Universidad Don Bosco

Facultad de Ingeniería

Escuela de Electrónica

PROTOTIPO EXPERIMENTAL PARA EL ESCANEO DE OBJETOS EN 3D

Alumnos:

Ricardo Ernesto Rivas Mendoza

Edwin Giovanny Contreras Nieto

Omar Antonio Méndez Martínez

CIUDADELA DON BOSCO SEPTIEMBRE DE 2004.

INTRODUCCIÓN

En El Salvador no existen equipos capaces de realizar el escaneo en tres dimensiones, sin embargo se cuentan con máquinas que pueden utilizar los datos generados por estos sistemas de escaneo, lo cual lleva a la subutilización de dichas máquinas. Por otro lado, programas como AUTOCAD®, MatLab®, entre otros, hacen uso objetos en tres dimensiones, los cuales deben diseñarse directamente en el programa u obtenerse a través de librerías extras.

Por todo lo anterior se diseñó un PROTOTIPO EXPERIMENTAL PARA EL ESCANEO DE OBJETOS EN TRES DIMENSIONES capaz de adquirir datos en coordenadas tridimensionales de un objeto, almacenarlos para su posterior utilización. Además el sistema podrá realizar una representación gráfica del objeto escaneado.

El sistema consistirá en un sensor de distancia y una plataforma giratoria en donde se colocará el objeto a escanear. El sensor medirá la distancia desde el punto de referencia hacia un punto de la superficie del objeto y se procesará para crear una coordenada tridimensional(x, y, z), que será archivada. Este proceso se repetirá moviendo el objeto y/o el sensor hasta terminar con la secuencia de escaneo. El patrón de escaneo será de *circunferencias superpuestas*. Luego se procesaran los datos obtenidos para unir todos los puntos, creando así, una superficie que cubra el contorno de las coordenadas tomadas con el escáner, para así obtener una imagen en 3D del objeto.

II. OBJETIVOS

2.1 OBJETIVO GENERAL

Desarrollar un prototipo experimental que permita el escaneo de objetos, para poder representarlos en un archivo virtual de datos en tres dimensiones, capaz de ser utilizado para diversas aplicaciones.

2.2 OBJETIVOS ESPECÍFICOS

- Adquirir datos del objeto escaneado en coordenadas tridimensionales
- Archivar los datos adquiridos en un formato que pueda ser utilizado por diversas aplicaciones.
- A partir de los datos archivados generar el modelo virtual en tres dimensiones para representar el objeto escaneado.

III. ALCANCES Y LIMITACIONES

3.1 ALCANCES

- El prototipo tomará los puntos de la superficie del objeto necesarios para formar una matriz de datos [*x, y, z*], que será archivada en un formato que pueda ser utilizado por diversas aplicaciones para generar el modelo virtual del objeto escaneado.

- El programa desarrollado será capaz de procesar los puntos almacenados, de manera que se forme una superficie que represente al objeto escaneado.

- Este proyecto establecerá un punto de partida para futuras investigaciones en el área del escaneo de objetos para su representación en tres dimensiones.

3.2 LIMITACIONES

El desarrollo de este proyecto se vio afectado por las siguientes limitaciones:

- Carencia en el país de investigaciones y proyectos previos que aportaran algún tipo de información para la implementación del prototipo desarrollado.

- Carencia en el mercado nacional de sensores adecuados para la aplicación desarrollada.
- Debido a la limitación anterior, fue necesario comprar a través de internet sensores para realizar las pruebas necesarias para determinar la mejor opción para el sensor de distancia, esto redundó en pérdida de tiempo y dinero.

- Costo elevado de sensores láser e infrarrojos apropiados para la aplicación desarrollada.

- Debido a la limitación económica, el prototipo se ha construido, principalmente, con partes mecánicas de diversos aparatos.

Las limitaciones del prototipo experimental para el escaneo de objetos en tres dimensiones son las siguientes:

- Las dimensiones de los objetos a escanear están limitadas por las características físicas del escáner (torque de los motores, dimensiones del escáner, etc.), por tanto los objeto a escanear no deberán ser de más de 10 cm de diámetro, 11.5 cm de altura y con un peso no mayor de una libra.

- El prototipo no es capaz de simular texturas ni colores de la superficie del objeto escaneado.

- Si el objeto esta hecho de varios materiales, la imagen que se obtenga representará un único volumen, independiente de los materiales que lo componen.

- Debido a que el sensor de distancia es mecánico, el tiempo de escaneo es relativamente largo.

- El sensor mecánico tiene diferentes márgenes de error, dependiendo de la superficie del objeto escaneado, así como de su forma.

IV. MARCO TEORICO

4.1 ESCANEO DEL OBJETO

El escaneo en rotatorio consiste en tomar datos de cierto punto del objeto en una secuencia anular de manera que se van formando anillos de datos que luego puestos uno sobre otro asemejan al objeto escaneado.

Hay dos maneras de escaneo rotatorio: Escaneo estático y escaneo dinámico.

- Escaneo rotatorio estático: Cuando el objeto gira sobre un eje y el sensor de distancia esta estático tomando los datos.

- Escaneo rotatorio dinámico: Cuando el objeto esta estático y el sensor de distancia esta girando alrededor del objeto tomando los datos.

En ambos escaneos los datos son los mismos lo que cambia es la forma en que se toman.

En la siguiente figura se muestra como se toman las coordenadas del objeto que se esta escaneando. (En este caso un objeto piramidal)

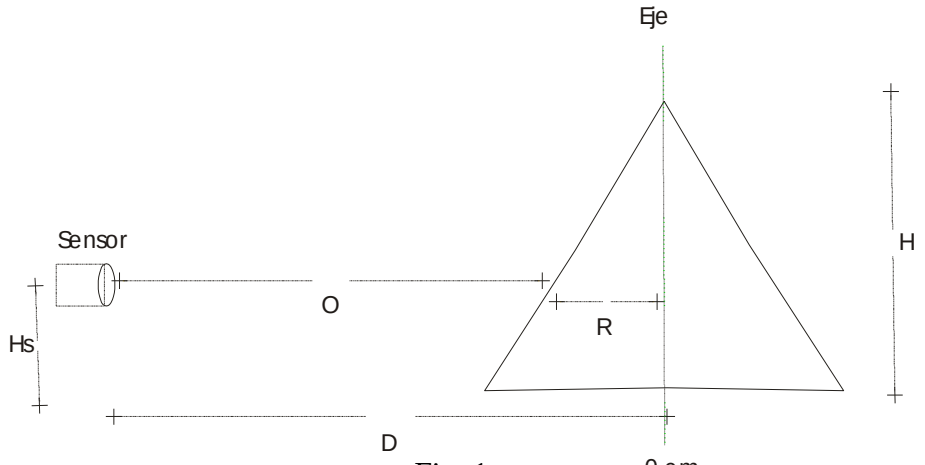


Fig. 1

Al observar la figura se pueden apreciar las siguientes distancias:

O: La distancia del sensor al punto del objeto donde esta enfocado.

R: La distancia desde el eje del objeto hacia el punto que el sensor esta escaneando.

D: La distancia desde el eje del objeto hacia el sensor.

H: La altura del objeto.

Hs: La altura en la que se encuentra el sensor al momento de escanear el punto del objeto.

Al colocar el eje del objeto en el 0 cm de nuestro marco de referencia, sabemos la distancia D. Luego si el sensor mide la distancia O y nosotros sabemos la distancia D, entonces podemos conocer la distancia R que seria el radio del objeto en ese punto.

D-O=R Eq. 1

Si además conocemos la altura del sensor(Hs) obtenemos las coordenadas siguientes:

X=R Eq. 2

Y=Hs Eq. 3

Si a todo esto agregamos el valor de θ el cual es el ángulo del sensor alrededor del objeto según nuestro marco de referencia, obtenemos las siguientes coordenadas:

R, θ,Hs

Donde R es el radio, θ es el ángulo y Hs la altura del punto que se esta escaneando, formando así coordenadas cilíndricas, de manera que si se toman varios puntos del objeto y se grafican cada uno de ellos, obtendremos círculos de puntos que asemejaran la forma del objeto escaneado. Un ejemplo de esto es el mostrado en la figura 2

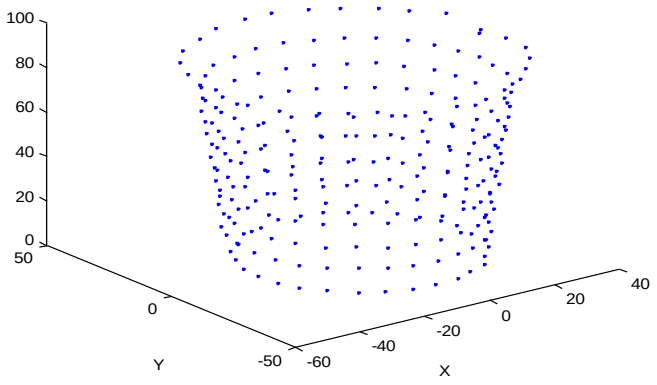


Fig. 2

4.2 SISTEMA DE COORDENADAS TRIDIMENSIONALES

En el espacio euclídeo tridimensional, que es el que usamos casi siempre, son sistemas en los que un punto del espacio P queda determinado como intersección de tres superficies que en dicho punto son perpendiculares (ortogonales) entre sí.

- **COORDENADAS CARTESIANAS.**

Para representar en el espacio euclídeo puntos, líneas. superficies, vectores, etc, utilizamos como referencia tres ejes de coordenadas perpendiculares entre sí tal como vemos en la Fig. 3. Utilizando como referencia estos ejes se pueden definir varios sistemas de coordenadas, de los cuales el más frecuentemente usado es el de coordenadas cartesianas (nombre dado en honor del filósofo y matemático francés René Descartes, 1596-1650). En este sistema un punto P está determinado por la intersección de 3 superficies (en este caso planos) perpendiculares entre sí (ortogonales) (Fig. 4) de ecuaciones $x = x_1$, $y = y_1$, $z = z_1$. Decimos que el punto P tiene como coordenadas: (x_1, y_1, z_1).

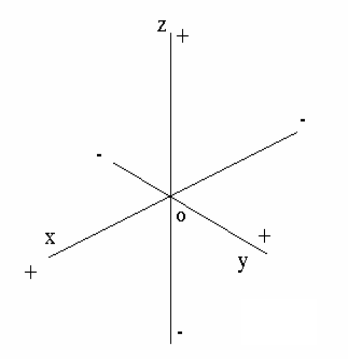


Fig. 3

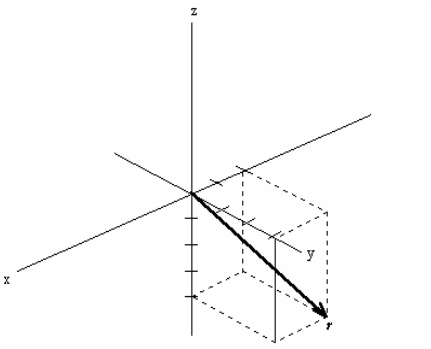


Fig. 4

- **COORDENADAS CILÍNDRICAS.**

En este sistema un punto P del espacio queda determinado por la intersección de 3 superficies: un cilindro cuyo eje es el eje z y su radio r; un semiplano limitado por el eje z y que forma un ángulo ϕ con ele eje x (c.c.w); un plano paralelo al plano x-y y a una distancia z del mismo. A dicho punto le asignamos 3 coordenadas: r, ϕ y z (z es la misma que en coordenadas cartesianas) fig. 5. Los rangos de cada una de ellas son:

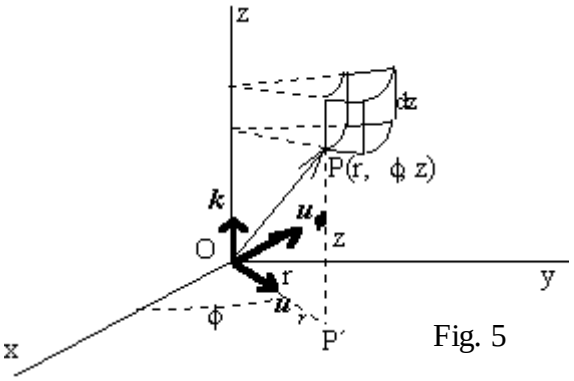


Fig. 5

$r \geq 0;\ 0 \leq \phi \leq 2\pi$ y $-\infty < z < +\infty$ Eq. 4

Si manteniendo r constante, hacemos variar ϕ y z, se generaría una superficie cilíndrica, cuyo eje coincidiría con el eje z de las coordenadas cartesianas rectangulares. Si mantenemos constante ϕ y variamos r y z, tendremos un plano que contendrá al eje z. Finalmente con z constante y ϕ y r variables, tendremos un plano perpendicular al eje z

