UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
CAMPUS ARAGÓN

"DISEÑO Y CONSTRUCCIÓN DE UN MODELADOR FÍSICO TRIDIMENSIONAL"

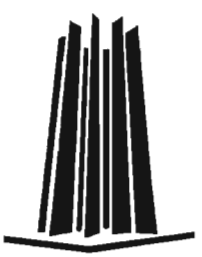
T E S I S

QUE PARA OBTENER EL TÍTULO DE:
INGENIERO EN COMPUTACIÓN

P R E S E N T A:

FELIPE DE JESÚS GUTIÉRREZ LÓPEZ

ASESOR DE TESIS:
M. EN C. MARCELO PÉREZ MEDEL



MÉXICO, [redacted] 2005

Agradecimientos:

"A Dios Por Permitirme Ser, Al Lado De Tan Bellos Seres"

PAG.

INDICE

PAG.

PAG.

PAG.

INTRODUCCIÓN

En la últimas décadas la tecnología ha avanzado de manera muy acelerada y sin duda lo que resulta más impresionante es el terreno que ha ganado la computadora y todo aquello que se comunica con ella, de tal manera que hoy en día es difícil imaginar el mundo sin esas máquinas revolucionarias.

En un principio las computadoras eran artefactos gigantescos, cuyo objetivo primordial se enfocaba al procesamiento de cálculos matemáticos bélicos y su costo de elaboración y mantenimiento eran muy elevados. Con el transcurrir del tiempo, el perfeccionamiento tecnológico redujo el tamaño y precio a tal grado, que empezaron a ocupar los espacios de escritorios en hogares y oficinas.

Para aprovechar las capacidades de las computadoras se han desarrollado una serie de accesorios auxiliares (periféricos) que permiten llevar a cabo tareas que resultan cotidianas como: imprimir, escanear, fotografiar, conectar a internet, almacenar grandes volúmenes de información, escuchar y grabar música, entre otras.

Los periféricos más comunes tienen un costo relativamente bajo en comparación con otros cuya existencia en el mercado resulta tan desconocida e inalcanzable como en el caso de: igualadores de color, bordadores, impresores de serigrafía y modeladores físicos.

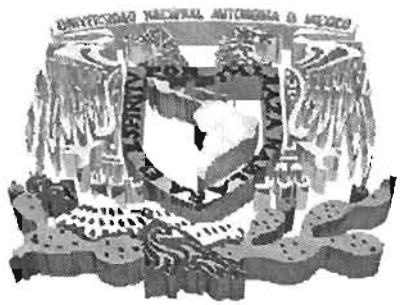
En el transcurso de la presente tesis, veremos que con los conocimientos adquiridos en la licenciatura en ingeniería en computación, es posible construir un modelador físico tridimensional reutilizando componentes de otros dispositivos en desuso (impresoras, escáners y ratones), además de que también ocuparemos chips comerciales con funciones específicas y programaremos otros según nuestras necesidades mediante un lenguaje de programación de hardware (VHDL).

El modelador físico tridimensional al que desde este momento llamaremos "Mofis", estará compuesto por dos partes: una máquina electromecánica periférica ("hardware") y un programa de computadora para dirigirla ("software").

El software a desarrollar ofrecerá la posibilidad de cargar imágenes planas, aplicarles diversos efectos especiales y generar su equivalente en tres dimensiones (campo de altura) de manera visual en la pantalla de un monitor, y de forma física en un bloque de madera que será esculpido por la máquina.

Finalmente hay que destacar que la información incluida en los capítulos siguientes no sólo pretende dar los conocimientos necesarios para la construcción de un solo tipo de periférico (un modelador), sino que también servirá de base para otro tipo de proyectos en los que la imaginación es el límite.

CAPÍTULO I



FUNDAMENTOS DE IMÁGENES Y MODELOS DIGITALES

Los seres humanos percibimos nuestro medio ambiente a través de los sentidos, y una parte importante de los datos que son enviados a nuestro cerebro son capturados por medio del sentido de la vista.

La palabra "imagen" nos hace pensar de manera inconsciente en un fragmento visual de nuestro entorno, que ha sido capturado en un determinado instante de tiempo.

Por medio de las imágenes podemos conocer ciertas propiedades de los objetos que nos rodean, tales como: su posición, color, tamaño, consistencia, entre otras.

Gracias a la información valiosa que nos ofrecen las imágenes sobre nuestro entorno, las utilizaremos como materia prima para la construcción de modelos físicos con relieve, y para ello definiremos a continuación algunos conceptos importantes.

1.1 CONCEPTOS BÁSICOS DE IMÁGENES

1.1.1 IMAGEN

Cuando escuchamos la palabra "imagen", pensamos en una fotografía, en la portada de una revista o periódico, en la escena de una película, en un póster, en el monitor de una computadora, en la televisión, etc.

Una imagen es un conjunto de puntos distribuidos sobre una superficie y además, cada uno de ellos tiene un color asociado. Dicho de otra manera, una imagen es un arreglo bidimensional de puntos cuyas propiedades son: posición y color[1].

Las imágenes las podemos clasificar de diversas maneras, ya sea dependiendo del número de colores que manejen, del lugar en el que se encuentren plasmadas, de la continuidad entre sus puntos, etc.

A continuación describiremos la clasificación de las imágenes según la continuidad entre sus puntos.

1.1.2 IMAGEN ANALÓGICA

Las cámaras convencionales como las que utilizan rollos fotográficos permiten obtener imágenes analógicas, en ellas todos los puntos mantienen continuidad, eso quiere decir que si ubicamos dos puntos cualesquiera sobre la superficie podremos darnos cuenta que dentro de éstos siempre existe un tercero.

En las imágenes de este tipo si aumentamos alguna región, no hay pérdida en la información gracias al fenómeno de continuidad antes mencionado. Veamos un ejemplo:

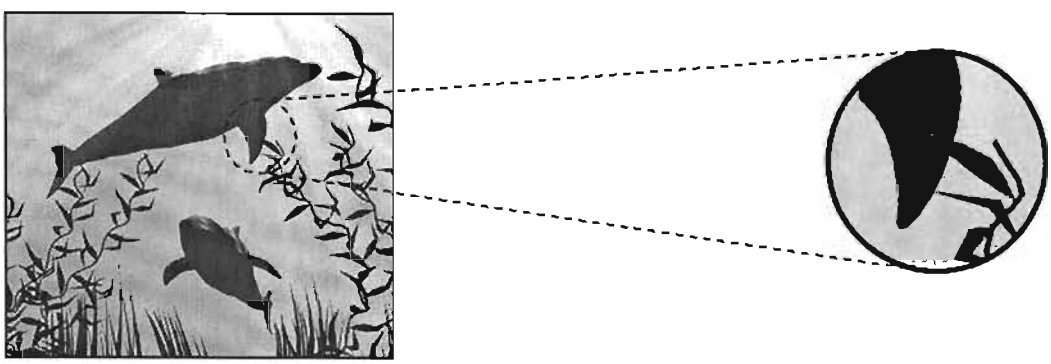


Fig. 1.1 La continuidad entre los puntos de una imagen analógica, nos permite amplificar cualquier zona sin que exista pérdida de información

[1] El negro por definición es la ausencia de color, pero lo consideraremos como tal para facilitar el manejo descriptivo.

Es preciso señalar que la imagen mostrada en la parte superior, fue introducida a la computadora con la ayuda de un escáner y por lo tanto se convirtió en digital, sin embargo, para efectos prácticos consideraremos que es analógica aunque en realidad su naturaleza haya cambiado.

1.1.3 PUNTO

La definición de punto señala que: "es el lugar en el que se produce la intersección de dos líneas".

Una imagen de tipo analógica está compuesta de muchos puntos situados a lo largo de su superficie, los cuales además de tener una posición también tienen un color determinado.

1.1.4 IMAGEN DIGITAL

Las imágenes digitales son aquellas que podemos encontrar en los monitores de computadoras, documentos elaborados por impresoras de inyección de tinta, láser, entre otros.

La diferencia entre las imágenes analógicas y las digitales radica en que las segundas, se componen de puntos discretos (ahora llamados "pixeles"), lo que significa que carecen de continuidad. Por lo tanto si ubicamos dos puntos sobre una superficie no siempre existirá un tercero entre ellos.

La discontinuidad de las imágenes digitales origina que la calidad de las mismas dependa en gran parte de la cantidad de pixeles de los que se componga, por lo que si aumentamos una región de la misma, será posible percibir que existe una pérdida de nitidez, tal y como se refleja en la figura 1.2:

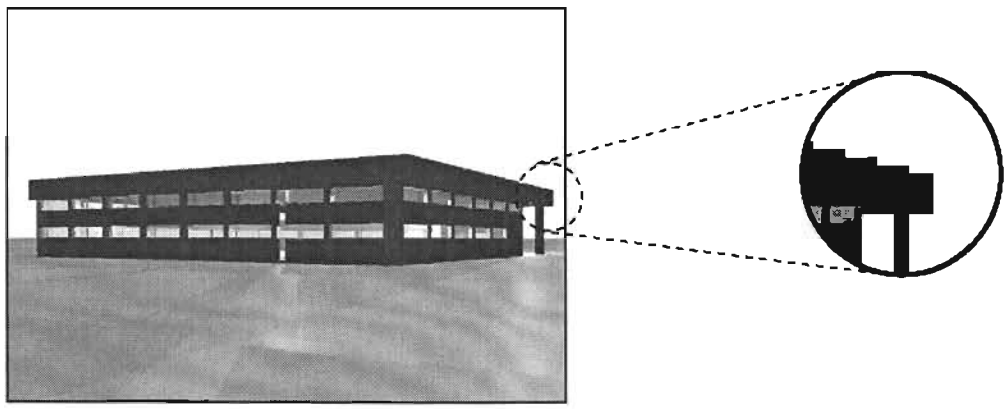


Fig. 1.2 En el círculo de aumento, se puede apreciar que la línea del techo que parecía recta, se ha convertido en una línea escalonada con pérdida de información como resultado del medio digital.

1.1.5 PIXEL

La unidad fundamental de una imagen digital recibe el nombre de: "pixel[2]", y al igual que los puntos también tienen una posición y un color asociado.

1.1.6 RESOLUCIÓN

Anteriormente se había mencionado que la nitidez de una imagen digital mantiene una estrecha relación con la cantidad de pixeles que la componen.

El concepto de resolución establece una relación entre la cantidad de pixeles que existen sobre el área de una imagen, lo que significa que a mayor resolución mejor será la calidad y viceversa.

La mayoría de los monitores que actualmente se venden en el mercado, permiten mostrar imágenes a resoluciones configurables que oscilan entre los 640x480, 800x600 y 1280x1024 pixeles, en tamaños de 14, 15 y 17" principalmente.

[2] Pixel es la contracción de "Picture-Element", que significa: Elemento de Imagen.

En las impresoras, la medida de resolución adquiere el nombre de dpi (dot per inch o puntos por pulgada), y algunas de ellas ofrecen la posibilidad de imprimir a 360x720, 720x1440, 1440x2880 dpi, etc.

Para entender mejor el concepto, generamos tres imágenes iguales del mismo tamaño pero con diferente resolución:

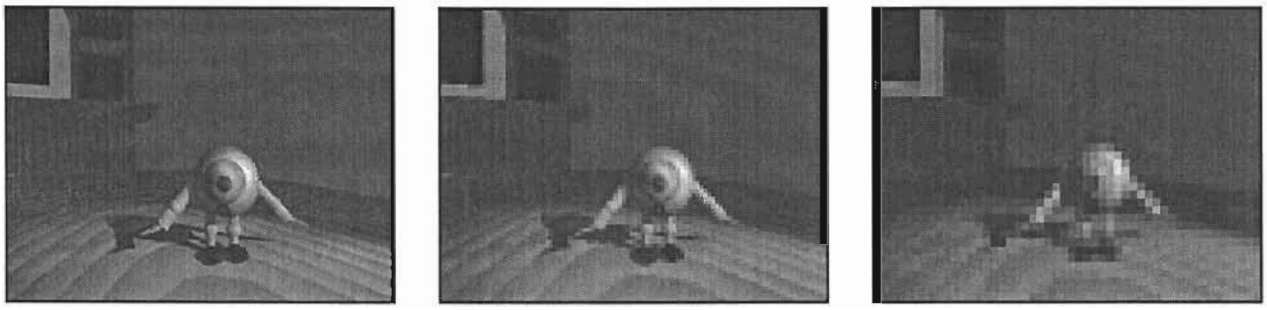


Fig. 1.3 La imagen con mejor calidad visual es la de mayor resolución (extrema izquierda), mientras que la peor es la de menor resolución (extrema derecha)

1.1.7 MODELOS DE COLOR

Los modelos de color, especifican la manera en la cual a partir de la mezcla de unos cuantos colores primarios o básicos, podemos obtener cualquier otro que se encuentre dentro de la gama de colores que muestra un arcoiris.

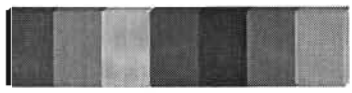


Fig. 1.4 Los colores del arcoiris son el resultado de la descomposición de la luz blanca o transparente

Existen diversos modelos de color como: RGB, CMY, HSB, entre otros. Sin embargo, sólo nos ocuparemos de los dos primeros debido a que nos servirán de base para el manejo de imágenes.

1.1.7.1 MODELO RGB

Las siglas de este modelo significan: **R**ed - **G**reen - **B**lue, lo cual indica que sus colores principales o primarios son: el rojo, verde y azul respectivamente, y que la mezcla de ellos con cierta intensidad permitirá obtener los demás.

Al modelo RGB se le conoce como de síntesis aditiva, debido a que la combinación de los tres colores a su mayor intensidad da como resultado el blanco y a la inversa el negro.

Supongamos que el valor mínimo de intensidad que pueden adquirir cada uno de los componentes del RGB es 0 y el máximo es de 255, y que deseamos obtener el color amarillo. Para lograrlo debemos de hacer las mezclas de la siguiente manera:

R:255 - *G*:255 - *B*:0

Cabe señalar que la cantidad de colores que podemos representar con intensidades de 256 valores como máximo por cada uno de los componentes, es de 16 millones y que dicha cantidad resulta suficiente para la percepción del ojo humano.

Enseguida se muestra una representación gráfica del modelo, que facilita su comprensión:

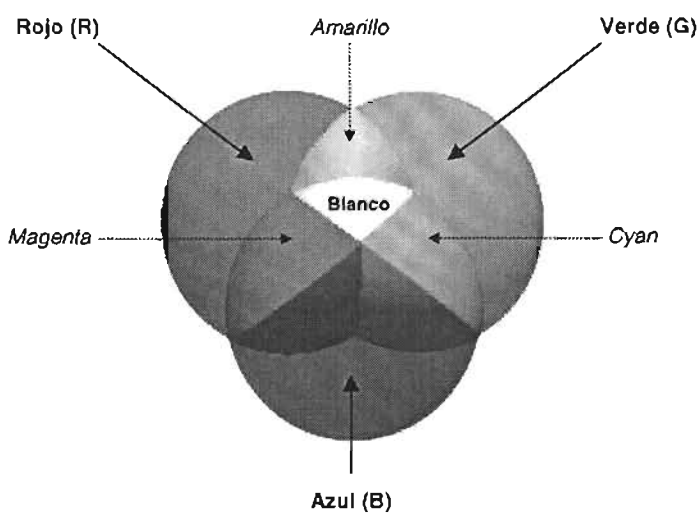


Fig. 1.5 La mezcla de los tres componentes del modelo RGB a su mayor intensidad da como resultado el color blanco

Los monitores de computadoras, televisores y proyectores, son algunos de los dispositivos que hacen uso de este modelo.

Un monitor de computadora a color se compone de una matriz de pixeles, y cada pixel a su vez se constituye de tres colores que corresponden al modelo RGB. En la figura siguiente ejemplificaremos ésto:

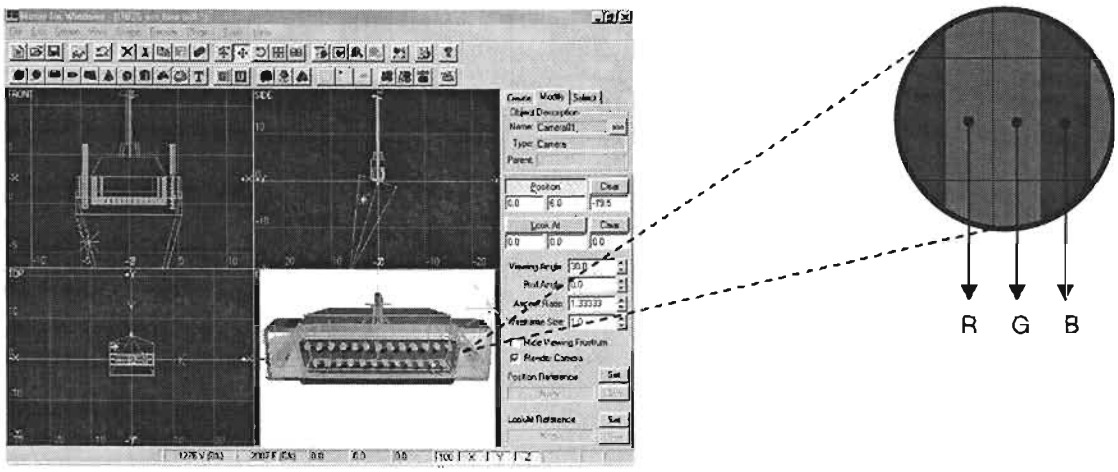


Fig. 1.6 Al aumentar cualquier sección del monitor de una computadora o televisor, se puede apreciar que cada pixel se compone de tres colores: Rojo, Verde y Azul (RGB)

1.1.7.2 MODELO CMY

Los colores principales o primarios de este modelo son: Cyan, Magenta y Amarillo, de ahí sus siglas (**C**yan, **M**agenta y **Y**ellow).

Es la versión opuesta al RGB por lo que se le llama de síntesis sustractiva, eso indica que la mezcla de sus colores a la máxima intensidad da como resultado el negro.

Al igual que con el modelo anterior, el color del pixel dependerá del valor que tengan cada uno de sus componentes. Suponiendo que el valor mínimo de intensidad para cada componente puede ser 0 y el máximo 255, y deseamos obtener el amarillo. La combinación tendrá que ser la siguiente:

C:0 - M:0 - Y:255

Si nos fijamos en la imagen del RGB de la explicación anterior, podremos darnos cuenta de que los colores que se obtienen al mezclar pares de colores son el CMY, y en la figura que mostraremos a continuación sobre este modelo veremos que sucede lo contrario:

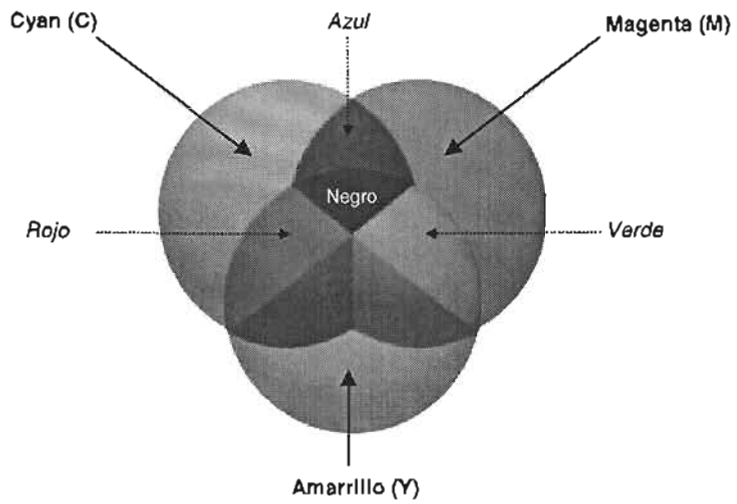


Fig. 1.7 En este modelo de síntesis sustractiva, los colores primarios son el CMY y los secundarios el RGB.

El modelo CMY se utiliza principalmente en las impresoras de inyección de tinta, sólo que para obtener mejores resultados añaden el color negro por separado, de tal manera que se maneja el término CMYK, en donde la K proviene de Black.

1.2 EFECTOS

En determinadas ocasiones existirá la necesidad de transformar nuestras imágenes para eliminarles el ruido, resaltar zonas no visibles y contornos, ocultar regiones, etc.

Existen una gran variedad de efectos que permiten manipular el espacio de pixeles para lograr la apariencia deseada, y algunos de los que describiremos en los puntos siguientes son: brillo, contraste y filtros (pasa alta, pasa baja, laplaciano, entre otros).

1.2.1 BRILLO Y CONTRASTE

Los conceptos de brillo y contraste en el procesamiento digital de imágenes son análogos a los manejados por los televisores, monitores de computadora, celulares, entre otros.

La función del primero es aumentar o disminuir el valor de intensidad de un pixel en la proporción deseada, de tal manera que es posible convertir en blanco lo que es negro (pasando por toda la gama de grises) y viceversa.

El contraste se utiliza para lograr cierta discriminación entre pixeles, esto quiere decir que los tonos claros tenderán a volverse blancos y los oscuros se moverán hacia el negro. Para aumentar el contraste, los valores de pixel que se localicen en el rango de 0 a 127 serán multiplicados por un factor debajo de 1, mientras que para los que van de 128 a 255 la transformación será con un valor superior a 1.

Con la intención de ejemplificar lo anteriormente mencionado, incrementaremos en un 50% el brillo y contraste de una imagen compuesta por 9 pixeles, cuyos valores se localizan en una escala de grises entre 0 y 255:

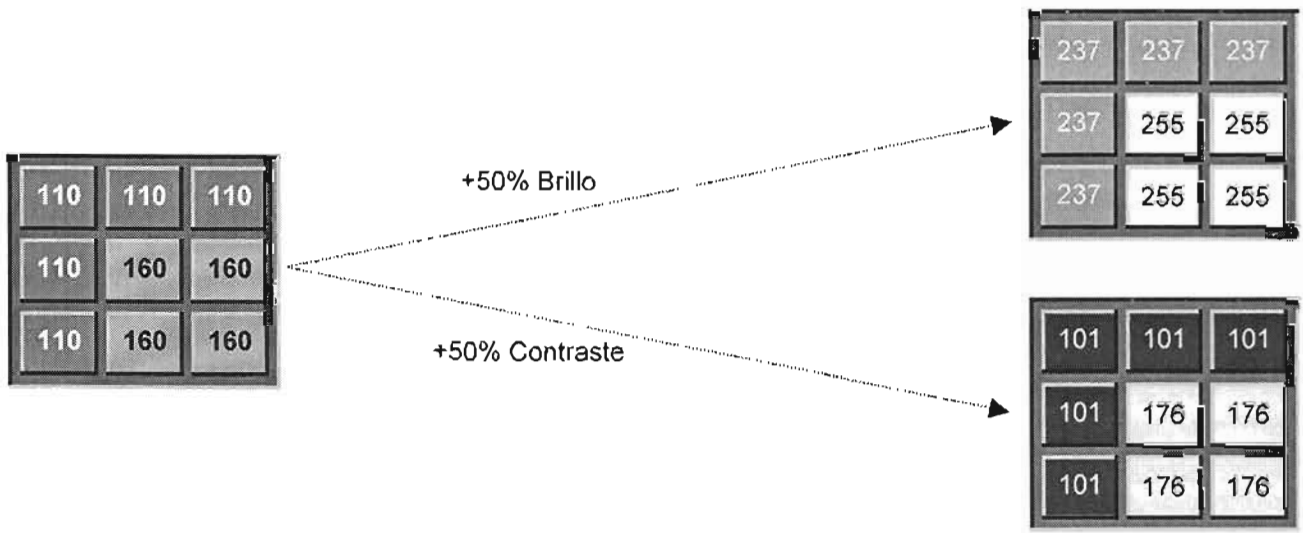


Fig. 1. 8 Al aumentar el brillo de la imagen original (izquierda) en un 50% todo se vuelve más claro pero con el contraste se origina una discriminación

Ahora ejemplificaremos el aumento de brillo y contraste al 50% pero con imágenes reales:

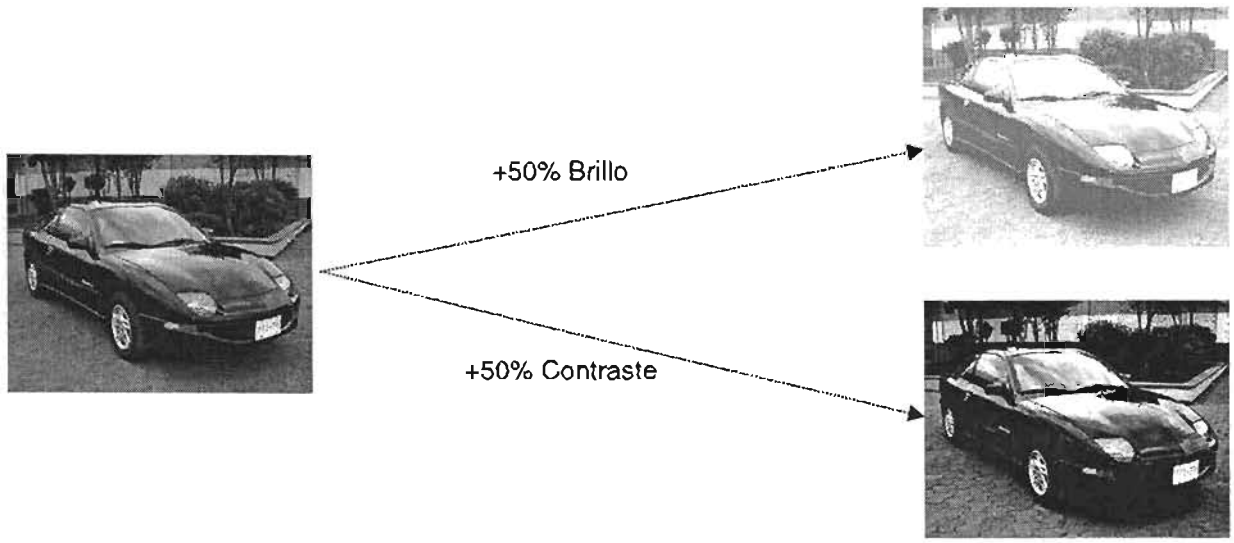


Fig. 1. 9 Incrementando el brillo de la imagen original (izquierda), todo se aclara, pero con el contraste los tonos oscuros se vuelven más oscuros y los claros más claros

1.2.2 FILTROS

La palabra filtro nos hace pensar en una especie de coladera que impide el paso a ciertos objetos a través de ella, de tal manera que sólo cruzarán los que cumplan con ciertas propiedades.

Un filtro en la computación gráfica se refiere a una matriz de valores y un factor, que permiten transformar una imagen en otra al barrer y multiplicar todos los pixeles.

De manera más formal, un filtro es una estructura numérica de transformación de imágenes que recibe el nombre de "kernel".

Al aplicar un filtro, cada pixel central es directamente afectado por los valores de sus vecinos, de tal manera entre mayor sea la ventana del kernel, mejor será el resultado que se obtenga.

En la figura 1.10 veremos un filtro de nueve pixeles distribuidos en una matriz de 3x3 junto con su factor:

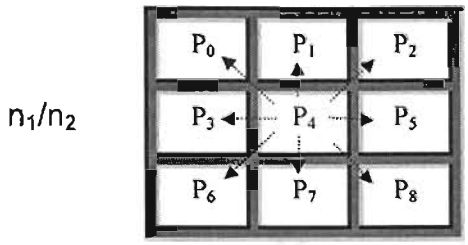


Fig. 1.10 El pixel central (P_4) será transformado por medio de los valores de sus pixeles vecinos (P_0,P_1,P_2, P_3, P_5, P_6, P_7 y P_8), y por el factor.

En los siguientes segmentos, mencionaremos algunos filtros para obtener efectos interesantes de transformación.

1.2.2.1 FILTRO PASA BAJA Y PASA ALTA

El primero de estos filtros permite que pasen las frecuencias bajas de una imagen mientras atenúa las de alta, produciendo un suavizado de la misma mediante la reducción de ruido.

El segundo filtro magnifica los cambios bruscos de intensidad de la imagen sin alterar las áreas planas.

Los valores correspondientes a cada una de las matrices de transformación de los filtros son:

$$1/9 \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad 1 \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Fig. 1.11 El kernel izquierdo corresponde al filtro pasa baja y el derecho al pasa alta

En la figura siguiente se pueden apreciar los resultados de aplicar estos filtros a una fotografía:

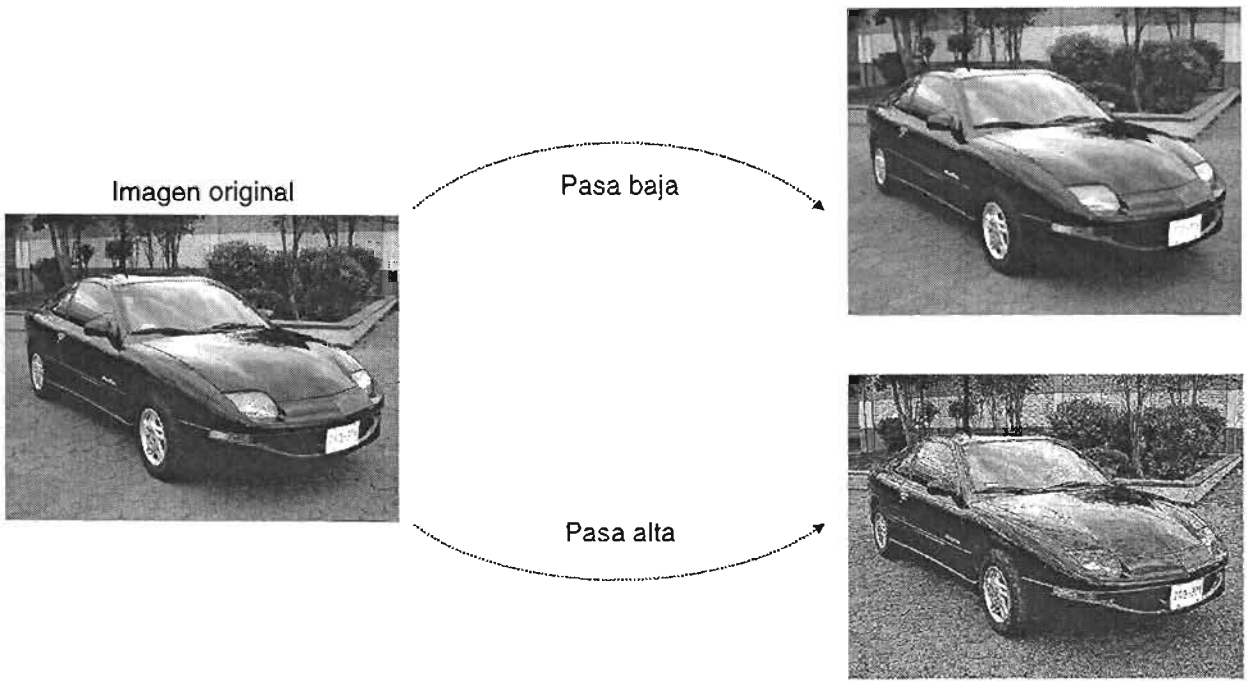


Fig. 1.12. La imagen original se suaviza con la ayuda del filtro pasa baja pero con el pasa alta las zonas con cambios de intensidad se resaltan

1.2.2.2 FILTRO LAPLACIANO Y SOBEL

Para encontrar los contornos de una imagen en cualquier dirección, se utiliza un filtro de nombre: "Laplaciano" y para resaltar los bordes en direcciones específicas existe el Sobel en sus diferentes modalidades: horizontal, vertical, diagonal y diagonal invertida.

En la figura 1.13 se incluyen los valores de los Kernels correspondientes a estos filtros:

Laplaciano

$$1 \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Sobel horizontal

$$1 \begin{bmatrix} -1 & -2 & 1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Sobel Vertical

$$1 \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Sobel Diagonal

$$1 \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix}$$

Sobel Diagonal Invertida

$$1 \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}$$

Fig. 1.13. Filtros para encontrar contornos

Los resultados de aplicar cada uno de los filtros a una imagen real se pueden apreciar en la figura siguiente:

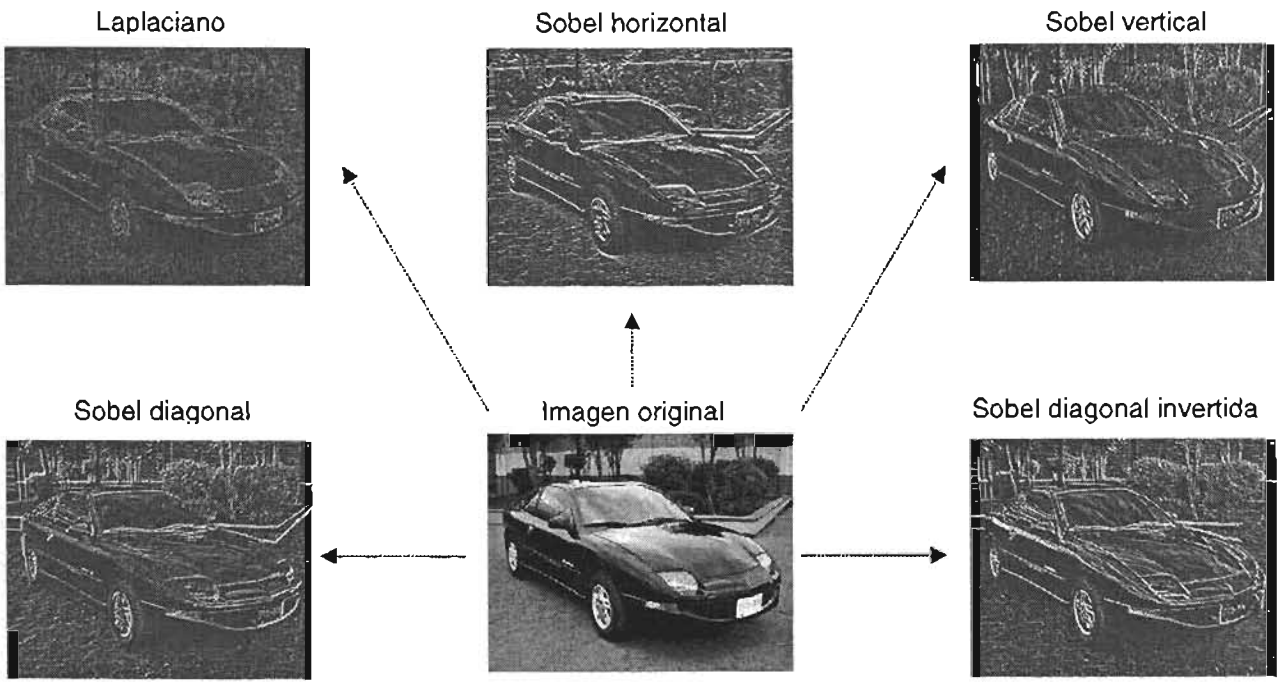


Fig. 1.14. Dependiendo del filtro que se utilice, será el contorno que se resalte

Con la ayuda de los filtros es posible obtener transformaciones muy importantes para el procesamiento digital de imágenes que a la vista parecen complejas, pero que son simples de programar.

1.3 FORMATOS GRÁFICOS

La manera en la que las imágenes digitales se almacenan dentro de los archivos, recibe el nombre de: "formato gráfico".

Un formato gráfico es una configuración para la representación de imágenes, de tal manera que dependiendo de dicha configuración será el tamaño, calidad y distribución de los pixeles dentro del archivo.

En el mercado existe una gran variedad de formatos, pero los más comunes son: BMP, GIF, JPG, PNG, PPM, entre otros.

En los apartados siguientes daremos una breve descripción de los formatos más utilizados en la actualidad, para conocer la importancia que tiene cada uno de ellos al momento de decidir en cual debemos de almacenar nuestras imágenes.

1.3.1 BMP

Este tipo de formato fue creado por Microsoft y sus siglas provienen de las palabras Bit MaP (mapa de bits).

En una imagen BMP, los valores de los pixeles son almacenados sin transformación alguna de abajo hacia arriba y de derecha a izquierda.

El BMP permite manejar imágenes monocromáticas (b/n), con escala de grises, a 16, 256 y 16,777,216 colores (24 bits)[3].

Debido a que los valores de cada pixel son almacenados tal cual en el archivo, no existe pérdida alguna en la imagen pero su mayor defecto es el gran espacio que ocupa.

[3] En una imagen de 24 bits, cada pixel se constituye de tres colores (Rojo, Verde y Azul [RGB]) y considerando que cada componente ocupa 8 bits y puede representar hasta 256 colores, al realizar la

En la siguiente imagen se puede observar una representación de la manera en la que se almacena el formato BMP:

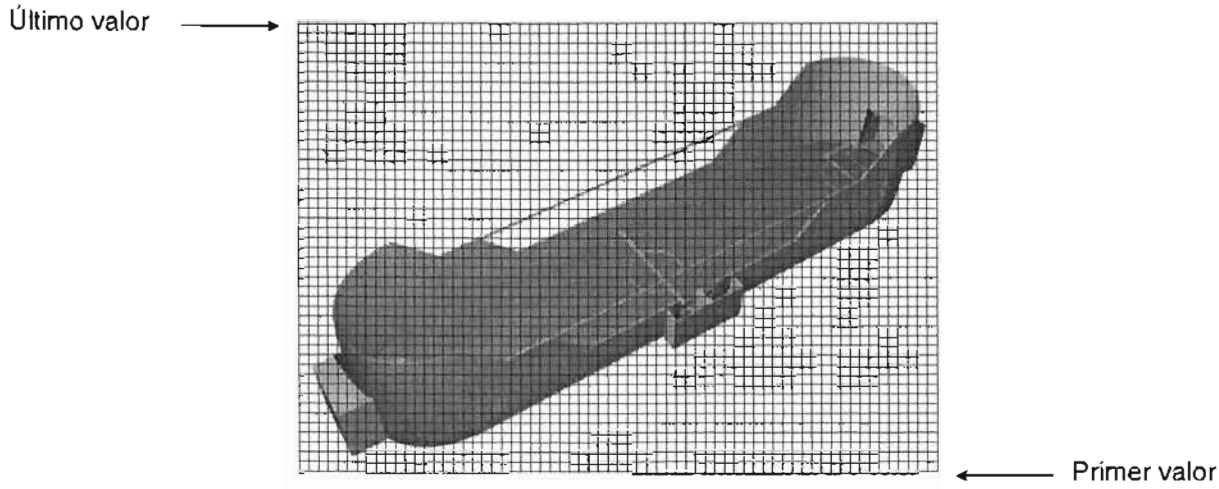


Fig. 1.15 El formato BMP almacena los valores de una imagen comenzando desde la esquina inferior derecha hacia la superior izquierda

1.3.2 GIF

Sus siglas provienen de Graphics Interchange Format (Formato de Intercambio Gráfico), es un formato gráfico comprimido[4] de 8 bits desarrollado por Compuserve en 1987. Utiliza el método de compresión LZW (Lempel, Ziv y Welch) que no produce pérdida de información, lo que da como resultado imágenes que ocupan muy poco espacio y con la nitidez original. Este método fue patentado por Unisys y por su compresión sobre cadenas de símbolos repetidos, es útil para imágenes tipo caricatura, con geometría simple y pocos colores cuya paleta no rebase los 256, pero no sirve para imágenes muy coloridas o con mucho detalle que presenten grandes cambios sobre la trama de pixeles de la imagen.

multiplicación de 256x256x256 obtenemos 16 millones de colores distintos que son suficientes para cubrir la gama de percepción visual del ojo humano.

