ESPOL
1958
ESCUELA SUPERIOR
POLITECNICA DEL LITORAL

Escuela Superior
POLITECNICA DEL LITORAL
Facultad de Ingeniería
en Electricidad y Computación

"Diseño y Construcción de un Brazo Mecánico controlado por un Computador Personal"

TESIS DE GRADO

Previa a la obtención del Título de
INGENIERO EN ELECTRICIDAD
Especialización: Electrónica

Presentada por

Adrian Oswaldo Arce Bastidas
Pablo Germán Parra Rosero

1997

AGRADECIMIENTO

A Dios.

Al Ing. HUGO VILLAVICENCIO, Director de Tesis, por su gran ayuda en la realización del presente trabajo.

Al Sr. RUFINO ASSAN, por su valiosa colaboración en la construcción y montaje de este proyecto.

Al personal DOCENTE Y ADMINISTRATIVO de la ESPOL, por todas las facilidades ofrecidas para el desarrollo de esta Tesis.

DEDICATORIA

A nuestros padres, por su gran apoyo durante toda nuestra formación estudiantil, les dedicamos la culminación de nuestra carrera.

A Pamela, para que en su vida futura este trabajo le sirva como fuente de inspiración, le dedicamos nuestro esfuerzo con mucho cariño.

MIEMBROS DEL TRIBUNAL

ING. ARMANDO ALTAMIRANO
PRESIDENTE DEL TRIBUNAL

ING. HUGO VILLAVICENCIO.
DIRECTOR DE TESIS

ING. RODRIGO BERREZUETA
MIEMBRO PRINCIPAL

ING. MIGUEL YAPUR
MIEMBRO PRINCIPAL

DECLARACION EXPRESA

"La responsabilidad por los hechos, ideas y doctrinas expuestas en esta tesis, me corresponden exclusivamente; y, el patrimonio intelectual de la misma, a la ESCUELA SUPERIOR POLITECNICA DEL LITORAL".

(Reglamento de Exámenes y Títulos Profesionales de la ESPOL).

ADRIAN ARCE BASTIDAS

PABLO PARRA ROSERO

RESUMEN

En los actuales momentos se ha vuelto muy necesario automatizar los sistemas de producción en serie, una de las alternativas más viables para lograr esto es el uso de los robots. Un modelo muy conocido es el Robot Vertical Articulado, este es utilizado actualmente por gran parte de las fábricas automatizadas de vehículos.

Aunque en el Ecuador el uso y desarrollo de la robótica es incipiente, en esta ocasión se ha diseñado y construido un prototipo básico de un robot industrial, utilizando los medios y la tecnología disponibles en el País. El proyecto desarrollado es un Brazo Mecánico controlado por un computador personal

El desarrollo de esta tesis se lo realiza en cinco capítulos que se definen brevemente a continuación:

El Capítulo I muestra un enfoque teórico de los principios básicos de la robótica, su clasificación e implicaciones sociales. Además se describe un Brazo Mecánico, con su sistema actuador y controlador.

En el Capítulo II se hace referencia al diseño del sistema mecánico, se describe el modelo utilizado y el dimensionamiento de las diferentes partes. Además se realiza un análisis estático de cada una de las articulaciones del

brazo, se detalla también los materiales utilizados y el ensamblaje las diferentes partes componentes del sistema mecánico. Se adjunta en este capítulo una sección de observaciones, dentro de la cual se anotan algunas de las experiencias suscitadas durante la construcción del proyecto.

El Capítulo III describe el diseño y montaje del sistema eléctrico, en este se detallan las diferentes partes que constituyen el sistema controlador del brazo, así, se hace referencia al computador, interfase digital, interfases de poder, motores, sensores y fuentes de poder. Cada una de las interfases esta descrita junto con su diagrama esquemático y el circuito impreso utilizado para su implementación.

El programa de control es descrito completamente en el Capítulo IV, en el que se detalla los procedimientos, funciones, rutinas y subrutinas utilizadas, escritas tanto en lenguaje C, como en lenguaje ensamblador. Además se adjuntan diagramas de flujo y el código del programa.

Finalmente, en el Capítulo V se ofrece un manual de operación para el usuario, el mismo que está orientado al manejo y control del brazo, también se describen las condiciones de falla, y los pasos a seguir para darles solución.

INDICE GENERAL

CAPITULO I

CONCEPTOS BASICOS DE ROBOTICA

CAPITULO II

DISEÑO DEL SISTEMA MECANICO

CAPITULO III

DISEÑO DEL SISTEMA ELECTRICO

CAPITULO IV

EL PROGRAMA DEL SISTEMA

CAPITULO V

MANUAL DEL USUARIO

INDICE DE FIGURAS

INTRODUCCION

El diseño y construcción de un robot en nuestro medio puede ser algo muy complicado e inusual, pero, con la elaboración de este proyecto se pretende demostrar que es factible la aplicación de la ingeniería electrónica en el desarrollo de proyectos de este tipo.

Investigar sobre los fundamentos básicos de la robótica, sus implicaciones en la industria y sus proyecciones futuras, entre otros es uno de los objetivos que se persiguen con el desarrollo de este trabajo.

La meta de este proyecto es diseñar e implementar un sistema robótico que sirva como base para la construcción de sistema similar, pero de uso industrial a mayor escala. Primordialmente se desea incentivar para que futuras generaciones de universitarios de nuestro país, emprendan investigaciones y desarrollen aplicaciones en el apasionante campo de la robótica.

Esta tesis pretende lograr dichos objetivos, utilizando las facilidades prestadas por lenguajes C y ensamblador, y aplicando las herramientas de la teoría electrónica puestas al alcance de la ingeniería. Finalmente como producto del desarrollo de esta tesis queda un sistema robótico completamente terminado y en pleno funcionamiento, apto para ser utilizado en investigaciones posteriores, para su mejoramiento.

CAPITULO I

CONCEPTOS BASICOS DE ROBOTICA

1.1 OBJETIVO

En los últimos años hemos visto que la tecnología se ha desarrollado de manera sorprendente, antes ya se vió como la energía mecánica reemplazo al músculo humano, hoy se trata de reemplazar al cerebro humano con circuitos electrónicos para el control de diversos tipos de procesos. Actualmente los robots no sólo se usan para experimentar, pués, su utilización se ha convertido en una imperiosa necesidad para el normal desenvolvimiento y desarrollo de la humanidad.

Este capítulo procura presentar ideas, fundamentos y conceptos básicos de lo que es la Robótica, que sirvan como guía para el diseño de un brazo mecánico controlado por computador, tomando como base el modelo de un robot vertical articulado.

1.2 INTRODUCCION

En este capítulo se expone la teoría de la constitución básica de un brazo mecánico y la interrelación existente entre sus componentes. Así, durante el desarrollo se detalla los diferentes tipos de junturas, enlaces y manejadores que existen y pueden ser utilizados en el diseño robótico.

Se da a conocer una breve reseña histórica de la Robótica, la definición de los robots y la clasificación de los mismos, sin descuidar las implicaciones sociales de la Robótica.

El enfoque del capítulo gira alrededor de lo que es un robot vertical articulado, pués, este es el modelo considerado en el diseño y construcción del proyecto, objeto de esta tesis. Dicho modelo ha sido seleccionado por ser el más popular de los robots, además por su volumen de cobertura de trabajo y por las facilidades que presta para su construcción en nuestro medio.

1.3 QUE ES UN ROBOT?

Algunas personas aspiran a encontrar en los robots una respuesta a los diferentes problemas que aquejan a la sociedad, basados en la idea de los robots aumentan los niveles de productividad, debido a que a diferencia de los humanos son capaces de realizar trabajos repetitivos y aburridos sin presentar cansancio o disgusto alguno y no necesariamente relizándolo con mayor rapidez o efectividad que un humano; pero, todo esto no es suficiente para solucionar ó poner fin a los problemas que afectan a una sociedad.

1.3.1 BREVE HISTORIA

Durante algún tiempo muchos escritores han tenido en los robots una fuente de inspiración, por ejemplo en 1921 el escritor Checoslovaco Karel Capek escribió una obra titulada R.U.R (Rossum's Universal Robots). En lenguaje Eslavo tal como Checo o Polaco, la palabra Robot significa trabajador.

Isaac Asimov en 1950 publica ROBOT I , en esta obra no se trata el aspecto físico de la máquina sino el proceso mental de una máquina que podría literalmente ser obligada a obedecer órdenes contradictorias. Es interesante notar que el desarrollo de

la obra de Asimov fue concordante con el desarrollo del computador en la esfera pública porque es al computadora la que ha hecho posible la existencia de los Robots hoy en día.

Por siglos el autómata mecánico ha sido hecho por las personas de acuerdo a su propia imagen, pero, cuando la tecnología computacional combinó aquellos autómatas con las capacidades computacionales algunas prácticas aplicaciones fueron posibles.

La gran mayoría de las industrias de los países desarrollados utilizan a los robots que desde hace algunos años atrás, y que han venido siendo producidos por compañías tales como IBM, Westinhouse, General Electric, Bendix, y Hobart, las mismas que mostraron públicamente algunos de sus modelos en el año de 1982, aunque antes en 1972 IBM ya había experimentado con Robots ; General Electric y Westinhouse lo habían hecho ya desde los mediados de 1970.

El primer robot industrial Japonés fue desarrollado en 1969, esto fue dos años después de que importaron el primer Robot VERSATRAN. El entusiasmo Japonés fue tan evidente que la Asociación de Robots Industriales del Japón (JIRA) fue fundada en 1971, cuatro años antes que el Instituto de Robots de América

(RIA) y seis años antes que la Asociación Británica de Robots (BRA). Se puede afirmar que Japón no se durmió en los laureles en asuntos de Robótica como otros países. Ellos tomaron la idea Americana y desarrollaron esa idea a su completo potencial.

Para el año de 1985 habían más de 50 fabricantes de Robots en los Estados Unidos, y más de 300 en todo el mundo.

Hoy en día Japón es el usuario más grande de Robots industriales en el mundo, incluso algunas compañías tales como SONY Corporation, SAMSUNG, Honda, entre otras han mostrado al mundo robots tipo mascota, robots antropomórficos que pueden realizar ciertas actividades tales como subir escaleras, jugar tenis, tocar piano, etc.

1.3.2 DEFINICION

Existen muchas definiciones de lo que es un robot, algunos expertos los definen como tontos, sin habilidad, máquinas de un solo brazo, capaces de ejecutar tareas simples. A continuación se transcribe la definición más aceptada que ha sido publicada por el Instituto de Robótica de América :

Un manipulador programable, multifuncional diseñado para mover materiales, partes, herramientas ó dispositivos especiales por medios de movimientos programados para la ejecución de una variedad de tareas.

Esta definición no cubre todo el concepto de un robot, pués, un robot es una máquina que puede ser programada y reprogramada para realizar una infinidad de tareas de acuerdo a la necesidad que tiene que satisfacer.

1.3.3 CLASIFICACION

Existen varias clasificaciones de los Robots, estas pueden ser tan estrechas o variadas dependiendo de los parámetros que se consideren. Por ejemplo en Japón se consideran seis distintas clasificaciones, mientras en Estados Unidos hay solamente dos grandes clasificaciones, aunque algunos expertos consideran la existencia de tres clases de Robots.

Los Americanos clasifican a los Robots como SERVO Y NO SERVO de acuerdo a su grado de evolución y niveles de sofisticación, esta clasificación también es conocida como de Primera y Segunda Generación. A los Robots no servo se los

conoce como no inteligentes y a los servo ser los divide como inteligentes y muy inteligentes.

El Robot no servo está constituido por un sistema de lazo abierto, esto implica que no tiene un mecanismo de realimentación, por lo cual no puede comparar su posición real con la posición programada.

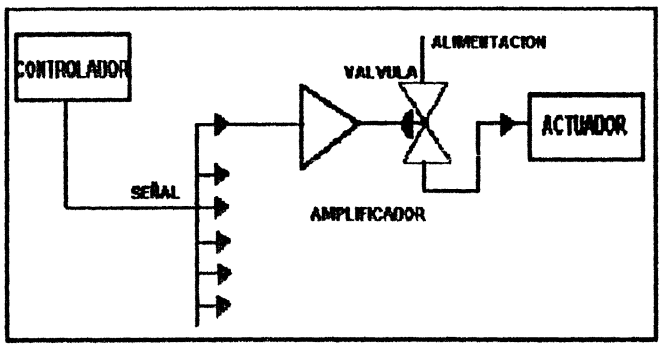


Fig. 1.3.3.a Diagrama No Servo

El Robot servo es más sofisticado y está constituido por un sistema de lazo cerrado, es decir que cuenta con un mecanismo de realimentación que le permite comparar su posición real con la posición programada. Esta comparación hace posible hacer los ajustes necesarios al sistema, siendo este proceso repetido hasta lograr la posición programada.

Dentro de los Robots tipo servo se encuentra una marcada diferencia entre los servo-inteligentes y los servo-muy inteligentes, esta es que los altamente inteligentes utilizan sensores externos además de los regulares usados internamente en las junturas mientras que los servo inteligentes no poseen este tipo de sensores externos. Los principales tipos de sensores externos son los de visión y los de tacto, estos proveen al robot de cierto grado de decisión que a su vez ayuda al Robot a tomar medidas correctivas y decisiones propias.

En Japón existen estándares industriales para definir robots y manipuladores:

Un manipulador es un dispositivo para manejar objetos como se desee sin tocarlos con las manos y tiene más de dos posibilidades de movimiento, tales como revolución, dentro-fuera, arriba-abajo, transporte izquierda-derecha, de manera que puede transportar espacialmente un objeto cogiéndolo, adheriéndolo y así por el estilo.Un Robot es definido como un sistema mecánico que tiene funciones flexibles de moción como los organismos vivientes o que combina tales funciones de moción con funciones inteligentes, y que

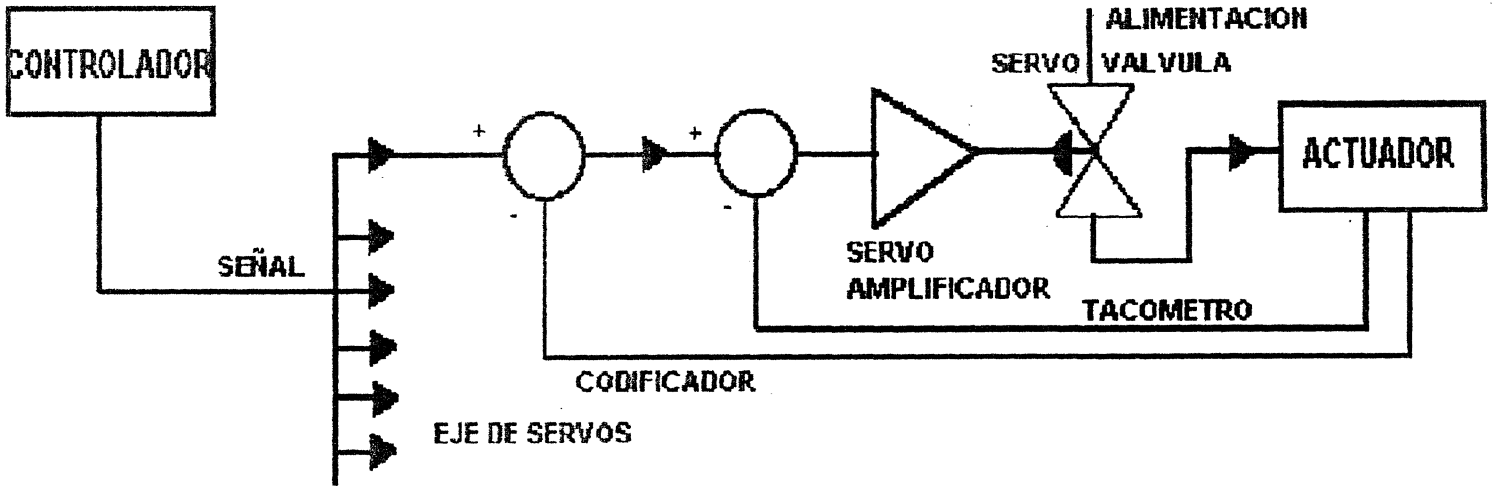


Fig.1.3.3.b Diagrama Servo

actúa en respuesta a lo que el humano quiere. En este contexto funciones inteligentes significan la habilidad para presentar al menos una de las siguientes características: juicio, reconocimiento ó aprendizaje.

Los diferentes grupos de Robots japoneses se clasifican de acuerdo a la entrada de información y métodos de enseñanza, así tenemos:

- Manipulador Manual
- Robot de Secuencia Fija
- Robot de Secuencia Variable
- Robot de Reproducción
- Robot Controlado Numéricamente
- Robot Inteligente

1.4 EL BRAZO MECANICO

Por lo descrito anteriormente se puede afirmar que un Robot es prácticamente un brazo mecánico, diseñado para ejecutar diversidad de movimientos y acciones controladas por el computador, el mismo que

recibe información del ambiente circundante al brazo a través de sensores.

En la figura 1.4.a se ilustran los tres principales componentes de un brazo mecánico visto como un sistema robótico.

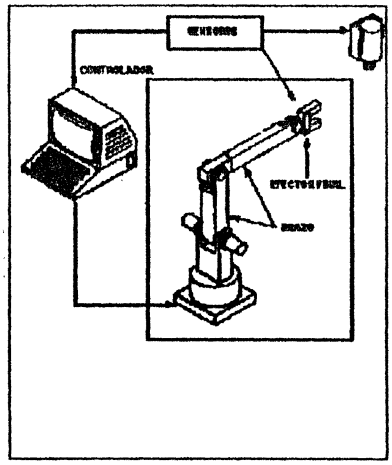


Fig. 1.4.a Las tres principales unidades de un Sistema Robótico

El brazo robótico presenta movimientos espaciales. Pegado al extremo tiene al efector final que es el encargado de llevar a cabo las tareas asignadas por el operador. El tipo de efector final varía de acuerdo al trabajo a realizar, por ejemplo en Robots usados para transferencia de objetos de un lugar a otro, el efector final es una tenaza.

Todos los brazos robóticos al igual que el brazo humano están constituidos por una serie de enlaces y junturas. Una juntura es aquella

parte de brazo que conecta dos enlaces y permite el movimiento relativo entre ellos.

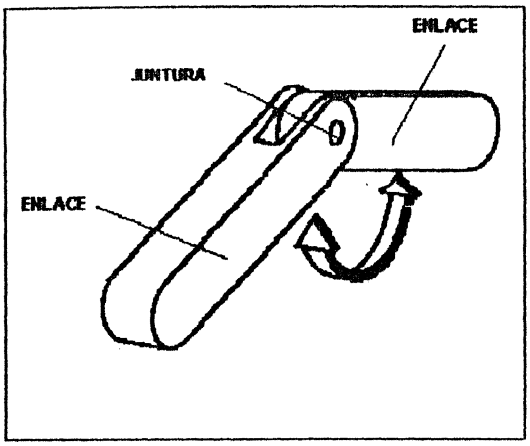


Fig. 1.4.b Juntura típica de un brazo mecánico

Los robots suelen tener una base que normalmente se fijan a una pared, al piso ó al techo; el primer enlace del robot es conectado a la base , y el último enlace es conectado al efector final.

1.4.1 TIPOS DE JUNTURAS

La gran mayoría de los brazos mecánicos tienen dos tipos de junturas, prismáticas y revolutas, aunque existe una tercera clase de juntura, la esférica ó también conocida como de bola y zócalo.

La Juntura Prismática permite movimientos lineales entre dos enlaces. Está compuesta de dos enlaces anidados, esto quiere decir que el un enlace se desliza sobre el otro. En otras palabras una parte puede moverse en línea recta hacia afuera ó hacia adentro en relación a la otra parte como se puede observar en la figura 1.4.1a

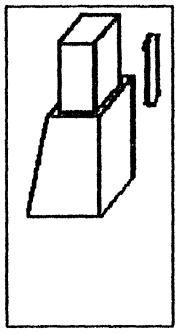


Fig.1.4.1.a Juntura Prismática

La Juntura Revoluta permite la rotación entre dos enlaces. Los dos enlaces están unidas por un eje pasador común da tal forma que uno de los enlaces puede moverse radialmente en forma de arco en relación a la otra como se detalla en la figura 1.4.1.b

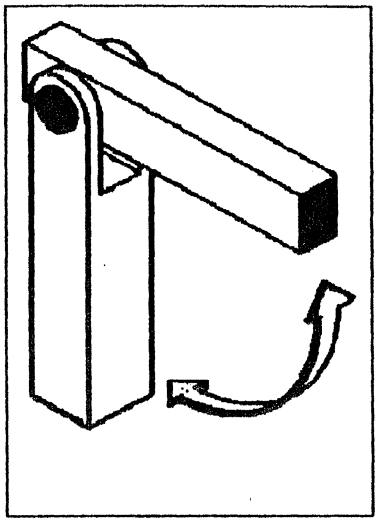


Fig. 1.4.1.B Juntura Revoluta

La Juntura Esférica funciona como una combinación de tres junturas revolutas permitiendo un movimiento rotacional alrededor de los tres ejes como se puede observar en la figura 1.4.1.c

1.4.2 Clasificación de los Robots según el Tipo de Juntura

Según el tipo de juntura los brazos robóticos pueden ser clasificados en cinco grupos:

- Cartesianos
- Cilíndricos

- Esféricos

- Horizontal Articulados

- Vertical Articulados

El código utilizado para estas clasificaciones consiste en un conjunto de tres letras referidas a los tipos de junturas (R para revoluta, P para prismática). Estas letras deben ser ordenadas comenzando con la más cercana a la base, por ejemplo PRP indica un robot cuya juntura de la base es prismática y cuya segunda y tercera juntura son revoluta y prismática respectivamente.

Los Robots Cartesianos tienen tres junturas prismáticas y por eso se los clasifican como PPP.

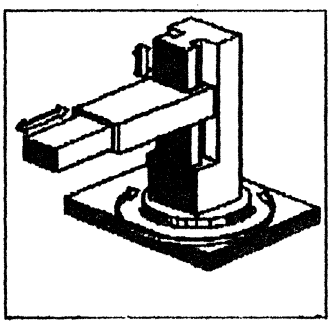


Fig. 1.4.2.b Robot Cilíndrico

Un Robot Cilíndrico está constituido por una juntura revoluta y dos junturas prismáticas. Su código es RPP.

El Robot Esférico tiene dos junturas revolutas y una prismática. Su código es RRP.

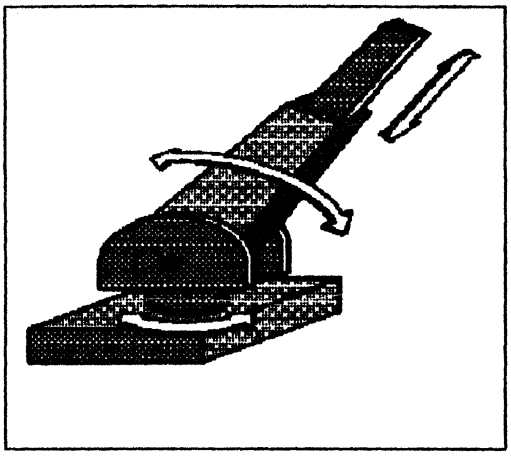


Fig.1.4.2.c Robot Esférico

El Robot Horizontal Articulado tiene dos junturas revolutas y una prismática. El código que lo representa es RRP.

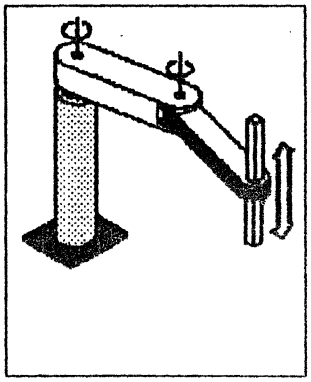


Fig.1.4.2.d Robot Horizontal Articulado

Un Robot Vertical Articulado incluye tres junturas revolutas y su código es RRR. Este tipo de Robot es el que se ha seleccionado para el diseño y construcción del proyecto objeto de esta tesis.

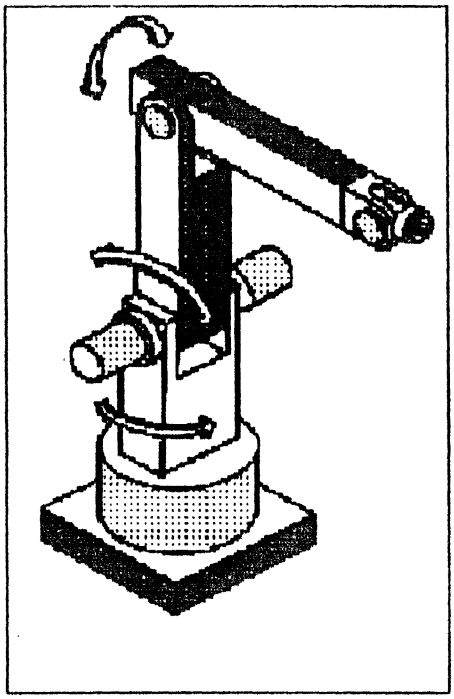


Fig.1.4.2.e Robot Vertical Articulado

1.4.3 VOLUMEN DE TRABAJO

En esta sección se muestra una serie de cálculos que facilitan el análisis de la capacidad de trabajo de los diferentes modelos de Robots.

El Robot Cartesiano es capaz de alcanzar cualquier punto dentro de un espacio cúbico de arista L, por lo tanto su volumen de trabajo es:

$$V_{\text{cartesiano}} = L^3 \quad (\text{unidades})^3$$

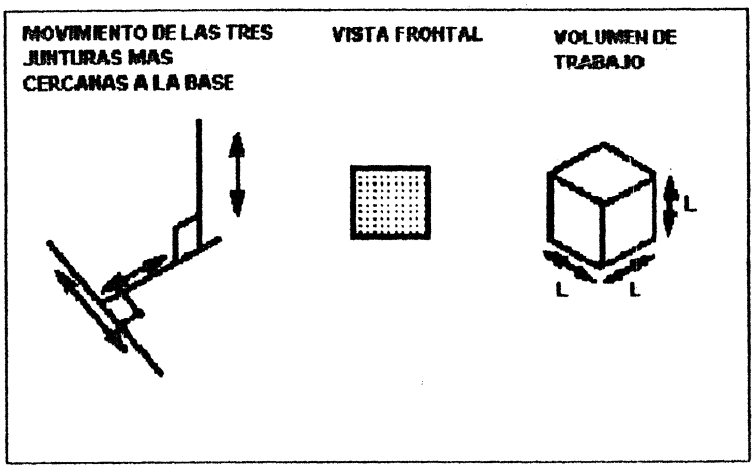


Fig.1.4.3.a Volumen de trabajo de un robot Cartesiano

Un modelo Cilíndrico puede alcanzar cualquier punto en un cilindro de altura L y radio 2L, excepto los puntos al interior del cilindro de altura L y radio L, entonces tenemos que su volumen de trabajo está dado por:

$$V_{\text{cilíndrico}} = L[\pi(2L)^2 - \pi L^2] = 3\pi L^3$$

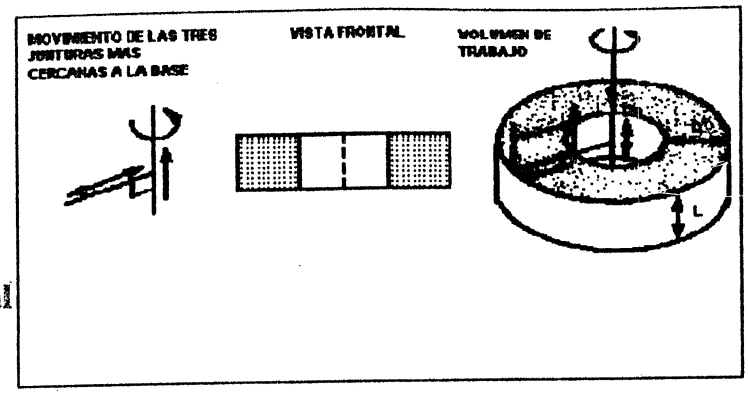


Fig 1.4.3.b Volumen de trabajo de un Robot Cilíndrico

En la mayoría de los Robots Cilíndricos el volumen de trabajo se limita al área cercana al eje de rotación.

El Robot Esférico puede alcanzar cualquier punto dentro de un espacio esférico de radio 2L, excepto los puntos al interior de la esfera de radio L. Su volumen de trabajo está determinado así:

$$V_{esférico} = (4\pi/3)\,(2L)^3 - (4\pi/3)L^3 = (28\pi/3)\,L^3$$

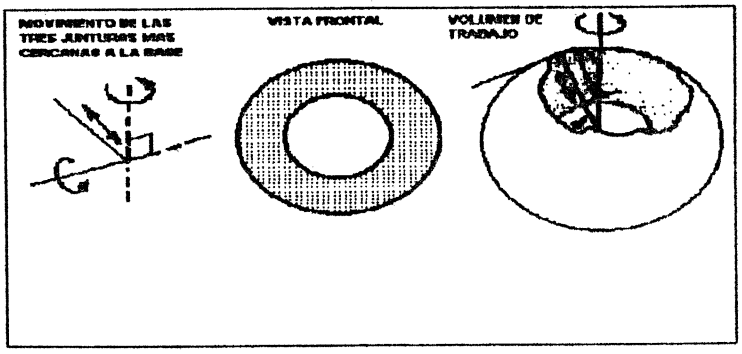


Fig.1.4.3.c Volumen de trabajo de un modelo Esférico

Un Robot Horizontal Articulado puede alcanzar cualquier punto en un cilíndro de altura L y radio 2L, su volumen de trabajo es:

$$V_{\text{horizontal articulado}} = L\pi(2L)^2 = 4\pi L^3$$

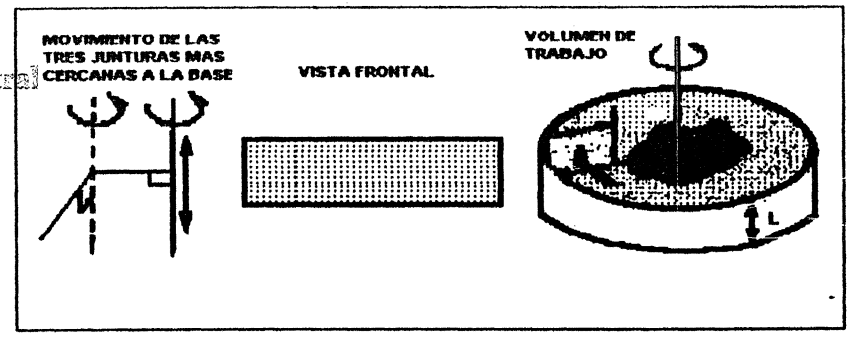


Fig.1.4.3.d Volumen de trabajo de un Robot Horizontal Articulado

El modelo Vertical Articulado puede alcanzar cualquier punto en una esfera de radio 2L, su volumen de trabajo es:

$$V_{\text{vertical articulado}} = (4\pi/3)(2L)^3 = (32\pi/3)\,L^3$$

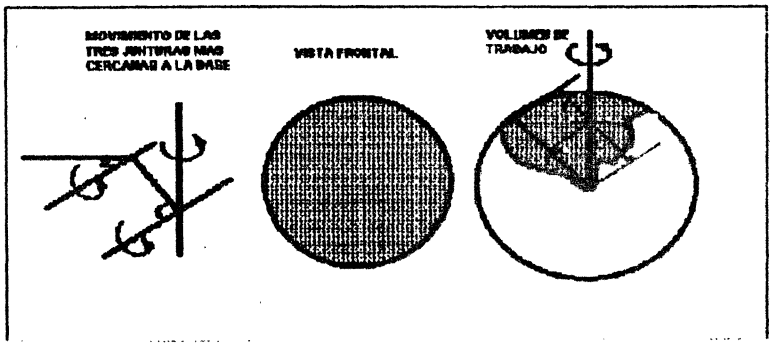


Fig.1.4.3.e Volumen de trabajo de un Robot Vertical Articulado

Debido a las limitaciones mecánicas el volumen de trabajo real es menor que el teórico, lo cual impide que el efector final alcance todos los puntos de su volumen teórico de trabajo.

1.4.4 ENLACES

Una consideración muy importante en la construcción de un Robot es el efecto de las cargas sobre las partes móviles. El brazo mecánico debe ser ligero en peso pero debe tener un alto grado de rigidez mecánica, si el brazo es muy pesado necesita motores más grandes, lo cual incrementa su costo, por otro lado un brazo con baja rigidez reduce la precisión del Robot debido a vibraciones y a la mala respuesta al estrés. Un ejemplo de un brazo con un bajo grado de rigidez es una caña de pescar en la cual la precisión con la que el efector final (anzuelo) trabaja es muy baja, y esto es medible en yardas. La precisión requerida por la mayoría de los Robots es medible en fracciones de pulgada.

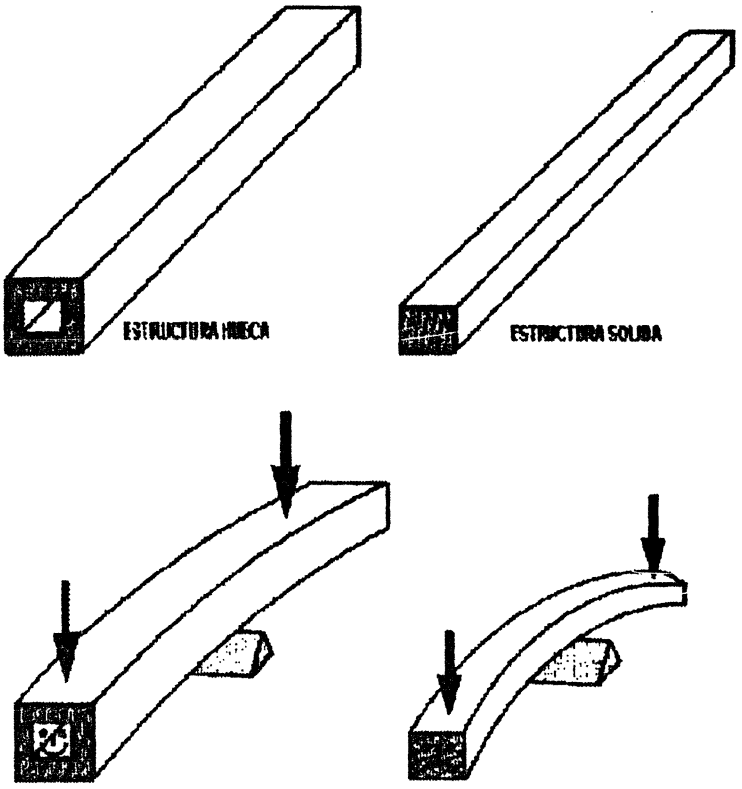


Fig.1.4.4.a Comparación de rigidez de dos tipos de Enlace.

1.4.5 UNIDADES DE MANEJO

Así como el brazo humano requiere del músculo para su movimiento, un brazo mecánico también requiere de unidades de

manejo que muevan las distintas partes del brazo. Existen varios tipos de manejadores, los cuales son clasificados como:

- Revolutos y Prismáticos
- Eléctricos, Hidráulicos y Neumáticos
- Directos e Indirectos

Los Manejadores Revolutos generalmente son los motores que una vez conectados a su fuente de energía, ya sea esta eléctrica, de presión, etc, responden con un movimiento rotacional. La carga unida al eje de este motor es movida por la rotación del eje.

Un Manejador Prismático es un cilíndro hidráulico o neumático formando una juntura prismática. El movimiento lineal puede resultar del movimiento rotacional por medio del uso de un mecanismo de tornillo sin fin.

Los Manejadores Eléctricos son motores eléctricos acoplados al Robot y alimentados por una fuente de energía eléctrica. Se usan varios tipos de motores eléctricos como manejadores de Robots: motores DC, motores de paso y motores AC.

La mayoría de los nuevos Robots son manejados por motores DC antes que por manejadores hidráulicos ó neumáticos,

principalmente debido al alto grado de precisión y a la simplicidad del control de motores eléctricos.

A continuación se describen las principales ventajas de los manejadores eléctricos.

- Permiten eficiencia y control preciso.
- Tienen una estructura simple y fácil de mantener
- No requieren de una fuente de energía costosa
- Su costo es relativamente bajo
- Son fáciles de conseguir en el mercado

Las principales desventajas de este tipo de manejadores son:

- Imposible mantener un torque constante a velocidades diferentes
- Son susceptibles al daño con cargas tan pesadas como para detener
  el motor
- Tienen una baja razón de poder de salida a peso, requiriendo
  motores grandes en el brazo

Los Manejadores Hidráulicos requieren de motores tipo bomba y de cilíndros de movimiento prismático. Estas unidades causan el movimiento de partes tales como pistones, utilizando aceite comprimido. Entre sus principales características se anotan:

- Mantienen un alto y constante torque sobre un amplio rango de velocidades. El momento permanece alto aún cuando arranca desde la velocidad cero.
- Permiten operaciones de precisión un tanto menores que los manejadores eléctricos, esto se debe a que el aceite es no compresible.
- Requieren de fuentes de energía costosas
- Necesitan de mucho mantenimiento
- Utilizan válvulas de precisión bastante caras

La estructura de los Manejadores Neumáticos es similar a la de los hidráulicos, incluyen motores neumáticos (compresores) para movimientos rotacionales, además utilizan cilíndros neumáticos para movimiento prismático. Sus principales características se mencionan a continuación.

- Posibilita extremadamente altas velocidades de operación

- Su costo es relativamente bajo

- Son sistemas de fácil mantenimiento

- No puede alcanzar altos niveles de precisión

- El brazo eatá sujeto a vibraciones momentáneas cuando el motor motor ó el cilindro se detienen.

Se llama Manejador Directo cuando el motor está directamente montado sobre las juntura a mover. Si el motor está montado lejos de la juntura, usualmente cerca de la base del robot, el movimiento de las junturas se da por medio de dispositivos de transmisión, los cuales pueden ser cadenas, bandas ó engranajes Un manejador dispuesto de esta manera es llamado Manejador Indirecto.

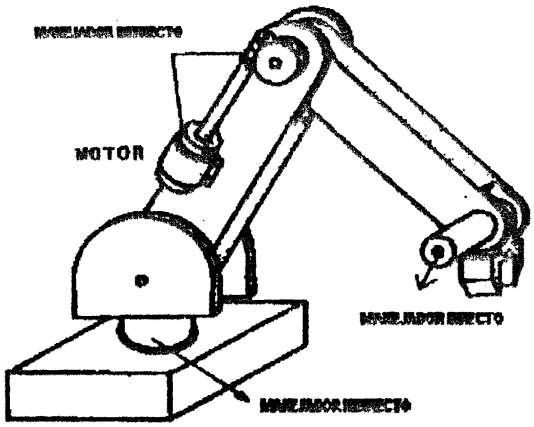


Fig.1.4.5.a Manejadores Directos e Indirectos

1.4.6 GRADOS DE LIBERTAD

Se conoce como grados de libertad al número de junturas que tiene un brazo robótico. Cada juntura permite un movimiento relativo entre dos enlaces con su respectivo grado de libertad. Cuando ocurre el movimiento relativo a lo largo ó alrededor de un eje simple, la juntura tiene un grado de libertad. Cuando el movimiento es a lo largo ó alrededor de más de un eje de movimiento, la juntura tiene dos ó tres grados de libertad.

La mayoría de los robots tienen entre cuatro y seis grados de libertad. Para comparación, el brazo humano desde el hombro hasta la muñeca tiene siete grados de libertad.

1.5 EFECTOR FINAL

El término efector final es una palabra genérica para todos los sistemas montados al final del Robot, esto es al final del enlace más lejano desde la base del robot, cuya tarea es coger objetos ó herramientas, y/ó transferirlos de un lugar a otro.

Como efectores finales pueden usarse tenazas, pistolas de soldar, herramientas de trabajo, sopletes para pintar, etc.

1.5.1 TENAZAS

Existe una gran variedad de tenazas utilizadas como efector final en un brazo mecánico, a continuación se mencionan algunas de ellas.

La Tenaza de Dos Dedos puede presentar movimiento paralelo o rotatorio de sus dedos. Su desventaja básica es la limitación para manejar objetos que sean más grandes que la abertura máxima de la tenaza.

