90
Zj

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

CONTROL POR COMPUTADORA DE UN MANIPULADOR DIDÁCTICO

T E S I S

QUE PARA OBTENER EL TITULO DE:
INGENIERO EN COMPUTACIÓN
P R E S E N T A:

NORMA ISABEL ORTEGA MARTÍNEZ

ASESOR: ING. MIGUEL ANGEL BAÑUELOS SAUCEDO

LABORATORIO DE ELECTRONICA
CENTRO DE INSTRUMENTOS

MEXICO, D.F. 1996

TESIS CON
FALLA DE ORIGEN

Con todo mi cariño y agradecimiento a:

Juana Martínez de Ortega por enseñarme a seguir, aún en los momentos más difíciles.

Juan Ortega Pineda por su cariño y comprensión.

Mis hermanos *Gaby, Sandra, Ytain, Miguel, Juan, y Jorge* por todo lo que significan para mi.

A *Margarita Capistrán Martínez* quien siempre ha sido como una hermana.

A *Adrián Hernández Alva* por sus consejos, apoyo y cariño.

Un reconocimiento a los Ingenieros José Castillo, Wilfredo Martínez, Gerardo Calva, Rosendo Fuentes, al M.C. José Luis Pérez Silva y en especial al Ing. Miguel Angel Bañuelos Saucedo, por haberme brindado su amistad y asesoria en la elaboración del presente trabajo.

Un agradecimiento a los estudiantes del Lab. de Electrónica, al personal académico, técnico y administrativo del Centro de Instrumentos, y a la Universidad Nacional Autónoma de México por su gran apoyo.

ÍNDICE

INTRODUCCIÓN

De todas las cosas existentes, el ser humano es considerado una máquina casi perfecta; es inteligente, puede realizar tareas livianas o pesadas, puede moverse alrededor de si mismo, tiene mecanismos de autodefensa, puede alimentarse, protegerse del peligro, puede sentir, etc. Otras criaturas vivientes poseen funciones similares, aunque no siempre en forma avanzada.

Un robot es definido como un manipulador multifuncional reprogramable capaz de manejar cargas, piezas, herramientas o dispositivos especiales según variadas trayectorias programadas para efectuar la tarea.

Los robots son a menudo modelados como humanos; si no en forma, al menos en función. Por décadas, los científicos han tratado de duplicar el cuerpo humano, para crear máquinas con inteligencia, fuerza, movilidad y mecanismos sensoriales. La meta no ha sido alcanzada todavía, y tal vez nunca lo sea.

Aunque el robot se encuentra muy lejos de ser un duplicado humano; en la actualidad es utilizado como una herramienta con mayor rendimiento.

Este trabajo es una aportación computacional y electrónica para la realización del control desde una computadora personal de un manipulador didáctico de cinco grados de libertad (5GL). El programa desarrollado opera bajo ambiente Windows. Se resolvió un problema no lineal de cinemática inversa, incluyendo un control manual y algunas otras opciones que serán explicadas en detalle en los capítulos subsecuentes.

ANTECEDENTES

En el Laboratorio de Electrónica del Centro de Instrumentos, UNAM; se cuenta con un brazo mecánico de cinco grados de libertad que tiene su propia electrónica, pero no incluye una comunicación con la computadora. La electrónica solamente permite tres de los cinco movimientos del manipulador y no se cuenta con los manuales para poder complementarla o modificarla.

Los movimientos de la mano del manipulador utilizan un sistema de poleas, las cuales tampoco se encuentran colocadas. Por ello deberá de adecuársele un tipo especial de cable que no dañe el plástico de las uniones de la mano y al mismo tiempo le permita tener un libre movimiento.

Se desea poner en funcionamiento al manipulador controlándolo desde una computadora, realizando un trabajo didáctico y que, al mismo tiempo, sea posible poder utilizarlo en proyectos subsecuentes.

Definición del Problema

En la actualidad existen múltiples tipos de manipuladores que son controlados por un *Teach Pendant* (Control Manual) o por determinado tipo de software, pero son muy pocos los que tienen un ambiente sencillo y fácil de comprender por cualquier persona que no este involucrada en el área de robótica.

La presente tesis es una pequeña aportación para aquellas personas que quisieran tener algún contacto con este tipo de manipuladores pero no saben como podrían utilizarlos.

Aparte de contar con un Control Manual, se desea resolver el problema de cinemática inversa; es decir, dado un punto en el espacio encontrar el valor en grados que cada articulación debe moverse.

Por otro lado, el programa debe ser capaz de permitir la programación de movimientos del manipulador, y tener una interacción amigable con el usuario haciéndole más sencillo el manipulador al robot.

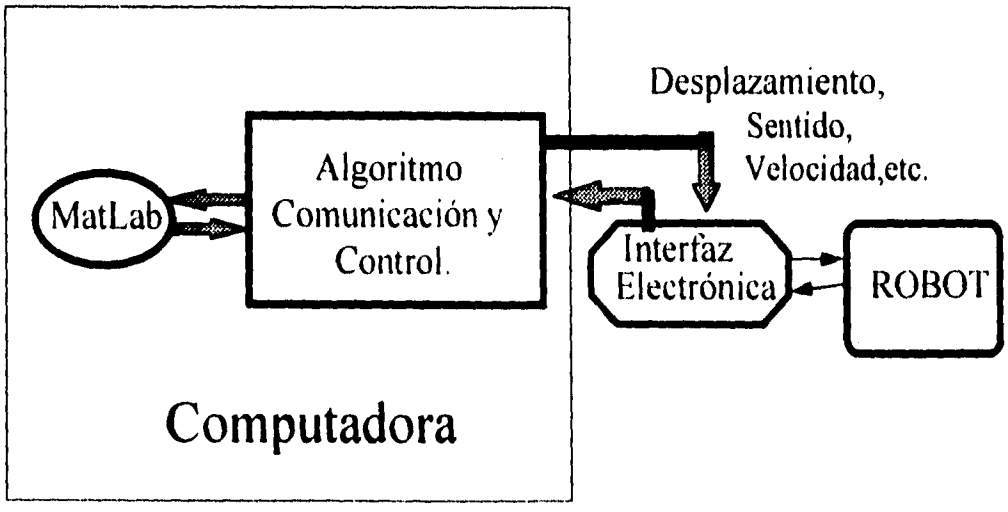
Se eligió hacerlo en ambiente Windows porque además de tener la habilidad de ejecutar más de una aplicación a la vez y transferir información entre ellas, cuenta con una rica interfaz gráfica. Ofreciéndole al usuario un ambiente de trabajo más intuitivo, sencillo de comprender y resulta más agradable para cualquier persona visualizar lo que se desea hacer.

Se decidió utilizar al interprete MatLab porque facilita la realización de programas en ambiente Windows, trabaja con matrices y vectores y se puede hacer acceso a código en lenguaje C. Además una de sus herramientas está orientada a redes neuronales que es un plan a futuro dentro del Centro de Instrumentos UNAM en donde se realizó esta tesis.

OBJETIVOS

- Obtener un modelado adecuado para controlar un manipulador didáctico de *5GL*.
- Realizar un algoritmo que dada una posición pueda generar los datos necesarios para que el manipulador alcance el punto especificado.
- Implementar el algoritmo en ambiente Windows y realizar una interacción por medio de un "ratón".
- Diseñar la interfaz electrónica para obtener la comunicación Robot <->Computadora.

El diagrama a bloques del plan a seguir es el siguiente:



1. Capítulo I

MODELADO

Introducción

Un elemento básico para el control de todos los manipuladores avanzados es la relación entre las coordenadas cartesianas del efector final y las coordenadas del manipulador. Un método directo es asignar sistemas de referencia en las uniones y obtener el valor de la posición del efector final en términos de sus coordenadas.

El objetivo de este capítulo es obtener el valor en grados de los ángulos de cada una de las cinco articulaciones del brazo mecánico (problema de cinemática inversa), considerando solamente las coordenadas cartesianas del punto destino y la orientación deseada.

Antecedentes

- Cuando un manipulador tiene menos de 6GL, no puede alcanzar todas las posiciones y orientaciones del espacio 3D. En muchas situaciones reales los manipuladores de 4 ó 5GL son empleados para operar fuera de un plano, pero evidentemente no pueden alcanzar metas generales.[1]

- Si el manipulador tiene 6GL, el modelado matemático solución iterativa utilizando notación de Denavit-Hartenberg puede ser aplicado directamente; pero si tiene más o menos GL debe haber consideraciones adicionales para hacer la matriz jacobiana cuadrada (R. Featherstone, 1983).

- Es conveniente utilizar un método con solución cerrada; al menos que se requieran encontrar todas las soluciones, en donde resulta más eficiente un método iterativo.

- Involucrar el cálculo e inversión de una matriz jacobiana de transformación, consume tiempo de proceso, además el proceso de inversión se complicará numéricamente cuando el determinante del jacobiano tienda a cero. (R. Featherstone 1983).

- Los puntos singulares son distinguidos en el momento en que el efector final pierde uno o más grados de libertad; es decir, hay una o más direcciones en las cuales no pueda moverse (Uchiyama 1979).Esto es, existe alguna dirección en el espacio cartesiano dentro de la cual es imposible mover la mano o, el punto seleccionado se encuentra fuera del campo de trabajo del robot.[2]

- El robot a modelar tiene 2GL en la muñeca (el ser humano tiene 3GL) ocasionando que el número de puntos singulares se incremente.

Análisis

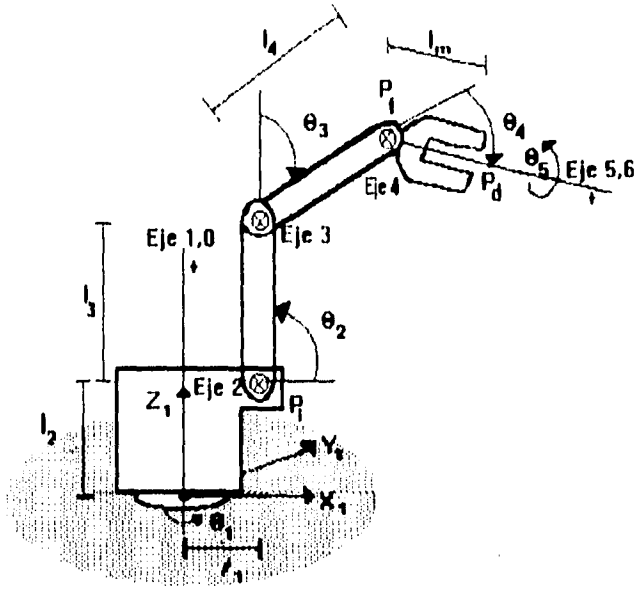
Fueron analizados dos métodos para el modelado del manipulador: el modelado inverso solución iterativa y el modelado matemático inverso solución geométrica.

a)Modelado Matemático Inverso Solución Iterativa

El modelado forma abierta o solución iterativa utiliza la notación de Denavit-Hartenberg para representar la orientación y la posición del manipulador; adoptando la filosofía de que en algún lugar hay un sistema de referencia universal de coordenadas dentro del cual se pueden describir todas las posiciones y orientaciones respecto a él o respecto a otro sistema de referencia.

Utiliza un operador de transformación formado por una matriz de rotación y un vector de posición, la dificultad de modelar un manipulador utilizando este método estriba en definir cuidadosamente los parámetros necesarios para formar cada uno de los operadores de transformación.

El diagrama para obtener el modelado utilizando la notación de Denavit-Hartenberg es el siguiente:



Representación de los parámetros para el modelado forma abierta

y el valor de los parámetros necesarios es el siguiente:

Tabla de Parámetros para Modelar al Manipulador 5GL

i	a_{i-1}	α_{i-1}	d_i	θ_i
0	0	0°	l_2	θ_1
1	l_1	270°	0	θ_2
2	l_3	0°	0	θ_3
3	l_4	0°	0	θ_4
4	0	270°	l_m	θ_5
5	0	-	-	-

Donde:

i: Es el número del eje de referencia.

a_{i-1}: Es la distancia más corta entre el eje de referencia i-1 y el eje i.

α_{i-1}: Es el ángulo formado entre el positivo del eje i-1 y el positivo del eje i alrededor de a_{i-1}.

d_i: Es la distancia medida sobre el eje i desde donde a_{i-1} corta al eje i hasta donde a_i corta el eje i.

θ_i: Es el ángulo medido al llevar la prolongación positiva de a_{i-1} hasta a_i positivo, medido alrededor de d_i.

Habiendo obtenido los parámetros del modelado y sustituyéndolos en la siguiente matriz de transformación[1]:

$$ {}^{i-1}_{i}T = \begin{vmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \cos(\alpha_{i-1})\sin\theta_i & \cos(\alpha_{i-1})\cos\theta_i & -\sin(\alpha_{i-1}) & -d_i\sin(\alpha_{i-1}) \\ \sin(\alpha_{i-1})\sin\theta_i & \sin(\alpha_{i-1})\cos\theta_i & \cos(\alpha_{i-1}) & d_i\cos(\alpha_{i-1}) \\ 0 & 0 & 0 & 1 \end{vmatrix} $$

se obtienen las matrices de transformación del eje i respecto al eje i-1.

[1] La matriz de transformación se encuentra dividida en una matriz de rotación 3X3(elementos 1,1 hasta 3,3); y un vector de posición 3X1(elementos 1,4;2,4 y 3,4).

Las matrices obtenidas fueron las siguientes:

$$ {}^{0}_{1}T = \begin{vmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & l_1 \\ 0 & 0 & 0 & 1 \end{vmatrix} $$

$$ {}^{1}_{2}T = \begin{vmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & l_1 \\ 0 & 0 & 1 & 0 \\ -\sin\theta_2 & -\cos\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} $$

$$ {}^{2}_{3}T = \begin{vmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & l_3 \\ \sin\theta_3 & \cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} $$

$$ {}^{3}_{4}T = \begin{vmatrix} \cos\theta_4 & -\sin\theta_4 & 0 & l_4 \\ \sin\theta_4 & \cos\theta_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} $$

$$ {}^{4}_{5}T = \begin{vmatrix} \cos\theta_5 & -\sin\theta_5 & 0 & 0 \\ 0 & 0 & 1 & l_m \\ -\sin\theta_5 & -\cos\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} $$

Después de multiplicar las matrices de transformación anteriores, se obtiene la matriz de transformación final. La matriz de 3X3 formada con las tres primeras columnas y los tres primeros renglones, representa la matriz de orientación y el vector 3X1 restante representa la posición; dando por resultado el modelado del manipulador para posicionarlo y orientarlo en un punto.

Se desarrolló un algoritmo en MatLab que obtuviera la posición final de un manipulador de 3GL a partir de un punto inicial; para observar, dentro de una situación casi real, las ventajas que se obtendrían al utilizar este método.

Al implementarlo se presentaron problemas de eficiencia, mostrando que el algoritmo era muy lento, debido a que el número de iteraciones a realizar puede ser grande, o porque dentro de su trayectoria existió algún punto singular que fue verificado e incrementó el tiempo de proceso. Se requirió en ocasiones hasta más de 1 minuto en encontrar la solución entre dos puntos muy cercanos. La prueba fue realizada en una computadora personal 486 a 66MHz.

Este método, al involucrar jacobianos e inversión de matrices, ocasiona que el proceso de cálculo sea mayor y en algunos casos no pueda determinarse la matriz inversa.

Al analizar el modelado inverso solución geométrica y realizarle la misma prueba de eficiencia, este demostró ser más apropiado.

b) Modelado Matemático Inverso Solución Geométrica

La solución utilizando este tipo de modelado es única, no intervienen matrices inversas ni jacobianos, por lo que, disminuye el tiempo de proceso. La solución, aún cuando la distancia sea grande consume el mismo tiempo de cálculo que para una distancia pequeña. Además resultó ser más sencillo de manipular en cuanto a las limitaciones mecánicas del brazo se refiere.

Características del manipulador.

El brazo mecánico a controlar cuenta con *cinco grados de libertad (5GL)*; ésto es, el mínimo número de parámetros independientes con los cuales puede posicionar y orientar un cuerpo en el espacio.

Se tiene un punto inicial P_i (cuyas coordenadas en centímetros respecto al eje de referencia E_1 son: (28.05,0,0.01)[cm]); del cual partirá a un punto especificado y denotado como P_d; de éste a otro y así sucesivamente. (Ver fig. 1.1).

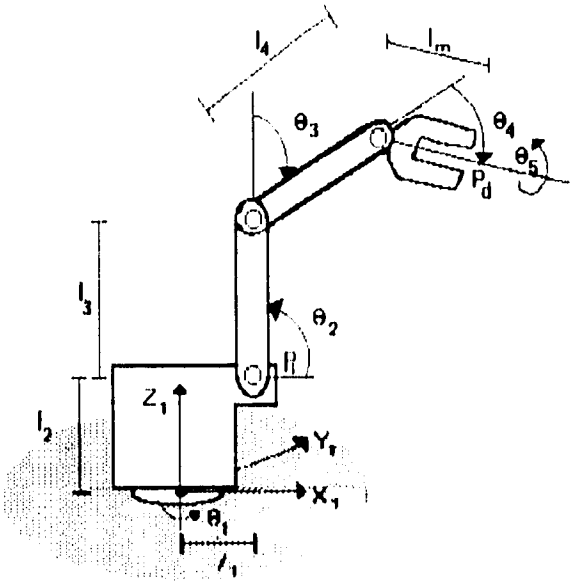


Figura 1.1. Modelado Geométrico.

Donde:

E_1 (X_1, Y_1, Z_1) es el sistema de referencia 1.

l_1 = 19 cm
l_2 = 14 cm
l_3 = 22.0 cm
l_4 = 22.0 cm longitudes de los eslabones

l_m = 16 cm la longitud el gripper

P_1 (28.05,0,0.01) punto de referencia

P_d punto al que se desea llevar al manipulador

$\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$ grados de libertad del manipulador

El *campo de trabajo* (espacio dentro del cual el manipulador puede moverse sin dificultad), esta limitado de la siguiente manera:

θ_1 (articulación del torso) no puede dar giros de 360°, solamente de su posición inicial a $\pm 165°$

θ_2 (articulación del brazo), tiene un alcance de 90° a -40°
θ_3 (articulación del codo) tiene un alcance de 155° a 0°
Sin embargo, su valor cambia proporcionalmente al cambio generado por θ_2; es decir, mientras θ_2 se mueva, θ_3 se mueve también, pero conservando su dirección
θ_4 (inclinación de la mano), tiene un alcance de -90° a 90° respecto a la horizontal
θ_5 (rotación de la mano) tiene un alcance de 0° a 180°.

Los datos a partir de los cuales se obtiene el modelado geométrico son la posición y la orientación del manipulador en el punto final. La posición está determinada por las coordenadas cartesianas, y la orientación por un ángulo de rotación y por uno de inclinación del gripper (conocidos en robótica como ángulos de roll y pitch respectivamente [3]).

En nuestro caso solamente necesitaremos conocer el valor del punto deseado $P_d[P_{dx}, P_{dy}, P_{dz}]$ y el ángulo de roll o θ_5; para obtener los valores de los desplazamientos angulares $\Delta\theta_i$ de las cuatro articulaciones.

Para obtener el valor de θ_1 a partir de las coordenadas cartesianas se realiza un análisis observando al manipulador desde z_1 dando por resultado el valor de θ_1 en términos de x_1 y y_1 como se muestra en la Fig. 1.2.

Vista Superior Manipulador Completo

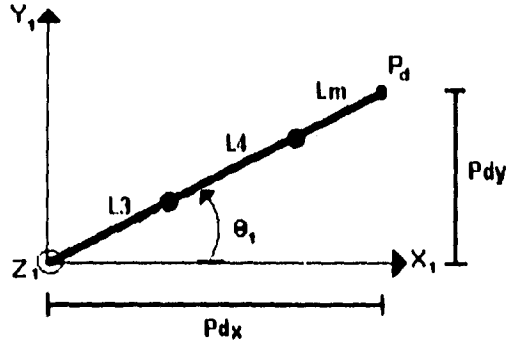


Figura 1.2.Representación en el plano XY del manipulador.

donde:

P_d es el punto deseado en el plano (x_1, y_1).

Dando por resultado:

$$\theta_1 = \text{arctg}(\frac{P_{dy}}{P_{dx}}) \qquad (1.1)$$

Suponiendo que el manipulador se encuentra en una posición inicial $P_{inicial}$ con un ángulo $\theta_{1\,inicial}$, podemos definir el incremento:

$$\Delta\theta_1 = \theta_1 - \theta_{1\,inicial} \qquad (1.2)$$

Es decir, la diferencia entre el ángulo θ_1 del punto inicial y el ángulo θ_1 obtenido de la ecuación 1.1.

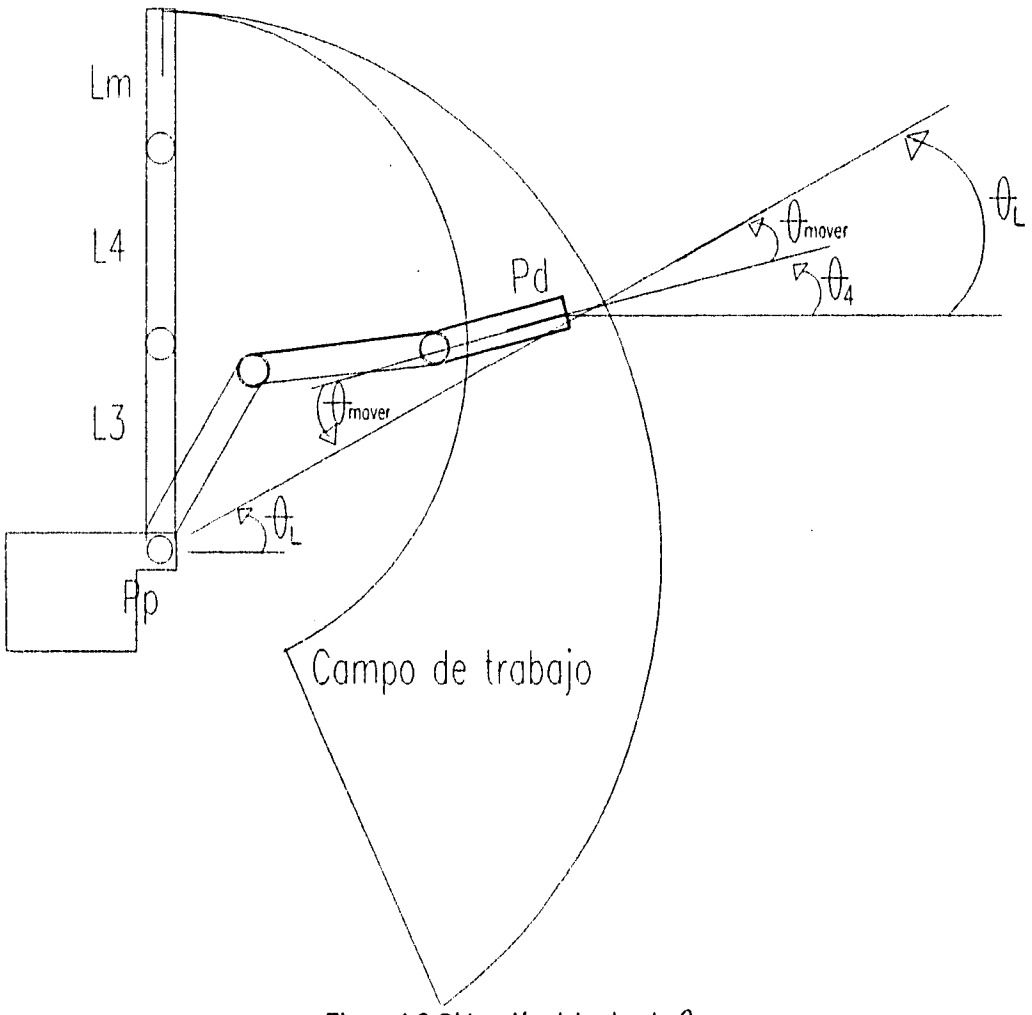


Figura 1.3.Obtención del valor de θ_1.

Para la obtención del valor de θ_2 y θ_3 debemos obtener las coordenadas del punto en la muñeca; para ello se sabe que el manipulador ya rotó $\Delta\theta_1$ en el eje z_1 colocándose en un nuevo plano que llamaremos plano N.

El problema es encontrar el valor más adecuado de θ_4, que nos permita fijar el valor del punto en la muñeca, de tal forma que θ_2 y θ_3 se encuentren dentro del intervalo de valores permitidos, eliminando puntos singulares.(ver figura 1.3).

Para ello, consideramos que al seleccionar puntos dentro del área de trabajo, las articulaciones se van alejando de la recta $\overline{L}$ (que va del punto Pf al punto Pp, ver fig. 1.3 y 1.4) y ésta separación aumenta conforme la magnitud de la recta disminuye.

Así, a los puntos más externos al área de trabajo les corresponde un valor de θ_{mover} [1] igual a cero, y ese valor aumenta conforme el punto elegido sea más interno.

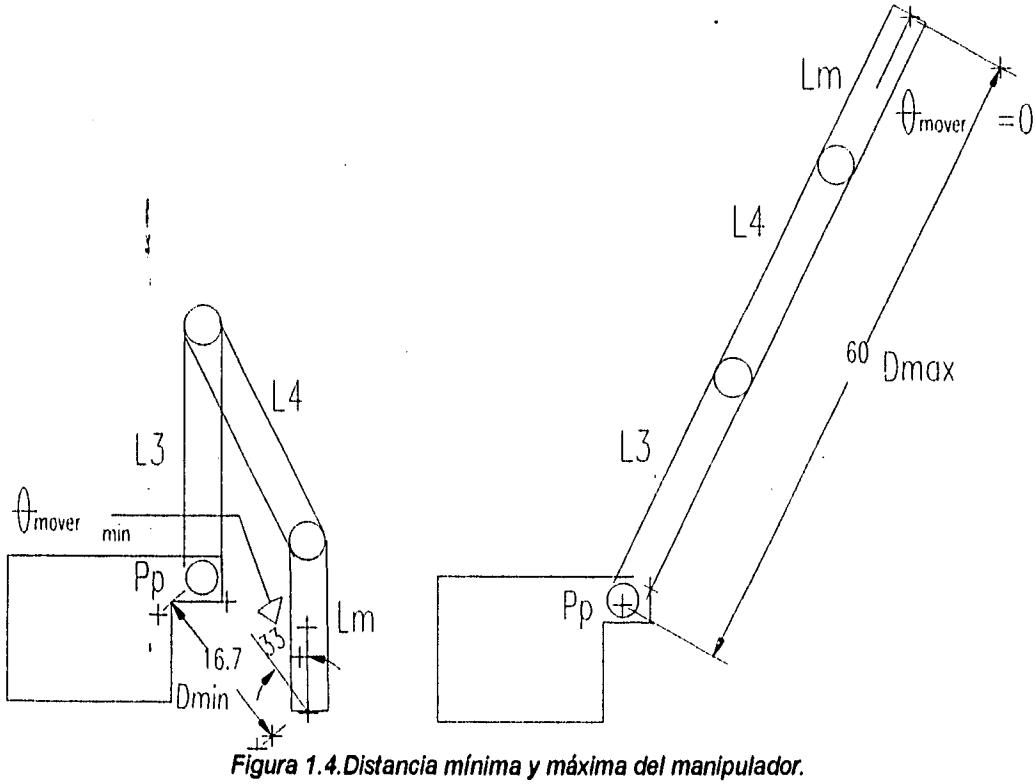


Figura 1.4.Distancia mínima y máxima del manipulador.

[1] Ángulo formado entre la recta $\overline{L}$ y el efector final.

La distancia máxima (desde el punto Pp hasta el efector) que puede alcanzar el manipulador es de 60cm (22cm brazo+22cm codo+16cm efector); con un valor de θ_{mover} igual a cero grados; la distancia mínima permitida es de 16.7636cm con un valor de θ_{mover} igual a 33.58°. (ver fig 1.4).