[4] La compresión es el mecanismo utilizado para almacenar cierta información en un menor espacio. Por ejemplo, con el método compresión de pixeles adyacentes si se tienen 150 pixeles blancos en la imagen, en lugar de guardar el valor blanco 150 veces, únicamente se almacena una vez el valor del pixel y el total de veces que se repite.

Al ser un formato de 8 bits, su paleta sólo permite representar 256 colores diferentes por lo que si adaptamos imágenes de mayor cantidad de bits (16, 24, 32), la imagen resultante tendrá una considerable pérdida de nitidez que será desagradable al ojo humano.

Existe una variante al GIF 87 original, llamado "GIF89a" que permite marcar colores como transparentes.

En la siguiente figura se muestra una imagen GIF de gran tamaño visual pero que ocupa muy poco espacio en disco:

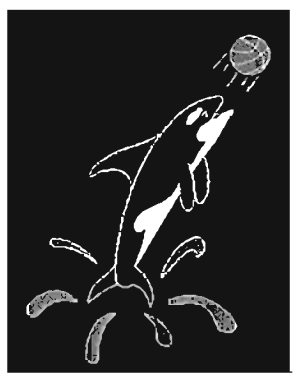


Fig. 1.16 El formato GIF es útil para imágenes tipo caricatura y con 256 colores como máximo

1.3.3 JPG

JPEG (Joint Photographic Experts Group) es un formato de 24 bits (RGB o CYMK) y escala de grises, con un algoritmo de alta compresión y con pérdida, especificado por la norma ISO 10918-1.

Para minimizar el tamaño de un archivo de imagen, este formato ocupa mecanismos matemáticos complejos y una reducción de los niveles de azul altos hacia valores menores, ya que el ojo humano tiene un menor grado de percepción de este tipo de tonos.

El método utilizado por el JPG permite comprimir fotografías o imágenes fotorrealistas con tonos continuos entre pixeles, además de que ofrece la capacidad de especificar cuanta pérdida de calidad de imagen está dispuesto a tolerar. Es malo

para imágenes con altos contrastes y formas simples, ya que suaviza los cambios de color, es decir, en donde exista una variación brusca de color como por ejemplo un borde, el algoritmo lo transformará en una línea, esto quiere decir que el deterioro de nitidez se hace evidente donde hay bordes o líneas.

A diferencia del GIF, el JPG si tiene pérdida de información a costa de su altísima compresión.

Si utilizamos este formato para modelar una imagen físicamente, es posible que los resultados no sean los adecuados debido a la reducción de fidelidad.

La figura 1.17 muestra una imagen multicolor almacenada en JPG:

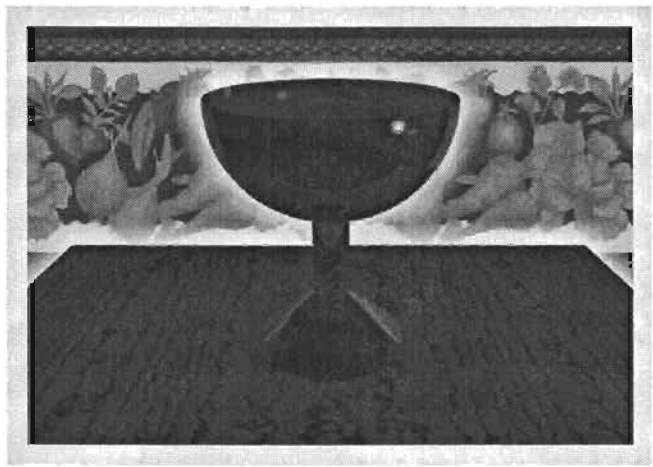


Fig. 1.17 El JPG ofrece una alta compresión pero con pérdida de información

1.4 MODELOS DIGITALES

A partir de la década de los 90´s, empezaron a proliferar las filmaciones en tercera dimensión como: Jurasic Park, Toy Story, Shrek, etc. Gracias al avance de la ciencia y la tecnología computacional, los objetos que antes se dibujaban de manera plana, ahora parecen tener un volumen determinado y por ende una mayor semejanza con la realidad.

Para representar nuestro entorno en tres dimensiones, se desarrollaron los modelos digitales, los cuales a partir de ciertos cálculos matemáticos permiten conocer

los valores de posición en el espacio y color de cada uno de los pixeles que forman a cada objeto.

Los modelos digitales más utilizados en la actualidad son: los que tratan superficies por medio de ecuaciones, mallas, campos de altura y de elevación. Veamos con un poco más de detalle a cada uno.

1.4.1 REPRESENTACIÓN DE SUPERFICIES POR MEDIO DE ECUACIONES

Este tipo de modelo como su nombre lo indica, utiliza ecuaciones matemáticas para simular la realidad, por ejemplo: esferas, cubos, pirámides, etc, lo cual da como resultado una muy buena representación de los objetos.

Los recursos de procesamiento para los cálculos son muy altos, además dificulta la creación de superficies irregulares tales como: personas, animales, plantas, autos, entre otros.

Uno de los programas más famoso para el modelado mediante ecuaciones es el POV-RAY (Persistence Of Vision Ray Tracing o Trazado de Rayos para la Persistencia de la Visión) cuya peculiaridad es su distribución libre, además de incluirse dentro de los paquetes del Sistema Operativo Linux.

La figura 1.18 fue creada bajo el entomo programable de POV-RAY:

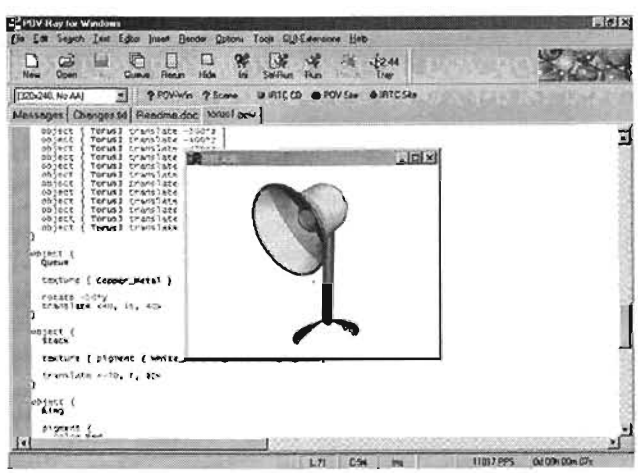


Fig 1.18 Lámpara generada con la ayuda del modelador en 3D "Pov-Ray"

1.4.2 REPRESENTACIÓN DE SUPERFICIES MEDIANTE MALLAS

Los polígonos son la materia prima de este modelo para representar cualquier tipo de superficies, por ejemplo, una esfera se compone de la unión de varios triángulos, de tal manera que la calidad de visualización será directamente proporcional a la cantidad de triángulos que constituyan a dicho objeto.

Es importante considerar que el consumo de recursos de procesamiento depende de la calidad que elijamos para los modelos.

Algunos programas que ofrecen modelado por mallas son: 3D Studio, Maya, Rinoceros, entre otros.

Fig. 1.19 Imagen creada con el software 3D-Studio.

1.4.3 CAMPO DE ALTURAS (HEIGHTFIELD)

Las imágenes digitales se caracterizan por representar superficies planas, es decir que están en dos dimensiones y que por lo tanto podemos visualizarlas en un eje de coordenadas virtual como el que posee un monitor.

Como se había mencionado anteriormente, las imágenes digitales están compuestas de pixeles, y cada pixel tiene asociada una posición y un color. Si consideramos que dicha posición se constituye a su vez dos valores; uno para el eje de

coordenadas X y otro para el Y, entonces cada pixel se define por tres datos, lo que nos permite construir imágenes tridimensionales a partir del mapeo de imágenes bidimensionales.

Un campo de altura o heightfield, es la visualización en relieve o en tres dimensiones de una imagen plana (2D). En la tabla 1.1 se ilustra el mapeo de transformación correspondiente:

Pixel 2D		Pixel 3D
X	⟶	X
Y	⟶	Y
Color	⟶	Z

Tabla 1.1 Al generar un campo de alturas o heightfield a partir de una imagen plana, las coordenadas X y Y conservan su significado mientras que el valor del color se transforma en profundidad

La conversión a heightfield de una imagen gris o monocromática es directa, pero si es a color, primero es necesario obtener un solo valor para cada pixel promediando sus componentes RGB.

Para crear modelos de altura, se debe de considerar que los valores que puede tomar un pixel van de 0 (negro) hasta 255 (blanco) y que por lo tanto dentro de ese rango debemos de especificar la profundidad del heightfield, de tal manera que con el negro nuestra altura sea mínima y con el blanco máxima. Revisemos gráficamente una barra de interpolación y su mapeo de altura para comprender mejor el concepto:

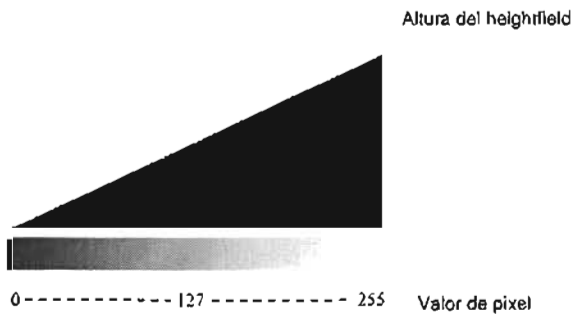


Fig. 1.20 Barra de interpolación en donde se puede apreciar que el negro no tiene altura y que conforme tiende al blanco crece hasta llegar al máximo

Ejemplifiquemos la creación de un campo de altura utilizando un logo plano de la UNAM:

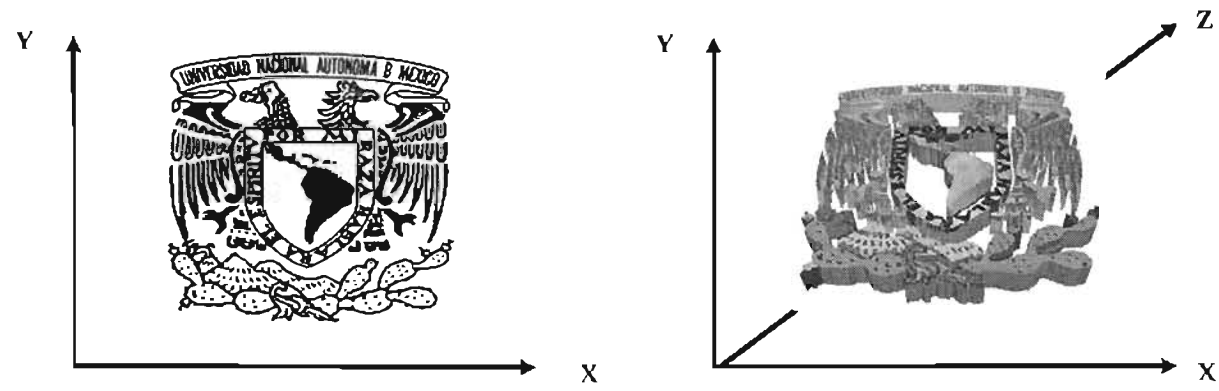


Fig. 1.21 La imagen plana o bidimensional (izquierda) se ha convertido en un campo de alturas (derecha) al transfomar cada pixel en líneas tridimensionales

1.4.4 MODELO DE ELEVACIÓN DIGITAL (DIGITAL ELEVATION MODEL O DEM)

El Modelo de Elevación Digital es muy similar a un campo de alturas sólo que esta estructura numérica de datos representa por sí sola la distribución espacial de la altitud de la superficie de un terreno.

La unidad principal de un DEM es un punto definido por una terna de valores, que son x y y para la localización cartesiana y la z para la altitud.

Existen diferentes tipos de DEM, pero el más utilizado es el de tipo raster o de matrices regulares, el cual es el resultado de superponer una retícula sobre el terreno y extraer la altitud media de cada celda. La retícula adopta normalmente la forma de una red regular de malla cuadrada. En esta estructura, la localización espacial de cada dato está determinado de forma implícita por su situación en la matriz, una vez definidos el origen y el valor del intervalo entre filas y columnas.

La matriz regular es la estructura más utilizada para la construcción de modelos DEM, ya que es una estructura de fácil manejo informático, además de ser simple de representar mediante estructuras lógicas como matrices de dos dimensiones.

La captura de información de un DEM se puede hacer principalmente mediante dos métodos que se describen enseguida:

1. **Directo**: la medición se hace de manera directa sobre el terreno, y se divide en:
    a. **Altimetría**: se ocupan altímetros radar o láser transportados por plataformas aéreas o satélites.
    b. **GPS**: el elemento principal para capturar los datos de la superficie es el GPS (global position system, o sistemas de localización global) y se basan en la triangulación que se puede lograr con las señales de satélites en el espacio.
    c. **Levantamiento topográfico**: se utilizan estaciones topográficas con salida digital.

2. **Indirecto**: las mediciones se logran a partir de documentos previos y se subdivide en:
    a. **Restitución**: los datos se obtienen con pares de imágenes que pueden ser:
        i. **Estéreo-imágenes digitales**: imágenes tomadas por satélites
        ii. **Estéreo-imágenes analógicas**: imágenes fotográficas convencionales
        iii. **Interferometría radar**: imágenes de interferencia de sensores radar.
    b. **Digitalización**: los mapas topográficos son la base de captura, y puede ser
        i. **Automática**: mediante escáner y vectorización
        ii. **Manual**: mediante tablero digitalizador.

A diferencia de los formatos anteriores, el DEM no contiene colores para cada

punto, pero es muy poderoso porque incluye valores de altura para cada posición que pueden representar superficies con valores fieles para los tres ejes (X, Y y Z).

Con los modelos de elevación digital, se puede visualizar cualquier región en el espacio con utilidades de todo tipo. En la figura 1.22 se incluye una sección de nuestro país construida a partir de los datos procesados de un DEM:

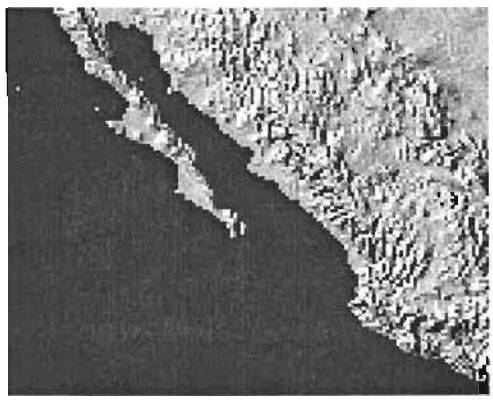


Fig. 1.22 La distribución de puntos en el espacio de un DEM, se puede modelar en tercera dimensión con la ayuda de una computadora y se le puede agregar textura para obtener cierto realismo

CAPÍTULO II

DISEÑO Y CONSTRUCCIÓN DEL HARDWARE

Las partes físicas de nuestro modelador como: cables, motores, circuitos integrados, estructuras metálicas y en general todo lo que podamos tocar, es a lo que llamaremos: "hardware".

En el presente capítulo se hará la descripción completa de cada uno de los componentes del hardware, para conocer la manera en que se llevarán a cabo las funciones de modelado mediante la interacción con la computadora.

2.1 TEORÍA DE OPERACIÓN DEL MODELADOR

El objetivo principal de esta tesis es diseñar y construir una máquina que sea capaz de crear modelos físicos tridimensionales (principalmente en madera) a partir de imágenes planas (aunque para futuras aplicaciones, sólo bastará con modificar el software[5] para que también reproduzca modelos de archivos tipo: DEM, 3D Studio, Autocad, entre otros), en otras palabras, el dispositivo será capaz de esculpir un trozo de madera con el campo de altura correspondiente a una imagen de dos dimensiones.

Para esculpir un campo de altura sobre un bloque de madera, la computadora leerá una imagen plana desde un archivo para extraer de cada pixel sus coordenadas en X y Y, junto con el valor del color. Los tres valores leídos serán enviados al hardware que estará conectado por el puerto paralelo, para perforar el bloque en la posición y profundidad exacta. Los valores de los pixeles de la imagen serán tratados uno a uno hasta terminar toda la imagen y el modelo.

2.2 DISEÑO DEL MODELADOR

El hardware del modelador estará compuesto de partes mecánicas, eléctricas, electrónicas y electromecánicas.

En los elementos mecánicos se incluyen: estructuras metálicas y plásticas, tornillos sin fin, tuercas, ejes guías, cajas de sujeción y bujes.

La composición eléctrica-electrónica del aparato será una mezcla de dispositivos analógicos y digitales, en donde encontraremos: resistencias, controladores de motores, fuente de poder, sensores, optoacopladores, centronics, y circuitos programables (GAL).

La parte neurálgica del modelador estará dada por los maestros electromecánicos del movimiento: los motores corriente continua y de paso.

Para describir el hardware partiremos del diseño visual tridimensional (ver fig. 2.1, 2.2, 2.3 y 2.4) y después explicaremos por separado cada uno de los componentes.

[5] En el capítulo III se describe todo lo relacionado con el software

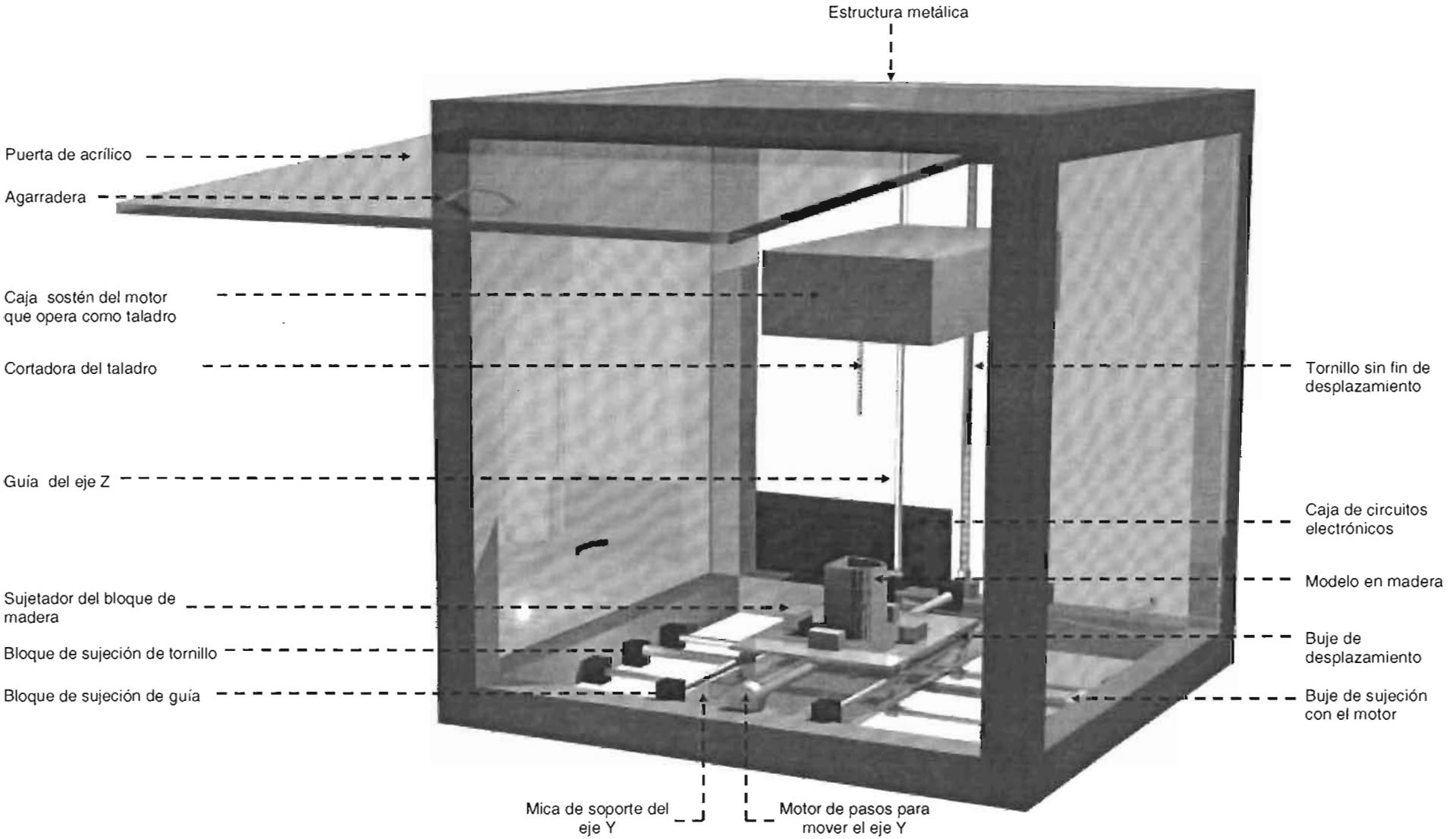


Fig. 2.1 Vista tridimensional frontal del hardware en la que se pueden observar tres ejes coordenados (X, Y y Z)

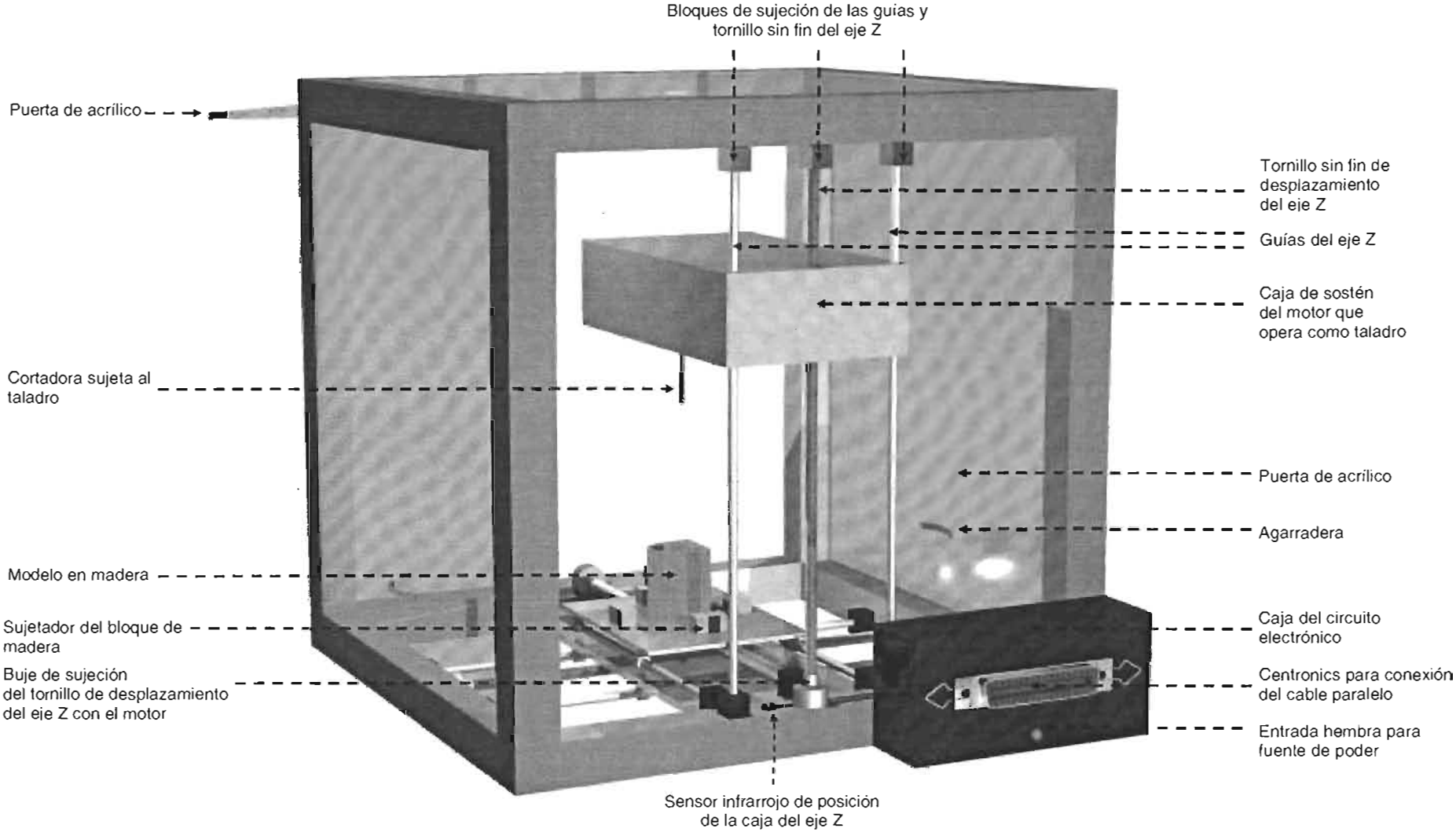


Fig. 2.2 Vista tridimensional de la parte posterior del hardware

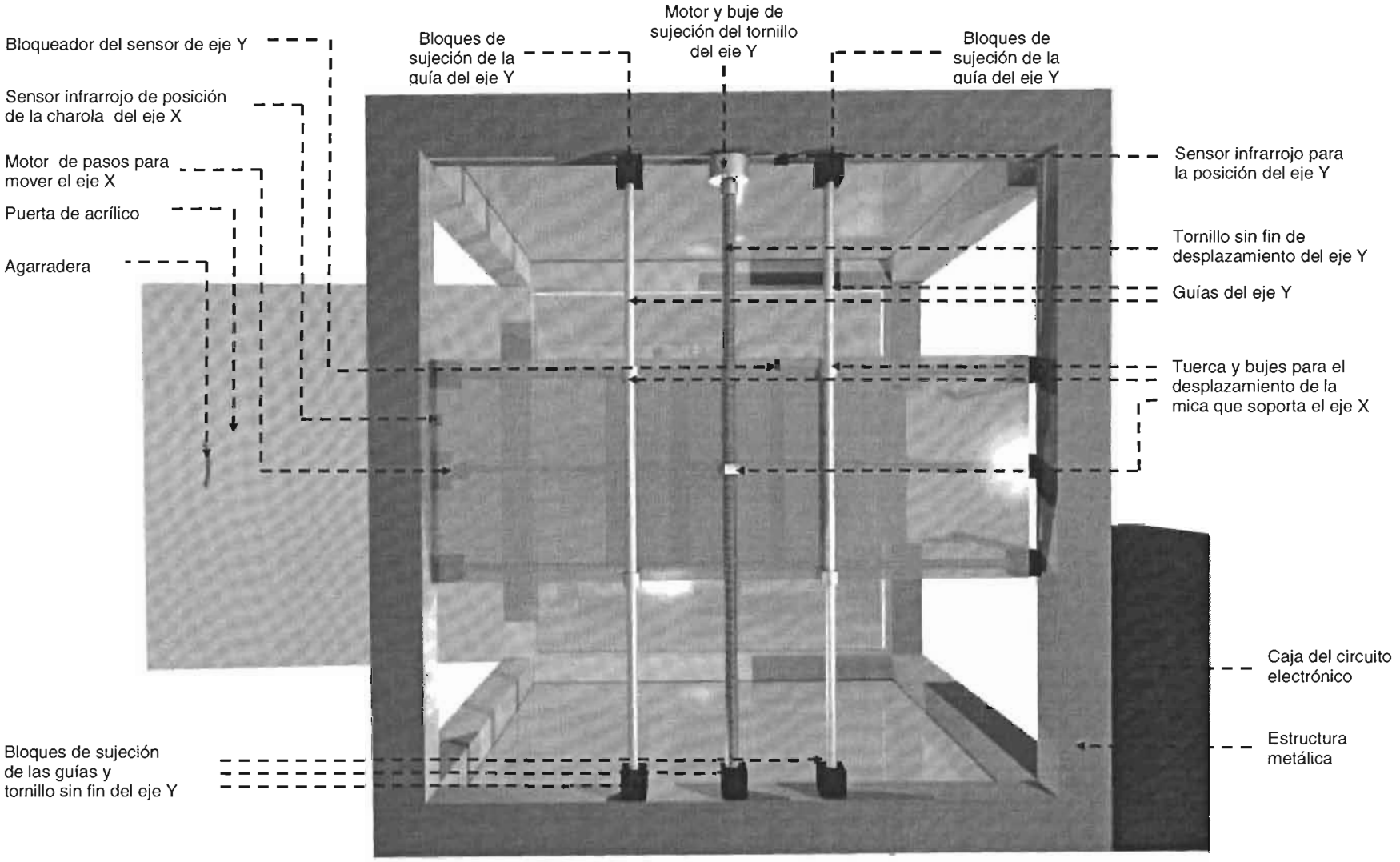


Fig. 2.3 Vista del hardware por debajo

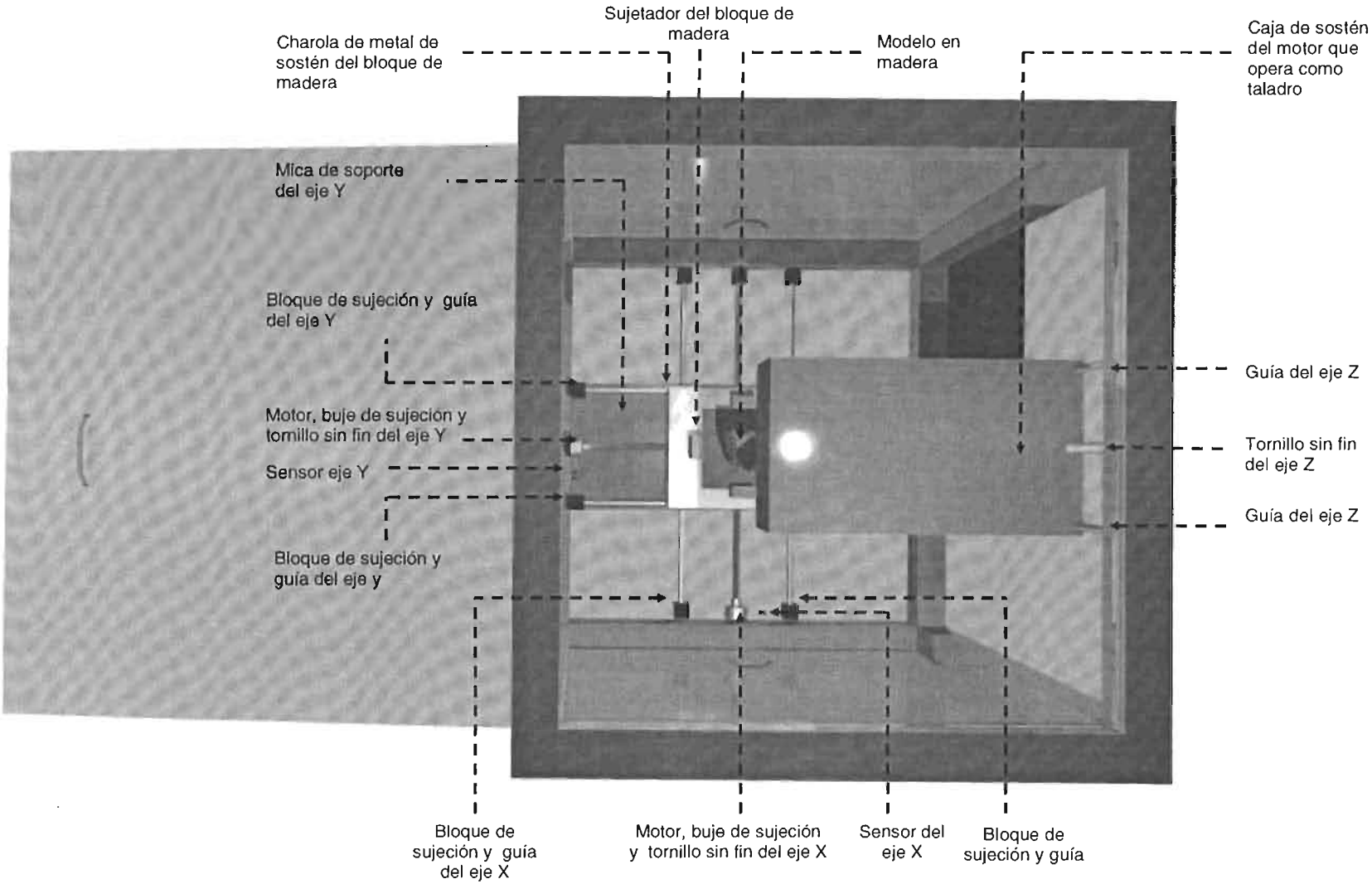


Fig. 2.4 Vista superior del hardware

El hardware estará compuesto de tres ejes de coordenadas que nos darán la oportunidad de lograr la tridimensionalidad que buscamos. El eje X y Y estarán dispuestos a manera de mesa de coordenadas y se moverán análogamente a la posición del pixel de la imagen que lee la computadora desde un archivo. Sobre esta mesa de coordenadas se moverá el bloque de madera, y el eje Z se encargará de la profundidad de la perforación (relacionada con el color del pixel) ya que sobre él estará montado una cortadora.

Los ejes del modelador tendrán la siguiente configuración:

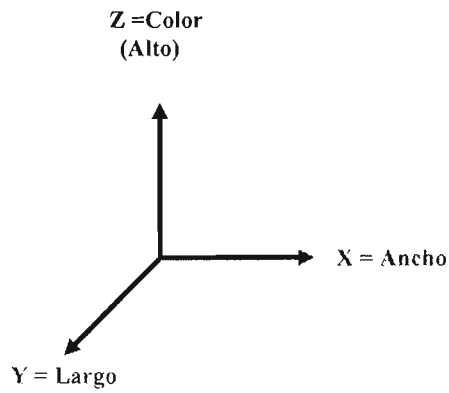


Fig. 2.5 Los ejes X y Y ofrecen la posibilidad de mover el bloque de madera a lo largo y ancho, mientras que el Z permite lograr la profundidad de perforación

Los tres ejes tendrán en sus extremos un juego de sensores que nos permitirán regresar la mesa de coordenadas y el taladro a su posición inicial o de arranque.

Por su parte, la estructura del modelador tendrá ángulos de soporte de aluminio para asegurar poco peso y gran dureza.

Veamos más de cerca cada uno de los componentes del hardware comenzando por la parte electromecánica.

2.3 COMPONENTES ELECTROMECÁNICOS (MOTORES)

Los motores son dispositivos electromecánicos con los que se pueden mover: automóviles, aviones, barcos, motocicletas, submarinos, entre otros. Son indispensables para la vida de los humanos y serán el corazón de nuestro hardware.

De manera más técnica, podemos decir que los motores son dispositivos transductores[6] de algún tipo de energía como la eléctrica, mecánica, etc., en energía cinética o de movimiento.

Para desarrollar este proyecto ocuparemos cuatro motores:

- *1 motor de corriente continua*: servirá como taladro y sobre él se montará la cortadora que perforará la madera.

- *3 motores de pasos*: su función será posicionar el taladro y el bloque de madera a transformar. Cada motor ocupará un lugar en el espacio y moverá un eje que corresponderá al eje cartesiano tridimensional (X, Y y Z).

Enseguida describiremos ambos tipos de motores para conocer su funcionalidad específica.

2.3.1 MOTORES DE CORRIENTE CONTINUA (TALADRO)

Algo que caracteriza a este tipo de motores es que su movimiento depende de la cantidad de energía eléctrica que reciban, es decir, mientras que sea alimentado girará en un determinado sentido y de manera constante (siempre y cuando la carga sea invariable y la puede vencer), lo cual hace difícil controlarlo en posiciones exactas. Estos motores los encontramos en: videocaseteras, estéreos, carritos a control remoto, etc.

[6] Transductor: dispositivo que permite convertir un tipo de energía en otra, por ejemplo, un foco se alimenta de energía eléctrica y la transforma en energía lumínica y calorifica.

El motor que fungirá como taladro será de este tipo, ya que no necesitamos controlar su posición exacta, sino que sólo nos bastará con echarlo a andar al momento de empezar la perforación del bloque de madera y detenerlo al terminar.

Nuestro taladro se ubicará y sujetará dentro de la caja negra del eje Z, pero en los diseños sólo se puede observar la cortadora. Veamos un gráfico completo:

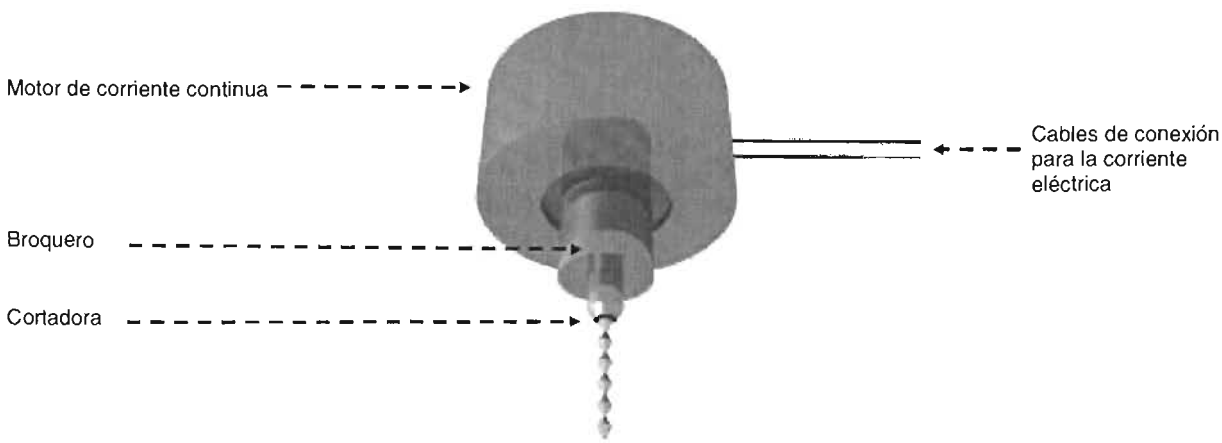


Fig. 2.6 El motor de corriente continua se moverá junto con el eje Z y tendrá sujeta una broca que perforará la madera

2.3.2 MOTORES DE PASOS (PARA EL DESPLAZAMIENTO DE LOS EJES X, Y Y Z)

Los motores de paso realizan un movimiento de ciertos grados (llamado "grado de paso") por cada pulso eléctrico que reciben.

Este tipo de motores a diferencia de los de corriente continua se les puede controlar su posición de manera exacta, por lo que serán ideales para manipular el movimiento del bloque de madera y la caja que sostendrá al taladro.

El bloque de madera debido a que estará montado sobre una mesa de coordenadas, se moverá con la ayuda de dos motores cuyo desplazamiento corresponderá a los ejes X y Y, mientras que el taladro se sujetará al correspondiente eje Z.

Los motores de paso pueden variar según los grados de giro que sean capaces de permitir en cada pulso, de esta manera, podemos tener de 1.8º, 3.6º hasta 90º, por

lo que si deseamos que uno de 1.8º complete un giro, se deberán de aplicar 200 pulsos[7].

Otra característica de estos motores, es que pueden quedarse completamente fijos si una o más de sus bobinas están energizadas, o totalmente libres en el caso contrario, es decir, cuando no existe corriente en ninguna bobina.

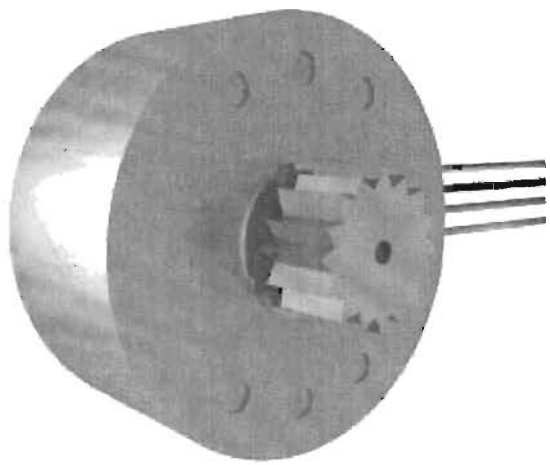


Fig 2.7 Motor de pasos para controlar la posición exacta de un eje de coordenadas

Estos motores se componen de dos partes principales que son: el rotor o parte giratoria, y el estator o parte fija. El rotor generalmente tiene imanes permanentes y se mueve gracias al campo magnético variable producido por la alimentación de corriente de las bobinas del estator.

