UNIVERSIDAD NACIONAL AUTONOMA
DE MEXICO

FACULTAD DE INGENIERIA

SISTEMA DE OPTICA ACTIVA DEL TELESCOPIO DE SAN PEDRO MARTIR, BAJA CALIFORNIA

TESIS
QUE PARA OBTENER EL TITULO DE
INGENIERO MECANICO ELECTRICISTA
AREA ELECTRONICA
PRESENTA:

JOSE CASTILLO HERNANDEZ

DIRECTOR DE TESIS:

ING. ROSENDO FUENTES GONZALEZ

MEXICO, D.F.1993

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

1 INTRODUCCION

1.1 NOTAS HISTORICAS

En 1929, la Universidad Nacional recibe su autonomía. En este mismo año recibe también, para su administración y cuidado, el Observatorio Astronómico Nacional *OAN*. Desde 1967, por acuerdo del Consejo Universitario, el observatorio es una dependencia del Instituto de Astronomía.

El *OAN* se encontraba situado en un principio en el Castillo de Chapultepec. En 1908 se estableció en Tacubaya, en el edificio que durante mucho tiempo se conoció como el Observatorio Astronómico de Tacubaya.

Conforme la ciudad de México fue creciendo, los factores en el medio fueron cambiando, haciendo que el lugar fuera poco propicio para el Telescopio; se debe de recordar que la ciudad a principios de siglo se encontraba con un ambiente limpio y tranquilo, con una iluminación mínima que no afectaba en forma importante al telescopio. Por tal motivo se pensó en reubicar el *OAN*.

Con la planeación y construcción de la Ciudad Universitaria, se dieron las condiciones propicias para trasladar el *OAN* a Tonantzintla, Puebla, a un predio contiguo al del Observatorio Astrofísico de la Secretaria de la Educación Pública y la dirección, las oficinas de investigadores , las bibliotecas, laboratorios y talleres, se mudaron a la Torre de Ciencias de la Ciudad Universitaria. Esta etapa de la *OAN* culmina en 1961 con el telescopio de un metro de diámetro instalado en Tonantzintla.

Una vez más, debido al crecimiento de la ciudad, se volvió a plantear la reubicación de el *OAN*.

1.2 SAN PEDRO MARTIR.

La sierra de San Pedro Mártir, fue el lugar seleccionado para la reubicación del *OAN* , después de un estudio con cartas meteorológicas y fotografías tomadas desde satélites artificiales. El factor de decisión, fue la poca

incidencia en la nubosidad, que presentaba la zona . En 1966 se inicia el estudio sobre las características del cielo en esa sierra.

Los estudios realizados mostraron que no solo hay un porcentaje muy alto de noches despejadas , sino que también la atmósfera es muy limpia y con muy bajo contenido de agua; la escasa turbulencia atmosférica hace que el tamaño de las imágenes estelares sea pequeño al presentarse un cielo nocturno muy obscuro, cosa que propicia la observación de objetos muy débiles. Un punto más a favor del nuevo lugar a seleccionar, es referente a la gran distancia que existe hacia los centros urbanos, siendo ésto una garantía para que se mantega una zona libre de contaminación.

Por lo anterior, se elegió la sierra de San pedro Mártir para el nuevo *OAN*, iniciándose las obras de construcción en 1967.

1.3 TELESCOPIOS

El *OAN* cuenta con tres telescopios , que son reflectores con foco Cassegrain; los diámetros de su óptica principal miden 84,150 y 212 cm. La óptica del primer telescopio, fue construido en el Instituto de Astronomía de la UNAM . El telescopio de 1.50 m es operado por el Instituto de Astronomía en convenio con la Universidad de Arizona. Finalmente , el telescopio de 212 cm de diámetro fue diseñado y parcialmente construido en el Instituto de Astronomía de la UNAM, con la ayuda de el Centro de Instrumentos;el resto del telescopio fue construido en partes por varias empresas estadounidenses.

1.4 SISTEMA DE OPTICA ACTIVA.

Dado que el paralaje que debe de existir entre los espejos del telescopio es de importancia fundamental, cada año en San Pedro Mártir, se realiza un proceso de mantenimiento de estos, el cual se basa fundamentalmente en verificar que los espejos estén en su posición adecuada. Este proceso se realiza en forma manual. Dado que este proceso puede tener fallas debido a factores humanos y mecánicos, se pensó en desarrollar un sistema que realice en forma automática la corrección en la posición de los espejos. El sistema será controlado

por una computadora, usando el puerto paralelo. Las funciones que debe de desempeñar el sistema, son las siguientes:

a).-Verificar continuamente el paralaje que existe en los espejos.

b).-De existir un error en el paralaje, deberá procesar información para realizar una corrección.

c).- Enviar una señal, que ponga en aviso al sistema de control de posición , así como los datos de las nuevas coordenadas.

d).- El sistema que controla la posición, deberá ubicar a los espejos en las coordenadas indicadas.

e) .- El sistema de posicionamiento, deberá de verificar que la ubicación de los espejos sea correcta, de no serlo, procesará con las coordenadas actuales, un grupo de nuevas coordenadas y realizará, una vez más el paso *d*.

El presente trabajo, realizará solo las funciones del sistema de posición. Los parámetros que deberá de manejar se muestran en la figura 1.1.

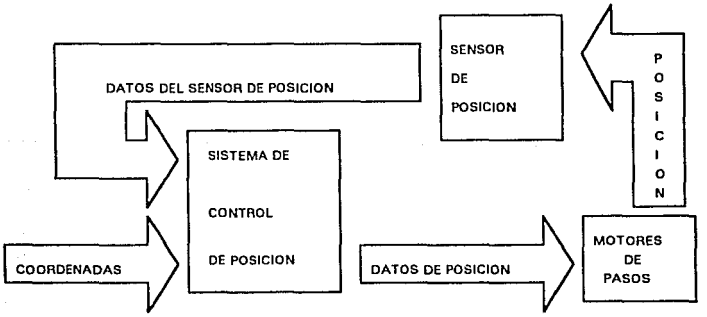


FIGURA 1.1 Descripción del sistema de control de posición

El sistema utiliza la filosofía de motores de pasos, los cuales basan su funcionamiento en un código especial, para poder realizar una fracción de giro denominado paso. Los datos de posición, corresponden al código que manejarán los motores de pasos, para realizar los movimientos pertinentes. Los motores de pasos se estudiarán con más detalle en el capítulo 3.

Las coordenadas, corresponden a la información que deberá procesar el sistema , para poder proporcionar el código adecuado a los motores de pasos. Las coordenadas son proporcionadas por el detector del paralaje de los espejos del telescopio.

El sensor de posición, se encarga de presentar en todo momento la posición de cada motor. El sensor de posición esta basado en un convertidor voltaje/frecuencia *VCO* . Su funcionamiento se estudiará en detalle en el capítulo 4. En este mismo capítulo se estudia la electrónica del sistema.

El software sobre el procesamiento de las coordenadas, datos del sensor de posición y datos de posición de los motores , se estudiaran en el capítulo 5

En el capítulo 2, se realiza un estudio sobre la óptica de los espejos del telescopio.

2 OPTICA

2.1 INTRODUCCION

En este capítulo se darán las nociones básicas sobre óptica, para comprender el funcionamiento de los espejos en un telescopio de tipo foco Cassegrain, también se analizará cual es el problema que se desea resolver.

2.2 ESPEJO CONVEXO

La figura 2.1 muestra un espejo esférico convexo de radio de curvatura *R*. El centro de la curvatura de la superficie se halla en *C*, el punto *P* es un objeto situado a la izquierda a una distancia *s*.

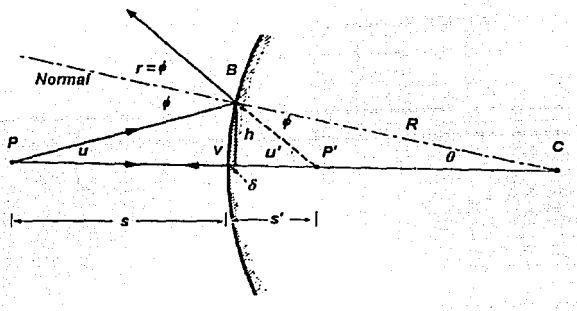


FIGURA 2.1 Análisis del espejo convexo

El rayo *PV* dirigido hacia *C*, incide normalmente sobre el espejo y se refleja, volviéndose sobre sí mismo. El punto *V* se denomina vértice, y la recta *PVC*, eje.

El rayo *PB*, que forma un ángulo cualquiera *u* con el eje, incide en *B* sobre la superficie, siendo ϕ el ángulo de incidencia y $r=\phi$ el de reflexión. Los rayos reflejados en *B* y *V*, prolongados hacia la derecha del espejo, se cortan en *P'* a una distancia *s'* a la derecha del vértice. Se calculará ahora la distancia imagen *s'*.

De los triángulos *PBC* y *P'BC* , y considerando que el ángulo exterior de un triángulo es igual a la suma de los ángulos interiores opuestos, se tiene:

$$\phi = u + \theta; \qquad u' = \phi + \theta;$$

a partir de las cuales, se obtiene :

$$u - u' = -2\theta \qquad (2.1)$$

Si denominamos a *h* la altura de *B* sobre el eje y a δ, la distancia al pie de la vertical, se obtiene:

$$\text{tg}\, u = \frac{h}{s+\delta}; \qquad \text{tg}\, u' = \frac{h}{s'-\delta}; \qquad \text{tg}\, \theta = \frac{h}{R-\delta}$$

si *u* es un ángulo pequeño, de tal forma que $\text{tg}(x) = x$, y si δ se desprecia frente *s'*, *s*, *R*, dado que *u* es pequeño, se tiene :

$$u = \frac{h}{s}; \qquad u' = \frac{h}{s'}; \qquad \theta = \frac{h}{R}$$

de donde, sustituyendo estos valores en la ecuación (2.1) y dividiendo entre *h*, se obtiene la siguiente relación:

$$\frac{1}{s} - \frac{1}{s'} = -\frac{2}{R} \qquad (2.2)$$

Por lo tanto, la distancia imagen *s'* puede ser calculada con base en *s* y *R*, sin considerar en ningun momento el ángulo *u (conservando las condiciones anteriormente dadas)*. Los rayos con un ángulo *u* pequeño, son denominados paraxiales, dado que son casi paralelos al eje.

Se debe de observar que si el ángulo *u* aumenta, el punto *P'* se aproxima al vértice, y un espejo esférico a diferencia de un espejo plano, no forma un punto imagen de un punto objeto. Esta propiedad se denomina aberración de esfericidad. Por oto lado si $R = \infty$, el espejo se convierte en plano.

Consideremos ahora un objeto de tamaño finito, representado por la flecha *PQ* perpendicular al eje *PV* de la figura 2.2.

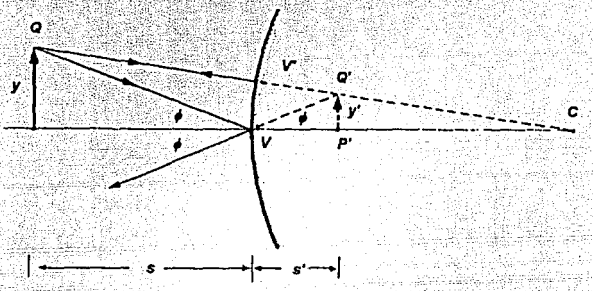


FIGURA 2.2 Construcción para determinar la altura de la imagen formada por un espejo convexo

La imagen de *P* formada por rayos paraxiales esta en *P'*. La distancia objeto para el punto *Q* es *QV'*, y puesto que ésta es mayor que *PV* , la distancia imagen *V'Q'* es menor que *VP'*. Por consiguiente, la imagen *P'Q'* no es recta sino curva , lo que constituye otra aberración de la superficies esfericas denominada curvatura de campo. Sin embargo si la altura *PQ* no es damasiado grande, la imagen es casi recta y perpendicular al eje.

Considerando lo anterior, calcularemos el aumento *m*. El rayo *QV'*, que incide normalmente, se refleja sobre sí mismo. El rayo *QV* forma un ángulo de incidencia ϕ y un ángulo de reflexión $r=\phi$. De los triángulos *QPV* y *Q'P'V*, se deduce:

$$\text{tg}\,\phi = \frac{y}{s} = \frac{y'}{s'}$$

si *y* e *y'* son las logitudes del objeto y de su imagen, la razon *y'/y* se denomina aumento *m*, por lo tanto:

$$m = \frac{y'}{y} = \frac{s'}{s} \qquad (2.3)$$

2.3 ESPEJO CONCAVO

Adoptando un convenio de signos algebraicos para *s*, *s'*, *R*, *y* e *y'* es innecesario deducir las fórmulas para el espejo cóncavo. Lo anterior se debe a que un punto objeto puede encontrarse a un lado u otro de una superficie reflectante o refrigente, de igual forma, lo mismo sucede para el punto imagen y para el centro de curvatura de una superficie esférica. Además, los objetos y las imágenes de tamaño finito pueden encontrarse por encima o por debajo del eje de la superficie.

El convenio de signos, que damos a continuación es utilizado por la mayor parte de los diseñadores de instrumentos ópticos.

1.- Dibujar los esquemas con la luz incidente propagándose de izquierda a derecha.

2.- La distancia objeto *s* es positiva si el objeto se encuentra a la izquierda de la superficie reflejante o refrigente.

3.- La distancia imagen *s'* son positiva si la imagen se halla a la derecha de la superficie.

4.- Los radios de curvatura *R* son positivos si el centro de curvatura se encuentra a la derecha de la superficie.

5.- Las dimensiones transversales *y* e *y'* son positivas si están por encima del eje.

De lo anterior, y verificando en la figura 2.3, se tiene:

$$\theta = u + \phi; \qquad u' = \phi + \theta;$$

de donde

$$u + u' = 2\theta \qquad (2.4)$$

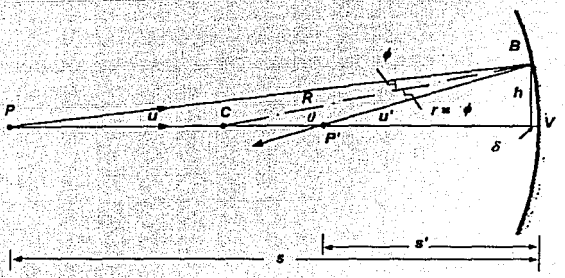


FIGURA 2.3 Análisis del espejo cóncavo

ademas, por trigonometría se tiene

$$\text{tg } u = \frac{h}{s-\delta}; \qquad \text{tg } u' = \frac{h}{-s'-\delta}; \qquad \text{tg } \theta = \frac{h}{-R-\delta}$$

haciendo las mismas consideraciones que para el espejo convexo obtenemos que:

$$u = \frac{h}{s}; \qquad u' = -\frac{h}{s'}; \qquad \theta = -\frac{h}{R} \qquad (2.5)$$

sustituyendo las ecuaciónes 2.5 en la ecuación 2.4, llegamos a la ecuación 2.6

$$\frac{1}{s} - \frac{1}{s'} = -\frac{2}{R} \qquad (2.6)$$

Por otro lado, para el aumento *m* se tiene:

$$\text{tg } \phi = \frac{y}{s} = \frac{-y'}{-s'} = \frac{y'}{s'}$$

de donde la ecuación para *m* esta dada por:

$$m = \frac{y'}{y} = \frac{s'}{s} \qquad (2.7)$$

Se puede observar que las ecuaciones son las mismas en un espejo cóncavo que en uno convexo. Lo anterior es cierto siempre que se utilicen los convenios de los signos.

2.4 FOCO Y DISTANCIA FOCAL

Si la distancia objeto *s*, es $s=\infty$, los rayos procedentes de un punto objeto que inciden en el espejo, son paralelos entre sí. Por lo tanto de la ecuación 2.6, se tiene:

$$\frac{1}{\infty}-\frac{1}{s'}=-\frac{2}{R'};\qquad s'=\frac{R}{2} \tag{2.8}$$

la ecuación 2.8, nos muestra que la distancia imagen, es igual a la mitad del radio de curvatura y tiene el mismo signo (recordar la convención de signos dada), ésto significa que si *R* es positivo, como en la figura 2.4a, el punto imagen *f* se halla a la derecha del espejo y es virtual, mientras que si *R* es negativo, como en la figura 2.4b, la imagen se encuentra a la izquierda y es real.

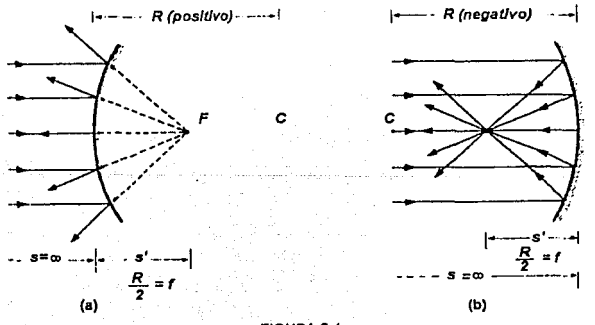


FIGURA 2.4

Por otro lado si la distancia imagen *s'*, es $s'=\infty$, de la ecuación 2.6 tenemos:

$$\frac{1}{s}-\frac{1}{\infty}=-\frac{2}{R};\qquad s=-\frac{R}{2} \tag{2.9}$$

en este caso, la distancia imagen es igual a la mitad del radio de curvatura, pero de signo contrario, es decir R es positivo, s es negativa y el objeto está a la derecha de la superficie,como en la figura 2.5a. De igual forma si R es negativo, s es positiva y el objeto se halla a la izquierda de la superficie como se muestra en la figura 2.5b

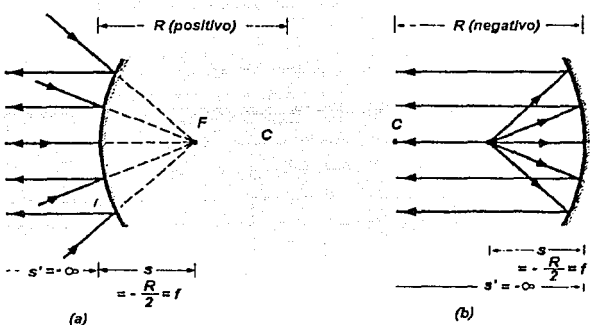


FIGURA 2.5

El punto f de las figuras 2.4 y 2.5, se denomina foco del espejo, y puede considerarse como un punto imagen de un punto objeto infinitamente distante situado sobre el eje, o como un punto objeto de un punto imagen infinitamente alejado. Con base en este principio, el espejo de un telescopio, forma en su foco la imagen de una estrella situada sobre el eje del espejo.

