UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO

FACULTAD DE INGENIERÍA MECÁNICA

DIVISIÓN DE ESTUDIOS DE POSGRADO

AUTOMATIZACIÓN DE LAS COORDENADAS DE UN MICROTORNO MANUAL BASADA EN MICROCONTROLADORES PIC"

TESIS

QUE PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS EN INGENIERÍA MECÁNICA

PRESENTA
DAVID ALVARADO ZAMORA

ASESOR
DOCTOR EN INGENIERÍA IGNACIO JUÁREZ CAMPOS

COASESOR
DOCTORA EN INGENIERÍA BIOMÉDICA LUCIA MÁRQUEZ PÉREZ

MORELIA, MICHOACÁN, ABRIL DE 2013

DEDICATORIA

Dedico este trabajo principalmente a mis padres y hermanas que incondicionalmente me han brindado su apoyo para el moldeo de mi personalidad y formación académica al alumbrarme con su experiencia y sabiduría en los momentos que la mente se distrae.

Hay una fuerza motriz más poderosa que el vapor,

la electricidad y la energía atómica: la voluntad.

Albert Einstein

AGRADECIMIENTOS

A mis compañeros de clase luego de que me indicaran rumbos de la vida con miles de anécdotas y experiencias que fuimos recogiendo con afán de llegar hasta el punto de nuestra graduación de este posgrado.

A mi asesor quien compartiera valiosos conocimientos ampliándome el panorama de la vida profesional.

A mi Facultad que me albergó durante toda mi formación académica con todo su personal docente y administrativo de alto nivel.

A mi Universidad que me ha tatuado con una identidad nicolaita que siempre portaré con singular orgullo.

RESUMEN

Este trabajo muestra las técnicas, procedimientos, programación y montaje de componentes necesarios para la automatización del posicionamiento del buril del microtorno manual modelo #OTMT-CO de la marca OTMT empleando el microcontrolador PIC 18F4550 que estará en comunicación con una aplicación desarrollada en el *Software Visual Basic 6.0* por medio del puerto USB de una computadora para lograr el control de motores a pasos que tendrá montados el torno antes mencionado.

La programación del microcontrolador PIC18F4550 es desarrollada con el compilador *MicroCode Studio 4.0.0.0* para la habilitación de la comunicación por medio del puerto USB y la interpretación de las órdenes provenientes de la aplicación (simulador) en la computadora.

La validación de este proyecto tiene lugar luego de que las piezas maquinadas por el microtorno manual #OTMT-CO fueron comparadas con otras dimensionadas de igual manera pero maquinadas en un torno de control numérico (*Dyna Myte 3000*), obteniendo una precisión superior a la de éste último cuando el sistema de control opera en unidades inglesas y ligeramente inferior en su operación en el sistema métrico.

ABSTRACT / SUMMARY

This work presents techniques, procedures, programming and assembly of the required components that are necessary for the automated positioning graver in a manual micro-lathe model # OTMT-CO OTMT brand, implementing a PIC 18F4550 microcontroller which will be in communication with an application developed in Visual Basic 6.0 software through an USB port computer to achieve the control of the stepper motors that the lathe mentioned above will have erected.

The PIC18F4550 microcontroller programming is developed with Microcode Studio 4.0.0.0 compiler for enabling communication via the USB port and the interpretation of the commands from the application (simulator) in the computer.

The validation of this project takes place after the parts machined by the manual micro-lathe # OTMT-CO were compared with other parts same dimensioned but machined in a computer numerical controlled lathe or CNC (Dyna Myte 3000), obtaining a higher accuracy than the last mentioned when the control system operates in English units and slightly lower in its operation in the metric system.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE TABLAS

ÍNDICE DIAGRAMAS

INTRODUCCIÓN

Una de las herramientas imprescindibles para la manufactura de piezas mecánicas y de ornamentación es el torno. El cual es una máquina capaz de producir piezas de revolución mediante la rotación de un material primario (figura 1) que será desbastado mediante una herramienta de corte, llámese buril, para desbastar en diferentes trayectorias y así retirar material de la pieza para dar formas y terminados (figura 3) en el material primario[1].

Fig. 1 Material primario.

Fig. 2 Torno manual.

Fig. 3 Piezas producidas por el torno.

Existe una gran variedad de estos equipos y entre ellos se encuentran los tornos artesanales, tornos manuales, tornos semiautomáticos, tornos copiadores, tornos CNC (control numérico por computadora *"computer numerical control"*), entre otros. Estos últimos son los más actuales y son controlados por lenguajes de programación de bajo nivel entre los que se encuentra: el código *G* y el código *M* además de que son los más precisos y costosos. Los temas tratados en este trabajo serán relacionados con la transformación de un torno manual a uno con características similares al CNC.

Los tornos manuales tienen limitantes muy importantes tales como la precisión, complejidad, y capacidad de producción en serie del producto final sobre todo en maquinados cónicos y esféricos[2].

Por lo anterior, la presente tesis expone una propuesta para mitigar estas limitantes con la elaboración e implementación de un sistema electromecánico para el control de un microtorno manual hasta lograr un control semiautomático y automático del mismo. Esto será a través de adaptaciones mecánicas entre torno y motores a pasos controlados por componentes electrónicos y microcontroladores que permitan el movimiento de los ejes de trabajo del microtorno. Esta adaptación permitirá aumentar la precisión de los maquinados en las piezas manufacturadas así como el tiempo de producción en serie y el grado de complejidad de las mismas[2].

Al realizar las modificaciones mencionadas anteriormente, el costo de las piezas maquinadas disminuirá debido a la reducción del número de empleados; por ejemplo, en algún taller donde se cuente con varios operadores de torno.

Cabe mencionar que existen trabajos similares donde se realiza la automatización de minitornos manuales a través de motores a pasos operados por *software* realizados en *LabView* y tarjetas controladoras para los motores a pasos.

PLANTEAMIENTO DEL PROBLEMA

En los tornos manuales se pueden observar limitantes claras al realizar maquinados que requieren de una sincronización en los movimientos de los ejes de trabajo (X y Z) que sean lo más exacto posible.

En el caso del microtorno manual #OTMT-CO, se pueden observar tres limitantes fundamentales las cuales se enumeran como sigue:

1. Volumen de producción en serie.
2. Realizar maquinados cónicos.
3. Realizar maquinados esféricos.

La primera es característica normal de cualquier torno manual y está determinada por el operador del torno al resultar fatigante tener largas jornadas de trabajo en un torno manual para realizar múltiples maquinados debido a la demanda de agudeza visual y la constante verificación de las dimensiones (generalmente con un vernier o micrómetro) hasta alcanzar las descritas por un plano de manufactura.

La segunda y tercera se presentan en un torno manual que no cuente con funciones mecánicas secundarias como una torre de herramienta con desplazamiento angular para los maquinados cónicos, y con un portaburil orbital para los esféricos, lo que imposibilita al microtorno #OTMT-CO para realizar este tipo de maquinados al carecer de estas adaptaciones.

Estas limitantes quedarían resueltas si se contase con un torno de control numérico debido a que es un conjunto de comandos (generalmente empleando código G y M) los programados para el maquinado de una pieza específica y “correrlo” las veces que resulten convenientes. La contraparte de esta solución se aprecia al cotizar los elevados costos de un torno con estas características.

Algunos investigadores en el ámbito de la automatización han desarrollado sistemas mecatrónicos que permiten el control de los movimientos de un torno para dar salida a las limitaciones que se han descrito anteriormente. Estos desarrollos tecnológicos emplean algún *Software* dirigido al campo de la automatización para el control de motores a pasos ó en algunos casos servomotores. Por citar un ejemplo de *Software* más empleado en la automatización se menciona *LabView 2011,* que en su licencia profesional tiene un costo de $7,500.00 USD lo que muchas veces resulta inaccesible para desarrollar proyectos con algún *Software* como estos.

OBJETIVO GENERAL

Diseñar y construir un sistema electrónico capaz de controlar un microtorno manual empleando motores a pasos para lograr los movimientos (en los ejes X y Z) accionando botones y basado en microcontroladores PIC (*Peripheral Interface Controller*) con el fin de superar las limitantes que poseen dichos tornos por sí mismos.

OBJETIVOS ESPECÍFICOS

1.- Estudiar y determinar un sistema de control electrónico que permita la manipulación de un microtorno a través de motores a pasos similar a un sistema de CNC (Control Numérico por Computadora).

2.- Analizar el sistema mecánico de transmisión manual del microtorno y diseñar e implementar uno nuevo que se adapte a las condiciones de automatización requeridas.

3.- Diseñar e implementar el sistema de control electrónico y la respectiva programación acorde a las necesidades de manufactura de las piezas, tales como geometrías de perfiles y acabados. Este sistema debe contar con un sistema de comunicación usuario máquina.

4.- Evaluar el desempeño del microtorno ya automatizado, mediante una comparación con tornos CNC similares.

JUSTIFICACIÓN

Actualmente el campo de diseño y construcción de mecanismos dependen en gran medida de los procedimientos y precisión del maquinado de sus componentes. Por esto existen equipos capaces de realizar maquinados en diversos materiales y según sea el grado de complejidad y de precisión se emplean equipos de manufactura específicos.

Los tornos manuales son una de las variantes de tornos existentes, la precisión de los maquinados desarrollados por este equipo, depende de la experiencia del operador y tiene limitantes en cierto tipo de maquinados. Por esto se pretende automatizar procesos que permitan superar las limitantes mencionadas en dicho microtorno. A su vez, como se ve en la figura 4, se aprecia el aumento en características favorables, como lo son la capacidad de realizar maquinados más complejos, aumento en el volumen de producción y mejora de la calidad con la respectiva homogenización de ésta al realizar la misma pieza en repetidas ocasiones.

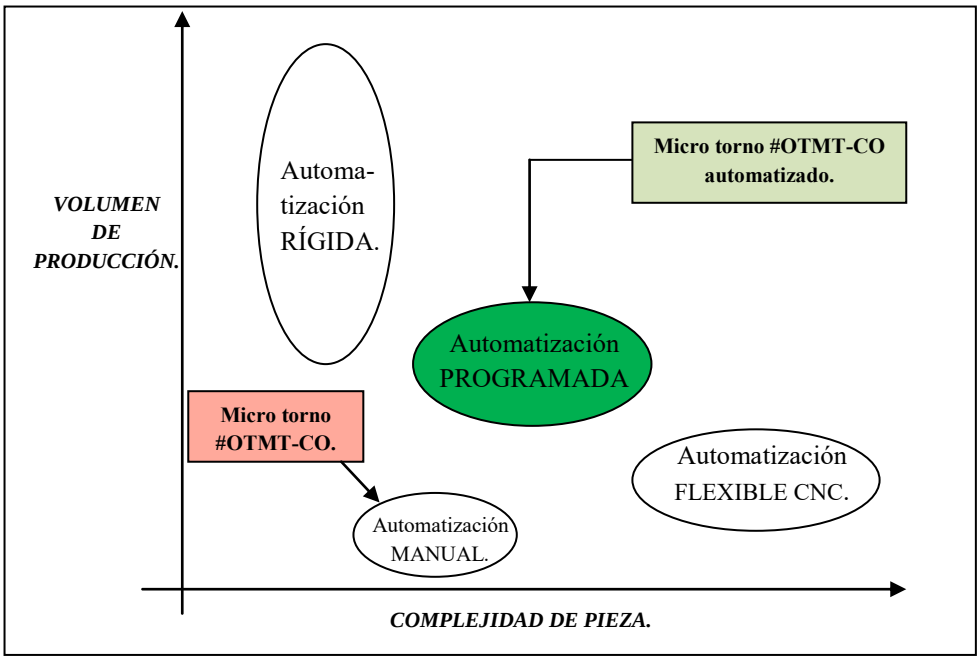


Fig. 4 tipos de automatización.

HIPÓTESIS

La automatización del posicionamiento de la herramienta de corte de un microtorno manual es posible, manteniendo un bajo costo en su transformación de manual a automático y con capacidades de desempeño similares a un torno CNC. Con la implementación de un simulador se logrará el control de motores a pasos que permitan realizar maquinados con geometría de mayor complejidad que las de un torno manual.

Al contar con una máquina que realice tareas de forma automática, se incrementará el volumen de producción así como las ganancias económicas de algún taller al incorporar menos personal con capacitación sobre el nuevo sistema de funcionamiento del torno.

Después de las modificaciones en el microtorno, las piezas maquinadas en el mismo podrán someterse a comparaciones con otras maquinadas en un torno de control numérico para la valoración del desempeño del microtorno y sus modificaciones.

METODOLOGÍA

Para el alcance de los objetivos planteados en esta tesis, a continuación se describe a manera de resumen los puntos clave que resuelven las necesidades de una automatización similar a los tornos equipados con control CNC.

1.- Modificación mecánica para la adaptación de los motores a pasos con el microtorno manual. En este punto se trata la creación de los soportes que apoyan a los motores a pasos al momento de su incorporación en el microtorno cuidando un ajuste aceptable en la alineación de eje del motor con el tornillo sinfín de los dos ejes del microtorno, así como alguna interferencia posible entre los soportes y la morfología inicial del microtorno.

2.- Maquinado de bridas de acoplamiento motor-tornillo sin fin. La fuerza torzora es transmitida de los motores a pasos hacia los tornillos sin fin a través de bridas maquinadas en aluminio.

3.- Creación de una aplicación desarrollada en *Visual Basic 6.0* (Simulador). Con motivo del envío y recepción de información de la interface controladora de los motores a pasos, esta aplicación tiene como objetivos la comunicación de los datos necesarios para los maquinados así como la obtención de los datos suministrados por el usuario para el previo trazo de los maquinados a realizar.

4.- Creación de interfaz controladora de los motores a pasos. Después del envío de datos provenientes del simulador, los datos son recibidos por una interfaz que transforma órdenes del simulador en órdenes para los motores a pasos a la vez que amplifica las señales electrónicas de baja y mediana potencia.

5.- Pruebas y comparaciones convenientes para evaluar el desempeño del microtorno al final de todas las modificaciones realizadas.

En el diagrama 1, se muestra el funcionamiento general del proceso para el control de los motores a pasos.

A lo largo de la redacción de esta tesis se muestran datos específicos sobre los puntos anteriormente expuestos.

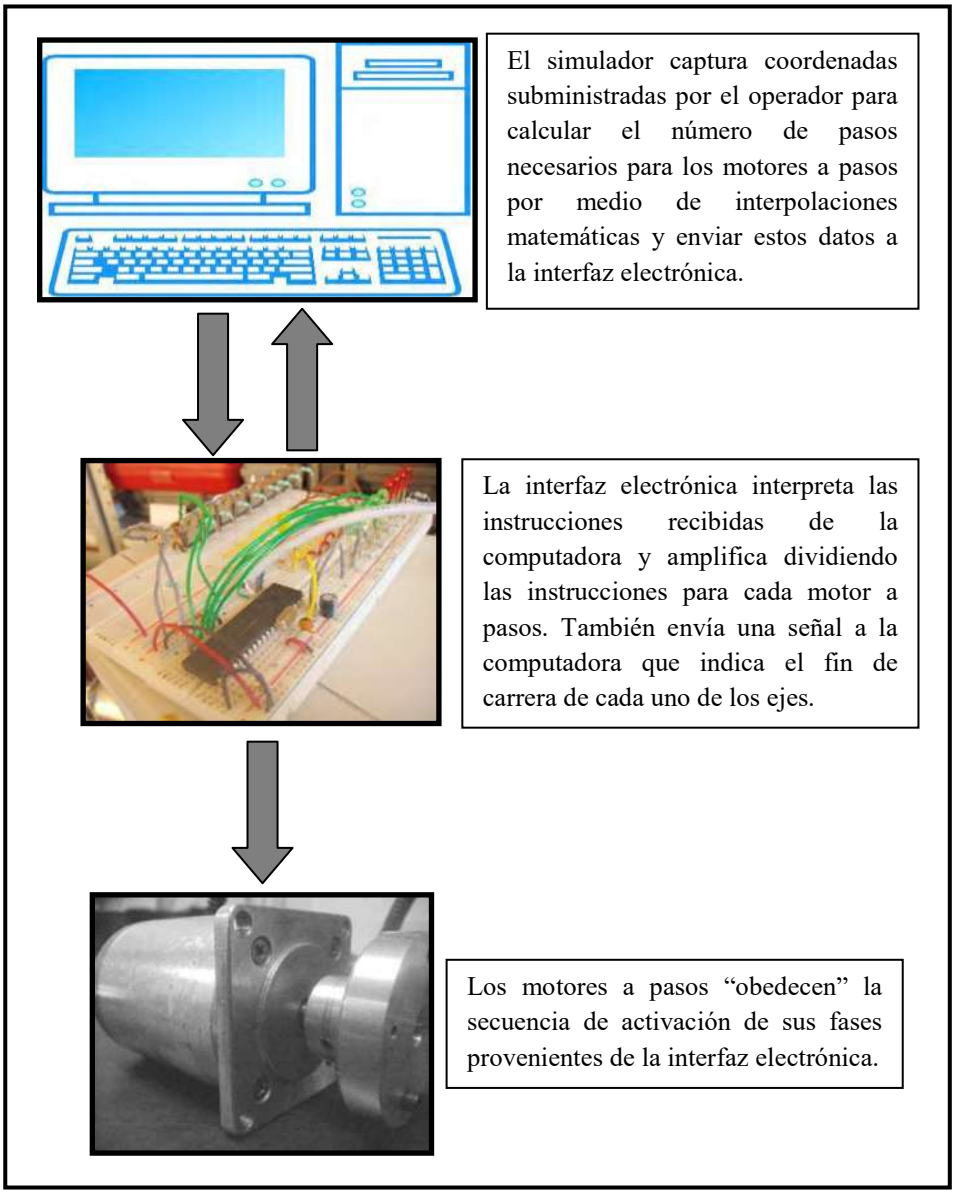


Diagrama 1.- Diagrama general de funcionamiento.