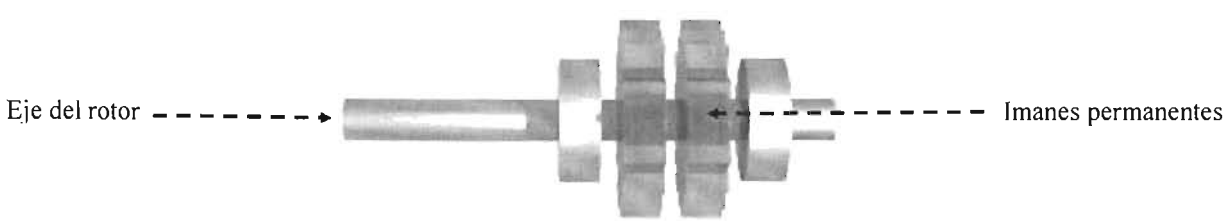


Fig. 2.8 El rotor gira sobre el campo magnético variable que generan las bobinas de un estator

[7] 1.8º x 200 pulsos=360º

Los motores de paso se pueden clasificar dependiendo de la forma en que se controlan, por lo que existen: bipolares y unipolares. En este proyecto utilizaremos del primer tipo, pero daremos la explicación de arnbos.

2.3.2.1 MOTORES DE PASO BIPOLARES

Se componen de cuatro cables y para lograr el movimiento es necesario que se hagan cambios de una determinada manera en los flujos de corriente que atraviesan las bobinas.

Veamos un esquema de estos motores, los cuales podremos extraer de irnpresoras y escáners principalmente:

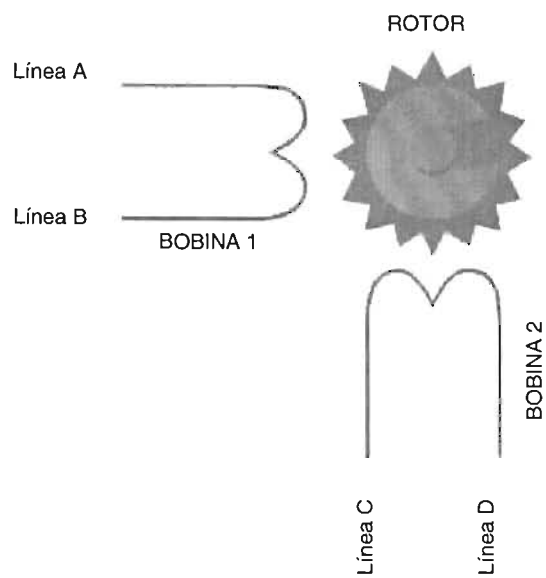


Fig. 2.9 Los motores bipolares generalmente constan de 4 líneas para controlar su giro

Los bipolares requieren que se invierta la polaridad de las bobinas para que se mueva el eje en un paso, y es necesario aplicar una secuencia para asegurar el giro en un determinado sentido y otra para el sentido inverso.

La secuencia de control de estos motores se muestra en la tabla 2.1:

	Línea				
Paso	A	B	C	D	Valor decimal
1	1	0	1	0	10
2	1	0	0	1	9
3	0	1	0	1	5
4	0	1	1	0	6

Tabla 2.1. Secuencia de giro del motor bipolar

Si deseamos que el motor gire en sentido contrario a lo producido por esta configuración, sólo bastará con invertir la secuencia, es decir en lugar de realizar los pasos en el orden 1,2,3 y 4, los haremos al revés (4,3,2 y 1).

2.3.2.2 MOTORES DE PASO UNIPOLARES

Se componen de 5 ó 6 cables y son más difíciles de conseguir que los anteriores. En la figura 2.10 se incluye el esquema correspondiente:

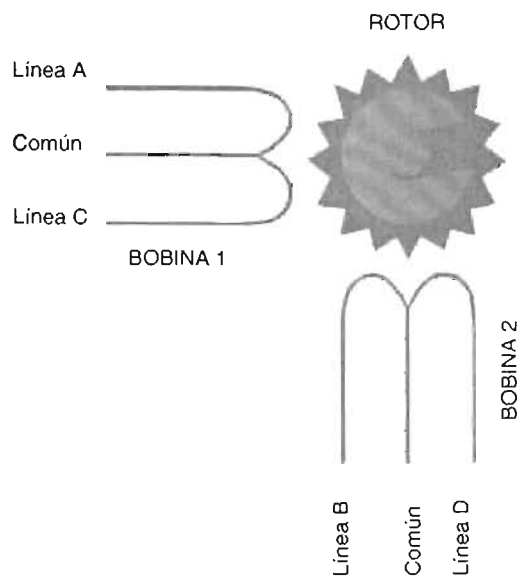


Fig. 2.10. El motor de pasos unipolar se compone de 5 ó 6 líneas para energizar las bobinas y crear los campos magnéticos que hacen girar al rotor.

Los motores unipolares se pueden manejar bajo tres tipos de configuraciones de secuenciamiento diferentes llamadas: One Step, Two Step y Half Step. Y enseguida describiremos cada una de ellas.

Configuración One Step o Wave Drive (Un Paso o de Onda)

Las bobinas son energizadas una a la vez, lo que permite consumir poca corriente pero tiene la desventaja de brindar el menor torque[8] y control. En la tabla 2.2 se incluye la secuencia de manejo correspondiente y en la figura 2.11 se puede incluye el esquema de operación de un motor:

	Línea				
Paso	A	B	C	D	Valor decimal
1	1	0	0	0	8
2	0	1	0	0	4
3	0	0	1	0	2
4	0	0	0	1	1

Tabla 2.2. Una a la vez son energizadas las bobinas

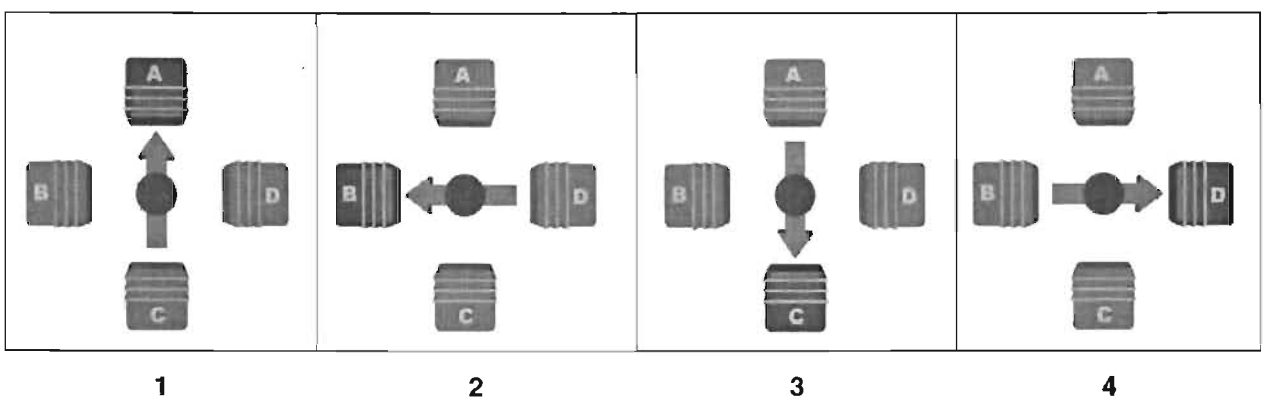


Fig. 2.11. La configuración de una onda es la que consume la menor cantidad de energía, pero ofrece un bajo torque.

Configuración Two Step (Dos Pasos)

Esta secuencia también se llama "normal" por ser la que recomiendan los fabricantes, y en ella siempre existen dos bobinas energizadas, lo cual permite tener un gran torque y control, pero consume mucha energía. En la tabla 2.3 se muestra la configuración respectiva y en la figura 2.12 se muestra el movimiento del rotor.

	Línea				
Paso	A	B	C	D	Valor decimal
1	1	1	0	0	12
2	0	1	1	0	6
3	0	0	1	1	3
4	1	0	0	1	9

Tabla 2.3. La mayor fuerza y control se logran con la configuración Two step

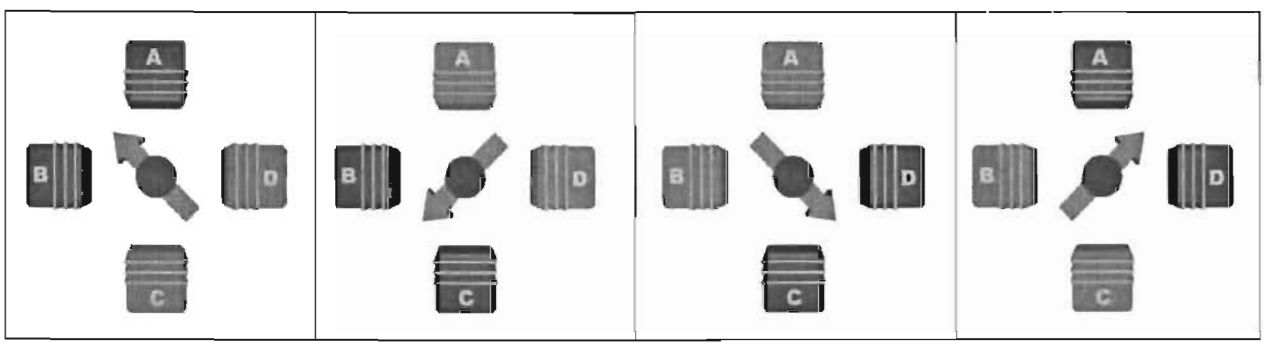


Fig. 2.12. Las bobinas se energizan de dos en dos

Configuración Half Step (Medio Paso)

Es una mezcla de las dos anteriores, es decir alterna la activación de dos bobinas con la de una. Su tabla de configuración es la 2.4 y la representación del movimiento se encuentran en la figura 2.13:

[8] El torque es la capacidad de un motor para vencer una fuerza opuesta a su movimiento.

	Línea				
Paso	A	B	C	D	Valor decimal
1	1	0	0	0	8
2	1	1	0	0	12
3	0	1	0	0	4
4	0	1	1	0	6
5	0	0	1	0	2
6	0	0	1	1	3
7	0	0	0	1	1
8	1	0	0	1	9

Tabla 2.4. Configuración para el funcionamiento de media onda

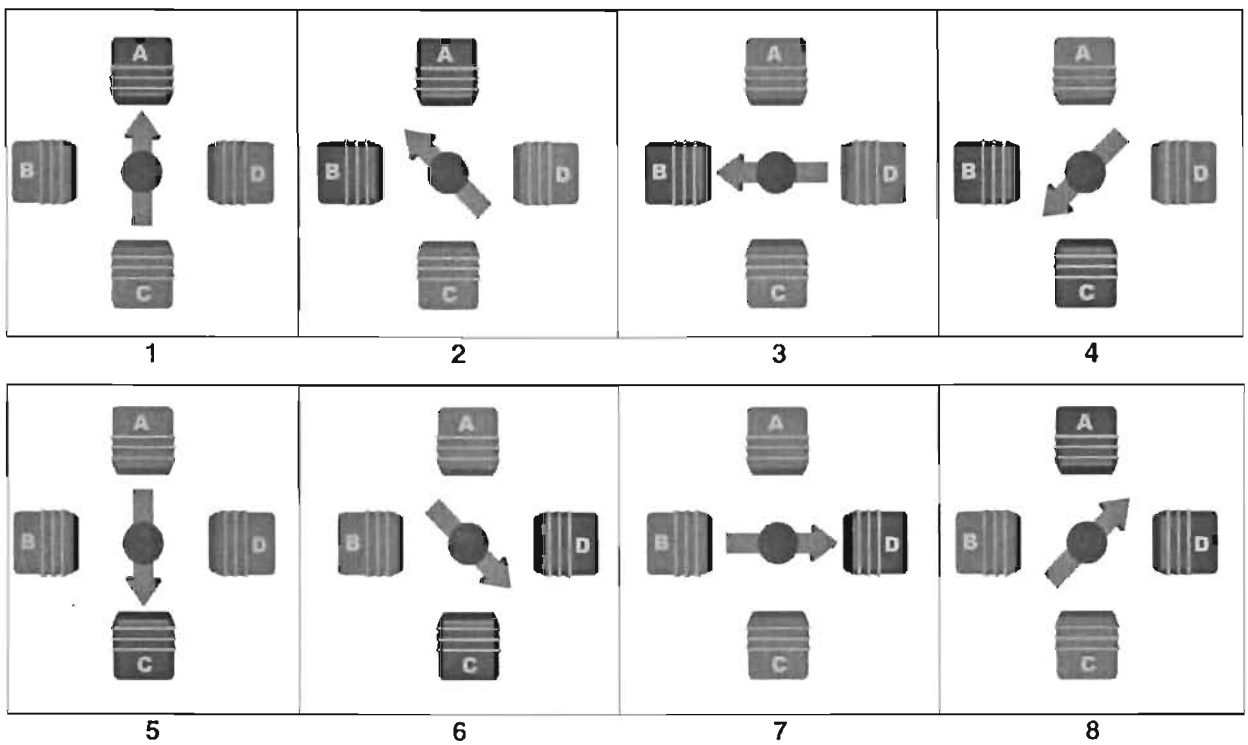


Fig. 2.13. La configuración de media onda mezcla la de una y dos fases, por lo que se deben aplicar ocho pulsos para acompletar un ciclo.

2.3.3 VELOCIDAD DE RESPUESTA DE LOS MOTORES DE PASOS

Cada motor de pasos tiene una frecuencia de respuesta ante el suministro de corriente, que depende de su marca y modelo. Esto quiere decir que poseen un límite de velocidad de acción.

Cuando la velocidad de aplicación de pulsos eléctricos al motor se encuentra dentro del rango soportado, dicho motor se moverá de manera normal, pero si el cambio de alimentación es más rápido que el soportado, es posible que no exista movimiento alguno, vibre o gire mal.

Es importante que el motor alcance la siguiente posición de giro antes de que se le envíe otra orden. Si no conocemos la velocidad de respuesta de un motor, es posible verificarla aplicándole pulsos en lapsos de tiempo que comiencen en 1 segundo y que gradualmente se disminuyan hasta encontrar un valor de respuesta adecuado.

2.3.4 PROCEDIMIENTO PARA CONOCER LAS LÍNEAS DEL MOTOR DE PASOS

Si no tenemos la hoja de especificación del motor de pasos, es posible identificar el significado de cada línea mediante el siguiente procedimiento:

Para los bipolares que generalmente son de 4 líneas, usaremos un multímetro a modo de óhmetro para verificar la continuidad entre pares de cables. Al primer par lo denominaremos como A y B, y al segundo como C y D. Después los conectaremos en el orden A,B,C y D, y aplicamos los pulsos de corriente de acuerdo a la tabla de configuración mostrada. Si el motor funciona mal tendremos que invertir A con B y C con D para lograr la buena operación.

Para los unipolares de 5 ó 6 líneas, primero aislamos el cable o los cables comunes que suelen ser del mismo color, pero sino, usando un multímetro verificamos la resistencia entre pares de cables y el común o los comunes serán los que tengan la mitad de valor de resistencia en relación a los demás, ya que dicho cable se encuentra a la mitad de dos bobinas tal y como se mostró en la figura 2.8.

Una vez encontrada la terminal común, es necesario descubrir las líneas A,B,C y D. Y para ello aplicaremos un voltaje de aproximadamente 12v al cable o cables

comunes mientras ponemos una de las otra líneas a tierra a la que llamaremos línea "A", después conectaremos a tierra uno a uno los cables restantes y observamos cual de ellos genera un movimiento hacia la izquierda y a ese lo denominamos "B", después lo safamos (sólo dejamos al A) y verificamos los dos restantes conectándolos uno a uno a tierra y observando cual de ellos genera un giro a la derecha y a ese lo nombramos "D", y por lógica el único cable restante será el "C".

2.4 COMPONENTES MECÁNICOS

Denominaremos componentes mecánicos a todos aquellos elementos estructurales o que no requieren de alimentación eléctrica para la consecución de su objetivo.

2.4.1 ESTRUCTURA

La caja mayor del modelador estará constituida de aluminio para lograr rigidez y poco peso estructural. Y las paredes serán de acrílico o mica transparente para permitir al espectador observar todo el proceso.

De acuerdo con la figura general, nuestra estructura tendrá la forma siguiente:

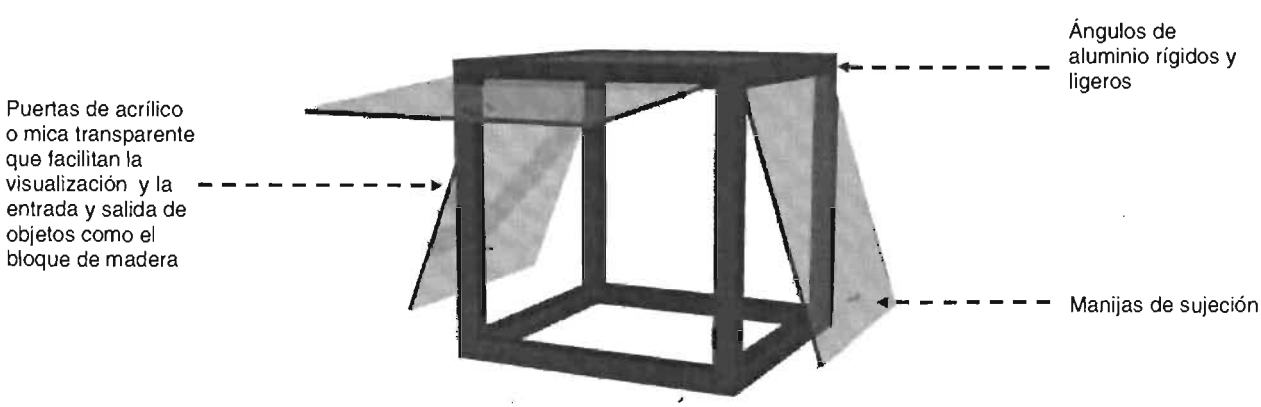


Fig. 2.14. La estructura es de aluminio y sobre ella se monta todo el modelador.

Las paredes se colocarán a manera de puertas que abren hacia arriba con la ayuda de bisagras y manijas.

2.4.2 CHAROLA DEL MODELO, SUJETADORES DEL BLOQUE DE MADERA, TUERCA Y BUJES DE DESPLAZAMIENTO

El bloque de madera que servirá de materia prima para la elaboración del modelo tridimensional, residirá sobre una base de aluminio y se inmovilizará con sujetadores desplazables.

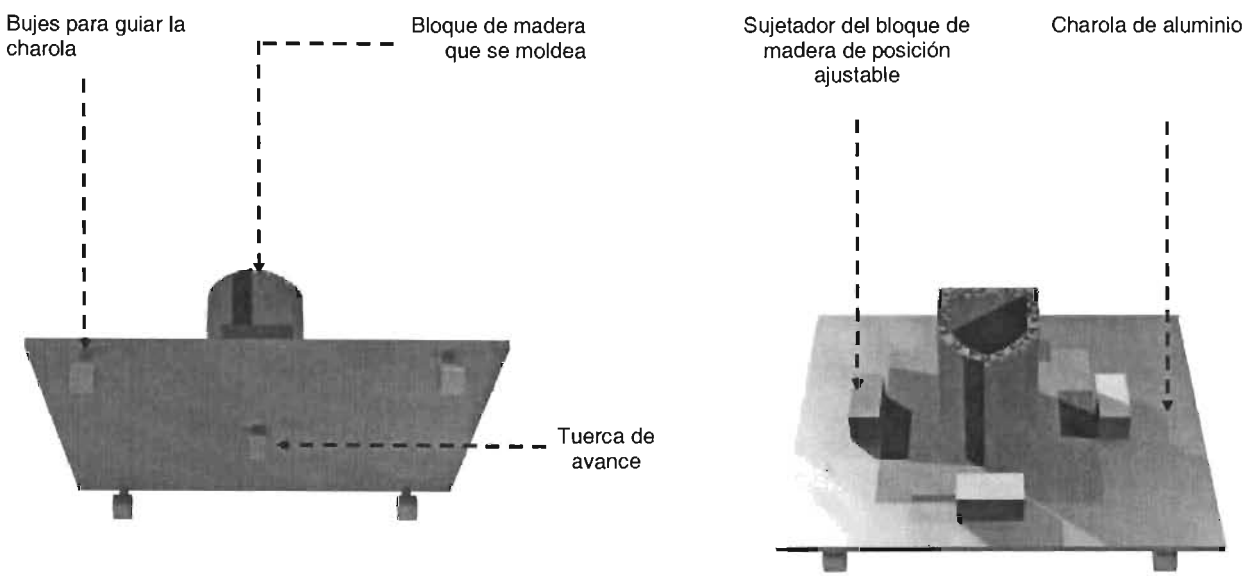


Fig. 2.15. La charola que sostiene el modelador se desplazará sobre un tornillo sin fin y dos guías.

La charola por debajo tiene una tuerca para moverse longitudinalmente al girar el tornillo sin fin que simula el eje Y, y cuatro bujes que rodean dos guías para asegurar el buen desplazamiento.

Cabe señalar que el material anterior se puede adecuar de desechos de impresoras y/o escáners.

2.4.3 TORNILLOS SIN FIN, GUÍAS Y BUJES Y BLOQUES DE SUJECIÓN

El movimiento de la charola anterior se logra con la rotación de un tornillo sin fin sobre el cual se embona una tuerca sujeta a dicha charola, y además se guía con otros dos ejes que a su vez están rodeados por bujes.

El encargado de hacer girar el tornillo sin fin es un motor de pasos que también tiene un buje de sujeción que los une. La figura 2.16 nos permitirá tener una mejor referencia:

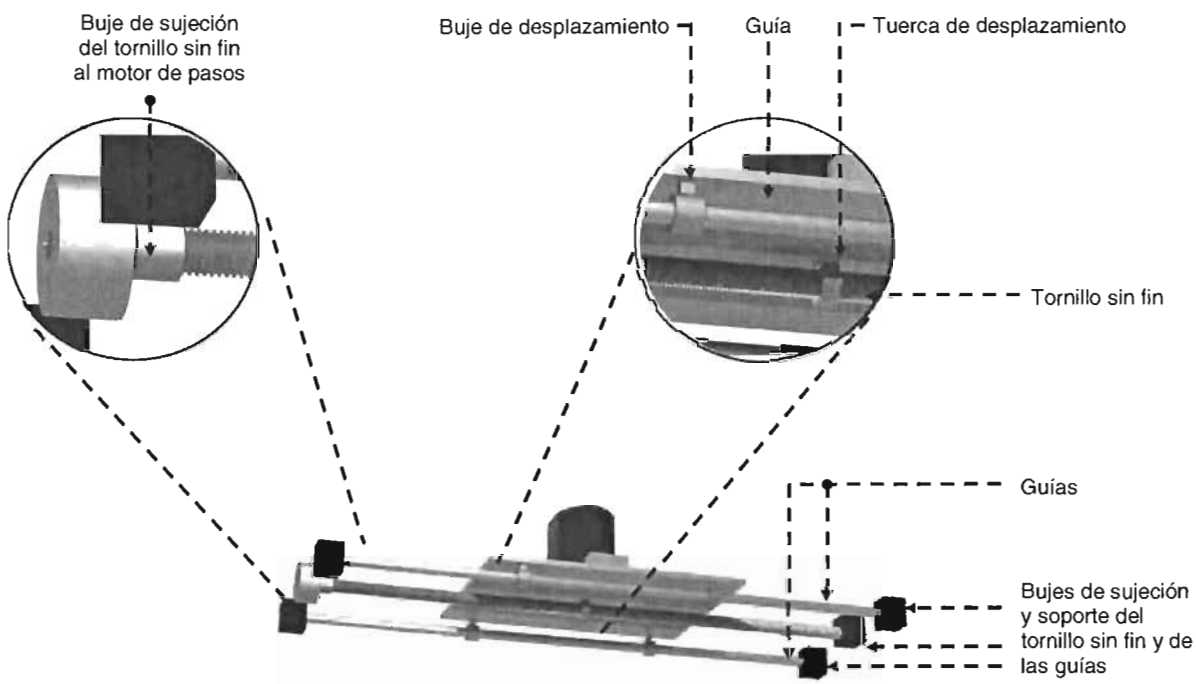


Fig. 2.16. Un motor de pasos mueve un tornillo sin fin el cual a su vez desplaza la charola

Para sostener las guías y el tornillo sin fin, se utilizan pequeños bloques de acrílico o metal, de los cuales sólo el central permite la rotación libre del tornillo.

2.4.4 CHAROLA DEL EJE X

El conjunto de componentes que están directamente ligados al eje Y, se desplazan sobre una charola grande que se mueve sobre el eje X. Veámosla:

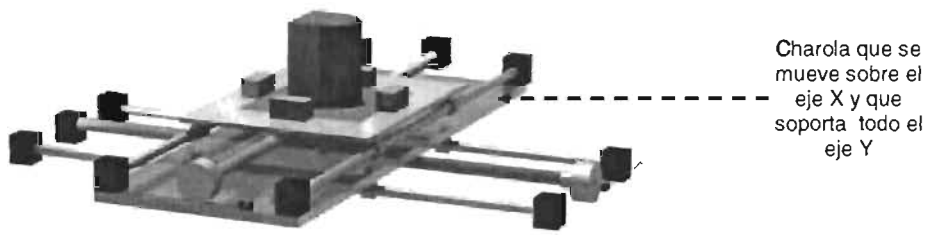


Fig. 2.17. Esta charola se mueve bajo el mismo concepto que aquella que sostiene al bloque de madera

2.4.5 CAJA DEL TALADRO Y DEL CIRCUITO ELECTRÓNICO

Para sostener el taladro se utilizará una caja que se moverá sobre el eje Z de igual manera que lo hacen las otras charolas, es decir con bujes, tuercas, etc.

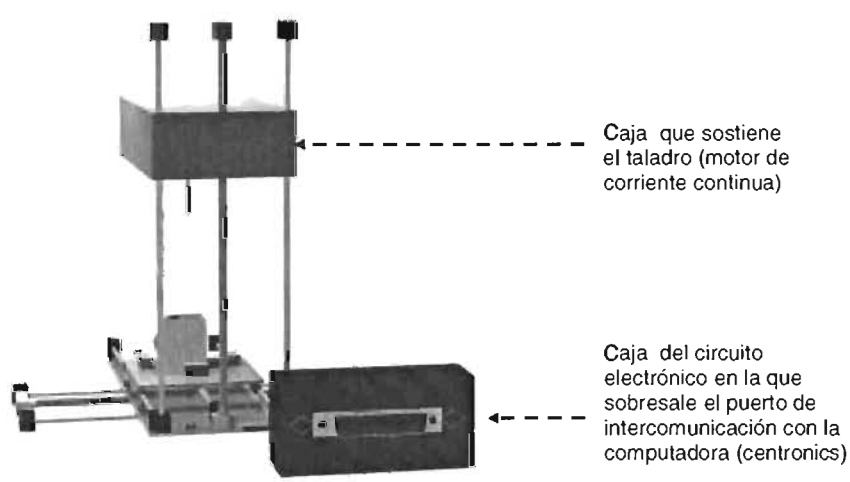


Fig. 2.18. Sobre el eje Z se monta el taladro para perforar el bloque de madera

2.5 COMPONENTES ELECTRÓNICOS

Para controlar los motores, necesitamos construir un circuito electrónico compuesto por una diversidad de elementos como: chips, leds, resistencias, conectores, etc.

Anteriormente describimos dos tipos de motores: bipolares y unipolares, y en base a ellos daremos dos alternativas de diseño aunque la principal y de mayor detalle será la que involucra a los bipolares.

2.5.1 CIRCUITO ELECTRÓNICO PARA MOTORES DE PASO BIPOLARES (PRINCIPAL ALTERNATIVA)

Comenzaremos por el diseño general y después desglosaremos de manera particular cada uno de sus componentes.

En esta alternativa usaremos motores bipolares y el L293D como circuito principal de control y potencia (driver), codificadores y decodificadores, sensores, resistencias y un conector para el cable paralelo de nombre: "centronics".

El centronics y los sensores son fáciles de extraer de ratones, impresoras y escáners, y los demás dispositivos deberán comprarse pero su costo es bajo.

En la figura 2.19 se incluye un diagrama de bloques sobre el circuito electrónico correspondiente:

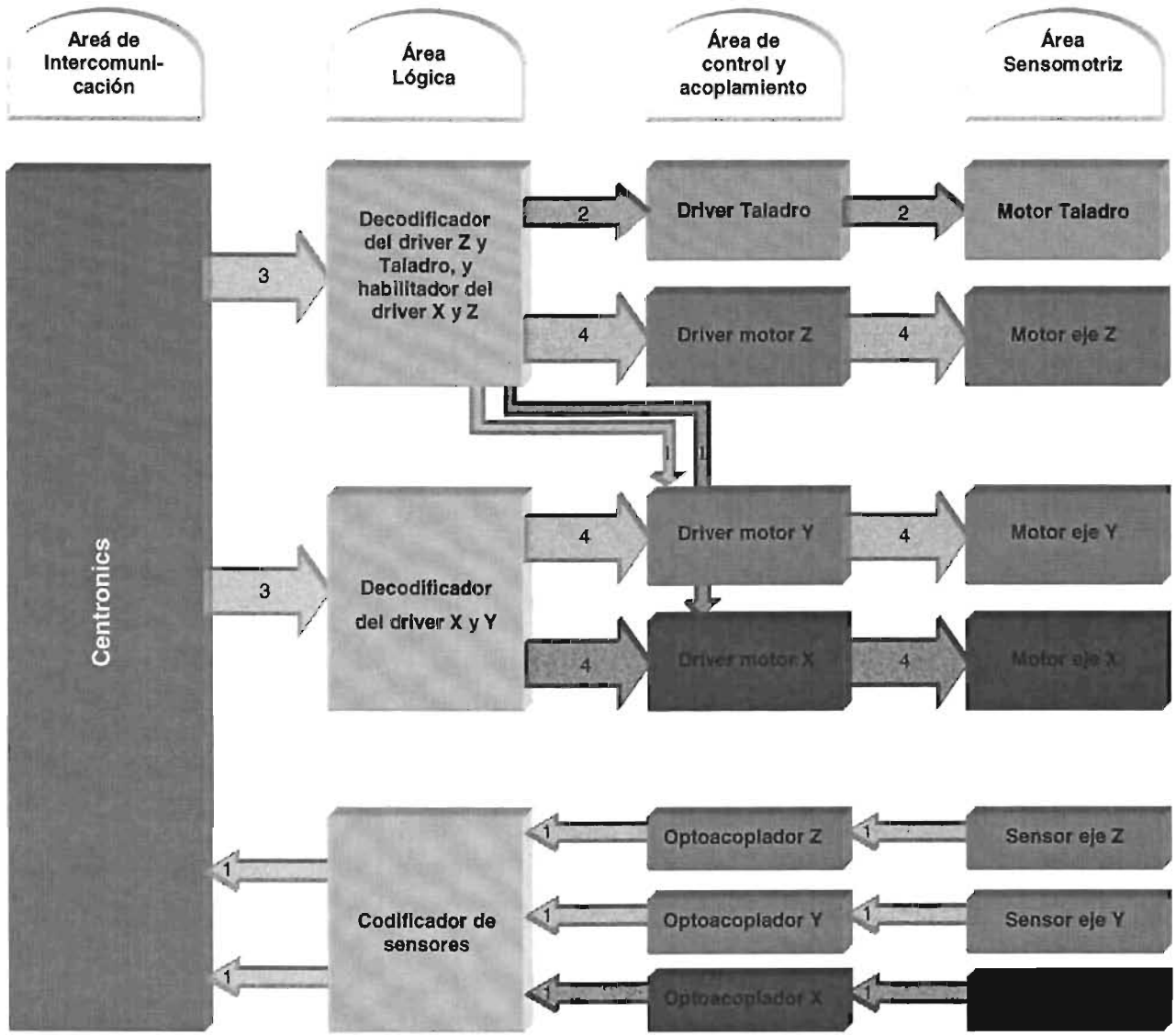


Fig. 2.19 Diagrama de bloques del circuito electrónico del Mofis que opera con motores de paso bipolares

Hemos divido el circuito en cuatro áreas y su explicación es la siguiente:

- *Área de intercomunicación:* incluye el conector para el cable paralelo llamado: "centronics", y será nuestro puerto de intercomunicación (entrada/salida de datos) con la corriputadora.

- *Área lógica:* se compone de dos decodificadores y un codificador que nos servirán para ocupar sólo un byte de información (8 bits o líneas de datos), de tal manera que la programación del software sea más sencilla.

- *Área de control y acoplamiento:* es una capa de potencia hacia los motores y de ajuste de las señales de entrada hacia la computadora para evitar dañarlo con sobrecargas.

- *Área sensomotriz:* contiene los motores de paso que mueven los ejes X, Y y Z, el motor de corriente continua que funciona como taladro y los sensores que detectan las posiciones de cada uno de los ejes.

Las tres primeras líneas que salen del centronics se envían a un decodificador que las transformará en ocho para que de ellas, dos muevan al motor taladro después de pasar por el circuito de control y potencia (driver), cuatro al motor del eje Z[9] que tarnbién deben atravesar un controlador y las dos restantes habilitarán o deshabitarán los drivers que manejan a los motores X y Y[10].

Las siguientes tres líneas de salida generan otras ocho al pasar por el decodificador del eje X y Y, y como es de esperarse se introducirán de cuatro en cuatro a los drivers X y Y, para que estos a su vez controlen los motores de los ejes respectivos.

Los tres sensores detectan las posiciones de los ejes X, Y y Z, y al ser bloqueados o desbloqueados por unas pequeñas laminitas envían cierta señal a los optoacopladores que impiden el paso de corrientes dañinas al puerto de comunicación, después esas líneas son codificadas en dos e introducidas al centronics para que al recibirlas la computadora, el software pueda analizarlas y decida si mueve los ejes a su posición original.

[9] Según la tabla de configuración de los motores bipolares y la que daremos un poco más adelante sobre los drivers L293D, necesitamos cuatro líneas para hacerlo girar.
[10] En las páginas siguientes de programación de los decodificadores y codificadores veremos la razón por la cual este decodificador controla la habilitación e inhabilitación de los drivers X y Y

2.5.2 CIRCUITO ELECTRÓNICO PARA MOTORES DE PASO UNIPOLARES (SEGUNDA ALTERNATIVA)

La otra alternativa de construcción del circuito electrónico involucra motores de paso unipolares y drivers UCN5804 que son costosos y muy complicados de conseguir pero que facilitan enormemente el control.

En esta opción nos limitaremos a diseñar el diagrama de bloques sin entrar en muchos detalles:

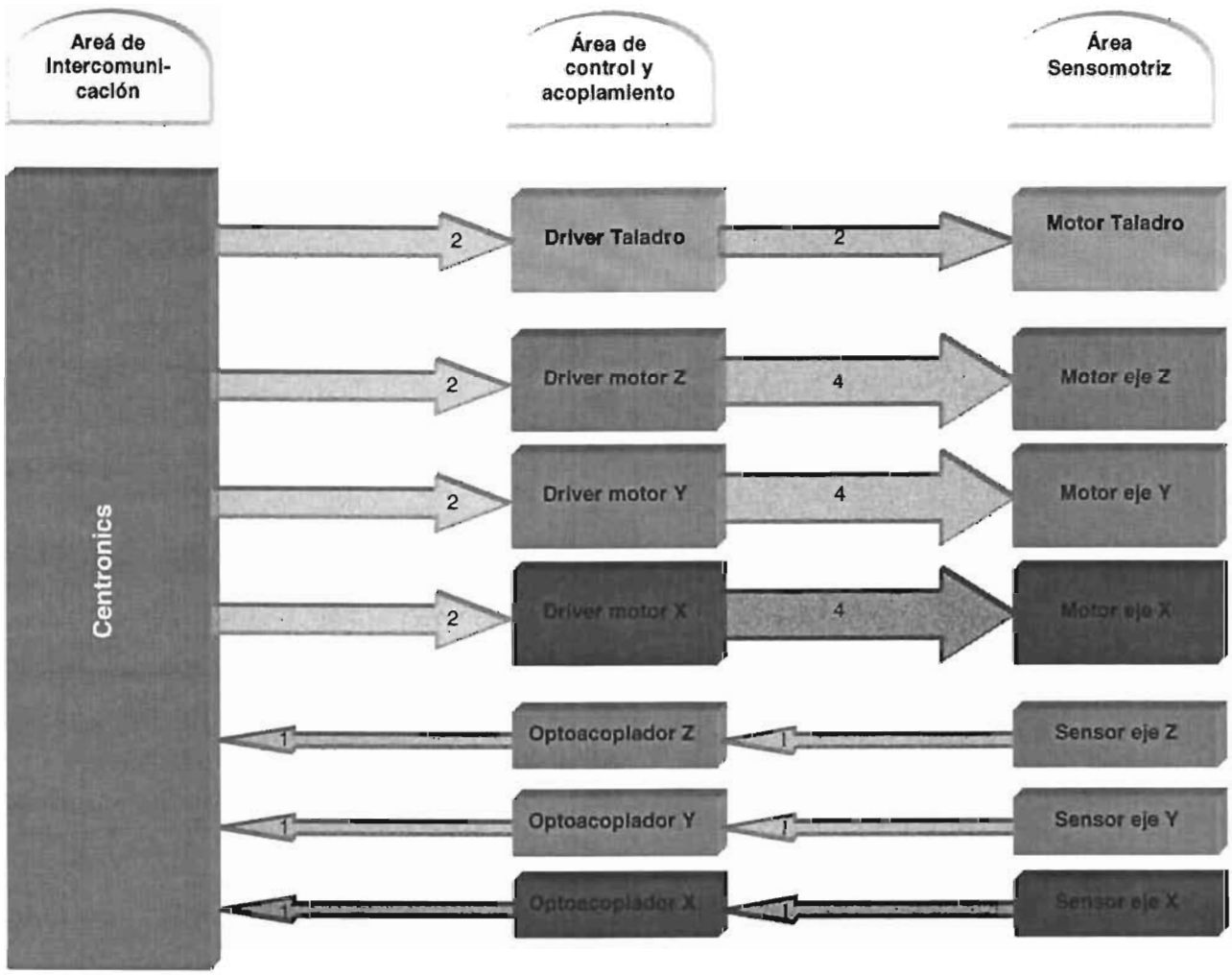


Fig. 2.20 Diagrama de bloques del circuito electrónico que opera con motores de paso unipolares

La explicación al diagrama anterior es similar a la de los bipolares, lo único que cambia es la ausencia de decodificadores y codificadores puesto que el controlador UNC5804 exige menos líneas de control según se verá más adelante.

2.5.3 PUERTOS DE INTERCOMUNICACIÓN

La entrada y salida de información desde y hacia una computadora es posible gracias a mecanismos de intercambio de datos llamados: "puertos", siendo los más comunes: el paralelo, usb, serie e infrarrojo. A través de ellos, es posible conectar dispositivo externos tales como: ratones, impresoras, escáners, cámaras fotográficas, entre otros.