La distancia entre el vértice de un objeto y el foco se denomina distancia focal del espejo y se representa por f y su valor para los espejos, según el estudio anterior es:

$$f=-\frac{R}{2} \tag{2.10}$$

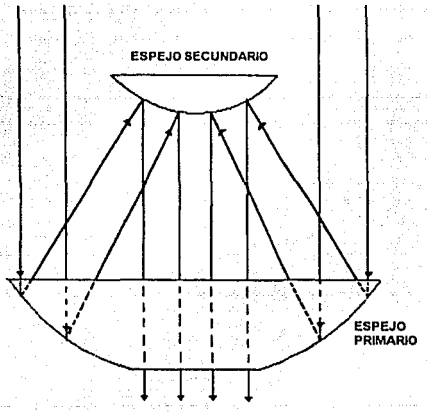
por lo tanto, la ecuación 2.6 puede ser escrita como:

$$\frac{1}{s} - \frac{1}{s'} = -\frac{1}{f} \qquad (2.11)$$

2.5 FOCO CASSEGRAIN

Como ya se mencionó en la introducción, los telescopios con que cuenta el Observatorio Astronómico Nacional, son de tipo foco Cassegrain; la figura 2.6 muestra en forma esquemática, la colocación que guardan los espejos en un telescopio con este tipo de foco.

FIGURA 2.6 Foco Cassegrain



El sistema se compone de dos espejos colocados uno frente a otro, donde uno es denominado primario, de tipo cóncavo, y otro es denominado secundario, de tipo convexo. Se debe de observar el orificio que tiene el espejo primario, a traves del cual pasará la imagen. Ambos espejos, están colocados en una estructura rígida.

Considerando la figura 2.6, si la distancia objeto es muy grande *(puede considerarse s=∞, dado que se está hablando de imágenes en el cielo)*, los rayos

que llegan al espejo primario, pueden ser considerados como paralelos y normales al plano x, considerando la teoría expuesta anteriormente, los rayos reflejados por el espejo, deberan de converger al foco real. Por otro lado, el espejo secundario esta colocado de tal forma, que su foco virtual, coincida con el foco real del primario, por lo tanto, los rayos que llegan al secundario son reflejados en forma paralela y normal al plano x, pasando a traves del orificio que se encuentra en el espejo primario.

2.6 PROBLEMA A SOLUCIONAR

Aún cuando la estructura que sostiene a los espejos, es considerada como rígida, ésto no es cierto, ya que puede tener pequeños cambios con la variación de la temperatura; debe recordarse que la Sierra de San Pedro Mártir presenta cambios bruscos en su temperatura en el transcurso del día a la noche. Estas variaciones en la estructura pueden llegar a ser tales que los focos de ambos espejos *(real para el primario y virtual para secundario)* no coincidan, dando así aberraciones notorias en las imágenes observadas. Por otro lado, cuando se realizan exposiciones fotográficas, el cielo está variando de posición, y aún cuando se tiene un sistema que regula el movimiento del telescopio para seguir el movimiento del cielo, no siempre se puede mantener la posición exacta, que debe de guardar el telescopio, para obtener una exposición nitida. Ademas como ya se mencionó, debido a factores mecánicos *(vibraciones, las características de los materiales, etc)* es necesario cada año realizar una calibración de los espejos.

Con base en lo anteriormente expuesto, el sistema de óptica activa, tiene por objetivó, corregir el error de la posición de los espejos, debida a fallas en el paralaje de los mismos.

3 MOTORES DE PASOS

3.1 ESTRUCTURA Y FUNCIONAMIENTO

El sistema de optica activa, esta basado en motores de pasos de tipo híbrido con devanado bifilar. Un motor de pasos de tipo híbrido es aquel que tiene un magneto permanente en su rotor. En la figura 3.1 se muestra la estructura interna de un motor de pasos híbrido.

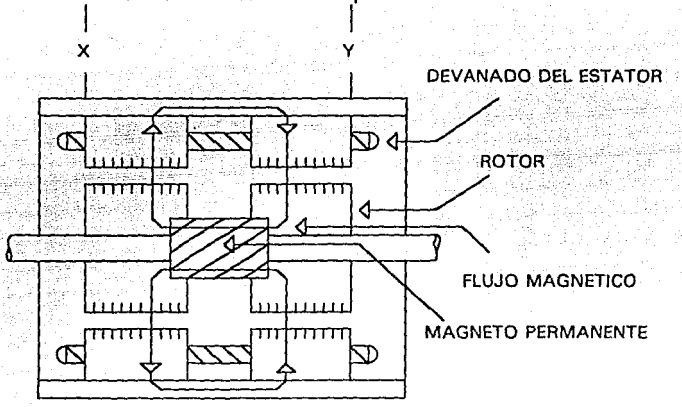


FIGURA 3.1 Estructura interna de un motor de pasos híbrido.

En la figura, se observa la ruta que sigue el flujo magnético a través de los polos del estator y el rotor.

La figura 3.2 , muestra un corte transversal de la figura 3.1, donde se observa la existencia de dos secciones *(X y Y)*.

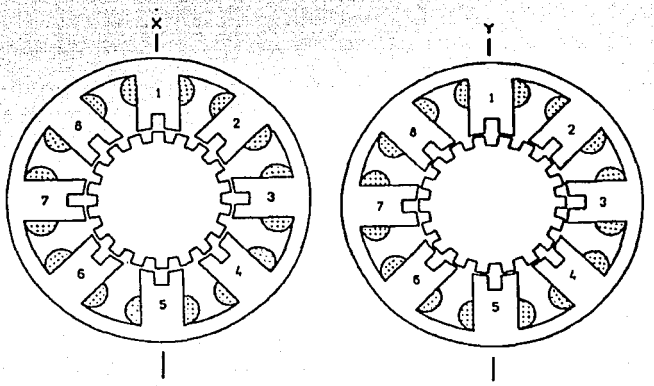


FIGURA 3.2 Corte transversal de un motor de pasos

Para el análisis se considera un motor típico con ocho polos en el estator. Cada polo cuenta con dos dientes (el número de dientes puede ser de dos a seis para motores típicos), dando un total de dieciseis dientes en el estator como se muestra en la figura 3.2.

Los polos en el estator, cuentan con devanados para activar o desactivar el flujo magnético en los polos del rotor. Los devanados se encuentran distribuidos en dos fases *(A y B)* de la siguiente manera :

a).- El fase *A* proporciona flujo a los polos 1, 3, 5 y 7.

b).- La fase *B* proporciona flujo a los polos 2, 4, 6, y 8.

Si la fase *A* es excitada con una corriente positiva, la dirección del campo magnético es directamente radial hacia afuera en los polos 3 y 7, pero radial hacia dentro en los polos 1 y 5. Un esquema similar se maneja en la fase *B*.

En la tabla 3.1, se muestra el comportamiento del flujo magnético de cada polo cuando alguna fase es excitada

TABLA 3.1

FASE	DIRECCION DE LA CORRIENTE	FLUJO MAGNETICO DE LOS POLOS	
A	POSITIVO	3 , 7	1 , 5
A	NEGATIVO	1 , 5	3 , 7
B	POSITIVO	4 , 8	2 , 6
B	NEGATIVO	2 , 6	4 , 8

Si la excitación es positiva en la fase *A*, el flujo de los polos 3 y 7 es radial hacia afuera, y radial hacia dentro en los polos 1 y 5, en la sección *X;* pero en la sección *Y*, el flujo es radial hacia dentro en los polos 3 y 7, y radial hacia afuera en los polos 1 y 5. Un esquema similar se maneja para la fase *B*.

El rotor al igual que el estator esta dentado; para nuestro caso el número de dientes en el rotor es de dieciocho. Si se observa la figura 3.2, los dientes de los polos del estator en la sección *X* están alineados con los dientes de los polos del estator de la sección *Y*. Por otro lado, los dientes en el rotor de la sección *X*, están defasados de sus correspondientes de la sección *Y* . Si el flujo magnético es concentrado en los polos por la excitación de una fase, entonces los dientes de el rotor tienden a alinearse con sus correspondientes dientes en el estator.

En la figura 3.2, al ser polarizada la fase *A*, en la sección *X* se observa un alineamiento entre los dientes de los polos 3 y 7 con sus correspondientes en el rotor. De igual forma en la sección *Y*, el alineamiento se realiza con los dientes de los polos 1 y 5 con sus correspondientes en el rotor.

La rotación continua del motor, es producida por una excitación secuencial de las fases *A* y *B*. Si la excitación en A es desactivada y es excitada la fase *B* con una corriente positiva ,el rotor deberá moverse un cierto ángulo para presentar una posición correcta, esto es, el alineamiento de los dientes del estator y rotor ocurre ahora en los polos 4 y 8 de la sección *X* y los polos 2 y 6 de la sección *Y*, produciendo un movimiento en el motor que se denominará paso.

En conclusión, para poder realizar un moviento continuo en el motor, deberá presentarse una secuencia de corrientes en las fases *A* y *B*, que pueden ser representadas como:

$$A+, B+, A-, B-, A+, B+,...$$

ésto para un sentido. Para que el movimiento sea en sentido contrario, la secuencia estará dada por :

$$A+, B-, A-, B+, A+, B-,...$$

El tamaño del paso, esta relacionado con el número de dientes del rotor *(P)*. Un ciclo completo de excitación, para un motor híbrido, consiste en cuatro estados y produce cuatro movimientos en el rotor, los cuales, son conocidos en conjunto como paso largo y están definidos como *tooth pitch:*

$$\text{TOOT PITCH} = \frac{360}{P} \quad [\text{grados}] \qquad (3.1)$$

Para el paso se tiene:

$$\text{PASO} = \frac{90}{P} \quad [\text{grados}] \qquad (3.2)$$

el motor de la figura 3.2 tiene diciocho dientes y por lo tanto tendrá pasos de 5 grados.

La teoría para motores híbridos es la misma sin importar el número de polos que existan en el rotor .

3.2 DRIVE DE MOTORES DE PASOS

El control de un sistema, basado en motores de pasos, es invariantemente de baja potencia, debido a su implementación con circuitos TTL (los circuitos *TTL* , manejan 5 V y 18 mA). Si estuvieramos considerando un

motor con las características nominales de 5 volts y 3 amperes, sería necesario una interfase que nos permitiera obtener la potencia requerida para manejar en forma aceptable las señales provenientes de los ciruitos *TTL*. A la interfase que proveé de esta potencia, se le conoce como *DRIVE* y está basada en transistores bipolares *(TBJ)*. Dichos circuitos trabajan por lo general en las regiones de corte y saturación, evitándose la región de amplificación por las pérdidas debidas al efecto Joule. Se hablará mas sobre el tema en el capítulo 3 donde se estudiará la electrónica del proyecto.

3.3 ERROR EN LA POSICION

Si una carga externa es aplicada al motor, cuando éste se encuentra excitado, el rotor tenderá a adoptar una posición dada, para lo cual el motor debe producir un torque suficiente, capaz de balancear el torque producido por la carga y mantener un equilibrio. El máximo torque que el motor puede producir, y por lo tanto la máxima carga que se puede aplicar bajo condiciones estáticas, es igual al máximo torque estático. Si la carga excede al máximo torque estático, el motor no puede mantener la carga en la posición demandada por la fase excitada.

El torque de la carga produce una posición de error estático, que puede ser deducida a partir de la característica de posición estática *TORQUE/ROTOR*. La figura 3.3, por ejemplo, muestra las características para un motor de 8 dientes en el rotor y un torque estático máximo de 1.2 Nm, a una corriente proporcional en la fase. Con un torque de la carga de 0.75 Nm, el motor debe moverse aproximadamente 4.83 grados de la posición del paso, hasta que desarrolle un torque que equilibre la carga.

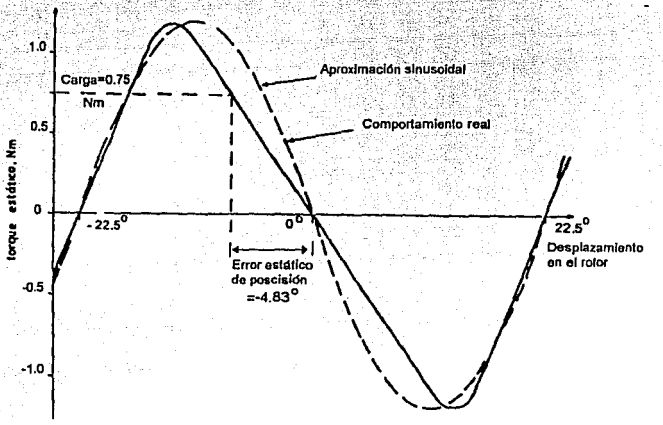


FIGURA 3.3 Relación torque/error estático

Una estimación del error de posición estático, puede ser obtenida porque, la característica de posición estático *TORQUE/ROTOR*, a una corriente apropiada de fase, se aproxima a una forma sinusoidal. Para un motor con P dientes de rotor, un torque estático máximo T_{PK} y un desplazamiento θ de la posición del paso, el torque desarrollado por el motor es aproximadamente:

$$T = -T_{PK} \times \text{sen } P\theta \quad [\text{N m}] \tag{3.3}$$

Cuando la carga es desplazada a la posición demandada por un ángulo θ_e , el torque de la carga (T_L) y del motor son iguales, por tanto:

$$T_L = T = -T_{PK} \times \sin P\theta_e \quad [\text{N m}] \tag{3.4}$$

despejando de la ecuación 3.4, el error de posición estático :

$$\theta_{em} = \frac{\sin^{-1}\left(\frac{-T_L}{T_{PK}}\right)}{P} \quad [\text{grados}] \tag{3.5}$$

Por lo tanto, el error de posición puede ser disminuido por el incremento en el torque estático máximo, es decir, utilizando un motor con mayor torque o por el uso de elementos mecánicos (estos serán discutidos más adelante). La ecuación 3.5 nos muestra que un número mayor en los dientes del estator, también modifica notablemente el error en la posición. Recordando que el paso del motor es inversamente proporcional al número de dientes del rotor, se puede ver que al tener un paso corto en un motor se reduce el error en la posición.

3.4 JUEGO DE ENGRANES

En algunas aplicaciones, un juego de engranes se encuentra interconectado entre la carga y el motor. Un sistema simple de engranes se muestra en la figura 3.4, en la cual, la relación que guardan los radios es de *1: N*, donde *N* es el número de revoluciones del motor que se debén producir para obtener una revolución en la carga.

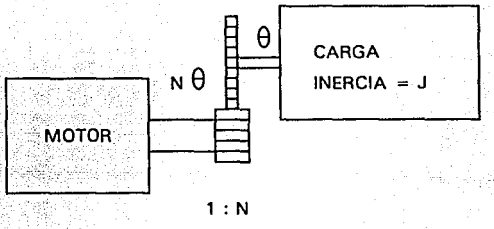


FIGURA 3.4 Sistema de engranes, para un motor de pasos

Si el torque de la carga es T_L, entonces, asumiendo que los efectos de la fricción del torque en el engrane son relativamente pequeños respecto al torque de carga, el torque en el motor es modificado por T_L / N. De definiendo a θ_{el} como el máximo error de posición en la carga, el motor operará con un error de posición estático de $\theta_{em} = N\theta_{el}$.

Por lo tanto, hay una considerable ventaja en el uso de engranes con gran radio que ligen la carga al motor, debido a que, los efectos del torque de la carga en el motor son reducidos, permitiendo que el error de posición decresca con respecto al caso donde el motor y carga están conectados directamente.

Si en la figura 3.4, la carga es conectada al motor por un juego de engranes con una relación de *3:1* y considerando los datos de la figura 3.3, el torque de carga efectiva en el motor es:

$$T_L = \frac{0.75}{3} = 0.25 \quad [\text{Nm}]$$

de igual forma se puede obtener el valor de error de posición en el motor, ésto es:

$$\theta_{em} = \frac{\sin^{-1}\left(\frac{-0.25}{1.2}\right)}{8} = -1.503 \quad [\text{GRADOS}]$$

así, el máximo error de posición en la carga es :

$$\theta_{el} = \frac{\theta_{em}}{N} = \frac{-1.503}{3} = -0.51 \quad [\text{GRADOS}]$$

lo cual nos reduce a un 10% del error obtenido en el caso de conectar directamente la carga al motor, .

Si durante la operación dinámica la carga es acelerada, entonces el torque aplicado es proporcional a la aceleración angular y la inercia de la carga está dada como:

$$T_L = J_L \frac{d^2\theta}{dt^2} \quad [\text{N m}] \qquad (3.6)$$

por lo tanto, el torque requerido para el motor es:

$$T_m = \frac{T_L}{N} = \frac{J_L \frac{d^2\theta_m}{dt^2}}{N^2} \quad [N\,m] \qquad (3.7)$$

donde

$$\theta_m = N\theta_L \quad [\text{grados}] \qquad (3.8)$$

De la ecuación 3.7, la inercia efectiva del motor es J / N^2; la inercia de la carga es reducida por el cuadrado del radio del engrane, por lo que un engrane de gran radio, permite una aceleración más rápida. Sin embargo, se debe mencionar que un paso del motor produce un movimiento en la carga, que es solamente una fracción *1/N* del motor de pasos. Si la carga es movida una distancia dada, utilizando un engrane de gran tamaño, se requiere que el motor mueva un número grande de pasos, por lo que el motor deberá realizar una gran cantidad de pasos en un tiempo razonable, para que la carga sea movida en forma aceptable. En forma inversa, con un radio pequeño en el engrane, la inercia de carga efectiva es alta y el motor es acelerado más lentamente, pero ésto se compensa la poca cantidad de pasos . Este comportamiento se puede visualizar en la tabla 3.2

TABLA 3.2

ALTO GRADO DE EL ENGRANE	BAJO RADIO DE EL ENGRANE
BAJA INERCIA REFLEJADA	ALTA INERCIA REFLEJADA
RAPIDA ACELERACION	LENTA ACELERACION
ALTA VELOCIDAD EN EL MOTOR	BAJA VELOCIDAD EN EL MOTOR

El uso de un engrane de gran radio, es usual cuando el movimiento de la carga envuelve períodos substanciales de aceleración y desaceleración. Un engrane de radio pequeño es comunmente usado cuando la capacidad de velocidad del motor es una restricción.

3.5 TORNILLO SIN FIN

Si una carga se sujeta a un tornillo sin fin, considerando una fuerza *F* y asumiendo que los efectos de fricción en el tornillo son despreciables, el trabajo realizado para mover la carga una distancia *x* es *Fx*. Lo anterior se puede visualizar en la figura 3.5.