CAPÍTULO I
ANTECEDENTES.

1.1- HISTORIA DEL TORNO

Fig. 1.1. Fresco Benni Hassan I. Egipto, 1,500 A.C.

Aunque no se conoce con precisión los orígenes del torno, se sabe que es de las máquinas más antiguas, teniendo como precursor las mesas rotatorias que se empleaban para las actividades de alfarería (*Fig. 1.1*). Algunas bibliografías mencionan la aparición del torno en los años 850 A.C. siendo un relieve hallado en la tumba de Petosiris (pontífice egipcio quién murió a fines del siglo I) la imagen más antigua que se tiene de una máquina similar. Posteriormente existieron algunas máquinas que eran impulsadas por un arquillo que permitía hacer girar el eje acoplado a la pieza a maquinar (ejecutándose a forma de violonchelo). En al año de 1250 nació el torno de pedal, mostrando un gran avance sobre el accionado por arquillo debido a que permitía al operador dejar las manos libres para manipular las herramientas de corte[3].

Ya en el siglo XV se implementó un sistema de transmisión por correas que permitía la rotación de la pieza de forma continua. Leonardo da Vinci también tuvo su aporte al trazar en su Códice Atlántico varias ideas acerca de los tornos a finales del siglo XV, siendo ideas sin poder ser llevadas a la realidad por falta de medios pero sirviendo de orientación para diseños posteriores[3].

Hacia el año de 1480 el pedal fue combinado con un vástago y una biela. Con la aplicación de este mecanismo nació el torno de accionamiento continuo, lo que implicaba el uso de un mecanismo biela-manivela, que debía ser combinado con un volante de inercia para superar los puntos muertos. Con estos adelantos se maquinaban metales no ferrosos como latón, cobre y

bronce, con algunas mejoras posteriores, el torno siguió empleándose durante varios siglos. De este diseño primitivo en madera, se introdujeron elementos de fundición tales como la rueda, el soporte del eje principal, contra punto, apoyo de herramientas, y hacia el año de 1586, el mandril[3].

Al comenzar la revolución industrial en Inglaterra, durante el siglo XVII, se desarrollaron tornos capaces de dar forma a una pieza metálica. El desarrollo del torno industrial para metales en el siglo XVIII hizo posible la producción en serie de piezas de precisión[3].

En la década de 1780 el inventor francés Jacques de Vaucanson construyó un torno industrial con un portaherramientas deslizante que se hacía avanzar mediante un tornillo manual. Hacia el año 1797 el inventor británico Henry Maudslay y el inventor estadounidense David Wilkinson mejoraron este torno conectando el portaherramientas deslizante con el 'husillo', que es la parte del torno que hace girar la pieza trabajada. Esta mejora permitió hacer avanzar la herramienta de corte a una velocidad constante (avance automático). En 1820, el mecánico estadounidense Thomas Blanchard inventó un torno en el que una rueda palpadora seguía el contorno de un patrón para una caja de fusil y guiaba la herramienta cortante para tornear una caja idéntica al patrón, dando así inicio a lo que se conoce como torno copiador[3].

El llamado torno revólver, desarrollado durante la década de 1840, incorpora un portaherramientas giratorio que soporta varias herramientas al mismo tiempo. En un torno revólver puede cambiarse de herramienta con sólo girar el portaherramientas y fijarlo en la posición deseada. Hacia finales del siglo XIX se desarrollaron tornos de revólver automáticos para cambiar las herramientas de forma automática. En 1833, Joseph Whitworth desarrolló innovaciones que contribuyeron de manera fundamental para que al paso de

unos años patentara un torno paralelo para cilindrar y roscar con bancada de guías planas y carro transversal automático, que tuvo una gran aceptación. Dos tornos que llevan incorporados elementos de sus patentes se conservan en la actualidad, uno de ellos construido en 1843, se conserva en el "Science Museum" de Londres. El otro, construido en 1850, se conserva en el "Birmingham Museum" en Gran Bretaña[3].

Fué J.G. Bodmer quien en 1839 tuvo la idea de construir tornos verticales. A finales del siglo XIX, este tipo de tornos eran fabricados en distintos tamaños y pesos[3].

1.2.- TORNOS ACTUALES

En la década de los 40´s el inventor norteamericano John T. Parsons y su ayudante Frank L. Stulen, desarrollaron una forma de automatización por medio del control numérico, el cual implicaba el uso de datos en un sistema de referencia para definir superficies de contorno para un maquinado determinado, es aquí donde tiene sus orígenes el torno CNC (control numérico por computadora *"computer numerical control"*)[4].

La aplicación de control numérico se puede dividir en dos grupos:

- Aplicación con máquinas herramienta (taladrado, laminado, torneado, etc.).
- Aplicación sin máquina herramienta (ensamblaje, trazado, inspección, etc.).

El principio de operación de todas las aplicaciones del control numérico es el control de la posición relativa de una herramienta o elemento de procesado con respecto al objeto a procesar[4].

Los sistemas de control en los tornos se pueden dividir de acuerdo al diagrama 1.1. El control por CNC, es actualmente el método de automatización que ofrece mejor precisión y rapidez en maquinados de torneado.

Diagrama 1.1 División de los tornos según su operación.

1.3.- TRABAJOS SIMILARES

Existen diversos trabajos en la automatización de tornos manuales sin llegar precisamente a un control CNC que utilice código G para las órdenes de movimiento. Mencionaremos dos ejemplos semejantes al expuesto en esta tesis dando una pequeña reseña de cada uno.

1. *Proyecto:* **Automatización y control de un mini-torno paralelo**.
   - *Desarrollador en:* Departamento de sistemas y automática de la facultad de ingeniería eléctrica, en la Universidad de Carabobo Venezuela.
   - *Descripción*: Automatización de un torno a través de motores a pasos implementando una tarjeta de movimiento PCI-7344 de National Instruments instalada en una computadora de escritorio empleando lenguaje de programación en LabView y librería de FlexMotion[5].

Fig. 1.2 Automatización de minitorno Carabobo[5].

2. *Proyecto:* **Diseño y construcción de un torno de control numérico**.

- *Desarrollado en:* Universidad EAFIT Medellín Colombia.
- *Descripción:* Automatización del torno implementando una interfaz RS-232 del puerto serial de la PC, un microcontrolador PIC 18F442, motores a pasos, entre otros varios circuitos electrónicos. La automatización de este torno comprende la utilización de paquetes de diseño CAD (Diseño Asistido por Computadora) para el trazado previo del maquinado para posteriormente traducirlo en instrucciones que desarrolle el torno[6].

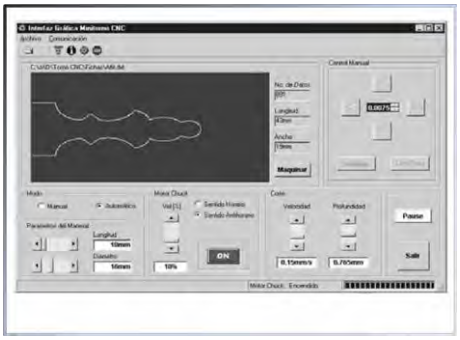


Fig. 1.3 Interfaz de automatización de torno UTP[6].

3. *Proyecto:* **Automatización de un torno paralelo, mediante un control numérico computarizado basado en PC.**

- *Desarrollado en:* Universidad Tecnológica de Pereira.
- *Descripción:* La implementación se realiza controlando los motores a pasos con una tarjeta de control CNC *Motion Control System,* en el apoyo para diseño de maquinado se emplea *AutoCAD,* para la traducción en código G; *MasterCAM,* y como *Software* de control; *Mach 2/3*[7].

Fig. 1.4 Minitorno Compact 5[7].

4. *Proyecto:* **Diseño de un sistema integrado para la conversión de un torno convencional a torno CNC.**

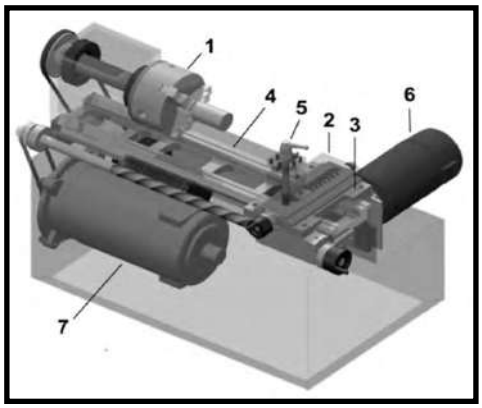


Fig. 1.5 Torno manual[8].

- *Desarrollado en:* Universidad de Antioquia.
- *Descripción:* Después del diseño y construcción de un minitorno en esta misma universidad, se dio a la tarea de diseñar un sistema de control numérico que cuenta con un microcontrolador PIC-SERVO para controlar motores DC, encoders de 1600 puntos por revolución, una tarjeta de potencia IGBT's y una aplicación desarrollada en *Visual Basic.NET*[8].

Cabe mencionar que la diferencia del trabajo expuesto en esta tesis a diferencia de los trabajos mencionados anteriormente, es el desarrollo del simulador y la programación en *Visual Basic 6.0* relativa para la realización de las interpolaciones de las coordenadas de los maquinados programados así como la programación en *MicroCode Studio 4.0.0.0* del microcontrolador PIC 18F4550 para la comunicación por medio del puerto USB para el envío y recepción de información con el simulador y hacia los motores a pasos.

1.4.- PARTES DEL TORNO MANUAL

Los tornos manuales ó de piso, están conformados por un número importante de piezas y controles, siendo los más importantes los enlistados en la figura 1.6[9].

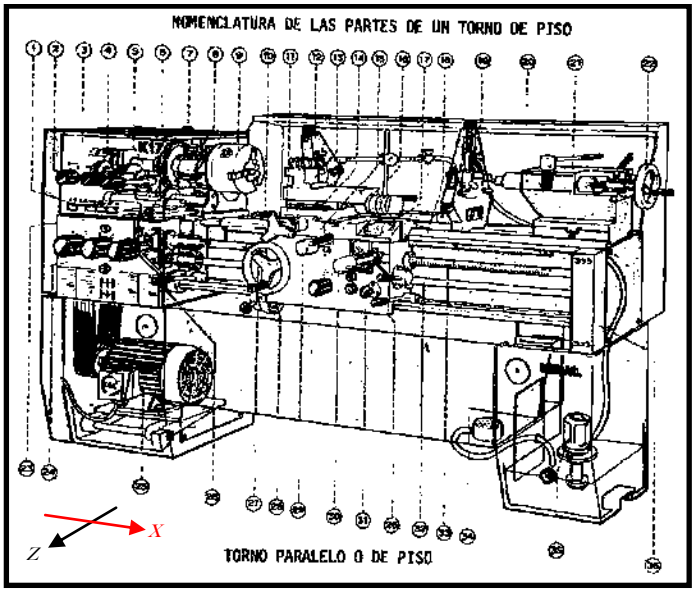


Fig. 1.6 Torno manual[9].

1	Botones de mando.	19	Luneta Fay.
2	Selector de avance.	20	Cubierta exterior.
3	Cabezal.	21	Contrapunto.
4	Engrane.	22	Volante del contrapunto.
5	Husillo del cabezal.	23	Nivel de aceite.
6	Engranes reductores.	24	Tablero selector de avances y roscados.
7	Visor del lubricante.	25	Motor.
8	Cojinete del husillo.	26	Palancas de embregue.
9	Chuck universal.	27	Palanca.
10	Volante del carro transversal.	28	Volante del carro longitudinal.
11	Carro transversal.	29	Palanca de avance automático transversal.
12	Luneta móvil (viajera).	30	Palanca de la tuerca dividida.
13	Portaherramientas simple.	31	Tablero.
14	Base graduada.	32	Barra para cilindrado.
15	Carro longitudinal.	33	Tornillo principal.
16	Carro auxiliar.	34	Colector de rebaba y aceite.
17	Indicador de carátula para roscado.	35	Bomba de lubricación.
18	Guía pusmática y bancada del carro principal.	36	Soporte de las barras.

Tabla 1.1.- Partes del torno[9].

1.5.- DATOS TÉCNICOS DEL MICROTORNO MODELO #OTMT-CO DE LA MARCA OTMT

La empresa ***OTMT TOOLS*** es la encargada de la producción del microtorno empleado en el desarrollo del proyecto de esta tesis. Los datos del desempeño de esta herramienta fueron obtenidos del manual muestrario proporcionado por la empresa.

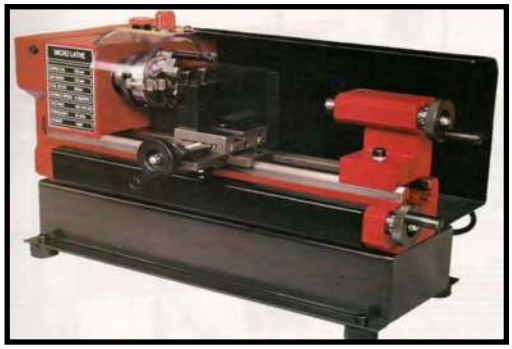


Fig. 1.7 Microtorno marca OTMT (#OTMT-CO)[10].

Utilizado principalmente para la fabricación de piezas pequeñas (joyería, aeromodelismo, etc.), el microtorno modelo #OTMT-CO de la marca OTMT, está dentro del grupo de tornos manuales, ya que el avance del portaherramienta es por medio de manivelas[10].

ESPECIFICACIONES:	
Volteo sobre la bancada	4-5/16" (110 mm)
Distancia entre puntos	5" (125 mm)
Agujero del Husillo	3/8" (10 mm)
Cuerda del Contrapunto	M14 x 1 mm
Rango de velocidades del husillo	100 - 3,800 RPM ± 10%
Carrera Longitudinal	2" (50 mm)
Poder del Motor	150 W, 110 V
Peso	13 Kg
Dimensiones (Cm)	44 L x 66 A x 61 Alto

Tabla 1.2 Especificaciones del Microtorno marca OTMT (#OTMT-CO)[10].

CAPÍTULO II

MODIFICACIONES MECÁNICAS Y ACOPLAMIENTO DE LOS MOTORES A PASOS.

2.1.- DISEÑO DE SOPORTES PARA MOTORES A PASOS

La morfología inicial del microtorno cuenta con manivelas (figura 2.1 y 2.2) para manipular de forma manual los desplazamientos en los dos ejes "X" y "Z" donde serán acoplados los motores a pasos por lo que existe inicialmente la necesidad de diseñar los soportes que se adapten tanto al microtorno como a los motores a pasos.

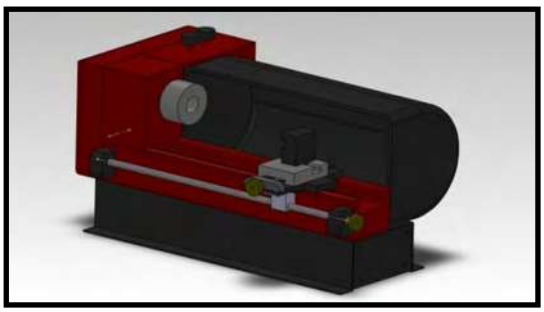


Fig. 2.1 Morfología inicial del microtorno modelo realizadfo en Solid Works 2011.

Fig. 2.2 Manivelas de control del microtorno.

Debido a que en esta tesis se tratan los temas relacionados con el control de los movimientos del buril no se atacan a profundidad temas referentes a la resistencia de los materiales de los cuales estén hechos los componentes mecánicos empleados en la modificación del microtorno, sin embargo a partir del apartado 2.2.2 se muestra una simulación realizada en ANSYS 14.0 de las deformaciones y esfuerzos *von Mises* que aparecen en los soportes al aplicar la fuerza torzora calculada en el apartado 2.2.1.

2.1.1.- ACOTAMIENTO DE LOS SOPORTES

Dicho lo anterior, el material empleado para la conformación de los soportes de los motores a pasos es acrílico de 5mm y los cortes realizados respetan los acotamientos mostrados en la figura 2.5 para el eje "X" y en la figura 2.3 para el eje "Z". Posteriormente los recortes son unidos empleando

cloroformo como adhesivo para el ensamblado de las piezas y finalmente tener una estructura como en las figuras 2.6 y 2.4 (eje "X" y "Z" respectivamente). Las dimensiones se obtuvieron luego de realizar un dibujo con escalas reales en el *software Solid Works 2011* incorporando los soportes y los motores a pasos para simular alguna posible colisión con las partes del microtorno.