- **CAMBIO DE COORDENADAS.**

Para transformar las coordenadas de un punto P de un sistema a otro no hay más que utilizar las correspondientes relaciones geométricas y trigonométricas derivadas de las definiciones vistas anteriormente. Así, si queremos expresar las coordenadas cilíndricas de un punto (módulos) (r; ϕ ; z) en función de sus coordenadas cartesianas (x; y; z), las relaciones a utilizar serán:

$$r = \sqrt{x^2 + y^2};\quad \phi = \cos^{-1}\left(\frac{x}{\sqrt{x^2 + y^2}}\right);\quad z = z \qquad \text{Eq. 5}$$

Las coordenadas cartesianas en función de las coordenadas cilíndricas:

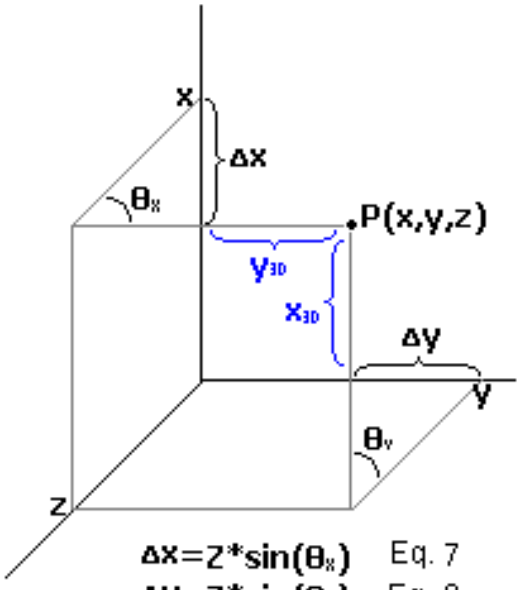
$$x = r\cos\phi;\quad y = r\,\text{sen}\,\phi;\quad z = z \qquad \text{Eq. 6}$$

4.3 REPRESENTACIÓN 3D EN DOS DIMENSIONES

A partir de la matriz Radio-Angulo-Altura, se obtiene los puntos X-Y-Z utilizando la conversión de coordenadas cilíndricas a rectangulares.

Ya con los puntos xyz podremos representar de forma sencilla 3D en 2D eliminando una coordenada, para el caso la coordenada Z.

Siempre se tienen dos dimensiones en la pantalla del computador, al colocar un punto xyz y sus ejes cartesianos en un plano vemos como Z es una diagonal y produce un desplazamiento ΔX y ΔY como se muestra en la figura 6.



$$\Delta x = Z * \sin(\theta_x) \quad \text{Eq. 7}$$

$$\Delta y = Z * \sin(\theta_y) \quad \text{Eq. 8}$$

$$x_{3D} = x - \Delta x \quad \text{Eq. 9}$$

$$y_{3D} = y - \Delta y \quad \text{Eq. 10}$$

Fig. 6

Este desplazamiento se puede calcular utilizando el ángulo que forma la diagonal Z con cada uno de los ejes X y Y y la diagonal Z utilizando Pitágoras:

$$\Delta X = Z * \sin(\theta_x) \qquad y \qquad \Delta Y = Z * \sin(\theta_y)$$

De esta forma tenemos un nuevo valor de X y Y para nuestra representación.

Hay que tener presente que esta es una representación sencilla, pues lo datos no son total mente reales y que en ningún momento se toma la perspectiva del observador que modifica las distancias reales de la coordenada Z.

4.4 ARCHIVO OBJ

El archivo OBJ es un formato de ficheros de objetos 3D estándar creado por Wavefront's Advanced Visualizer. Los ficheros Object están basados en texto soportando tanto geometría poligonal como de forma libre (curvas y superficies).

No es la intención de este documento ocuparse de todos los aspectos del formato de OBJ, puesto que la mayoría de los aspectos del formato no se utilizan a menudo. Se centrará solamente en esos aspectos del formato de OBJ que sean necesarios para almacenar cimas y caras con agrupar y etiquetas materiales.

A continuación se muestra la descripción de OBJ de un cubo.

```
mtllib colors.mtl
v -0.500000 -0.500000  0.500000
v  0.500000 -0.500000  0.500000
v -0.500000  0.500000  0.500000
v  0.500000  0.500000  0.500000
v -0.500000  0.500000 -0.500000
v  0.500000  0.500000 -0.500000
v -0.500000 -0.500000 -0.500000
v  0.500000 -0.500000 -0.500000
g cube
usemtl Color1
f 1 2 4 3
f 3 4 6 5
f 5 6 8 7
usemtl Color2
f 7 8 2 1
f 2 8 6 4
f 7 1 3 5
```

El resultado es el mostrado por la siguiente figura:

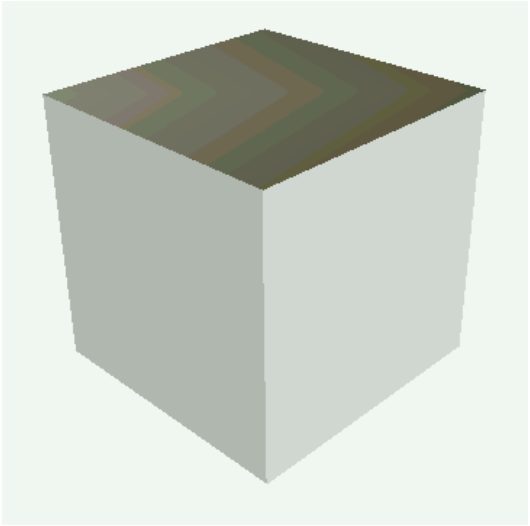


Fig. 7

DESCRIPCIÓN DEL FORMATO OBJ

Una cima es denotada por la bandera "v" al principio de la línea, seguido por las coordenadas de x, y, z de la cima. Cada cima se numera según el orden de la creación. Por lo tanto, la primera cima definida en el archivo es la cima número uno. Una cima se puede definir en cualquier parte en el archivo mientras se defina antes de cualquier cara de la cual sea un miembro.

Una cara es denotada por la bandera "f" al principio de la línea seguida por una lista por lo menos de tres cimas que definan la cara. Las caras se pueden construir a partir de cualquier cima que se haya definido ya en el archivo, y no hay límite superior a cuántas cimas puede contener una cara. Es posible definir las caras que no se exhibirán correctamente. Para evitar esto, todas las cimas que abarcan la cara deben ser uniplanar. La manera más simple de asegurar esto es hacer todos los triángulos de las caras, aunque ésta dará lugar a un archivo más grande.

Nota: La enumeración de la cima comienza en uno, NO en cero. Una cara que procura referirse a la cima 0 dará lugar generalmente "Error fuera de escala".

La bandera "g" al principio de la línea define a un grupo, seguido por el nombre de grupo. Aunque no se hacen caso a los grupos, estos sirven como separadores convenientes, lógicos para los archivos de OBJ que contienen más de un objeto, y son útiles al escribir a lectores de OBJ. Cuando definen a un grupo sigue siendo el grupo actual hasta que definen a otro grupo. Cada cara nueva se hace un miembro del grupo actual. Si no se especifica a ningún

grupo en el archivo, todo en el archivo está en el grupo de defecto. No hay ayuda para los grupos jerarquizados.

Aunque el formato de OBJ no contiene descripciones del material o del color, proporciona una manera de especificar que ciertas caras deben tener diversos materiales. Los materiales son denotados por la bandera del "usemtl", seguida por el nombre del material. Como grupos, los materiales se aplican a todas las caras que siguen la declaración material hasta que se declara otro material.. La localización del archivo material se especifica en la primera línea del archivo OBJ y es denotada generalmente por la bandera "mtllib" seguida por la trayectoria y el nombre del archivo de MTL.

.

Nota: Dos banderas adicionales, "vt" y "vn," se pueden encontrar en algunos archivos OBJ. La bandera "vt" denota una cima de la textura. Las cimas de la textura se determinan como las imágenes de la textura mapeada a la superficie de un pedazo de geometría. La bandera "vn" indica una cima normal. Los normales de la cima sirven para mezclar visualmente sobre los bordes de polígonos individuales dando por resultado una superficie que parece más lisa. Si estas banderas aparecen en un archivo, entonces el formato de la bandera de "f" será alterado como sigue:

f v/vt/vn v/vt/vn v/vt/vn ...

La declaración de "f " sería como sigue:

f 1/1/1 2/2/2 3/3/4 4/4/3...

4.5 FORMATOS DE ARCHIVOS

A continuación se mencionan otros archivos a los cuales se puede transformar un archivo OBJ:

Extensión	Tipo de archivo
*.asc	3D Studio ASC
*.x	Direct X model
*.dxf	AutoCAD DXF
*.cpp	Cpp OpenGL
*.peo, *.geo	Homeworld Geometry
*lwo, *.lw	Ligthwave Object
*.stl	Stereo lithography
*.cob	TrueSpace Object
*.3ds	3D Studio

4.6 CARACTERISTICAS GENERALES DE VRML

- VRML 97: ISO Standard, revisión de VRML 2.0, dic. 1997
- VRML es un lenguaje de descripción de formas y ambientes tridimensionales interactivos
- Un browser VRML es un programa que interpreta archivos.wrl, presenta su contenido al usuario y le permite a éste interactuar con el mundo VRML mediante una interfase
- Los archivos .wrl se pueden cargar localmente o desde Internet
- Un browser VRML puede ser:
    - VRML helper-application
    - VRML plug-in en un browser HTML

- **HERRAMIENTAS NECESARIAS PARA CREAR UN VRML**
    - Un Editor de texto
    - Una aplicación "constructora" o builder de VRML97
    - Un modelador 3D y un conversor a formato VRML97
    - Un generador de formas

- **ESTRUCTURA DE UN ARCHIVO VRML**

Un archivo .wrl contiene:

- Encabezado: #VRML V2.0 utf8
    - V2.0 es la versión
    - utf8 es un standard de caracteres internacionales UCS (Universal Character Set) Transformation Format, 8-bit

- Comentarios: comienzan con el carácter #

- Nodos: describen el contenido de la escena (formas, luces, sonidos, etc.)

<tipo _ nodo> { conjunto de <campo> <valor> }

    - el tipo identifica al nodo (Shape, Light, ElevationGrid, etc.)
    - un campo describe atributos del nodo
        - nombre del campo (height, radius, coordIndex, etc.)
        - tiene un tipo de dato asociado (SFInt32, SFFloat, etc.)
        - tiene un valor por defecto

- **GRAFICACIÓN DE PUNTOS EN VRML**

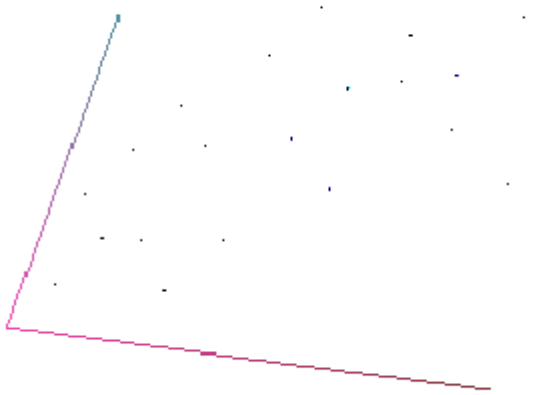


Fig. 8

- El nodo de geometría PointSet crea un conjunto de puntos

```
Shape {
   appearance Appearance {
        material Material{...}
   }
   geometry PointSet {
        coord Coordinate {
            point [
               1.0 1.0 0.0,
               2.0 4.0 0.0,
               . . .
            ]
        }
       . . .
   }
}
```

- **CREACIÓN DE POLÍGONOS BASADOS EN PUNTOS EN VRML**

Fig. 9

- El nodo de geometría IndexedFaceSet crea un conjunto de polígonos.

```
Shape {
     appearance Appearance {
          material Material{...}
     }
    geometry  IndexedFaceSet {
          coord Coordinate {
               point [
                 0.0  0.0  0.0,
                 5.5  5.0  0.88,
                 . . .]
          }
          coordIndex [0,  1,..., -1,
                    0, 12,..., 7, -1,
          .... ]
          solid TRUE
          ccw TRUE
          convex FALSE
    }
}
```

- coord tiene como valor un nodo Coordinate (idem IndexedLineSet)
- Se listan los índices de las coordenadas de las caras en coordIndex. El índice -1 indica fin de polígono, el cual se cierra conectando automáticamente el último índice con el primero.
- solid indica si la forma es sólida, es decir, si las caras internas no se dibujan.
- ccw indica si las caras son en sentido antihorario (counter-clockwise), es decir, si las caras son externas.
- convex indica si las caras son convexas. Las caras cóncavas se parten en múltiples caras convexas.