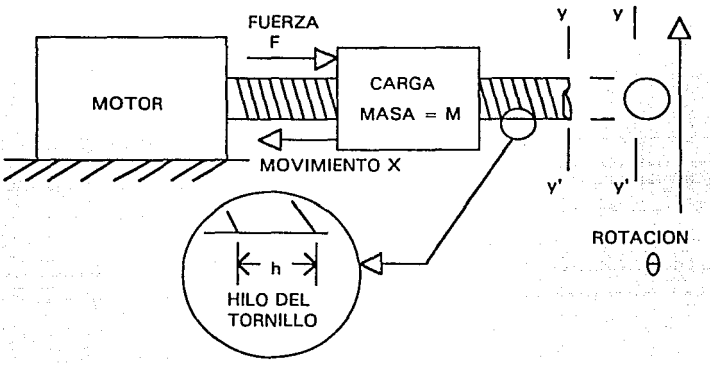


FIGURA 3.5 Tornillo sin fin en un motor de pasos

La relación entre el torque necesario y la posición angular del motor, está dada por la ecuación 3.9 :

$$F \times x = T_L \times \theta \qquad (3.9)$$

donde θ y x están relacionadas por la ecuación

$$\frac{\theta}{2\Pi} = \frac{x}{h} \qquad (3.10)$$

sustituyendo esta ecuación 3.10 en 3.9 y despejando

$$T_L = \frac{F \times x}{\theta} = \frac{F \times h}{2\Pi} \quad [N\,m] \qquad (3.11)$$

con base en estas relaciones, podemos calcular el error de posición del tornillo, cuando la carga esta sujeta a una fuerza *(se puede considerar al peso como la fuerza que afecta al sistema, cuando el tornillo está dispuesto en una posición vertical)*. Partiendo de calcular el torque efectivo de la carga, se puede deducir el error de posición del motor; con este dato, utilizando la ecuación 3.10, podemos obtener finalmente el error de posición en el tornillo.

Si la carga es acelerada, la fuerza requerida es proporcional a la masa de la carga por la aceleración:

$$F = M\frac{d^2x}{dt^2} \quad [N] \tag{3.12}$$

sustituyendo este valor en la ecuación 3.11:

$$T_L = \frac{F \times h}{2\Pi} = \frac{M \times h\frac{d^2x}{dt^2}}{2\Pi} \quad [N\,m] \tag{3.13}$$

si ahora sustituimos x en terminos de θ de la ecuación 3.10:

$$T_L = M\left(\frac{h}{2\Pi}\right)^2\left(\frac{d^2\theta}{dt^2}\right) \quad [N\,m] \tag{3.14}$$

donde la inercia de la carga en el motor es:

$$J = M\left(\frac{h}{2\Pi}\right)^2 \tag{3.15}$$

Desde el punto de vista de una posición estática, el reducir el *hilo en el tornillo* reduce el torque de la carga en el motor. Para un hilo de tornillo corto, la inercia efectiva de la carga es reducida y el motor puede acelerarse rápidamente, pero ésto nos sujeta a un incremento lineal dependiente de la velocidad para poder generar cada paso del motor.

4 ELECTRONICA

4.1 CARACTERISTICAS

El sistema de óptica activa, al estar basados en motores de pasos, necesita de una electrónica, que cumpla con las siguientes características:

a).- Permitir el control a partir de la computadora, vía puerto paralelo.
b).- Manejar seis motores de pasos *(ésto incluye las etapas de potencia para cada motor).*
c).- La existencia de una etapa que permita aislar a la computadora, de la electrónica externa.
e).- Manejar seis sensores de posicion *(cada uno corresponde a un motor).*
f).- Que ocupe el menor espacio posible.
g).- Que sea barato.
h).- Su construcción deberá ser con elementos fáciles de conseguir en el mercado.

En la figura 4.1, se muestra en diagrama de bloques la electrónica del sistema.

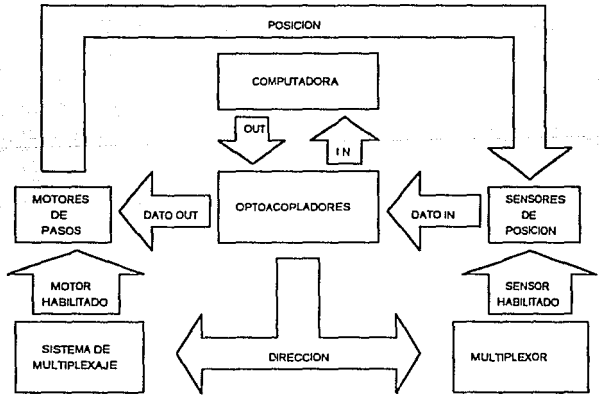


FIGURA 4.1 Diagrama a bloques de la electrónica

En la figura 4.1 se puede apreciar que, dependiendo de las direcciones proporcionadas por la computadora, un motor y un sensor son habilitados; además el motor recibe el código adecuado para realizar un movimento.

El sensor al ser habilitado, presenta a su salida, el dato pertinente de la posición referente al motor que esté en funcionamiento. Los optoacopladores, aislan a la computadora de la electrónica externa .

A continuación, se estudiarán con mayor detalle los elementos que intervienen en la parte electrónica del proyecto.

4.2 INTERFASE DE ACOPLAMIENTO

Una de los renglones que todo sistema debe de guardar, es el referente a la protección del mismo; en nuestro caso, al utilizar la computadora como controlador, es de vital importancia protegerla contra posibles transitorios que puedan dañarla en forma parcial o total.

Con base en lo anterior, la interfase de acoplamiento tiene la tarea de aislar al controlador *(computadora)*, de parte electrónica ; ésto se realizó utilizando optoacopladores. En la figura 4.2, se muestra en forma esquemática, el funcionamiento de estos.

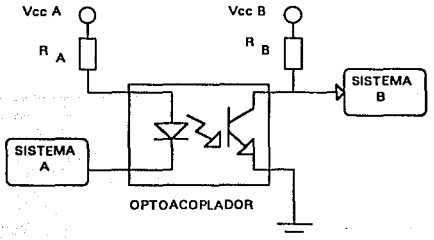


FIGURA 4.2 Esquema de un optoacoplador

Cuando el led transmisor es polarizado, emite una luz proporcional a la corriente de la base del fototransistor receptor. En el proyecto se utilizó al fototransistor en sus estados de corte y saturación.

4.3 SISTEMA DE MULTIPLEXAJE

Cada motor necesita de cuatro bits para que pueda realizar su funcionamiento en forma correcta; si además el sistema cuenta con seis motores, ésto nos da un total de 24 bits o 3 bytes para el control del servomecanismo. Esto es una desventaja, debido a que la computadora nos proporciona una palabra de ocho bits *(o un byte)* como líneas de salida utilizando el puerto paralelo. Lo anterior es solucionado al utilizar un sistema de multiplexaje externo.

Un sistema de multiplexaje, nos permite manejar, de un conjunto de n grupos de líneas de información, solo un grupo del conjunto, a partir de una dirección dada. Si consideramos el código correspondiente al movimiento de los seis motores como conjunto de n grupos de líneas, de los cuales solo nos interesa un solo grupo a la vez, estaremos hablando de un sistema de multiplexaje. Dado que la dirección del motor es la misma para el sensor de posición, denominaremos como puerto, al conjunto motor-sensor. No se debe de confundir al puerto paralelo (que es el puerto de la computadora), con los puertos externos (motor-sensor). El sistema emplea de la siguiente forma los bits de salida procedentes del puerto:

a).- Cuatro bits nos proporcionarán el código correspondiente a un movimiento en el motor.

b).-Tres bits nos proporcionarán la dirección del motor, se debe observarque estos bits nos dan ocho posibles combinaciones, por consiguiente ocho posibles direcciones.

c).- El octavo bit no es utilizado.

El sistema de multiplexaje ésta constituido por un decodificador que habilita el puerto pertinente. La habilitación depende de la dirección que envía el puerto paralelo de la computadora.

4.4 DECODIFICADOR DE PUERTO.

Un código binario de n bits es capaz de representar hasta 2^n elementos distintos de información codificada. Un codificador es un circuito combinacional que convierte la información binaria de n líneas de entrada a un máximo de 2^n líneas de salida. La figura 4.3 muestra un decodificador $n \times 2^n$.

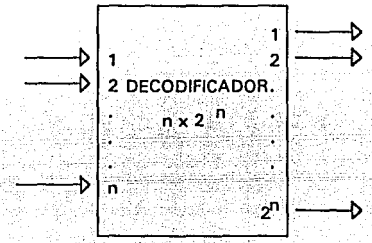


FIGURA 4.3

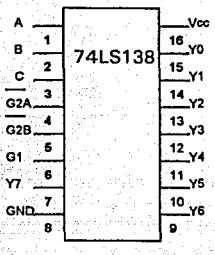
En un decodificador de línea, dependiendo de la combinación en las líneas de entrada, se habilita una y solo una línea de salida. Lo anterior se puede ilustrar mediante una tabla 4.1 de un decodificador de 2 x 4

TABLA 4.1

DEC 2 X 4		
A	B	L
0	0	1
0	1	2
1	0	3
1	1	4

En el sistema se utilizó un decodificador *74LS138*. El circuito es un decodificador en línea, de tecnología TTL low power Schottky. El circuito cuenta con tres entradas binarias de selección *(A, B y C)*. Si el circuito está habilitado, estas entradas determinan cuál de las ocho salidas, normalmente altas, es mandada a un estado bajo.

Cuenta con dos entradas de habilitación *(G2A y G2B)* activas bajas y una entrada de hablitación *(G1)* activa alta, que son utilizadas para la conexión en cascada de decodificadores. Se muestra a continuación el circuito y su tabla de verdad.



INPUTS					OUTPUTS							
ENABLE		SELECT										
G1	$\overline{G2}$*	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

* $\overline{G2} = G2A + G2B$

FIGURA 4.4 Circuito 74LS138

4.5 PUERTOS

Como ya se mencionó, la computadora sólo nos proporciona una palabra de ocho bits *(un byte)* en su puerto . Si tenemos que manejar 6 motores independientes, es necesario la creación de puertos. La operación de los puertos es independiente entre sí, es decir, solo un puerto podrá operar a la vez en el sistema. Por tal razón, los puertos deberán recibir una señal de habilitación, así como el código correspondiente a un movimiento del motor. La señal de habilitación es proporcionada por el decodificador de puerto, y el código del movimiento por un bus de datos, proporcionado por un buffer. Para la implementación de los puertos de salida, se utilizaron flip-flop's que son circuitos de tipo secuencial.

Antes de continuar, es importante hacer una diferencia entre los circuitos denominados combinacionales *(como es el caso del decodificador y del multiplexor)* y los circuitos secuenciales *(como los flip-flop's)*.

Un circuito combinacional, las salidas en cualquier momento dependen por completo de las entradas presentes en ese instante, no presentan elementos de memoria.

Para explicar un circuito secuencial, nos auxiliaremos de la figura 4.5.

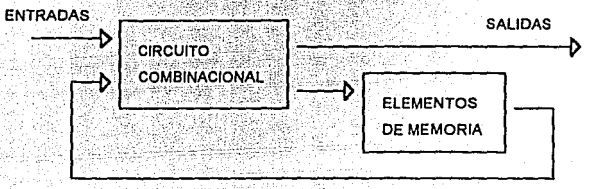


FIGURA 4.5 Diagrama a bloques de un circuito secuencial

El diagrama de la figura, consta de un circuito combinacional al que se conectan elementos de memoria para formar una trayectoria de retroalimentación. Los elementos de memoria son dispositivos capaces de almacenar información binaria. La información binaria almacenada en los elementos de memoria en cualquier momento dado definen el estado del circuito secuencial.

El circuito secuencial recibe información binaria de entradas externas. Estas entradas, junto con el estado presente de los elementos de memoria, determinan el valor binario en las terminales de salida.

El diagrama de bloques muestra que las salidas externas en un circuito secuencial son funciones no solo de las entradas externas sino también del estado presente de los elementos de memoria. El siguiente estado de los elementos de memoria también es una función de las entradas externas y del estado presente. Por tanto, un circuito secuencial está especificado por una secuencia de tiempo de entradas salidas y estados internos.

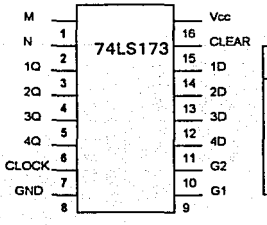
Hay dos tipos principales de circuitos secuenciales. Su clasificación depende del temporizado de señales.

CIRCUITO SECUENCIAL ASINCRONO. El comportamiento de un circuito secuencial asíncrono, depende del orden en el cuál cambian sus señales de entrada y puede afectarse en cualquier instante de tiempo. Los elementos de memoria ,que por lo común se utilizan en los circuitos secuenciales asíncronos, son dispositivos de retardo de tiempo, por tal razón en ocasiones son inestables.

CIRCUITO SECUENCIAL SINCRONO. Un circuito síncrono es un sistema cuyo comportamiento puede definirse por el conocimiento de sus señales en instantes discretos de tiempo. Los sistemas lógicos secuenciales síncronos usan amplitudes fijas, como niveles de voltajes para las señales binarias. La sincronización se logra por un dispositivo temporizador. Los elementos de memoria que se usan en los circuitos secuenciales, se llaman flip-flop's.

Un circuito flip-flop puede mantener un estado binario en forma indefinida (en tanto se suministre potencia en el circuito) hasta que recibe la dirección de una señal de entrada para cambiar de estado. La diferencia principal entre los diversos tipos de flip-flop's, radica en el número de entradas que poseen y en la manera en la cual, las entradas afectan al estado binario. También determinan las condiciones para cambiar el estado en los elementos de memoria.

En el diseño del sistema se utilizaron flip-flop D. el circuito utilizado fue el circuito *74LS173*. A continuación se muestra el circuito así como su la tabla de verdad en la figura 4.6.



CLEAR	CLOCK	G1	G2	D	Q
H	X	X	X	X	L
L	L	X	X	X	Qo
L	^	H	X	X	Qo
L	^	X	H	X	Qo
L	^	L	L	L	L
L	^	L	L	H	H

FIGURA 4.6 Circuito 74LS173

El circuito *74LS173*, contiene cuatro flip-flop's que operan en forma

síncrona dependiendo de una señal de reloj común. El circuito cuenta con salida tres estados, que se encontrará presente si las terminales *M* ó *N* (o ambas) se encuentran conectadas en un estado alto. Estas señales en el sistema se encuentran conectadas en un nivel alto.

Existe dos señales de habilitación para el circuito *(G1 y G2)*, las cuales deberán estar en un estado bajo, para permitir el paso de la información que se encuentra a la entrada del circuito. Si una o ambas señales están en un nivel alto, la información a la salida es retenida en forma indefinida. Esta característica es aprovechada en el sistema para la habilitación del puerto, conectándose cada una de las líneas de salida del decodificador a entradas de habilitación del flip-flop.

La entrada de *clear* en el circuito debe conectarse a un nivel bajo, de otra forma el circuito en sus salidas presentarán estados bajos.

Para la señal de sincronización se utilizó un circuito astable conformado por un temporizador *LM555*. La frecuencia a la que debe de operar el reloj deberá de ser mayor que la velocidad en que se transmita la información de la computadoara a los puertos, sin embargo ésto se puede controlar por medio de software, debiéndose de recordar que la velocidad de respuesta en los motores está limitada por el torque de la carga. Con base en lo anterior, se diseñó un reloj de una frecuencia de 10 kHz, el cual es lo suficientemente rápido, para cubrir las necesidades del sistema.

4.6 DRIVE

Una corriente bidireccional puede fluir en los devanados unifilares de un motor híbrido convencional, pudiendo producir un campo bidireccional en los polos del estator. El mismo resultado se obtiene con un devanado bifilar. Un devanado bifilar consta de dos devanados colocados en sentido opuesto en el polo del estator, ésto se ilustra para un polo en la figura 4.7.

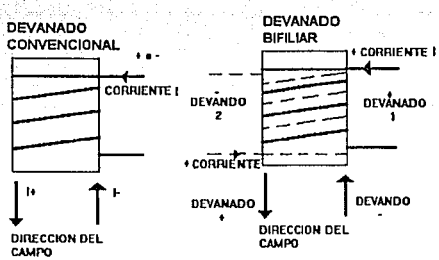


FIGURA 4.7

Dependiendo de la dirección del campo, uno de los devanados es excitado por una corriente unidireccional . En la figura 4.7 el campo producido por una corriente positiva, en el arreglo convencional, se encuentra disponible en el devando bifilar. El efecto de una corriente negativa en el devanado convencional, está dispuesto en el devanado bifilar, por la excitación positiva en su devanado negativo .Los dos devanados bifilares de una fase pueden ser excitados por separado mediante un circuito de drive unipolar.

4.7 DRIVE UNIPOLAR

Un circuito de drive unipolar se muestra en la figura 4.8. El drive es excitado por el control de baja potencia.

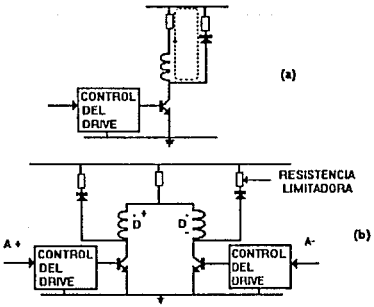


FIGURA 4.8

El principio del drive es simple; la señal de control coloca al transistor en los estados de operación de corte o de saturación. El devanado es excitado, cuando el transistor es saturado por una corriente suficientemente alta en la base. Bajo estas condiciones el voltaje de polarización, en su totalidad (idealmente), es aplicado al devanado y a la resistencia limitadora, puesto que el voltaje colector-emisor del transistor es relativamente pequeño (típicamente de entre 0.1 y 0.2 volts).

El voltaje de polarización *(Vs)* es escogido para que produzca una corriente (I) que será aplicada a la resistencia total de la fase, la cual es igual a la suma de las impedancias del devanado *(r)* y la resistencia limitadora *(R)*; el voltaje de polarización está dado por:

$$V_s = I(r + R) \qquad (4.1)$$

Una consecuencia de la inductancia del devanado,es la corriente de fase, que no puede ser switcheada para apagarse instantáneamente. Si el switcheo en la base del transistor es repentino, aparecerá un gran voltaje inducido entre el colector y el emisor, pudiendo causar *(en corto o largo plazo)*, un daño permanente en el circuito del Drive.

La posibilidad anteriormente mencionada queda salvada si existe una ruta alterna, que puede ser utilizada para descargar en forma efectiva la corriente del devanado desactivado. El circuito utilizado para crear la ruta de descarga, es llamado frewheeling, cuyo funcionamiento se explicará a continuación.

Cuando el transistor es apagado, la corriente puede continuar fluyendo a través de la ruta marcada por un diodo y una resistencia *(opcional)* en serie conectados entre el colector y la fuente de polarización. Si la corriente es establecida , la razón del valor máximo de voltaje (*Vce max*) a través del transistor switcheado, ocurre en el instante siguiente al que el transistor es abierto. La corriente *I* no inicia su decrecimiento y fluye a través de la ruta marcada por el diodo y la resistencia R_f, el máximo valor de voltaje colector-emisor, es definido entonces por la ecuación 4.2

$$V_{cc_{max}} = V_s + R_f \times I \qquad (4.2)$$

La corriente de la fase por lo tanto decae a través del circuito freewheeling y la energía magnética almacenada en el devanado es disipada en la resistencia del circuito.

Para un motor híbrido bifilar de dos fases, cada fase puede ser excitada por separado por un circuito drive unipolar, sin embargo, una alternativa es la *REJA O SHARE* que consiste en colocar una resistencia limitadora entre los dos devanados unifilares como se muestra en la figura 4.8b. Hay ahora solamente dos pares de transistores y diodos en este circuito completo de drive

4.8 SENSOR DE POSICION

En la mayoría de los casos de robótica es de importancia fundamental conocer la posición del sistema en el instante que se requiera. El sensor se puede dividir en dos partes fundamentales: el transductor y el convertidor *(este último en algunos casos no es necesario)*.