Las Tenazas de Tres Dedos son similares a las de dos dedos pero permiten tomar con más seguridad objetos de forma triangular, circular ó irregular.

Existen Tenazas para Objetos Cilíndricos que tienen dos dedos, los cuales tienen ranuras circulares de diferente diámetro para poder coger objetos cilíndricos de distintas medidas.

Las Tenazas para Objetos Frágiles sirven para tomar objetos de contextura delicada, existen varios tipos de estas tenazas, uno de ellos incluye dos dedos flexibles que pueden deformarse hacia

dentro para tomar un objeto frágil. Los dedos están controlados por aire comprimido.

También hay Tenazas de Vacío y Electromagnéticas, las de vacío están diseñadas para adherirse a superficies planas, lo que se logra creando un vacío; el objeto permanece unido mientras el vacío exista. La tenaza electromagnética está diseñada para unirse a objetos ferromagnéticos. Ambas tenazas son muy eficientes debido a que pueden tomar objetos de varias medidas y no requieren gran precisión de posicionamiento.

1.5.2 HERRAMIENTAS COMO EFECTORES FINALES

En algunos tipos de trabajo surge la necesidad de que la tenaza pueda manejar objetos de diferentes formas y medidas, además el uso de una herramienta tal como un destornillador, un taladro puede alternarse con el uso de una tenaza.

Hasta la fecha no se ha inventado una tenaza que sea capaz de coger todos los objetos, por lo tanto se vuelve necesario crear una unidad llamada "Cambiador Automático de Tenaza", lo que es en realidad un adaptador que permite que la tenaza sea cambiada por otra de acuerdo a la necesidad, siempre y cuando no se aumente el peso del efector final.

1.6 IMPLICACIONES SOCIALES

El tema de la incidencia social de la Robótica suele ser muy delicado y complejo, pués, muchas personas piensan que el Robot ha desplazado al hombre de sus puestos de trabajo y que le ha restado importancia a las cualidades y destrezas humanas para desempeñar un gran número de actividades, pero, debe considerarse que aún estamos en las primeras etapas de la Robótica y que falta mucho por hacer en este campo como para pensar que el Robot pueda marginar o sustituir al hombre de la manera que se ha pensado.

Por otro lado deben analizarse todo el sinnúmero de beneficios que el hombre ha alcanzado gracias al desarrollo de la Robótica, hoy en día aquellos trabajos peligrosos, repetitivos, aburridos y muy cansados han sido asignados a los Robots para salvaguardar la integridad de muchos obreros que por cierto no necesariamente han sido marginados de su empleo, sino, que se los ha especializado en otras tareas ó a su vez se los ha puesto al frente del control y supervisión del trabajo desarrollado por los Robots.

Se debe considerar además que con la intervención de los Robots se han reducido los espacios físicos dentro de las industrias y se ha reducido además el exceso de personal, y de esta manera se ha

conseguido optimizar los procesos de producción y aumentar la productividad de la industria.

Como ejemplo de los beneficios que la Robótica la ha brindado a la humanidad podemos citar al Robot SOJOURNER, utilizado en el proyecto espacial de exploración MARS PATHFINDER, llevado a cabo en el mes de julio de 1997. Dicho Robot le permitido al hombre obtener una importantísima cantidad de información acerca del vecino planeta gracias a su facilidad de adaptación al medio. Esto por ejemplo es una actividad sencillamente imposible de ser realizada por humano alguno.

En Ecuador, el desarrollo de la Robótica es prácticamente nulo, este país tiene todavía grandes deficiencias socioeconómicas , las mismas que son un gran obstáculo para el desarrollo de cualquier actividad de investigación, pués, el gobierno tiene que atender problemas más apremiantes de la sociedad. Son las Universidades y Escuelas politécnicas del país las llamadas a incentivar y apoyar los trabajos de investigación que tengan relación con el desarrollo de la Electrónica en el medio, para de esta manera sentar una base sólida que permita efectuar avances dentro del campo de la Robótica. Existen muchas cosas que se pudieran hacer en el país y se evitaría el hecho de solamente comprar la tecnología y limitarse a utilizarla.

Uno de los principales objetivos del desarrollo de este proyecto de tesis ha sido el de incentivar a los estudiantes a desarrollar temas afines a la Robótica y de esta manera crear conciencia de que el desarrollo de un pueblo depende sus habitantes.

CAPITULO II

DISEÑO DEL SISTEMA MECANICO

2.1 OBJETIVO

El análisis y descripción de un Sistema Robótico suele ser bastante complejo y extenso, en vista de esto durante el desarrollo de esta tesis se busca analizar únicamente los parámetros básicos requeridos para el diseño y construcción, tanto de la parte actuadora como de la parte controladora de un brazo mecánico.

Con el fin de detallar cada uno de los sistemas y partes constitutivas del brazo mecánico, se ha procedido a dividir el problema de la construcción del mismo en dos grandes partes: Diseño Eléctrico y Diseño Mecánico. El objetivo de este capítulo es desarrollar el análisis del diseño del sistema mecánico.

Se desea dar una concepción general de la estructura básica del brazo y la interrelación de cada una de sus partes, además se definen las características y el trabajo que debe efectuar el brazo mecánico.

2.2 INTRODUCCIÓN

El Sistema Mecánico de un brazo robótico consta de varias partes tales como enlaces, junturas, engranajes, motores, etc. En este capítulo se describe a cada una de estas partes, su dimensionamiento, características, y función que desempeñan.

Se detallan completamente a los dos efectores finales utilizados, además se anotan experiencias sobre el ensamblaje y se desarrolla un análisis estático de la estructura del brazo mecánico.

Para una mejor comprensión, la descripción del sistema se lo hace por partes, es decir se analiza a cada una da las articulaciones por separado, sin descuidar la relación existente entre uno y otro enlace ó entre sus junturas .

2.3 MODELO UTILIZADO

Para la construcción del brazo mecánico se eligió como modelo base, el de un Robot Vertical Articulado debido a su gran popularidad y a su gran capacidad para actuar dentro de un espacio físico poco restringido. Este modelo es el que menos problemas ofrece para su construcción y además el que mejor responde a los propósitos de esta tesis.

Una vez seleccionado el modelo base se ha optado por construir un Robot tipo No Servo, es decir que no cuenta con un sistema de realimentación, pero sí con un sistema de control directo. Esta opción se ha elegido en vista de que el manejo de motores de paso no necesariamente requieren de un servomecanismo para su control . La decisión de utilizar motores de paso se debe a que ofrecen mayor torque y mejores facilidades de manejo que otro tipo de motores.

El brazo mecánico tiene junturas de tipo revoluta , las mismas que no unen directamente a los enlaces entre sí, sino que lo hacen indirectamente a través de reductores mecánicos. El brazo ha sido montado sobre una estructura en forma de paralelepípedo, esta es capaz de soportar todo el peso del brazo en sus diversas posiciones, además en dicha base están alojados los circuitos de manejo y poder del sistema.

Para efectuar su trabajo, el brazo debe tener cuatro grados de libertad, esto quiere decir que existirán cuatro junturas, siendo la más alejada a la base la encargada de mover el efector final, esto unido a tres enlaces le permitirán ubicarse con gran facilidad y flexibilidad en un punto deseado, dentro de su volumen de trabajo.

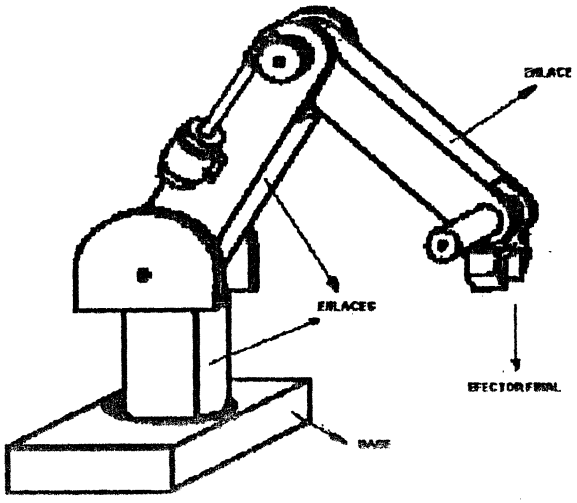


Fig.2.3.a Gráfico del Modelo utilizado

Se puede observar en la figura 2.3.a un bosquejo bastante generalizado de lo que es el brazo mecánico, más adelante se detalla a cada una de las partes componentes del mismo con todas sus propiedades, dimensiones y características.

2.4 DIMENSIONAMIENTO

En esta sección se procede a especificar medidas de longitud y peso de cada una de las articulaciones por separado, incluyendo enlaces,

junturas y efector final, para con estos datos realizar posteriormente un análisis estático del cual se desprenderán las características de los motores a utilizarse.

Para el dimensionamiento de las partes constitutivas del brazo se ha considerado el volumen de trabajo que se desea cubrir, tomando en cuenta que el robot constituye nada más que un prototipo de un brazo mecánico industrial. El volumen de trabajo estimado que desarrollará el brazo es de 10 pies cúbicos. Posteriormente se determina el volumen de trabajo exacto que desarrolla el brazo terminado.

A continuación se hace el detalle empezando por la articulación más lejana a la base, nombrada como Articulación # 1, y terminando con la Articulación # 4, que es la más cercana a la base del brazo.

2.4.1 ARTICULACION # 1

- Peso del Efector Final más carga: 1.0 lb.

- Longitud del Efector Final: 8.0 plg.

El peso y longitud del motor a utilizarse en esta articulación se lo determinará más adelante en el análisis estático.

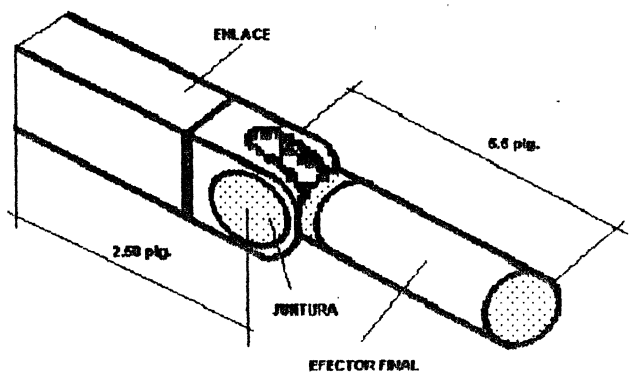


Fig.2.4.1.a Gráfico dimensionado de la Articulación # 1

2.4.2 ARTICULACION # 2

- Longitud del enlace: 6.0 plg.

- Peso de reductor más juntura 1.00 lb.

- Peso del enlace 0.30 lb.

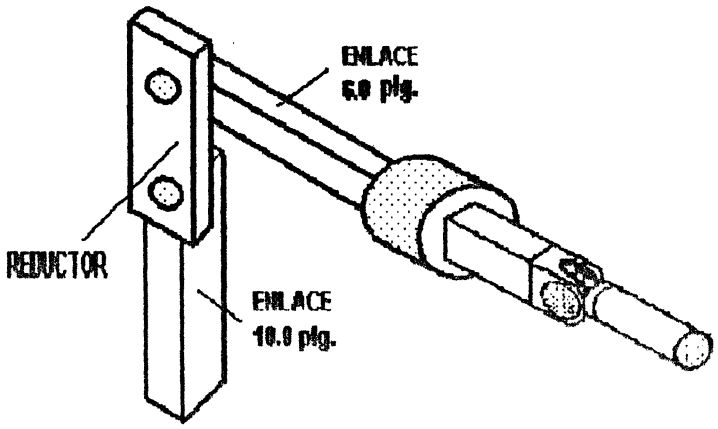


Fig. 2.4.2.a Gráfico dimensionado de la articulación # 2

El peso y longitud del motor para esta articulación es determinado posteriormente en el análisis estático.

2.4.3 ARTICULACION # 3

- Longitud del enlace: 10.0 plg.

- Peso del enlace: 0.45 lb.

- Separación entre enlaces: 1.20 plg.

El peso y dimensionamiento del motor para esta articulación es obtenido más adelante.

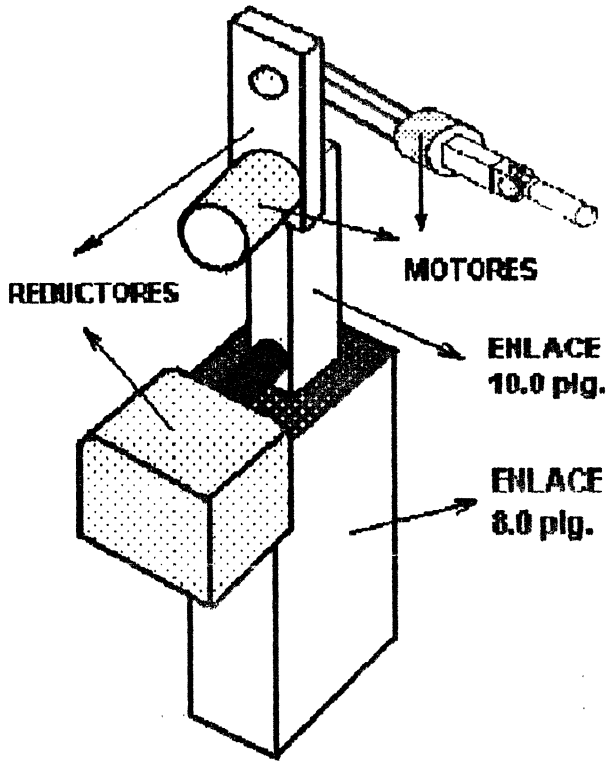


Fig 2.4.3.a Gráfico dimensionado de la articulación # 3

2.4.4 ARTICULACION # 4

- Longitud del enlace: 8.00 plg.

- Peso del enlace: 3.50 lb.

- Peso del Reductor secundario 2.50 lb.

- Peso del Reductor primario 1.50 lb.

Igualmente el dimensionamiento del motor para esta articulación se lo obtendrá en el análisis estático del sistema.

En esta articulación se da la particularidad que el enlace esta unido tanto a su propio reductor como al reductor correspondiente a la articulación # 3.

En la figua 2.4.4.a no se puede observar claramente la articulación # 4 del brazo, pero se muestra la estructura prácticamente completa, incluyendo la base del brazo mecánico. La base es una estructura metálica en forma de paralelepípedo con las siguientes medidas:

- Largo: 19.70 plg.

- Ancho: 15.80 plg.

- Profundidad: 6.0 plg.

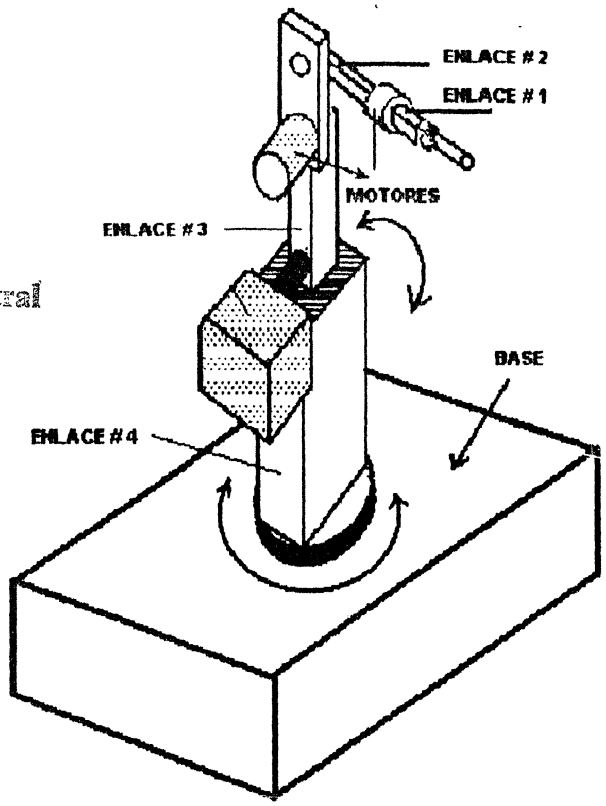


Fig. 2.4.4.a Bosquejo General de la Estructura del Brazo

2.5 ANALISIS ESTATICO

2.5.1 JUNTURA # 1

Esta articulación esta asociada con el manejo del efector final, se utilizará un motor AC que será el encargado de hacer girar dicho efector a la posición deseada De todas las posiciones espaciales posibles del efector final, la que presentará mayor oposición al motor será aquella cuando el brazo este totalmente extendido

horizontalmente y, el eje longitudinal del efector se encuentre en ángulo recto con el eje longitudinal común de los enlaces dos y tres , esto se ilustra en la figura 2.5.1.a.

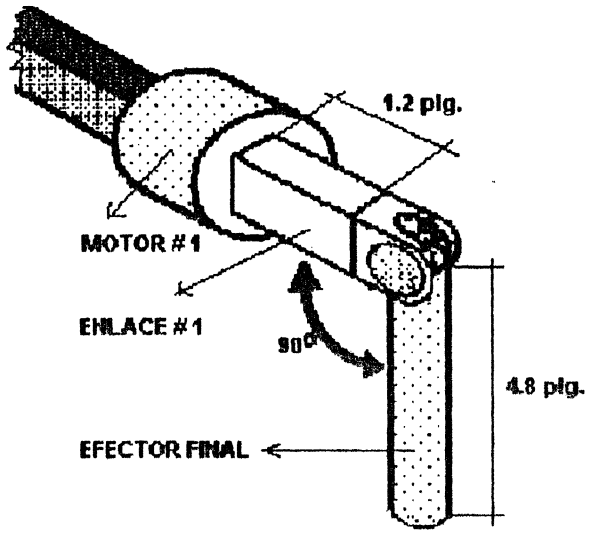


Fig. 2.5.1.a Posición Crítica del efector final

Se puede considerar la masa del efector final como uniformemente distribuida a lo largo de su estructura, esta hipótesis es válida en cuanto dicho efector en este caso es un motor DC que consta de imanes y devanado uniforme. Por tanto el torque mínimo del motor #1 necesario para mantener el sistema efector final-juntura #1 en equilibrio estático deberá ser:

$$T_{M\,1} = W_{EF}\,(L1 + L_{EF} / 2)$$

donde:

T_{M1} es el torque mínimo desarrollado por el motor # 1

W_{EF} es el peso máximo del efector final (lb)

L_{EF} es la longitud del efector final

L_1 es la longitud del enlace #1 (plg), por lo tanto:

T_{M1}=(1lb)(1.2plg + 4.8/ 2 plg)=3.6 lb-plg

Entonces el motor # 1 deberá tener un torque mínimo de 3.6lb-plg para hacer girar al efector final a la posición deseada.

En base a este análisis se selecciona un motor AC cuyas características se detallan más adelante. Para el análisis de la siguiente articulación se debe considerar el peso de dicho motor, el mismo que es de 1.5 lb.

2.5.2 JUNTURA # 2

El motor asociado a esta articulación será el motor # 2, y estará encargado de hacer rotar al conjunto enlace # 2 - efector final (con sus componentes asociados), con respecto al enlace # 3.

El torque de oposición máximo presentado a este motor ocurre cuando el eje longitudinal del enlace # 2 y el eje longitudinal del efector final se encuentran alineados y en posición horizontal, cabe resaltar que en esta posición se da el máximo alcance del brazo. Esto se ilustra en la figura 2.5.2.a

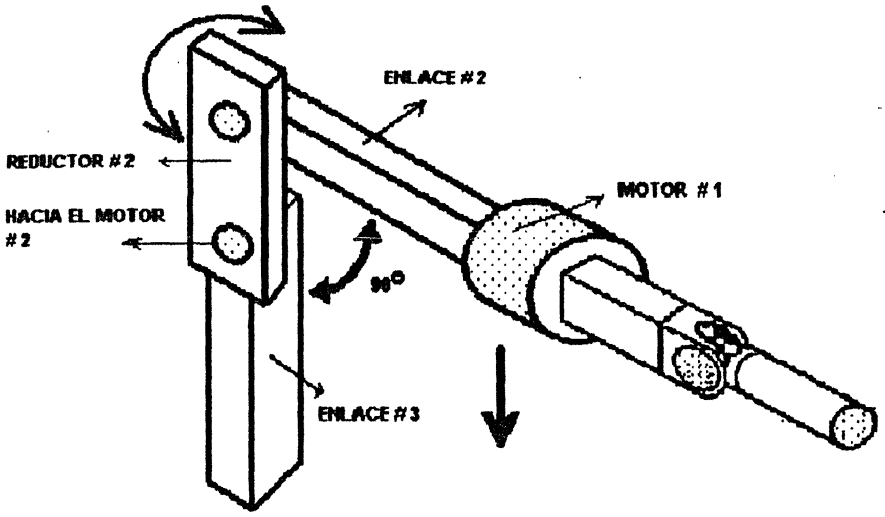


Fig.2.5.2.a Posición Crítica del enlace # 2

Por ser el enlace # 2 (L2), una estructura hueca regular de espesor constante podemos asumir que su masa esta distribuida uniformemente a lo largo de ella , por tanto se puede considerar que su centro de gravedad esta ubicado en la mitad de dicha estructura. Entonces, el motor # 2 y el conjunto de engranajes asociados con él deberán desarrollar un torque suficiente para mantener al sistema equilibrado estáticamente, y además

suficiente pata lograr mover la estructura. Este torque viene dado por :

$T_2 = W_{L2}(L_2/2) + W_{EF}(L_2+L_1+L_{M1}+L_{EF}/2) + W_{M1}(L_2 +L_{M1}/2)$

donde,

$W_{L2} = 0.5$ lb peso del enlace # 2

$L_2 = 6.0$ plg longitud enlace # 2

$L_{M1} = 2.0$ plg longitud motor # 1

$W_{M1} = 1.5$ lb peso motor # 1

Así,

$T2 = (0.5lb)(6plg/2) + 1lb(6+1.2+4.8/2+2)plg+1.5lb(6+2/2)plg$

$T2 = 1.5$ lb-plg + 11.6 lb-plg + 10.5 lb-plg = 23.6 lb-plg

Este es un torque demasiado alto para ser manejado por un motor razonablemente pequeño, acoplado directamente a la juntura # 2 por tanto necesitamos de un conjunto de engranajes que multiplique el torque del motor pero que no reduzcan demasiado la velocidad del mismo. Se cuenta con un reductor de velocidad que no es más que un conjunto de engranajes acoplados formando una estructura maciza, cuya razón de transmisión es de 1:150 , suficiente para permitirnos escoger un motor de pequeñas

dimensiones, entonces este motor deberá desarrollar un torque mínimo de :

$$TM2 = 23.6 \text{ lb-plg}(1 / 150) = 0.15 \text{ lb-plg}$$

Posteriormente se describen lasa características del motor seleccionado, cuyo peso es de 1.25 lb.

2.5.3 JUNTURA # 3

El motor asociado a esta articulación será el motor # 3, y estará encargado de hacer rotar al conjunto, enlace # 3 + enlace # 2 + efector final (y a los componentes asociados con cada uno de ellos), con respecto al enlace # 4.

El torque de oposición máximo presentado a este motor ocurre cuando los ejes longitudinales de los enlaces dos y tres y el eje longitudinal del efector final se encuentran todos alineados y en posición horizontal, cabe resaltar que en esta posición se da el máximo alcance del brazo. Esto se ilustra en la figura 2.5.3.a

Entonces, el motor # 3 y el conjunto de engranajes asociados con él deberán desarrollar un torque suficiente para mantener al

sistema equilibrado estáticamente, y además suficiente para lograr mover la estructura. Este torque viene dado por :

$$T3 = WL3(L3/2) + (WM2+WR)L3 + (L3+LR)(WL2+WM1+WEF)+T2$$

Aquí:

WM2 = 1.25 lb peso del motor # 2

WR = 1.0 lb peso del reductor ubicado en la juntura dos

L3 = 10.0 plg longitud de enlace # 3

LR = 1,2 plg separación entre los enlaces dos y tres

$$T3 = 0.75\ lb(10/2plg) + (10\ plg)(1.25+1)lb+(10+1.2)(0.5+1.5+1)lb\text{-}plg + 23.6\ lb\text{-}plg$$

$$T3 = 3.75\ lb\text{-}plg + 22.5\ lb\text{-}plg + 33.6\ lb\text{-}plg + 23.6\ lb\text{-}plg = 83.45\ lb\text{-}plg$$

Para poder satisfacer la necesidad de torque de esta articulación se requiere de un motor bastante grande, es por esta razón que se ha procedido a utilizar un sistema reductor que proporciona una relación de aumento de torque de 1: 144. Con la ayuda de este reductor se facilita la selección de un motor considerablemente

pequeño y de acuerdo a la estructura del brazo mecánico, cuyo peso es de 1.75 lb.

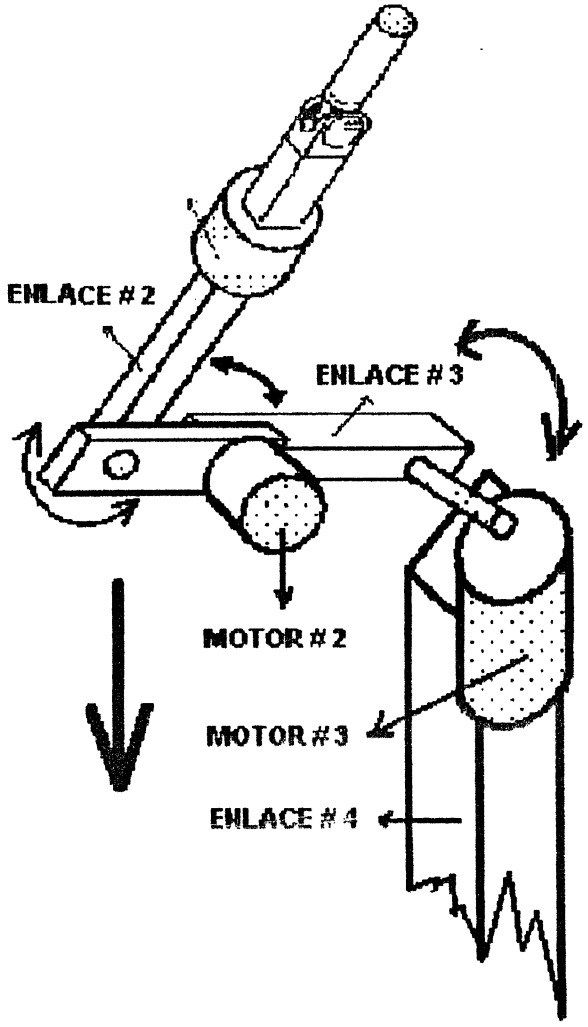


Fig. 2.5.3.a Interacción entre los enlaces 3 y 4

2.5.4 JUNTURA # 4

Sobre esta articulación actuará la mayor del peso de la estructura del brazo mecánico, en vista de esto se ha creído conveniente

utilizar un sistema mecánico que tienda a anular las fuerzas ejercidas por el brazo sobre la base. El sistema más efectivo para cumplir con este propósito resulta ser aquel que utiliza dos rodamientos dispuestos concéntricamente, abrazando el eje que conecta al brazo con la base del mismo. La ubicación de los rodamientos es tal que al encontrarse separados uno del otro logran equilibrar las fuerzas ejercidas sobre el eje del brazo, permitiendo de esta manera utilizar un reductor simple y un motor relativamente pequeño para lograr el movimiento de todo el brazo mecánico. El motor # 4 será el encargado de hacer rotar esta juntura y deberá cumplir con las siguientes especificaciones:

- Deberá ser de 1.8°/paso (permitirá una buena resolución)

- Deberá acelerarse hasta una velocidad de 50 pasos/seg. en 0.1seg

Para simplificar el cálculo del motor # 4 se asumirá que la estructura es equivalente a un cilindro macizo con un radio de 2plg y un peso total de WT=14lb. Su momento de inercia viene dado por:

$$I = (W_T R^2)/2$$

$$I = (14)(2^2)/2 = 27{,}12 \text{ lb-pl}^{s^2}$$

Entonces el torque del motor deberá ser :

$T=I\alpha$ donde α es la aceleración del sistema

$T=(27{,}12\ lb\text{-}plg^2)(50\ pasos/seg.\div 0.1seg)(1{,}8\ \pi/180)(1/24)$

$T= 17.75$ oz-plg.

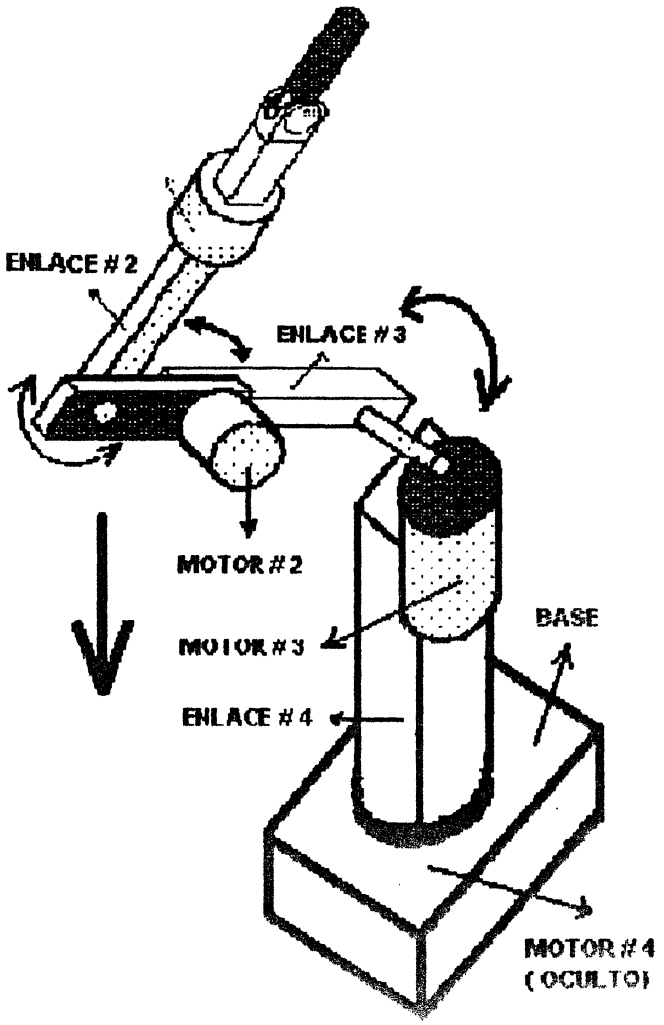


Fig. 2.5.4.a Brazo acoplado a su base.

En el análisis estático desarrollado para cada una de las articulaciones no se ha dimensionado las pérdidas por fricción y las pérdidas inherentes de los engranajes, por esta razón se han utilizado reductores

sobredimensionados de tal manera que la selección de los motores no sea errónea. Igualmente cabe anotar que en los cálculos de torque para las articulaciones 2 y 3, se ha agregado al peso de los enlaces el peso de sus respectivas acoplaciones, es por esta razón que en los cálculos aparecen valores de peso de 0.5 lb y de 0.75 lb para los enlaces 2 y 3, en lugar de 0.30 lb y 0.45 lb respectivamente, que son los valores del peso de cada enlace como se describe en su respectivo dimensionamiento.

2.6 SELECCION DE MATERIALES

En esta sección se detalla el tipo, calidad y cantidad de los diferentes materiales utilizados para la construcción del brazo mecánico, y además se justifica su selección. Se describen aquellas partes y componentes analizados en las secciones anteriores de este capítulo, dejando la descripción de las partes eléctricas para el capítulo tres.

2.6.1 ENLACES

Del análisis teórico expuesto en el capítulo uno se, desprende que el material ideal para el enlace de un brazo mecánico debe ser ligero en peso y con un alto grado de rigidez, de tal manera que soporte el peso de los motores y del resto de la estructura. En vista de esto se ha hecho la siguiente selección:

En el enlace # 1 se utiliza tubo cuadrado de hierro de estructura hueca de 1.0 plg., acoplado al motor a través de un dado cilíndrico de hierro dulce. Se utiliza hierro por la rigidez que se necesita en este enlace y además por que su longitud es bastante pequeña, lo cual no afecta mayormente en el peso total de la articulación.

Para los enlaces 2 y 3 se utiliza tubo cuadrado de aluminio de estructura hueca, de 1.0 plg de espesor. En este caso se opta por el aluminio hueco debido a la considerable longitud de los enlaces, la misma que hace imposible el uso de un tipo de material más pesado, pues, esto implicaría el uso de motores más grandes y la elevación en los costos.

En el enlace # 4 se utiliza un material mucho más fuerte que el utilizado para los otros enlaces, puesto que este enlace soporta todo el peso de la estructura colgante del brazo y por tal motivo debe estar obligatoriamente constituido de un material de gran rigidez. Se utiliza tubo cuadrado de hierro dulce, de estructura hueca y de 2.0 plg de espesor.

Para la acoplación entre los enlaces y sus respectivos reductores, al igual que en el enlace # 1, se utilizan dados de hierro dulce, los

mismos que ofrecen una buena rigidez y gran facilidad de acoplación.

Para la construcción de la base se ha utilizado ángulo de hierro de 0.5 plg. Como esta parte es fija, no ha sido crítica la selección del tipo de material, además este permite una fácil manipulación.

2.6.2 REDUCTORES Y ENGRANAJES

Para la articulación # 1 no ha sido necesaria la utilización de engranajes o reductores extras, puesto que el motor seleccionado para esta articulación tiene incluido internamente un reductor cuya relación de reducción es de 1: 1800, la misma que es suficiente y sobredimensionada para cubrir con la necesidad de torque en esta articulación.

En la articulación # 2 se utiliza un reductor de 10 plg^2 de superficie por 0.5 plg de espesor, constituido externamente por un material producto de una aleación de aluminio y estaño, lo cual hace que el peso de este reductor sea muy bajo. La relación de este reductor es de 1 : 150.

Para la articulación # 3, además de utilizar un reductor, se utiliza también una pareja de engranajes para incrementar aún más la relación de reducción.

El reductor tiene una superficie de 15.0 plg^2 por 3.0 plg de espesor, internamente utiliza un eje sinfín acoplado a un engranaje circular como se ilustra en la figura 2.6.2.a.

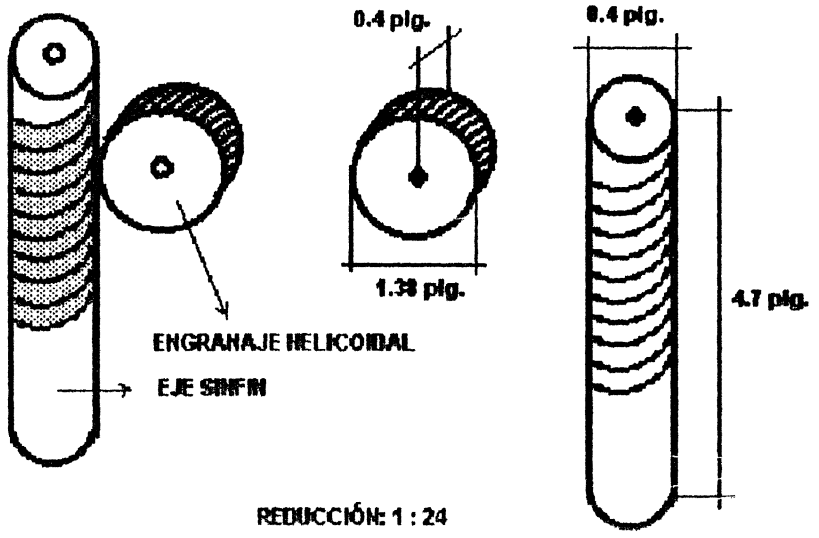


Fig. 2.6.2.a Estructura interna del Reductor

La relación de reducción del reductor es de 1 : 24, en vista de que esta relación no es suficiente para cubrir la necesidad de torque de esta articulación, se acopla una pareja de engranajes extras con una relación de 1 : 6. El material del cual están constituidos estos engranajes es el hierro dulce, mientras el engranaje del reductor

es de bronce fosfórico y el eje sinfín de acero. Más detalles se observan en la figura 2.6.2.b

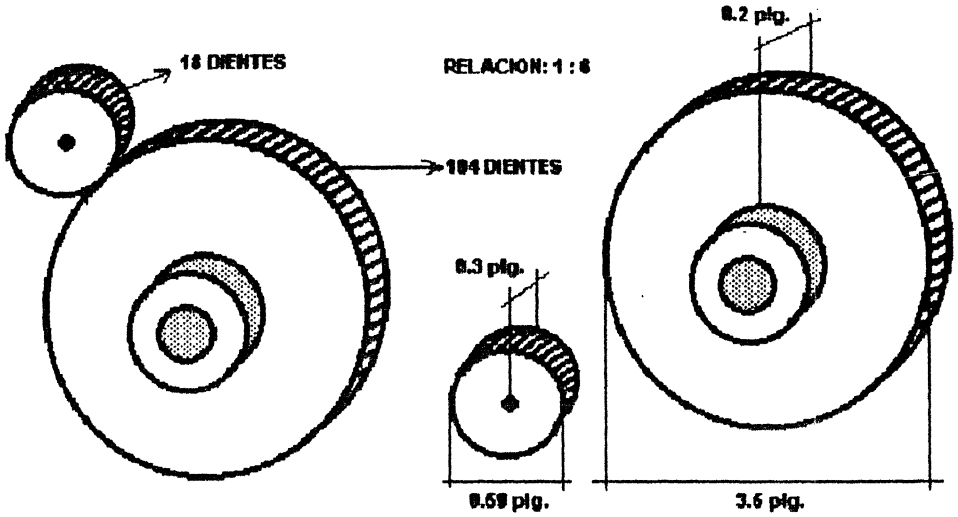


Fig.2.6.2.b Juego de Engranajes de la articulación # 3.

Igualmente en la base, para lograr el movimiento de todo el brazo, se utiliza una pareja de engranajes con una relación de reducción de 1 : 4.5, ambos constituidos de hierro dulce. Algunas características extras se pueden observar en la figura 2.6.2.c.

La descripción del efector final no se hace en esta sección, puesto se en este mismo capítulo de dedica un punto para detallarlo completamente.

Dificultades y más acotaciones sobre la selección y obtención de materiales son descritas en la sección de observaciones.

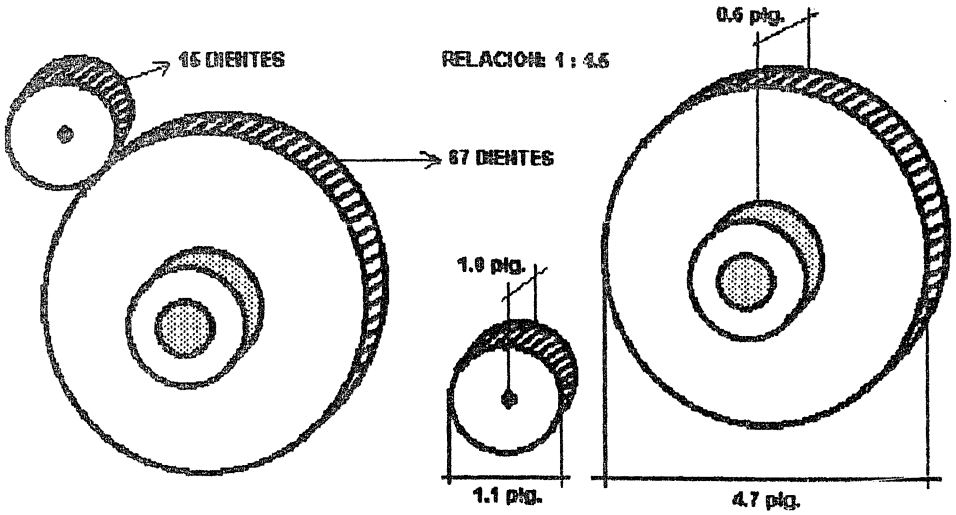


Fig. 2.6.2.c Engranajes de la articulación # 4 (Base).

2.7 EFECTOR FINAL

Para este brazo mecánico se han seleccionado dos tipos de efector final, con la finalidad de utilizarlos alternativamente de acuerdo a la necesidad del trabajo a desarrollar. El un efector final es una herramienta eléctrica de uso múltiple y el otro efector es un electroimán.

2.7.1 HERRAMIENTA ELECTRICA

Entre las principales características de esta herramienta tenemos las siguientes.

- Posee un motor DC de 12 voltios.

