un costo bajo y un fácil control mediante el uso de una computadora.

Con el fin de que el motor de pasos gire, una secuencia particular en binario se necesita como entrada al motor, es decir, una secuencia binaria por paso. Para cambiar la dirección del motor la secuencia en binario se invierte.

Un dato importante del brazo robot es la relación entre los pasos del motor y la rotación de las articulaciones del brazo.

Por otro lado, el torque es la fuerza rotatoria que produce un motor. La salida de torque del motor de pasos varía con su velocidad, a baja velocidad el torque es máximo pero al incrementar repentinamente la velocidad el motor podría no resbalar y no se obtendría ningún torque. El sistema de software determina la máxima velocidad permitida para evitar que el brazo resbale, asímismo, también es tomado en cuenta el peor caso en cuanto a que los miembros del brazo estén a su máxima extensión horizontal requiriendo con ello el máximo torque del motor.

Así pués, moviendo el brazo robot de una posición a otra, se requiere de la rotación de más de una de sus articulaciones de manera coordinada como en el caso nuestro. Existen robots cuyo

movimiento es secuencial, es decir, cada articulación se mueve en diferente tiempo para llegar a la posición indicada.

En general, el movimiento coordinado es más suave y rápido que el movimiento secuencial. El sistema de software está diseñado para producir un movimiento coordinado utilizando tres motores de pasos para mover el brazo de una posición a otra. Para llevar a cabo la coordinación, el motor de pasos está temporizado, es decir, a cada motor le llega un tren de pulsos diferentes a intervalos regulares de tiempo durante el movimiento.

V.3.2 FUENTE DE PODER

El análisis que a continuación se menciona se limita a las fuentes de alimentación que transforman la potencia de entrada de corriente alterna en potencia de salida de corriente directa. La función de la fuente de alimentación regulada es mantener el voltaje constante en las terminales de salida. En nuestro caso, se diseñó 1 fuente de alimentación para cada uno de los motores de pasos, debido a que son de desecho, cada uno de ellos tiene parámetros de operación diferentes (voltaje de polarización, corriente en cada devanado, etc.).

Los parámetros básicos que describen la fuentes de alimentación son:

a) Ondulación de rizo: Componente de CA que se superpone sobre la componente de CD.

b) Regulación de línea: Cambio del valor de estado permanente de voltaje de salida que acompaña a un cambio de voltaje de línea, con todas las demás condiciones constantes.

c) Regulación de carga: Cambio del valor del estado permanente del voltaje de salida que acompaña a un cambio en la corriente de carga, con todas las demás condiciones constantes.

La fuente de alimentación es del tipo rectificador de puente de onda completa bipolar y las especificaciones para esta fuente son:

- Corriente de carga de 0 a 6 Amp
- Voltaje de entrada de 127 V de CA 60 Hz
- Voltaje de salida de 0 a 5 V de CD
- Capacitor de filtro de 1000 microfaradios, 16V

- Puente de diodos de 7 Amps

Las etapas de la fuente de alimentación son:

A) ETAPA REDUCTORA

El voltaje de línea es reducido a un nivel que pueda ser usado en el circuito. Analizando los datos proporcionados por las hojas de datos de los reguladores utilizados se mandó a construir un transformador que diera la corriente y el voltaje necesarios para la alimentación de cada motor.

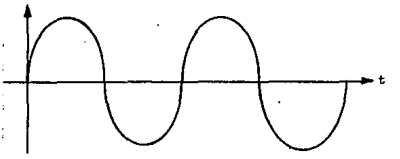


fig 3 Voltaje de CA

B) ETAPA RECTIFICADORA

Se usó un rectificador mediante un puente de diodos para convertir las componentes negativas de la señal en positivas. El valor en corriente de éste puente es de 6 A.

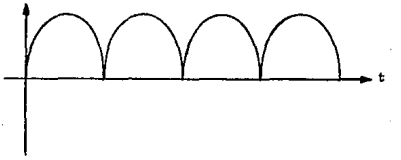


fig 4 Voltaje Contínuo Pulsante

C) ETAPA DE FILTRADO

Se utilizó un filtro para eliminar las ondulaciones o rizo de la señal entregando una señal de CD a la salida. Este filtro es un capacitor.

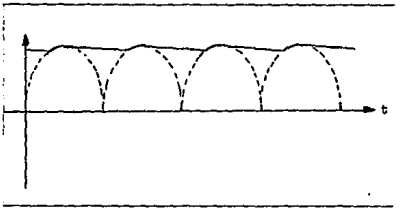


fig 5 Voltaje de CD

D) ETAPA REGULADORA

En esta etapa se usaron reguladores (Circuitos Integrados) los cuales, como su nombre lo indica, mantienen el voltaje de salida en un rango deseado.

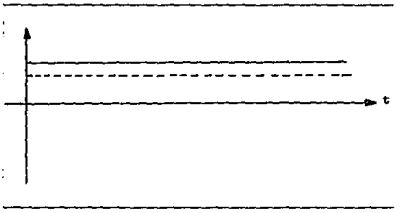
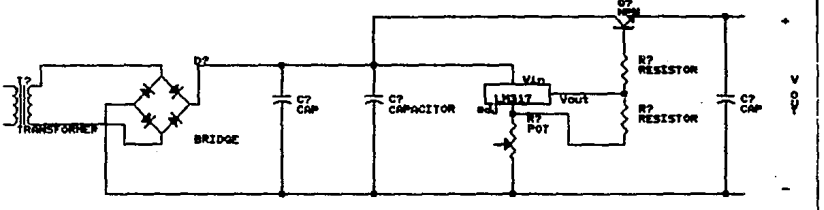


fig 6 Voltaje CD por uso de CI

Resumiendo, un circuito rectificador convierte un voltaje de alterna en voltaje continuo pulsante, un circuito de filtro disminuye las variaciones que presentan dichos pulsos para obtener un voltaje de corriente directa el cual es regulado mediante CI.

A continuación se presenta el diagram de la fuente de poder:



FUENTE DE PODER DE VOLTAJE VARIABLE		
Size A	Document Number	REV
Date: June 29, 1994	Sheet of	

V.3.2.1 JUSTIFICACION DE LA FUENTE DE PODER

La fuente de poder polarizará a los motores y deberá ser de un voltaje variable, ya que como se dijo anteriormente, los motores de pasos tienen características diferentes por ser de desecho.

V.3.3 TARJETA CON RELEVADORES DE ESTADO SOLIDO (SSR)

Se utilizó una tarjeta PCLD-786 que tiene un manejador de relevadores de potencia de estado sólido de AC y DC (SSR Solid State Relay). La PCLD-786 es una tarjeta que puede controlar hasta 8 módulos de relevadores de estado sólido y hasta 8 relevadores externos. Está diseñada para manejar 16 entradas digitales mediante un puerto de 20 pines. La tarjeta está optoacoplada y aislada entre la etapa digital y analógica.

Los módulos de DC (Corriente Directa) proveen alta confiabilidad, rápida respuesta en tiempo y alta inmunidad al ruido por lo cual no están constituídos por sus contrapartes mecánicas.

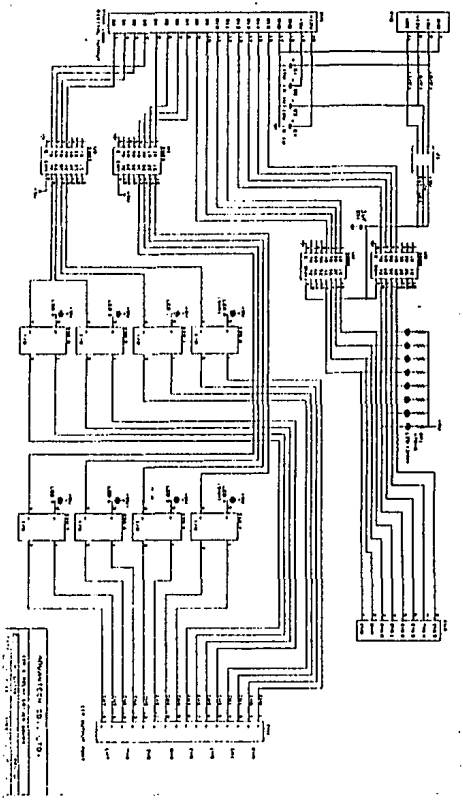
La tarjeta tiene 8 salidas digitales para manejar relevadores externos. Los voltajes a manejar pueden ser seleccionados a +5V, +12V o voltaje externo de DC (en caso de que la fuente de poder se sobrecargue).

Cada módulo está equipado con un led para monitorear el estado en el que se encuentra el relevador de estado sólido.

Especificaciones:

- Tipo de módulo PCLD-ODC5.
- Voltaje de salida de operación de 5 a 60 V de CD a 3 A.
- Corriente transitoria de la fuente en 1 seg igual a 5 A.
- Tiempo de encendido/apagado = 750 microsegundos máx (1333 Hz).

Por todo lo anterior, a continuación se presentan los diagramas que conforman a la tarjeta PCLD-786 y las características de los módulos de relevadores de estado sólido PCLD-ODC5



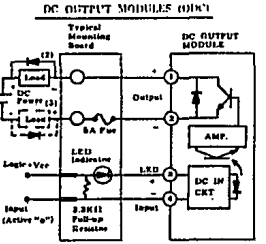


fig 9 PCLD-ODC5

V.3.3.1 JUSTIFICACION DE LA TARJETA DE ESTADO SOLIDO

Durante la realización de la tesis, se emplearon diversos drivers (circuito controlador de motores de pasos) para manejar prototipos a escala del brazo robot; se observó que para cada prototipo se tenía que implementar un circuito que se adecuara a las características eléctricas de los motores de pasos ya que se necesitaba una corriente diferente.

Con el fin de dar la mayor flexibilidad al usuario final del robot industrial, se decidió tener una tarjeta que no sólamente pudiera manejar motores de pasos pequeños sino que permitiera el uso de motores de DC cuya corriente no necesitara de diseñar otro circuito, sino la simple adición de un relevador externo que pudiera manejar toda la corriente que necesitara el motor.

CAPITULO VI

PRUEBAS DE CAMPO

PRUEBAS DE CAMPO

En este capítulo se dividieron las pruebas de campo en tres partes principales:

- Pruebas del Software
- Pruebas de hardware para el diseño del control
- Acoplamiento de ambas etapas en el prototipo final

VI.1 PRUEBAS DEL SOFTWARE

Las pruebas del software se realizaron en un principio con la ayuda de interfases que permitían conocer las señales que eran enviadas desde la computadora al brazo robot pudiendo comprobar la buena operación del programa y de su manejo interactivo con el usuario.

Se analizaron los modelos matemáticos que permitían determinar la trayectoria descrita por el brazo robot a partir de los pulsos que generaba el programa.

Asimismo, al lograr que el brazo se moviera, el siguiente paso fue el de generar los tipos de trayectorias en los cuales se iba a desplazar el robot como la trayectoria recta y curva. También fue necesario crear un editor que le permitiera al usuario editar sus propios programas que indicaran el trabajo a desarrollar por el brazo robot.

Finalmente, se presentó al usuario un programa interactivo, sencillo y fácil de utilizar.

VI.2 PRUEBAS DEL HARDWARE

Las pruebas de hardware se realizaron mandando un tren de pulsos mediante un generador de funciones y un circuito diseñado por nosotros para generar la secuencia que permitiría el movimiento bidireccional de un motor de pasos. Es importante mencionar que ya generada la secuencia digital, ésta era enviada a una tarjeta de relevadores de estado sólido que permitían enviar los pulsos de corriente a los motores de pasos.

Dentro de estas pruebas fue de gran trascendencia la caracterización de los motores de pasos ya que ninguno tenía sus datos de placa completos y hubo que analizar experimentalmente los datos faltantes.

VI.3 ACOPLAMIENTO DE AMBAS SEÑALES AL PROTOTIPO FINAL

Por último, se tomaron las señales que proporciona la computadora para manejar tres motores de pasos y la acción de una pistola para pintura concluyendo el proyecto de Tesis.

En esta etapa se observó la importancia de tomar en cuenta el área de trabajo del brazo robot ya que una falla en el software podría provocar un daño físico al robot y/o a todo el hardware que lo maneja.

A pesar que la computadora es capaz de enviar los pulsos a una frecuencia muy alta, existe la limitante de que el brazo robot no podría moverse a esa velocidad ya que a mayor frecuencia menor torque o par en el motor y llegaría el momento en el cual el brazo ya no podría continuar desplazándose.