Considerando la ecuación de una recta para definir el valor de θ_{mover} se tiene:

$$\theta_{mover} - \theta_{mover\,min} = m\,(dist - dist\,min) \qquad (1.3.1)$$

$$m = \frac{(\theta_{mover-max} - \theta_{mover-min})}{(dist\_max - dist\_min)} \qquad (1.3.2)$$

despejando θ_{mover} y reduciendo las ecuaciones 1.3.1 y 1.3.2 se obtiene:

$$\theta_{mover} = -0.0776\,dist + 46.60 \qquad (1.3)$$

Siendo esta ecuación la que nos permitirá conocer el valor de θ_{mover} dado el valor de la distancia. La distancia es calculada de la siguiente manera:

$$dist = \sqrt{(Pd_x - Pp_x)^2 + (Pd_y - Pp_y)^2 + (Pd_z - Pp_z)^2} \qquad (1.4)$$

El valor de θ_L (ver fig 1.3), es obtenido de la siguiente manera:

$$\theta_L = arcsin\left(\frac{Pdz - Ppz}{dist}\right) \qquad (1.5)$$

El valor de θ_4 es:

$$\theta_4 = \theta_L - \theta_{mover} \qquad (1.6)$$

Conociendo el valor del punto deseado $P_d[P_{dx}, P_{dy}, P_{dz}]$, las dimensiones del gripper, el valor de θ_4 y utilizando proyecciones (ver fig. 1.5) se puede obtener el valor del punto en la muñeca.

Dando por resultado las siguientes ecuaciones:

$$proy_{xy} = l_m \cos\theta_4 \qquad (1.7)$$

$$(P_{dz} - P_{fz}) = l_m \sin\theta_4 \qquad (1.8)$$

$$(P_{dy} - P_{fy}) = proy_{xy} \sin\theta_1 \qquad (1.9)$$

$$(P_{dx} - P_{fx}) = proy_{xy} \cos\theta_1 \qquad (1.10)$$

Utilizando las ecuaciones 1.8,1.9,1.10 y despejando P_{fx}, P_{fy}, P_{fx}, obtenemos el valor del punto P_f o punto de la muñeca como:

$$P_f = [P_{fx}, P_{fy}, P_{fz}] \qquad (1.11)$$

Vista Lateral de la Mano

Manipulador Completo,Vista Lateral

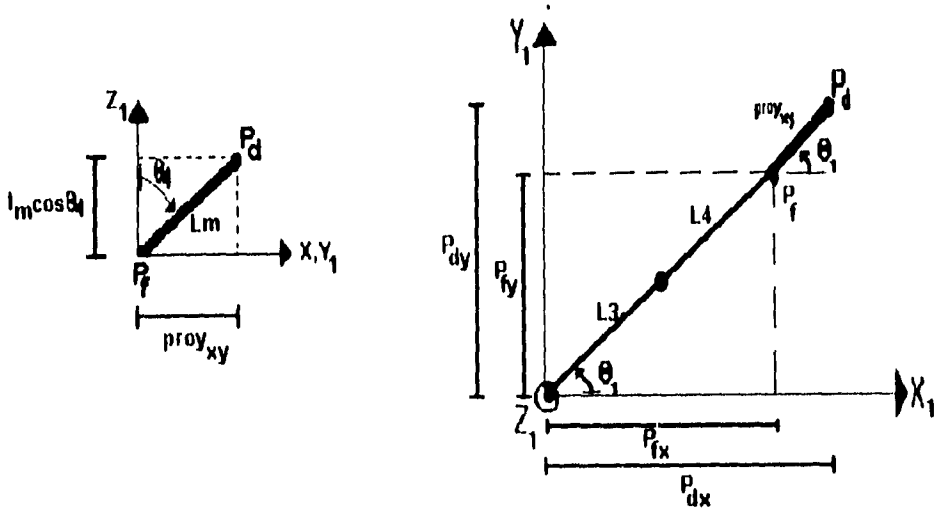


Figura 1.5. Representación del Gripper y Manipulador.

Para obtener el valor de θ_2 (ver fig.1.6).

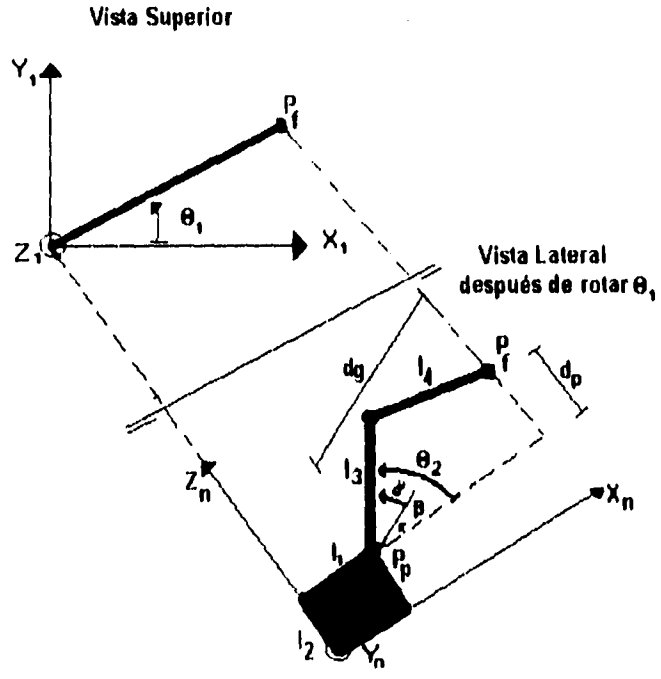


Figura 1.6.Obtención del valor de θ_2.

Haciendo una rotación canónica alrededor del eje z_1, θ_1 podemos obtener el valor del punto P_p después de la rotación.

$$P_{p\ rotado} = \text{Rot}z_1(\theta_1)\ P_{p\ inicial} \qquad (1.12)$$

donde:

$$Rot\ z_1(\theta_1) = \begin{vmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{vmatrix} \qquad (1.13)$$

De esta manera la distancia desde el punto P_p después de la rotación hasta el punto P_f puede calcularse así:

$$d_g = \sqrt{((P_{fx} - P_{px})^2 + (P_{fy} - P_{py})^2 + (P_{fz} - P_{pz})^2)} \qquad (1.14)$$

El valor de θ_2 es la suma de dos ángulos: α y β. El valor de α se encuentra utilizando la ley de los cosenos:

$$\alpha = \arccos\left(\frac{l_1^2 + d_g^2 - l_2^2}{2 l_1 d_g}\right) \qquad (1.15)$$

Para el valor de ß, se tiene que la distancia pequeña formada por el P_f y l_2 a lo largo del eje z en el nuevo plano;

$$d_p = P_{fz} - l_2 \qquad (1.16)$$

utilizando las ecuaciones 1.14 y 1.16:

$$\beta = \arcsin\left(\frac{d_p}{d_g}\right) \qquad (1.17)$$

Dando por resultado:

$$\theta_2 = \alpha + \beta \qquad (1.18)$$

La diferencia entre el valor del ángulo $\theta_{2\ inicial}$ y el ángulo θ_2 generado por las ecuaciones es:

$$\Delta\theta_2 = \theta_2 - \theta_{2\ inicial} \qquad (1.19)$$

y representa el movimiento físico de la segunda articulación.

Para θ_3; (ver fig.1.7).

El valor de γ, por ley de los cosenos:

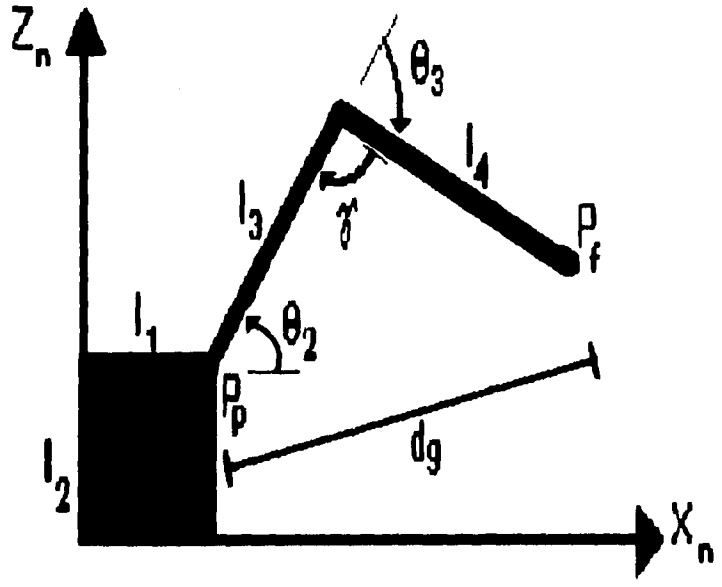


Figura 1.7. Obtención de γ.

$$\gamma = \arccos\left(\frac{l_1^2 + l_3^2 - d_g^2}{2 l_1 l_3}\right) \qquad (1.20)$$

$$\theta_3 = 180 - \gamma \qquad (1.21)$$

Considerando que $\theta_{3\,mod}$ representa el cambio proporcional que sufre θ_3 al cambio de θ_2, la diferencia entre el valor inicial de θ_3 ($\theta_{3\,inicial}$) y su valor final quedaría definida por:

$$\Delta\theta_3 = \theta_3 - \theta_{3\,mod} \qquad (1.22)$$

La relación entre $\theta_{3\,mod}$ y $\theta_{3\,inicial}$ se aprecia mejor de manera gráfica en la fig. 1.8.

donde:

A es el ángulo que se conserva durante el movimiento de θ_2

P_m es la posición de l_4 después de haber llevado $\theta_{2\,inicial}$ a θ_2.

Para obtener la relación algebraica se tiene:

$$\theta_{3\,inicial} = 180 - \gamma_{inicial} \qquad (1.23)$$

por ángulos internos del triángulo:

$$\gamma_{inicial} = 180 - \theta_{2\,inicial} - A \qquad (1.24)$$

$$\gamma_{mod} = 180 - \theta_2 - A \qquad (1.25)$$

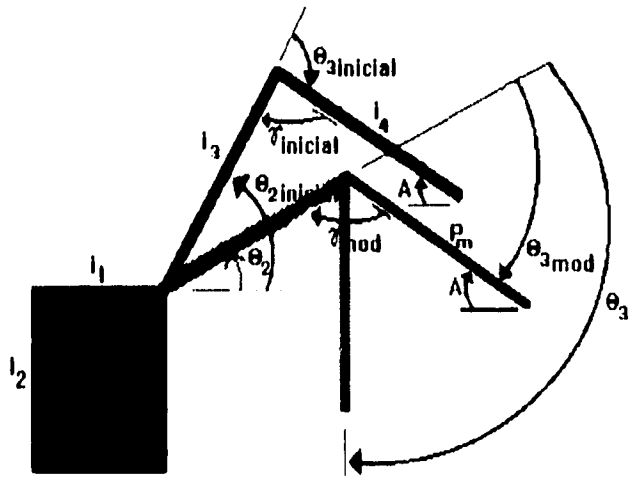


Figura 1.8. Cambio de θ_3 proporcional al movimiento de θ_2.

utilizando la ec.1.21:

$$\theta_{3\,mod} = 180 - \gamma_{mod} \qquad (1.26)$$

igualando la ec.1.24 con la ec.1.25 y simplificando:

$$\theta_{2\,inicial} - \theta_2 = \gamma_{mod} - \gamma_{inicial} \qquad (1.27)$$

utilizando la ec. 1.19 :

$$-\Delta\theta_2 + \gamma_{inicial} = \gamma_{mod} \qquad (1.28)$$

sustituyendo la ec. 1.26 en 1.22:

$$\Delta\theta_3 = \theta_3 - (180 - \gamma_{mod}) \qquad (1.29)$$

de la ec.1.28:

$$\Delta\theta_3 = \theta_3 - (180 + \Delta\theta_2 - \gamma_{inicial}) \qquad (1.30)$$

despejando y sustituyendo $\gamma_{inicial}$ de la ec. 1.23, obtenemos el valor de $\Delta\theta_3$, en términos de valores conocidos:

$$\Delta\theta_3 = \theta_3 - \theta_{3inicial} - \Delta\theta_2 \qquad (1.31)$$

y representa el movimiento que deberá realizar la tercera articulación.

Hasta este punto del modelado, el manipulador ya está posicionado e inclinado en el punto deseado, utilizando la ec.1.6:

$$\Delta \theta_4 = \theta_4 - \theta_{4\,inicial} \qquad (1.32)$$

Ahora sólo falta una parte de la orientación y es el ángulo θ_5 que nos indica que tan rotado lo queremos en ese punto. Este valor es dato y tiene un intervalo de:

$$0 < \theta_5 < 180 \qquad (1.33)$$

y además:

$$\Delta \theta_5 = \theta_5 - \theta_{5\,inicial} \qquad (1.34)$$

Para que el punto deseado se encuentre dentro del campo de trabajo según las características únicas del manipulador, el valor de la distancia d_g tiene un intervalo. Para obtener el límite; se utilizan las ecuaciones 1.11 y 1.16 dando por resultado:

$$\frac{l_3^2 + d_g^2 - l_4^2}{2 l_3 d_g} \geq 1 \qquad (1.35)$$

y

$$\frac{l_3^2 + l_4^2 - d_g^2}{2 l_3 l_4} \geq 1 \qquad (1.36)$$

despejando y resolviendo para d_g; se tiene:

$$(l_4 - l_3) \leq d_g \leq (l_3 + l_4) \qquad (1.37)$$

Este límite evitará colocar al manipulador en un punto incongruente con sus características.

Referencias

[1]. John J.Craig. *Introduction to Robotics Mechanics and Control.* Ed. Addison-Wesley,México, Second Edition.

[2] Asada Slotine J. *Robot Control and Analisis.*

[3] Antti J.Koivo. *Fundamentals for Control of Robotic Manipulators.*

2. Capítulo II

AMBIENTE WINDOWS

Introducción

En este capítulo se presenta la realización de un programa de control con una interfaz gráfica fácil de entender y la obtención de datos que permitan al manipulador, colocado en un punto de referencia, desplazarse a un punto designado por el usuario.

Considerando que el usuario puede ser una persona que no tenga conocimientos de robótica, resulta conveniente que la pantalla represente el objetivo del sistema lo más claro posible. Para ello, se reservará un área dentro de la pantalla para representar tanto la posición de referencia del manipulador como el espacio que éste puede alcanzar. Utilizando como comunicación el "ratón", el usuario sólo deberá especificar la coordenada a la que desea mover el manipulador.

Representación Gráfica

Los datos necesarios para que un algoritmo en base al modelado forma cerrada (de acuerdo al capítulo I) funcione, son: θ_i y $P_d(x,y,z)$ respecto al sistema de referencia 1.

Debe recordarse que MatLab realizará la representación gráfica para el usuario y como salida dará los datos generados por el modelado $(\Delta\theta_1,\Delta\theta_2,\Delta\theta_3,\Delta\theta_4,\Delta\theta_5)$ y la secuencia de movimiento de cada una de las articulaciones. Los datos restantes para mover al manipulador (cantidad de movimiento por articulación, sentido, señal a motores, etc.) son manejados desde Borland C++ debido a que la comunicación entre la PC y la interfaz electrónica (ver capítulo siguiente para más detalles) no es realizada desde MatLab sino desde Borland C++ V4.0.

La solución para la representación del Campo de Trabajo de Robot en la pantalla de la computadora, fue graficar el área de trabajo utilizando dos vistas, Vista Lateral y Vista Superior, de tal manera que al concluir la selección de un punto en cada uno de las vistas se obtiene el valor de un punto en el espacio.

La Vista Lateral del robot representa la altura máxima y mínima que pueden ser alcanzadas; y, por las características propias del robot no se puede tener un desplazamiento igual en la Vista Superior para todas las alturas. La vista superior deberá representar solamente el desplazamiento permitido dada una altura.

Se consideró dentro del diseño, que el usuario tuviera acceso a una pequeña explicación de lo representado gráficamente y, fue conveniente una demostración así como mensajes de lo que está sucediendo en el momento.

La mayoría de los manipuladores cuentan con un Control Manual (Teach Pendant). En nuestro caso, el Control Manual será representado gráficamente, recordando que, el movimiento del manipulador utilizando el Control Manual será independiente del movimiento realizado al ejecutar la opción de resolver el problema de cinemática inversa.

El funcionamiento del Control Manual será simulando botones de presión, teclas, barras de desplazamiento y radio buttons (utilerías de MatLab); cada uno de estos controles genera un bloque de datos, los cuales serán procesados por el programa en lenguaje C.

MatLab tiene herramientas que facilitan la realización de una GUI (Graphical User Interface) pero, para tener una mayor velocidad de proceso, resulta conveniente vectorizar las operaciones a realizar.

Diseño

La pantalla se encuentra dividida en dos secciones: una dedicada exclusivamente al control del robot por medio de la selección de dos puntos con el "ratón", y la segunda corresponde al control del robot por medio de su Control Manual.

a)Control Manual

La parte derecha contendrá la representación gráfica del Control Manual (ver fig. 2.1).

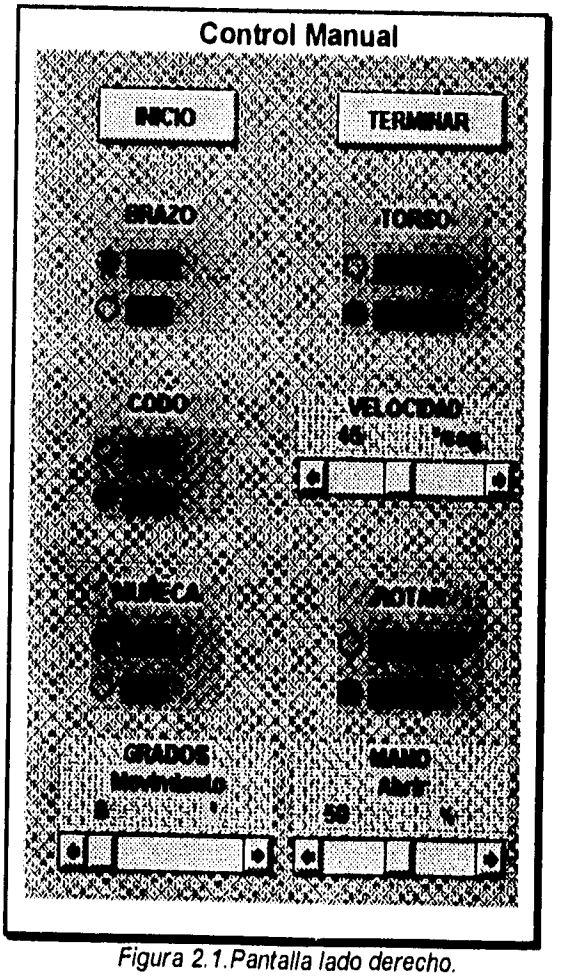


Figura 2.1.Pantalla lado derecho.

y se consideran las siguientes opciones al momento de presionar el botón de control correspondiente:

Inicio: Moverá al robot a la posición de referencia.

Torso:

Izq.- Rota el torso del robot a la izquierda.
Der.- Rota el torso del robot a la derecha.

Brazo:

Sube.- Mueve la segunda articulación del robot hacia arriba.
Baja .- Mueve la segunda articulación del robot hacia abajo.

Codo:

Sube.- Mueve la tercera articulación del robot hacia arriba.
Baja .- Mueve la tercera articulación del robot hacia abajo.

Muñeca:

Sube.- Mueve la mano del robot hacia arriba.
Baja .- Mueve la mano del robot hacia abajo.

Rotar:

Izq. .- Rota la mano del robot a la izquierda.
Der.- Rota la mano del robot a la derecha.

Grados: Controla la cantidad de movimiento de las cinco articulaciones.

Mano: Controla la cantidad que se abrirá o cerrará la mano.

Velocidad: Controla la velocidad de los seis motores del manipulador.

Salir: Termina la ejecución del sistema y regresa el control a Windows.

Para poder mover al manipulador utilizando el Control Manual, se debe seleccionar en la barra deslizable nombrada "Movimiento en Grados" la cantidad en grados que uno desea mover a la articulación; este valor puede ir desde tres hasta 100 grados, después debe seleccionarse el control correspondiente a la articulación[2].

Cada una de las articulaciones cuenta con un límite y algunas con una restricción.[3]

---

[2]Si selecciona alguna articulación y la barra deslizable de Movimiento tiene un número menor a 3 el movimiento no se realiza; lo mismo sucede para el control de velocidad.

[3]Al movimiento del brazo, el codo trata de conservar su ángulo y al movimiento del codo, la inclinación de la mano trata de conservar su ángulo.

Por ejemplo, si el manipulador se encuentra en su posición inicial y se desea mover al codo a través de su intervalo de 0 a 155°; esto debe hacerse de la siguiente manera:

Considerar que, cada vez que el codo suba, la mano tratará de conservar su ángulo, pareciendo que la mano se está moviendo hacia abajo; es decir, se va inclinando y llegará un momento en el cual la inclinación de la mano presione el sensor de que no puede seguir el movimiento, en ese momento, el codo dejará de moverse. Para evitar esta limitación, deberá moverse primero la inclinación de la mano hasta el límite contrario; es decir, hacia arriba y después empezar con el movimiento del codo.

1.- Seleccionar el máximo valor que permite el deslizador de movimiento(100)
2.- Seleccionar el control del codo hacia arriba.
3.- Seleccionar el valor de 55 grados en el deslizador de movimiento.
4.- Seleccionar el control del codo hacia arriba.

La barra deslizable para la velocidad tiene un intervalo de 0 hasta 90 grados por minuto; no permitiendo el movimiento si su valor es menor a 15 grados por minuto; esto es debido a que la velocidad sería demasiado baja y por la posición del manipulador esa velocidad no es suficiente para moverlo. En cualquier otro caso el movimiento está permitido.

Si el usuario trata de llevar alguna articulación, por ejemplo al brazo, más allá de su límite, un sonido acompañará al mensaje "Error en el Brazo", y el movimiento no es realizado.

b) Campo de Trabajo.

En cuanto a la parte izquierda de la pantalla, ésta es ocupada por la representación gráfica del Campo de Trabajo del Manipulador.

Al analizar el campo de trabajo del manipulador, se obtuvo la representación gráfica mostrada en las figuras 2.2 y 2.3.

La figura 2.2 es una vista lateral del manipulador. El área sombreada representa el área que puede ser alcanzada de forma vertical por el manipulador, en otras palabras, es la representación de todas las alturas que puede alcanzar.

Vista Lateral

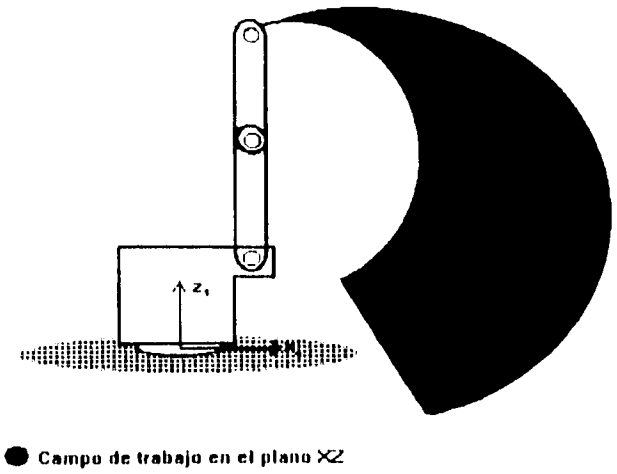


Figura 2.2.Manipulador Vista Lateral.

La siguiente gráfica es una vista desde el eje z del sistema de referencia 1 (ver Capítulo I), y representa las longitudes máximas y mínimas que el manipulador puede tener. (ver fig. 2.3).

Vista Superior

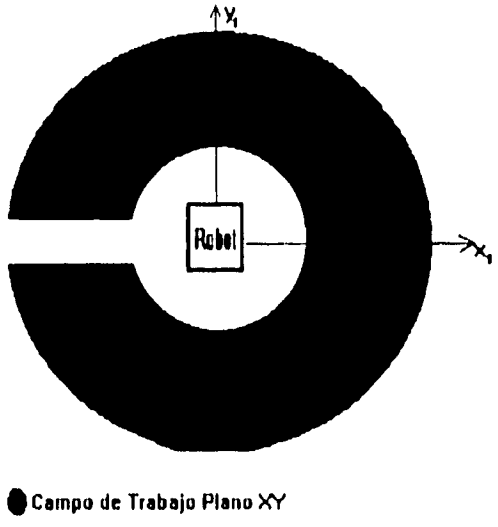


Figura 2.3.Manipulador Vista Superior.

La representación real en pantalla del Campo de Trabajo del manipulador es como lo muestra la figura 2.4. La parte superior de la figura muestra el campo de trabajo del manipulador desde una vista lateral así como

al manipulador en su posición de referencia. De la misma manera, la parte inferior muestra el campo de trabajo del manipulador desde una vista superior y al manipulador en su posición de inicio.

Dentro de esta área reservada de la pantalla, además de poder realizar la solución al problema de cinemática inversa se cuenta con las siguientes opciones:

Inicio:

Regresa al manipulador a su posición de referencia.

En el caso de un problema de cinemática inversa es necesario tener una referencia de donde se encuentra el manipulador y poder calcular cuánto le faltaría para llegar al punto indicado. Esta posición puede ser cualquiera, pero en base a ella se realizan todos los cálculos para el movimiento; la posición de referencia para el manipulador en cuestión es como se indica en la figura 2.4. La posición fue elegida considerando la colocación de los sensores del manipulador.

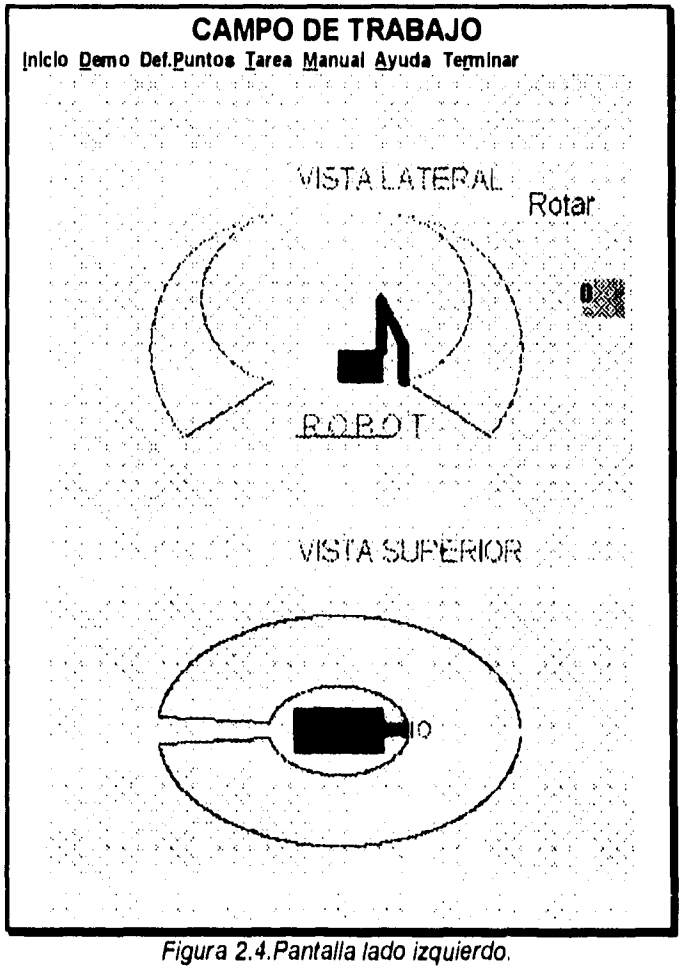


Figura 2.4.Pantalla lado izquierdo.

Demo:

Verifica que el manipulador se encuentre en su posición de referencia y después sigue una línea recta, esta línea se encuentra dentro de su área de trabajo.