- Tiene un reductor incluido internamente.

- Cuenta con un acoplador múltiple que le permite adaptarse a un gran número de herramientas tales como: destornilladores de varias formas y medidas, dados para manejo de pernos o tornillos de cabeza hexagonal de diferentes tamaños, brocas, avellanadoras, entre otras.

- Tiene la capacidad de girar en dos sentidos.

- Su posición puede ser variada a través del programa de control y también manualmente.

- El peso de este efector es de 1.0 lb., el cual es aceptable para ser manejado por el brazo mecánico.

- Sus características eléctricas y su circuito de manejo serán descritos más adelante en el capítulo del diseño eléctrico.

- Es fácilmente desmontable y ajustable, además su instalación eléctrica consta de dos cables simplemente.

En la figura 2.7.1.a se muestra un gráfico ilustrativo de este efector final.

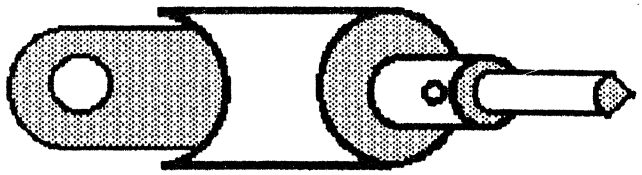


Fig. 2.7.1.a Herramienta de aplicación múltiple.

2.7.2 ELECTROIMAN

Como alternativa al uso de la herramienta múltiple como efector final, se presenta la utilización de un electroimán, el mismo que le permitirá al brazo mecánico transportar objetos metálicos (magnetizables) de un lugar a otro. Entre sus principales características citamos las siguientes:

- Este efector requiere una tensión de 12 voltios DC.

- Su tamaño y peso son similares a los presentados por el efector anteriormente descrito.

- Tiene la capacidad de coger objetos de distinta forma y tamaño, siempre que el peso de estos estén dentro del rango de capacidad del efector.

- Ofrece gran facilidad de manejo, puesto que para su control no se requiere de mucha precisión como la requerida para el control de la herramienta múltiple.

En la figura 2.7.2.a se muestra un bosquejo de la apariencia externa de este efector final.

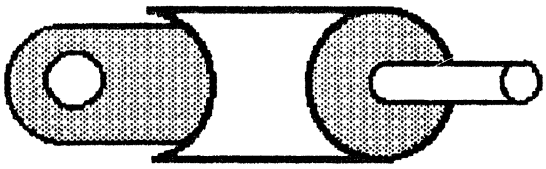


Fig 2.7.2.a Electroimán como efector final.

2.8 ENSAMBLAJE MECANICO

Existen varios aspectos importantes sobre el ensamblaje del brazo mecánico, algunos de ellas se mencionan a continuación.

- Cada una de las articulaciones es totalmente desmontable, al igual que cada uno de los componentes de las articulaciones tales como: enlaces, motores, reductores y acopladores. Esto facilita el armado y desarmado del brazo para su correspondiente transportación ó mantenimiento.

La base está diseñada de tal forma que tiene la capacidad de sostener a toda la estructura del brazo, además permite el alojamiento de las fuentes de poder y de los circuitos eléctricos manejadores del brazo. Las tapas de esta base también son desmontables para facilitar la manipulación de sus componentes internos.

Para facilitar la conexión eléctrica entre la parte externa del brazo y la circuitería interna ubicada en la caja de la base, se ha diseñado un eje hueco oculto, el mismo que une el enlace cuatro con la base del brazo. Este eje permite el libre movimiento de la estructura externa, sin que esto produzca torceduras o remordeduras en los cables eléctricos.

El brazo tiene ciertos movimientos bruscos ó erráticos, debido a la fuga mecánica existente en las diferentes acoplaciones de reductores, engranajes y enlaces.

2.9 OBSERVACIONES

En esta sección se anotan ciertas experiencias suscitadas durante la construcción del brazo mecánico.

- En el montaje de la articulación # 3 se utilizó inicialmente varios tipos de reductores mecánicos, los mismos que fueron desechados. debido a

que su estructura resultó ser muy frágil, aunque la relación de reducción era satisfactoria.

Debido a los torques ejercidos por cada una de las partes colgantes del brazo, fue necesario utilizar reductores con una relación alta de reducción, esto dio como resultado que el movimiento de las articulaciones resultó ser relativamente lento. Al querer elevar dicha velocidad se notó que resultaba imposible debido a que los motores perdían torque con el aumento de velocidad. El cambiar los motores utilizados por motores más potentes implicaba un redimensionamiento de toda la estructura del brazo, y además el hecho de perder la línea del modelo base. Cabe recordar que el proyecto planteado como tema de esta tesis es un prototipo de un brazo mecánico industrial.

Algunas partes y componentes utilizados en el montaje mecánico del brazo han tenido que ser extraídas de equipos viejos, debido a la no existencia de estos en el mercado. En otros casos, como por ejemplo, para la obtención del reductor utilizado en la articulación # 3, se recurrió a desarmar una pulidora eléctrica en estado de funcionamiento.

CAPITULO III

DISEÑO DEL SISTEMA ELECTRICO

3.1 OBJETIVO

Una vez construido y ensamblado el sistema mecánico del brazo robótico, es necesario determinar un nexo de conexión de esta parte con el sistema de control. El sistema controlador del brazo está constituido básicamente por un computador, y la conexión de este con el sistema mecánico no puede ser directa, se necesita implementar una interfase de potencia a través de la cual pueda controlarse sin problema alguno al brazo mecánico. Con el fin de analizar y detallar los diferentes circuitos y elementos eléctricos utilizados en las interfaces mencionadas, se ha creído conveniente dedicar este capítulo para cumplir con dicho objetivo.

3.2 INTRODUCCION

Este capítulo está dedicado a detallar todo el sistema eléctrico del brazo robótico. Entre otros aspectos se describe a la Interface Periférica Programable, la función que desempeña y la justificación de su utilización.

Con los resultados obtenidos del análisis mecánico desarrollado en el capítulo dos, se expone en esta sección un detalle de los motores seleccionados, incluyendo sus características, condiciones de funcionamiento, circuitos manejadores y secuencias de control. Además se describen circuitos y elementos de apoyo, tales como fuentes de poder, sensores, tarjetas de circuito impreso, entre otros.

3.3 INTERFACES COMPUTADOR - MOTORES

El control del brazo mecánico se lo hace a través de un computador personal, el mismo que envía las señales de control por el puerto paralelo, dichas señales son receptadas por una interfase digital la que a su vez distribuye las señales hacia los diferentes circuitos de control de los motores. El brazo cuenta con un sistema de sensores que envían señales hacia la interfase digital para luego ser leídas por el computador a través del puerto paralelo. Esta descripción de fimcionamiento es bosquejada en el diagrama de bloques general mostrado en la figura 3.3.a.

En esta sección se detalla ampliamente cada una de las partes mostradas en el diagrama de bloques, así entonces se presenta un diagrama de la arquitectura de un computador, la descripción del manejo de la interfase periférica programable (8255) a través del puerto paralelo. Además se hace el análisis del circuito de control y se muestra las pistas de dicho circuito.

3.3.1 EL COMPUTADOR

En esta sección se presenta una descripción general del computador orientada a su utilización en esta tesis.

DIAGRAMA DE BLOQUES GENERAL

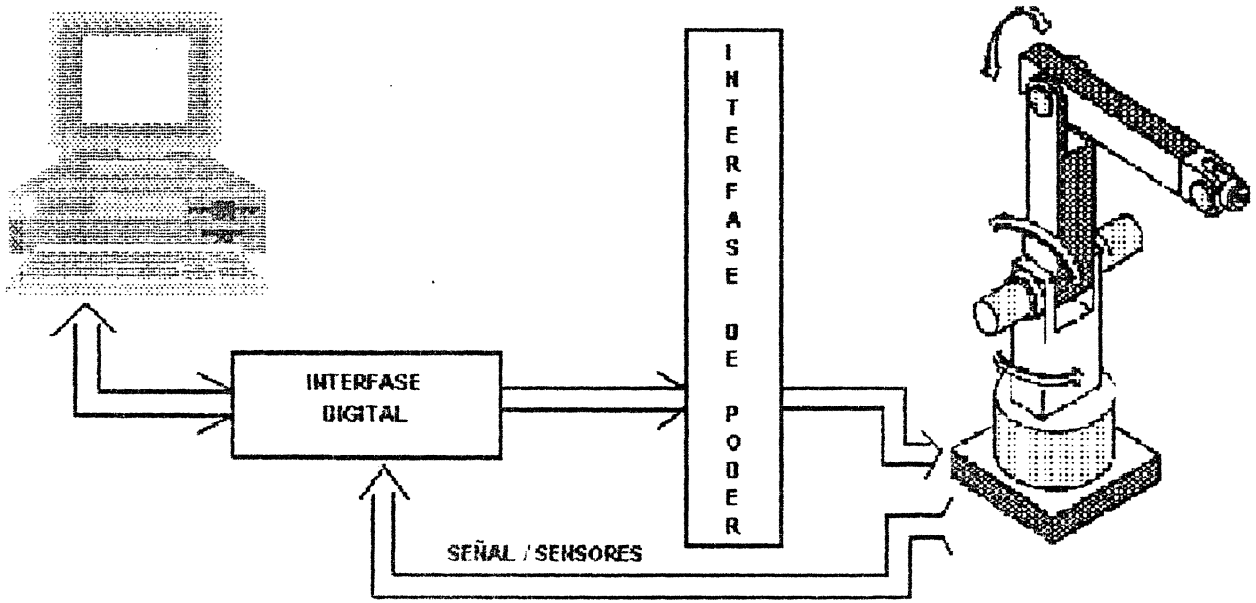


Fig. 3.3.a Diagrama de bloques del brazo mecánico

La definición más básica de computador reconoce que es un sistema o dispositivo capaz de aceptar información, aplicarle un proceso determinado a esta información, ejecutar acciones de acuerdo a esta información procesada y, finalmente presentar de por algún medio los resultados de estos procesos. La función esencial de cualquier computador es tomar un conjunto de datos y realizar un conjunto de secuencias de acuerdo a estos datos. Se puede encontrar dos clasificaciones principales para lo anterior, la primera tiene que ver con el procesamiento de datos que han sido ingresados al computador por algún medio, por ejemplo teclado, modem, disquetes, etc. y utilizados por el usuario posteriormente para formar por ejemplo una base de datos o para realizar algún cálculo. La segunda clasificación tiene que ver con lo que generalmente se denomina procesamiento en tiempo real, aquí los datos son ingresados e instantáneamente se analizan y , usualmente se genera una acción apropiada en respuesta a estos análisis. El brazo mecánico utiliza básicamente un software controlador que responde en tiempo real de acuerdo a las entradas desde el teclado por parte del operador.

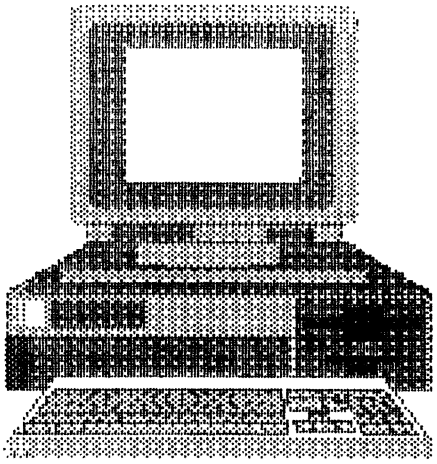


Fig.3.3.1.a Computador Personal

Los componentes básicos de un computador son : los bancos de memoria, la unidad central de proceso (CPU), unidad de almacenamiento, y los dispositivos de entrada salida (E/S). En los bancos de memoria se almacenan los códigos de las instrucciones que debe ejecutar el CPU, generalmente estos códigos pertenecen a programas de utilidad o de sistema operativo. Estos bancos de memoria son comúnmente chips de estado sólido, cuya densidad de almacenamiento es continuamente mejorada por los fabricantes. El CPU es la unidad encargada de la sincronización, también tiene unidades especiales de almacenamiento, además de estar encargada de la ejecución de las instrucciones almacenadas en la memoria, el componente más

importante del CPU es el microprocesador. Normalmente el poder de un CPU esta determinado por la cantidad de instrucciones por segundo que puede ejecutar, este parámetro es continuamente mejorado con el desarrollo de nuevas tecnologías, por ejemplo el microprocesador 8086 en su versión básica desarrollado por INTEL en el año 1978 ejecuta 2.5 millones de instrucciones por segundo (MIPS), en cambio los procesadores Pentium sacados a la luz en 1995 ejecutan hasta 100 MIPS. Otro parámetro utilizado para medir el poder de un CPU es el conjunto de instrucciones que puede manejar así como la manera de manejarlas . Los elementos internos del microprocesador pueden caer en cualquiera de las siguientes tres categorías: registros, elementos de instrucción y la unidad aritmética lógica (ALU).

Los registros son elementos individuales que están dentro del microprocesador e interaccionan estrechamente con la información almacenada en la memoria, están clasificados de acuerdo a la función que realizan :

1. Registros acumuladores, usados para escribir datos con los cuales se realizará alguna operación aritmética o lógica.

normalmente se asocia al registro acumulador un registro especial de 1 bit llamado de enlace o acarreo, es usado para ejecutar

operaciones de rotación de los datos almacenados en el acumulador.

2. Registro de estado, este registro esta compuesto de cinco flip-flops especiales:

a. El flip-flop de acarreo mencionado antes

b. El flip-flop negativo, el cual se establece en nivel lógico alto cuando el acumulador tiene un valor negativo.

c. El flip-flop o bandera cero que se establece en alto cuando el contenido de una operación da un resultado cero en el acumulador.

d. El flip-flop de sobreflujo, se activa cuando el resulatado de una operación es mayor que el máximo valor posible de representar con el número de bits del acumulador.

e. El flip-flop o bandera de interrupciones, el microprocesador atenderá una interrupción solo cuando este bit este activado.

4. Registro decodificador de instrucciones, aquí se almacena el código de operación de una instrucción .

5. Registro contador de programa, aquí se almacena la dirección de la siguiente instrucción que debe ser ejecutada, esta dirección también puede ser modificada por el programa, permitiendo así la ejecución de instrucciones localizadas en otras partes del programa.

La sección de E/S, provee al computador de una interfase con el mundo externo, permitiendo la conexión de dispositivos tales como teclados, monitores, mouses, joisticks. Esta sección es fundamental para la aplicación de las computadoras en la robótica puesto que le permite comunicarse con el exterior para recibir o enviar datos. Las unidades de almacenamiento sirven para almacenar grandes cantidades de información, la conforman principalmente los discos duros.

En la figura 3.3.1.b se muestra un diagrama de la arquitectura interna del computador, diagrama general válido para la gran mayoría de los computadores existentes en el mercado actualmente.

Debe conocerse que el brazo mecánico objeto de esta tesis puede ser controlado con cualquier computador con la tecnología de IBM, independientemente del tipo de microprocesador que la máquina posea.

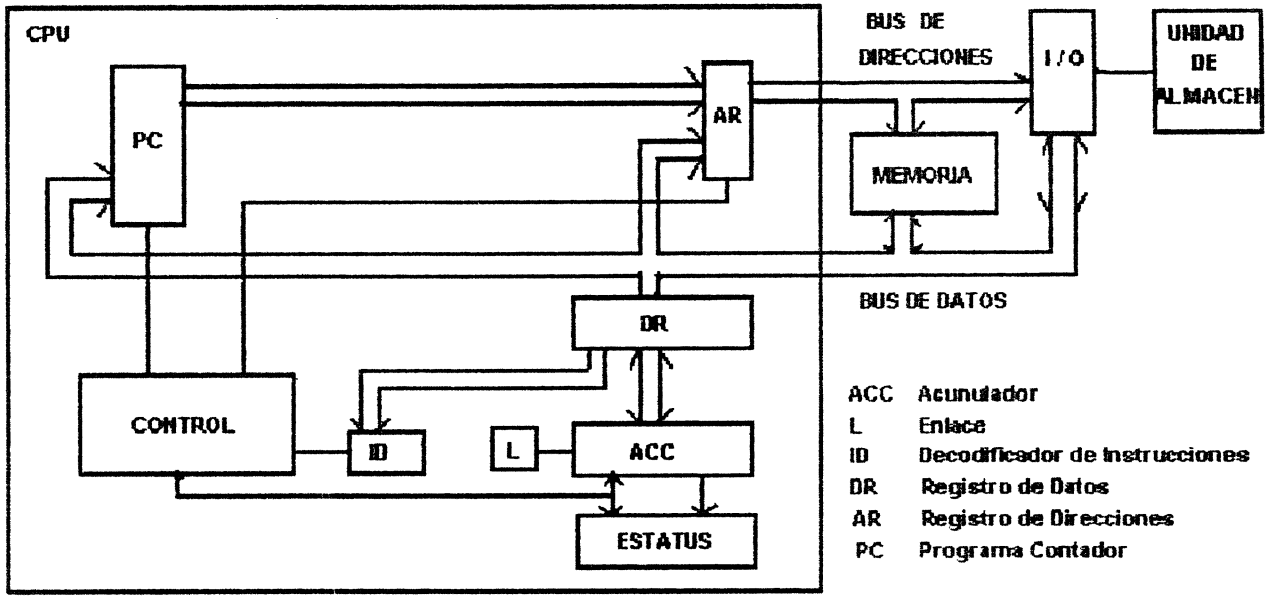


Fig.3.3.1.b Arquitectura Básica de un Computador

3.3.2 INTERFASE PERIFERICA PROGRAMABLE (PPI 8255)

El corazón de la interfase entre el puerto paralelo del computador y los actuadores eléctricos que manejan cada una de las articulaciones del brazo mecánico es la Interfase Periférica Programable 8255 (en adelante PPI 8255), el cual es un circuito integrado de 40 pines desarrollado inicialmente por INTEL. Su función principal dentro del circuito puede ser vista como la de decodificar las señales provenientes del puerto paralelo del computador, dichas señales son bytes generados por el programa manejador, que llevan las secuencias adecuadas de pulsos, necesarios para el correcto funcionamiento de los motores asociados a cada una de las articulaciones.

De acuerdo al apéndice D, la PPI 8255 consta esencialmente de tres puertos de E/S de ocho bits cada uno, estos son conocidos como puerto A, puerto B, y puerto C; pueden ser programados como grupos independientes de 12 terminales llamados grupo A y grupo B ó como puertos independientes de acuerdo a la palabra de control y dependiendo de las necesidades especificadas. Igualmente este integrado cuenta con una patilla de selección llamada Chip Select ó CS/ (la barrita arriba significa que es activa en nivel bajo), este pin es decodificado de acuerdo a la dirección que tendrá dentro del mapa de E/S, en este caso el

pin CS/ estará conectado siempre a cero voltios , es decir la PPI estará siempre habilitada; esto no representa problema puesto que el integrado empezará a recibir tanto los bytes de control como de datos únicamente cuando dentro del programa principal se direccione al puerto paralelo activo del computador. La selección de los registros de operación ó de control se logra por medio de los pines A1 y A0, para mayor comodidad se muestra en la tabla I la asignación de valores para A1 y A0 de acuerdo al registro que se desee accesar. En este caso el control de estos dos pines se lo hace por medio del programa principal, esto es necesario porque se maneja a la PPI con las señales de control proporcionadas por el puerto paralelo del computador.

A1	A0	FUNCION
0	0	Puerto A
0	1	Puerto B
1	0	Puerto C
1	1	Registro de Comando

Tabla I. Asignación de puertos E/S para la PPI 8255.

Otra señal fundamental para el correcto funcionamiento de la PPI es WR/, esta señal debe estar en nivel bajo cuando se escriba un byte en la PPI, dicho byte pude estar destinado al registro de control ó al de datos, como se detalla a continuación

Antes de tratar de escribir datos o palabras de control en la PPI, es necesario generar las señales A1, A0 y WR/ para que esta funcione adecuadamente. Esto no fuera necesario si se trabajára directamente con las señales del microprocesador puesto que este las genera automáticamente en cada ciclo de canal por tanto no tuviera sentido generarlas, a continuación se discute la simulación de estas señales. En la figura 3.3.2.a se muestra el diagrama de tiempo de las señales A1, A0, WR/ , en la manera que deben ser producidas por el programa .

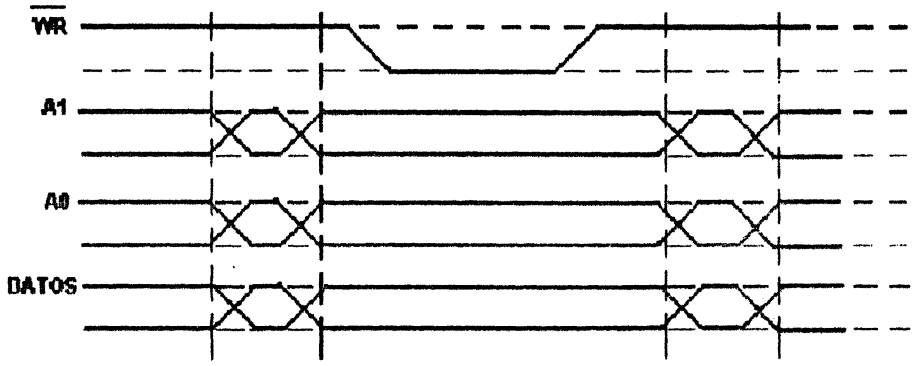


Fig 3.3.2.a Diagrama de tiempo para señales de control.

Se utilizan las señales de control del puerto paralelo que corresponden a la dirección 37A H, que tienen la siguiente relación con los pines de control del 8255.

INTERFAZ CENTRONICS	PIN DB25	PPI 8255	PIN 8255
STROBE	1	WR	36
AUTO FD	14	A1	9
INIT	16	A0	8

Tabla II. Pines de control de la PPI 8255.

En el programa se generan cada una de estas señales mediante procedimientos que necesitan ser llamados cada vez que se quiera escribir en la PPI, ya sea para programarla o para sacar datos.

Para este caso en particular se programa a la PPI en modo cero-salida lo que la habilita para que se pueda escribir las secuencias de pulsos necesarios para los motores. Se utiliza al puerto A para controlar un par de motores y al puerto B para controlar a los otros dos, en la figura 3.3.2.b se muestra la palabra de control necesaria para lograr esto.

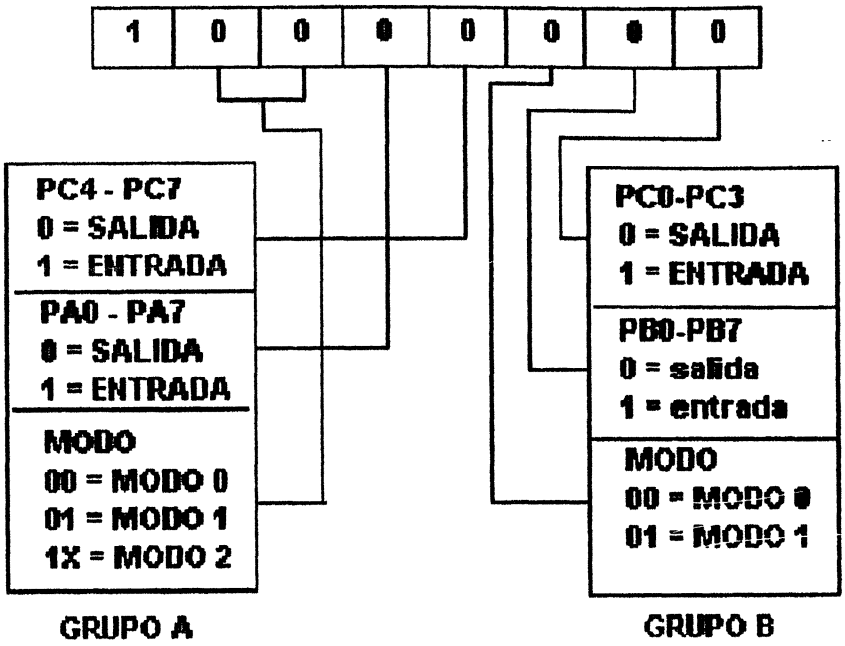


Fig. 3.3.2.b Palabra de comando para el 8255.

En el cuadro se pueden observar las distintas alternativas para programar el funcionamiento de la PPI 8255, con sus diferentes modos de operación y sus dos distintos grupos de trabajo.

Por los ocho pines de datos de la PPI pasaran tanto los bytes de control como los bytes de operación para cada motor. La correspondencia con los pines de la interfaz CENTRONICS es la siguiente.

INTERFAZ CENTRONICS	PIN DB25	PPI 8255	PIN 8255
D0	2	D0	34
D1	3	D1	33
D2	4	D2	32
D3	5	D3	31
D4	6	D4	30
D5	7	D5	29
D6	8	D6	28
D7	9	D7	27

Tabla III. Correspondencia de señiales entre el puerto paralelo y el 8255.

Entonces, para escribir en la PPI un byte de datos el programa de control debe hacer lo siguiente:

1) Llevar a alto a la señal WR/, establecer los valores deseados de A1 y A0 de acuerdo a si queremos escribir dato ó control.

2) Direccionar el puerto de datos (dirección 378 H)

3) Sacar byte de datos o control

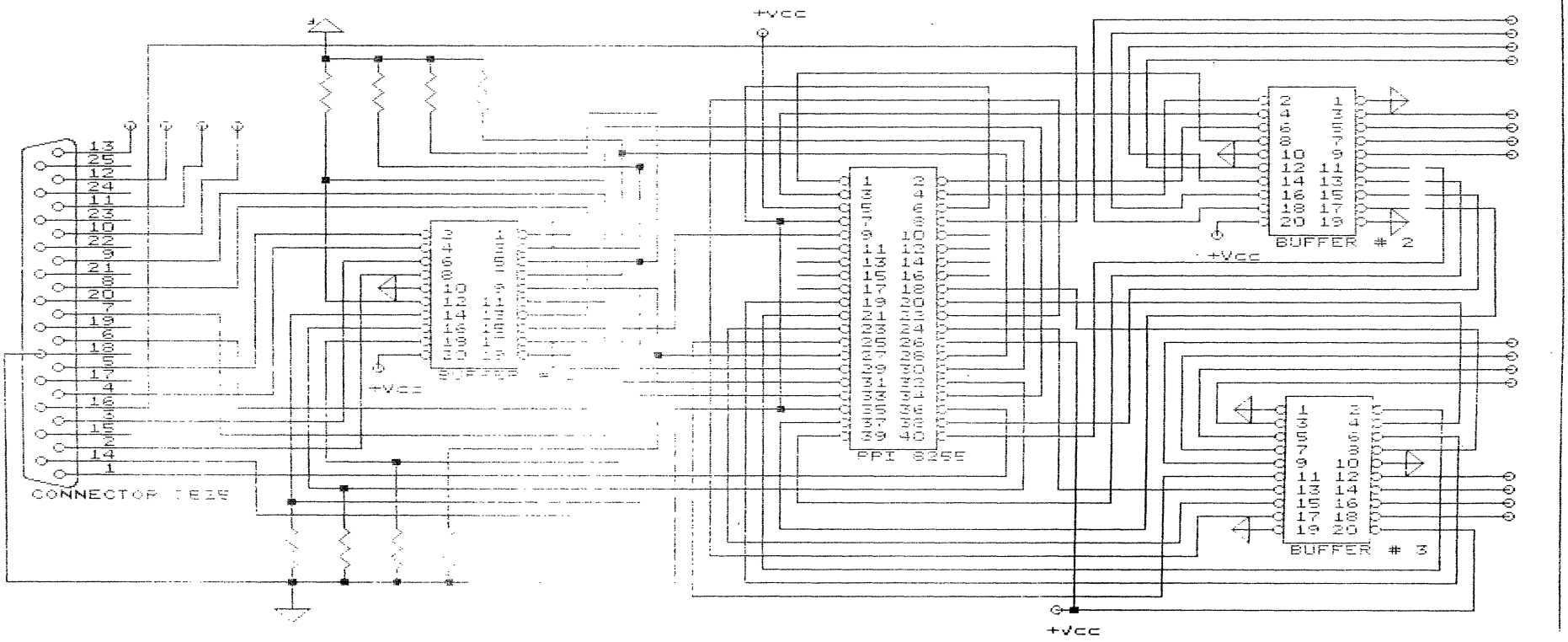
4) Llevar a nivel bajo a la señal WR/, A1 y A0 no cambian.

5) Retardo

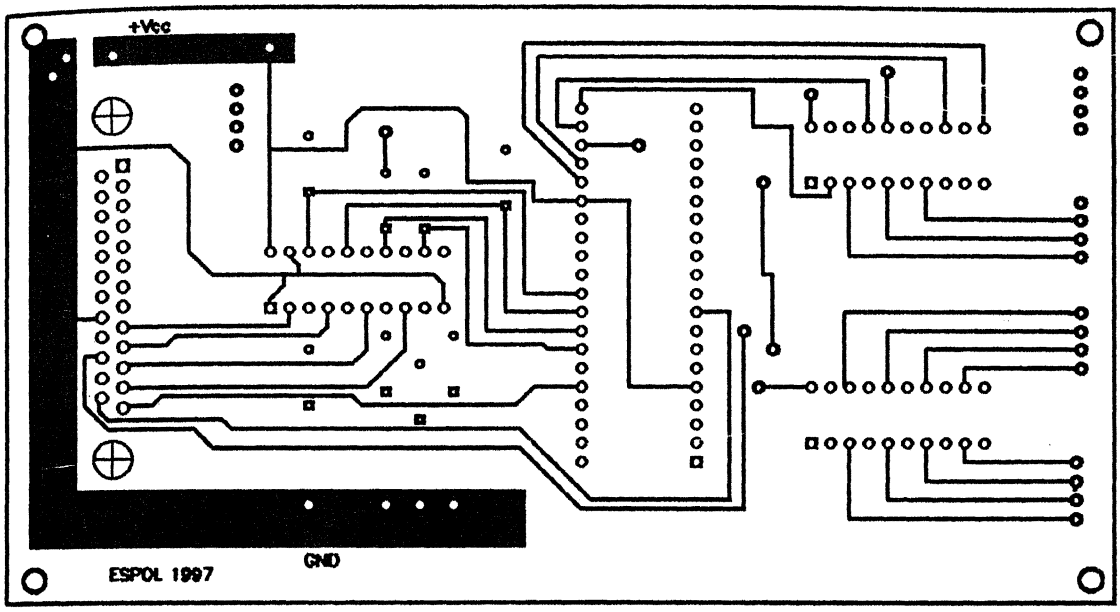
6) Llevar a nivel alto a la señal WR/, A1 y A0 pueden cambiar si es necesario.

El retardo es necesario debido a que la PPI es un dispositivo de E/S relativamente lento, por lo tanto si trabaja con máquinas de más de 8 Mhz se tendrá que extender mediante el programa la duración de las señales WR/, A1, A0 para cumplir con los requerimientos mínimos del dispositivo. En el programa principal se utilizará procedimientos de espera para lograr esto.

La tarjeta en la que estará ubicada la PPI será denominada principal, siendo esta el componente más importante de esta interfase. La conexión entre el puerto paralelo del computador y esta tarjeta no es directa sino a través del CI 74LS244, esto se hace principalmente para proteger a las salidas del puerto de cualquier cortocircuito o anomalía en la tarjeta . Por este buffer pasan las líneas de datos del puerto hacia la PPI, las líneas de



Title	INTERFASE DIGITAL	
Size: A	Document Number: Fig.3.3.2.c	REV



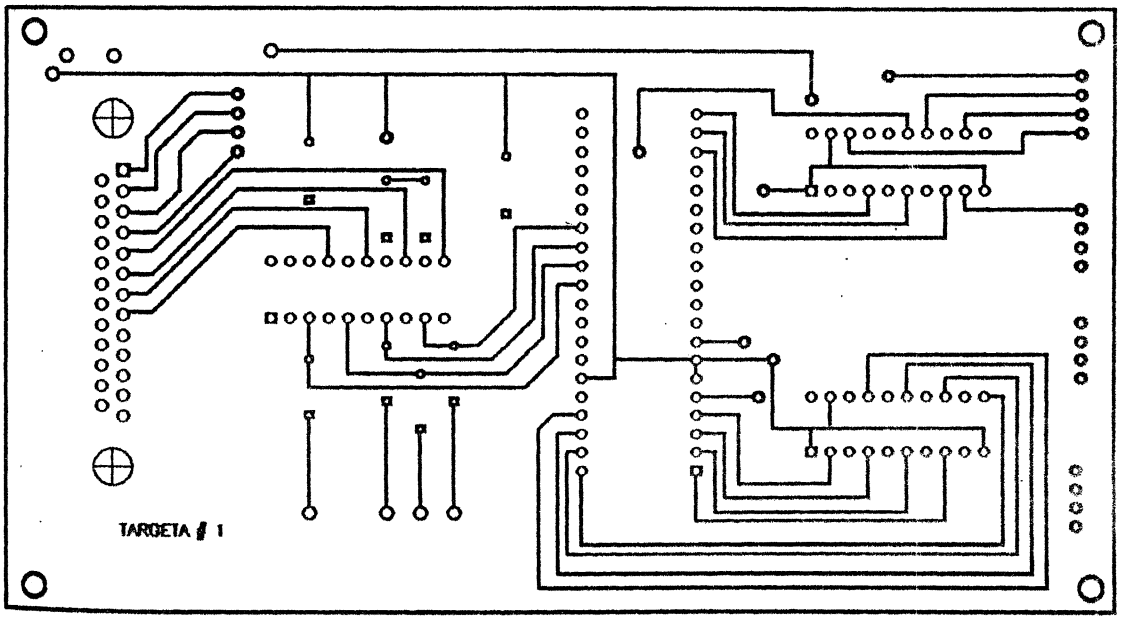


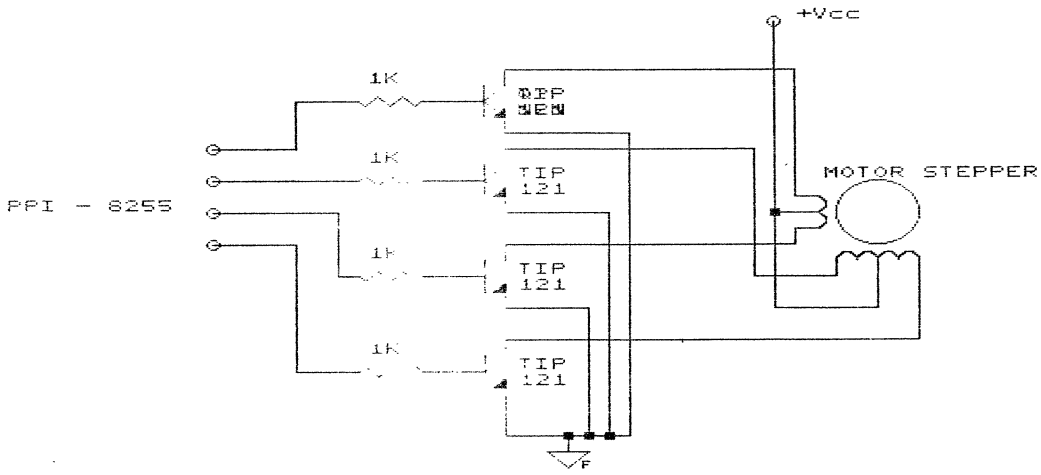
fig. 3.3.2.d

control A1, A0 y WR/ pasan directamente desde el puerto hacia la PPI. El diagrama esquemático de este circuito se muestra en la figura 3.3.2.c.

Luego de que las señales de datos han llegado a la PPI, esta las reparte a cada tarjeta manejadora de cada motor de acuerdo a los requerimientos del usuario. Esto lo hace a través de buffers 74LS244 con la finalidad de proporcionar la corriente de base necesaria para saturar cada transistor, así como para aislar al circuito de fuerza de la PPI. En cada salida de los buffer se ha conectado una resistencia de 1K hacia tierra con la finalidad de evitar comportamientos erróneos de los motores cuando las salidas de la PPI están en estado indeterminado. El gráfico del circuito impreso utilizado para esta interfase es mostrado en la figura 3.3.2.d

3.3.3 INTERFASE DE PODER PARA MOTORES DE PASO

Los pulsos digitales enviados por el computador a cada uno de los motores necesitan de algún dispositivo electrónico que actúe a manera de switch para dejar pasar la corriente a cada una de las bobinas de estos, se utiliza entonces un arreglo de transistores tipo Darlington que trabajan de la siguiente manera:



ESPOL		
Title: MANEJADOR DE MOTOR DE PASOS.		
Size: A	Document Number: Fig. 3.3.3.a	REV: OK

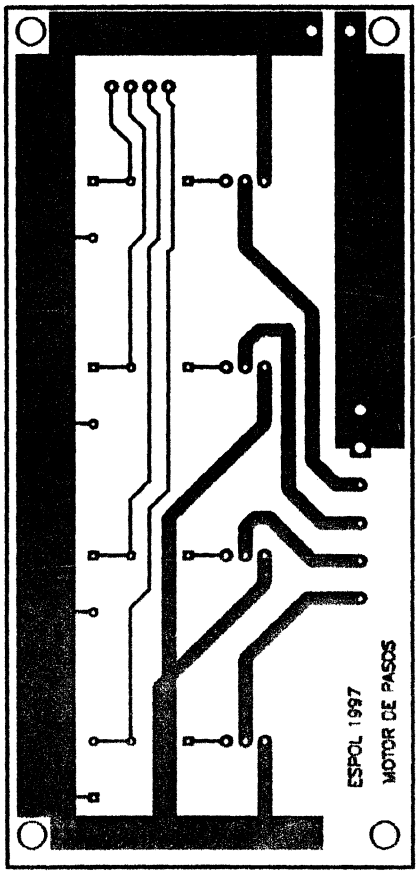


Fig. 3.3.3.b

Los pulsos de nivel bajo llegan a la base del transistor a través de una resistencia de 1K, este valor de voltaje es como máximo de 0.8V debido a que proviene de un dispositivo TTL (buffer 74LS244 cuyas características se detallan en el anexo D), por tanto no es suficiente para polarizar directamente las dos junturas base-emisor del Darlington, para esto se necesitaría 0,7 V+ 0.7V =1,4V, en estas condiciones no habrá flujo de corriente hacia la respectiva bobina. Cuando el pulso pase a alto, será de por lo menos 3,4V de acuerdo a la lógica TTL, esto acompañado del valor adecuado de la resistencia harán que el transistor vaya a saturación dando un valor de voltaje de colector a emisor de 0,3V, de esta manera habrá flujo de corriente desde la fuente de voltaje hacia la respectiva bobina. El diagrama esquemático de este circuito se muestra en la figura 3.3.3.a

El transistor utilizado deberá ser capaz de soportar la corriente demandada por cada bobina de cada motor así como disipar el calor producido cada vez que entra en conducción, igualmente deberá presentar una buena frecuencia de switcheo, tomando en cuenta lo anteriores se utilizará el Transistor tipo Darlington TIP121, cuyas principales características se presentan en la tabla.IV.

Voltaje colector-emisor V_{CE} (V)	Voltaje base-emisor V_{BE} (V)	Corriente de colector I_C (A)	Potencia Disipada (W)
100	5	8	65

Tabla IV. Características del transistor TIP121.