4.7 VISUAL BASIC

- **HISTORIA**

Microsoft lanzó Visual Basic en 1987. Fue la primer herramienta de desarrollo visual que desarrollo Microsoft, y se hizo para competir con C, C++, Pascal y otros marcas conocidas de lenguajes de programación. Desde el comienzo Visual Basic no fue un éxito. No fue hasta el lanzamiento de la versión 2.0 en 1991 que la gente realmente descubrió el potencial del lenguaje, y con el lanzamiento de la versión 3.0 se convirtió en el lenguaje de programación de más rápido crecimiento en el mercado.

- **QUÉ ES VISUAL BASIC**

Los programadores han experimentado un gran cambio en sus muchos años de programa maquinas. Por ejemplo, lo que se pudo crear en minutos con Visual Basic, tomaría días en Pascal o C++. Visual Basic provee muchos sistemas de herramientas interesantes para ayudar a construir excitantes aplicaciones. Visual Basic provee estas herramientas para hacer la vida mucho más fácil, ya que todo el código difícil esta ya escrito para el usuario.

- **CARACTERÍSTICAS SIGNIFICANTES DEL LENGUAJE**

Visual Basic no es solo un lenguaje de programación, sino también un desarrollador de ambientes completamente gráficos. Este ambiente ayuda al programador que con poca

programación rápidamente experimente útiles aplicaciones de Windows con la habilidad de usar objetos OLE(Object Linking Embedding), tales como las hojas de Excel.

El punto de venta principal de Visual Basic es la facilidad con la que permite crear programas gráficos agradables a la vista con unos pocos códigos del programador.

Al trabajar el programador en el ambiente grafico, varios códigos del programa se genera automáticamente por el programa. El objeto principal de Visual Basic es llamado una **forma.** Cuando se abre un proyecto nuevo, una ventana similar a esta aparece:

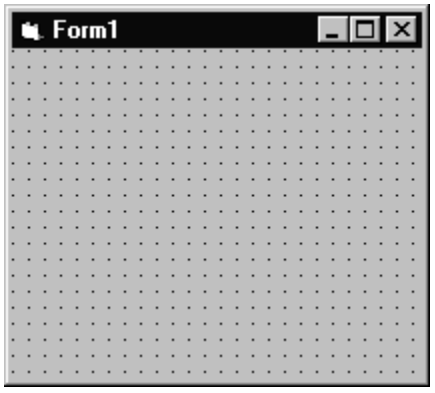


Fig. 10

Esta forma eventualmente se incorpora al programa como una ventana. A esta forma se le añaden controles. Controles son cosas como cajas de texto, cajas de chequeo y botones de comando. Estos se añaden simplemente eligiéndolos de la herramienta de botones e insertándolos en la forma. La herramienta de botones luce como esto:

Fig. 11

Una vez que se creen las formas y los controles, usted puede cambiar las características (el aspecto, estructura, etc.) relacionadas con esos objetos en eso la ventana particular de las características de los objetos. De esta ventana, usted elige la característica que usted desea cambiar de la lista y cambiar sus settings correspondientes. Aquí está un ejemplo de una ventana de las características:

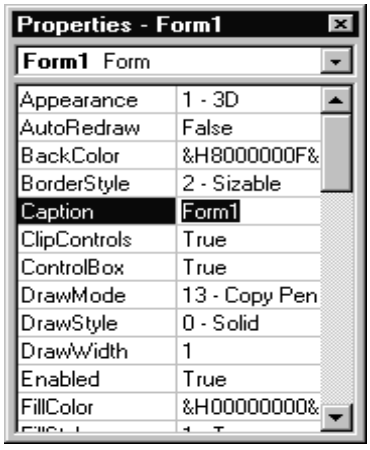


Fig. 12

Finalmente, se puede agregar acontecimientos a sus controles. Los acontecimientos son respuestas a las acciones realizadas en controles. Por ejemplo, en el programa de "Hello World", cuando usted da clic en el botón de comando de encendido en la forma el acontecimiento se acciona que es la salida del mensaje "Hello World" a la pantalla. Todo esto se hace en la ventana de código de Basic:

Fig. 13

Una vez que se ha abierto la ventana de código, se selecciona el object para crear un evento para la acción de presionar el botón. Se puede abrir la caja de códigos para una forma en particular seleccionándola de la ventana de proyectos y seleccionando el botón **View Code.** La ventana de proyectos contiene una lista de proyectos asociados con el proyecto actual. El siguiente es un ejemplo de una ventana de proyecto:

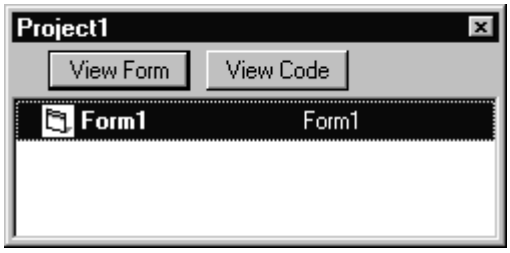


Fig. 14

4.8 BASIC STAMP/PBASIC

- **BASIC STAMP**

Basic Stamp tiene sus antecedentes desde 1979, su creador Chip Gracey tuvo sus primeras experiencias en programación en las computadoras Apple II. Su entusiasmo en el Basic de Apple fue tal que fue mas allá de los experimentos normales pasando a ensayar con interfases a circuitos de electrónica.

Parallax tiene sus orígenes desde 1986 al asociarse Chip Gracey con Lance Walley, los primeros productos tuvieron como base el microcontrolador Intel 8051. Esta empresa se consolidó al diseñar su primer Basic Stamp, en 1992. La base de Basic Stamp son los microcontroladores Microchip serie PIC. Parallax distribuye Basic Stamp en varias versiones, esto dependiendo de las características funcionales que se adapten a la aplicación del usuario.

El Basic Stamp BS2-IC se encuentra en una presentación de tableta de ensamble de 24 pines, igual a un circuito integrado Dual-In-line, excepto para la versión BS2p, en la cual la tableta de ensamble esta presente en configuración de 40 pines.

La razón del éxito de los Basic Stamp es que conjugan alta funcionalidad con un encapsulado muy compacto. Sus 16 pines de entrada/salida permiten la programación de

manera muy versátil y totalmente completa tanto a nivel de bit como nibble, byte y palabra, en cualquier combinación, ya sea entrada o salida.

Potencia e inicialización: Señales PWR, GND, RES, +5Vcc. Este bloque incluye la señal de inicialización para el microcontrolador y subsistemas que complementan el BS2-IC. La señal PWR funciona como entrada para el regulador de voltaje interno, en caso de que vaya a ser necesario interconectar dispositivos adicionales utilizar un regulador de voltaje externo (7805) e interconectar su salida a la patilla +5Vcc y dejar PWR sin conectar. Claro para poder hacer esto, es necesario que se diseñe una fuente de voltaje externa.

Entrada/Salida: Señales P0 a P15. Este juego de pines es totalmente programable. Tanto en modo grupo de pines como individual, asimismo, función entrada o salida. El nombre de estas señales es compatible entre todas las versiones de Basic Stamp, el número dependerá de la versión que se este utilizando. Para el uso del compilador de Basic en la programación es necesario su interconexión a la PC por medio de un cable serial. Sistema de Memoria.

El componente principal del sistema es el microcontrolador Pic16C57. La tableta del Stamp cuenta con una memoria programable de 2Kb utilizada para almacenar los programas. Este es el propósito del chip 24LC16B, el cual es una memoria eléctricamente programable (EEPROM). El sistema de memoria del Basic Stamp

Descripción de las patillas del Basic Stamp BS2-IC, todas sus versiones de 24 pines, incluyendo el JAVELIN.

Pin	Nombre	Descripción
1	Tx	Salida serial. Se conecta el pin 2 del DB9 del serial de la PC (Rx)
2	Rx	Entrada serial. Se conecta al pin 3 del DB9 del serial de la PC (Tx)
3	ATN	Reset activo en alto. Conectado al pin 4 del DB9 del serial de la PC (DTR)
4	GNDS	Tierra de la interfaz serial. Conectado al pin 5 del DB9 del serial de la PC (GND)
5	P0	Pin de E/S 0 Cada pin puede entregar hasta 20mA y consumir 25mA.
6	P1	Pin de E/S 1
7	P2	Pin de E/S 2
8	P3	Pin de E/S 3
9	P4	Pin de E/S 4
10	P5	Pin de E/S 5
11	P6	Pin de E/S 6
12	P7	Pin de E/S 7 Puede agruparse a los pines P0-P7 y P8-P15 como bytes. En conjunto pueden entregar hasta 40mA y consumir 50mA.
13	P8	Pin de E/S 8
14	P9	Pin de E/S 9
15	P10	Pin de E/S 10
16	P11	Pin de E/S 11
17	P12	Pin de E/S 12
18	P13	Pin de E/S 13
19	P14	Pin de E/S 14
20	P15	Pin de E/S 15
21	+5Vcc	Entrada de +5 volts de la fuente de alimentación. Consumo de 7mA en modo normal, 50uA en modo Standby o de bajo consumo de energía.
22	RES	Reset activo en bajo. Una señal de al menos 100mS en nivel bajo es suficiente para resetear el sistema.
23	GND	Tierra de Circuito,
24	PWR	Entrada del regulador de voltaje interno. De +6 a +15 volts, por lo general si se quiere diseñar un sistema portátil su magnitud puede ser de +9 volts o alimentar la patilla 21 con un regulador de voltaje a 5 volts .

Parallax viene trabajando en la fabricación de productos basados en microcontroladores desde hace años y en su metodología obtuvieron excelentes resultados utilizando un

conjunto de dispositivos y programas que ellos mismos habían creado. En 1992 decidieron comercializar esas herramientas y hoy día las emplean profesionales y empresas de todo el mundo. Además, dada su sencillez de manejo se ha impuesto como elemento de trabajo ideal en los centros educativos de todos los niveles: Formación Profesional, IES, Escuelas de Ingeniería Técnica, etc. Parallax desarrolló estas herramientas para que el personal especializado de una empresa, con conocimientos básicos de Electrónica y Programación, fuese capaz de participar e implementar proyectos del área de su especialidad.

- **LENGUAJE PBASIC**

La gran aportación de Parallax, consiste en el lenguaje que utiliza para la programación de sus sistemas microcontroladores, soportados por un excelente hardware con un funcionamiento muy fiable en las condiciones más duras. Dicho lenguaje, El PBASIC, es el más fácil del mundo ya que es una derivación del tradicional BASIC, aunque orientado a la explotación de los recursos y operatividad de los microcontroladores.
En la sección de anexos se encuentra un set de instrucciones de PBASIC.
En las páginas de Internet de Parallax (www.parallaxinc.com) se puede encontrar una amplia información. Además puede obtener libremente el software para el PC y bajarse varios tutoriales en español sobre experiencias muy interesantes que facilitan el aprendizaje.