Para controlar nuestro modelador usaremos el puerto paralelo por su simpleza de manejo y daremos su explicación junto a la del serie, usb e infrarrojo para el caso en el que se desee construir otro periférico diferente.

2.5.3.1 PUERTO SERIE

La palabra serie se refiere a una secuencia de elementos, números, etc. dispuestos uno detrás del otro, de tal manera que se puede crear una fila con ellos.

El puerto serie es el más simple de los puertos, y como su nombre lo indica, envía y recibe los datos de manera secuencial, es decir uno por uno.

A este tipo de puerto también se le llama puerto: "COM" o de comunicaciones, ya que su objetivo inicial era comunicar a través de un módem, pero después también se comenzó a utilizar para la conexión de ratones, impresoras, pocket PC's, handhelds, Palms, entre otros.

La velocidad de transferencia que alcanza es de aproximadamente 115 kbps, siendo suficiente para dispositivos como los anteriormente mencionados, pero resulta inútil para hardware que demanda altas tasas de transferencia.

El COM sólo requiere de una línea de datos para el envío y otro para la recepción, lo que permite que la comunicación sea bidireccional y simultánea.

Para liberar al procesador de la tarea de comunicación con el exterior, el puerto utiliza un chip llamado: UART (Universal Asynchronous Receiver/Transmitter o Transmisor/Receptor Asíncrono Universal), el cual se encarga de capturar los datos que se le envían desde el interior de la computadora y transferirlos de manera seriada hacia el dispositivo de comunicación externa basándose en un protocolo[11] . También hace la tarea inversa, es decir, la de recibir los datos del exterior en forma seriada para convertirlos en formatos de palabras y transmitirlos hacia dentro del sistema. Además del UART, también cuenta con dos dispositivos de memoria llamados buffers de los cuales, uno permite retener la información que será enviada al exterior y el otro para la que se ha recibido. La capacidad de dichos buffers oscila entre los 16 y 64 kb.

El conector del puerto recibe el nombre de: RS-232 y los conectores del cable para conexión se llaman: DB9.

En la figura siguiente se muestra un DB9 y la distribución de sus 9 terminales (de ahí el nombre):

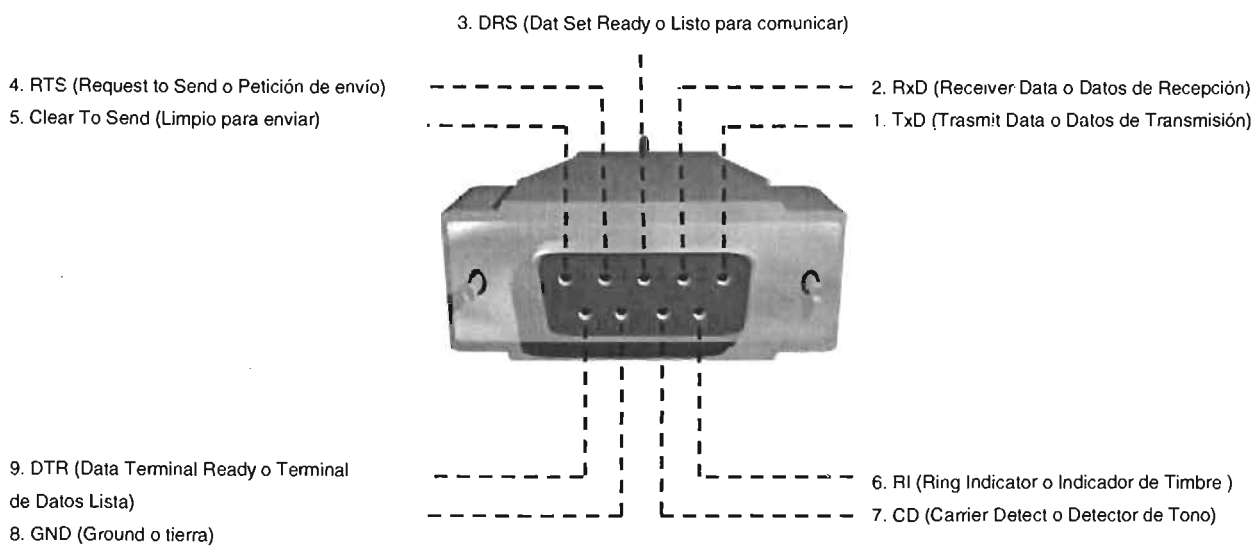


Fig. 2.21 Conector para transmisión serie

La descripción de cada una de las terminales se incluye en la tabla 2.5.

[11] Un protocolo es un conjunto de reglas que especifican la manera en la cual se comunicarán dos dispositivos.

Pin	Nombre		Descripción
1	TxD (Transmitter Data)	Datos de transmisión	Permite transmitir los datos
2	RxD (Receiver Data)	Datos de recepción	Permite recibir los datos
3	DSR (Data Set Ready)	Listo para comunicar	Indica que la comunicación puede empezar
4	RTS (Request To Send)	Petición de envío	Sirve para preguntar al módem si esta listo para el intercambio de información.
5	CTS (Clear To Send)	Limpio para enviar	Se utiliza para recibir la contestación del módem a la petición hecha en el pin anterior.
6	RI (Ring Indicator)	Indicador de timbre	Su función es la de detectar la entrada de una llamada.
7	CD (Carrier Detect)	Detector de tono o portadora	Señala si el módem está conectado a una línea telefónica con tono.
8	GND (Ground)	Tierra	Referencia para todas las señales.
9	DTR	Terminal de Datos Lista	Indica el estado de la terminal

Tabla 2.5 Distribución de pines del puerto serie

2.5.3.2 PUERTO PARALELO

Hasta hace poco tiempo, la mayoría de las impresoras y escáners se conectaban a la computadora a través de un puerto de ranura ancha situado en la parte posterior compuesto por 25 agujeros, denominado: "puerto LPT" (Line Port Terminal o Puerto Terminal de Línea).

Este puerto a diferencia del serial, permite enviar y recibir varios datos al mismo tiempo (8 bits ó 1 byte), de ahí el nombre que lo identifica. El paralelo generalmente se direcciona por default en la ubicación hexadecimal &378, aunque puede cambiar.

El conector para el LPT es el DB-25 y los dispositivos periféricos utilizan en su puerto de intercomunicación una ranura llamada "Centronics", que consta de 36 líneas.

El tipo de transmisión del puerto paralelo es half-duplex, eso quiere decir que su comunicación es bidireccional pero que cuando envía no recibe y viceversa.

Las señales de salida del puerto llevan una corriente de 20mA y el potencial o voltaje de salida alto es de 5v de C.C., mientras que el potencial bajo es de 0.5v.

Utilizaremos este puerto para comunicarnos con el Mofis por su facilidad de manejo.

Veamos con mayor detalle la composición de un conector DB-25 y la descripción de sus pines:

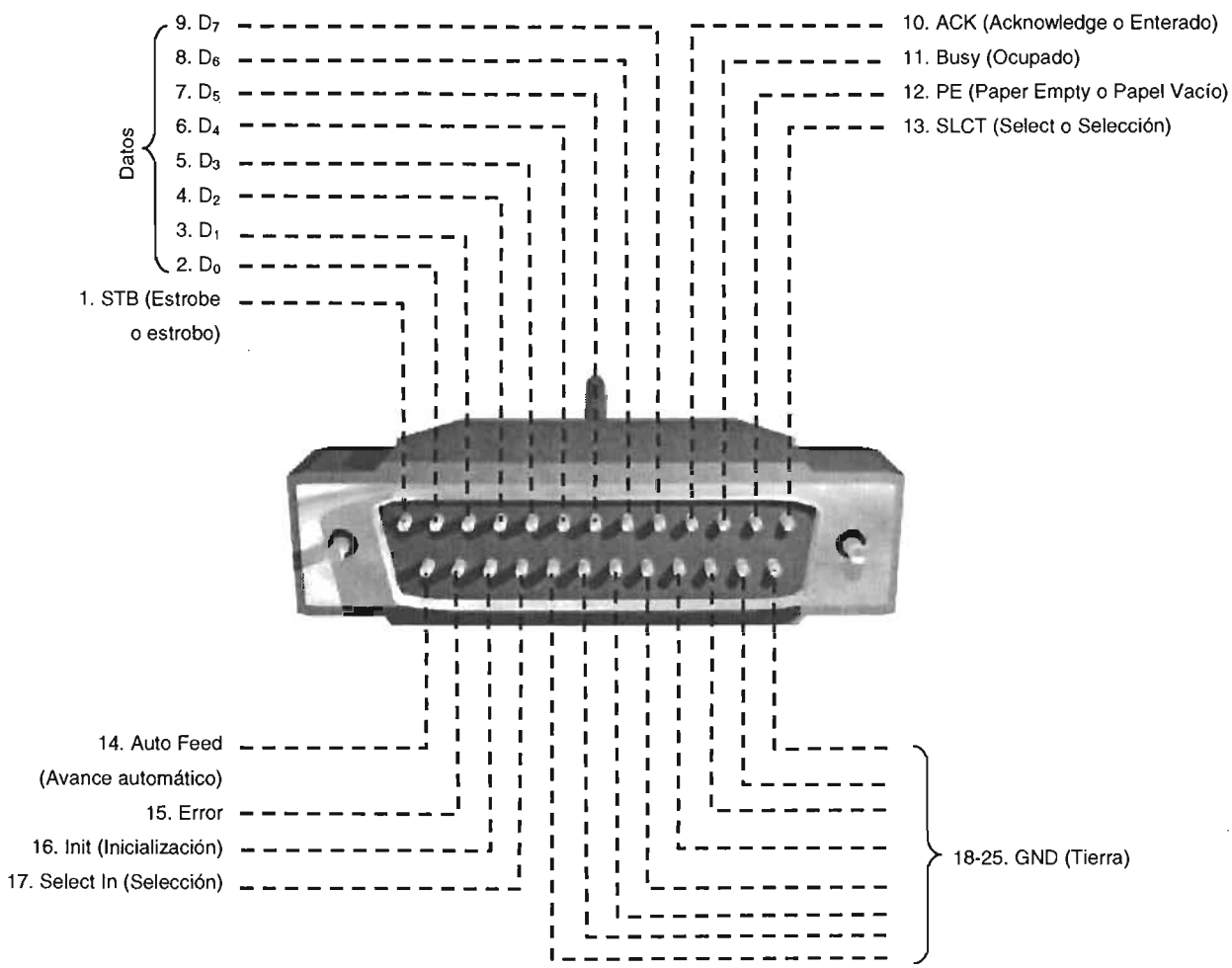


Fig. 2.22 El conector DB-25 incluye un conjunto de líneas para la transmisión de 8 bits en forma paralela

Pin	Nombre		Descripción
1	STROBE	Estrobo	Hace una cambio de nivel lógico, para indicar al dispositivo externo que ha enviado un byte al buffer para que lo lea.
2	D_0	Dato 0	Dato cero o menos significativo (LSB o Lower Significative Bit)
3	D_1	Dato 1	Dato uno
4	D_2	Dato 2	Dato dos
5	D_3	Dato 3	Dato tres
6	D_4	Dato 4	Dato cuatro
7	D_5	Dato 5	Dato cinco
8	D_6	Dato 6	Dato seis
9	D_7	Dato 7	Dato siete o más significativo (MSB o Major Significative Bit)
10	ACK (Acknwledge)	Enterado	Línea de entrada que indica a la computadora que ya se ha leído el dato y que puede escribir otro.
11	BUSY	Ocupado	Indica que se encuentra ocupada la impresora y por lo tanto no puede recibir datos.
12	PE (Paper empty)	Vació de papel	Le indica a la computadora que la impresora no tiene papel
13	SLCT (Select)	Selector	Señala que la impresora está lista para usarse, es decir, que esta en línea.
14	AUTO FEED XT	Avance automático	Sirve para regresar al inicio de la línea de la impresora.
15	ERROR	Error	La impresora indica que le ha sucedido un error.
16	INIT	Inicialización	Si existe un cambio de estado, la impresora se reinicializa.
17	SLCT IN	Selección	Su función es la de poner fuera de servicio a la impresora.
18-25	GND	Tierra	Referencia para todas las señales

Tabla 2.6 La distribución de los pines del puerto paralelo demuestran que fue diseñado principalmente para enlazar impresoras

2.5.3.3 PUERTO USB

Los puertos serie y paralelo tienen muchas limitantes, como: velocidad de transferencia reducida, imposibilidad de conectar más de un dispositivo a la vez y de reconocerlos "al vuelo" o "en caliente".

Las siglas USB provienen de las palabras Universal Serial Bus (Bus Serial Universal), y es que en la actualidad este puerto permite conectar prácticamente cualquier tipo de periférico como: teclados, ratones, bocinas, impresoras, lectores externos de CD, módems, webcams, cámaras fotográficas digitales, escáners, entre otros.

El USB es un estándar que permite conectar hasta 127 dispositivos partiendo de un único conector. Con una velocidad de 12 Mbps[12] (Versión 1.1) y 480 Mbps (Versión 2.0), el objetivo del USB es eliminar las carencias del puerto serie y paralelo, además ambos puertos sólo permiten conectar un dispositivo al mismo tiempo. El USB con una manguera flexible de pocos hilos consigue velocidades muy por encima de las que se pueden transmitir con otros tipos de puertos.

El USB tiene un gran ancho de banda y permite añadir dispositivos "al vuelo", esto es, sin apagar la computadora o el dispositivo que se va a conectar.

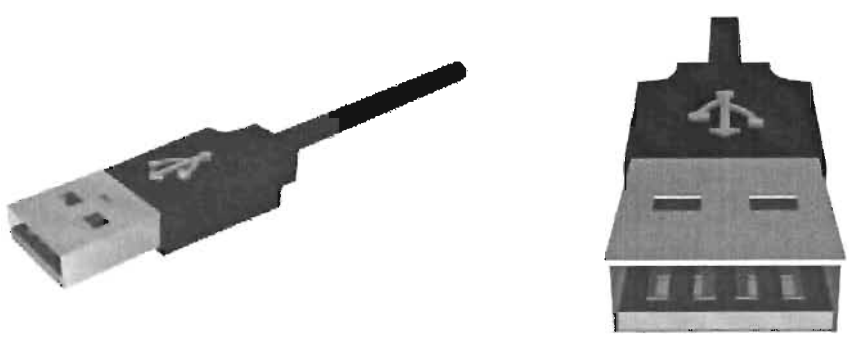


Fig. 2.23 Conector USB tipo "A" visto desde dos ángulos diferentes

[12] Mbps son las siglas de Mega Bits Per Second (Mega Bits Por Segundo). Es una unidad de medida de transmisión de datos entre la computadora y los periféricos

Un cable USB como se puede apreciar en la figura 2.21, se compone de sólo cuatro líneas, de las cuales, una alimenta con 5v y una corriente de 5400mA, otra sirve de tierra y las otras forman un par trenzado que permite el envío y recepción de información.

2.5.3.4 PUERTO INFRARROJO

Los dispositivos electrónicos más comunes que utilizan la comunicación por rayos infrarrojos (IrDA), son los controles remotos de los televisores y estéreos.

Este tipo de puerto permite comunicación inalámbrica de alcance relativamente corto. Las siglas IrDA significan: Infrared Data Association o Asociación de Datos Infrarrojos.

El IrDA brinda la posibilidad de conectar equipos portátiles como: laptops, Pocket PCs, Handhelds, Palms, Impresoras, concentradores, entre otros.

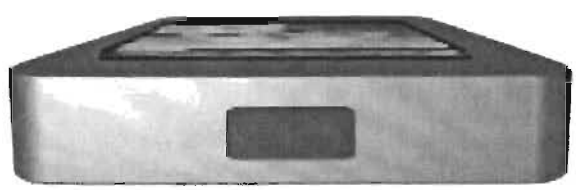


Fig. 2.24 Vista frontal del panel infrarrojo de una computadora de bolsillo o Pocket PC

Existen otras formas de comunicar una computadora como: Wireless, Bluetooth, entre otras, pero su aprovechamiento requiere de explicaciones fuera del alcance de nuestros objetivos.

2.5.4 CONTROLADORES

Para manejar los motores de corriente continua (uno que ocuparemos como taladro) y tres de pasos (como ejes de coordenadas de desplazamiento), además de

utilizar energía eléctrica para lograr el movimiento, también requeriremos de circuitos electrónicos que nos permitan comandarlos y que al mismo tiempo nos ofrezcan mayor potencia de salida. Dichos circuitos son denominados: "controladores o drivers.

Para este proyecto en el que ocuparemos motores de paso bipolares, el controlador que utilizaremos será el L239D el cual también servirá para comandar al de corriente continua.

En las secciones siguientes describiremos al circuito L293D y también al UCN5804 para el caso en el que contemos con motores de paso unipolares.

2.5.4.1 CONTROLADOR PARA MOTORES DE PASO BIPOLARES Y DE CORRIENTE CONTINUA (L293D)

Este circuito integrado es un driver de 4 canales que nos ofrece 600 mA por canal (suficientes para el modelador). Cada uno de los canales se controla por señales compatibles TTL y cada pareja de canales se conecta a una señal de control que permite habilitarlas/deshabilitarlas.

El L293D no sólo controla motores de paso bipolares, sin que también ofrece la posibilidad de manejar dos motores de corriente continua en un solo sentido o uno sólo en dos sentidos

Este tipo de chip es muy comercial, fácil de encontrar y de costo reducido. Sus características más relevantes son:

- Protección contra sobrecalentamientos.
- 600 mA de corriente por canal.
- Facilidad para habilitar y deshabilitar canales.
- Alta inmunidad ante el ruido.
- Diodos internos de protección contra corrientes de retorno originadas por el arranque de los motores.
- Alimentación separada para las cargas de la lógica[13].

[13] La descripción del circuito fue obtenida del la hoja de especificaciones del L293D de SGS-Thompsons Microelectronics

Ahora veamos una representación del L293D junto con el significado de cada uno de sus pines o "patitas":

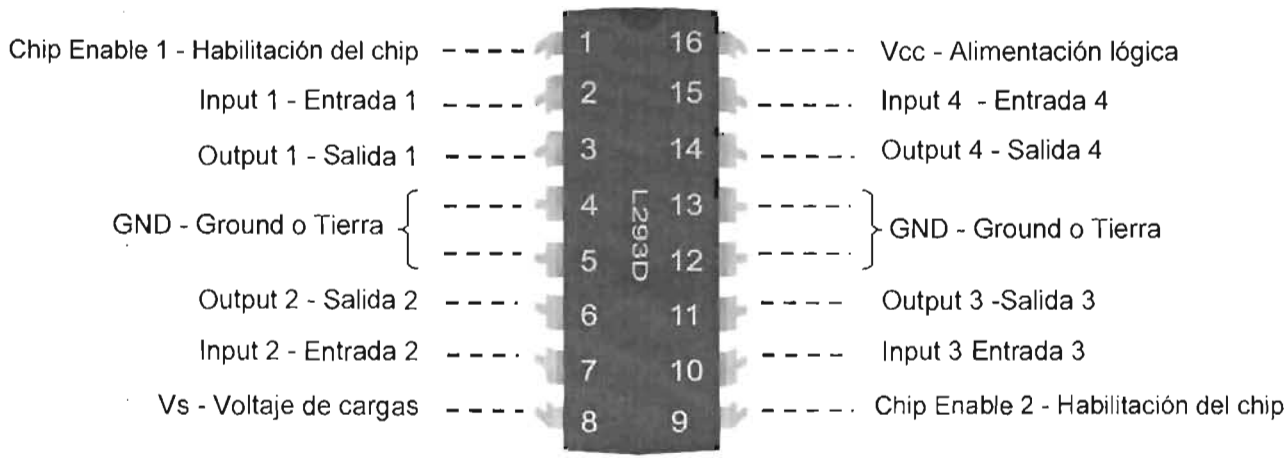


Fig. 2.25 Esquema del L293D para controlar motores de corriente continua y de pasos

Para comprender la funcionalidad de cada uno de los pines, en la tabla 2.7 se incluye la descripción:

Pin	Nombre		Descripción
1	Chip Enable 1	Habilitación de las salidas de la izquierda	Permite habilitar los canales de salida 1 y 2
2	Input 1	Entrada 1	Línea de entrada del canal 1
3	Output 1	Salida 1	Salida del canal 1
4	GND	Tierra	Tierra o referencia
5	GND		
6	Output 2	Salida 2	Salida del canal 2
7	Input 2	Entrada 2	Línea de entrada del canal 2
8	Vs	Voltaje suministro	Recibe la alimentación para las cargas
9	Chip Enable 2	Habilitación de las salidas de la derecha	Permite habilitar los canales de salida 3 y 4
10	Input 3	Entrada 3	Línea de entrada del canal 3
11	Output 3	Salida 3	Salida del canal 3
12	GND	Tierra	Tierra o referencia
13	GND		
14	Output 4	Salida 4	Salida del canal 4
15	Input 4	Entrada 4	Línea de entrada del canal 4
16	Vss	Voltaje lógico	Entrada de alimentación para la parte lógica del circuito

Tabla. 2.7 Tabla descriptiva del L293D

Para controlar un motor de pasos bipolares, debemos de conectarlo según el esquema de la figura 2.26:

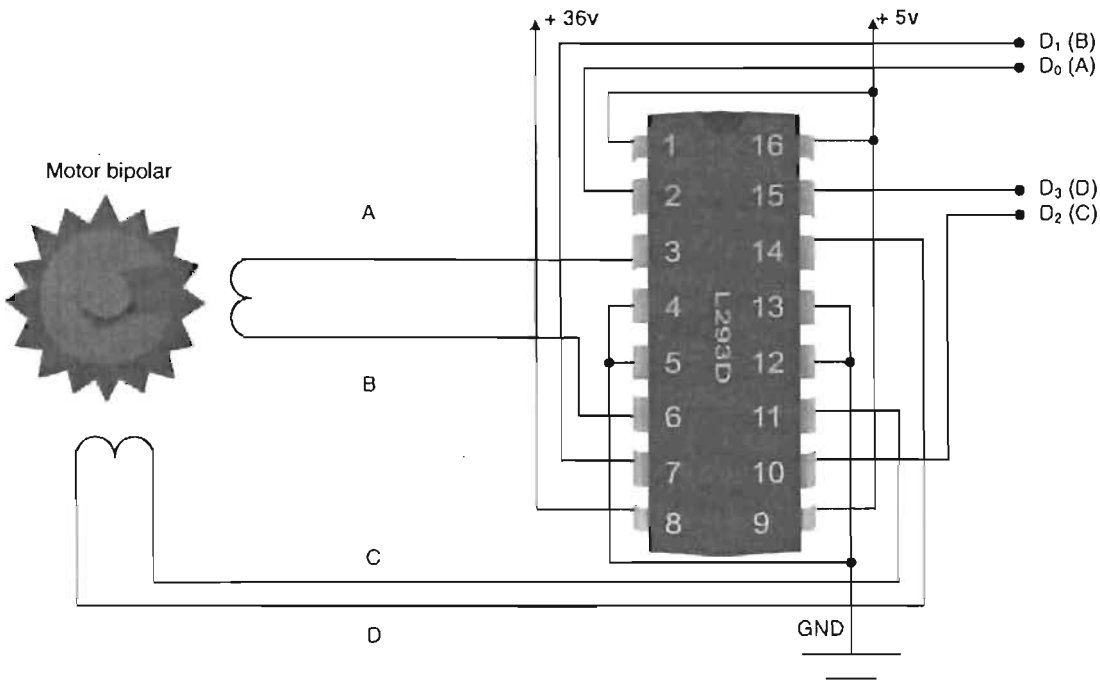


Fig. 2.26. Conexión de un motor de pasos bipolar (las entradas de control son D_0, D_1, D_2 y D_3)

Debido a que también usaremos un motor de corriente continua para el taladro del Mofis, es necesario que veamos la configuración de conexión a utilizar:

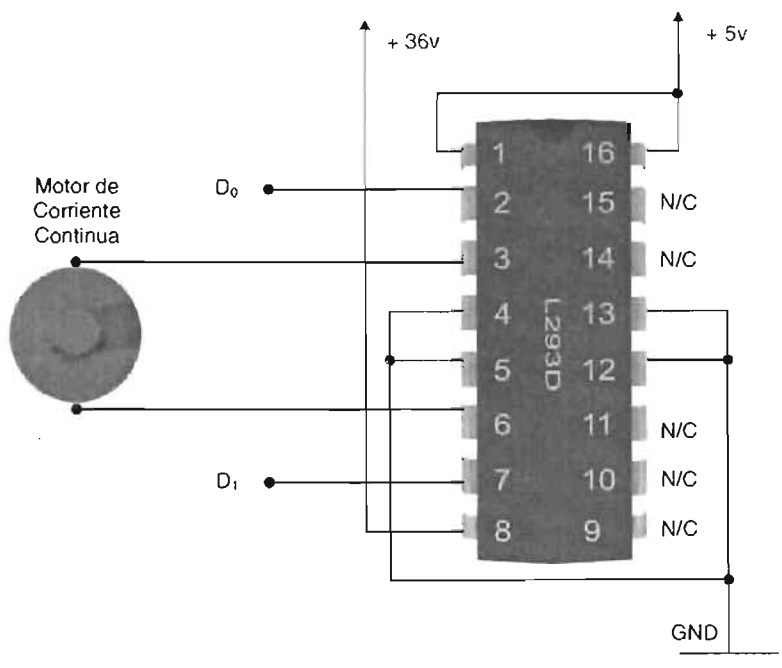


Fig. 2.27. Conexión de un motor de corriente continua (D_0 y D_1 son las líneas de control)

Para que el L293D controle el motor de pasos en un sentido u otro, sólo bastará con aplicar pulsos eléctricos de 5v según la tabla de configuración siguiente:

	Línea				
Paso	A	B	C	D	Valor decimal
1	1	0	1	0	10
2	1	0	0	1	9
3	0	1	0	1	5
4	0	1	1	0	6

Tabla 2.8. Secuencia de pulsos para hacer girar el motor de pasos en sentido horario (para el otro sentido se invierte la secuencia)

Si queremos que este mismo circuito controle un motor de corriente continua, la alimentación de los pines de control deberá ser la siguiente:

Datos		
D_1	D_0	Sentido de giro
0	0	Ninguno
0	1	Antihorario
1	0	Horario
1	1	Ninguno

Tabla 2.9. Configuración para control de motores de corriente continua

En las tablas anteriores se puede observar que no incluimos diodos de protección para el circuito, puesto que ya los contiene de manera interna.

2.5.4.2 CONTROLADOR PARA MOTORES DE PASO UNIPOLARES (UCN5804)

Este chip nos permite controlar motores de paso unipolares de una manera muy simple, también nos da la posibilidad de elegir el tipo de formato de control (un paso o de o de onda, dos pasos y medio paso).

A diferencia del L293D, el UCN5804 es muy difícil de conseguir y su costo es mayor.

El UCN5804 combina el bajo consumo de la lógica CMOS con salidas de alta-corriente y alto–voltaje. Ofrece un control y manejo total para un motor de pasos unipolar de cuatro fases con salidas de corriente continua de 1.25 A por fase y 35 V.

El formato de una fase energiza una bobina al mismo tiempo en la secuencia A-B-C-D o viceversa. Este tipo de configuración consume la menor cantidad de corriente pero es la de menor torque. La de dos fases es la que ofrece el mejor torque de retención al energizar dos fases adyacentes en cada posición, la secuencia es AB-BC-CD-DA o viceversa. Finalmente la de media fase, alterna entre una y dos fases, por lo que su secuencia es: A-AB-B-BC-C-CD-D-DA o viceversa.

Enlistemos ahora las características principales de este controlador:

- Corriente máxima de salida 1.5 A
- Voltaje de salida 35 V
- Formatos de onda de salida de una, dos y media fase.
- Diodos de protección internos
- Habilitación de salidas
- Control de dirección.
- Reseteo al encender (Power-On Reset)
- Circuitería interna de bloqueo térmico.[14]

[14] La descripción del circuito fue extraída de la hoja de especificaciones del UCN5804 ofrecida por Allegro MicroSystems

A continuación veremos una imagen del chip junto con el nombre de cada uno de sus pines:

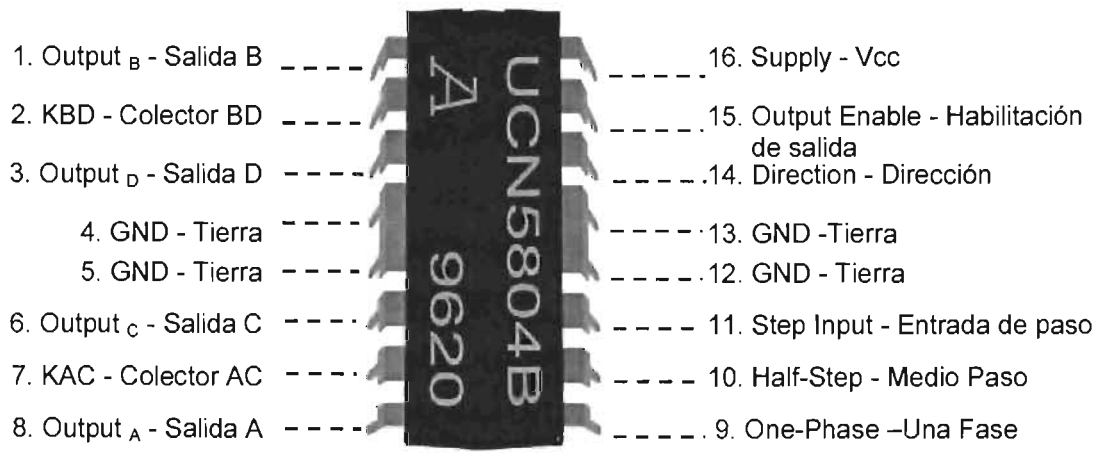


Fig. 2.28 Circuito integrado UCN5804 para controlar motores de paso unipolares

La tabla 2.10 ofrece la de definición de cada uno de los pines o "patitas" del UCN5804.

Pin	Nombre		Descripción
1	Output $_B$	Salida B	Segunda línea para el motor de pasos.
2	KBD	Colector BD	Permite controlar la amplificación de salida a las líneas BD.
3	Output $_D$	Salida D	Cuarta y última línea para el motor de pasos.
4	GND	Tierra	Tierra
5	GND	Tierra	Tierra
6	Output $_C$	Salida C	Tercera línea para el motor de pasos.
7	KAC	Colector AC	Permite controlar la amplificación de salida a las líneas AC.
8	Output $_A$	Salida A	Primera línea para el motor de pasos.
9	One-Phase	Una Fase	Este pin junto con el 10 determinan el formato del motor. Su configuración se muestra más adelante
10	Half-Step	Medio Paso	Conjuntamente con el pin 9 determina la configuración del motor.
11	Step Input)	Entrada de paso	Recibe los pulsos de reloj que hacen girar el motor
12	GND	Tierra	Tierra
13	GND	Tierra	Tierra
14	Direction	Dirección	Permite indicar la dirección de giro del motor
15	Output Enable	Habilitación de salida	Su función es la de habilitar o deshabilitar las salidas hacia el motor
16	Suplí	Vcc	Alimentación del circuito

Tabla. 2.10 Tabla descriptiva de los pines del circuito UCN5804

Debido a que la fuerza de torque para un proyecto como éste es muy importante, solamente proporcionaremos de manera explícita la configuración correspondiente para que un motor de pasos unipolar opere en dos fases.

Los pines que definen el tipo de fase son el 9 y 10, y las salidas hacia el motor serán las que corresponden a las líneas A,B,C y D. Revisemos la tabla de configuración correspondiente:

Pin 9 (Half Step)=L , Pin 10 (One Phase)=L				
Paso	**A**	**B**	**C**	**D**
POR	1	0	0	1
1	1	0	0	1
2	1	1	0	0
3	0	1	1	0
4	0	0	1	1

Tabla. 2.11 Configuración del UCN5804 para que accione en motores en dos fases

El POR (Power on Reset o Reinicio al encender), nos inicializa el circuito para poder aplicar cada uno de los cuatro pasos, que se deben realizar de manera cíclica.

La manera de conectar un motor unipolar para operar en dos fases es:

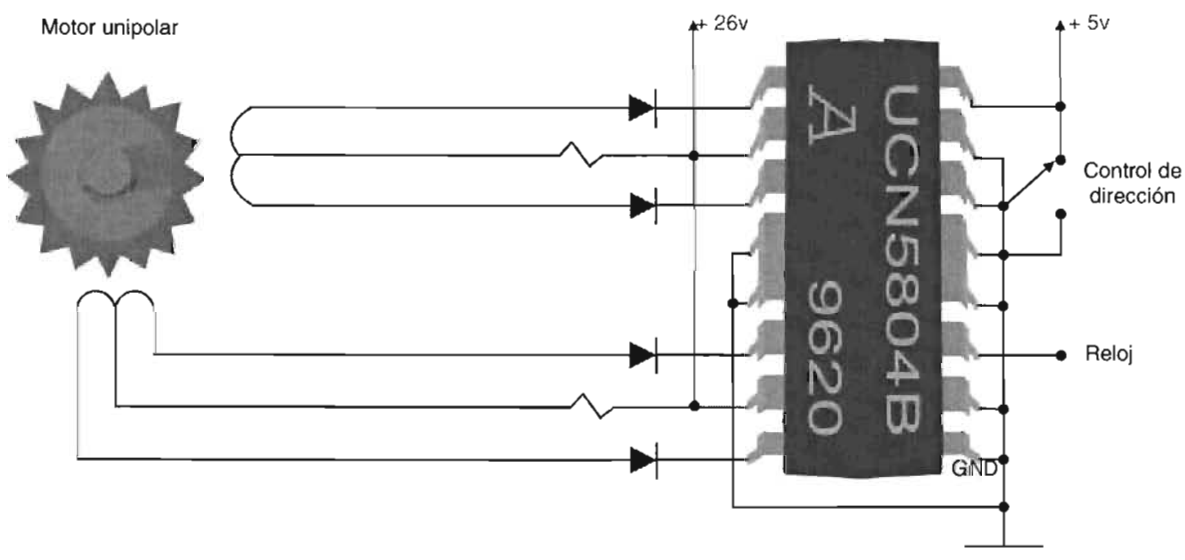


Fig. 2.29. Esquema de conexión del unipolar al UC5804 para su comportamiento en dos fases

2.5.5 DECODIFICADORES Y CODIFICADORES (PROGRAMABLES)

El puerto paralelo maneja 8 bits de e/s que resultan insuficientes para controlar a todos los motores y sensores, y una solución a dicho problema se encuentra en aprovechar otras líneas o crear decodificadores y codificadores intermediarios.

En este proyecto utilizaremos dispositivos electrónicos programables GAL20V8B como decodificadores y codificadores, y la manera de programarlos será mediante el lenguaje VHDL.

2.5.5.1 DISPOSITIVOS PROGRAMABLES (GAL20V8B)

Estos chips pueden ser programados en lenguaje VHDL y grabados en grabadores de memorias para que realicen las funciones que deseemos.

La figura 2.30 incluye la distribución de pines del GAL20V8B:

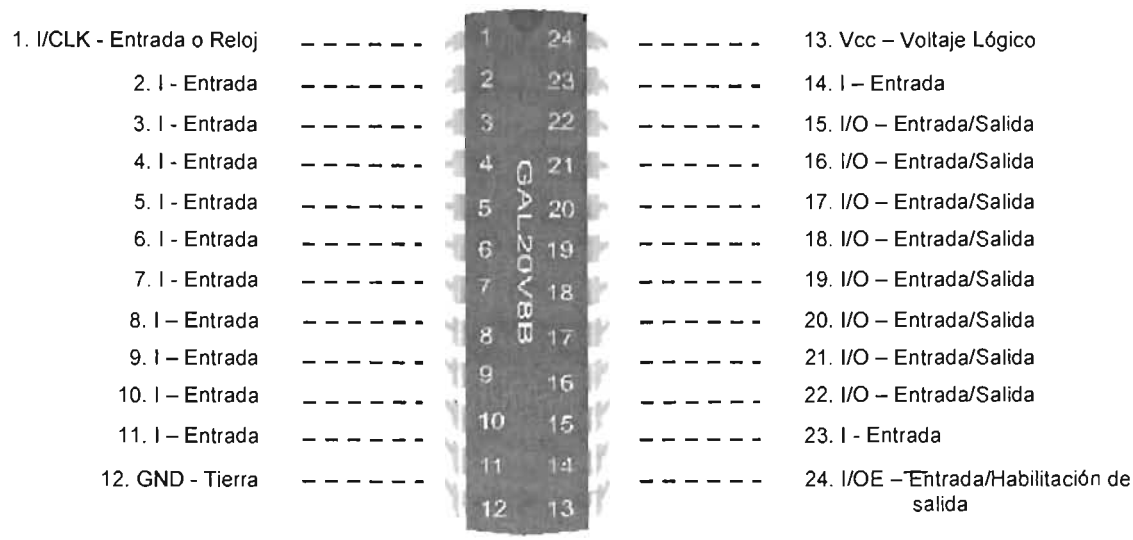


Fig. 2.30 Dispositivo programable GAL20V8B

Antes de programar un circuito de este tipo, es preciso elaborar una tabla de operación.

El primer decodificador a desarrollar servirá de intermediario entre los controladores del motor de corriente continua (taladro), de pasos (para el eje Z) y para habilitar los drivers X y Y. El segundo se encargará de los correspondientes al eje Y y X. Mientras que el único codificador a usar manejará los tres sensores.

En la tabla 2.12 se incluye el modo de operación del primer decodificador:

	Entradas			Salidas							
	Datos puerto LPT			Motor Taladro		Motor Z				Habilitación	
										Y	X
No.	D_2	D_1	D_0	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0	1
3	0	1	1	0	0	0	0	0	0	1	0
4	1	0	0	0	1	1	0	1	0	0	0
5	1	0	1	0	1	1	0	0	1	0	0
6	1	1	0	0	1	0	1	0	1	0	0
7	1	1	1	0	1	0	1	1	0	0	0

Tabla 2.12. Decodificación para mover el motor de pasos del eje Z, el taladro y para habilitar los motores de la mesa de coordenadas

En la tabla anterior se puede observar que tan sólo tres líneas de entrada, pueden ser decodificadas en ocho de salida, y que además de controlar dos motores (corriente continua y de pasos), es posible habilitar y deshabilitar los controladores del eje Y y X para evitar sobrecalentamientos.