El circuito manejador de cada motor de pasos consta entonces de cuatro resistencias 1K conectadas entre las salidas de los buffer y las bases de cada transistor, por tanto habrá cuatro transistores, uno para cada hilo de cada motor de pasos (los dos hilos restantes se conectar a Vcc según se explica en el anexo C), en cada tarjeta manejadora también se incluyen cuatro resistencias de 1K conectadas entre las salidas de los buffer y tierra , su objetivo es asegurar que estas salidas estén en nivel bajo cuando sus respectivas entradas estén inactivas, esto sucede especialmente cuando se enciende la fuente del sistema y el programa principal todavía no esta corriendo, evitando así el fimcionamiento inoportuno de los motores . Debido a que el manejo de los motores de paso es similar para los tres motores

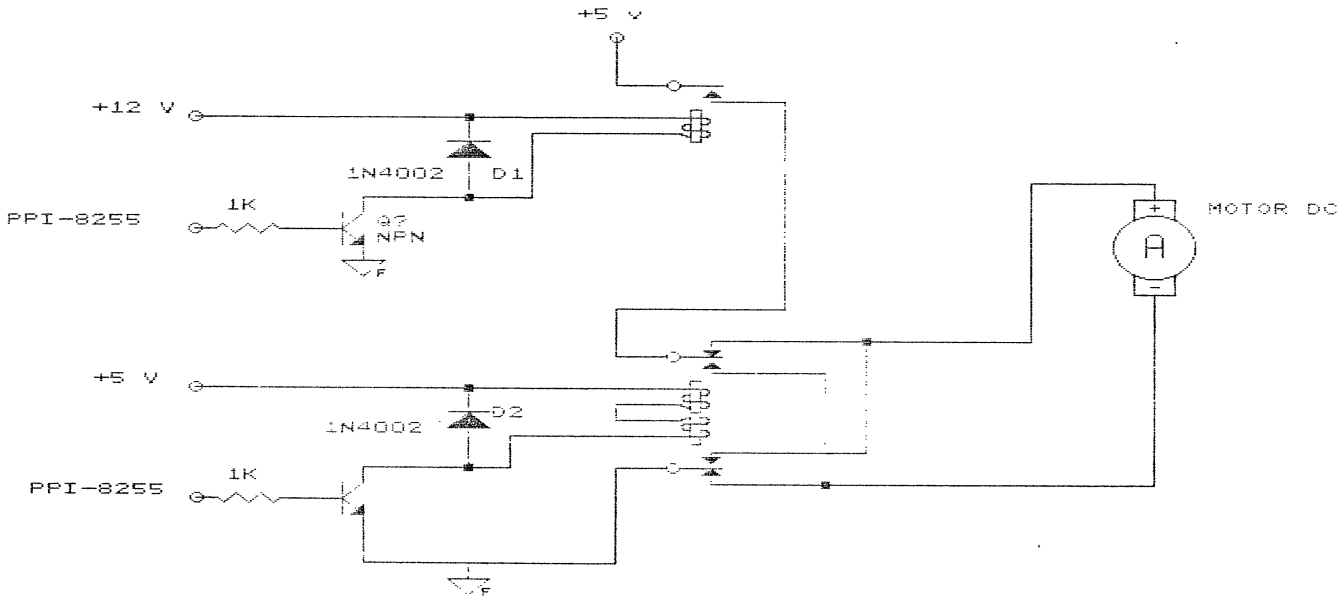
utilizados en este proyecto, se necesitarán tres circuitos similares.

El gráfico del circuito impreso utilizado par la implementación de esta interface se muestra en la figura 3.3.3.b.

3.3.4 INTERFASE DE PODER PARA EL MOTOR DC

Esta interfase tiene como finalidad, controlar la energización del motor D.C que se utilizará como efector final, además de gobernar el sentido de giro de dicho motor. Esto se logra utilizando un par de relés cuya energización o desenergización depende de pulsos provenientes del computador y generados mediante programa.

Estos pulsos llegan desde el puerto paralelo hacia la PPI, y se escriben en el puerto B, bits B5 y B6. De aquí son tomados hacia las bases de dos transistores a través de resistencias de 1K, no sin antes pasar por un buffer 74LS244, estos valores de resistencias son suficientes para saturar a los transistores cuando el programa envía un nivel lógico alto, en estas condiciones los relés se energizan y cierran sus contactos, los que son utilizados de manera conveniente para lograr la inversión de giro o energizar el motor. Los relés están conectados entre los colectores y la fuente de voltaje (+5V o +12V según sea el tipo de relé). Para controlar



ESPOL		
Title: MANEJADOR DE MOTOR DC		
Size: A	Document Number: Fig. 3.3.4.b	REV: OK
Date: December 3, 1997	Sheet 1 of 1	

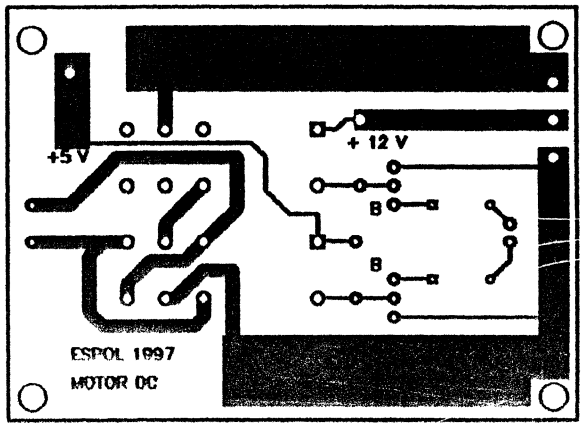


Fig. 3.3.4.a

la dirección de giro del motor D.C se tiene que invertir la polaridad de sus terminales. Como se puede ver en la fig. 3.3.4.a, cuando el relé no está energizado el voltaje de la fuente polariza al motor a través de los contactos normalmente cerrados F1 y F2, y, cuando mediante programa se ordena su energización los contactos F1 y F2 se abren y como contrapartida, los contactos R1 y R2 se cierran polarizando al motor en forma contraria a la anterior, dando como resultado el giro del motor en el otro sentido. El contacto S del relé # 2 actúa simplemente como un switch que permite o no la energización del motor D.C.

La tarjeta manejadora consta de dos transistores BD135, dos relés de 5V y 12V , además de dos resistencias de 1K que llegan a la base de los transistores como se detalla en el diagrama esquemático.

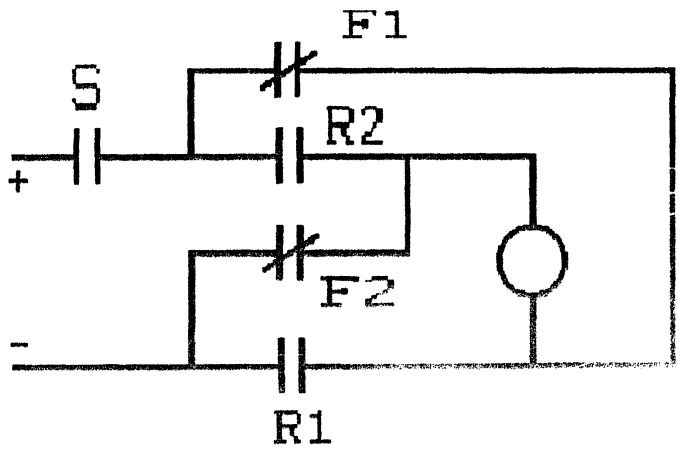


Fig.3.3.4.a Circuito de control para el motor DC

También se incluyen dos diodos conectados en paralelo con la bobina de cada relé con la finalidad de que exista un camino cerrado para la corriente de la bobina cuando se pone en corto el transistor, evitando así la generación de un voltaje excesivamente elevado que pueda afectar a la circuiteriía restante del sistema electrónico

En la figura 3.3.4.b se muestra el diagrama esquemático de esta interfase, el gráfico del circuito impreso se presenta en la figura 3.3.4.c.

3.3.5 INTERFASE DE PODER PARA EL MOTOR AC

Este circuito controlará la energización o desenergización del motor A.C, que gobernará la articulación # 1, además de el sentido de giro de este motor, mediante pulsos provenientes del computador. Estos pulsos se escriben en los bits # y # del puerto B de la PPI, luego pasan por un buffer 74LS244 para llegar a la base de cada transistor a través de resistencias de 1K, en cuyos colectores estarán conectados relés. De uno de estos relés se utiliza sus contactos para hacer el intercambio de fases del motor, logrando así la inversión de giro del mismo. Del otro relé se utiliza sus contactos a manera de switch para energizar o no energizar al motor. El esquema de la fig. 3.3.5.a, muestra el

circuito utilizado para lograr el funcionamiento del motor A.C y la inversión de giro del mismo.

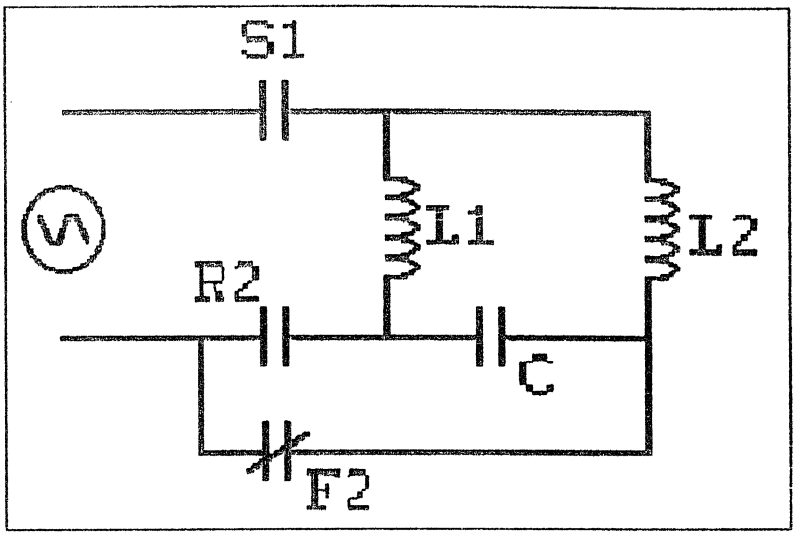
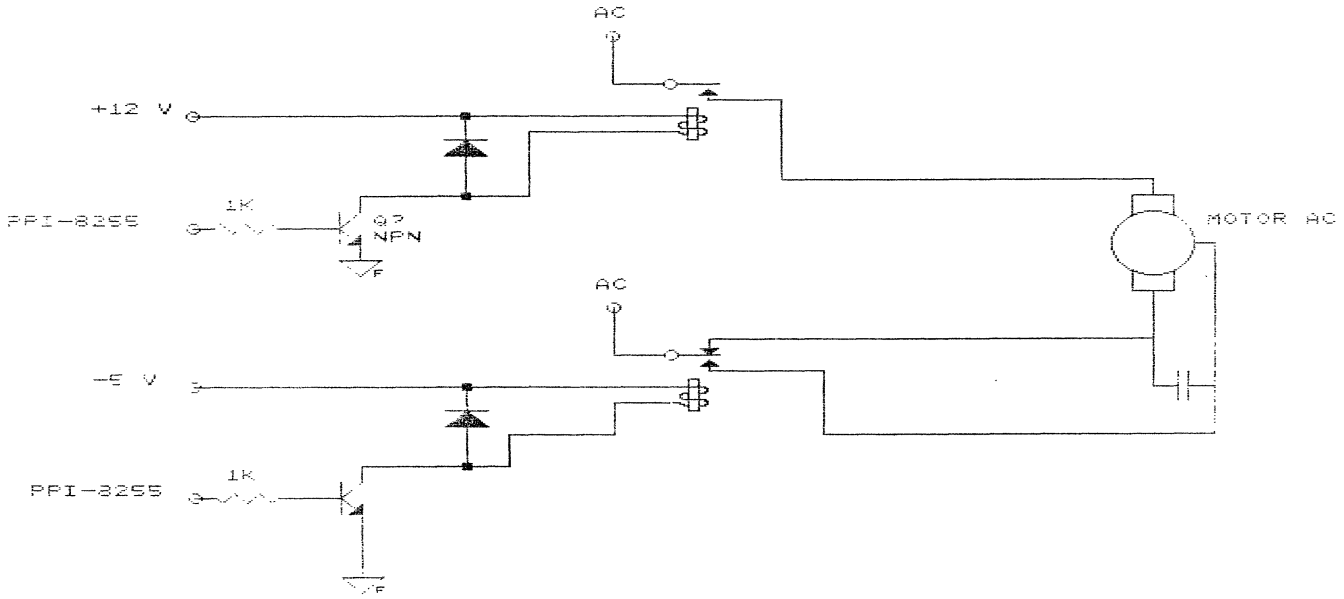


Fig.3.3.5.a Circuito de control para motor AC

La tarjeta manejadora consta de dos transistores BD135, dos resistencias de 1K conectadas una cada una a la base de cada transistor, un relé conectado entre el colector de cada transistor y su respectiva fuente (+5V o +12V según corresponda), además de los diodos protectores conectados en paralelo con la bobina de cada relé, como se muestra en el diagrama esquemático de la figura 3.3.5.b. El circuito impreso para la implementación de esta interfase se muestra en la figura 3.3.5.c.



ESPOL		
Title: MANEJADOR DE MOTOR AC		
Size: A	Document Number: Fig. 3.3.5.b	REV:
Date: December 3, 1997	Sheet 1 of 1	

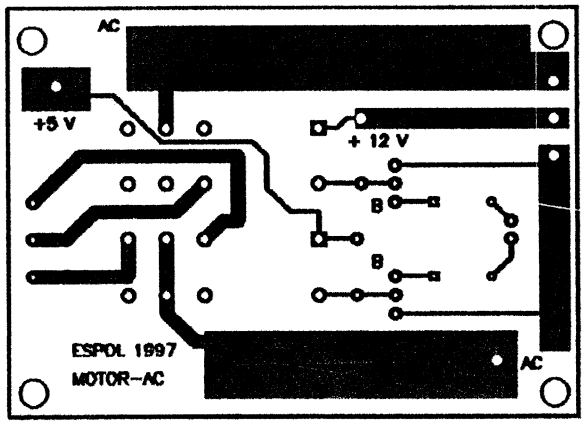


Fig. 3.3.5.c

3.4 DESCRIPCION DE LOS MOTORES UTILIZADOS

En esta sección se describen las características generales de los cinco motores utilizados en el brazo mecánico, más no se hace un detallamiento teórico de cada uno de ellos, puesto que en el anexo C de esta monografía se describen ampliamente los fundamentos de los motores de paso, motores AC y motores DC.

3.4.1 MOTOR # 1

Tipo:	motor AC
Fabricante:	Fuji Electrochemical
Potencia:	12 vatios
Voltaje nominal	120 voltios AC
Torque:	0.4 lb-plg.
Diámetro del eje:	0.2 plg.
Peso:	1.5 lb
Número de terminales:	3
Velocidad:	1800 r.p.m.

Observaciones:

Este motor necesita un capacitor de 0.5 uF para su arranque, además cuenta con un relé que actuá como freno mecánico del motor.

Una ventaja de este motor es que tiene un reductor mecánico incluido, lo cual le permite entregar un torque de 780 lb-plg. Y una velocidad de 1.0 r.p.m.

3.4.2 MOTOR # 2

Tipo:	motor de pasos
Fabricante:	Sanyo Denky
Potencia:	10.0 vatios
Voltaje nominal	4.1 voltios
Torque:	1.0 lb-plg.
Diámetro del eje:	0.2 plg.
Peso:	1.25 lb.
Número de terminales:	6

Observaciones:

Este motor de pasos proporciona 200 pasos por revolución, es decir que por cada paso avanza 1.8°.

3.4.3 MOTOR # 3

Tipo:	motor de pasos
Fabricante:	Sanyo Denky

Potencia:	15 vatios
Voltaje nominal	2.6 voltios
Torque:	3.0 lb-plg.
Diámetro del eje:	0.2 plg.
Peso:	1.75 lb.
Número de terminales:	6

Observaciones

Este motor es de alta precisión, puesto que proporciona 1.8° por paso, además genera un gran torque sin ser físicamente de gran tamaño.

3.4.4 MOTOR # 4

Tipo:	motor de pasos
Fabricante:	Sanyo Denky
Potencia:	20 vatios
Voltaje nominal:	2.5 voltios
Torque:	4.0 lb-plg.
Diámetro del eje:	0.25 plg.
Peso:	40 onzas
Número de terminales:	6

3.4.5 MOTOR # 5

Tipo:	motor DC
Fabricante:	Fujitsu
Potencia:	20 vatios
Voltaje nominal:	12 voltios.
Torque:	0.5 lb-plg
Diámetro del eje:	0.2 plg.
Peso:	1.0 lb.
Número de terminales:	2

Observaciones:

Este motor constituye básicamente el efector final del brazo mecánico, tiene un reductor mecánico incluido internamente, el mismo que le otorga un gran torque y una velocidad razonable, suficientes como para desarrollar el trabajo destinado.

3.5 FUENTES DE PODER

Para normal funcionamiento del brazo robótico ha sido necesario utilizar dos fuentes de poder, debido a que se ha destinado una fuente exclusivamente para la circuitería de control, de tal manera que las caídas de voltaje ocasionadas por los circuitos de fuerza no afecten de

ninguna manera a los circuitos de control. Las dos fuentes utilizadas en el proyecto se describen a continuación.

3.5.1 FUENTE PARA CIRCUITOS DE CONTROL

Tipo:	Fuente DC conmutada
Fabricante:	SONY Corporation.
Voltaje de alimentación:	90 - 130 voltios AC
Salidas:	5.0 voltios - 7.0 amperios
	12.0 voltios - 1.0 amperios
Potencia Salida:	50 vatios

Observaciones:

Este tipo de fuente proporciona un voltaje regulado con un porcentaje de rizado bastante bajo, además cuenta con circuitos de protección contra corto circuito y a pesar de su gran capacidad de salida su tamaño físico es considerablemente pequeño.

3.5.2 FUENTE PARA CIRCUITOS DE FUERZA

Tipo:	Fuente DC conmutada
Fabricante:	KING - YEAR
Voltaje de alimentación:	110 voltios AC

Salidas:	5.0 voltios - 15.0 amperios
	12.0 voltios - 4.2 amperios
	-12.0 voltios - 0.25 amperios
	-5.0 voltios - 0.30 amperios
Potencia Salida:	130 vatios

Observaciones:

Este tipo de fuente es comúnmente usada por la mayoría de los computadores personales, cuenta con circuitos de protección contra corto circuito y poseen un sistema de ventilación interno.

3.6 SENSORES DE LIMITE

Con la finalidad de controlar ó limitar la libertad de movimiento de las distintas articulaciones del brazo robótico, han sido colocados switches mecánicos de dos posiciones, los mismos que son los encargados de sensar los límites de abertura máximo de cada enlace del brazo. Su función consiste en enviar una señal eléctrica cuando un enlace llega a su tope máximo de abertura, esto se consigue provocando que mecánicamente el switch se cierre cuando entre en contacto con el enlace de acercamiento. En la figura 3.6.a se puede apreciar un bosquejo de la forma y ubicación de estos sensores. En total se utilizan

ocho sensores de este tipo, dos por cada articulación, de esta forma se consigue controlar las posiciones extremas de los enlaces en las dos direcciones opuestas.

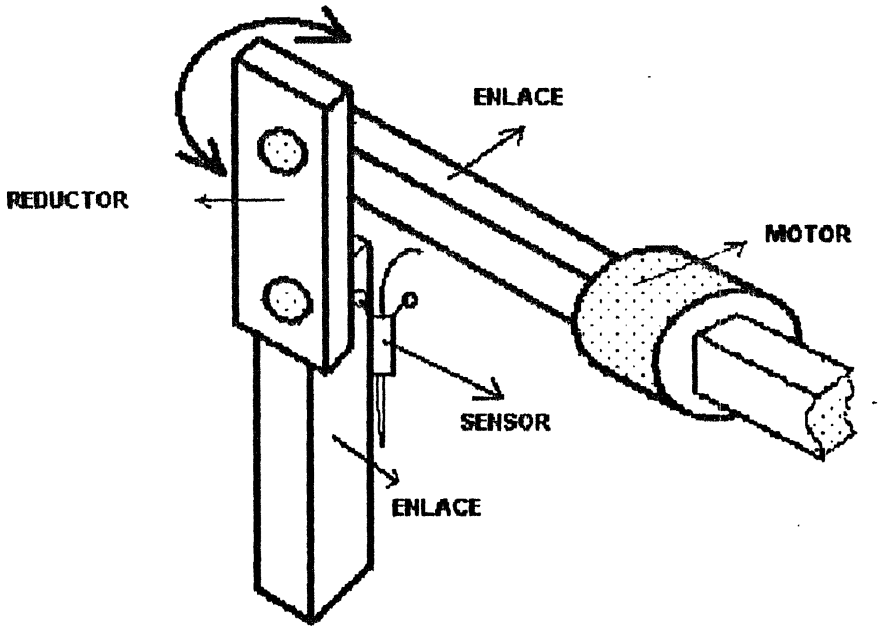
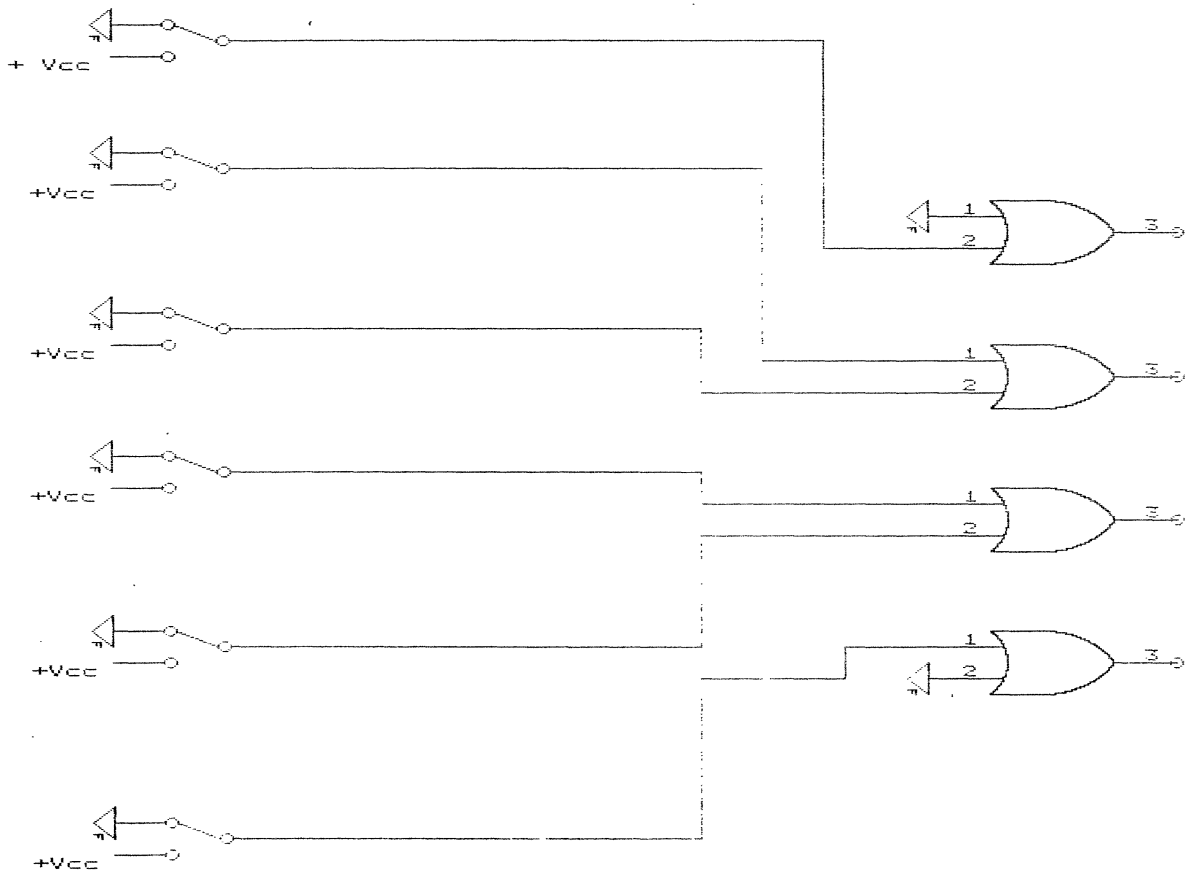


Fig.3.6.a Sensor Limitador de Movimiento

En la figura 3.6.b se muestra el diagrama esquemático del circuito manejador de los sensores. Como se puede observar básicamente un integrado 7432 que tiene cuatro puertas lógicas de tipo OR constituye toda la circuitería necesaria para entregar las señales a la interfase digital para que a través de ella el computador pueda leer la información requerida.



HACIA EL PUERTO PARALELO

ESPOL		
Title: CIRCUITO DE LECTURA PARA SWITCHES DE LIMITE		
Size: A	Document Number: Fig. 3.6.b	REV
Date: December 3, 1997	Sheet 1 of 1	

Se muestra en la figura 3.6.c el gráfico del circuito impreso requerido para la implementación del circuito manejador de los sensores del sistema.

Los sensores utilizados en este sistema eléctrico no constituyen ó no forman parte de un sistema de realimentación, puesto que el robot objeto de este estudio es de tipo NO SERVO. Dichos sensores como ya se descrito anteriormente, sólo son parte de un mecanismo de apoyo para limitar el movimiento del brazo mecánico.

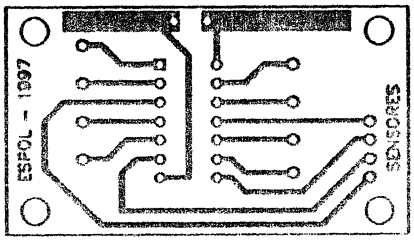


Fig. 3.6.c

CAPITULO IV

EL PROGRAMA DEL SISTEMA

4.1 INTRODUCCION

Habiendo concluido con el análisis eléctrico y mecánico del sistema, en este capítulo se busca detallar la estructura del programa de control del brazo mecánico. Para lograr esto se ha procedido a ilustrar el programa con diagramas de flujo y adividir el mismo en procedimientos generales, específicos y especiales.

El programa principal está desarrollado en lenguaje C, el mismo que utiliza programas de apoyo desarrollados en lenguaje ensamblador, por este motivo llamaremos al programa en lenguje C "programa anfitrión".

Son catorce los programas escritos en lenguaje ensamblador, cada articulación utiliza dos programas de control, al igual que para cada efector final totalizando doce procedimientos, además existe un procedimiento especial para programar la PPI 8255, y otro procedimiento desarrollado para el funcionamiento automático del sistema.

4.2 PARAMETROS UTILIZADOS EN EL PROGRAMA.

El programa "controlador del brazo mecánico", denominado TORTUGAI, ha sido escrito con la ayuda del compilador de Turbo C++ versión 1.0, cuyas librerias de funciones nos han permitido desarrollar una interfase amigable con el usuario, lo que hubiese sido mucho más complicado y extenso utilizando lenguaje ensamblador. Este programa puede ser ejecutado en cualquier computador cuyo microprocesador sea de la familia INTEL versión 8086 o superior, incluida la versión PENTIUM, considerando cambios en los retardos de los procedimientos de apoyo según corresponda.

Los programas de apoyo que son llamados por el programa anfitrión son los siguintes:

- ART1D.EXE
- ART1I.EXE
- ART2D.EXE
- ART2I.EXE
- ART3D.EXE
- ART3I.EXE
- ART4D.EXE
- ART4I.EXE
- ART5D.EXE
- ART5I.EXE
- ART6C.EXE

- ART6S.EXE
- MAINAU.EXE
- PPI.EXE

los cuales fueron ensamblados y encadenados por separado utilizando los programas TASM..EXE y TLINK.EXE respectivamente.

4.3 PROGRAMA PRINCIPAL.

4.3.1 DIAGRAMA DE FLUJO

Como se observa en la figura 4.3.1, el programa pincipal tiene una lógica de fácil ejecución. Luego de cargar el programa entramos a la pantalla de presentación, estando aquí debemos presionar la tecla ENTER, para llegar al menú principal. El programa nos ubica automáticamente en la opción "MANUAL" del menú principal, presionando la tecla ENTER se visualiza el menú de acción de esta opción, caso contrario con las teclas de dirección podremos ubicarnos en las opciones "AUTOMATICO" o "AYUDA" , según se requiera.

Con el fin de explicar de la mejor manera posible, se ha procedido a fraccionar el programa y a detallarlo secuencialmente en un orden adecuado.

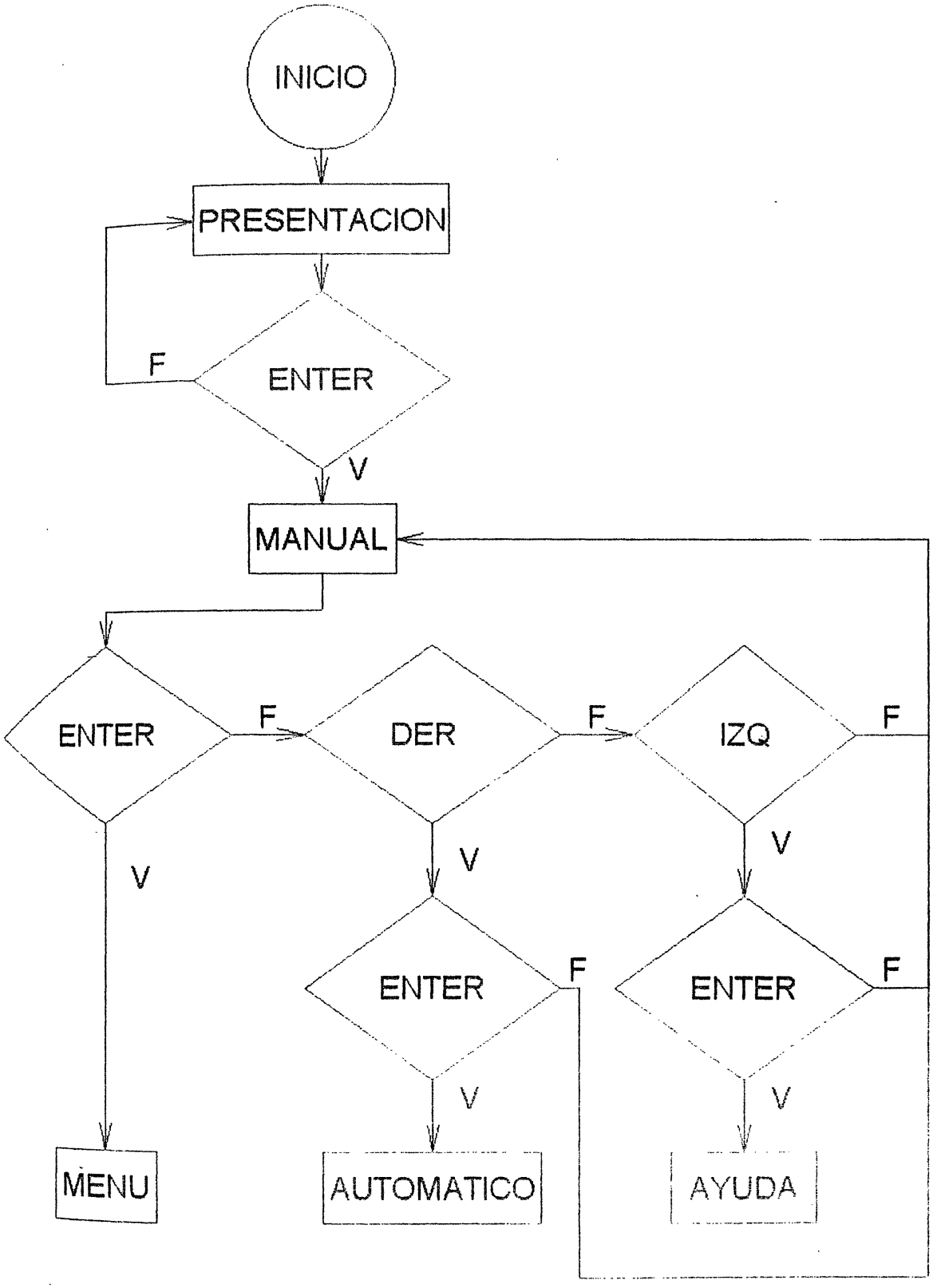

INICIO
PRESENTACION
ENTER
F
V
MANUAL
ENTER
F
DER
F
IZQ
F
V
V
V
ENTER
F
ENTER
F
V
V
MENU
AUTOMATICO
AYUDA

Fig. 4.3.1

4.3.2 ESTRUCTURA PRINCIPAL

```
/* Funcion Principal */
void main(void)
{
   int salir = 0;
   int pos = MENU;
   int posx = 0;
   int posy = 3;
   int ymenor, ymayor, aux1, aux2;
   char c;
   int
ban1d=0,ban1i=0,ban2d=0,ban2i=0,ban3d=0,ban3i=0,ban4d=0,b
an4i=0;
   int cont;
   presentacion();
   menu_inicial();
   spawnl(P_WAIT,"PPI.exe",NULL);
   while(salir == 0)
   {
   gotoxy(80,25);
   c = getch();
   if(c == ESC)
```

```
{
   if(pos == SUBMENU)
   {
        borra_submenu(ymenor,ymayor);
        posx = aux1;
        pos = MENU;
   }
   if(pos == EJECUCION)
   {
        borra_submenu(ymenor,ymayor);
        ymenor = 3;
        ymayor = ymenor + 5;
        posy = aux2;
        pos = SUBMENU;
   }
}
if(c == ENTER)
{
   if(pos == MENU)
   {
        if(posx == 1)
                spawnl(P_WAIT,"MAINAU.exe",NULL);
                else
```

```
        {
          aux1 = posx;
          if(posx == 0)
          {
               ymenor = 3;
               ymayor = ymenor + 5;
          }
          if(posx == 2)
          {
               ymenor = 9;
               ymayor = ymenor + 2;
          }
          submenu(ymenor,ymayor);
          pos = SUBMENU;
          posy = ymenor;
        }
    }
    else
    {
        if(pos == SUBMENU)
        {
          aux2 = posy;
          if(posx == 0)
```

```
{
    if((posy >= 3) && (posy <= 7))
    {
      ymenor = 12;
      ymayor = ymenor + 1;
    }
    else
    {
      ymenor = 14;
      ymayor = ymenor + 1;
    }
    submenu(ymenor,ymayor);
    pos = EJECUCION;
    posy = ymenor;
}
if(posx == 2)
{
    if(posy == 9)
    {
      borra_submenu(ymenor,ymayor);
      acerca_de();
      pos = MENU;
    }
```

```
                if(posy == 10)
                {
                   borra_submenu(ymenor,ymayor);
                   ayuda();
                   pos = MENU;
                }
                if(posy == 11)
                   salir = 1;
             }
          }
```

LIBRERIAS:

```
/* Librerias utilizadas */
#include <stdio.h>
#include <string.h>
#include <dos.h>
#include <conio.h>
#include <process.h>
```

CONSTANTES:

```
/* Constantes utilizadas */
#define NUMERO_BARRAS 16
#define ESC 27
#define ENTER 13
```

```
#define DERECHA 77
#define IZQUIERDA 75
#define ARRIBA 72
#define ABAJO 80
#define MENU 0
#define SUBMENU 1
#define EJECUCION 2
```

BARRAS:

```
/* Estructura para las barras */
struct barra
{
   int xini, xfin, y;
   char *titulo;
};
```

VARIABLES:

```
/* Variables globales */
struct barra barras[NUMERO_BARRAS] = {
                                    { 2, 24, 4, "  M A N U A L  "
},
                                    { 29, 51, 4, " A U T O M A T I C
O " },
```

```
                                    { 56, 78, 4, "    A Y U D A    "
},
                                    { 2, 24, 8, "  ARTICULACION
1  " },
                                    { 2, 24, 11, "  ARTICULACION
2  " },
                                    { 2, 24, 14, "  ARTICULACION
3  " },
                                    { 2, 24, 17, "  ARTICULACION
4  " },
                                    { 2, 24, 20, "  EFECTO FINAL
1  " },
                                    { 2, 24, 23, "  EFECTO FINAL
2  " },
                                    { 56, 78, 8, "  ACERCA DE ...
" },
                                    { 56, 78, 11, "  AYUDA
GENERAL  " },
                                    { 56, 78, 14, "    TERMINAR
" },
                                    { 29, 51, 11, "  GIRO
DERECHA  " },
```

```
                                    { 29, 51, 14, "  GIRO
IZQUIERDA  " },
                                    { 29, 51, 20, "   COGER
" },
                                    { 29, 51, 23, "   SOLTAR
" }
                                };
int color_texto = BLACK;
int color_fondo = WHITE;
int color_marca = LIGHTGREEN;
void MAINSEN(void);
void MAINAU(void);
int SENSO;
union REGS regist;
```

4.3.3 FUNCIONES AUXILIARES

VENTANAS:

```
/* Funcion que dibuja una ventana */
void ventana(int xini, int yini, int xfin, int yfin, int color_borde, int
color_fondo)
```

```
{
  static int i, j;
  textcolor(color_borde);
  textbackground(color_fondo);
  gotoxy(xini,yini);
  cprintf("É");
  gotoxy(xfin,yini);
  cprintf("»");
  gotoxy(xini,yfin);
  cprintf("È");
  gotoxy(xfin,yfin);
  cprintf("¼");
  for(i = xini + 1; i < xfin; i++)
  {
  gotoxy(i,yini);
  cprintf("Í");
  gotoxy(i,yfin);
  cprintf("Í");
  }
for(i = yini + 1; i < yfin; i++)
{
gotoxy(xini,i);
cprintf("º");
```

```
    gotoxy(xfin,i);
    cprintf("o");
    }
    for(i = xini + 1; i < xfin; i++)
    for(j = yini + 1; j < yfin; j++)
    {
       gotoxy(i,j);
       cprintf(" ");
    }
}
```

RECORRER:

/* Funcion que recorre el nombre del alumno a traves de la pantalla */

```
void recorre(char *s, int xini, int x, int y)
{
   static int i, len;
   static char aux[30], temp[30];
   aux[0] = '\0';
   temp[0] = '\0';
   len = strlen(s);
   for(i = len - 1; i > 0; i--)
   {
   aux[0] = s[i];
```

```
    aux[1] = '\0';
    strcat(aux,temp);
    strcpy(temp,aux);
    gotoxy(xini,y);
    cprintf("%s",temp);
    delay(100);
    }
   strcpy(temp," ");
   strcat(temp,s);
   for(i = xini; i < x; i++)
   {
     gotoxy(i,y);
     cprintf("%s",temp);
     delay(100);
   }
}
```

PRESENTACION:

```
/* Funcion que crea la pantalla de presentacion */
void presentacion(void)
{
   char *cadena1 = "ESCUELA SUPERIOR POLITECNICA DEL
LITORAL";
```

```
char *cadena2 = "FIEC";
char *cadena3 = "PROYECTO DE GRADO";
char *cadena4 = "TEMA DEL PROYECTO:";
char *cadena5 = "BRAZO MECANICO";
char *cadena6 = "PROFESOR DIRECTOR:";
char *cadena7 = "Ing. Hugo Villavicencio Villavicencio";
char *cadena8 = "ALUMNOS INTEGRANTES:";
char *cadena9 = "Guayaquil, Noviembre de 1997";
char *nombre1 = "Pablo Germ n Parra Rosero";
char *nombre2 = "Adrian Oswaldo Arce Bastidas";
clrscr();
ventana(5,20,75,23,LIGHTCYAN,BROWN);
textcolor(BLACK);
gotoxy(21,4);
cprintf("%s",cadena1);
gotoxy(39,6);
cprintf("%s",cadena2);
gotoxy(32,8);
cprintf("%s",cadena3);
gotoxy(7,10);
cprintf("%s",cadena4);
gotoxy(7,13);
cprintf("%s",cadena6);
```

```
   gotoxy(7,16);
   cprintf("%s",cadena8);
   gotoxy(27,21);
   cprintf("%s",cadena9);
   textcolor(BLUE);
   gotoxy(27,11);
   cprintf("%s",cadena5);
   gotoxy(27,14);
   cprintf("%s",cadena7);
   recorre(nombre1,6,27,17);
   recorre(nombre2,6,27,18);
   textbackground(BLACK);
   textcolor(WHITE);
   gotoxy(19,24);
   cprintf("Presione una tecla para iniciar el programa.");
   getch();
}
```

Cuando se presiona una tecla para iniciar el programa, al mismo tiempo se ordena ejecutar el programa PPI.EXE.