4.8.1 TRANSDUCTOR DE POSICION.

Existen una gran variedad de elementos transductores disponibles para detectar posición; en nuestro caso examinaremos el transductor de tipo resistivo. Los potenciómetros son dispositivos analógicos cuya tensión de salida es proporcional a la posición de un cursor. La figura 4.9 muestra un potenciómetro común.

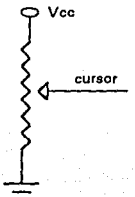


FIGURA 4.9 Potenciómetro

Si se aplica una tensión a través de los elementos resistivos . La tensión en el cursor es proporcional a la relación entre la resistencia de un lado del cursor y la resistencia total del elemento. En esencia, el potenciómetro actúa como una red de condición de tensión (divisor de tensión). Es decir la energía que cruza al elemento resistivo es dividida en dos partes por el cursor. La medición de esta energía, proporciona la posición del cursor. La función del potenciómetro se puede representar mediante la siguiente función:

$$V_0(t) = K_P \times \theta(t) \qquad (4.3)$$

Donde $V_0(t)$ es la tensión de salida, K_P es la constante de tensión del potenciómetro en voltios por radianes (o voltios por centímetro en el caso de un potenciómetro lineal). $\theta(t)$ es la posición del potenciómetro en radianes . Puesto que un potenciómetro requiere una tensión de excitación, con objeto de calcular $V_0(t)$ se puede utilizar la fórmula:

$$V_o = V_{ex} \frac{\theta_{act}}{\theta_{tot}} \qquad (4.4)$$

Donde V_{ex} es la tensión de exitación, θ_{tot} es el avance total disponible del cursor y θ_{act} es la posición actual del cursor.

Los rangos de desplazamiento angular en los transductores potenciométricos están comprendidos en general en un rango de 10° a 350° . En el área de instrumentación se cuenta con potenciómetros especiales denominados multivuelta, que requieren más de una vuelta para poder alcanzar su escala total.

Como transductor se utilizará un potenciómetro de precisión; su característica principal radica en su linealidad, la cual es de 0.25%, además de presentar 10 vueltas por escala completa.

4.8.2 CONVERTIDOR VOLTAJE/FRECUENCIA.

Para un sensor de posición convencional, el transductor potenciométrico es suficiente. Sin embargo para que la información pueda ser leída por la computadora es necesario un elemento que pueda convertir la información analógica a información digital. Para un convertidor A/D convencional se necesita por lo menos de 8 bits de entrada en un puerto para que la información sea leída. Esta es una limitante, ya que en el puerto paralelo, el número de bits de entrada, está restringido a un número menor. Si además se considera el rango de lecturas que nos podría proporcionar una palabra de 8 bits *(256 posibilidades, una vuelta de un potenciómetro corresponde a 360)* se observará que esta posibilidad es poco viable. Otro tipo de convertidor A/D es el convertidor voltaje frecuencia *(V.C.O)*.

El sensor de posición *(S.P)* está basado en un V.C.O. Este elemento transforma un valor de voltaje en un tren de pulsos con una frecuencia definida. La ventaja que presenta, radica en que sólo se necesita una sola línea (un pin en el puerto de la compuadora) para transmitir la información. Su desventaja es que se necesita de un programa capaz de procesar correctamente la frecuencia que éste nos entrega. El programa para la lectura de frecuencia fue desarrollado en el Centro de Instrumentos, en el laboratorio de electrónica. La frecuencia que el S.P. entrega, es transmitida a la computadora vía puerto paralelo, para después ser procesada por el software que controlará al sistema de multiplexaje y a los motores de pasos.

4.8.3 OPERACION A DIAGRAMA DE BLOQUES DE EL LM131

Se utilizó el circuito LM131 para la conversión voltaje/frecuencia. Dentro de sus características mas importantes se tiene:

- Escala de frecuencia completa de 1 a 100 kHz
- Baja disipación de potencia del orden de 15 mW a 5V
- Maneja tensiones en un rango de 4 a 40 V
- Pulsos de salida compatibles con TTL, CMOS, etc.
- Linealidad máxima de 0.01 %

- Bajo costo.

La figura 4.10 muestra el diagrama a bloques del LM131, que consta de un comparador de entrada, un one shot timer y de un switch de corriente. La teoría de operación se describe a continuación.

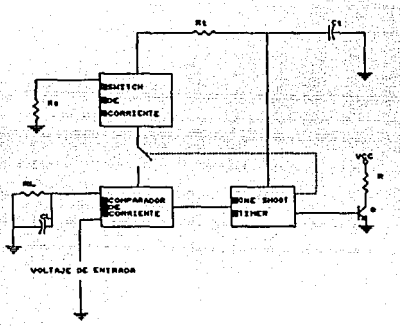


FIGURA 4.10 Diagrama de bloques del V.C.O.

Se compara el voltaje de entrada V_1 contra un voltaje de referencia V_x. Si el voltaje V_1 es mayor que el voltaje de referencia, el comparador dispara el one shot. El one shot mantiene encendido tanto al transistor de salida como al switch de corriente por un periodo de $1.1\,R_tC_t$. Durante este periodo la corriente fluye a la salida del switch de corriente, aportando una cantidad de carga $Q = i \times t$ dentro del capacitor C_L, éste se carga hasta un voltaje mayor al del voltaje V_1. Al término del periodo la corriente apagará al switch, dando así un reset al one shot. Ahora no hay corriente que fluya a través de la salida del switch de corriente, y el capacitor se puede descargar gradualmente a través de la resistencia R_L, hasta que el valor de V_x sea igual al nivel de V_1, entonces el comparador vuelve a disparar el one shot, iniciandose así otro ciclo de trabajo del convertidor. La corriente que fluye dentro del capacitor C_L es exactamente

$$i_{AVE} = i\,(1.1\,R_1C_1)\,f \qquad (4.5)$$

y la corriente que fluye hacia afuera del capacitor C_L es exactamente

$$\frac{V_x}{R_L} \cong \frac{V_{in}}{R_L} \qquad (4.6)$$

Si el voltaje de entrada es duplicado, la frecuencia de salida también es duplicada, manteniéndose así un balance entre la entrada y la salida. El convertidor puede proporcionar una relación lineal entre el voltaje y la frecuencia, ésto siempre que no se excedan las limitaciones del circuito.

El one shot timer, está compuesto por un flip-flop RS y un comparador de tiempo conectado a la red R_1C_1. Cuando el comparador de entrada detecta un voltaje en el pin 7 mayor que en el pin 6, el flip-flop pasa a estado alto, encendiendo de esta manera al switch de corriente y el transistor de salida es colocado en un estado de saturación . Cuando el voltaje en el pin 5 sube arriba de 2/3 Vcc el comparador de tiempo causa un reset en flip-flop. El reset en el flip-flop provoca que el transistor pase a un estado de corte y el switch de corriente se apague.

De lo anterior, se observa que, el máximo voltaje en el comparador de entrada debe de ser de 2/3 del voltaje de alimentación. Si el voltaje sobrepasa este valor, la operación del integrado sale de control, es decir, su característica de linealidad se pierde.

4.8.4 CIRCUITO UTILIZADO.

La figura 4.11 muestra el circuito utilizado en el proyecto.

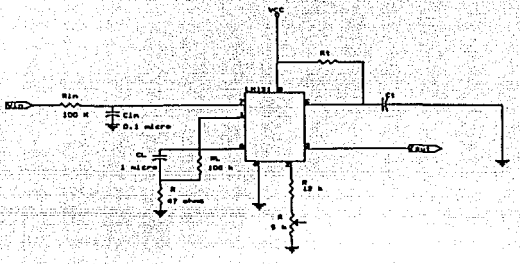


FIGURA 4.11 Convertido voltaje/frecuencia

El resistor R_{IN}= 100k, se encuentra conectado en el pin 7, de tal forma que la corriente inversa del pin 7, pueda cancelar la corriente de inversa del pin 6 y ayude a proveer una frecuencia de offset mínima.

El capacitor C_{IN} es conectado al pin 7 y a tierra, formando junto con el resistor R_{IN} un filtro para el voltaje de entrada. Los valores para este capacitor varían entre 0.01 μf y 0.1 μf. Para casos donde se requiera un mayor filtrado es preferible utilizar un capacitor de 1 μf.

Un resistor de 47 ohms en serie con el capacitor C_L de 1μf, es conectado para proporcionar un efecto de histéresis, que auxilie al comparador para proporcionar una excelente linealidad (tipicamente de 0.03%).

La resistencia R_S, conectada en el pin 2 , está integrada de dos resistencias: una fija de 12 kΩ mas una resistencia variable de 5kΩ para ajuste. El ajuste es referente a la tolerancia en la ganancia del circuito integrado y a la tolerancia en los valores del los resistores R_t , R_L y el capacitor C_t. La frecuencia de salida esta dada por el fabricante como:

$$f_{out} = \left(\frac{V_{IN}}{2.09\ V}\right)\left(\frac{R_S}{R_L}\right)\left(\frac{1}{R_tC_t}\right) \qquad (4.7)$$

4.8.5 MULTIPLEXOR ANALOGICO

Si observamos al sistema como un sistema realimentado, la señal de realimentación es proporcionada por el sensor de posición, la cual es particular de cada motor, por esta razón es necesario multiplexar esta señal. Esto se puede solucionar de dos formas:

a).- Creando seis sensores de posición y utilizando la dirección de cada motor para manejar un multiplexor digital, de tal manera que, de las señales provenientes de los sensores, solo una sea transferida a un "pin" en el puerto de entrada y de esta manera pueda ser procesada por la computadora.

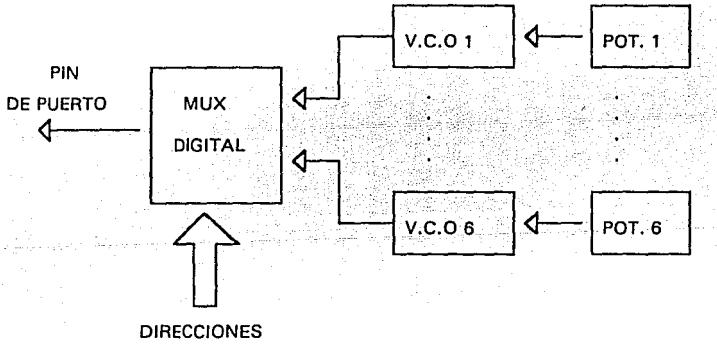


FIGURA 4.12 Diseño con multiplexor digital

b).- Utilizar un solo sensor de posición, el cual pueda controlar los seis transductores (potenciómetros) por medio de un multiplexor analógico controlado por la dirección de cada motor

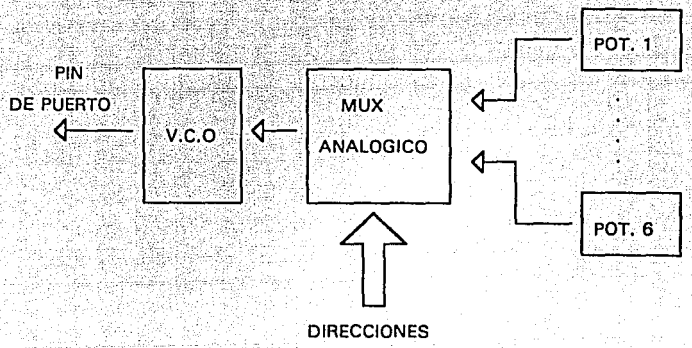
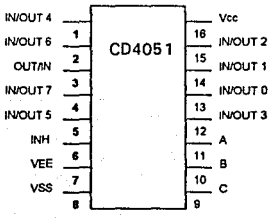


FIGURA 4.13 Diseño con multiplexor analógico

La primera opción implica un mayor gasto y mayor volumen. En la segunda opción, se ahorra espacio y dinero. Por lo anteriormente mencionado, la opción mas viable es la segunda.

El CD4051 es un circuito multiplexor/demultiplexor analógico y controlado por direccionamiento digital, fabricado con tecnología CMOS. El circuito puede manejar en forma típica, tensiones entre los 3 y 16 volts. A continuación se muestra en la figura 4.14 el circuito y su tabla de verdad.



INHIBIT	C	B	A	OUT
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	*	*	*	NONE

FIGURA 4.14 Cicuito CD4051, multiplexor analógico

Las líneas *A,B,C* manejan el direccionamiento del circuito y la linea INH es una línea de habilitación de entrada.

Debido a que el circuito en realidad está compuesto por switches analógicos, dependiendo de donde se tomen las entradas o salidas, el circuito funcionará ya sea como multiplexor o demultiplexor, es decir, que presenta un comportamiento bidireccional en todo momento,

Funcionando como multiplexor, las líneas de direccionamiento del circuito selecionan entre las ocho entradas (0 - 7) cuál deberá pasar a la línea de salida.

Con los elementos anteriormente estudiados es posible ahora mostrar el módulo completo del sensor de posición.

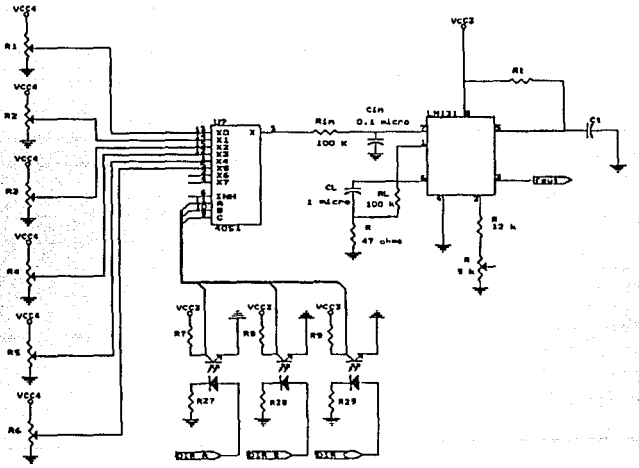


FIGURA 4.15 Sensor de posición

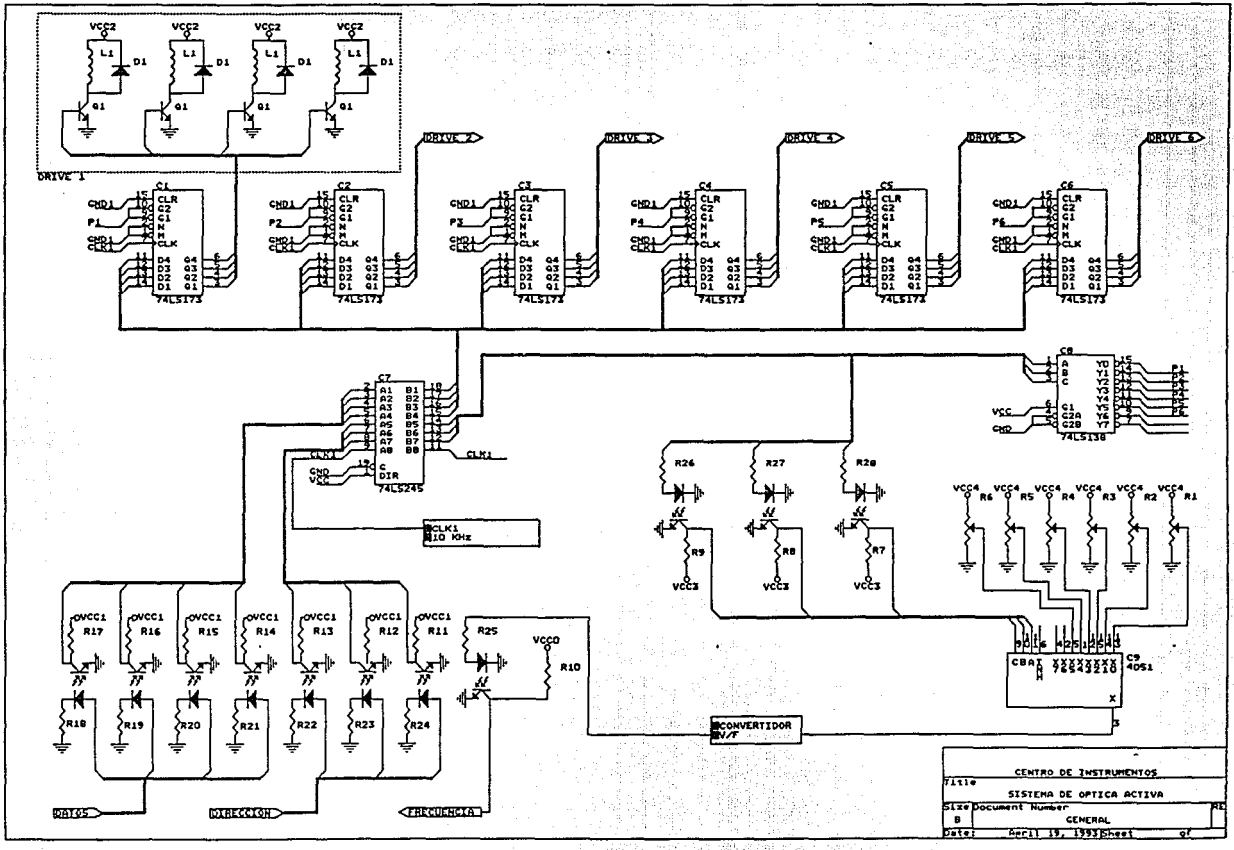
En la figura 4.15 se observan tres fuentes , Vcc_0, Vcc_3, Vcc_4 cada una corresponde a 5, 12 y 8 volts. La fuente Vcc_0 corresponde a de la computadora.

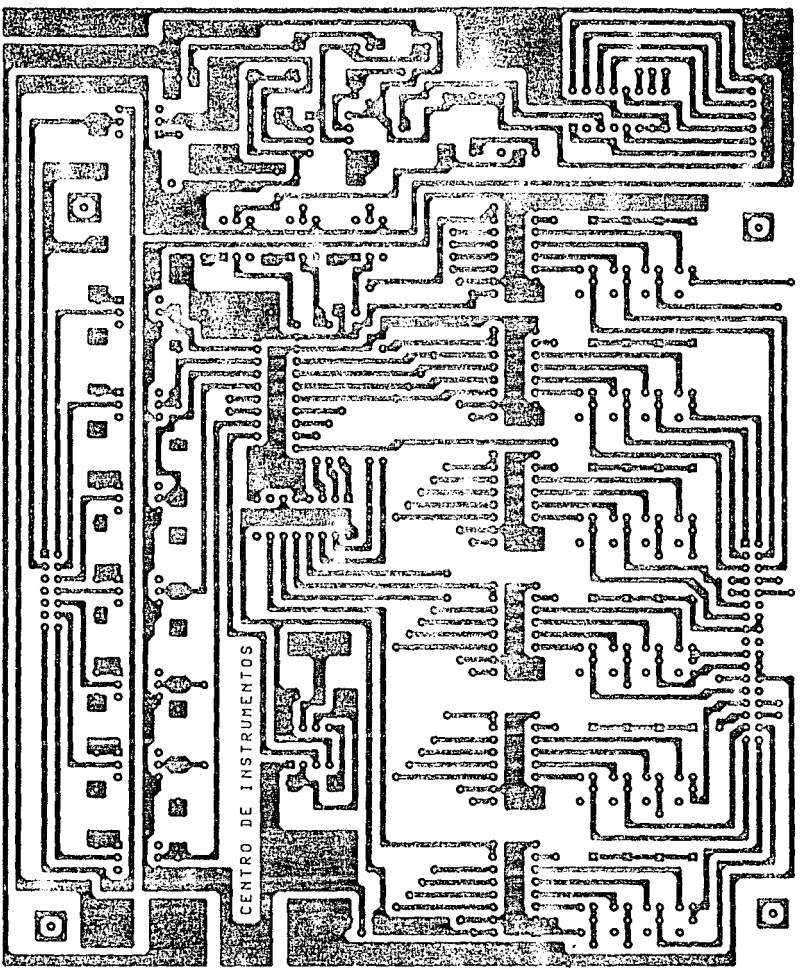
Como se había mencionado, el máximo rango de voltaje de entrada debe ser 2/3 del voltaje de alimentación, por lo tanto, considerando que podemos disminuir el voltaje de Vcc_3 con un regulador de 8 volts (no existen reguladores de 9 volts), el rango de medición sera de 0 a 8 volts.

