Agradecimientos

Le doy gracias a Dios por darme unos Padres Maravillosos, a los que nunca podré pagarles todos los esfuerzos y sacrificios que han hecho por mi. Por poner en mi camino a Elizabeth C. B. Que me ha ayudado y apoyado en todo momento.

Le agradezco a mis Tíos y familiares por todo el apoyo que me han dado.

Agradezco sinceramente a mi asesor M. C. Felipe Santiago Espinosa por la paciencia que tuvo conmigo en la realización de este trabajo, a todos los profesores que me han dado apoyo, consejos y enseñanza, a los señores del taller de metales y maderas por la gran ayuda que me han brindado.

Gracias Dios mío por darme fuerza de voluntad, conocimientos, optimismo y paciencia en los momentos difíciles que he vivido.

Les agradezco a todas las personas que de alguna u otra forma me dieron ánimos y palabras de apoyo para terminar este trabajo.

A la memoria de mis abuelitos fallecidos Fidencio Vázquez y Marcelo Gallardo. ††

Sinceramente

Jorge L. Vázquez Gallardo

CAPITULO 1

INTRODUCCION

1.1 ANTECEDENTES

Los circuitos impresos forman parte importante en la electrónica, ya que sobre ellos se ensamblan o montan la mayoría de los componentes, como son circuitos integrados, resistencias, diodos, transistores y demás componentes de mediano y pequeño tamaño. Con los circuitos impresos se reemplazan los métodos antiguos de armado sobre chasises grandes, donde las conexiones se hacían por medio de una gran cantidad de cables, lo que gastaba mucho tiempo y espacio, además de dificultar el mantenimiento.

Los circuitos impresos prestan una doble función: como soporte de los componentes y como conexión para formar el alambrado del circuito. El circuito impreso es importante en la electrónica, ya que sin él no se habría alcanzado el desarrollo de la tecnología electrónica a los niveles que hoy ha llegado.

A continuación se describen los tres métodos comúnmente utilizados por los alumnos de esta institución, para elaborar circuitos impresos:

El primer método requiere un plumón permanente ó calcomanías (rapid circuit), éste tiene la ventaja de requerir escasos recursos aunque la forma de elaboración es demasiado rudimentaria, ya que manualmente se realiza todo el diseño por lo que su elaboración es rápida pero de baja calidad, mostrando muchas deficiencias debido a que no es un método preciso y depende directamente de la persona que lo realiza.

El segundo método puede desarrollarse en un taller de serigrafía. En este el diseño se realiza con ayuda de software (orcad, pcb, protel, etc.), hecho el diseño se imprime en un acetato, se pasa a la malla y se hace el pintado de la placa. El resultado final depende de la

experiencia de la persona que la realice ya que el pintado puede tener manchas no deseadas, rayas entre pistas, falta de pintura en las pistas, etc.

El tercer método está basado en el uso del negativo de diseño y foto resina. En este, al igual que el descrito anteriormente, el diseño se realiza con software, después en un laboratorio de fotografía se obtiene el negativo del diseño, se le aplica la foto resina a la placa y con el negativo encontrado se expone a la luz. Aquí se mejora el resultado, pero las desventajas principales son: es necesario un laboratorio de fotografía para la obtención de negativos, la foto resina es difícil de conseguir en algunos lugares y tiene un alto costo.

Es por eso que en el presente trabajo se presenta el diseño de una herramienta con la cual se espera alcanzar un método para la realización de circuitos impresos rápido y fácil, un método con pocas etapas pero eficiente.

1.2 OBJETIVOS DE LA TESIS

El objetivo de este trabajo es la realización de un Sistema para el desarrollo automatizado de circuitos impresos, el cual es un sistema de control con las siguientes características:

- Es un sistema de Lazo Abierto
- El tipo de control es digital y es realizado desde una PC
- Controla la posición de dos ejes manejados por motores de pasos y un tercer eje manejado por un motor de corriente directa.
- Por medio de software se define el comportamiento del sistema

El Proyecto está constituido por tres partes fundamentales:

1. El Graficador (parte electromecánica), directamente acoplada a los ejes para las diferentes posiciones.
2. La Tarjeta Controladora, funciona como interfaz para el desempeño eficiente del graficador.
3. Un programa para darle información a la tarjeta controladora e interactuar directamente con el usuario.

En este contexto se planteó el desarrollo de la presente tesis; como se observa se realizaron trabajos en hardware, software y acoplamiento mecánico. El desarrollo del software se efectuó en lenguaje C.

Con todo esto se pretende conseguir un sistema automático con mejores características que permita:

- Minimizar los costos para la fabricación de circuitos impresos
- Facilitar la elaboración de circuitos en serie
- Poder realizar circuitos impresos sin previa experiencia.

Como se muestra en la figura 1.1 el sistema consta de tres etapas para las cuales se dará una breve descripción a continuación.

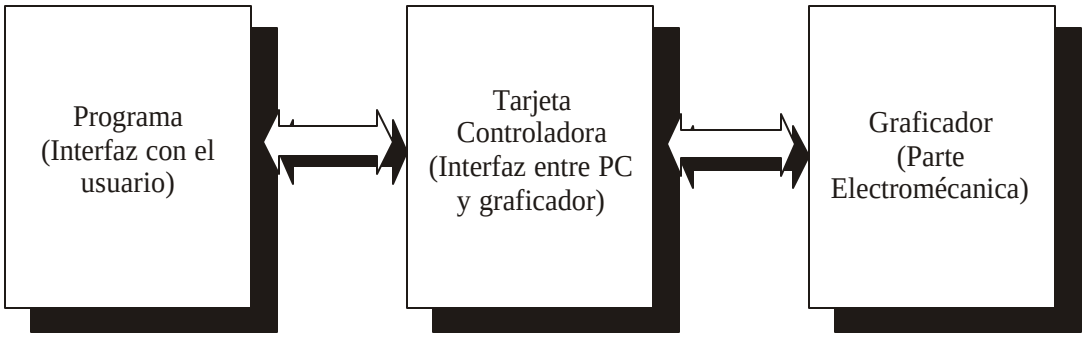


Fig. 1.1 Generalidades del proyecto.

1.2.1 El Graficador (parte electromecánica)

Esta etapa es la más importante ya que dependiendo de la precisión que se obtenga, los resultados serán aceptables y el sistema cumplirá todos los objetivos propuestos. Aquí es donde se realizará la impresión del circuito impreso sobre la placa de cobre.

La parte electromecánica esta compuesta de:

- Dos motores de paso para el movimiento de los ejes ***X*** y ***Y***.
- Un motor de corriente directa para manejar el movimiento en ***Z***.
- Una etapa de potencia para que sea posible el funcionamiento de los motores.
- Un sistema de engranes para que el movimiento de los motores sea preciso y eficiente.

1.2.2 La Tarjeta Controladora

Esta etapa va a estar conformada por una tarjeta basada en el bus de expansión ISA (*Industry Standard Architecture,* Arquitectura Estándar para la Fabricación de Prototipos) conectada de forma directa a la PC, para que funcione como interfaz entre la PC y el graficador. La tarjeta contendrá:

- Un manejador de Puertos para manipular el graficador.
- Un decodificador para decodificar la dirección a utilizar de la PC.

1.2.3 El programa (o software)

El programa sirve como medio de comunicación o interfaz entre usuario y computadora. También es el encargado de suministrarle información a la tarjeta controladora para que ésta interactúe directamente con el graficador.

La programación será realizada en el lenguaje de programación C, bajo un estilo de programación estructurada, en modo DOS, para que el sistema pueda ser manejado en computadoras cuyos recursos sean escasos.

CAPITULO 2

PARTE MECÁNICA O GRAFICADOR

2.1 INTRODUCCIÓN

En este capítulo se describe la forma como está constituida la parte mecánica o graficador. El graficador se encuentra formado por los siguientes bloques:

1. **Dos sistemas de engranes:** Para mejorar la precisión en el control a través de los ejes X y Y.
2. **Etapa de potencia:** Encargada de suministrar la corriente necesaria para el manejo de los motores de pasos y el motor de corriente directa.
3. **Dos motores de pasos:** Para el control de la posición en los ejes X y Y.
4. **Un motor de corriente directa:** Para el control de la posición en el eje Z.

En la figura 2.1 se muestra un diagrama a bloques del graficador con sus diferentes elementos.

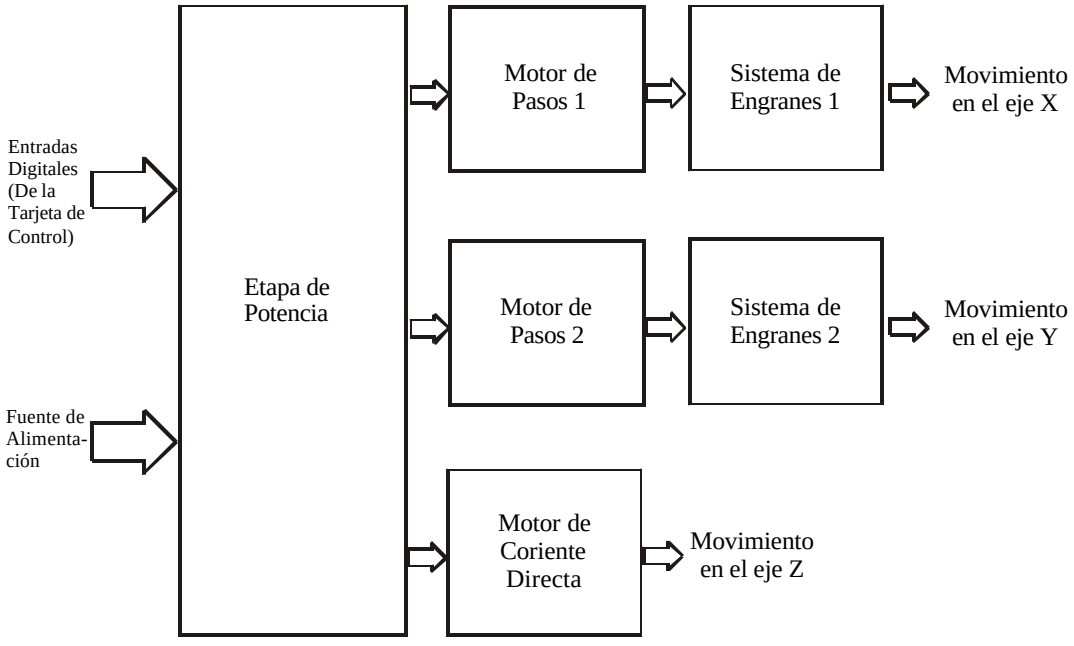


Fig 2.1 Diagrama a bloques del graficador.

2.2 CONTROL DE LAZO CERRADO Y LAZO ABIERTO

El sistema desarrollado es un sistema de control de lazo abierto definido a continuación, también se define un sistema de lazo cerrado para un mejor entendimiento y diferenciación entre estos.

Sistemas de Control en Lazo Abierto[7]: Los sistemas en los cuales la salida no afecta la acción de control se denominan *Sistemas de Control en Lazo Abierto*. En otras palabras, en un sistema de control en lazo abierto no se mide la salida ni se retroalimenta para compararla con la entrada. Un ejemplo práctico se encuentra en una lavadora. El remojo, el lavado y el enjuague en la lavadora operan con una base de tiempo. La máquina no mide la señal de salida, que es la limpieza de la ropa.

En cualquier sistema de control de lazo abierto, la salida no se compara con la entrada de referencia. Por tanto, a cada entrada corresponde una salida; como resultado, la precisión del sistema depende de la calibración. Ante la presencia de perturbaciones, un sistema de control en lazo abierto presenta fallas al realizar la tarea deseada. Es evidente que estos sistemas no son de control retroalimentado. Por lo tanto cualquier sistema de control que opere con una base de tiempo es de lazo abierto. Por ejemplo, el control del tránsito mediante señales operadas con una base de tiempo funciona también en lazo abierto, en la fig. 2.2 se muestra un diagrama a bloques de un sistema de lazo abierto.

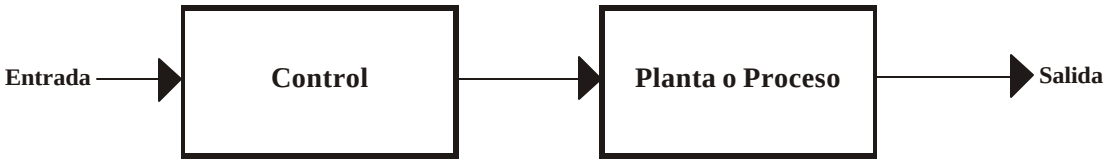


Fig. 2.2 Sistema de control de lazo abierto.

Sistemas de Control en Lazo Cerrado[7]: Los sistemas de control retroalimentados se denominan también *Sistemas de Control en Lazo Cerrado*. En la práctica, los términos control retroalimentado y control en lazo cerrado se usan indistintamente. En un sistema de control en lazo cerrado, se alimenta al controlador la señal de error de actuación, que es la diferencia entre la señal de entrada y la señal de retroalimentación (que puede ser la señal de salida misma o una función de la señal de salida y sus derivadas y/o integrales), a fin de reducir el error y llevar la salida del sistema a un valor conveniente. El término control en lazo cerrado siempre implica el uso de una acción de control retroalimentado para reducir el error del sistema. En la fig. 2.3 se muestra un diagrama a bloques de un sistema de lazo cerrado.

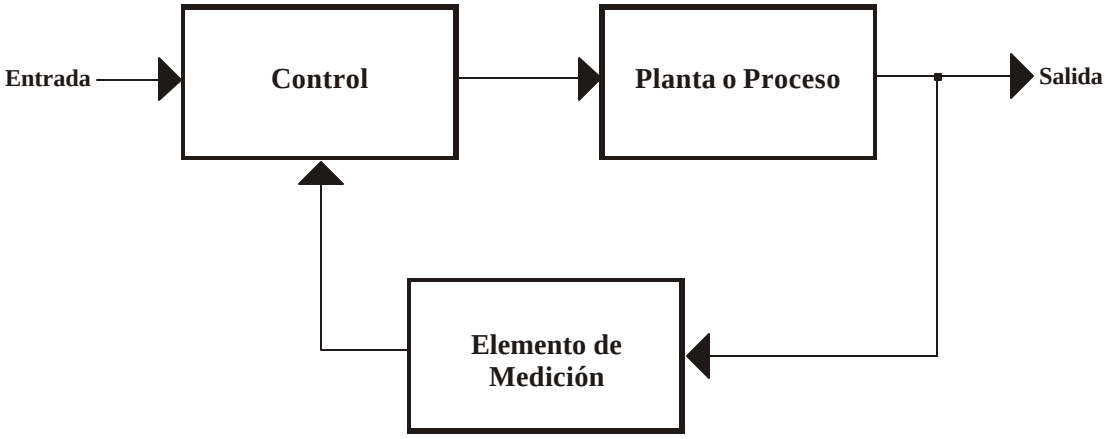


Fig. 2.3 Sistema de control de lazo cerrado.

2.3 SISTEMA DE ENGRANES

Se utilizan engranes en este sistema para obtener mayor fuerza en el movimiento de los motores, así como también para aumentar la precisión al reducir la distancia recorrida entre paso y paso, aumentando de esta forma la eficiencia y el funcionamiento total del sistema.

Esto se logra por medio de el acoplamiento de varios engranes los cuales al interconectarse entre si hacen que las distancias y la velocidad entre cada engrane varíe.

En la figura 2.4 se muestra un arreglo simple basado en dos engranes, el engrane A es mas pequeño que el engrane B, y tomando en cuenta la siguiente relación:

$$\frac{q_B}{q_A} = \frac{N_A}{N_B} \quad (2.1)$$

Donde N es el número de dientes y q el ángulo de desplazamiento, como se muestra en la figura 2.4. El engrane A con 12 dientes (N_A) y el engrane B con 48 dientes (N_B), si el engrane A gira 360°(q_A), el engrane B solo girará 90°(q_B). Suponiendo que el engrane A está manejado por un motor que produce un movimiento de 1° por paso, se requiere de 360 pasos para una vuelta. Al conectar el engrane mayor, con el mismo motor se requerirá de 1440 pasos para el mismo desplazamiento mejorando el movimiento a 0.25° por paso.

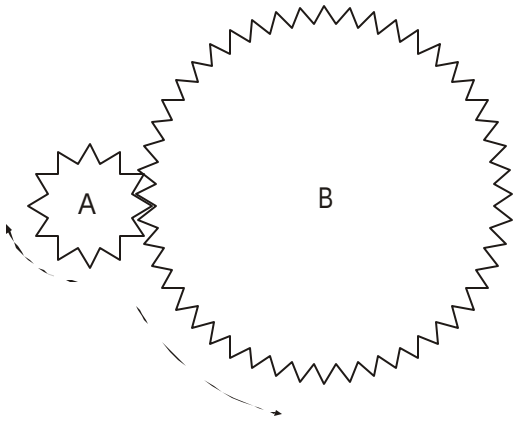


Figura 2.4 Un arreglo de dos engranes.

Por lo cual, entre más pequeño sea un engrane con respecto al otro, aunque el más pequeño girará mas vueltas a una velocidad más rápida que el engrane más grande, la distancia entre pasos se reduce. Otro aspecto importante es el sentido en que giran los engranes ya que estos tienen que girar en forma opuesta uno del otro. Cabe mencionar que los engranes también multiplican la fuerza mecánica producida por el motor.

2.4 ETAPA DE POTENCIA

El objetivo de la etapa de potencia es aumentar la corriente reforzando a las señales provenientes de la tarjeta de control, para esto se utilizan transistores darlington con los que se obtiene la corriente necesaria para el funcionamiento de los motores. Ya que la configuración darlington es una de las más eficientes para amplificar la corriente.

El Transistor Darlington o conexión darlington[6]. Es una conexión muy popular de dos transistores de unión bipolar para funcionar como un solo transistor con una ”superbeta”, la conexión que se ilustra en la figura 2.5 muestra esto. La principal característica de la conexión Darlington es que el transistor actúa como una sola unidad con ganancia de corriente que es producto de las ganancias de corriente de dos transistores por separado. Si la conexión se hace utilizando dos transistores individuales con ganancia de corriente β_1 y β_2, la ganancia resultante para la configuración es

$$\beta_D = \beta_1 \beta_2 \qquad (2.2)$$

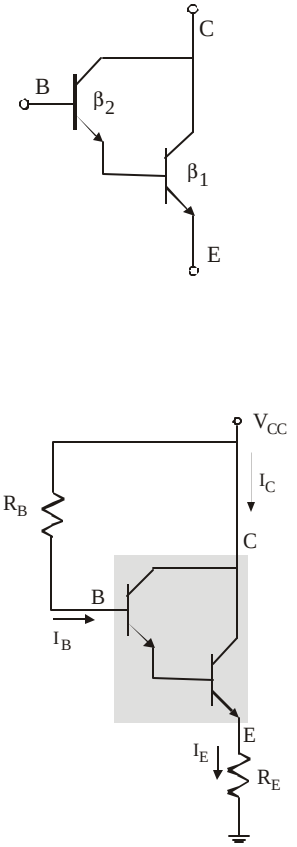


Figura 2.5 Conexión del Transistor Darlington.

Figura 2.6 Circuito Básico de Polarización Darlington.