BARRAS:

```
/* Funcion que crea una barra */
void llena_barra(struct barra bar, int color_fondo, int
color_letras)
```

```
{
   textcolor(color_letras);
   textbackground(color_fondo);
   gotoxy(bar.xini,bar.y - 1);
   cprintf("ÚÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ¿");
   gotoxy(bar.xini,bar.y);
   cprintf("³%s³",bar.titulo);
   gotoxy(bar.xini,bar.y + 1);
   cprintf("ÀÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÙ");
}
```

MENU PRINCIPAL:

```
/* Funcion que crea el menu principal */
void menu_inicial(void)
{
/* COLOR DE LA PANTALLA*/
     textbackground(RED);
   clrscr();
   textcolor(BLACK);
   textbackground(BROWN);
   gotoxy(26,1);
   cprintf(" B R A Z O   M E C A N I C O ");
   llena_barra(barras[0],color_marca,BLACK);
   llena_barra(barras[1],color_fondo,BLACK);
```

```
   llena_barra(barras[2],color_fondo,BLACK);
}
```

SUBMENU :

```
/* Funcion que crea un submenu del menu principal */
void submenu(int menor, int mayor)
{
   static int i;
   llena_barra(barras[menor],color_marca,BLACK);
   for(i = menor + 1; i <= mayor; i++)
   llena_barra(barras[i],color_fondo,BLACK);
}
```

SALIR DE SUBMENU:

```
/* Funcion que elimina un submenu */
void borra_submenu(int menor, int mayor)
{
   static int i;
   for(i = menor; i <= mayor; i++)
   llena_barra(barras[i],RED,RED);
}
```

PANTALLAS DE SUBMENU:

```
/* Funcion que crea la pantalla Acerca De */
void acerca_de(void)
{
```

```
static char c;
ventana(10,8,70,25,WHITE,LIGHTBLUE);
textcolor(BLACK);
gotoxy(34,9);
cprintf("Acerca De ...");
gotoxy(34,10);
cprintf("_____");
textcolor(WHITE);
gotoxy(14,12);
cprintf("La responsabilidad por los hechos, ideas y");
gotoxy(14,13);
cprintf("doctrinas expuestas en esta Tesis, nos corresponden");
gotoxy(14,14);
cprintf("exclusivamente; y, el patrimonio intelectual de la");
gotoxy(14,15);
cprintf("misma, a la ESCUELA SUPERIOR POLITECNICA
DEL LITORAL.");
gotoxy(14,17);
cprintf("(Reglamento de Ex menes y Tjtulos Profesionales de");
gotoxy(14,18);
cprintf("la ESPOL)");
gotoxy(14,20);
```

```
   cprintf("ADRIAN ARCE B.                PABLO PARRA
R.");
   gotoxy(35,24);
   cprintf("Salir:");
   textcolor(LIGHTRED);
   gotoxy(42,24);
   cprintf("ESC");
   do
   {
    c=getch();
   }
   while(c != ESC);
   ventana(10,8,70,25,RED,RED);
}
```

PANTALLA AYUDA

```
/* Funcion que crea la pantalla de Ayuda */
void ayuda(void)
{
   static char c;
   ventana(10,8,70,25,WHITE,LIGHTBLUE);
   textcolor(BLACK);
   gotoxy(34,9);
   cprintf("Ayuda General");
```

```
   gotoxy(34,10);
   cprintf("_____");
   textcolor(WHITE);
   gotoxy(14,12);
   cprintf("Para escoger una opci¢n utilice las teclas de direc-");
   gotoxy(14,13);
   cprintf("cionamiento y luego presione ENTER.");
   gotoxy(14,15);
   cprintf("Las opciones disponibles son las siguientes:");
   gotoxy(14,17);
   cprintf("MANUAL: Al escoger esta opci¢n se presentar un
men£");
   gotoxy(14,18);
   cprintf("      de opciones para escoger la parte del BRAZO");
   gotoxy(14,19);
   cprintf("      MECANICO que quiera moverse, en forma
manual.");
   gotoxy(14,21);
   cprintf("AUTOMATICO: Esta opci¢n permite que el BRAZO
MECANICO");
   gotoxy(14,22);
   cprintf("          funcione en forma autom tica.");
   textcolor(BLACK);
```

```
gotoxy(20,24);
cprintf("Siguiente:");
textcolor(LIGHTRED);
gotoxy(31,24);
cprintf("ENTER");
textcolor(BLACK);
gotoxy(50,24);
cprintf("Salir:");
textcolor(LIGHTRED);
gotoxy(57,24);
cprintf("ESC");
do
{
c=getch();
}
while((c != ESC) && (c != ENTER));
if(c == ENTER)
{
textcolor(WHITE);
gotoxy(14,12);
cprintf("AYUDA: Al escoger esta opciģn se presentar  um
menŁ  ");
gotoxy(14,13);
```

```
    cprintf("      de opciones que dan informaci¢n general del   ");
    gotoxy(14,14);
    cprintf("      programa.                                     ");
    gotoxy(14,15);
    cprintf("                                                    ");
    gotoxy(14,16);
    cprintf("ARTICULACION n: (n puede ser 1,2,3, o 4) Al
escoger  ");
    gotoxy(14,17);
    cprintf("            esta opci¢n se presentar  2 opciones,");
    gotoxy(14,18);
    cprintf("            que permitir  mover esa articulaci¢n.");
    gotoxy(14,19);
    cprintf("                                                    ");
    gotoxy(14,20);
    cprintf("EFECTOR FINAL 1: Al escoger esta opci¢n el
 usuario   ");
    gotoxy(14,21);
    cprintf("            podr  escoger la direcci¢n de giro  ");
    gotoxy(14,22);
    cprintf("            del taladro.                        ");
    }
    do
```

```
{
c=getch();
}
while((c != ESC) && (c != ENTER));
if(c == ENTER)
{
textcolor(WHITE);
gotoxy(14,12);
cprintf("EFECTOR FINAL 2: Al escoger esta opci¢n el
usuario  ");
gotoxy(14,13);
cprintf("            podr  escoger entre tomar o soltar  ");
gotoxy(14,14);
cprintf("            objeto utilizando un electroim n.  ");
gotoxy(14,15);
cprintf("                                               ");
gotoxy(14,16);
cprintf("ACERCA DE: Esta opci¢n nos presentar  una
aclaraci¢n ");
gotoxy(14,17);
cprintf("        importante sobre el programa.          ");
gotoxy(14,18);
cprintf("                                               ");
```

```
    gotoxy(14,19);
    cprintf("AYUDA GENERAL: Al seleccionar esta opci¢n se
presenta");
    gotoxy(14,20);
    cprintf("          la pantalla actual.          ");
    gotoxy(14,21);
    cprintf("                                       ");
    gotoxy(14,22);
    cprintf("TERMINAR: Con esta opci¢n podemos salir del
programa.");
    }
    do
    {
    c=getch();
    }
    while((c != ESC) && (c != ENTER));
    if(c == ENTER)
    {
    textcolor(WHITE);
    gotoxy(14,12);
    cprintf("IZQUIERDA: Esta opci¢n permite mover la respectiva
");
    gotoxy(14,13);
```

```
    cprintf("        Articulaci¢n o Efector Final 1 a la      ");
    gotoxy(14,14);
    cprintf("        izquierda.                               ");
    gotoxy(14,15);
    cprintf("                                                 ");
    gotoxy(14,16);
    cprintf("DERECHA: Esta opci¢n permite mover la respectiva
");
    gotoxy(14,17);
    cprintf("        Articulaci¢n o Efector Final 1 a la derecha.");
    gotoxy(14,18);
    cprintf("                                                 ");
    gotoxy(14,19);
    cprintf("COGER: Con esta opci¢n el Efector Final 2 tomar  un
");
    gotoxy(14,20);
    cprintf("      objeto met lico.                           ");
    gotoxy(14,21);
    cprintf("                                                 ");
    gotoxy(14,22);
    cprintf("SOLTAR: Permite al Efector Final 2 soltar el objeto.
");
    gotoxy(14,24);
```

```
    cprintf("                                    ");
    }
    gotoxy(35,24);
    cprintf("Salir:");
```

ESTRUCTURA PRINCIPAL:

```
/* Funcion Principal */
void main(void)
{
    int salir = 0;
    int pos = MENU;
    int posx = 0;
    int posy = 3;
    int ymenor, ymayor, aux1, aux2;
    char c;
    int
ban1d=0,ban1i=0,ban2d=0,ban2i=0,ban3d=0,ban3i=0,ban4d=0,b
an4i=0;
    int cont;
    presentacion();
    menu_inicial();
    spawnl(P_WAIT,"PPI.exe",NULL);
    while(salir == 0)
    {
```

```
gotoxy(80,25);
c = getch();
if(c == ESC)
{
  if(pos == SUBMENU)
  {
      borra_submenu(ymenor,ymayor);
      posx = aux1;
      pos = MENU;
  }
  if(pos == EJECUCION)
  {
      borra_submenu(ymenor,ymayor);
      ymenor = 3;
      ymayor = ymenor + 5;
      posy = aux2;
      pos = SUBMENU;
  }
}
if(c == ENTER)
{
  if(pos == MENU)
  {
```

```
        if(posx == 1)
                spawnl(P_WAIT,"MAINAU.exe",NULL);
                else
        {
          aux1 = posx;
          if(posx == 0)
          {
                ymenor = 3;
                ymayor = ymenor + 5;
          }
          if(posx == 2)
          {
                ymenor = 9;
                ymayor = ymenor + 2;
          }
          submenu(ymenor,ymayor);
          pos = SUBMENU;
          posy = ymenor;
        }
}
else
{
      if(pos == SUBMENU)
```

```
{
   aux2 = posy;
   if(posx == 0)
   {
        if((posy >= 3) && (posy <= 7))
        {
          ymenor = 12;
          ymayor = ymenor + 1;
        }
        else
        {
          ymenor = 14;
          ymayor = ymenor + 1;
        }
        submenu(ymenor,ymayor);
        pos = EJECUCION;
        posy = ymenor;
   }
   if(posx == 2)
   {
        if(posy == 9)
        {
          borra_submenu(ymenor,ymayor);
```

```
            acerca_de();
            pos = MENU;
        }
        if(posy == 10)
        {
            borra_submenu(ymenor,ymayor);
            ayuda();
            pos = MENU;
        }
        if(posy == 11)
            salir = 1;
    }
}
```

4.4 PROCEDIMIENTOS GENERALES.

Se ha denominado procedimientos generales a aquellas parets del programa que se ejecutan al ingresar en cualquiera de las tres opciones del menú principal. En la sección anterior (4.3), se describió la opción de ayuda dentro del programa principal, puesto que esta consta solamente de finciones auxiliares, quedándonos por describir los procedimientos de control mnual y automático.

4.4.1 CONTROL MANUAL.

AL presionar la tecla ENTER en esta opción del menú principal, el programa abre un submenú con seis opciones de selección. Para explicar la lógica del funcionamiento de esta parte del programa se ha procedido a ilustrar el diagrama de flujo de una de las opciones del submenú "MANUAL". La lógica de funcionamiento para las restantes opciones es idéntica a la detallada en esta sección.

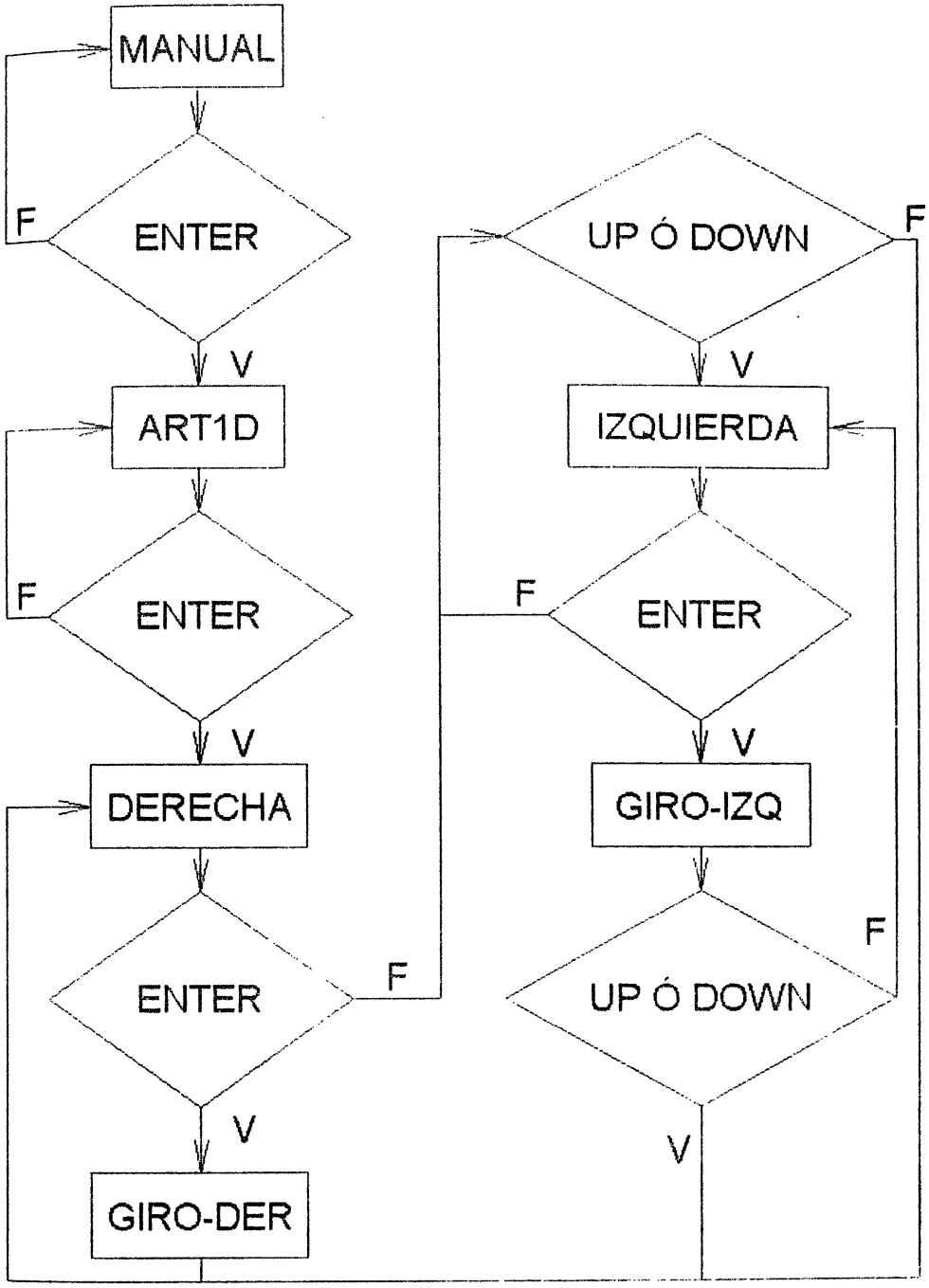

MANUAL
ENTER
F
V
ART1D
ENTER
F
V
DERECHA
ENTER
F
V
GIRO-DER
UP Ó DOWN
F
V
IZQUIERDA
ENTER
F
V
GIRO-IZQ
UP Ó DOWN
F
V

Fig. 4.4.1

Se ha seleccionado el procedimiento para mover la articulación # 1. Como se puede apreciar en el diagrama de flujo, al presionar la tecla ENTER en la opcion MANUAL, automaticamente se visualiza el submenú con la opción "ARTICULACION # 1 " resaltada, lo que implica que el programa está en espera de un ENTER para abrir un submenú de ejecución. Si no se presiona ENTER, el programa permite la opción de selección mediante la utilización de las teclas de dirección, de esta manera se podrá trabajar con cualquier otra articulación o efector final.

Estando dentro del menú de ejecución de cualquier opción escojida (ARTICULACION # 1 en este caso), se podrá seleccionar el sentido de giro de la articulación, presionando una de las teclas de dirección, hacia arriba ó hacia abajo. Sel valida la selección con un ENTER y el programa en este instante ordena al ejecución del programa respectivo. En cuelquier caso para salir de un submenú, el programa espera por la validación de la tecla ESC y de ser así se ubica en el menú superior correspondiente. A continuación se detalla el código de la parte del programa correspondiente a la opción manual del menú principal. La descripción es completa y considera a todas las obciones del submenú MANUAL con los respectivos submenús

de ejecución para cada una de las articulaciones y efectores finales:

```
{
    if(pos == EJECUCION)
    {
        textbackground(BLACK);
        if(aux2 == 3)
        {  if(posy == 12 && ban1d==0)
           { for(cont=1;cont<=30;cont++)
             { MAINSEN();
        SENSO=(SENSO ^ 255);   //NEGANDO
        SENSO=(SENSO & 128);    //10000000b
        if(SENSO==128) goto DECIDIR1D;
              ban1i=0;
           SIPUEDE1D:
              spawnl(P_WAIT,"ART1D.exe",NULL);
           }
           goto ART1I;

           DECIDIR1D:
          if(ban1i==1) goto SIPUEDE1D;
          else ban1d=1;
```

```
   }
   ART1I:
   if(posy == 13 && ban1i==0)
   { for(cont=1;cont<=30;cont++)
    { MAINSEN();
SENSO=(SENSO ^ 255);   //NEGANDO
SENSO=(SENSO & 128);    //10000000b
if(SENSO==128) goto DECIDIR1I;
ban1d=0;
    SIPUEDE1I:
spawnl(P_WAIT,"ART1I.exe",NULL);
    }
    goto ART2;
    DECIDIR1I:
if(ban1d==1) goto SIPUEDE1I;
else ban1i=1;
   }
}

ART2:
if(aux2 == 4)
{
  if(posy == 12 && ban2d==0)
```

```
    { for(cont=1;cont<=30;cont++)
      { MAINSEN();
    SENSO=(SENSO & 16);      //00010000b
    if(SENSO==16) goto DECIDIR2D;
          ban2i=0;
      SIPUEDE2D:
    spawnl(P_WAIT,"ART2D.exe",NULL);
      }
      goto ART2I;
      DECIDIR2D:
    if(ban2i==1) goto SIPUEDE2D;
          else ban2d=1;
     }
     ART2I:
     if(posy == 13 && ban2i==0)
     { for(cont=1;cont<=30;cont++)
       { MAINSEN();
    SENSO=(SENSO & 16);      //00010000b
    if(SENSO==16) goto DECIDIR2I;
    ban2d=0;
SIPUEDE2I:

          spawnl(P_WAIT,"ART2I.exe",NULL);
```

```
        }
        goto ART3;
        DECIDIR2I:
    if(ban2d==1) goto SIPUEDE2I;
    else ban2i=1;
      }
    }
    ART3:
    if(aux2 == 5)
    {
      if(posy == 12 && ban3d==0)
      { for(cont=1;cont<=30;cont++)
        { MAINSEN();
    SENSO=(SENSO & 32);//00100000b
    if(SENSO==32) goto DECIDIR3D;
    ban3i=0;
SIPUEDE3D:
    spawnl(P_WAIT,"ART3D.exe",NULL);
        }
        goto ART3I;
        DECIDIR3D:
            if(ban3i==1) goto SIPUEDE3D;
            else ban3d=1;
```

```
            }
            ART3I:
            if(posy == 13 && ban3i==0)
            { for(cont=1;cont<=30;cont++)
              { MAINSEN();
SENSO=(SENSO & 32);      //00100000b
if(SENSO==32) goto DECIDIR3I;
        ban3d=0;
            SIPUEDE3I:
                spawnl(P_WAIT,"ART3I.exe",NULL);
            }
            goto ART4;
            DECIDIR3I:
                if(ban3d==1) goto SIPUEDE3I;
                else ban3i=1;
            }
          }
          ART4:
          if(aux2 == 6)
          {
            if(posy == 12 && ban4d==0)
            { for(cont=1;cont<=100;cont++)
              { MAINSEN();
```

```
SENSO=(SENSO & 64);      //01000000b
if(SENSO==64) goto DECIDIR4D;
        ban4i=0;
    SIPUEDE4D:
spawnl(P_WAIT,"ART4DA.exe",NULL);
    }
    spawnl(P_WAIT,"PPI.exe",NULL);
    goto ART4I;
    DECIDIR4D:
        if(ban4i==1) goto SIPUEDE4D;
        else ban4d=1;
    spawnl(P_WAIT,"PPI.exe",NULL);
  }
  ART4I:
  spawnl(P_WAIT,"ART4DB.exe",NULL);
  if(posy == 13 && ban4i==0)
  { for(cont=1;cont<=100;cont++)
    { MAINSEN();
SENSO=(SENSO & 64);      //01000000b
if(SENSO==64) goto DECIDIR4I;
        ban4d=0;
    SIPUEDE4I:
        spawnl(P_WAIT,"ART4IA.exe",NULL);
```

```
    }
    spawnl(P_WAIT,"PPI.exe",NULL);
    spawnl(P_WAIT,"ART4IB.exe",NULL);
    goto ART5;
    DECIDIR4I:
        if(ban4d==1) goto SIPUEDE4I;
        else ban4i=1;
    spawnl(P_WAIT,"PPI.exe",NULL);
    spawnl(P_WAIT,"ART4IB.exe",NULL);
  }
}
ART5:
if(aux2 == 7)
{
  if(posy == 12)
  { spawnl(P_WAIT,"ART5D.exe",NULL); }
  if(posy == 13)
  { spawnl(P_WAIT,"ART5I.exe",NULL); }
}
if(aux2 == 8)
{
  if(posy == 14)
  { spawnl(P_WAIT,"ART6C.exe",NULL); }
```

```
                    if(posy == 15)
                    { spawnl(P_WAIT,"ART6S.exe",NULL); }
                }
            }
        }
    }
}
if(c == 0)
{
    c = getch();
    if((c == DERECHA) && (pos == MENU))
    {
        llena_barra(barras[posx],color_fondo,BLACK);
        posx++;
        if(posx > 2)
            posx = 0;
        llena_barra(barras[posx],color_marca,BLACK);
    }
    if((c == IZQUIERDA) && (pos == MENU))
    {
        llena_barra(barras[posx],color_fondo,BLACK);
        posx--;
        if(posx < 0)
```

```
            posx = 2;
        llena_barra(barras[posx],color_marca,BLACK);
    }
    if((c == ARRIBA) && (pos != MENU))
    {
        llena_barra(barras[posy],color_fondo,BLACK);
        posy--;
        if(posy < ymenor)
            posy = ymayor;
        llena_barra(barras[posy],color_marca,BLACK);
    }
    if((c == ABAJO) && (pos != MENU))
    {
        llena_barra(barras[posy],color_fondo,BLACK);
        posy++;
        if(posy > ymayor)
            posy = ymenor;
        llena_barra(barras[posy],color_marca,BLACK);
    }
}
}
textcolor(WHITE);
textbackground(BLACK);
```

```
   clrscr();
}
void MAINSEN(void)
{ unsigned char temp;
 temp=inportb(0x379);
 SENSO=(temp & 248);        //11111000b
}
```

Como podemos observar, para las cuatro articulaciones se hace el llamado a la funcion MAINSEN, dicha función se encarga de leer desde el puerto paralelo el estado de los sensores de límite. El dato tomado por esta función es almacenado en la variable SENSO, valor de la cual es consultado en el procedimiento desarrollado para cada una de las cuatro articulaciones, dependiendo de su valor se permite ó no la ejecución de los programas correspondiente a cada una de las articulaciones.

4.4.2 CONTROL AUTOMATICO

Esta es otra de las opciones disponibles en el menú principal del programa. El programa principal se encarga de llamar a la ejecución de un programa desarrollado en lenguaje ENSAMBLADOR cuando esta opción es seleccionada y validada pulsando la tecla ENTER.

En la figura 4.4.2 se ilustra el diagrama de flujo de la lógica de funcionamiento del programa MAINAU.EXE, que es el encargado de hacer ejecutar una determinada secuencia de acciones al brazo mecánico.

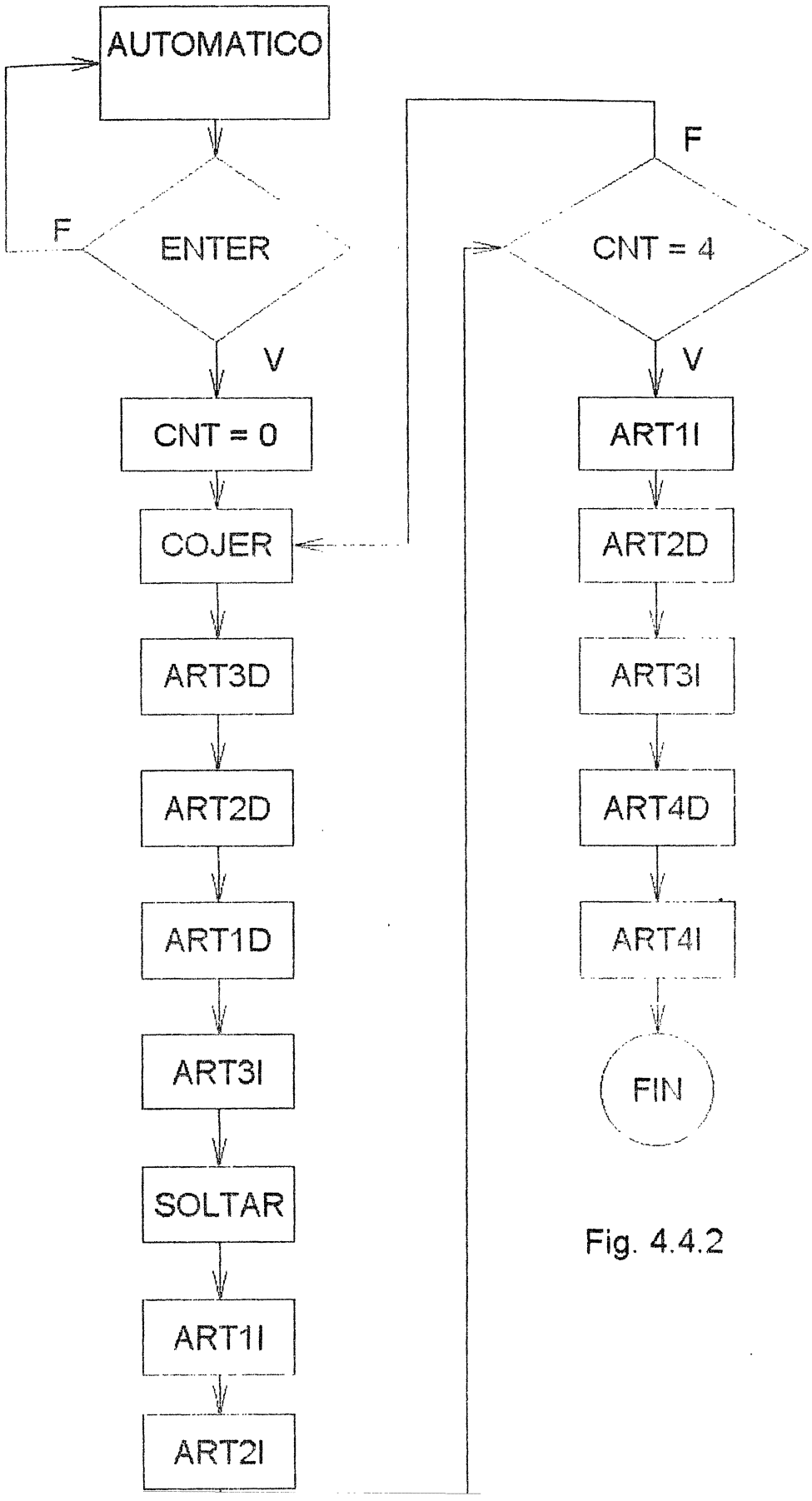

AUTOMATICO
ENTER
F
V
CNT = 0
COJER
ART3D
ART2D
ART1D
ART3I
SOLTAR
ART1I
ART2I
CNT = 4
F
V
ART1I
ART2D
ART3I
ART4D
ART4I
FIN

Fig. 4.4.2

A continuación se detalla el codigo del programa MAINAU.ASM, el mismo que utiliza procedimientos especiales que se detallan más adelente en este trabajo.

```
;-------------------------------------------------------------------------
;-------------------------------------------------------------------------
  TITLE MAINAU
  PILA SEGMENT PARA  PUBLIC 'STACK'
        DW 10 DUP(?)
  PILA ENDS
;-------------------------------------------------------------------------
DATO SEGMENT PARA PUBLIC 'DATA'

DATO ENDS
;-------------------------------------------------------------------------
CODIGO SEGMENT PARA PUBLIC 'CODE'
;-------------------------------------------------------------------------
MAINTIC PROC FAR                    ;PROGRAMA PRINCIPAL
 ASSUME CS:CODIGO, DS:DATO,SS:PILA
 PUSH DS
 SUB AX,AX
 PUSH AX
 MOV AX,DATO
 MOV DS,AX
 CALL MATIC
 RET
MAINTIC ENDP
;-------------------------------------------------------------------------
;-------------------------------------------------------------------------
MATIC PROC NEAR
```

```
PUSH AX
PUSH BX
PUSH CX
PUSH DX
CALL FRENTE5
CALL RETARDO
mov cx,20
qx:
call frente3
loop qx
MOV CX,100
QA:
CALL FRENTE2
LOOP QA
CALL RETARDO
MOV CX,40
QB:
CALL FRENTE1
LOOP QB
CALL RETARDO
 mov cx,20
qq:
call retro3
loop qq
call retardo
CALL RETRO5
MOV CX,35
QC:
CALL RETRO1
LOOP QC
```

```
 CALL RETARDO
 MOV CX,100
QD:
 CALL RETRO2
 LOOP QD
 CALL RETARDO
;-----------------------------------------
 CALL FRENTE5
 CALL RETARDO
 MOV CX,20
 RA:
 CALL FRENTE3
 LOOP RA
 MOV CX,100
QE:
 CALL FRENTE2
 LOOP QE
 CALL RETARDO
 MOV CX,35
QF:
 CALL FRENTE1
 LOOP QF
 CALL RETARDO
 mov cx,20
 qt:
 CALL RETRO3
 loop qt
 CALL RETRO5
 MOV CX,30
QG:
```

```
CALL RETRO1
LOOP QG
CALL RETARDO
MOV CX,100
QH:
CALL RETRO2
LOOP QH
CALL RETARDO
;--------------------------------
CALL FRENTE5
CALL RETARDO
MOV CX,20
RB:
CALL FRENTE3
LOOP RB
MOV CX,100
QI:
CALL FRENTE2
LOOP QI
CALL RETARDO
MOV CX,30
QJ:
CALL FRENTE1
LOOP QJ
CALL RETARDO
mov cx,20
qu:
CALL RETRO3
loop qu
CALL RETRO5
```

```
 MOV CX,25
QK:
 CALL RETRO1
 LOOP QK
 CALL RETARDO
 MOV CX,100
QL:
 CALL RETRO2
 LOOP QL
 CALL RETARDO
;-------------------------------------------------
 CALL FRENTE5
 CALL RETARDO
 MOV CX,20
 RC:
 CALL FRENTE3
 LOOP RC
 MOV CX,100
QM:
 CALL FRENTE2
 LOOP QM
 CALL RETARDO
 MOV CX,25
QN:
 CALL FRENTE1
 LOOP QN
 CALL RETARDO
mov cx,20
 qv:
 CALL RETRO3
```

```
loop qv
CALL RETRO5
MOV CX,20
QO:
CALL RETRO1
LOOP QO
MOV CX,40
Qs:
call frente3
loop qs
CALL RETARDO
MOV CX,200
QP:
CALL FRENTE2
LOOP QP
CALL RETARDO
MOV CX,40
QR:
CALL RETRO3
LOOP QR
CALL RETARDO
CALL FRENTE4
CALL RETARDO
CALL RETRO4
CALL RETARDO
POP DX
POP CX
POP BX
POP AX
RET
```

```
MATIC ENDP

;--------------------------------------------------------------------------
FRENTE1 PROC NEAR
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV AL,10011001B
 CALL SALA
 call retardo
 CALL RETARDO
 MOV AL,10010001B
 CALL SALA
 CALL RETARDO
 call retardo
 MOV AL,10010101B
 CALL SALA
 CALL RETARDO
 call retardo
 MOV AL,10010100B
 CALL SALA
 CALL RETARDO
 call retardo
 MOV AL,10010110B
 CALL SALA
 CALL RETARDO
 call retardo
 MOV AL,10010010B
 CALL SALA
```

```
 CALL RETARDO
 call retardo
 MOV AL,10011010B
 CALL SALA
 CALL RETARDO
 call retardo
 MOV AL,10011000B
 CALL SALA
 CALL RETARDO
 call retardo
 MOV AL,10011001B
 CALL SALA
 CALL RETARDO
 call retardo
 POP DX
 POP CX
 POP BX
 POP AX
RET
FRENTE1 ENDP
;--------------------------------------------------------------------------
RETRO1 PROC NEAR
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV AL,10011001B
 CALL SALA
 CALL RETARDO
 call retardo
```

```
MOV AL,10011000B
CALL SALA
CALL RETARDO
call retardo
MOV AL,10011010B
CALL SALA
CALL RETARDO
call retardo
MOV AL,10010010B
CALL SALA
CALL RETARDO
call retardo
MOV AL,10010110B
CALL SALA
CALL RETARDO
call retardo
MOV AL,10010100B
CALL SALA
CALL RETARDO
call retardo
MOV AL,10010101B
CALL SALA
CALL RETARDO
call retardo
MOV AL,10010001B
CALL SALA
CALL RETARDO
call retardo
MOV AL,10011001B
CALL SALA
```

```
 CALL RETARDO
 call retardo
 POP DX
 POP CX
 POP BX
 POP AX
RET
RETRO1 ENDP
;--------------------------------------------------------------------------
FRENTE2 PROC NEAR
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV AL,10011001B
 CALL SALA
 CALL RETARDO
 MOV AL,00011001B
 CALL SALA
 CALL RETARDO
 MOV AL,01011001B
 CALL SALA
 CALL RETARDO
 MOV AL,01001001B
 CALL SALA
 CALL RETARDO
 MOV AL,01101001B
 CALL SALA
 CALL RETARDO
 MOV AL,00101001B
```

```
 CALL SALA
 CALL RETARDO
 MOV AL,10101001B
 CALL SALA
 CALL RETARDO
 MOV AL,10001001B
 CALL SALA
 CALL RETARDO
 MOV AL,10011001B
 CALL SALA
 CALL RETARDO
 POP DX
 POP CX
 POP BX
 POP AX
RET
FRENTE2 ENDP
;--------------------------------------------------------------------
RETRO2 PROC NEAR
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV AL,10011001B
 CALL SALA
 CALL RETARDO
 MOV AL,10001001B
 CALL SALA
 CALL RETARDO
 MOV AL,10101001B
```

```
 CALL SALA
 CALL RETARDO
 MOV AL,00101001B
 CALL SALA
 CALL RETARDO
 MOV AL,01101001B
 CALL SALA
 CALL RETARDO
 MOV AL,01001001B
 CALL SALA
 CALL RETARDO
 MOV AL,01011001B
 CALL SALA
 CALL RETARDO
 MOV AL,00011001B
 CALL SALA
 CALL RETARDO
 MOV AL,10011001B
 CALL SALA
 CALL RETARDO
 POP DX
 POP CX
 POP BX
 POP AX
RET
RETRO2 ENDP
;-------------------------------------------------------------------------
FRENTE3 PROC NEAR
 PUSH AX
 PUSH BX
```

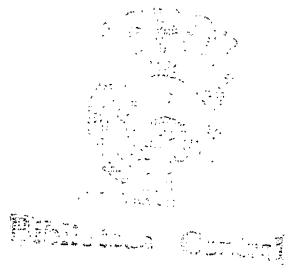
```
PUSH CX
PUSH DX
MOV AL,01101001B
CALL SALB
CALL RETARDO
MOV AL,01100001B
CALL SALB
CALL RETARDO
MOV AL,01100101B
CALL SALB
CALL RETARDO
MOV AL,01100100B
CALL SALB
CALL RETARDO
MOV AL,01100110B
CALL SALB
CALL RETARDO
MOV AL,01100010B
CALL SALB
CALL RETARDO
MOV AL,01101010B
CALL SALB
CALL RETARDO
MOV AL,01101000B
CALL SALB
CALL RETARDO
MOV AL,01101001B
CALL SALB
CALL RETARDO
POP DX
```

```
 POP CX
 POP BX
 POP AX
RET
FRENTE3 ENDP
;--------------------------------------------------------------------------
RETRO3 PROC NEAR
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV AL,01101001B
 CALL SALB
 CALL RETARDO
 MOV AL,01101000B
 CALL SALB
 CALL RETARDO
 MOV AL,01101010B
 CALL SALB
 CALL RETARDO
 MOV AL,01100010B
 CALL SALB
 CALL RETARDO
 MOV AL,01100110B
 CALL SALB
 CALL RETARDO
 MOV AL,01100100B
 CALL SALB
 CALL RETARDO
 MOV AL,01100101B
```

```
 CALL SALB
 CALL RETARDO
 MOV AL,01100001B
 CALL SALB
 CALL RETARDO
 MOV AL,01101001B
 CALL SALB
 CALL RETARDO
 POP DX
 POP CX
 POP BX
 POP AX
RET
RETRO3 ENDP
;-------------------------------------------------------------------------
FRENTE4 PROC NEAR
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV CX,400
 TEST3:
 MOV AL,10011001B
 CALL SALB
 CALL RETARDO
 LOOP TEST3
 MOV AL,00001001B
 CALL SALB
 call retardo
 POP DX
```

```
 POP CX
 POP BX
 POP AX
RET
FRENTE4 ENDP
;-------------------------------------------------------------------------
RETRO4 PROC NEAR
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV CX,400
 TEST4:
 MOV AL,00011001B
 CALL SALB
 CALL RETARDO
 LOOP TEST4
 MOV AL,00001001B
 CALL SALB
 call retardo
 POP DX
 POP CX
 POP BX
 POP AX
RET
RETRO4 ENDP
;-------------------------------------------------------------------------
FRENTE5 PROC NEAR
 PUSH AX
 PUSH BX
```

```
 PUSH CX
 PUSH DX
 MOV AL,01101001B
 CALL SALB
 CALL RETARDO
 POP DX
 POP CX
 POP BX
 POP AX
RET
FRENTE5 ENDP
;-------------------------------------------------------------------------
RETRO5 PROC NEAR
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV AL,00001001B
 CALL SALB
 call retardo
 POP DX
 POP CX
 POP BX
 POP AX
RET
RETRO5 ENDP
;-------------------------------------------------------------------------
 RETARDO PROC NEAR
 PUSH CX
 MOV CX,10
```



```
R1:
PUSH CX
MOV CX,1000
R2:
NOP
NOP
NOP
LOOP R2
POP CX
LOOP R1
POP CX
RET
RETARDO ENDP
;-------------------------------------------------------------------------
        SALB PROC NEAR
            PUSH AX
            MOV DX,37AH
            MOV AL,00001000B
            OUT DX,AL
            NOP
            POP AX
            MOV DX,378H
            OUT DX,AL
            NOP
            MOV DX,37AH
            MOV AL,00001001B
            OUT DX,AL
            NOP
            MOV AL,00001000B
            OUT DX,AL
```

```
        NOP
        RET
      SALB ENDP
;-------------------------------------------------------------------
    SALA PROC NEAR
        PUSH AX
        MOV DX,37AH
        MOV AL,00001010B
        OUT DX,AL
        NOP
        POP AX
        MOV DX,378H
        OUT DX,AL
        NOP
        MOV DX,37AH
        MOV AL,00001011B
        OUT DX,AL
        NOP
        MOV AL,00001010B
        OUT DX,AL
        NOP
        RET
      SALA ENDP
;-------------------------------------------------------------------
CODIGO ENDS
END MAINTIC
;-------------------------------------------------------------------
;-------------------------------------------------------------------
```

4.5 PROCEDIMIENTOS ESPECÍFICOS

Se han denominado procedimientos específicos a aquellos programas desarrollados en lenguaje Ensamblador, que son los encargados de enviar la información requerida para las diferentes actividades del brazo mecánico. Estos procedimientos son muy puntuales y exactos, pués interactúan directamente con la PPI-8255, y a través de esta y de las interfaces de poder, con los motores de las diferentes articulaciones, y con los efectores finales.

Estos procedimientos específicos, al igual que el procedimiento general de control automático, requieren de procedimientos especiales, los mismos que han sido desarrollados en lenguaje ensamblador y que son detallados en la sección 4.6.