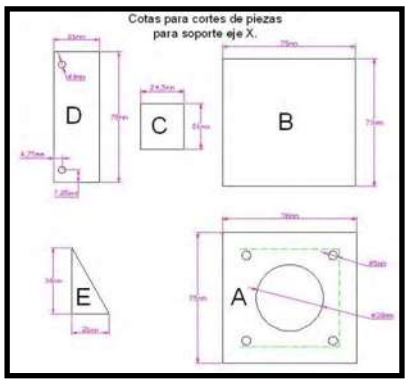


Fig. 2.3 Acotamiento de piezas del soporte eje "Z".

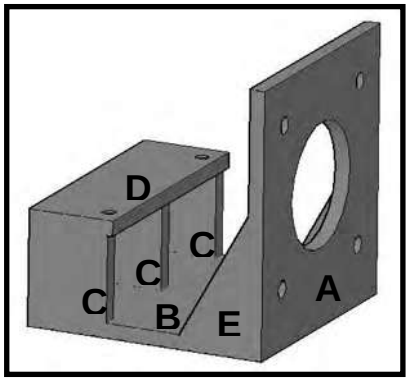


Fig. 2.4 Ensamble del soporte eje "Z".

Para obtener la mayor precisión posible en el dimensionamiento de cada pieza de los recortes de acrílico, todos los recortes se realizaron con ayuda de un cortador laser. Este maquinado facilitó en gran medida el armado debido a que todas las piezas tienen una homogeneidad muy aceptable y cada pieza embona en su respectivo lugar con un error mínimo en cuanto a dimensionamiento y cuadratura de la estructura.

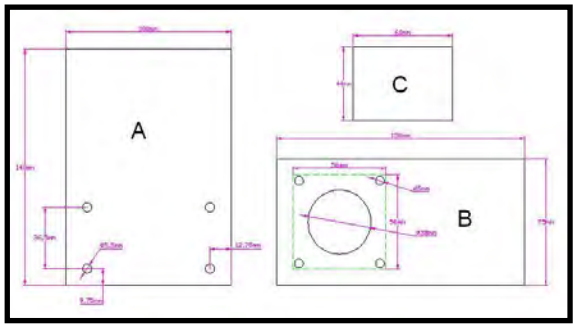


Fig. 2.5 Acotamiento de piezas del soporte eje "X".

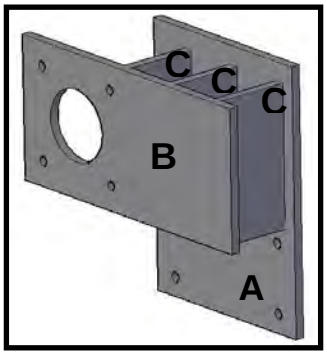


Fig. 2.6 Ensamble del soporte eje "X".

La morfología final de los soportes nos permite un acoplamiento microtorno-soporte donde no tenemos ningún tipo de interferencia en las partes restantes del microtorno como lo son soportes de tornillos sin fin de los ejes “X” y “Z”, movimientos de la torre porta herramienta, ni de los switchs de fin de carrera por lo que es preciso iniciar el diseño de las bridas de acoplamiento microtorno-motor a pasos.

2.1.2.- SOPORTES MONTADOS EN EL MICROTORNO

Una vez fabricados los soportes, estos son fijados al microtorno; en el caso del soporte del eje “X” es fijado mediante cuatro tornillos de Ø5/32” x 3/4” y un paso de 32 dientes por pulgada con sus respectivas arandelas plana y de presión (figura 2.7).

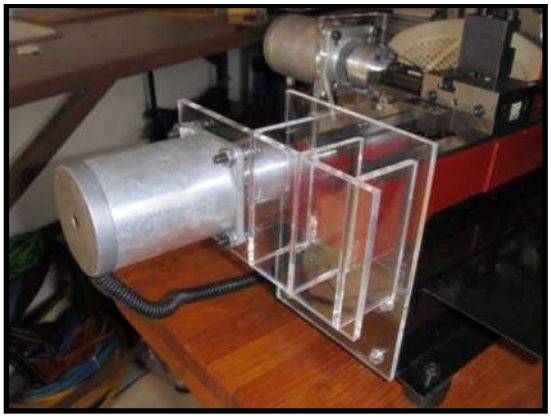


Fig. 2.7 Ensamblaje real de soporte eje “X”.

Se tiene la hipótesis de que el soporte del eje “X” será el que tenga mayor deformación cuando el motor a pasos esté ejerciendo una fuerza torzora sobre el soporte, pero esta cuestión será confirmada en la simulación en ANSYS en un apartado más delante de esta tesis.

En el caso del soporte del eje “Z”, la sujeción es por medio de dos tornillos de Ø9/64” x 1” y un paso de 40 dientes por pulgada con arandelas de

plana y de presión con los que ya contaba el microtorno, únicamente se aumentó la longitud del tornillo que inicialmente era de 13/32” (figura 2.8).

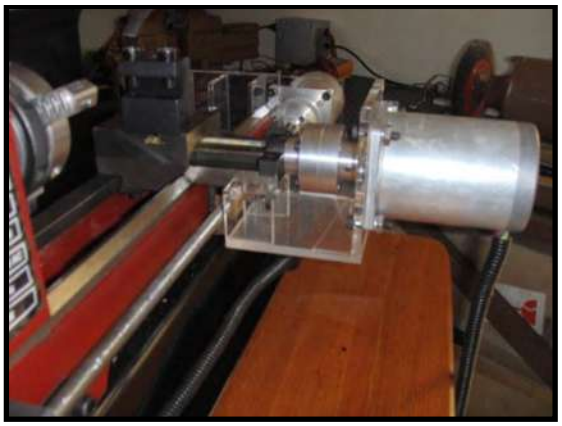


Fig. 2.8 Ensamblaje real de soporte eje “Z”.

De esta manera es como los soportes de los motores a pasos están acoplados al microtorno presentando una leve deformación al rotar debido a la rigidez de las bridas de acoplamiento de los motores a pasos.

2.2.- SIMULACIÓN DEL COMPORTAMIENTO DE LOS SOPORTES DE LOS MOTORES A PASOS

Con la finalidad de conseguir cierta fiabilidad sobre la capacidad de los soportes de los motores a pasos, se reporta la siguiente simulación generada en ANSYS donde se muestran los esfuerzos a los que están sometidos dichos soportes tanto del motor del eje “X” como del motor del eje “Z”; así como la simulación correspondiente a la deformación de cada uno de ellos. Los esfuerzos y deformaciones son generados por el torque producido por cada motor.

Es de vital importancia aclarar que el propósito de este trabajo de tesis finalmente tiene mayor injerencia sobre los temas relacionados con el control de los movimientos de la herramienta de corte del microtorno, por lo que en la

simulación presentada en este punto se obtienen valores iniciales con una técnica que pudiera ser mejorada sobre todo en el cálculo del torque de los motores a pasos y de esta manera obtener un resultado con mayor exactitud.

2.2.1.- CÁLCULO DEL TORQUE DE LOS MOTORES A PASOS

Los motores a pasos unipolares que han sido empleados en este proyecto son motores que se obtuvieron de un almacén de "chatarra" industrial por lo que no se cuenta con los datos del fabricante donde se informe los valores de alimentación eléctricos y del torque que es indispensable para la simulación que aquí se menciona. Por lo anterior, se desarrolló un mecanismo que con el principio básico de brazo de palanca permite el cálculo del torque de manera aproximada. El mecanismo consiste en la fabricación de una base de madera MDF (*Medium Density Fibreboard*) para soportar el motor a pasos el cual tiene acoplada una palanca de acrílico de 3mm de espesor que transmite la fuerza de torsión del motor a una determinada distancia como se muestra en la figura 2.9, la fuerza es transmitida por la palanca de acrílico a una distancia de 10 cm del eje de giro del motor a pasos para posteriormente ser medida por una báscula ubicada en la plataforma de la base de madera.

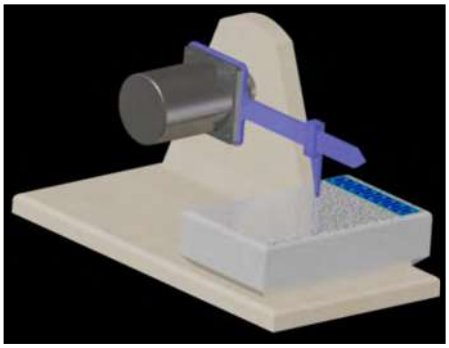


Fig. 2.9 Mecanismo para cálculo del torque.

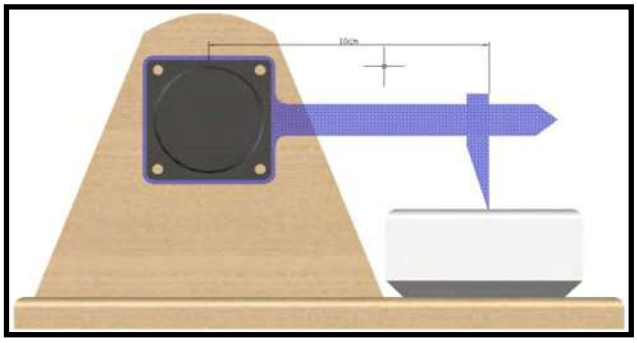


Fig. 2.10 Distancia de aplicación de la fuerza.

El valor obtenido por la báscula es de 0.3 kgf con el punto de apoyo a 10cm (figura 2.10) del eje de giro del motor a pasos por lo que según la fórmula (2.1) para el cálculo del torque los datos quedan:

$$\tau = Fxd \tag{2.1}$$

Al sustituir los valores:

$$\tau = (0.3kgf)x(0.1m) \tag{2.2}$$

$$\tau = 0.03kgf.m \tag{2.3}$$

Ahora bien, para el ingreso de estos valores en el software ANSYS es necesario obtener el equivalente en Newtons-metro por lo que convirtiendo valores tenemos:

$$\tau = \frac{0.03kgf.m\ x\ 9.81N.m}{1kgf.m} \tag{2.4}$$

$$\tau = 0.2943N.m \tag{2.5}$$

.

Debido a que la deformación es mínima es conveniente convertir los valores para la simulación a N.mm para obtener cifras sin exponentes en base 10 por lo tanto:

$$0.2943N.m\left(\frac{1000mm}{1m}\right) = 294.3N.mm \tag{2.6}$$

2.2.2.- SIMULACIÓN DEL SOPORTE DEL EJE "X"

Una vez obtenido el valor del torque en las unidades convenientes es necesaria la consideración del lugar de aplicación de las fuerzas. Debido a que el motor desarrolla un torque sobre los soportes de los motores a pasos (Referente al punto 2.1 de esta tesis), el torque calculado anteriormente es dividido entre cuatro considerando aplicar las fuerzas del torque de forma tangencial al círculo que comprende las 4 cavidades de los tornillos que aseguran la sujeción de los motores a pasos y los soportes (figura 2.11).

$$\tau_e = \frac{294.3N.mm}{4} \tag{2.7}$$

$$\tau_e = 73.575N.mm \tag{2.8}$$

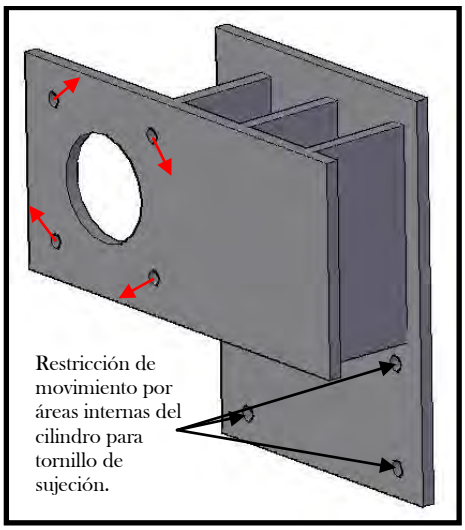


Fig. 2.11 Aplicación de las fuerzas para la simulación.

La restricción de movimiento para la simulación tiene lugar en los 4 orificios de la placa vertical; la restricción está considerada por áreas que conforman al cilindro interior de cada orificio.

Al realizar el experimento para calcular el torque de los motores a pasos se utilizó un brazo de palanca de 10cm de longitud, por lo que para la simulación se calcula un torque equivalente a la distancia entre la flecha del motor a pasos y el orificio de los tornillos de sujeción (figura 2.12) donde se considera la aplicación de la fuerza del torque.

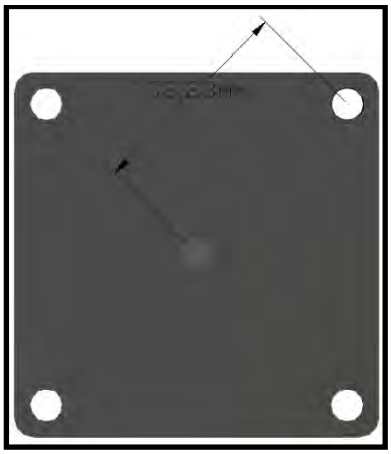


Fig. 2.12 Distancia de aplicación de fuerza de torque.

$$\tau_{final} = \tau_{equivalente} \tag{2.9}$$

$$(F_f)(d_f) = (\tau_e) \tag{2.10}$$

$$(F_f) = \frac{(\tau_e)}{(d_f)} \tag{2.11}$$

$$(F_f) = \frac{(73.575N.mm)}{(33.23mm)} \tag{2.12}$$

$$(F_f) = 2.2141N \tag{2.13}$$

La fuerza final F_f obtenida anteriormente es el valor empleado en la simulación de los soportes de los motores a pasos. Para realizar la simulación

en ANSYS solo quedan pendientes dos datos referentes al material que se va a simular:

El módulo de Young:

$$E = 6\text{GPa (Para el acrílico)[9]}$$

Para conveniencia de los datos de salida la conversión de éste parámetro como se muestra enseguida:

$$6GPa = 6x10^9 \frac{N}{m^2}$$

Por lo que m^2 debe ser convertido a mm^2 ya que es la unidad de longitud que se tomó en cuenta en los cálculos anteriores, por lo tanto:

$$6x10^9 \frac{N}{m^2}\left(\frac{1m^2}{1000000mm^2}\right) = 6000\frac{N}{mm^2}$$

Y por último el coeficiente de Poisson:

$$v = 0.33\ [9]$$

Al tener los datos necesarios para la simulación y correr la simulación se obtienen los datos aproximados de deformación de cada eje así como de los esfuerzos *von Mises*.

Después de la simulación como era de esperarse, se registra que en el soporte del eje "X" la deformación o desplazamiento tiene lugar en el extremo de la placa más retirada del lugar de restricción (figura 2.11) según lo mostrado en la figura 2.13. El valor máximo de desplazamiento es de $0.000478mm$. En cuanto a los esfuerzos *von Mises,* se encuentra la mayor

concentración de esfuerzos en la zona en que se aplicaron las fuerzas como se muestra en la figura 2.14 y teniendo un valor máximo de $0.674347\,\frac{N}{mm^2}$, por lo que se puede determinar que el material es apto para realizar las pruebas necesarias para la continuidad del proyecto.

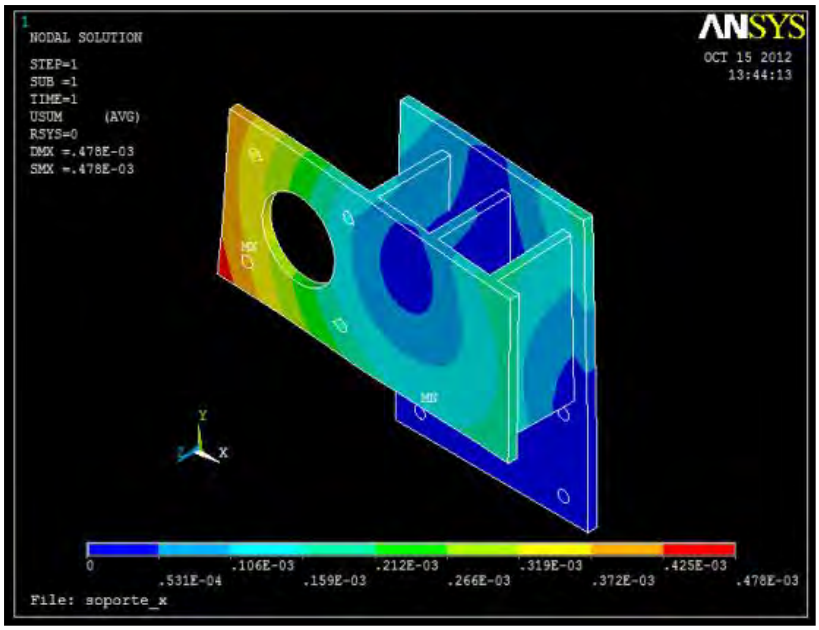


Fig. 2.13 Resultado de simulación de deformación eje "X".