Finalmente, ya acopladas las etapas, los circuitos fueron colocados dentro de varias cajas metálicas para protegerlos del medio ambiente y de un eventual contacto por parte del usuario dejando únicamente en el exterior los controles necesarios para la correcta operación del brazo robot.

CAPITULO VII

CONCLUSIONES

CONCLUSIONES

Al finalizar el trabajo de Tesis, nos dimos cuenta de varios detalles que no habíamos experimentado en nuestra etapa como estudiantes, entre ellas están:

- El proceso de investigación que nos ayudó a conocer todos los conceptos involucrados con robótica.
- La interacción en trabajos multidisciplinarios con personas con conocimientos en otras áreas diferentes a la computación.
- La aplicación de los conceptos que aprendimos en la División de Ciencias Básicas en problemas matemáticos que se fueron presentando, dándole la importancia que realmente tienen.
- El diseño de circuitos en función de requerimientos propios.
- La utilización de tecnología existente acoplandola dentro de un sistema para simplificar las modificaciones que se pudieran realizar a futuro dando una mayor vida útil al Sistema de Manufactura Flexible.
- La investigación de antecedentes con personas con experiencia en las áreas involucradas en el SMF en las cuales se ha desarrollado tecnología atendiendo a sus sugerencias y consejos.
- El trabajo en equipo que nos permitió hacer más eficiente las soluciones a los problemas que se presentaron durante el desarrollo de la tesis.
- El uso global de los conocimientos adquiridos durante nuestra etapa como estudiantes dentro de la Facultad de Ingeniería, en la cual fue muy importante la metodología en la cual se planteaban y resolvían los problemas.
- La detección y corrección de las fallas que se ocasionaban tanto por software como por hardware durante la operación de cada una de las etapas en las que se fue desarrollando el SMF.

- Evitar que la presión psicológica que se genera al no poder resolver un problema se traduzca en tensiones que impidan seguir avanzando en el trabajo.
- Trabajar con la robótica como una área de desarrollo que actualmente tiene mucho auge y aplicación en las industrias de nuestro país.

CAPITULO VIII

BIBLIOGRAFIA

BIBLIOGRAFIA

- D. McCloy, D. M. J. Harris, ROBOTICA, UNA INTRODUCCION, Limusa, México, D.F. 1993
- Manual PCL-838, STEPPING MOTOR CONTROL CARD
- Guilder, J., "Focus on Steppin Motors", ELECTRONIC DESIGN, U.S.A Octuber 25, 1977
- Paul Richard, ROBOT MANIPULATORS: MATHEMATICS, PROGRAMMING AND CONTROL, Massachusetts Noviembre 1981
- Mark W. Spong. M. Vidyasagar, ROBOT DYNAMICS AND CONTROL, John Wiley & Sons. 1989.
- John J. Craig. INTRODUCTION TO ROBOTICS. MECHANICS AND CONTROL, Second Edition 1989.
- Vukobratovic. Potkonjak, DYNAMICS OF MANIPULATION ROBOTS. THEORY AND APPLICATION, Springer-Verlag, 1982.
- Ranjan Mukherjee. Yoshihico Nakumara, FORMULATION AND EFFICIENT COMPUTATION OF INVERSE DYNAMICS OF SPACE ROBOT, Transactions on Robotics and Automation. Vol. 8 Num. 3. June 1992.
- Mikell P. Groover. et al, INDUSTRIAL ROBOTICS. TECHNOLOGY, PROGRAMMING AND APPLICATIONS, Mc Graw-Hill. International Editions. 1987.
- Ignacio Juárez Campos, DISEÑO Y CONSTRUCCIÓN DE UN MANIPULADOR DE 7 GRADOS DE LIBERTAD, Segunda Reunión Nacional de Estudiantes de Posgrado en Ingeniería Mecánica. Facultad de Ingeniería Mecánica, Eléctrica y Electrónica. Universidad de Guanajuato. Salamanca, Gto. 1992
- J. L. Meriam and L. G. Kraige, DYNAMICS, Second Edition. Volume 2. Wiley. 1987
- William R. Tanner, INDUSTRIAL ROBOTS, First Edition. Volume 1. 1979.
- ELECTRÓNICA PRÁCTICA RESISTOR, Año 1, Num. 2. Suplemento especial. Noviembre 1992.

- ELECTRÓNICA PRÁCTICA RESISTOR, Año 1, Num. 3. Suplemento especial. Diciembre 1992.
- ELECTRÓNICA PRÁCTICA RESISTOR, Año 2, Num. 1. Suplemento especial. Enero 1993.
- ELECTRÓNICA PRÁCTICA RESISTOR, Año 2, Num. 2. Suplemento especial. Febrero 1993.
- ELECTRÓNICA PRÁCTICA RESISTOR, Año 2, Num. 3. Suplemento especial. Marzo 1993.
- ELECTRÓNICA PRÁCTICA RESISTOR, Año 2, Num. 4. Suplemento especial. Abril 1993.
- ELECTRÓNICA PRACTICA RESISTOR, Año 2, Num. 5. Suplemento especial. Mayo 1993.
- ELECTRÓNICA PRÁCTICA RESISTOR, Año 2, Num. 6. Suplemento especial. Junio 1993.
- ELECTRÓNICA PRÁCTICA RESISTOR, Año 2, Num. 7. Suplemento especial. Julio 1993.
- ELECTRÓNICA PRÁCTICA RESISTOR, Año 2, Num. 8. Suplemento especial. Agosto 1993.
- SKF, Catalogo General, 1989.
- Catálogo de Bandas y Poleas BRECO FLEX. TIMING BELTS, TRANSMISSION DEVELOPMENTS Co. LTD Germany.
- Catálogo Hi-Drive Systems, ESPECIFICACIONES DE SERVOMOTORES ELÉCTRICOS DE CORRIENTE CONTINUA, Tokio, Japón.
- Tsuboi, Y., A MINI-COMPUTER CONTROLED INDUSTRIAL ROBOT WITH OPTICAL SENSORS IN GRIPPER, Cambridge, 1973
- Will, P. M., COMPUTER CONTROLLED MECHANICAL ASSEMBLY, Chicago, Illinois 1975
- Denavit J & Hartenberg, R. S. (1955), A KINEMATIC NOTATION FOR LOWER PAIR MECHANICS BASED ON MATARICES, Trans. ASME Vol. 77 June 1955

APENDICE A

MANUAL DEL USUARIO

I OBJETIVO PRINCIPAL DEL SOFTWARE

El objetivo general del sistema, es controlar el movimiento de un brazo mecánico de tres articulaciones con el fin de que realice una tarea especifica que puede ser reprogramada cuantas veces se desee.

Se utiliza como comunicación base una interfaz electrónica que convierte los grados a los que se deberá mover cada articulación en pulsos con la potencia suficiente para mover el motor de cada articulación.

II ARRANQUE DEL SISTEMA

Para que el sistema pueda ser utilizado, se debe contar con una Computadora que maneje cualquier versión de Sistema Operativo MS-DOS.

Posteriormente se verificara que el Brazo Robot este conectado a la computadora y a cualquier línea de corriente.

Si lo anterior ha sido revisado, se enciende la computadora y se llama al software del sistema llamado ROBOT.

- Si se tiene instalado un disco duro aparecerá el prompt c> y solo se tecleara ROBOT quedando de la siguiente forma:
c> ROBOT

- Si no se cuenta con disco duro aparecerá el prompt a> y se hará lo mismo que en el punto anterior quedando:
a> ROBOT

III MODULOS DEL SOFTWARE

El sistema consta de cuatro modulos identificados con los siguientes nombres:

- **MENÚ MANUAL**
  Con este modulo se puede mover el brazo robot en forma manual, así es posible inicializar las funciones mandando el brazo a Home. Posteriormente,

se podrá mover cada articulación del brazo independientemente.

- **MENÚ AUTOMÁTICO**
  Con este modulo se puede mover el brazo robot en forma automática, así es posible que su trayectoria de un punto a otro la realice en forma recta o circular.

- **MENÚ EDITOR**
  Se basa principalmente, en la creación de programas que contengan instrucciones de la tarea que deberá realizar el brazo robot.

- **TERMINAR**
  Esta opción nos permite terminar la sesión de trabajo saliendo al Sistema Operativo.

IV MODULO DEL MENÚ MANUAL

Al elegir esta opción, se presentara una ventana con tres opciones:

1 HOME
Con esta opción se inicializan las funciones del brazo robot.
Nota: ES INDISPENSABLE EMPEZAR CON ESTA OPCIÓN PARA EL BUEN FUNCIONAMIENTO DEL SISTEMA.

2 MOVER BRAZO
Si se escoge esta opción, aparecerán nuevas instrucciones que harán posible el movimiento del brazo robot a donde se desee. La posición puede ser grabada en cualquier numero de localidad de memoria entre el 1 y el 9.

Aparecen también los valores de los grados y las coordenadas en las que el brazo robot se encuentra.

Si se desea salir del modo manual, la opción de ESC regresa el programa al Menú Principal.

3 SALIR
Con esta opción, se regresa el programa al Menú Principal.

V MODULO DEL MENÚ AUTOMÁTICO

Esta opción realiza trayectorias recta y circular, por lo tanto, es necesario tener de antemano grabadas, las posiciones del brazo robot que se utilizaran para el desarrollo de una tarea deseada.

Se presenta una ventana con tres opciones:

TRAYECTORIA

1 **RECTA**
Al elegir esta opción, se desea que el brazo robot describa una trayectoria recta ente dos puntos. Se pedirá solo el Punto Final de la trayectoria pues el Punto Inicial es aquel en donde el brazo robot se encuentra.

2 **CIRCULAR**
Al elegir esta opción, se desea que el brazo robot describa una trayectoria curva ente tres puntos. Se pedirán el Punto Medio y el Punto Final de la trayectoria, pues el Punto Inicial es aquel en donde el brazo robot se encuentra.

3 **SALIR**
Con esta opción, se regresa el programa al Menú Principal.

VI MODULO DEL EDITOR

Al elegir este menú, se presentara una ventana con las siguientes opciones:

1) **CREAR PROGRAMA**
Al entrar a esta opción, se presentara una ventana en la que se pide el nombre del archivo a crear; en el se almacenara la serie de instrucciones de la tarea deseada.El nombre constara de máximo 8 caracteres (no se debe poner extensión).
Si el nombre ya existe se desplegara una nota para rescribir o no el archivo.Posteriormente, se desplegara un editor con una ayuda de instrucciones en la parte

inferior de la pantalla.
Para elegir alguna opción del menú de ayuda, solo se deberán presionar las teclas de funciones especiales como son F1..F7, INS y DEL.A continuación, se explicara cada instrucción de la ayuda:

F1 **MOVEL** Opción que realiza una trayectoria recta. Al elegirla, se pide el Punto Final y el valor de la localidad se dará con las flechas horizontales, oprimiendo ENTER para aceptar la localidad.

F2 **MOVEC** Opción que realiza una trayectoria curva. Al elegirla, se piden el Punto Medio y el Punto Final. Los valores de las localidades se darán con las flechas horizontales oprimiendo ENTER para aceptar cada punto.

F3 **VELOCIDAD** Opción que cambia la velocidad de movimiento del brazo. Al elegirla, se pide el Valor de Velocidad a la que se desea se ejecute la tarea. El valor de la velocidad se encuentra en un rango de 1 a 3 y se elige con las flechas horizontales, oprimiendo ENTER para aceptar la velocidad.

F4 **DISPARO** Opción que acciona la pistola de pintura.

F5 **ALTO** Opción que desactiva la pistola de pintura.

F6 **MENÚ** Opción que almacena el programa en memoria regresando automáticamente al Menú de Editor.

F7 **EJECUTA** Opción que ejecuta el programa que esta editado en pantalla.

INS **INSERTA LINEA** Opción que inserta una linea en la posición que se encuentre el cursor.

DEL **BORRA LINEA** Opción que borra la linea en la que se encuentra el cursor. El cursor puede desplazarse sobre las

instrucciones del programa
utilizando las flechas verticales.

Si se desea salir del Editor sin salvar el programa se puede utilizar la tecla de ESC.

2) **CARGAR PROGRAMA**
Al entrar a esta opción, se presentara una ventana en la que se pide el nombre del archivo a llamar. Si el archivo no existe marca un ERROR y regresa nuevamente al Menú de Editor, en caso contrario desplegara el editor con el programa elegido.