Def.Puntos:

Esta opción resuelve el problema de cinemática inversa. Pedirá al usuario un punto dentro de el campo de trabajo del manipulador y llevará al manipulador a esa posición. La secuencia es la siguiente:

1.- Aparece el mensaje de que el manipulador está verificando que se encuentra en su posición de referencia para poder iniciar el movimiento

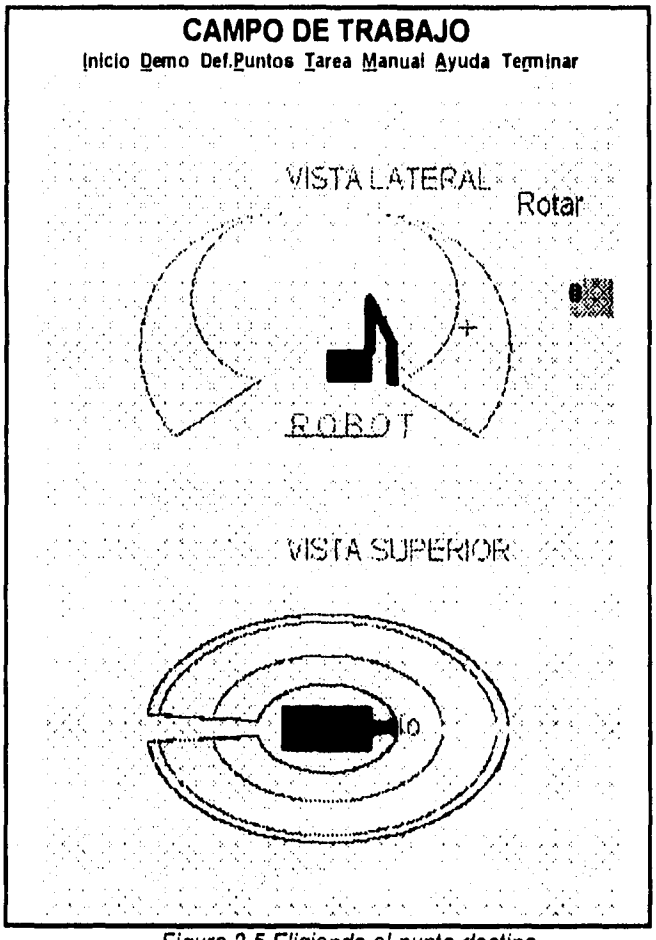


Figura 2.5.Eligiendo el punto destino.

2.- Pide al usuario el nombre del archivo en el cual desea se guarden los datos de los movimientos que realizará el manipulador.

3.- Aparece el mensaje de seleccionar utilizando el "ratón", la altura del punto destino, para ello se utiliza la vista lateral (fig. 2.5) del manipulador y el punto es marcado con una cruz. El punto debe encontrarse dentro del área marcada por las líneas; en caso de elegir un punto fuera del área, se pide otro hasta que el punto seleccionado se encuentre dentro del área permitida.

4.- Dada la altura elegida, el manipulador sólo puede extenderse cierto intervalo, este intervalo es marcado con círculos dentro del área permitida (ver figura 2.5) y aparece el mensaje de seleccionar la extensión del punto destino dentro del área permitida. El usuario deberá seleccionar un punto dentro de esa área, en caso de elegir algún punto fuera de esa nueva área, éste será ignorado.

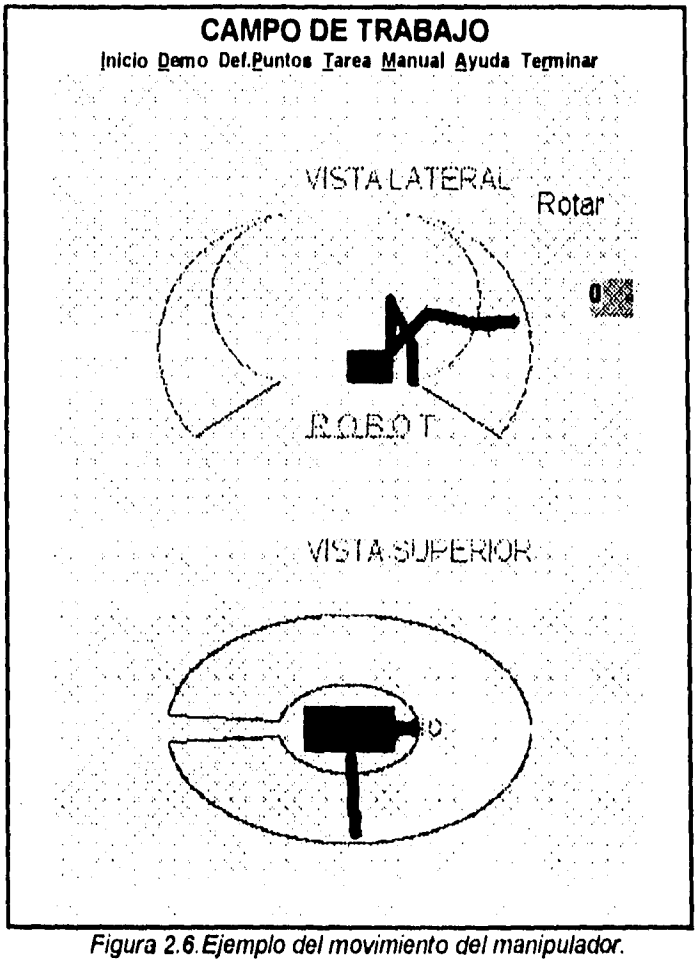


Figura 2.6.Ejemplo del movimiento del manipulador.

5.- El programa procesa los datos y realiza los cálculos necesarios para poder activar las articulaciones necesarias al mismo tiempo. En caso de existir algún error con el manipulador se desplegará un mensaje, en cualquier otro caso, el manipulador se moverá al punto elegido y en la pantalla se

presentará cómo debió quedar el manipulador después de haber realizado el movimiento (ver figura 2.6).

6.- Se despliega un mensaje con la coordenada que fue elegida y se pregunta si se desean más datos, de ser así, el proceso se repite hasta que el usuario lo desee.

Tarea:

Repite una secuencia de movimientos previamente realizados.

Su funcionamiento es el siguiente:

1.- Se despliega un mensaje indicándole al usuario que debe elegir el nombre de algún archivo de datos. Este archivo tuvo que ser generado previamente por la opción de Def Punto o Manual.

2.- Verifica que el manipulador se encuentre en su posición de referencia y se inicia el movimiento del manipulador. Esta opción es utilizada para repetir movimientos o tareas varias veces.

El usuario puede elegir el archivo Tomar.Txt, el cual activa al manipulador para tomar y dejar un objeto.

Manual:

A diferencia de la opción Def. Punto en la cual se utiliza el "ratón" para definir los puntos; en esta opción se utiliza el teclado. Se pide el nombre del archivo de datos, se pide la coordenada del punto destino. Este punto debe encontrarse dentro su área de trabajo, y al igual que en la opción Def. Punto; si el punto es erróneo el mensaje es desplegado.

Ayuda:

Despliega la información mostrada en el siguiente recuadro.

> Este sistema está diseñado para controlar un manipulador de cinco grados de libertad utilizando la computadora. Se resuelve el problema de cinemática inversa y cuenta además con un Control Manual.
>
> El manipulador es de marca Zenith y utiliza un sistema de poleas para efectuar dos de sus cinco Movimientos independientes. (Ver Rotación e Inclinación del Efector Final o Mano). Debe evitarse en lo posible el tensar demasiado este cable porque puede romperse; en ese caso debe colocarse otro siguiendo las indicaciones del Apéndice B incluido en el trabajo escrito. Sin olvidar que esto provocará un cambio

en los movimientos del manipulador y quizá se requiera un ajuste dentro del software.

Se necesita tener una tarjeta de expansión utilizando un circuito integrado 8255 insertada a la computadora; y cuya salida debe ser conectada al manipulador utilizando un conector DB25. El segundo conector (tipo EIS) del manipulador debe ser conectado al circuito AM9513, insertado previamente en la computadora.

La opción DEFINIR PUNTOS es la que permite seleccionar un punto para mover al brazo de su posición inicial a la definida en la pantalla. Para ello se cuenta con 2 gráficas que muestran el campo de trabajo del robot desde dos vistas, vista superior y vista lateral; las regiones dentro de las curvas representan el espacio en el cual el robot se puede mover. Para mover al brazo mecánico se activa esta opción y se selecciona una coordenada (dentro del espacio permitido); resultando la definición de un punto en 3 dimensiones.

La opción DEMO es una demostración de algunos movimientos que el brazo puede realizar.

La opción INICIO, coloca al brazo en su posición inicial o más recomendable para iniciar una tarea especifica.

La figura de la derecha corresponde al Teach Pendant del brazo, o CONTROL MANUAL. El control de cada uno de los motores que forman al brazo mecánico se realiza de manera independiente, permitiendo mover al brazo articulación por articulación.

La opción TAREA ejecuta los movimientos que ha hecho el manipulador y los ejecuta automáticamente después de elegir el archivo adecuado.

En el espacio denotado como Rotar se escribe el ángulo de rotación de la mano.

La opción MANUAL resuelve también el algoritmo de cinemática inversa pero a diferencia del Definir Puntos en éste se utiliza el teclado para dar los datos.

La opción de AYUDA despliega esta información.

La opción TERMINAR lleva al manipulador a su posición inicial y termina el programa.

Terminar:

Lleva al manipulador a su posición de referencia, cierra la aplicación y regresa el control a Windows.

Implementación

Los diagramas de flujo para el desarrollo del algoritmo se muestran en las figuras 2.7 y 2.8.

Diagrama de flujo[Torso, Brazo, Codo, Rotar, Muñeca y Mano].

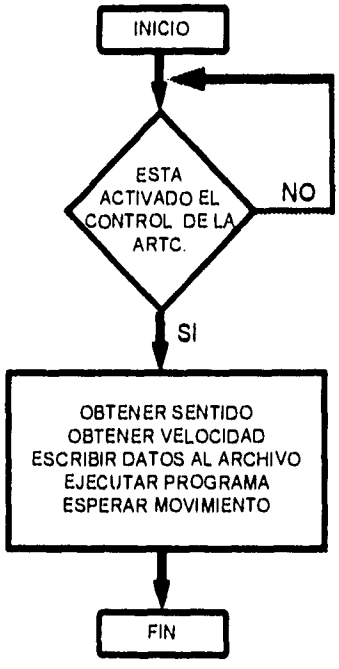


Figura 2.7.Diagrama para cada articulación utilizando el Control Manual.

La figura 2.7 representa el diagrama básico para el control de cada una de las articulaciones del manipulador así como de la mano o efector final. Se pregunta una vez por articulación; de tal manera que al momento de seleccionar el control correspondiente del Control Manual la articulación es activada.

En la figura 2.8 está representado el diagrama general que da solución al problema de cinemática inversa. Las ecuaciones utilizadas se pueden consultar en el capítulo 1, y los programas se encuentran totalmente documentados.

Diagrama de Flujo para el Campo de Trabajo

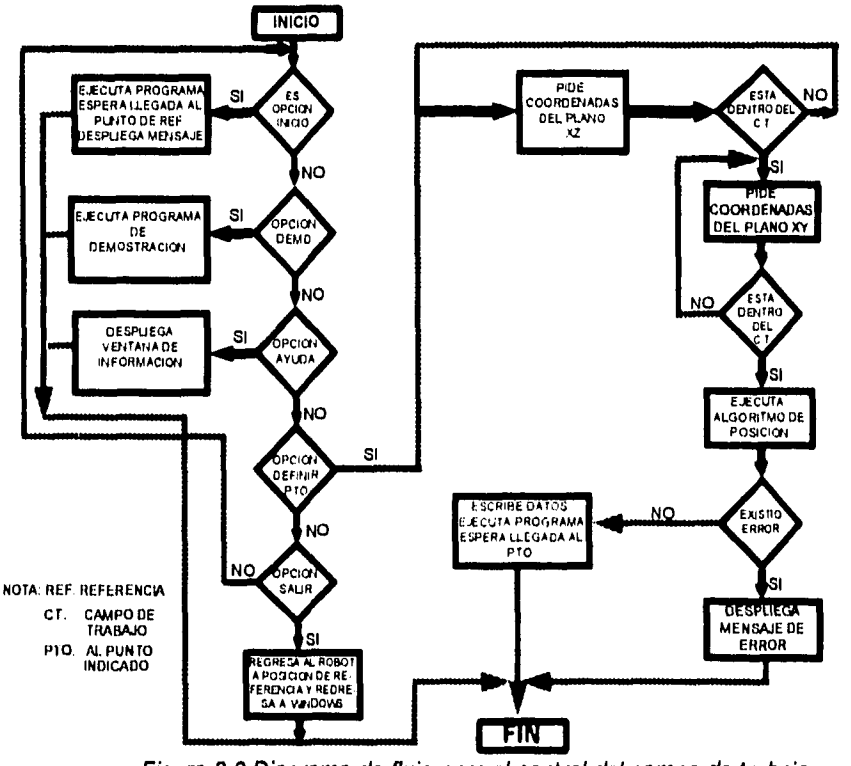


Figura 2.8 Diagrama de flujo para el control del campo de trabajo

Después de implementar los diagramas de flujo para MatLab y realizarle pruebas de funcionamiento; la codificación final del programa principal y de las funciones necesarias es la siguiente.

Programas y funciones en MatLab

```
%%Modificado MAYO 96
%%Programado por Norma Isabel Ortega Martinez
%% Ene 1995 Última modificación Marzo 96
function [angul,Ptopp,continuar] = angprog(angulosan,Ptop,Ptof,t5,archivo);
%%function [angul,Ptopp,continuar] = angprog(angulosan,Ptop,Ptof,t5,archivo);
%%    Donde:
%%      angul     Contiene los angulos del algoritmo.
%%      anguro    Son los angulos a mandar al robot.
%%      Ptopp     Es el punto en otro plano donde el
%%                punto pequeño  quedo.
%%      angulosan  Son los angulos anteriores.
%%      Ptop       Es el punto pequeno anterior.
%%      Ptof      Es el punto final al que se quiere llegar.
%% t5       Angulos de rotacion
```

```
% Archivo   Nombre del archivo para guardar datos.
%************************OBTENCION DE TETA1*************************

if ((Ptof(2) == 0) & (Ptof(1) < 0))    % x negativa, y =0
          ante1 = 160;
elseif ((Ptof(1) < 0) & (Ptof(2) > 0))  % x negativa, y positiva
   aux1 =atan(Ptof(2) /(Ptof(1)));
   ante4 = cte*aux1;
   if (ante4 > -20)                % por no dar vuelta completa
         ante1 = 160;
   else
        ante1 = 180+ ante4;
   end;
elseif ((Ptof(1) < 0 ) & (Ptof(2) <0))  % x negativa, y negativa
   aux1=atan(Ptof(2) /(Ptof(1)));
   ante4= cte*aux1;
 if (ante4 < 20)
     ante1 = -160;
   else
         ante1 =-180+ ante4;
   end;
else                           % cualquier otro caso
   aux1=atan(Ptof(2) /(Ptof(1)+0.00001));
   ante1= cte*aux1;              % ante1 contiene el valor del angulo t1
end;                           % dado por el algoritmo


%********Angulos a moverse y punto pequeño**********************************
anguro(1) = (ante1 - angulosan(1))   %a moverse en grados
aux2 = ((2*pi)/360)*anguro(1);       %en radianes
Ptopp =[cos(aux2), -sin(aux2),0;sin(aux2), cos(aux2),0;0,0,1]*Ptop

          %Punto pequeño ya rotado
torso =(((2*pi)/360)*ante1)          %algoritmo en radianes

%*********Se busca el valor adecuado de tet4**********************************

teta4 = parat4(Ptof,Ptopp)
%disp('es el elegido en grados');
teta4 = ((2*pi/360)* teta4);             %radianes

%*********Obteniendo el punto en la muñeca************************************
Ptof = Ptof-16*[cos(teta4)*cos(torso);cos(teta4)*sin(torso);sin(teta4)]

%******Obtencion de los ANGULOS RESTANTES*************************************
dg = sqrt((Ptof(1)-Ptopp(1))^2 + (Ptof(2)-Ptopp(2))^2 +(Ptof(3)-Ptopp(3))^2)

a = (dg /44);
b = (Ptof(3) - 14)/dg;
c = ((968-dg^2)/968);

if (((a <= 1.000001) &  (a >= -1.000001)) & ((b <= 1.0000009) & (b >= -1.0000009)) & ((c <= 1.000001) &
(c >= -1.000001)))
  continuar = 1;
```

```
else
 %disp('serian complejos');
 continuar = 0;
end;
if ((dg <= 44.000009) & (dg>9.4) & (continuar == 1))
   alfa = cte * acos(a);              % a = dg /44
                                %dp = (Ptof(3)-14);
   beta = cte * asin(b);              % b = (Ptof(3)-14)/dg
          ante2 = alfa+beta;
   gama = cte * acos(c);               % c =(968-dg^2)/968
   ante3 = 180 - gama;
          t4 = cte*teta4;                  % en grados
   angul =[ante1;ante2;ante3;t4;t5];    % angulos del algoritmo

   anguro(1) = ante1 - angulosan(1);    % angulos a mandar para mov.
   anguro(2) = ante2 -angulosan(2);
   anguro(3) = ante3 -angulosan(3) - anguro(2);
   anguro(4) = t4 - angulosan(4);
   anguro(5) = t5 - angulosan(5);
   anguro    = fix(anguro);
  %******Para limitantes mecánicas*********************************************

          if (angul(2)<-40) | (angul(2)>90.5) | (angul(3) < 0) |(angul(3) >= 155.3)| ...
          (angul(4) < -94) |(angul(4) >90) |(abs(anguro(5))>160)
                venerror;
          end;
%**********Inicia comunicacion con el exterior*************************
         if (continuar == 1 ),
          fid1 = fopen('c:\lan\norma\progmat\angupru.txt','w');
          fclose(fid1);
          %ooooooooooooooo
          %Archivo para ejecutar el movimiento.
          !c:\lan\norma\progc\prucams.pif
          fid1 = fopen(archivo,'a');
          count1 = fprintf(fid1,'%f\n',anguro);
          fclose(fid1);
           end;
else
  venerror;                        % Ventana de error
  continuar = 0;                    % Error por distancia.
end;
%*************** Termina ANGPROG.M *********************
%Modificacion MAYO 96
%Programado por Norma Isabel Ortega Martinez
%*******Para llegar a su posicion inicial********
%function sal = fhome(act)
% si el valor de act es 1 aparece el mensaje de
%que llego a home o 0 no aparece  el mensaje

!c:\lan\norma\progc\casitas.pif
close(avises);
sal = errcasa;
%*****************Termina fHome.M
function teenc = parat4(Ptodes,ptop);
```

```
%Programa realizado por Norma Isabel Ortega Martinez.
%Modificado Feb 1996
%Fancion que encontrara el valor  adecuado de teta4
%function teenc = parat4(Ptodes,ptop);
%Ptodes   Es el punto real al que se quiere llegar
%ptop     Es el punto pequeño ya rotado

dn = sqrt((Ptodes(1)-ptop(1))^2 + (Ptodes(2)-ptop(2))^2 +(Ptodes(3)-ptop(3))^2);
tetamover = (-0.776*(dn) + 46.60)*((2*pi)/360);
oper = (Ptodes(3) - 14) / (dn +.000001);
if ((oper < 1) |(oper > -1))
  tetalin = asin(oper);
  teenc =  (tetalin-tetamover);
  teenc = (teenc*(360/(2*pi)));          % en grados
else
          teenc = 0;
          disp('serian complejos');
end
%*********Termina Parat4 M
%************************DEFPF*****************************
%Modificado en MAYO 96
%Programado por Norma Isabel Ortega Martinez
%Programa  para resolver el problema de cinemática inversa.
%**** Dado un punto encontrar el valor de los ángulos.*****
%Utilizadno el mouse para obtener los datos.
% *****************Incia en su posicion de referencia.******
%pause off;
clc;
homereal = [28.29;0;0];  %lo lleva a su posicion inicial
errhome = fhome(0);
fvenguard;             %ventana de aviso para guardar datos
namefile=guarda;     %Archivo elegido
fid1 = fopen(namefile,'w');
           count1 = fprintf(fid1,'%f\n\r',");
           disp('limpia el archivo elegido');
fclose(fid1);
if errhome == [1;2;3;4;5;6]
          homep = [28.29; 0; 0];     %punto de inicio en centimetros
          ptoant = homep;
          angulosant =[0;90;155;-90;0];
          ptopeq = [19;0;14];
          continua = 1;
          xprim = ptoant(1);
          s1  = 'red';
          vez = 1;
          venta = 0;
          er=1;
          xf =28.29;
          yf=0;
          zf=0;
 while (continua ==1)
           if (er ==1)                % si no hay error es igual
                    puntoanterior = [xf;yf;zf];
           end
```

```
        while (sigue == 1)
          if (venta < 3)
                ventana(210,'el punto  XZ','para el recorrido');
          end
          [a,c] = ginput(1);
          [sigue,modman] = limalt(a,c,1);
          xprim = a;
          zf = c;
        end;
        plot(xprim,zf,'r+');              % xz deseado
        [ce,xpeq1,xgran1] = extdfun(zf,xprim,s1);  % Plano extension
                radio=80;
                contin = 1;
                while ( (radio <xpeq1) |(radio>xgran1) |(contin == 1) )
                        if (venta < 3 )
                                ventana(50,'el   punto   XY','en los circulos');
                        end
                        [a,b]= ginput(1);
                        contin = limext(a,b);
                        radio = sqrt(a^2 +b^2);
                end;
                venta = venta+1;
                xf = a; yf = b;
                plot(xf,yf,'r+');             %Punto deseado
                puntofinal =[xf;yf;zf]
                close(cea);
                close(ce);
                if (vez == 2)
                        close(al);
                        close(bl);
                        vez =1;
                end;
 %********************** Algoritmo para angulos.*****************************
                [anguli,ptop,er] = angprog(angulosant,ptopeq,round(puntofinal),te5,namefile);
                if (er == 1)
                        angulosant = anguli;        % Si es uno no existio error
                        ptopeq = ptop;
                        ptoant =[xf;yf;zf];
                else
                       angulosant = angulosant
                       ptopeq=ptopeq
                       ptoant =puntoanterior
                end;
                otropto;                    %Abre la ventana de otrop
        end;                        %while
        set(opn,'value',1);
        close(otro);
        if (vez == 2)
         close(al);
         close(bl);
        end;
else
        venerror;
end;
```

```
%************************************ Termina delpd
% ***** Control Manual**********
%Modificado Feb96
%Realiza la interface grafica del control manual
%Realizacion de cada uno de los controles para cada articulacion
%del manipulador
%Programado por Norma Isabel Ortega Martinez
%tema 468,40,300,450
tea = figure('position',[350,5 235 450],
                              'NumberTitle','off',
                              'MenuBar','none',
                              'Name','CONTROL MANUAL',
                              'Resize','off',
        'Color',[.8 .7 .8]);
 uicontrol(tea,
    'Style','push',
    'Position',[175 400 58 30],
    'String','TERMINAR',
    'CallBack','[illegible]');

tor_izq = uicontrol(tea,
    'Style','radio',
    'Position',[180 325 85 25],
    'BackgroundColor',[.2 .75 .7],
    'String','Izquierda',
    'CallBack',[
    'set(tor_izq,'Value',1),
 set(tor_der,'Value',[illegible]

 tor_der = uicontrol(tea,
    'Style','radio',
    'Position',[180 290 85 25],
    'BackgroundColor',[.2 .75 .7],
          'String','Derecha',
    'Value',0,

 CallBack',[set(tor_der,'Value',1), set(tor_izq,'Value',0) [illegible]
 [illegible]

 % [illegible]
 %----------------TRASLACION

 tr_[illegible] = uicontrol(tea,
    'Style','text',
    'String','TRASLACION',
          'Position',[illegible],
    'BackgroundColor',[illegible]

 uicontrol(tea,'Style','text','String',[illegible]
 tr_[illegible] = uicontrol(tea,
    'Style','slider',
          'Position',[illegible],
    'Value',[illegible],
    'CallBack',[illegible]
```

```
%********************************** Termina defpf
%***** Control Manual**********
%Modificado Feb96
%Realiza la interface gráfica del control manual
%Realización de cada uno de los controles para cada articulación
%del manipulador.
%Programado por Norma Isabel Ortega Martinez
%tenia 468,40,300,450
tea = figure('position',[350,5,285,450],...
                                    'NumberTitle','off',...
                                    'MenuBar','none',...
                                    'Name','CONTROL MANUAL ©',...
                                    'Resize','off',...
        'Color',[.5,.7,.8]);
 uicontrol(tea,...
    'Style','push',...
    'Position',[175,400,98,30],...
    'String','TERMINAR',...
    'CallBack','salida');

tor_izq =  uicontrol(tea,...
    'Style','radio',...
    'Position',[180,325,83,25],...
    'BackgroundColor',[.2,.75,.7],...
    'String','Izquierda',...
    'Callback',[...
    'set(tor_izq,"Value",1),
'set(tor_der,"Value",0),ftocobra(str2num(get(datgra,"String")),str2num(get(datv,"String")),0,1)']);

tor_der =  uicontrol(tea,...
    'Style','radio',...
    'Position',[180,300,83,25],...
    'BackgroundColor',[.2,.75,.7],...
        'String',' Derecha',...
    'Value',1,...

'Callback',['set(tor_der,"Value",1),set(tor_izq,"Value",0),'ftocobra(str2num(get(datgra,"String")),str2num(get
(datv,"String")),1,1)']);

%... código similar para las demás articualciones.
%-------------------GRADOS

txt_gra = uicontrol(tea,...
    'Style','text',...
    'String','GRADOS',...
        'Position',[20,65,123,20],...
    'BackgroundColor',[.4,.84,.9]);

 uicontrol(tea,'Style','text', 'String',' Movimiento','Position',[20,50,123,20],  'BackgroundColor',[.4,.85,.9]);
sli_gra =  uicontrol(tea,...
    'Style','slider',...
        'Position',[20,15,123,20],...
    'Value',0,...
    'Callback', 'set(datgra,"string",'num2str(100*get(sli_gra,"Val"))')']
```

```
   'Callback',[...
   'datomano = get(datm,"String");,'...
   'set(datm,"string",',...
   'num2str(100*get(sli_mano,"Val"))),',...
   'fmanof(str2num(get(datv,"String")),str2num(get(datm,"String")),str2num(datomano))'],...
          'ButtonDownFcn','fmanoar');
%lo anterior indica que se mueve el slider arrastrando el mouse
%pero para ejecutar la rutina se hace clic con el boton izquierdo del mouse
%****** Para conexion del puerto*******
%******Configuracion del puerto
%Modificado Feb96

dos('c:\lan\norma\progc\confi |');
text('Position',[.1 .8],'String','  Puede Conectar ');
text('Position',[.1 .5],'String',' el Manipulador al ');
text('Position',[.1 .2],'String','    puerto 300 H.');
dos('c:\lan\norma\progc\confi.exe |');
pause(3);
close(configu);
%*************termina venconfi
%****** Para trayectoria o demostracion
%Modificacion Feb96
%Realiza una demostracion de movimientos
%donde se encuentran los datos para el demo
fid1 = fopen('c:\lan\norma\progmat\nomarchc.txt','w'); %arch para movim.
 fclose(fid1);
!c:\lan\norma\progc\tareas.pif
%Termina demo.

%************************************************************
%Programa que  ejecutara una trayectoria con los datos dados
%Programa modificado en Feb 96
%Pedira el  nombre del archivo que se desea ejecutar
%************************************************************
%fhome(0);
demo = figure('Position',[250 195 270 150],...
text('Position',[0  .8],'String','Elige el Archivo de');
close(demo);
caminomat = abrira      %path y nombre del archivo abierto
if caminomat ~= 0
        fid1 =fopen('c:\lan\norma\progmat\nomarchc.txt','w'); %arch para movim.
        fclose(fid1);
                %La seguridad e integridad del archivo dependera del usuario.
                %el manipulador solorealiza movimientos dentro de sus limites
                %Ejecuta el programa para el manipulador
        !c:\lan\norma\progc\tareas.pif
end;
%****** Termina Tarea.M  ******************
%Campo de Trabajo del Robot.
%Programado por NORMA ISABEL ORTEGA MARTINEZ
%Modificado MAYO 96
%Dibujara el area de trabajo del manipulador
rob = figure('Position',[5,5,340,450],...
homm = uimenu(rob,'Label','&Inicio','CallBack','fhome(1)');  % se regresa a su posicion
```

```
                                        %inicial
ejtra = uimenu(rob,Label','&Demo',CallBack','trayec');                  %Realiza una demostracion
def_pun = uimenu(rob, 'Label','Def. &Puntos', 'CallBack','defpf');      %algoritmo de cinematica inversa
ta = uimenu(rob,'Label','&Tarea', 'CallBack','tarea');                  %repeticion de movimientos
ma = uimenu(rob,'Label','&Manual', 'CallBack','manual');   %cinemática inversa desde teclado
ay = uimenu(rob,.'Label','&Ayuda',. 'CallBack','ayuda');                        %despliegue de ayuda
sali = uimenu(rob,Label','Te&rminar','CallBack','salida');              %regresa a home y sale de matlab
f2=text('Position',[.9 .9],'String','Rotar');
sha = alturadi;                 %se dibuja el plano de la ALTURA
s ='blue';
w = text(14,2,'Inicio');
set(w,'Color','w');
manipula([19;0;14],90,155,-90);
base;
q = text(-15,-20,'R O B O T');      set(q,'Color',[.2 .5 .5]);
q = line([-18,25],[-24,-24]);       set(q,'Color',[.2 .5 .5]);
hr,xpeq1,xgra1]= extdfun(-1,20,s);  %dibujo del plano de extensión
base2;
q=line([19;28.2],[0;0]);
set(q,'Linewidth',5);
set(q,'Color',[.1 .1 .8]);
r = text(14,3,'Inicio');            set(r,'Color',[.2 .5 .5]);
%**************************Termina  Campo de Trabajo
%Ejecuta el programa para mover al efector final
%Modificado Feb 96
!c:\lan\norma\progc\manom.pif
%*************Termina Fmanoar.M

function [h,xpeq,xgran] = extdfun(zf,xa,s);
%Modificado MAYO 96
%*****Para el  Plano de extension*********
%***** O  Vista Superior *****************
%Dibuja el area permitida dada una altura;
        xpeq = 0;
        xgran = 0;
        if ((0<=zf)&(zf<=74))
                xpeq = sqrt(1444 - (zf-36).^2)+19;
                xgran = sqrt(3600 - (zf-14).^2)+19;
        elseif (zf>=74)                 % Limites
                xpeq = 19;
                xgran = 19;
        elseif ((-45 <= zf) & (zf<0))
                xpeq = (zf -18.94) /(-0.669);
                xgran = sqrt(3600 - (zf-14).^2)+19;
        end;
if ((xpeq == 0) & (xgran == 0))
        disp('error de extension');
else
 ejexy = axes('position',[.1 .05 .8 .35]);
        xpeq = abs(xpeq);
        x1 =(xpeq-.5):.09:(xpeq);
        i=1:length(x1);
                y1 =sqrt((xpeq.^2) -(x1.^2));
        hold on;
```

```
        plot(x1,y1,s);
        plot(x1,-y1,s);
        plot(x2,y2,s);
        plot(x2,-y2,s);
        pl = line([-(xgran-.5),-(xpeq-.5)],[-y2(1),-y1(1)]);
        set(pl,'Color',s),
        sl= line([-(xgran-.5),-(xpeq-.5)],[y2(1),y1(1)]);
        set(sl,'Color',s);
        set(ejexy,'visible','off');
        set(ejexy,'Xlim',[-100,100],'Ylim',[-100,100]);
end
h = ejexy;
%******************** Termina funcion del plano de extension.
% Dibuja la vista lateral del manipulador
% Modificacion MAYO 96
% Programado por Norma Isabel Ortega Martinez
% Retorna el identificador de la figura creada.

ejexz = axes('position',[.1 .5 .8 .4]);

        y3 =-24.56:1:74;              % Area de Trabajo
        x3 =0;
        i=1:length(y3);
        x3 =sqrt(3600 - (y3-14).^2) +19;

        y4 =0:1:74;                   % Limite de Area
        x4 =0;
        i=1:length(y4);
        x4 =sqrt(1444 - (y4-36).^2)+19;

  plot(x3,y3,'b.');
  hold on;
  plot(-x3,y3,'b.');
  f= text(-17,90,'VISTA LATERAL');
  set(f,'Color',[.2 .5 .3]);
  plot(x4,y4,'b.');
plot(-x4,y4,'b.');
  xp =line([28.29,64.96],[0,-24.56]);
  set(xp,'Color','b');
  xp = line([-28.29,-64.96],[0,-24.56]);
  set(xp,'Color','b');
  set(ejexz,'visible','off');

h = ejexz;
% ***************TERMINA funcion alturadi.
```