V. DISEÑO Y DESARROLLO DEL PROTOTIPO

5.1 MECÁNICA

5.1.1 MECANISMO SENSOR DE DISTANCIA

- **VISTAS**

Isométrico

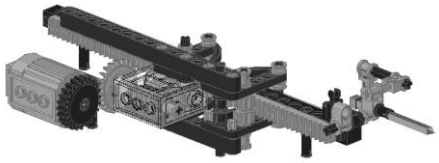


Fig. 15

Frontal

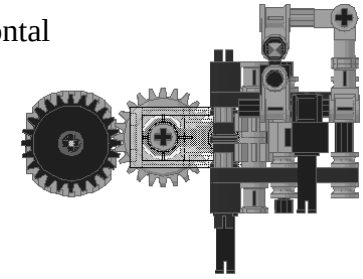


Fig. 16

Derecha

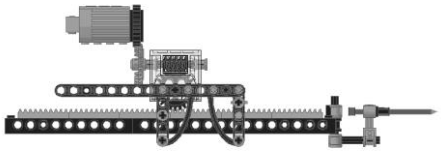


Fig. 17

Izquierda

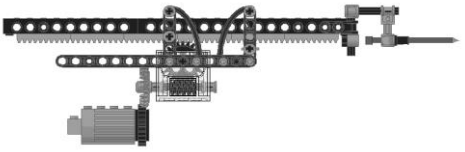


Fig. 18

Arriba

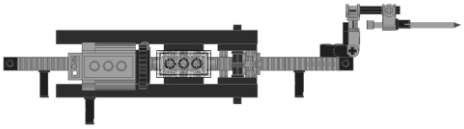


Fig. 19

Abajo

Fig. 20

- **SISTEMA REDUCTOR DEL SENSOR DE DISTANCIA:**

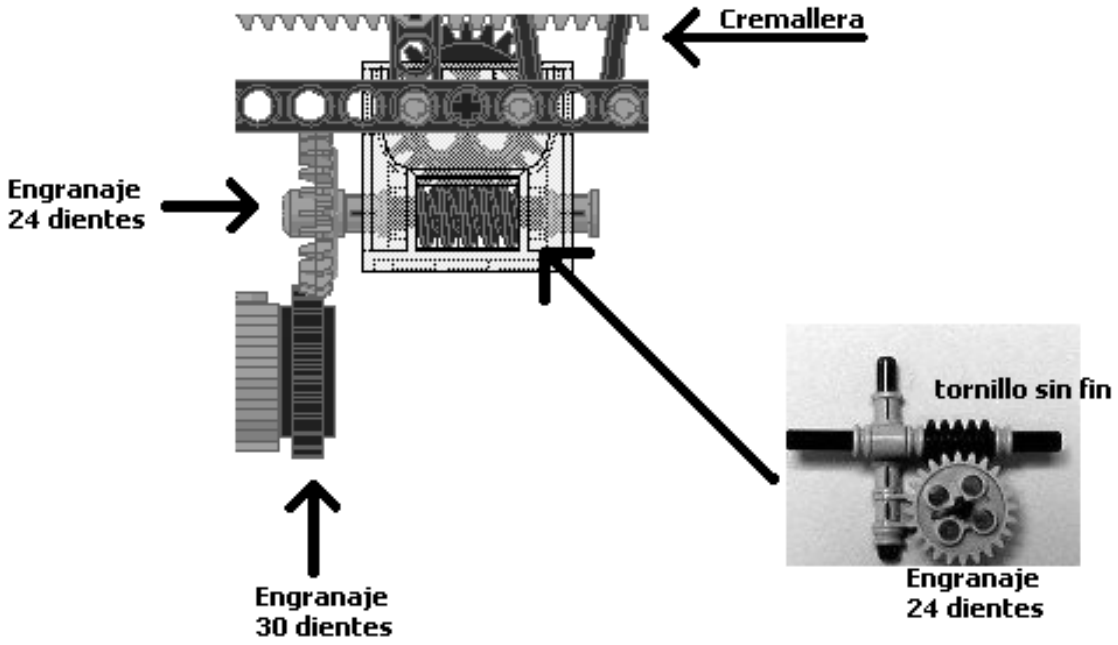


Fig. 21

Con el cual se obtiene un factor de multiplicación de 30 sobre 24 en el primer par de engranajes (engranaje 30 dientes – engranaje 24 dientes), y luego un factor de reducción de 1 sobre 24 en el segundo par de engranajes (engranaje 24 dientes- tronillo sin fin), lo que nos da un resultado de reducción de velocidad igual a:

$$30/24 * 1/24 = 0.0520833333,$$

Y un factor multiplicación de fuerza igual a: 19.2

Así junto con el sistema de cremallera obtenemos la siguiente relación de desplazamiento:

Paso del motor: 3.75º

Radio de engranaje de 30 dientes: 14.325 mm

Con cada paso se obtiene un desplazamiento de: 14.325 * 3.75º = 0.9375690576 mm

Con un factor de reducción de 0.05208333333 el sensor se desplazara *0.0488317217 mm* por cada paso del motor, por tanto, se necesitaran de 2047.8491526136 pasos para lograr una distancia de 100 mm la cual corresponde al máximo desplazamiento del sensor.

5.1.2 PLATAFORMA GIRATORIA.

- **VISTA**

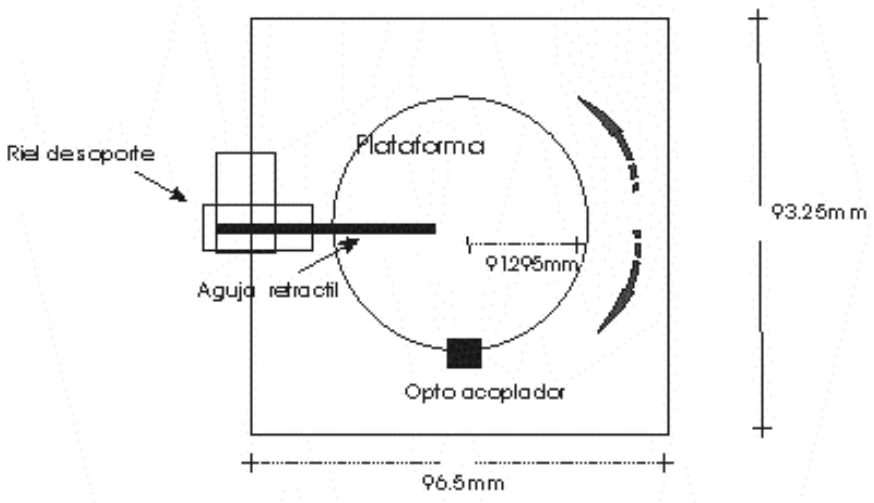


Fig. 22

- **SISTEMA REDUCTOR DE MOVIMIENTO CIRCULAR:**

Radio del motor = 5.75 mm

Radio de plataforma = 91.925 mm

Factor de reducción: 5.75 / 91.925 = 0.0625509927

Con cada paso se obtiene un desplazamiento en el motor de: 3.75º

El resultante desplazamiento angular de la plataforma giratoria es:

0.0625509927 * 3.75º = 0.2345662226º

Para completar 360º se necesitan 1534.7478251969 pasos.

5.1.3 RIEL DE SOPORTE VERTICAL.

- **VISTA**

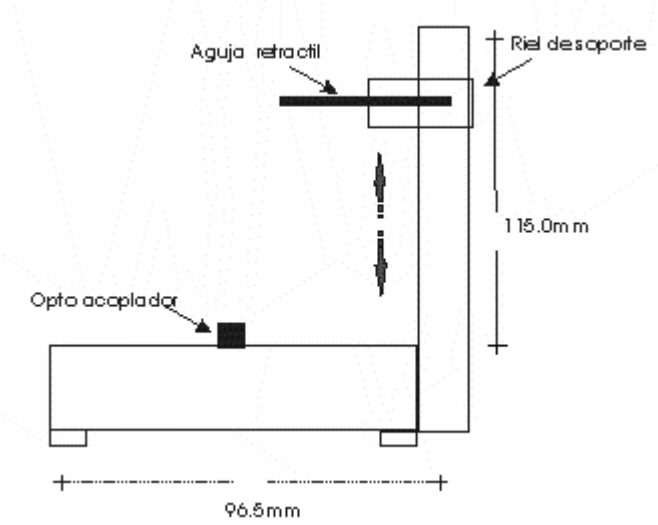


Fig. 23

- **SISTEMA REDUCTOR DE MOVIMIENTO VERTICAL:**

½ Paso del motor: 1.875º

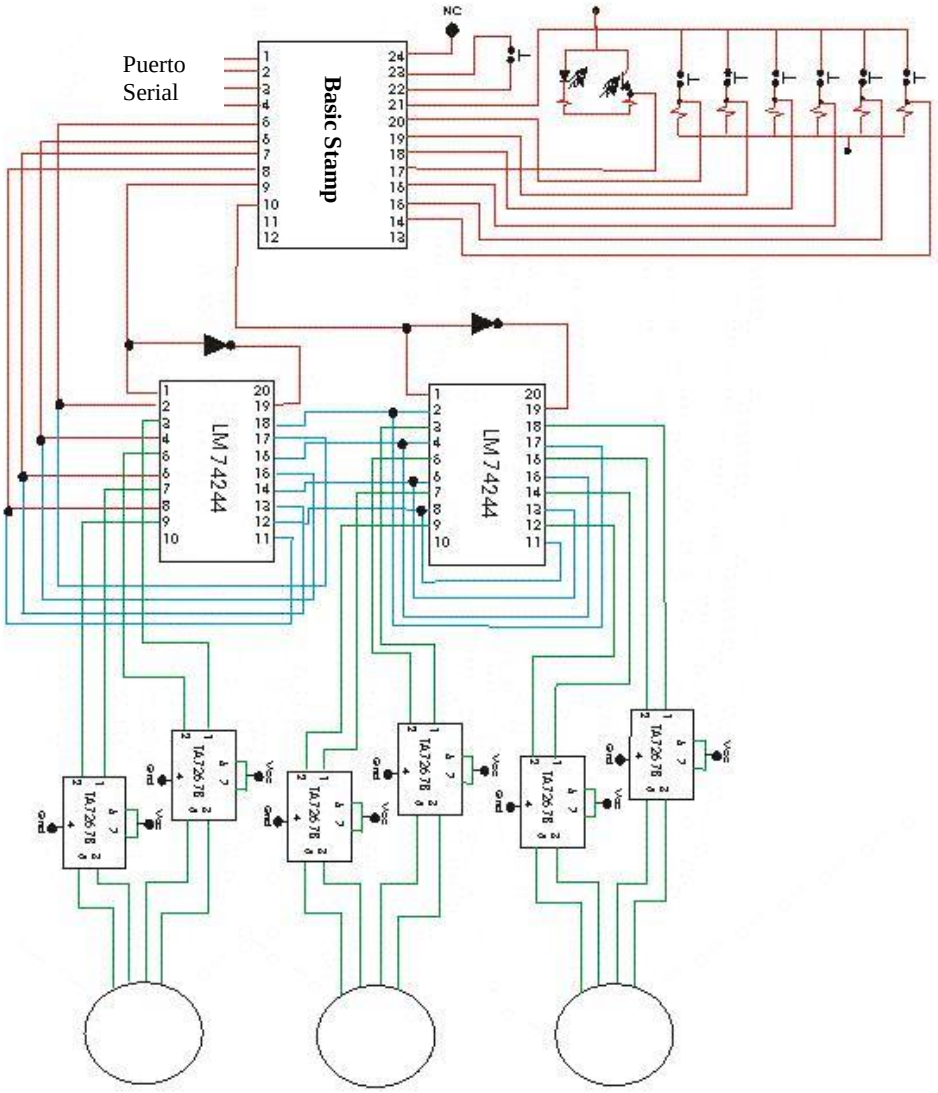
Radio del motor: 5.75 mm

Con cada ½ paso se obtiene un desplazamiento de: 5.75 * 1.875º = 0.18816831 mm vertical.

Por tanto para lograr el máximo desplazamiento vertical, que es igual a 215mm, se necesitaran de 1142.5940956796 medios pasos.

5.2 ELECTRÓNICA

- **DIAGRAMA DEL CIRCUITO.**



Motores Paso

Fig. 24

- **DESCRIPCIÓN DEL CIRCUITO**

El Microprocesador Basic Stamp es el encargado de dirigir la rutina de escaneo y consta de 24 pines tal como se muestra a continuación.

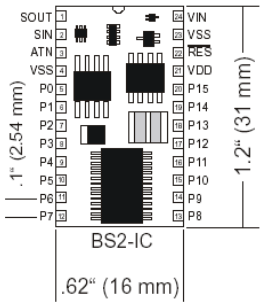


Fig. 25

En tabla siguiente se detalla la función de cada uno de los pines.

Pin	Nombre	Descripción
1	SOUT	Salida serial de datos
2	SIN	Entrada serial de datos
3	ATN	Atención: para programación desde la PC
4	VSS	Tierra del sistema
5-20	P0-P15	Puertos 0-15
21	VDD	Alimentación de 5 voltios
22	RES	Reset
23	VSS	Tierra del sistema
24	VIN	Entrada de alimentación irregulada (5.5-15.VDC)

Para la rutina de escaneo se han asignado las siguientes funciones a los pines del Basic Stamp.

Pin	Asignación
1-4	Transmisión serial hacia la PC
5-8	Transmisión hacia los motores paso
9 y 10	Múltiplexación de motores (elegimos que motor se mueve)
11-13	No se utilizan
14	Brazo mecánico en su posición mas alta
15	Brazo mecánico en su posición más baja
16	Brazo mecánico extendido al máximo
17	Brazo mecánico contraído al máximo
18	Aguja toca objeto
20	Sensor de 360º (Sensado por medio de un sensor óptico)
21	alimentación de 5 voltios regulada
22-23	Reset del programa

Con los pines 5, 6, 7 y 8 del Basic Stamp mandamos la secuencia de movimiento de los motores.

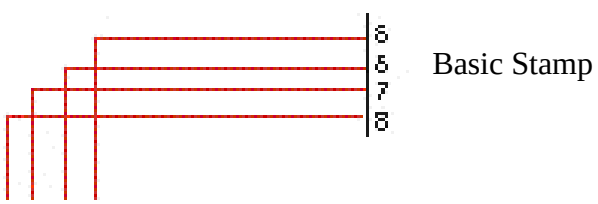


Fig. 26

La tabla siguiente nos muestra la secuencia de los motores.