3) **LISTAR PROGRAMAS**
Al entrar a esta opción, se presentara una ventana en la que se desplegaran todos los archivos creados con anterioridad.

4) **BORRAR PROGRAMA**
Al entrar a esta opción, se presentara una ventana en la que se pide el nombre del archivo a borrar. Si el archivo no existe marca un ERROR y regresa nuevamente al Menú de Editor, en caso contrario desplegara otra ventana en la que pregunta si realmente se desea borrar o no dicho archivo.

5) **RENOMBRAR PROGRAMA**
Al entrar a esta opción, se presentara una ventana en la que se pide el nombre del archivo a renombrar. Si el archivo no existe marca un ERROR y regresa nuevamente al Menú de Editor, en caso contrario desplegara otra ventana en la que pregunta el nuevo nombre del archivo.

6) **SALIR**
Con esta opción, se regresa el programa al Menú Principal.

APENDICE B

SISTEMA DE SOFTWARE DE CONTROL DEL BRAZO ROBOT

```
program union;
uses crt,dos,unidad,graph;
const
     l1=10;      {longitud del eslabon 1}
     l2=10;      {longitud del eslabon 2}
     l3=10;      {longitud del eslabon 3}
type
    posicion= recorD      {registro que almacena la posicion en
                          coordenadas}
              x:real;     {cartesianas y polares de un punto}
              y:real;
              z:real;
              th:real;
              tc:real;
              tm:real;
              end;
    arre = array [1..10] of posicion;
var
   th,tc,tm: real;   {angulos iniciales en grados}
   x,y,z:real;       {puntos intermedios de la trayectoria de la
                     recta}
   xi,yi,zi:real;    {coordenadas del punto inicial de la recta}
   xf,yf,zf:real;    {coordenadas del punto final de la recta}
   arreglo: arre;    {array [1..10] of posicion; almacena las
                     posiciones del brazo}
   cont: integer;    {indica el numero de la posicion a grabar}
   op: char;         {espera un caracter del teclado}
   medio,inicio,fin:integer;  {contadores auxiliares para las
                              trayectorias}
   i,j,k,l:integer; {contadores auxiliares}
   p,q,r:integer;
   home:boolean;     {indica si el brazo esta o no en posicion de
                     home}
   dispa:boolean;    {indica si hace el disparo o no}
   arhombro,arcodo,armano:real;    {angulos en radianes del home}
   conthombro,contcodo,contmano:integer;     {contadores
                                             auxiliares de
                                             lasecuencia de
                                             moviviento de cada
                                             articulacion}


procedure menu_principal; forward;
procedure motor(th,tc,tm:real;home,dispa:boolean); forward;
```

```
procedure newton (var t1,t2,t3,px,py,pz:real);
{Este procedimiento tiene el objetivo de proporcionar el angulo
final a que debe rotar cada articulacion para llegar a una posicion
determinada, teniendo como datos los angulos de la posicion en que
se encuentra (en grados) y el punto al que debe llegar}
type
    matriz=array [0..2,0..2] of real;    {matriz auxiliar de 3x3}
    vector=array [0..2] of real;         {vector auxiliar de 3
                                          elementos}
var
    A,B,C,D,E,F,G,H,I:real;    {Derivadas parciales del jacobiano}
    r1,r2,r3:real;             {angulos iniciales en radianes}
    t11,t22,t33: real;         {angulos finales en grados}
    det:real;                  {determinante del jacobiano}
    adj : matriz;              {matriz adjunta del jacobiano}
    f1,f2,f3:real;             {funciones que proporcionan la
                               posicion del punto inicial}
    resta:vectoR;              {vector resultante de la resta de el
                               punto final menos las coordenadas
                               del punto inicial}
    multi:vector;              {vector resultante de la
                               multiplicacion de la matriz inversa
                               del jacobiano por el vector resta}
    ch :char;                  {espera un caracter para continuar}

procedure transformarad (var t1,t2,t3,r1,r2,r3:real);
{Procedimiento que transforma angulos a radianes}
begin
    r1:=t1*3.1416/180;
    r2:=t2*3.1416/180;
    r3:=t3*3.1416/180;
end;

procedure transformang (var t1,t2,t3,t11,t22,t33:real);
{Procedimiento que transforma radianes a angulos}
begin
    t11:=t1*180/3.1416;
    t22:=t2*180/3.1416;
    t33:=t3*180/3.1416;
end;

begin
   transformarad (t1,t2,t3,r1,r2,r3);
   t1:=r1;
   t2:=r2;
   t3:=r3;
```

```
repeat
  f1:=    (l3*cos(t1)*cos(t2)*cos(t3))  +  (l2*cos(t1)*cos(t2))  -
          (l3*cos(t1)*sin(t2)*sin(t3));
  f2:=    (l3*sin(t1)*cos(t2)*cos(t3))  +  (l2*sin(t1)*cos(t2))  -
          (l3*sin(t1)*sin(t2)*sin(t3));
  f3:=    (l3*sin(t2)*cos(t3))    +    (l2*sin(t2))    +
          (l3*cos(t2)*sin(t3)) + l1;
  A:=     (-l3*sin(t1)*cos(t2)*cos(t3)) + (l2*sin(t1)*cos(t2)) +
          (l3*sin(t1)*sin(t2)*sin(t3));
  B:=     (-l3*sin(t2)*cos(t1)*cos(t3)) - (l2*sin(t2)*cos(t1)) -
          (l3*cos(t1)*cos(t2)*sin(t3));
  C:=     (-l3*cos(t1)*cos(t2)*cos(t3)) - (l2*cos(t1)*cos(t2)) -
          (l3*cos(t1)*sin(t2)*cos(t3));
  D:=     (l3*cos(t1)*cos(t2)*cos(t3))  +  (l2*cos(t1)*cos(t2))  -
          (l3*cos(t1)*sin(t2)*sin(t3));
  E:=     (-l3*sin(t1)*sin(t2)*cos(t3)) - (l2*sin(t1)*sin(t2)) -
          (l3*sin(t1)*cos(t2)*sin(t3));
  F:=     (-l3*sin(t1)*cos(t2)*sin(t3)) - (l2*sin(t1)*cos(t2)) -
          (l3*sin(t1)*sin(t2)*cos(t3));
  G:=     0;
  H:=     (l3*cos(t2)*cos(t3))    +    (l2*cos(t2))    -
          (l3*sin(t3)*sin(t2));
  I:=.    (-l3*sin(t2)*sin(t3)) + (l3*cos(t2)*cos(t3));
  det:=   (A*E*I)-(A*F*H)+(B*F*G)-(B*D*I)+(C*D*H)-(C*E*G);
  adj[0,0]:=   (E*I)  -  (F*H);
  adj[0,1]:=   (C*H)  -  (B*I);
  adj[0,2]:=   (B*F)  -  (E*C);
  adj[1,0]:=   (F*G)  -  (D*I);
  adj[1,1]:=   (A*I)  -  (G*C);
  adj[1,2]:=   (C*D)  -  (A*F);
  adj[2,0]:=   (D*H)  -  (E*G);
  adj[2,1]:=   (B*G)  -  (A*H);
  adj[2,2]:=   (E*A)  -  (B*D);
  resta[0]:=  px-f1;
  resta[1]:=  py-f2;
  resta[2]:=  pz-f3;
  multi[0]:=  (1/det)    *    ((adj[0,0]*resta[0])    +
              (adj[0,1]*resta[1]) + (adj[0,2]*resta[2]));
  multi[1]:=  (1/det)    *    ((adj[1,0]*resta[0])    +
              (adj[1,1]*resta[1]) + (adj[1,2]*resta[2]));
  multi[2]:=  (1/det)    *    ((adj[2,0]*resta[0])    +
              (adj[2,1]*resta[1]) + (adj[2,2]*resta[2]));
  t11:=   t1+multi[0];
  t22:=   t2+multi[1];
  t33:=   t3+multi[2];
  t1:=    t11;
```

```
   t2:=   t22;
   t3:=   t33;
   until    ((resta[0]<0.001)    and        (resta[1]<0.001)    and
          (resta[2]<0.001));
   t1:=   ((t1/6.2832)-trunc(t1/6.2832))*6.2832;
   t2:=   ((t2/6.2832)-trunc(t2/6.2832))*6.2832;
   t3:=   ((t3/6.2832)-trunc(t3/6.2832))*6.2832;
   transformang (t1,t2,t3,t11,t22,t33);
   th:=   t11;
   tc:=   t22;
   tm:=   t33;
end;

procedure mueve_brazo;
var
   i:integer;
   cont:integer;
   x1,y1,z1:real;
   tecla: char;

procedure pantal;
var
   i:integer;
begin
cuadrosimple(17,11,31,13,7,0);
textcolor(4);
gotoxy (19,12);
write ('F7');
textcolor(0);
gotoxy (23,12);
write ('DISPARO');
textcolor(4);
gotoxy (53,12);
write ('ESC');
textcolor(0);
gotoxy (56,12);
write (' SALIR');
textcolor(0);
gotoxy(21,17); write('Hombro');
gotoxy(39,17); write('Codo');
gotoxy(56,17); write('Mano');
cuadrosimple(19,14,28,16,7,0);
textcolor (4);
gotoxy(20,15); write(' F1 ');
textcolor (0);
gotoxy(24,15); write('H > ');
```

```
cuadrosimple(36,14,45,16,7,0);
textcolor (4);
gotoxy(37,15); write(' F3 ');
textcolor (0);
gotoxy(41,15); write('C ^ ');
cuadrosimple(53,14,62,16,7,0);
textcolor (4);
gotoxy(54,15); write(' F5 ');
textcolor (0);
gotoxy(58,15); write('M ^ ');
cuadrosimple(19,18,28,20,7,0);
textcolor(4);
gotoxy(20,19); write(' F2  ');
textcolor (0);
gotoxy(24,19); write('H < ');
cuadrosimple(36,18,45,20,7,0);
textcolor (4);
gotoxy(37,19); write(' F4  ');
textcolor (0);
gotoxy(41,19); write('C v ');
cuadrosimple(53,18,62,20,7,0);
textcolor (4);
gotoxy(54,19); write(' F6  ');
textcolor (0);
gotoxy(58,19); write('M v ');
textcolor (0);
gotoxy(23,22);
write ('Localidad   1  2  3  4  5  6  7  8  9');
end;

procedure graba_pos(th,tc,tm,x1,y1,z1:real;tecla:char);
var
   c:char;
begin
case tecla of
          #49:begin
              cont:=1;
             end;
          #50:begin
              cont:=2;
             end;
          #51:begin
              cont:=3;
             end;
          #52:begin
              cont:=4;
             end;
```

```
          #53:begin
              cont:=5;
             end;
          #54:begin
              cont:=6;
             end;
          #55:begin
              cont:=7;
             end;
          #56:begin
              cont:=8;
             end;
          #57:begin
              cont:=9;
             end;
          #27:begin
              cont:=10;
             end;
        end;
x1:= (l3*cos(th)*cos(tc)*cos(tm))    +    (l2*cos(th)*cos(tc))    -
     (l3*cos(th)*sin(tc)*sin(tm));
y1:= (l3*sin(th)*cos(tc)*cos(tm))    +    (l2*sin(th)*cos(tc))    -
     (l3*sin(th)*sin(tc)*sin(tm));
z1:= (l3*sin(tc)*cos(tm)) + (l2*sin(tc)) + (l3*cos(tc)*sin(tm)) +
     l1;
arreglo[cont].x:=   x1;
arreglo[cont].y:=   y1;
arreglo[cont].z:=   z1;
arreglo[cont].th:=  th;
arreglo[cont].tc:=  tc;
arreglo[cont].tm:=  tm;
if (cont<>10 ) then
   begin
   case cont of
        1: begin
           gotoxy (35,22);
           textbackground(4);
           textcolor(128);
           write ('1');
           delay(1300);
           textbackground(7);
           textcolor(0);
           gotoxy (35,22);
           write ('1');
           end;
        2: begin
```

```
        gotoxy (38,22);
        textbackground(4);
        textcolor(128);
        write ('2');
        delay(1300);
        textbackground(7);
        textcolor(0);
        gotoxy (38,22);
        write ('2');
        end;
    3: begin
        gotoxy (41,22);
        textbackground(4);
        textcolor(128);
        write ('3');
        delay(1300);
        textbackground(7);
        textcolor(0);
        gotoxy (41,22);
        write ('3');
        end;
    4: begin
        gotoxy (44,22);
        textbackground(4);
        textcolor(128);
        write ('4');
        delay(1300);
        textbackground(7);
        textcolor(0);
        gotoxy (44,22);
        write ('4');
        end;
    5: begin
        gotoxy (47,22);
        textbackground(4);
        textcolor(128);
        write ('5');
        delay(1300);
        textbackground(7);
        textcolor(0);
        gotoxy (47,22);
        write ('5');
        end;
    6: begin
        gotoxy (50,22);
        textbackground(4);
```

```
            textcolor(128);
            write ('6');
            delay(1300);
            textbackground(7);
            textcolor(0);
            gotoxy (50,22);
            write ('6');
            end;
         7: begin
            gotoxy (53,22);
            textbackground(4);
            textcolor(128);
            write ('7');
            delay(1300);
            textbackground(7);
            textcolor(0);
            gotoxy (53,22);
            write ('7');
            end;
         8: begin
            gotoxy (56,22);
            textbackground(4);
            textcolor(128);
            write ('8');
            delay(1300);
            textbackground(7);
            textcolor(0);
            gotoxy (56,22);
            write ('8');
            end;
         9: begin
            gotoxy (59,22);
            textbackground(4);
            textcolor(128);
            write ('9');
            delay(1300);
            textbackground(7);
            textcolor(0);
            gotoxy (59,22);
            write ('9');
            end;
         end;
   end;
end;
```

```
procedure mover(var th,tc,tm,x1,y1,z1 :real);
var
    seguir :                    boolean;
    tecla  :                    char;