3. Capítulo III

COMUNICACIÓN Y CONTROL

Introducción

La comunicación entre la PC y la interfaz electrónica puede realizarse utilizando el puerto paralelo de la impresora con el cual, aunque el número de salidas y entradas no es exactamente el requerido por el control del manipulador, puede descodificarse o multiplexarse hasta obtener las entradas y salidas necesarias, ésto implicaría el uso de descodificadores y multiplexores. Resulta poco flexible, ya que si en un futuro se desea incrementar el número de sensores en el manipulador, tal vez tanto el programa de control como el diseño electrónico deben modificarse para obtener más líneas de control.

Considerando lo anterior, la comunicación se realizará utilizando un puerto paralelo diseñado con base a un PPI (Programmable Peripheral Interface) D8255AC2 que permite tener un puerto paralelo de 25 pines ofreciendo la ventaja de poder configurarse como tres puertos diferentes de ocho bits cada uno, y además pueden ser utilizados como de entrada o salida según se requiera. En comparación con una tarjeta de expansión (otra alternativa), evita el uso de latches, descodificadores y señales de control extras, teniendo como resultado un puerto flexible y una interacción *Entrada/Salida* más fácil de manipular.

El manipulador cuenta con seis motores de corriente directa a 12V para realizar sus movimientos, cada uno de los motores cuenta con un codificador óptico, el cual nos puede indicar la cantidad de movimiento realizado.

El siguiente diagrama nos da una idea del ciclo de comunicación que existe entre la computadora y el manipulador.

Ciclo de Comunicación

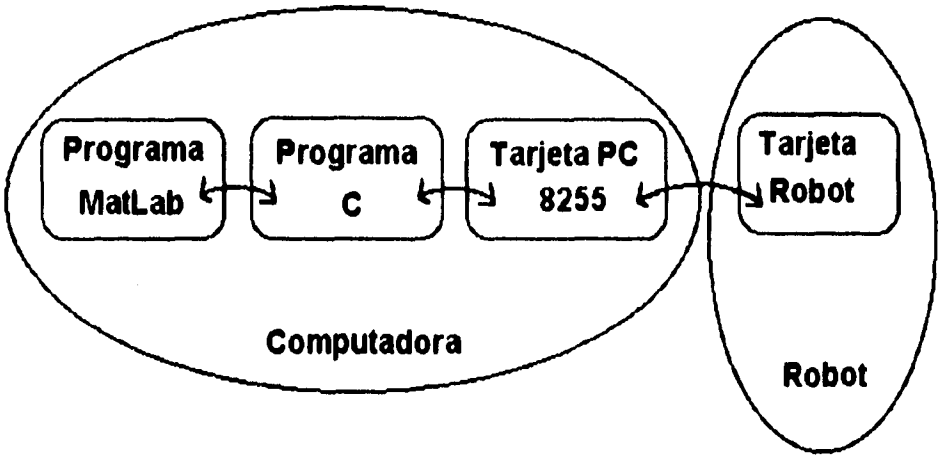


Diagrama de Comunicación PC<->Robot.

Análisis

Hasta el momento se cuenta simplemente con los resultados generados por el algoritmo en MatLab. Ahora debemos complementarlos para poder controlar al manipulador.

El problema no es solamente mover al manipulador; en realidad, son tres los problemas o casos a resolver:

Caso 1.- Al utilizar la opción Definir Punto, la posición seleccionada deberá ser alcanzable; en caso contrario se mostrará un mensaje de error.

En el capítulo anterior, al delimitar el campo de trabajo del manipulador se evita el envío de datos erróneos y se presenta el mensaje de error; solucionando, de esta manera el problema.

Caso 2.- Estando el manipulador en cualquier posición debe ser posible llevarlo a su posición inicial.

Caso 3.- Cuando se utilice el Control Manual, y el manipulador haya llegado a su límite, no se debe permitir el movimiento al menos que sea efectuado en el sentido contrario.

Los casos dos y tres deberán ser resueltos con el programa en MatLab, el programa de Control y Comunicación en lenguaje C y por la interfaz electrónica.

No se incluyen las otras opciones del menú de control (ver figura 2.4) como Tarea, Manual, etc. Debido a que la comunicación y control en éstas sólo es una variante de los tres casos mencionados anteriormente.

En cualquiera de los casos, para mover el manipulador es necesario saber cual o cuales articulaciones van a moverse, la cantidad de grados a moverse por cada una, el sentido, y la velocidad.

Una vez que esto se haya procesado de manera adecuada, tanto por la interfaz como por el programa de Control, y el manipulador se encuentre en movimiento, se verificará (utilizando los codificadores ópticos de cada motor) si la cantidad de movimiento corresponde a la cantidad deseada (teniendo un control realimentado) y hasta que se igualen u ocurra algún error, el problema termina y el manipulador estará en la posición deseada o se desplegará el mensaje de error.

Diseño

El diagrama a bloques para la comunicación desde la computadora hasta el manipulador se muestra en la figura 3.1.

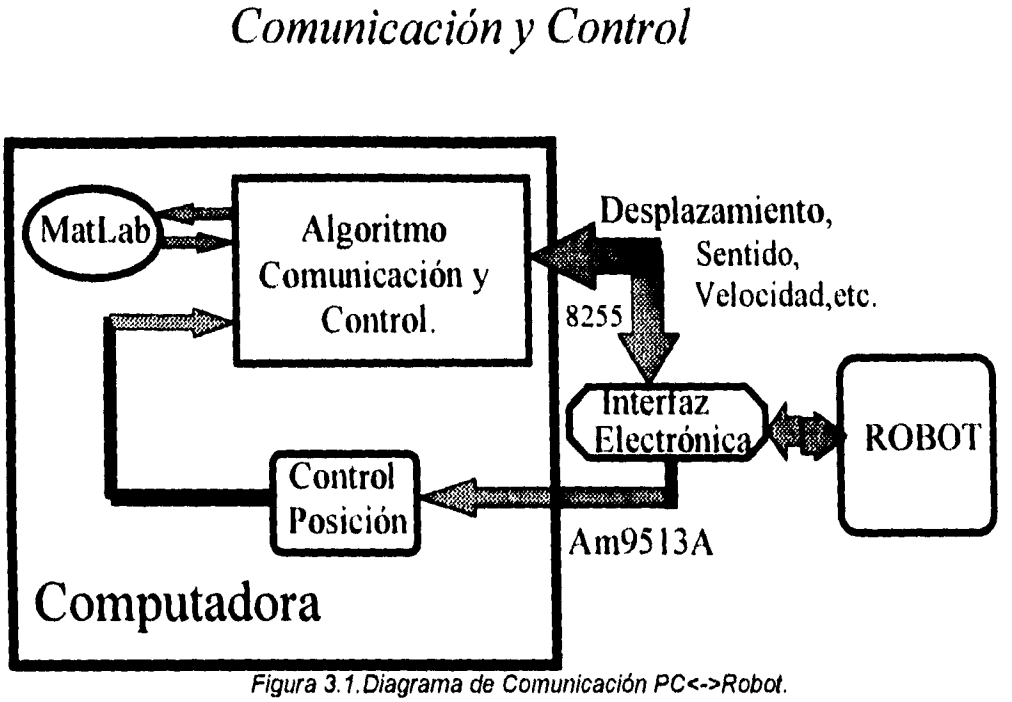


Figura 3.1.Diagrama de Comunicación PC<->Robot.

Se decidió colocar un circuito integrado AM9513A STC (circuito con cinco contadores controlados individualmente de 16 bits) como sistema redundante para tener mayor seguridad de que el manipulador llegue a la posición indicada. El procedimiento es indicado posteriormente.

Se utilizó un puerto paralelo con base en un PPI 8255 para obtener las señales necesarias de comunicación; para mayor detalle, el diagrama del D8255AC2 se encuentra en el Apéndice D.

El circuito integrado 8255 cuenta con 25 pines que pueden configurarse para tener tres puertos paralelos (A,B,C) de ocho bits cada uno. Su configuración dependerá de los datos que necesitamos de salida y de entrada.

La configuración del puerto paralelo se muestra en la tabla 3.1.

Tabla 3.1

Puerto y Número Pin	Función	Entrada/Salida
A0, 1	Dato1	S
A1, 2	Dato2	S
A2, 3	Dato3	S
A3, 4	Dato4	S

A4,5	Dato5	S
A5, 6	Dato6	S
A6, 7	Dato7	S
A7, 8	Dato8	S
B0, 9	SWT	E
B1, 10	SWB	E
B2, 11	SWC	E
B3, 12	SWR2	E
B4, 13	SWMM	E
B5, 14	SWMANO	E
B6, 15	LLegó	E
B7, 16	-	-
C0, 17	Control Dec1	S
C1, 18	Control Dec2	S
C2, 19	Control Dec3	S
C3, 20	Control Dec4	S
C4, 21	RST	S
C5, 22	-	-
C6, 23	-	-
C7, 24	-	-
25	GND	-

Tabla 3.1.Puerto Paralelo D8255AC2.

La configuración anterior está estrechamente relacionada con la interfaz electrónica (Capítulo 4); pero por el momento sólo se explicará como se envían y reciben las señales y cuales son.

Para ello, el algoritmo de Comunicación y Control será explicado considerando los tres casos mencionados anteriormente.

Caso 1.- Llevar al manipulador de una posición a otra habiendo seleccionado sólo un punto. (Opciones Manual o Def. Punto del menú Campo de Trabajo ver fig. 2.4)

Antes de continuar es necesario aclarar un aspecto muy importante, el manipulador en cuestión se encuentra muy limitado mecánicamente; además, el movimiento de algunas de sus articulaciones depende directamente de otras, ocasionando que se deba tener cuidado con el orden en el cual se mueve cada una de ellas.

El algoritmo realizado en MatLab genera varios bloques de datos; en uno se encuentran cinco datos que indican la cantidad de movimiento por cada articulación, y en otro se almacena la secuencia de los movimientos que se hayan seleccionado para poder repetirlos utilizando la opción Tarea (ver Capítulo 2).

El programa *PRUCAM.C*, es el encargado de manipular cada uno de esos bloques de datos generando a su vez *datos finales*, los que son utilizados por la interfaz electrónica permitiendo el movimiento del manipulador. (ver figura 3.1).

El programa conoce la transformación necesaria para obtener todos los datos necesarios para habilitar cada una de las articulaciones del manipulador.

La secuencia de envío de datos para cada articulación será la siguiente:

Envía el sentido, cantidad de movimiento y el valor de la velocidad. Esta secuencia hará que el manipulador responda con el movimiento de las articulaciones especificadas, hasta haber alcanzado el punto que se le haya indicado. En caso de ser necesario, el programa corregirá el movimiento de la articulación a la cual le faltó o sobró movimiento, utilizando para ello el circuito AM9513 y las funciones incluidas en el programa para controlar la posición.

Caso 2.- Llevar al manipulador de cualquier punto a su posición de referencia.

Para llevar al manipulador a su posición de referencia, las articulaciones son movidas en una secuencia especial para evitar el daño físico de alguna de ellas.

Para entender los algoritmos que serán explicados a continuación, es necesario conocer el sentido de giro de cada articulación, éstos son mostrados en la figura 3.2.

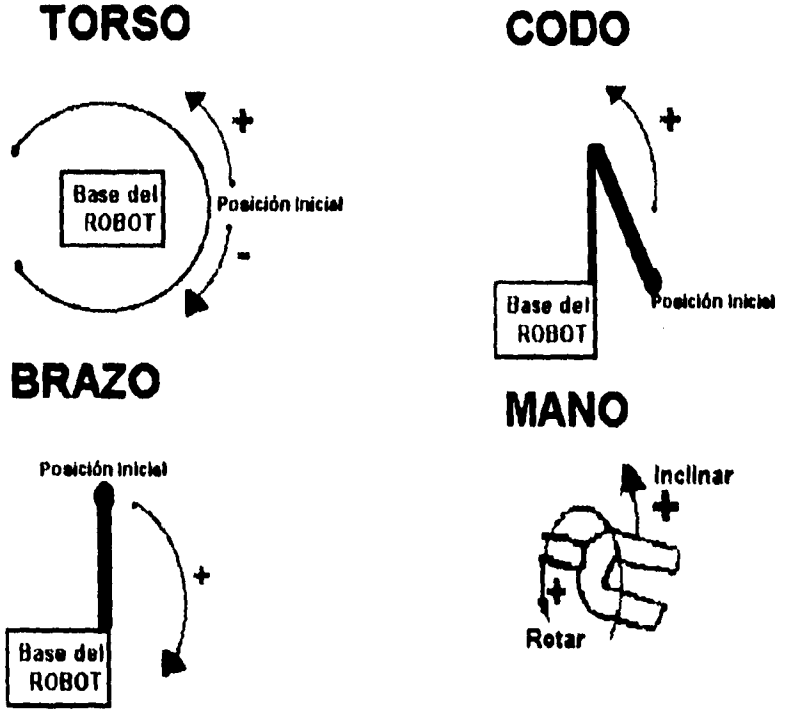


Figura 3.2.Sentidos para cada articulación.

Considerando que el torso puede estar a la izquierda o a la derecha, se necesita saber primero hacia a donde se debe girar, para ello se cuenta con un switch precisamente en donde se consideró 0º. El SWT (switch del torso ver figura 3.3), se encuentra presionado cuando el manipulador se encuentra a la derecha y deja de presionarse cuando está a la izquierda.

Considerando lo anterior, el primer paso del algoritmo es leer el valor del SWT para obtener el sentido en el cual deberá girar.

El programa envía el sentido, una velocidad promedio, y está continuamente registrando el sensor hasta que éste cambie su valor original; en ese momento, el torso ya habrá llegado a su posición de referencia.

El Brazo y el Codo funcionan de manera similar, ambos cuentan con un switch, SWB (switch del brazo) y SWC (switch del codo). Para estas dos articulaciones no es necesario saber si el sentido es hacia arriba o hacia abajo; por ejemplo para el Brazo, si éste se ha movido, necesariamente tuvo que haber sido hacia abajo; es decir, debe tener un ángulo menor a su posición de referencia es decir menor a 90º.

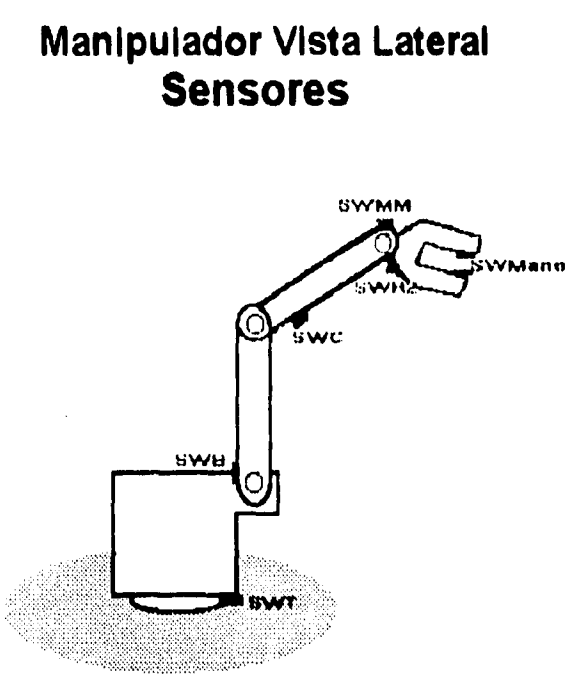


Figura 3.3.Localización de los sensores del Manipulador.

El algoritmo leerá el valor del SWB, si éste está presionado no se hace nada, en caso contrario se mandará el sentido negativo; es decir, hacia arriba hasta que el SWB sea presionado.

En cuanto al Codo, el algoritmo hace algo similar a lo que se hizo con el Brazo. De tal manera que tanto el Brazo como el Codo queden en su posición de referencia.

Los dos grados de libertad que tiene el manipulador en su muñeca son el de Rotación y el de Inclinación. Su posición de referencia está dada por un switch (SWMM) que se encuentra en la parte superior de la muñeca y por el SWR2 localizado en la muñeca.

El algoritmo lee el valor del SWMM, si éste se encuentra presionado, indicará que no es necesaria una inclinación hacia arriba; a continuación leerá el valor del SWR2, en caso de no estar presionado, se deberán mandar los datos para rotar hacia la izquierda a la muñeca hasta que el SWR2 sea presionado.

Ahora, sólo resta cerrar la mano, en caso de encontrarse abierta; para ello se cuenta con el SWMano, si éste se encuentra presionado, indica que la mano está totalmente cerrada; en caso contrario, se enviarán los datos necesarios para activar el motor que controla la mano y esperar a que el SWMano se presione.

Considerando la dependencia mecánica entre brazo-codo, codo-mano y los algoritmos brevemente explicados arriba, el programa *CASITA.C* logra que el manipulador llegue a su posición de referencia (ver código al final del capítulo). Cabe resaltar que la secuencia del programa casita se obtuvo de la observación minuciosa del comportamiento del manipulador.

Caso 3.- Llevar al manipulador a cualquier posición utilizando su Control Manual.

Cada uno de los controles del Control Manual que aparecen en la pantalla de la computadora (ver figura 2.1) cuentan con un programa en Matlab, que obtiene la velocidad marcada en la barra de desplazamiento, la dirección de articulación que se va a mover y el sentido. Estos datos son almacenados en un dicque para después ser manipulados por el programa *TODOJUNTA.C* (en caso de ser el torso, brazo o todo el que se desee mover) o *MUNECA.C* (para la mano, muñeca e inclinación) cuyo código se encuentra al final del capítulo.

El programa correspondiente hace las transformaciones necesarias y envía los datos resultantes a la interfaz electrónica. En caso de existir algún error por ejemplo, haber presionado el control de subir el brazo y este se encuentra en su límite, la interfaz electrónica se encargará de desarrollar esa movimiento avisando al programa *TODOJUNTA.C* y este al usuario, del error ocurrido, inmediatamente se regresa el control al programa *PROGRAMA.M* el cual se encargará de pedir otros datos e iniciar el proceso nuevamente.

En cuanto al Codo, el algoritmo hace algo similar a lo que se hizo con el Brazo. De tal manera que tanto el Brazo como el Codo queden en su posición de referencia.

Los dos grados de libertad que tiene el manipulador en su muñeca son el de Rotación y el de Inclinación. Su posición de referencia está dada por un switch (SWMM) que se encuentra en la parte superior de la muñeca y por el SWR2 localizado en la muñeca.

El algoritmo lee el valor del SWMM, si éste se encuentra presionado, indicará que no es necesaria una inclinación hacia arriba; a continuación leerá el valor del SWR2, en caso de no estar presionado, se deberán mandar los datos para rotar hacia la izquierda a la muñeca hasta que el SWR2 sea presionado.

Ahora, sólo resta cerrar la mano, en caso de encontrarse abierta; para ello se cuenta con el SWMano, si éste se encuentra presionado, indica que la mano está totalmente cerrada; en caso contrario, se enviarán los datos necesarios para activar el motor que controla la mano y esperar a que el SWMano se presione.

Considerando la dependencia mecánica entre brazo-codo, codo-mano y los algoritmos brevemente explicados arriba, el programa *CASITA.C* logra que el manipulador llegue a su posición de referencia (ver código al final del capítulo). Cabe resaltar que la secuencia del programa casita se obtuvo de la observación minuciosa del comportamiento del manipulador.

Caso 3.- Llevar al manipulador a cualquier posición utilizando su Control Manual.

Cada uno de los controles del Control Manual que aparecen en la pantalla de la computadora (ver figura 2.1) cuentan con un programa en MatLab que obtiene la velocidad marcada en la barra de desplazamiento de velocidad, la articulación que se va a mover y el sentido. Estos datos son almacenados en un bloque para después ser manipulados por el programa *TOCOBRA.C* (en caso de ser el torso, brazo, o codo el que se debe mover) o *MUNEROT.C* (para la mano rotación e inclinación) cuyo código se encuentra al final del capítulo.

El programa correspondiente hace las transformaciones necesarias y envía los datos resultantes a la interfaz electrónica. En caso de existir algún error, por ejemplo, haber presionado el control de subir el Brazo y éste se encuentre ya en su límite, la interfaz electrónica se encargará de deshabilitar ese movimiento avisando al programa Tocobra.C, y éste al usuario, del error ocurrido. Inmediatamente se regresa el control al programa Ftocobra.M el cual se encargará de pedir otros datos e iniciar el proceso nuevamente.

Desarrollo e Implementación

Para probar que los algoritmos funcionaran correctamente se realizaron programas independientes para cada articulación, los cuales fueron muy importantes para poder desarrollar el programa que habilita a las cinco articulaciones a la vez.

La implementación de los programas individuales, en seudocódigo es la siguiente:

1.- Enviar al puerto la cantidad de movimiento
2.- Enviar el sentido
3.- Enviar la velocidad
4.- Sensar el movimiento
5.- Comparar el movimiento registrado con el movimiento final
6.- Repetir las veces que se desee.

Lo importante es revisar el movimiento del manipulador, primero articulación por articulación y después las cinco a la vez.

Una vez realizadas las pruebas necesarias a cada uno de los algoritmos explicados anteriormente, y de verificar el buen funcionamiento de cada articulación por separado, se desarrollaron los programas principales cuya implementación, se presenta a continuación.