La manera de programar la tabla anterior en lenguaje VHDL con su respectiva documentación, se muestra en la figura 2.31:

```
-- 1)UTILIZACIÓN DE LAS LIBRERIAS
library ieee;
use ieee.std_logic_1164.all;

-- 2)DEFINICIÓN DE LA ENTIDAD DECODIFICADORA PARA EL EJE Z Y EL TALADRO
entity deco_zcc is port(
 entradas: in std_logic_vector (2 downto 0);  --3 BITS DE SALIDA DEL LPT
 salidas: out std_logic_vector (7 downto 0)); --8 BITS PARA MOVER EL MOTOR Z,
end deco_zcc;                                 --  EL TALADRO, Y HABILITAR LOS
                                              --  MOTORES Y y X

-- 3)DEFINICION DEL COMPORTAMIENTO DEL DECODIFICADOR
architecture arq_deco_zcc of deco_zcc is
begin
 process(entradas) begin
  case entradas is
   when "000" =>salidas<= "00000000";     --DESHABILITA TODOS LOS MOTORES
   when "001" =>salidas<= "00000000";     --DESHABILITA TODOS LOS MOTORES
   when "010" =>salidas<= "00000001";     --HABILITA EL MOTOR X
   when "011" =>salidas<= "00000010";     --HABILITA EL MOTOR Y
   when "100" =>salidas<= "01101000";     --ARRANCA TALADRO Y PASO 1 DE Z
   when "101" =>salidas<= "01100100";     --ARRANCA TALADRO Y PASO 2 DE Z
   when "110" =>salidas<= "01010100";     --ARRANCA TALADRO Y PASO 3 DE Z
   when others =>salidas<="01011000";     --ARRANCA TALADRO Y PASO 4 DE Z
  end case;
 end process;
end arq_deco_zcc;
```

Fig. 2.31 Nuestra codificación en VHDL se reduce a tres bloques: 1) Uso de librerías, 2) Definición de variables de e/s y 3) Reglas de comportamiento

En el primer bloque se incluyen las librerías necesarias para el funcionamiento de las instrucciones básicas. En el segundo se definen el total de variables de entrada y salida. Y en el último se escribe la manera en la que responderá el circuito según la entrada que reciba.

La escritura y compilación del programa se hará sobre el compilador de lenguaje VHDL: "Galaxy".

Antes de compilar el programa, debemos de especificar el dispositivo sobre el que se generará el archivo de mapa de fusibles, en este caso será: "PALCE20V8-25PC".

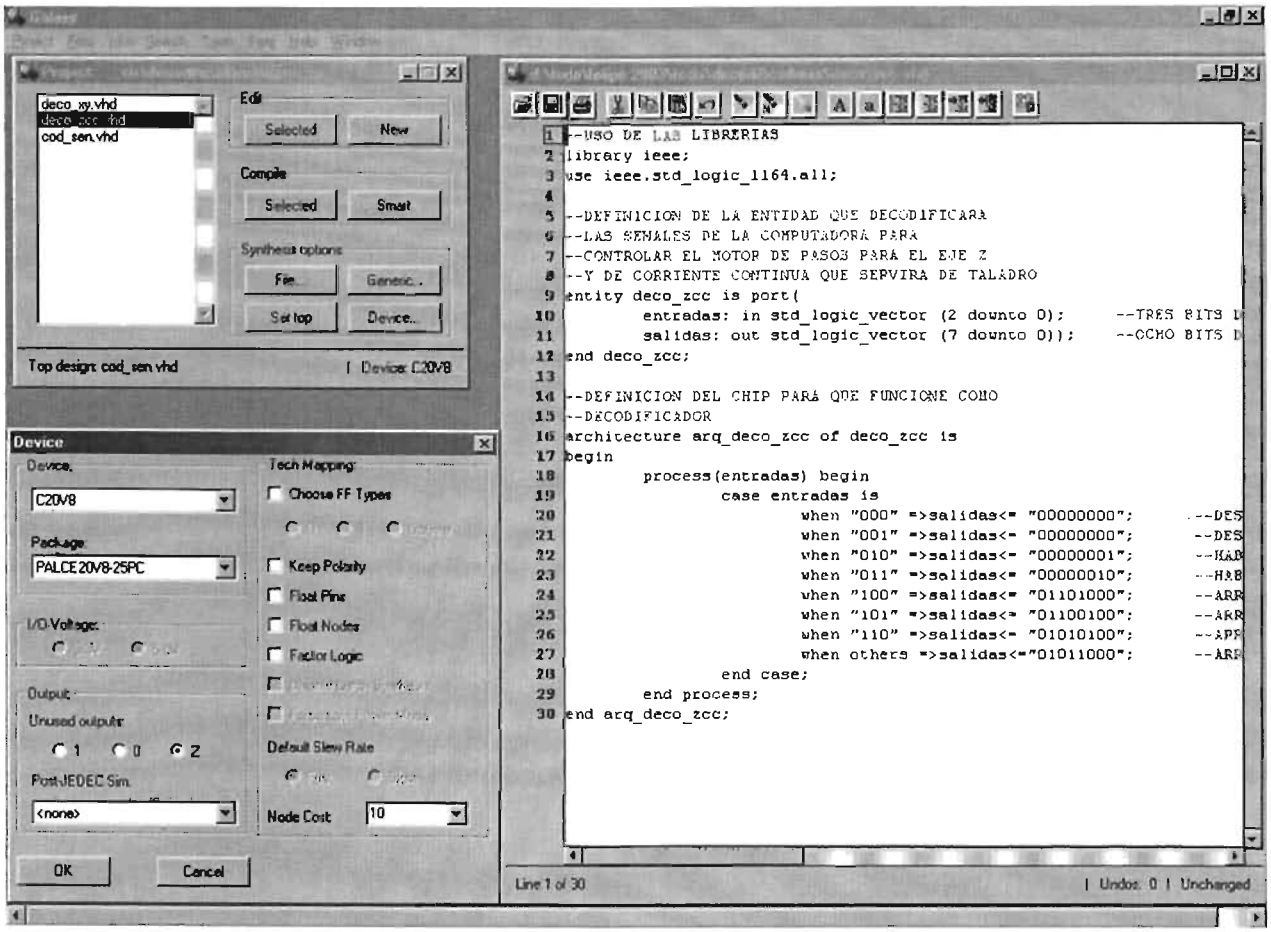


Fig. 2.32. Pantalla del Galaxy para la edición y compilación del programa decodificador

Después de compilar, el Galaxy genera un archivo de texto llamado igual que nuestro archivo pero con extensión .rpt, en el que se incluye la asignación de variables a los pines, lo que permite saber como conectar el circuito. En la figura 2.33 se muestra sólo el fragmento del .rpt que nos interesa:

```
    | | | | | | |
  ------------------
-|                  |-
-|                  |-
-|                  |-
-|    CYPRESS       |-
-|                  |-
-|                  |-   Warp VHDL Synthesis Compiler: Version 4 IR x95
-|                  |-   Copyright (C) 1991, 1992, 1993,
  |__________________|     1994, 1995, 1996, 1997, 1998 Cypress Semiconductor
    | | | | | | |

========================================================================
Compiling:  deco_zcc.vhd
Options:     -q -yv2 -e10 -w100 -o2 -ygs -fP -v10 -dc20v8 -pPALCE20V8-25PC -a
deco_zcc.vhd
========================================================================

                                C20V8A
                    ---------------------------------------
entradas_0 =|  1|                                    |24|* not used
entradas_1 =|  2|                                    |23|* not used
entradas_2 =|  3|                                    |22|= salidas_2
  not used *|  4|                                    |21|= salidas_7
  not used *|  5|                                    |20|= salidas_6
  not used *|  6|                                    |19|= salidas_5
  not used *|  7|                                    |18|= salidas_4
  not used *|  8|                                    |17|= salidas_1
  not used *|  9|                                    |16|= salidas_0
  not used *|10|                                     |15|= salidas_3
  not used *|11|                                     |14|* not used
  not used *|12|                                     |13|* not used
                    ---------------------------------------
```

Fig. 2.33. Las líneas provenientes del puerto paralelo se introducirán en los pines 1, 2 y 3, mientras que las salidas hacia los controladores ocuparán los espacios del 15 al 22

Otro archivo que genera el Galaxy es el correspondiente al mapa de fusibles (.jed), el cual sirve para grabar sobre el chip lo programado.

El archivo .jed se deberá abrir sobre un programa que comande un programador universal, como el "SUPERPRO" que se incluye en la figura 2.34:

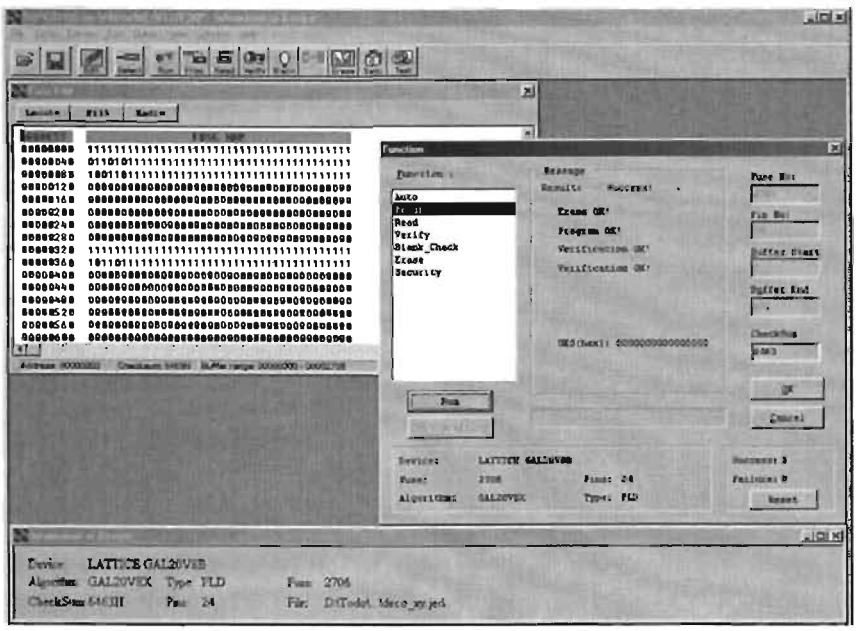


Fig. 2.34. El archivo mapa de fusibles (FUSE MAP) permitirá al SUPERPRO grabar el decodificador

Una vez abierto el .jed, debemos especificarle al SUPERPRO que grabaremos una GAL20V8B:

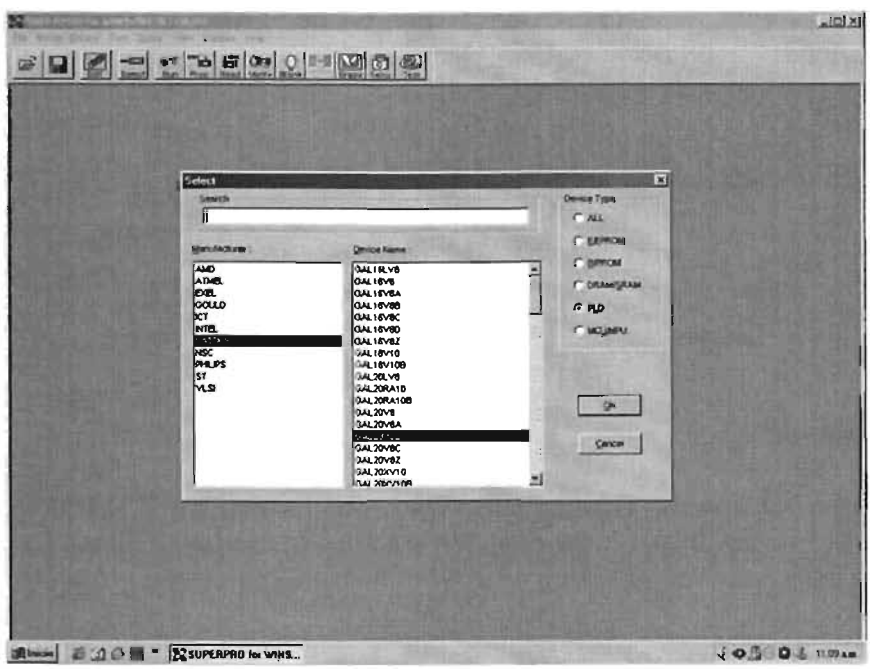


Fig. 2.35. De la lista de PLD's de Lattice, elegimos el dispositivo GAL20V8B

Finalmente programamos físicamente el circuito en el programador universal SUPERPRO/Z.

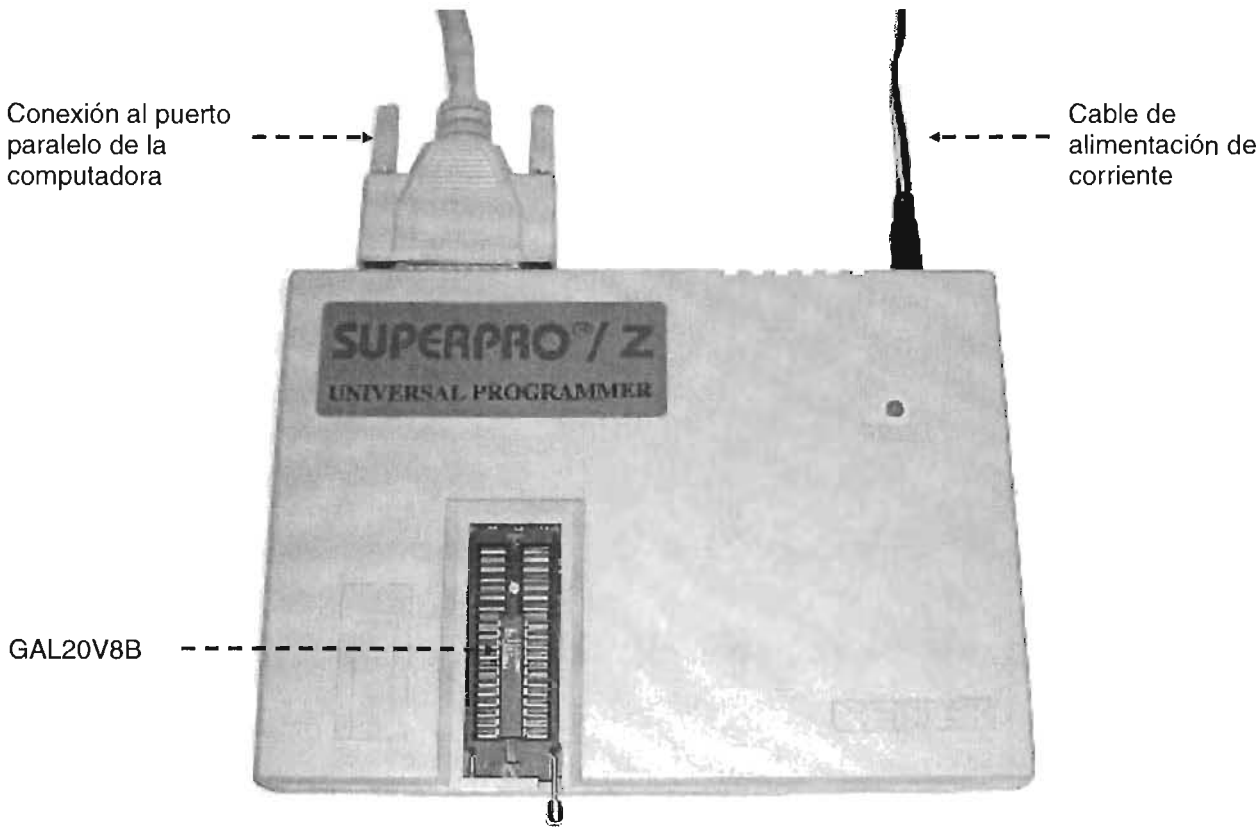


Fig. 2.36. Montamos la GAL20V8B sobre el grabador de memorias SUPERPRO/Z y seleccionamos en el software la opción de programar

El proceso anterior se deberá repetir para el decodificador del eje Y y X y codificador de sensores, por lo que en las hojas siguientes sólo incluiremos las tablas de operación, programa VHDL y archivo .rpt de los mismos.

	ENTRADAS			SALIDAS							
	Datos			Motor X				Motor Y			
No.	D_5	D_4	D_3	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	0	0	0	0	0	0	0	1	0	1	0
1	0	0	1	0	0	0	0	1	0	0	1
2	0	1	0	0	0	0	0	0	1	0	1
3	0	1	1	0	0	0	0	0	1	1	0
4	1	0	0	1	0	1	0	0	0	0	0
5	1	0	1	1	0	0	1	0	0	0	0
6	1	1	0	0	1	0	1	0	0	0	0
7	1	1	1	0	1	1	0	0	0	0	0

Tabla 2.13. Decodificación para los motores X y Y que controlan la mesa de coordenadas

```
--UTILIZACIÓN DE LAS LIBRERIAS
library ieee;
use ieee.std_logic_1164.all;

--DEFINICIÓN DE LA ENTIDAD DECODIFICARA PARA LOS EJES X Y Y
entity deco_xy is port(
 entradas: in std_logic_vector (2 downto 0);--3 BITS DEL LPT
 salidas: out std_logic_vector(7 downto 0));--8 BITS PARA MOVER MOTORES
end deco_xy;                                        --  DEL EJE X y Y

--DEFINICION DEL COMPORTAMIENTO DEL DECODIFICADOR
architecture arq_deco_xy of deco_xy is
begin
      process(entradas) begin
            case entradas is
                  --GIRO DEL MOTOR QUE CONTROLA EL EJE Y
                  when  "000" =>salidas<= "00001010"; --PASO 1
                  when  "001" =>salidas<= "00001001"; --PASO 2
                  when  "010" =>salidas<= "00000101"; --PASO 3
                  when  "011" =>salidas<= "00000110"; --PASO 4

                  --GIRO DEL MOTOR QUE CONTROLA EL EJE X
                  when  "100" =>salidas<= "10100000"; --PASO 1
                  when  "101" =>salidas<= "10010000"; --PASO 2
                  when  "110" =>salidas<= "01010000"; --PASO 3
                  when others =>salidas<= "01100000"; --PASO 4
            end case;
      end process;
end arq_deco_xy;
```

Fig. 2.37 Programa VHDL de los motores Y y X

```
                              C20V8A

                      ________________________________
entradas_0 =| 1|                                |24|* not used
entradas_1 =| 2|                                |23|* not used
entradas_2 =| 3|                                |22|= salidas_0
  not used *| 4|                                |21|= salidas_7
  not used *| 5|                                |20|= salidas_6
  not used *| 6|                                |19|= salidas_3
  not used *| 7|                                |18|= salidas_2
  not used *| 8|                                |17|= salidas_5
  not used *| 9|                                |16|= salidas_4
  not used *|10|                                |15|= salidas_1
  not used *|11|                                |14|* not used
  not used *|12|                                |13|* not used
                      ________________________________
```

Fig. 2.38 Archivo .rpt de los motores Y y X

Ahora sólo nos resta ver la codificación[15] de los sensores:

	Entradas			Salidas	
	Sensores			Datos puerto LPT	
No.	Z	Y	X	D_7	D_6
0	0	0	0	0	0
1	0	0	1	0	1
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	1
5	1	0	1	0	1
6	1	1	0	1	0
7	1	1	1	0	1

Tabla 2.14. Codificación de los tres sensores que detectan la posición de los ejes

[15] En los decodificadores las líneas de entrada son menores que las de salida, mientras que en los codificadores es a la inversa

```
--UTILIZACIÓN DE LAS LIBRERIAS
library ieee;
use ieee.std_logic_1164.all;

--DEFINICION DE LA ENTIDAD QUE CODIFICARA LAS SENALES DE LOS SENSORES
entity cod_sen is port(
  entradas: in std_logic_vector (2 downto 0);  --3 BITS PARA SENSORES
  salidas: out std_logic_vector (1 downto 0)); --2 BITS PARA EL LPT
end cod_sen;

--DEFINICION DEL COMPORTAMIENTO DEL CODIFICADOR DE SENSORES
architecture arq_cod_sen of cod_sen is
begin
process(entradas) begin
  case entradas is
    when "000" =>salidas<= "00";          --TODOS LOS EJES EN POSICION
    when "001" =>salidas<= "01";          --EJE X FUERA DE POSICIÓN
    when "010" =>salidas<= "10";          --EJE Y FUERA DE POSICIÓN
    when "011" =>salidas<= "01";          --EJE X FUERA DE POSICIÓN
    when "100" =>salidas<= "11";          --EJE Z FUERA DE POSICIÓN
    when "101" =>salidas<= "01";          --EJE X FUERA DE POSICIÓN
    when "110" =>salidas<= "10";          --EJE Y FUERA DE POSICIÓN
    when others =>salidas<="01";          --EJE X FUERA DE POSICIÓN
  end case;
end process;
end arq_cod_sen;
```

Fig. 2.39 Programa VHDL de los sensores

```
                           C20V8A
                 -----------------------------------
entradas_0 =| 1|                                 |24|* not used
entradas_1 =| 2|                                 |23|* not used
entradas_2 =| 3|                                 |22|= salidas_0
  not used *| 4|                                 |21|* not used
  not used *| 5|                                 |20|* not used
  not used *| 6|                                 |19|* not used
  not used *| 7|                                 |18|* not used
  not used *| 8|                                 |17|* not used
  not used *| 9|                                 |16|* not used
  not used *|10|                                 |15|= salidas_1
  not used *|11|                                 |14|* not used
  not used *|12|                                 |13|* not used
                 -----------------------------------
```

Fig. 2.40 Archivo .rpt de los sensores en los que se puede ver que tres entradas se convierten en dos

2.5.6 OPTOACOPLADORES

Los puertos de una computadora manejan voltajes y corrientes pequeñas, y un simple corto circuito puede inutilizarlos.

Para evitar la entrada de corrientes y voltajes no deseados; utilizamos dentro de la circuitería unos chips llamados: "optoacopladores" , cuya función principal es aislar las corrientes de entrada de los sensores.

En la imagen de abajo se muestra este tipo de dispositivo y su conexión correspondiente:

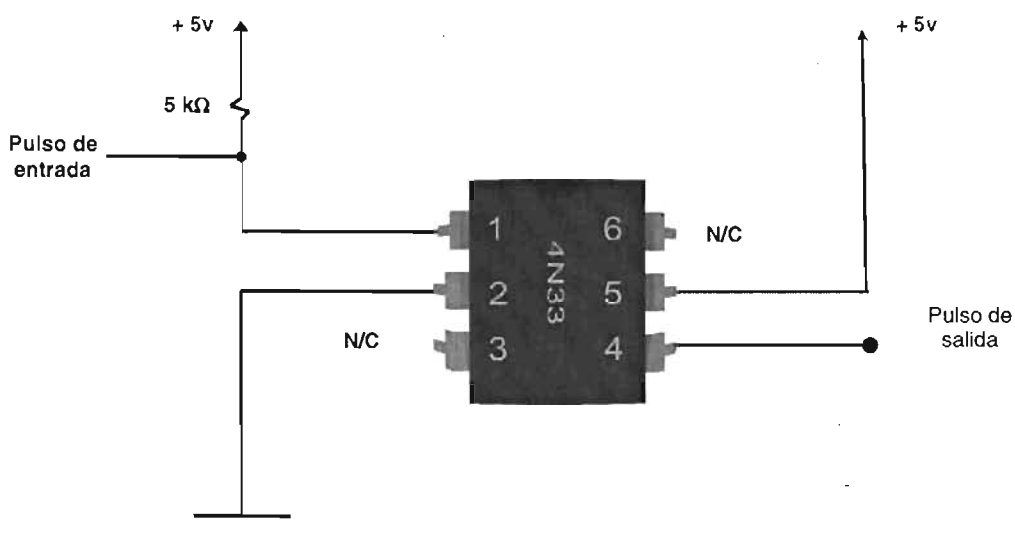


Fig. 2.41 Los optoacopladores protegen el puerto de la computadora

Estos dispositivos son baratos y fáciles de conseguir, y ocuparemos uno para cada sensor.

2.5.7 SENSORES

Al igual que las impresoras, que cuando se encienden realizan rutinas de reinicialización, nuestro modelador deberá de hacer lo mismo para asegurar que siempre que se comience un nuevo modelo físico no existan desajustes de posición.

En el hardware existirán tres sensores infrarrojos (compuestos por un emisor y un receptor), uno para cada eje coordenado que se bloqueará o desbloqueará con la ayuda de pequeñas laminitas posicionadas en las charolas. Con ellos podremos registrar desde la computadora cual de los ejes es el que debemos de regresar a su posición inicial.

La configuración del sensor que ocuparemos es la siguiente:

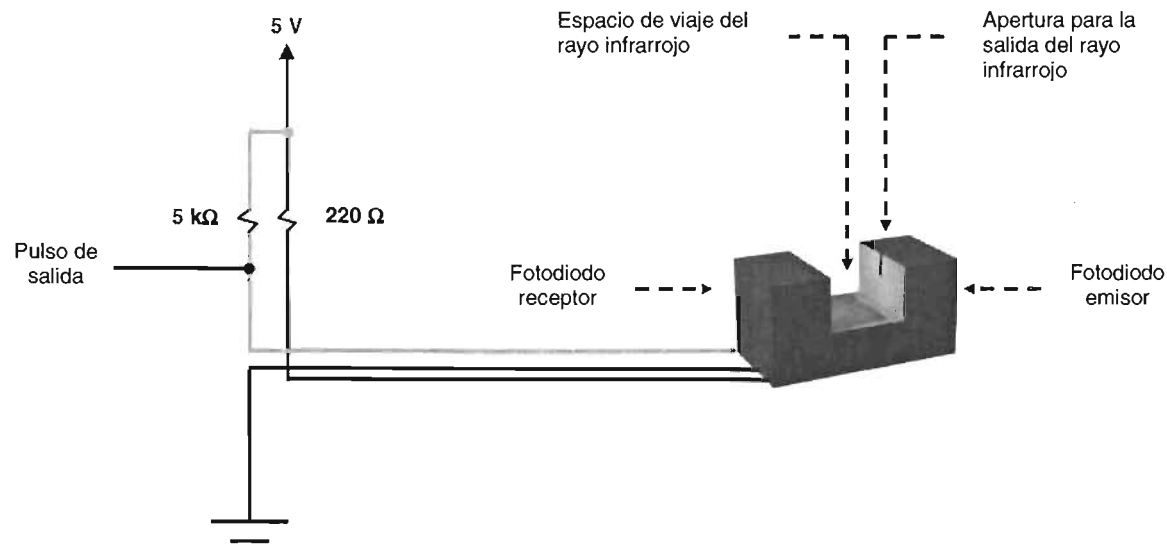


Fig. 2.42. Al bloquearse el rayo infrarrojo con la lámina que hay debajo de las charolas el pulso de salida cambia de bajo a alto

Los sensores de este género son fáciles de extraer de impresoras de inyección de tinta y ratones.

2.5.8 FUENTE DE PODER

Para que nuestro circuito electrónico funcione, necesitaremos de voltajes en el rango de 5 - 36 V de corriente continua, para lo cual utilizaremos transformadores reductores o fuentes de poder que conviertan la energía eléctrica de nuestros hogares cuyo voltaje oscila entre los 110 y 127 V de corriente alterna C.A. y los 60 Hz.

La fuente de poder también la obtendremos de de impresoras en desuso y su forma es la siguiente:

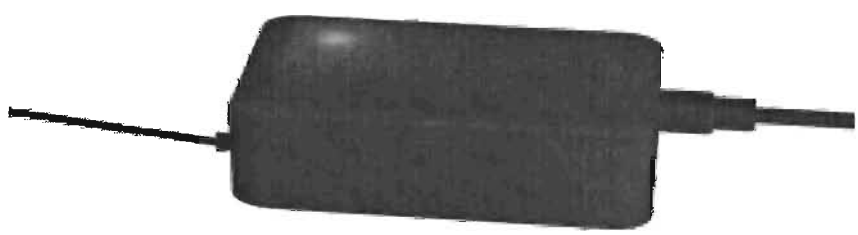


Fig. 2.43 La fuente de poder reduce la energía a nuestras necesidades y la transforma de alterna a continua

En las páginas siguientes mostraremos el diagrama del circuito electrónico resultante junto con las fotografías reales del mismo.

2.5.9 DIAGRAMA DEL CIRCUITO ELECTRÓNICO

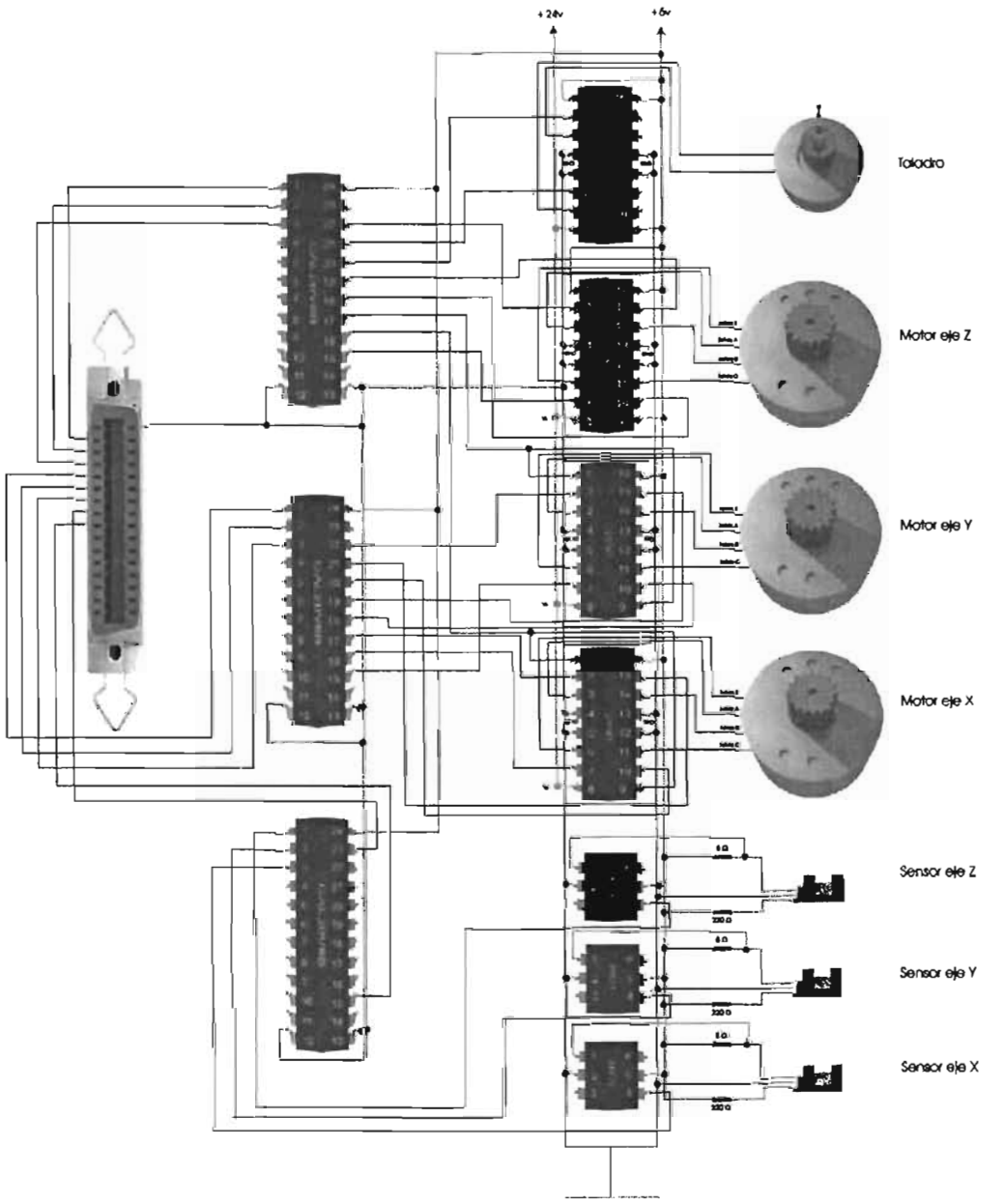


Fig. 2.44 Diagrama electrónico

2.5.10 FOTOGRAFIAS REALES DEL CIRCUITO ELECTRÓNICO

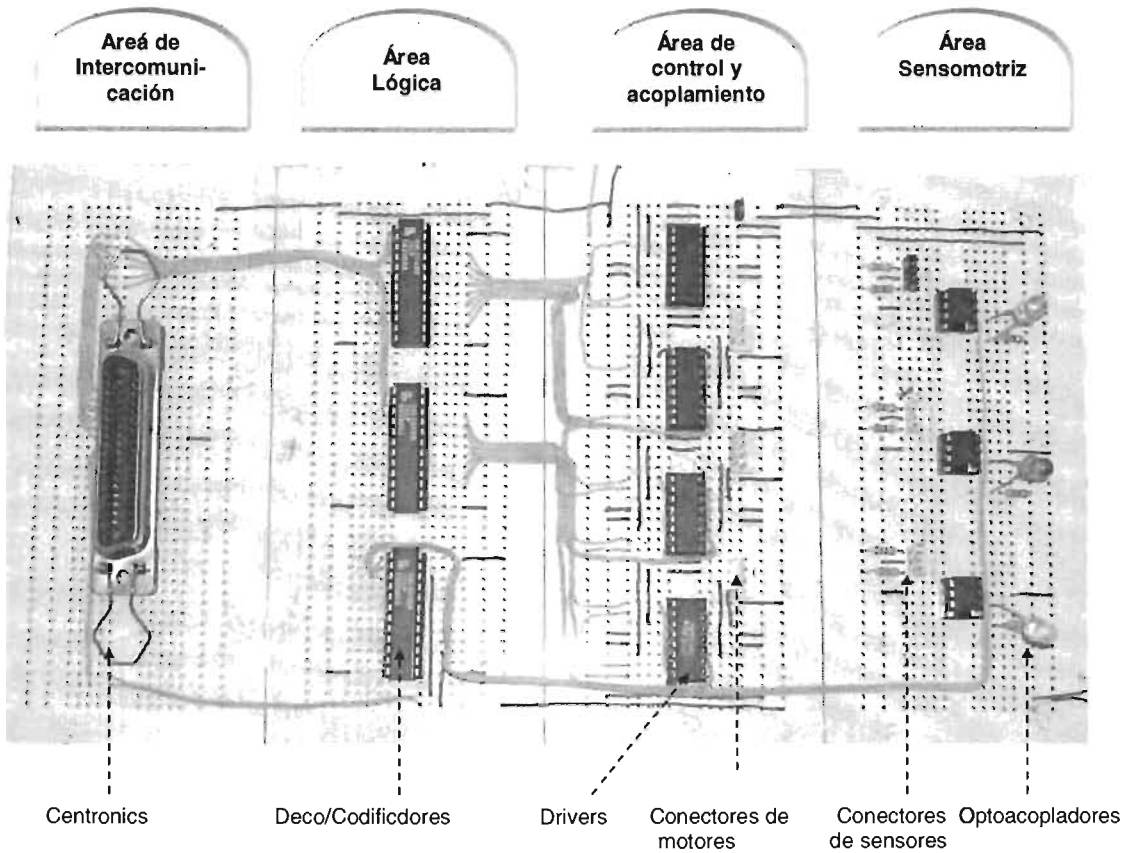


Fig. 2.45 Fotografía real del circuito electrónico

Este circuito se monta en la parte trasera del modelador, cuyas fotografías reales se muestran en las páginas siguientes:

2.5.11 FOTOGRAFÍAS REALES DEL MODELADOR

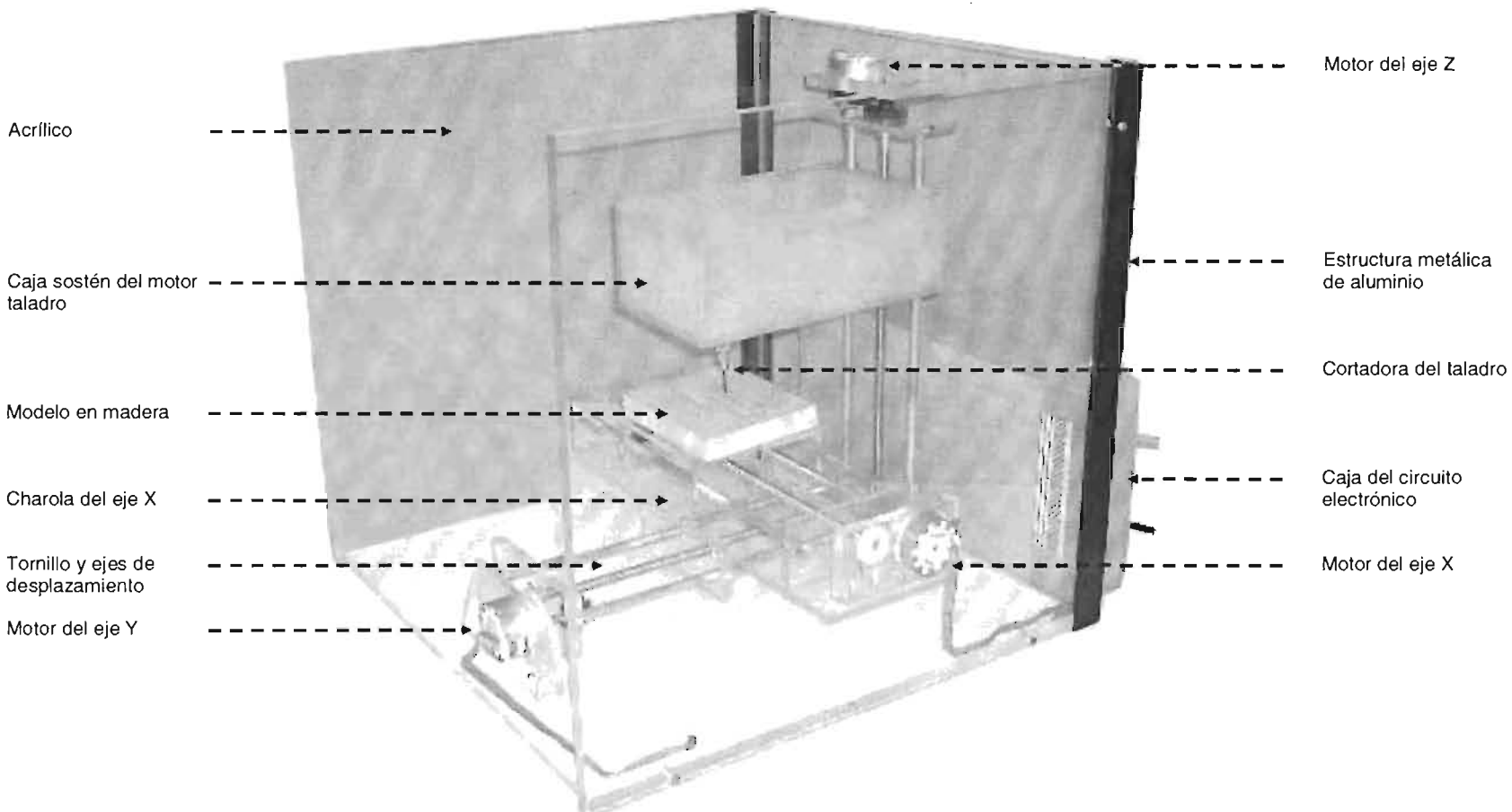


Fig. 2.46. Vista tridimensional frontal del hardware en la que se pueden observar los tres ejes coordenados (X,Y,Z)