En la figura 2.6 se muestra la forma básica para la polarización de un transistor darlington con muy alta ganancia de corriente, β_D. La corriente de base puede calcularse a partir de:

$$I_B = \frac{V_{CC} - V_{BE}}{R_B + ß_D R_E} \qquad (2.3)$$

Esta ecuación es la misma que la utilizada para los transistores comunes, sólo que aquí el valor de β_D es mucho mayor y el valor de V_{BE} es más grande. La corriente de emisor es por lo tanto

$$I_E = (ß_D + 1)I_B \approx ß_D R_E \tag{2.4}$$

Los voltajes en cd son

$$V_E = I_E R_E \tag{2.5}$$

$$V_B = V_E + V_{BE} \tag{2.6}$$

Ya definido anteriormente lo que es un transistor darlington o conexión darlington se explica como se utiliza en el graficador. Lo que se hace en este caso es conectar un transistor directamente a cada una de las fases del motor suministrando así la corriente necesaria para el movimiento de este, en la siguiente sección se define la forma en que se realiza el control de los motores de pasos. En la figura 2.7 se muestra la conexión utilizada

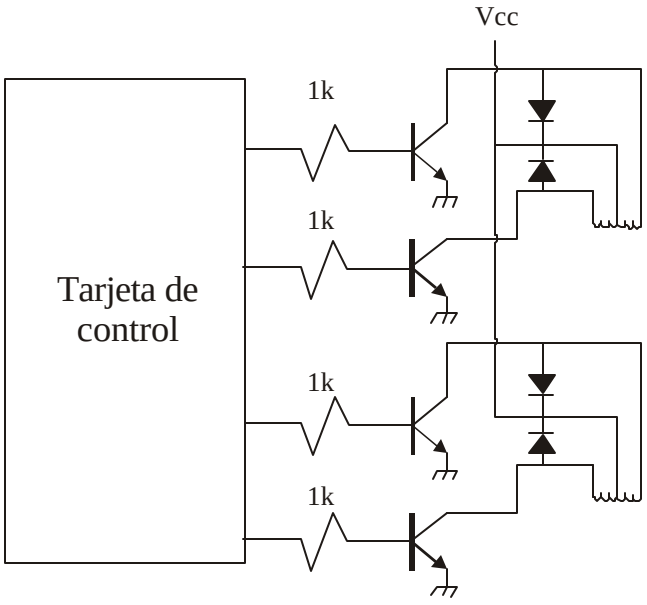


Figura 2.7 Conexión de la etapa de potencia para los motores de paso, los transistores en el esquema son TIP120. El bloque con la leyenda Tarjeta de Control se explica en el siguiente capitulo.

Además de los motores de pasos fue necesario usar un motor de C. D. Para subir o bajar al plumón que trabajará sobre la placa. La corriente requerida por este motor se obtiene de una configuración de transistores conocida como puente H, la cual se muestra en la figura 2.8.

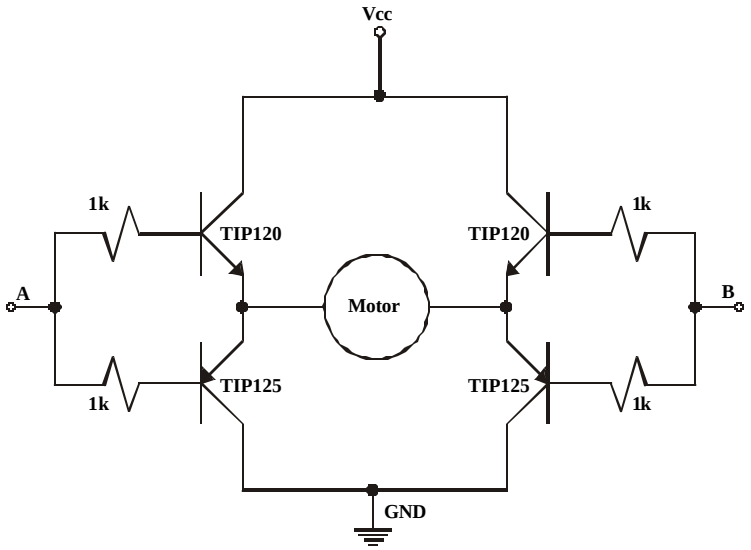


Fig. 2.8 Etapa de potencia para controlar el movimiento en el motor de corriente directa de acuerdo a las entradas A y B.

2.5 MOTOR DE CORRIENTE DIRECTA

Los dispositivos rotatorios de conversión de energía electromecánica son conocidos como *máquinas rotatorias*. Están clasificados como *máquinas de corriente directa* si sus salidas son en corriente directa o si la energía de entrada a la máquina proviene de una fuente de corriente directa. Se llaman *máquinas de corriente alterna* si su salida es periódica o si la energía primaria de entrada proviene de una fuente de corriente alterna.

Una máquina rotatoria se llama *generador* si convierte energía mecánica en eléctrica y *motor* si la energía de tipo eléctrica la convierte en mecánica. En principio, una máquina puede usarse indistintamente como generador o motor, pero consideraciones prácticas de diseño favorecen su uso en uno de los dos tipos [7].

El motor de corriente directa contiene imanes permanentes que están magnetizados radialmente. El flujo magnético f surge de la cara del polo sur hacia el polo norte como se muestra en la figura 2.9. La trayectoria de flujo magnético se completa a través del armazón de acero del motor. Y al aplicársele un voltaje en V_A las escobillas magnetizan el rotor de forma que cada sección esta alternada entre norte y sur haciendo que se produzca un movimiento de atracción o rechazo entre los polos del rotor y los polos del estator generando un movimiento circular mecánico [2].

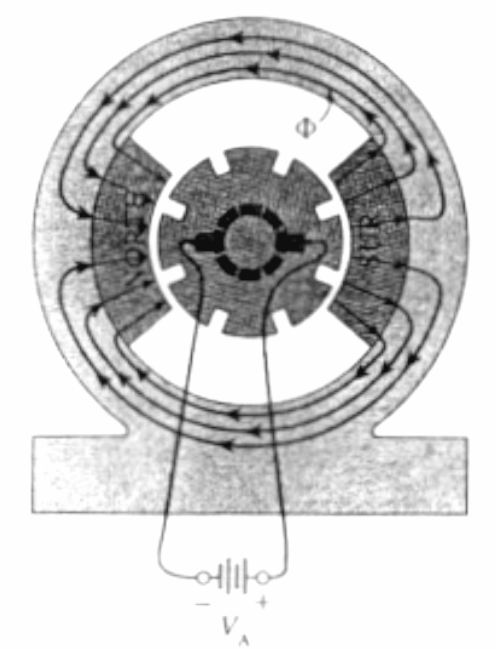


Figura 2.9 Cortes transversales de un motor convencional de imán permanente con dos polos magnetizados radialmente.

Este movimiento mecánico hace que a su vez las escobillas cambien la magnetización de las secciones del rotor y por lo tanto cambien sus polos, haciendo que el movimiento circular continúe periódicamente. Este movimiento depende directamente del voltaje V_A aplicado. Cabe mencionar que si se invierte la polaridad del voltaje, el movimiento circular cambiará de sentido.

El motor de corriente directa se utiliza para controlar el movimiento del eje Z, la dirección del movimiento depende de un código binario de dos bits, dependiendo de su valor el motor girará hacia la izquierda o hacia la derecha y de acuerdo al giro que

el motor realice la pluma subirá o bajará. En la tabla 2.1 se muestra el código y el movimiento que se realiza.

POSICIÓN DE LA PLUMA	Código binario en las entradas A y B de la Figura 2.6
Abajo	01
Arriba	10

Tabla 2.1 Código utilizado para el movimiento de la pluma.

2.5 MOTORES PASO A PASO [2]

Los motores paso a paso forman parte de los motores llamados ***motores de conmutación electrónica***. Son los más apropiados para la rotación continua de velocidad ajustable.

Los motores paso a paso son diferentes de los demás motores de cd; no tienen escobillas ni conmutador mecánico. En su lugar, la acción de conmutación necesaria para la función de motor de cd es lograda por transistores externos. Es más, el rotor no tiene devanado de armadura. Simplemente es una colección de imanes permanentes salientes, como se muestra en la figura 2.10.

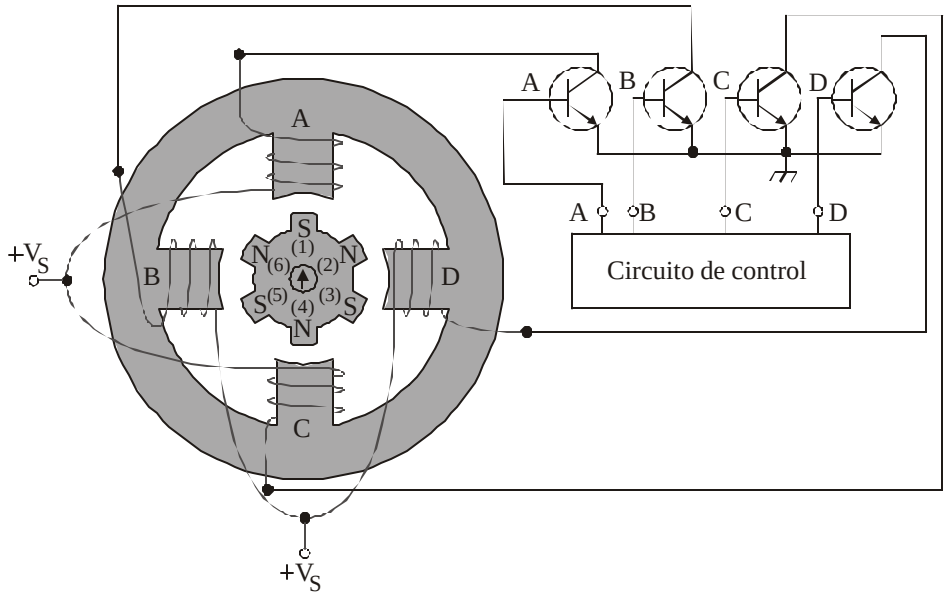


Figura 2.10 Con cuatro polos de estator y seis polos de rotor, este motor paso a paso tiene un ángulo de paso natural de 30°. El rotor se muestra en la posición de 0° (flecha de posición imaginaria apuntando hacia arriba, a la posición de las 12 en punto).

Los cuatro devanados de los polos del estator y sus transistores controladores se identifican con las etiquetas A, B, C y D. Cuando el circuito de control enciende un transistor, hay un flujo de corriente desde la fuente de alimentación de cd ($+V_s$), que pasa por el devanado donde el transistor se encendió y a través del mismo transistor a tierra. Cada devanado está enrollado de tal manera que cuando sea energizado su polo se vuelve norte magnético. Su flujo emerge de la cara del polo, pasa a través del rotor, entonces completa su trayectoria entrando en la cara del polo directamente opuesto a él. Por ejemplo, si el transistor A energiza al polo A de la figura 2.10, el flujo creado por ese polo completa su trayectoria a través del polo C y a través del armazón del motor. Por tanto, C automáticamente se vuelve un polo sur, aun cuando su devanado no lleva corriente.

En la figura 2.10 los polos permanentes del rotor están etiquetados del 1 a 6, siendo sur los polos 1, 3 y 5. Los polos con los números 2, 4 y 6, son norte.

El principio de operación de un motor paso a paso de imán permanente es el siguiente: el polo del estator energizado que se vuelve norte magnético activo atrae el polo sur más cercano del rotor para alinearlo con él. Esta acción de producción de par es ayudada por el polo sur pasivo del estator (del lado opuesto del estator), atrayendo el rotor nor te opuesto para alinearlo con él.

Por ejemplo, en la figura 2.10, si el transistor A es encendido, el polo A del estator que está en la posición de las 12 en punto es el norte activo. En el instante mostrado, ya ha atraído el polo sur 1 del rotor para alinearlo con él.

También, el polo C del estator que apunta hacia las 6 en punto, es el sur pasivo. Ha atraído el polo norte 4 del rotor para alinearlo con él.

Definamos la posición del rotor mostrada en la figura 2.10 como la posición de 0°. Se muestra una flecha imaginaria de posición en el eje que apunta hacia arriba indicando las 12 en punto. A medida que gira el eje del rotor, podemos describir su nueva posición dando la dirección en la que apunta la flecha imaginaria.

Cuando el circuito de control apaga al transistor A y enciende simultáneamente el transistor B. El polo B del estator se convierte en el norte activo, el polo D del estator se vuelve el sur pasivo. Los polos A y C se vuelve neutrales, se desmagnetizan. El polo B del estator atrae el polo sur 5 del rotor y el polo D del estator atrae el polo norte 2 del rotor. El rotor se mueve en dirección de las manecillas del reloj 30°, por lo que los polos del rotor se alinean con los polos del estator. Se dice que el motor toma "un paso" de 30°. La flecha imaginaria de posición apunta hacia la una en punto.

Una vez que se ha dado el paso de 30° el controlador puede apagar el transistor B y seguir la secuencia que se registra en la tabla 2.2, pasando del segundo renglón al tercer renglón y así sucesivamente.

POSICIÓN DEL EJE (GRADOS)	TRANSISTOR **ENCENDIDO**
0	A
30	B
60	C
90	D
120	A
150	B
180	C
210	D
240	A
270	B
300	C
330	D
360	A

TABLA 2.2 Secuencia de conmutación de los transistores para tomar pasos completos en la dirección de las manecillas del reloj.

El circuito de control apaga el transistor B y simultáneamente enciende el transistor C. Esto hace que el polo C del estator se vuelva el norte y que el polo A se vuelva sur pasivo. El polo sur 3 está apenas a 30° de distancia en este momento, por lo

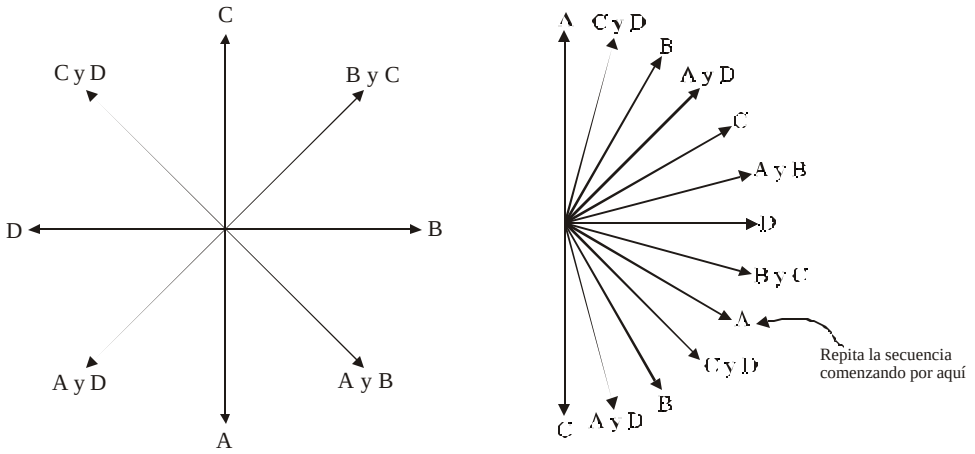
que se mueve para alinearse con él. El motor ha tomado otro paso de 30° en dirección de las manecillas del reloj, como se indica en la tabla 2.2. La flecha imaginaria de posición apunta, a 60° de su posición inicial. Y así continuando con el controlador disparando los transistores en la secuencia repetida ABCD. Deberá seguir los pasos del motor durante una rotación de 360°, hasta la parte inferior de la tabla 2.2. Es importante notar qué el polo del estator se ha vuelto norte y que el polo sur del rotor está a 30° de él.

POSICIÓN DEL EJE (GRADOS)	TRANSISTOR **ENCENDIDO**
0	A
-30	D
-60	C
-90	B
-120	A
-150	D
-180	C
-210	B

TABLA 2.3 Secuencia de conmutación de transistores para lograr que el motor de la figura 2.8 tome pasos completos en dirección contraria a las manecillas del reloj.

Para que un motor paso a paso realice giros en dirección contraria a las manecillas del reloj, la forma de hacerlo es simple, solo se requiere que el circuito de control encienda los transistores de conmutación en la secuencia inversa, DCBA. Esto se muestra en la tabla 2.3. Comenzando por la posición de 0°, la posición inicial que se muestra en la figura 2.10 donde el circuito de control ha encendido a A, el controlador primero conmuta al transistor D. En la figura 2.10, esto hace que el polo sur 3 del rotor se mueva 30° en dirección contraria a las manecillas del reloj para alinearse con el polo norte D del estator y, por supuesto, alineando el polo norte 6 del rotor con el polo sur B pasivo del estator. Se sigue la rotación en dirección contraria a las manecillas del reloj del motor paso a paso haciendo referencia a la tabla 2.3 y a la figura 2.10. Esta secuencia es similar solo que el motor se mueve en sentido inverso.

Medios pasos. Es posible conseguir que el motor de la figura 2.10 avance pasos de 15°, llamados medios pasos. La secuencia de conmutación de los transistores se da en la tabla 2.4. Al comparar la figura 2.11 (a), muestra la dirección del flujo magnético del estator del motor para cada fila de la tabla 2.4. La figura 2.11 (b) muestra los grados que gira el motor.



(a) (b)

FIGURA 2.11 (a) Dirección del flujo neto del estator para cada posibilidad de conmutación de transistores (cada renglón) de la tabla 2.4. (b) Posición de la flecha imaginaria del eje para cada fila de la tabla 2.3.

POSICIÓN DEL EJE (GRADOS)	TRANSISTOR **ENCENDIDO**
0	A
15	C y D
30	B
45	A y D
60	C
75	A y B
90	D
105	B y C
120	A

TABLA 2.4 Secuencia de conmutación de los transistores para lograr que el motor de la figura 2.10 tome medios pasos(de 15 °) en la dirección de las manecillas del reloj.

Los motores que se utilizan en este trabajo son de 7.5° entre paso y paso y la secuencia de conmutación de los transistores se muestra en la tabla 2.5.

POSICIÓN DEL EJE (GRADOS)	TRANSISTOR ENCENDIDO
0	A y C
7.5	A
15	A y D
22.5	D
30	C
37.5	B y D
45	B
52.5	B y C
60	C
67.5	A y C

TABLA 2.5 Secuencia de conmutación de los transistores para lograr que el motor utilizado en el sistema tome pasos de 7.5° en dirección de las manecillas del reloj.

2.6 UNIENDO LAS PARTES

En la figura 2.1 se mostró un diagrama a bloques del graficador y a lo largo del capítulo se han descrito las partes más importantes que lo conforman, en el apéndice A se muestran los diagramas del circuito de potencia, sin embargo también fue necesario el uso de otros elementos entre los que destacan:

- Base de lámina que sostiene a los motores y es el soporte de los demás elementos.
- Unos rodillos que hacen que los engranes puedan mover la placa de cobre.
- Banda dentada que permite mover a la pluma a lo largo del eje transversal.
- Resortes y ejes metálicos para mantener a la placa en movimiento.

El resultado final se muestra en la figura 2.12, diferentes vistas del graficador.

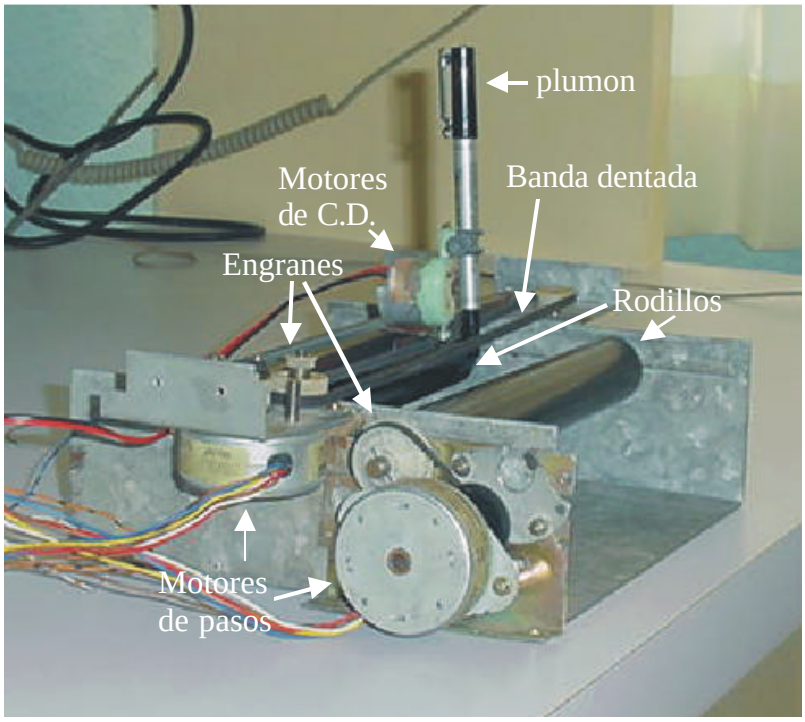


Figura 2.12(a) Vista lateral del graficador.