4.5.1 PROCEDIMIENTO ART1D.ASM

Este procedimiento es el encargado de enviar la información necesaria para que la articulación # 1 se mueva hacia la derecha. Con cada cada ejecución de este procedimiento el motor de pasos gira 3.6°.Dependiendo del reductor mecánico utilizado para cada artiulación y del tamaño del lazo implementado en el programa anfitrión, la articulación tendrá un giro considerablemente perceptible o inperceptible.

```
;-------------------------------------------------------------------------

  TITLE ART1D
  PILA SEGMENT PARA  PUBLIC 'STACK'
        DW 10 DUP(?)
  PILA ENDS
;-------------------------------------------------------------------------
DATO SEGMENT PARA PUBLIC 'DATA'
DATO ENDS
;-------------------------------------------------------------------------
CODIGO SEGMENT PARA PUBLIC 'CODE'
;-------------------------------------------------------------------------
MAIN1D PROC FAR                 ;PROGRAMA PRINCIPAL
 ASSUME CS:CODIGO, DS:DATO,SS:PILA
 PUSH DS
 SUB AX,AX
 PUSH AX
 MOV AX,DATO
 MOV DS,AX
 CALL D1
 RET
MAIN1D ENDP
;-------------------------------------------------------------------------
D1 PROC NEAR                    ;PROCEDIMIENTO PRINCIPAL
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV AL,10011001B
 CALL SALA
 CALL RETARDO
 MOV AL,10010001B
 CALL SALA
 CALL RETARDO
```

```
MOV AL,10010101B
CALL SALA
CALL RETARDO
MOV AL,10010100B
CALL SALA
CALL RETARDO
MOV AL,10010110B
CALL SALA
CALL RETARDO
MOV AL,10010010B
CALL SALA
CALL RETARDO
MOV AL,10011010B
CALL SALA
CALL RETARDO
MOV AL,10011000B
CALL SALA
CALL RETARDO
MOV AL,10011001B
CALL SALA
CALL RETARDO
POP DX
POP CX
POP BX
POP AX
RET
D1 ENDP
;--------------------------------------------------------------------
CODIGO ENDS
END MAIN1D
```

El procedimiento DI PROC NEAR es el encargado de enviar la secuencia respectiva para que el motor gire hacia la derecha. Dicha secuencia es enviada utilizando los cuatro bits menos significativos del puerto A de la PPI-8255.

Los procedimientos RETARDO PROC NEAR y SALA PROC NEAR son procedimientos especiales.

4.5.2 PROCEDIMIENTO ART1I.ASM

Este procedimiento es el encargado de enviar la información necesaria para que la articulación # 1 se mueva hacia la izquierda. Con cada cada ejecución de este procedimiento el motor de pasos gira 3.6°.

```
;--------------------------------------------------------------------
  TITLE ART1I
  PILA SEGMENT PARA  PUBLIC 'STACK'
        DW 10 DUP(?)
  PILA ENDS
;--------------------------------------------------------------------
DATO SEGMENT PARA PUBLIC 'DATA'
DATO ENDS
;--------------------------------------------------------------------
CODIGO SEGMENT PARA PUBLIC 'CODE'
;--------------------------------------------------------------------
```

```
MAIN1I PROC FAR                    ;PROGRAMA PRINCIPAL
 ASSUME CS:CODIGO, DS:DATO,SS:PILA
 PUSH DS
 SUB AX,AX
 PUSH AX
 MOV AX,DATO
 MOV DS,AX
 CALL D1
 RET
MAIN1I ENDP
;-------------------------------------------------------------------------
D1 PROC NEAR                       ;PROCEDIMIENTO PRINCIPAL
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV AL,10011001B
 CALL SALA
 CALL RETARDO
 MOV AL,10011000B
 CALL SALA
 CALL RETARDO
 MOV AL,10011010B
 CALL SALA
 CALL RETARDO
 MOV AL,10010010B
 CALL SALA
 CALL RETARDO
 MOV AL,10010110B
 CALL SALA
```

```
CALL RETARDO
MOV AL,10010100B
CALL SALA
CALL RETARDO
MOV AL,10010101B
CALL SALA
CALL RETARDO
MOV AL,10010001B
CALL SALA
CALL RETARDO
MOV AL,10011001B
CALL SALA
CALL RETARDO
POP DX
POP CX
POP BX
POP AX
RET
D1 ENDP
CODIGO ENDS
END MAIN1I
;-------------------------------------------------------------------------
```

El procedimiento DI PROC NEAR es el encargado de enviar la secuencia respectiva para que el motor gire hacia la izquierda. Dicha secuencia es enviada utilizando los cuatro bits menos significativos del puerto A de la PPI-8255.

Los procedimientos RETARDO PROC NEAR y SALA PROC NEAR son procedimientos especiales.

4.5.3 PROCEDIMIENTO ART2D.ASM

Este procedimiento es el encargado de enviar la información necesaria para que la articulación # 2 se mueva hacia la derecha. Con cada cada ejecución de este procedimiento el motor de pasos gira 3.6°.

```
;-------------------------------------------------------------------------
  TITLE ART2D
  PILA SEGMENT PARA  PUBLIC 'STACK'
        DW 10 DUP(?)
  PILA ENDS
;-------------------------------------------------------------------------
DATO SEGMENT PARA PUBLIC 'DATA'
DATO ENDS
;-------------------------------------------------------------------------
CODIGO SEGMENT PARA PUBLIC 'CODE'
;-------------------------------------------------------------------------
MAIN2D PROC FAR                    ;PROGRAMA PRINCIPAL
 ASSUME CS:CODIGO, DS:DATO,SS:PILA
 PUSH DS
 SUB AX,AX
 PUSH AX
 MOV AX,DATO
 MOV DS,AX
 CALL D1
 RET
```

```
MAIN2D ENDP
;--------------------------------------------------------------------------
D1 PROC NEAR                    ;PROCEDIMIENTO PRINCIPAL
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV AL,10011001B
 CALL SALA
 CALL RETARDO
 MOV AL,00011001B
 CALL SALA
 CALL RETARDO
 MOV AL,01011001B
 CALL SALA
 CALL RETARDO
 MOV AL,01001001B
 CALL SALA
 CALL RETARDO
 MOV AL,01101001B
 CALL SALA
 CALL RETARDO
 MOV AL,00101001B
 CALL SALA
 CALL RETARDO
 MOV AL,10101001B
 CALL SALA
 CALL RETARDO
 MOV AL,10001001B
 CALL SALA
```

```
CALL RETARDO
MOV AL,10011001B
CALL SALA
CALL RETARDO
POP DX
POP CX
POP BX
POP AX
RET
D1 ENDP
;--------------------------------------------------------------------

CODIGO ENDS
END MAIN2D
;--------------------------------------------------------------------
```

El procedimiento DI PROC NEAR es el encargado de enviar la secuencia respectiva para que el motor gire hacia la derecha. Dicha secuencia es enviada utilizando los cuatro bits más significativos del puerto A de la PPI-8255.

Los procedimientos RETARDO PROC NEAR y SALA PROC NEAR son procedimientos especiales.

4.5.4 PROCEDIMIENTO ART2L.ASM

Este procedimiento es el encargado de enviar la información necesaria para que la articulación # 2 se mueva hacia la

izquierda. Con cada cada ejecución de este procedimiento el motor de pasos gira 3.6°.

```
;--------------------------------------------------------------------
  TITLE ART2I
  PILA SEGMENT PARA  PUBLIC 'STACK'
        DW 10 DUP(?)
  PILA ENDS
;--------------------------------------------------------------------
DATO SEGMENT PARA PUBLIC 'DATA'
DATO ENDS
;--------------------------------------------------------------------
CODIGO SEGMENT PARA PUBLIC 'CODE'
;--------------------------------------------------------------------
MAIN2I PROC FAR                    ;PROGRAMA PRINCIPAL
 ASSUME CS:CODIGO, DS:DATO,SS:PILA
 PUSH DS
 SUB AX,AX
 PUSH AX
 MOV AX,DATO
 MOV DS,AX
 CALL D1
 RET
MAIN2I ENDP
;--------------------------------------------------------------------
D1 PROC NEAR                ;PROCEDIMIENTO PRINCIPAL
 PUSH AX
 PUSH BX
 PUSH CX
```

```
PUSH DX
MOV AL,10011001B
CALL SALA
CALL RETARDO
MOV AL,10001001B
CALL SALA
CALL RETARDO
MOV AL,10101001B
CALL SALA
CALL RETARDO
MOV AL,00101001B
CALL SALA
CALL RETARDO
MOV AL,01101001B
CALL SALA
CALL RETARDO
MOV AL,01001001B
CALL SALA
CALL RETARDO
MOV AL,01011001B
CALL SALA
CALL RETARDO
MOV AL,00011001B
CALL SALA
CALL RETARDO
MOV AL,10011001B
CALL SALA
CALL RETARDO
POP DX
POP CX
```

```
 POP BX
 POP AX
 RET
D1 ENDP
;--------------------------------------------------------------------
CODIGO ENDS
END MAIN2I
;--------------------------------------------------------------------
```

El procedimiento DI PROC NEAR es el encargado de enviar la secuencia respectiva para que el motor gire hacia la izquierda. Dicha secuencia es enviada utilizando los cuatro bits más significativos del puerto A de la PPI-8255.

Los procedimientos RETARDO PROC NEAR y SALA PROC NEAR son procedimientos especiales.

4.5.5 PROCEDIMIENTO ART3D.ASM

Este procedimiento es el encargado de enviar la información necesaria para que la articulación # 3 se mueva hacia la derecha. Con cada cada ejecución de este procedimiento el motor de pasos gira 3.6°.

```
;--------------------------------------------------------------------

  TITLE ART3D
```

```
  PILA SEGMENT PARA  PUBLIC 'STACK'
        DW 10 DUP(?)
  PILA ENDS
;-------------------------------------------------------------------------
DATO SEGMENT PARA PUBLIC 'DATA'
DATO ENDS
;-------------------------------------------------------------------------
CODIGO SEGMENT PARA PUBLIC 'CODE'
;-------------------------------------------------------------------------
MAIN3D PROC FAR                ;PROGRAMA PRINCIPAL
 ASSUME CS:CODIGO, DS:DATO,SS:PILA
 PUSH DS
 SUB AX,AX
 PUSH AX
 MOV AX,DATO
 MOV DS,AX
 CALL D1
 RET
MAIN3D ENDP
;-------------------------------------------------------------------------
D1 PROC NEAR                   ;PROCEDIMIENTO PRINCIPAL
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV AL,01101001B
 CALL SALB
 CALL RETARDO
 MOV AL,01100001B
 CALL SALB
```

```
CALL RETARDO
MOV AL,01100101B
CALL SALB
CALL RETARDO
MOV AL,01100100B
CALL SALB
CALL RETARDO
MOV AL,01100110B
CALL SALB
CALL RETARDO
MOV AL,01100010B
CALL SALB
CALL RETARDO
MOV AL,01101010B
CALL SALB
CALL RETARDO
MOV AL,01101000B
CALL SALB
CALL RETARDO
MOV AL,01101001B
CALL SALB
CALL RETARDO
POP DX
POP CX
POP BX
POP AX
RET
D1 ENDP
;--------------------------------------------------------------------
CODIGO ENDS
```

```
END MAIN3D
;-------------------------------------------------------------------------
```

El procedimiento DI PROC NEAR es el encargado de enviar la secuencia respectiva para que el motor gire hacia la derecha. Dicha secuencia es enviada utilizando los cuatro bits menos significativos del puerto B de la PPI-8255.

Los procedimientos RETARDO PROC NEAR y SALB PROC NEAR son procedimientos especiales.

4.5.6 PROCEDIMIENTO ART3I.ASM

Este procedimiento es el encargado de enviar la información necesaria para que la articulación # 3 se mueva hacia la izquierda. Con cada cada ejecución de este procedimiento el motor de pasos gira 3.6°.

```
;-------------------------------------------------------------------------
  TITLE ART3I
  PILA SEGMENT PARA PUBLIC 'STACK'
        DW 10 DUP(?)
  PILA ENDS
;-------------------------------------------------------------------------
DATO SEGMENT PARA PUBLIC 'DATA'
DATO ENDS
;-------------------------------------------------------------------------
CODIGO SEGMENT PARA PUBLIC 'CODE'
```

```
;-------------------------------------------------------------------------
MAIN3I PROC FAR                    ;PROGRAMA PRINCIPAL
 ASSUME CS:CODIGO, DS:DATO,SS:PILA
 PUSH DS
 SUB AX,AX
 PUSH AX
 MOV AX,DATO
 MOV DS,AX
 CALL D1
 RET
MAIN3I ENDP
;-------------------------------------------------------------------------
D1 PROC NEAR                       ;PROCEDIMIENTO PRINCIPAL
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV AL,01101001B
 CALL SALB
 CALL RETARDO
 MOV AL,01101000B
 CALL SALB
 CALL RETARDO
 MOV AL,01101010B
 CALL SALB
 CALL RETARDO
 MOV AL,01100010B
 CALL SALB
 CALL RETARDO
 MOV AL,01100110B
```

```
CALL SALB
CALL RETARDO
MOV AL,01100100B
CALL SALB
CALL RETARDO
MOV AL,01100101B
CALL SALB
CALL RETARDO
MOV AL,01100001B
CALL SALB
CALL RETARDO
MOV AL,01101001B
CALL SALB
CALL RETARDO
POP DX
POP CX
POP BX
POP AX
RET
D1 ENDP
;-------------------------------------------------------------------------
CODIGO ENDS
END MAIN3I
;-------------------------------------------------------------------------
```

El procedimiento DI PROC NEAR es el encargado de enviar la secuencia respectiva para que el motor gire hacia la izquierda.

Dicha secuencia es enviada utilizando los cuatro bits menos significativos del puerto B de la PPI-8255.

Los procedimientos RETARDO PROC NEAR y SALB PROC NEAR son procedimientos especiales.

4.5.7 PROCEDIMIENTO ART4D.ASM

Este procedimiento es el encargado de enviar la información necesaria para que la articulación # 4 se mueva hacia la derecha. Con cada cada ejecución de este procedimiento el motor AC gira 15.0°.

```
;-------------------------------------------------------------------------
  TITLE ART4DA
  PILA SEGMENT PARA  PUBLIC 'STACK'
        DW 10 DUP(?)
  PILA ENDS
;-------------------------------------------------------------------------
DATO SEGMENT PARA PUBLIC 'DATA'
DATO ENDS
;-------------------------------------------------------------------------
CODIGO SEGMENT PARA PUBLIC 'CODE'
;-------------------------------------------------------------------------
MAIN4D PROC FAR                    ;PROGRAMA PRINCIPAL
 ASSUME CS:CODIGO, DS:DATO,SS:PILA
 PUSH DS
 SUB AX,AX
```

```
 PUSH AX
 MOV AX,DATO
 MOV DS,AX
 CALL D1
 RET
MAIN4D ENDP
;-------------------------------------------------------------------------
D1 PROC NEAR                ;PROCEDIMIENTO PRINCIPAL
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV AL,10010000B
 CALL SALB
 CALL RETARDO
 POP DX
 POP CX
 POP BX
 POP AX
 RET
D1 ENDP
;-------------------------------------------------------------------------
CODIGO ENDS
END MAIN4D

;-------------------------------------------------------------------------
```

El procedimiento DI PROC NEAR es el encargado de enviar la palabra respectiva para que el motor gire hacia la derecha. Dicha

palabra es enviada utilizando los cuatro bits más significativos del puerto B de la PPI-8255.

Los procedimientos RETARDO PROC NEAR y SALB PROC NEAR son procedimientos especiales.

4.5.8 PROCEDIMIENTO ART4I.ASM

Este procedimiento es el encargado de enviar la información necesaria para que la articulación # 4 se mueva hacia la izquierda. Con cada cada ejecución de este procedimiento el motor AC gira 15.0°.

```
;-------------------------------------------------------------------------
  TITLE ART4I
  PILA SEGMENT PARA  PUBLIC 'STACK'
        DW 10 DUP(?)
  PILA ENDS
;-------------------------------------------------------------------------
DATO SEGMENT PARA PUBLIC 'DATA'
DATO ENDS
;-------------------------------------------------------------------------
CODIGO SEGMENT PARA PUBLIC 'CODE'
;-------------------------------------------------------------------------
MAIN4I PROC FAR                    ;PROGRAMA PRINCIPAL
 ASSUME CS:CODIGO, DS:DATO,SS:PILA
 PUSH DS
 SUB AX,AX
```

```
 PUSH AX
 MOV AX,DATO
 MOV DS,AX
 CALL D1
 RET
MAIN4I ENDP
;--------------------------------------------------------------------
D1 PROC NEAR                ;PROCEDIMIENTO PRINCIPAL
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV AL,00010000B
 CALL SALB
 CALL RETARDO
 POP DX
 POP CX
 POP BX
 POP AX
 RET
D1 ENDP
;--------------------------------------------------------------------
CODIGO ENDS
END MAIN4I
;--------------------------------------------------------------------
```

El procedimiento DI PROC NEAR es el encargado de enviar la palabra respectiva para que el motor gire hacia la izquierda. Dicha palabra es enviada utilizando los cuatro bits más significativos del puerto B de la PPI-8255.

Los procedimientos RETARDO PROC NEAR y SALB PROC NEAR son procedimientos especiales.

4.5.9 PROCEDIMIENTO ART5D.ASM

Este procedimiento es el encargado de enviar la información necesaria para que el efector final # 1 se gire hacia la derecha. Con cada cada ejecución de este procedimiento el motor DC (taladro) gira 5.0 revoluciones aproximadamente.

```
;-------------------------------------------------------------------------
  TITLE ART5D
  PILA SEGMENT PARA  PUBLIC 'STACK'
        DW 10 DUP(?)
  PILA ENDS
;-------------------------------------------------------------------------
DATO SEGMENT PARA PUBLIC 'DATA'
DATO ENDS
;-------------------------------------------------------------------------
CODIGO SEGMENT PARA PUBLIC 'CODE'
;-------------------------------------------------------------------------
MAIN5D PROC FAR                    ;PROGRAMA PRINCIPAL
 ASSUME CS:CODIGO, DS:DATO,SS:PILA
 PUSH DS
 SUB AX,AX
 PUSH AX
 MOV AX,DATO
 MOV DS,AX
 CALL D1
 RET
MAIN5D ENDP
```

```
;-------------------------------------------------------------------
D1 PROC NEAR                ;PROCEDIMIENTO PRINCIPAL
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV AL,01100000B
 CALL SALB
 CALL RETARDO
 POP DX
 POP CX
 POP BX
 POP AX
 RET
D1 ENDP
;-------------------------------------------------------------------
CODIGO ENDS
END MAIN5D
;-------------------------------------------------------------------
```

El procedimiento DI PROC NEAR es el encargado de enviar la palabra respectiva para que el efector final gire hacia la derecha. Dicha palabra es enviada utilizando los cuatro bits más significativos del puerto B de la PPI-8255.

Los procedimientos RETARDO PROC NEAR y SALB PROC NEAR son procedimientos especiales.

4.5.10 PROCEDIMIENTO ART5I.ASM

Este procedimiento es el encargado de enviar la información necesaria para que el efector final # 1 se mueva hacia la izquierda. Con cada cada ejecución de este procedimiento el motor DC gira 5.0 revoluciones aproximadamente.

```
;--------------------------------------------------------------------------
  TITLE ART5I
  PILA SEGMENT PARA  PUBLIC 'STACK'
        DW 10 DUP(?)
  PILA ENDS
;--------------------------------------------------------------------------
DATO SEGMENT PARA PUBLIC 'DATA'
DATO ENDS
;--------------------------------------------------------------------------
CODIGO SEGMENT PARA PUBLIC 'CODE'
;--------------------------------------------------------------------------
MAIN5I PROC FAR                 ;PROGRAMA PRINCIPAL
 ASSUME CS:CODIGO, DS:DATO,SS:PILA
 PUSH DS
 SUB AX,AX
 PUSH AX
 MOV AX,DATO
 MOV DS,AX
 CALL D1
 RET
MAIN5I ENDP
;--------------------------------------------------------------------------
D1 PROC NEAR                    ;PROCEDIMIENTO PRINCIPAL
 PUSH AX
 PUSH BX
```

```
 PUSH CX
 PUSH DX
 MOV AL,00100000B
 CALL SALB
 CALL RETARDO
 POP DX
 POP CX
 POP BX
 POP AX
 RET
D1 ENDP
;--------------------------------------------------------------------------
CODIGO ENDS
END MAIN5I
;--------------------------------------------------------------------------
```

El procedimiento DI PROC NEAR es el encargado de enviar la palabra respectiva para que el efector final gire hacia la izquierda. Dicha palabra es enviada utilizando los cuatro bits más significativos del puerto B de la PPI-8255.

Los procedimientos RETARDO PROC NEAR y SALB PROC NEAR son procedimientos especiales.

4.5.11 PROCEDIMIENTO ART6C.ASM

Este procedimiento es el encargado de enviar la información necesaria para que el efector final # 2 coja un objeto, y no lo suelte hasta que el programa hací lo decida.

```
;--------------------------------------------------------------------
  TITLE ART6C
  PILA SEGMENT PARA  PUBLIC 'STACK'
        DW 10 DUP(?)
  PILA ENDS
;--------------------------------------------------------------------
DATO SEGMENT PARA PUBLIC 'DATA'
DATO ENDS
;--------------------------------------------------------------------
CODIGO SEGMENT PARA PUBLIC 'CODE'
;--------------------------------------------------------------------
MAIN6C PROC FAR                ;PROGRAMA PRINCIPAL
 ASSUME CS:CODIGO, DS:DATO,SS:PILA
 PUSH DS
 SUB AX,AX
 PUSH AX
 MOV AX,DATO
 MOV DS,AX
 CALL D1
 RET
MAIN6C ENDP
;--------------------------------------------------------------------
D1 PROC NEAR                   ;PROCEDIMIENTO PRINCIPAL
 PUSH AX
 PUSH BX
 PUSH CX
```

```
PUSH DX
MOV AL,01100000B
CALL SALB
CALL RETARDO
POP DX
POP CX
POP BX
POP AX
RET
D1 ENDP
;--------------------------------------------------------------------------
CODIGO ENDS
END MAIN6C
;--------------------------------------------------------------------------
```

El procedimiento DI PROC NEAR es el encargado de enviar la palabra respectiva para que el efector final agarre un objeto. Dicha palabra es enviada utilizando los cuatro bits más significativos del puerto B de la PPI-8255.

Los procedimientos RETARDO PROC NEAR y SALB PROC NEAR son procedimientos especiales.

4.5.12 PROCEDIMIENTO ART6S.ASM

Este procedimiento es el encargado de enviar la información necesaria para que el efector final # 2 suelte un objeto.

```
;--------------------------------------------------------------------------
```

```
  TITLE ART6S
  PILA SEGMENT PARA  PUBLIC 'STACK'
        DW 10 DUP(?)
  PILA ENDS
;-------------------------------------------------------------------------
DATO SEGMENT PARA PUBLIC 'DATA'
DATO ENDS
;-------------------------------------------------------------------------
CODIGO SEGMENT PARA PUBLIC 'CODE'
;-------------------------------------------------------------------------
MAIN6S PROC FAR                  ;PROGRAMA PRINCIPAL
 ASSUME CS:CODIGO, DS:DATO,SS:PILA
 PUSH DS
 SUB AX,AX
 PUSH AX
 MOV AX,DATO
 MOV DS,AX
 CALL D1
 RET
MAIN6S ENDP
;-------------------------------------------------------------------------
D1 PROC NEAR                     ;PROCEDIMIENTO PRINCIPAL
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV AL,00000000B
 CALL SALB
 CALL RETARDO
 POP DX
```

```
POP CX
POP BX
POP AX
RET
D1 ENDP
;--------------------------------------------------------------------
CODIGO ENDS
END MAIN6S
;--------------------------------------------------------------------
```


El procedimiento DI PROC NEAR es el encargado de enviar la palabra respectiva para que el efector final suelte un objeto. Dicha palabra es enviada utilizando los cuatro bits más significativos del puerto B de la PPI-8255.

Los procedimientos RETARDO PROC NEAR y SALB PROC NEAR son procedimientos especiales.

4.6 PROCEDIMIENTOS ESPECIALES

Dentro de los procedimientos genetrales y específicos, se utilizan procedimientos especiales, llamados así en primer lugar por que son comunes para la totalidad de los procedimientos específicos y para el procedimiento automático, y e segundo lugar porque através de estos se

controlan a la PPI-8255, simulando las señales que normalmente el microprocesador envía a este periférico.

4.6.1 PROCEDIMIENTO PPI.ASM

Este es el más importante de los procedimientos especiales, pués mediante este procedimiento se consigue programar a la interface periférica, simulando las señales del microprocesador: WR, A1 y A0. De esta manera se logra programar al 8255 para que trabaje en modo "0".

```
;--------------------------------------------------------------------------
  TITLE PPI
  PILA SEGMENT PARA  PUBLIC 'STACK'
        DW 10 DUP(?)
  PILA ENDS
;--------------------------------------------------------------------------
DATO SEGMENT PARA PUBLIC 'DATA'
DATO ENDS
;--------------------------------------------------------------------------
CODIGO SEGMENT PARA PUBLIC 'CODE'
;--------------------------------------------------------------------------
MAINPPI PROC FAR                    ;PROGRAMA PRINCIPAL
 ASSUME CS:CODIGO, DS:DATO,SS:PILA
 PUSH DS
 SUB AX,AX
 PUSH AX
```

```
MOV AX,DATO
MOV DS,AX
CALL UNO
RET
MAINPPI ENDP
;--------------------------------------------------------------------------
        UNO PROC NEAR
        PUSH AX
        PUSH BX
        PUSH CX
        PUSH DX
        MOV DX,37AH
        MOV AL,00000100B
        MOV AH,0
        OUT DX,AL
        nop
        CALL RETARDO
        MOV AL,00001101B
        OUT DX,AL
        nop
        CALL RETARDO
        MOV DX,378H
        MOV AL,10000000B
```

```
        OUT DX,AL
        nop
        CALL RETARDO
        MOV DX,37AH
        MOV AL,00001100B
        OUT DX,AL
        nop
        CALL RETARDO
        POP DX
        POP CX
        POP BX
        POP AX
        RET
        UNO ENDP
;-------------------------------------------------------------------------
CODIGO END
END MAINPPI
;-------------------------------------------------------------------------
```

El procedimiento UNO PROC NEAR es el encargado de enviar los pulsos de simulación a través del puerto paralelo.

Este procedimiento especial es ejecutado desde el programa principal justo en el instante antes de entrar al menú principal del programa anfitrión.

4.6.2 PROCEDIMIENTO SALA.ASM

Este procedimiento es el encargado de configurar a la PPI- 8255 para que el dato que va a recibir desde el puerto paralelo, lo escriba en el puerto A.

```
;--------------------------------------------------------------------------
    SALA PROC NEAR
        PUSH AX
        MOV DX,37AH
        MOV AL,00001010B
        OUT DX,AL
        nop
        nop
        nop
        POP AX
        MOV DX,378H
        OUT DX,AL
        nop
        nop
        nop
        MOV DX,37AH
        MOV AL,00001011B
        OUT DX,AL
        nop
```

```
        nop
        nop
        MOV AL,00001010B
        OUT DX,AL
        nop
        nop
        nop
        RET
    SALA ENDP
;-------------------------------------------------------------------------
```

Como se puede notar en el código de este procedimiento, también aquí se simulan las señales del microprocesador.

4.6.3 PROCEDIMIENTO SALB.ASM

Este procedimiento es el encargado de configurar a la PPI- 8255 para que el dato que va a recibir desde el puerto paralelo, lo escriba en el puerto B.

```
;-------------------------------------------------------------------------
    SALB PROC NEAR
        PUSH AX
        MOV DX,37AH
        MOV AL,00001000B
        OUT DX,AL
        NOP
        NOP
```

```
NOP
POP AX
MOV DX,378H
OUT DX,AL
NOP
NOP
NOP
MOV DX,37AH
MOV AL,00001001B
OUT DX,AL
NOP
NOP
NOP
MOV AL,00001000B
OUT DX,AL
NOP
NOP
NOP
RET
SALB ENDP
;---------------------------------------------------------------
```

Como se puede notar en el código de este procedimiento, también aquí se simulan las señales del microprocesador.

4.6.4 PROCEDIMIENTO RETARDO.ASM

Este procedimiento es el encargado de proporcionar los tiempos de retardo necesarios en ciertas operaciones del

microprocesador, y principalmente utilizados entre el envío de una palabra y otra durante la ejecución de las secuencias de funcionamiento de los motores.

```
;-------------------------------------------------------------------------
 RETARDO PROC NEAR
 PUSH CX
 MOV CX,10
 R1:
 PUSH CX
 MOV CX,1000
 R2:
 NOP
 NOP
 LOOP R2
 POP CX
 LOOP R1
 POP CX
 RET
RETARDO ENDP
;-------------------------------------------------------------------------
```

Como se puede notar con la implementación de un doble lazo, podemos controlar facilmente el tiempo de retardo de acuerdo a la rapidez con que se quiera mover a los motores ó de acuerdo al microprocesador que se utilice.

CAPITULO V

MANUAL DEL USUARIO

5.1 OBJETIVO

Con fin de proporcionar una guía para el normal funcionamiento del sistema desarrollado, se ha creído conveniente detallar algunas reglas y normas para la operación del mismo.

Entre otros aspectos, en este capítulo se detallan: Requisitos necesarios para usar el programa, reglas de funcionamiento, guía para el control manual de trabajo, guía para el control automático de trabajo, apuntes acerca de la operación del programa, condiciones de falla, solución de fallas.

Además se proporciona reglas para el montaje y desmontaje tanto del sistema eléctrico como del sistema mecánico.

5.2 REQUISITOS NECESARIOS PARA USAR EL PROGRAMA.

Entre los aspectos más importantes que deben coinsiderarse para utilizar el programa controlador del sistema denominado TORTUGA.EXE, se anota lo siguiente:

- Utilizar cualquier computador con microprocesador INTEL.

- Se recomienda utilizar un monitor a color para aprovechar la interfase gráfica que el programa ofrece al usuario.

- Para conectar la computadora con el brazo mecánico es preciso utilizar la interface de entrada salida proporcionada, puesto que el protocolo establecido es propio y único para este proyecto.

- El puerto paralelo de la máquina a utilizarse debe estar configurado con la dirección 378H.

- Se recomienda tener al alcance todos los archivos auxiliares, para el supuesto caso que se requiera la edición de los mismos.

- En caso de requerir una nueva edición tanto del programa principal como de los auxiliares se necesita contar con la ayuda del compilador Turbo C++, además de los archivos TASM.EXE, y TLINK.EXE para ensamblar y encadenar estos archivos.

5.3 REGLAS DE FUNCIONAMIENTO

Para empezar a operar con el brazo mecánico deben tenerse en cuenta las siguiente normas:

- Revisar la conección entre el computador y el brazo mecánico.

- Encender el computador y cargar el programa TORTUGA.EXE desde el indicador de mandatos del DOS.

- Alimentar la fuente de poder del brazo mecánico.(110 VAC).

- Presionar la tecla ENTER para cargar el menú principal del programa, en este momento el sistema está bajo el control del computador.

- Energizar el sistema eléctrico del brazo con el botón rojo ubicado el la parte frontal de la base del brazo mecánico.

Ahora el usuario puede hacer uso de cualquiera de las opciones disponibles en los diferentes menús del programa controlador del sistema. Las normas para el uso de estas opciones se describen más adelante, y además se las encuentra en la opción de ayuda del menú principal.

5.4 CONTROL MANUAL DE TRABAJO

Si se requiere de este modo de operación, se deben considerar los siguientes aspectos:

- Del menú principal debe seleccionarse la opción "MANUAL", la misma que se iluminará al ser seleccionada, presionando la tecla ENTER se validará dicha opción.

- Una vez seleccionada esta opción se abrirá un submenú de opciones, que le permite seleccionar entre operar con cualquiera de las articulaciones ó efectores finales del brazo. Con las teclas de dirección Ud. Puede seleccionar cualquiera de las opciones disponibles, la misma que será validada cuando presione la tecla ENTER. Debe tomarse en cuenta que al presionar ENTER será validada la opción que se encuentre iluminada en ese momento. Si se desea salir de este menú se debe presionar la tecla ESC, opción con la cual el programa se ubica en el menú principal.

- Si se ha validado cualquiera de las opciones del submenú "MANUAL", se activará un submenú de ejecución que le permitirá seleccionar entre uno y otro sentido de giro o entre una u otra actividad, dependiendo del efector final seleccionado. Igualmente en este caso se validará la opción escogida presionanmdo la tecla

ENTER , o a su vez se puede abandonar este submenú presionando la tecla ESC, opción con la cual el programa se ubica en el submenú "MANUAL".

- Cuando se opte por utilizar la articulación # 4 se debe tener en cuenta que para realizar cualquier otra actividad, debe volverse a energizar el circuito.

5.5 CONTROL AUTOMATICO DE TRABAJO.

Para trabajar en el modo automático, el usuario solamente debe mover el brazo hacia una posición de partida considerando las siguientes recomendaciones:

- Manualmente deberá posicionarse en el extremo izquierdo al frente de la cara principal de la base, con la punta del efector final (electroimán) despegada un centímetro de la mesa de trabajo, aquí estará el primer objeto que será transportado por el brazo mecánico.

- Deberán ubicarse tres obetos más, a la dercha de la posición inicial del brazo y separados cinco centímetros entre si. La posición de dichos objetos deberá formar un arco de circunferecia considerablemente abierto.

Habiendo concluido con los pasos anteriores se procede a validar la opción "AUTOMATICO", presionando la tecla ENTER, el programa se encargará de guiar al brazo para que recorra los cuatro objetos de uno en uno y los deposite en un lugar determinado, luego el brazo mueve sus cuatro articulaciones, terminando así la secuencia automática de trabajo.

Cabe anotarse que esta secuencia ha sido editada de acuerdo al criterio particular de los autores de este proyecto. Más adelante se indicará como cambiar la secuencia automática a criterio del usuario.

5.6 EDICION DE NUEVAS SECUENCIAS DE TRABAJO

El usuario puede editar su propia secuencia de trabajo para incluirla en el control automático de trabajo, para lo cual debe considerarse las siguentes recomendaciones:

- El usuario debe poseer los conocimientos básicos de programación en lenguaje ensamblador.

- Se debe editar el programa MAINAU.ASM y dentro de este hacer los cambios que se creyeren convenientes en el procedimiento MATIC PROC NEAR, modificando lazos y llamados a procedimientos de la manera que se desee hacer.

- Una vez editado el programa MAINAU.ASM se procede a ensamblarlo y encadenarlo, para luego compilar el programa TORTUGA.C y de esta manera actualizar los cambios realizados en el sistema.

5.7 APUNTES ACERCA DE LA OPERACION DEL PROGRAMA

En esta sección se proporcionan recomendaciones para modificar la velocidad de operación del brazo mecánico y para trabajar con microprocesadores de distintas velocidades.

5.7.1 CAMBIO DE TIEMPOS DE RETARDO

En el supuesto caso que se desee modificar la velocidad de operación del brazo mecánico, se puede recurrir a la opción de activar o desactivar el turbo de la máquina utilizada, si esta lo permite. Otra manera de lograrlo es editando los procedimientos de retardo, ya sea en los programas auxiliares o en el programa principal, en el caso de los programas auxiliares, todos cuentan con un procedimiento de retardo, al mismo que se le pueden modificar los valores de las variables que contienen el número de repeticiones de operaciones de espera que realiza el microprocesador. Para realizar dichos cambios se deberá editar

cada uno de los programas auxiliares, para luego ensamblarlos y encadenarlos debidamente.

5.7.2 OPERCION CON DISTINTOS MICROPROCESADORES

En vista de que el sistema puede trabajar con distintos computadores que cumplan con los requerimientos básicos de funcionamiento, puede presentarse el caso de trabajar con microprocesadores mucho más rápidos o lentos, frente a lo cual el usuario debe adecuar el programa subiendo o bajando los tiempos de retardo según sea el caso.

En el caso de usar microprocesadores muy rápidos deberá aumentarse los tiempos de retardo a tal punto que el programa proporcione velocidades razonables al sistema mecánico. Si el microprocesador es muy lento deben recortarse los tiempos de retardo. La edición de los tiempos de retardo se hará siguiendo las recomendaciones mencionadas en la sección 5.7.1.

5.8 CONDICIONES DE FALLA

Durante la operación o funcionamiento del sistema pueden presentarse algunas falla, algunas de ellas se las menciona a continuación:

- Al energizar por primera vez el circuito del brazo puede darse el caso de que la fuente de poder no arranque. Igualmente la fuente se apagará cuando se utilice la articulación # 4, esto no es precisamente una falla sino que obedece a una orden del programa.

- Al conectar la alimentación de la fuente, si no se han seguido los pasos recomendados, puede ser que la articulación # 4 entre en funcionamiento indeseadamente.

- Si se cambia de efector final y no se utiliza el programa adecuado el sistema trabajará de manera incorrecta.

5.9 SOLUCION DE FALLAS

Ante el supuesto caso de que sucedan las fallas antes mencionadas, en esta sección se ofrece la solución para cada una de ellas.

- Si al energizar por primera vez el circuito, la fuente no arranca, deberá ordenarse desde el computador cualquier actividad o movimiento del brazo, excepto la activación de la articulación # 4, para luego proceder a arrancar la fuente el número de intentos que sean necesarios.

- Si la articulación # 4 se mueve involuntariamente, inmediatamente deberá cargarse el programa de control, de esta manear la

articulación dejará de moverse, o a su vez deberá quitarse la alimentación de la fuente para evitar que los cables del efector final se enreden.

- Deberá utilizarse el programa correspondiente de acuerdo al efector final que se desee utilizar, esto es TORTUGA!, para el caso del efector final # 1 (herramienta eléctrica), TORTUGA para el caso del efector final # 2 (electroimán).

CONCLUSIONES Y RECOMENDACIONES

Habiendo concluido el diseño y construcción del brazo mecánico, objeto de esta tesis, se pueden anotar diferentes aspectos, experiencias y recomendaciones que deben considerarse en el análisis de este trabajo.

El sistema electrónico de control del brazo puede trabajar a diferentes velocidades, tan bajas o altas como el usuario desee y de acuerdo al tipo de microprocesador que utiliza, pero, existe una gran limitante en el sistema, esta es, la lenta respuesta de los componentes mecánicos. Es conocido que la respuesta de los sistemas mecánicos es mucho más lenta que la de los eléctricos, por esto, la respuesta del brazo al movimiento es lenta, este deficiencia puede ser corregida con el uso de motores DC conjuntamente con reductores mecánicos, cuya relación de amplificación de torque sea suficiente para la aplicación en particular, para de esta manera lograr elevar la velocidad del sistema sin perder torque. Cabe anotar que el sistema de control del brazo para el manejo de motores DC, debería ser modificado, lo que elevaría su complejidad y costo al tener que usar sistemas de retroalimentación.

En este proyecto, la interfase entre el programa y el usuario es muy amigable, pero podría ser mejorada a futuro utilizando periféricos tales como mouse, joysticks, sistemas de control remoto, grabado automático de secuencias e incluso se puede independizar al brazo del computador, para lo que se

necesitaría una unidad de control adjunta, cuyo corazón bien podría ser un microprocesador de bajo costo que cumpla las funciones requeridas. Todas estas sugerencias son válidas para el mejoramiento a futuro del sistema, y constituyen una base sobre la cual pueden trabajar otras personas interesadas.

Luego de realizar un análisis económico, se ha comparado el costo de la construcción de este proyecto en este país frente a la posibilidad de importar un brazo mecánico similar, se puede concluir que definitivamente resulta más barato construirlo aquí y, más importante que esto, queda la satisfacción de haber utilizado nuestros propios medios, conocimientos y recursos, así como de tener la oportunidad de incursionar en un campo de desarrollo tan vertiginoso y diverso como el de la robótica.

No esta demás recomendar a la ESPOL, a través de la FIEC, la implementación de un Laboratorio de Robótica paralelamente con el dictado de una materia que brinde el fundamento teórico, para de esta manera promover el desarrollo de esta disciplina en nuestro medio.