Cabe mencionar que el módulo de el sensor es independiente del módulo de multiplexaje, con el fin de evitar transitorios debidos al funcionamiento en los motores. Con base en lo anterior, se utilizaron optoacopladores que aíslan a ambos sistemas.

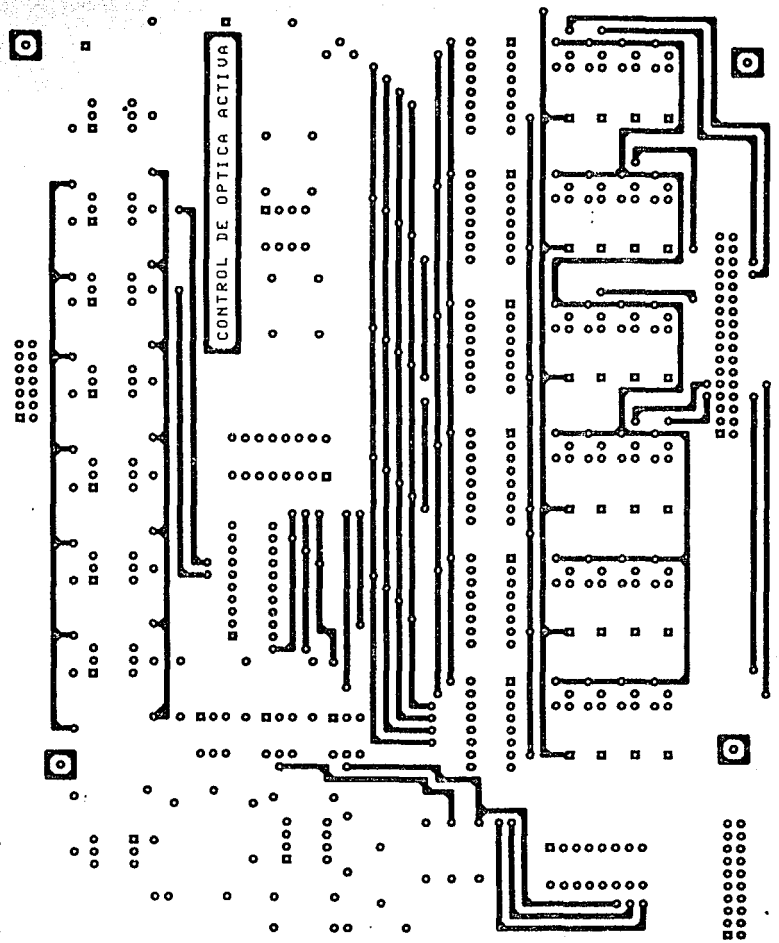
4.9 CIRCUITOS Y LISTAS DE PARTES

Con los elementos anteriormente estudiados, es posible mostrar ahora el circuito completo, así como, los circuitos impresos y la lista de partes, que comprende a la parte electrónica del sistema.

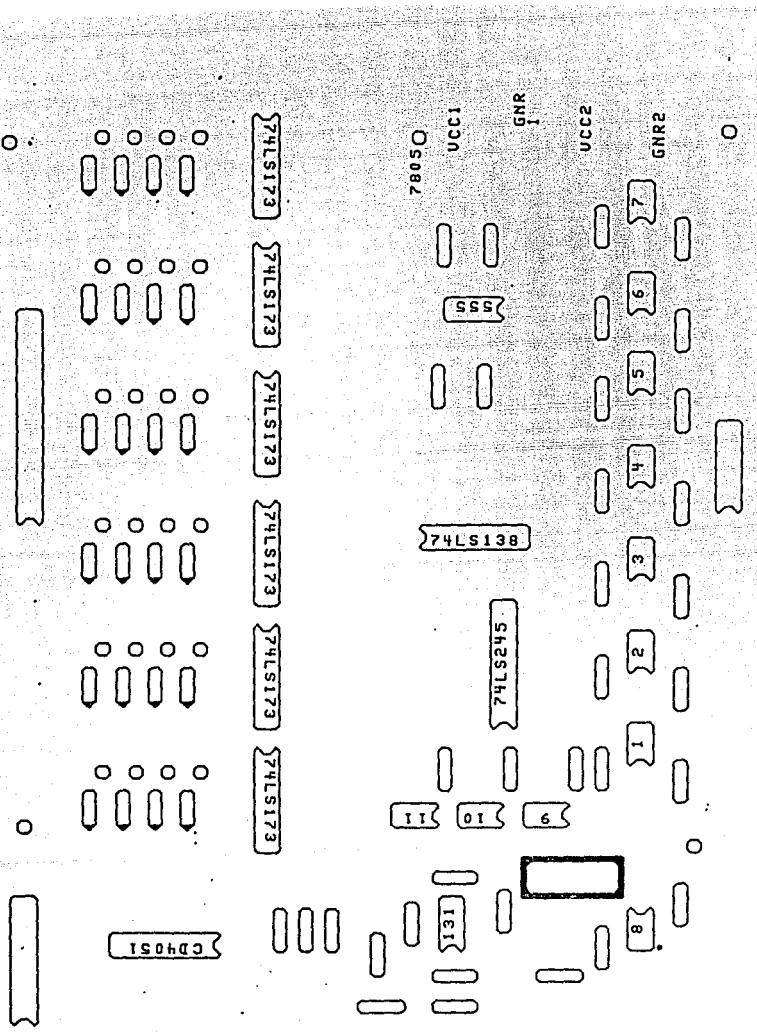

DRIVE 1
DRIVE 2
DRIVE 3
DRIVE 4
DRIVE 5
DRIVE 6
74LS173
74LS245
74LS138
4051
CLK1
10 KHz
CONVERTIDOR
A/F
RAYOS
DIRECCION
FRECUENCIA
CENTRO DE INSTRUMENTOS
SISTEMA DE OPTICA ACTIVA
GENERAL



CARA DE SOLDADURAS



CARA DE COMPONENTES


74LS173
74LS173
74LS173
74LS173
74LS173
74LS173
CD4051
7805Q
VCC1
GNR 1
555
74LS138
74LS245
VCC2
GNR2
7
6
5
4
3
2
1
11
10
9
13
8

MASCARILLA

ELEMENTO	DENOMINACION
C1 - C6	74LS173
C7	74LS245
C8	74LS138
C9	CD4051
C10	LM131
C11	LM555
C12 -C22	4N28
Q1-Q6	TIP41C
D1 - D6	IN4001
R1 - R6	10 KΩ
R7 - R17	12 KΩ
R18 - R28	220 Ω
R29	12 KΩ
R30	5 KΩ
R31	47 Ω
R_{in}	100 KΩ
R_L	100 KΩ
R_t	100 KΩ
C_{in}	0.1 µf
C_L	1 µf
C_t	0.01 µf

5 SOFTWARE

5.1 INTRODUCCION

El software del sistema de óptica activa, es una parte de un sistema para la operación del telescopio. La ejecución del módulo depende de la existencia de error en el paralaje entre los espejos.

Al presentarse un error en el paralaje de los espejos, existe un programa encargado de detectar la falla, así como de procesar información referente a ésta. Dentro de la información se contempla, el proporcionar las coordenadas para realizar la corrección pertinente. Este programa no está contemplado dentro esta tesis. Las coordenadas son enviadas a un módulo encargado de procesarlas para proporcionar información que pemita operar los motores de pasos, así como conocer la posición actual que presenta cada motor. Este programa es llamado: *Programa de control de óptica activa*

Para el diseño del software, se planteó un modelo en diagrama de bloques donde se observan las entradas y salidas que debe de contemplar. Este modelo se muestra en la figura 5.1.

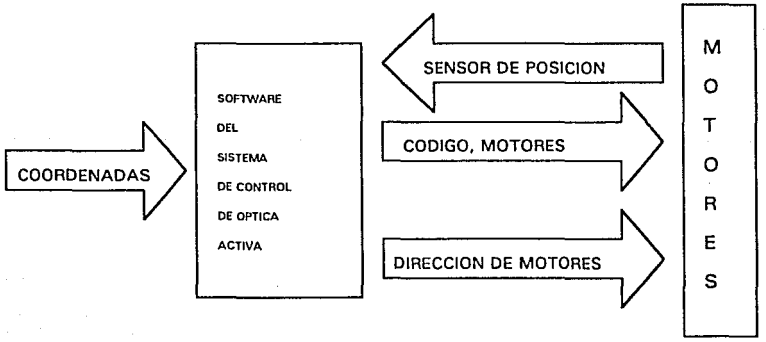


FIGURA 5.1

en esta figura se puede observar que los datos de la frecuencia de cada motor y las coordenadas de una nueva posición, son los datos de entrada. Los datos de salida son los códigos que deben recibir los motores de pasos para realizar sus movimientos, así como la dirección del puerto que se debe de habilitar en el hardware.

5.2 OPERACION BASICA

El diseño de software esta basado en tres procesos fundamentales denominados como:

a).- Adquisición de datos.
c).- Procesamiento de datos
b).- Transmisión de datos.

ADQUISICION DE DATOS. La computadora debe enviar la dirección del sensor de posición que se desea habilitar. Una vez que el puerto esta en línea, se procede a sensar la frecuencia que se encuentra presente en el puerto de entrada.

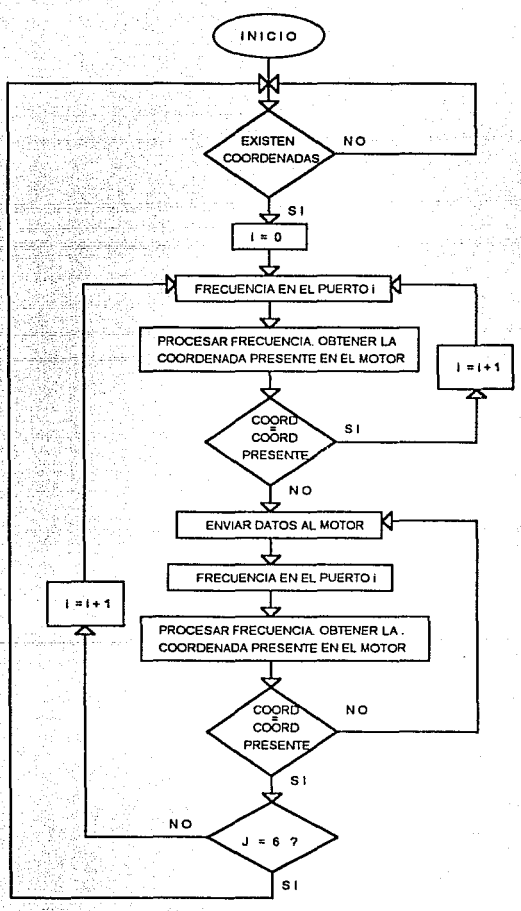
PROCESAMIENTO DE DATOS. El dato de la frecuencia sensada, es procesado para obtener su valor correspondiente en posición. El valor de posición es comparado con los valores de coordenadas enviados previamente por el programa principal. Si los valores comparados no son iguales, se procesa una nueva posición, ésto es, se obtiene el número de pasos proporcional a la diferencia en las coordenadas comparadas.

TRANSMISION DE DATOS. La computadora envía la dirección del motor que se desea habilitar. A continuación se envía la información correspondiente al movimiento que debe realizar el motor. La dirección y el código referente al movimiento en el motor, son enviados vía puerto paralelo.

Antes de mencionar la interacción de los procesos, es importante mencionar que los procesos se repiten para cada motor, es decir, la adquisición, procesamiento y transmisión de datos se realiza en cada motor por separado. También es importante mencionar que cada vez que se invoque al programa

para corregir una posición , el programa verificará las coordenadas que existan en los seis motores, aún cuando en uno de ellos, ésto no fuese necesario. Este proceso se puede visualizar mediante el diagrama de flujo de la figura 5.2

FIGURA 5.2



El programa inicia con la recepción de las coordenadas. Al recibir

las coordenadas, da principio a un proceso de adquisición de datos. Una vez que se tienen el dato de la frecuencia, del sensor correspondiente, la información es enviada al procesamiento de datos, donde, dependiendo de el resultado de comparación, se pasará ya sea, a la transmisión de datos o en su defecto se iniciará un nuevo ciclo para un nuevo motor. Si el siguiente proceso es el referente a la transmisión de datos, al término de éste, se debe de verificar que el movimiento en el motor haya sido el correcto, ésto se lleva cabo mediante una nueva adquisición de datos, seguida de un procesamiento de los mismos. Si al hacer la comparación entre ambas posiciones (la presente y a la que se quería llegar) existe una diferencia, da inicio nuevamente al proceso de transmisión de datos. El proceso de verificación se llevará a cabo cuantas veces sea necesario. Al término de éste proceso, el programa iniciará un nuevo ciclo para otro motor, ésto dependiendo de que todavía existan movimientos por realizar.

5.3 SESION TIPICA DE TRABAJO

Describiremos ahora una sesión tipica de trabajo en el software. Esta descripción tiene por objetivo, mostrar en forma detallada el sofware del programa, de tal manera que las nociones adquiridas al principio del capítulo, por la descripción a bloques del sistema sean mayormente entendidas. Se debe observar que en la descripción, se denominan puertos al conjunto motor-sensor de posición.

En el sistema de espejos, se ha detectado una falla en el paralaje . La detección es hecha por sensores ópticos, que proporcionan la información a un programa principal (tanto los sensores ópticos, como el programa principal, salen del alcance de la tesis).

El programa principal al recibir la información de falla, procesa información referente a una posición correcta. La información de las nuevas coordenadas es enviada al programa de control de óptica activa.

El programa de óptica activa, recibe la información. La información es

almacenada en un arreglo unidimensional de tipo registro, el arreglo es global, es decir, la información puede ser accesada en cualquier parte del programa, sin la necesidad de que sea pasado como parámetro a los procedimientos. Las variables del registro, así como su contenido se muestran a continuación:

a).- *D_PORT.* El número de puerto
b).- *N_PASO.* El desplazamiento que se debe de realizar
c).- *DIG.* El sentido del giro (este dato se carga una vez que se ha procesado la información sobre la posición)

Una vez que la información de las nuevas coordenadas son transferidas, el programa debe presentar la posición actual de cada cordenada. Con el fin de presentar una visualización de los dato de posición, el programa llama al procedimiento *COORD_OPER,* cuyo codigo es presentado a continuación

```
procedure coord_oper;

var i:integer;

begin
        pantalla(3,6,44,20,0,7,0);
        pantalla(5,5,47,19,7,0,0);
        mov(17,6,30,6,0,15,0,' INSTRUCCION ');
        pantalla(52,6,76,21,0,7,0);
        pantalla(54,5,77,20,7,0,0);
        mov(56,6,75,6,0,15,0,'COORDENADAS ');
        mov(56,8,66,8,0,15,1,  'x1  ');
        mov(56,10,66,10,0,15,1,'x2  ');
        mov(56,12,66,12,0,15,1,'x3  ');
        mov(56,14,66,14,0,15,1,'x4  ');
        mov(56,16,66,16,0,15,1,'x4  ');
        mov(56,18,66,18,0,15,1,'x4  ');
        i:=0;
        repeat
        port[numport]:= i;for j:=1 to 20000 do;
        coordenadas(i);inc(i,16);until(i=96);
end;
```

Los archivos que este procedimiento utiliza en forma directa o indirecta son

comentados a continuación:

PROCEDIMIENTO PANTALLA. Se encarga de generar una ventana. Las coordenadas, atributo *(color de la letra)* y fondo son definidas a criterio del programador. El código se muestra a continuación.

```
procedure pantalla(x1,y1,x2,y2,color_fon,color_let,ap:byte);
begin
        textbackground(color_fon);
        textcolor(color_let);
        window(x1,y1,x2,y2);
        if ap=0 then clrscr;
end;
```

PROCEDIMIENTO MOV. Funciona en forma similar al procedimiento pantalla, con la diferencia, de que coloca un letrero dentro de la ventana generada. El código se muestra a continuación .

```
procedure mov(x1,y1,x2,y2,cf,cl,ap:integer;var1:string);

begin
        pantalla(x1,y1,x2,y2,cf,cl,ap);
        write(var1);c_off;
end;
```

PROCEDIMIENTO COORDENADA. Tiene por objetivo, direccionar un puerto en particular; dependiendo del puerto que se desea sensar, el procedimiento llama a *COORD1*. El código se muestra a continuación.

```
procedure coordenada(ext:integer);
begin
case ext of

      0 :       coord1(8);
      16:       coord1(10);
      32:       coord1(12);
      48:       coord1(14);
      64:       coord1(16);
      80:       coord1(18);
end;end;
```

PROCEDIMIENTO COORD1. En este procedimiento, se hace uso de otro procedimiento llamado *FREC*, el cual se encaragara de leer la frecuencia presente en el puerto paralelo. El dato de frecuencia es procesado para obtener un valor sobre la posición real del motor. El dato es visualizado mediante el procedimiento *PANTALLA_ESC*. Los parámetros que recibe este procedimiento son referentes a las coordenadas, fondo y atributo del caracter, necesarios para generar una pantalla, así como el dato (tipo numérico), que se desea visualizar. A continuación son mostrados los códigos de los procedimientos *COORD1, FREC y PANTALLA_ESC*

```
procedure coord1(y:integer);

var
long:real;

begin
        frec;long:=frecuencia1*klong;
        pantalla_esc(65,y,74,y,4,15,0,long);
end;


procedure frec;
var
   cyc,n,n1,t,f,x,y,ch,cl:real;
begin
   dato1:=0;
      dato2:=0;
      dato3:=0;
      dato4:=0;
      ensambla(dato1,dato2,dato3,dato4);
      x:=256 mod dato1;
      y=(dato1/256);cl:=256 mod dato3;
      ch:=(dato3/256);
      n:=12*x+(256*12+1)*y-4;
      cyc:=dato3;n1:=n+11*(cyc-1)+4*ch;
      t:=1/7759683*n1/cyc;
      f:=1.39221241*(1/t)/10;
frecuencia1:= int((f*100+0.5)/100);
frecuencia1:=10*round(trunc(frecuencia1)*0.1);
end;

procedure pantalla_esc(x1,y1,x2,y2,color_fon,
```

```
                color_let,ap:byte;dato:real);

begin
        textbackground(color_fon);
        textcolor(color_let);window(x1,y1,x2,y2);
        if ap=0 then clrscr;write(' ',dato:5:2);
end;
```