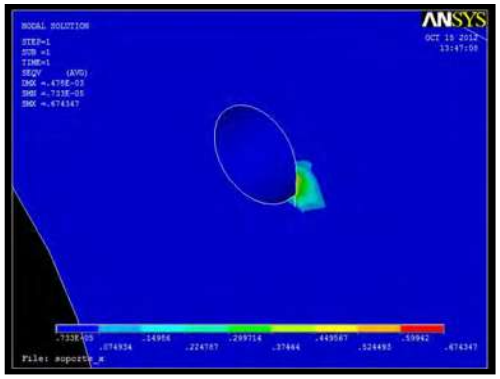


Fig. 2.14 Punto de mayor concentración de esfuerzos (von Mises) eje "X".

Únicamente es importante señalar que en la simulación, la construcción de las piezas mostradas está comprendida por una sola pieza y no existe algún factor que determine el pegamento que se emplea para la unión de los cortes de acrílico, lo que puede alterar ciertos parámetros de la simulación.

2.2.3.- SIMULACIÓN DEL SOPORTE DEL EJE "Z"

Para la simulación del soporte del eje "Z", el cálculo de la fuerza torzora y las consideraciones en el punto de interacción con el soporte, son las mismas que en el soporte del eje "X" a excepción del área de restricción de movimiento que se indicará en el *software* ANSYS, la cual, se determinó en toda el área que se señala en la figura 2.15 debido a que el soporte del eje "Z" tiene contacto en toda esta superficie con la placa de soporte del carro portaherramientas del microtorno y no únicamente en los orificios para los tornillos de sujeción como es el caso del soporte para el motor a pasos del eje "X". Esta observación se aprecia con mayor detalle en la figura 2.15.

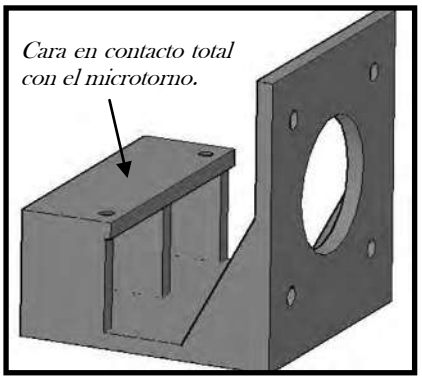


Fig. 2.15 Consideración de restricción de movimiento del eje "Z".

Al desarrollar la simulación se obtuvieron los resultados de deformación y de esfuerzos *von Mises* como se muestran en las figuras 2.16 y 2.17 respectivamente. En la figura 2.16 se observa la mayor deformación en la parte superior izquierda con un valor máximo de $0.00211mm$. En la figura 2.17 se muestran los resultados relacionados con los esfuerzos *von Mises* teniendo como valor máximo $0.384229\frac{N}{mm^2}$, lo que nos permite saber que el soporte es apto para continuar con el desarrollo del proyecto.

Las fuerzas aplicadas sobre el soporte del eje "Z" se aplicaron, al igual que en el soporte del eje "X", sobre el círculo que comprende los orificios de los tornillos de sujeción en forma tangencial.

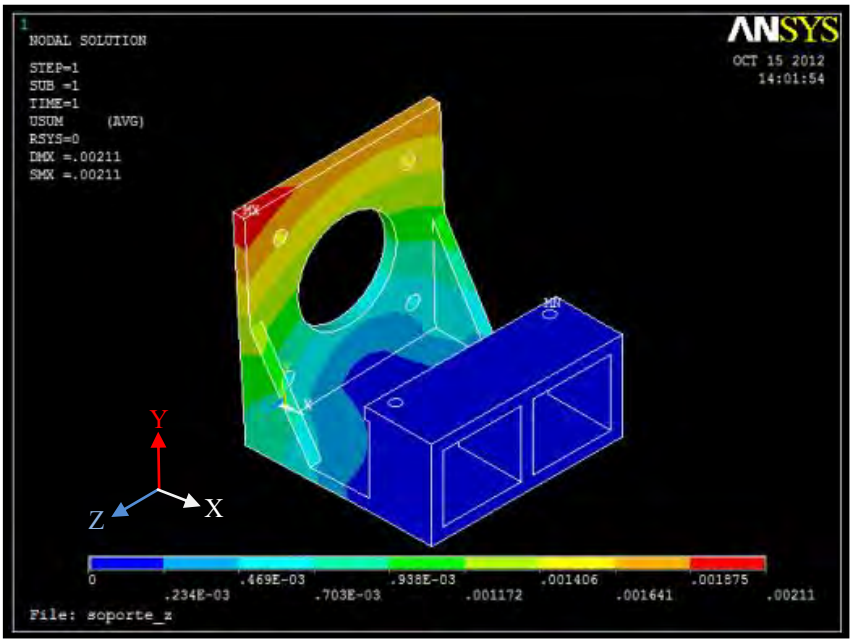


Fig. 2.16 Resultados simulación para la deformación eje "Z".

En la figura siguiente se aprecia que la mayor concentración de esfuerzos *von Mises* se encuentra en el punto donde fue considerada la aplicación de la fuerza torzara.

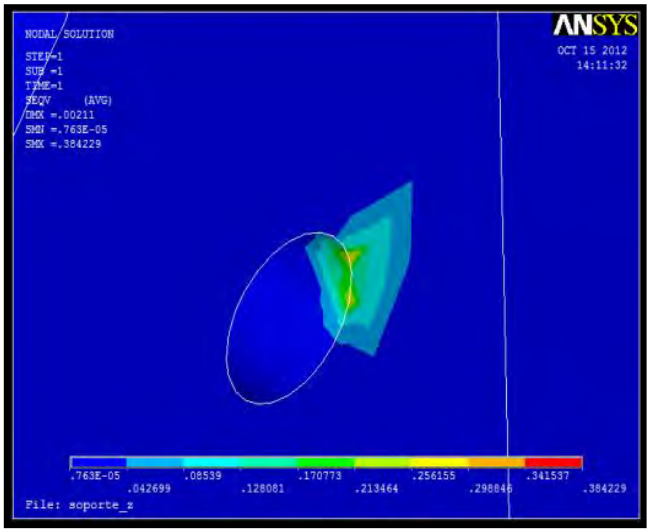


Fig. 2.17 Resultado de simulación para los esfuerzos von Mises eje "Z".

2.3.- DISEÑO DE BRIDAS DE ACOPLAMIENTO

Originalmente el microtorno tenía manivelas para los dos ejes de trabajo acopladas a los tornillos sin fin por medio de un roscado Ø5mm x 19mm y paso de 0.8mm por vuelta, en uno de los extremos posteriormente hace apriete a contratuerca con una tuerca para el roscado descrito, por consiguiente, la sujeción de la brida de acoplamiento tiene que ajustarse a los mismos requerimientos de la forma de sujeción de la manivela (figura 2.18).

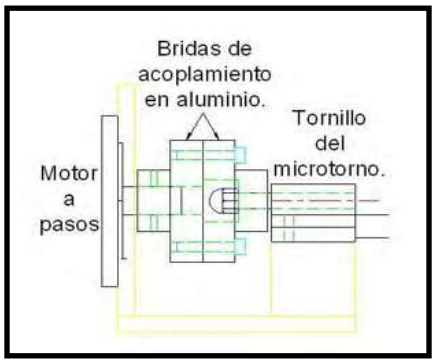


Fig. 2.18 Ensamblaje de Bridas.

Por otro lado, la brida que esta acoplada a los motores a pasos esta maquinada al diámetro de la flecha del los mismos y asegura su sujeción por medio de dos opresores.

2.3.1.- ACOTAMIENTO DE LAS BRIDAS

Las bridas están dimensionadas como se muestra en la figura 2.19, estas bridas son de tipo rígido, por lo que no tienen manera de amortizar los esfuerzos de flexión producidos por el desalineamiento de los ejes donde son montadas, estos esfuerzos sin disipar, provocan una ligera deformación en los soportes de los motores a pasos cuando éstos están en rotación. La elasticidad del acrílico de los soportes, permite absorber los esfuerzos mencionados y después de la simulación realizada en el apartado 2.2.2, se observa que la

magnitud de la deformación en cada soporte, tiene valores muy pequeños que permiten continuar con el propósito principal del proyecto.

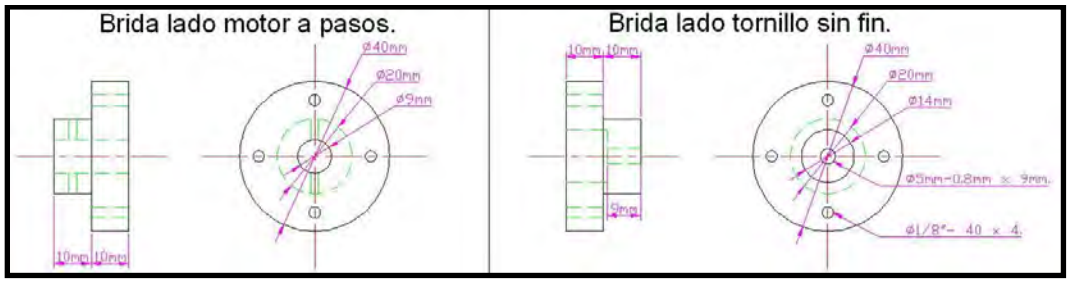


Fig. 2.19 Dimensionamiento de las bridas.

Una vista general del orden de ensamblaje de los componentes mencionados se observa en la figura 2.20, se pueden apreciar los soportes en acrílico referentes al punto 2.1 de esta tesis.

Fig. 2.20 Ensamblaje de los componentes mecánicos.

CAPÍTULO III

SIMULADOR DESARROLLADO EN *VISUAL BASIC 6.0*.

En busca de un método capaz de realizar una simulación previa al maquinado con la finalidad de corroborar el dimensionamiento de la pieza maquinada y los movimientos del buril, es desarrollada una aplicación en el *software Visual Basic 6.0* que por medio de ventanas contenidas dentro del mismo, muestra información esencial para la ejecución de maquinados.

3.1.- PARTES DEL SIMULADOR

El simulador está agrupado de tal forma que todas sus partes pueden ser ubicadas fácilmente por: información, mando y captura de coordenadas.

Fig. 3.1 Vista general del simulador.

3.1.1.- INFORMACIÓN DEL SIMULADOR

El simulador dota al operador de información esencial para realizar los maquinados tal como la ubicación del buril, número de maquinados específicos, velocidad de avance en el control manual (figura 3.1).

INFORMACIÓN DE UBICACIÓN DEL BURIL

Es indispensable conocer la ubicación de la herramienta de corte para evitar daños, tanto a la pieza por maquinar como a los componentes del

microtorno, por lo que el simulador cuenta con las ventanas de información para el monitoreo de la ubicación del buril, en estas ventanas se muestra la ubicación del buril en número de pasos respecto al cero máquina y en su equivalencia en pulgadas.

INFORMACIÓN GENERAL

El simulador informa al operador de las dimensiones en número de pasos y su equivalente en pulgadas de la pieza antes del maquinado, luego de realizar un rectificado y un refrentado por medio del botón CAPTURA CERO PIEZA. El rectificado y el refrentado se pueden desarrollar de una forma semiautomática por medio de los botones del control manual o en su defecto con la subrutina programada en el apartado de MANDO RÁPIDO del cual se profundiza más adelante.

INFORMACIÓN DE NÚMERO DE MAQUINADOS

En la parte inferior de la ventana de INFORMACIÓN GENERAL del simulador, se muestra el número de maquinados programados teniendo como límite 4 maquinados de cada tipo.

INFORMACIÓN DE VELOCIDAD DEL CONTROL MANUAL

Cuando el modo CONTROL MANUAL está activado, el simulador permite el ajuste de la velocidad entre paso y paso de los movimientos en los ejes “X” y “Z”, la modificación de este parámetro se realiza con el botón VEL. MANUAL y el valor registrado se muestra en la ventana de CONTROL MANUAL. El valor ingresado está dimensionado en milisegundos y el rango

admisible para este ajuste es de 1 – 255 milisegundos. Este ajuste tiene como objetivo el no cometer errores, como colisionar el buril con la pieza a maquinar al acercar el buril de forma semiautomática hacia donde será el cero pieza, para después memorizarlo con el botón CAPTURA CERO PIEZA .

3.2.- FUNCIONES DEL SIMULADOR

El simulador es capaz de realizar un trazo previo de la pieza maquinada, así como la exhibición de los movimientos que realizará el buril para la obtención del maquinado deseado a través de la pantalla de graficado del maquinado, la cual, muestra el lugar del cero máquina y el área donde la herramienta de corte tiene libertad de desarrollar todos sus movimientos. También permite la configuración de algunos parámetros extras, como profundidad de corte entre pasadas, viaje rápido del buril a coordenadas concretas, autocalibración del buril y por último la obtención de las coordenadas y datos necesarios para el trazo de la pieza con los desbastes requeridos. La manipulación de estas funciones se obtienen con los botones contenidos en las ventanas de CONTROL MANUAL, CAPTURA DE COORDENADAS y MANDO RAPIDO del simulador (figura 3.2).

Fig. 3.2 Descripción de cuadros de funciones del simulador.

3.2.1.-FUNCIÓN DE AUTOCALIBRACIÓN DEL BURIL

La autocalibración es una subrutina programada en el simulador que tiene dos objetivos específicos dentro de la automatización del microtorno:

1.- La ubicación del buril en las coordenadas *HOME.*

2.- El cálculo de los "pasos muertos" en el cambio de sentido del movimiento en cada eje.

Después de "correr" el simulador, inicialmente sólo está habilitado el botón de INICIO GENERAL para no dar oportunidad de realizar ninguna otra operación del simulador antes de garantizar el posicionamiento del buril en las coordenadas *HOME* y referenciar así las coordenadas subsecuentes para los movimientos (figura 3.3). Al presionar el botón de INICIO GENERAL, el simulador ordena a los motores a pasos girar de tal forma que los valores en los dos ejes del microtorno se incrementen hasta alcanzar la coordenada *HOME* del eje "Z", para cuando este eje esté calibrado da paso al incremento de valores en el eje "X". Al terminar el ajuste del eje "X", la rutina de auto ajuste termina habilitando botones para instrucciones siguientes.

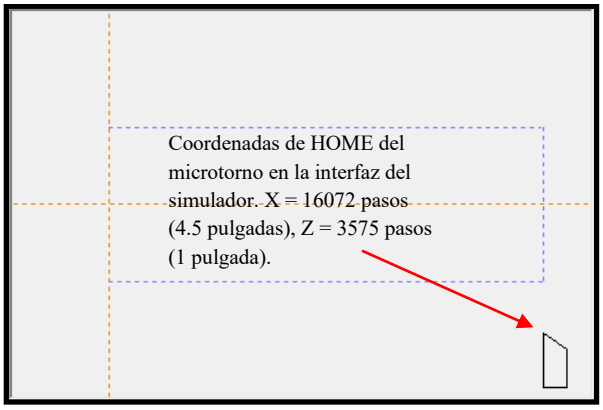


Fig. 3.3 Ubicación HOME del simulador.

El segundo objetivo de la rutina de autocalibración es necesaria debido a que, en los cambios de sentido de los movimientos en cada eje, se tiene un juego entre el tornillo sin fin y la tuerca incorporada en el carro portaherramienta, produciendo pasos del motor en los que no se tiene desplazamiento axial por parte del buril ("pasos muertos"); esto sucede debido que el microtorno no está equipado con tornillos embalados que no tienen juego entre los dientes de su roscado y su tuerca.

El cálculo de los "pasos muertos" tiene lugar antes de que la autocalibración termine, es decir, cuando la rutina de autocalibración comienza, el motor a pasos del eje "Z" comienza a girar aumentando el valor de la coordenada de este eje hasta presionar un interruptor que está ubicado en su fin de carrera (coordenada *HOME* del eje "Z"), lo anterior proporciona una señal de 5 volts que recibe el microcontrolador y a su vez, es enviada hacia el simulador provocando que el motor a pasos invierta el sentido de giro hasta que se deje de presionar el interruptor y la señal recibida por el microcontrolador se pierda, es aquí cuando este motor a pasos deja de funcionar y da inicio a la misma rutina del motor a pasos encargado del movimiento del eje "X". Cuando el sentido de giro es invertido el número de pasos necesario para la desactivación del interruptor es contabilizado en una variable que posteriormente será sumada en los cambios de sentido de rotación de los

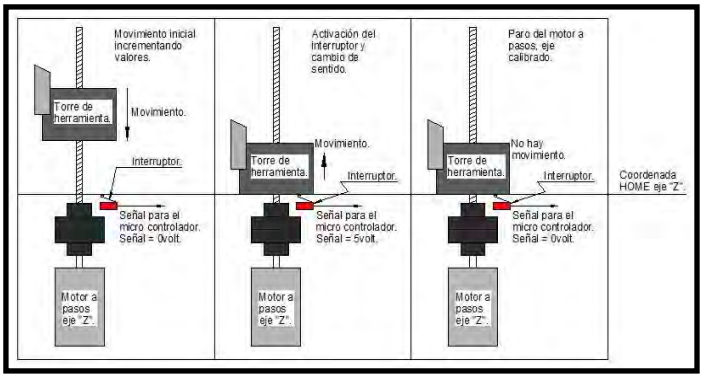


Fig. 3.4Representación del cálculo de pasos muertos.

motores a pasos. Al sumar el número de “pasos muertos” en los cambios de rotación se tiene una ubicación más precisa del buril (figura 3.4).