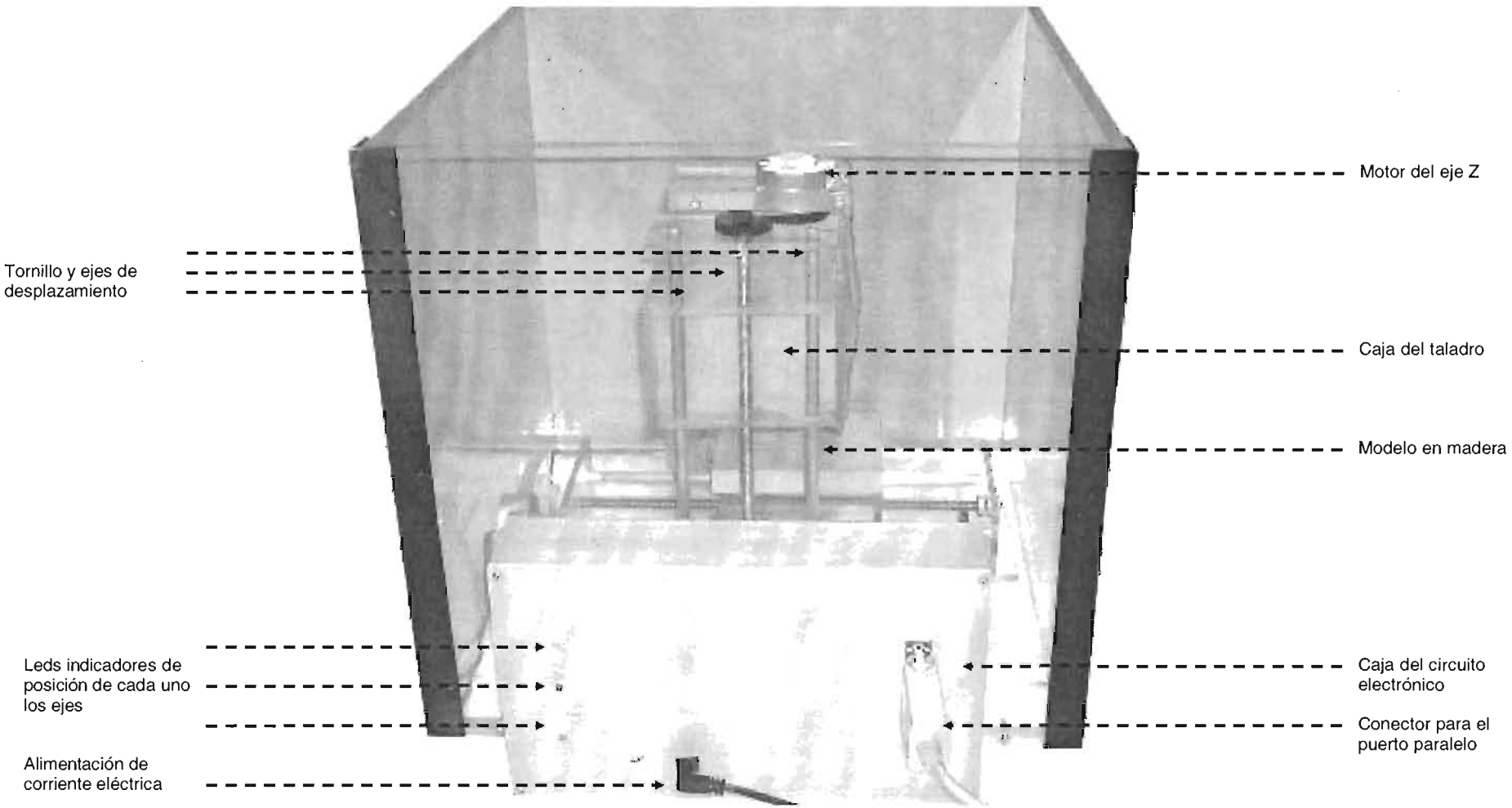


Fig. 2.47. Vista tridimensional de la parte posterior del hardware

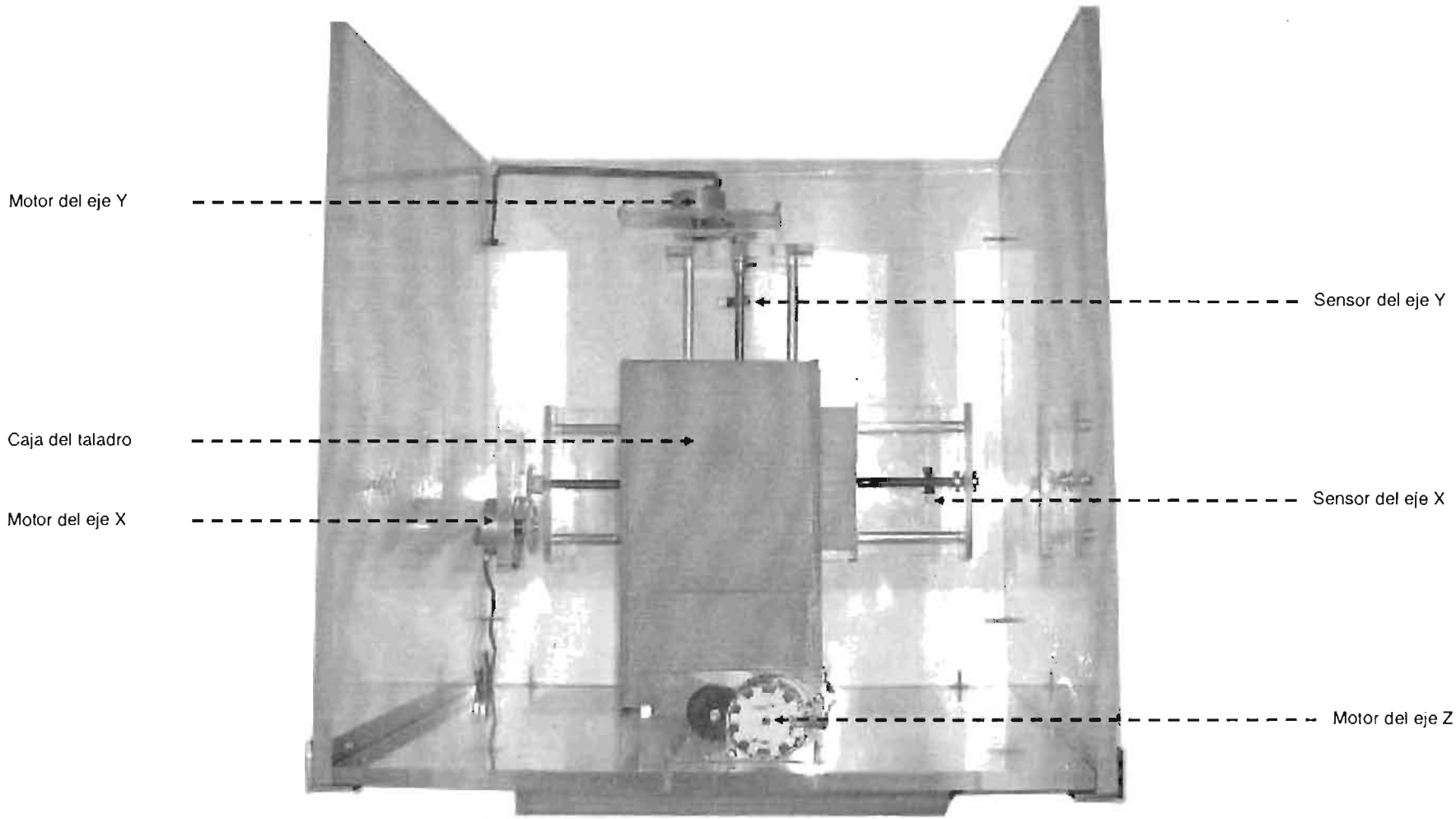


Fig. 2.48. Vista superior del harware

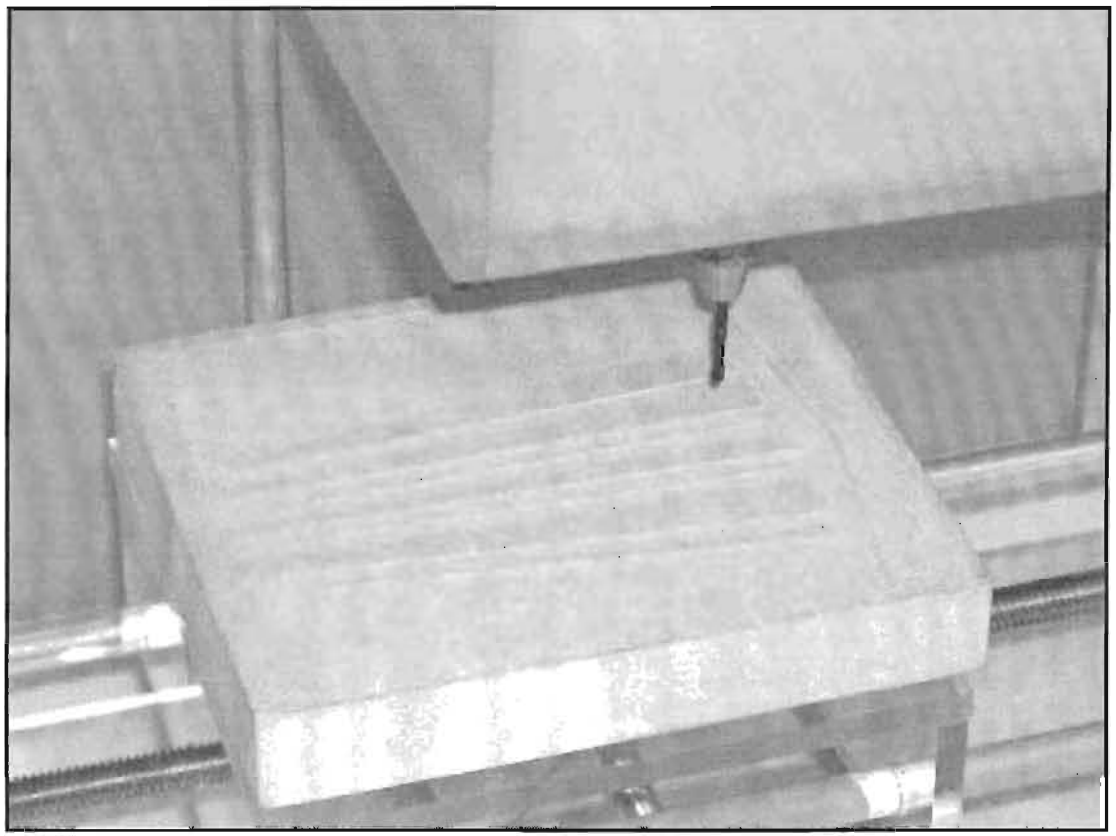


Fig. 2.49. Proceso de modelado de las torres de la ENEP "Aragón"

CAPÍTULO III

DISEÑO E IMPLEMENTACIÓN DEL SOFTWARE

De nada nos serviría tener la máquina modeladora si no contaramos con un mecanismo de control para comandarla.

El software o conjunto de programas de computadora, nos ayudarán a mover el hardware para esculpir los modelos en madera.

Para facilitar la comprensión del proceso de modelación a través de la interacción usuario-computadora-software-hardware, elaboramos el esquema siguiente:

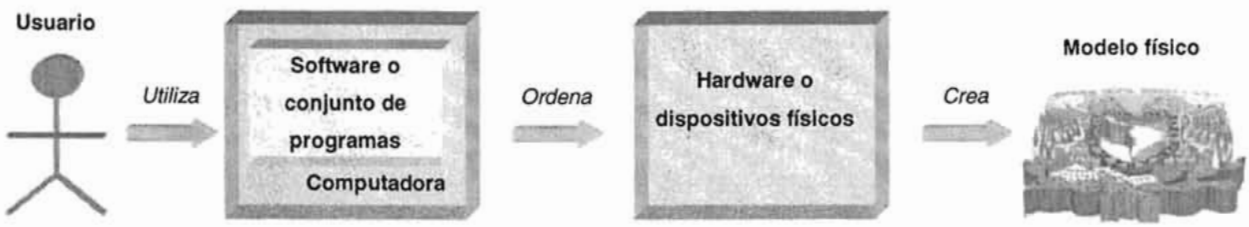


Fig. 3.1 Proceso general de elaboración de un modelo físico

Como podemos observar en el esquema anterior, un usuario utiliza una computadora y el software respectivo para manipular el modelador físico, el cual a su vez esculpe un bloque de madera.

A lo largo de este capítulo seguiremos el proceso de creación del software, basándonos en ciertos pasos enmarcados por una buena metodología de ingeniería de software: "El Proceso Unificado"

3.1 DEFINICIÓN DE REQUERIMIENTOS

El software que se desea desarrollar, brindará a cualquier usuario la capacidad de visualizar en la pantalla de un monitor imágenes planas capturadas previamente, transformarlas con efectos especiales, y generar el modelo tridimensional equivalente (campo de altura). Además de lo anterior, el programa de computadora brindará la posibilidad de comandar el hardware modelador para crear en un bloque de madera el campo de alturas correspondiente.

3.1.1 REQUERIMIENTOS FUNCIONALES

Las principales características de funcionalidad que necesitamos que cubra nuestro software, son las siguientes:

- Ingresar al sistema
- Abrir archivo de imagen
  - JPG
  - GIF
  - BMP
- Aplicar efectos
  - Brillo
  - Contraste
  - Filtros
    - Pasa Baja
    - Pasa alta
    - Laplaciano
    - Sobel
      - Horizontal
      - Vertical
      - Diagonal normal
      - Diagonal invertida

- Ajustable

➢ Configurar escena tridimensional

- Posición de la cámara
- Ángulo de visión
- Objeto de enfoque
- Posición de la fuente de luz
- Color de la fuente de luz
- Color de fondo

➢ Crear campo de altura visual a partir de:

- Imagen original
- Imagen transformada

➢ Crear modelo físico a partir de:

- Imagen original
- Imagen transformada

➢ Salir del sistema

Con lo que respecta a los actores, el único involucrado en el uso del sistema será:

➢ Usuario

A partir de los anteriores requisitos funcionales, crearemos el diagrama general de casos de uso del sistema para ofrecer una mejor panorámica de las actividades propias de nuestro software:

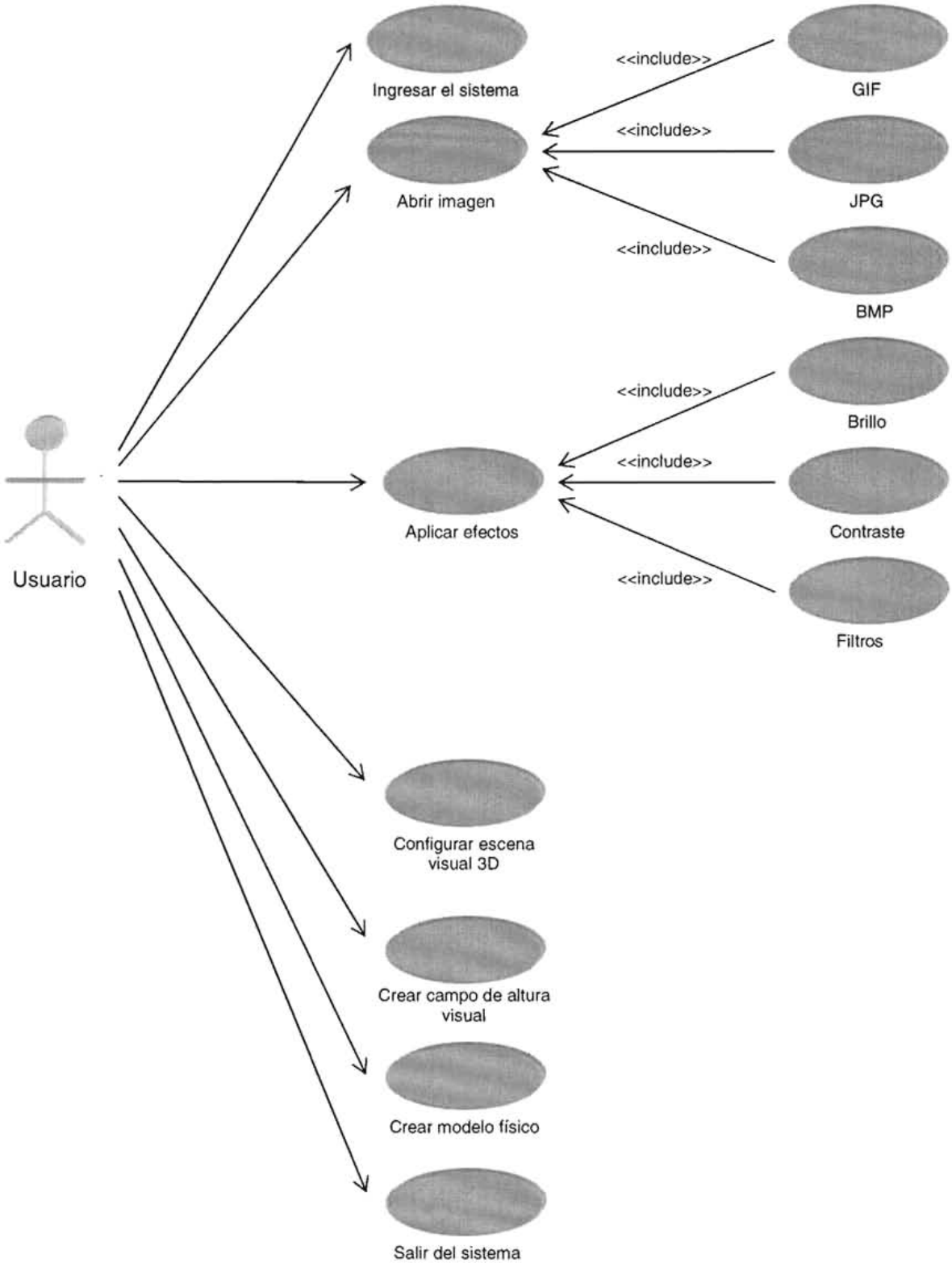


Fig. 3.2 Primera parte del diagrama de casos de la funcionalidad del software

Continuación:

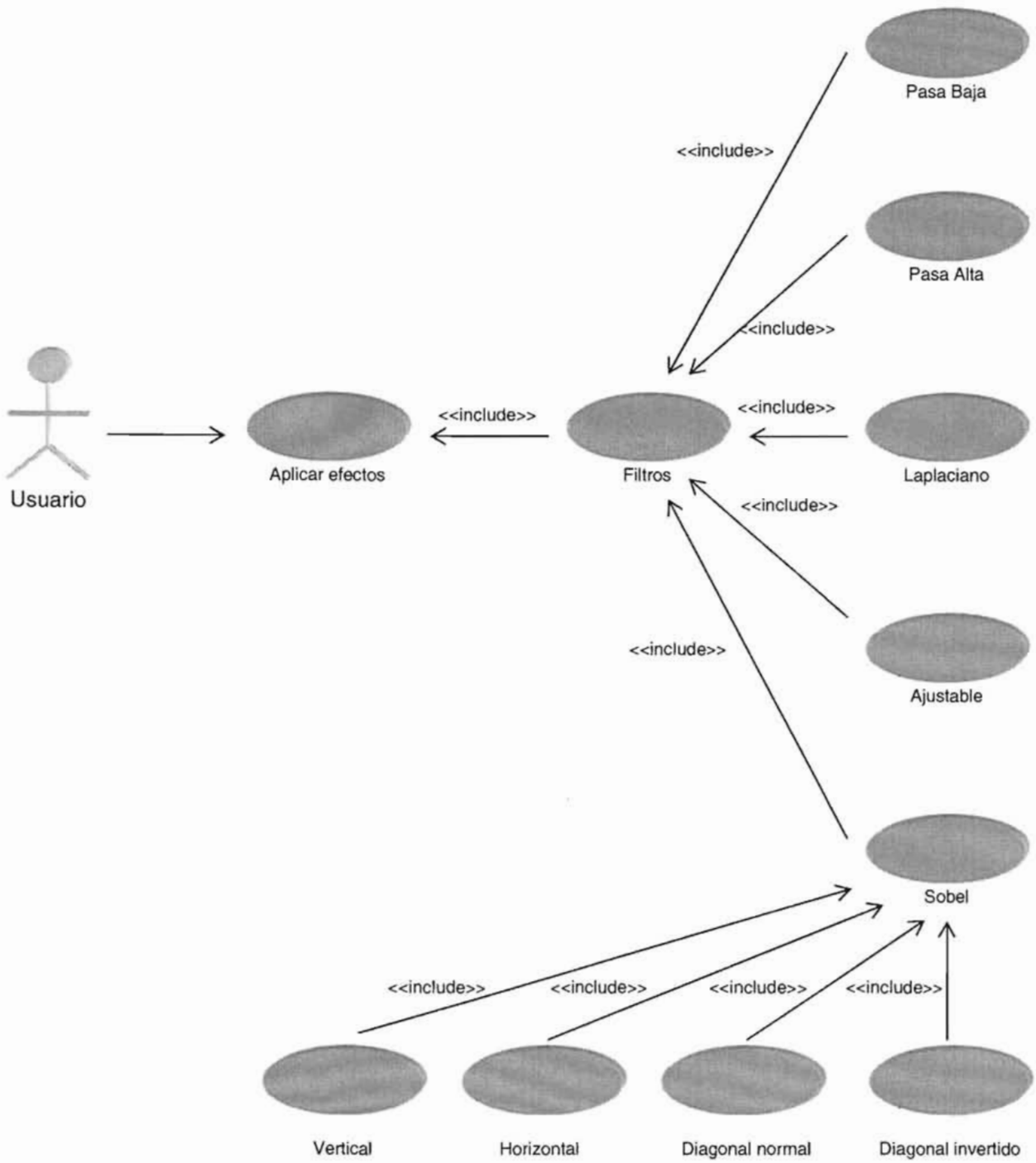


Fig. 3.2 Continuación del diagrama de casos de uso

Ahora que ya hemos definido los requerimientos funcionales del sistema, listaremos los no funcionales.

3.1.2 REQUERIMIENTOS NO FUNCIONALES

- Look and feel:
  - Funcionará de manera óptima con tarjetas de video y monitores con resolución mínima de 600x800 pixeles y 16 millones de colores.
  - El entorno visual deberá ser simple.

- Usabilidad:
  - El manejo de pantallas deberá ser lo más sencillo y útil.
  - Cualquier persona con conocimientos mínimos en computación podrá aprender su funcionamiento y operarlo de manera rápida.
  - Deberá de reducir las posibilidades de error.

- Desempeño:
  - Todos los resultados tendrán que ser lo más exactos posibles.
  - El sistema sólo soportará un usuario a la vez
  - Los archivos de imágenes tendrán 320x240 pixeles de resolución como máximo.

- Operacionales:
  - Hardware:
    - Computadora PC con las características mínimas siguientes:
      - Procesador intel o AMD con velocidad mínima de 233MHz
      - 32Mb en memoria RAM
      - Espacio libre en Disco Duro de 300 MB.

  - Software:
    - Cualquiera de los siguientes sistemas operativos:

- Windows 95
- Windows 98
- Windows Me
- Windows NT
- Windows 2000
- Windows XP

➢ Herramientas:

- Visual Studio 6 ó Visual Studio .net
- Componentes OCX para el uso de librerías OpenGL en Visual Basic
- Librería de enlace dinámico (dll) para manejo del puerto paralelo.

➢ Portabilidad:

- El sistema sólo podrá ser accesado de manera local y bajo los sistemas operativos anteriormente señalados.

➢ Seguridad:

- No habrá restricciones de acceso

➢ Peticiones culturales y políticas:

- El idioma del sistema será el español.

Ahora que hemos definido todos los requerimientos, pasaremos a la etapa de análisis.

3.2 ANÁLISIS

Comenzaremos por decir que la arquitectura del software estará dada en tres capas únicamente:

1. *Interfaz Humana (IH):* estará constituida por el conjunto de ventanas a través de las cuales se comunicará el usuario con el sistema, en esta capa se encuentra la parte visual de interacción humano - computadora.

2. *Dominio del problema (DP):* estará definida por aquellas clases que resuelven el problema, en esta capa se encuentra la mayor parte de la programación, y se encarga de comunicar a la interfaz humana con los datos y viceversa.

3. *Modelo de datos (MD):* aunque en la realidad no tengamos una capa que incluya una base de datos como tal, diremos que nuestro soporte serán los datos que obtengamos de imágenes.

Echemos un vistazo a la representación visual de la arquitectura:

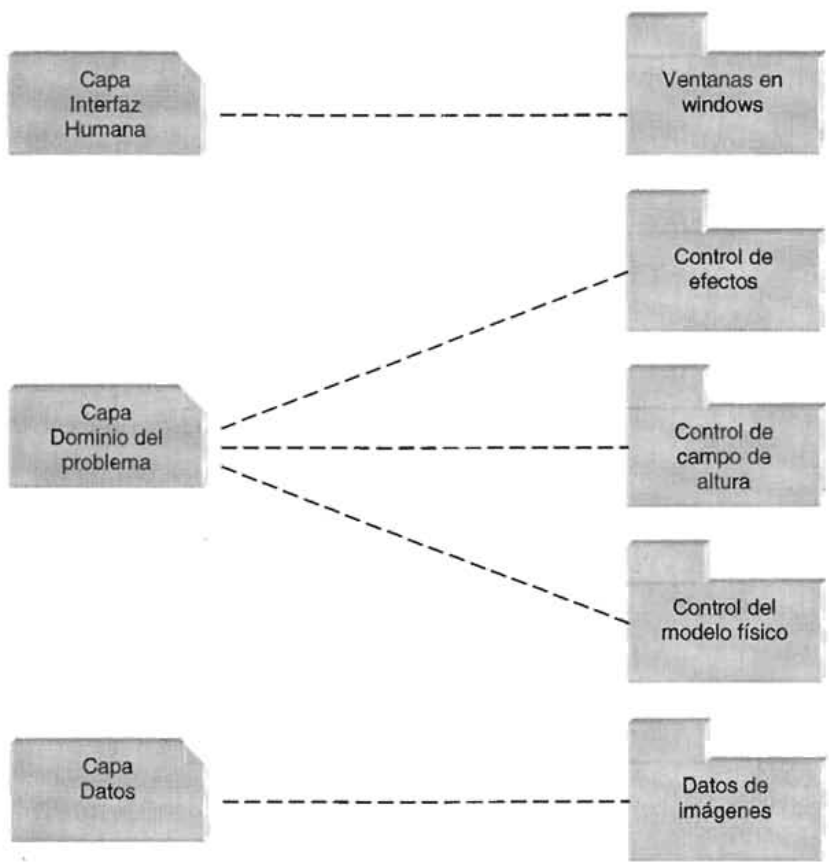


Fig. 3.3 La arquitectura del sistema estará compuesta de tres capas

Ahora definiremos el diagrama de clases correspondiente al dominio del problema:

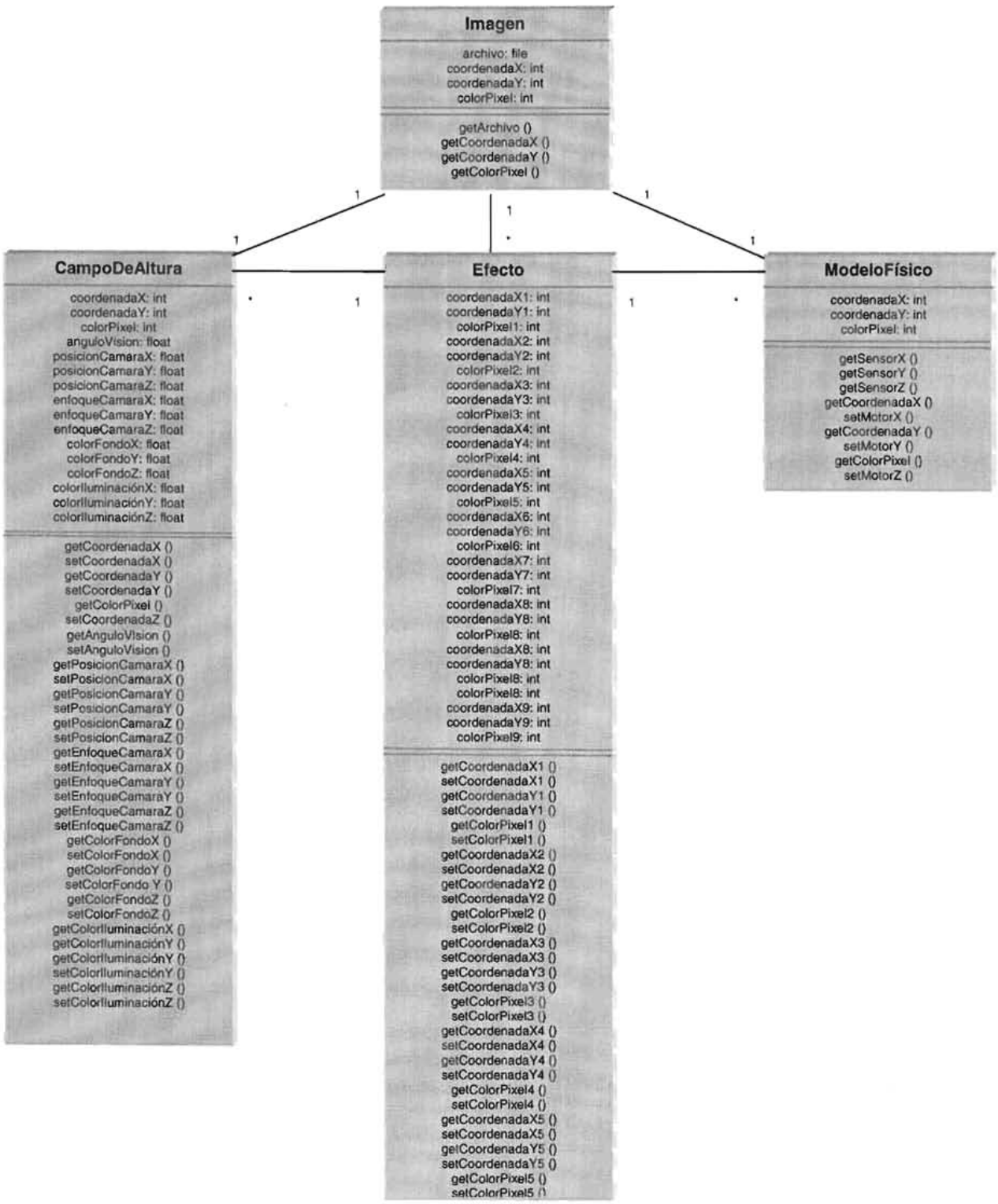


Fig. 3.4 a) Primera parte del diagrama de clases del software. A partir de una imagen se puede generar su campo de altura, aplicarle efectos y modelarla físicamente

Continuación:

Fig. 3.4 b) Continuación del diagrama de clases

3.3 DISEÑO

Para diseñar el software nos basaremos en un patrón conocido como: "capas" o "layers" en inglés. Al estructurarlo de esa manera aislaremos la interfaz, el dominio del problema y el modelo de datos.

Fig 3.5 El diseño en capas permite independencia entre ellas

Necesidades:

Para elaborar el software se requerirá de una computadora con las características mínimas siguientes:

- Hardware:
  - Procesador Intel o AMD con velocidad de procesamiento de 233 MHz.
  - 64 Mb en RAM.
  - Monitor color VGA de 14" y resolución mínima de 800x600
  - Tarjeta de vídeo de 2Mb.
  - Disco Duro de 2 Gb.

- Software:
  - Sistema Operativo Windows 98.
  - Visual Studio 6.0 o .NET
  - Librerías de OpenGL para VB6 de nombre: glxctl.ocx, glxctl.oca y vbogl.tlb
  - Librería de enlace dinámico: inpout.dll

Es tiempo de que revisemos el diagrama de estados referente a la navegación sobre la interfaz del software:

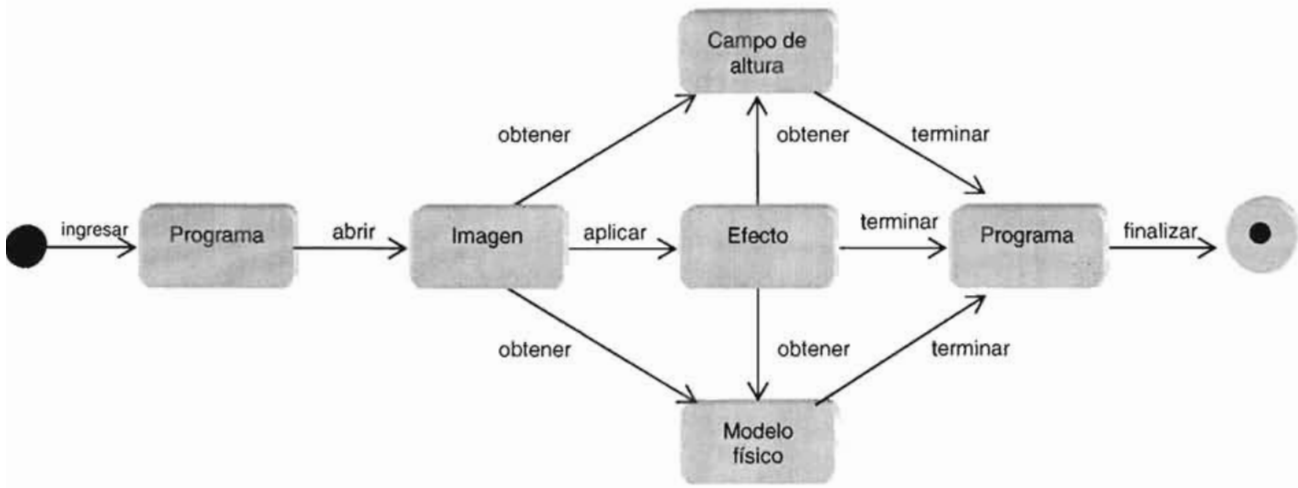


Fig 3.6 Diagrama de estados para observar la navegabilidad a través de la interfaz del sistema

Antes de continuar con otra fase, tendremos que aclarar dos aspectos fundamentales para la realización del campo de alturas de manera visual y para el modelo físico.

Para el campo de alturas, debemos de considerar que la representación será en tres dimensiones, y que para programarlo en Visual Basic (por simplicidad) será necesario utilizar las famosas librerías OpenGL (Open Graphic Library o Librería de Gráficos Abierta), la cual es una interfaz de aplicación de programa o API (Aplication Program Interface). Y para enviar datos al puerto paralelo ocuparemos una librería de enlace directo como: "inpout.dll".

3.3.1 OPENGL

Todo lo que podemos proyectar en la pantalla de un monitor en realidad es plano y para lograr el efecto óptico de tridimensionalidad se utilizan ciertos modelos matemáticos de iluminación y transformación cuya complejidad es relativamente alta.

Las librerías OpenGL[16] (Open Graphics Libraries) son un conjunto de códigos de computadora escritos para diferentes lenguajes y plataformas, cuyas funciones facilitan el trazado de imágenes en 2D y 3D (rendering).

Fig. 3.7 Logotipo de las librerías gráficas de OpenGL

Con las librerías OpenGL convertiremos a cada uno de los pixeles que leeremos de la imagen plana en una línea tridimensional, de tal manera que la ubicación en el espacio de dicha línea corresponderá a la posición en el plano del pixel y la altura será igual al valor del color.

[16] El sitio oficial de OpenGL es: http://www.opengl.org/

Para construir una imagen en tres dimensiones, debemos de considerar algunos conceptos indispensables como son: espacio tridimensional, cámara, fuente de luz, objeto y fondo principalmente, los cuales comprenderemos mediante la figura 3.8

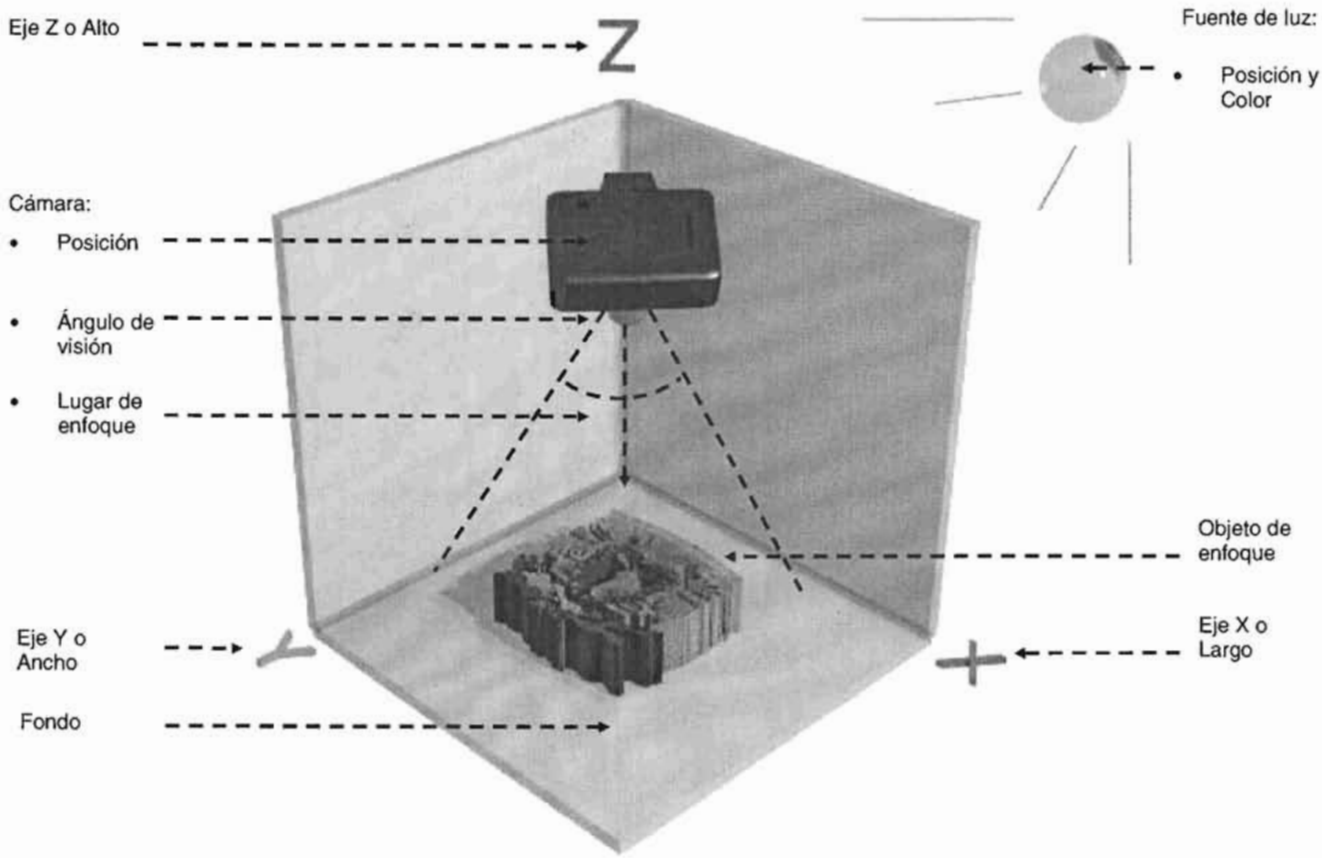


Fig. 3.8 Imagen tridimensional computarizado

Para crear una escena tridimensional (representada por la perspectiva generada con los ejes X, Y y Z), necesitamos definir el color del fondo y posicionar una fuente de luz (especificando su color) para iluminar a los objetos que se construyan (como el logo de la UNAM anterior). Y también debemos de especificar la posición de una cámara cuya función será capturar todo aquello que enfoque sobre el ángulo de visión deseado.

3.3.2 INPOUT.DLL

Visual Basic 6 carece de una función para enviar/recibir información desde el puerto paralelo y debido a que utilizaremos este lenguaje para la construcción del software, ocuparemos una librería de enlace dinámico (dll) descargable desde internet cuyo nombre es: inpout.dll[17]

3.4 CONSTRUCCIÓN

Esta fase abarca la elaboración del software y en ella incluiremos todas las ventanas de interacción del humano con la computadora, al mismo tiempo que mostraremos algunos segmentos de código[18].

La primer pantalla que aparece al momento de cargar el programa es:

Fig. 3.9 Pantalla de presentación

La ventana de arriba se despliega al arrancar el programa y permanece visible durante cinco segundos, después de esa aparece la principal que se muestra en la figura 3.10.

[17] El sitio de descarga de la librería inpout.dll es:
http://www.programmersheaven.com/zone1/cat247/122.htm
[18] El código completo del software se anexa en el CD.