Una vez que el programa principal ha transferido la información sobre las nuevas coordenadas del sistema, y en la pantalla aparece el formato donde se visualizara los datos referentes a las coordenadas, así como los movimientos que deberán de realizar los motores, el programa del control óptico invoca el procedimiento *PROS_DAT*. Este procedimiento se encarga de procesar la información contenida en las variables de uno de los registros. Para realizar el procesamiento, es necesario conocer la posición presente del motor que se desea mover (anteriormente se realizo una lectura de frecuencia en cada puero, sin embargo esta informacion no se guardo en ninguna variable), es decir, es necesario volver a leer la frecuencia presente en el puerto; debido a que existen seis puertos, debemos de habilitar solo uno, y esto lo llevamos a cabo por medio del procedimiento *DIREC_PUERTO* . Este procedimiento, recibe como parámetro, la variable *D_PORT* y regresa como parámetro la variable *EXT*. El parámetro *EXT*, define la dirección del puerto. Una vez que el puerto ha sido habilitado, se puede dar paso a la lectura de frecuencia, a partir de la cual de definira el numero de pasos y el sentido del giro para el motor en cuestión. Los datos del número de pasos y sentido de giro son pasados como parámetros a *PASO1*.

```
procedure pros_dat(dir_i:integer;palabra:string);

var
com_paso:real;

begin
      with cod_op[dir_i] do
      begin
         direc_puerto(ext,dport);
         port[numport]:=ext+15;
```

```
        frec;
        com_paso:=npaso;
        paso_in :=k_paso*(frecuencia1 - 100);
        npaso_ver := npaso*k_long;
        npaso := trunc(npaso_ver-paso_in);
        ver_dat(dir_i,palabra);
        if npaso<0 then
              begin
                 dig:=0;npaso:=-npaso;
              end
        else
              dig:=1;paso1(dir_i);
        coordenadas1(ext);
    end;
end;


procedure direc_puerto (var ext:integer;dport:integer);
begin
        case dport of
            1:      ext:=0;
            2:      ext:=16;
            3:      ext:=32;
            4:      ext:=48;
            5:      ext:=64;
            6:      ext:=80;
        end;
end;
```

PROCEDIMIENTO PASO1, se encarga de enviar el número de pasos necesarios para que el motor se coloque en la posición adecuada. El código de éste procedimiento se muestra a continuación. Como ya se mencionó, un paso corresponde a una palabra de 4 bits.

```
procedure paso1(pas_i:integer);
var
t,i:integer;
vel_r:longint;
vel1:real;

begin
    with cod_op[pas_i] do
    begin
```

```
t:=0;vel1:=4;vel_r:= 100*trunc(exp(vel1));
while npaso<>t do
  begin
    if t<npaso then
        begin
            t:= t+1;
            port[numport]:= 10 + ext - dig;
            if t<(0.9*npaso) then arranca(vel_r,vel1,vel)
            else if t>=(0.9*npaso)
            then desa(vel_r,vel1,vel);for i:=1 to vel_r do;
        end;
    if t<npaso then
        begin
            t := t+1;
            port[numport]:= 6 + ext - dig;
            if t<(0.9*npaso) then arranca(vel_r,vel1,vel)
            else if t>=(0.9*npaso) then
            desa(vel_r,vel1,vel);for i:=1 to vel_r do;
        end;
    if t<npaso then
        begin
            t := t+1;
            port[numport]:= 5 + ext + dig;
            if t<(0.9*npaso) then arranca(vel_r,vel1,vel)
            else if t>=(0.9*npaso) then
            desa(vel_r,vel1,vel);for i:=1 to vel_r do;
        end;
    if t<npaso then
        begin
            t := t +1;
            port[numport]:= 9 + ext + dig;
            if t<(0.9*npaso) then arranca(vel_r,vel1,vel)
            else if t>=(0.9*npaso) then
            desa(vel_r,vel1,vel);for i:=1 to vel_r do;
        end;
  end;end;port[numport] := 9;for i:= 1 to 5000 do;
end;
```

Debido a que el motor debe de vencer la inercia a que está sometido por el sistema, se diseñó un procedimiento que le permita una aceleración creciente, hasta llegar al máximo permisible por el sistema. Este procedimiento es llamado dentro el sistema como *ARRANCA*.

PROCEDIMIENTO ARRANCA. Se encarga de suministrar una velocidad creciente en cada paso. Este proceso se llevará a cabo, mientras que el numero de paso presente, sea menor a 90% del número de pasos que se deberán realizar. El código del procedimiento *ARRANCA* se muestra a continuación.

```
procedure arranca(var vel_i:longint;var vel1:real;

vel_c:integer);

const
valor=0.01;
begin
      if vel_i>vel_c then
      begin
      vel1:=vel1-valor;vel_i:= 100*trunc(exp(vel1));
      end
      else
      begin
      vel1:=vel1-0;vel_i:= 100*trunc(exp(vel1));
      end;
end;
```

PROCEDIMIENTO DESA. Una vez que se ha sobrepasado el 90% de el número de pasos, este procedimiento se encarga de dismimuir la velocidad en cada paso, ésto con el propósito de evitar al máximo el error en posición final que se nos presente en el puerto. Tanto *DESA* como *ARRANCA*, están basados en una función de tipo exponecial. El código de *DESA* es mostrado a continuación.

```
procedure desa(var vel_i:longint;

var vel1:real;vel_c:integer);

const valor=0.05;
begin
if vel_i<6000 then
      begin
      vel1:=vel1+valor;vel_i:= 100*trunc(exp(vel1));
      end
 else
      begin
```

```
    vel1:=vel1+0;vel_i:= 100*trunc(exp(vel1));
    end;end;
```

Al terminar de enviar el número de pasos , el programa llama a el procedimiento LIMPIA

PROCEDIMIENTO LIMPIA. Con el fin de ahorrar energía, así como de evitar calentamiento innecesario en la fuente del sistema, se envía una palabra de 4 bits de "ceros", que permita colocar en estado de corte a los transistores de los drives del motor. Su código es mostrado a continuación.

```
procedure lim_pi;
var i:integer;
begin
        port[numport]:=0;for  i:= 1 to 1000 do;
        port[numport]:=16;for i:= 1 to 1000 do;
        port[numport]:=32;for i:= 1 to 1000 do;
        port[numport]:=48;for i:= 1 to 1000 do;
        port[numport]:=64;for i:= 1 to 1000 do;
      port[numport]:=80;for i:= 1 to 1000 do;
end;
```

Al término del procedimiento *LIMPIA*, se debe verificar si la posición presente es la misma que la posición pedida al inicio de la rutina. Una vez más, se llama a la rutina *FREC*, se procesa el valor de la frecuencia, se compara contra el dato pedido y dependiendo del valor de la comparación se decide si es necesario un nuevo movimiento. Si los datos comparados no son iguales, llama a *PASO1*. Este proceso se realizará cuantas veces sea necesario. Si al comparar los datos, éstos son iguales, el procedimiento *COORDENADAS1* es llamado. De esta manera se visualiza la posición final del motor correspondiente.

Una vez que se ha concluido este proceso, el programa deberá atender a otro movimiento (si lo hay) en el motor siguiente, es decir se invoca de nuevo al procedimiento *PROS_DAT* y a los procedimientos que interactúan con este.

Cuando todas las coordenadas fueron procesadas, el programa termina limpiando la pantalla.
El programa completo, se encuentra en el apéndice B.

COMENTARIOS FINALES

El sistema de óptica activa, diseñado en el Centro de Instrumentos, cumplió en forma satisfactoria los objetivos correspondientes al prototipo. A continuación se presentan algunas sugerencias tendientes a mejorar el prototipo y que podrian aplicarse a futuro.

El utilizar el multiplexor analógico en los sensores de posición, nos obliga a reducir el voltaje de entrada máximo de 2/3Vcc a 1/2Vcc; lo anterior se debe a que la respuesta del multiplexor analógico es mas rápida que la respuesta del convertidor voltaje/frecuencia, lo cual provoca que el sistema salga de estabilidad cuando existen cambios rápidos en el voltaje de entrada.

El rango recomendado de frecuencia de entrada a la computadora (y por consiguiente para la operación del convertidor voltaje/ frecuencia), es del orden de 20 kHz. Lo anterior, se debe a que el programa diseñado para la lectura de frecuencia, no permite un rango mayor a 30 kHz.

El tiempo aproximado para la lectura de la frecuencia, fue de 0.73 seg. , (medido experimentalmente). Este tiempo, puede ser mejorado si se utiliza una máquina de mayor velocidad; para las pruebas, se utilizó, una máquina XT 8086.

El tiempo mínimo aproximado para transmitir una secuencia de cuatro pasos, fue de 0.03 seg. Debido a las características del motor, éste tiempo solo puede ser mejorado, cambiando a un motor que tenga un tiempo de respuesta menor.

La fuente que alimenta al sensor de posición tiene un alto porcentaje de regulación, con el fin de evitar variaciones por ruido, que pueden afectan al momento de realizar la lectura.

Es necesario buscar alternativas para los sensores de posición los cuales son un punto clave del sistema, debido a que en las pruebas realizadas, se encontro un error de ±5 herts, al sensar la frecuencia, la cual esta estrechamente relacionada con la posición. Este error puede ser modificado, si en lugar de sensar una sola vez la frecuencia, se sensa un varias veces, a fin de obtener un

valor promedio. Lo que implica un tiempo mayor de procesamiento para la computadora.

El desplazamiento por paso no es un problema, dado que se puede obtener mayor precisión, con solo cambiar la relación de engranes entre motor y tornillo sin fin; el punto clave, sigue siendo el conocer la posición en cada coordenada. De acuerdo con el apéndice A, se observa que un incremento de 25 µm (minimo incremento posible) implica un aumento de 10 Hz en la frecuencia sensada por la computadora. Si deseamos aumentar la precisión del sistema se tienen dos posibilidades:

a).- Aumentar el número de vueltas del transductor potenciométrico (actualmente de dispone de 10 vueltas y el máximo comercial es de 12).

b).- Aumentar la velocidad de lectura de la frecuencia. Sin embargo existen diferentes limitantes a este respecto debido a que el programa permite hasta 30 kHz, y aún utilizando una computadora muy rápida, la escala total de frecuencia requerida para un incremento de 1 µm es del orden de 500 kHz.

Aunque el tiempo de procesamiento no es un parámetro crítico en el sistema, se pueden lograr algunas mejoras notorias en éste , si las etapas de visualización son depuradas.

Si bien el utilizar un juego de engranes nos presenta algunas ventajas en cuanto a velocidad, torque, etc.; estos tienen desventajas en el error de posición, debido al juego inherente de fabricación, por lo que se recomienda evitarlo, por lo menos en el sensor de posición.

La filosofía del sistema desarrollado en éste proyecto, puede ser aplicada en otros sistemas que requieran del uso de servomecanismos, con la ventaja de tener un mayor rango de posibilidades al tener a la computadora como controlador.

APENDICE A

A continuación se propone un acoplamiento mecanico entre el motor, tornillo sin fin y potenciómetro

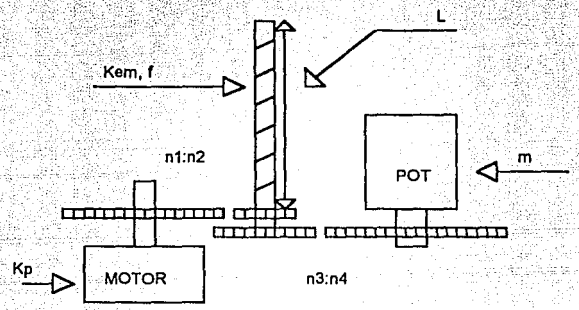


FIGURA a.1 Conexión propuesta para el motor, el transductor y el tornillo sin fín

Para la presente discusión la nomenglatura utilizada es la siguiente:

- V_{pot} : vuelta del potenciómetro.
- V_{mot} : vuelta del motor.
- V_{em} : vuelta en el tornillo sin fin
- f_T: frecuencia máxima aceptable
- L : carrera total del tornillo sin fín
- n1, n2, n3, n4 : relación de engranes
- K_p: número de pasos en una vuelta del motor
- K_{em} : número de vueltas en el tornillo sin fin
- m : número de vueltas del potenciómetro

Las relaciones que existen entre el rango de frecuencia, las vueltas en el potenciómetro, el número de pasos en el motor, etc., son obtenidas a continuación.

Considerando lo anterior se tiene:

$$n1\ [V_{mot}] = n2\ [V_{em}] \qquad (a.1)$$

$$n3\ [V_{em}] = n4\ [V_{pot}] \qquad (a.2)$$

$$1\ [V_{pot}] = 360\ [\text{grados}] \qquad (a.3)$$

$$1\ [V_{mot}] = K_p\ [\text{pasos}] \qquad (a.4)$$

$$m\ [V_{pot}] = f_T\ [\text{Hz}] \qquad (a.5)$$

$$1\ [V_{em}] = \frac{L}{K_{em}}\ [\text{cm}] \qquad (a.6)$$

$$\frac{K_{em}}{m} = \frac{n3}{n4} \qquad (a.7)$$

sustituyendo a.1 en a.2

$$\frac{n1 \times n3}{n2 \times n4}\ [V_{mot}] = 1\ [V_{pot}] \qquad (a.8)$$

Sustituyendo a.3 y a.4 en 4.8

$$1\ [\text{paso}] = \frac{n2 \times n4 \times 360}{n1 \times n3 \times K_p}\ [\text{grados}] \qquad (a.9)$$

Sustituyendo a.3 en a.5

$$1\ [\text{grado}] = \frac{f_T}{360 \times m}\ [\text{Hz}] \qquad (a.10)$$

susituyendo a.10 en a.9

$$1\ [\text{paso}] = \frac{n2 \times n4 \times f_T}{n1 \times n3 \times K_p \times m}\ [\text{Hz}] \qquad (a.11)$$

sustituyendo a.6 en a.1

$$\frac{n1}{n2}[V_{mot}] = \frac{L}{K_{em}}[cm] \qquad (a.12)$$

sustituyendo a.4 en a.12

$$1\,[paso] = \frac{n2 \times L}{n1 \times K_p \times K_{em}}\,[cm] \qquad (4.20)$$

Considerando las ecuaciones y los siguientes datos

m=10	n1=2	n2=1
n3=2	n4=1	L= 5cm
Kp=480	Kem=20	ft=20,000 Hz

se tiene en teoría:

1[PASO] = 2.083 [Hz]

1[PASO] = 5.2 [μm]

1[μm] = 0.4 Hz

Considerando un error de ± 5 [Hz] (este es un dato experimental),se tiene que:

25 [μm]= 10 [Hz]

A partir de las ecuaciones, es posible conocer, el número de pasos necesarios para obtener un valor en cm, Hertz, grados en el potenciómetro. Además, manipulando estas ecuaciones, es posible conocer la relación que existe entre la posición y la frecuencia. A continuacion se dan