3.2.2.- CONTROL MANUAL

La ventana de control manual (figura 3.5) permite manipular el buril en tiempo real para realizar el rectificado y refrentado de la pieza a maquinar, esto es sumamente necesario para el trazo de las dimensiones iniciales de la pieza y referenciar que valores son admisibles para los trazos de los maquinados deseados.

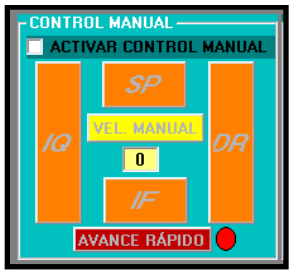


Fig. 3.5 Control manual.

CONTROL MANUAL EN EL RECTIFICADO Y REFRENTADO

Al terminar la rutina de autocalibración el simulador está listo para montar el material primario que será maquinado (figura 3.6), al tener el material primario montado en el usillo, éste debe de refrentarse y rectificarse para obtener las coordenadas cero pieza. Para esta tarea, inicialmente es necesario practicar un refrentado en el material primario, para lo cual, se acerca el buril hasta el punto donde roce la cara del material primario para después desbastarla hasta tener una geometría plana, de aquí se obtiene el valor de la coordenada del eje “X” que será usada para la ubicación del buril en las coordenadas “cero pieza” del eje “X”. Posteriormente el buril regresa a la última coordenada empleada en el refrentado para iniciar la tarea del

rectificado del material primario. De la misma manera que en el refrentado, el buril se acerca al material primario para desbastar material pero ahora en sentido axial hasta conseguir un contorno recto. La última coordenada empleada en este desbaste será la empleada en la ubicación del buril en las coordenadas "cero pieza" sobre el eje "Z", luego de este proceso se tienen ambas coordenadas para el trazo de la pieza refrentada y rectificada (figura 3.7).

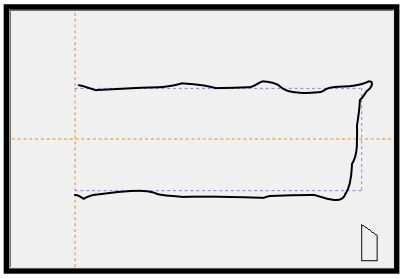


Fig. 3.6 Pieza antes de rectificar y refrentar.

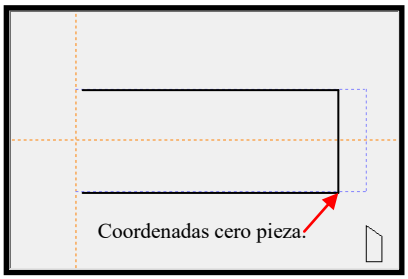


Fig. 3.7 Pieza después de rectificar y refrentar.

BOTONES DE CONTROL DE EJES "X" Y "Z"

Para lograr un refrentado y rectificado de la pieza inicial se vuelve necesario el movimiento del buril a través de los eje "X" y "Z" de forma independiente. Estos movimientos se realizan con los botones SP, IF, IQ y DR contenidos en el apartado de control manual (figura 3.5) donde se puede manipular el movimiento en uno de los dos ejes a la vez y con el ajuste de velocidad de esta función es posible la obtención de acercamientos en forma lenta o rápida dependiendo de las necesidades del maquinado. El botón VEL. MANUAL permite ingresar un valor en el rango de 1-255 en milisegundos que determina la pausa entre paso y paso en relación al botón de movimiento que se esté presionando. Este valor es inversamente proporcional a la velocidad de desplazamiento que reflejará el buril. Para un movimiento rápido con estos botones se presiona el botón "AVANCE RAPIDO" y el valor de la

pausa será 1 milisegundo, al desaccionarlo, el valor de la pausa entre los pasos de los motores será el último ingresado por el operador.

3.2.3.- CAPTURA DE COORDENADAS

El simulador necesita valores para procesar y referenciarse tales como las coordenadas "cero pieza", las coordenadas *"home"*, y las que trazarán algún maquinado y las coordenadas para el uso de la ventana de "mando rápido", estos datos se obtienen por medio de la ventana "captura de datos" (figura 3.8).

Fig. 3.8 Captura de datos.

CAPTURA CERO PIEZA	Memoriza y traza las coordenadas cero pieza.
CILÍNDRICO	Requiere coordenadas de maquinado cilíndrico.
CÓNICO	Requiere coordenadas de maquinado cónico.
ESFÉRICO	Requiere coordenadas de maquinado esférico.
HOME	Manda la herramienta a Coordenadas HOME.

Tabla 3.1 Funciones de los botones de captura de datos.

CAPTURA CERO PIEZA

Después de refrentar y rectificar la pieza a maquinar el buril debe ser ubicado en los valores máximos de la pieza tanto en el eje "X" como en el eje "Z" para memorizarlos como las coordenadas "cero pieza" y así determinar los valores máximos posibles de los maquinados que se programarán. Hasta este momento después de presionar el botón CAPTURA CERO PIEZA es que el simulador dibuja la geometría inicial de la pieza a maquinar.

AGREGAR MAQUINADOS.

Al capturar lo que serán las coordenadas “cero pieza”, el simulador está habilitado para determinar si las coordenadas de programación de los siguientes maquinados están dentro de las dimensiones de la pieza, si esta condición se cumple, la programación de los maquinados “correrá”, de lo contrario el simulador emite un aviso de “coordenadas invalidas”.

BOTÓN CILÍNDRICO.

El botón CILÍNDRICO permite realizar maquinados con una geometría de este tipo, para lo cual al presionar este botón el simulador requerirá información de las coordenadas clave para el previo trazo como se muestra en la siguiente figura.

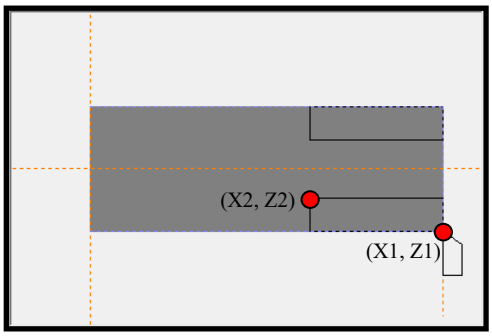


Fig. 3.9 Coordenadas para maquinado cilíndrico.

Con estos pares de coordenadas es posible el trazo del maquinado cilíndrico así como también es posible la ejecución de la programación para los recorridos del buril para este maquinado.

BOTÓN CÓNICO

A partir del maquinado cónico se vuelve necesario el empleo del cálculo con interpolaciones para la determinación de la coordenada final del maquinado en función de la profundidad de corte. El tema del cálculo de interpolaciones se abordará más adelante.

El botón CÓNICO está restringido para trazar y maquinar piezas con la geometría mostrada en la figura 3.11, debido a que el microtorno no cuenta con cambio de una herramienta de corte a la derecha, si se intenta realizar un maquinado restringido se corre el riesgo de una colisión entre la herramienta de corte y la morfología que no se desea tocar de la pieza a maquinar.

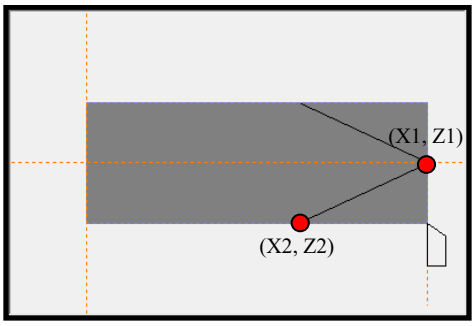


Fig. 3.10 Maquinado cónico válido.

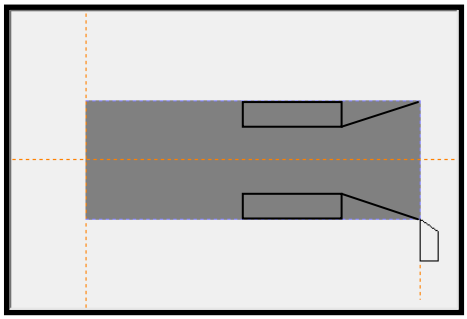


Fig. 3.11 Maquinado cónico no permitido.

Con el par de coordenadas que se ilustra en la figura 3.11, se puede trazar el maquinado y realizar la interpolación necesaria para los movimientos del buril hasta alcanzar la morfología del trazo.

BOTÓN ESFÉRICO

El maquinado esférico es el más complejo de realizar a la hora de desarrollar las ecuaciones para su interpolación debido a que se utilizan

ecuaciones de segundo grado, lo que en ciertos rangos arroja dos resultados para una misma posición en determinada coordenada, lo que a su vez es aprovechable para realizar maquinados esféricos cóncavos y convexos. La determinación de un maquinado cóncavo o convexo tiene lugar al ingresar el dato del radio, si el valor del dato es positivo, tanto el trazo como el maquinado serán convexos y si el valor del dato es negativo serán cóncavos.

Es muy importante el corroborar la existencia de los maquinados esféricos, esto hace referencia a la comprobación de los datos entre punto inicial (X1, Z1), punto final (X2, Z2) y radio sean congruentes ya que de lo contrario el simulador tendrá un conflicto con un valor del radio menor al posible para el trazo de un maquinado esférico. Este inconveniente puede ser resuelto al realizar el trazo del maquinado previamente en algún *software* que sea capaz de trazar los datos del maquinado planeado y se pueda obtener el dato del radio, un ejemplo del *software* puede ser *AutoCAD*.

MAQUINADO ESFÉRICO CONVEXO

Para realizar un maquinado esférico convexo (figura 3.12) es necesario oprimir el botón ESFÉRICO y el simulador responde con una ventana donde se requieren los datos X1, Z1, X2, Z2 y el radio con valor positivo.

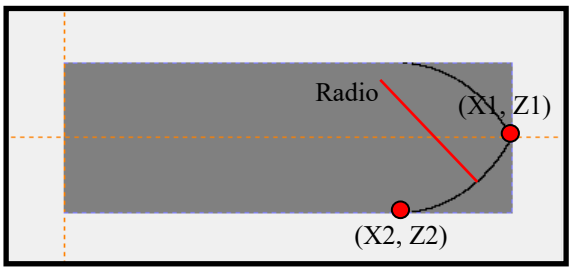


Fig. 3.12 Maquinado esférico convexo.

MAQUINADO ESFÉRICO CÓNCAVO

De manera similar, para el maquinado esférico cóncavo es necesario ingresar los parámetros mencionados en el punto anterior a diferencia de que el valor ingresado del radio tiene que ser negativo, de esta manera el simulador interpreta que el maquinado requerido tiene concavidad y se obtiene un maquinado con la forma de la figura 3.13.

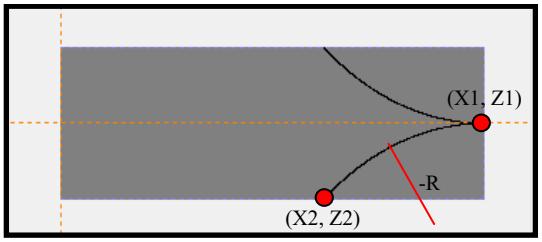


Fig. 3.13 Maquinado esférico cóncavo.

BOTÓN HOME

El botón HOME está programado para enviar el buril de forma automática a las coordenadas *"home"* (figura 3.3). Puede ser utilizado en el tiempo que sea del uso del simulador, la única restricción de este botón es a la hora de que el simulador esté en tiempo de ejecución de algún maquinado para lo cual se tendría que oprimir primeramente el botón PARO DE EMERGENCIA.

3.2.4.- MANDO RÁPIDO

El apartado de mando rápido es empleado para mandar el buril a las coordenadas X, Z que se ingresen, al especificar este par de coordenadas el simulador ordena a los motores a pasos igualar las coordenadas actuales con las ingresadas en este apartado de una manera automática, evitando así, que el

operador mantenga presionado algún botón para el envío de la herramienta de corte a alguna coordenada específica. El empleo de esta utilidad resulta muy ventajosa al realizar desbastes rectos como lo es el caso de los refrentados y rectificaciones iniciales.

Fig. 3.14 Mando rápido.

BOTÓN PARO DE EMERGENCIA

El botón PARO DE EMERGENCIA tiene como objetivo el detener cualquier movimiento automático que esté realizando el microtorno.

BOTÓN MAQUINAR

Para ejecutar los maquinados previamente programados en el simulador, es necesario oprimir el botón MAQUINAR (figura 3.15), una vez que se presione éste botón el simulador se encargará de calcular y realizar todos los maquinados previamente programados.

El simulador está programado para jerarquizar los maquinados en el orden de simplicidad, ésto es, primeramente todos los maquinados cilíndricos, posteriormente realiza todos los cónicos y por último los maquinados esféricos como se muestra en diagrama del apartado del anexo 1.

En el caso de que no existan maquinados cilíndricos el simulador está programado para verificar si se tiene algún maquinado cónico, si la respuesta es afirmativa, el simulador ejecutará los maquinados cónicos existentes, de lo contrario hará la misma tarea para el maquinado esférico.

Una vez oprimiendo el botón MAQUINAR, se observará que varias opciones del simulador quedarán inhabilitadas con el objeto de evitar algún fallo en la pieza maquinada. Para interrumpir algún maquinado en proceso será suficiente con oprimir el botón PARO DE EMERGENCIA.

Fig. 3.15 Botón para iniciar los maquinados.

3.2.5.- MAQUINADO PASO POR PASO

Este selector tiene el propósito de seguir el maquinado paso por paso, es decir, el simulador ejecuta el primer maquinado programado y al terminar pregunta si desea seguir con el siguiente maquinado.

Fig. 3.16 Opción maquinado paso a paso.

3.3.- PROGRAMACIÓN

La comunicación entre el simulador y la interfaz de control es vía USB 2.0, para lograr esta comunicación se utilizó la herramienta *EasyHid USB Wisard* del *software Micro Code Studio 4.0.0.0,* la cual se describe de manera profunda en el apartado 5.3 de esta tesis. Por el momento se menciona únicamente que esta herramienta genera una carpeta con archivos que son necesarios para la comunicación entre la computadora y ciertos tipos de microcontroladores por medio del puerto USB. La programación integra en *Visual Basic 6.0* se muestra en el anexo 7.

Uno de los archivos generados por la herramienta antes mencionada tiene la programación inicial para el *software Visual Basic 5.0* que también es ejecutable en la versión *6.0* y que es el empleado en esta tesis.

3.3.1.- PROGRAMACIÓN INICIAL PARA COMUNICACIÓN POR USB 2.0 PARA *VISUAL BASIC 6.0*

Una vez ejecutada la herramienta *EasyHid USB Wisard* del *software Micro Code Studio 4.0.0.0*, se abre el archivo válido para *Visual Basic 6.0* y se encuentra la programación como se muestra en el anexo 2 de esta tesis. Ésta programación como ya se ha mencionado son únicamente, líneas de programación iniciales que sirven para la comunicación por medio del puerto USB y son generadas por la herramienta mencionada inicialmente en este párrafo. Para desarrollar la aplicación en *Visual Basic 6.0* se debe abrir el código generado por *EasyHid USB Wisard.* En este archivo existen dos carpetas, la primera llamada formularios, que contiene el *Main Form* o formulario principal en donde se desarrolla el programa, la segunda carpeta

llamada “Módulos” contiene unas líneas de código que generan un archivo con extensión dll. Un dll es un acrónimo de *“Dynamic Linking Library”,* término con el que se refiere a los archivos con código ejecutable que se cargan bajo demanda del programa por parte del sistema operativo para realizar tareas específicas. Este módulo contiene el dll para el controlador *HID,* el cual se enlaza con el *Firmware* del microcontrolador para poder realizar la comunicación USB. No se necesita hacer ningún cambio en este módulo[10].

En el formulario principal se encuentran otras líneas de código, que interactúan con el código del módulo anterior para poder realizar la comunicación USB. Estas líneas de código son las encargadas de realizar el proceso de enumeración USB, es decir, pedirle al microcontrolador alguna información como su ID, a que vendedor pertenece, además detecta que se ha conectado un dispositivo USB con características de *HID,* como es el caso del PIC 18F4550, luego lo configura y lo deja listo para que pueda intercambiar información con el computador[10].

Al igual que el microcontrolador, en este programa existe un número determinado de *Buffers,* tanto de entrada como de salida, en este caso se tiene 9 *Buffers* que van desde cero hasta el ocho respectivamente. El *BufferIn(0)* y el *BufferOut(0)*,no se pueden utilizar, ya que estos *Buffers* contienen el reporte *ID* por lo tanto los *Buffers* que se pueden utilizar para transmisión de datos son desde el uno al ocho. En *Visual Basic 6.0* los *Buffers* son mayores en uno a los *Buffers* del microcontrolador, por ejemplo el *Buffer(1)* del PIC corresponde al *BufferIn(2)* de *Visual Basic 6.0.*

Gracias a que la programación en *Visual Basic 6.0* es orientada a objetos, no es necesaria la redacción de muchas líneas de código, por ende pueden hacerse algunas pruebas con relativa rapidez al asignarle a un botón la tarea de transmitir un dato al microcontrolador, para esto es necesario asignarle un valor a uno de los *Buffers,* por ejemplo: *BufferOut(8)=1*, ahora para transmitir el dato debe colocar el siguiente código:

hidWriteEx VendorID, ProductID, BufferOut(0)

Y para recibirlo en el microcontrolador queda alojado en el *Buffer(7).* El valor adquirido en este *Buffer(7)* del microcontrolador, ahora puede ser manipulado de la forma que convenga.