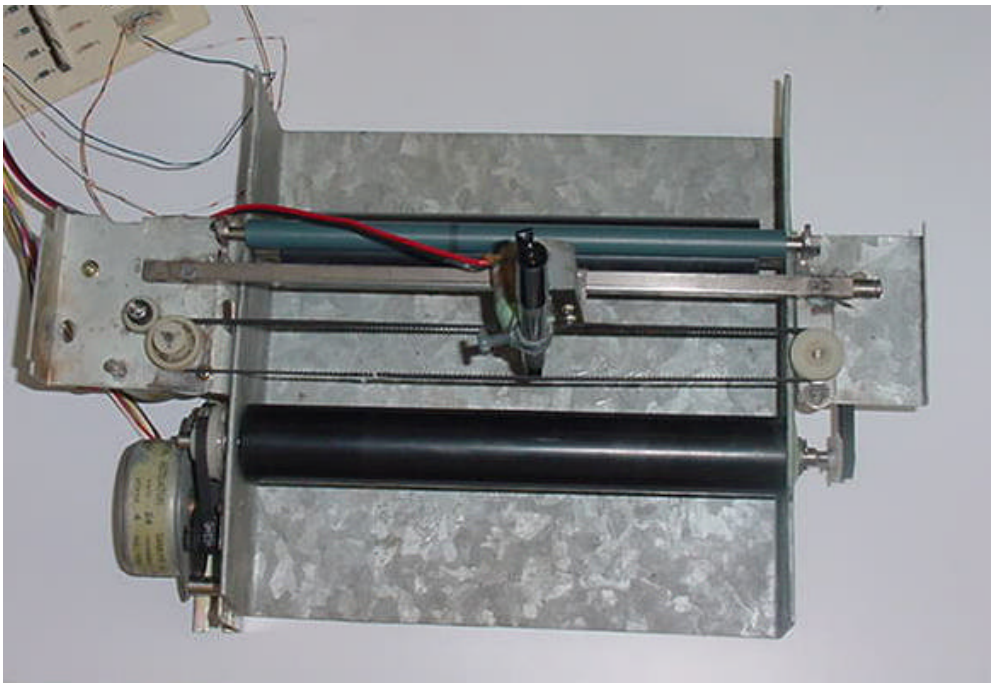
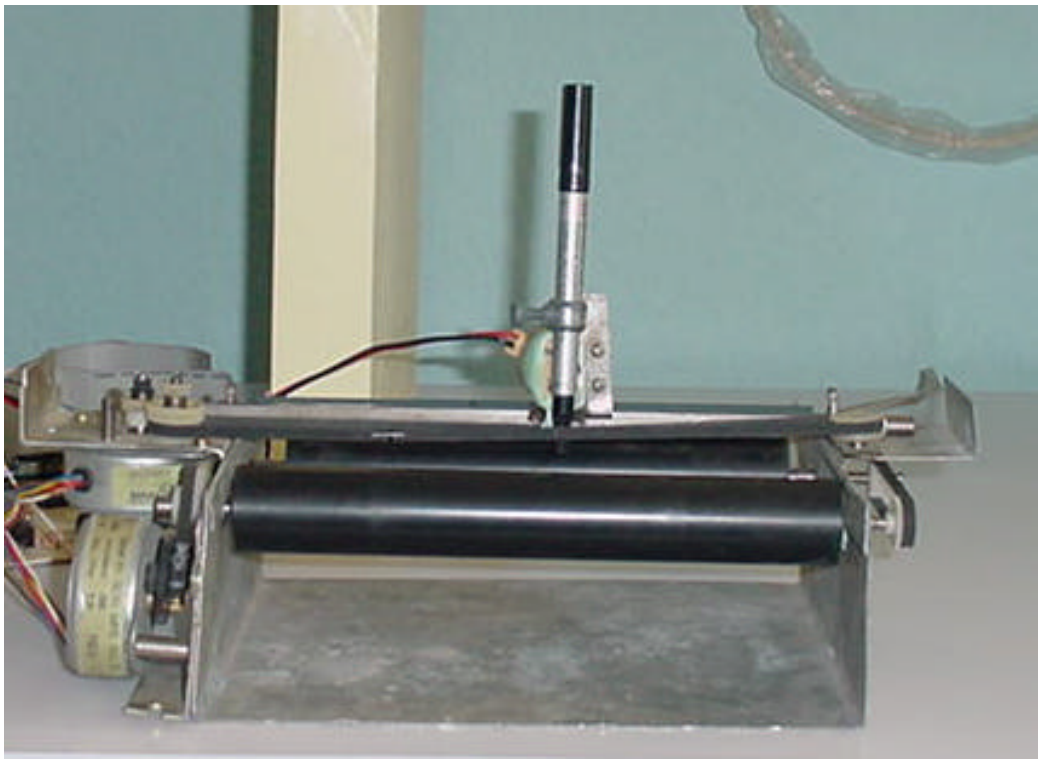


Figura 2.12(b) Vista aérea del graficador.



(c)

Fig. 2.12 (c) Vista frontal del graficador.

CAPITULO 3

LA TARJETA CONTROLADORA

3.1 INTRODUCCIÓN

En esta sección se describe la forma en que esta constituida la tarjeta controladora que es el medio con el cual se comunica el programa y el graficador o parte mecánica. Para lograr esto la tarjeta contiene:

1. **Un bus ISA (*Industry Standard Architecture*):** Con el cual se tienen salidas o entradas de datos de la PC, además de voltajes y tierra.
2. **Un manejador de puertos:** El 8255 es un PPI (Interfaz Periférica Programable), se utiliza para el manejo de los datos que se van a mandar al graficador.
3. **Un decodificador de direcciones:** Para utilizar el mínimo de componentes, se utilizó la GAL22V10.

3.2 EL BUS ISA DE LA PC DE IBM[4]

El bus ISA de la PC de IBM es una extensión del bus del microprocesador. Está demultiplexado y enriquecido por señales adicionales que ayudan al acceso directo a memoria (DMA, Direct Memory Access), interrupciones y otras aplicaciones. Todas las señales son niveles TTL (Lógica Transistor - Transitor), y también tiene pines de Vcc y tierra. La figura 3.1 muestra los 62 pines del bus ISA de IBM PC. Enseguida se describen con detalle las diferentes señales que lo integran, todas las señales se activan en alto a menos que se especifique lo contrario.

3.2.1 Las Direcciones A0 a A19

Estos 20 pines conforman las líneas de dirección de la memoria de la PC. A0 es el bit menos significativo (LSB) y A19 es el bit más significativo (MSB).

Estas líneas son manejadas para accesar al procesador o para controlar el acceso directo a memoria.

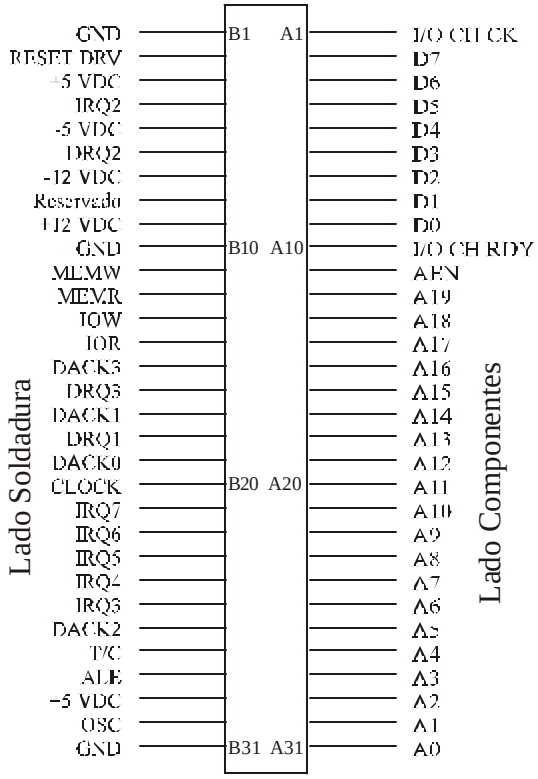


Figura 3.1 Sistema del bus ISA

3.2.2 Los Datos D0 a D7

Estos 8 pines son un bus de datos bidireccional, D0 es el bit menos significativo y D7 es el bit más significativo. Durante la inicialización en su ciclo de escritura, el procesador provee los datos sobre el bus de datos antes del flanco de subida de la entrada/salida ($\overline{IOW}$) o señal de escritura de memoria ($\overline{MEMW}$), en el cual el reloj de datos esta presente en el puerto de salida o memoria. Durante la inicialización del ciclo de lectura del bus, la entrada del puerto debe tener datos sobre el bus de datos antes del flanco de subida de la entrada/salida ($\overline{IOR}$) o señal de lectura de memoria ($\overline{MEMR}$), y el cual asegura el dato en el microprocesador.

3.2.3 $\overline{\text{MEMR}}$, $\overline{\text{MEMW}}$, $\overline{\text{IOR}}$, $\overline{\text{IOW}}$

Estas señales de operación para el control de lectura y escritura se activan en nivel bajo. Como se menciona anteriormente, pueden ser generadas por el procesador o por el controlador de DMA.

3.2.4 ALE (Address Latch Enable)

Para el sistema de bus de la PC, el ALE indica el comienzo de un ciclo de inicialización de bus, cuando esta señal es mantenida, el sistema del bus de datos no contiene información en la dirección. La función ALE no demultiplexa las direcciones del bus de datos.

3.2.5 AEN (Address Enable)

Esta señal es editada por el controlador de DMA, indica que un ciclo de DMA esta en progreso. Esto es normalmente usado para deshabilitar la entrada/salida de los puertos lógicos decodificados durante un ciclo de DMA de tal manera que las direcciones de memoria de DMA no se usan inadvertidamente como una entrada/salida de la dirección del puerto. Esta situación podría ocurrir desde que IOR o IOW puedan activarse durante un ciclo de DMA.

3.2.6 OSC(oscilador), CLOCK

OSC es el sistema de reloj de alta velocidad con un periodo de 70 ns (14.31818 MHz) y un 50% de ciclo de trabajo. El CLOCK es una tercera parte de la frecuencia del oscilador (4.77 MHz). Esta es la frecuencia de operación del microprocesador Intel 8088. Este tiene un periodo de 210 ns y un 33% de ciclo de trabajo.

3.2.7 Interrupciones IRQ2 – IRQ7

Estas seis líneas de entrada son para solicitar una interrupción al procesador. Tienen prioridades, IRQ2 tiene la más alta prioridad y IRQ7 tiene la más baja. Una interrupción es generada por un flanco en la línea IRQ y se queda retenida en alto hasta que es reconocida por el procesador.

3.2.8 I/O CH RDY I/O (Canal listo)

Esta es una señal de entrada usada para generar estados de espera, los cuales extienden la longitud del bus del microprocesador en ciclos para memoria lenta y dispositivos de entrada y salida.

3.2.9 I/O CH CK I/O (Canal de verificación)

Esta es una señal activada en bajo usada para informar al procesador que hay un error de paridad en la memoria o en los dispositivos de entrada y salida.

3.2.10 RESET DRV (Manejador de reset)

Esta señal es usada para inicializar el sistema lógico si esta prendido o en caso de que no haya nivel de voltaje el rango de operación después del encendido esta señal es sincronizada con el flanco de bajada del OSC.

3.2.11 DRQ1 a DRQ3 (respuesta DMA)

Estas líneas de entrada son respuestas de canales asíncronos usados por dispositivos periféricos para ganancia de un servicio DMA. Una línea DRQ puede mantenerse en alto hasta que la correspondiente línea DACK no este en bajo. Note que la DRQ0 no está disponible en el bus, ésta es usada para refrescar el sistema de memoria dinámico.

3.2.12 DACK0 A DACK3 (señales de reconocimiento DMA)

Estas líneas se activan en nivel bajo son usadas para reconocer solicitudes de DMA y para refrescar la memoria dinámica (DACK0).

3.2.13 T/C(estación de conteo)

Esta línea da un pulso cuando la estación de conteo ha terminado y ha llegado al canal de DMA.

3.2.14 Limitaciones de Alimentación

Las especificaciones del suministro de alimentación en la PC, van desde 5V aproximadamente 4 A están disponibles en 5 ranuras o slots. Si otros cuatro slots son usados para características de tarjetas de expansión, entonces menos de 1 A esta disponible para la tarjeta prototipo.

3.2.15 Alimentación de Desacoplamiento

La instalación de cables y conexiones que se realizan en las computadoras agregan inductancia a la fuente de alimentación, la cual hace que la alimentación sufra variaciones de voltaje llamados comúnmente transitorios o fluctuaciones de voltaje. Para resolver este problema se utilizan los capacitores que son de gran importancia en los circuitos, ya que sirven para disminuir la alimentación transitoria. Para grandes y lentas fluctuaciones de alimentación se usan capacitores con valores de 10 a 50 μF. Para alta frecuencia y pequeños transitorios se usan capacitores de cerámica en el rango de 10 a 100 nF, estos capacitores son normalmente colocados entre los pines de tierra y alimentación de los dispositivos como por ejemplo chips TTL, manejadores de bus, transmisores receptores, chips de gran escala de integración y dispositivos de conmutación de alta velocidad.

3.2.16 Mapeo de E/S en la PC de IBM

En las instrucciones para generar las señales correctas, se necesita conocer las direcciones de los puertos de E/S y el mapa de asignación de puertos. El diseño de la PC tiene 10 bits para direcciones de puertos, del bit A0 a A9, para un total de 1024 direcciones de puertos. El mapa de direcciones de puertos de E/S esta dividido en dos partes. Las 512 direcciones de la 0000H a 01FFH (Estas son en notación hexadecimal) son asignadas a la tarjeta madre. Las direcciones entre la 0200H a 03FFH, son 512 direcciones, están disponibles para 5 tarjetas. La figura 3.2 muestra que de las 512 direcciones de puertos ya han sido asignadas para características de tarjeta de IBM. Las direcciones designadas para la tarjeta prototipo van de la 0300H a 31FH, solamente 32 direcciones de puerto.

Dirección Hex.	Uso	
000-00F	DMA chip 8237A-5	
020-021	Interrupción 8259A	
040-043	Timer 8253-5	Asignado a la tarjeta madre
060-063	PPI 8255-A	
080-083	DMA registros de página	
0Ax	NMI registros de máscara	
0Cx	Reservado	
0Ex	Reservado	
100-1FF	No usados	
200-20F	Control de Juegos	
210-217	Unidad de Expansión	
220-24F	Reservado	
278-27F	Reservado	
2F0-2F7	Reservado	
2F8-2FF	Comunicación Asíncrona (2)	
300-31F	**Tarjeta Prototipo**	
320-32F	Disco Flexible	Asignado para tarjetas con puertos
378-37F	Impresora	
380-38C	Comunicación SDLC	
380-389	Comunicación Binaria Síncrona (2)	
3A0-3A9	Comunicación Binaria Síncrona (1)	
3B0-3BF	IBM monocromático display/printer	
3C0-3CF	Reservado	
3D0-3DF	Gráficos de color	
3E0-3F7	Reservado	
3F0-3F7	Diskette	
3F8-3FF	Comunicación Asíncrona (1)	

Figura 3.2 Mapa de Direcciones de Entrada/Salida de la PC.

3.3 MANEJADOR DE PUERTOS

El circuito integrado 8255 es una interfaz periférica programable (PPI), un componente muy popular de bajo costo para interfaces, que se encuentra en muchas

aplicaciones. En la figura 3.3 se ilustra el diagrama de base del 8255, sus tres puertos de entrada/salida (Puertos A, B y C) se programan en 2 grupos de 12 terminales. Las conexiones del primer grupo llamado grupo A constan de todo el puerto A (PA7-PA0) y de la mitad superior del puerto C (PC7-PC4); el segundo grupo llamado grupo B consiste de todo el puerto B (PB7-PB0) y la mitad inferior del puerto C (PC3-PC0). Tiene una terminal para que se habilite llamada CS (Chip Select) que necesita estar activada para leer o escribir en un puerto o también programar.

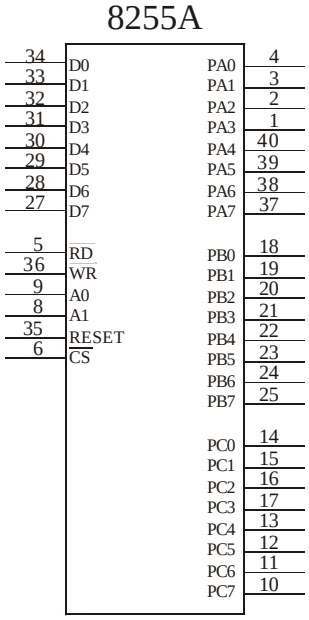


Figura 3.3 Diagrama base del 8255.

La selección de sus registros se logra por medio de las terminales A1 y A0, que seleccionan un registro interno para programación u operación. En la tabla 3.1 se muestran las asignaciones de puertos de E/S (Entrada/Salida) usadas para la programación y acceso a esos puertos. En una computadora personal un 8255 o su equivalente se decodifica en los puertos 60H -63H de E/S [5].

A1	***A0***	Función
0	0	Puerto A
0	1	Puerto B
1	0	Puerto C
1	1	Registro de comando

Tabla 3.1 Asignación de puertos de E/S para el 8255.

3.3.1 Programación del 8255

El 8255 es muy fácil de usar, para poder leer o escribir en él, la entrada CS debe ser cero lógico y si se quiere mandar el registro de comando al 8255, las terminales A1 y A0 deben tener uno lógico, Las terminales D7,D6,D5,D4,D3,D2,D1,D0 se conectan de acuerdo al tipo de funcionamiento que se requiera en dicho dispositivo, en la figura 3.4 se muestran las diferentes opciones que se pueden utilizar.

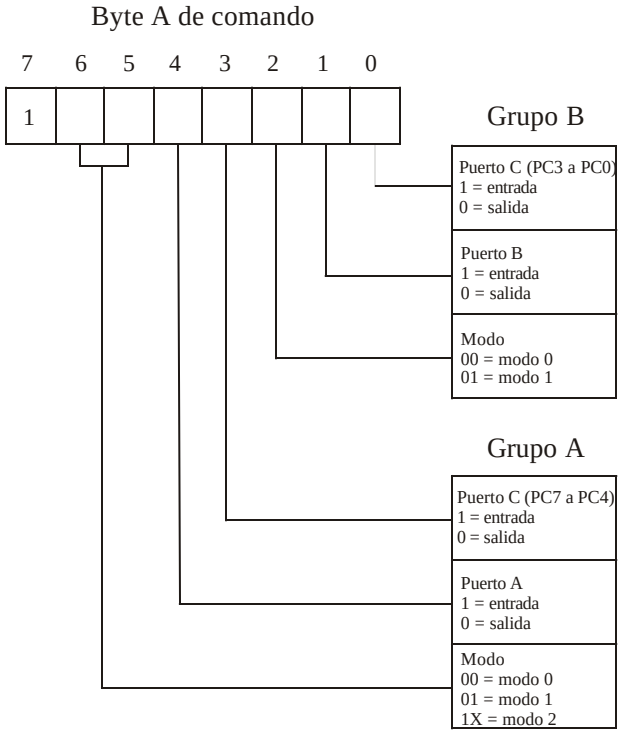


Figura 3.4 El byte de comando para el registro de control del 8255, (a) programación de los puertos A, B

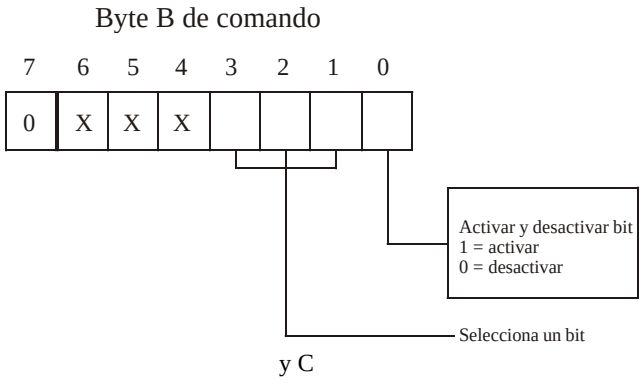


Figura 3.4 (b) Activa o desactiva el bit indicado del puerto C en el campo s*elecciona un bit.*

Funcionamiento en Modo 0 [5]: Este permite que el 8255 actúe como registro de entrada o como dispositivo de salida con registro transparente.