Pines	Codigo1	Codigo2	Codigo3	Codigo4	Codigo5	Codigo6	Codigo7	Codigo8
5	1	1	1	1	1	0	0	0
6	0	0	0	0	0	0	1	1
7	1	0	0	0	1	1	1	0
8	0	0	1	0	0	0	0	0

Para elegir a que motor se va a enviar los datos, se ocuparon dos integrados LM74244. A continuación se muestra su diagrama interno:

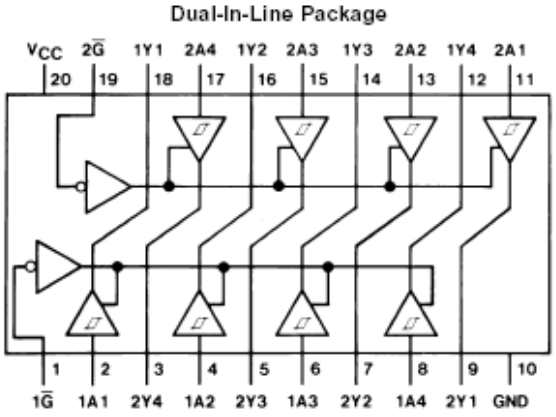
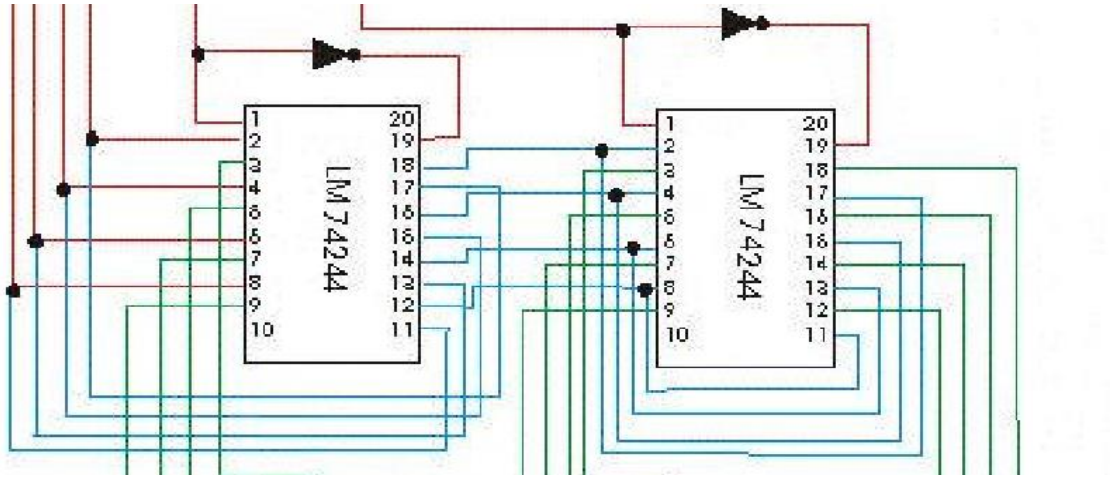


Fig. 27

BASIC STAMP

Pines 5 al 8 Pin 9 Pin10



TA7267B 1 TA7267B 2 TA7267B 3

Fig. 28

Como se puede observar en el diagrama interno del 74244(Fig. 27) las entradas de los pines 1 y 19 son negadas. De manera que para elegir que se activen las salidas 2A del integrado se tiene que tener un cero lógico en el pin 1.

Con las compuertas NOT negamos los impulsos de los pines 9 y 10 para poder habilitar las salidas 1A de los integrados. (Fig. 28)

De manera que los pines 9 y 10 controlan la multiplexacion de impulsos a los motores.

Con la tabla siguiente podemos ver cual es la secuencia de los pines para elegir el motor que se quiere mover.

Pin	Motor 1	Motor 2	Motor 3	Ninguno
9	1	0	0	1

10	0	1	0	1

Luego que se elige a que motor se enviara los pulsos estos pasan a través de los TA7267B, los cuales son Puentes H que amplifican la corriente hacia los motores tal como se muestra en la siguiente figura.

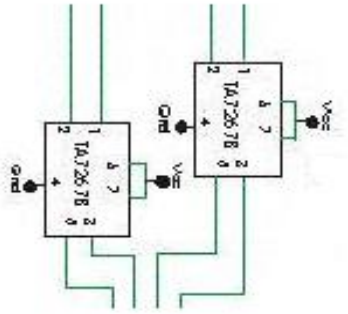


Fig. 29

A continuación se muestra el diagrama interno del TA7267B.

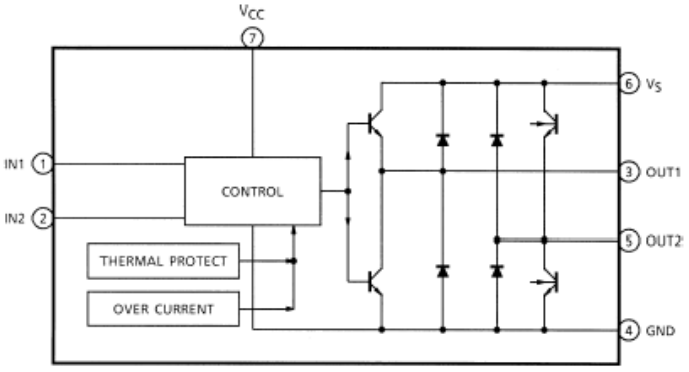
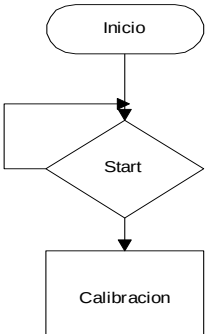


Fig. 30

5.3 PROGRAMACIÓN



1

Fig. 31

- **Flujograma**

- **EXPLICACIÓN FLUJOGRAMA**

1. START: mientras no se presione el pulsador START, el proceso de escaneo no da inicio.
2. Una vez presionado el pulsador START se comienza el proceso de escaneo con la calibración del sensor.
3. Luego de la calibración la aguja se posiciona en Z = 0, la plataforma gira hasta quedar en $\theta = 0°$.
4. La aguja comienza a extenderse
5. Si la aguja toca la superficie del objeto pasa a tomar el dato de posición; si no pasa a preguntarse si llegó a su extensión máxima.
6. Si la aguja llegó a su extensión máxima pasa a retraer aguja; si no la aguja continua extendiéndose.
7. Cuando la aguja toca la superficie del objeto se hacen las conversiones matemáticas para obtener las coordenadas de ese punto y se almacena.
8. Si la aguja tocó a la superficie o llegó a su extensión máxima se retrae.
9. una vez retraída la aguja la plataforma gira 'n' grados (dependerá de la precisión)
10. Si la plataforma ha completado un giro completo, es decir 360°, pasará subir aguja; si no pasa a extender aguja.
11. Cuando ha girado 360° la aguja sube 'x' milímetros (dependerá de la precisión).
12. Si la aguja llega a su altura máxima finaliza proceso de escaneo.

- **PROGRAMA MICROCONTROLADOR BASIC STAMP**

```
'{$STAMP BS2sx}
DIRA=%1111
DIR4=%1
DIR5=%1
SecPAP VAR NIB(8)
Index VAR NIB
indexG VAR NIB
indexH VAR NIB
indexA VAR NIB
counterH VAR WORD
counterG VAR WORD
counterA VAR WORD
Acalib VAR WORD
Hcalib var WORD
i VAR BYTE
x VAR NIB


secpap(0)=%1010
secpap(1)=%1000
secpap(2)=%1001
secpap(3)=%0001
secpap(4)=%0101
secpap(5)=%0100
secpap(6)=%0110
secpap(7)=%0010

perate:

perateK:                        'esperando orden de inicio de escaneo

if in13=0 then perateK
PAUSE 500

'<<<<<<<<<<<<<CALIBRACION Y PUNTOS INICIALES>>>>>>>>>>>>>

out4=1                          'seleccionando A
iniA0:
IF in15=1 then iniA1            'llevando el sensor a su punto final, para calibracio
        index=index-2
        x=5
        GOSUB movingFP
        if in13=1 then perateI  'saltamos a darle tiempo para colocar pin de calibracion
GOTO iniA0

perateI:                        'perdiendo el tiempo
PAUSE 1000
perateL:
SEROUT 16, 17405, ["cal"]
        OUT5=%1
        PAUSE 50
        OUT5=%0
        PAUSE 50
if in13=0 then perateL
```

```
PAUSE 500
goto iniA0                              'regresamos a punto final

iniA1:
IF in14=1 then iniA2                    'llevando el sensor a su punto inicial, calculo de calibracion
        index=index+2
        counterA=counterA+1
        x=5
        GOSUB movingFP
GOTO iniA1

iniA2:
Acalib=2783-counterA
indexA=index                            ' Paso A


out4=0                                  'seleccionando H y G
out5=0                                  'seleccionando H
iniH0:
IF in10=1 then iniH                     'llevando el sensor al punto inicial abajo
        index=index-1
        x=12
        GOSUB movingHP

GOTO iniH0
indexH=index                            ' Paso H
iniH:                                   'H inicial
counterH=0

out5=1                                  'seleccionando G
iniG0:
if in12=0 then iniG                     'llevando plataforma a 0 grados
        index=index-2
        x=5
        GOSUB movingFP
GOTO iniG0


'<<<<<<<<<<<<<<<<<<<<<<RUTINA DE ESCANEO>>>>>>>>>>>>>>>>>>>>>>>
iniG:
                                        'G inicial, 0 grados
GOSUB pasoop                            ' Paso optoacoplador


Gini:

out4=1

                                        'RUTINA DE DISTANCIA
index=indexA
counterA=Acalib
GOSUB distance
indexA=index

out5=1
                                        'activando motor G
```

```
index=indexG
FOR i = 0 to 49
        index=index-2
        counterG=counterG+1
        x=10
        GOSUB movingFP
        indexG=index
        if in12=0 then grado0           'llegando a 0 grados
NEXT

GOTO Gini

grado0:
GOSUB pasoop                            'paso de optoacoplador

out5=0
counterG=0
                                                'activando motor H
index=indexH
FOR i = 0 to 24
        index=index+1
        counterH=counterH+1
        x=14
        GOSUB movingHP
        indexH=index
        if in11=1 then perate           'llegando a H max
NEXT

GOTO Gini


                                '>>>>>>RUTINA DE MEDICION DE DISTANCIA<<<<<<<<
distance:

out4=1                                  'seleccionando A
finA:
IF in15=1 then inicAd                   'llevando el sensor a su punto final
        index=index-2
        counterA=counterA+1
x=5
        GOSUB movingFP


GOTO finA

inicAd:

'*************************TRANSMISION SERIAL A VB*****************************
SEROUT 16, 1021, ["dis"]
pause 2
SEROUT 16, 1021, [DEC counterA,"A",DEC counterG,"G",DEC counterH,"H"]                   'transmision
de datos
pause 350
        'tiempo para transmisión
'*******************************************************************************
```

```
inicA:
IF in14=1 then yaA                      'llevando el sensor a su punto inicial
        index=index+2
        counterA=counterA-1
x=5
        GOSUB movingFP

GOTO inicA
yaA:

out4=0
RETURN

                                        '>>>>>PASO DEL OPTOACOPLADOR<<<<<<<
pasoop:

FOR i = 0 to 49
        index=index-2
        counterG=counterG+1
        GOSUB movingFP
NEXT
        indexG=index

RETURN

                                        '>>>>RUTINA PARA MOVER MOTOR HALF step<<<<
movingHP:

if index<>8 then elsei                  'solo cuatro pasos
        index=0
elsei:
if index<>15 then endifi        'si hay un -1
        index=7
endifi:

OUTA=SecPAP(Index)                      'mandando dato a los motores

PAUSE x
        outA=0

RETURN
                                        '>>>>RUTINA PARA MOVER MOTOR FULL step<<<<
movingFP:

if index<>8 then elseif                 'solo cuatro pasos
        index=0
elseif:
if index<>14 then endifif       'si hay un -1
        index=6
endifif:

OUTA=SecPAP(Index)                      'mandando dato a los motores

PAUSE x
        outA=0
```

```
RETURN
```

- **PROGRAMA VISUAL BASIC**

- **Orden de comunicación con microcontrolador:**

```
Private Sub BtnRead_Click()
ReDim dataROH(1 To 10000, 0 To 2), dataXYZ(1 To 10000, 0 To 2)
BtnNoRead.Enabled = True
BtnRead.Enabled = False
BTNwfile.Enabled = False
BTNrfile.Enabled = False
MSComm1.PortOpen = True
Reading = True
i = 1
Do While Reading
   DoEvents

   If MSComm1.InBufferCount = 3 Then
      DatoRecibido = MSComm1.Input
      MSComm1.InBufferCount = 0
      Label0.Caption = DatoRecibido
      If DatoRecibido = "dis" Then DataDis 'ADQUISICION DE DATOS DE SCANER
   End If
Loop
```

```
End Sub
```