    home   :                    boolean;
    check  :                    real;
    tg1,tg2:real;
begin
pantal;
dispa:=FALSE;
textcolor(1);
gotoxy (16,7); write ('GRADOS');
textcolor(0);
gotoxy(30,7);  write('H ');
gotoxy(45,7);  write('C ');
gotoxy(58,7);  write('M ');
textcolor(5);
gotoxy(32,7);  write(th*180/pi:3:2);
gotoxy(47,7);  write(tc*180/pi:3:2);
gotoxy(60,7);  write(tm*180/pi:3:2);
seguir:=true;
home:= false;
tecla:=#13;
while seguir do
      begin
      x1:=      (l3*cos(th)*cos(tc)*cos(tm)) + (l2*cos(th)*cos(tc))
                - (l3*cos(th)*sin(tc)*sin(tm));
      y1:=      (l3*sin(th)*cos(tc)*cos(tm)) + (l2*sin(th)*cos(tc))
                - (l3*sin(th)*sin(tc)*sin(tm));
      z1:=      (l3*sin(tc)*cos(tm))    +    (l2*sin(tc))    +
                (l3*cos(tc)*sin(tm)) + l1;
      textcolor(1);
      gotoxy (16,9);      write ('COORDENADAS');
      textcolor(0);
      gotoxy(30,9);       write('X ');
      gotoxy(45,9);       write('Y ');
      gotoxy(58,9);       write('Z ');
      textcolor(5);
      gotoxy(32,9);       write(x1:3:2);
      gotoxy(47,9);       write(y1:3:2);
      gotoxy(60,9);       write(z1:3:2);
      tecla:=readkey;
      if tecla=#00  then
         tecla:=readkey;
      case tecla of
           #48: begin
```

```
      graba_pos(th,tc,tm,x1,y1,z1,tecla);
      end;
#49: begin
      graba_pos(th,tc,tm,x1,y1,z1,tecla);
      end;
#50: begin
      graba_pos(th,tc,tm,x1,y1,z1,tecla);
      end;
#51: begin
      graba_pos(th,tc,tm,x1,y1,z1,tecla);
      end;
#52: begin
      graba_pos(th,tc,tm,x1,y1,z1,tecla);
      end;
#53: begin
      graba_pos(th,tc,tm,x1,y1,z1,tecla);
      end;
#54: begin
      graba_pos(th,tc,tm,x1,y1,z1,tecla);
      end;
#55: begin
      graba_pos(th,tc,tm,x1,y1,z1,tecla);
      end;
#56: begin
      graba_pos(th,tc,tm,x1,y1,z1,tecla);
      end;
#57: begin
      graba_pos(th,tc,tm,x1,y1,z1,tecla);
      end;
';': begin                                              {F1}
      check:=th+(0.6*pi/180);
      if (check<(-2*pi)) or (check>(2*pi)) then
         begin
         sound(200);
         delay(500);
         nosound;
         end
      else
          begin
          th:=th+(0.6*pi/180);
          motor(th,tc,tm,FALSE,dispa);
          textcolor(7);
          gotoxy(32,7);write('       ');
          textcolor(5);
          gotoxy(32,7);write(th*180/pi:3:2);
          end;
      end;
```

```
'<': begin                                                    {F2}
     check:=th-(0.6*pi/180);
     if (check<(-2*pi)) or (check>(2*pi)) then
        begin
        sound(200);
        delay(500);
        nosound;
        end
     else
         begin
         th:=th-(0.6*pi/180);
         motor(th,tc,tm,FALSE,dispa);
         textcolor(7);
         gotoxy(32,7);write('        ');
         textcolor(5);
         gotoxy(32,7);write(th*180/pi:3:2);
         end;
     end;
'=': begin                                                    {F3}
     check:=tc+(0.6*pi/180);
     if (check<(-2*pi)) or (check>(2*pi)) then
        begin
        sound(200);
        delay(500);
        nosound;
        end
     else
         begin
         tc:=tc+(0.6*pi/180);
         motor(th,tc,tm,FALSE,dispa);
         textcolor(7);
         gotoxy(47,7);write('        ');
         textcolor(5);
         gotoxy(47,7);write(tc*180/pi:3:2);
         end;
     end;
'>': begin                                                    {F4}
     check:=tc-(0.6*pi/180);
     if (check<(-2*pi)) or (check>(2*pi)) then
        begin
        sound(200);
        delay(500);
        nosound;
        end
     else
         begin
```

```
            tc:=tc-(0.6*pi/180);
            motor(th,tc,tm,FALSE,dispa);
            textcolor(7);
            gotoxy(47,7);write('       ');
            textcolor(5);
            gotoxy(47,7);write(tc*180/pi:3:2);
            end;
        end;
    '?': begin                                          {F5}
        check:=tm+(0.6*pi/180);
        if (check<(-2*pi)) or (check>(2*pi)) then
           begin
           sound(200);
           delay(500);
           nosound;
           end
        else
            begin
            tm:=tm+(0.6*pi/180);
            motor(th,tc,tm,FALSE,dispa);
            textcolor(7);
            gotoxy(60,7);write('       ');
            textcolor(5);
            gotoxy(60,7);write(tm*180/pi:3:2);
            end;
        end;
    '@': begin                                          {F6}
        check:=tm-(0.6*pi/180);
        if (check<(-2*pi)) or (check>(2*pi)) then
           begin
           sound(200);
           delay(500);
           nosound;
           end
        else
            begin
            tm:=tm-(0.6*pi/180);
            motor(th,tc,tm,FALSE,dispa);
            textcolor(7);
            gotoxy(60,7);write('       ');
            textcolor(5);
            gotoxy(60,7);write(tm*180/pi:3:2);
            end;
        end;
    'A': begin                                          {F7}
        dispa:=TRUE;
```

```
                motor(th,tc,tm,FALSE,dispa);
                textcolor(4);
                gotoxy (19,12);
                write ('  F8');
                textcolor(0);
                gotoxy (23,12);
                write ('  ALTO ');
                end;
           'B': begin                                          {F8}
                dispa:=FALSE;
                motor(th,tc,tm,FALSE,dispa);
                textcolor(4);
                gotoxy (19,12);
                write ('F7  ');
                textcolor(0);
                gotoxy (23,12);
                write ('DISPARO');
                end;
           #27: begin
                seguir:=false;
                graba_pos(th,tc,tm,x1,y1,z1,tecla);
                end;
           else
            begin
              write(chr(7));
            end;
         end;
       end;
    end; (* fin de mueve_brazo*)
begin
x1:= 10;
y1:= 10;
z1:= 10;
dispa:=   FALSE;
mover(th,tc,tm,x1,y1,z1);
end;

procedure recta (inicio,fin:integer);
{Procedimiento que calcula una trayectoria recta dados dos puntos}
const
    divi=10;  {numero de divisiones de la recta}
type
    tabl= record
            th:real;
            tc:real;
            tm:real;
            end;
```

```
var
   t:real;       {parametro de la ecuacion vectorial de la recta}
   inc:real;     {incremento de los puntos intermedios de la recta}
   i:integer;    {contador auxiliar}
   op:char;      {espera un caracter para continuar}
   tha,tca,tma:real;      {angulos auxiliares}
   vuelve:boolean;  {variable que se utiliza para repetir la recta
                     en caso de que newton converge de manera
                     inestable}
   seguro:boolean;  {variable que se utiliza para que imprima la
                     tabla al derecho o al reves}
   aux:integer;     {variable auxiliar}
   tabla:array [1..divi] of tabl;  {arreglo que guarda los valores
                                    de los angulos que enviaran al
                                    puerto}
begin
cuadrodoble(25,7,53,9,7,7);
textcolor(4);
gotoxy(29,8);
writeln ('CALCULANDO TRAYECTORIA ');
t:=1/divi;
inc:=t;
vuelve:=  false;
seguro:=  false;
repeat
if vuelve then
   begin
   aux:=  fin;
   fin:=  inicio;
   inicio:=    aux;
   seguro:=    true;
   end;
x:=  arreglo[inicio].x;
y:=  arreglo[inicio].y;
z:=  arreglo[inicio].z;
th:= arreglo[inicio].th*180/3.1416;
tc:= arreglo[inicio].tc*180/3.1416;
tm:= arreglo[inicio].tm*180/3.1416;
for i:=1 to divi do
    begin
    tha:=th;
    tca:=tc;
    tma:=tm;
    if seguro then
       begin
       tabla[divi+1-i].th:=th;
```

```
        tabla[divi+1-i].tc:=tc;
        tabla[divi+1-i].tm:=tm;
        end
    else
        begin
        tabla[i].th:=th;
        tabla[i].tc:=tc;
        tabla[i].tm:=tm;
        end;
    x:= arreglo[inicio].x + (t*(arreglo[fin].x-arreglo[inicio].x));
    y:= arreglo[inicio].y + (t*(arreglo[fin].y-arreglo[inicio].y));
    z:= arreglo[inicio].z + (t*(arreglo[fin].z-arreglo[inicio].z));
    newton (th,tc,tm,x,y,z);
    if (((abs(th)-abs(tha)) > 90) or ((abs(tc)-abs(tca)) > 90) or
        ((abs(tm)-abs(tma)) > 90) )
        then
        begin
        i:=divi;
        vuelve:=true;
        end
    else
        begin
        vuelve:=false;
        t:=t+inc;
        end;
    end;
until vuelve=false;
for i:=1 to divi do
    begin
    th:=tabla[i].th*3.1416/180;
    tc:=tabla[i].tc*3.1416/180;
    tm:=tabla[i].tm*3.1416/180;
    motor(th,tc,tm,FALSE,dispa);
    delay(1500);
    end;
th:= arreglo[fin].th;
tc:= arreglo[fin].tc;
tm:= arreglo[fin].tm;
motor(th,tc,tm,FALSE,dispa);
end;

procedure circular(inicio,medio,fin:integer);
{Programa que calcula una trayectoria curva teniendo como datos
tres puntos, se utiliza interpolacion de Lagrange para generar
un polinomio de segundo grado}
```

```
const
     n1=10;
     n2=10;     {numero de divisiones de cada trayectoria}
type
    tabl1=record
            y:real;
            z:real;
            end;
    tabl2=record
            th:real;
            tc:real;
            tm:real;
            end;
var
   y,z:real;    {incrementos en las coordenadas de la trayectoria}
   yi,zi:real;  {coordenadas del punto inicial}
   ym,zm:real;  {coordenadas del punto intermiedio}
   yf,zf:real;  {coordenadas del punto final}
   op:char;     {espera un caracter para continuar}
   tha,tca,tma:real;      {angulos auxiliares}
   aux:integer;      {variable auxiliar}
   aux2:integer;     {variable auxiliar}
   n3:integer;  {variable auxiliar}
   d1:real;     {distancia lineal del punto inicial al intermedio}
   d2:real;     {distancia lineal del punto intermedio al final}
   i:integer;   {contador auxiliar}
   inc1,inc2:real;  {incremento de cada trazectoria dependiendo
                     del numero de divisiones en funcion de la
                     longitud}
   tabla:array [1..21] of tabl1;    {arreglo que guarda los valores
                                     de las coordenadas que se
                                     enviaran al newton}
   tabla2:array [1..21] of tabl2;   {arreglo que guarda los valores
                                     delos angulos que enviaran al
                                     puerto}
   vuelve:boolean;  {variable que se utiliza para repetir la curva
                     caso de que newton converge de manera
                     inestable}
   seguro:boolean;  {variable que se utiliza para que imprima la
                     tabla al derecho o al reves}

procedure divide(var yi,zi,ym,zm,yf,zf:real);
{Este procedimiento proporciona el numero de divisiones de las
trayectorias en funcion de su longitud}

begin
```

```
d1:= sqrt( sqr(ym-yi) + sqr(zm-zi) );
d2:= sqrt( sqr(yf-ym) + sqr(zf-zm) );
{if d1>20 then n1:=10
else n1:=10;
if d2>20 then n2:=10
else n2:=10;}
end;
begin
cuadrodoble(25,7,53,9,7,7);
textcolor(4);
gotoxy(29,8);
writeln ('CALCULANDO TRAYECTORIA');
vuelve:=false;
seguro:=false;
yi:=arreglo[inicio].y;
zi:=arreglo[inicio].z;
ym:=arreglo[medio].y;
zm:=arreglo[medio].z;
yf:=arreglo[fin].y;
zf:=arreglo[fin].z;
th:=arreglo[inicio].th;
tc:=arreglo[inicio].tc;
tm:=arreglo[inicio].tm;
divide (yi,zi,ym,zm,yf,zf);
inc1:=(ym-yi)/n1;
if inc1<>0 then
   begin
   y:=yi;
   for i:=1 to n1+1 do
       begin
       z:=      ( ((y-ym)*(y-yf))/((yi-ym)*(yi-yf)) ) * zi  +
                ( ((y-yi)*(y-yf))/((ym-yi)*(ym-yf)) ) * zm  +
                ( ((y-yi)*(y-ym))/((yf-yi)*(yf-ym)) ) * zf  ;
{       writeln ('y ',y:2:2,' ','z ',z:2:2);}
       tabla[i].y:=y;
       tabla[i].z:=z;
       x:=5;
       y:=y+inc1;
       end;
   end
else
   begin
   inc1:=(zm-zi)/n1;
   z:=zi;
   for i:=1 to n1+1 do
       begin
```

```
        tabla[i].y:=y;
        tabla[i].z:=z;
        z:=z+inc1;
        end;
    end;
inc2:=(yf-ym)/n2;
if inc2<>0 then
    begin
    y:=ym;
    for i:=1 to n2+1 do
        begin
        z:=     ( ((y-ym)*(y-yf))/((yi-ym)*(yi-yf)) ) * zi  +