APENDICE

A

LISTADO DEL PROGRAMA

```
/* PROGRAMA PRINCIPAL */
/* Librerias utilizadas */
#include <stdio.h>
#include <string.h>
#include <dos.h>
#include <conio.h>
#include <process.h>

/* Constantes utilizadas */
#define NUMERO_BARRAS 16
#define ESC 27
#define ENTER 13
#define DERECHA 77
#define IZQUIERDA 75
#define ARRIBA 72
#define ABAJO 80
#define MENU 0
#define SUBMENU 1
#define EJECUCION 2

/* Estructura para las barras */
struct barra
{
   int xini, xfin, y;
   char *titulo;
};

/* Variables globales */
struct barra barras[NUMERO_BARRAS] = {
                                { 2, 24, 4, "   M A N U A L   " },
                                { 29, 51, 4, "A U T O M A T I C O" },
                                { 56, 78, 4, "    A Y U D A    " },
                                { 2, 24, 8, " ARTICULACION 1  " },
                                { 2, 24, 11, " ARTICULACION 2  " },
                                { 2, 24, 14, " ARTICULACION 3  " },
                                { 2, 24, 17, " ARTICULACION 4  " },
```

```
                                { 2, 24, 20, "  EFECTO FINAL 1  " },
                                { 2, 24, 23, "  EFECTO FINAL 2  " },
                                { 56, 78, 8, "   ACERCA DE ...   " },
                                { 56, 78, 11, "   AYUDA GENERAL   " },
                                { 56, 78, 14, "    TERMINAR     " },
                                { 29, 51, 11, "   GIRO DERECHA   " },
                                { 29, 51, 14, "  GIRO IZQUIERDA  " },
                                { 29, 51, 20, "    C O G E R     " },
                                { 29, 51, 23, "   S O L T A R    " }
                                  };
int color_texto = BLACK;
int color_fondo = WHITE;
int color_marca = LIGHTGREEN;
int color_back = CYAN;

void MAINSEN(void);
void MAINAU(void);
int SENSO;

union REGS regist;

/* Funcion que dibuja una ventana */
void ventana(int xini, int yini, int xfin, int yfin, int color_borde, int
color_fondo)
{
   static int i, j;

   textcolor(color_borde);
   textbackground(color_fondo);
   gotoxy(xini,yini);
   cprintf("É");
   gotoxy(xfin,yini);
   cprintf("»");
   gotoxy(xini,yfin);
   cprintf("È");
   gotoxy(xfin,yfin);
   cprintf("¼");
   for(i = xini + 1; i < xfin; i++)
   {
        gotoxy(i,yini);
        cprintf("Í");
        gotoxy(i,yfin);
        cprintf("Í");
   }
   for(i = yini + 1; i < yfin; i++)
   {
```

```
        gotoxy(xini,i);
        cprintf("º");
        gotoxy(xfin,i);
        cprintf("º");
    }
    for(i = xini + 1; i < xfin; i++)
        for(j = yini + 1; j < yfin; j++)
        {
            gotoxy(i,j);
            cprintf(" ");
        }
}

/* Funcion que recorre el nombre del alumno a traves de la pantalla */
void recorre(char *s, int xini, int x, int y)
{
    static int i, len;
    static char aux[30], temp[30];

    aux[0] = '\0';
    temp[0] = '\0';
    len = strlen(s);
    for(i = len - 1; i > 0; i--)
    {
        aux[0] = s[i];
        aux[1] = '\0';
        strcat(aux,temp);
        strcpy(temp,aux);
        gotoxy(xini,y);
        cprintf("%s",temp);
        delay(100);
    }
    strcpy(temp," ");
    strcat(temp,s);
    for(i = xini; i < x; i++)
    {
      gotoxy(i,y);
      cprintf("%s",temp);
      delay(100);
    }
}

/* Funcion que crea la pantalla de presentacion */
void presentacion(void)
{
    char *cadena1 = "ESCUELA SUPERIOR POLITECNICA DEL LITORAL";
```

```
    char *cadena2 = "FIEC";
    char *cadena3 = "PROYECTO DE GRADO";
    char *cadena4 = "TEMA DEL PROYECTO:";
    char *cadena5 = "BRAZO MECANICO";
    char *cadena6 = "PROFESOR DIRECTOR:";
    char *cadena7 = "Ing. Hugo Villavicencio Villavicencio";
    char *cadena8 = "ALUMNOS INTEGRANTES:";
    char *cadena9 = "Guayaquil, Noviembre de 1997";
    char *nombre1 = "Pablo Germ n Parra Rosero";
    char *nombre2 = "Adrian Oswaldo Arce Bastidas";

    clrscr();
    ventana(5,2,75,23,LIGHTCYAN,BROWN);
    textcolor(BLACK);
    gotoxy(21,4);
    cprintf("%s",cadena1);
    gotoxy(39,6);
    cprintf("%s",cadena2);
    gotoxy(32,8);
    cprintf("%s",cadena3);
    gotoxy(7,10);
    cprintf("%s",cadena4);
    gotoxy(7,13);
    cprintf("%s",cadena6);
    gotoxy(7,16);
    cprintf("%s",cadena8);
    gotoxy(27,21);
    cprintf("%s",cadena9);
    textcolor(BLUE);
    gotoxy(27,11);
    cprintf("%s",cadena5);
    gotoxy(27,14);
    cprintf("%s",cadena7);
    recorre(nombre1,6,27,17);
    recorre(nombre2,6,27,18);
    textbackground(BLACK);
    textcolor(WHITE);
    gotoxy(19,24);
    cprintf("Presione una tecla para iniciar el programa.");
    getch();
}

/* Funcion que crea una barra */
void llena_barra(struct barra bar, int color_fondo, int color_letras)
{
    textcolor(color_letras);
```

```
   textbackground(color_fondo);
   gotoxy(bar.xini,bar.y - 1);
   cprintf("ÚÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ¿");
   gotoxy(bar.xini,bar.y);
   cprintf("³%s³",bar.titulo);
   gotoxy(bar.xini,bar.y + 1);
   cprintf("ÀÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÙ");
}

/* Funcion que crea el menu principal */
void menu_inicial(void)
{
/* AQUI SE VA A MODIFICAR LA PANTALLA*/
        textbackground(color_back);

   clrscr();
   textcolor(BLACK);
   textbackground(BROWN);
   gotoxy(26,1);
   cprintf(" B R A Z O   M E C A N I C O ");
   llena_barra(barras[0],color_marca,BLACK);
   llena_barra(barras[1],color_fondo,BLACK);
   llena_barra(barras[2],color_fondo,BLACK);
}

/* Funcion que crea un submenu del menu principal */
void submenu(int menor, int mayor)
{
   static int i;

   llena_barra(barras[menor],color_marca,BLACK);
   for(i = menor + 1; i <= mayor; i++)
       llena_barra(barras[i],color_fondo,BLACK);
}

/* Funcion que elimina un submenu */
void borra_submenu(int menor, int mayor)
{
   static int i;

   for(i = menor; i <= mayor; i++)
       llena_barra(barras[i],color_back,color_back);
}

/* Funcion que crea la pantalla Acerca De */
void acerca_de(void)
```

```
{
   static char c;

   ventana(10,8,70,25,WHITE,LIGHTBLUE);
   textcolor(BLACK);
   gotoxy(34,9);
   cprintf("Acerca De ...");
   gotoxy(34,10);
   cprintf("_____");
   textcolor(WHITE);
   gotoxy(14,12);
   cprintf("La responsabilidad por los hechos, ideas y");
   gotoxy(14,13);
   cprintf("doctrinas expuestas en esta Tesis, nos corresponden");
   gotoxy(14,14);
   cprintf("exclusivamente; y, el patrimonio intelectual de la");
   gotoxy(14,15);
   cprintf("misma, a la ESCUELA SUPERIOR POLITECNICA DEL
LITORAL.");
   gotoxy(14,17);
   cprintf("(Reglamento de Ex menes y T¡tulos Profesionales de");
   gotoxy(14,18);
   cprintf("la ESPOL)");
   gotoxy(14,20);
   cprintf("ADRIAN ARCE B.                    PABLO PARRA R.");
   gotoxy(35,24);
   cprintf("Salir:");
   textcolor(LIGHTRED);
   gotoxy(42,24);
   cprintf("ESC");
   do
   {
        c=getch();
   }
   while(c != ESC);
   ventana(10,8,70,25,color_back,color_back);
}

/* Funcion que crea la pantalla de Ayuda */
void ayuda(void)
{
   static char c;

   ventana(10,8,70,25,WHITE,LIGHTBLUE);
   textcolor(BLACK);
   gotoxy(34,9);
```

```
    cprintf("Ayuda General");
    gotoxy(34,10);
    cprintf("_____");
    textcolor(WHITE);
    gotoxy(14,12);
    cprintf("Para escoger una opci¢n utilice las teclas de direc-");
    gotoxy(14,13);
    cprintf("cionamiento y luego presione ENTER.");
    gotoxy(14,15);
    cprintf("Las opciones disponibles son las siguientes:");
    gotoxy(14,17);
    cprintf("MANUAL: Al escoger esta opci¢n se presentar  un men£");
    gotoxy(14,18);
    cprintf("       de opciones para escoger la parte del BRAZO");
    gotoxy(14,19);
    cprintf("       MECANICO que quiera moverse, en forma manual.");
    gotoxy(14,21);
    cprintf("AUTOMATICO: Esta opci¢n permite que el BRAZO
MECANICO");
    gotoxy(14,22);
    cprintf("          funcione en forma autom tica.");
    textcolor(BLACK);
    gotoxy(20,24);
    cprintf("Siguiente:");
    textcolor(LIGHTRED);
    gotoxy(31,24);
    cprintf("ENTER");
    textcolor(BLACK);
    gotoxy(50,24);
    cprintf("Salir:");
    textcolor(LIGHTRED);
    gotoxy(57,24);
    cprintf("ESC");
    do
    {
        c=getch();
    }
    while((c != ESC) && (c != ENTER));
    if(c == ENTER)
    {
        textcolor(WHITE);
        gotoxy(14,12);
        cprintf("AYUDA: Al escoger esta opci¢n se presentar  un men£  ");
        gotoxy(14,13);
        cprintf("       de opciones que dan informaci¢n general del   ");
        gotoxy(14,14);
```

```
        cprintf("     programa.                          ");
        gotoxy(14,15);
        cprintf("                                        ");
        gotoxy(14,16);
        cprintf("ARTICULACION n: (n puede ser 1,2,3, o 4) Al escoger  ");
        gotoxy(14,17);
        cprintf("            esta opci¢n se presentar  2 opciones,");
        gotoxy(14,18);
        cprintf("            que permitir  mover esa articulaci¢n.");
        gotoxy(14,19);
        cprintf("                                        ");
        gotoxy(14,20);
        cprintf("EFECTOR FINAL 1: Al escoger esta opci¢n el usuario   ");
        gotoxy(14,21);
        cprintf("            podr  escoger la direcci¢n de giro  ");
        gotoxy(14,22);
        cprintf("            del taladro.                ");
    }
    do
    {
        c=getch();
    }
    while((c != ESC) && (c != ENTER));
    if(c == ENTER)
    {
        textcolor(WHITE);
        gotoxy(14,12);
        cprintf("EFECTOR FINAL 2: Al escoger esta opci¢n el usuario   ");
        gotoxy(14,13);
        cprintf("            podr  escoger entre tomar o soltar  ");
        gotoxy(14,14);
        cprintf("            objeto utilizando un electroim n.   ");
        gotoxy(14,15);
        cprintf("                                        ");
        gotoxy(14,16);
        cprintf("ACERCA DE: Esta opci¢n nos presentar  una aclaraci¢n ");
        gotoxy(14,17);
        cprintf("        importante sobre el programa.          ");
        gotoxy(14,18);
        cprintf("                                        ");
        gotoxy(14,19);
        cprintf("AYUDA GENERAL: Al seleccionar esta opci¢n se
presenta");
        gotoxy(14,20);
        cprintf("            la pantalla actual.             ");
        gotoxy(14,21);
```

```
        cprintf("                                        ");
        gotoxy(14,22);
        cprintf("TERMINAR: Con esta opci¢n podemos salir del programa.");
}
do
{
        c=getch();
}
while((c != ESC) && (c != ENTER));
if(c == ENTER)
{
        textcolor(WHITE);
        gotoxy(14,12);
        cprintf("IZQUIERDA: Esta opci¢n permite mover la respectiva   ");
        gotoxy(14,13);
        cprintf("         Articulaci¢n o Efector Final 1 a la       ");
        gotoxy(14,14);
        cprintf("         izquierda.                                ");
        gotoxy(14,15);
        cprintf("                                                   ");
        gotoxy(14,16);
        cprintf("DERECHA: Esta opci¢n permite mover la respectiva     ");
        gotoxy(14,17);
        cprintf("         Articulaci¢n o Efector Final 1 a la derecha.");
        gotoxy(14,18);
        cprintf("                                                   ");
        gotoxy(14,19);
        cprintf("COGER: Con esta opci¢n el Efector Final 2 tomar un  ");
        gotoxy(14,20);
        cprintf("       objeto met lico.                            ");
        gotoxy(14,21);
        cprintf("                                                   ");
        gotoxy(14,22);
        cprintf("SOLTAR: Permite al Efector Final 2 soltar el objeto. ");
        gotoxy(14,24);
        cprintf("                                                   ");
}
gotoxy(35,24);
cprintf("Salir:");
textcolor(LIGHTRED);
gotoxy(42,24);
cprintf("ESC");
do
{
        c=getch();
}
```

```
   while(c != ESC);
   ventana(10,8,70,25,color_back,color_back);
}


/* Funcion Principal */
void main(void)
{
   int salir = 0;
   int pos = MENU;
   int posx = 0;
   int posy = 3;
   int ymenor, ymayor, aux1, aux2;
   char c;
   int
ban1d=0,ban1i=0,ban2d=0,ban2i=0,ban3d=0,ban3i=0,ban4d=0,ban4i=0;
   int cont;

   presentacion();
   menu_inicial();
   spawnl(P_WAIT,"PPI.exe",NULL);
   while(salir == 0)
   {
        gotoxy(80,25);
        c = getch();
        if(c == ESC)
        {
          if(pos == SUBMENU)
          {
               borra_submenu(ymenor,ymayor);
               posx = aux1;
               pos = MENU;
          }
          if(pos == EJECUCION)
          {
               borra_submenu(ymenor,ymayor);
               ymenor = 3;
               ymayor = ymenor + 5;
               posy = aux2;
               pos = SUBMENU;
          }
        }
        if(c == ENTER)
        {
          if(pos == MENU)
          {
```

```
        if(posx == 1)
                spawnl(P_WAIT,"MAINAU.exe",NULL);
                else
        {
           aux1 = posx;
           if(posx == 0)
           {
                ymenor = 3;
                ymayor = ymenor + 5;
           }
           if(posx == 2)
           {
                ymenor = 9;
                ymayor = ymenor + 2;
           }
           submenu(ymenor,ymayor);
           pos = SUBMENU;
           posy = ymenor;
        }
}
else
{
        if(pos == SUBMENU)
        {
           aux2 = posy;
           if(posx == 0)
           {
                if((posy >= 3) && (posy <= 7))
                {
                   ymenor = 12;
                   ymayor = ymenor + 1;
                }
                else
                {
                   ymenor = 14;
                   ymayor = ymenor + 1;
                }
                submenu(ymenor,ymayor);
                pos = EJECUCION;
                posy = ymenor;
           }
           if(posx == 2)
           {
                if(posy == 9)
                {
                   borra_submenu(ymenor,ymayor);
```

```
            acerca_de();
            pos = MENU;
          }
          if(posy == 10)
          {
            borra_submenu(ymenor,ymayor);
            ayuda();
            pos = MENU;
          }
          if(posy == 11)
            salir = 1;
      }
}
else
{
   if(pos == EJECUCION)
   {
        textbackground(BLACK);
        if(aux2 == 3)
        {  if(posy == 12 && ban1d==0)
           { for(cont=1;cont<=30;cont++)
             { MAINSEN();
                SENSO=(SENSO ^ 255);   //NEGANDO
                SENSO=(SENSO & 128);    //10000000b
                if(SENSO==128) goto DECIDIR1D;
                ban1i=0;

             SIPUEDE1D:
             ;
                spawnl(P_WAIT,"ART1D.exe",NULL);
             }
             goto ART1I;

             DECIDIR1D:
             ;
                if(ban1i==1) goto SIPUEDE1D;
                else ban1d=1;
           }

           ART1I:
           ;
           if(posy == 13 && ban1i==0)
           { for(cont=1;cont<=30;cont++)
             { MAINSEN();
                SENSO=(SENSO ^ 255);   //NEGANDO
                SENSO=(SENSO & 128);    //10000000b
```

```
        if(SENSO==128) goto DECIDIR1I;
        ban1d=0;

    SIPUEDE1I:
    ;
        spawnl(P_WAIT,"ART1I.exe",NULL);
    }
    goto ART2;

    DECIDIR1I:
    ;
        if(ban1d==1) goto SIPUEDE1I;
        else ban1i=1;
   }
}

ART2:
;
if(aux2 == 4)
{
   if(posy == 12 && ban2d==0)
   { for(cont=1;cont<=30;cont++)
     { MAINSEN();
        SENSO=(SENSO & 16);      //00010000b
        if(SENSO==16) goto DECIDIR2D;
        ban2i=0;

     SIPUEDE2D:
     ;
        spawnl(P_WAIT,"ART2D.exe",NULL);
     }
     goto ART2I;

     DECIDIR2D:
     ;
        if(ban2i==1) goto SIPUEDE2D;
        else ban2d=1;
   }

   ART2I:
   ;
   if(posy == 13 && ban2i==0)
   { for(cont=1;cont<=30;cont++)
     { MAINSEN();
        SENSO=(SENSO & 16);      //00010000b
        if(SENSO==16) goto DECIDIR2I;
```

```
        ban2d=0;

    SIPUEDE2I:
    ;
        spawnl(P_WAIT,"ART2I.exe",NULL);
    }
    goto ART3;

    DECIDIR2I:
    ;
        if(ban2d==1) goto SIPUEDE2I;
        else ban2i=1;
   }
}

ART3:
;
if(aux2 == 5)
{
   if(posy == 12 && ban3d==0)
   { for(cont=1;cont<=30;cont++)
     { MAINSEN();
        SENSO=(SENSO & 32);      //00100000b
        if(SENSO==32) goto DECIDIR3D;
        ban3i=0;

     SIPUEDE3D:
     ;
        spawnl(P_WAIT,"ART3D.exe",NULL);
     }
     goto ART3I;

     DECIDIR3D:
     ;
        if(ban3i==1) goto SIPUEDE3D;
        else ban3d=1;
   }

   ART3I:
   ;
   if(posy == 13 && ban3i==0)
   { for(cont=1;cont<=30;cont++)
     { MAINSEN();
        SENSO=(SENSO & 32);      //00100000b
        if(SENSO==32) goto DECIDIR3I;
        ban3d=0;
```

```
    SIPUEDE3I:
    ;
        spawnl(P_WAIT,"ART3I.exe",NULL);
    }
    goto ART4;

    DECIDIR3I:
    ;
        if(ban3d==1) goto SIPUEDE3I;
        else ban3i=1;
   }
}

ART4:
;
if(aux2 == 6)
{
   if(posy == 12 && ban4d==0)
   { for(cont=1;cont<=100;cont++)
     { MAINSEN();
        SENSO=(SENSO & 64);      //01000000b
        if(SENSO==64) goto DECIDIR4D;
        ban4i=0;

     SIPUEDE4D:
     ;
        spawnl(P_WAIT,"ART4DA.exe",NULL);
     }
     spawnl(P_WAIT,"PPI.exe",NULL);
     goto ART4I;

     DECIDIR4D:
     ;
        if(ban4i==1) goto SIPUEDE4D;
        else ban4d=1;
     spawnl(P_WAIT,"PPI.exe",NULL);
   }

   ART4I:
   ;
   spawnl(P_WAIT,"ART4DB.exe",NULL);


   if(posy == 13 && ban4i==0)
   { for(cont=1;cont<=100;cont++)
```

```
                    { MAINSEN();
                      SENSO=(SENSO & 64);      //01000000b
                      if(SENSO==64) goto DECIDIR4I;
                      ban4d=0;

                    SIPUEDE4I:
                    ;
                      spawnl(P_WAIT,"ART4IA.exe",NULL);
                    }
                    spawnl(P_WAIT,"PPI.exe",NULL);
                    spawnl(P_WAIT,"ART4IB.exe",NULL);
                    goto ART5;

                    DECIDIR4I:
                    ;
                      if(ban4d==1) goto SIPUEDE4I;
                      else ban4i=1;
                    spawnl(P_WAIT,"PPI.exe",NULL);
                    spawnl(P_WAIT,"ART4IB.exe",NULL);
                   }
                 }

                 ART5:
                 ;

                 if(aux2 == 7)
                 {
                   if(posy == 12)
                   { spawnl(P_WAIT,"ART5D.exe",NULL); }
                   if(posy == 13)
                   { spawnl(P_WAIT,"ART5I.exe",NULL); }
                 }
                 if(aux2 == 8)
                 {
                   if(posy == 14)
                   { spawnl(P_WAIT,"ART6C.exe",NULL); }
                   if(posy == 15)
                   { spawnl(P_WAIT,"ART6S.exe",NULL); }
                 }
            }
          }
      }
}
if(c == 0)
{
   c = getch();
```

```
            if((c == DERECHA) && (pos == MENU))
            {
                llena_barra(barras[posx],color_fondo,BLACK);
                posx++;
                if(posx > 2)
                   posx = 0;
                llena_barra(barras[posx],color_marca,BLACK);
            }
            if((c == IZQUIERDA) && (pos == MENU))
            {
                llena_barra(barras[posx],color_fondo,BLACK);
                posx--;
                if(posx < 0)
                   posx = 2;
                llena_barra(barras[posx],color_marca,BLACK);
            }
            if((c == ARRIBA) && (pos != MENU))
            {
                llena_barra(barras[posy],color_fondo,BLACK);
                posy--;
                if(posy < ymenor)
                   posy = ymayor;
                llena_barra(barras[posy],color_marca,BLACK);
            }
            if((c == ABAJO) && (pos != MENU))
            {
                llena_barra(barras[posy],color_fondo,BLACK);
                posy++;
                if(posy > ymayor)
                   posy = ymenor;
                llena_barra(barras[posy],color_marca,BLACK);
            }
         }
      }
      textcolor(WHITE);
      textbackground(BLACK);
      clrscr();
}

void MAINSEN(void)
{  unsigned char temp;
   temp=inportb(0x379);

   SENSO=(temp & 248);        //11111000b
}
```

```
; PROGRAMAS AUXILIARES
;-------------------------------------------------------------------------

   TITLE PPI
   PILA SEGMENT PARA  PUBLIC 'STACK'
          DW 10 DUP(?)
   PILA ENDS
;-------------------------------------------------------------------------
DATO SEGMENT PARA PUBLIC 'DATA'
DATO ENDS
;-------------------------------------------------------------------------
CODIGO SEGMENT PARA PUBLIC 'CODE'
;-------------------------------------------------------------------------
MAINPPI PROC FAR                    ;PROGRAMA PRINCIPAL
  ASSUME CS:CODIGO, DS:DATO,SS:PILA
  PUSH DS
  SUB AX,AX
  PUSH AX
  MOV AX,DATO
  MOV DS,AX
  CALL UNO
  RET
MAINPPI ENDP
;-------------------------------------------------------------------------
          UNO PROC NEAR
          PUSH AX
          PUSH BX
          PUSH CX
          PUSH DX
          MOV DX,37AH
          MOV AL,00000100B
          MOV AH,0
          OUT DX,AL
          nop
          CALL RETARDO
          MOV AL,00001101B
          OUT DX,AL
          nop
          CALL RETARDO
          MOV DX,378H
          MOV AL,10000000B
          OUT DX,AL
          nop
          CALL RETARDO
```

```
        MOV DX,37AH
        MOV AL,00001100B
        OUT DX,AL
        nop
        CALL RETARDO
        POP DX
        POP CX
        POP BX
        POP AX
        RET
        UNO ENDP
;-------------------------------------------------------------------------
 RETARDO PROC NEAR
 PUSH CX
 MOV CX,10
 R1:
 PUSH CX
 MOV CX,1000
 R2:
 NOP
 NOP
 NOP
 LOOP R2
 POP CX
 LOOP R1
 POP CX
 RET
RETARDO ENDP
;-------------------------------------------------------------------------
CODIGO ENDS
END MAINPPI
;-------------------------------------------------------------------------



;-------------------------------------------------------------------------

  TITLE ART1D
  PILA SEGMENT PARA  PUBLIC 'STACK'
            DW 10 DUP(?)
  PILA ENDS
;-------------------------------------------------------------------------
DATO SEGMENT PARA PUBLIC 'DATA'
DATO ENDS
;-------------------------------------------------------------------------
CODIGO SEGMENT PARA PUBLIC 'CODE'
```

```
;-------------------------------------------------------------------------------
MAIN1D PROC FAR                    ;PROGRAMA PRINCIPAL
 ASSUME CS:CODIGO, DS:DATO,SS:PILA
 PUSH DS
 SUB AX,AX
 PUSH AX
 MOV AX,DATO
 MOV DS,AX
 CALL D1
 RET
MAIN1D ENDP
;-------------------------------------------------------------------------------
D1 PROC NEAR                       ;PROCEDIMIENTO PRINCIPAL
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV AL,10011001B
 CALL SALA
 CALL RETARDO
 MOV AL,10010001B
 CALL SALA
 CALL RETARDO
 MOV AL,10010101B
 CALL SALA
 CALL RETARDO
 MOV AL,10010100B
 CALL SALA
 CALL RETARDO
 MOV AL,10010110B
 CALL SALA
 CALL RETARDO
 MOV AL,10010010B
 CALL SALA
 CALL RETARDO
 MOV AL,10011010B
 CALL SALA
 CALL RETARDO
 MOV AL,10011000B
 CALL SALA
 CALL RETARDO
 MOV AL,10011001B
 CALL SALA
 CALL RETARDO
 POP DX
 POP CX
```

```
 POP BX
 POP AX
 RET
D1 ENDP
;-------------------------------------------------------------------------------
 RETARDO PROC NEAR
 PUSH CX
 MOV CX,10
 R1:
 PUSH CX
 MOV CX,1000
 R2:
 NOP
 NOP
 NOP
 LOOP R2
 POP CX
 LOOP R1
 POP CX
 RET
RETARDO ENDP
;-------------------------------------------------------------------------------
        SALA PROC NEAR
            PUSH AX
            MOV DX,37AH
            MOV AL,00001010B
            OUT DX,AL
            nop
            nop
            nop
            POP AX
            MOV DX,378H
            OUT DX,AL
            nop
            nop
            nop
            MOV DX,37AH
            MOV AL,00001011B
            OUT DX,AL
            nop
            nop
            nop
            MOV AL,00001010B
            OUT DX,AL
            nop
            nop
```

```
            nop
            RET
          SALA ENDP
;-------------------------------------------------------------------------
CODIGO ENDS
END MAIN1D
;-------------------------------------------------------------------------------


;-------------------------------------------------------------------------------

   TITLE ART1I
   PILA SEGMENT PARA  PUBLIC 'STACK'
          DW 10 DUP(?)
   PILA ENDS
;-------------------------------------------------------------------------------
DATO SEGMENT PARA PUBLIC 'DATA'
DATO ENDS
;-------------------------------------------------------------------------------
CODIGO SEGMENT PARA PUBLIC 'CODE'
;-------------------------------------------------------------------------------
MAIN1I PROC FAR                    ;PROGRAMA PRINCIPAL
  ASSUME CS:CODIGO, DS:DATO,SS:PILA
  PUSH DS
  SUB AX,AX
  PUSH AX
  MOV AX,DATO
  MOV DS,AX
  CALL D1
  RET
MAIN1I ENDP
;-------------------------------------------------------------------------------
D1 PROC NEAR                       ;PROCEDIMIENTO PRINCIPAL
  PUSH AX
  PUSH BX
  PUSH CX
  PUSH DX
  MOV AL,10011001B
  CALL SALA
  CALL RETARDO
  MOV AL,10011000B
  CALL SALA
  CALL RETARDO
  MOV AL,10011010B
```

```
 CALL SALA
 CALL RETARDO
 MOV AL,10010010B
 CALL SALA
 CALL RETARDO
 MOV AL,10010110B
 CALL SALA
 CALL RETARDO
 MOV AL,10010100B
 CALL SALA
 CALL RETARDO
 MOV AL,10010101B
 CALL SALA
 CALL RETARDO
 MOV AL,10010001B
 CALL SALA
 CALL RETARDO
 MOV AL,10011001B
 CALL SALA
 CALL RETARDO
 POP DX
 POP CX
 POP BX
 POP AX
 RET
D1 ENDP
;-------------------------------------------------------------------------------
 RETARDO PROC NEAR
 PUSH CX
 MOV CX,10
 R1:
 PUSH CX
 MOV CX,1000
 R2:
 NOP
 NOP
 NOP
 LOOP R2
 POP CX
 LOOP R1
 POP CX
 RET
RETARDO ENDP
;-------------------------------------------------------------------------------
      SALA PROC NEAR
         PUSH AX
```

```
        MOV DX,37AH
        MOV AL,00001010B
        OUT DX,AL
        nop
        nop
        nop
        POP AX
        MOV DX,378H
        OUT DX,AL
        nop
        nop
        nop
        MOV DX,37AH
        MOV AL,00001011B
        OUT DX,AL
        nop
        nop
        nop
        MOV AL,00001010B
        OUT DX,AL
        nop
        nop
        nop
        RET
    SALA ENDP
;-------------------------------------------------------------------------
CODIGO ENDS
END MAIN1I
;-------------------------------------------------------------------------------



;-------------------------------------------------------------------------------

  TITLE ART2D
  PILA SEGMENT PARA  PUBLIC 'STACK'
       DW 10 DUP(?)
  PILA ENDS
;-------------------------------------------------------------------------------
DATO SEGMENT PARA PUBLIC 'DATA'
DATO ENDS
;-------------------------------------------------------------------------------
CODIGO SEGMENT PARA PUBLIC 'CODE'
;-------------------------------------------------------------------------------
MAIN2D PROC FAR                 ;PROGRAMA PRINCIPAL
```

```
  ASSUME CS:CODIGO, DS:DATO,SS:PILA
  PUSH DS
  SUB AX,AX
  PUSH AX
  MOV AX,DATO
  MOV DS,AX
  CALL D1
  RET
MAIN2D ENDP
;-------------------------------------------------------------------------------
D1 PROC NEAR                    ;PROCEDIMIENTO PRINCIPAL
  PUSH AX
  PUSH BX
  PUSH CX
  PUSH DX
  MOV AL,10011001B
  CALL SALA
  CALL RETARDO
  MOV AL,00011001B
  CALL SALA
  CALL RETARDO
  MOV AL,01011001B
  CALL SALA
  CALL RETARDO
  MOV AL,01001001B
  CALL SALA
  CALL RETARDO
  MOV AL,01101001B
  CALL SALA
  CALL RETARDO
  MOV AL,00101001B
  CALL SALA
  CALL RETARDO
  MOV AL,10101001B
  CALL SALA
  CALL RETARDO
  MOV AL,10001001B
  CALL SALA
  CALL RETARDO
  MOV AL,10011001B
  CALL SALA
  CALL RETARDO
  POP DX
  POP CX
  POP BX
  POP AX
```

```
 RET
D1 ENDP
;-------------------------------------------------------------------------------
 RETARDO PROC NEAR
 PUSH CX
 MOV CX,10
 R1:
 PUSH CX
 MOV CX,1000
 R2:
 NOP
 NOP
 NOP
 LOOP R2
 POP CX
 LOOP R1
 POP CX
 RET
RETARDO ENDP
;-------------------------------------------------------------------------------
      SALA PROC NEAR
         PUSH AX
         MOV DX,37AH
         MOV AL,00001010B
         OUT DX,AL
         nop
         nop
         nop
         POP AX
         MOV DX,378H
         OUT DX,AL
         nop
         nop
         nop
         MOV DX,37AH
         MOV AL,00001011B
         OUT DX,AL
         nop
         nop
         nop
         MOV AL,00001010B
         OUT DX,AL
         nop
         nop
         nop
         RET
```

```
        SALA ENDP
;-------------------------------------------------------------------------
CODIGO ENDS
END MAIN2D
;-------------------------------------------------------------------------------



;-------------------------------------------------------------------------------

   TITLE ART2I
   PILA SEGMENT PARA  PUBLIC 'STACK'
          DW 10 DUP(?)
   PILA ENDS
;-------------------------------------------------------------------------------
DATO SEGMENT PARA PUBLIC 'DATA'
DATO ENDS
;-------------------------------------------------------------------------------
CODIGO SEGMENT PARA PUBLIC 'CODE'
;-------------------------------------------------------------------------------
MAIN2I PROC FAR                    ;PROGRAMA PRINCIPAL
  ASSUME CS:CODIGO, DS:DATO,SS:PILA
  PUSH DS
  SUB AX,AX
  PUSH AX
  MOV AX,DATO
  MOV DS,AX
  CALL D1
  RET
MAIN2I ENDP
;-------------------------------------------------------------------------------
D1 PROC NEAR                   ;PROCEDIMIENTO PRINCIPAL
  PUSH AX
  PUSH BX
  PUSH CX
  PUSH DX
  MOV AL,10011001B
  CALL SALA
  CALL RETARDO
  MOV AL,10001001B
  CALL SALA
  CALL RETARDO
  MOV AL,10101001B
  CALL SALA
  CALL RETARDO
```

```
  MOV AL,00101001B
  CALL SALA
  CALL RETARDO
  MOV AL,01101001B
  CALL SALA
  CALL RETARDO
  MOV AL,01001001B
  CALL SALA
  CALL RETARDO
  MOV AL,01011001B
  CALL SALA
  CALL RETARDO
  MOV AL,00011001B
  CALL SALA
  CALL RETARDO
  MOV AL,10011001B
  CALL SALA
  CALL RETARDO
  POP DX
  POP CX
  POP BX
  POP AX
  RET
D1 ENDP
;-------------------------------------------------------------------------------
  RETARDO PROC NEAR
  PUSH CX
  MOV CX,10
  R1:
  PUSH CX
  MOV CX,1000
  R2:
  NOP
  NOP
  NOP
  LOOP R2
  POP CX
  LOOP R1
  POP CX
  RET
RETARDO ENDP
;-------------------------------------------------------------------------------
        SALA PROC NEAR
           PUSH AX
           MOV DX,37AH
           MOV AL,00001010B
```

```
        OUT DX,AL
        nop
        nop
        nop
        POP AX
        MOV DX,378H
        OUT DX,AL
        nop
        nop
        nop
        MOV DX,37AH
        MOV AL,00001011B
        OUT DX,AL
        nop
        nop
        nop
        MOV AL,00001010B
        OUT DX,AL
        nop
        nop
        nop
        RET
    SALA ENDP
;-------------------------------------------------------------------------
CODIGO ENDS
END MAIN2I
;-------------------------------------------------------------------------------



;-------------------------------------------------------------------------------

  TITLE ART3D
  PILA SEGMENT PARA  PUBLIC 'STACK'
        DW 10 DUP(?)
  PILA ENDS
;-------------------------------------------------------------------------------
DATO SEGMENT PARA PUBLIC 'DATA'
DATO ENDS
;-------------------------------------------------------------------------
CODIGO SEGMENT PARA PUBLIC 'CODE'
;-------------------------------------------------------------------------
MAIN3D PROC FAR                    ;PROGRAMA PRINCIPAL
 ASSUME CS:CODIGO, DS:DATO,SS:PILA
 PUSH DS
```

```
 SUB AX,AX
 PUSH AX
 MOV AX,DATO
 MOV DS,AX
 CALL D1
 RET
MAIN3D ENDP
;-------------------------------------------------------------------------------
D1 PROC NEAR                    ;PROCEDIMIENTO PRINCIPAL
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV AL,01101001B
 CALL SALB
 CALL RETARDO
 MOV AL,01100001B
 CALL SALB
 CALL RETARDO
 MOV AL,01100101B
 CALL SALB
 CALL RETARDO
 MOV AL,01100100B
 CALL SALB
 CALL RETARDO
 MOV AL,01100110B
 CALL SALB
 CALL RETARDO
 MOV AL,01100010B
 CALL SALB
 CALL RETARDO
 MOV AL,01101010B
 CALL SALB
 CALL RETARDO
 MOV AL,01101000B
 CALL SALB
 CALL RETARDO
 MOV AL,01101001B
 CALL SALB
 CALL RETARDO
 POP DX
 POP CX
 POP BX
 POP AX
 RET
D1 ENDP
```

```
;-------------------------------------------------------------------------
 RETARDO PROC NEAR
 PUSH CX
 MOV CX,10
 R1:
 PUSH CX
 MOV CX,1000
 R2:
 NOP
 NOP
 NOP
 LOOP R2
 POP CX
 LOOP R1
 POP CX
 RET
RETARDO ENDP
;-------------------------------------------------------------------------
      SALB PROC NEAR
         PUSH AX
         MOV DX,37AH
         MOV AL,00001000B
         OUT DX,AL
         NOP
         NOP
         NOP
         POP AX
         MOV DX,378H
         OUT DX,AL
         NOP
         NOP
         NOP
         MOV DX,37AH
         MOV AL,00001001B
         OUT DX,AL
         NOP
         NOP
         NOP
         MOV AL,00001000B
         OUT DX,AL
         NOP
         NOP
         NOP
         RET
       SALB ENDP
```

```
;-------------------------------------------------------------------------
CODIGO ENDS
END MAIN3D
;-------------------------------------------------------------------------------



;-------------------------------------------------------------------------------

   TITLE ART3I
   PILA SEGMENT PARA  PUBLIC 'STACK'
          DW 10 DUP(?)
   PILA ENDS
;-------------------------------------------------------------------------------
DATO SEGMENT PARA PUBLIC 'DATA'
DATO ENDS
;-------------------------------------------------------------------------------
CODIGO SEGMENT PARA PUBLIC 'CODE'
;-------------------------------------------------------------------------------
MAIN3I PROC FAR                    ;PROGRAMA PRINCIPAL
  ASSUME CS:CODIGO, DS:DATO,SS:PILA
  PUSH DS
  SUB AX,AX
  PUSH AX
  MOV AX,DATO
  MOV DS,AX
  CALL D1
  RET
MAIN3I ENDP
;-------------------------------------------------------------------------------
D1 PROC NEAR                       ;PROCEDIMIENTO PRINCIPAL
  PUSH AX
  PUSH BX
  PUSH CX
  PUSH DX
  MOV AL,01101001B
  CALL SALB
  CALL RETARDO
  MOV AL,01101000B
  CALL SALB
  CALL RETARDO
  MOV AL,01101010B
  CALL SALB
  CALL RETARDO
  MOV AL,01100010B
```

```
 CALL SALB
 CALL RETARDO
 MOV AL,01100110B
 CALL SALB
 CALL RETARDO
 MOV AL,01100100B
 CALL SALB
 CALL RETARDO
 MOV AL,01100101B
 CALL SALB
 CALL RETARDO
 MOV AL,01100001B
 CALL SALB
 CALL RETARDO
 MOV AL,01101001B
 CALL SALB
 CALL RETARDO
 POP DX
 POP CX
 POP BX
 POP AX
 RET
D1 ENDP
;-------------------------------------------------------------------------------
 RETARDO PROC NEAR
 PUSH CX
 MOV CX,10
 R1:
 PUSH CX
 MOV CX,1000
 R2:
 NOP
 NOP
 NOP
 LOOP R2
 POP CX
 LOOP R1
 POP CX
 RET
RETARDO ENDP
;-------------------------------------------------------------------------------
      SALB PROC NEAR
         PUSH AX
         MOV DX,37AH
         MOV AL,00001000B
         OUT DX,AL
```

```
        NOP
        NOP
        NOP
        POP AX
        MOV DX,378H
        OUT DX,AL
        NOP
        NOP
        NOP
        MOV DX,37AH
        MOV AL,00001001B
        OUT DX,AL
        NOP
        NOP
        NOP
        MOV AL,00001000B
        OUT DX,AL
        NOP
        NOP
        NOP
        RET
      SALB ENDP

;-------------------------------------------------------------------------
CODIGO ENDS
END MAIN3I
;-------------------------------------------------------------------------------



;-------------------------------------------------------------------------------

  TITLE ART4DA
  PILA SEGMENT PARA  PUBLIC 'STACK'
       DW 10 DUP(?)
  PILA ENDS
;-------------------------------------------------------------------------------
DATO SEGMENT PARA PUBLIC 'DATA'
DATO ENDS
;-------------------------------------------------------------------------------
CODIGO SEGMENT PARA PUBLIC 'CODE'
;-------------------------------------------------------------------------------
MAIN4D PROC FAR                 ;PROGRAMA PRINCIPAL
 ASSUME CS:CODIGO, DS:DATO,SS:PILA
 PUSH DS
```

```
 SUB AX,AX
 PUSH AX
 MOV AX,DATO
 MOV DS,AX
 CALL D1
 RET
MAIN4D ENDP
;-------------------------------------------------------------------------------
D1 PROC NEAR                    ;PROCEDIMIENTO PRINCIPAL
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV AL,10010000B
 CALL SALB
 CALL RETARDO
 POP DX
 POP CX
 POP BX
 POP AX
 RET
D1 ENDP
;-------------------------------------------------------------------------------
 RETARDO PROC NEAR
 PUSH CX
 MOV CX,10
 R1:
 PUSH CX
 MOV CX,1000
 R2:
 NOP
 NOP
 NOP
 LOOP R2
 POP CX
 LOOP R1
 POP CX
 RET
RETARDO ENDP
;-------------------------------------------------------------------------------
      SALB PROC NEAR
         PUSH AX
         MOV DX,37AH
         MOV AL,00001000B
         OUT DX,AL
         NOP
```

```
          NOP
          NOP
          POP AX
          MOV DX,378H
          OUT DX,AL
          NOP
          NOP
          NOP
          MOV DX,37AH
          MOV AL,00001001B
          OUT DX,AL
          NOP
          NOP
          NOP
          MOV AL,00001000B
          OUT DX,AL
          NOP
          NOP
          NOP
          RET
         SALB ENDP

;-----------------------------------------------------------------------
CODIGO ENDS
END MAIN4D
;------------------------------------------------------------------------------




;-----------------------------------------------------------------------------

  TITLE ART4I
  PILA SEGMENT PARA  PUBLIC 'STACK'
         DW 10 DUP(?)
  PILA ENDS
;------------------------------------------------------------------------------
DATO SEGMENT PARA PUBLIC 'DATA'
DATO ENDS
;-----------------------------------------------------------------------------
CODIGO SEGMENT PARA PUBLIC 'CODE'
;-----------------------------------------------------------------------------
MAIN4I PROC FAR                    ;PROGRAMA PRINCIPAL
 ASSUME CS:CODIGO, DS:DATO,SS:PILA
 PUSH DS
 SUB AX,AX
```

```
 PUSH AX
 MOV AX,DATO
 MOV DS,AX
 CALL D1
 RET
MAIN4I ENDP
;-------------------------------------------------------------------------------
D1 PROC NEAR                    ;PROCEDIMIENTO PRINCIPAL
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV AL,00010000B
 CALL SALB
 CALL RETARDO
 POP DX
 POP CX
 POP BX
 POP AX
 RET
D1 ENDP
;-------------------------------------------------------------------------------
 RETARDO PROC NEAR
 PUSH CX
 MOV CX,10
 R1:
 PUSH CX
 MOV CX,1000
 R2:
 NOP
 NOP
 NOP
 LOOP R2
 POP CX
 LOOP R1
 POP CX
 RET
RETARDO ENDP
;-------------------------------------------------------------------------------
      SALB PROC NEAR
         PUSH AX
         MOV DX,37AH
         MOV AL,00001000B
         OUT DX,AL
         NOP
         NOP
```

```
        NOP
        POP AX
        MOV DX,378H
        OUT DX,AL
        NOP
        NOP
        NOP
        MOV DX,37AH
        MOV AL,00001001B
        OUT DX,AL
        NOP
        NOP
        NOP
        MOV AL,00001000B
        OUT DX,AL
        NOP
        NOP
        NOP
        RET
      SALB ENDP
;-------------------------------------------------------------------------
CODIGO ENDS
END MAIN4I
;-------------------------------------------------------------------------
```



```
;-------------------------------------------------------------------------

  TITLE ART5D
  PILA SEGMENT PARA  PUBLIC 'STACK'
        DW 10 DUP(?)
  PILA ENDS
;-------------------------------------------------------------------------
DATO SEGMENT PARA PUBLIC 'DATA'
DATO ENDS
;-------------------------------------------------------------------------
CODIGO SEGMENT PARA PUBLIC 'CODE'
;-------------------------------------------------------------------------
PUBLIC MAIN5D
MAIN5D PROC FAR                    ;PROGRAMA PRINCIPAL
 ASSUME CS:CODIGO, DS:DATO,SS:PILA
 PUSH DS
 SUB AX,AX
 PUSH AX
```

```
 MOV AX,DATO
 MOV DS,AX
 CALL D1
 RET
MAIN5D ENDP
;-------------------------------------------------------------------------------
D1 PROC NEAR                    ;PROCEDIMIENTO PRINCIPAL
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV CX,50
 LAXA:
 MOV AL,01100000B
 CALL SALB
 CALL RETARDO
 LOOP LAXA
 MOV AL,00000000B
 CALL SALB
 CALL RETARDO
 POP DX
 POP CX
 POP BX
 POP AX
 RET
D1 ENDP
;-------------------------------------------------------------------------------
 RETARDO PROC NEAR
 PUSH CX
 MOV CX,10
 R1:
 PUSH CX
 MOV CX,1000
 R2:
 NOP
 NOP
 NOP
 LOOP R2
 POP CX
 LOOP R1
 POP CX
 RET
RETARDO ENDP
;-------------------------------------------------------------------------------
      SALB PROC NEAR
         PUSH AX
```

```
        MOV DX,37AH
        MOV AL,00001000B
        OUT DX,AL
        NOP
        NOP
        NOP
        POP AX
        MOV DX,378H
        OUT DX,AL
        NOP
        NOP
        NOP
        MOV DX,37AH
        MOV AL,00001001B
        OUT DX,AL
        NOP
        NOP
        NOP
        MOV AL,00001000B
        OUT DX,AL
        NOP
        NOP
        NOP
        RET
      SALB ENDP
;-------------------------------------------------------------------------
CODIGO ENDS
END MAIN5D
;-------------------------------------------------------------------------------



;-------------------------------------------------------------------------------

  TITLE ART5I
  PILA SEGMENT PARA  PUBLIC 'STACK'
        DW 10 DUP(?)
  PILA ENDS
;-------------------------------------------------------------------------------
DATO SEGMENT PARA PUBLIC 'DATA'
DATO ENDS
;-------------------------------------------------------------------------------
CODIGO SEGMENT PARA PUBLIC 'CODE'
;-------------------------------------------------------------------------------
PUBLIC MAIN5I
```

```
MAIN5I PROC FAR                    ;PROGRAMA PRINCIPAL
 ASSUME CS:CODIGO, DS:DATO,SS:PILA
 PUSH DS
 SUB AX,AX
 PUSH AX
 MOV AX,DATO
 MOV DS,AX
 CALL D1
 RET
MAIN5I ENDP
;-------------------------------------------------------------------------------
D1 PROC NEAR                       ;PROCEDIMIENTO PRINCIPAL
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV CX,50
 LAXA:
 MOV AL,00100000B
 CALL SALB
 CALL RETARDO
 LOOP LAXA
 MOV AL,00000000B
 CALL SALB
 CALL RETARDO
 POP DX
 POP CX
 POP BX
 POP AX
 RET
D1 ENDP
;-------------------------------------------------------------------------------
 RETARDO PROC NEAR
 PUSH CX
 MOV CX,10
 R1:
 PUSH CX
 MOV CX,1000
 R2:
 NOP
 NOP
 NOP
 LOOP R2
 POP CX
 LOOP R1
 POP CX
```

```
 RET
RETARDO ENDP
;-------------------------------------------------------------------------------
        SALB PROC NEAR
           PUSH AX
           MOV DX,37AH
           MOV AL,00001000B
           OUT DX,AL
           NOP
           NOP
           NOP
           POP AX
           MOV DX,378H
           OUT DX,AL
           NOP
           NOP
           NOP
           MOV DX,37AH
           MOV AL,00001001B
           OUT DX,AL
           NOP
           NOP
           NOP
           MOV AL,00001000B
           OUT DX,AL
           NOP
           NOP
           NOP
           RET
         SALB ENDP
;-------------------------------------------------------------------------------
CODIGO ENDS
END MAIN5I
;-------------------------------------------------------------------------------




;-------------------------------------------------------------------------------

  TITLE ART6C
  PILA SEGMENT PARA  PUBLIC 'STACK'
          DW 10 DUP(?)
  PILA ENDS
;-------------------------------------------------------------------------------
DATO SEGMENT PARA PUBLIC 'DATA'
```

```
DATO ENDS
;-------------------------------------------------------------------------
CODIGO SEGMENT PARA PUBLIC 'CODE'
;-------------------------------------------------------------------------
MAIN6C PROC FAR                    ;PROGRAMA PRINCIPAL
 ASSUME CS:CODIGO, DS:DATO,SS:PILA
 PUSH DS
 SUB AX,AX
 PUSH AX
 MOV AX,DATO
 MOV DS,AX
 CALL D1
 RET
MAIN6C ENDP
;-------------------------------------------------------------------------
D1 PROC NEAR                       ;PROCEDIMIENTO PRINCIPAL
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV AL,01100000B
 CALL SALB
 CALL RETARDO
 POP DX
 POP CX
 POP BX
 POP AX
 RET
D1 ENDP
;-------------------------------------------------------------------------
 RETARDO PROC NEAR
 PUSH CX
 MOV CX,10
 R1:
 PUSH CX
 MOV CX,1000
 R2:
 NOP
 NOP
 NOP
 LOOP R2
 POP CX
 LOOP R1
 POP CX
 RET
RETARDO ENDP
```

```
;-------------------------------------------------------------------------
      SALB PROC NEAR
         PUSH AX
         MOV DX,37AH
         MOV AL,00001000B
         OUT DX,AL
         NOP
         NOP
         NOP
         POP AX
         MOV DX,378H
         OUT DX,AL
         NOP
         NOP
         NOP
         MOV DX,37AH
         MOV AL,00001001B
         OUT DX,AL
         NOP
         NOP
         NOP
         MOV AL,00001000B
         OUT DX,AL
         NOP
         NOP
         NOP
         RET
        SALB ENDP
;-------------------------------------------------------------------------
CODIGO ENDS
END MAIN6C
;-------------------------------------------------------------------------




;-------------------------------------------------------------------------

  TITLE ART6S
  PILA SEGMENT PARA  PUBLIC 'STACK'
         DW 10 DUP(?)
  PILA ENDS
;-------------------------------------------------------------------------
DATO SEGMENT PARA PUBLIC 'DATA'
DATO ENDS
;-------------------------------------------------------------------------
```

```
CODIGO SEGMENT PARA PUBLIC 'CODE'
;-------------------------------------------------------------------------------
PUBLIC MAIN6S
MAIN6S PROC FAR                    ;PROGRAMA PRINCIPAL
 ASSUME CS:CODIGO, DS:DATO,SS:PILA
 PUSH DS
 SUB AX,AX
 PUSH AX
 MOV AX,DATO
 MOV DS,AX
 CALL D1
 RET
MAIN6S ENDP
;-------------------------------------------------------------------------------
D1 PROC NEAR                       ;PROCEDIMIENTO PRINCIPAL
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV AL,00000000B
 CALL SALB
 CALL RETARDO
 POP DX
 POP CX
 POP BX
 POP AX
 RET
D1 ENDP
;-------------------------------------------------------------------------------
 RETARDO PROC NEAR
 PUSH CX
 MOV CX,10
 R1:
 PUSH CX
 MOV CX,1000
 R2:
 NOP
 NOP
 NOP
 LOOP R2
 POP CX
 LOOP R1
 POP CX
 RET
RETARDO ENDP
;-------------------------------------------------------------------------------
```

```
      SALB PROC NEAR
         PUSH AX
         MOV DX,37AH
         MOV AL,00001000B
         OUT DX,AL
         NOP
         NOP
         NOP
         POP AX
         MOV DX,378H
         OUT DX,AL
         NOP
         NOP
         NOP
         MOV DX,37AH
         MOV AL,00001001B
         OUT DX,AL
         NOP
         NOP
         NOP
         MOV AL,00001000B
         OUT DX,AL
         NOP
         NOP
         NOP
         RET
       SALB ENDP
;-------------------------------------------------------------------------
CODIGO ENDS
END MAIN6S
;-------------------------------------------------------------------------------
;-------------------------------------------------------------------------
;-------------------------------------------------------------------------
  TITLE MAINAU
  PILA SEGMENT PARA  PUBLIC 'STACK'
          DW 10 DUP(?)
  PILA ENDS
;-------------------------------------------------------------------------------
DATO SEGMENT PARA PUBLIC 'DATA'

DATO ENDS
;-------------------------------------------------------------------------
```

```
CODIGO SEGMENT PARA PUBLIC 'CODE'
;-------------------------------------------------------------------------------
MAINTIC PROC FAR                    ;PROGRAMA PRINCIPAL
 ASSUME CS:CODIGO, DS:DATO,SS:PILA
 PUSH DS
 SUB AX,AX
 PUSH AX
 MOV AX,DATO
 MOV DS,AX
 CALL MATIC
 RET
MAINTIC ENDP
;-------------------------------------------------------------------------------
MATIC PROC NEAR
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 CALL FRENTE5
 CALL RETARDO
 mov cx,20
 qx:
 call frente3
 loop qx
 MOV CX,100
QA:
 CALL FRENTE2
 LOOP QA
 CALL RETARDO
 MOV CX,40
QB:
 CALL FRENTE1
 LOOP QB
CALL RETARDO
 mov cx,20
 qq:
 call retro3
 loop qq
 call retardo
 CALL RETRO5
 MOV CX,35
QC:
 CALL RETRO1
 LOOP QC
 CALL RETARDO
 MOV CX,100
```

```
QD:
 CALL RETRO2
 LOOP QD
 CALL RETARDO
;-----------------------------------------
 CALL FRENTE5
 CALL RETARDO
 MOV CX,20
 RA:
 CALL FRENTE3
 LOOP RA
 MOV CX,100
QE:
 CALL FRENTE2
 LOOP QE
 CALL RETARDO
 MOV CX,35
QF:
 CALL FRENTE1
 LOOP QF
 CALL RETARDO
 mov cx,20
 qt:
 CALL RETRO3
 loop qt
 CALL RETRO5
 MOV CX,30
QG:
 CALL RETRO1
 LOOP QG
 CALL RETARDO
 MOV CX,100
QH:
 CALL RETRO2
 LOOP QH
 CALL RETARDO
;--------------------------------
 CALL FRENTE5
 CALL RETARDO
 MOV CX,20
 RB:
 CALL FRENTE3
 LOOP RB
 MOV CX,100
QI:
 CALL FRENTE2
```

```
LOOP QI
CALL RETARDO
MOV CX,30
QJ:
CALL FRENTE1
LOOP QJ
CALL RETARDO
mov cx,20
qu:
CALL RETRO3
loop qu
CALL RETRO5
MOV CX,25
QK:
CALL RETRO1
LOOP QK
CALL RETARDO
MOV CX,100
QL:
CALL RETRO2
LOOP QL
CALL RETARDO

;-----------------------------------------------------
CALL FRENTE5
CALL RETARDO
MOV CX,20
RC:
CALL FRENTE3
LOOP RC
MOV CX,100
QM:
CALL FRENTE2
LOOP QM
CALL RETARDO
MOV CX,25
QN:
CALL FRENTE1
LOOP QN
CALL RETARDO
mov cx,20
qv:
CALL RETRO3
loop qv
CALL RETRO5
MOV CX,20
```

```
 QO:
 CALL RETRO1
 LOOP QO
 MOV CX,40
 Qs:
 call frente3
 loop qs
 CALL RETARDO
 MOV CX,200
 QP:
 CALL FRENTE2
 LOOP QP
 CALL RETARDO
 MOV CX,40
 QR:
 CALL RETRO3
 LOOP QR
 CALL RETARDO
 CALL FRENTE4
 CALL RETARDO
 CALL RETRO4
 CALL RETARDO
 POP DX
 POP CX
 POP BX
 POP AX
RET
MATIC ENDP

;-------------------------------------------------------------------------
FRENTE1 PROC NEAR
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV AL,10011001B
 CALL SALA
 call retardo
 CALL RETARDO
 MOV AL,10010001B
 CALL SALA
 CALL RETARDO
 call retardo
 MOV AL,10010101B
 CALL SALA
 CALL RETARDO
```

```
 call retardo
 MOV AL,10010100B
 CALL SALA
 CALL RETARDO
 call retardo
 MOV AL,10010110B
 CALL SALA
 CALL RETARDO
 call retardo
 MOV AL,10010010B
 CALL SALA
 CALL RETARDO
 call retardo
 MOV AL,10011010B
 CALL SALA
 CALL RETARDO
 call retardo
 MOV AL,10011000B
 CALL SALA
 CALL RETARDO
 call retardo
 MOV AL,10011001B
 CALL SALA
 CALL RETARDO
 call retardo
 POP DX
 POP CX
 POP BX
 POP AX
RET
FRENTE1 ENDP
;-------------------------------------------------------------------------
RETRO1 PROC NEAR
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV AL,10011001B
 CALL SALA
 CALL RETARDO
 call retardo
 MOV AL,10011000B
 CALL SALA
 CALL RETARDO
 call retardo
 MOV AL,10011010B
```

```
 CALL SALA
 CALL RETARDO
 call retardo
 MOV AL,10010010B
 CALL SALA
 CALL RETARDO
 call retardo
 MOV AL,10010110B
 CALL SALA
 CALL RETARDO
 call retardo
 MOV AL,10010100B
 CALL SALA
 CALL RETARDO
 call retardo
 MOV AL,10010101B
 CALL SALA
 CALL RETARDO
 call retardo
 MOV AL,10010001B
 CALL SALA
 CALL RETARDO
 call retardo
 MOV AL,10011001B
 CALL SALA
 CALL RETARDO
 call retardo
 POP DX
 POP CX
 POP BX
 POP AX
RET
RETRO1 ENDP
;--------------------------------------------------------------------------
FRENTE2 PROC NEAR
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV AL,10011001B
 CALL SALA
 CALL RETARDO
 MOV AL,00011001B
 CALL SALA
 CALL RETARDO
 MOV AL,01011001B
```

```
 CALL SALA
 CALL RETARDO
 call retardo
 MOV AL,10010010B
 CALL SALA
 CALL RETARDO
 call retardo
 MOV AL,10010110B
 CALL SALA
 CALL RETARDO
 call retardo
 MOV AL,10010100B
 CALL SALA
 CALL RETARDO
 call retardo
 MOV AL,10010101B
 CALL SALA
 CALL RETARDO
 call retardo
 MOV AL,10010001B
 CALL SALA
 CALL RETARDO
 call retardo
 MOV AL,10011001B
 CALL SALA
 CALL RETARDO
 call retardo
 POP DX
 POP CX
 POP BX
 POP AX
RET
RETRO1 ENDP
;-------------------------------------------------------------------------
FRENTE2 PROC NEAR
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV AL,10011001B
 CALL SALA
 CALL RETARDO
 MOV AL,00011001B
 CALL SALA
 CALL RETARDO
 MOV AL,01011001B
```

```
 CALL SALA
 CALL RETARDO
 MOV AL,01001001B
 CALL SALA
 CALL RETARDO
 MOV AL,01101001B
 CALL SALA
 CALL RETARDO
 MOV AL,00101001B
 CALL SALA
 CALL RETARDO
 MOV AL,10101001B
 CALL SALA
 CALL RETARDO
 MOV AL,10001001B
 CALL SALA
 CALL RETARDO
 MOV AL,10011001B
 CALL SALA
 CALL RETARDO
 POP DX
 POP CX
 POP BX
 POP AX
RET
FRENTE2 ENDP
;-------------------------------------------------------------------------
RETRO2 PROC NEAR
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV AL,10011001B
 CALL SALA
 CALL RETARDO
 MOV AL,10001001B
 CALL SALA
 CALL RETARDO
 MOV AL,10101001B
 CALL SALA
 CALL RETARDO
 MOV AL,00101001B
 CALL SALA
 CALL RETARDO
 MOV AL,01101001B
 CALL SALA
```

```
 CALL RETARDO
 MOV AL,01001001B
 CALL SALA
 CALL RETARDO
 MOV AL,01011001B
 CALL SALA
 CALL RETARDO
 MOV AL,00011001B
 CALL SALA
 CALL RETARDO
 MOV AL,10011001B
 CALL SALA
 CALL RETARDO
 POP DX
 POP CX
 POP BX
 POP AX
RET
RETRO2 ENDP
;-------------------------------------------------------------------------
FRENTE3 PROC NEAR
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV AL,01101001B
 CALL SALB
 CALL RETARDO
 MOV AL,01100001B
 CALL SALB
 CALL RETARDO
 MOV AL,01100101B
 CALL SALB
 CALL RETARDO
 MOV AL,01100100B
 CALL SALB
 CALL RETARDO
 MOV AL,01100110B
 CALL SALB
 CALL RETARDO
 MOV AL,01100010B
 CALL SALB
 CALL RETARDO
 MOV AL,01101010B
 CALL SALB
 CALL RETARDO
```

```
 MOV AL,01101000B
 CALL SALB
 CALL RETARDO
 MOV AL,01101001B
 CALL SALB
 CALL RETARDO
 POP DX
 POP CX
 POP BX
 POP AX
RET
FRENTE3 ENDP
;--------------------------------------------------------------------------
RETRO3 PROC NEAR
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV AL,01101001B
 CALL SALB
 CALL RETARDO
 MOV AL,01101000B
 CALL SALB
 CALL RETARDO
 MOV AL,01101010B
 CALL SALB
 CALL RETARDO
 MOV AL,01100010B
 CALL SALB
 CALL RETARDO
 MOV AL,01100110B
 CALL SALB
 CALL RETARDO
 MOV AL,01100100B
 CALL SALB
 CALL RETARDO
 MOV AL,01100101B
 CALL SALB
 CALL RETARDO
 MOV AL,01100001B
 CALL SALB
 CALL RETARDO
 MOV AL,01101001B
 CALL SALB
 CALL RETARDO
 POP DX
```

```
 POP CX
 POP BX
 POP AX
RET
RETRO3 ENDP
;-------------------------------------------------------------------------
FRENTE4 PROC NEAR
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV CX,400
 TEST3:
 MOV AL,10011001B
 CALL SALB
 CALL RETARDO
 LOOP TEST3
 MOV AL,00001001B
 CALL SALB
 call retardo
 POP DX
 POP CX
 POP BX
 POP AX
RET
FRENTE4 ENDP
;-------------------------------------------------------------------------
RETRO4 PROC NEAR
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV CX,400
 TEST4:
 MOV AL,00011001B
 CALL SALB
 CALL RETARDO
 LOOP TEST4
 MOV AL,00001001B
 CALL SALB
 call retardo
 POP DX
 POP CX
 POP BX
 POP AX
RET
```

```
RETRO4 ENDP
;-------------------------------------------------------------------------
FRENTE5 PROC NEAR
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV AL,01101001B
 CALL SALB
 CALL RETARDO
 POP DX
 POP CX
 POP BX
 POP AX
 RET
FRENTE5 ENDP
;-------------------------------------------------------------------------
RETRO5 PROC NEAR
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 MOV AL,00001001B
 CALL SALB
 call retardo
 POP DX
 POP CX
 POP BX
 POP AX
RET
RETRO5 ENDP
;-------------------------------------------------------------------------
 RETARDO PROC NEAR
 PUSH CX
 MOV CX,10
 R1:
 PUSH CX
 MOV CX,1000
 R2:
 NOP
 NOP
 NOP
 LOOP R2
 POP CX
 LOOP R1
 POP CX
```

```
 RET
RETARDO ENDP
;-------------------------------------------------------------------------------
        SALB PROC NEAR
           PUSH AX
           MOV DX,37AH
           MOV AL,00001000B
           OUT DX,AL
           NOP
           POP AX
           MOV DX,378H
           OUT DX,AL
           NOP
           MOV DX,37AH
           MOV AL,00001001B
           OUT DX,AL
           NOP
           MOV AL,00001000B
           OUT DX,AL
           NOP

           RET
         SALB ENDP
;-------------------------------------------------------------------------
        SALA PROC NEAR
           PUSH AX
           MOV DX,37AH
           MOV AL,00001010B
           OUT DX,AL
           NOP
           POP AX
           MOV DX,378H
           OUT DX,AL
           NOP
           MOV DX,37AH
           MOV AL,00001011B
           OUT DX,AL
           NOP
           MOV AL,00001010B
           OUT DX,AL
           NOP
           RET
         SALA ENDP
;-------------------------------------------------------------------------
CODIGO ENDS
END MAINTIC
```