En cambio cuando el microcontrolador envíe un dato al computadora, es necesario colocar el dato en el *Buffer* del microcontrolador, es decir, por ejemplo USB*Buffer(3)=2*, luego se ejecuta el comando *DoUSBOut* en ese momento se dirige a esta subrutina y envía el dato hacia el computador para recibirlo en el *BufferIn(4)*[10].

3.4.- MATEMÁTICA PARA LA INTERPOLACIÓN DE COORDENADAS

Interpolación es el proceso mediante el cual, conocidos los valores que toma una función en dos puntos (A y B), se determina con cierto grado de exactitud los valores de un tercer punto (C) comprendido entre A y B. Esta definición es empleada para el desarrollo de dos de los maquinados para los cuales está programado el simulador, el cónico y el esférico; debido a que en

el maquinado cilíndrico tiene los viajes de la herramienta bien establecidos por los dos pares de coordenadas que definen el trazo del maquinado.

3.4.1.- MAQUINADO CILÍNDRICO

Como se mencionó anteriormente en este tipo de maquinado no es necesario el desarrollo del cálculo de alguna interpolación debido a que con las coordenadas del punto A y de las coordenadas del punto B es suficiente para determinar hasta qué puntos debe llegar la herramienta de corte (Referente a la figura 3.17).

Este maquinado también puede ser utilizado para las tareas de refrentado y rectificado de una pieza, sólo hay que recordar que el simulador tiene capacidad de almacenar 3 maquinados de este tipo.

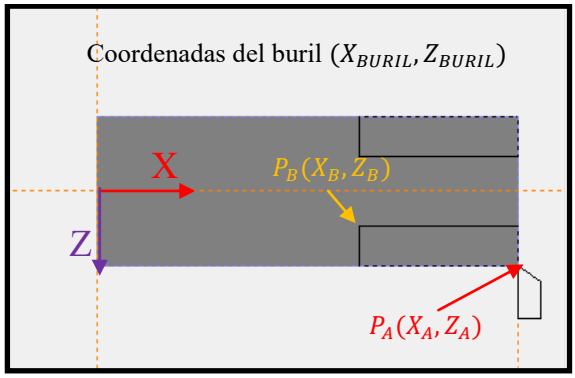


Fig. 3.17 Coordenadas para maquinado cilíndrico.

Una vez que las coordenadas de la herramienta de corte son iguales a las coordenadas X_A, Z_A del trazo programado, únicamente es necesario igualar ahora la coordenada X_{BURIL} de la herramienta de corte hasta la coordenada X_B del punto B para posteriormente sacar el buril unos cuantos milímetros (en la coordenada Z_{BURIL}) para el desahogo de la viruta cortada y nuevamente posicionar el buril en la coordenada X_A, subsecuentemente a esto la

herramienta debe de desarrollar un movimiento de profundidad de corte el cual tiene el valor programado por el usuario del simulador. Este bucle es repetido hasta que la coordenada Z_{BURIL} sea igual a la coordenada Z_B del maquinado programado para desarrollar una última pasada del bucle anterior para dar el terminado de la pieza y finalmente, si no existen otros maquinados programados, la herramienta de corte es llevada hasta las coordenadas *HOME*. La rutina descrita anteriormente se resume en el diagrama del anexo 3 de esta tesis. La profundidad de corte programada por el operador es la misma para todos los maquinados programados, ya que ésta se determina en base a las características del material a maquinar.

3.4.2.- INTERPOLACIÓN PARA EL MAQUINADO CÓNICO

Para realizar el maquinado cónico la situación cambia debido a que la coordenada X_{NUEVA} no es la misma entre el punto A y el punto B a medida que la coordenada del buril Z_{BURIL} se acerca a la coordenada Z_A (figura 3.18), por lo que es necesario el cálculo de esta nueva coordenada para lo cual se emplean ecuaciones relacionadas con geometría analítica como se muestra a continuación:

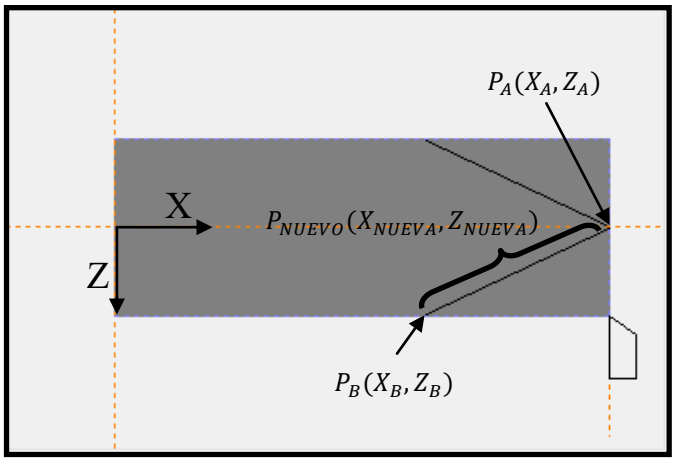


Fig. 3.18 Coordenadas interpolación maquinado cónico.

Debido a que los únicos datos disponibles son las coordenadas del punto A (X_A, Z_A) y del punto B (X_B, Z_B), los valores de estos datos son sustituidos en la ecuación (4.1) para el cálculo de la pendiente de una recta:

$$m = \frac{Z_B - Z_A}{X_B - X_A} \tag{4.1}$$

Así el valor de la pendiente m es empleado en la ecuación de la recta determinada por un punto y su pendiente para obtener:

$$Z_A = mX_A + a \tag{4.2}$$

En la ecuación (4.2) los valores de a para sustituirla después en la misma ecuación con la diferencia de que en el lugar de la abscisa y la ordenada estarán los valores que cambiarán en función de la profundidad de corte:

$$a = Z_A - mX_A \tag{4.3}$$

El cálculo de a de la ecuación (4.3) se realiza con valores que son proporcionados desde la programación del simulador por lo que la matemática desarrollada hasta este momento no tiene conflicto.

La incógnita necesaria para el maquinado es una coordenada en el eje de las abscisas (X_{NUEVA}) en relación a una ordenada (Z_{NUEVA}) que está determinada por la profundidad de corte, por lo que empleando la misma ecuación (4.3) pero con las variables que estarán cambiantes durante el maquinado quedaría:

$$X_{NUEVA} = \frac{Z_{NUEVA} - a}{m} \tag{4.4}$$

Como Z_{NUEVA} es una variable que cambia en función de la profundidad de corte, cada que ésta obtenga un valor distinto en cada ciclo del bucle de maquinado cónico (Anexo 4) se obtiene un valor para X_{NUEVA} y finalmente se tiene el valor de la coordenada final para el viaje del buril en el eje "X" del maquinado. El diagrama de flujo del anexo 4 muestra la lógica del cálculo.

3.4.3.- INTERPOLACIÓN DEL MAQUINADO ESFÉRICO

Como se ha mencionado anteriormente en esta tesis, los datos disponibles para el cálculo de la interpolación de este maquinado, son el punto inicial (X_a, Z_a), punto final (X_b, Z_b) y el valor del radio; éste último puede ser positivo para maquinados convexos o negativo para maquinados cóncavos, al emplear esta ecuación de segundo grado (4.5), implica que tiene dos soluciones posibles, es esta misma situación la que es aprovechable para realizar maquinados cóncavos y convexos; uno de los resultados satisface el maquinado cóncavo y el otro el convexo.

$$(X - h)^2 + (Z - k)^2 = r^2 \qquad (4.5)$$

Inicialmente dados los datos de inicio, fin y radio del segmento del círculo, se determinan las coordenadas del centro del círculo, generando dos ecuaciones con los datos disponibles para igualarlas y reducir los términos semejantes (4.6), (4.7):

$$(X_a - h)^2 + (Z_a - k)^2 = r^2 \qquad (4.6)$$
$$(X_b - h)^2 + (Z_b - k)^2 = r^2 \qquad (4.7)$$

Desarrollando cuadrados para reducción de términos:

$$X_a{}^2 - 2X_a h + h^2 + Z_a{}^2 - 2Z_a k + k^2 - r^2 = 0 \quad (4.8)$$

$$-X_b{}^2 + 2X_b h - h^2 - Z_b{}^2 + 2Z_b k - k^2 + r^2 = 0 \quad (4.9)$$

Al reducir términos queda:

$$2h(X_b - X_a) + 2k(Z_b - Z_a) = -X_a{}^2 + X_b{}^2 - Z_a{}^2 + Z_b{}^2 \quad (4.10)$$

Despejando h queda:

$$h = \frac{-2k(Z_b - Z_a) - X_a{}^2 + X_b{}^2 - Z_a{}^2 + Z_b{}^2}{2(X_b - X_a)} \quad (4.11)$$

Donde para facilitar la ecuación (4.11) se sustituye:

Símbolo.	Valor.
α	$-2(Z_b - Z_a)$
β	$-X_a{}^2 + X_b{}^2 - Z_a{}^2 + Z_b{}^2$
γ	$2(X_b - X_a)$
δ	α/γ
ε	β/γ

Tabla 3.2 Equivalencias de ecuación 4.9.

Por lo que la ecuación 4.11 queda como se muestra a continuación:

$$h = \delta k + \varepsilon \quad (4.12)$$

Al sustituir la ecuación 4.12 en la ecuación 4.8, desarrollando operaciones y organizando términos, queda:

$$(\delta^2 + 1)k^2 + 2(X_a\delta + \delta\varepsilon - Z_a)k + (X_a - 2\varepsilon)X_a + \varepsilon^3 + Z_a{}^2 - r^2 = 0 \quad (4.13)$$

La ecuación 4.13 es resuelta con la fórmula general para ecuaciones de segundo grado 4.14:

$$k_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{4.14}$$

Y las dos soluciones posibles están determinadas por las ecuaciones (4.15) y (4.16):

$$k_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \tag{4.15}$$

$$k_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \tag{4.16}$$

Una vez aplicadas las ecuaciones 4.15 y 4.16 se obtienen 2 resultados de k, que ayudarán a determinar las dos coordenadas del centro del círculo posibles, después de sustituirlas en la ecuación 4.5. Cuando se programa un maquinado esférico y el radio es negativo, la coordenada del centro del círculo es k_1 generando un maquinado cóncavo, y al programar un maquinado esférico con el valor del radio positivo la coordenada del centro del circulo k toma el valor de k_2 para generar un maquinado convexo. Para calcular las coordenadas del centro del círculo h_1 y h_2, basta con sustituir los valores calculados k_1 y k_2 en la ecuación 4.5 respectivamente, y así se cuenta con el valor de los dos pares de coordenadas de los centros de los círculos posibles (figura 3.19).

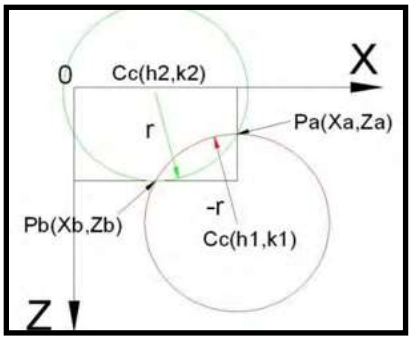


Fig. 3.19 Maquinados esféricos posibles.

INTERPOLACIÓN MAQUINADO ESFÉRICO CONVEXO

Cuando el operador del simulador genera la programación de un maquinado esférico, el simulador mostrará un aviso requiriendo la información que es necesaria primeramente para el trazo del maquinado, y posteriormente para el cálculo de las interpolaciones de los viajes de la herramienta. El simulador pedirá el punto inicial ($P_a(X_a, Z_a)$), punto final ($P_b(X_b, Z_b)$) y el valor del radio (figura 3.20).

Para generar un maquinado convexo el valor del radio debe de ser un valor positivo, de manera que el simulador calculará las coordenadas del centro del círculo necesarias para el maquinado convexo. Cabe mencionar que los puntos inicial y final tienen que estar dentro del área donde existe el material, de lo contrario, el simulador mostrará un aviso de "coordenada no válida".

Cuando el simulador ha calculado las coordenadas del centro del círculo, se emplea la ecuación 4.5 para hacer los despejes correspondientes y que nos quede la ecuación 4.15, que será la utilizada en la programación de *Visual Basic 6.0* para el cálculo de la coordenada "X_N".

$$X_N = \sqrt{r^2 - \left(Z_N - k_{1,2}\right)^2} + h_{1,2} \tag{4.15}$$

El valor de "Z_N" es la diferencia entre la profundidad de corte programada previamente por el operador, y el valor de la coordenada de la ubicación del buril en el eje "Z", cuando "Z_N" ha sido calculada, se emplea la ecuación 4.15 para determinar la coordenada "X_N".

"X_N" Tiene el valor de la coordenada a la que el buril tiene que llegar para dar la geometría programada, y este proceso se repite hasta que "Z_N" sea igual a la coordenada "P_a" (ver figura 3.20).

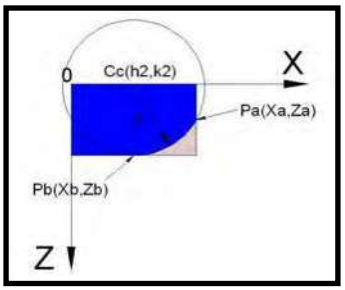


Fig. 3.20 Datos para maquinado convexo.

INTERPOLACIÓN MAQUINADO ESFÉRICO CÓNCAVO

Para realizar un maquinado esférico cóncavo, la única diferencia respecto al maquinado esférico convexo, es el momento de proporcionar los datos al simulador. Éstos serán, punto inicial ($P_a(X_a, Z_a)$), punto final ($P_b(X_b, Z_b)$) y el valor del radio anteponiendo el signo "-" (menos). De esta manera el simulador interpreta que el maquinado requerido tiene concavidad (figura 3.21). La tarea de interpolación es calculada con la misma ecuación 4.15 del maquinado esférico convexo, con la diferencia de que las coordenadas del centro del circulo son los valores almacenados en las variables del centro del círculo h_1, k_1.

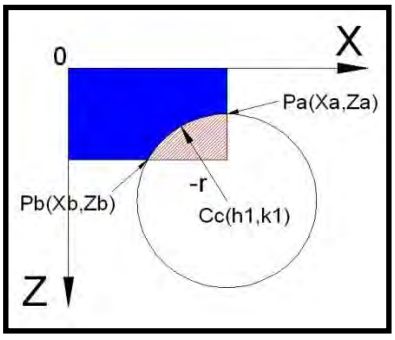


Fig. 3.21 Datos para maquinado cóncavo.

CAPITULO IV

PROGRAMACIÓN DEL MICROCONTROLADOR PIC 18F4550.

Para la recepción de los datos enviados por el simulador hacia los motores a pasos es necesario la implementación de un arreglo electrónico (interfaz), que reciba cualquier información proveniente del simulador, la procese, y luego la amplifique para proveer la potencia necesaria a los motores a pasos. La interfaz electrónica tiene como cerebro al microcontrolador PIC 18F4550 que tiene la peculiaridad de soportar la comunicación por puerto USB 2.0.

Un microcontrolador es un circuito integrado que incluye en su interior las tres unidades funcionales de una computadora: unidad central de procesamiento, memoria y periféricos de entrada y salida.

Los microcontroladores son diseñados para reducir el costo y el consumo de energía de un sistema en particular. Por eso el tamaño de la unidad central de procesamiento, la cantidad de memoria y los periféricos incluidos dependerán de la aplicación.

Tanto la industria de los computadores comerciales como la de los microcontroladores están inclinados hacia la filosofía *"reduced instruction set computer"* con sus siglas abreviadas RISC (Computadores de juego de instrucciones reducido). En estos procesadores el repertorio de instrucciones máquina es muy reducido y las instrucciones son simples y generalmente se ejecutan en un ciclo. Los PIC son una familia de microcontroladores tipo RISC, fabricados por *Microchip Tecnology Inc.*y derivados del PIC1650, originalmente desarrollado por la división de *General Instrument.*

El nombre actual no es un acrónimo. En realidad, el nombre completo es *PICmicro*, aunque generalmente se utiliza como *Peripheral Interface Controller* (controlador de interfaz periférico).

4.1.- PIC 18F4550

El PIC 18F4550 (figura 4.1) de la empresa *Microchip,* es de la familia 18FXXX y tiene una arquitectura RISC avanzada de *Hardware* que le permite trabajar con 8 bits de datos y acepta hasta 32Kbytes de programación. También cuenta con característica especiales para la comunicación por puerto USB gracias al periférico que tiene integrado.

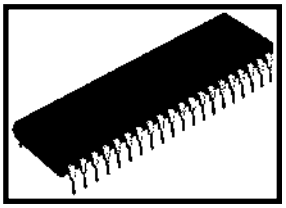


Fig. 4.1 Morfología del PIC 18F4550.

En la tabla 4.1 se muestran las características y funciones principales de este microcontrolador PIC 18F4550.