                ( ((y-yi)*(y-yf))/((ym-yi)*(ym-yf)) ) * zm  +
                ( ((y-yi)*(y-ym))/((yf-yi)*(yf-ym)) ) * zf  ;

        tabla[n1+i].y:=y;
        tabla[n1+i].z:=z;
        y:=y+inc2;
        end;
    end
else
    begin
    inc2:=(zf-zm)/n2;
    z:=zm;
    for i:=1 to n2+1 do
        begin
        tabla[n1+i].y:=y;
        tabla[n1+i].z:=z;
        z:=z+inc2;
        end;
    end;
repeat
if vuelve then
    begin
    aux2:=11;
    aux:=medio;
    medio:=inicio;
    inicio:=aux;
    seguro:=true;
    end
 else
    aux2:=1;
th:= arreglo[inicio].th*180/3.1416;
tc:= arreglo[inicio].tc*180/3.1416;
tm:= arreglo[inicio].tm*180/3.1416;
for i:=1 to n1+1 do
```

```
    begin
    tha:=th;
    tca:=tc;
    tma:=tm;
    newton (th,tc,tm,x,tabla[aux2].y,tabla[aux2].z);
    if ( ((abs(th)-abs(tha)) > 90) or ((abs(tc)-abs(tca)) > 90) or
         ((abs(tm)-abs(tma)) > 90) )
        then
        begin
        i:=n1;
        vuelve:=true;
        end
    else
        begin
        vuelve:=false;
        end;
    if seguro then
       begin
       tabla2[n1+1-i].th:=th;
       tabla2[n1+1-i].tc:=tc;
       tabla2[n1+1-i].tm:=tm;
       aux2:=aux2-1;
       end
    else
        begin
        tabla2[i].th:=th;
        tabla2[i].tc:=tc;
        tabla2[i].tm:=tm;
        aux2:=aux2+1;
        end;
    end;
until vuelve=false;
vuelve:=false;
seguro:=false;
repeat
if vuelve then
   begin
   aux2:=21;
   aux:=fin;
   fin:=medio;
   medio:=aux;
   seguro:=true;
   end
 else
   aux2:=11;
th:= arreglo[medio].th*180/3.1416;
```

```
tc:= arreglo[medio].tc*180/3.1416;
tm:= arreglo[medio].tm*180/3.1416;
n3:= n1;
for i:=1 to n2+1 do
    begin
    tha:=th;
    tca:=tc;
    tma:=tm;
    newton (th,tc,tm,x,tabla[aux2].y,tabla[aux2].z);
    if ( ((abs(th)-abs(tha)) > 90) or ((abs(tc)-abs(tca)) > 90) or
         ((abs(tm)-abs(tma)) > 90) )
        then
        begin
        i:=n1;
        vuelve:=true;
        end
    else
        begin
        vuelve:=false;
        end;
    if seguro then
       begin
       tabla2[n3+1-i].th:=     th;
       tabla2[n3+1-i].tc:=     tc;
       tabla2[n3+1-i].tm:=     tm;
       aux2:=aux2-1;
       end
    else
        begin
        tabla2[n3+i].th:=      th;
        tabla2[n3+i].tc:=      tc;
        tabla2[n3+i].tm:=      tm;
        aux2:=aux2+1;
        end;
    end;
until vuelve=false;
for i:=1 to n1+1 do
    begin
    th:=  tabla2[i].th*3.1416/180;
    tc:=  tabla2[i].tc*3.1416/180;
    tm:=  tabla2[i].tm*3.1416/180;
    motor(th,tc,tm,FALSE,dispa);
    delay(1500);
    end;
for i:=n1 to n1+n2+1 do
    begin
```

```
    th:=  tabla2[i].th*3.1416/180;
    tc:=  tabla2[i].tc*3.1416/180;
    tm:=  tabla2[i].tm*3.1416/180;
    motor(th,tc,tm,FALSE,dispa);
    delay(1500);
    end;
end;

procedure motor(th,tc,tm:real;home,dispa:boolean);
{Este procedimiento convierte los grados a moverse (en radianes) de
cada articulacion en pulsos electricos que se transmiten por el
puerto paralelo a motores de pasos}

var
   salida: array [1..4] of integer;     {vector que almacena la
                                         secuencia de movimiento}
   hombro,codo,mano:real;               {angulos en grados de
                                         cada articulacion}


   vechombro,veccodo,vecmano:real; {almacena el numero de pulsos
                                    que debe moverse cada
                                    articulacion}
   salhombro,salcodo,salmano:integer;   {trunca el valor de
                                         vechombro, veccodo y
                                         vecmano para que el
                                         numero de pulsos sea
                                         entero}

   tecla:char; {espera una tecla para continuar}
   i,j,k:integer;    {contadores auxiliares}
   h,c,m:integer;    {valor absoluto de los pulsos a moverse de
                     cada articulacion,para saber que motor se
                     mueve mas rapido y cual mas lento}
bandera:integer;     {proporciona que
                     articulacion debe moverse antes que las demas}
   mayor,medio,menor:integer; {nos indican que articulacion es la
                               que debe moverse mas rapido o lento}
   disp: integer;    {manda el pulso a puerto para el disparo}

procedure procmano;
{Procedimeinto para dar la secuencia de giro al motor de pasos de
la mano}
begin
if (salmano>0) then
   begin
```

```
   dec (contmano);
   if (contmano<1) then contmano:=4;
   if (contmano>4) then contmano:=1;
   if dispa then disp:=64 else disp:=0;
   port[$3BC]:=16+disp;
   delay(100);
   port[$3BC]:=128+16+disp;
   delay(100);
   end
else
    begin
    inc (contmano);
    if (contmano<1) then contmano:=4;
    if (contmano>4) then contmano:=1;
    if dispa then disp:=64 else disp:=0;
    port[$3BC]:=32+disp;
    delay(100);
    port[$3BC]:=128+32+disp;
    delay(100);
    end;
end;

procedure prochombro;
{Procedimeinto para dar la secuencia de giro al motor de pasos del
hombro}
begin
if (salhombro>0) then
   begin
   dec (conthombro);
   if (conthombro<1) then conthombro:=4;
   if (conthombro>4) then conthombro:=1;
   if dispa then disp:=64 else disp:=0;
   port[$3BC]:=1+disp;
   delay(100);
   port[$3BC]:=128+1+disp;
   delay(100);
   end
else
    begin
    inc (conthombro);
    if (conthombro<1) then conthombro:=4;
    if (conthombro>4) then conthombro:=1;
    if dispa then disp:=64 else disp:=0;
    port[$3BC]:=2+disp;
    delay(100);
    port[$3BC]:=128+2+disp;
```

```
    delay(100);
    end;
end;

procedure proccodo;
{Procedimeinto para dar la secuencia de giro al motor de pasos del
codo}
begin
if (salcodo>0) then
   begin
   dec (contcodo);
   if (contcodo<1) then contcodo:=4;
   if (contcodo>4) then contcodo:=1;
   if dispa then disp:=64 else disp:=0;
   port[$3BC]:=4+disp;
   delay(100);
   port[$3BC]:=128+4+disp;
   delay(100);
   end
else
    begin
    inc (contcodo);
    if (contcodo<1) then contcodo:=4;
    if (contcodo>4) then contcodo:=1;
    if dispa then disp:=64 else disp:=0;
    port[$3BC]:=8+disp;
    delay(100);
    port[$3BC]:=128+8+disp;
    delay(100);
    end;
end;
begin
textcolor(0);
salida[1]:=$A;
salida[2]:=$9;
salida[3]:=$5;
salida[4]:=$6;
hombro:=th*180/3.1416;
codo:=tc*180/3.1416;
mano:=tm*180/3.1416;
if (home=TRUE) then
   begin
   cuadrosimple(25,7,53,9,7,7);
   textcolor(4);
   gotoxy(27,8);
   write('DETECTANDO POSICION INICIAL');delay(1000);
```

```
   arhombro:=0;
   arcodo:=0;
   armano:=-90;
   conthombro:=1;
   contcodo:=1;
   contmano:=1;
   port[$3BC]:=63;
   delay(100);
   port[$3BC]:=128+63;
   delay(100);
   gotoxy(60,2);
   write(salida[conthombro]:2,'    ',salida[contcodo]:2,'
         ',salida[contmano]:2);
   cuadrosimple(25,7,57,9,7,7);
   end
else
    begin
    vechombro:=round ((hombro- arhombro)/0.6);
    veccodo:=round ((codo- arcodo)/0.6);
    vecmano:=round ((mano- armano)/0.6);
    salhombro:=trunc (vechombro);
    salcodo:=trunc (veccodo);
    salmano:=trunc (vecmano);
    h:=abs(salhombro);
    c:=abs(salcodo);
    m:=abs(salmano);
    if ((vechombro=0) and (veccodo=0) ) then
       begin
       if (vecmano<>0) then
          begin
          procmano;
          gotoxy(60,2);
          write(salida[conthombro]:2,'    ',salida[contcodo]:2,'
          ',salida[contmano]:2);
          delay(50);
          end;
       end;
    if ((vechombro=0) and (vecmano=0) ) then
       begin
       if (veccodo<>0) then
          begin
          proccodo;
          gotoxy(60,2);
          write(salida[conthombro]:2,' ',salida[contcodo]:2,'
```

```
        ',salida[contmano]:2);
        delay(50);
        end;
    end;
if ( (veccodo=0) and (vecmano=0)) then
    begin
    if (vechombro<>0) then
        begin
        prochombro;
        gotoxy(60,2);
        write(salida[conthombro]:2,'     ',salida[contcodo]:2,'
        ',salida[contmano]:2);
        delay(50);
        end;
    end
else
begin
if ((h>c) and (h>m)) then
    begin
    if (c>m) then
        begin
        mayor:=h;
        medio:=c;
        menor:=m;
        bandera:=1;
        end;
    if (m>c) then
         begin
         mayor:=h;
         medio:=m;
         menor:=c;
         bandera:=2;
         end;
    end;
if ((c>h) and (c>m)) then
    begin
    if (h>m) then
        begin
        mayor:=c;
        medio:=h;
        menor:=m;
        bandera:=3;
        end;
    if (m>h) then
         begin
```

```
        mayor:=c;
        medio:=m;
        menor:=h;
        bandera:=4;
        end;
   end;
if ((m>h) and (m>c)) then
   begin
   if (h>c) then
      begin
      mayor:=m;
      medio:=h;
      menor:=c;
      bandera:=5;
      end;
   if (c>h) then
       begin
       mayor:=m;
       medio:=c;
       menor:=h;
       bandera:=6;
       end;
   end;
   if ((h=c) and (c=m)) then
      begin
      mayor:=h;
      medio:=c;
      menor:=m;
      bandera:=7;
      end;
   if ((h=c) and (c<>m)) then
      begin
      if (c>m) then
         begin
         mayor:=h;
         medio:=c;
         menor:=m;
         bandera:=1;
         end;
      if (m>c) then
         begin
         mayor:=m;
         medio:=c;
         menor:=h;
         bandera:=6;
         end;
```

```
            end;
         if ((h=m) and (c<>m)) then
            begin
            if (c>m) then
               begin
               mayor:=c;
               medio:=h;
               menor:=m;
               bandera:=3;
               end;
            if (m>c) then
               begin
               mayor:=m;
               medio:=h;
               menor:=c;
               bandera:=5;
               end;
            end;
         if ((c=m) and (c<>h)) then
            begin
            if (h>c) then
               begin
               mayor:=h;
               medio:=c;
               menor:=m;
               bandera:=1;
               end;
            if (c>h) then
               begin
               mayor:=c;
               medio:=m;
               menor:=h;
               bandera:=4;
               end;
            end;
if menor=0 then
   begin
   for j:=1 to medio do
      begin
      if bandera=1 then proccodo;
      if bandera=2 then procmano;
      if bandera=3 then prochombro;
      if bandera=4 then procmano;
      if bandera=5 then prochombro;
      if bandera=6 then proccodo;
      if bandera=7 then proccodo;
```

```
        for i:=1 to trunc(mayor/medio) do
            begin
            if bandera=1 then prochombro;
            if bandera=2 then prochombro;
            if bandera=3 then proccodo;
            if bandera=4 then proccodo;
            if bandera=5 then procmano;
            if bandera=6 then procmano;
            if bandera=7 then prochombro;
            gotoxy(60,2);
            write(salida[conthombro]:2,'     ',salida[contcodo]:2,'
           ',salida[contmano]:2);
            delay (50);
            end;
         end;
   end
else
begin
for k:=1 to menor do
    begin
    if bandera=1 then procmano;
    if bandera=2 then proccodo;
    if bandera=3 then procmano;
    if bandera=4 then prochombro;
    if bandera=5 then proccodo;
    if bandera=6 then prochombro;
    if bandera=7 then procmano;
    for j:=1 to trunc(medio/menor) do
        begin
        if bandera=1 then proccodo;
        if bandera=2 then procmano;
        if bandera=3 then prochombro;
        if bandera=4 then procmano;
        if bandera=5 then prochombro;
        if bandera=6 then proccodo;
        if bandera=7 then proccodo;
        for i:=1 to trunc(mayor/medio) do
            begin
            if bandera=1 then prochombro;
            if bandera=2 then prochombro;
            if bandera=3 then proccodo;
            if bandera=4 then proccodo;
            if bandera=5 then procmano;
            if bandera=6 then procmano;
            if bandera=7 then prochombro;
```

```
            end;