APENDICE B

```
TESIS: SISTEMA DE OPTICA ACTIVA DEL TELESCOPIO DE SAN PEDRO
MARTIR, BAJA CALIFORNIA.
AUTOR: CASTILLO HERNANDEZ JOSE.
LABORATORIO DE ELECTRONICA.
CENTRO DE INSTRUMENTOS.

program robot;
uses Crt,Dos;

type
   dat_rob = record

       cordX,cordY,cordZ,npaso:real;
       dport,vel,dig,oper,indicador:integer;
   end;
   oper_gen=array[1..200] of dat_rob;
   pal = string[20];
   cod = array [1..10] of string[20];
   inf_dat = array[0..200] of string[20];
   string30=string[30];
const
      klong       = 0.5;
      kroo        = 0.5;
      kree        = 0.5;
      kzeta       = 0.5;
      krot        = 0.5;
      kcerrar     = 0.5;
      minV        = 000;
      maxV        = 40000;
      minX        = -100;
      maxX        = 100;
      minY        = -100;
      maxY        = 100;
      minZ        = 0;
      maxZ        = 100;
      k_long      = 20;
      k_paso      = 0.5;
      k_frec      = 20;
      numport     = 888;
      puerto      = 889;(* 889;*) (* 957 *)
      puerto1     = 888;(* 888;*) (* 956 *)


VAR
   DATO1,DATO2,DATO3,DATO4:WORD;
    cod_op : oper_gen;
    i,j,t:integer;
    pos_pant,puert,cont,ext,paso,pc: integer;
    tdir: char;
    frecuencia1,
    paso_in,
```

```
    npaso_ver  : Real;
    tdat       :char;
    palabra,
    nombre     :string[20];
    archa      : text;
    color      :integer;
    ch         :char;
    val1       :longint;
    let        :cod;


function letra(dato:integer):char;
begin
case dato of
65..90   : letra:=chr(dato);
97..122  : letra:=chr(dato);
33..57   : letra:=chr(dato);
else       letra:=chr(20);
end;
end;
procedure pantalla(x1,y1,x2,y2,color_fon,color_let,ap:byte);
begin
textbackground(color_fon);
textcolor(color_let);
window(x1,y1,x2,y2);
if ap=0 then clrscr;
end;

procedure pantalla_esc(x1,y1,x2,y2,color_fon,
color_let,ap:byte;dato:real);
begin
textbackground(color_fon);
textcolor(color_let);
window(x1,y1,x2,y2);
if ap=0 then clrscr;
write(' ',dato:5:0);
end;

procedure pantalla_lec(x1,y1,x2,y2,a,b:integer);
begin
    pantalla(x1,y1,x2,y2,a,b,0);
    pantalla(x1+1,y1,x2,y2+3,a,b,1);
end;

function tecla_oper:longint;
begin
    ch:=readKEY;
    if ch=#0 then
    begin
      ch := ReadKey;
      tecla_oper:= Ord(Ch)+1000;
```

```
    end
    else tecla_oper:=ord(ch);
end;

procedure
mov_dat(x1,y1,x2,y2,cf,cl,ap,i:integer;var1:string);

begin
  pantalla(x1,y1,x2,y2,cf,cl,ap);
  write(i);
  gotoxy(6,1);
  write( var1);c_off;
end;

procedure mov(x1,y1,x2,y2,cf,cl,ap:integer;var1:string);
begin
  pantalla(x1,y1,x2,y2,cf,cl,ap);
  write(var1);c_off;
end;
procedure ver_archivo(ind1:integer;var ind2:integer);
var
j1  :file of byte;
opcion,result,contenido:integer;
begin
ind2:=0;
opcion:=115;
while ((opcion=115)or(opcion=83))and(ind2=0) do
begin
    repeat
    pantalla(2,24,30,24,1,15,1);
    write(' DESEAS CONTINUAR (S/N)? ');
    opcion:=tecla_oper;
    until(opcion=115)or(opcion=83)
    or(opcion=110)or(opcion=78);
    pantalla(2,24,80,24,7,0,0);
    if (opcion=115)or(opcion=83) then
    begin
    pantalla(15,5,30,5,15,0,0);
    write(' ARCHIVO ?');
    pantalla_lec(15,7,30,7,7,0);
    c_on;
    readln(nombre);
    c_off;
    assign(j1,nombre);
    {$I-}
    reset(j1);
    {$I+}
    result := ioresult;
    if result=0 then
    begin
    contenido:=filesize(j1);
```

```
    close (j1);
    end;
    if (result<>0) or (contenido= 0) then
       begin
       if (ind1=1)and(nombre<>'') then
         begin
         assign(archa,nombre);
         rewrite(archa);
         pantalla(15,5,30,8,1,0,0);
         pantalla(2,24,40,24,1,7,1);
         write(' ARCHIVO ',NOMBRE);
         ind2:=1;
         end
       else
         begin
         if (result=0)and(nombre<>'') then
            begin
            if (contenido=0) then;
               BEGIN
               pantalla(50,24,80,24,4,15,1);
               textcolor(255);
               write(' ARCHIVO VACIO ');
               end;
            end
         else
            begin
            pantalla(50,24,80,24,4,15,1);
            textcolor(255);
            write(' ARCHIVO INEXISTENTE ');
            end;
         end;
       end
    else
       begin
       assign(archa,nombre);
       reset (archa);
       pantalla(15,5,30,8,1,0,0);
       pantalla(2,24,40,24,1,15,1);
       write(' ARCHIVO ',NOMBRE,' ');
       ind2:=2;
       end;
    end;
end;
end;

procedure ver_archivo1(var ind2:integer);
var
j1  :file of byte;
opcion,result,contenido:integer;
```

```
begin
ind2:=0;
opcion:=115;
while ((opcion=115)or(opcion=83))and(ind2=0) do
begin
    repeat
    pantalla(2,24,30,24,1,15,1);
    write(' DESEAS CONTINUAR (S/N)? ');
    opcion:=tecla_oper;
    until(opcion=115)or
    (opcion=83)or(opcion=110)or(opcion=78);
    pantalla(2,24,80,24,7,0,0);
    if (opcion=115)or(opcion=83) then
        begin
            pantalla(2,24,13,24,1,15,1);
            write(' ARCHIVO ?');
            pantalla(13,24,14,24,4,15,0);
            pantalla(13,24,60,24,4,15,1);
            c_on;
            readln(nombre);
            pantalla(2,24,70,24,7,0,0);
            c_off;
            assign(j1,nombre);
            {$I-}
            reset(j1);
            {$I+}
            result := ioresult;
            if result=0 then
            begin
            contenido:=filesize(j1);
            close (j1);
            end;
            if((result=0)AND(contenido=1)AND(nombre<>''))
            or((result<>0)AND(nombre<>'')) then
                begin
                assign(archa,nombre);
                rewrite(archa);
                pantalla(2,24,70,24,7,0,0);
                pantalla(2,24,40,24,1,7,1);
                write(' ARCHIVO ',NOMBRE);
                ind2:=1;
                end
            else if (result=0)and
                (contenido<>0)and(nombre<>'') then
                begin
                    repeat
                    pantalla(2,24,50,24,1,31,1);
                    write(' ARCHIVO CON INFORMACION
                    CONTINUAR (S/N)? ');
                    opcion:=tecla_oper;
                    pantalla(2,24,50,24,7,35,0);
```

```
                    until(opcion=115)or
                    (opcion=83)or(opcion=110)
                    or(opcion=78);
                    if (opcion=115)or(opcion=83) then
                        begin
                        assign(archa,nombre);
                        rewrite(archa);
                        pantalla(2,24,40,24,1,7,1);
                        write(' ARCHIVO ',NOMBRE);
                        ind2:=1;
                        end
                     else opcion:=110;
                end;
        end
        else ind2:=0;
    end;
end;

procedure letrero(var codigo:pal ;var1:string);
begin
        pantalla(44,10,80,10,7,0,1);
        write(var1);
        pantalla_lec(44,13,75,13,7,0);
        c_on;
        readln(codigo);
        pantalla(44,10,75,14,1,7,0);
        c_off;
end;

procedure lis_ta(j,pc:integer;codigo:inf_dat);
var
i:integer;
begin
    i:=1;
    while (i<11)and(j<pc) do
    begin
        write(j);
        gotoxy(6,i);
        writeln(codigo[j]);
        inc(i);
        inc(j)
    end;
    write(j);
    gotoxy(6,i);
    write(codigo[j]);
end;

PROCEDURE FREC;
var
   CYC,N,N1,T,F,X,Y,CH,CL:REAL;
BEGIN
```

```
     DATO1:=0;DATO2:=0;DATO3:=0;DATO4:=0;
     ENSAMBLA(dato1,dato2,dato3,dato4);
     X:=256 MOD DATO1;
     Y:=(DATO1/256);
     CL:=256 MOD DATO3;
     CH:=(DATO3/256);
     N:=12*X+(256*12+1)*Y-4;
     CYC:=DATO3;
     N1:=N+11*(CYC-1)+4*CH;
     T:=1/7759683*N1/CYC;
     F:=1.39221241*(1/T)/10;
     FRECUENCIA1:= INT((F*100+0.5)/100);
END;

procedure coord1(Y:integer);
var
long:real;
   begin
      frec;
      long:=frecuencia1*klong;
      pantalla_esc(65,y,74,y,4,15,0,long);
   end;

procedure coordenadas1(ext:integer);
begin
case ext of

    0 :       coord1(8);
    16:       coord1(10);
    32:       coord1(12);
    48:       coord1(14);
    64:       coord1(16);
    80:       coord1(18);
end;
end;
procedure coord2(Y:integer);
var
long:real;

   begin
      frec;
      pantalla_esc(19,y,30,y,4,15,0,frecuencia1);
      long:=frecuencia1*klong;
      pantalla_esc(60,y,70,y,4,15,0,long);
   end;

procedure coordenadas2(ext:integer);
begin
case ext of
    0 :       coord2(8);
    16:       coord2(10);
```

```
    32:       coord2(12);
    48:       coord2(14);
    64:       coord2(16);
    80:       coord2(18);
end;
end;

procedure pos_dat(pos_pant,pc:integer;palabra:string);
begin
with cod_op[pc] do
  begin
  mov(6,pos_pant,46,pos_pant,0,15,0,palabra);
  end;
end;

procedure ver_dat(pc:integer;palabra:string);
var
oper:integer;
val1:real;
begin
if cont<=9 then oper:=1
else oper:=2;
case  oper of
  1:   pos_dat(pos_pant,pc,palabra);
  2:   begin
         cont:=1;
         pos_pant:=8;
         pantalla(5,7,47,19,7,0,0);
         pos_dat(pos_pant,pc,palabra);
       end;
end;
pos_pant:=pos_pant + 1;inc(cont);
end;

procedure arranca(varvel_i:longint;var
                  vel1:real;vel_c:integer);
const
valor=0.01;
begin
    if vel_i>vel_c then
    begin
    vel1:=vel1-valor;
    vel_i:= 100*trunc(exp(vel1));
    end
    else
    begin
    vel1:=vel1-0;
    vel_i:= 100*trunc(exp(vel1));
    end;
end;
```

```
procedure desa(var vel_i:longint;var
                vel1:real;vel_c:integer);
const
valor=0.05;
begin
if vel_i<6000 then
    begin
    vel1:=vel1+valor;
    vel_i:= 100*trunc(exp(vel1));
    end
 else
    begin
    vel1:=vel1+0;
    vel_i:= 100*trunc(exp(vel1));
    end;
end;

procedure paso1(pas_i:integer);
var
t,i:integer;
vel_r:longint;
vel1:real;

begin
   with cod_op[pas_i] do
   begin
   t:=0;
   vel1:=4;
   vel_r:= 100*trunc(exp(vel1));
   while npaso<>t do
    begin
      if t<npaso then
          begin
             t:= t+1;
             port[numport]:= 10 + ext - dig;
             if t<(0.9*npaso) then
             arranca(vel_r,vel1,vel)
             else if t>=(0.9*npaso) then
             desa(vel_r,vel1,vel);
             for i:=1 to vel_r do;
          end;
      if t<npaso then
          begin
             t := t+1;
             port[numport]:= 6 + ext - dig;
             if t<(0.9*npaso) then
             arranca(vel_r,vel1,vel)
             else if t>=(0.9*npaso) then
             desa(vel_r,vel1,vel);
             for i:=1 to vel_r do;
          end;
```

```
      if t<npaso then
           begin
             t := t+1;
             port[numport]:= 5 + ext + dig;
             if t<(0.9*npaso) then
             arranca(vel_r,vel1,vel)
             else if t>=(0.9*npaso) then
             desa(vel_r,vel1,vel);
             for i:=1 to vel_r do;
           end;
      if t<npaso then
           begin
             t := t +1;
             port[numport]:= 9 + ext + dig;
             if t<(0.9*npaso) then
             arranca(vel_r,vel1,vel)
             else if t>=(0.9*npaso) then
             desa(vel_r,vel1,vel);
             for i:=1 to vel_r do;
           end;
   end;
   end;
   port[numport] := 9;
   for i:= 1 to 5000 do;
end;

procedure direc_puerto (var ext:integer;dport:integer);
begin
      case dport of
         1:   ext:=0;
         2:   ext:=16;
         3:   ext:=32;
         4:   ext:=48;
         5:   ext:=64;
         6:   ext:=80;
      end;
end;

procedure pros_dat(dir_i:integer;palabra:string);
var
com_paso:real;
begin
    with cod_op[dir_i] do
    begin
      direc_puerto(ext,dport);
      port[numport]:=ext+15;
      frec;
      com_paso:=npaso;
      paso_in :=k_paso*(frecuencia1 - 100);
      npaso_ver := npaso*k_long;
      npaso := trunc(npaso_ver-paso_in);
```

```
      ver_dat(dir_i,palabra);
      if npaso<0 then
          begin
            dig:=0;
            npaso:=-npaso;
          end
      else
          dig:=1;
      paso1(dir_i);
      coordenadas1(ext);
    end;
end;

procedure lim_pi;
var i:integer;
begin
          port[numport]:=0;
          for i:= 1 to 1000 do;
          port[numport]:=16;
          for i:= 1 to 1000 do;
          port[numport]:=32;
          for i:= 1 to 1000 do;
          port[numport]:=48;
          for i:= 1 to 1000 do;
          port[numport]:=64;
          for i:= 1 to 1000 do;
          port[numport]:=80;
          for i:= 1 to 1000 do;
end;

procedure mov_inst(i:integer;var
                   var1:pal;var band:integer);
var
num1,num2,num3,num4,
num5,num6,apunt1,apunt2,
apunt3,apunt4,apunt5 :integer;
inst1,inst2,inst3,inst4,
inst5,var2,var3,var4 : string[20];
Begin
   with cod_op[i] do
   begin
      num1 := pos ('(',var1);
      num2 := pos (',',var1);
      num4 := pos (')',var1);
      var2 := copy (var1,num2+1, num4-1-num2);
      num3 := pos(',',var2);
      num3 :=num2+num3;
      inst1 := copy (var1,num1+1,num2-1-num1);
      inst2 := copy (var1,num2+1, num3-1-num2);
      inst3 := copy (var1,num3+1, num4-1-num3);
      val(inst1,dport,apunt1);
```

```
      val(inst2,npaso,apunt2);
      val(inst3,vel,apunt3);
      vel:=vel*100;
      if (apunt2<> 0) then
         begin
            pantalla(2,24,40,24,1,255,1);
            write(' ERROR EN EL DESPLAZAMIENTO  ',inst2);
            band:=0;
         end;
      if (apunt1 = 0)and(band=1) then
         begin
              case dport of
              1..6  : ;
              else    begin
                      pantalla(2,24,40,24,1,255,1);
                      write(' NO EXISTE MOTOR   ',inst1);
                      band := 0;
                      end;
              end;
         end;
      if (apunt3=0)and(band=1)
        and(vel>=minV)and(vel<=maxV) then
      else    begin
              pantalla(2,24,40,24,1,255,1);
              write(' VELOCIDAD FUERA DE RANGO ',inst3);
              band := 0;
              end;
  end
end;

procedure codigo(i:integer;var var1:pal;var band:integer);
var
num1 :integer;
inst1 : string[20];
begin
   with cod_op[i] do
   begin
        num1:= pos('(',var1);
        inst1 := copy (var1,1,num1-1);
        if (inst1 = 'mov')or(inst1='MOV') then
            begin
               indicador:=1;
               band:=1;
               mov_inst(i,var1,band);
            end
        else
            begin
               pantalla(2,24,40,24,1,255,1);
               write(' ERROR DE SINTAXIS  ',inst1);
            end;
   end;
```

```
end;

procedure coord_oper;
var
   i:integer;
begin
      pantalla(3,6,44,20,0,7,0);
      pantalla(5,5,47,19,7,0,0);
      mov(17,6,30,6,0,15,0,' INSTRUCCION ');
      pantalla(52,6,76,21,0,7,0);
      pantalla(54,5,77,20,7,0,0);
      mov(56,6,75,6,0,15,0,'     COORDENADAS ');
      mov(56,8,66,8,0,15,1,'    x1  ');
      mov(56,10,66,10,0,15,1,'   x2  ');
      mov(56,12,66,12,0,15,1,'   x3  ');
      mov(56,14,66,14,0,15,1,'   x4  ');
      mov(56,16,66,16,0,15,1,'   x4  ');
      mov(56,18,66,18,0,15,1,'   x4  ');
      i:=0;
      repeat
      port[numport]:= i;
      for j:=1 to 20000 do;
      coordenadas1(i);
      inc(i,16);
      until(i=96);
end;


procedure rec_inst;
var
tp_inst:inf_dat;
var1:pal;
band,result,
te_cla,contenido,ind1,indi,t:integer;
begin
  ver_archivo(0,ind1);
  if ind1=2 then
     begin
     pc:=0;
     while not eof (archa) do
          begin
          pc:=pc+1;
          readln(archa,palabra);
          tp_inst[pc] := palabra;
          end;
     close (archa);
     j:=0;
     indi:=1;
     pantalla(10,7,37,21,0,0,0);
     pantalla(11,6,38,20,7,0,0);
     while j<>pc do
```

```
      begin
      j:=j+1;
      repeat
        if j-indi = 11 then
        inc(indi);
        pantalla(12,8,37,19,7,0,0);
        c_off;
        lis_ta(indi,j,tp_inst);
        for t:=1 to 1000 do;
        band:=0;
        var1:= tp_inst[j];
        codigo (j,var1,band);
        if band=0 then
          begin
          letrero(var1,' NUEVA INSTRUCCION ');
          pantalla(2,24,80,24,7,0,0);
          pantalla(2,24,80,24,1,15,1);
          end;
          write(' ARCHIVO ',NOMBRE,' ');
        tp_inst[j]:=var1;
      until(band=1);
      end;
   pantalla(10,6,38,21,1,7,0);
   assign(archa,nombre);
   rewrite (archa);
   pc := 0;
   while pc<>j do
      begin
      pc:=pc+1;
      palabra:=tp_inst[pc];
      writeln(archa,palabra);
      end;
   close(archa);
   pantalla(2,24,40,24,1,15,1);
   write(' ARCHIVO ',NOMBRE,' ');
   coord_oper;
   pos_pant:=8;
   cont:=1;
   for i := 1 to pc do
       begin
       with cod_op[i] do
          begin
          pros_dat(i,tp_inst[i]);
          end;
       end;
pantalla(35,24,80,24,4,31,1);
write(' PRESIONE ENTER PARA CONTINUAR ');
repeat
until(tecla_oper=13);
lim_pi;
pantalla(2,24,80,24,7,0,0);
```

```
    end;
end;

procedure numero_pasos (var t_man,repe:integer;ext:integer);
begin
          if t_man=13 then
          begin
          mov(20,24,35,24,1,15,1,' PASOS ? ');
          pantalla(36,24,50,24,4,15,0);
          read (repe);
          pantalla(20,24,70,24,7,15,0);
             case ext of
              0:    pantalla(37,8,45,8,4,15,0);
              16:   pantalla(37,10,45,10,4,15,0);
              32:   pantalla(37,12,45,12,4,15,0);
              48:   pantalla(37,14,45,14,4,15,0);
              64:   pantalla(37,16,45,16,4,15,0);
              80:   pantalla(37,18,45,18,4,15,0);
             end;
             clrscr;
          write(' ',repe:5);
          t_man:=tecla_oper;
          end;
end;

procedure paso_manual(var t_man,repe:integer;
                      pas_i,ext,indica1:integer);
var
cont_i,i,paso_temp,desicion,sal:integer;

begin
  with cod_op[pas_i] do
  begin
     cont_i:=1;
    repeat
          if(t_man=1075) then
          dig:=1
          else if (t_man=1077) then
          dig:=0;
          if ((t_man=1075)or(t_man=1077))
             and(indica1=0)then
             begin
             desicion:=t_man;
             repeat
                 while not(keypressed) do
                    begin
                       port[numport]:= 10 + ext - dig;
                    for i:=1 to vel do;
                       port[numport]:= 6 + ext - dig;
                    for i:=1 to vel do;
                       port[numport]:= 5 + ext + dig;
```

```
                for i:=1 to vel do;
                   port[numport]:= 9 + ext + dig;
                for i:=1 to vel do;
                end;
                t_man:=tecla_oper;
             until(desicion<>t_man);
          end
       else if ((t_man=1075)or(t_man=1077))
          and(indica1=1)then
          begin
          desicion:=t_man;
          repeat
          sal:=1;
              while not(keypressed)and(sal<=repe) do
                 begin
                    port[numport]:= 10 + ext - dig;
                 for i:=1 to vel do;
                    port[numport]:= 6 + ext - dig;
                 for i:=1 to vel do;
                    port[numport]:= 5 + ext + dig;
                 for i:=1 to vel do;
                    port[numport]:= 9 + ext + dig;
                 for i:=1 to vel do;
                 inc(sal);
                 if(sal>repe) then
                 coordenadas2(ext);