Funcionamiento en Modo 1 [5]:

- ***Entrada mediante habilitación*.** Este funcionamiento hace que el puerto A o el puerto B funcionen como registros de entrada. Esto permite que los datos externos se almacenen en el puerto hasta que el microprocesador está listo para leerlos. El puerto C se utiliza también en el funcionamiento en modo 1, no para datos, sino para señales de control o de "reconocimiento" que hacen al puerto A o B como puertos de entrada mediante una señal de habilitación estroboscópica.
- ***Salida mediante habilitación*.** El funcionamiento con salida por habilitación estroboscópica es similar a la salida en Modo 0, excepto que se incluyen las señales de control para que haya un protocolo de reconocimiento.

Funcionamiento en Modo 2 (Bidireccional) [5]: En el modo 2 solo esta permitido para el grupo A, el puerto A se vuelve bidireccional y permite transmitir y recibir datos en las mismas ocho terminales. Un canal de datos bidireccional es útil cuando hay una interfaz entre dos computadoras. También se utiliza para la interfaz paralela estandar IEEE-488, de alta velocidad (canal de instrumentación de uso general, GPIB).

Para programar el 8255, por ejemplo si se requiere de configurar dos puertos de salida y uno de entrada, entonces, tomemos los puertos A y B como salidas y el puerto C con entrada, por lo cual necesitamos CS en cero lógico, A1 y A0 en uno lógico ambas terminales, para accesar al registro de comandos. La palabra de control 1000 1001 en D7D6D5D4 D3D2D1D0 respectivamente para que los puertos A y B sean salidas y el puerto C sea entrada.

Otro ejemplo sencillo, es la configuración que se utilizó en este proyecto, en donde los tres puertos funcionan como salida, CS en cero lógico, A1 y A0 en uno lógico para accesar al registro de comandos, la palabra de control es 1000 0000 en D7D6D5D4 D3D2D1D0, quedando los puertos A, B y C como salidas.

Para usar los diferentes puertos del 8255 se necesita, el CS este en cero lógico, A0 y A1 en cero lógico, para accesar al puerto A, A1 y A0 en 0 y 1 respectivamente para el puerto B, A1 y A0 en 1 y 0 respectivamente para el puerto C. Si los puertos están configurados como salidas, la palabra que se coloque en D7D6D5D4 D3D2D1D0 se obtendrá en el puerto que este en uso en ese momento, por el contrario si su configuración es como entrada, lo que se coloque en el puerto en uso se obtendrá en D7D6D5D4 D3D2D1D0.

3.4 DECODIFICADOR DE DIRECCIONES

La parte de la decodificación de las direcciones del bus ISA se hace por medio de una GAL22V10 que es un arreglo lógico programable. En la figura 3.5 se muestra la arquitectura de este dispositivo que tiene la función de mapear las direcciones disponibles de la PC, las direcciones son la 300, 301, 302 y 303 (estas direcciones son en hexadecimal). El programa que se grabó a este dispositivo está realizado en el formato Opal.

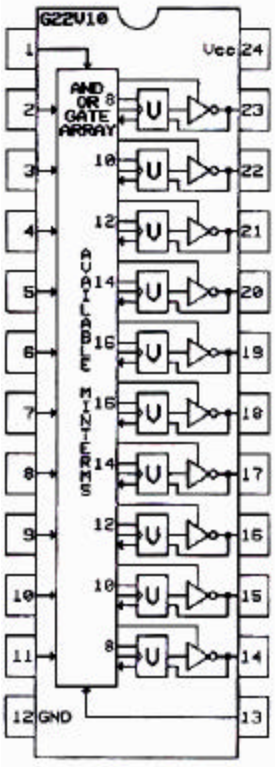


Figura 3.5 Arquitectura de una GAL22V10.

La GAL22V10 en la figura 3.5 se muestra claramente la arquitectura con la cual puede simplificar en un solo dispositivo diseños muy complejos de compuertas contadores, etcétera y tiene en su configuración interna:

- Como entradas solamente los pines: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13.
- Como entradas o salidas los pines: 14, 15, 16, 17, 18, 19, 20, 21, 22, 23.

Ventajas del uso de GALs

- Reducción del número de componentes. Con ello se logra aumentar la confiabilidad, disminuir el consumo de energía, menor ruido y menor superficie ocupada en la placa de circuito impreso.
- Facilidad para el cambio del diseño.

La tarjeta controladora se maneja por medio de un programa realizado en BorlandC, la GAL22V10 esta programada para decodificar de la dirección 300h a 303h al tener esas direcciones disponibles en el bus de la PC, la GAL22V20 manda a habilitar el 8255 por medio de un cero lógico en CS, mandándose también A1 y A0 en uno lógico, después se manda la palabra de control 1000 0000 a D7D6D5D4 D3D2D1D0 respectivamente para que todos los puertos del 8255 funcionen como salida. Al hacer todo esto ahora ya podemos mandar datos a cualquiera de los tres puertos, y funcionará adecuadamente la tarjeta.

La señal IOW del bus de la PC y WR del 8255 van conectadas entre si para el control de escritura que en el 8255 manejamos como salida, por el contrario IOR del bus de la PC y RD del 8255 que se conectan entre si, controlan la lectura o las entradas.

En la figura 3.6 se muestra un diagrama a bloques de las conexiones de la tarjeta utilizando unos LEDS para su prueba. Para un mejor entendimiento en el apéndice A se muestra el circuito completo de la tarjeta controladora.

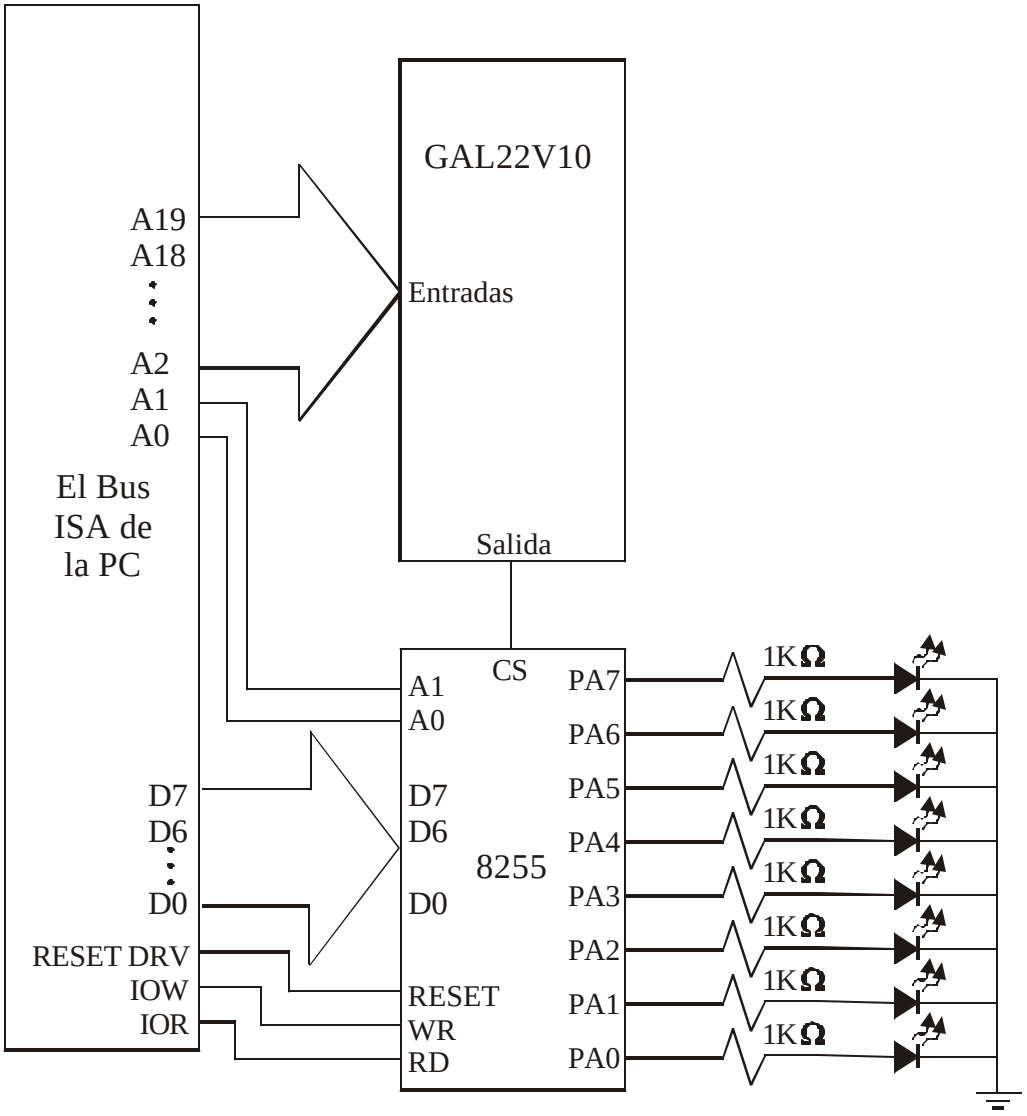


Figura 3.6 Diagrama a bloques de la tarjeta controladora.

En la figura 3.7 se muestra el programa para que los LED’s de la figura 3.6 prendan alternadamente empezando por el bit menos significativo hasta el mas significativo.

```
#include <stdio.h>
#include <dos.h>
#include <conio.h>

int cad[ ]={0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80},i=0; //
Datos a mandar
int main(void)
{
   outportb(0x303, 0x80);      // Palabra de control para
configurar el 8255
   delay(100);
   while(!kbhit()){
     outportb(0x300, cad[i]);    // Se mandan los datos por el
puerto A
     delay(500);
     i++;
     if(i==8) i=0;
   }
   outportb(0x300,0x00);
   return 0;
}
```

Figura 3.7 Programa en BorlandC para probar la tarjeta controladora.

En la Figura 3.8 se muestra el programa que se utilizó para que la GAL22V10 decodifique las direcciones 300,301,302 y 303 hexadecimal.

```
Begin header
      Decodificador de la direccion 300H para el bus de expansion
End header

Begin definition
     Device GAL22V10;

     Inputs A2=1,A3=2,A4=3,A5=4,A6=5,A7=6,A8=7,A9=8,A10=9,A11=10;
     Inputs A12=11,A13=13,A14=14,A15=15,A16=16,A17=17,A18=18,A19=19;

     Outputs /CS=20,/CS1=21;
End definition

Begin EQUATIONS

CS=A19&/A18&/A17&/A16&/A15&/A14&/A13&/A12&/A11&/A10&/A9&/A8&/A7&/A6&/A
5&/A4&/A3&A2;
CS1=A19&/A18&/A17&/A16&/A15&/A14&/A13&/A12&/A11&/A10&/A9&/A8&/A7&/A6&/
A5&/A4&/A3&A2;

End EQUATIONS
```

EL PROGRAMA

4.1 INTRODUCCION

El software, o la parte correspondiente a programas, es otro elemento importante en el desarrollo del sistema, quizás el más importante puesto que define el comportamiento del mismo. El software esta diseñado para ejecutarse en una PC IBM o compatible, esta formado por un proyecto llamado *GRAFICA.PRJ* dentro del cual se tienen los siguientes módulos: *PRINCIPAL.C, LISTA.H Y SVGACC.LIB.*

El programa tiene como objetivo apoyar al usuario en la realización y diseño del circuito, así como también se encarga de manipular al graficador o parte mecánica mandándole instrucciones a través de la tarjeta de control.

Para el desarrollo del programa se aplicaron las reglas de la programación estructurada que es un estilo sencillo de programación. Para la manipulación de datos se utilizó una estructura conocida como *Lista Ligada*.

4.1.1 LISTAS LIGADAS [3]

Una lista ligada es una estructura de datos que puede ser usada para mantener un número de objetos de datos. La lista tiene un límite para almacenar los datos, debido a la cantidad de memoria disponible.

Una lista ligada esta hecha sobre un número de *nodos*, cada uno de los cuales mantiene un elemento de datos, y también un *apuntador* al nodo siguiente.

Apuntador: **El estándar de C permite al programador hacer referencia a la localidad de memoria de los objetos al igual que a los objetos mismos(es decir, al contenido de tales localidades de memoria). Por ejemplo, si x se declara como entero, &x se refiere a la localidad de memoria que se ha reservado para que**

contenga x. &x se denomina un apuntador. Entonces un apuntador es aquel que nos indica la localidad de memoria de x dato.

Nodo: **Esta formado por dos o mas campos, uno o mas de información y uno o dos campos de dirección al siguiente o al anterior(apuntador), los cuales contienen la dirección del nodo siguiente o del nodo anterior.**

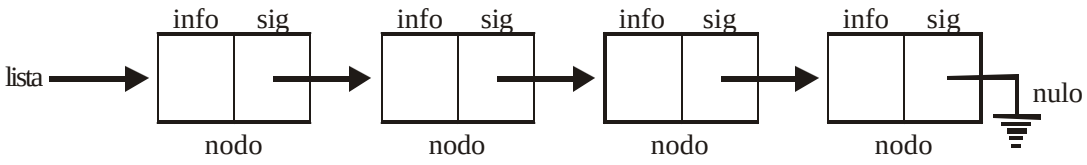


Figura 4.1 **Una lista ligada.**

4.2 CARACTERISTICAS DEL PROYECTO

GRAFICA.EXE Es un proyecto realizado en BorlandC (versión 3.1) que contiene las siguientes características:

- **Es un programa para ejecutarse en un sistema operativo MS-DOS versión 6.0 en adelante.**
- **Cuenta con un conjunto de Menús para seleccionar las diferentes opciones, la selección se puede hacer a través de el ratón proporcionando una interfaz de fácil manejo.**
- **Para su ejecución, no necesita una computadora poderosa, los requerimientos mínimos son: Procesador 80486, espacio en disco duro de 80Kbytes, espacio libre en RAM de 12Mbytes y monitor VGA.**
- **Se configura automáticamente el modo de video para obtener una resolución de 640x480 píxeles.**

El programa se compone de la siguientes partes:

- Dos listas ligadas independientes: Estas dos listas llevan el control de las pistas y nodos que se van editando.
- Funciones para lectura y escritura a archivos.
- Funciones para el manejo del menú principal.

- Funciones para el manejo de cada submenú.
- Funciones para el manejo del bus de la PC para mandarle instrucciones a la tarjeta controladora para que manipule el graficador.

4.3 ORGANIZACIÓN DEL PROGRAMA

El programa se estructura de la siguiente forma: La función o programa principal y un conjunto de funciones diversas que por su función se clasifican en cinco grupos: Funciones de entrada y salida, funciones para establecer la comunicación con el bus ISA, funciones que dan forma a las ventanas y una miscelánea de funciones. En figura 4.2 se muestra el diagrama de flujo de la función principal.

La función principal entra en un ciclo "do - while" para ejecutarse continuamente desplegando el menú principal en espera de la elección del usuario. Son cuatro los comandos que se presentan al usuario; a continuación se listan y se describe de manera breve lo que cada uno realiza. *Sólo se puede acceder a los comandos por medio del ratón o mouse***.**

Archivo: **Le despliega al usuario tres subcomandos o un nuevo submenú a elegir. Las tres opciones son:**

1. *Nuevo* **: Abre un nuevo archivo para empezar a diseñar un impreso.**
2. *Cargar:* **Despliega una ventana en espera de el nombre del archivo a cargar, si el nombre del archivo es correcto despliega el diseño en pantalla. El nombre del archivo tiene como extensión .GRA si cuando se le da el nombre no se pone la extensión el programa da por hecho que tiene esa extensión, lo busca y lo carga.**
3. *Salvar:* **Despliega una ventana esperando el nuevo nombre del archivo para guardar el diseño.** ***El nombre del archivo que se le de al diseño no necesita incluir extensión ya que el programa le asigna por default .GRA.***

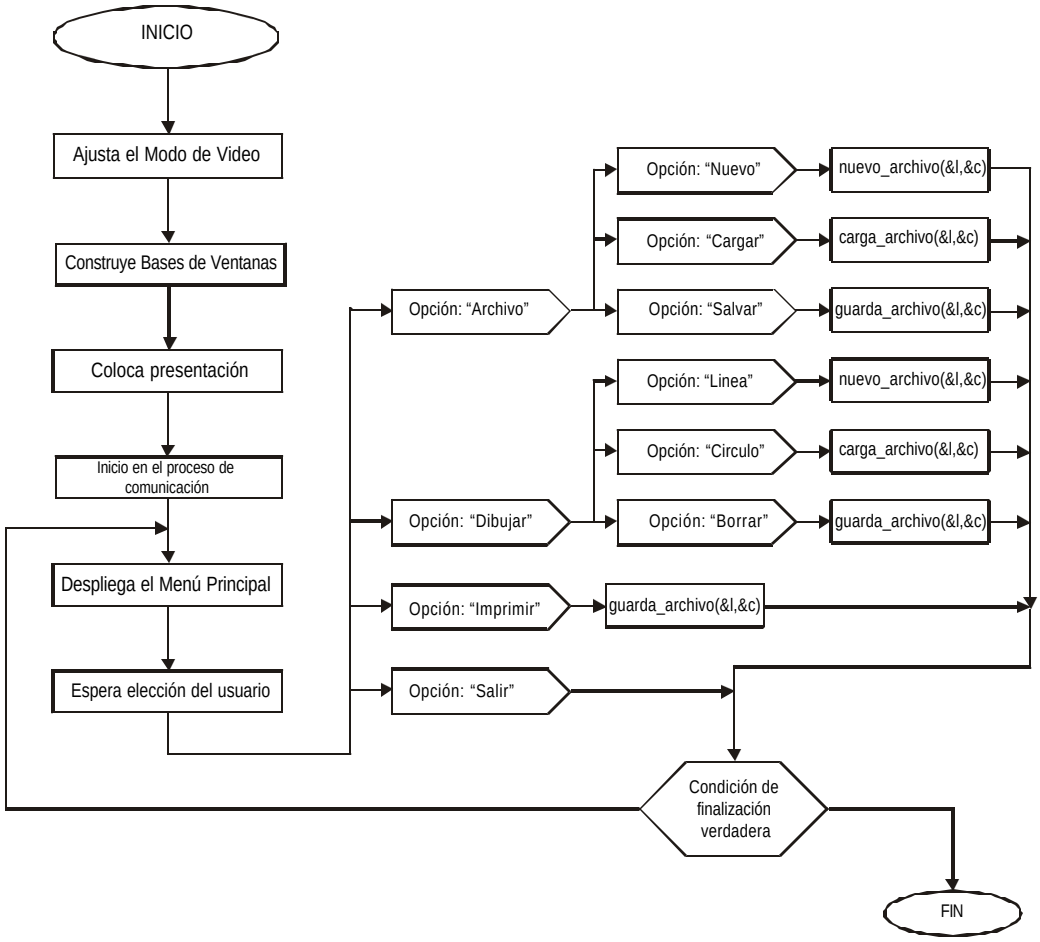


Figura 4.2 **Diagrama de flujo de la función principal de GRAFICA.**

Dibujar: **Al igual que el primer comando despliega tres subcomandos u otro submenú a elegir. Sus subcomandos son:**

1. *Línea*: **Con este subcomando el usuario puede realizar líneas rectas a cero noventa y cuarenta y cinco grados respectivamente para esto el cursor del mouse cambia para facilitar el manejo de este. Para realizar una línea el usuario debe dar un "clic" en el botón izquierdo y mantenerlo presionado hasta que la línea alcance la ubicación y longitud deseada. Con un "clic" en el botón derecho se termina la ejecución de este subcomando.**
2. *Círculos*: **Con este subcomando el usuario puede colocar donas o círculos de manera muy simple, solo tiene que posicionar el mouse en el lugar donde se desee colocar la dona y con un "clic" en el botón izquierdo será colocada, si el usuario ya no desea usar este subcomando, con un "clic" en el botón derecho terminará su ejecución.**

3. *Borrar*: **Por medio de este subcomando el usuario puede borrar líneas o donas que ya no requiera dentro del diseño que actualmente este realizando, para borrar el objeto deseado basta solo con posicionarse sobre el y dar un "clic" con el botón izquierdo, después de esto aparecerá una ventana preguntando si desea borrar la línea o dona resaltada de un color diferente al que aparecía antes de seleccionarla para borrar. Si ya no se desea borrar mas objetos, al igual que los dos subcomandos anteriores, con un clic en el botón derecho del mouse se termina la ejecución.**

Imprimir: **Este comando como su nombre lo indica manda a imprimir el diseño que se tenga en pantalla. Para esto la placa de cobre ya debe de estar lista y en posición en el graficador para que éste realice su tarea.**

Salir: **Este comando hace que la condición de finalización del programa se haga verdadera.**

El diseño que se va realizando en pantalla se almacena en una lista ligada la cual se define a continuación:

```
struct punto {
      int a1,b1,a2,b2;
      struct punto *sig1;};
typedef struct punto *a_punto;
```

La reservación de memoria se hace de manera dinámica.