        end;
    end;
    end;
    end;
end;
if dispa then
   begin
   port[$3BC]:=64;
   delay(100);
   port[$3BC]:=128+64;
   delay(100);
   end
else
    begin
    port[$3BC]:=0;
    delay(100);
    port[$3BC]:=128;
    delay(100);
    end;
arhombro:=th*180/3.1416;
arcodo:=tc*180/3.1416;
armano:=tm*180/3.1416;
end;

procedure edit(var arreglo: arre);
{Programa que realiza la funcion de un editor para almacenar un
secuencia
 de instrucciones que muevan el brazo}
const
     minimo=1; {minimo valor que puede tomar la velocidad, el
               movimiento lineal y circular}
type
    arreglo1= record
              num_linea: integer;  {numero de linea de cada
                                    instruccion}
              instruc: string[10]; {nombre de la instruccion}
              par_1: integer; {primer parametro de la instruccion}
              par_2: integer; {segundo parametro de la
                               instruccion}
              end;
var
   p,q,r: integer;   {contadores auxiliares}
   valor: integer;   {valor que puede tomar cada parametro}
   seguir: boolean; {indica si se desea salir o no del editor}
   lee_tecla: boolean;    {indica si ya leyo una tecla}
```

```
   tecla: char;      {lee el codigo de una tecla}
   linea: array [1..21] of arreglo1;     {almacena cada registro
                                          formado por numero de
                                          linea, nombre de la
                                          instruccion y parametros}
   maximo:integer;   {valor maximo que puede tomar cada
                                    parametro}
   archivo: text;    {archivo para almacenar la secuencia de
                     instrucciones del editor}
   op:char;      {espera un caracter para continuar}
   opcion:char;      {almacena la opcion del menu}
   nombre_arch:string[12];      {guarda el nombre del archivo}
   nuevo_arch:string[12];       {guarda el nombre nuevo del archivo}
   fileExist:boolean;      {regresa si existe o no un archivo}
   dirinfo: SearchRec;     {informacion sobre nombre de archivos
                           utilizando la unidad DOS}

procedure inicializa;
{Procedimiento que inicializa el valor de cada campo del registro}
var
   p:integer;
begin
     for p:=1 to 21 do
         begin
         linea[p-1].num_linea:=0;
         linea[p-1].instruc:='
         linea[p-1].par_1:=0;
         linea[p-1].par_2:=0;
         end;
end;

function flecha(cont:integer;identifica:char;p:integer):integer;
{Funcion que proporciona un numero que indica valor del parametro
en funcion de las teclas de flechas horizontales}
var
   entra:boolean;    {se utiliza para forzar la introduccion de un
                     parametro antes de salir de la funcion}
begin
valor:=minimo;
entra:=true;
case identifica of
     'l': maximo:=10;
     'c': maximo:=20;
     'v': maximo:=3;
     end;
repeat
```

```
      tecla:=readkey;
      case tecla of
           #75: begin                              {flecha izquierda}
                valor:=valor-1;
                if valor=minimo-1 then
                   valor:=maximo;
                gotoxy(cont,p);
                write (valor:2);
                entra:=false;
                end;
           #77: begin                              {flecha derecha}
                valor:=valor+1;
                if valor=maximo+1 then
                   valor:=minimo;
                gotoxy(cont,p);
                write (valor:2);
                entra:=false;
                end;
           end;
until ((tecla=#13) and (entra=false));
flecha:=valor;
end;

procedure ayudas;
{Procedimiento que muestra las ayudas al usuario}
begin
     cuadrosimple(2,22,79,25,1,15);
     gotoxy (5,23); write ('F1');
     gotoxy (12,23); write ('F2');
     gotoxy (21,23); write ('F3');
     gotoxy (31,23); write ('F4');
     gotoxy (39,23); write ('F5');
     gotoxy (45,23); write ('F6');
     gotoxy (53,23); write ('INS');
     gotoxy (64,23); write ('DEL');
     gotoxy (74,23); write ('F7');
     gotoxy (4,24); write ('MOVEL');
     gotoxy (11,24); write ('MOVEC');
     gotoxy (18,24); write ('VELOCIDAD');
     gotoxy (29,24); write ('DISPARO');
     gotoxy (38,24); write ('ALTO');
     gotoxy (44,24); write ('MENU');
     gotoxy (50,24); write ('INS LINEA');
     gotoxy (61,24); write ('DEL LINEA');
     gotoxy (72,24); write ('EJECUTA');
end;
```

```
procedure editor(valida:boolean);
var
   aux:char;    {variable que guarda los caracteres de la
                instruccion}
   l,p,r:integer;    {contador auxiliar}

begin
     textbackground(1);
     clrscr;
     textcolor(15);
     p:=1;
     q:=0;
     r:=0;
     tecla:=#13;
     seguir:=true;
     lee_tecla:=false;
     ayudas;
     if valida=true then
     begin
     {$I-}
     assign(archivo,nombre_arch);
     reset(archivo);
     p:=1;
     while not eof (archivo) do
           begin
           readln(archivo,linea[p].num_linea,linea[p].instruc,
           linea[p].par_1,linea[p].par_2);
           gotoxy (5,p);
           write(linea[p].num_linea:2);
           gotoxy (8,p);
           l:=1;
           for r:=1 to 10 do
               begin
               aux:=linea[p].instruc[r];
               if aux<>' ' then
                  begin
                  write(aux);
                  linea[p].instruc[l]:=aux;
                  l:=l+1;
                  end;
               end;
           for r:=l to 10 do
               linea[p].instruc[r]:=' ';
           if linea[p].par_1<>0 then
              begin
              gotoxy(18,p);
```

```
            write(linea[p].par_1:2);
            end;
         if linea[p].par_2<>0 then
            begin
            gotoxy(22,p);
            write(linea[p].par_2:2);
            end;
         q:=q+1;
         p:=p+1;
         end;
         close (archivo);
         {$I+}
end
else
     begin
     inicializa;
     p:=1;
     end;
while ((seguir) and (q<=20))  do
   begin
   gotoxy(60,1);
   write(' p= ',p:2,' q= ',q:2,' r= ',r:2);
   gotoxy( 5,p);
   if lee_tecla=false then
      begin
      tecla:=readkey;
      if tecla=#00  then
         tecla:=readkey;
      end;
    case tecla of
     ';': begin                                    { F1 }
          if (p<>21) then
          begin
          linea[p].num_linea:=p;
          linea[p].instruc:='MOVEL';
          write (p:2,' MOVEL ');
          linea[p].par_1:=flecha(18,'l',p);
          linea[p].par_2:=0;
          p:=p+1;
          if lee_tecla=false then
             q:=p-1;
          lee_tecla:=false;
          end;
          end;
     '<': begin                                    { F2 }
          if (p<>21) then
```

```
        begin
        linea[p].num_linea:=p;
        linea[p].instruc:='MOVEC';
        write (p:2,' MOVEC ');
        linea[p].par_1:=flecha(18,'c',p);
        gotoxy(22,p);
        linea[p].par_2:=flecha(22,'c',p);
        p:=p+1;
        if lee_tecla=false then
           q:=p-1;
        lee_tecla:=false;
        end;
        end;
'=': begin                                        { F3 }
        if (p<>21) then
        begin
        linea[p].num_linea:=p;
        linea[p].instruc:='VELOCIDAD';
        write (p:2,' VELOCIDAD ');
        linea[p].par_1:=flecha(18,'v',p);
        linea[p].par_2:=0;
        p:=p+1;
        if lee_tecla=false then
           q:=p-1;
        lee_tecla:=false;
        end;
        end;
'>': begin                                        { F4 }
        if (p<>21) then
        begin
        linea[p].num_linea:=p;
        linea[p].instruc:='DISPARO';
        write (p:2,' DISPARO ');
        linea[p].par_1:=0;
        linea[p].par_2:=0;
        p:=p+1;
        if lee_tecla=false then
           q:=p-1;
        lee_tecla:=false;
        end;
        end;
'?': begin                                        { F5 }
        if (p<>21) then
        begin
        linea[p].num_linea:=p;
        linea[p].instruc:='ALTO';
```

```
                write (p:2,' ALTO ');
                linea[p].par_1:=0;
                linea[p].par_2:=0;
                p:=p+1;
                if lee_tecla=false then
                   q:=p-1;
                lee_tecla:=false;
                end;
                end;
           '@': begin                                    { F6 }
                {GRABAR DATOS}
                assign(archivo,nombre_arch);
                rewrite(archivo);
                for p:=1 to q do
                    begin
                    linea[p].instruc:=' '+linea[p].instruc;
                     writeln(archivo,linea[p].num_linea:2,linea[p]
                     .instruc:10,
                     linea[p].par_1:3,linea[p].par_2:3);
                    end;
                close (archivo);
                seguir:=false;
                end;
           'A': begin                                    { F7 }
                for i:=1 to q do
                    begin
                    case linea[i].instruc[1] of
                         'M': begin
                              if linea[i].instruc[5]='L' then
                                 begin
                                 recta(10,linea[i].par_1);

arreglo[10].x:=     arreglo[linea[i].par_1].x;

arreglo[10].y:=     arreglo[linea[i].par_1].y;

arreglo[10].z:=     arreglo[linea[i].par_1].z;

arreglo[10].th:=    arreglo[linea[i].par_1].th;

arreglo[10].tc:=    arreglo[linea[i].par_1].tc;

arreglo[10].tm:=    arreglo[linea[i].par_1].tm;
                                 end
                              else
                                  begin
```

```
circular(10,linea[i].par_1,linea[i].par_2);

arreglo[10].x:=arreglo[linea[i].par_2].x;

arreglo[10].y:=arreglo[linea[i].par_2].y;

arreglo[10].z:=arreglo[linea[i].par_2].z;

arreglo[10].th:=arreglo[linea[i].par_2].th;

arreglo[10].tc:=arreglo[linea[i].par_2].tc;

arreglo[10].tm:=arreglo[linea[i].par_2].tm;
                                    end;
                              end;
                         'V': begin
                              writeln ('velocidad');
                              end;
                         'D': begin
                              writeln ('disparo');
                              end;
                         'A': begin
                              writeln ('alto');
                              end;
                         end;
                    end;
               end;
          'S': begin                                    [Suprimir]
               if q>=p then
               begin
               while linea[p].num_linea<>q do
                     begin
                     linea[p].num_linea:=p;
                     linea[p].instruc:=linea[p+1].instruc;
                     linea[p].par_1:=linea[p+1].par_1;
                     linea[p].par_2:=linea[p+1].par_2;
                     gotoxy (5,p);
                     write (p:2,' ',linea[p].instruc,'
                     if linea[p].par_1<>0 then
                        begin
                        gotoxy (18,p);
                        write(linea[p].par_1:2);
                        end
                     else
                         begin
```

```
                    gotoxy (18,p);
                    write('    ');
                    end;
               if linea[p].par_2<>0 then
                  begin
                  gotoxy (22,p);
                  write(linea[p].par_2:2);
                  end
               else
                    begin
                    gotoxy (22,p);
                    write('    ');
                    end;
               p:=p+1;
               end;
          gotoxy (5,q);
          linea[q].num_linea:=0;
          linea[q].instruc:='              ';
          linea[q].par_1:=0;
          linea[q].par_2:=0;
          write ('

          gotoxy (5,q);
          p:=q;
          if lee_tecla=false then
             q:=q-1;