- **Rutina de lectura de dato serial y calculo de coordenadas XYZ:**

```
Private Sub DataDis()
Dataring = True
DataIn = False
Timer1.Enabled = True
Do While Dataring
DoEvents
Loop
CalcXYZ     'CALCULO DE COORDENADAS CILINDRICAS Y RECTANGULARES
End Sub
```

- **Temporizador para sincronización comunicación microcontrolador - PC:**

```
Private Sub Timer1_Timer()
If DataIn Then
DatoRecibido = MSComm1.Input
label1.Caption = DatoRecibido
MSComm1.InBufferCount = 0
Dataring = False
Timer1.Enabled = False
End If
DataIn = True
End Sub
```

- **Cálculo de coordenadas cartesianas:**

```
Private Sub CalcXYZ()
tempO = ""
j = 1
Do While Mid(DatoRecibido, j, 1) <> "A"
   tempO = tempO & Mid(DatoRecibido, j, 1)
```

```
   j = j + 1
Loop
j = j + 1
dataROH(i, 0) = (2783 - Val(tempO))
If dataROH(i, 0) < 0 Then dataROH(i, 0) = 0
dataROH(i, 0) = dataROH(i, o) / 2783 * 100 'mm, radio
tempO = ""
Do While Mid(DatoRecibido, j, 1) <> "G"
   tempO = tempO & Mid(DatoRecibido, j, 1)
   j = j + 1
Loop
j = j + 1
dataROH(i, 1) = Val(tempO) / 1494 * 360 'grados
tempO = ""
Do While Mid(DatoRecibido, j, 1) <> "H"
   tempO = tempO & Mid(DatoRecibido, j, 1)
   j = j + 1
Loop
dataROH(i, 2) = Val(tempO) / 1016 * 215 'mm, altura
dataXYZ(i, 2) = dataROH(i, 2)   'Z
dataXYZ(i, 0) = dataROH(i, 0) * Cos(dataROH(i, 1) * pI / 180)   'X
dataXYZ(i, 1) = dataROH(i, 0) * Sin(dataROH(i, 1) * pI / 180)   'Y
Debug.Print dataROH(i, 0), dataROH(i, 1), dataROH(i, 2)
Debug.Print dataXYZ(i, 0), dataXYZ(i, 1), dataXYZ(i, 2)
i = i + 1
End Sub
```

- **Escritura de archivos TXT:**

```
Private Sub BTNwfile_Click()
nameFile = InputBox("Nombre de archivo:", "SAVE...", "e3d")
Label2.Caption = nameFile
Label3.Caption = "guardando archivo..."
ProgressBar1.Max = 10000
DoEvents
ProgressBar1.Visible = True
```

```
BTNwfile.Enabled = False
BtnRead.Enabled = False
Open "c:\e3d\" & nameFile & "roh.txt" For Output As #1
Open "c:\e3d\" & nameFile & "xyz.txt" For Output As #2
Open "c:\e3d\" & nameFile & "x.txt" For Output As #3
Open "c:\e3d\" & nameFile & "y.txt" For Output As #4
Open "c:\e3d\" & nameFile & "z.txt" For Output As #5

For II = 1 To 10000
ProgressBar1.Value = II
   Print #1, Str(dataROH(II, 0)) & " " & Str(dataROH(II, 1)) & " " & Str(dataROH(II, 2))
   Print #2, Str(dataXYZ(II, 0)) & " " & Str(dataXYZ(II, 1)) & " " & Str(dataXYZ(II, 2))
Print #3, (dataXYZ(II, 0))
Print #4, (dataXYZ(II, 1))
Print #5, (dataXYZ(II, 2))

Next
Close #1
Close #2
Close #3
Close #4
Close #5
BTNwfile.Enabled = True
BtnRead.Enabled = True
ProgressBar1.Visible = False
Label3.Caption = ""
End Sub
```

- **Lectura de archivos TXT, necesarios para representación en 3d dentro de Visual Basic:**

```
Private Sub BTNrfile_Click()
Frame1.Enabled = True
```

```
BTNrfile.Enabled = False
BTNwfile.Enabled = False
BtnRead.Enabled = False
ReDim dataXYZ(1 To 10000, 0 To 2)
nameFile = InputBox("Nombre de Archivo:", "OPEN...", "e3d")

Label2.Caption = nameFile
Label3.Caption = "cargando archivo..."
ProgressBar1.Max = 10000

DoEvents
ProgressBar1.Visible = True
Open "c:\e3d\" & nameFile & "xyz.txt" For Input As #1
For II = 1 To 10000
'ProgressBar1.Value = II
Line Input #1, Datoleido
j = 2
Do While Mid(Datoleido, j, 1) <> " "
j = j + 1
Loop
tempO = Left(Datoleido, j - 1)
dataXYZ(II, 0) = Val(tempO)
j = j + 2
Do While Mid(Datoleido, j, 1) <> " "
j = j + 1
Loop
dataXYZ(II, 1) = Val(Mid(Datoleido, Len(tempO) + 1, j - Len(tempO) - 1))
dataXYZ(II, 2) = Val(Right(Datoleido, Len(Datoleido) - j))
Debug.Print dataXYZ(II, 0), dataXYZ(II, 1), dataXYZ(II, 2)
If dataXYZ(II, 0) = 0 And dataXYZ(II, 1) = 0 And dataXYZ(II, 2) = 0 Then GoTo FinFile
Next
FinFile:
Debug.Print II
MaxDataII = II
Close #1

Label3.Caption = ""
ProgressBar1.Visible = False
```

```
BTNrfile.Enabled = True
BTNwfile.Enabled = True
BtnRead.Enabled = True
End Sub
```

- **Calculo y presentación en 3D dentro de Visual Basic:**

```
Private Sub BTN3D_Click()
Dim dataXY3D() As Single, Z1 As Single, Z2 As Single
ReDim dataXY3D(1 To MaxDataII, 1 To 2)
Picture1.Cls
For II = 1 To MaxDataII
dataXY3D(II, 1) = dataXYZ(II, 0) - dataXYZ(II, 1) * Sin(tetaX * pI / 180)
dataXY3D(II, 2) = dataXYZ(II, 2) - dataXYZ(II, 1) * Sin(tetaY * pI / 180)
Next
If OPTpoint Then
   For II = 1 To MaxDataII
   Picture1.PSet ((dataXY3D(II, 1) + OffsetX), (flipVAR - (dataXY3D(II, 2) + OffsetZ))), RGB((II + 25) / (MaxDataII / 3) * 225, (II + 25) / (MaxDataII / 3 * 2) * 225, (II + 25) / MaxDataII * 225)
   Next
ElseIf OPTline Then
   II = 1
   For II = 1 To MaxDataII - 1
   i = II
   Z1 = dataXYZ(II, 2)
   Z2 = Z1
      Do While Z1 = Z2
      II = II + 1
      Picture1.Line ((dataXY3D(II, 1) + OffsetX), (flipVAR - (dataXY3D(II, 2) + OffsetZ)))-((dataXY3D(II - 1, 1) + OffsetX), (flipVAR - (dataXY3D(II - 1, 2) + OffsetZ))), RGB((II + 25) / (MaxDataII / 3 * 2) * 225, (II + 25) / (MaxDataII) * 225, (II + 25) / (MaxDataII / 3) * 225)
      Z2 = dataXYZ(II + 1, 2)
      Loop
   Picture1.Line ((dataXY3D(II - 1, 1) + OffsetX), (flipVAR - (dataXY3D(II - 1, 2) + OffsetZ)))-((dataXY3D(i, 1) + OffsetX), (flipVAR - (dataXY3D(i, 2) + OffsetZ))), RGB((II + 25) / (MaxDataII / 3 * 2) * 250, (II + 25) / (MaxDataII) * 225, (II + 25) / (MaxDataII / 3) * 225)
```

```
   Next
Else
   II = 1
   For II = 1 To MaxDataII - 1
   i = II
   Z1 = dataXYZ(II, 2)
   Z2 = Z1
      Do While Z1 = Z2
      II = II + 1
      Picture1.Line ((dataXY3D(II, 1) + OffsetX), (flipVAR - (dataXY3D(II, 2) + OffsetZ)))-((dataXY3D(II - 1, 1) + OffsetX), (flipVAR - (dataXY3D(II - 1, 2) + OffsetZ))), RGB((II + 25) / (MaxDataII / 3 * 2) * 225, (II + 25) / (MaxDataII) * 225, (II + 25) / (MaxDataII / 3) * 225)
       Picture1.PSet ((dataXY3D(II, 1) + OffsetX), (flipVAR - (dataXY3D(II, 2) + OffsetZ))), vbRed
      Z2 = dataXYZ(II + 1, 2)
      Loop
   Picture1.Line ((dataXY3D(II - 1, 1) + OffsetX), (flipVAR - (dataXY3D(II - 1, 2) + OffsetZ)))-((dataXY3D(i, 1) + OffsetX), (flipVAR - (dataXY3D(i, 2) + OffsetZ))), RGB((II + 25) / (MaxDataII / 3 * 2) * 250, (II + 25) / (MaxDataII) * 225, (II + 25) / (MaxDataII / 3) * 225)
   Picture1.PSet ((dataXY3D(II - 1, 1) + OffsetX), (flipVAR - (dataXY3D(II - 1, 2) + OffsetZ))), vbRed
   Next

End If
i = 1
End Sub
```

VI. FUNCIONAMIENTO

6.1 VENTANA DEL PROGRAMA

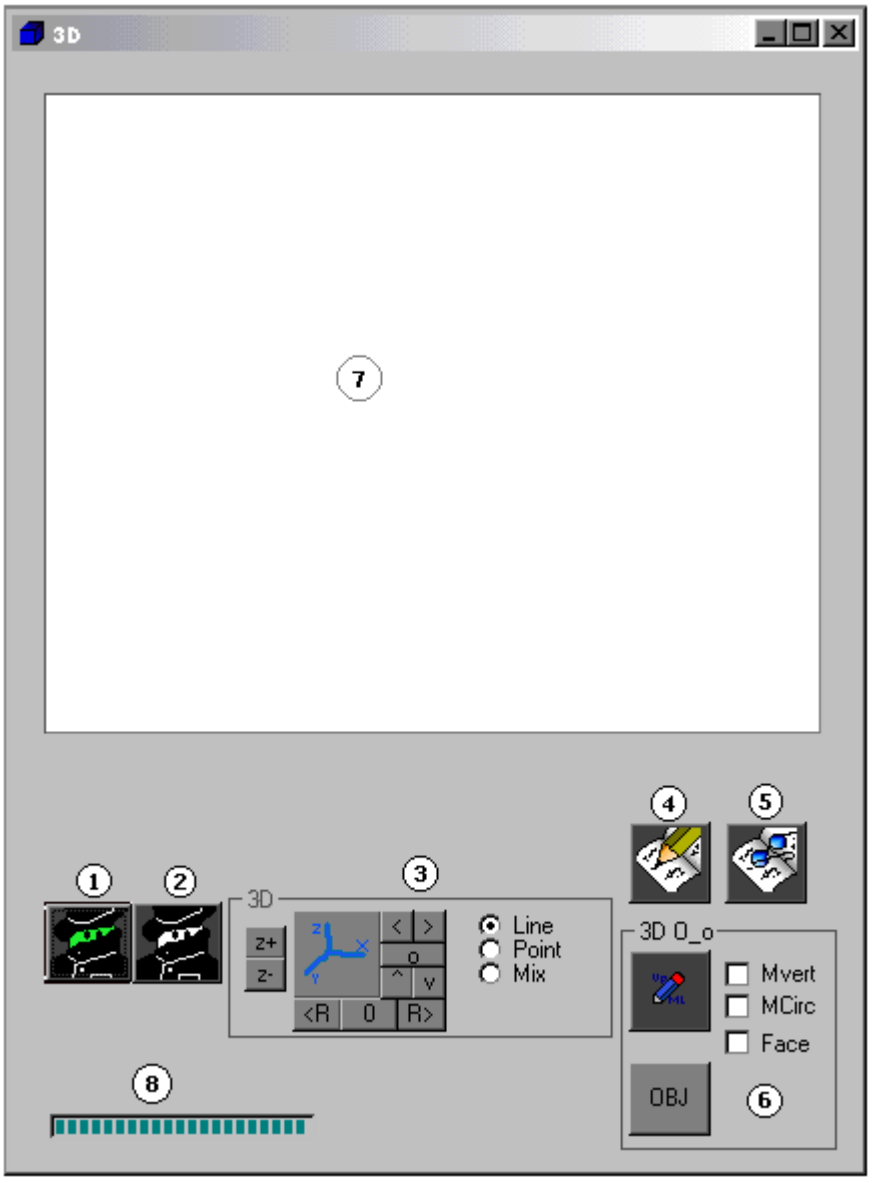


Fig. 32

La figura 32 muestra la ventana del programa que controla el Prototipo experimental para el escaneo de objetos en tres dimensiones.