4.4 VINCULACION DEL PROGRAMA CON EL GRAFICADOR

Aquí se explica brevemente la forma en que se resolvieron los problemas surgidos en la elaboración del programa, así como la forma de manipular los datos proporcionados por el usuario y mandarlos a el graficador.

4.4.1 LLENADO DE LAS LISTAS

El programa utiliza listas ligadas dinámicas para almacenar las coordenadas de las líneas del diseño que se vaya realizando la lista se llena de la siguiente forma:

Como primer paso se reserva memoria para la nueva línea; después de esto se capturan las cuatro coordenadas (x1,y1,x2,y2) que nos indican la posición de la

nueva línea. Al realizar la captura se procede a agregarla a la lista para que sea más entendible. A continuación se muestra el código que se utilizó para esta tarea.

```
a_punto crear_punto(int  a, int  b, int aa, int bb) {
a_punto  t;
t = (a_punto) malloc(sizeof(struct punto));
t -> a1 = a;
t -> b1 = b;
t -> a2 = aa;
t -> b2 = bb;
t -> sig1 = NULL;
return t;
}

void agregar_punto(struct  linea  *l, a_punto  p) {
a_punto t;
t = l -> linea_ini;
l -> linea_ini = p;
p -> sig1 = t;
}
```

Donde *a_punto* es un apuntador del tipo *struct punto,* para un mejor entendimiento del programa en el apéndice A se muestra el código completo.

Con la función que se muestra a continuación se agrega la nueva línea a la lista.

```
void funcion(int z1,int z2,int z3,int z4){
 a_punto   p;
 p = crear_punto(z1,z2,z3,z4);
 agregar_punto(&l, p);
}
```

La variable *&l* es el apuntador a la lista de todas las líneas que se tienen en el diseño , z1, z2, z3, z4 corresponden a las coordenadas de la nueva línea a insertar x1,y1,x2,y2 respectivamente. En la figura 4.3 se muestra un diseño de dos líneas y

su respectiva lista, en la figura 4.4 se agregó una línea al diseño y se muestra la nueva lista de líneas.

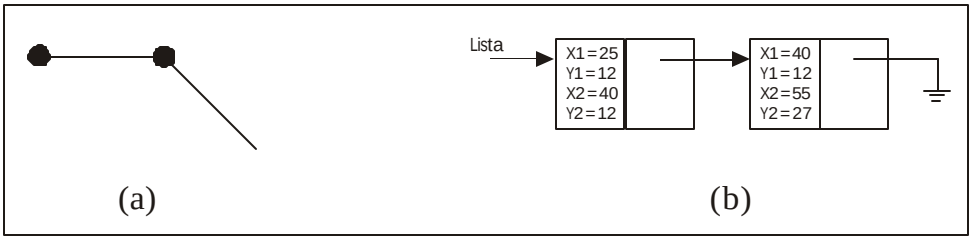


Figura 4.3 **(a) Un diseño sencillo y (b) como esta conformada la lista de líneas.**

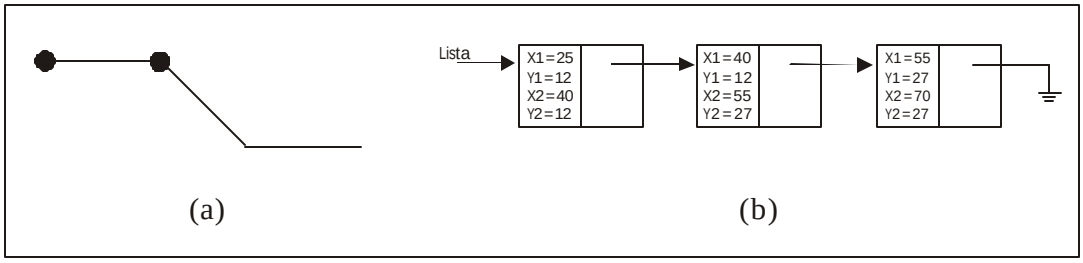


Figura 4.4**(a) El mismo diseño con una nueva línea. (b) La nueva lista de líneas.**

De manera similar se realiza la captura de las listas de las donas que se van agregando al diseño. El código es similar y a continuación se muestra:

```
a_centro crear_centro(int zx,int zy){
a_centro t;
t = (a_centro) malloc(sizeof(struct centro));
t -> xc = zx;
t -> yc = zy;
t -> sig2 = NULL;
return t;
}
void agregar_centro(struct circulo *circ,a_centro cent){
a_centro t;
t = circ -> circulo_ini;
circ -> circulo_ini = cent;
cent -> sig2 = t;
}
```

De igual forma con la siguiente función se agrega la nueva dona a la lista de donas.

```
void funcion2(int zx,int zy){
 a_centro cir1;
 cir1 = crear_centro(zx,zy);
 agregar_centro(&cir,cir1); }
```

Donde *&cir* es el apuntador a la lista principal de donas del diseño que se realice en pantalla. En la figura 4.5 se muestra el mismo diseño del ejemplo anterior con la lista de donas que se tienen, en la figura 4.6 se agrega una nueva dona y se muestra su nueva lista.

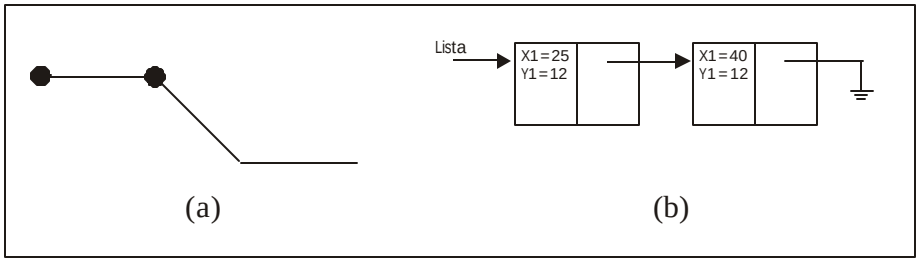


Figura 4.5 **Un diseño sencillo y (b) como esta conformada la lista de donas.**

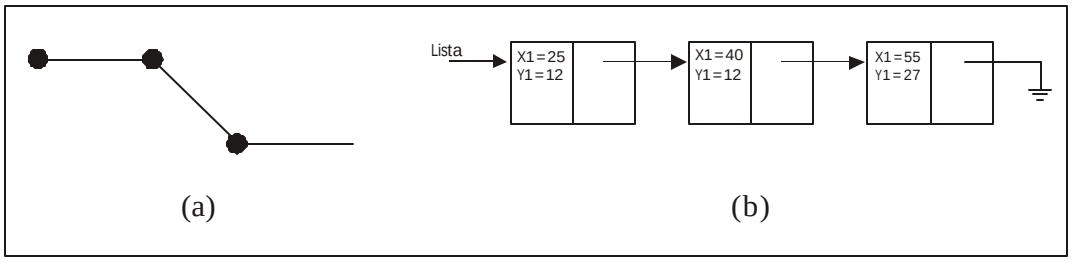


Figura 4.6 **(a) El mismo diseño con una nueva dona. (b) La nueva lista de donas.**

4.4.2 FORMATO DEL ARCHIVO

Para que el usuario pueda seguir realizando su diseño después o para que vuelva a imprimirlo en otra ocasión, el programa tiene la forma de guardar los diseños que el usuario va creando en archivos, la extensión de estos archivos es *.GRA* (*graficador*) se le puso esta extensión para identificar el archivo de manera mas simple.

La forma en que se guarda la información de las listas en un archivo *.GRA* es de manera simple, se guardan primero todas las coordenadas de la lista de líneas, identificándose por una *l* al inicio de el conjunto de coordenadas de cada línea y después se procede a guardar las lista de donas que al igual que las líneas se identifica con una *c* después de la *c* va el par de coordenadas de ubicación de la dona. El final del archivo se identifica por una letra *f* (final). Para que se tenga una mejor idea de como queda finalmente un archivo *.GRA* en la figura 4.7 se muestra el archivo que corresponde a el ejemplo anterior.

l25,12,40,12l40,12,55,27l55,27,70,27c25,12c40,12c55,27

Figura 4.7 **El formato del archivo .GRA.**

4.4.3 IMPRESIÓN

La impresión se realiza después que se convierten todos los datos de las listas tanto las líneas como las donas en una sola matriz de unos y ceros, en donde los unos representan puntos o píxeles y los ceros representan espacios en blanco o dicho de otra forma es el lugar donde no se muestra punto o píxel alguno.

Al mandarse a imprimir un diseño lo primero que se hace es crear un archivo llamado *matriz.txt* después las funciones que se encarga de realizar el llenado se llaman *manda_lineas* y *manda_cir* que mandan las líneas y las donas respectivamente a el archivo *matriz.txt* que esta lleno de ceros por ser un diseño en blanco, al crear un diseño el usuario y mandarlo a imprimir, este se manda a la matriz.

Al tener la matriz con el diseño que el usuario realizó y al mandar a imprimir, la función que se encarga de leer los datos de la *matriz* y mandarlos a el graficador para obtener el resultado final se llama *graficar*. Cuando la función *graficar* lee un "uno" significa que el plumón debe bajar y pintar un punto en la placa de cobre, por el contrario si lee un "cero", el plumón debe subir para no pintar. En la figura 4.8 se muestra el ejemplo de cómo queda el archivo MATRIZ.TXT.

Para acelerar la impresión, la función *graficar* hace la lectura de los datos del archivo MATRIZ.TXT por renglones y para cada renglón lleva una cuenta del total de "unos" encontrados, para que cuando se pinte un punto, la cuenta se decremente y de esta forma pueda determinarse si es necesario continuar pintando en ese renglón o bien, avanzar al siguiente.

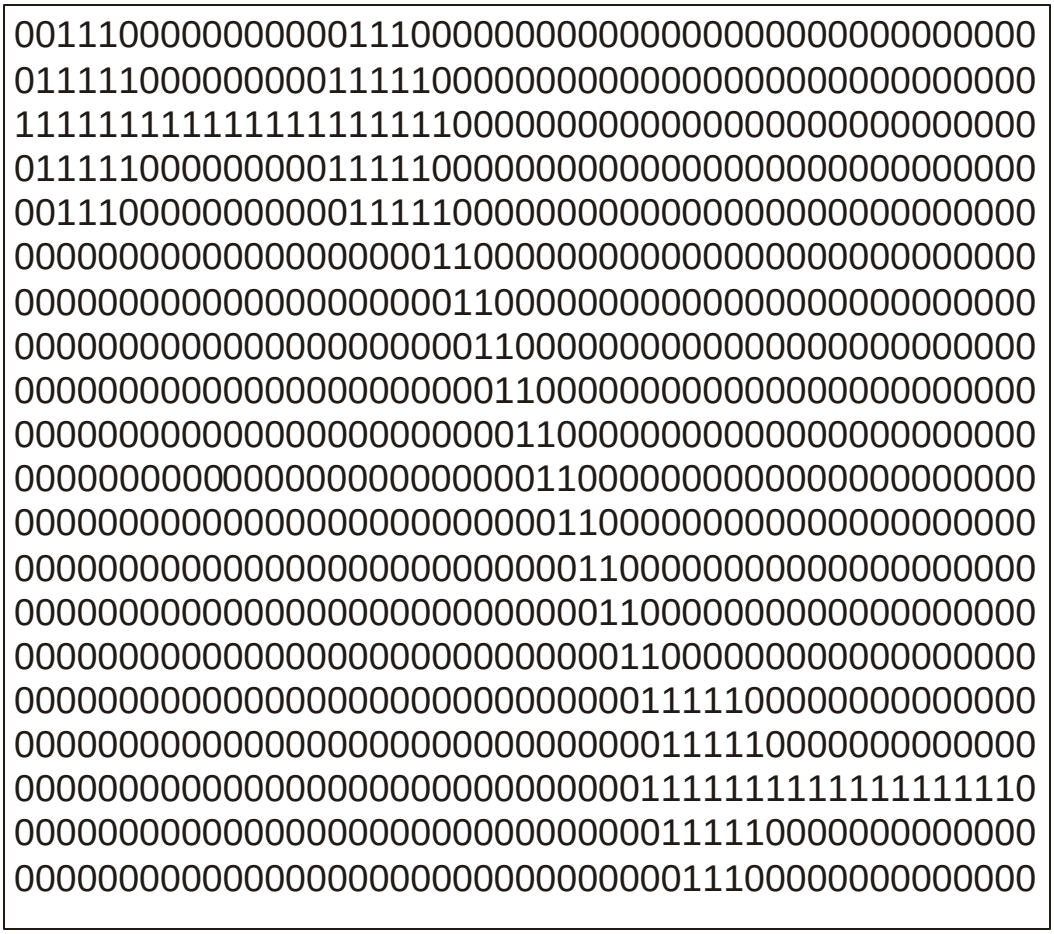


Figura 4.8 **Fragmento de MATRIZ.TXT solo tomando los datos del ejemplo.**

4.5 FUNCIONES QUE INTEGRAN EL PROGRAMA

Además de la función principal y de las funciones de biblioteca, el programa utiliza funciones ya elaboradas para el manejo de gráficos y manejo del ratón. Las funciones que se elaboraron para este programa están distribuidas en cinco grupos funcionales. En esta sección se describen a las funciones bajo el siguiente formato (parte del formato puede omitirse en caso de no ser requerido):

Nombre:	**Nombre de la función**
Definición:	**Definición y objetivos**
Parámetros:	**Tipo de parámetros recibidos**
Retorna:	**Tipo de valor de retorno**

Nombres de las funciones principales y descripción de cada una de ellas.

Nombre:	**menu_archivo.**
Descripción:	**Despliega el submenú archivo, el cual tiene tres subcomandos.**

Nombre:	**nuevo_archivo.**
Descripción:	**Esta subfunción abre un nuevo archivo para empezar a realizar el diseño en pantalla.**
Parámetros:	**struct linea *l (apuntador a la estructura que lleva el control de la posición y numero de líneas), struct circulo *cir (apuntador a la estructura que lleva el control de la posición y número de donas).**
Retorna:	**Un apuntador a struct linea *l y un apuntador a struct circulo *cir vacíos.**

Nombre:	**carga_archivo.**
Descripción:	**Abre un archivo ya existente desplegando una ventanita en espera del nombre del archivo a cargar.**
Parámetros:	**struct linea *l (apuntador a la estructura que lleva el control de la posición y numero de líneas), struct circulo *cir (apuntador a la estructura que lleva el control de la posición y número donas).**
Retorna:	**Un apuntador a struct linea *l y un apuntador a struct circulo *cir con los datos del diseño cargado.**

Nombre:	**guarda_archivo**
Descripción:	**Guarda el archivo en el mismo documento o en otro diferente.**
Parámetros:	**Un apuntador a struct linea *l y un apuntador a struct circulo *cir.**
Retorna:	**Un apuntador a struct linea *l y un apuntador a struct circulo *cir.**
Nombre:	**menu_dibujar**
Descripción:	**Despliega el submenú Dibujar, el cual tiene tres subcomandos.**

Nombre:	**menu_dibujar**
Descripción:	**Despliega el submenú Dibujar, el cual tiene tres subcomandos.**

Nombre:	**dibuja_linea**
Descripción:	**Limita el movimiento del mouse a un área la cual es en donde se puede realizar el diseño. Dibuja líneas en el área de diseño.**

Nombre:	**dibuja_circulo**
Descripción:	**Limita el movimiento del mouse a un área la cual es en donde se puede realizar el diseño. Dibuja donas en el área de diseño.**

Nombre:	**funcion_borrar**
Descripción:	**Al igual que las dos funciones anteriores restringe el movimiento del mouse y borra líneas o donas no deseadas en el diseño.**
Parámetros:	**struct linea *l (apuntador a la estructura que lleva el control de las líneas en el diseño) y struct circulo *cir (apuntador a la estructura que lleva el control de las donas en el diseño).**
Retorna:	**struct linea *l (apuntador a la estructura que lleva el control de las líneas en el diseño) y struct circulo *cir (apuntador a la estructura que lleva el control de las donas en el diseño).**

Nombre:	**imprimir**
Descripción:	**Esta función se encarga de procesar toda la información del diseño en pantalla para después mandarla al graficador.**
Parámetros:	**struct linea *l (apuntador a la estructura que lleva el control de las líneas en el diseño) y struct circulo *cir (apuntador a la estructura que lleva el control de las donas en el diseño).**

Nombre:	**salir**
Descripción:	**Función encargada de hacer que la condición de finalización del programa sea verdadera.**

Otras funciones de apoyo.

Nombre:	**inicialinea**
Descripción:	**Inicia la estructura tipo líneaen nulo.**
Parámetros:	**struct linea *l (apuntador de tipo estruct linea)**

Nombre:	**crear_punto**
Descripción:	**Reserva memoria para crear una nueva línea con solo cuatro puntos o coordenadas.**
Parámetros:	**a_punto (un apuntador a la estructura punto), int a, int b, int aa y int bb (coordenadas x1,y1,x2,y2 que indican respectivamente el inicio y fin de la nueva línea).**
Retorna:	**Un apuntador de tipo a_punto para el cual se reservó memoria.**

Nombre:	**agregar_punto**
Descripción:	**Agrega la nueva línea a la lista ligada de líneas del diseño actual que se ve desplegado en pantalla.**
Parámetros:	**struct linea *l (recibe el apuntador que lleva el control de todas las líneas), a_punto p (y aquí recibe al apuntador de la memoria reservada).**

Nombre:	**iniciacirculo**
Descripción:	**Inicia la estructura de tipo circulo en nulo**
Parámetros:	**struct circulo *q (recibe el apuntador que lleva el control principal a la estructura de las donas)**

Nombre:	**crear_centro**
Descripción:	**Reserva memoria para crear un nuevo circulo con sus respectivas coordenadas xy.**
Parámetros:	**int zx, int zy (recibe las coordenadas de ubicación del nuevo elemento para reservar la memoria).**
Retorna:	**Un apuntador de tipo a_centro para el cual se reservó memoria.**
Nombre:	**recorre_dibujo**

Descripción:	**Hace un recorrido a todo el diseño que se va mostrando en pantalla tanto en líneas con el círculos.**
Parámetros:	**struct linea *l y struct circulo *cir (apuntadores a la lista principal de líneas y círculos).**

Nombre:	**funcion**
Descripción:	**Realiza el almacenamiento en memoria de la nueva línea con la ayuda de las funciones, crear_punto y agregar_punto.**
Parámetros:	**int z1, int z2, int z3, int z4 (coordenadas de la nueva línea).**

Nombre:	**funcion2**
Descripción:	**Realiza el almacenamiento en memoria del nuevo circulo con la ayuda de las funciones, crear_centro y agregar_centro.**
Parámetros:	**int z1, int z2, int z3, int z4 (coordenadas del nuevo circulo).**

Nombre:	**graficar**
Descripción:	**Función que se encarga de mandar las instrucciones necesarias al graficador para que este imprima el diseño en la placa de cobre por medio del plumón.**
Nombre:	**manda_lineas**
Descripción:	**Convierte las líneas del diseño de coordenadas a puntos en la matriz que se utiliza para mandar a imprimir.**
Parámetros:	**Int lx1, int ly1, int lx2, int ly2 (coordenadas de la linea a mandar a la matriz).**

Nombre:	**manda_cir**
Descripción:	**Convierte las donas del diseño de coordenadas a puntos en la matriz que se utiliza para mandar a imprimir.**
Parámetros:	**int cx1, int cy1(coordenadas del circulo a mandar a la matriz).**

En el apéndice B se muestra el código completo del programa, para entender mejor su funcionamiento.

CAPITULO 5

RESULTADOS Y CONCLUSIONES

5.1 RESULTADOS

En la figura 5.1. se muestra un diseño sencillo realizado en el programa, utilizando las herramientas de dibujo.