          lee_tecla:=false;
          end;
          end;
     'R': begin                                    {Insertar}
          r:=q;
          if (p<>21) then
          begin
          while (r>=p) do
               begin
               linea[r+1].num_linea:=r+1;
               linea[r+1].instruc:=linea[r].instruc;
               linea[r+1].par_1:=linea[r].par_1;
               linea[r+1].par_2:=linea[r].par_2;
               gotoxy (5,r+1);
               write (r+1:2,' ',linea[r+1].instruc,'     ');
               if linea[r+1].par_1<>0 then
                  begin
                  gotoxy (18,r+1);
                  write(linea[r+1].par_1:2);
                  end
```

```
                else
                    begin
                    gotoxy (18,r+1);
                    write('   ');
                    end;
                if linea[r+1].par_2<>0 then
                   begin
                   gotoxy (22,r+1);
                   write(linea[r+1].par_2:2);
                   end
                else
                    begin
                    gotoxy (22,r+1);
                    write('   ');
                    end;
                r:=r-1;
                end;
          gotoxy (5,p);
          linea[p].num_linea:=0;
          linea[p].instruc:='
          linea[p].par_1:=0;
          linea[p].par_2:=0;
          write ('
');
          gotoxy (5,p);
          repeat
          tecla:=readkey;
          if tecla=#00  then
             begin
             tecla:=readkey;
             gotoxy (5,p);
             end;
          until ((tecla=';') or (tecla='<') or
                 (tecla='=') or (tecla='>') or
                 (tecla='?'));
          lee_tecla:=true;
          q:=q+1;
          end;
          end;
     #72: begin                                    {flecha arriba}
          if p<=1 then
             p:=1
          else
              p:=p-1;
          gotoxy (5,p);
          end;
```

```
          #80: begin                                    {flechas abajo}
               if p>21 then
                  p:=20
               else
                   begin
                   if p>=(q+1) then
                      p:=q+1
                   else
                      p:=p+1;
                   end;
               gotoxy (5,p);
               end;
          #13: writeln;
          #27: seguir:=false;
          end;
       end;
textbackground(0);
clrscr;
cuadrodoble(1,1,80,24,1,15);
end;

procedure menu;
begin
textbackground(1);
cuadrodoble(7,3,70,22,1,1);
cuadrodoble(27,3,51,5,0,15);
gotoxy (32,4);
write ('MENU DEL EDITOR');
cuadrodoble(26,7,52,22,7,0);
textcolor(0);
gotoxy(30,9);
write('1) Crear programa ');
gotoxy(30,11);
write('2) Cargar programa ');
gotoxy(30,13);
write('3) Listar programas ');
gotoxy(30,15);
write('4) Borrar programa ');
gotoxy(30,17);
write('5) Renombrar programa ');
gotoxy(30,19);
write('6) Terminar ');
repeat
      gotoxy(38,21);
      write('Opcion ');
      opcion:=readkey;
```

```
until ((opcion='6') or (opcion='5') or (opcion='4') or (opcion='3')
      or (opcion='2') or (opcion='1'));
end;

begin {programa principal editor}
repeat
menu;
case opcion of
     '1': begin
          cuadrodoble(11,13,68,15,5,0);
          gotoxy (13,14);
          textcolor(0);
          write ('Nombre del archivo (maximo de 8 caracteres) ');
          readln (nombre_arch);
          nombre_arch:=nombre_arch+'.dat';
          {$I-}
          assign(archivo,nombre_arch);
          reset(archivo);
          fileExist:=( (IOresult=0) and (nombre_arch<>'') );
          if fileExist then
             begin
             close(archivo);
             {$I+}
             cuadrodoble(12,14,67,14,5,5);
             cuadrodoble(11,13,68,15,5,0);
             textcolor(0);
             gotoxy (13,14);
             write ('ARCHIVO EXISTENTE, QUIERES ESCRIBIR SOBRE EL?
S/N ');
             read (op);
             if ((op='s') or (op='S')) then
                begin
                editor(false);
                end;
             end
          else
              editor(false);
          end;
     '2': begin
          cuadrodoble(11,13,68,15,5,0);
          gotoxy (13,14);
          textcolor(0);
          write ('Nombre del archivo (maximo de 8 caracteres) ');
          readln (nombre_arch);
          nombre_arch:=nombre_arch+'.dat';
          {$I-}
```

```
      assign(archivo,nombre_arch);
      reset(archivo);
      fileExist:=( (IOresult=0) and (nombre_arch<>'') );
      if fileExist then
         begin
         p:=1;
         close(archivo);
         {$I+}
         editor(true);
         end
      else
          begin
          cuadrodoble(12,14,67,14,5,5);
          cuadrodoble(11,13,68,15,5,0);
          textcolor(0);
          gotoxy (23,14);
          write ('!ERROR!, NO EXISTE ESE ARCHIVO');
          delay(1500);
          end;
      end;
'3': begin
      cuadrodoble(3,3,78,23,1,1);
      textcolor(15);
      p:=5;
      FindFirst('*.DAT',Anyfile,DirInfo);
      while DosError=0 do
            begin
            gotoxy (35,p);
            write (DirInfo.name);
            findnext(DirInfo);
            p:=p+1;
            end;
      cuadrodoble(23,21,56,23,5,0);
      textcolor(0);
      gotoxy (24,22);
      write ('Oprime una tecla para continuar ');
      read (op);
      cuadrodoble(23,21,56,23,1,1);
      end;
'4': begin
      cuadrodoble(11,13,68,15,5,0);
      gotoxy (18,14);
      textcolor(0);
      write ('Nombre del archivo a borrar ');
      readln (nombre_arch);
      nombre_arch:=nombre_arch+'.dat';
```

```
        {$I-}
        assign(archivo,nombre_arch);
        reset(archivo);
        fileExist:=((IOresult=0) and (nombre_arch<>''));
        if fileExist then
           begin
           close(archivo);
           {$I+}
           cuadrodoble(12,14,67,14,5,5);
           cuadrodoble(11,13,68,15,5,0);
           textcolor(0);
           gotoxy (13,14);
           write ('ESTAS SEGURO QUE DESEAS BORRAR A
           ',nombre_arch:12,' S/N ? ');
           read (op);
           if ((op='s') or (op='S')) then
              erase(archivo);
           end
        else
            begin
            cuadrodoble(12,14,67,14,5,5);
            cuadrodoble(11,13,68,15,5,0);
            textcolor(0);
            gotoxy (23,14);
            write ('¡ERROR! NO EXISTE ESE ARCHIVO ');
            delay(1500);
            end;
        end;
   '5': begin
        cuadrodoble(11,13,68,15,5,0);
        textcolor(0);
        gotoxy (18,14);
        write ('Nombre del archivo ');
        readln (nombre_arch);
        nombre_arch:=nombre_arch+'.dat';
        {$I-}
        assign(archivo,nombre_arch);
        reset(archivo);
        fileExist:=((IOresult=0) and (nombre_arch<>''));
        if fileExist then
           begin
           cuadrodoble(12,14,67,14,5,5);
           cuadrodoble(11,13,68,15,5,0);
           textcolor(0);
           gotoxy (18,14);
           write ('Nombre del nuevo archivo ');
```

```
            read (nuevo_arch);
            nuevo_arch:=nuevo_arch+'.dat';
            rename(archivo,nuevo_arch);
            close(archivo);
            {$I+}
            end
        else
            begin
            cuadrodoble(12,14,67,14,5,5);
            cuadrodoble(11,13,68,15,5,0);
            textcolor(0);
            gotoxy (23,14);
            write ('¡ERROR! NO EXISTE ESE ARCHIVO ');
            delay(1500);
            end;
        end;
    end;
until (opcion='6');
menu_principal;
end;

procedure menu_automatico;
{Este procedimiento nos da el menu del modo automatico y hace
llamadas
a los procedimientos recta y circular}
begin
cuadrodoble(14,2,68,23,1,1);      {ventana que borra}
{cuadrodoble(1,1,80,24,1,15);}
cuadrodoble(27,2,51,4,0,15);
gotoxy (32,3);
write ('MODO AUTOMATICO');
cuadrodoble(24,6,54,10,7,0);
cuadrodoble(24,10,54,23,7,0);
gotoxy (24,10);
write(' =');
gotoxy (54,10);
write('= ');
textcolor(1);
gotoxy(35,11);
write('TRAYECTORIA');
cuadrosimple (31,13,39,15,7,0);
textcolor(4);
gotoxy (35,14);
write ('1');
textcolor(0);
gotoxy (41,14);
```

```
write (' RECTA');
cuadrosimple (31,16,39,18,7,0);
textcolor(4);
gotoxy (35,17);
write ('2');
textcolor(0);
gotoxy (41,17);
write (' CIRCULAR');
cuadrosimple (31,19,39,21,7,0);
textcolor(4);
gotoxy (35,20);
write ('3');
textcolor(0);
gotoxy (41,20);
write (' SALIR');
repeat
textcolor(7);
op:=readkey;
textcolor(0);
case op of
     '1': begin
          inicio:=10;
          textcolor(5);
          gotoxy(27,8);
          writeln ('PUNTO FINAL ');
          textcolor(1);
          gotoxy(50,8);
          read (fin);
          recta (inicio,fin);
          arreglo[10].x:=arreglo[fin].x;
          arreglo[10].y:=arreglo[fin].y;
          arreglo[10].z:=arreglo[fin].z;
          arreglo[10].th:=arreglo[fin].th;
          arreglo[10].tc:=arreglo[fin].tc;
          arreglo[10].tm:=arreglo[fin].tm;
          cuadrodoble(25,7,53,9,7,7);
          end;
     '2': begin
          inicio:=10;
          textcolor(5);
          gotoxy(27,8);
          writeln ('PUNTO MEDIO ');
          textcolor(1);
          gotoxy(50,8);
          read (medio);
          cuadrodoble(25,7,53,9,7,7);
```

```
            textcolor(5);
            gotoxy(27,8);
            writeln ('PUNTO FINAL ');
            textcolor(1);
            gotoxy(50,8);
            read (fin);
            circular(inicio,medio,fin);
            arreglo[10].x:=arreglo[fin].x;
            arreglo[10].y:=arreglo[fin].y;
            arreglo[10].z:=arreglo[fin].z;
            arreglo[10].th:=arreglo[fin].th;
            arreglo[10].tc:=arreglo[fin].tc;
            arreglo[10].tm:=arreglo[fin].tm;
            cuadrodoble(25,7,53,9,7,7);
            end;
      end;
until op='3';
menu_principal;
end;

procedure menu_manual;
{Este procedimiento nos da el menu del modo manual y hace llamadas
a los procedimientos motor y graba_pos}
begin
cuadrodoble(27,4,51,22,1,1);
cuadrodoble(29,2,53,4,0,15);
gotoxy (36,3);
write ('MODO MANUAL');
cuadrodoble(14,6,68,10,7,0);
cuadrodoble(14,10,68,23,7,0);
gotoxy (14,10);
write('=');
gotoxy (68,10);
write('=');
cuadrodoble (17,11,31,13,7,0);
textcolor(4);
gotoxy (21,12);
write ('1');
textcolor(0);
gotoxy (22,12);
write (' HOME');
cuadrodoble (34,11,48,13,7,0);
textcolor(4);
gotoxy (35,12);
write ('2');
textcolor(0);
```

```
gotoxy (36,12);
write (' MOVER BRAZO');
cuadrodoble (51,11,65,13,7,0);
textcolor(4);
gotoxy (55,12);
write ('3');
textcolor(0);
gotoxy (56,12);
write (' SALIR');
repeat
textcolor(7);
op:=readkey;
textcolor(0);
case op of
     '1': begin
          th:=0;
          tc:=0;
          tm:=-1.5708;
          dispa:=FALSE;
          motor(th,tc,tm,TRUE,dispa);
          end;
     '2': begin
          mueve_brazo;
          op:='3';
          end;
     end;
until (op='3');
menu_principal;
end;

procedure menu_principal;
{Este procedimiento es el menu principal y hace llamadas a
menu_manual,
menu_automatico y editor}
begin
cuadrodoble(9,2,68,23,1,1);      {ventana que borra}
cuadrodoble(27,4,51,6,0,15);
gotoxy (30,5);
write ('EDITOR DE LENGUAJE');
cuadrodoble(28,9,50,22,7,0);
textcolor(0);
cuadrodoble(28,9,50,11,7,0);
gotoxy (28,11);
write('=');
gotoxy (50,11);
write('=');
```

```
gotoxy (32,10);
write ('MENU PRINCIPAL');
gotoxy (30,13);
write ('1.- Modo Manual');
gotoxy (30,15);
write ('2.- Modo Automatico');
gotoxy (30,17);
write ('3.- Editor');
gotoxy (30,19);
write ('4.- Terminar');
gotoxy (37,21);
write ('Opcion ');
repeat
textcolor(0);
gotoxy (44,21);
op:=readkey;
case op of
     '1': begin
          menu_manual;
          end;
     '2': begin
          menu_automatico;
          end;
     '3': begin
          edit(arreglo);
          end;
    end;
until op='4';
end;

begin {programa principal}
cuadrodoble(0,0,85,25,0,0);
cuadrodoble(1,1,80,24,1,15);
textbackground(1);
menu_principal;
textbackground(0);
textcolor(15);
clrscr;
end.
```