CARACTERISTICAS	PIC 18F4550
Frecuencia de operación.	Hasta 48 MHz
Memoria de programa (Bytes).	32.768
Memoria RAM (Bytes).	2.048
Memoria EEPROM datos (Bytes).	256
Interrupciones.	20
Líneas de E/S.	35
Temporizadores.	4
Módulos de comparación/Captura/PWM(CCP).	1
Canales de comunicación serie.	MSSP,EUSART
Canal USB.	1
Puerto paralelo de transmisión de datos (SPP).	1
Canales de conversión A/D de 10 Bits.	13
Comparadores analógicos.	2
Juego de instrucciones.	75 (83 ext.)
Encapsulado.	PDIP 40 pines QFN 40 pines TQFP 40 pines.

Tabla 4.1 Características PIC 18F4550.

4.1.1.- PUERTOS DEL PIC 18F4550

El PIC 18F4550 dispone de 5 puertos de entrada y salida; que incluyen un total de 35 terminales, en la tabla 4.2 se da una pequeña descripción de los puertos.

PUERTOS	LINEAS DE ENTRADA /SALIDA
PUERTO A	7 LINEAS DE ENTRADA/SALIDA
PUERTO B	8 LINEAS DE ENTRADA/SALIDA
PUERTO C	6 LINEAS DE ENTRADA/SALIDA + 2 LINEAS DE ENTRADA
PUERTO D	8 LINEAS DE ENTRADA/SALIDA
PUERTO E	3 LINEAS DE ENTRADA/SALIDA + 1 LINEAS DE ENTRADA

Tabla 4.2 Entradas y salidas de los puertos.

En la figura 4.2 se muestra la disposición de cada terminal así como de los puertos y funciones de cada una de ellas, para resolver alguna duda inherente a los tratados en esta tesis se sugiere consultar el manual del fabricante.

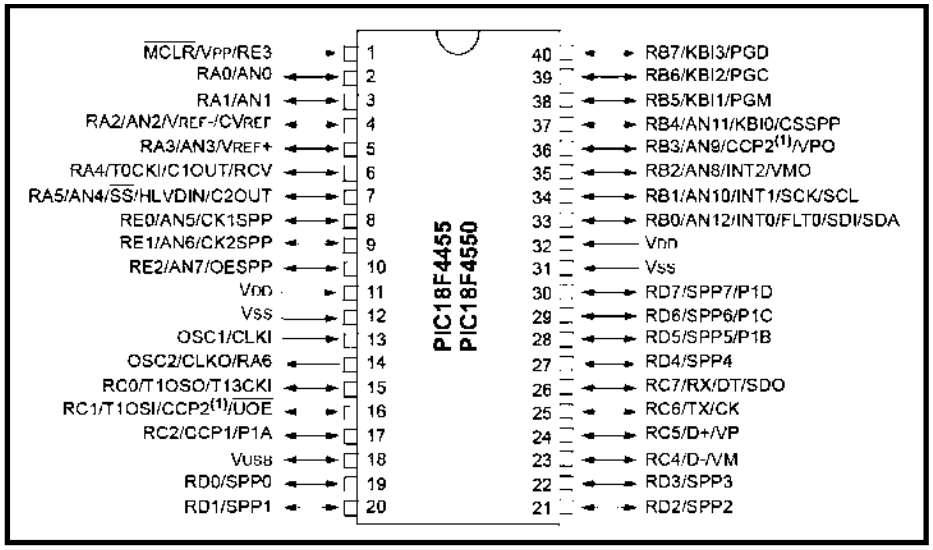


Fig. 4.2 Disposición de los terminales del PIC 16F4550.

4.2- PROGRAMACIÓN DEL PIC 18F4550

La programación del PIC se desarrolló en el *software MicroCode Studio 4.0.0.0*, este *software* ofrece la herramienta *EasyHid USB Wisard* mencionada en el capítulo anterior, la cual también genera un programa para el microcontrolador con cierto número de líneas de programación que son necesarias para la comunicación por puerto USB (Anexo 5).

Tanto las variables como la programación desarrollada para la comunicación por USB está expuesta en el anexo 6 de esta tesis.

4.3.- EMPLEO DE LA HERRAMIENTA *EASYHID USB WISARD* DEL *SOFTWARE MICROCODE STUDIO 4.0.0.0*

Para la creación de un programa para comunicación por USB con la herramienta *EasyHid USB Wisard* es necesario ejecutar el *software MicroCode Studio 4.0.0.0* de inmediato se observará la pantalla de la figura 4.3 y en la barra de tareas en el apartado *View* aparece la opción del uso de la herramienta antes mencionada.

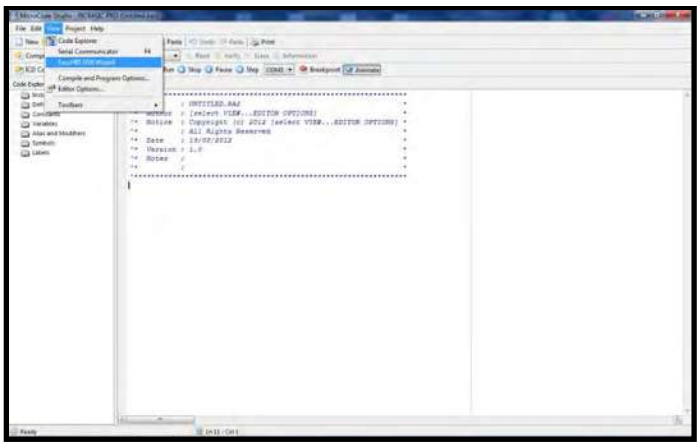


Fig. 4.3 Ventana inicial herramianta EasyHid USB Wisard.

Después de seleccionar la herramienta *EasyHid USB Wisard* seguirá la configuración del número de identificación del propietario del *software* y número de identificación del producto para el cual *MicroCode Studio 4.0.0.0* ofrece unos valores predeterminados (figura 4.4) y que en el caso de este proyecto no se han alterado.

Estos valores determinan la privacidad del proyecto desarrollado con este *software*, por omisión, los valores mostrados en la figura 4.4 son los empleados en este proyecto.

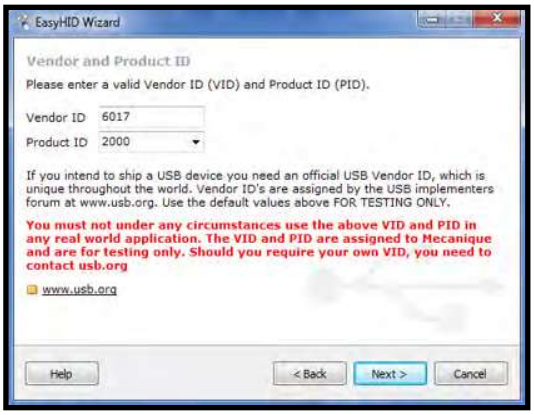


Fig. 4.4 Datos del software proveedor.

Estos valores pueden ser modificados después de adquirir una licencia especial por parte de los proveedores de *MicroCode Studio*.

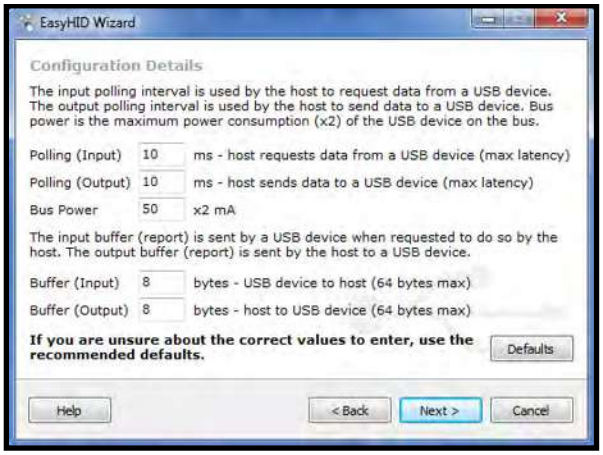


Fig. 4.5 Opciones para capacidad de intercambio de información.

La figura 4.5 muestra una primera pantalla para configuraciones relacionadas para la comunicación con el Host (Computador).

El host ejecuta una comunicación bidireccional con el dispositivo USB, intercambiando información continuamente durante un determinado intervalo de tiempo. Es posible modificar este intervalo dependiendo las necesidades del sistema, por omisión se manejan tiempos de 10ms. También es posible modificar la corriente que el bus USB entrega al dispositivo; es importante tener en cuenta que el bus de la computadora entrega una corriente máxima de 500mA, en caso de que el dispositivo requiera una corriente mayor se debe implementar una fuente externa que nos proporcione una mayor corriente, en el caso del microcontrolador 18F4550, éste funciona proporcionando los 5 Volts y la corriente delimitada a 500mA provenientes del computador.

La información se transmite a través de *Buffers* tanto de entrada como de salida, el número máximo de *Buffers* que se pueden implementar es de 64 y a su vez cada *Buffer* transmite 256 bits.

Después de estas configuraciones y dar click en siguiente puede cambiarse el nombre del proyecto y la ubicación donde será guardado. Una observación que llega a contraer bastantes problemas, es que el nombre del proyecto no debe contener más de 25 caracteres, al igual que la ubicación donde será guardado el proyecto tendrá que ser lo más cercano al directorio raíz del computador debido a que se generan problemas para la compilación del proyecto por el tamaño de la extensión del archivo.

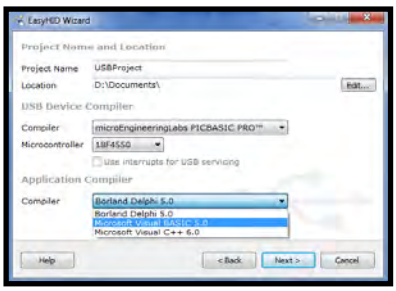


Fig. 4.6 Datos del proyecto y configuración de componentes.

También en la figura 4.6 se observa la posibilidad de seleccionar el microcontrolador empleado en el proyecto así como el compilador para el cual se desea que *Easy HID Wisard* genere las líneas de código, que para nuestro caso es *PicBasic Pro* y por último el *software* del computador que estará en comunicación con el microcontrolador (Visual Basic 5.0).

Al terminar la generación de los archivos necesarios para la comunicación por el puerto USB, la herramienta *EasyHid USB Wisard* envía una pantalla donde se avisa que el proceso fue realizado con éxito (figura 4.7). Al dar por terminados los pasos necesarios descritos anteriormente, será generada una carpeta con los archivos necesarios para la comunicación por medio del puerto USB de la computadora.

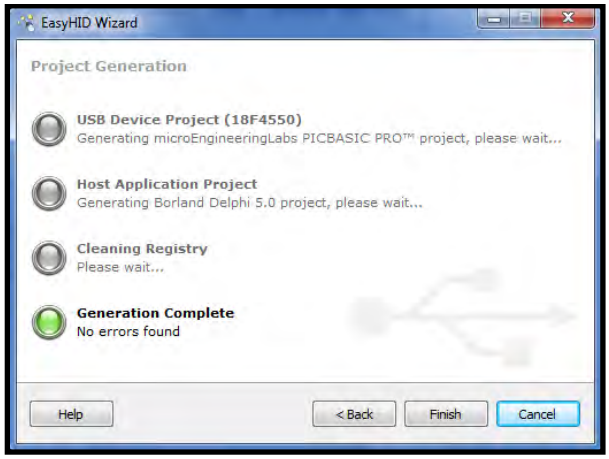


Fig. 4.7 Pantalla final de EasyHID Wizard.

CAPÍTULO V
ELECTRÓNICA EMPLEADA EN EL PROYECTO.

Para la interacción entre motores a pasos y el simulador es necesaria la creación de una interfaz que permita la comunicación electrónica de las órdenes provenientes del simulador, esta comunicación realmente es la secuencia de activación de las fases de los motores a pasos que deberá ser amplificada con la potencia necesaria por la misma interfaz. Por otro lado, la interfaz también envía señales recibidas.

5.1.- MOTORES A PASOS

Los motores a pasos son dispositivos electromecánicos que transforman energía eléctrica en mecánica en forma rotacional. Los motores a pasos cuentan con un rotor donde se monta un arreglo de imanes permanentes y por un cierto número de bobinas exitadoras bobinadas en su estator también conocidas como fases que cuando son energizadas con cierta secuencia "acomodan" la flecha del motor a pasos en diferentes posiciones hasta hacerlo girar 360º, si la secuencia sigue corriendo el motor seguirá rotando, cuando la secuencia de activación de las fases es detenida el motor a pasos se enclava y opondrá cierta resistencia al movimiento giratorio debido a que unas de sus fases están energizadas y tenderá a mantener el eje del motor en una sola posición; cuando esta fase es desactivada la flecha del motor pierde fuerza para mantener una posición fija.

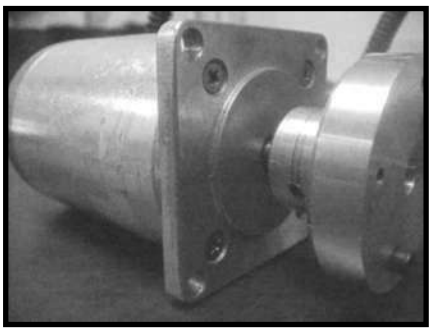


Fig. 5.1 Motor a pasos Unipolar.

Existen dos tipos de motores a pasos de imán permanente que son utilizados en robótica:

1.- Unipolares 2.-Bipolares

Para el trabajo descrito en esta tesis se emplearon motores a pasos unipolares de 5 cables, estos motores tienen un total de 200 pasos para sumar un movimiento circular de 360º y son alimentados con 5Volt, la razón del empleo de éstos, es porque en el caso de los motores a pasos bipolares es necesario la implementación de un puente "H" lo que aumentaría el número de componentes electrónicos en la interfaz de control.

5.1.1.- MOTORES A PASOS UNIPOLARES

Los motores a pasos unipolares suelen tener 5 ó 6 cables dependiendo de su conexionado interno, de los cuales 4 usualmente son los que reciben los pulsos de secuencia para determinar el sentido de rotación y la duración en cada paso y los restantes sirven para alimentación del mismo motor. Un ejemplo que permite la comprensión del funcionamiento de estos motores es la expuesta en la figura 5.2, donde también se hace notar la parte en la que interviene la interfaz electrónica de la cual se menciona en el apartado 5.2.1.

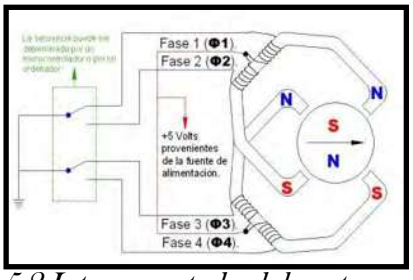


Fig. 5.2 Interconectado del motor a pasos.

5.1.2.- SECUENCIA DE ACTIVACIÓN DE FASES

Existen tres secuencias diferentes para la activación de los motores unipolares que depende de la forma de activación de las fases, al emplear

alguna de éstas, se alteran variables del motor como lo pueden ser el torque, velocidad de giro entre otras. Todas las secuencias de activación de fases comienzan nuevamente por el paso 1 después de llegar al paso final, que en las dos primeras es en el paso 4 y la última el paso 8. Para provocar un movimiento inverso, la secuencia tiene que ejecutarse de manera inversa, del último paso hasta el paso 1.

SECUENCIA NORMAL

Con esta secuencia el motor avanza un paso por cada cambio en la secuencia, y debido a que siempre hay al menos dos fases energizadas el campo magnético es mayor y se obtiene un mayor torque de pasos y de retención que en las secuencias posteriores. Esta secuencia es la empleada para el desarrollo de esta tesis. La secuencia está ilustrada en la figura 5.3 así como los valores del voltaje en la tabla 5.1.

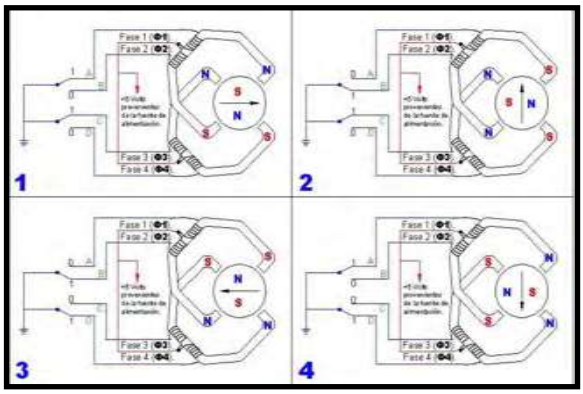


Fig. 5.3 Secuencia normal de rotación.

PASO	Fase 1 (Φ1)	Fase 2 (Φ2)	Fase 3 (Φ3)	Fase 4 (Φ4)
1	1	0	1	0
2	0	1	1	0
3	0	1	0	1
4	1	0	0	1

Tabla 5.1 Secuencia de activación de fases "Secuencia Normal".