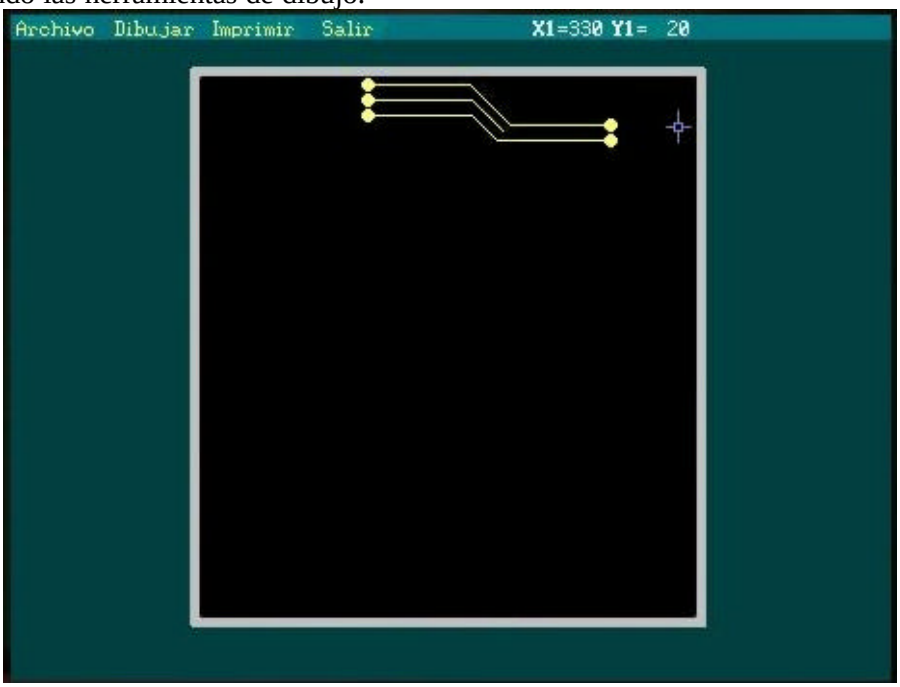


Fig. 5.1 Un diseño sencillo hecho en el programa.

Al concluirse el diseño en el programa, el siguiente paso que se efectuó fué la colocación de la placa de cobre en el graficador, como se muestra en la figura 5.2 colocando el plumón para que este listo y en espera de que el usuario seleccione la opción imprimir.

Cabe aclarar que la placa de cobre debe quedar libre de polvo y grasa para obtener un circuito impreso de mejor calidad.

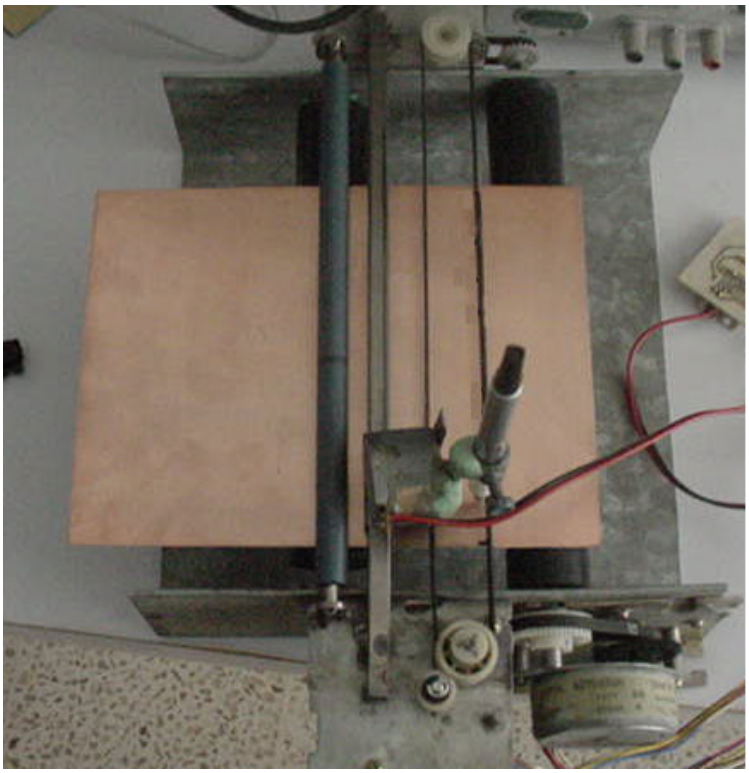


Fig. 5.2 Colocación de la placa de cobre en el graficador.

En la figura 5.3 se muestra al graficador imprimiendo en la placa de cobre el diseño previamente editado en el programa.

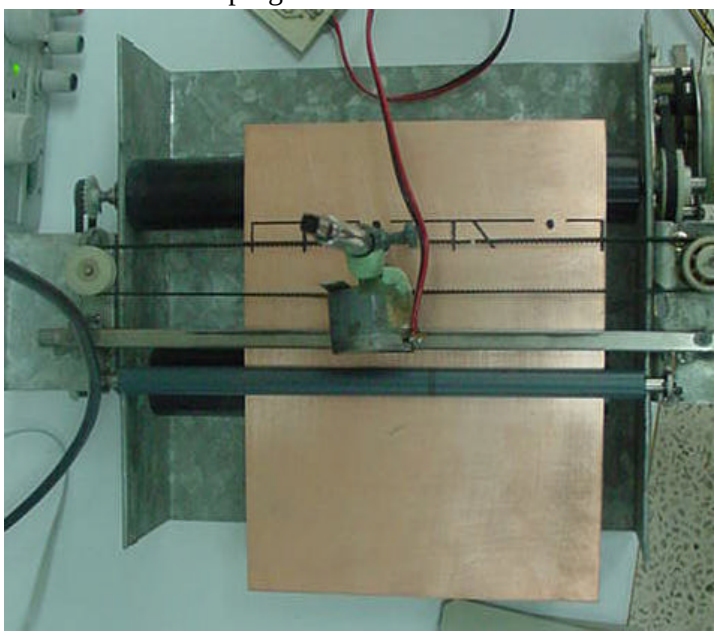


Fig. 5.3 Imprimiendo.

En la figura 5.4 se muestra como quedó el diseño al finalizar la impresión. El cual está listo para introducirse al cloruro férrico.

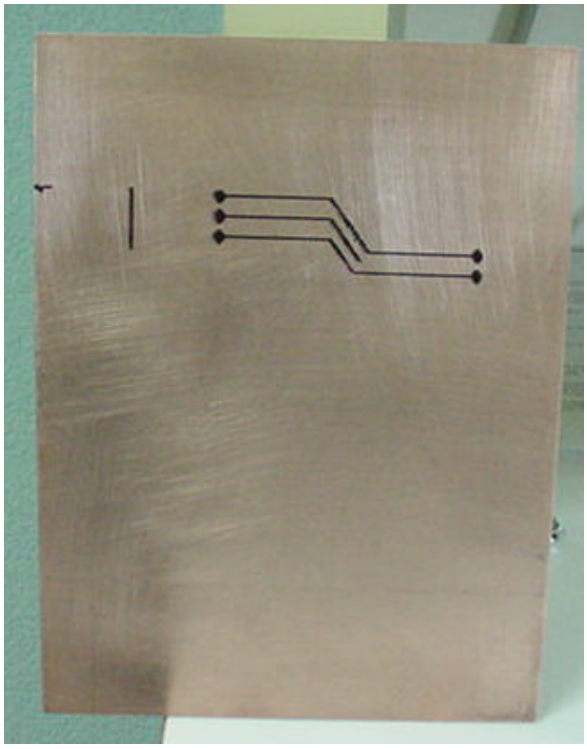


Fig. 5.4 El diseño después de la impresión en la placa de cobre.

En la figura 5.5 se muestra el circuito impreso obtenido. La integridad del circuito impreso se verificó al realizar la prueba de continuidad entre pistas.

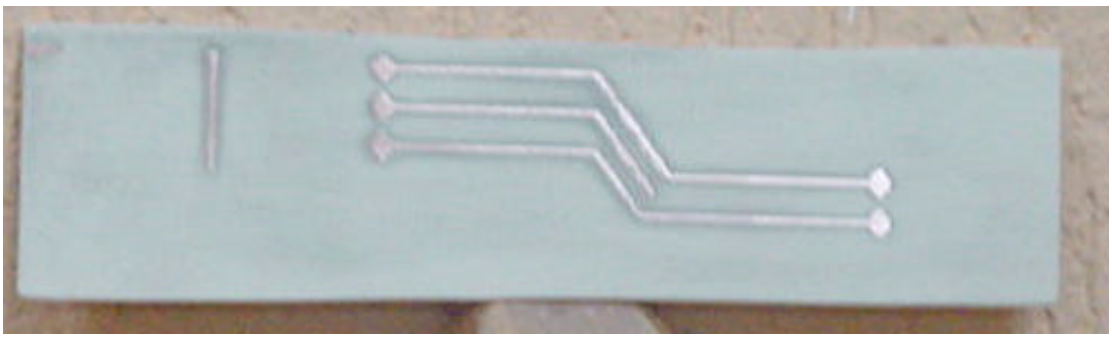


Fig. 5.5 El resultado después de aplicarle el cloruro férrico.

El graficador obtenido tiene una resolución aceptable para los ejes X y Y. Un píxel equivale a 0.425mm en la placa de cobre, lo cual nos da líneas y donas de buen tamaño para realizar diseños aceptables.

La forma de control para el motor del eje X equivale a un píxel por paso, mientras que para el control del eje Y su equidad cambia de 10 pasos por píxel. Por lo que se obtienen pistas de 0.7mm de ancho con un plumón de punta delgada.

El tamaño de la placa de cobre que se puede utilizar en el graficador es de 15x20 centímetros, siendo 15x15 centímetros la parte útil.

5.2 CONCLUSIONES

Este trabajo se realizó utilizando piezas de impresoras como son motores de pasos, engranes, rodillos y bandas. Otras piezas se tomaron de cartuchos para impresoras láser y piezas que se improvisaron y adaptaron a las necesidades del proyecto, con el fin de obtener el mejor resultado posible, con ciertas limitaciones, por la falta de experiencia, pero con gran esfuerzo. Por lo tanto:

La elaboración de los circuitos impresos podrá realizarse sin inconvenientes, como pueden ser: la falta de experiencia, falta de materiales requeridos para la elaboración delcircuito impreso, etcétera.

También se minimizan los costos de elaboración, al dejar de consumir los materiales para negativos y foto resina, que son de precios elevados y difíciles de conseguir en la región.

Se facilita la elaboración en serie de un diseño por que el usuario puede imprimirlo las veces que lo requiera.

Otro aspecto que se mejora es el tiempo invertido en la fabricación de un circuito impreso, este se reduce por que al finalizarse la impresión, inmediatamente la placa con el diseño puede introducirse en el cloruro férrico para obtener el resultado final.

5.3 FUTURAS EXPECTATIVAS

Las mejoras que se sugieren para este trabajo son:

- ✓ Ampliar su resolución con un sistema de engranes para realizar circuitos impresos mas pequeños.
- ✓ Implementarle otras herramientas de utilidad, por ejemplo un taladro para realizar perforaciones del circuito impreso
- ✓ Emplear inyección de tinta, u otra forma de impresión sobre la placa, para tener una precisión mas fina.
- ✓ Transportar el programa de usuario hacia el ambiente Windows, agregando librerías que incluyan módulos prediseñados.
- ✓ Tener la opción de convertir de otros formatos (Orcad, Protel, etc) a el formato utilizado para el graficador.

APENDICE A

Diagrama Electrónico y Diseño de la Tarjeta de Control y de la Etapa de Potencia.

Tarjeta Lado Soldadura

Tarjeta Lado Componentes

Lado Soldadura

LISTA DE COMPONENTES

TARJETA DE CONTROL

a) 1 Circuito Integrado GAL22V10
b) 1 Circuito Integrado 82C55A
c) 1 Conector DB25 Hembra

ETAPA DE POTENCIA

a) 8 Transistores Darlington TIP120
b) 2 Circuitos Integrados CD4050
c) 8 Resistencias de 1KΩ
d) 8 Diodos 1N4001
e) 1 Conector DB25 Macho