APENDICE C

SUPLEMENTO DEL SISTEMA DE SOFTWARE

```
unit unidad;
interface
   uses
      crt;
      procedure cuadrodoble(x1,y1,x2,y2:word;fondo,linea:byte);
      procedure cuadrosimple(x1,y1,x2,y2:word;fondo,linea:byte);
      procedure pantal;
      procedure panta_m_a;
      procedure panta_m_m;
      procedure panta_m_p;
      procedure panta_m_e;
      procedure ayudas;
      procedure limpia;
      procedure limpia_fin;

implementation


   procedure cuadrodoble(x1,y1,x2,y2:word;fondo,linea:byte);
   var
      j,i:integer;


   begin
      textbackground(fondo);
      textcolor(linea);
      gotoxy(x1,y1);
      write('=');
      gotoxy(x2,y1);
      write('=');
      gotoxy(x1,y2);
      write('=');
      gotoxy(x2,y2);
      write('=');
      for i:=(x1+1) to (x2-1) do
      begin
         gotoxy(i,y1);
         write('=');
         gotoxy(i,y2);
         write('=');
      end;
      for i:=(y1+1) to (y2-1) do
      begin
         gotoxy(x1,i);
         write(' ');
         gotoxy(x2,i);
```

```
      write('');
   end;
   for i:=(y1+1) to (y2-1) do
      for j:=(x1+1) to (x2-1) do
      begin
         gotoxy(j,i);
         write(' ');
      end;
end;


procedure cuadrosimple(x1,y1,x2,y2:word;fondo,linea:byte);
var
   j,i:integer;

begin
   textbackground(fondo);
   textcolor(linea);
   gotoxy(x1,y1);
   write('');
   gotoxy(x2,y1);
   write('');
   gotoxy(x1,y2);
   write('');
   gotoxy(x2,y2);
   write('');
   for i:=(x1+1) to (x2-1) do
   begin
      gotoxy(i,y1);
      write('');
      gotoxy(i,y2);
      write('');
   end;
   for i:=(y1+1) to (y2-1) do
   begin
      gotoxy(x1,i);
      write('');
      gotoxy(x2,i);
      write('');
   end;
   for i:=(y1+1) to (y2-1) do
      for j:=(x1+1) to (x2-1) do
      begin
         gotoxy(j,i);
         write(' ');
      end;
```

```
end;


procedure pantal;
var
   i:integer;

begin
cuadrosimple(17,11,31,13,7,0);
textcolor(4);
gotoxy (19,12);
write ('f7');
textcolor(0);
gotoxy (23,12);
write ('disparo');
textcolor(4);
gotoxy (53,12);
write ('esc');
textcolor(0);
gotoxy (56,12);
write (' salir');
textcolor(0);
gotoxy(21,17);write('hombro');
gotoxy(39,17);write('codo');
gotoxy(56,17);write('mano');
cuadrosimple(19,14,28,16,7,0);
textcolor (4);
gotoxy(20,15);write(' f1 ');
textcolor (0);
gotoxy(24,15);write('h > ');
cuadrosimple(36,14,45,16,7,0);
textcolor (4);
gotoxy(37,15);write(' f3 ');
textcolor (0);
gotoxy(41,15);write('c ^ ');
cuadrosimple(53,14,62,16,7,0);
textcolor (4);
gotoxy(54,15);write(' f5 ');
textcolor (0);
gotoxy(58,15);write('m ^ ');
cuadrosimple(19,18,28,20,7,0);
textcolor(4);
gotoxy(20,19);write(' f2  ');
textcolor (0);
gotoxy(24,19);write('h < ');
cuadrosimple(36,18,45,20,7,0);
```

```
textcolor (4);
gotoxy(37,19);write(' f4  ');
textcolor (0);
gotoxy(41,19);write('c v ');
cuadrosimple(53,18,62,20,7,0);
textcolor (4);
gotoxy(54,19);write(' f6  ');
textcolor (0);
gotoxy(58,19);write('m v ');
textcolor (0);
gotoxy(23,22);
write ('localidad   1  2  3  4  5  6  7  8  9');
end;

procedure panta_m_a;
begin
cuadrodoble(14,2,68,23,1,1);
cuadrodoble(27,2,51,4,0,15);
gotoxy (32,3);
write ('modo automatico');
cuadrodoble(24,6,54,10,7,0);
cuadrodoble(24,10,54,23,7,0);
gotoxy (24,10);
write('╓');
gotoxy (54,10);
write('╖');
textcolor(1);
gotoxy(35,11);
write('trayectoria');
cuadrosimple (31,13,39,15,7,0);
textcolor(4);
gotoxy (35,14);
write ('1');
textcolor(0);
gotoxy (41,14);
write (' recta');
cuadrosimple (31,16,39,18,7,0);
textcolor(4);
gotoxy (35,17);
write ('2');
textcolor(0);
gotoxy (41,17);
write (' circular');
cuadrosimple (31,19,39,21,7,0);
textcolor(4);
gotoxy (35,20);
```

```
write ('3');
textcolor(0);
gotoxy (41,20);
write (' salir');
end;

procedure panta_m_m;
begin
cuadrodoble(27,4,51,22,1,1);
cuadrodoble(29,2,53,4,0,15);
gotoxy (36,3);
write ('modo manual');
cuadrodoble(14,6,68,10,7,0);
cuadrodoble(14,10,68,23,7,0);
gotoxy (14,10);
write('╒');
gotoxy (68,10);
write('╕');
cuadrodoble (17,11,31,13,7,0);
textcolor(4);
gotoxy (21,12);
write ('1');
textcolor(0);
gotoxy (22,12);
write (' home');
cuadrodoble (34,11,48,13,7,0);
textcolor(4);
gotoxy (35,12);
write ('2');
textcolor(0);
gotoxy (36,12);
write (' mover brazo');
cuadrodoble (51,11,65,13,7,0);
textcolor(4);
gotoxy (55,12);
write ('3');
textcolor(0);
gotoxy (56,12);
write (' salir');
end;

procedure panta_m_p;
begin
cuadrodoble(9,2,68,23,1,1);
cuadrodoble(27,4,51,6,0,15);
gotoxy (30,5);
```

```
write ('editor de lenguaje');
cuadrodoble(28,9,50,22,7,0);
textcolor(0);
cuadrodoble(28,9,50,11,7,0);
gotoxy (28,11);
write('╠');
gotoxy (50,11);
write('╣');
gotoxy (32,10);
write ('menu principal');
gotoxy (30,13);
write ('1.- Modo manual');
gotoxy (30,15);
write ('2.- Modo automatico');
gotoxy (30,17);
write ('3.- Editor');
gotoxy (30,19);
write ('4.- Terminar');
gotoxy (37,21);
write ('opcion ');
end;

procedure panta_m_e;
begin
textbackground(1);
cuadrodoble(7,3,70,22,1,1);
cuadrodoble(27,3,51,5,0,15);
gotoxy (32,4);
write ('menu del editor');
cuadrodoble(26,7,52,22,7,0);
textcolor(0);
gotoxy(30,9);
write('1) crear programa ');
gotoxy(30,11);
write('2) cargar programa ');
gotoxy(30,13);
write('3) listar programas ');
gotoxy(30,15);
write('4) borrar programa ');
gotoxy(30,17);
write('5) renombrar programa ');
gotoxy(30,19);
write('6) terminar ');
end;
```

```
  procedure ayudas;
  {procedimiento que muestra las ayudas al usuario}
  begin
    cuadrosimple(2,22,79,25,1,15);
    gotoxy (5,23); write ('f1');
    gotoxy (12,23); write ('f2');
    gotoxy (21,23); write ('f3');
    gotoxy (31,23); write ('f4');
    gotoxy (39,23); write ('f5');
    gotoxy (45,23); write ('f6');
    gotoxy (53,23); write ('ins');
    gotoxy (64,23); write ('del');
    gotoxy (74,23); write ('f7');
    gotoxy (4,24); write ('movel');
    gotoxy (11,24); write ('movec');
    gotoxy (18,24); write ('velocidad');
    gotoxy (29,24); write ('disparo');
    gotoxy (38,24); write ('alto');
    gotoxy (44,24); write ('menu');
    gotoxy (50,24); write ('ins linea');
    gotoxy (61,24); write ('del linea');
    gotoxy (72,24); write ('ejecuta');
  end;

  procedure limpia;
  begin
       cuadrodoble(0,0,85,25,0,0);
       cuadrodoble(1,1,80,24,1,15);
       textbackground(1);
  end;

  procedure limpia_fin;
  begin
       textbackground(0);
       textcolor(15);
       clrscr;
  end;

end.
```