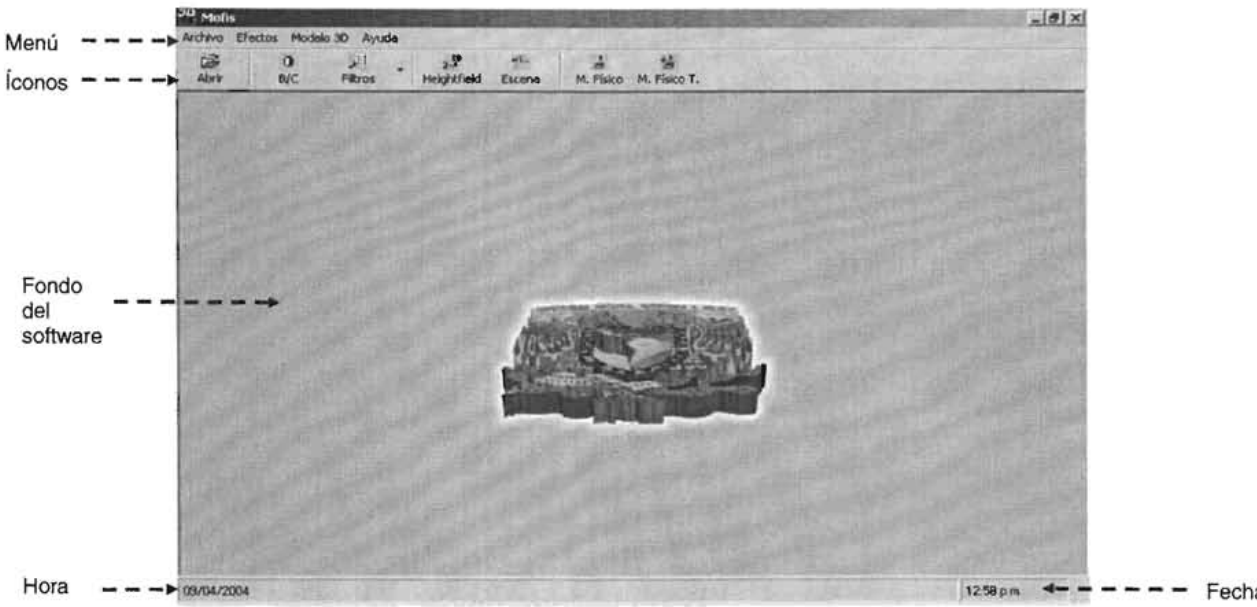


Fig. 3.10 Venta maestra a partir de la que se controlan todas las funciones del software

A través de la ventana principal se pueden realizar todas las tareas a las que nos habíamos comprometidos en la fase de requerimientos, es decir: abrir imágenes, aplicarles efectos y crear los modelos de altura visuales y físicos.

Revisemos cada una de las pantallas que componen nuestro sistema de cómputo, empezando por la de abrir (figura 3.11):

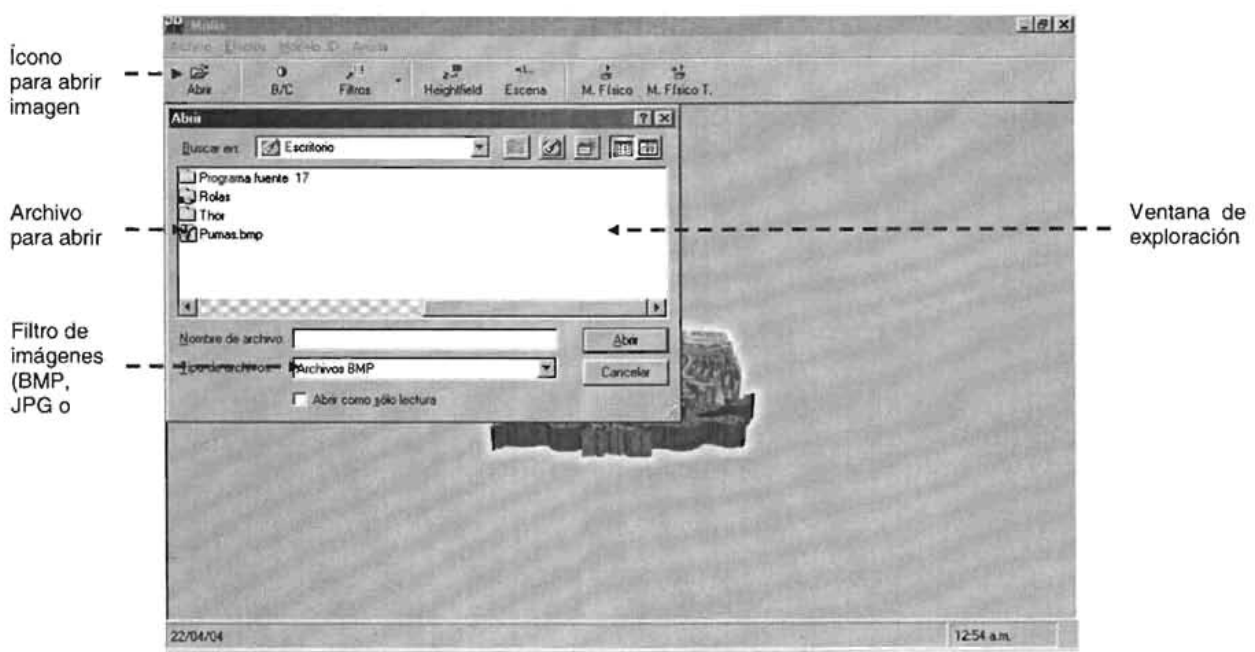


Fig. 3.11 La opción de abrir nos permite seleccionar el archivo de imagen (JPG, GIF o BMP)

La interfaz anterior como todas las que mostraremos más adelante, la podremos visualizar seleccionando dentro del menú la opción deseada o presionando el ícono equivalente.

Después de seleccionar un archivo de imagen, le aplicaremos los efectos de contraste y brillo de manera separada y con valores al 80% para reafirmar los conceptos mostrados en el primer capítulo:

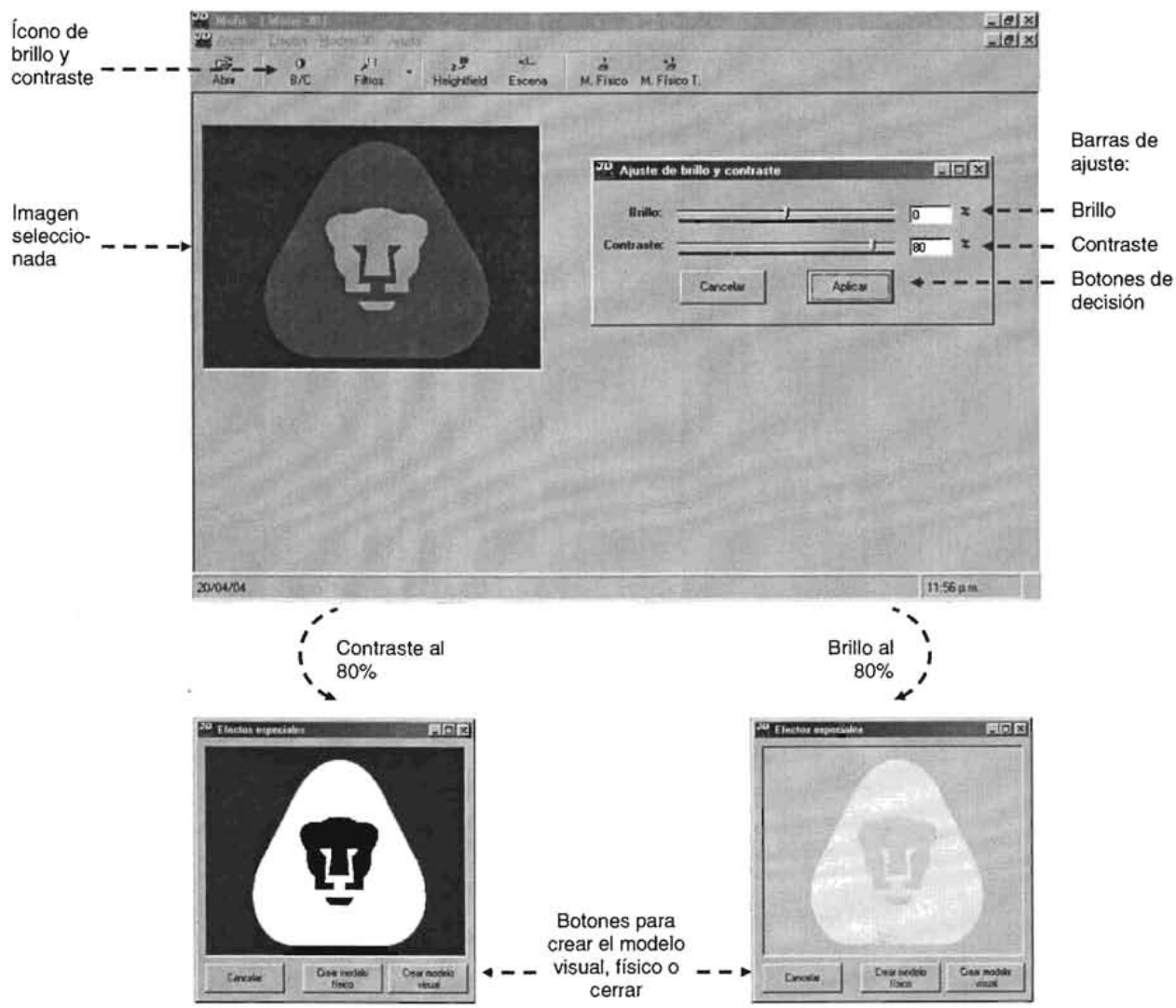


Fig. 3.12 Podemos alterar el brillo y el contraste de manera separada o conjunta.

Existen otros tipos de efectos que podemos aplicar para obtener resaltes,

definición de contornos, suavizados, etc. Dichos efectos ya los mencionamos anteriormente (filtros) y los utilizaremos uno a uno:

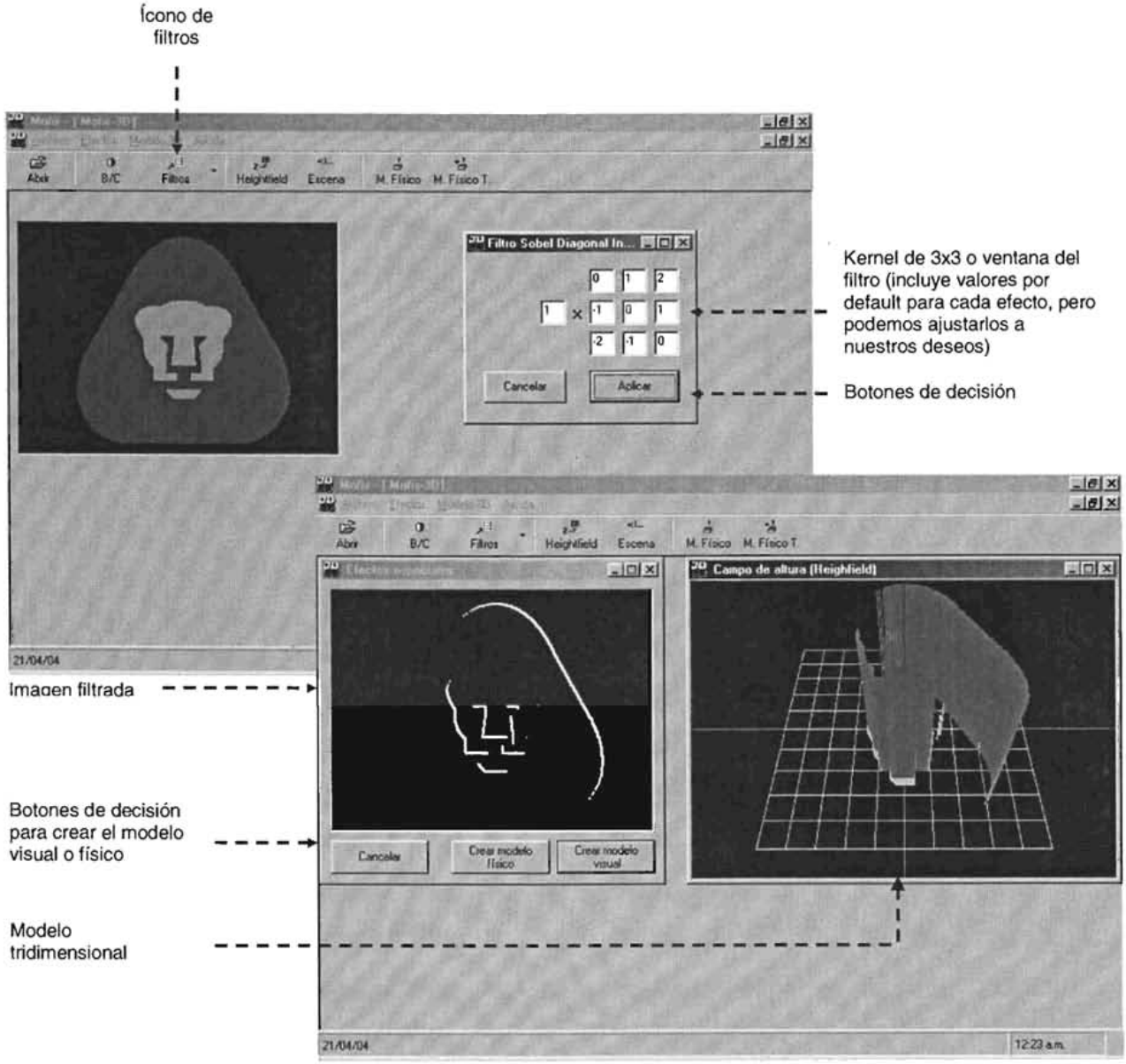


Fig. 3.13 Efecto especial del filtro Sobel Diagonal invertida y su generación en modelo visual tridimensional

Cuando seleccionamos la opción del filtro en el menú o en el ícono, se despliegan todos los posibles filtros a elegir. Una vez que aplicamos alguno de ellos, el sistema nos ofrece la capacidad de crear su equivalente en tres dimensiones de forma visual o física.

En esta parte incluiremos el código de operación del filtro dentro de la clase efecto y después ilustraremos los resultados de aplicar todos los demás:

```
'********** TRANSFORMACIÓN DE UNA IMAGEN CON UN FILTRO DE TRES POR TRES **********'

For j = 0 To altoImagen - 3
    For i = 0 To anchoImagen – 3

      frmProgreso.barraProgreso.Value = progreso
      progreso = progreso + 1

      matriz(0, 0) = (frmVistas.picPrincipal.Point(i, j) / 65536) * kernel0
      matriz(0, 1) = (frmVistas.picPrincipal.Point(i, j + 1) / 65536) * kernel1
      matriz(0, 2) = (frmVistas.picPrincipal.Point(i, j + 2) / 65536) * kernel2
      matriz(1, 0) = (frmVistas.picPrincipal.Point(i + 1, j) / 65536) * kernel3
      matriz(1, 1) = (frmVistas.picPrincipal.Point(i + 1, j + 1) / 65536) * kernel4
      matriz(1, 2) = (frmVistas.picPrincipal.Point(i + 1, j + 2) / 65536) * kernel5
      matriz(2, 0) = (frmVistas.picPrincipal.Point(i + 2, j) / 65536) * kernel6
      matriz(2, 1) = (frmVistas.picPrincipal.Point(i + 2, j + 1) / 65536) * kernel7
      matriz(2, 2) = (frmVistas.picPrincipal.Point(i + 2, j + 1) / 65536) * kernel8

      color = (matriz(0, 0) + matriz(0, 1) + matriz(0, 2) + matriz(1, 0) + matriz(1, 1) +    matriz(1, 2) +
      matriz(2, 0) + matriz(2, 1) + matriz(2, 2)) / factor

      If color < 0 Then
            color = 0
       End If

     frmEfectos.picEfecto.PSet (i, j), RGB(color, color, color)
     DoEvents
   Next i
 Next j

'********** FIN DEL CÓDIGO **********'
```

En el código anterior se puede observar que la matriz de 3x3 correspondiente a un kernel, barre y multiplica los valores de pixel de una imagen abierta sobre un control llamado: "picPrincipal" el cual a su vez se encuentra ubicado sobre el formulario: "frmVistas" y el valor resultante se divide con respecto al valor del factor.

La parte de la barra de progreso, sirve para dibujar el avance a manera de cuadritos porcentuales y el DoEvents permite que el procesador no se bloquee durante el proceso de transformación.

Revisemos ahora los resultados de la aplicación de todos efectos restantes:

Pasa baja

Pasa Alta

Laplaciano

Sobel Horizontal

Sobel Vertical

Sobel Diagonal Normal

Fig. 3.14 El filtro Sobel Diagonal invertido ya lo mostramos en las páginas anteriores y el ajustable no aparece porque sólo basta con que modifiquemos los valores de la ventana de transformación para obtener cualquier otro efecto no especificado

En el siguiente paso, configuraremos las variables de la escena que alojará el diseño visual tridimensional, empezando por la cámara, después la fuente de luz y finalmente el fondo:

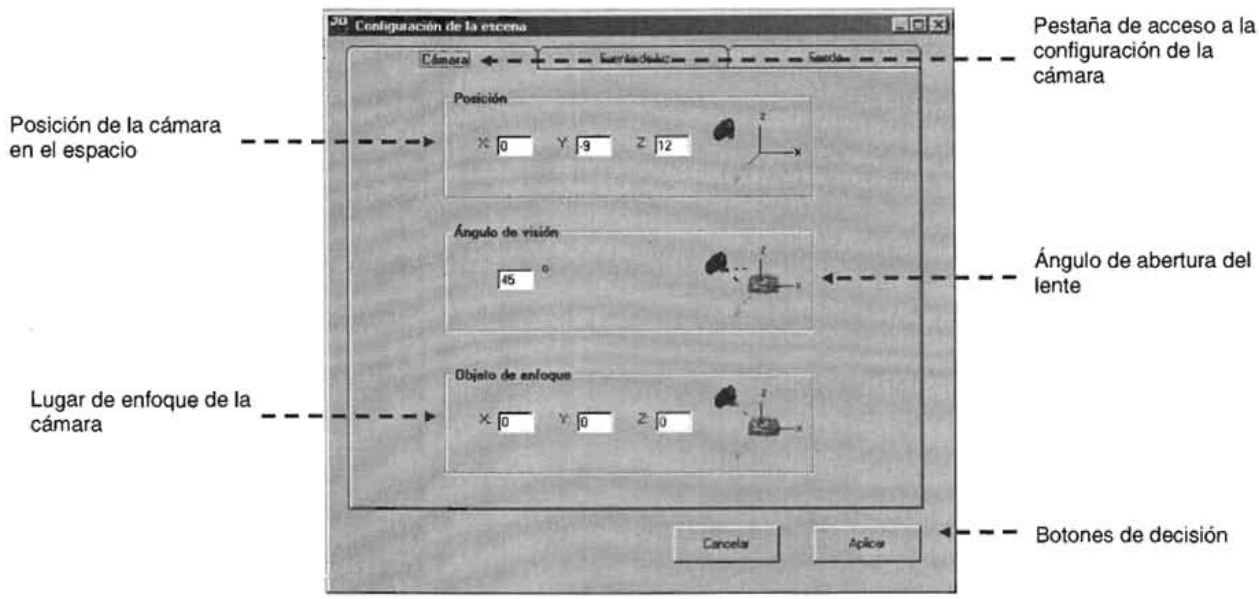


Fig. 3.15 Para visualizar un modelo tridimensional con la cámara virtual, podemos configurar el sitio desde donde veremos un objeto, el ángulo y dirección de visión

En la fase de diseño mencionamos que para crear escenas en tres dimensiones, nos auxiliaríamos de unas librerías descargables desde internet llamadas "OpenGL" escritas para visual Basic, con dichas librerías, podemos construir y configurar un control que para dibujar objetos tridimensionales.

El código de configuración de la cámara es el siguiente:

```
' ************************************** CONFIGURACIÓN DE LA CÁMARA ********************************** '

With gCtl.Camera
   '---------- ÁNGULO
   .FieldOfView = camaraAnguloVision
   '---------- POSICIÓN
   .SetEyePos camaraPosicionX, camaraPosicionY, camaraPosicionZ
   '---------- LUGAR DE MIRADA
   .SetTargetPos  camaraObjetoEnfoqueX, camaraObjetoEnfoqueY, camaraObjetoEnfoqueZ      End With

'************************************************** FIN DEL CÓDIGO ************************************************'
```

La configuración de la cámara del ambiente 3D es muy sencilla y sólo exige el paso de parámetros a los métodos respectivos: FieldOfView: determina el campo de visión, SetEyePos: ubicación de la cámara y SetTargetPos: lugar de enfoque

La figura 3.15 refiere a la fuente de luz que se encargará de iluminar a nuestros objetos:

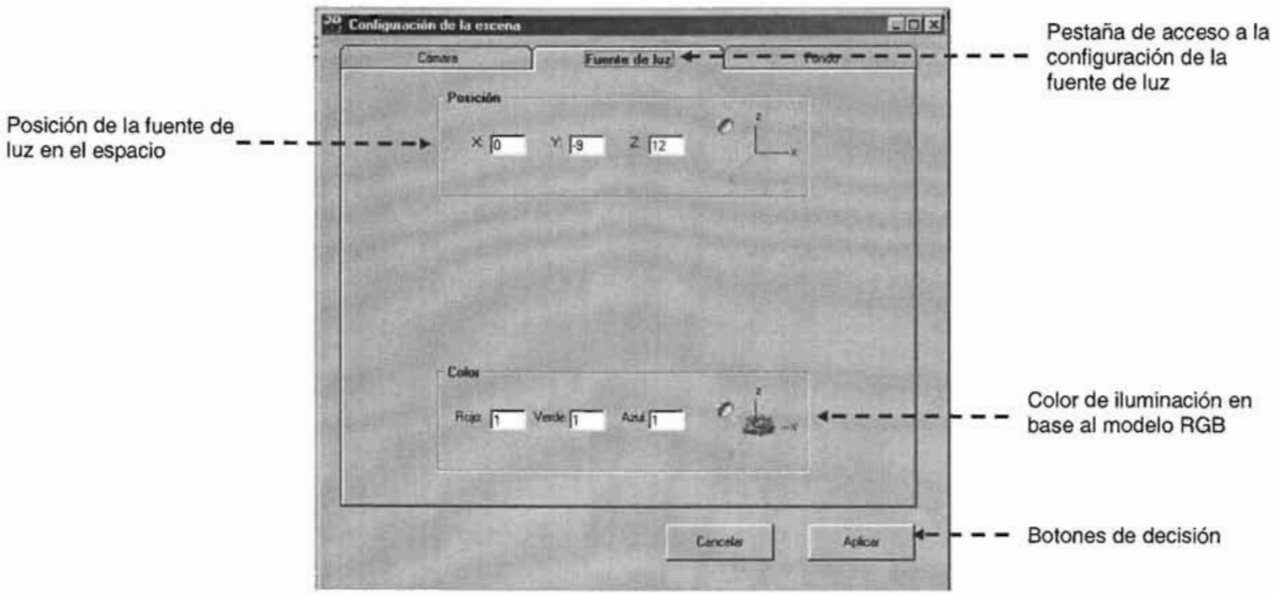


Fig. 3.15 La fuente de luz ilumina la escena y todo lo que está en ella

El código en Visual Basic con ayuda de OpenGL para la puesta a punto de la iluminación es:

```
'************************************** CONFIGURACIÓN DE LA FUENTE DE LUZ ********************************

  With gCtl.Lights.Item(liLight0)
    .SetAmbient 0.1, 0.1, 0.1
    .SetDiffuse luzColorRojo, luzColorVerde, luzColorAzul
    .SetPosition luzPosicionX, luzPosicionY, luzPosicionZ
    .Enabled = True
  End With

'************************************************** FIN DEL CÓDIGO ***********************************************
```

Para ajustar el fondo, sólo debemos de indicar su color, tal y como se muestra en la ventana de abajo:

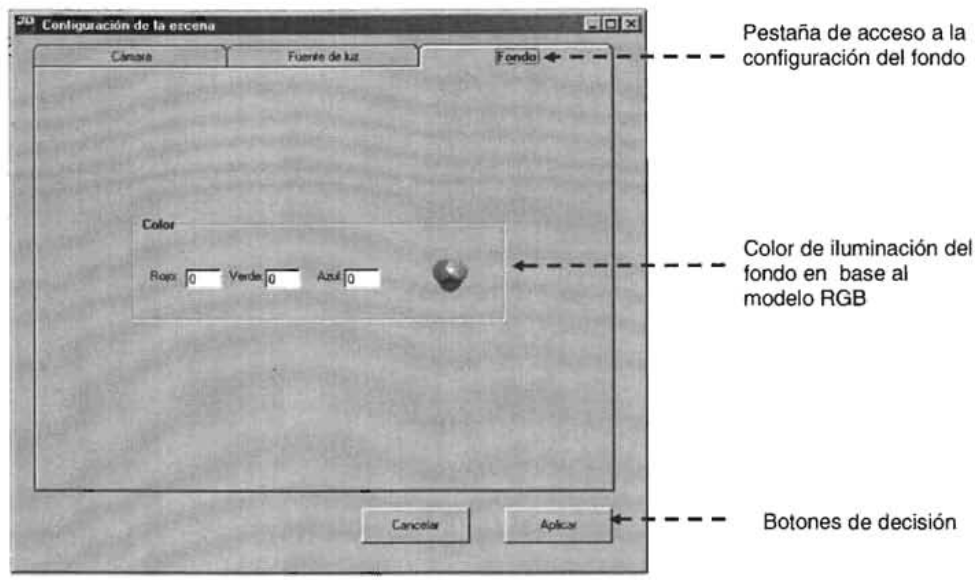


Fig. 3.16 El fondo se define en RGB

El código en OpenGL para visual Basic que controla el fondo de la escena es el siguiente:

```
'********************************* CONFIGURACIÓN DEL FONDO DE LA ESCENA ****************************** '

    glClearColor fondoRojo, fondoVerde, fondoAzul, 0

'************************************************** FIN DEL CÓDIGO ***********************************************'
```

La función glClearColor define el color del fondo y recibe los valores del modelo RGB.

Ahora veamos como construir un modelo visual en 3D a partir de una imagen plana. Aunque anteriormente incluimos uno junto con los efectos, en esta sección lo volveremos a hacer pero partiendo de una imagen pura:

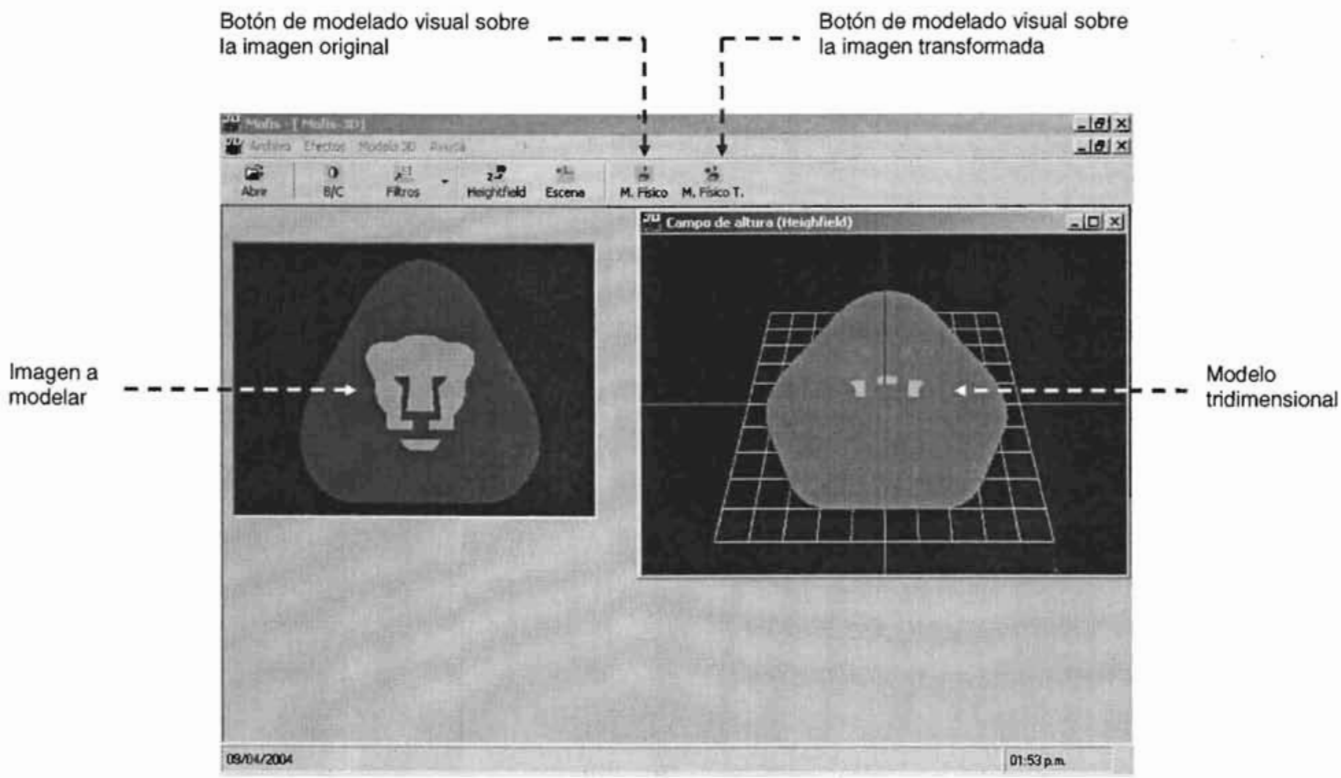


Fig. 3.17 El logotipo plano de los pumas se convierte en un modelo visual en 3D

Los pixeles de la imagen plana (izquierda) se transforman en líneas tridimensionales con la ayuda del código OpenGL siguiente:

```
'********************************* CREACIÓN DE LÍNEAS TRIDIMENSIONALES *********************************'

    glLineWidth grosor                  'DEFINE EL GROSOR DE LA LÍNEA

    glBegin bmLines
        glVertex3d x1, y1, z1           'UBICACIÓN DEL PRIMER PUNTO
        glVertex3d x2, y2, z2           'UBICACIÓN DEL SEGUNDO PUNTO
    glEnd

'************************************************** FIN DEL CÓDIGO **************************************************'
```

Con el método glLineWidth especificamos el grosor que tendrá nuestra línea, mientras que con glVertex3d definimos los dos puntos en el espacio sobre los que se trazará dicha línea.

La figura 3.18 corresponde a la creación física del modelo tridimensional, lo cual es el punto neurálgico de toda la tesis; es el lugar en el que se conjuga la interacción humano-computadora-periférico.

Para crear un modelo en madera, lo podemos hacer de manera indistinta desde una imagen pura o desde una previamente transformada con la ayuda de los diferentes efectos.

Al seleccionar la opción de creación del modelo físico a partir de la imagen original o transformada, la ventana que aparece es la siguiente:

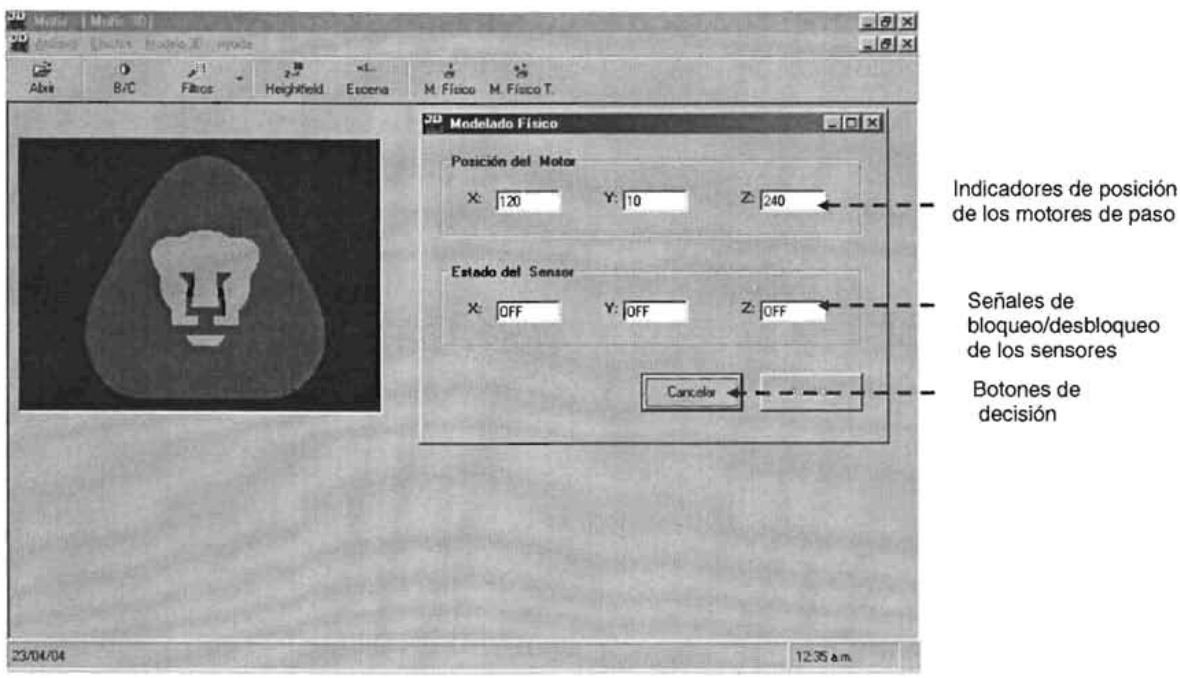


Fig. 3.18 Al momento de crear el modelo en madera, lo único que veremos será una ventana de estado de progreso de los motores junto con el sensado de los mismos

En la figura anterior, se pueden observar dos tercias de valores. La primera de ellas muestra de manera dinámica la posición de cada uno de los motores que contribuyen en la perforación de la madera. Y la segunda muestra el estado de los sensores que detectan si los motores están fuera de su posición inicial.

Si nuestra imagen a modelar es de 320x240 como en el ejemplo de arriba, veremos que en la pantalla el valor de X avanzará desde 0 hasta 320 de manera reiterativa un total de 240 veces (equivalentes al valor de Y), mientras que el resultado de la multiplicación anterior especificará el número de veces que el eje Z se moverá para perforar la madera en la profundidad especificada por el color del pixel.

La sección de sensores tiene por objetivo asegurar la reinicialización de la base de modelado y el taladro, antes de comenzar un nuevo trabajo. Por ejemplo, el valor de "OFF" en X señala que la plataforma se ha desplazado por lo menos una unidad fuera de su punto de arranque, y así sucesivamente para Y y Z. Por otra parte, un valor "ON" indicará que la situación es inversa.

Para comandar desde la computadora los motores del hardware, es necesario enviar y recibir información desde el puerto paralelo mediante el auxilio de una librería de enlace dinámico previamente mencionada (inpout.dll). Y para usar sus funciones desde cualquier formulario, es preciso crear un módulo .bas en el que se definan los métodos de entrada y salida (Inp y Out):

```
Public Declare Function Inp Lib "inpout32.dll" Alias "Inp32" (ByVal DirPuerto As Integer) As Integer
Public Declare Sub Out Lib "inpout32.dll" Alias "Out32" (ByVal DirPuerto As Integer, ByVal Value As Integer)
```

El uso de los métodos exige el paso de la dirección numérica del puerto paralelo, el cual suele ser 378 en hexadecimal:

```
DirPuerto = &H378
```

Un ejemplo de salida de datos para comandar un motor de pasos bipolares según la tabla 2.1 es:

```
Out DirPuerto, 10
Out DirPuerto, 9
Out DirPuerto, 5
Out DirPuerto, 6
```

Para la parte que corresponde a la entrada de datos desde los sensores, el código es:

valorSensores = Inp(DirPuerto)

Finalmente, la pantalla "Acerca de..." muestra datos sobre la creación del software:

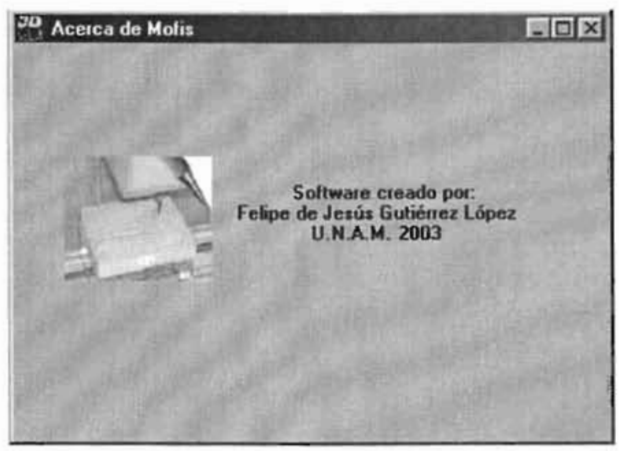


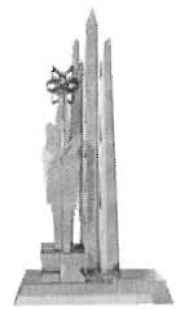
Fig. 3.19 Ventana Acerca de...

La siguiente fase a seguir dentro de la metodología del Proceso Unificado sería la de pruebas, sin embargo ya la hemos realizado de cierta manera en esta etapa por lo que evitaremos definirla, al igual que la implementación y el mantenimiento.

El apartado contiguo englobará algunos de los beneficios explicados a lo largo de esta tesis, junto con ejemplos de aplicación práctica del Mofis.

CAPÍTULO IV

APLICACIONES



En capítulos anteriores definimos algunos conceptos sobre: imágenes, hardware y software para la construcción del modelador físico tridimensional deseado. De aquí en adelante veremos algunas de las aplicaciones que podremos darle al Mofis y también mencionaremos otro tipo de periféricos que pueden elaborarse.

4.1 ADORNOS Y PUBLICIDAD

La forma cotidiana de vivir del ser humano se rige en gran parte por el consumismo impulsado por la astucia de la mercadotecnia, y es por eso que las empresas invierten un porcentaje importante de sus utilidades en publicidad para asegurar su difusión y crecimiento.

La mercadotecnia para lograr sus objetivos de impacto, se apoya de cualquier mecanismo que sea capaz de transmitir un mensaje certero, siendo sus principales objetivos los siguientes:

- Que lo producido sea visto por la mayor cantidad de personas.
- Que lo producido sea lo más atractivo posible y que abarque el mayor número de sentidos.
- Que lo producido logre un impacto psicológico fuerte (principalmente en las personas a las cuales va dirigido).

Para lograr lo anterior, la mercadotecnia se apoya principalmente de los medios de comunicación masivos como: la televisión, radio, internet, revistas, periódicos, espectaculares, entre otros.

Grandes empresas como: Coca Cola, Pepsi, Nike, Microsoft, Bimbo, Telmex,

Aeroméxico, entre otras, destinan varios millones de dólares en su publicidad, de ahí su alta popularidad y la inclinación de muchas personas hacia ellos.

Con el Mofis podremos crear pequeñas esculturas publicitarias de madera que acompañen escritorios de oficinas, marcos de puertas, muebles, etc.