B

EL PUERTO PARALELO

La forma más común de sacar datos hacia el mundo externo desde un computador, es sin duda con la ayuda del puerto paralelo, al cual se conecta normalmente la impresora. Pero no es muy común utilizar este puerto para entrada de datos desde interfaces externas que no sea la misma impresora. Debido a que en esta tesis se ha utilizado al puerto paralelo del computador para sacar datos como también para introducirlos, se ha creído conveniente utilizar esta sección para dar una breve explicación de las señales que proporciona esta interfaz tanto para entrada como para salida de datos, obviamente la explicación dada está orientada a su utilización en esta tesis.

PROTOCOLOS Y CONECTORES

El puerto paralelo (También conocido como interfaz de E/S Centronics), es un estándar diseñado especialmente para conectar un computador a una impresora, esta interfaz trasmite en paralelo los datos que han de imprimirse, además proporciona varias señales que el ordenador utiliza para trasmitir señales de control hacia la impresora, y también para recibir señales de estado desde la impresora. Lo anterior permite que entre la impresora y el computador haya un correcto entendimiento para el buen funcionamiento de la comunicación.

Para facilitar la descripción de las señales de la interfaz, se la dividirá en tres partes que se mencionan a continuación:

a.- Salida de datos

b.- Salida de señales de control.

c.- Entrada de señales de control.

Salida de datos.- Estas señales corresponden a los pines 2 al 9 de el conector DB25, en esta tesis se direcciona a estos ocho bits de salida mediante la instrucción OUT de lenguaje ensamblador, dirección 378H. No hay que olvidar que la instrucción OUT saca los datos almacenados en el registro AL, en la tabla adjunta se muestra entonces la correspondencia entre cada bit del registro AL y el pin del conector DB25 por donde sale.

BIT REGISTRO AL	SALIDA DATOS	PIN DB25
AL0	DATO 0	2
AL1	DATO 1	3
AL2	DATO 2	4
AL3	DATO 3	5
AL4	DATO 4	6
AL5	DATO 5	7
AL6	DATO 6	8
AL7	DATO 7	9

Salida de señales de control.- Estas señales corresponden a los pines 1, 14, 16, 17 del conector DB25, se direcciona en 37AH igualmente se utiliza la instrucción OUT del lenguaje ensamblador para hacer referencia a estos pines, debido a su uso especial con las impresoras estos pines tienen nombres bien definidos, a continuación se mencionan:

STROBE/ : Usada para indicar a la impresora cuándo son válidos los datos que se presentan en las líneas de datos. Esta señal sale invertida con respecto al valor que se le de en el registro AL.

AUTO FD: Esta señal es usada para indicar a la impresora que ejecute de manera automática una impresión de línea en blanco, seguida de un retorno de carro. La señal sale directa de acuerdo al valor que se de en el registro AL.

INIT/ : Envía una señal que obliga a la impresora a parar la impresión y a borrar todos los datos del buffer. Esta señal sale invertida con respecto al valor asignado en el registro AL.

SLCT IN/ : Permite o no la selección de la impresora. También esta señal sale invertida con respecto al valor asignado en el registro AL.

Como se explicó en el Capítulo # 3, estas señales se utilizan para controlar la programación de la PPI-8255.

En la tabla adjunta se muestra la correspondencia entre estos pines y cada bit del registro AL al cual se refiere.

BIT REGISTRO AL	SALIDA DATOS	PIN DB25
AL0	STROBE/	1
AL1	AUTO FD	14
AL2	INIT/	16
AL3	SLCT IN/	17

Entrada de señales de control.- Corresponden a los pines 10, 11, 12,13, 15 del conector DB25, se direccionan en 379H, se usa la instrucción IN para ingresar los valores de estos pines. En su trabajo con la impresora tienen funciones y nombres especiales:

ACK: Indica al computador que puede enviar más datos. Entra sin inversión al registro AL.

BUSY/ : Indica al ordenador que no puede aceptar más datos porque se ha llenado el buffer. Entra invertido al registro AL con respecto a su valor externo.

PE : Indica al ordenador que no hay papel en la impresora. Entra sin inversión al registro AL.

SL : Indica al computador que la impresora está presente. Entra sin inversión al registro AL.

ERROR : Informa al computador que hay un error en la impresora y que no va a poder recibir más datos mientras no se corrija dicho error. Entra sin inversión al registro AL. En esta tesis se utiliza estos pines para informar al computador que se han activado los switches de límite. La correspondencia de estos pines con los bits de AL se muestran en el cuadro.

BIT REGISTRO AL	SALIDA DATOS	PIN DB25
AL0	ACK	10
AL1	BUSY/	11
AL2	PE	12
AL3	ERROR	15
AL4	SL	13

C

MOTORES DE PASO

DESCRIPCION BASICA

El motor de pasos es un dispositivo que traduce pulsos eléctricos en movimientos mecánicos. El eje del motor rota un ángulo determinado por cada pulso o excitación que recibe. Este desplazamiento angular se repite con precisión por cada nuevo pulso que este motor recibe, con la ayuda de un circuito manejador apropiado. El resultado de este movimiento preciso, fijo y repetido es la habilidad de este motor para posicionarse precisamente. Esto contrasta con el motor DC convencional, cuyo eje tiene un movimiento libre si no cuenta con un circuito de control de posición. El motor de pasos por lo tanto permite el control de su velocidad y dirección de movimiento. La repetibilidad (la habilidad para posicionar una carga a través de un mismo patrón de movimientos un múltiple número de veces), es muy grande. El único error introducido por el motor de pasos es su error de paso simple, que es generalmente menor del 5% de su paso. Más significantemente, este error es no acumulativo sin importar la distancia posicionada o el número de veces que el reposicionamiento tenga lugar. El motor de pasos es normalmente controlado por una fuente de poder DC y una circuiteria manejadora, que se ha discutido en capítulos anteriores.

TERMINOLOGIA DE MOTORES DE PASO

Debido a que el uso de los motores de paso es muy amplio, hay una terminología que requiere definición.

A. Angulo de paso.- El eje del motor rota un ángulo específico cada vez que la polaridad de sus devanados es cambiada. Este ángulo es llamado el ángulo de pasos, y se especifica en grados.

B. Pasos por revolución.- Este término describe el número total de pasos requeridos para que el eje del motor rote 360°, o lo que es lo mismo de una vuelta completa. El número de pasos por revolución se calcula dividiendo el ángulo de paso entre 360.

C. Pasos por segundo.- El número de movimientos angulares que el motor desarrolla por segundo.

D. Precisión de paso.- Generalmente se expresa en porcentaje e indica el error total introducido por el motor en un paso dado por este. Este error es no acumulativo, es decir no se incrementa a pasar de el número de pasos que este de.

E. Torque de retención.- Con el motor de pasos detenido, es el monto de torque, desde una fuente externa, requerida para sacar al motor de su posición de retención. Este torque es medido con voltaje aplicado al

motor. Es una característica básica de los motores de paso y proporciona estabilidad bajo condiciones de descanso o detención.

F. Manejadores.- Un término usado para describir la circuiteria que controla al motor de pasos, usualmente consiste de una fuente de poder, lógica de secuencia y componentes de conmutación.

ANGULOS DE PASO DISPONIBLES

En el cuadro adjunto se muestra los típicos ángulos de paso para motores que se encuentra normalmente disponibles.

ANGULO DE PASO (GRADOS)	PASOS POR REVOLUCION
0.72	500
1.8	200
2.0	180
2.5	144
5.0	72
15.0	24

CONSTRUCCION BASICA Y OPERACION

La operación de un motor de pasos es relacionada con la teoría básica de los magnetos permanentes donde "iguales se repelen" y "opuestos se atraen". Considere la figura 1: si el devanado del estator es energizado de forma tal que A es el polo norte y B es el polo sur y el magneto que es el rotor permanente es posicionado con la polaridad que se indica, es virtualmente imposible determinar la dirección de rotación. Sin embargo, si como se muestra en la figura 2, se agregan dos polos estáticos adicionales C y D con las polaridades mostradas, nosotros podríamos predecir la dirección de rotación de se motor. En este caso, la dirección sería en contra de las manecillas del reloj con el rotor alineándose entre el "promedio" del polo sur y el "promedio" del polo norte, tal como se muestra en la figura 3.

Para permitir una mejor resolución de paso, se puede agregar cuatro polos más y además se puede "dentar" al rotor y al motor, esto influye en el ángulo de paso que alcanzará el motor. En la figura 4 se muestra la configuración de un motor con 1.8° de ángulo de paso.

LA SECUENCIA DE CONMUTACION DE OCHO PASOS

La secuencia de ocho pasos que se usa en esta tesis es conocida también como "medio paso electrónico". Bajo estas condiciones, el motor se

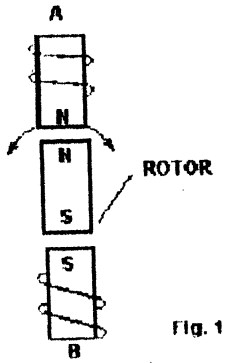


Fig. 1

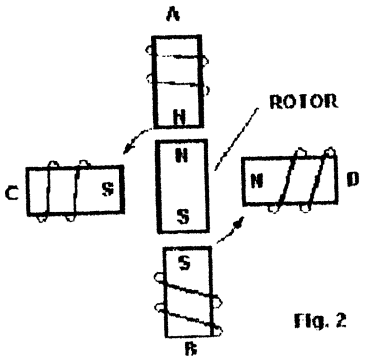


Fig. 2

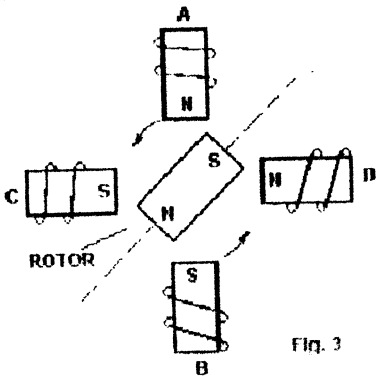


Fig. 3

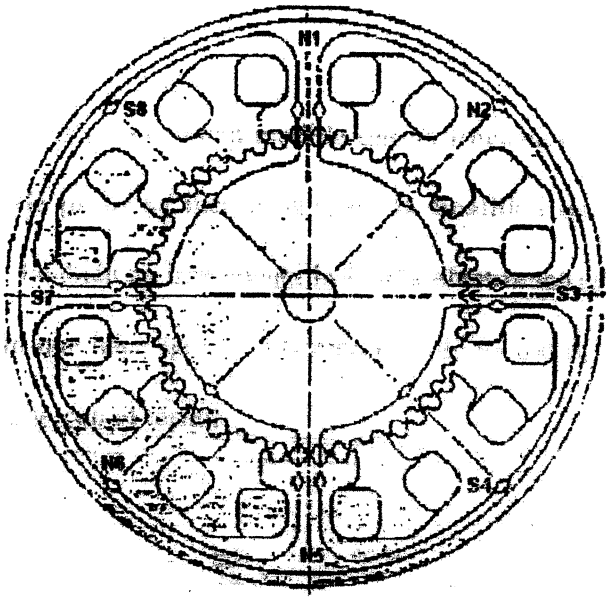


Fig. 4

mueve la mitad de su distancia normal por cada paso. Por ejemplo, un motor de 1.8° de paso, se moverá solamente 0.9°, igualmente los demás modelos. Las ventajas de operación en este modo incluyen una resolución más fina, la reducción de amplitudes resonantes y la posibilidad de manejo a mayores velocidades. En la figura 5 se presenta la secuencia de conmutación para alcanzar este modo de operación.

No está de más anotar que a medida que la frecuencia de conmutación del motor de pasos se incrementa con la finalidad de incrementar la velocidad, también se incrementa la fuerza electromotriz producida por el motor, lo que hace que la corriente de este motor se decremente, lo mismo sucede con el torque.

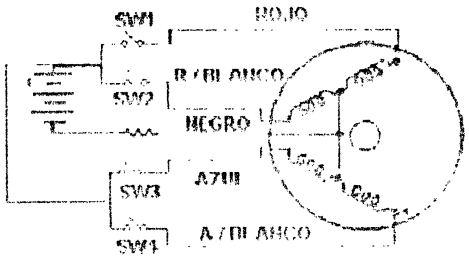
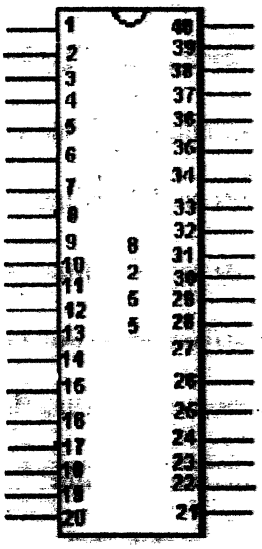


Fig. 5

D

INTERFASE PERIFERICA PROGRAMABLE

8255



DESCRIPCION DE LOS PINES

1.- PA-3, bit # 3 de entrada / salida del puerto A

2.- PA-2, bit # 2 de entrada / salida del puerto A

3.- PA-1, bit # 1 de entrada / salida del puerto A

4.- PA-0, bit # 0 de entrada / salida del puerto A

5.- RD/, pin de habilitación de lectura

6.- CS/, pin de habilitación del integrado

7.- Tierra

8.- A1, pin de programación

9.- A0, pin de programación

10.- PC-7, bit # 7 de entrada / salida del puerto C

11.- PC-6, bit # 6 de entrada / salida del puerto C

12.- PC-5, bit # 5 de entrada / salida del puerto C

13.- PC-4, bit # 4 de entrada / salida del puerto C

14.- PC-0, bit # 0 de entrada / salida del puerto C

15.- PC-1, bit # 1 de entrada / salida del puerto C

16.- PC-2, bit # 2 de entrada / salida del puerto C

17.- PC-3, bit # 3 de entrada / salida del puerto C

18.- PB-0, bit # 0 de entrada / salida del puerto B

19.- PB-1, bit # 1 de entrada / salida del puerto B

20.- PB-2, bit # 2 de entrada / salida del puerto B

21.- PB-3, bit # 3 de entrada / salida del puerto B

22.- PB-4, bit # 4 de entrada / salida del puerto B

23.- PB-5, bit # 5 de entrada / salida del puerto B

24.- PB-6, bit # 6 de entrada / salida del puerto B

25.- PB-7, bit # 7 de entrada / salida del puerto B

26.- Vcc

27.- D7, pin de entrada / salida de datos

28.- D6, pin de entrada / salida de datos

29.- D5, pin de entrada / salida de datos

30.- D4, pin de entrada / salida de datos

31.- D3, pin de entrada / salida de datos

32.- D2, pin de entrada / salida de datos

33.- D1, pin de entrada / salida de datos

34.- D0, pin de entrada / salida de datos

35.- Reset

36.- WR/, pin de habilitación de escritura

37.- PA-7, bit # 7 de entrada / salida del puerto A

38- PA-6, bit # 6 de entrada / salida del puerto A

39.- PA-5, bit # 5 de entrada / salida del puerto A

40.- PA-4, bit # 4 de entrada / salida del puerto A

BIBLIOGRAFIA

1. MURRAY - PAPPAS, Manual de lenguaje C/C $^{++}$ 7, Mc Graw Hill, 1996.

2. MOSHE SHOHAM, Conceptos Básicos de Robótica,

3. MASTERTON, Robótica, Collins, Londres,1988

4. KAFFRISSEN, Industrial Robotics, Reston & Publishing, 1985

5. BARRY BREY, Los Microprocesadores INTEL, Prentice-Hall, 1994.