A continuación se explica cada una de sus partes, según su numeración:

1- Inicio de escaneo: El computador envía el mando al scanner 3D para el inicio del escaneo, el cual realiza su calibración respectiva y luego el escaneo y transmisión de datos al computador para su procesamiento.
2- Fin de escaneo: Se envía la orden de finalización de escaneo al scanner 3D y el computador presenta la opción de guardar los datos recibidos en un archivo de texto para su posterior uso.
3- Representación en 2D de los datos escaneados 3D, con la posibilidad de aumentar la imagen, girar la imagen, dibujar puntos o líneas para una mejor apreciación.
4- Guardar TXT: Almacenamiento de archivos *.txt para el posterior uso. Se almacenan datos en coordenadas cilíndrica y en coordenadas rectangulares.
5- Lectura TXT: Lectura de archivos *.txt en coordenadas cilíndricas, para representación en 2D y escritura de archivos *.wrl y *.obj
6- Escritura de archivos de imágenes tridimensionales: *.wrl (VRML) y *.obj
7- Imagen 2D del objeto escaneado en 3D
8- Barra de progreso

6.2 CAPTURA DE PUNTOS

Para la rutina de escaneo nuestro sistema consta de tres partes principales: una plataforma giratoria, un riel de soporte, y una aguja retráctil.

El movimiento de cada una de las partes es controlado por medio motores pasos, uno para mover la plataforma, el otro para mover la guja de arriba a bajo por medio del riel y el tercero para extender y retraer la aguja.

El objetivo es que la aguja toque cada ciertos grados el objeto a escanear y que por medio de un micro-switch el microcontrolador sepa cuando la aguja ha hecho contacto. Para rotar el objeto se utiliza la plataforma giratoria y para subir y bajar la aguja se usa el riel de soporte que consiste en una banda acoplada a un motor paso sobre la cual esta la aguja con un sistema que le permite extenderse y retraerse

Para empezar el proceso de escaneo, la aguja tiene que estar completamente retraída y en la posición más baja. Por esto al principio se da una mini-rutina donde la aguja se coloca en la posición inicial y se retrae completamente, a la vez que la plataforma rota hasta llegar a lo que seria 0º que seria nuestro punto de partida para que comience a escanear.

Luego al comenzar el proceso la plataforma girara Xº y luego la aguja comenzara a avanzar hacia el centro de la plataforma hasta que haga contacto con algo que active el micro-switch o hasta que llegue a su alcance máximo; cualquiera de los dos eventos hará que la

aguja se retraiga y la plataforma girara nuevamente Xº y seguirá nuevamente el proceso hasta que se terminen de cubrir los 360º de la plataforma.

Luego de terminar una vuelta completa la aguja subirá X cm y el proceso comenzara nuevamente hasta que de la vuelta completa y subirá otra vez Xº y seguirá en esa rutina hasta que se cubra el volumen del objeto completamente.

6.3 NUBE DE PUNTOS

El resultado del proceso será una nube de puntos que recrean la superficie del objeto y que toman la forma de círculos que contornan la forma del objeto escaneado, estos puntos son almacenados en los siguientes archivos:

Extensión	Datos	Programa asociado
.txt (archivo de texto)	Coordenadas cilíndricas	Notepad
.txt (archivo de texto)	Coordenadas rectangulares	Notepad
.wrl (archivo VRML)	nstituciones del objeto	Internet Explorer con el debido intérprete.
.obj (archivo OBJECT)	Nube de puntos	Caligari Truespace 4, Metacreations Bryce3D, Alias/Wavefront Maya, Cinema 4DXL, SweetMollyGrace, Metacreations Poser3, etc.

Como ejemplo tenemos la siguiente figura en la que se grafica la nube de puntos del escaneo de un vaso:

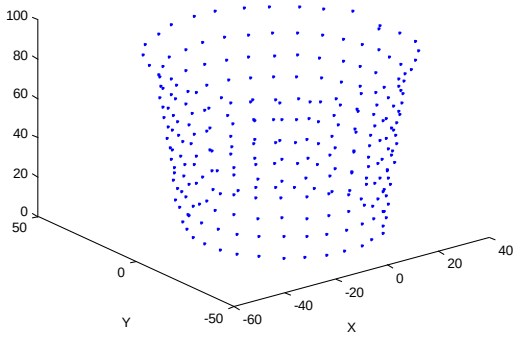


Fig. 33

Para tener una idea de la secuencia de escaneo se unen los puntos desde el primero que se escaneo hasta el ultimo para observar mejor como se tomaron los datos, y se presentan en la siguiente figura:

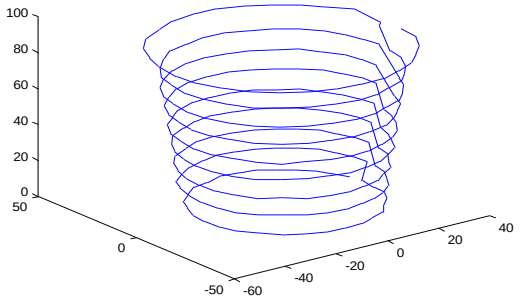


Fig .34

Aquí se puede ver que punto fue el primer dato tomado con el nstitu y como continuo el proceso hasta escanear el objeto completo.

7. RECURSOS UTILIZADOS

7.1 HARDWARE:

nstituciones or Basic Stamp® BS2SX.

3 Motores PAP M42SP-4N

Elementos mecánicos de LEGO®

Equipo electrónico para Interfaz con PC

- 6 Puentes H: TA7267
- 2 74LS244
- 6 Opto acopladores P721
- 7404
- LEDS
- Chatarra electrónica y mecánica:
- Carro de impresora de inyección
- nstitucio JVC, para la plataforma giratoria
- Aguja de acero
- Cables
- conectores

7.2 SOFTWARE:

- nstit Basic®
- Lenguaje de nstitucione Pbasic®
- MatLab®

8. REFERENCIAS

- http://www.parallax.com Sensores ultrasónicos y microcontrolador Basic Stamp
- http://www.aliatron.com/parallax/pbasic_sp.htm
- http://www.zenstar.com/dxfobj.htm
- http://www.righthemisphere.com/sitemap.htm
- http://graphics.stanford.edu/~kapu/scanners.html Escáner 3D comerciales, nstituc, nstituc, nstituciones, etc.
- Papers sobre triangulación óptica como: http://www-2.cs.cmu.edu/~seitz/course/SIGG99/papers/curl95.pdf
- Papers sobre medición ultrasónica de distancias.
- Hojas Técnicas

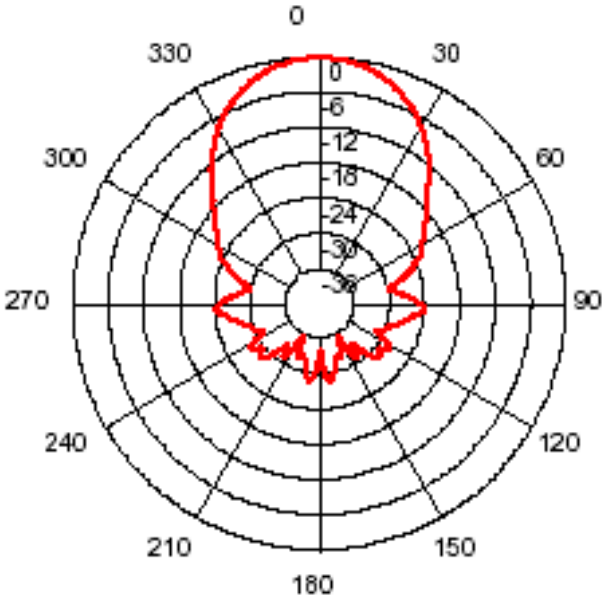
9. ANEXOS

9.1 PRUEBAS REALIZADAS

Se realizaron pruebas con dos sensores de distancia, el sensor de ultrasonido y el sensor infrarrojo. A continuación se describen los resultados de dichas pruebas.

SENSOR DE ULTRASONIDO

A continuación se muestra un diagrama polar de las mediciones que se tomaron con dicho sensor:



Como se puede observar la precisión que tiene el sensor no es la ideal para la aplicación que queremos implementar, por lo tanto los objetos que se escanearon no representaron ni por cerca la forma del objeto real. A continuación se muestran algunas conclusiones de las pruebas:

- Alcance mínimo de 3 centímetros
- Alcance máximo de 3 metros
- No mide la distancia directa al punto especifico al que se esta enfocando

SENSOR INFRARROJO

Al hacer pruebas con el sensor infrarrojo pudimos observar que los valores que media no eran constantes y el rango de variación era de 0.9 voltios aproximadamente, por lo que no se podía tomar como una medición exacta. Además de todo eso el sensor tenia las siguientes limitaciones:

- Sensibilidad a la luz
- Sensibilidad al color del objeto
- Datos demasiado variables
- Rango de detección de 10 a 80 cm
- Resolución insuficiente para la aplicación

9.2 ESPECIFICACIONES DEL BASIC STAMP

Stamp Specifications (revised 09/02)

Released Products	Rev.D / BS1-IC	BS2-IC	BS2e-IC	BS2sx-IC
Package	PCB w/Proto / 14-pin SIP	24-pin DIP	24-pin DIP	24-pin DIP
Package Size (L x W x H)	2.5" x 1.5" x .5" / 1.4" x .6" x .1"	1.2" x 0.6" x 0.4"	1.2" x 0.6" x 0.4"	1.2" x 0.6" x 0.4"
Environment	0º - 70º C* (32º - 158º F) **	0º - 70º C* (32º - 158º F) **	0º - 70º C* (32º - 158º F) **	0º - 70º C* (32º - 158º F) **
Microcontroller	Microchip PIC16C56c	Microchip PIC16C57c	Scenix SX28AC	Scenix SX28AC
Processor Speed	4 MHz	20 MHz	20 MHz	50 MHz
Program Execution Speed	~2,000 instructions/sec.	~4,000 instructions/sec.	~4,000 instructions/sec.	~10,000 instructions/sec.
RAM Size	16 Bytes (2 I/O, 14 Variable)	32 Bytes (6 I/O, 26 Variable)	32 Bytes (6 I/O, 26 Variable)	32 Bytes (6 I/O, 26 Variable)
Scratch Pad RAM	N/A	N/A	64 Bytes	64 Bytes
EEPROM (Program) Size	256 Bytes, ~80 instructions	2K Bytes, ~500 instructions	8 x 2K Bytes, ~4,000 inst.	8 x 2K Bytes, ~4,000 inst.
Number of I/O pins	8	16 + 2 Dedicated Serial	16 + 2 Dedicated Serial	16 + 2 Dedicated Serial
Voltage Requirements	5 - 15 vdc	5 - 15 vdc	5 - 12 vdc	5 - 12 vdc
Current Draw @ 5V	2mA Run / 20µA Sleep	8 mA Run / 100 µA Sleep	20 mA Run / 100 µA Sleep	60 mA Run / 200 µA Sleep
Source / Sink Current per I/O	20 mA / 25 mA	20 mA / 25 mA	30 mA / 30 mA	30 mA / 30 mA
Source / Sink Current per unit	40 mA / 50 mA	40 mA / 50 mA per 8 I/O pins	60 mA / 60 mA per 8 I/O pins	60 mA / 60 mA per 8 I/O pins
PBASIC Commands	32	36	39	39
PC Programming Interface	Parallel Port	Serial Port (9600 baud)	Serial Port (9600 baud)	Serial Port (9600 baud)
DOS Text Editor	STAMP.EXE	STAMP2.EXE	STAMP2E.EXE	STAMP2SX.EXE
Windows Text Editor	N/A	Stampw.exe (v1.04 and up)	Stampw.exe (v1.096 and up)	Stampw.exe (v1.091 and up)