Programas y librerías utilizadas.

```
//Libreria que inicializa la tarjeta PCTIO para poder controlar a los
//contadores que se requieren.

#if !defined _LIBDEF_H
#define _LIBDEF_H

//Valores de registros y comandos para utilizar la tarjeta PCTIO
//Direccion Base para la Tarjeta PCTIO
#include <dos.h>
#include <stdio.h>
#include <conio.h>
#define         PCTIO_BASE              0x01A0
////////////////////// Am9513A
// Registros Internos para el Am9513A
#define  MASTER_MODE                    0x17        // Apuntadores al registro de modo
#define  C2_MODE                        0x02        // Registro de Modo del Contador #2
#define  C4_MODE                        0x04        // Registro de Modo del Contador #4
// Apuntadores a los registros de Carga
#define  C2_LOAD                        0x0A        //Registro de Carga para Contador #2
#define  C4_LOAD                        0x0C        //Registro de Carga para Contador #4
```

```
                                                    // Apuntadores al Registro de
Mantener Dato
#define C2_HOLD                 0x12                // Registro Mantener del Contador #2
#define C4_HOLD                 0x14                // Registro Mantener del Contador #4
// Comandos
#define RESET_9513          0xFF                    // Reset Maestro
#define DISARM_C2           0xC2                    // Deshabilita el Contador #2
#define DISARM_C4           0xC8                    // Deshabilita el Contador #4
#define LOAD_ARM_C2             0x62                    // Carga y Arma al  #2
#define LOAD_ARM_C4             0x68                    // Carga y Arma al #4
#define LOAD_ARM_ALL        0x6F            // Carga y Arma todos los cont. en uso
#define SAVE_C2                 0xa2                    // Salva el contenido del Contador #2
#define SAVE_C4                 0xa8                    // Salva el contenido del Contador #4
#define DISARM_SAVE_C2          0x82            // Deshabilita y salva al Contador #2
#define DISARM_SAVE_C4          0x88            // Deshabilita y salva al Contador #4
// Existen dos  Amd9513A en  la tarjeta PCTIO-10
#define         cmd_port                0x0001
#define         data_port       0x0000
#define         stc_a                   0x0000
#define         STCA_COMMAND_PORT       (PCTIO_BASE | stc_a     | cmd_port)
#define         STCA_DATA_PORT          (PCTIO_BASE | stc_a     | data_port)
// STCA
#define         COMMAND_PORT    STCA_COMMAND_PORT
#define         DATA_PORT       STCA_DATA_PORT
//*************************************************************
//STCB
#define         stc_b                   0x0002
#define         STCb_COMMAND_PORT       (PCTIO_BASE | stc_b     | cmd_port)
#define         STCb_DATA_PORT          (PCTIO_BASE | stc_b     | data_port)
#define         COMMAND_PORTb   STCb_COMMAND_PORT
#define         DATA_PORTb        STCb_DATA_PORT
 Funciones
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//Funcion inipctio  inicializa al master mode
inipctio(void)
{
//Configura el master mode,binario y un bus de datos
outp(COMMAND_PORT,MASTER_MODE);
outp(DATA_PORT,0X00);  //bit bajo
outp(DATA_PORT,0X51);  //bit alto
return(0);
}

//++++++++++++++++++++++++++++++++++
//Funcion que prepara al contador 2
prepcont2(void)
{
//Configurando contadores
//registro de modo del contador dos CONTAR BINARIO HACIA ARRIBA FUENTE scr2
outp(COMMAND_PORT,C2_MODE);
outp(DATA_PORT,0X08);    //menos sig.
outp(DATA_PORT,0x02);    //mas significativo
```

```
outp(COMMAND_PORT,C2_LOAD);   //carga al contador dos
outp(DATA_PORT,0x00);
outp(DATA_PORT,0x00);
outp(COMMAND_PORT,LOAD_ARM_C2);  //Carga y prepara el contador dos

return(0);
}

//++++++++++++++++++++++++++++++++++++
//Prepara al contador 4
prepcont4(void)
{
//Configurando contadores
//registro de modo del contador cuatro CONTAR BINARIO HACIA ARRIBA FUENTE scr2
outp(COMMAND_PORT,C4_MODE);
outp(DATA_PORT,0X08);    //menos sig.
outp(DATA_PORT,0x04);    //mas significativo

outp(COMMAND_PORT,C4_LOAD);   //carga al contador cuatro
outp(DATA_PORT,0x00);
outp(DATA_PORT,0x00);

outp(COMMAND_PORT,LOAD_ARM_C4);  //Carga y arma el contador cuatro
return(0);
}
//++++++++++++++++++++++++++++++++++++
//Prepara al contador 7
prepcont7(void)
{
//Configurando contadores
//registro de modo del contador cuatro CONTAR BINARIO HACIA ARRIBA FUENTE scr2
outp(COMMAND_PORTb,C2_MODE);
outp(DATA_PORTb,0X08);    //menos sig.
outp(DATA_PORTb,0x02);    //mas significativo

outp(COMMAND_PORTb,C2_LOAD);   //carga al contador cinco
outp(DATA_PORTb,0x00);
outp(DATA_PORTb,0x00);

outp(COMMAND_PORTb,LOAD_ARM_C2);  //Carga y arma el contador cinco
return(0);
}
//++++++++++++++++++++++++++++++++++++
//Prepara al contador 9
prepcont9(void)
{
//Configurando contadores
//registro de modo del contador cuatro CONTAR BINARIO HACIA ARRIBA FUENTE scr2
outp(COMMAND_PORTb,C4_MODE);
outp(DATA_PORTb,0X08);        //menos sig.
outp(DATA_PORTb,0x04);        //mas significativo

outp(COMMAND_PORTb,C4_LOAD);   //carga al contador cinco
outp(DATA_PORTb,0x00);
```

```
outp(DATA_PORTb,0x00);

outp(COMMAND_PORTb,LOAD_ARM_C4);  //Carga y arma el contador cinco
return(0);
}

//+++++++++++++++++++++++++++++++++++++++++
//cierra registros del contador 2
registros2(void)
{outp(COMMAND_PORT,DISARM_C2);      //desarma o deshabilita al contador 2
return(0);}
//+++++++++++++++++++++++++++++++++++++++++
//Cierra registros del contador 4
registros4(void)
{outp(COMMAND_PORT,DISARM_C4);      //desarma o deshabilita al contador4
return(0);
}
//+++++++++++++++++++++++++++++++++++++++++
//Cierra registros del contador 7
registros7(void)
{outp(COMMAND_PORTb,DISARM_C2);      //desarma o deshabilita al contador 7
return(0);
}
//+++++++++++++++++++++++++++++++++
//cierra registros del contador 9
registros9(void)
{outp(COMMAND_PORTb,DISARM_C4);      //desarma o deshabilita al contador 9
return(0);
}

//+++++++++++++++++++++++++++++++++
//las siguientes funciones
//leeran los contadores  y
//regresan  la cantidad de movimiento
//de las diferentes articulaciones.
int leecontorso(void)
{
int aux1,aux2,lectura2;
        lectura2 =0;
        outp(COMMAND_PORT,SAVE_C2);                    // salve al contador 2
        outp(COMMAND_PORT,C2_HOLD);
        //menos sig
        aux1 = (unsigned char)inp(DATA_PORT);     //leyendo del puerto el registro hold
        //mas sig
        aux2  = (unsigned char)inp(DATA_PORT);     //leyendo del puerto el registro hold
        lectura2 =(int)((256*aux2)+aux1);
return(lectura2);
}
//***************FUNCIONES UTILIZADAS**********
//************* funcion de error
//*******************************
errores(char dato[10])
{
int gdriver = DETECT, gmode, errorcode,a,b;
```

```
struct textsettingstype textinfo;
initgraph(&gdriver, &gmode, "c:\\bc4\\bgi");
        errorcode = graphresult();
        if (errorcode != grOk)
        {printf("Graphics error: %s\n", grapherrormsg(errorcode));
                printf("Presiona una tecla para salir:");
                getch();
                exit(1);
        }
        setcolor(1);
        bar(0,0,getmaxx(),getmaxy());
        a = getmaxx()/4;
        b = getmaxy()/4;
        setcolor(3);
        setfillstyle(1,9);
        bar(a,b,getmaxx()-a,getmaxy()-b);
        setlinestyle(0,1,3);
        line(a,b,getmaxx()-a,b);
        line(a,b,a,getmaxy()-b);
        line(a,getmaxy()-b,getmaxx()-a,getmaxy()-b);
        line(getmaxx()-a,b,getmaxx()-a,getmaxy()-b);
        settextstyle(1,0,6);
        outtextxy(a+70,b+20,"ERROR ");
        sound(1227);
        outtextxy(a+70,b+85," !!! ");
        settextstyle(1,0,5);
        outtextxy(a+25,b+145, dato);
        gettextsettings(&textinfo);
        delay(1500);
        nosound();
closegraph();
return 0;
}
vueltas(float factor, float valorn,int *primdat,int *segdat);
int ciclotra(float vel);
//*****************************************************************
//******************  FUNCIONES UTILIZADAS *****************
//velocidad  maxima 91 grados por  minuto
//********obtiene ciclo de trabajo y
//********lo manda al puerto
//****************************************
ciclotra(float vel)
{
float   constante;
int dato;
        constante = (255/91.0);  //vel max 91 grados por min.  regla de tres
        dato = ceil(vel*constante);  // 255 corresponde a pwm = ff
return(dato);
}

//**********Funcion para obtencion de vueltas
//*********************************************
vueltas(float factor,float valorn,int *primdat,int *segdat)
{
```

```
int vuel;                       //Dado el factor de conversión se obtiene
int dat1, dat2;                 //la cantidad de vueltas por grado
        vuel= abs(floor(valorn*factor));
        dat1 = floor(vuel/256);
        *primdat = dat1;
        if (dat1 != 0)
                dat2 = floor(vuel -(dat1*256));
        else
          dat2 = vuel;
        *segdat = dat2;
return(0);
}
Funciones para su posición de referencia.
//*************************************
//**************** FUNCION TORSO
//*********************************
ftorso(void)
{
int entrada,swt1,swt2,error,conta;  //printf("\n es torso a home");
        error = 1;          conta = 1;
        entrada = inportb(0x301);
        swt1 = entrada & 1;         //printf("el valor de swt1 %x\n",swt1);
        if (swt1 == 1)              //esta presionado entonces el sent es neg
         mandasenti(0);
        else                                    //  el sentido es positivo
         mandasenti(1);                         //habilita el sentido pos del tor
        delay(300);
        words(torsovu,vmax,vmax);
        pwm(torsopw,50);                        //habilita el pwm del torso
        entrada = inportb(0x301);               //lee nuevamente al puerto
        swt2 = entrada & 1;                             //para esperar a que cambie.

        while ((swt2 == swt1) && (conta != reloj))
        {  delay(30);
                entrada = inportb(0x301);
                swt2 = entrada & 1;
                conta++;                                        //incrementa contador
        }
        pwm(torsopw,0);             //FDeshabilita el motor
        mandareset();
        mandareset();
        if (conta != reloj)          //printf("llego a home el torso \n");
          error = 0;
        else
          error = 1;  //printf("hubo error y va de nuevo en el torso\n");

        mandareset();
return(error);
}

//*************BRAZO
//*******************
fbrazo(void)
{
```

```
int entrada,swb,swc,conta,error,swmm;
{     ,                    //printf("\n es el brazo a home ");
conta = 1;  error = 0;
entrada = inportb(0x301);
swb = entrada & 2;
if (swb == 0)                  //no esta presionado entonces el sent es pos
{
 mandasenti(2);               //habilita el sentido pos del brazo
 words(brazovu,vmax,vmax);       //habilita el menos y mas sig del brazo
 pwm(brazopw,pwm50);
 entrada = inportb(0x301);     //  lee nuevamente al puerto
 swb = entrada & 2;            //   para esperar a que cambie.
 while ((swb == 0) && (conta != reloj))
 {    entrada = inportb(0x301);
          swb = entrada & 2;
          swc = entrada & 4;
          swmm = entrada & 16;

          if (swc == 4)
          {//printf("\n se oprimio el del codo por lo tanto manda el máximo negativo\n");
                        mandareset();
                        mandasenti(0);               //sentido codo contraria a home
                        words(codovu,vmax,vmax);     //numero de vueltas
                        pwm(codopw,pwm50);
                        delay(2000);   //dando tiempo a que llegue al otro lado
                        //printf("\n se encendera nuevamente el brazo\n");
                        mandasenti(2);               // habilita el sentido pos del brazo
                        words(brazovu,vmax,vmax);    //vueltas brazo
                        pwm(brazopw,pwm50);
          }
          if (swmm == 16)
          {  //printf("\n se oprimio el de inclinar por lo tanto manda el máximo negativo\n");
                        mandareset();
                        mandasenti(0);    //sentido inclinacion contraria a home
                        words(roteincvu,vmax,vmax);
                        pwm(roteincvu,pwm50);
                        delay(2000);    //dando tiempo a que llegue al otro lado
                        mandareset();
                        //printf("\n se encendera nuevamente el brazo\n");
                        mandasenti(2);
                        words(brazovu,vmax,vmax);
                        pwm(brazopw,pwm50);
          }
                pwm(brazopw,pwm50);
                pwm(brazopw,pwm50);
          conta++;
 }
 mandareset();
 if (conta != reloj)
 { error = 0;       //printf("llego a home el brazo\n");
 }
 else
 {// printf("no puede llegar  el brazo\n");
          error =1;
```

```
 }
}
else   //esta presionado
 error = 0;
mandareset();
}
return(error);
}

//************************ Funcion Codo
//*************************************
fcodo(void)
{
int swc,swmm,entrada,error,conta;
{//printf("\n es el codo a home");
entrada = inportb(0x301);
swc = entrada & 4;
conta = 1;
error = 0;
if (swc == 0)                          // no esta presionado entonces el sent es pos
{       mandasenti(4);                          // habilita el sentido pos del codo
        words(codovu,vmax,vmax);
        pwm(codopw,pwm50);
        entrada = inportb(0x301);    //  lee nuevamente al puerto
        swc = entrada & 4;           //  para esperar a que cambie.

        while ((swc == 0) && (conta != reloj))
        {       delay(100);
                entrada = inportb(0x301);
                swc = entrada & 4;
                swmm = entrada & 16;
 //printf("Esto es el Codo,entrada %x   contador %d \n",entrada,conta);
                if (swmm == 16)
                {       mandareset();  //printf("\n manda sentido contrario a inclinar\n");
                        mandasenti(0);  //sentido inclinacion contraria a home
                        words(roteincvu,vmax,vmax);
                        pwm(roteincpw,pwm50);
                        delay(2000);  //dando tiempo a que llegue al otro lado
                        mandareset();//printf("\n encendera nuevamente al codo\n");
                        mandasenti(4);      // habilita el sentido pos del codo
                        words(codovu,vmax,vmax);
                        pwm(codopw,pwm50);
                }
                if (fmod(conta,10) == 0)
                {//printf("se habilitara nuevamente el pwm del codo %d\n",conta);
                        pwm(codopw,pwm50);
                }
                conta++;
        }
        mandareset();
        if (conta == reloj)
        { error = 1; //printf("hubo error en el codo\n");
        }
        else
```

```
            { error = 0; //printf("llego codo a home\n");
            }
}
else
        error = 0;
mandareset();
}
return(error);
}
//*****************  Funcion Rotar
//*********************************
frotar(void)
{
int entrada,foto,conta, error;
{//printf("\n es rotar a home");
error = 0;
conta = 1;
entrada = inportb(0x301);
foto = entrada & 8;

if (foto  == 8)                         // alineados
        error = 0;
else       // se necesita rotar y solo se permite hacia la der.
{
 mandasenti(8);  //es para habilitar el sensor swr2 que es el de home
 words(roteincvu,vmax,vmax);
 pwm(roteincpw,pwm50);

 entrada = inportb(0x301);    //  lee nuevamente al puerto
 foto = entrada & 8;          //  para esperar a que cambie.
 while ((foto == 0) && (conta != reloj))
 { entrada = inportb(0x301);      //se repite hasta que se alinie
          foto = entrada & 8;
          if (fmod(conta,100) == 0)
          {// printf("se habilitara nuevamente el pwm del rotar %d\n",conta)
                pwm(roteincpw,pwm50);
          }
          conta++;
 }
 mandareset();

 if (conta !=reloj)
 { error = 0;     // printf("llego rotar a home\n");
 }
 else
 { error = 1;     // printf("hubo error en rotar\n");
 }
}
}
return(error);
}
//****************** Funcion Inclinar
//************************************
finclinar(int cuanto)
```

```
{
int swmm, entrada,error, cont;
//printf("\n es inclinar a home");
entrada = inport(0x301);
swmm = entrada & 16;
error = 0;
cont =1;

if (swmm == 16)                    // esta presionado
        error = 0;
else
{       mandasenti(24);            // inclinacion positiva
        words(roteincvu,vmax,vmax);
        pwm(roteincpw,pwm50);
        entrada = inportb(0x301);      // lee nuevamente al puerto
        swmm = entrada & 16;           // para esperar a que cambie.
        while ((swmm == 0) && (cont != cuanto))
        {  entrada = inportb(0x301);      //se repite hasta que se suba
           swmm = entrada & 16;
           if (fmod(cont,100) == 0)
           {//printf("se habilitara nuevamente el pwm de inclinar%d\n",cont
                pwm(roteincpw,pwm50);
              }
              cont++;
        }
        mandareset();
        if (cont == cuanto)
        {  error = 1;//printf("error en  inclinar\n");
        }
        else
        {  error = 0;//printf("llego  a home inclinar\n");
        }
}
return(error);
}

//******************Funcion  Mano *****
//****************************************
fmano(void)
{
int swcerrar,entrada,error,conta;
{
conta = 1;
error = 0;
entrada = inportb(0x301);
swcerrar = entrada &  32;

if (swcerrar == 32)        // esta presionado
        error =0;
else
{          outportb(0x300,0x60);
           outportb(0x302,0x14);  // manda pwm mano
           outportb(0x302,0x1e);
```

```
        entrada = inportb(0x301);    //  lee nuevamente al puerto
        swcerrar = entrada & 32;        //  para esperar a que cambie.

        while ((swcerrar == 0) && (conta != reloj))
        {  delay(100);
                entrada = inportb(0x301);   //se repite hasta que se cierre
                swcerrar = entrada & 32;
 printf("Cerrar la mano, entrada  %x contador %d \n",entrada,conta);
                conta++;
        }
        mandareset();
        if (conta == reloj)
                error = 1;
        else
                error =0;
}
}
return(error);
}
//*****************************************************************************
//*****************************************************************************
//*****************************************************************************
//Funciones que mandan los datos al puerto 8255
#define  inter  110          //retardo entre dato y dato
#define  reloj  5000     //espera para seguir el algoritmo
#define torsopw  16
#define brazopw 18
#define codopw 17
#define roteincpw  19
#define torsovu  23
#define brazovu 25
#define codovu 24
#define roteincvu 26
#define pwm50  89
#define vmax  239
#define factor  3.5
#define facod  13.43
#define facbra  12.5
#define facrot  3.1
#define facinc  3.0
#define limcont 2

Funciones
//**********************************
//**********************************

 //*****Funcion para determinar los sentidos
//*****************************************
sentidos(float val1,float val2,float val3,float val4,float val5)
{
int p1,p2,p3,p4,p5,sentido;          //El sentido se encuentra
        p1 =0;   p2=0;    p3=0;    p4=0; p5=0;    //en un solo registro
        if (val1 > 0.0)
         p1 = 1;
```

```
        if (val2 > 0.0)  //teta2 brazo
          p2 = 2;
        if (val3 > 0.0)  //teta3 codo
          p3 = 4;
        if ((val4 == 0.0 ) && (val5 > 0.0)) //rotacion positiva derecha
        {                           //sin inclinacion
                p4 =0;
                p5 = 16;
        }
        if ((val4 == 0.0) && (val5 < 0.0)) //rotacion negativa
        {  p4 = 8;
                p5 = 0;
        }
        if ((val4 != 0.0) &&  (val4 > 0.0))
        {  p4 = 8;                  //inclinacion  positiva
                p5 = 16;
        }
        if (( val4 != 0.0) && (val4 < 0.0)) //inclinacion negativa
        {  p4 = 0;
                p5 = 0;
        }
        sentido = p1+p2+p3+p4+p5;
return(sentido);
}

//Funcion PWM*************
//************************
pwm(int articul,int cantidad)
{ /*si es torso arti = 16 decimal 10h teta1
          si es brazo arti = 18 decimal 12h teta2
          si es codo arti = 17 decimal 11h teta3
          si es rotar o inclinar arti = 19d 13h teta4
          cantidad es el PWM*/

 outportb(0x300,cantidad);  delay(inter);
 outportb(0x302,articul); delay(inter);
}

//********Funcion words
//**********************
words(int articulacion,int masig,int menosig)
{
/* articulacion para torso 23d 17h teta1
          articulacion para brazo 25d 19h teta2
          articulacion para codo 24d 18h  teta3
          articulacion para rotar e inc  26d 1ah teta4
          massig, menosig numero de vueltas mas y menos significativas
*/
 outportb(0x300,menosig); delay(inter);
 outportb(0x302,articulacion); delay(inter);
 outportb(0x300,masig); delay(inter);
 outportb(0x302,articulacion); delay(inter);
 outportb(0x302,0x1e); delay(inter);
}
```

```
//*********Funcion mandasentidos
//*******************************
mandasenti(int sentid)
{
 outportb(0x300,sentid); delay(inter);
 outportb(0x302,0x15); delay(inter);
 outportb(0x302,0x1e); delay(inter);
}

//*********Funcion MandaReset
//***************************
mandareset(void)
{ outportb(0x300,0x00);delay(inter);
 outportb(0x302,0x1e); delay(inter);
 outportb(0x302,0x0e); delay(inter);
 outportb(0x302,0x1e); delay(inter);
}

//*********Funcion Configura
//***************************
configura(void)
{
 outportb(0x303,0x82);delay(inter);
 mandareset();
return(0);
}
//Funcion que verificara que llegue el torso al lugar indicado
//utilizando el contador dos
//Regresa la diferencia entre el dato de entrada difer y las vueltas dadas
//los datos de entrada son el datot para obtener el sentido
//y difer que es la diferencia del error obtenido con anterioridad
int llegotorso(float datot,int difer)
{
int vnewmas,vnewmenos;
int cont,conta,dadas1,ant1,sent;
int diferen;

cont =0;ant1=0;conta=0;dadas1 =0;vnewmas=0;vnewmenos=0;
sent = sentidos(datot,0.0,0.0,0.0,0.0,0.0);
vueltas(1.0,(float)difer,&vnewmas,&vnewmenos);
mandasenti(sent);
words(torsovu,vnewmas,vnewmenos);
prepcont2();
while ((dadas1 < abs(difer))  &&(conta <3))
{
        pwmes(torsopw,85);
        dadas1 = leecontorso();
        if (fmod(cont,490) == 0)
                ant1 =leecontorso();
        if ((fmod(cont,620) == 0) &&(ant1 == dadas1))
                conta++;
        cont++;
```

```
}
mandareset();
registros2();
diferen= abs(difer)-dadas1;
return(diferen);
}

//*********************************************************************
//*********************************************************************
//*********************************************************************
// Función encargada de ordenar, calcular y habilitar las cinco
// articulaciones al mismo tiempo.
//funcion que repite el proceso
//para mover al manipulador continuamente
//***************************************
mandacinco(float dato1,float dato2,float dato3,float dato4,float dato5)
{  int va1,vb1,va2,vb2,va3,vb3,va4,vb4,va5,vb5,sent;
        int ct1,ct2,ct3,ct4,ct5;
        int dadas1,dadas2,dadas3,dadas4,total1,total2,total3
                ,total4,total1r,total2r,total3r,total4r;
        int ant1,ant2,ant3,ant4,cont;
        int banerro,diferencia,conta,datoaux;
        int errortorso,errorbrazo,errorcodo,errorinch;
        int dadas5,total5r;
banerro= 0;
                                //Termina obtencion de datos0
dato1 = dato1 * (-1.0);                 //por la correccion de sentido

//*********** inicia obtencion de valores para el manipulador**************
//*****************************************************************************
vueltas(factor,dato1,&va1,&vb1);  // mas signif va1
vueltas(facbra,dato2,&va2,&vb2);  //originalmente era 12
vueltas(facod,dato3,&va3,&vb3);
vueltas(facinc,dato4,&va4,&vb4);
sent = sentidos(dato1,dato2,dato3,dato4,dato5);  //obtiene sentidos
//obtiene ciclos de trabajo
//si existen datos existe pwm
ct1 = 0;ct2 =0;ct3=0;ct4=0;ct5=0;
if (dato1 != 0.0)
 ct1= 99;
if (dato2 != 0.0)
 ct2= pwm50;
if (dato3 != 0.0)
 ct3= pwm50;
if (dato4 != 0.0)
 ct4= 50;
if (dato5 != 0.0)
 ct5= 50;
mandareset();
mandasenti(sent);       //manda sentidos
words(torsovu,va1,vb1);  //manda la cantidad de movimiento
words(brazovu,va2,vb2);
words(codovu,va3,vb3);
words(roteincvu,va4,vb4); //para inclinacion
```

```
prepcont2();          //prepara a los contadores antes de
prepcont4();          //empezar el movimiento
prepcont7();
prepcont9();
//SOLO SE HA CONSIDERADO HASTA LA INCLINACION OJOOOOOOOO
//*****************************************************
total1r =0;total2r =0;total3r =0;total4r =0;
if (dato1 != 0.0)
total1r = (256*va1)+vb1;
if (dato2 != 0.0)
total2r = (256*va2)+vb2;
if (dato3 != 0.0)
total3r = (256*va3)+vb3;
if (dato4 != 0.0)
total4r = (256*va4)+vb4;
pwm(torsopw,ct1);                    //Se habilitan los cinco motores
pwm(codopw,ct3);
pwm(brazopw,ct2);
pwm(roteincpw,ct4);   //inclinacion
//Se mandaron todos excepto la rotacion
dadas1=0;dadas2=0;dadas3=0;dadas4=0;
ant4 =1;ant2=1;ant3=1;ant1=1;
total1=0;total2=0;total3=0;total4=0;
//Lo siguiente son las tolerancia s de error permitidas.
//Es decir, se permite  cierto porcentaje de error
//dentro del movimiento.
if (total1r > 0)
  total1 = abs(total1r-((int)(0.02*total1r)));
if (total2r > 0)
  total2 = abs(total2r-((int)(0.025*total2r)));
if (total3r > 0)
  total3 = abs(total3r-((int)(0.019*total3r)));
if (total4r > 0)
  total4 = abs(total4r-((int)(0.02*total4r)));
cont=0;
// si durante tres ciclos las cuentas  son iguales indicara que el
//manipulador se reusa a realizar el movimiento; quiza porque se
//encuentra en el limite o  su posición no lo permite.
while(cont< 3)
{       ant3= leecontcodo();
        if (ant3 > total3)
                pwm(codopw,0);
                                //se registraran las anteriores
        //-------------torso
        dadas1 = leecontorso();
        if ((dadas1 < total1)&&(ant1 < total1))
                pwm(torsopw,99);
        else
          pwm(torsopw,0);
        //----------------------

        ant4 =leecontrot();
        if (ant4 > total4)
                pwm(roteincpw,0);
```

```
        //----------brazo
        dadas2=leecontbrazo();
        if ((dadas2 < total2)&&(ant2 < total2))
          pwm(brazopw,75);
        else
          pwm(brazopw,0);
        //----------

        //----------codo
        dadas3 = leecontcodo();
        if ((dadas3 < total3)&&(ant3 < total3))
                pwm(codopw,76);      //4c
        else
                pwm(codopw,0);
        //----------

        if (((dadas4 == ant4)&&(dadas3 == ant3))&&((dadas2 == ant2)&&(dadas1 == ant1)))
          cont++;
        ant1= leecontorso();

        ant2= leecontbrazo();
        if (ant2 > total2)
                pwm(brazopw,0);
        //---------rotar
        dadas4 = leecontrot();
        if ((dadas4 <total4)&&(ant4 < total4))
          pwm(roteincpw,64);  //pwm del 40%
        else
          pwm(roteincpw,0);
        //----------
}                                       //termina ciclo
mandareset();
dadas1 = leecontorso();
dadas2 = leecontbrazo();
dadas3 = leecontcodo();
dadas4 = leecontrot();
//torso
registros2();
//brazo
registros7();
//codo
registros4();
//rotar
registros9();
//programa para ir a la posición de referencia.
//printf("\n Va inclinacion para permitir movimiento\n");
entri =finclinar(poq);
mandareset();
//printf(" Va rotar en sentido negativo para permitir movimiento");
mandasenti(16);
words(roteincvu,0,90);
pwm(roteincpw,pwm50);
delay(150);                     //esperando el movimiento
```

```
//Considerando que no esta en home
//+++++++++++++++ TORSO    ++++++++++++++++++++++++++++++
//printf("\n va el torso a home \n");
entrt = ftorso();            //********** torso *****
if (entrt!= 1)
{       //printf("llego torso a home\n");
        mandareset();
        //++++++++++++ BRAZO  ++++++++++++++++++++++++++++++++++++
        //printf("\nVa el brazo a home\n");
        entrb = fbrazo();          //verifica al codo y al inclinar
        if (entrb != 1)
        {       //printf("llego el brazo a home\n");
                //+++++++++++++++ CODO ++++++++++++++++++++++++++++++++
                //printf("\n Va el codo a home \n");
                entrc = fcodo();           //******* codo ******
                if (entrc != 1)            //sensa a inclinar
                {
                        //printf("llego codo a home\n");
                        mandareset();
                        //+++++++++++++++++ INCLINAR  ++++++++++++++++++++
                        //printf("\va inclinar a home\n");
                        entri = finclinar(8000);  //********* inclinar ****
                        mandareset();
                        if (entri !=1)
                        {
                                mandasenti(0);
                                dadas =0;
                                prepcont9();
                                dadas= leecontrol();
                                mandareset();
                                words(roteincvu,0,250);
// manda 90 grados a inclinar para su posicion de home
// considerando que 90 grados equivalen a 250
                                pwm(roteincpw,pwm50);
                                conta =0;
                                while ((dadas < 273) &&(conta != 6000))
                                {       dadas= leecontrol();
                                        pwmes(roteincpw,pwm50);
                                        conta++;
                                }
                                mandareset();
        //printf("dadas en inclinar es %d ",dadas);//getch();
                                registros9();
        //printf("\n Es el contador para esperar a que incli a -90 \n");
                                if (conta == reloj)
                                { mandareset();
                                  errores("INCLINACION");
                                }
//como el  angulo de inclinacion se conserva; al final se pondra en
//el valor de -65 respecto a la horizontal
                                //++++++++++++++++++ ROTAR ++++++++++++++++++
                                //printf(" Va el rotar a home \n");
                                entrr = frotar();  //********* rotar ******
                                if (entrr != 1)
```

```
                { //      printf("llego rotar a home y termina el programa");
/*printf("\nManda el efector a home cierra la mano cualquier tecla\n");
                                        getch();
                                        clrscr();
                                        //entrm = fmano(); //*****  mano ******
                                        /*if (entrm == 1)
                                        errores(" EL EFECTOR"); */
                                }
                                else
                                        errores(" AL  ROTAR");
                        }
                        else
                        {
                         errores(" AL INCLINAR");
                        }
                }
                else  //error codo
                {
                        errores(" EL  CODO");       //lleva al codo a home
                }
        }                       //no error el brazo
        else  //error el brazo
        {
         errores(" EL BRAZO");
         mandareset();
        } //else error brazo
}       //no error torso
else     //error torso  if del entrttorso
{// getch();
 errores(" EL  TORSO ");
}//else de error torso
entrm = 0;
mandareset();
errmat(entrt,entrc, entrb, entrr, entri, entrm);
mandareset();
cierragraf();
return(0);
}
//Programa realizado por Norma Isabel Ortega Martinez
//Realiza la transmisión y la trasformación de los datos necesarios
//para poder controlar la inclinación y la rotación de la mano.
main()
{
//Control manual rotar e inclinar
//si tiene sentidos iguales sera incli
//y si no será rotacion
FILE *ri;
float vel,sent1,sent2,grados;
int senti,vul1,vul2,conta,ctra;
int total1r,total1,dadas1,cont,ant1,diferencia;
mensajes();
configura();
mandareset();
ri = fopen("c:\\lan\\norma\\progmat\\munerot.txt","r");
```

```
if (ri == NULL)
        exit(1);
fscanf(ri,"%f",&grados);
fscanf(ri,"%f",&vel);
fscanf(ri,"%f",&sent1);
fscanf(ri,"%f",&sent2);
fclose(ri);
cont =0;
conta=0;
senti = floor(sent1)*16 + floor(sent2)*8;
if ((senti != 0) && (senti != 8) && (senti != 16) && (senti != 24))
        errores(" !! MANUAL !!");
else
{
        ctra = ciclotra(vel);
        vueltas(facrot,grados,&vul1,&vul2);
        mandasenti(senti);
        wordses(roteincvu,vul1,vul2);
        prepcont9();
        pwmes(roteincpw,ctra);
        totall r=floor(abs(grados)*facrot);
        totall = abs(totall r-((int)(0.02*totall r)));
        dadas1= leecontrot();

        while ((dadas1 < totall) && (conta<2))
        { pwmes(roteincpw,ctra);
                dadas1 = leecontrot();
                if (fmod(cont,490) == 0)
                        ant1 =leecontrot();
                if ((fmod(cont,620) == 0) &&(ant1 == dadas1))
                        conta++;
                cont++;
        }
        diferencia =(dadas1-totall r);
                        //printf("la diferencia es %d ",diferencia);
                        //getch();
        mandareset();
        if ( abs(diferencia) >20)
        {
                if ((senti == 0) ||(senti == 24))
                        errores("INC MANUAL");
                else
                  errores("ROT MANUAL");
        }
}
cierragraf();

}
//Inicia programa de cinematica inversa PRUCAM.C
//Obtiene la transformación de datos necesaria,
//transmite y recibe los datos utilizando el puerto paralelo 8255.
{       FILE *prueb;
        float dato1,dato2,dato3,dato4,dato5;
        mensajes();    //mensaje de OK PREOCESO
```

```
inipctio();    //Inicializa la tarjeta PCTIO10
configura();    //Configura A de salida B entrada y C          de salida
prueb = fopen("c:\\lan\\norma\\progmat\\angupru.txt","r");
        if (prueb == NULL)
                exit(1);
        fscanf(prueb,"%f",&dato1);        //teta1
        fscanf(prueb,"%f",&dato2);        //teta2
        fscanf(prueb,"%f",&dato3);        //teta3
        fscanf(prueb,"%f",&dato4);        //teta4
        fscanf(prueb,"%f",&dato5);        //teta5
fclose(prueb);
erro = mandacinco(dato1,dato2,dato3,dato4,dato5);
if (erro !=0)
        errores("EL PUNTO");
return(0);
} //fin del main PRUCAM.c************
//Programa Realizado por Norma Isabel Ortega Martinez
//Para la realización de la opción TAREA o repetición de
//los datos seleccionados se utiliza este programa
main()
{
FILE *taree;
float dat1,dat2,dat3,dat4,dat5;
int contador,iteracion,bandera,itera;
FILE *nombre;
char *caminito;
int banderror;
char nombrecom[33];
//Inicia la repeticion de movimientos
nombre = fopen("c:\\lan\\norma\\progmat\\nomarchc.txt","r");
        if (nombre == NULL)
                exit(1);
        fscanf(nombre,"%s",&nombrecom);
//printf("el directorio fue %s\n",nombrecom);
fclose(nombre);
//getch();
caminito =nombrecom;
taree = fopen(caminito,"r");
//taree = fopen ("c:\\lan\\norma\\progmat\\demo.txt","r");
        if (taree == NULL)
        {   printf("error de archivo terminara");
                        getch();
                        exit(1);
        }
        contador = -1;
        do
        {
                fscanf(taree,"%f",&dat1);
                contador++;
        }while (fgetc(taree) != EOF);
fclose(taree);
bandera = 0;
if (fmod(contador,5) == 0)
 bandera = 1;                    //continua
```

```
else
{ printf("error de archivo");
 getch();
}
mensajes();
configura();
inipctio();
if (bandera == 1)
{
 iteracion = contador /5;
 taree = fopen(caminito,"r");
        if (taree == NULL)
                exit(1);
        itera =0;
        banderror = 0;
        mandareset();
        while (itera < iteracion)
        {
                fscanf(taree,"%f",&dat1);
                fscanf(taree,"%f",&dat2);
                fscanf(taree,"%f",&dat3);
                fscanf(taree,"%f",&dat4);
                fscanf(taree,"%f",&dat5);
                banderror = mandacinco(dat1,dat2,dat3,dat4,dat5);
                if (banderror == 1)
                {itera = iteracion+2;           //error
                 errores("  TAREA");
                }
                else
                 itera++;
        }
  fclose(taree);
 }
 else
 {  printf("\n error de calculos");
 delay(700);}
 mandareset();
 cierragraf();
 return(0);
 }          //fin del main de tarea
```