                 end;
          t_man:=tecla_oper;
          if sal=repe then
          case desicion of
          1077: t_man:=1075;
          1075: t_man:=1077;
          end;
          until(desicion<>t_man);
          end ;
       coordenadas2(ext);
  if(t_man<>1059)and(t_man<>1060)then
  begin
       if (indica1=0)and(t_man<>13)and(t_man<>1075)
       and(t_man<>1077)and(t_man<>1072)
       and(t_man<>1080)and(t_man<>27) then
       t_man:=tecla_oper
       else if (indica1=1)and(sal<>repe) then
            begin
            if (t_man<>13)and(t_man<>1075)and
            (t_man<>1077)and(t_man<>1072)and
            (t_man<>1080)and(t_man<>27) then
            t_man:=tecla_oper;
            end
       else if(indica1=1)and(sal=repe) then
            begin
```

```
                if (t_man<>13)and(t_man<>1075)and
                (t_man<>1077)and(t_man<>1072)and
                (t_man<>1080)and(t_man<>27) then
                t_man:=tecla_oper;
                end;
                numero_pasos (t_man,repe,ext);
      end;
   until ((t_man=1059)or(t_man=1060)or
   (t_man=1072)or(t_man=1080)or(t_man=27));
   lim_pi;
   end;
end;

procedure pantalla_modo_man;
var
j,ext, cuenta_1,cuenta_2,cuenta_3,cuenta_4,
cuenta_5,cuenta_6:integer;
begin
        cuenta_1:=10;
        cuenta_2:=10;
        cuenta_3:=10;
        cuenta_4:=10;
        cuenta_5:=10;
        cuenta_6:=10;
        pantalla(4,5,76,21,0,15,0);
        mov(19,6,35,6,7,0,1,' FRECUENCIA ');
        mov(37,6,50,6,7,0,1,'  PASOS  ');
        mov(57,6,72,6,7,0,1,' COORDENADAS ');
        mov(6,8,15,8,4,15,1,' motor1 ');
        mov(6,10,15,10,7,0,1,' motor2 ');
        mov(6,12,15,12,7,0,1,' motor3 ');
        mov(6,14,15,14,7,0,1,' motor4 ');
        mov(6,16,15,16,7,0,1,' motor5 ');
        mov(6,18,15,18,7,0,1,' motor6 ');
        mov(6,20,15,20,1,15,1,' libre  ');
        mov(55,8,62,8,7,0,1,  ' x1 ');
        mov(55,10,62,10,7,0,1,' x2 ');
        mov(55,12,62,12,7,0,1,' x3 ');
        mov(55,14,62,14,7,0,1,' x4 ');
        mov(55,16,62,16,7,0,1,' x5 ');
        mov(55,18,62,18,7,0,1,' x6 ');
        ext:=0;
        repeat
        port[numport]:= ext;
        for j:=1 to 20000 do;
        coordenadas2(ext);
        inc(ext,16);
        until(ext=96);
        pantalla(37,8,45,8,4,15,0);
        write(' ',cuenta_1:5);
        pantalla(37,10,45,10,4,15,0);
```

```
        write(' ',cuenta_1:5);
        pantalla(37,12,45,12,4,15,0);
        write(' ',cuenta_1:5);
        pantalla(37,14,45,14,4,15,0);
        write(' ',cuenta_1:5);
        pantalla(37,16,45,16,4,15,0);
        write(' ',cuenta_1:5);
        pantalla(37,18,45,18,4,15,0);
        write(' ',cuenta_1:5);
end;

procedure menu_modo_man(tecla_man,color_ind:integer;
                        var cont_m,cont_a:integer);
begin
   if tecla_man=1080 then
      begin
      inc (cont_m);
      if cont_m=8 then cont_m:=1;
      end
   else if tecla_man=1072 then
      begin
      dec(cont_m);
      if cont_m=0 then cont_m:=7;
      end;
   if tecla_man=1080 then
      begin
      if cont_m=1 then cont_a:=7
      else cont_a:=cont_m-1;
      end
   else if tecla_man=1072 then
      begin
      if cont_m=7 then cont_a:=1
      else cont_a:=cont_m+1;
      end;
   case cont_a of
      1: mov(6,8,15,8,15,0,1,' motor1 ');
      2: mov(6,10,15,10,15,0,1,' motor2 ');
      3: mov(6,12,15,12,15,0,1,' motor3 ');
      4: mov(6,14,15,14,15,0,1,' motor4 ');
      5: mov(6,16,15,16,15,0,1,' motor5 ');
      6: mov(6,18,15,18,15,0,1,' motor6 ');
      7: mov(6,20,15,20,1,color_ind,1,' libre ');end;
   case cont_m of
      1: mov(6,8,15,8,4,15,1,' motor1 ');
      2: mov(6,10,15,10,4,15,1,' motor2 ');
      3: mov(6,12,15,12,4,15,1,' motor3 ');
      4: mov(6,14,15,14,4,15,1,' motor4 ');
      5: mov(6,16,15,16,4,15,1,' motor5 ');
      6: mov(6,18,15,18,4,15,1,' motor6 ');
      7: mov(6,20,15,20,4,color_ind,1,' libre');end;
end;
```

```
procedure modo_man;
type
    arreglo_manual=array [1..100] of string;
var
    tecla_man,t,i,cont_m,cont_a,
    cuenta_1,cuenta_2,cuenta_3,cuenta_4,
    cuenta_5,cuenta_6,indica1,color_ind,
    opcion_sec,cont_dat,ind2,
    i_dat,bandera_dat:integer;
    long,roo,ree,zeta,rot,cerrar:real;
    pd_manual:string30;
    d_manual:arreglo_manual;

begin
    with cod_op[1] do
       begin
       cuenta_1:=10;
       cuenta_2:=10;
       cuenta_3:=10;
       cuenta_4:=10;
       cuenta_5:=10;
       cuenta_6:=10;
       vel:=450;
       cont_a:=1;
       cont_m:=1;
       indica1:=1;
       bandera_dat:=0;
       i_dat:=1;
       color_ind:=15;
       pantalla_modo_man;
       port[numport]:=0;
       ext:=0;
       tecla_man:=1;
       repeat
           repeat
           menu_modo_man(tecla_man,color,cont_m,cont_a);
           tecla_man:=tecla_oper;
           pantalla(18,20,70,20,0,15,0);
           until (tecla_man=27)or(tecla_man=13)or
              (tecla_man=1075)or(tecla_man=1077);
              if (tecla_man<>27) then
                  begin
                  case cont_m of

                  1:        begin
                               ext:=0;
                               numero_pasos
                               (tecla,cuenta_1,ext);
                               paso_manual
                               (tecla_man,cuenta_1,
```

```
                1,ext,indica1);
                end;

2:          begin
                ext:=16;
                numero_pasos
                (tecla_man,cuenta_2,ext);
                paso_manual
                (tecla_man,cuenta_2,
                1,ext,indica1);
                end;

3:          begin
                ext:=32;
                numero_pasos
                (tecla_man,cuenta_3,ext);
                paso_manual
                (tecla_man,cuenta_3,
                1,ext,indica1);
                end;

4:          begin
                ext:=48;
                numero_pasos
                (tecla_man,cuenta_4,ext);
                paso_manual
                (tecla_man,cuenta_4,
                1,ext,indica1);
                end;

5:          begin
                ext:=64;
                numero_pasos
                (tecla_man,cuenta_5,ext);
                paso_manual
                (tecla_man,cuenta_5,
                1,ext,indica1);
                end;

6:          begin
                ext:=80;
                numero_pasos
                (tecla_man,cuenta_6,ext);
                paso_manual
                (tecla_man,cuenta_6,
                1,ext,indica1);
                end;

7:           begin
                    if indica1=1 then
                    begin
```

```
                                    indica1:=0;
                                    color_ind:=31;
                                    end
                                    else if indica1=0 then
                                    begin
                                    indica1:=1;
                                    color_ind:=15;
                                    end;
                                  end;
                    end;
                    end;
        until(tecla_man=27);
end;
pantalla(2,24,70,24,7,0,0);
end;

procedure mod_edit(var i1,i,pc,renp:integer;
                   var codigo:inf_dat;
                   var bandera:integer);
var
var_dat:pal;
pc2,val_tem:integer;
let1:cod;
                begin
                case i1 of
                  1082:       begin
                              pc2:=pc;
                              while(pc2>=i) do
                                 begin
                                 codigo[pc2+1]:=codigo[pc2];
                                 dec(pc2);
                                 end;
                              if pc<>0 then
                                 begin
                                 inc(pc2);
                                 inc(pc);
                                 end
                              else
                                 begin
                                 pc2:=1;
                                 pc:=1;
                                 end;
                              letrero(var_dat,
                               'INSTRUCCION A INSERTAR ?');
                              codigo[pc2]:=var_dat;
                              pantalla(12,8,37,18,7,0,0);
                              lis_ta(i-renp+1,pc,codigo);
                              val_tem:=0;
                              bandera:=1;
                              end;
                    13:       begin
```

```
                        letrero(var_dat,
                         'INSTRUCCION SOBREESCRIBIR?');
                        codigo[i]:=var_dat;
                        pantalla(12,8,37,18,7,0,0);
                        if pc = 0 then
                        pc:=1;
                        lis_ta(i-renp+1,pc,codigo);
                        val_tem:=0;
                        bandera:=1;
                        end;
                1083:   begin
                        if ((i=1)and(i=pc)or
                        (i=1)and(pc=0)) then
                        begin
                        pantalla(12,8,37,18,7,0,0);
                        pantalla(12,8,37,8,1,7,0);
                        codigo[1]:=' ';
                        pc:=0;
                        end
                        else
                        begin
                        if (i<>pc) then
                        begin
                        for pc2:= i to pc do
                        codigo[pc2]:=codigo[pc2+1];
                        end
                        else if (i=pc)and(renp<>1)then
                        begin
                        dec(i);
                        dec(renp);
                        end
                        else dec(i);
                        dec(pc);
                        pantalla(12,8,37,18,7,0,0);
                        lis_ta(i-renp+1,pc,codigo);
                        end;
                        val_tem:=0;
                        bandera:=1;
                        end;
            end;
            end;


procedure cambios;

var
codigo : inf_dat;
var_dat:string[20];
te_cla,h,pc,val2,indi,
pc2,renp,i,val1,i1,
result,band,bandera,
```

```
contenido,ind1:integer;
let1:cod;
begin
    ver_archivo(1,ind1);
    if(ind1=1)or(ind1=2) then
    begin
    pantalla(10,7,37,21,0,0,0);
    pantalla(11,6,38,20,7,0,0);
    pantalla(12,8,37,19,7,0,0);
    end;
    if (ind1=1) then
       begin
       pc2:=1;
       pc:=1;
       indi:=1;
       pantalla(44,10,80,10,7,0,1);
       write(' INSTRUCCION ');
       c_on;
       pantalla_lec(44,12,75,12,7,0);
       te_cla:=tecla_oper;
       while (te_cla<>27) do
          begin
          if (letra(te_cla)<>chr(20)) then
             begin
             write(letra(te_cla));
             readln(var_dat);
             codigo[pc2]:=concat(letra(te_cla),var_dat);
             if pc-indi = 11 then
             inc(indi);
             pantalla(12,8,37,19,7,0,0);
             c_off;
             lis_ta(indi,pc,codigo);
             inc(pc);
             inc(pc2);
             end;
          pantalla_lec(44,12,75,12,7,0);
          c_on;
          te_cla:=tecla_oper;
          end;
       c_off;
       dec(pc);
       for h:= 1 to pc do
       writeln(archa,codigo[h]);
       close (archa);
       end;
    if (ind1=2) then
      begin
       pc:=1;
       while not(eof(archa)) do
          begin
          readln(archa,var_dat);
```

```
    codigo[pc]:=var_dat;
    inc(pc);
    end;
dec(pc);
close(archa);
pantalla(12,8,37,19,7,0,0);
lis_ta(1,pc,codigo);
renp:=1;
i:=1;
mov_dat(12,8,37,8,1,15,0,i,codigo[i]);
bandera:=0;
pantalla(22,24,80,24,4,15,1);
write(' ESC:MENU ENTER:SOBRESCRIBIR
      INSERT:INSERTAR SUPR:BORRAR ');
textcolor(14);
gotoxy(2,1);
write('ESC:');
gotoxy(11,1);
write('ENTER:');
gotoxy(30,1);
write('INSERT:');
gotoxy(46,1);
write('SUPR:');
repeat
  val1:=tecla_oper;
  if(val1=1072)or(val1=1081)or(val1=1073)
  or(val1=1082)or(val1=1083)or
  (val1=1080)or(val1=13)then
  begin
  case val1 of
  1081:    begin
             if (i+10)<pc then
               begin
               i:= i+12-renp;
               pantalla(12,8,37,18,7,0,0);
               lis_ta(i,pc,codigo);
               i:=i+renp-1;
               end
             else if (i+11-renp)<pc then
               begin
               i:=i+1;
               pantalla(12,8,37,18,7,0,0);
               lis_ta(i,pc,codigo);
               if (pc-i) < renp then
               renp:=pc-i+1;
               i:=i+renp-1;
               end
             else  if (i<>pc) then
               begin
               mov_dat(12,renp+7,37,renp+7,
               7,0,0,i,codigo[i]);
```

```
                renp:= renp+pc-i;
                i:=pc;
                end;
          end;
1073:     begin
            if (i-10)>1 then
               begin
               i:= i-10-renp;
               if(i<=0) then
               i:=1;
               pantalla(12,8,37,18,7,0,0);
               lis_ta(i,pc,codigo);
               i:=i+renp-1;
               end
            else if (i-renp)>1 then
               begin
               pantalla(12,8,37,18,7,0,0);
               if (pc-i) < renp then
               renp:=pc-i+1;
               i:=1;
               lis_ta(i,pc,codigo);
               i:=i+renp-1;
               end
            else  if (i<>1) then
               begin
               mov_dat(12,renp+7,37,renp+7,
               7,0,0,i,codigo[i]);
               renp:= 1;
               i:=1;
               end;
          end;

1072:     begin
            if (renp>1)and(i>1) then
               begin
               mov_dat(12,renp+7,37,renp+7,
               7,0,0,i,codigo[i]);
               dec(i);
               dec(renp);
               end
            else if (i>1)and(renp=1) then
               begin
               dec(i);
               pantalla(12,8,37,18,7,0,0);
               lis_ta(i,pc,codigo);
               end;
            end;

1080:     begin
            if (renp<11)and(i<pc) then
               begin
```

```
                        mov_dat(12,renp+7,37,renp+7,
                        7,0,0,i,codigo[i]);
                        inc(i);
                        inc(renp);
                        end
                      else if (i<pc)and(renp=11) then
                        begin
                        pantalla(12,8,37,18,7,0,0);
                        inc(i);
                        lis_ta(i-10,pc,codigo);
                        end;
                      end;
 1082,1083,13:       mod_edit(val1,i,pc,renp,
                     codigo,bandera);
            end;
            mov_dat(12,renp+7,37,renp+7,
            1,15,0,i,codigo[i]);
            end;
       until (val1=27);
       if bandera=1 then
         begin
         assign(archa,nombre);
         rewrite(archa);
         for h:= 1 to pc do
         writeln(archa,codigo[h]);
         close (archa);
         end;
      end;
      pantalla(1,24,80,24,7,0,0);
end;

procedure pantalla_inicial;
begin
pantalla(1,1,80,25,7,0,0);
pantalla(2,2,79,2,1,7,0);
write('   EDITAR OPERAR MODO_MAN SALIR  ');
textcolor(14);
gotoxy(4,1);
write('E');
gotoxy(22,1);
write('O');
gotoxy(40,1);
write('M');
gotoxy(58,1);
write('S');
pantalla(2,4,79,22,1,7,0);
end;

procedure letra_ilu(col1,color:integer);
begin
   gotoxy(1,1);
```

```
   textcolor(color);
   case col1 of
        1: write('E');
        2: write('O');
        3: write('M');
        4: write('S');
   end;
end;

procedure menu;
var
cola,i,colp,col1,sal_b,opcion:integer;
begin
let[1]:='EDITAR';
let[2]:='OPERARA';
let[3]:='MODO_MAN';
let[4]:='SALIR';
pantalla_inicial;
colp:=5;
col1:=1;
mov(colp,2,colp+17,2,4,15,1,let[1]);
gotoxy(1,1);
textcolor(30);
write('E');
repeat
  pantalla(2,4,79,22,1,7,0);
  repeat
   repeat
      cola:=colp;
      val1:=tecla_oper;
      if (val1=1075)or(val1=1077)or(val1=13)
      or(val1=69)or(val1=79) then sal_b:=1
      else if (val1=77)or(val1=83)or(val1=101)or(val1=111)
      or(val1=109)or(val1=115) then sal_b:=1
      else sal_b:=0;
   until(sal_b=1);
   mov(cola,2,cola+17,2,1,7,0,let[col1]);
   letra_ilu(col1,14);
   case val1 of
   1075 :     begin
                if colp = 5 then
                   begin
                     colp:=59;
                     col1:=4;
                   end
                else
                    begin
                      dec(colp,18);
                      dec(col1);
                    end;
                end;
```

```
      1077 :      begin
                  if colp = 59 then
                    begin
                      colp:=5;
                      col1:=1;
                    end
                  else
                    begin
                      inc(colp,18);
                      inc(col1);
                    end;
                  end;
      69,101:     begin
                    colp:=5;
                    col1:=1;
                    val1:=13
                  end;
      79,111:     begin
                    colp:=23;
                    col1:=2;
                    val1:=13;
                  end;
      77,109:     begin
                    colp:=41;
                    col1:=3;
                    val1:=13;
                  end;
      83,115:     begin
                    colp:=59;
                    col1:=4;
                    val1:=13;
                  end;
         end;
      mov(colp,2,colp+17,2,4,15,1,let[col1]);
      letra_ilu(col1,30);
   until (val1=13);
   letra_ilu(col1,15);
   case col1 of
      1: cambios;
      2: rec_inst;
      3: modo_man;
      4: begin
         mov(10,24,60,24,4,31,1,' DESEA SALIR ? (S/N) ');
         repeat
         opcion := tecla_oper;
         until(opcion=115)or(opcion=83)or(opcion=110)
         or(opcion=78);
         pantalla(10,24,60,24,7,0,0);
         if (opcion=115)or(opcion=83) then val1:=24
         else val1:=10;
         end;
```

```
end;
mov(colp,2,colp+17,2,4,15,1,let[col1]);
letra_ilu(col1,30);
until (val1=24);
end;

begin
  textbackground(0);
  clrscr;
  pantalla_inicial;
  menu;
  c_on;
  pantalla(1,1,80,25,0,15,0);
  lim_pi;
  end.
```

BIBLIOGRAFIA

P. P. Acarnley Stepping: *"Motors: a Guide to Modern Theory and Practice"*, Second Edition, Peter Peregrinus Ltd,1984

Ramakant A. Gayakward: *"Op-Amps and Linear Integrated Circuits"*, Second Edition, Prentice-Hall, 1988.

Donald L. Schilling, Charles Belove: *"Electronic Circuits"*, Third Edition, Mc Graw Hill,1989

Boylestad Nashelsky: *" Electrónica, Teoría de Circuitos"*, Prentice-Hall, Primera Edicion,1983.

Francis W. Sears, Mark W. Zemansky: *"Fisica General"*, Quinta Edición, Aguilar,1981

M. Morris Mano: *"Diseño Digital"*, Primera Edición, Prentice Hall,1987

Databook: *"Linear"*, National Semiconductor Corporation, 1983.

Databook: *"CMOS Logic"*, National Semiconductor Corporation, 1988.

Databook: *"LS/S/TTL"*, National Semiconductor Corporation, 1987.