Veamos en la figura 4.1 un modelo en madera creado a partir del logo de la marca nike:

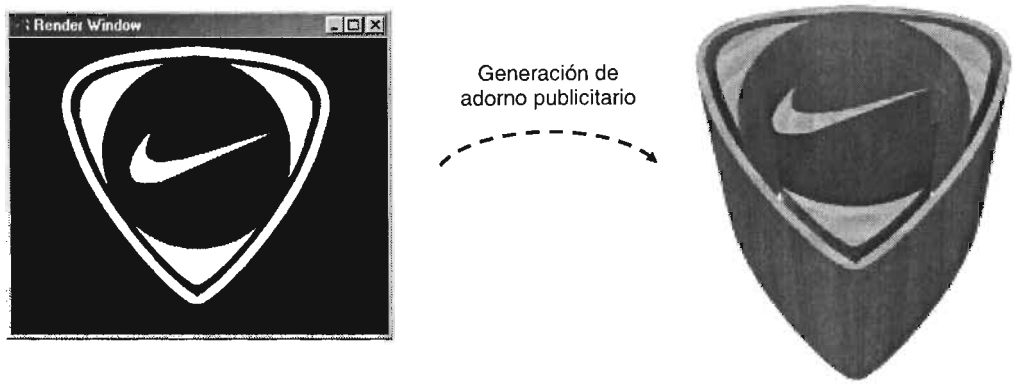


Fig. 4.1 Los adornos creativos y duraderos, son recursos importantes para la difusión de una marca

4.2 MOLDES

Una de las aplicaciones más interesantes del Mofis se encuentra en la elaboración de moldes para la reproducción de objetos.

Para crear un molde es preciso tomar en cuenta que la imagen a modelar deberá estar reflejada horizontalmente y los pixeles importantes deberán tener una tendencia hacia el negro, de tal manera que se obtenga un hueco en el que se pueda verter algún material endurecible.

En la figura 4.2 se incluye el molde para una neurona y el empaque de una bomba de agua:

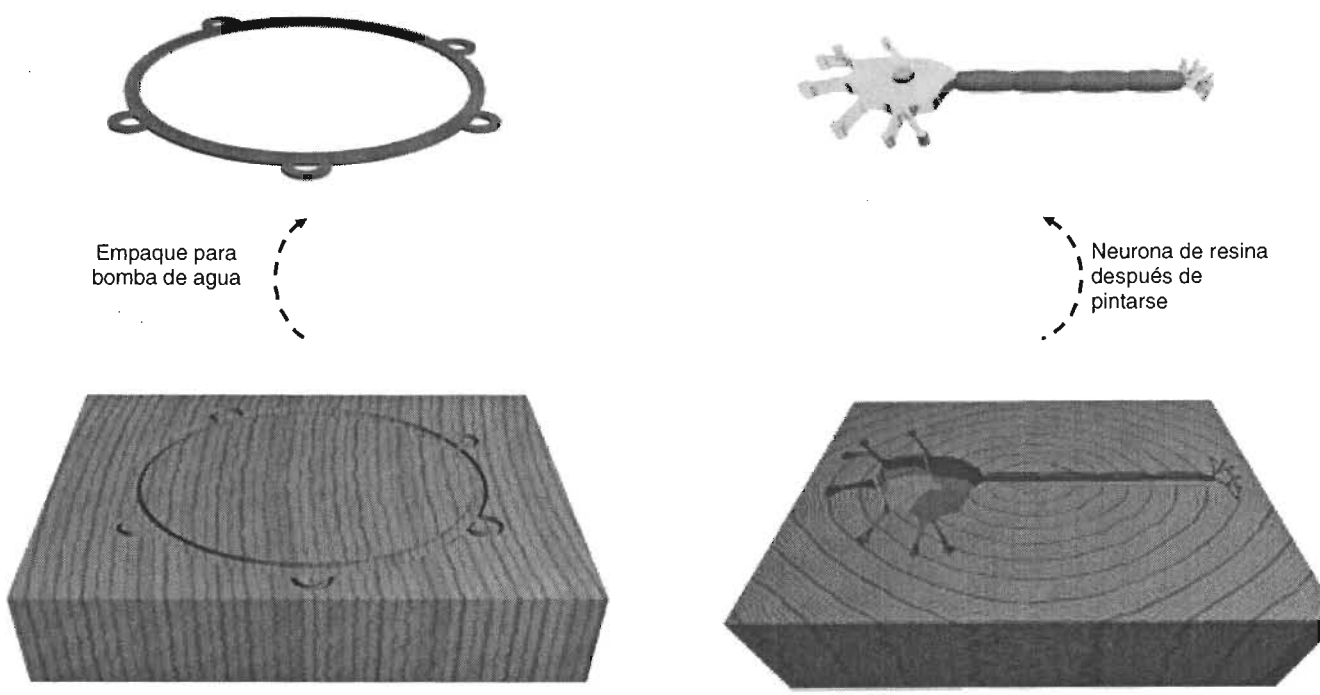


Fig. 4.2 A partir de un solo molde se pueden crear muchas copias de un objeto

4.3 MÉDICAS

La medicina es un campo de vital importancia para la investigación debido a su relación con la salud humana, por lo que si el Mofis tiene aplicación en este ramo, será muy relevante.

Cuando se toman radiografías a huesos rotos, la parte afectada por lo regular se identifica de manera inmediata por las fisuras que se presentan en forma de astillas, las cuales según su existencia y distribución serán factor clave de reacomodo de las partes separadas.

Con la ayuda del Mofis, será posible crear la representación tridimensional de radiografías de cualquier tipo, y en la imagen siguiente se ilustra el modelado de una radiografía bucal:

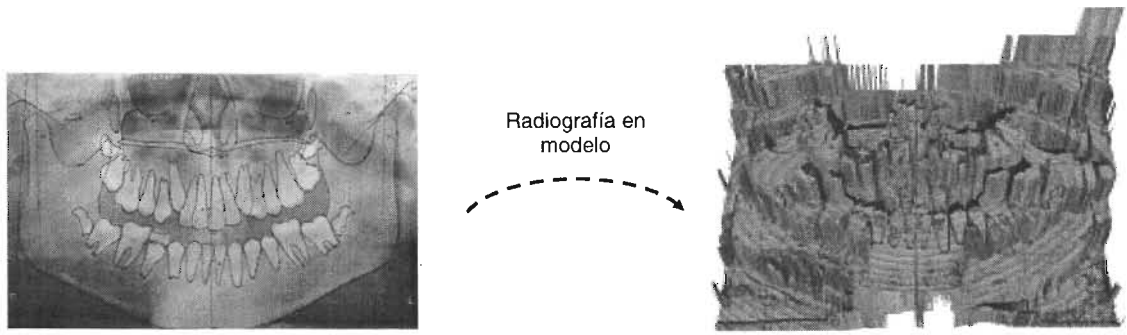


Fig. 4.3 En la madera se obtiene una impresión en relieve de los huesos faciales

4.4 ESCÁNER TRIDIMENSIONAL

Como se dijo al inicio, la información de esta tesis no sólo sirve para construir un Modelador Tridimensional, sino que también da la pauta para la elaboración de otros periféricos, es por eso que veremos algunos ejemplos de lo que podemos hacer.

Los escáners o dispositivos periféricos con los cuales es posible digitalizar fotografías, texto u objetos, son aparatos que rastrean una superficie plana para obtener una imagen bidimensional en el monitor de una computadora. Los escáners actúan a manera de cámaras fotográficas que capturan una imagen en 2D.

Para construir un escáner en 3D que nos permita obtener imágenes espaciales, es necesario que contemos con una cámara digital que se conecte a la computadora, una base giratoria con un motor de pasos, un objeto a ser modelado, dispositivos electrónicos de control y alimentación de energía, una pantalla azul y un software capaz de procesar la entrada de imágenes y de generación del modelo en 3D .

El funcionamiento del escáner 3D es el siguiente:

Una cámara digital tomará varias fotos a un objeto en todo su alrededor, para obtener un conjunto de imágenes en los 360° del mismo. El motor de pasos se encargará de girar la charola sobre la cual se encuentre el objeto. La pantalla azul servirá para procesar de una manera más simple las imágenes capturadas. El software controlará la cámara y el motor, recibirá las imágenes desde la cámara, las procesará mediante filtrado, construirá un modelo de alambre, y renderizará[19] la imagen en 3D.

A cada fotografía, el software le aplicará un proceso de filtrado para después sacarle el contorno y generar un modelo de alambre completo (con vectores) que al aplicarle un relleno o textura permita obtener imágenes realistas después del respectivo renderizado.

Con el mismo software se podrían exportar los archivos para ser manejados por programas como: POV Ray, 3D Studio, Maya, entre otros.

En la figura 4.4 se ilustra el funcionamiento de este periférico:

[19] Renderizar es el proceso de trazado de un imagen

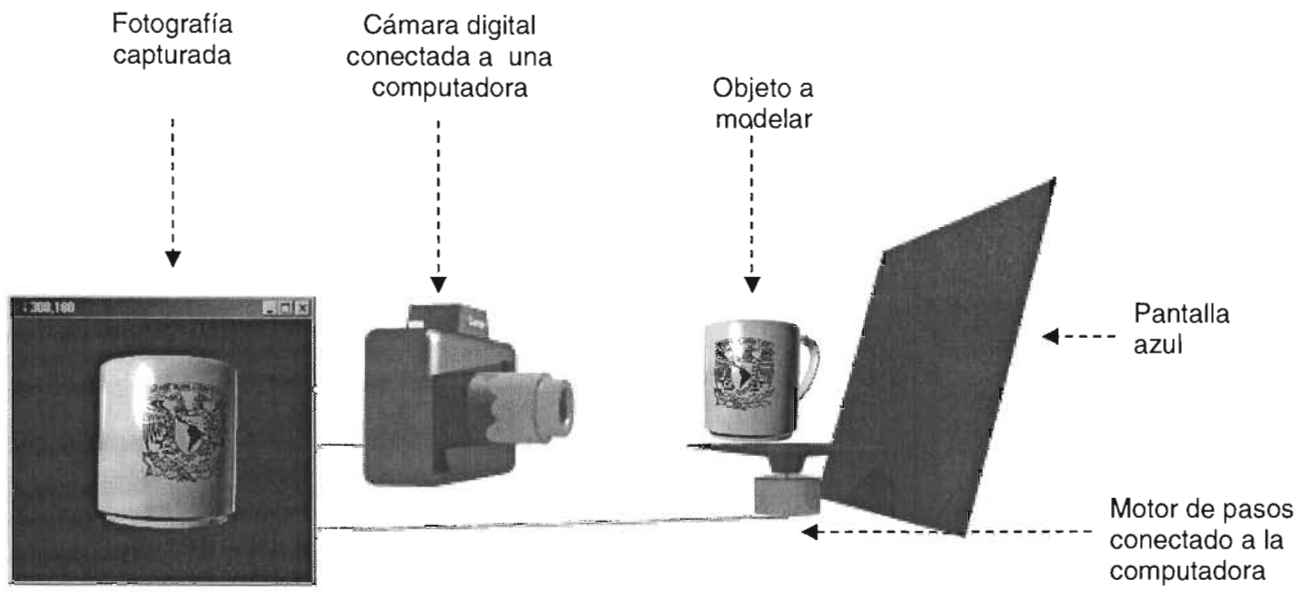


Fig. 4.4. La cámara toma fotos a un objeto que gira 360º con la ayuda de un motor de pasos controlado por computadora

El software que controla y procesa las imágenes, crea el modelo visual tridimensional tal y como se muestra en la imagen de abajo:

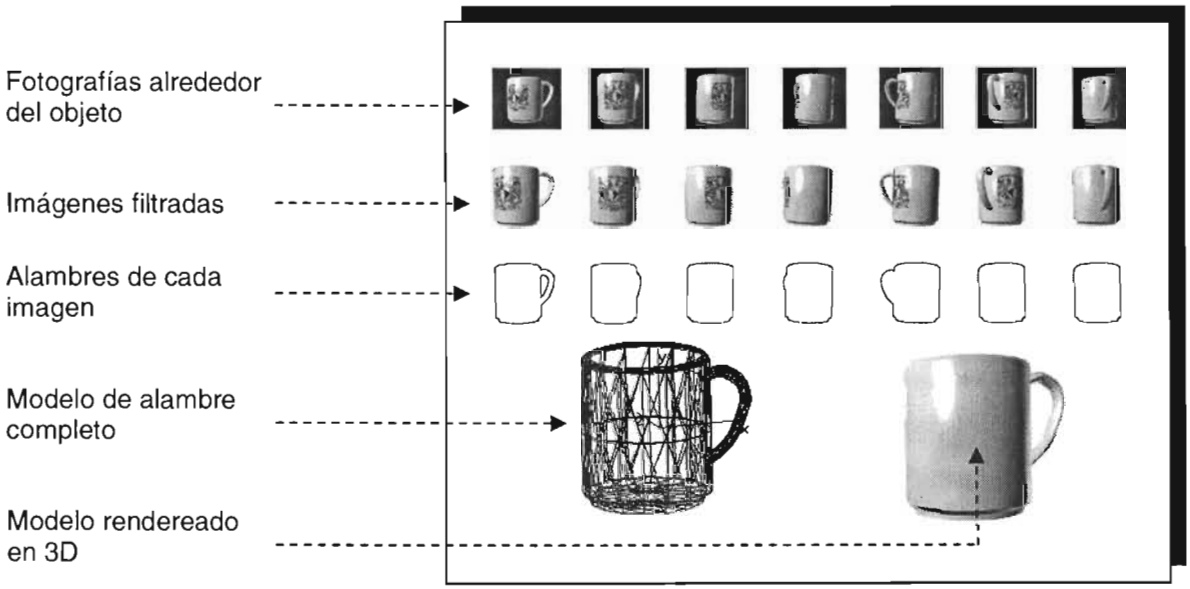


Fig. 4.5 De cada fotografía se obtiene un alambre que se unirá con todos los generados a lo largo de las fotos en 360º, a partir de ellos se construye un modelo de alambre completo que al renderearse generará el modelo visual en 3D.

4.5 FILMADORA EN 360º

Las gran mayoría de las películas que hasta ahora se han producido sólo pueden verse de una sola perspectiva debido a que solamente existe una grabación lineal de la misma; aunque existan muchas tomas desde diferentes ángulos.

Con algunas películas en DVD pueden verse ciertos fragmentos animados desde perspectivas diferentes, tal y como si eligiéramos la cámara desde donde quisiéramos ver la escena, sin embargo, nunca en la historia del cine, se ha producido una película con total perspectiva en 360º , es decir que nosotros decidamos en cualquier momento el ángulo que deseamos ver.

Para construir una filmadora en 360º, se necesita de cuatro filmadoras unidas a un centro dispuestas a manera de círculo, dichas filmadoras deben ser conectadas a la computadora y se deberá de contar con un programa de mezclado y visualización de video y sonido.

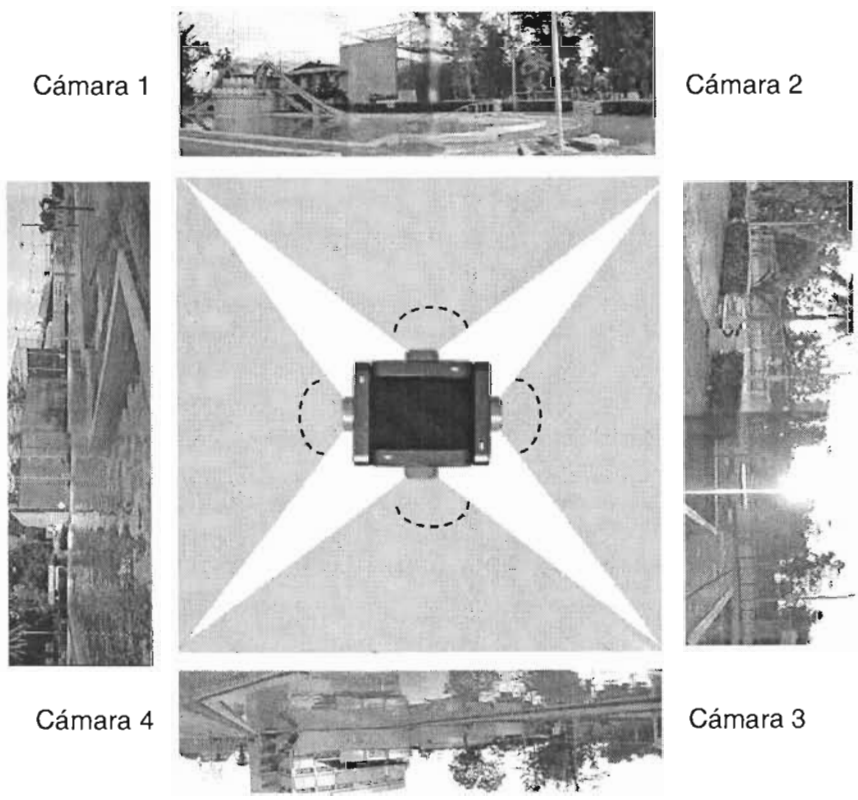


Fig. 4.6 Vista superior en la que las filmadoras graban cubriendo los 360º de la escena

Cada una de las filmadoras cubrirá 90º de la escena y lo grabado se almacenará en 4 archivos de audio y video comprimidos en formato MPEG-4 para ocupar un espacio reducido y alta calidad.

Un software de visualización será el responsable de crear la magia, ya que con un control de giro el espectador podrá ver en cualquier momento la escena desde el ángulo deseado. Dicho software mostrará la parte proporcional de video y audio de dos cámaras al mismo tiempo. Veamos el siguiente dibujo:

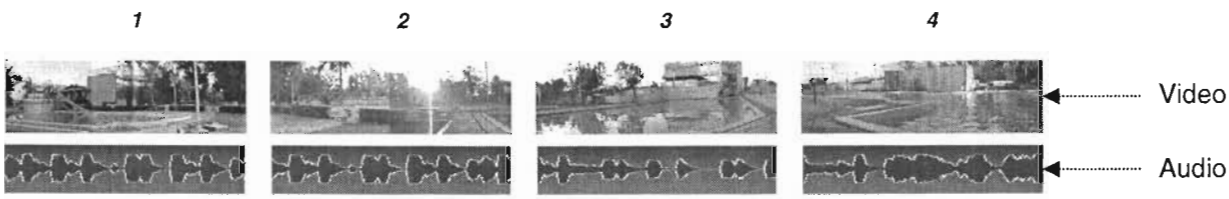


Fig. 4.7 Archivos de audio y video en 360º

Suponiendo que la pantalla de inicio es la que se almacena en el archivo 1 y el espectador comienza a girar su visión un 20% hacia lo que capturó la cámara dos. El software se encargará de mezclar 80% de audio y video del archivo 1 y 20% del segundo.

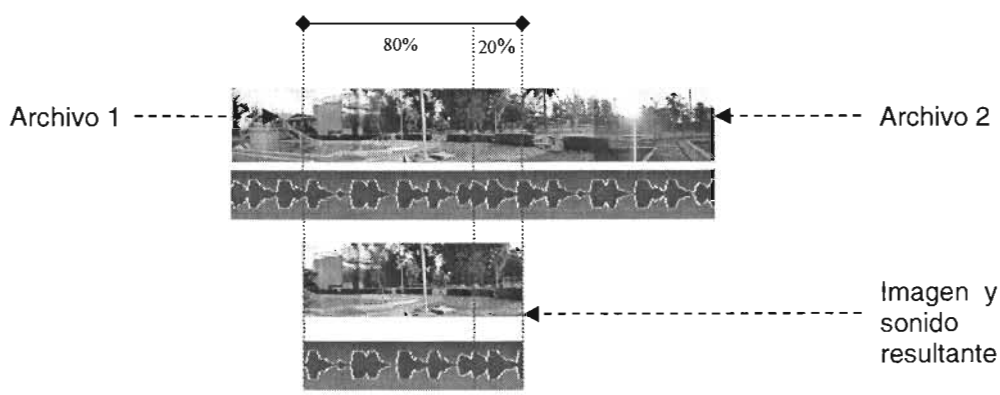


Fig. 4.8 La imagen 1 se mezcló con la 2 dando la perspectiva de giro

Y así como se mezcló la imagen 1 con la 2, la 2 se puede mezclar con la 3, la 3 con la 4 y la cuatro con la 1, de tal manera que podremos ver en cualquier momento la escena desde cualquier perspectiva.

4.6 MONITOR DE LEDS

En algunos establecimientos como: bancos, supermercados, verificentros, estadios de futbol, etc, se utilizan pantallas electrónicas para desplegar mensajes de gran atracción. Dichas pantallas están compuestas por una matriz de leds que permiten desplegar mensajes y/o imágenes con diversos efectos.

Partiendo del mismo concepto, es posible construir un pequeño monitor a color que reciba la información directamente de una computadora para mostrarla al público espectador.

Para la elaboración del minimonitor de leds, se necesitan:

- n x m^{20} minileds cuadrados de color rojo, esa misma cantidad en verde y azul, con sus respectivas resistencias.
- Una placa matriz donde distribuir espacialmente los minileds
- Un circuito electrónico que convierta las señales digitales de la computadora en señales analógicas y que seleccione a manera de coordenadas el pixel que debe recibir la información en el tiempo requerido.
- Una fuente de poder capaz de alimentar todo el circuito electrónico.
- Cables de conexión con el puerto de la computadora a utilizar
- Un software que envie la información del monitor hacia el circuito de control del minimonitor.

El software barrerá los pixeles del monitor de la computadora a una frecuencia de 12 fps o cuadros por segundo y enviará los valores de cada uno de los componentes de color hacia el circuito electrónico que convertirá dichos valores en señales analógicas para ser recibidas por cada trío de leds en la posición respectiva, de tal manera que se muestren en el minimonitor 12 imágenes por segundo.

A continuación se muestra un esquema visual del minimonitor de leds:

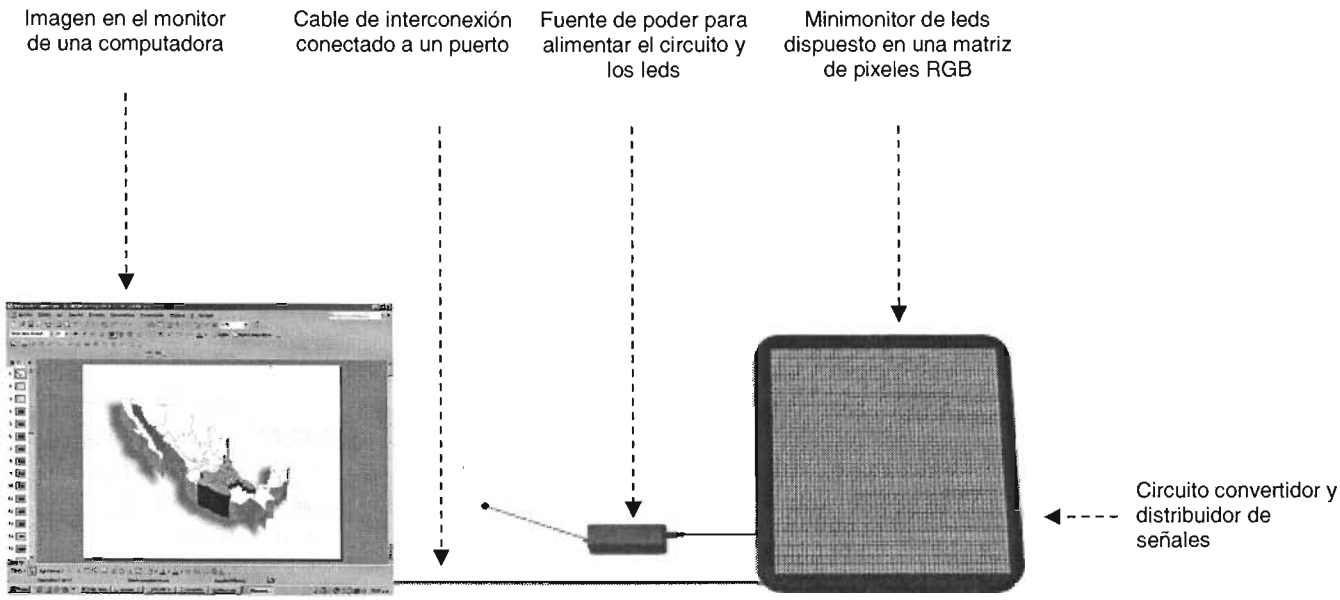


Fig. 4.9 La imagen del monitor de una computadora es enviada al minimonitor por uno de sus puertos, para ser mostrada en pixeles compuestos por leds rojo-verde-azul

[20] nxm es la resolución que daremos al minimonitor, es decir, si deseamos que sea de 320 x 240, la multiplicación dirá que necesitamos esa cantidad en leds rojos, verdes y azules.

4.7 MODELADOR FÍSICO TRIDIMENSIONAL LÁSER

El modelador planteado en esta tesis utiliza como herramienta de esculpido un taladro y como materia prima un bloque de madera, sin embargo en la actualidad existen periféricos que realizan en esencia la mismo tarea pero utilizan como herramienta de modelado un rayo láser controlado y como materia prima una sustancia que se cataliza[21] con el mismo rayo.

El principio de funcionamiento de este modelador es similar al del Mofis, sólo que ahora lo que controlamos es la potencia y dirección de un rayo láser. Veamos el esquema siguiente:

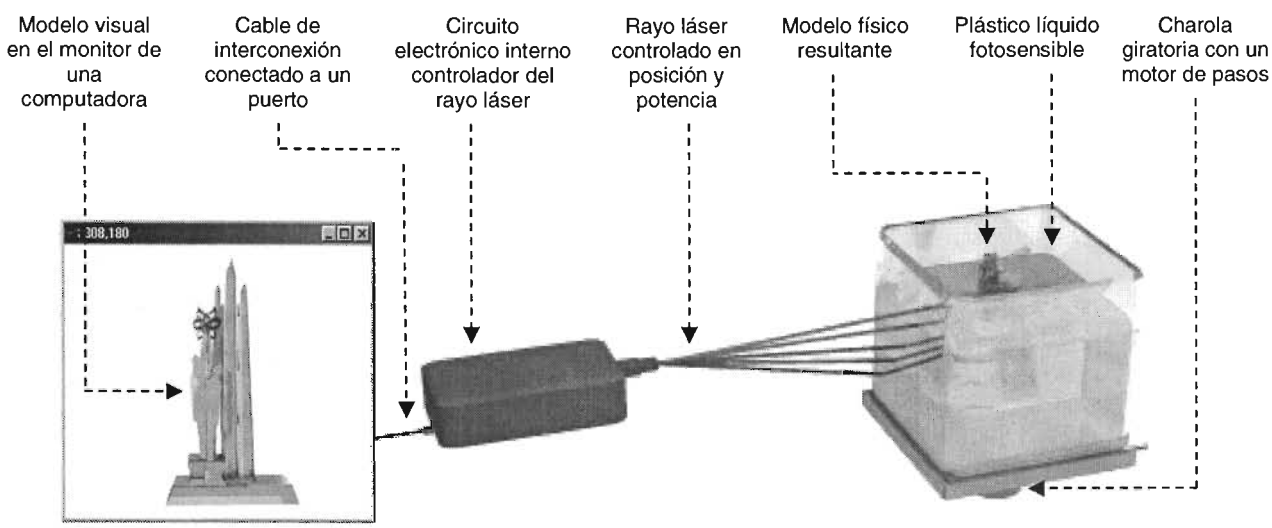


Fig. 4.10 El rayo láser moldea una figura al acelerar el endurecimiento del plástico líquido fotosensible

Cabe señalar que este tipo de periféricos son muy costosos por lo que su manufactura debe ser muy compleja.

[21] La catalización es un proceso de aceleración o retardo de un proceso químico, en este caso el rayo láser agiliza el endurecimiento del líquido modelador.

4.8 MODELOS DE ELEVACIÓN DIGITAL

En el primer capítulo, se definieron algunos conceptos importantes relacionados con los formatos gráficos de las imágenes, dentro ellos se encontraba el Modelo de Elevación Digital o DEM, el cual es una estructura numérica con ternas de valores que representan puntos espaciales.

Con archivos de tipo DEM es posible representar el mundo entero en tres dimensiones, e inclusive existen organizaciones internacionales que poseen bases de datos de muchas regiones dentro de las cuales se encuentra nuestro país con sus bellezas naturales.

El software del Mofis lee archivos de tipo: GIF, JPG y BMP, sin embargo no posee la capacidad de procesar datos de tipo DEM, por lo que para futuras actualizaciones sólo bastará con crear un módulo de lectura para que el hardware elabore los modelos correspondientes.

Logrando las adaptaciones necesarias para la lectura de estos archivos, el Mofis podría crear físicamente mapas de cualquier parte del mundo (de internet se pueden descargar archivos DEM gratuitos).

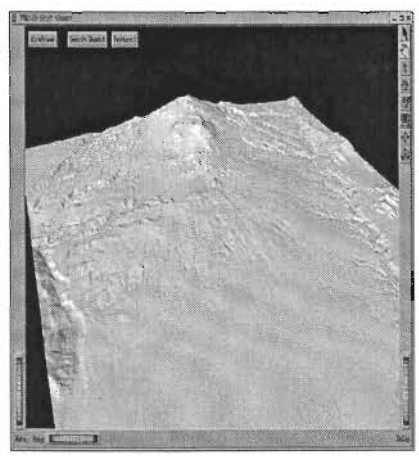


Fig. 4.11 Modelo de Elevación Digita de la República Mexicana que podría modelarse físicamente en un futuro

4.9 IGUALADOR DE COLORES

A parte de los proyectos ya mencionados, con la teoría de esta tesis es posible construir otros como: bordadoras, igualadores de colores, sismógrafos, plotters, etc., sin embargo cada uno de ellos exige un estudio y trabajo profundo

Sin entrar en mucho detalle mencionaremos la manera de crear un igualador de colores:

Cuando vamos a un establecimiento que dice en un letrero "se igualan colores por computadora" resulta de todo impresionante saber que el color pensado será el que obtendremos, sin embargo, después de haber leído en el primer capítulo lo referente a los modelos de color como el CMY, es fácil darnos cuenta que la respuesta secreta lleva esa dirección.

Para crear un igualador de colores será necesario contar con tres recipientes para cada pintura del modelo CMY, es decir, con los componentes Cyan, Magenta y Amarillo, también se deberán de tener tres válvulas que controlen el vaciado de cada uno de los colores sobre un mismo recipiente y en la proporción correcta para que al mezclar se obtenga el color exacto.

Los recipientes de color deben de incluir sensores que detecten los niveles de pintura para que se pueda saber el momento en el que se encuentren en nivel bajo.

Suponiendo que alguna persona desea que se le iguale algún color a partir de una lámina de auto, u otro objeto, únicamente se tendrá que escanear la superficie del mismo y mediante el software del igualador se separarán los colores respectivos para dar la instrucción de vaciado de cada recipiente.

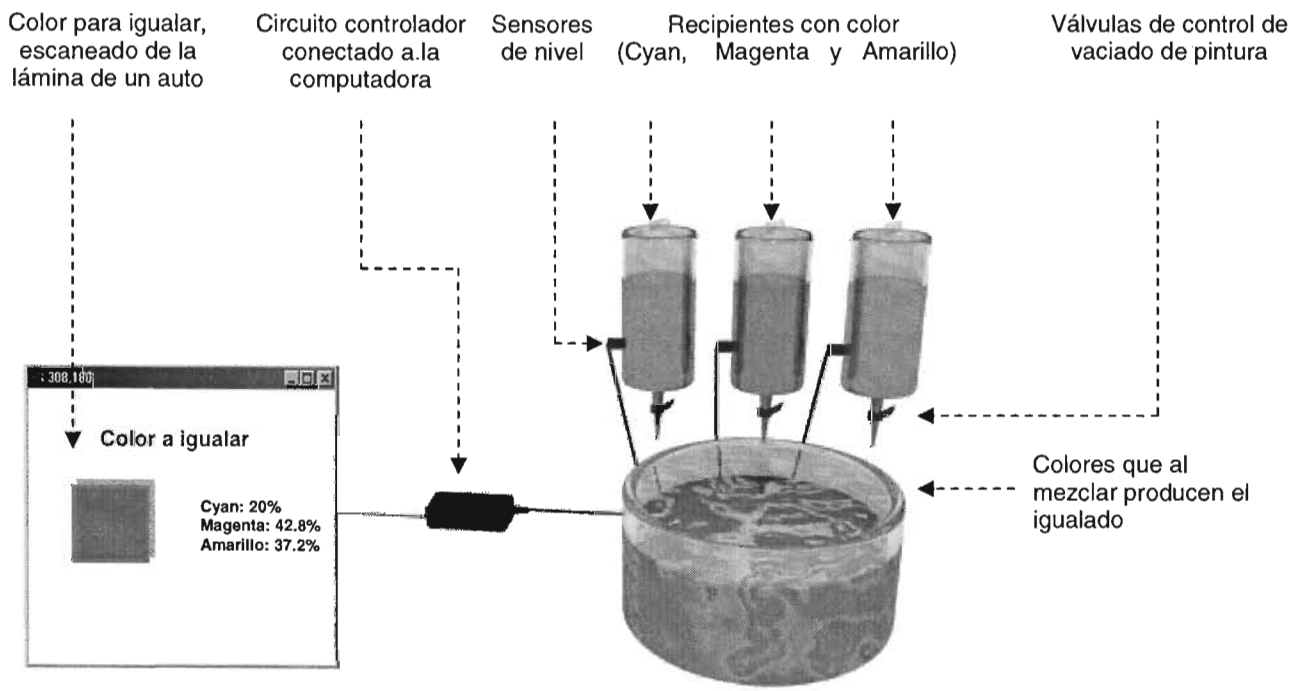


Fig 4.12. Con la ayuda del circuito principal, se controlan las válvulas para verter la proporción exacta de cada color y lograr el igualado

Una vez que se tiene toda la pintura en un solo bote, sólo hará falta mezclarla para asegurar que se obtenga un tono uniforme, para lo cual también podremos construir una máquina revolvedora.

CONCLUSIONES

A lo largo de la presente tesis hemos demostrado que con los conocimientos adquiridos en la licenciatura en computación, es posible construir periféricos de aplicación interesante como lo es: un modelador físico tridimensional, reutilizando componentes físicos (impresoras, escáners y ratones), y lógicos (librerías de software como: OpenGL e inpoutl.dll).

Hemos visto que la elaboración de un periférico computacional requiere de la composición de una parte física llamada: "hardware" (motores, estructuras, cables, circuitos integrados, etc.) capaz de realizar nuestro trabajo, y de una lógica: "software" (conjunto de programas) cuya función sea comandar al hardware.

Tratándose de hardware podemos concluir que:

- Para lograr movimientos exactos y controlados es necesario utilizar motores de paso (bipolares o unipolares), mientras que para el caso contrario se pueden usar los de corriente continua.
- Para convertir el movimiento circular de un motor en longitudinal, se pueden utilizar tornillos o bandas sin fin.
- Para facilitar y potenciar el manejo de motores de paso y corriente continua, lo ideal es adquirir circuitos integrados comerciales llamados drivers o controladores.
- Para realizar funciones de control digital complejas no disponibles en circuitos comerciales, lo recomendable es programar circuitos "PLD" (Dispositivos Lógicos Programables) mediante el lenguaje para hardware "VHDL" y un programador universal.
- El mejor puerto de intercomunicación de periféricos es el USB por su alta velocidad de transferencia y capacidad para reconocer dispositivos "al vuelo",

pero su uso requiere de un circuito de control.

- Para proteger los puertos de la computadora de la entrada de corrientes dañinas, se pueden usar dispositivos de acoplamiento de señales como los optoacopladores.
- Con la ayuda de sensores podemos conocer ciertas variables internas/externas a nuestro periférico, tales como: posición, temperatura, iluminación, sonido, entre otros.
- Los dispositivos electrónicos para funcionar requieren de corriente continua controlada, para lo cual es necesario utilizar transformadores reductores.

Para el software concluimos que:

- Un desarrollo de software bueno y controlado se puede lograr siguiendo una metodología de diseño como: "El Proceso Unificado".
- La construcción del software en capas, facilita la división y manejo de tareas en: interfaz humana, dominio del problema y datos.
- Es posible generar una infinidad de efectos especiales mediante el simple cambio de los valores de un filtro de transformación o "kernel".
- Debido a la carencia de Visual Basic para comunicarse con el puerto paralelo, es necesario hacer uso de algún módulo externo como el: inpout.dll, que permita lograr dicha funcionalidad.
- El reuso de las librerías OpenGL facilita la generación de un ambiente tridimensional visual desde lenguajes de programación como: "Visual Basic".

Para modelado concluimos que:

- Todas las imágenes que consideramos planas o bidimensionales, se componen de puntos que podemos representar en el espacio tridimensional mediante la transformación del valor del color a su equivalente en el eje Z cartesiano, de tal manera que es posible generar campos de altura visuales (en el monitor) y físicos (en madera).
- Las aplicaciones del modelador son extensas y van desde simples adornos de

escritorio hasta la obtención de objetos médicos.

- Para lograr los mejores resultados de modelado, es preciso asegurar que las imágenes sean almacenadas en formatos sin pérdida de información como: bmp o gif, ya que el jpg puede producir modelos extraños como resultado de su proceso de compresión con pérdida.
- La generación de moldes con el Mofis exige que la imagen esté reflejada y que el objeto en cuestión se componga de tonos oscuros para asegurar el hueco de vaciado de alguna sustancia líquida endurecible.
- En esta versión de software sólo es posible procesar imágenes planas (bmp, gif y jpg), pero sólo basta con programar la lectura de otro tipo de flujos para la generación de modelos visuales y físicos a partir de archivos: POV, 3D Studio, DEM, entre otros.

Para la construcción de otros periféricos concluimos que:

- La teoría expuesta en esta tesis sólo ofrece información completa para desarrollar el modelador físico tridimensional, sin embargo, da las bases para la construcción de otro tipo de periféricos los cuales requieren de una investigación más amplia.

En conclusión, podemos decir que la creación de algún tipo de periférico no es una labor sencilla pero tampoco es una tarea que escape a las capacidades de un Ingeniero en Computación de la ENEP "Aragón", y más aún, que el abanico de posibilidades de desarrollo sólo está restringido a los recursos y límites de su imaginación.

BIBLIOGRAFÍA

CEBALLOS, Javier
Enciclopedia de Microsoft Visual Basic 6
Ed. Alfaomega ra-ma, México, 2000
p.p. 1028

MAXINEZ, David G. y Alcalá Jessica
VHDL "El arte de programar sistemas digitales"
Ed. CECSA, México, 2002
p.p. 352

DEMBOWSKI, Klaus
El gran libro del hardware
Ed. Marcombo, España, 2000
p.p. 956

MANO, Morris M.
Lógica Digital y Diseño de Computadores
Ed. PHH, 12ª ed, México 2002
p.p. 636

WORBOYS, Michael F.
GIS: A Computing Perspective
Ed. Taylor & Francis Publishers, E.U.A, 1995
p.p. 146

BOYLESTAD, Robert
Electrónica Teoría de circuitos
Ed. Prentice-Hall, México, 8ª ed, 1998
p.p. 784

DORF, Richard C.,
CIRCUITOS ELÉCTRICOS, Introducción al análisis y diseño
Ed. Alfaomega, México, 1995.
p.p. 418

BENNET, David
Visual C++ 5 para desarrolladores
Ed. Prentice Hall, México, 1998
p.p. 1062

NATHAN, Gurewich
Aprendiendo Visual Basic 5 en 21 días
Ed. SAMS, México, 1998
p.p. 832

GREG, Perry
Aprendiendo Visual Basic 5
Ed. SAMS, México, 1998
p.p. 496

OpenGL:
www.opengl.org/

Allegro Microsystems:
www.allegromicro.com

Lattice:
www.lattice.com

ST Microelectronics:
www.st.com