4. Capítulo IV

INTERFAZ ELECTRÓNICA

Introducción

Actualmente, para desarrollar un sistema electrónico se cuenta con diversos tipos de circuitos integrados LSI, VLSI, etc; sólo depende del diseñador encontrar los circuitos integrados que mejor se adapten a sus requerimientos.

La interfaz electrónica para el manipulador en cuestión, requiere de lo siguiente:

-Realización de seis controladores para motores de corriente directa tipo PWM(Modulación por Ancho de Pulso); los cuales pueden ser requeridos al mismo tiempo.
-Recepción de señales de control y datos desde la PC.
-Control de Lazo Cerrado.
-Monitoreo de los sensores del manipulador.
-Control para desactivación de los motores.

Una parte de la interfaz electrónica fue diseñada utilizando un sistema de desarrollo de lógica programable, el cual permite desarrollar sistemas digitales de alta escala de integración en un sólo circuito integrado. Este tipo de sistemas tienen la ventaja de poder desarrollar sistemas de propósito específico lo que permite que las arquitecturas sean más eficientes tanto en espacio y velocidad de procesamiento como en tiempo de construcción. Una de sus desventajas fue que el circuito utilizado es OTP (programable sólo una vez) y en caso de programarlo y existir algún error de diseño, éste ya no podría ser corregido.

Este circuito, así como el software y hardware necesarios para programarlo se encontraban disponibles en el Lab. de Electrónica del Centro de Instrumentos; siendo este trabajo una gran oportunidad para probar algo que todavía no había sido utilizado.

Diseño

La interfaz electrónica fue diseñada de forma modular[1] para facilitar su entendimiento y poder cambiar algún elemento sin tener que modificar todo el diseño.

El diagrama a bloques de la Interfaz Electrónica se muestra en la figura 4.1.

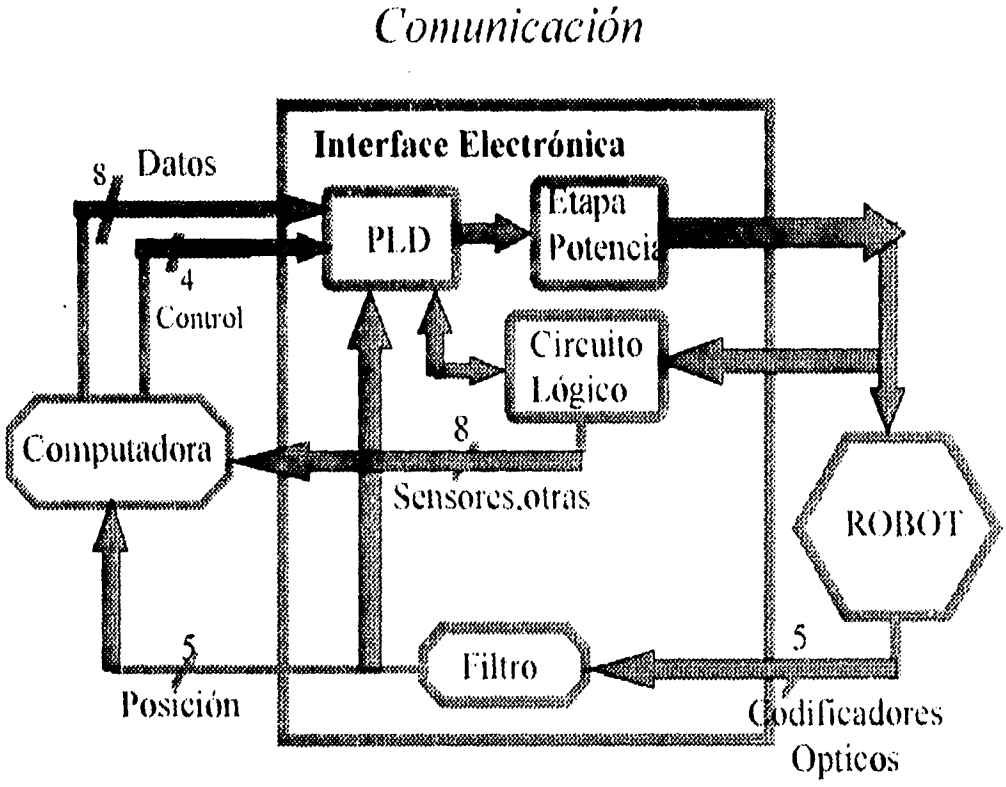


Figura 4.1. Diagrama General Interfaz Electrónica.

a) PLD

Primero serán explicados los cuatro módulos internos que integran al sistema programado en el circuito PLD EPM7256EGC192-20 ALTERA MAX, para después explicar los módulos complementarios de la Interfaz.

El primero de ellos nombrado FPWM o función que realiza un control tipo PWM. Requiere como datos una palabra de 8 bits que representará el

[1]Los diagramas de cada módulo son presentados al final del capítulo.

ciclo de trabajo de PWM requerido y una señal para la desactivación del PWM generado. La salida de este módulo es un PWM con ciclo de trabajo variable del 1% al 100% y con una frecuencia fija de reloj de 50Hz; frecuencia adecuada para evitar vibraciones en los motores.

Por otro lado se necesita tener un conteo de la cantidad de movimiento; es decir cuantas vueltas debe dar el codificador óptico para poder colocar al manipulador, por ejemplo, a 90° de su posición de inicio. El número máximo de vueltas por articulación, que los codificadores ópticos pueden dar es variable; pero mayor en todos los casos a 3500[vueltas].

Considerando lo anterior se desarrolló el módulo denominado CONTABIT2 el cual es un contador decreciente módulo 65536 (16 bits), que genera como salida un nivel bajo al momento que la cuenta que le haya sido cargada previamente (número de vueltas) llegue a cero. Este módulo tiene como entradas la señal del codificador óptico del motor, control de cuenta y carga.

Para la descodificación de los datos de la computadora, se desarrolló un tercer módulo, denominado DECF, el cual es el encargado de descodificar los datos de la computadora para activar los registros necesarios. Para ello cuenta con cuatro entradas de control, ocho de datos y una de reset. Este módulo tiene incorporados cinco FPWM (uno por cada articulación) y es el encargado de activarlos en el momento preciso, así mismo controla los registros que guardan el valor del sentido de cada uno de los motores.

El cuarto y último módulo, denominado TODO, incorpora al DECF y a cinco módulos CONTABIT2, de tal manera que representa a todo el sistema que fue programado en el circuito. Cuenta con cinco entradas provenientes de los cinco codificadores ópticos de los motores, las entradas mencionadas en DECF; es decir, en total el sistema programado cuenta con 24 entradas y 20 salidas las cuales son explicadas en la Tabla 4.1 y Tabla 4.2. Para más detalle acerca del circuito referirse al Apéndice A.

Señales de Entrada al PLD.

No. Pin	Nombre	Descripción
U10	Reloj	Señal de reloj para el sistema
R12	Clear	Señal de Reset (lógica negativa).
T13,R10,T10 R13,U12,A14, U13,U11.	Datos [8..1]	Datos provenientes de Puerto A

P9,T9,R9,U9	Control[4..1]	Datos para selección de registros. Provienen del Puerto C[0..3]
E16	Desmano	Si es igual a cero volts desactiva el movimiento del motor de la Mano.
G15	DesMyR	Desactiva los motores que controlan la Inclinación y la Rotación de la Mano.
K15	DesBra	Desactiva el motor del Brazo.
T17	DesCodo	Desactiva el motor del Codo.
J14	DesTor	Desactiva el motor del Torso.
H15	EncoTor	Entrada del Codificador óptico del motor del Torso.
T14	EncoCo	Entrada del Codificador óptico del motor del Codo.
T15	EncoBra	Entrada del Codificador óptico del motor del Brazo.
D15	EncoRyM	Entrada del Codificador óptico de los motores de Rotación e Inclinación.
F15	EncoMano	Entrada del Codificador óptico del motor de la Mano.

Tabla 4.1.Señales de Entrada.

Señales de Salida del PLD.

No. Pin	Nombre	Descripción
B1	PWMBra	Salida tipo PWM para el motor del brazo.
N1	PWMCodo	Salida tipo PWM para el motor del codo.
U1	PWMTor	Salida tipo PWM para el motor del torso.
B9	PWMyR	Salida tipo PWM para los motores de rotación e inclinación.
B6	PWMano	Salida tipo PWM para el motor de la mano.
L16	Sent1	Sentido del torso.
T12	Sent2	Sentido del brazo.
T11	Sent3	Sentido del codo.
M17 L15	Sent4 Sent5	Sentido de la rotación y la inclinación.
L17	SentMano	Sentido de la mano.

F3	CerTor	Señal activada en bajo cuando el contador del Torso ha llegado a cero.
J2	CerCodo	Igual que la anterior pero para el Codo.
U6	CerBra	Igual que la anterior pero para el Brazo.
C9	CerRot	Igual que la anterior pero para Rotación e Inclinación.
F17	CerMano	Igual que la anterior pero para la Mano.
J16	Dec11	Activada cuando el manipulador ha llegado al punto.
K16	Dec13	Señal directa del descodificador del módulo DECF.
K17	Dec14	Señal directa del descodificador del módulo DECF.
J15	Dec15	Señal directa del descodificador del módulo DECF.

Tabla 4.2. Señales de Salida.

Cada una de las entradas al PLD cuenta con su respectiva resistencia de "pull up" y sus salidas con un "buffer", porque el circuito es de tecnología CMOS y de esta manera se tiene una correcta interfaz con lógica TTL.

b) MÓDULOS COMPLEMENTARIOS

Los módulos que complementan al circuito programado para tener la interfaz electrónica completa son los siguientes:

Módulo Circuito Lógico

Dadas determinadas condiciones de los sensores del manipulador y de la dirección de los movimientos de las articulaciones, deben ser inmovilizados uno o más motores.

Para no incluir todas las condiciones que se pudiesen presentar, se muestran las ecuaciones lógicas minimizadas, mismas que fueron programadas utilizando el simulador Design Center 6.1 y el programador SuperPro en un GAL22V10-25LNC. Resultando más conveniente que si se hubiese realizado utilizando compuertas.

Ecuaciones Minimizadas

$$desrei = swr1 swnm\, sent5 + swr1 swnm\, sent4 + swnm\, sent4 sent5 + swr2 swnm\, sent4 + swr2 swnm\, swnm\, sent5 + swr2\, sent4 sent5$$

$$desbra = swnbra swncad\, sentbra + swnmbra\, sentbra swcod + swbra swnbra swcod\, sentbra$$

$$descod = swncod\, swnm\, sentcod + swncod\, sentcod\, swnm + swcod\, swncod\, swnm\, sentcod$$

$$destor = swn1\, sentor + swn2\, sentor$$

Donde:

desrei = Señal para deshabilitar los motores de rotación e inclinación
destor = Señal para deshabilitar el motor del torso
desbra = Señal para deshabilitar el motor del brazo
descod= Señal para deshabilitar el motor del codo

El proceso programado en el circuito GAL22V10 se muestra a continuación.

```
PROCEDURE GAL( INPUT swbra, swnbra
   , swcod, swncod, sentbra, swnm, swnm
   , sentcod, swr1, swr2, sent4, sent5
   , swn1, swn2, sentor,cerbra,cerotar,rstgral; OUTPUT desbra
   , descod, desrei, destor );
   VIRTUAL NODE cbra,crot;

cbra =
((/swnbra)*(/swncod)*(/sentbra))+(swnbra*(/swcod)*sentbra)+((/swbra)*(/swnbra)*(/swcod)*sentbra);

descod =((/swncod)*(/swnm)*(/sentcod)) +(swncod*(/swnm)*sentcod) +
((/swcod)*(/swncod)*(/swnm)*sentcod);


destor = ((/swn1)*(/sentor))+((/swn2)*sentor);

crot =
((/swr1)*swnm*sent5)+((/swr1)*(/swnm)*(/sent4))+((/swnm)*(/sent4)*(/sent5))+((/swr2)*(/swnm)*sent4)+
(swr2*(/swnm)*(/swnm)*sent5)+ ((/swr2)*sent4*(/sent5));

desbra = cerbra*rstgral*cbra;

desrei = cerotar*rstgral*crot;

END GAL;
```

El diagrama del Gal22V10 se muestra en la figura 4.2.

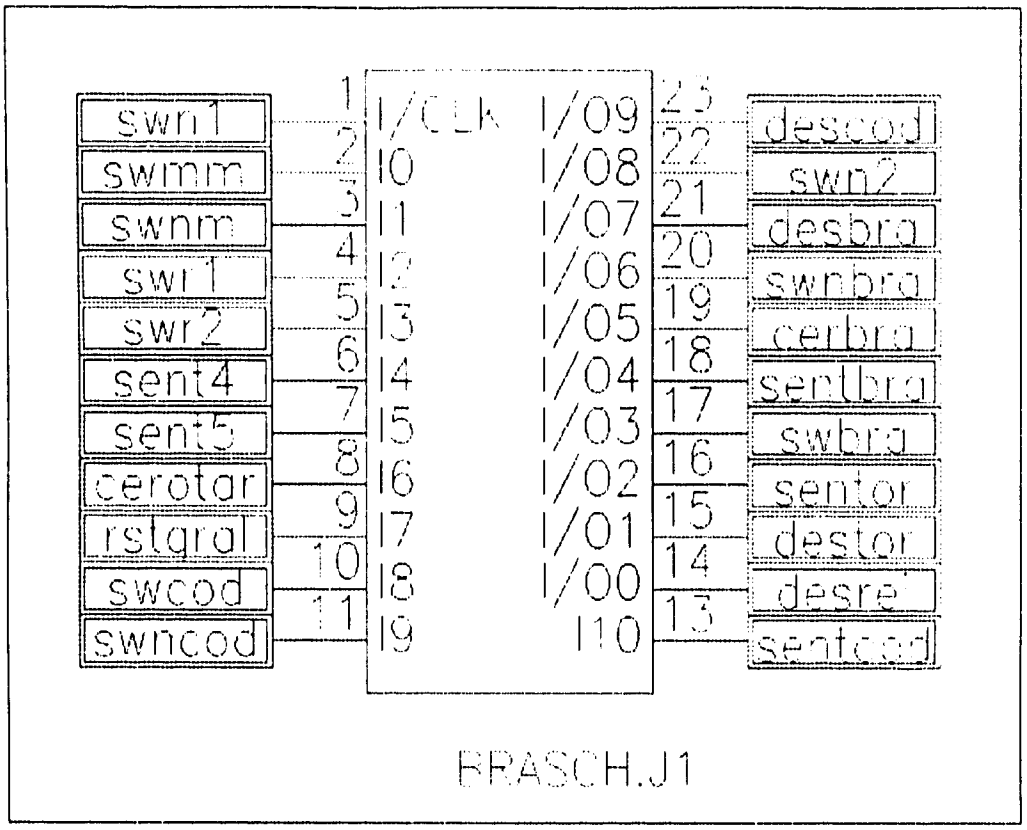


Figura 4.2. GAL22V10.

Módulo Filtro

Se utilizó un circuito LM556 en configuración monoestable para filtrar las señales recibidas de los codificadores ópticos de cada uno de los motores.

Módulo Etapa de Potencia

Para la realización de la etapa de potencia se utilizó un circuito LM18293 (controlador de motores de baja potencia) que permite como entrada una señal TTL y como salida el voltaje que alimenta al motor. Las especificaciones, así como la configuración utilizada se muestran en el Apéndice C. Este dispositivo es un circuito tipo puente H integrado; que permite controlar dos motores tanto en el sentido de movimiento como en velocidad.

DIAGRAMAS

La Interfaz electrónica se muestra en diagramas individuales para cada articulación. Estos diagramas muestran la unión entre cada uno de los circuitos que han sido explicados. Posteriormente, se muestran los diagramas correspondientes al PLD.

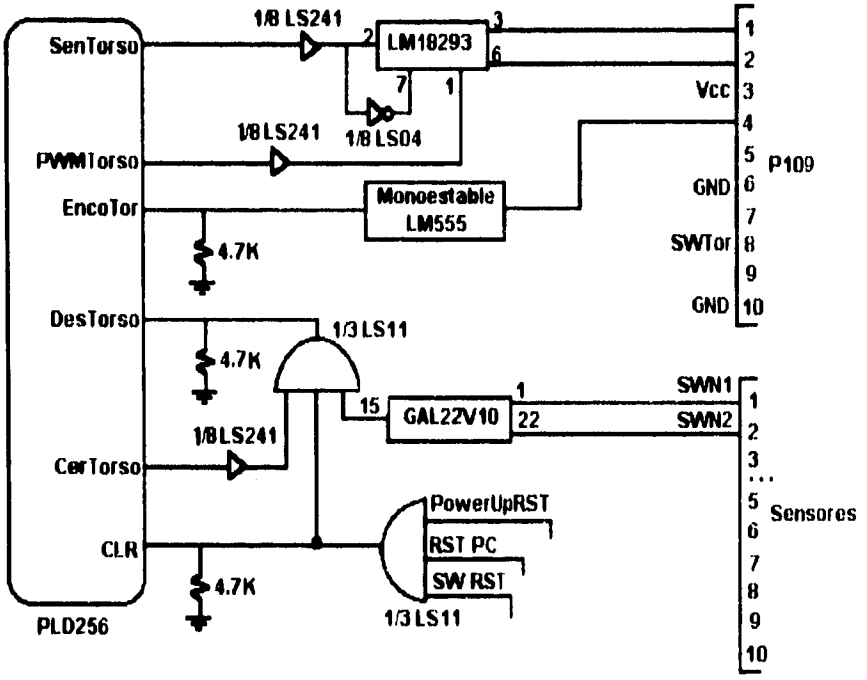


Figura 4.3. Diagrama para el torso.

El funcionamiento del diagrama mostrado en la fig 4.3 es de la siguiente manera:

El programa de comunicación Prucam.C envía el sentido de giro del torso, la velocidad del motor y la cantidad de movimiento, estas señales se procesan por el PLD. El sentido y la velocidad son enviados al circuito LM18293 y de allí al conector P109 (pines 1 y 2) correspondientes al motor del torso. La cantidad de movimiento es almacenada en el contador para el torso del módulo Contabit2 del PLD.

Al momento de girar el motor, el codificador óptico girará, generando la señal recibida en el pin 4 del conector P109, esta señal es filtrada a través de un circuito LM555 configuración monoestable (generando un tren de pulsos) llegando al PLD como la señal EncoTor. Esta señal irá decrementando al contador que ha sido cargado previamente.

Cuando la cuenta llegue a cero, la señal CerTor generará un nivel bajo que pasa por el circuito LS11 y desactiva al motor a través de la señal DesTorso del PLD.

El circuito GAL22V10 tiene dos entradas del torso SWN1,y SWN2 que son los límites para movimiento izquierdo y derecho del torso; el GAL genera un nivel bajo cuando alguno de los sensores es presionado y el movimiento del torso no puede continuar.

Por último, si alguna de las tres entradas al circuito LS11 (CerTorso, CLR, GAL) tiene nivel bajo, el motor no se moverá hasta que las tres lo permitan; es decir, hasta que no este en reset, la cuenta no sea cero y si pueda moverse la articulación.

El funcionamiento de los diagramas restantes es similar variando solamente en los sensores que intervienen y, en el caso del brazo y la mano el circuito LS11 esta incluido en el GAL22V10.

Por otro lado, el manipulador cuenta con un sistema de poleas para realizar el movimiento de inclinación y rotación de la muñeca, el diagrama que muestra como es colocado el juego de cables, así como el diagrama de la fuente de alimentación y otros, se encuentra incluido en el Apéndice B.