APENDICE B

GRAFICA.PRJ: LISTADO DE LA PROGRAMACION EN BORLANDC

PRINCIPAL.CPP

```
/*************** Aquí se empieza el programa principal
****************
#include <stdio.h>
#include <alloc.h>
#include <process.h>
#include <stdlib.h>
#include <conio.h>
#include <dos.h>
#include <io.h>
#include "listas.h"
#include "svgacc.h"
#include "svgademo.h"
#define  TRUE   1
#define  FALSE  0

void menu_archivo(void);
void menu_dibujar(void);
void dibuja_linea(void);
void dibuja_circulo(void);
void funcion_borrar(struct linea *l, struct circulo *cir);
imprimir(struct linea *l,struct circulo *cir);
void manda_lineas(int lx1, int ly1, int lx2, int ly2);
void manda_cir(int cx1, int cy1);
void graficar(void);

struct linea l;
struct circulo cir;

FILE *nuevo,*motor,*motor2; long int jorge1;
int vmode, x, y, x1, y1, n=0, n1=0, n2=0, n3=0, mbuts, f=0;
int f1=0,f2=0,f3=0,f4=0,c1=0,c2=0,c3=0,m1=0,m2=0,m3=0;
char text[50],cad[30];
char textclr[]="";

void main(void)
{
 inicialinea(&l);
 iniciacirculo(&cir);
 vmode = videomodeget();
 if ( !whichvga() )      exit(1);
 if ( whichmem() < 512)  exit(1);
 if ( !whichmouse())     exit(1);

 res640();                                // Inicia el modo 640x480
 fillscreen(0);                           // Colorea toda la pantalla
 drwfillbox(SET,150,0,3,639,18);
 sprintf(cad, "Archivo");
 drwstring(SET,14,150,cad,10,5);
 sprintf(cad, "Dibujar");
 drwstring(SET,14,150,cad,80,5);
 sprintf(cad, "Imprimir");
 drwstring(SET,14,150,cad,150,5);
 sprintf(cad, "Salir");
```

```
drwstring(SET,14,150,cad,230,5);
drwfillbox(SET,200,0,19,640,480);
drwfillbox(SET,25,139,43,501,455);
drwfillbox(SET,0,145,49,495,449);

outportb(0x303, 0x80);
delay(100);
outportb(0x300, 0x00);
delay(100);
outportb(0x301, 0x00);
delay(100);

mouseenter();
mousecursorset(&manomousecursor);
mouseshow();

do{

//****** Efecto de la barra en el menu Archivo ******
mousestatus(&x,&y,&mbuts);
if(!f1 && (x>=5&&x<=72)&&(y>=3&&y<=15)){
mousehide();
drwfillbox(SET,155,5,3,72,18);
sprintf(cad, "Archivo");
drwstring(SET,14,155,cad,10,5);
mouseshow();
f1=1;
}

if(f1 && ((x<5||x>72)||(y<3||y>15))){
mousehide();
drwfillbox(SET,150,5,3,72,18);
sprintf(cad, "Archivo");
drwstring(SET,14,150,cad,10,5);
mouseshow();
f1=0;
}

//****** Efecto de la barra en el menu Dibujar ******
if(!f2 && (x>=73&&x<=144)&&(y>=3&&y<=15)){
mousehide();
drwfillbox(SET,155,73,3,144,18);
sprintf(cad, "Dibujar");
drwstring(SET,14,155,cad,80,5);
mouseshow();
f2=1;
}

if(f2 && ((x<73||x>144)||(y<3||y>15))){
mousehide();
drwfillbox(SET,150,73,3,144,18);
sprintf(cad, "Dibujar");
drwstring(SET,14,150,cad,80,5);
mouseshow();
f2=0;
}


//****** Efecto de la barra en el menu Imprimir ******
if(!f3 && (x>=145&&x<=222)&&(y>=3&&y<=15)){
mousehide();
```

```
 drwfillbox(SET,155,145,3,222,18);
 sprintf(cad, "Imprimir");
 drwstring(SET,14,155,cad,150,5);
 mouseshow();
 f3=1;
 }

 if(f3 && ((x<145||x>222)||(y<3||y>15))){
 mousehide();
 drwfillbox(SET,150,145,3,222,18);
 sprintf(cad, "Imprimir");
 drwstring(SET,14,150,cad,150,5);
 mouseshow();
 f3=0;
 }
 //****** Efecto de la barra en el menu Salir ******
 if(!f4 && (x>=223&&x<=277)&&(y>=3&&y<=15)){
 mousehide();
 drwfillbox(SET,155,223,3,277,18);
 sprintf(cad, "Salir");
 drwstring(SET,14,155,cad,230,5);
 mouseshow();
 f4=1;
 }

 if(f4 && ((x<223||x>277)||(y<3||y>15))){
 mousehide();
 drwfillbox(SET,150,223,3,277,18);
 sprintf(cad, "Salir");
 drwstring(SET,14,150,cad,230,5);
 mouseshow();
 f4=0;
 }
 mousebutpress(1,&x,&y,&n,&mbuts);
 if((n!=0) && ((x>=5&&x<=72)&&(y>=3&&y<=15)))
 menu_archivo();
 else if((n!=0) && ((x>=73&&x<=144)&&(y>=3&&y<=15)))
 menu_dibujar();
 else if((n!=0) && (x>=145&&x<=222)&&(y>=3&&y<=15))
 imprimir(&l,&cir);
 else if((n!=0) && ((x>=223&&x<=277)&&(y>=3&&y<=15)))
 n3 = 1;
 }while(n3 == 0 );

 mousehide();
 mouseexit();
 videomodeset(vmode);
 exit(0);
}




//************************** FUNCIONES ************************

// ************************* Menu Archivo ************************
void menu_archivo(void){
 mousehide();
 drwfillbox(SET,150,5,19,72,70);
 sprintf(cad, "Nuevo");
```

```
drwstring(SET,14,150,cad,10,22);
sprintf(cad, "Cargar");
drwstring(SET,14,150,cad,10,39);
sprintf(cad, "Salvar");
drwstring(SET,14,150,cad,10,56);
mouseshow();

do{
 //****** Efecto de la barra de la opcion Nuevo ******
 mousestatus(&x,&y,&mbuts);
 if(!c1&&((x>=5&&x<=72)&&(y>=19&&y<=37))){
 mousehide();
 drwfillbox(SET,155,5,19,72,37);
 sprintf(cad, "Nuevo");
 drwstring(SET,14,155,cad,10,22);
 mouseshow();
 c1 = 1;
 }

 if(c1 && ((x<5||x>72)||(y<19||y>37))){
 mousehide();
 drwfillbox(SET,150,5,19,72,37);
 sprintf(cad, "Nuevo");
 drwstring(SET,14,150,cad,10,22);
 mouseshow();
 c1 = 0;
 }
 //****** Efecto de la barra de la opcion Cargar ******
 if(!c2&&((x>=5&&x<=72)&&(y>=38&&y<=54))){
 mousehide();
 drwfillbox(SET,155,5,38,72,54);
 sprintf(cad, "Cargar");
 drwstring(SET,14,155,cad,10,39);
 mouseshow();
 c2 = 1;
 }

 if(c2 &&((x<5||x>72)||(y<38||y>54))){
 mousehide();
 drwfillbox(SET,150,5,38,72,54);
 sprintf(cad, "Cargar");
 drwstring(SET,14,150,cad,10,39);
 mouseshow();
 c2 = 0;
 }
 //****** Efecto de la barra de la opcion Salvar ******
 if(!c3&&((x>=5&&x<=72)&&(y>=55&&y<=70))){
 mousehide();
 drwfillbox(SET,155,5,54,72,70);
 sprintf(cad, "Salvar");
 drwstring(SET,14,155,cad,10,56);
 mouseshow();
 c3 = 1;
 }

 if(c3 &&((x<5||x>72)||(y<55||y>70))){
 mousehide();
 drwfillbox(SET,150,5,54,72,70);
 sprintf(cad, "Salvar");
 drwstring(SET,14,150,cad,10,56);
 mouseshow();
```

```
  c3 = 0;
  }
  mousebutpress(2,&x,&y,&m1,&mbuts);
  mousebutpress(1,&x,&y,&m2,&mbuts);
  if(m1!=0||m2!=0){
  mousehide();
  drwfillbox(SET,0,5,19,72,70);
  mouseshow();
  }
  if(m2!=0&&((x>=5&&x<=72)&&(y>=19&&y<=37)))
  nuevo_archivo(&l,&cir);
  else if(m2!=0&&((x>=5&&x<=72)&&(y>=38&&y<=54)))
  carga_archivo(&l,&cir);
  else if(m2!=0&&((x>=5&&x<=72)&&(y>=55&&y<=70)))
  guarda_archivo(&l,&cir);
 }while(m1 == 0 && m2 == 0);
}

// ************************* Menu Dibujar
*****************************

void menu_dibujar(void){
c1=c2=c3=m1=m2=0;
mousehide();
 drwfillbox(SET,150,73,19,144,70);
 sprintf(cad, "Linea");
 drwstring(SET,14,150,cad,80,22);
 sprintf(cad, "Circulo");
 drwstring(SET,14,150,cad,80,39);
 sprintf(cad, "Borrar");
 drwstring(SET,14,150,cad,80,56);
 mouseshow();

 do{
  //****** Efecto de la barra de la opcion Linea ******
  mousestatus(&x,&y,&mbuts);
  if(!c1&&((x>=73&&x<=144)&&(y>=19&&y<=37))){
  mousehide();
  drwfillbox(SET,155,73,19,144,37);
  sprintf(cad, "Linea");
  drwstring(SET,14,155,cad,80,22);
  mouseshow();
  c1 = 1;
  }

  if(c1 && ((x<73||x>144)||(y<19||y>37))){
  mousehide();
  drwfillbox(SET,150,73,19,144,37);
  sprintf(cad, "Linea");
  drwstring(SET,14,150,cad,80,22);
  mouseshow();
  c1 = 0;
  }


  //****** Efecto de la barra de la opcion Circulo ******
  if(!c2&&((x>=73&&x<=144)&&(y>=38&&y<=54))){
  mousehide();
  drwfillbox(SET,155,73,38,144,54);
  sprintf(cad, "Circulo");
  drwstring(SET,14,155,cad,80,39);
```

```
  mouseshow();
  c2 = 1;
  }

  if(c2 &&((x<73||x>144)||(y<38||y>54))){
  mousehide();
  drwfillbox(SET,150,73,38,144,54);
  sprintf(cad, "Circulo");
  drwstring(SET,14,150,cad,80,39);
  mouseshow();
  c2 = 0;
  }
  //****** Efecto de la barra de la opcion Borrar ******
  if(!c3&&((x>=73&&x<=144)&&(y>=55&&y<=70))){
  mousehide();
  drwfillbox(SET,155,73,54,144,70);
  sprintf(cad, "Borrar");
  drwstring(SET,14,155,cad,80,56);
  mouseshow();
  c3 = 1;
  }

  if(c3 &&((x<73||x>144)||(y<55||y>70))){
  mousehide();
  drwfillbox(SET,150,73,54,144,70);
  sprintf(cad, "Borrar");
  drwstring(SET,14,150,cad,80,56);
  mouseshow();
  c3 = 0;
  }

  mousebutpress(2,&x,&y,&m1,&mbuts);
  mousebutpress(1,&x,&y,&m2,&mbuts);
  if(m1!=0||m2!=0){
  mousehide();
  drwfillbox(SET,0,73,19,144,70);
  mouseshow();
  }
  if(m2!=0 && x>=73&&x<=144&&y>=19&&y<=37)
  dibuja_linea();
  else if(m2!=0 && x>=73&&x<=144&&y>=38&&y<=54)
  dibuja_circulo();
  else if(m2!=0 && x>=73&&x<=144&&y>=55&&y<=70)
  funcion_borrar(&l,&cir);
 }while(m1 == 0 && m2==0);
}


// *************************** IMPRIMIR
*****************************

// Hace un recorrido completo a todo el dibujo
imprimir(struct linea *l,struct circulo *cir) {
a_punto  t;
a_centro t1;

system("copy matriz1.txt matriz.txt");

t = l -> linea_ini;
      while( t != NULL ) {
```

```
            manda_lineas((t->a1)-145,(t->b1)-49,(t->a2)-145,(t->b2)-
49);
            t = t -> sig1;
      }
t1 = cir->circulo_ini;
      while(t1 !=NULL){
            manda_cir((t1->xc)-145,(t1->yc)-49);
            t1 = t1 ->sig2;
      }
graficar();
}

// ******* Esta funcion se encargara de las líneas en la matriz
*******
void manda_lineas(int lx1, int ly1, int lx2, int ly2)
{
long int jo1,jo2;
unsigned  char  uno = '1';
nuevo = fopen("matriz.txt","r+");

if(lx1==lx2&&ly1==ly2){
 fseek(nuevo,((ly1-1)*350)+lx1,SEEK_SET);
 fwrite(&uno, sizeof(unsigned char), 1,nuevo);  }
 else if(lx1==lx2&&ly1<ly2)
  for(jo1=ly1;jo1<=ly2;jo1++){
  fseek(nuevo, ((jo1-1)*350)+lx1, SEEK_SET);
  fwrite(&uno, sizeof(unsigned char), 1,nuevo);
  fseek(nuevo, ((jo1-1)*350)+lx1+1, SEEK_SET);
  fwrite(&uno, sizeof(unsigned char), 1,nuevo); }
      else if(lx1<lx2&&ly1==ly2)
       for(jo1=lx1;jo1<=lx2;jo1++){
       fseek(nuevo, ((ly1-1)*350)+jo1, SEEK_SET);
       fwrite(&uno, sizeof(unsigned char), 1,nuevo); }
       else if(lx1==lx2&&ly1>ly2)
        for(jo1=ly2;jo1<=ly1;jo1++){
        fseek(nuevo, ((jo1-1)*350)+lx1, SEEK_SET);
        fwrite(&uno, sizeof(unsigned char), 1,nuevo);
        fseek(nuevo, ((jo1-1)*350)+lx1+1, SEEK_SET);
        fwrite(&uno, sizeof(unsigned char), 1,nuevo); }
        else if(lx1>lx2&&ly1==ly2)
            for(jo1=lx2;jo1<=lx1;jo1++){
            fseek(nuevo, ((ly1-1)*350)+jo1, SEEK_SET);
            fwrite(&uno, sizeof(unsigned char), 1,nuevo); }
            else if(lx1<lx2&&ly1<ly2)
             for(jo1=lx1,jo2=ly1;jo1<=lx2&&jo2<=ly2;jo1++,jo2++){
             fseek(nuevo, ((jo2-1)*350)+jo1, SEEK_SET);
             fwrite(&uno, sizeof(unsigned char), 1,nuevo);
             fseek(nuevo, ((jo2-1)*350)+jo1+1, SEEK_SET);
             fwrite(&uno, sizeof(unsigned char), 1,nuevo); }
             else if(lx1<lx2&&ly1>ly2)
              for(jo1=lx1,jo2=ly1;jo1<=lx2&&jo2>=ly2;jo1++,jo2--){
              fseek(nuevo, ((jo2-1)*350)+jo1, SEEK_SET);
              fwrite(&uno, sizeof(unsigned char), 1,nuevo);
              fseek(nuevo, ((jo2-1)*350)+jo1+1, SEEK_SET);
              fwrite(&uno, sizeof(unsigned char), 1,nuevo);}
              else if(lx1>lx2&&ly1>ly2)
                  for(jo1=lx2,jo2=ly2;jo1<=lx1&&jo2<=ly1;jo1++,jo2++){
                  fseek(nuevo, ((jo2-1)*350)+jo1, SEEK_SET);
                  fwrite(&uno, sizeof(unsigned char), 1,nuevo);
                  fseek(nuevo, ((jo2-1)*350)+jo1+1, SEEK_SET);
                  fwrite(&uno, sizeof(unsigned char), 1,nuevo); }
```

```
                  else if(lx1>lx2&&ly1<ly2)
                   for(jo1=lx2,jo2=ly2;jo1<=lx1&&jo2>=ly1;jo1++,jo2--){
                   fseek(nuevo, ((jo2-1)*350)+jo1, SEEK_SET);
                   fwrite(&uno, sizeof(unsigned char), 1,nuevo);
                   fseek(nuevo, ((jo2-1)*350)+jo1+1, SEEK_SET);
                   fwrite(&uno, sizeof(unsigned char), 1,nuevo); }
 fclose(nuevo);
}
// ******* Esta funcion se encargara de las donas para la matriz
******
void manda_cir(int cx1, int cy1){
  int i1;
  unsigned  char  uno = '1';
  nuevo = fopen("matriz.txt","r+");
  for(i1=1;i1<=4;i1++) {
  fseek(nuevo, ((cy1-i1)*350)+cx1, SEEK_SET);
  fwrite(&uno, sizeof(unsigned char), 1,nuevo);}

  for(i1=0;i1<=2;i1++) {
  fseek(nuevo, ((cy1+i1)*350)+cx1, SEEK_SET);
  fwrite(&uno, sizeof(unsigned char), 1,nuevo);}

  for(i1=-4;i1<=2;i1++) {
  fseek(nuevo, ((cy1+i1)*350)+(cx1-1), SEEK_SET);
  fwrite(&uno, sizeof(unsigned char), 1,nuevo);
  fseek(nuevo, ((cy1+i1)*350)+(cx1+1), SEEK_SET);
  fwrite(&uno, sizeof(unsigned char), 1,nuevo);}

  for(i1=-3;i1<=1;i1++) {
  fseek(nuevo, ((cy1+i1)*350)+(cx1-2), SEEK_SET);
  fwrite(&uno, sizeof(unsigned char), 1,nuevo);
  fseek(nuevo, ((cy1+i1)*350)+(cx1+2), SEEK_SET);
  fwrite(&uno, sizeof(unsigned char), 1,nuevo);}

  for(i1=-2;i1<=0;i1++) {
  fseek(nuevo, ((cy1+i1)*350)+(cx1-3), SEEK_SET);
  fwrite(&uno, sizeof(unsigned char), 1,nuevo);
  fseek(nuevo, ((cy1+i1)*350)+(cx1+3), SEEK_SET);
  fwrite(&uno, sizeof(unsigned char), 1,nuevo);}

  fclose(nuevo);
}

//** Esta funcion se encarga de mandar las instrucciones al graficador
**
//************ la informacion la lee del archivo matriz.txt
*************

void graficar(void)
{
 int cad2[]={0xa0,0x80,0x90,0x10,0x50,0x40,0x60,0x20},j=0,i=0;
 int cad1[]={0x0a,0x08,0x09,0x01,0x05,0x04,0x06,0x02};
 int motor1,cont1,cont,cmot2;
 unsigned char dato;

//********************* Inicializa los puertos
*************************
 outportb(0x303,0x80);
 delay(100);
 outportb(0x300,0x00);
 delay(100);
```

```
 outportb(0x301,0x00);
 delay(100);

//************** se preparan los motores de pasos
*********************

 outportb(0x301,2);
 delay(50);

 for(i=0;i<=7;i++) {
      outportb(0x300,cad1[i]);
      delay(50);      }
 for(i=7;i>=0;i--) {
      outportb(0x300,cad1[i]);
      delay(50);      }
 for(i=0;i<=7;i++) {
      outportb(0x300,cad2[i]);
      delay(50);      }
 for(i=7;i>=0;i--) {
      outportb(0x300,cad2[i]);
      delay(50);      }


 cont1=1; cmot2=cont=motor1=0;
 nuevo = fopen("matriz.txt","r"); // Se abre el archivo del diseño

 while(cont<=399){
 while(dato!=EOF){
 fseek(nuevo,cont1+(cont*350),SEEK_SET);  //aqui se controla la
posición
 fread(&dato,sizeof(unsigned char), 1,nuevo); //de la matriz
 if(dato=='0')      {      // Si el dato es cero el plumón se mantiene
arriba
      outportb(0x301,2);
      delay(40);     }
 if(dato=='1')      {      // Si es uno el plumón se baja para marcar
      outportb(0x301,1);
      delay(40);     }
 outportb(0x300,cad1[motor1]);
 delay(40);
 motor1++;
 if(motor1==8) motor1=0;
 cont1++;
 if(cont1==351) break;
      }

 cont1=350; motor1=motor1-1;
 while(dato!=EOF){
 outportb(0x300,cad1[motor1]);
 delay(40);
 fseek(nuevo,cont1+(cont*350),SEEK_SET);
 fread(&dato,sizeof(unsigned char), 1,nuevo);
 if(dato=='0')      {
      outportb(0x301,2);
      delay(40);     }
 if(dato=='1')      {
      outportb(0x301,1);
      delay(40);     }
 motor1--;
 if(motor1==-1) motor1=7;
 cont1--;
```

```
 if(cont1==0){
 cont++; break;}  }

 for(i=0;i<=9;i++) {
      outportb(0x300,cad2[cmot2]);
      delay(40); cmot2++;
      if(cmot2==8) cmot2=0;}

 cont1=1; motor1=0;
 }

 outportb(0x300,0x00);
 outportb(0x301,0x00);
 fclose(nuevo);

}
// ********************* Guarda en memoria la linea
*******************
void funcion(int z1,int z2,int z3,int z4){
 a_punto   p;
 p = crear_punto(z1,z2,z3,z4);
 agregar_punto(&l, p);
}

// ******************** Guarda en memoria el circulo
******************
void funcion2(int zx,int zy){
 a_centro cir1;
 cir1 = crear_centro(zx,zy);
 agregar_centro(&cir,cir1);
}
// ************************** Dibuja Círculos
*************************
void dibuja_circulo(void){
 n=n1=0;
 mousehide();
 mousecursorset(&miomousecursor);
 mouseshow();
 mousestatus(&x,&y,&mbuts);
 mouserangeset(149,53,490,445);
 mouselocset(x,y);
 do{
      mousebutpress(1,&x,&y,&n,&mbuts);
      if(n!=0){
      mousehide();
      drwfillcircle(OR,14,x,y,3);
      funcion2(x,y);

      drwfillbox(SET,200,0,19,640,480);
      drwfillbox(SET,25,139,43,501,455);
      drwfillbox(SET,0,145,49,495,449);

      recorre_dibujo(&l,&cir);
      mouseshow();
      }
      mousestatus(&x,&y,&mbuts);
      sprintf(cad,"X1=%3d Y1=%3d",x-145,y-49);
      drwstring(SET,15,150,cad,380,5);
 mousebutpress(2,&x,&y,&n1,&mbuts);
 }while(n1==0);
mousehide();
```

```
mousecursorset(&manomousecursor);
mouseshow();
mouserangeset(0,0,639,479);
mouselocset(x,y);
n=n1=0;
}

// ************************* Dibuja Líneas
*****************************
void dibuja_linea(void){
n=n1=n2=f=0;
mousehide();
mousecursorset(&miomousecursor);
mouseshow();
mousestatus(&x,&y,&mbuts);
mouserangeset(146,50,493,448);
mouselocset(x,y);
do
{
 mousebutpress(1,&x,&y,&n,&mbuts);
  if(n!=0){
  mousehide();
  mousestatus (&x,&y,&mbuts);
  x1 = x;
  y1 = y;
  f = 1;
  mouseshow();
  }
 mousebutrelease(1,&x,&y,&n1,&mbuts);
  if(n1 != 0){
  mousehide();
      if(x == x1 && y == y1){
      drwline(OR,4,x1,y1,x,y);
      funcion(x1,y1,x,y);
      }
      if(x != x1 && y == y1){
      drwline(OR,4,x1,y1,x,y);
      funcion(x1,y1,x,y);
      }
      if(x == x1 && y != y1){
      drwline(OR,4,x1,y1,x,y);
      funcion(x1,y1,x,y);
      }
      if(x != x1 && y != y1 && abs(x-x1) == abs(y-y1)){
      drwline(OR,4,x1,y1,x,y);
      funcion(x1,y1,x,y);
      }

      drwfillbox(SET,200,0,19,640,480);
      drwfillbox(SET,25,139,43,501,455);
      drwfillbox(SET,0,145,49,495,449);

      recorre_dibujo(&l,&cir);
      drwfillbox(SET,150,0,3,639,18);
      sprintf(cad, "Archivo");
      drwstring(SET,14,150,cad,10,5);
      sprintf(cad, "Dibujar");
      drwstring(SET,14,150,cad,80,5);
      sprintf(cad, "Imprimir");
      drwstring(SET,14,150,cad,150,5);
      sprintf(cad, "Salir");
```

```
      drwstring(SET,14,150,cad,230,5);

      mouseshow();
      f=0;
  }
 mousestatus(&x,&y,&mbuts);
  if(f && (x1 != x || y1 != y)){
  mousestatus(&x,&y,&mbuts);
  drwline(SET,15,x1,y1,x,y);
  sdelay(2);
  drwline(AND,0,x1,y1,x,y);
  sprintf(cad,"X1=%3d Y1=%3d X2=%3d Y2=%3d ",x1-145,y1-49,x-145,y-49);
  drwstring(SET,15,150,cad,380,5);
  }
  mousestatus(&x,&y,&mbuts);
  if(!f){
  sprintf(cad,"X1=%3d Y1=%3d ",x-145,y-49);
  drwstring(SET,15,150,cad,380,5);
  }
 mousebutpress(2,&x,&y,&n2,&mbuts);
 }while(n2 == 0);
 mousehide();
 mousecursorset(&manomousecursor);
 mouseshow();
 mouserangeset(0,0,639,479);
 mouselocset(x,y);
 n=n1=n2=0;
}




// ************************ Funcion para Borrar
***********************
void funcion_borrar(struct linea *l, struct circulo *cir){
  a_punto t,t1;
  a_centro c,c1;
  char s,cad[40];
  n=n1=n2=f=0;
  mousehide();
  mousecursorset(&miomousecursor);
  mouseshow();
  mousestatus(&x,&y,&mbuts);
  mouserangeset(146,50,494,448);
  mouselocset(x,y);
  do{
  mousebutpress(1,&x,&y,&n,&mbuts);
  if(n!=0&&f==0){

  mousehide();
  mousestatus (&x,&y,&mbuts);
  x1 = x;
  y1 = y;
  f = 1;

  t = t1 = l->linea_ini;
  while(t1 != NULL){
  if(t1->a1==t1->a2&&t1->a1==x1&&t1->b1<=t1->b2&&y1>=t1->b1&&y1<=t1-
>b2){
       drwline(SET,2,t1->a1,t1->b1,t1->a2,t1->b2);
```

```
       sprintf(cad,"Desea borrar la linea resaltada [s/n]: ");
       drwstring(SET,15,150,cad,150,100);
       fflush(stdin);
       s = '\x0';
       s = getch();
       fflush(stdin);
       if(s == 's'||s =='S') {
            if(t1 == l->linea_ini)
              l->linea_ini = t1->sig1;
            else
              t->sig1 = t1->sig1;
            free(t1);
            break;
       }
  drwline(SET,14,t1->a1,t1->b1,t1->a2,t1->b2);
  s='\x0';
  }
  if(t1->a1==t1->a2&&t1->a1==x1&&t1->b2<=t1->b1&&y1>=t1->b2&&y1<=t1-
>b1){
       drwline(SET,2,t1->a1,t1->b1,t1->a2,t1->b2);
       sprintf(cad,"Desea borrar la linea resaltada [s/n]: ");
       drwstring(SET,15,150,cad,150,100);
       fflush(stdin);
       s = '\x0';
       s = getch();
       fflush(stdin);
       if(s == 's'||s =='S') {
            if(t1 == l->linea_ini)
              l->linea_ini = t1->sig1;
            else
              t->sig1 = t1->sig1;
            free(t1);
            break;
       }
  drwline(SET,14,t1->a1,t1->b1,t1->a2,t1->b2);
  s='\x0';
  }
  if(t1->b1==t1->b2&&t1->b1==y1&&t1->a1<=t1->a2&&x1>=t1->a1&&x1<=t1-
>a2){
       drwline(SET,2,t1->a1,t1->b1,t1->a2,t1->b2);
       sprintf(cad,"Desea borrar la linea resaltada [s/n]: ");
       drwstring(SET,15,150,cad,150,100);
       fflush(stdin);
       s = '\x0';
       s = getch();
       fflush(stdin);
       if(s == 's'||s =='S') {
            if(t1 == l->linea_ini)
              l->linea_ini = t1->sig1;
            else
              t->sig1 = t1->sig1;
            free(t1);
            break;
       }
  drwline(SET,14,t1->a1,t1->b1,t1->a2,t1->b2);
  s='\x0';
  }
  if(t1->b1==t1->b2&&t1->b1==y1&&t1->a2<=t1->a1&&x1>=t1->a2&&x1<=t1-
>a1){
       drwline(SET,2,t1->a1,t1->b1,t1->a2,t1->b2);
       sprintf(cad,"Desea borrar la linea resaltada [s/n]: ");
```

```
       drwstring(SET,15,150,cad,150,100);
       fflush(stdin);
       s = '\x0';
       s = getch();
       fflush(stdin);
       if(s == 's'||s =='S') {
            if(t1 == l->linea_ini)
              l->linea_ini = t1->sig1;
            else
              t->sig1 = t1->sig1;
            free(t1);
            break;
       }
  drwline(SET,14,t1->a1,t1->b1,t1->a2,t1->b2);
  s='\x0';
  }
  if(t1->a1<t1->a2&&t1->b1<t1->b2&&(x1-t1->a1)==(y1-t1->b1)&&t1-
>a1<=x1&&
            x1<=t1->a2&&t1->b1<=y1&&y1<=t1->b2){
       drwline(SET,2,t1->a1,t1->b1,t1->a2,t1->b2);
       sprintf(cad,"Desea borrar la linea resaltada [s/n]: ");
       drwstring(SET,15,150,cad,150,100);
       fflush(stdin);
       s = '\x0';
       s = getch();
       fflush(stdin);
       if(s == 's'||s =='S') {
            if(t1 == l->linea_ini)
              l->linea_ini = t1->sig1;
            else
              t->sig1 = t1->sig1;
            free(t1);
            break;
       }
  drwline(SET,14,t1->a1,t1->b1,t1->a2,t1->b2);
  s='\x0';
  }
  if(t1->a2<t1->a1&&t1->b2<t1->b1&&(x1-t1->a2)==(y1-t1->b2)&&t1-
>a2<=x1&&
            x1<=t1->a1&&t1->b2<=y1&&y1<=t1->b1){
       drwline(SET,2,t1->a1,t1->b1,t1->a2,t1->b2);
       sprintf(cad,"Desea borrar la linea resaltada [s/n]: ");
       drwstring(SET,15,150,cad,150,100);
       fflush(stdin);
       s = '\x0';
       s = getch();
       fflush(stdin);
       if(s == 's'||s =='S') {
            if(t1 == l->linea_ini)
              l->linea_ini = t1->sig1;
            else
              t->sig1 = t1->sig1;
            free(t1);
            break;
       }
  drwline(SET,14,t1->a1,t1->b1,t1->a2,t1->b2);
  s='\x0';
  }
  if(t1->a1<t1->a2&&t1->b2<t1->b1&&(x1-t1->a1)==(t1->b1-y1)&&t1-
>a1<=x1&&
            x1<=t1->a2&&t1->b2<=y1&&y1<=t1->b1){
```

```
       drwline(SET,2,t1->a1,t1->b1,t1->a2,t1->b2);
       sprintf(cad,"Desea borrar la linea resaltada [s/n]: ");
       drwstring(SET,15,150,cad,150,100);
       fflush(stdin);
       s = '\x0';
       s = getch();
       fflush(stdin);
       if(s == 's'||s =='S') {
            if(t1 == l->linea_ini)
              l->linea_ini = t1->sig1;
            else
              t->sig1 = t1->sig1;
            free(t1);
            break;
       }
  drwline(SET,14,t1->a1,t1->b1,t1->a2,t1->b2);
  s='\x0';
  }
  if(t1->a2<t1->a1&&t1->b1<t1->b2&&(x1-t1->a2)==(t1->b2-y1)&&t1-
>a2<=x1&&
            x1<=t1->a1&&t1->b1<=y1&&y1<=t1->b2){
       drwline(SET,2,t1->a1,t1->b1,t1->a2,t1->b2);
       sprintf(cad,"Desea borrar la linea resaltada [s/n]: ");
       drwstring(SET,15,150,cad,150,100);
       fflush(stdin);
       s = '\x0';
       s = getch();
       fflush(stdin);
       if(s == 's'||s =='S') {
            if(t1 == l->linea_ini)
              l->linea_ini = t1->sig1;
            else
              t->sig1 = t1->sig1;
            free(t1);
            break;
       }
  drwline(SET,14,t1->a1,t1->b1,t1->a2,t1->b2);
  s='\x0';
  }
  t = t1;
  t1 = t1->sig1;
  }

  c = c1 = cir->circulo_ini;
  while(c1!=NULL){
  if(x1>=(c1->xc-2)&&x1<=(c1->xc+2)&&y1>=(c1->yc-2)&&y1<=(c1->yc+2)){
      drwfillcircle(SET,2,c1->xc,c1->yc,3);
      sprintf(cad,"Desea borrar el circulo resaltado [s/n]: ");
       drwstring(SET,15,150,cad,150,100);
       fflush(stdin);
       s = '\x0';
       s = getch();
       fflush(stdin);
       if(s == 's'||s =='S') {
            if(c1 == cir->circulo_ini)
              cir->circulo_ini = c1->sig2;
            else
              c->sig2 = c1->sig2;
            free(c1);
            break;
       }
```

```
  drwfillcircle(SET,14,c1->xc,c1->yc,3);
  s='\x0';
  }
  c = c1;
  c1 = c1->sig2;
  }
  fflush(stdin);

  drwfillbox(SET,200,0,19,640,480);
  drwfillbox(SET,25,139,43,501,455);
  drwfillbox(SET,0,145,49,495,449);
  recorre_dibujo(l,cir);
  mouseshow();
  }
  mousebutrelease(1,&x,&y,&n1,&mbuts);
  if(n1 != 0) f = 0;
  s='\x0';
  mousestatus(&x,&y,&mbuts);
  sprintf(cad,"X1=%3d Y1=%3d",x,y);
  drwstring(SET,15,150,cad,380,5);
  mousebutpress(2,&x,&y,&n2,&mbuts);
  fflush(stdin);
  s='\x0';

  }while(n2 == 0);
 mousehide();
 mousecursorset(&manomousecursor);
 mouseshow();
 mouserangeset(0,0,639,479);
 mouselocset(x,y);
 n=n1=n2=0;
}
```