Released Products	BS2p24 -IC	BS2p40 -IC	BS2pe-IC	Java Stamp
Package	24-pin DIP	40-pin DIP	24-pin DIP	24-pin DIP
Package Size (L x W x H)	1.2" x 0.6" x 0.4"	2.1" x 0.6" x 0.4"	1.2" x 0.6" x 0.4"	1.24" x 0.60" x 0.45"
Environment	0º - 70º C* (32º - 158º F) **	0º - 70º C* (32º - 158º F) **	0º - 70º C* (32º - 158º F) **	0º - 70º C* (32º - 158º F) **
Microcontroller	Scenix SX48AC	Scenix SX48AC	Ubicom SX48AC	Ubicom SX48AC
Processor Speed	20 MHz Turbo	20 MHz Turbo	8 MHz Turbo	25 MHz Turbo
Program Execution Speed	~12,000 instructions/sec.	~12,000 instructions/sec.	~6000/sec.	~8,500 instructions/sec. ***
RAM Size	38 Bytes (12 I/O, 26 Variable)	38 Bytes (12 I/O, 26 Variable)	38 Bytes (12 I/O, 26 Variable)	32768 Bytes
Scratch Pad RAM	128 Bytes	128 Bytes	128 Bytes	N/A
EEPROM (Program) Size	8 x 2K Bytes, ~4,000 inst.	8 x 2K Bytes, ~4,000 inst.	16 x 2K Bytes (16 K for source)	32768 Bytes
Number of I/O pins	16 + 2 Dedicated Serial	32 + 2 Dedicated Serial	16 + 2 Dedicated Serial	16
Voltage Requirements	5 - 12 vdc	5 - 12 vdc	5 - 12 vdc	5 vdc (reg), 6-24 vdc (unreg)
Current Draw @ 5V	40 mA Run / 400 µA Sleep	40 mA Run / 400 µA Sleep	15mA Run / 60µA Sleep	60mA Run / 13µA Sleep
Source / Sink Current per I/O	30 mA / 30 mA	30 mA / 30 mA	30 mA / 30 mA	30 mA / 30 mA
Source / Sink Current per unit	60 mA / 60 mA per 8 I/O pins	60 mA / 60 mA per 8 I/O pins	60 mA / 60 mA per 8 I/O pins	60 mA / 60 mA per 8 I/O pins
PBASIC Commands	55	55	55	0 (Java)
PC Programming Interface	Serial Port	Serial Port	Serial Port	Serial Port
DOS Text Editor	STAMP2P.EXE	STAMP2P.EXE	N/A	N/A
Windows Text Editor	Stampw.exe (v1.1 and up)	Stampw.exe (v1.1 and up)	Stampw.exe (v1.33 and up)	Javelin Stamp IDE

* Industrial Models Available - Environment is -40º - 85º C (-40º - 185º F)**

** 70% Non-Condensing Humidity

9.3 HOJAS TECNICAS

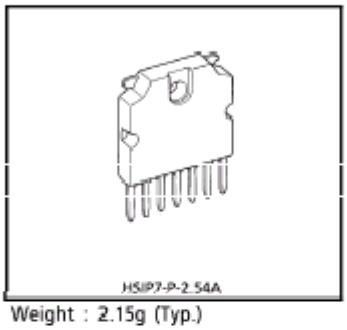
TOSHIBA TA7267BP

TOSHIBA BIPOLAR LINEAR INTEGRATED CIRCUIT SILICON MONOLITHIC

TA7267BP

BRIDGE DRIVER

The TA7267BP is a Bridge Driver for brushed DC Motor Rotation control.
Forward Rotation, Reverse Rotation, Stop and Braking operations are available.
It's designed for Loading and Reel Motor driver for VCR and Tape Deck. and any other consumer and industrial applications.
TA7267BP have Operation Supply Voltage terminal and Motor Driving Supply Voltage terminal independently, therefore Servo control operation is applicable.



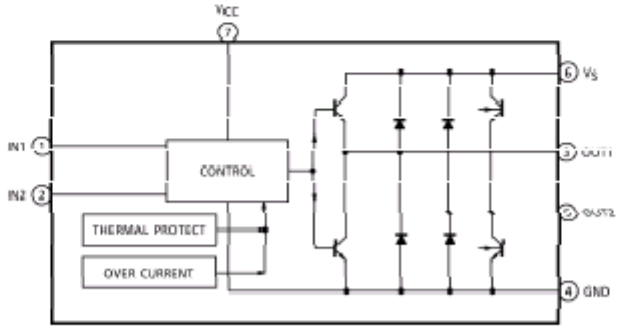
HSIP7-P-2.54A

Weight : 2.15g (Typ.)

FEATURES

- Output Current Up to 1.0A (AVE.), and 3.0A (PEAK).
- 4 Function Modes (CW, CCW, STOP and Brake) are Controlled by 2 Logic Signals Fed Into 2 Input Terminals.
- Build in Over Current Protector and Thermal Shut Down Circuit.
- Operating Voltage Range : $V_{CC\ (opr.)} = 6 \sim 18V$, $V_{S\ (opr.)} = 0 \sim 18V$

BLOCK DIAGRAM



SN54LS240, SN54LS241, SN54LS244, SN54S240, SN54S241, SN54S244
SN74LS240, SN74LS241, SN74LS244, SN74S240, SN74S241, SN74S244
OCTAL BUFFERS AND LINE DRIVERS WITH 3-STATE OUTPUTS

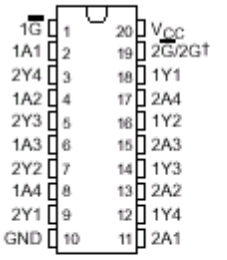
SDLS144B – APRIL 1985 – REVISED FEBRUARY 2002

- 3-State Outputs Drive Bus Lines or Buffer Memory Address Registers
- PNP Inputs Reduce DC Loading
- Hysteresis at Inputs Improves Noise Margins

description

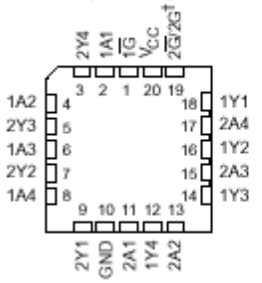
These octal buffers and line drivers are designed specifically to improve both the performance and density of three-state memory address drivers, clock drivers, and bus-oriented receivers and transmitters. The designer has a choice of selected combinations of inverting and noninverting outputs, symmetrical, active-low output-control ($\overline{G}$) inputs, and complementary output-control (G and $\overline{G}$) inputs. These devices feature high fan-out, improved fan-in, and 400-mV noise margin. The SN74LS' and SN74S' devices can be used to drive terminated lines down to 133 Ω.

SN54LS', SN54S' . . . J OR W PACKAGE
SN74LS240, SN74LS244 . . . DB, DW, N, OR NS PACKAGE
SN74LS241 . . . DW, N, OR NS PACKAGE
SN74S' . . . DW OR N PACKAGE
(TOP VIEW)



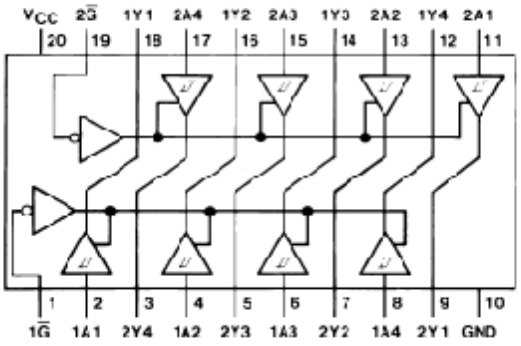
† 2G for 'LS241 and 'S241 or $2\overline{G}$ for all other drivers.

SN54LS', SN54S' . . . FK PACKAGE
(TOP VIEW)



† 2G for 'LS241 and 'S241 or $2\overline{G}$ for all other drivers.

Dual-In-Line Package

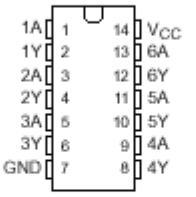


- Dependable Texas Instruments Quality and Reliability

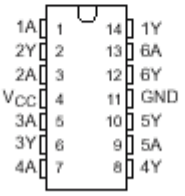
description/ordering information

These devices contain six independent inverters.

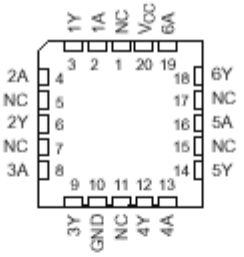
SN5404 . . . J PACKAGE
SN54LS04, SN54S04 . . . J OR W PACKAGE
SN7404, SN74S04 . . . D, N, OR NS PACKAGE
SN74LS04 . . . D, DB, N, OR NS PACKAGE
(TOP VIEW)



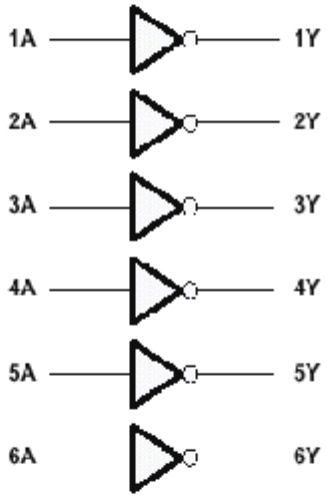
SN5404 . . . W PACKAGE
(TOP VIEW)



SN54LS04, SN54S04 . . . FK PACKAGE
(TOP VIEW)



NC – No internal connection



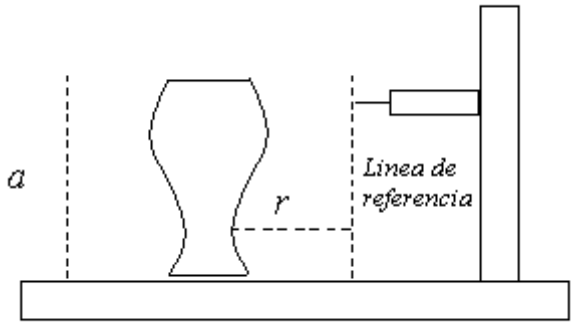
$Y = \overline{A}$

9.4 TIEMPO ESTIMADO DE ESCANEO

A continuación se muestra una tabla con los tiempos de escaneo aproximado dependiendo de la altura y radio de calibración del objeto que se va a escanear.

a \ r	5	13	21	29	37	45	53	61	69	77	85
5	7	9,2	11,4	13,6	15,8	18	20,2	22,4	24,6	26,8	29
10	10,5	13,8	17,1	20,4	23,7	27	30,3	33,6	36,9	40,2	43,5
15	14	18,4	22,8	27,2	31,6	36	40,4	44,8	49,2	53,6	58
20	17,5	23	28,5	34	39,5	45	50,5	56	61,5	67	72,5
25	21	27,6	34,2	40,8	47,4	54	60,6	67,2	73,8	80,4	87
30	24,5	32,2	39,9	47,6	55,3	63	70,7	78,4	86,1	93,8	101,5
35	28	36,8	45,6	54,4	63,2	72	80,8	89,6	98,4	107,2	116
40	31,5	41,4	51,3	61,2	71,1	81	90,9	100,8	110,7	120,6	130,5
45	35	46	57	68	79	90	101	112	123	134	145
50	38,5	50,6	62,7	74,8	86,9	99	111,1	123,2	135,3	147,4	159,5
55	42	55,2	68,4	81,6	94,8	108	121,2	134,4	147,6	160,8	174
60	45,5	59,8	74,1	88,4	102,7	117	131,3	145,6	159,9	174,2	188,5
65	49	64,4	79,8	95,2	110,6	126	141,4	156,8	172,2	187,6	203
70	52,5	69	85,5	102	118,5	135	151,5	168	184,5	201	217,5
75	56	73,6	91,2	108,8	126,4	144	161,6	179,2	196,8	214,4	232
80	59,5	78,2	96,9	115,6	134,3	153	171,7	190,4	209,1	227,8	246,5
85	63	82,8	102,6	122,4	142,2	162	181,8	201,6	221,4	241,2	261
90	66,5	87,4	108,3	129,2	150,1	171	191,9	212,8	233,7	254,6	275,5
95	70	92	114	136	158	180	202	224	246	268	290
100	73,5	96,6	119,7	142,8	165,9	189	212,1	235,2	258,3	281,4	304,5
105	77	101,2	125,4	149,6	173,8	198	222,2	246,4	270,6	294,8	319
110	80,5	105,8	131,1	156,4	181,7	207	232,3	257,6	282,9	308,2	333,5

Definiendo el radio de calibración como el radio que hay existe entre el objeto y el sensor en su línea de referencia lo cual se muestra en la siguiente figura:



Donde :

r = Radio de calibración

a = Altura del objeto

De lo anterior se define la siguiente función para un estimado de tiempo de escaneo. Para lo cual es recomendable tomar el radio de calibración como el radio máximo que existe en un objeto irregular, como se muestra en la figura anterior, de esta forma se asegura un tiempo máximo de escaneo:

$$T = \frac{\left[7 + 2.2\left(\frac{r-5}{8}\right)\right]}{2} \cdot \left(2 + \frac{a-5}{5}\right)$$