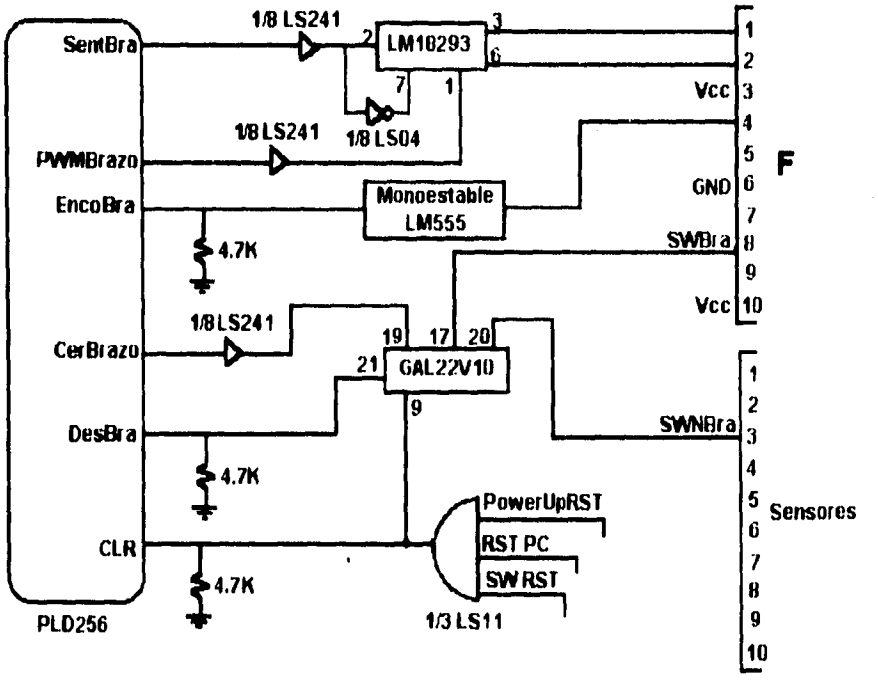


Figura 4.4.Diagrama para el brazo.

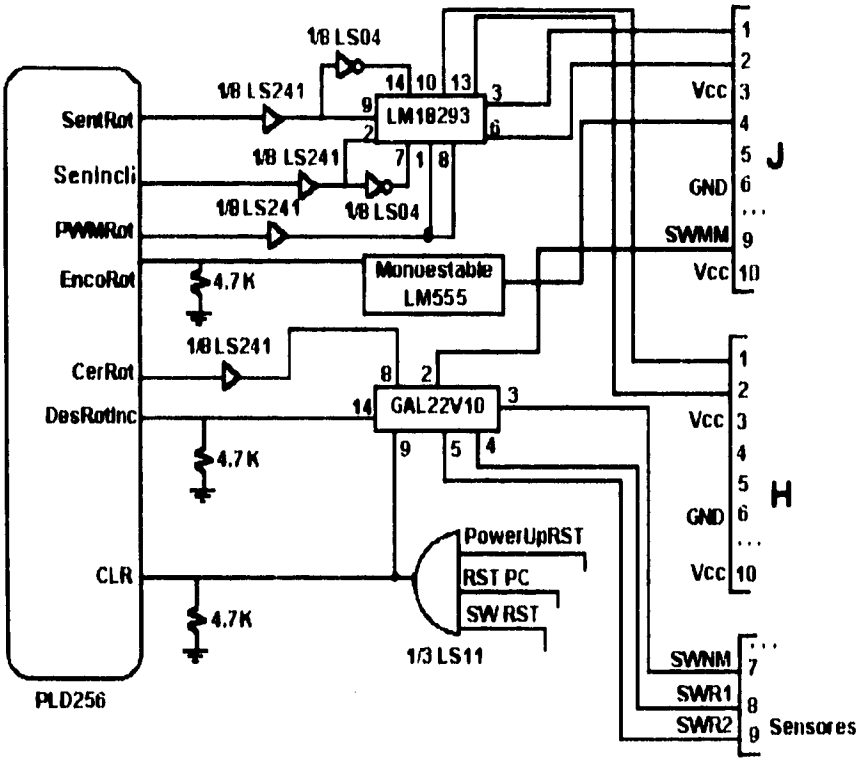


Figura 4.5. Diagrama para la Mano Rotación e Inclinación.

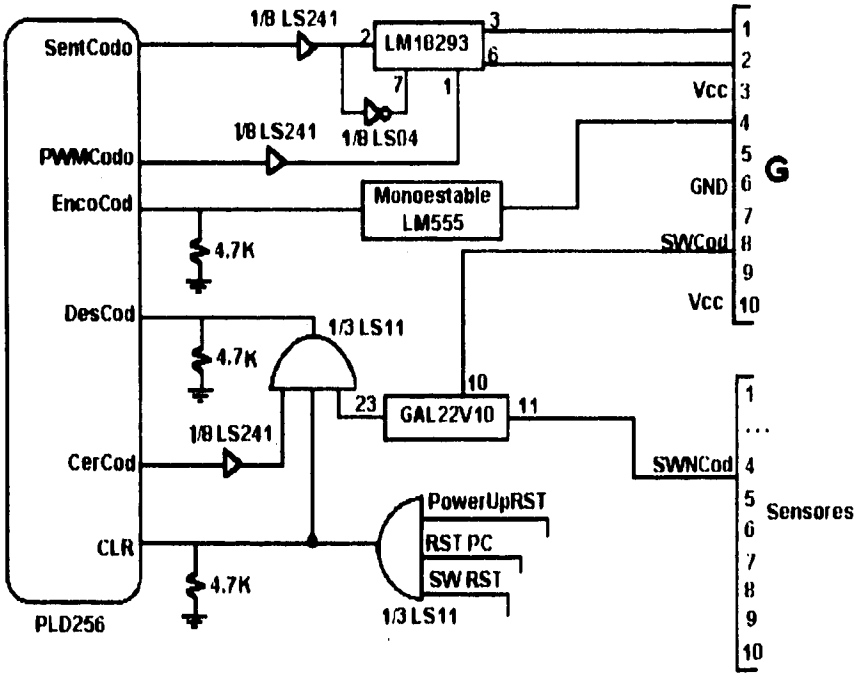
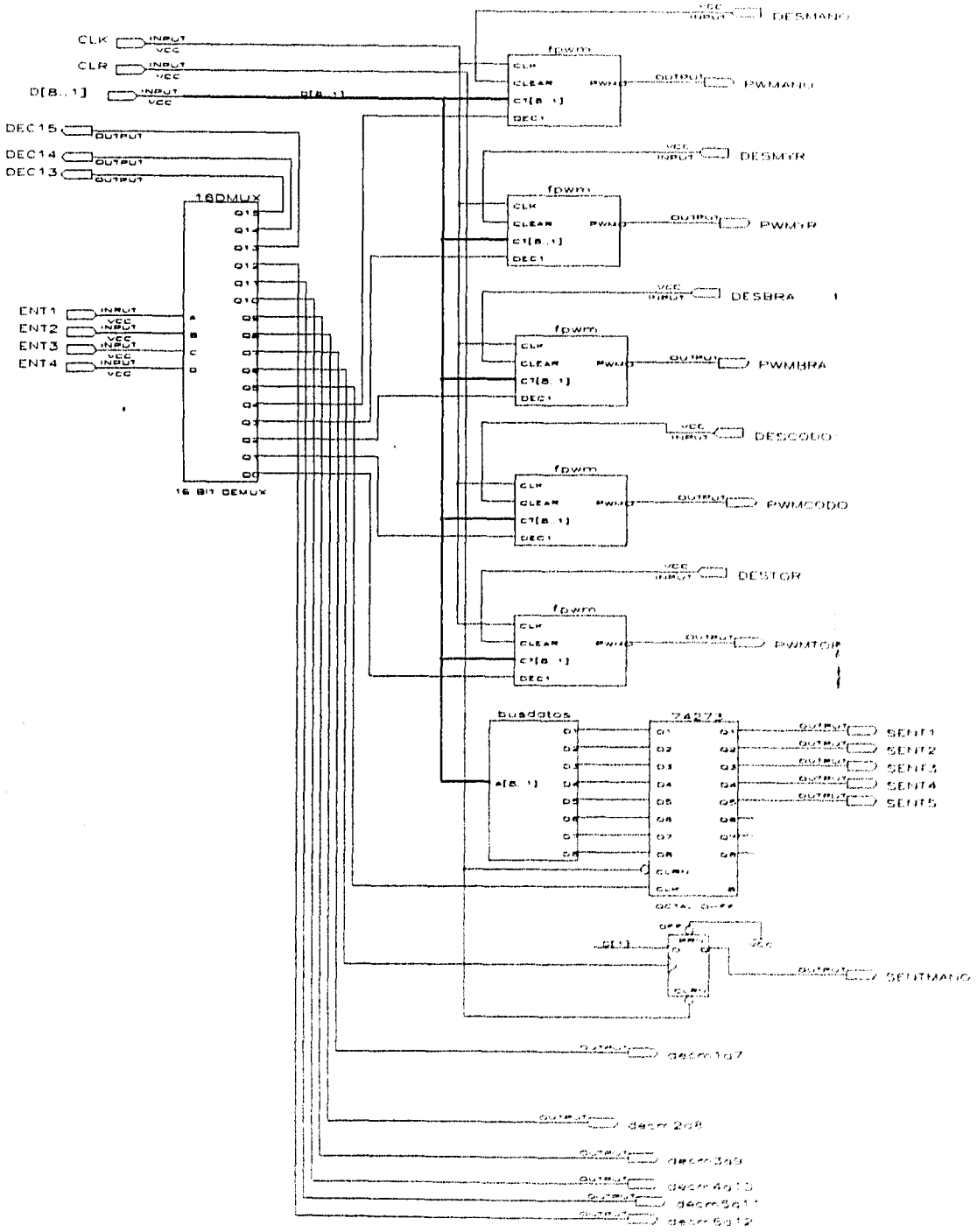


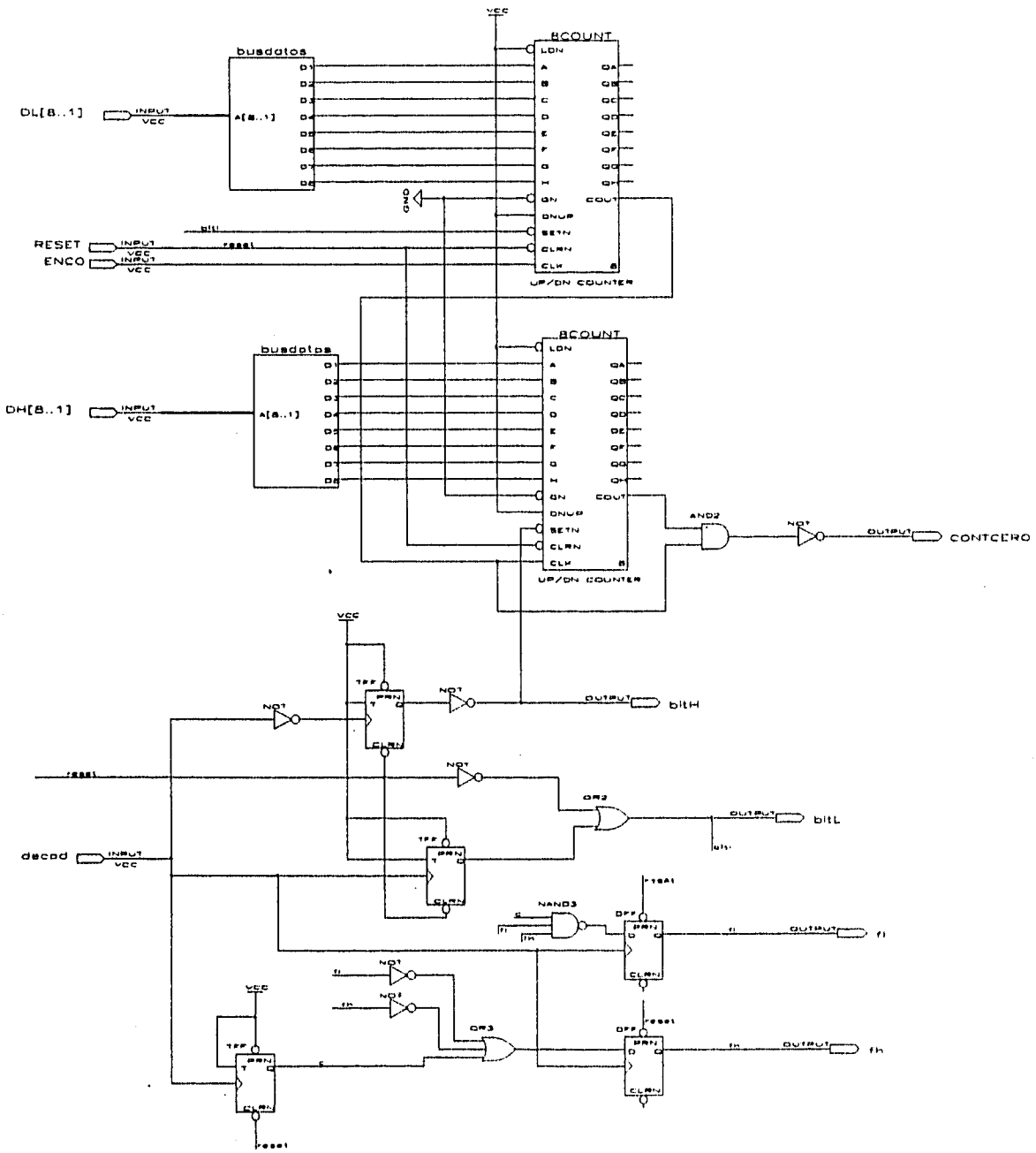
Figura 4.6. Diagrama para el codo.

```
FUNCTION 8MCOMPB (A[7..0],B[7..0]) RETURNS(ALTB,AEQB,AGTB,AEB[7..0]);

SUBDESIGN FPWM
(CLK,CLEAR,CT[8..1],DEC1        :INPUT;         //Definición de entradas
PWMO                            :OUTPUT;)               //y salidas

VARIABLE
COMPA     : 8MCOMPB;                            //Comparador de 8 bits
CONT[8..1] : DFF;                               //Registro para contador módulo 256
CW          : SRFF;                             //Flip Flop RS
RCT[8..1]  : DFF;                               //Registro 8 bits para ciclo de trabajo


BEGIN
RCT[1].PRN = GND;                               //Inicialización de señales de
RCT[8..2].PRN =VCC;                             //Control a los flip flop's
RCT[].CLK =DEC1;                                //y definir entradas
RCT[].CLRN = CLEAR;
RCT[8..2] = CT[8..2];
RCT[1] = VCC;


CW.CLRN = CLEAR;                                //señal externa para inicializar en ceros a los flipflop's
CW.R = COMPA.AEQB ;
CW.CLK = CLK;                                   //reloj externo

CONT[].CLK = CLK;                               //se construye el contador módulo 256
CONT[].CLRN = CLEAR;
CONT[].D = CONT[].Q + 1 ;
COMPA.A[] = RCT[].Q;                            //se compara la cuenta del contador y el valor del
registro
COMPA.B[] = CONT[].Q;                                   //del PWM

PWMO = CW.Q;                                            //Se define como salida PWM, la salida del
flipflop RS

IF CONT[].Q == B"00000000" THEN                 //Se obliga a conmutar a cero al flipflop RS
dependiendo
        CW.S = VCC;                             //de la cuenta del contador.
ELSE
        CW.S = GND;
END IF;
END;
```

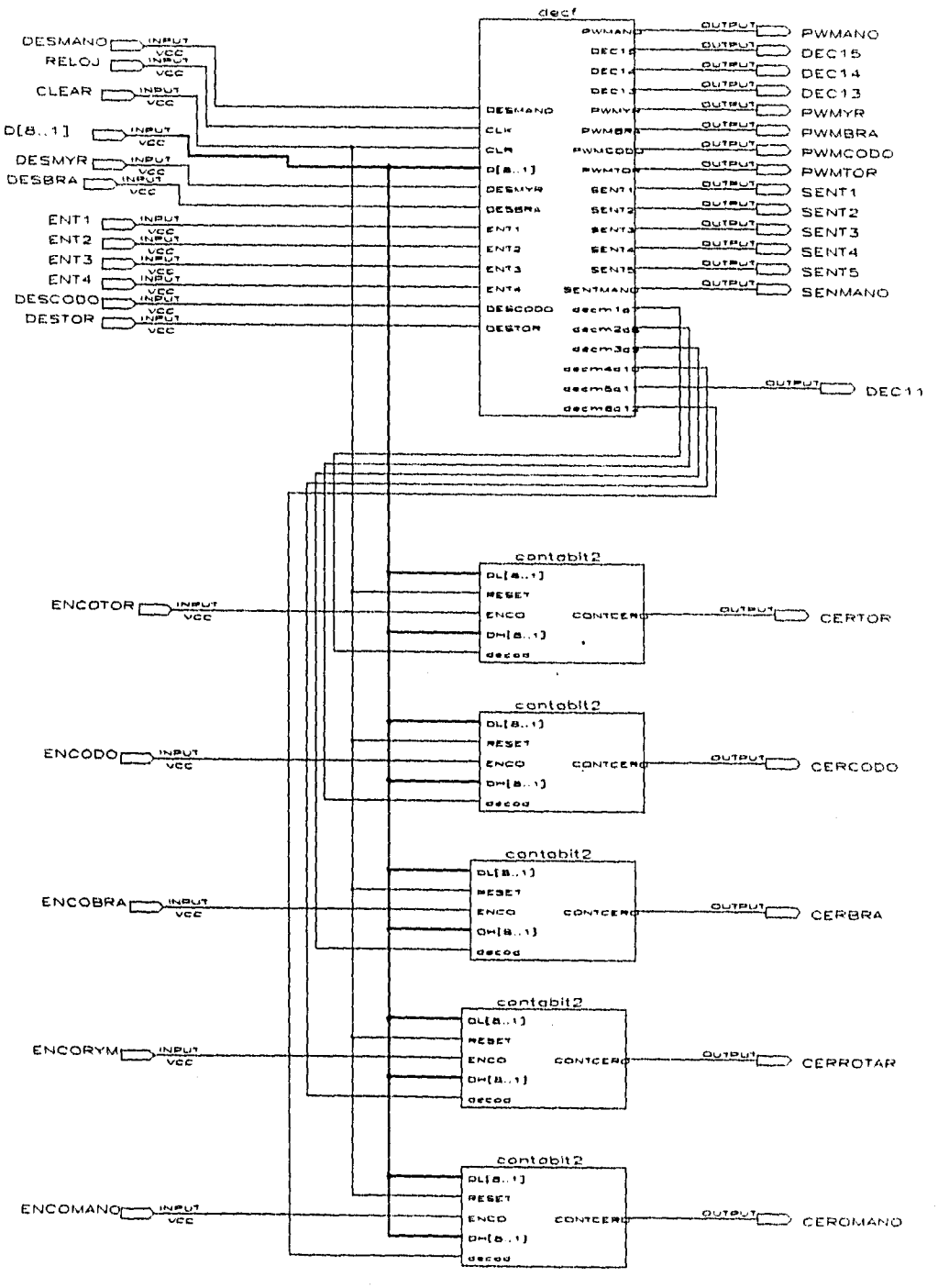
Módulo FPWM.TDF



Módulo DECF.GDF



Módulo CONTABIT.GDF



Módulo TODO.GDF

CONCLUSIONES

El sistema, además de poseer un programa capaz de permitir una interfaz gráfica amigable y una comunicación sencilla al exterior, reduciendo el tiempo de entrenamiento de los usuarios, cuenta también con un hardware flexible.

Para lograr lo anterior, el diseño de la interfaz electrónica, así como el programa que realiza todo el control, fue realizado de manera modular permitiendo realizar modificaciones dentro de cualquier módulo sin alterar de manera significativa el diseño original. Se puso énfasis en una adecuada documentación.

Todas las partes que forman al sistema presentado son importantes pero, quizá una de las partes decisivas en este proyecto fue obtener un valor adecuado de θ_4 (ángulo de inclinación), ya que este valor depende de si el punto deseado es alcanzado o no. Si se varía un poco la inclinación de la mano, el punto puede ser inalcanzable aún estando dentro del campo de trabajo.

Se logró implementar un módulo que controlara a seis motores de corriente directa utilizando un PWM variable del 1% a 100% del ciclo de trabajo, y con una frecuencia que si se requiere podría ser también variable. Este diseño fue el único programado en un circuito OTP con todas las desventajas que esto implica. Este módulo funcionó sin tener que incluir partes extras para su funcionamiento.

El software que permite llevar al manipulador a su posición de referencia estando en cualquier punto, cuenta con rutinas no tan sencillas de comprender, pero la observación fue determinante en ese aspecto.

El conocimiento adquirido a lo largo del desarrollo de este proyecto fue muy amplio, permitiéndome utilizar herramientas de diversos tipos tanto en software como en hardware; oportunidad que recibí en el Laboratorio de Electrónica del Centro de Instrumentos UNAM.

6. BIBLIOGRAFIA

John J. Craig. *Introduction to Robotics Mechanics and Control* Ed. Addison Wesley,Second Edition, México.

National Semiconductor Corp. *Linear Data Book*. Ed. National Semiconductor Corp.

Microsoft Windows. *Matlab User Reference Guide*. Ed. Microsoft Windows.

Microsoft Windows. *MatLab Guide User Interface*. Ed. Microsoft Windows.

Altera Max. *Data Book*. Altera Corporation. 1993.

Motorola Inc. *Schottky TTL Data*. 1983

APENDICE A


ALTERA
EPM7256MC208
ALTERA
EPM7096LC84
ALTERA
EPM7192QC160
ALTERA
EPM7160QC160
ALTERA
EPM7064LC68
ALTERA

MAX 7000

Programmable Logic Device Family

August 1993, ver. 1 — Data Sheet

Features...

- High-performance, erasable CMOS devices based on second-generation Multiple Array MatriX (MAX) architecture
- Complete EPLD family with logic densities up to 10,000 available gates (5,000 usable gates). See Table 1.
- Fast, 7.5-ns pin-to-pin logic delays with up to 125-MHz counter frequencies (including interconnect)
- Programmable power-saver mode for up to 50% power reduction in each macrocell
- Programmable Security Bit for total protection of proprietary designs
- Configurable expander product-term distribution allowing up to 32 product terms in each macrocell
- 44 to 208 pins available in J-lead, pin-grid array (PGA), and quad flat pack (QFP) packages, including 1-mm thin quad flat pack (TQFP)
- High pin-to-logic ratio with user-defined I/O options for pin-intensive applications such as 32-bit microprocessor interface logic
- Enhanced Programmable Interconnect Array (PIA) that provides a fast, fixed delay from any internal source to any destination in the device
- Advanced macrocell to efficiently place logic for optimum speed and density
- Programmable flipflops providing individual Clear, Preset, Clock, and Clock Enable controls
- Independent clocking of all registers from array or global Clock signals
- 3.3-V operation and advanced power management features provided by EPM7032V device

Table 1. MAX 7000 Device Features

Feature	EPM7032	EPM7032V	EPM7064	EPM7096	EPM7128	EPM7160	EPM7192	EPM7256
Available Gates	1,200	1,200	2,500	3,600	5,000	6,400	7,500	10,000
Usable Gates	600	600	1,250	1,800	2,500	3,200	3,750	5,000
Macrocells	32	32	64	96	128	160	192	256
Max. User I/O	36	36	68	76	100	104	124	164
t_{PD} (ns)	7.5	12	7.5	7.5	10	12	12	20
t_{ASU} (ns)	3	4	3	3	3	4	4	4
t_{CO} (ns)	4.5	7	4.5	4.5	5	6	6	12
f_{CNT} (MHz)	125	90.9	125	125	100	90.9	90.9	62.5

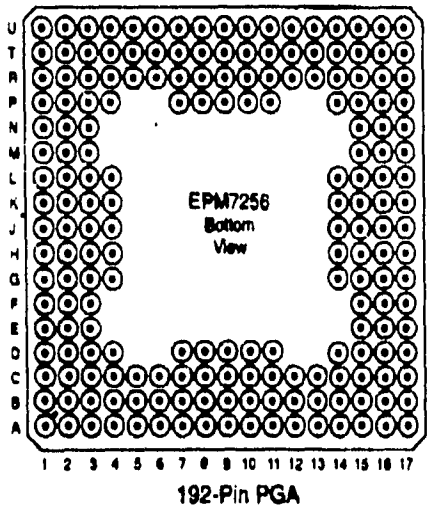
EPM7256 EPLD

Features

- High-density, erasable CMOS EPLD based on second-generation MAX architecture
  - 5,000 usable gates
  - Combinatorial speeds with t_{PD} = 20 ns (Higher speed versions under development)
  - Counter frequencies up to 62.5 MHz
- Advanced 0.8-micron CMOS EPROM technology
- Programmable I/O architecture with up to 164 inputs or 160 outputs
- 256 advanced macrocells to efficiently implement registered and complex combinatorial logic
- Configurable expander product-term distribution allowing up to 32 product terms in a single macrocell
- Available in the following packages (see Figure 34):
  - 192-pin pin-grid array (PGA)
  - 208-pin power quad flat pack (RQFP)
  - 208-pin metal quad flat pack (MQFP)

Figure 34. EPM7256 Package Pin-Out Diagrams

Package outlines not drawn to scale. See Tables 13 and 14 in this data sheet for pin-out information.