LISTAS.H

```
# ifndef _LISTAS_H_
# define _LISTAS_H_

struct punto {
      int a1,b1,a2,b2;
      struct punto *sig1;
};

typedef struct punto *a_punto;

struct linea {
 a_punto linea_ini;
};

struct centro {
      int xc,yc;
      struct centro *sig2;
};

typedef struct centro *a_centro;

struct circulo{
 a_centro circulo_ini;
};

a_punto crear_punto(int  a, int  b, int aa, int bb);
```

```
void agregar_punto(struct  linea  *l, a_punto  p);
void  recorre_dibujo(struct linea *l,struct circulo *cir);
void iniciacirculo(struct circulo *q);
void inicialinea(struct linea *l);
a_centro crear_centro(int zx,int zy);
void agregar_centro(struct circulo *cir,a_centro cent);
void guarda_archivo(struct linea *l,struct circulo *cir);
void carga_archivo(struct linea *l,struct circulo *cir);
void nuevo_archivo(struct linea *l,struct circulo *cir);
# endif
```

LISTAS.CPP

```
#include <stdio.h>
#include <alloc.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include "svgacc.h"
#include "listas.h"

void inicialinea(struct linea *l){
l->linea_ini = NULL;
}

a_punto crear_punto(int  a, int  b, int aa, int bb) {
a_punto  t;
t = (a_punto) malloc(sizeof(struct punto));
t -> a1 = a;
t -> b1 = b;
t -> a2 = aa;
t -> b2 = bb;
t -> sig1 = NULL;
return t;
}

void agregar_punto(struct  linea  *l, a_punto  p) {
a_punto t;
t = l -> linea_ini;
l -> linea_ini = p;
p -> sig1 = t;
}

// Inicializa la estructura circulo
void iniciacirculo(struct circulo *q){
q->circulo_ini = NULL;
}

// Crea una variable y reserva memoria
a_centro crear_centro(int zx,int zy){
a_centro t;
```

```
t = (a_centro) malloc(sizeof(struct centro));
t -> xc = zx;
t -> yc = zy;
t -> sig2 = NULL;
return t;
}

// Funcion para agregar un elemento a la estructura circulo
void agregar_centro(struct circulo *circ,a_centro cent){
a_centro t;
t = circ -> circulo_ini;
circ -> circulo_ini = cent;
cent -> sig2 = t;
}

// Hace un recorrido completo a todo el dibujo
void  recorre_dibujo(struct linea *l,struct circulo *cir) {
a_punto  t;
a_centro t1;

t = l -> linea_ini;
      while( t != NULL ) {
            drwline(OR,14,t->a1,t->b1,t->a2,t->b2);
            t = t -> sig1;
      }
t1 = cir->circulo_ini;
      while(t1 !=NULL){
            drwfillcircle(OR,14,t1->xc,t1->yc,3);
            t1 = t1 ->sig2;
      }
}

// ********************** Guarda un Archivo ***********************
void guarda_archivo(struct linea *l,struct circulo *cir) {
int  i = 0, j = 0;
FILE *in;
a_punto  t;
a_centro t1;
char cad1[30]="",*cad2=".gra",*ptr,cad[30],c[4],c1;
 mousehide();
 if(l->linea_ini!=NULL||cir->circulo_ini!=NULL)
 {
 sprintf(cad,"Nombre: ");
 drwfillbox(SET,150,120,80,500,120);
 drwstring(SET,15,0,cad,130,100);
 c1 = getch();
 if(c1 != '\r'|| c1 != '\x1B')
  while(c1 != '\r'&& c1 != '\x1B'){
      if(c1 == '\b'&& i>0){
      j = i = i-1;
       while(cad1[j]!='\x0'){
       cad1[j]='\x0'; j++;     }
      i--;
      }
  else
  cad1[i] = c1;
  drwfillbox(SET,150,194,99,500,119);
  j = strcmp(cad1,"\b");
  if(j == 0) {cad1[0]='\x0'; i=0;}
  if(j!=0)
  drwstring(SET,15,0,cad1,194,100);
```

```
  fflush(stdin);
  c1 = getch();
  i++;
  j = strcmp(cad1,"");
  if(j==0) i=0;
  }
  if(c1=='\x1B')
      drwfillbox(SET,0,120,80,500,120);
 else{
 ptr = strstr(cad1,cad2);
 if(ptr == NULL) strcat(cad1,cad2);


      if ((in = fopen(cad1, "wt"))== NULL)
      {
       sprintf(cad,"Error en el archivo %s",cad1);
       drwstring(SET,15,0,cad,180,200);
       fflush(stdin);
       getch();
       sprintf(cad,"
");
       drwstring(SET,15,0,cad,180,200);
      }
      else {
      t = l -> linea_ini;
      while( t != NULL ) {
        fputc('l',in);
        t->a1;
        itoa(t->a1,c,10);
            j=0;
            while(c[j] != '\x0'){
            fputc(c[j],in); j++;}
        fputc(',',in);
        t->b1;
        itoa(t->b1,c,10);
            j=0;
            while(c[j] != '\x0'){
            fputc(c[j],in); j++;}
        fputc(',',in);
        t->a2;
        itoa(t->a2,c,10);
            j=0;
            while(c[j] != '\x0'){
            fputc(c[j],in); j++;}
        fputc(',',in);
        t->b2;
        itoa(t->b2,c,10);
            j=0;
            while(c[j] != '\x0'){
            fputc(c[j],in); j++;}
        t = t -> sig1;
      }
      t1 = cir->circulo_ini;
      while(t1 !=NULL){
      fputc('c',in);
      t1->xc;
      itoa(t1->xc,c,10);
            j=0;
            while(c[j] != '\x0'){
            fputc(c[j],in); j++;}
      fputc(',',in);
```

```
      t1->yc;
      itoa(t1->yc,c,10);
            j=0;
            while(c[j] != '\x0'){
            fputc(c[j],in); j++;}
      t1 = t1 ->sig2;
  }
fputc('f',in);
fclose(in);
}

drwfillbox(SET,200,0,19,640,480);
drwfillbox(SET,25,139,43,501,455);
drwfillbox(SET,0,145,49,495,449);
recorre_dibujo(l,cir);
}
}  //   Llave final del primer if
mouseshow();
}

// ********************** Crea un Nuevo Archivo ********************
void nuevo_archivo(struct linea *l,struct circulo *cir){
a_punto t,t1;
a_centro t2,t3;
 char cad[40],c1;
 mousehide();
 if(l->linea_ini != NULL||cir->circulo_ini!=NULL){
 drwfillbox(SET,150,150,80,480,145);
 sprintf(cad,"-Advertencia!");
 drwstring(SET,15,150,cad,260,90);
 sprintf(cad,"¨Desea continuar sin grabar los cambios");
 drwstring(SET,15,150,cad,160,104);
 sprintf(cad,"del archivo en uso [s/n]: ");
 drwstring(SET,15,150,cad,220,118);
 fflush(stdin);
 c1 = getch();
 fflush(stdin);
 if(c1 == 's' || c1 == 'S'){
      mousehide();
      t = l -> linea_ini;
       while(t != NULL){
       t1 = t;
       t = t -> sig1;
       free(t1);
       }
      l -> linea_ini = NULL;
      t2 = cir->circulo_ini;
       while(t2 != NULL){
       t3 = t2;
       t2 = t2 ->sig2;
       free(t3);
       }
      cir -> circulo_ini = NULL;
      drwfillbox(SET,200,0,19,640,480);
      drwfillbox(SET,25,139,43,501,455);
      drwfillbox(SET,0,145,49,495,449);
      mouseshow();
      }
 }

drwfillbox(SET,200,0,19,640,480);
```

```
drwfillbox(SET,25,139,43,501,455);
drwfillbox(SET,0,145,49,495,449);
recorre_dibujo(l,cir);
mouseshow();
}
// ********************** Carga un Archivo **********************
void carga_archivo(struct linea *l,struct circulo *cir){
int i = 0, j = 0,x1,y1,x2,y2;
FILE *in;
a_punto  t,t1;
a_centro t2,t3;
char cad1[40]="",*cad2=".gra",*ptr,cad[40],c[4],c1;
 mousehide();
 if(l->linea_ini != NULL||cir->circulo_ini!=NULL){
 drwfillbox(SET,150,150,80,480,145);
 sprintf(cad,"-Advertencia!");
 drwstring(SET,15,150,cad,260,90);
 sprintf(cad,"¨Desea continuar sin grabar los cambios");
 drwstring(SET,15,150,cad,160,104);
 sprintf(cad,"del archivo en uso [s/n]: ");
 drwstring(SET,15,150,cad,220,118);
 c1 = getch();
 if(c1 == 's' || c1 == 'S'){
      t = l -> linea_ini;
       while(t != NULL){
       t1 = t;
       t = t -> sig1;
       free(t1);
       }
      l -> linea_ini = NULL;
      t2 = cir->circulo_ini;
       while(t2 != NULL){
       t3 = t2;
       t2 = t2 ->sig2;
       free(t3);
       }
      cir -> circulo_ini = NULL;
      }
      drwfillbox(SET,200,0,19,640,480);
      drwfillbox(SET,25,139,43,501,455);
      drwfillbox(SET,0,145,49,495,449);
 }

 if(l->linea_ini == NULL && cir->circulo_ini == NULL){
 sprintf(cad,"Nombre: ");
 drwfillbox(SET,150,120,80,500,120);
 drwstring(SET,15,0,cad,130,100);
 strcpy(cad1,"");
 cad1[0] = '\x0';
 mousehide();
 c1 = getch();
 if(c1 != '\r'|| c1 != '\x1B')
  while(c1 != '\r'&& c1 != '\x1B'){
      if(c1 == '\b'&& i>0){
      j = i = i-1;
       while(cad1[j]!='\x0'){
       cad1[j]='\x0'; j++;     }
      i--;
      }
  else
  cad1[i] = c1;
```

```
  drwfillbox(SET,150,194,99,500,119);
  j = strcmp(cad1,"\b");
  if(j == 0) {cad1[0]='\x0'; i=0;}
  if(j!=0)
  drwstring(SET,15,0,cad1,194,100);
  c1 = getch();
  i++;
  j = strcmp(cad1,"");
  if(j==0) i=0;
  }
  if(c1=='\x1B'){
      drwfillbox(SET,200,0,19,640,480);
      drwfillbox(SET,25,139,43,501,455);
      drwfillbox(SET,0,145,49,495,449);
  }
 else{
 ptr = strstr(cad1,cad2);
 if(ptr == NULL) strcat(cad1,cad2);
 if ((in = fopen(cad1, "rt"))== NULL)
      {
       sprintf(cad,"Error en el archivo %s",cad1);
       drwstring(SET,15,0,cad,300,200);
       getch();
       sprintf(cad,"
");
       drwstring(SET,15,0,cad,300,200);
      }
  else {

      c1 = fgetc(in);
      while(c1=='l') {
        j = 0;
        c1 =fgetc(in);
        while(c1!=','){
        c[j] = c1;
        c1 = fgetc(in);
        j++;          }
        x1 = atoi(c);
        for(j=0;j<4;j++)
            c[j] = '\x0';
        j = 0;
        c1 = fgetc(in);
        while(c1!=','){
        c[j] = c1;
        c1 = fgetc(in);
        j++;          }
        y1 = atoi(c);
        for(j=0;j<4;j++)
            c[j] = '\x0';
        j = 0;
        c1 =fgetc(in);
        while(c1!=','){
        c[j] = c1;
        c1 = fgetc(in);
        j++;          }
        x2 = atoi(c);
        for(j=0;j<4;j++)
            c[j] = '\x0';
        j = 0;
        c1 = fgetc(in);
        while(c1!='l'&&c1!='c'&&c1!='f'){
```

```
        c[j] = c1;
        c1 = fgetc(in);
        j++;          }
        y2 = atoi(c);
        for(j=0;j<4;j++)
            c[j] = '\x0';
        t = crear_punto(x1,y1,x2,y2);
        agregar_punto(l,t);
      }
      while(c1=='c') {
        j = 0;
        c1 =fgetc(in);
        while(c1!=','){
        c[j] = c1;
        c1 = fgetc(in);
        j++;          }
        x1 = atoi(c);
        for(j=0;j<4;j++)
            c[j] = '\x0';
        j = 0;
        c1 = fgetc(in);
        while(c1!='c'&&c1!='f'){
        c[j] = c1;
        c1 = fgetc(in);
        j++;          }
        y1 = atoi(c);
        for(j=0;j<4;j++)
            c[j] = '\x0';
        t2 = crear_centro(x1,y1);
        agregar_centro(cir,t2);
      }
      if(c1=='f')
      fclose(in);
      }
  fclose(in);
  }
 }
drwfillbox(SET,200,0,19,640,480);
drwfillbox(SET,25,139,43,501,455);
drwfillbox(SET,0,145,49,495,449);
recorre_dibujo(l,cir);
mouseshow();
}
```

BIBLIOGRAFÍA

[1] Como Programar en C/C++ Segunda Edición.
H. M. Deitel, P. J. Deitel, Editorial Prentice Hall, 1995.

[2] Electrónica Industrial Moderna Tercera Edición.
Timothy J. Maloney Editorial Prentice Hall, 1997.

[3] Estructuras de Datos con C y C++ Segunda Edición.
Yedidyah Langsam, Aarón M. Tenenbaum, Editorial Prentice Hall, 1997.

[4] Interfacing Sensors to The IBM PC.
Willis J. Tompkins, John G. Webster, Editorial Prentice Hall, 1988.

[5] Los Microprocesadores Intel, Arquitectura Programación e Interfaces.
Barry B. Brey, Editorial Prentice Hall, 1994.

[6] Principios de Electrónica Quinta Edición.
Albert Paul Malvino, Editorial McGraw-Hill, 1993.

[7] Control de Posición de un Motor con Carga Variable
Felipe Santiago Espinosa, Tesis de Licenciatura. Universidad Autónoma de Puebla,
1997.