192-Pin PGA

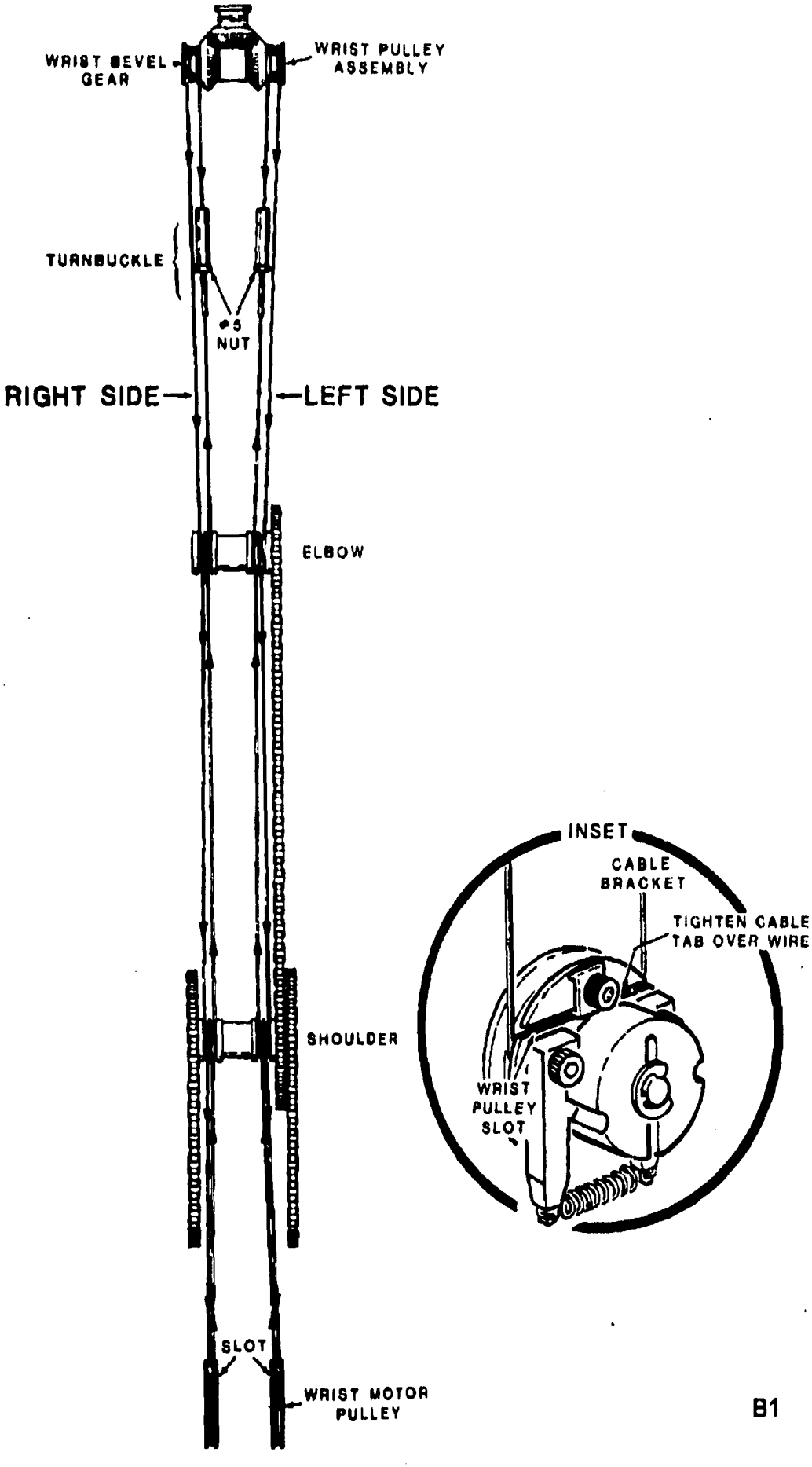
208-Pin QFP

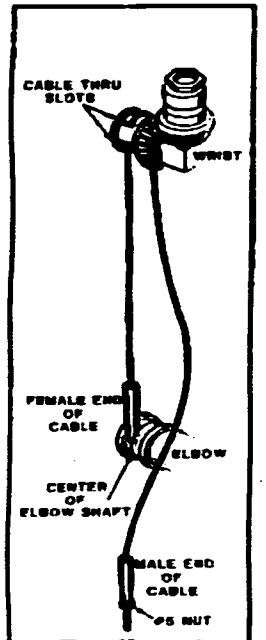
General Description

The Altera EPM7256 is a high-density, high-performance CMOS EPLD based on Altera's second-generation MAX architecture. See Figure 35. Fabricated on a 0.8-micron EPROM technology, the EPM7256 provides 5,000 usable gates, counter speeds of 62.5 MHz and propagation delays of

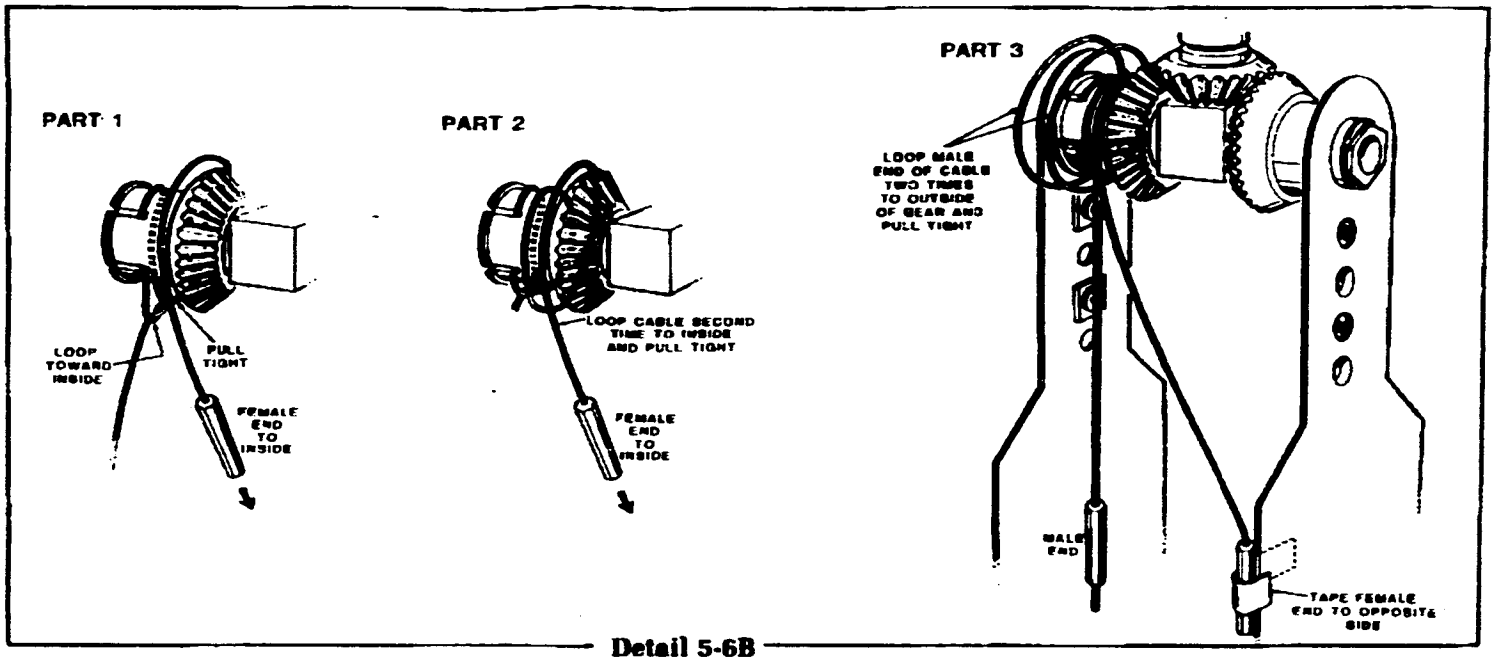
APENDICE B

Diagramas Complementarios

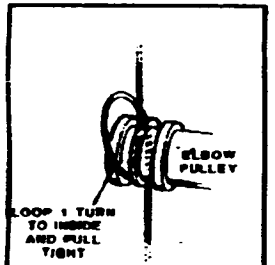

WRIST BEVEL GEAR
WRIST PULLEY ASSEMBLY
TURNBUCKLE
Ø5 NUT
RIGHT SIDE
LEFT SIDE
ELBOW
SHOULDER
SLOT
WRIST MOTOR PULLEY
INSET
CABLE BRACKET
TIGHTEN CABLE TAB OVER WIRE
WRIST PULLEY SLOT



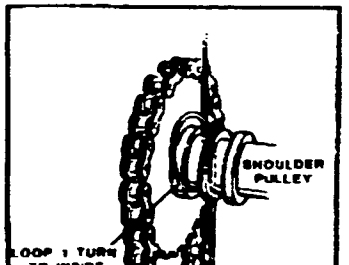
Detail 5-6A



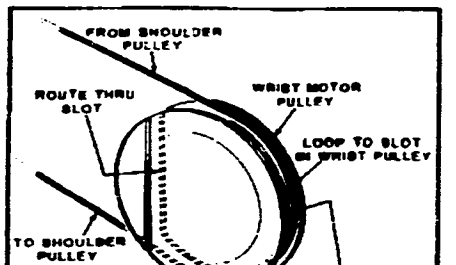
Detail 5-6B



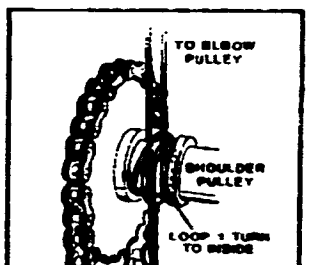
Detail 5-6C



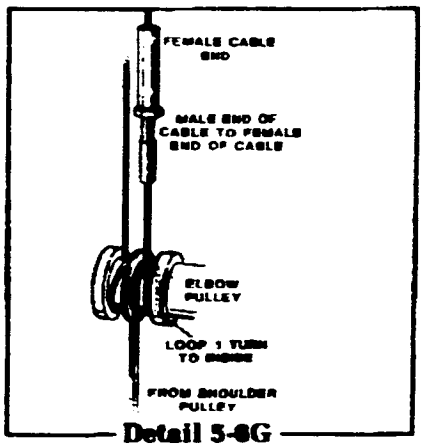
Detail 5-6D



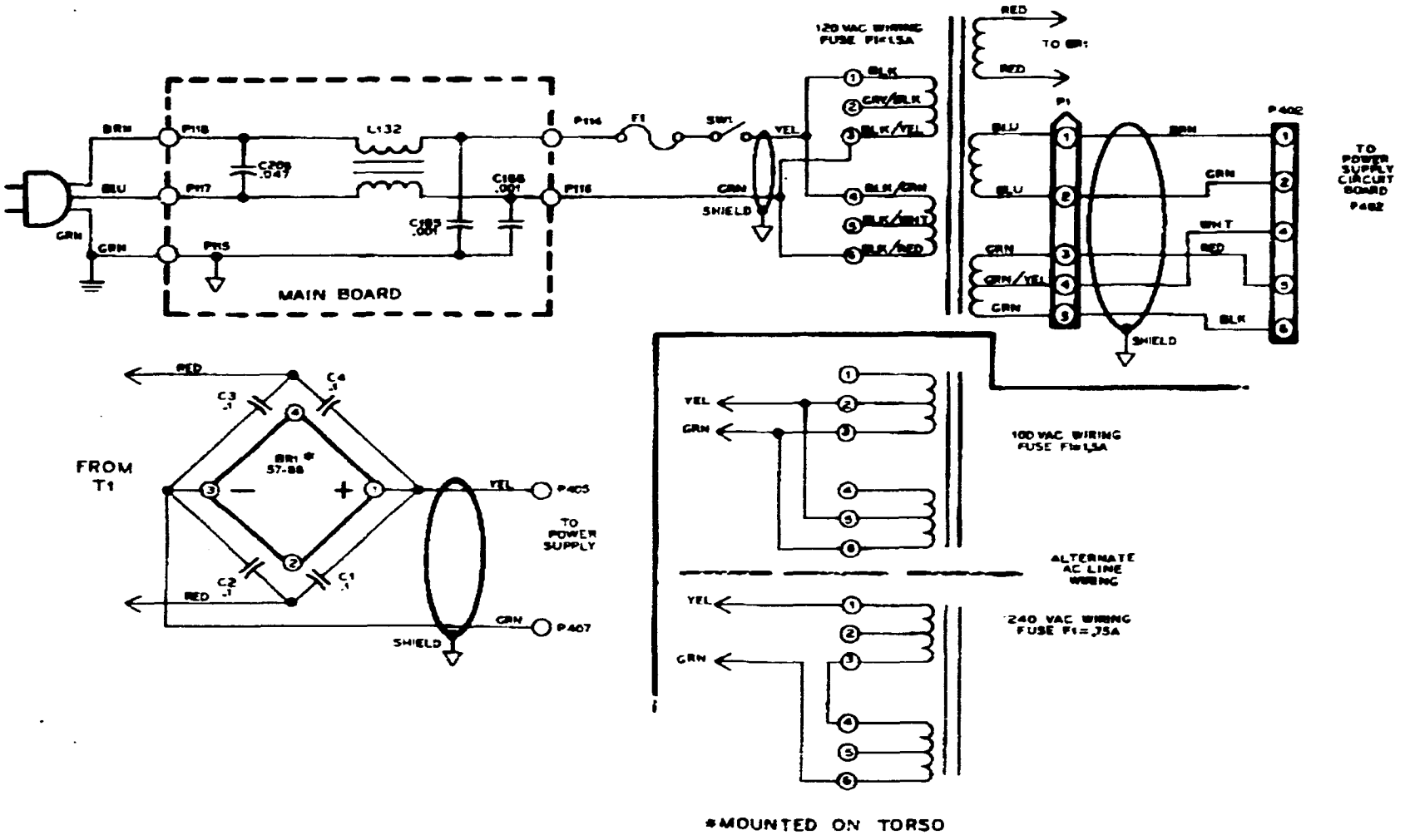
Detail 5-6E



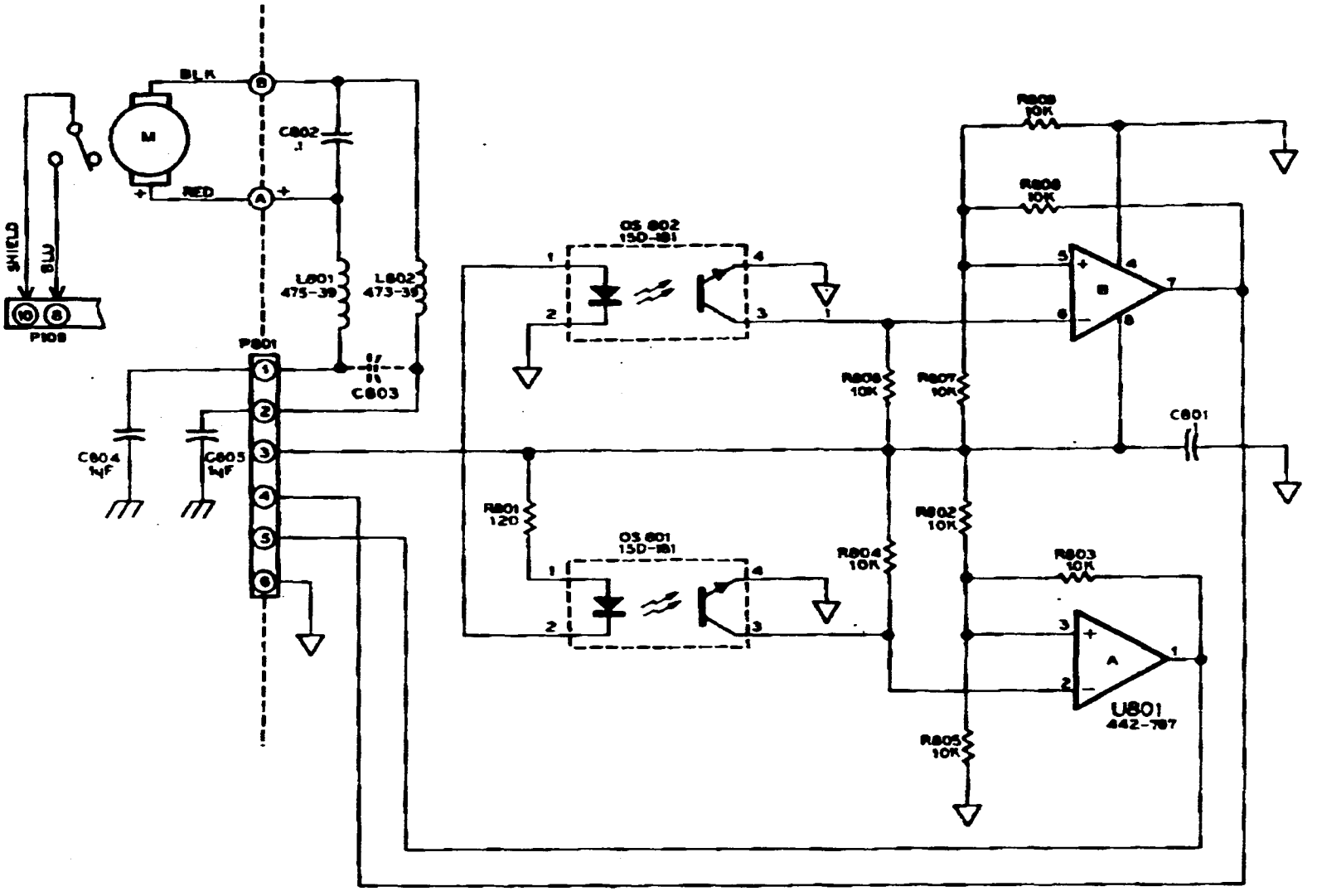
Detail 5-6F



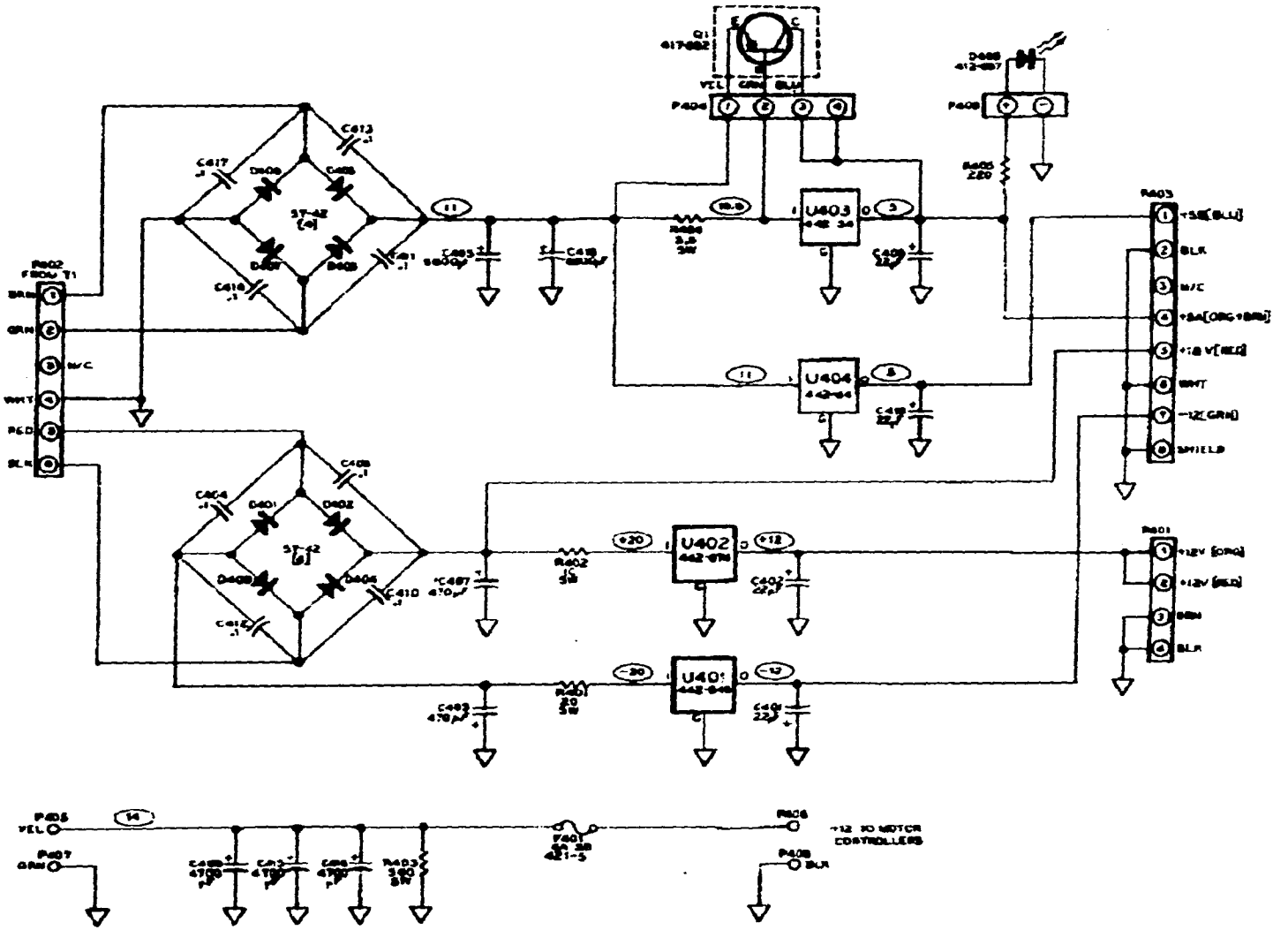
Detail 5-6G



TORSO WIRING



TORSO, ARM, ELBOW AND WRIST MOTOR ENCODER CIRCUIT BOARDS



POWER SUPPLY CIRCUIT BOARD

APENDICE C

Circuito LM18293

National Semiconductor

LM18293 Four Channel Push-Pull Driver

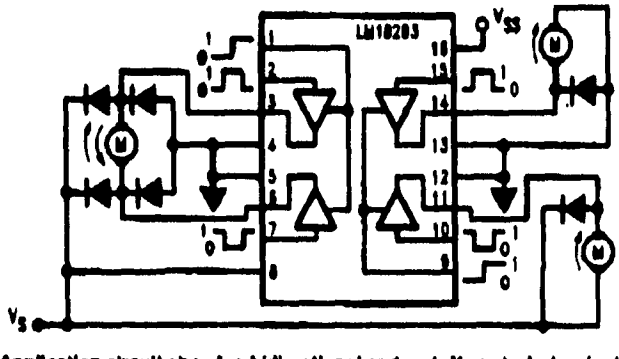
General Description

The LM18293 is designed to drive DC loads up to one amp. Typical applications include driving such inductive loads as solenoids, relays and stepper motors along with driving switching power transistors and use as a buffer for low level logic signals. The four inputs accept standard TTL and DTL levels for ease of interfacing. Two enable pins are provided that also accept the standard TTL and DTL levels. Each enable controls 2 channels and when an enable pin is disabled (tied low), the corresponding outputs are forced to the TRI-STATE® condition. If the enable pins are not connected (i.e., floating), the circuit will function as if it has been enabled. Separate pins are provided for the main power supply (pin 8), and the logic supply (pin 16). This allows a lower voltage to be used to bias up the logic resulting in reduced power dissipation. The chip is packaged in a specially designed 16 pin power DIP. The 4 center pins of this package are tied together and form the die paddle inside the package. This provides much better heat sinking capability than most other DIP packages available. The device is capable of operating at voltages up to 36 volts.

Features

- 1A output current capability per channel
- Pin for pin replacement for L293B
- Special 16 pin power DIP package
- 36 volt operation
- Internal thermal overload protection
- Logical "0" input voltage up to 1.5 volts results in high noise immunity

Typical Connection



TL/H/8706-1

FIGURE 1. Application circuit showing bidirectional and on/off control of a single DC motor using two outputs and unidirectional on/off function of two DC motors using a single output each.

Order Number LM18293N
NS Package Number N16A

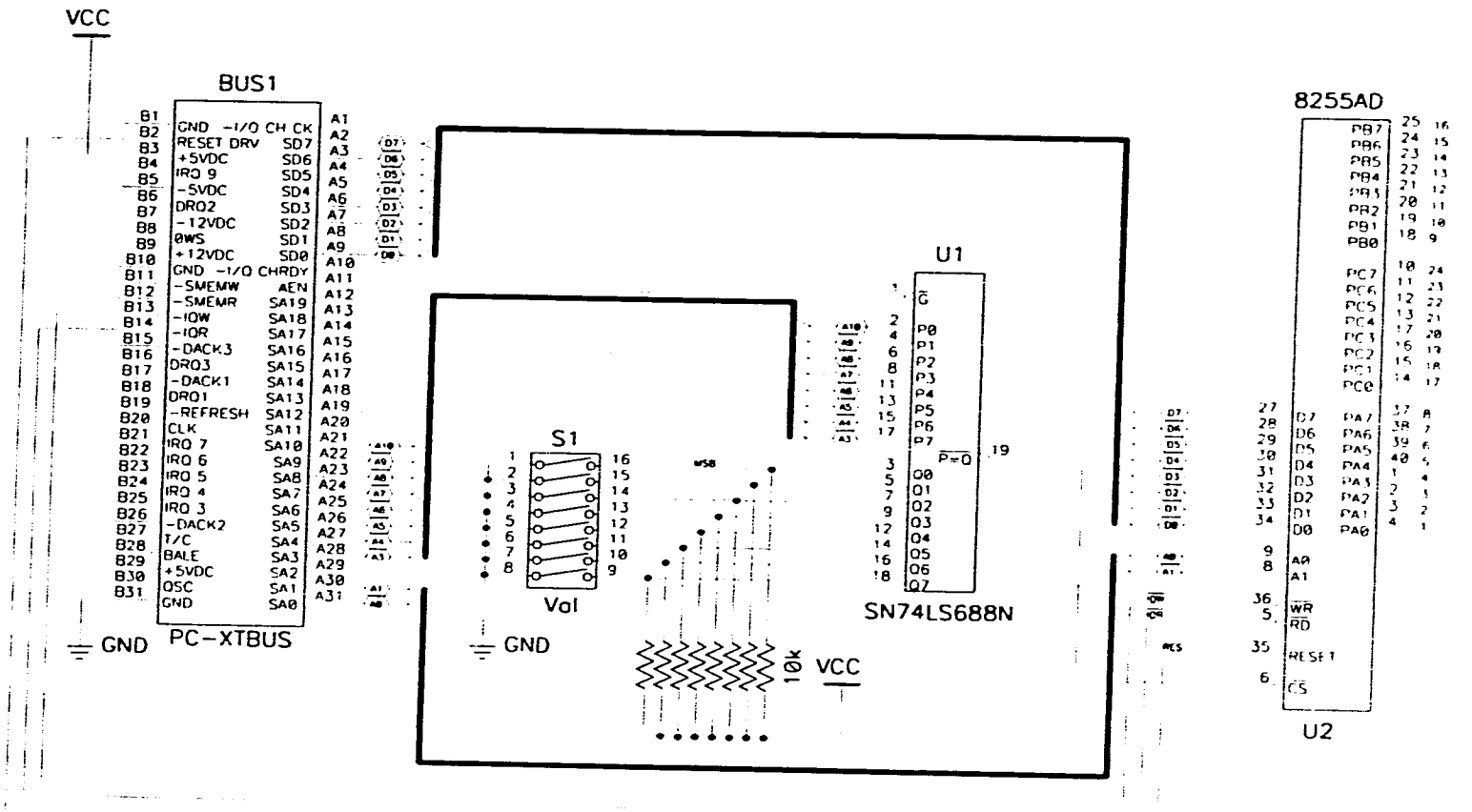
APENDICE D

Implementación Puerto Paralelo D8255AC2

APENDICE D

Implementación Puerto Paralelo D8255AC2

8255AD


VCC
BUS1
GND
PC-XTBUS
S1
Vol
GND
U1
SN74LS688N
10k
VCC
8255AD
U2

APENDICE E

Circuito AM9513

Am9513A

System Timing Controller

FINAL

DISTINCTIVE CHARACTERISTICS

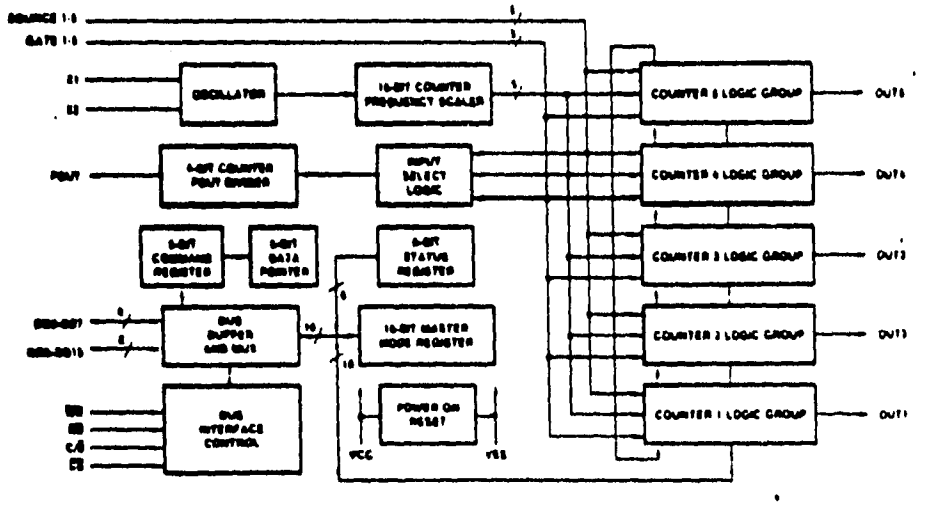
- Five independent 16-bit counters
- High speed counting rates
- Up/down and binary/BCD counting
- Internal oscillator frequency source
- Tapped frequency scaler
- Programmable frequency output
- 8-bit or 16-bit bus interface
- Time-of-day option
- Alarm comparators on counters 1 and 2
- Complex duty cycle outputs
- One-shot or continuous outputs
- Programmable count/gate source selection
- Programmable input and output polarities
- Programmable gating functions
- Retriggering capability
- +5 volt power supply
- Standard 40-pin package
- SMD/DESC qualified

GENERAL DESCRIPTION

The Am9513A System Timing Controller is an LSI circuit designed to service many types of counting, sequencing and timing applications. It provides the capability for programmable frequency synthesis, high resolution programmable duty cycle waveforms, retriggerable digital one-shots, time-of-day clocking, coincidence alarms, complex pulse generation, high resolution baud rate generation, frequency shift keying, stop-watching timing, event count accumulation, waveform analysis, etc. A variety of programmable operating modes and control features allows the Am9513A to be personalized for particular applications as well as dynamically reconfigured under program control.

The STC includes five general-purpose 16-bit counters. A variety of internal frequency sources and external pins may be selected as inputs for individual counters with software selectable active-high or active-low input polarity. Both hardware and software gating of each counter is available. Three-state outputs for each counter provide pulses or levels and can be active-high or active-low. The counters can be programmed to count up or down in either binary or BCD. The host processor may read an accumulated count at any time without disturbing the counting process. Any of the counters may be internally concatenated to form any effective counter length up to 80 bits.

BLOCK DIAGRAM



BD003381

Publication # 01731 Rev. F Amendment /0
Issue Date: August 1988