SECUENCIA DE PASO SIMPLE

Esta secuencia de pasos es la más simple de todas; se activa sólo una bobina a la vez. En algunos motores esto brinda un funcionamiento más suave, pero por otro lado al estar sólo una bobina activada, el torque de cada paso y el enclavamiento es menor. La velocidad de rotación también es afectada con esta secuencia de pasos debido a que al existir un campo magnético más débil el rotor es arrastrado con menor fuerza y el motor podría perder pasos si la velocidad del cambio de secuencia es muy rápido. La secuencia de activación de fases se muestra en la figura 5.4 y tabla 5.2.

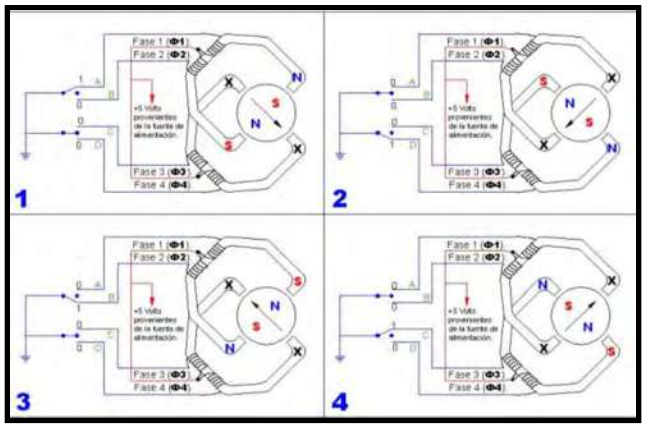


Fig. 5.4 Secuencia "Paso Simple" de rotación.

PASO	Fase 1 (Φ1)	Fase 2 (Φ2)	Fase 3 (Φ3)	Fase 4 (Φ4)
1	1	0	0	0
2	0	0	0	1
3	0	1	0	0
4	0	0	1	0

Tabla 5.2 Secuencia de activación "Paso simple".

SECUENCIA DE MEDIO PASO

En esta secuencia se activan las bobinas de tal forma que se realice un movimiento igual a la mitad del paso real. Para ello se activan primero dos

bobinas y luego sólo una y así sucesivamente. En la figura 5.5 y 5.6, y en la tabla 5.3 se muestra la secuencia de pasos, a diferencia de las anteriores secuencias, ésta está comprendida de ocho movimientos en lugar de cuatro.

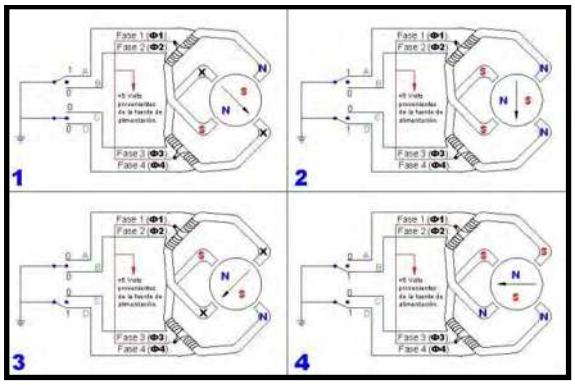


Fig. 5.5 Secuencia "Medio Paso" de rotación.

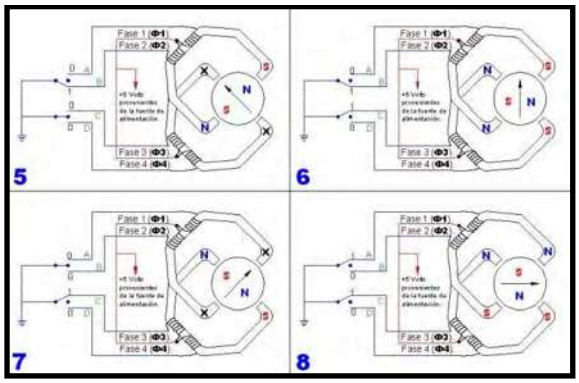


Fig. 5.6 Secuencia "Medio Paso" de rotación continuación.

PASO	Fase 1 (Φ1)	Fase 2 (Φ2)	Fase 3 (Φ3)	Fase 4 (Φ4)
1	1	0	0	0
2	1	0	0	1
3	0	0	0	1
4	0	1	0	1
5	0	1	0	0
6	0	1	1	0
7	0	0	1	0
8	1	0	1	0

Tabla 5.3 Secuencia de activación "Medio paso".

5.1.3.- MOTORES A PASOS BIPOLARES

Este tipo de motores a pasos tienen generalmente cuatro cables de salida. Necesitan ciertas manipulaciones para ser controlados, debido a que requieren del cambio de dirección del flujo de corriente a través de las bobinas en la secuencia apropiada para realizar un movimiento. Es necesario además un puente "H" por cada bobina del motor, es decir que para controlar un motor

a pasos de cuatro cables (dos bobinas), se necesitan dos puentes H para su activación.

Debido a que en lo relativo a esta tesis se emplean motores a pasos unipolares, esto es todo lo que se mencionará acerca de los motores a pasos bipolares.

5.2.- CONEXIONADO DE LA PARTE ELECTRÓNICA

Una vez que ya se ha descrito la programación del simulador, la del microcontrolador PIC, y el modo de operación del proyecto, se realiza la descripción de la interfaz electrónica con los aditamentos necesarios para su óptimo rendimiento.

En la figura 5.7 se muestra todo el conexionado de la interfaz electrónica del proyecto. Cabe resaltar que el modo de comunicación es vía USB por medio del microcontrolador PIC (Referente a punto 5.2).

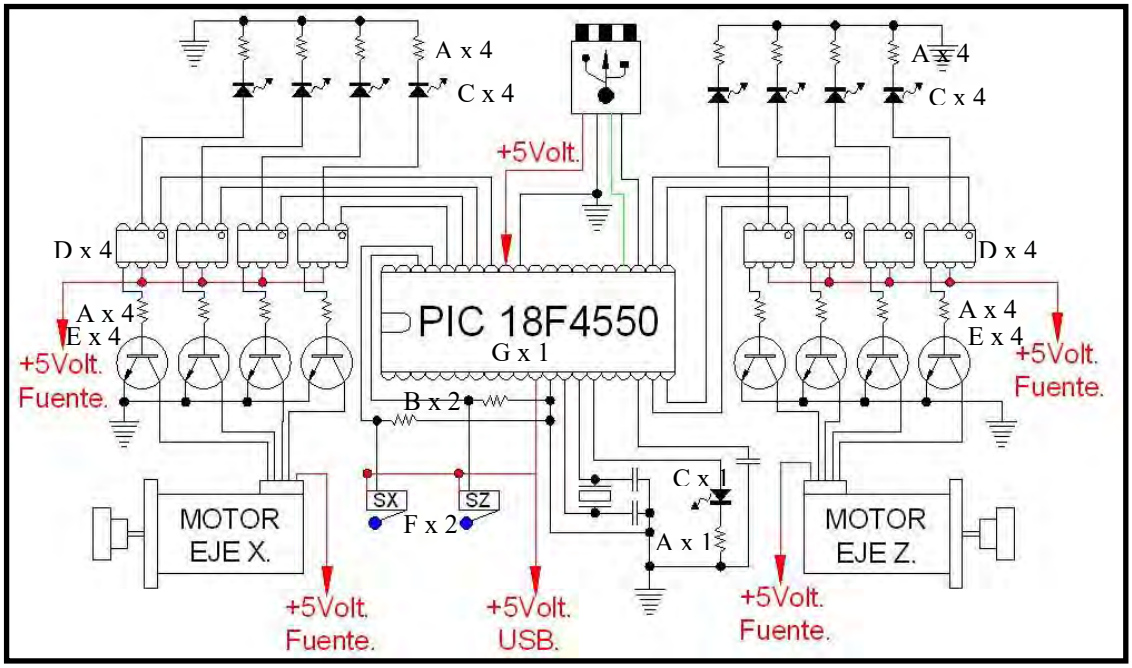


Fig. 5.7 Conexionado interno de la interfaz electrónica.

En la tabla 5.4 se observan el número y tipo de componentes que conforman la interfaz electrónica para la interacción entre el simulador (Sistema Maestro), y los motores a pasos (Sistema Esclavo). La circuitería necesaria para el funcionamiento del microcontrolador está compuesta por un oscilador de 20Mhz y dos capacitores cerámicos de 22µF en la manera que muestra la figura 5.7.

DESIGNACIÓN	DESCRIPCIÓN	CANTIDAD TOTAL
A	RESISTENCIA 470Ω 1/4W	17
B	RESISTENCIA 1K Ω 1/4W	2
C	LED'S	9
D	OPTOACOPLADOR 4N30	9
E	TRANSISTOR TIP120	8
F	SWITCH ON-OFF (NORMALMENTE CERRADO)	2
G	MICROCONTROLADOR PIC 18F4550	1

Tabla 5.4 Componentes electrónicos de la interfaz.

5.2.1.- RESEÑA DEL FUNCIONAMIENTO DE LA INTERFAZ ELECTRÓNICA

Inicialmente la información es transmitida y recibida por medio del USB que está conectado al PIC 18F4550 por medio de los pines 23 y 24. Dos de los cables del conector USB proporcionan alimentación al microcontrolador, esto es, la computadora a la que se conecta la interfaz, alimenta con voltaje y amperaje suficiente para la operación del microcontrolador y los LED's indicadores. Posterior a que la información ha sido recibida por el microcontrolador, es procesada y en respuesta, muestra la información que entrega a los motores a pasos por medio de los LED's que se encuentran conectados a los optoacopladores conectados a sus respectivas

resistencias limitadores de corriente, el puerto B del microcontrolador es el empleado para el eje X del microtorno, mientras que el puerto D es empleado para el eje Z. La información enviada por estos puertos es la secuencia de activación de las bobinas de los motores a pasos (Referente a punto 6.1).

La interfaz tiene montados 8 optoacopladores 4N30 para el aislamiento del voltaje proveniente del puerto USB de la computadora, así como para la protección del microcontrolador de algún corto circuito en la parte electrónica de mediana potencia. Enseguida, en la parte de "salida" del optoacoplador 4N30 (Parte del fototransistor), están montadas las resistencias limitadoras de corriente de los transistores TIP120, que es donde se genera la amplificación de la corriente enviada finalmente a las bobinas de los motores a pasos. La interfaz cuenta con dos switch´s que son activados al momento de la autocalibración (referente al punto 3.2.1 de esta tesis) del microtorno en el fin de carrera de los dos ejes.

CAPITULO VI
RESULTADOS

6.1.- RESULTADOS OBTENIDOS

Los resultados obtenidos en este trabajo de tesis son evaluados en base a comparaciones entre:

1. Comparación entre el tiempo empleado para la manufactura de algún maquinado sin la modificación y el tiempo consumido después de la modificación del microtorno mod. #OTMT-CO.
2. Comparación de la precisión del dimensionamiento entre una pieza maquinada sin las modificaciones realizadas en el microtorno #OTMT-CO y otra maquinada con ayuda del simulador y las modificaciones realizadas.
3. Comparación de la precisión en el dimensionamiento de una pieza maquinada en el microtorno con las modificaciones expuestas en esta tesis y otra pieza maquinada con las mismas dimensiones pero empleando el torno CNC con el que cuenta el laboratorio de CNC de la U.M.S.N.H. el torno *Dyna Myte 3000*.
4. Comparación del terminado superficial de las piezas maquinadas del punto anterior.
5. Comparación entre costo de la modificación del microtorno mod. #OTMT-CO y el costo del torno *Dyna Myte 3000*.

6.1.1.- MEJORA DE TIEMPOS EN MAQUINADOS

Al inicio de este proyecto el microtorno mod. #OTMT-CO era un microtorno del tipo manual, las piezas maquinadas en forma manual fueron únicamente del tipo cilíndricas debido a que la manufactura de alguna pieza en

que interviniera el movimiento sincronizado de ambos ejes, fue imposible. Por lo que en la gráfica 6.1 se observa la comparativa entre el tiempo empleado para realizar 3 piezas con maquinados cilíndricos de reducción de diámetro (rectificado) en madera de pino, de Ø1" hasta Ø0.6" lo que implica la ejecución de 4 pasadas con una profundidad de corte de 0.05" (1.27mm) y una longitud de rectificado de 1" como se muestra en la figura 6.1.

Nótese que las dimensiones de la figura 6.1 están dadas en pulgadas, aunque el microtorno también esta posibilitado para maquinar en milímetros.

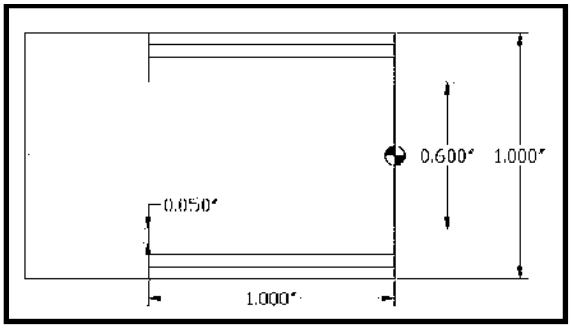
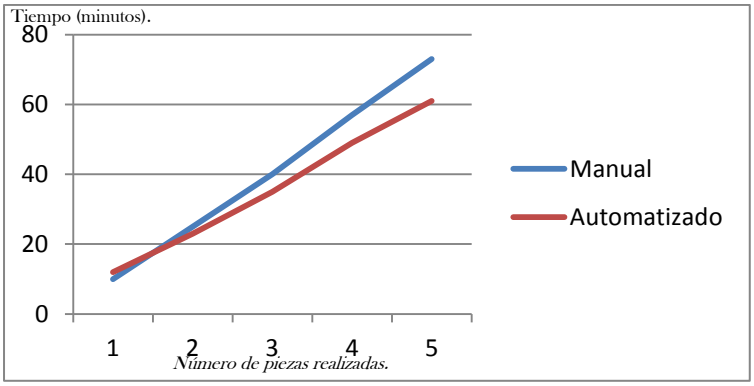


Fig. 6.1 Dimensionamiento de maquinado de prueba (rectificado).

Es importante asimilar que si la profundidad de corte es de 0.05" la pieza se reduce el doble de esta medida, es decir, después de la primer pasada la pieza quedará con un diámetro de Ø0.9" y así sucesivamente hasta alcanzar el diámetro final.



Gráfica 6.1 Comparativa de tiempos para maquinado cilíndrico (Rectificado).

Es importante señalar que en los datos de la gráfica 6.1 no fue agregado ningún factor que afectara el rendimiento del microtorno relacionado con la fatiga del operador después de permanecer maquinando durante más de 30 minutos.

El ahorro de tiempo al realizar el maquinado de la figura 6.1 es de 5 minutos después de 3 piezas con el empleo de los dispositivos de control en el microtorno. Al término de 5 piezas el ahorro de tiempo es de 10 minutos lo que habla de un resultado positivo en cuanto a la productividad del microtorno después de su automatización.

6.1.2.- COMPARACIÓN DE LA PRECISIÓN DEL MICROTORNO #OTMT-CO ORIGINAL Y DESPUÉS DE LA AUTOMATIZACIÓN

Otra de las finalidades de la automatización de una máquina herramienta es el aumento en la precisión en los productos finales que ésta presenta. Al practicar las modificaciones en el microtorno #OTMT-CO se observó un aumento de precisión al maquinar la pieza de la figura 6.1 de una manera manual, esto es, antes de la modificación y después de la misma, los datos relativos a esta comparación están expuestos en la tabla 6.1. Este ensayo se realizó en dos ocasiones y los datos que se muestran son los diámetros finales de cada una de las piezas respectivamente.

PIEZA.	MAQUINADO MANUAL.	MAQUINADO AUTOMÁTICO.
A	0.55”	0.601”
B	0.59”	0.601”

Tabla 6.1 Valores de diámetros finales.

Como se puede observar, el diámetro obtenido en los maquinados realizados en forma manual presentan valores más alejados del deseado y una variación entre el maquinado A y el B de 4 centésimas de pulgada lo que

señala un defecto en la homogenización de piezas que se produzcan en serie. Por otro lado, al analizar las dimensiones de las piezas realizadas en el microtorno automatizado, presentan un error de precisión de 1 milésima de pulgada y una buena homogenización de las dimensiones de los diámetros entre la pieza A y la B.

Una consideración muy importante es que el microtorno con sus mecanismos de control originales, permite la posibilidad de algún error a la hora de maquinar alguna pieza de manera que el material maquinado quede inservible para obtener las dimensiones deseadas, esto es, si el operador por error da una profundidad de corte mayor a la indicada por algún plano de manufactura, la pieza quedará inhabilitada al no presentar un diámetro mayor al requerido. Al realizarle las modificaciones al microtorno, la programación del simulador nunca enviará (de forma natural) información que produzca un desplazamiento del buril erróneo.

6.1.3.- COMPARACIÓN DE LA PRECISIÓN DEL MICROTORNO #OTMT-CO CON SUS MODIFICACIONES Y EL TORNO CNC *DYNA MYTE 3000*

Con el afán de medir la precisión obtenida del microtorno #OTMT-CO después de las modificaciones mencionadas (figura 6.3), las piezas maquinadas con este torno fueron comparadas con las maquinadas por un torno de control numérico (*Dyna Myte 3000* figura 6.2). Las acotaciones de las piezas maquinadas por ambos tornos están exhibidas en las figuras 6.4 y 6.5.

La comparación consistirá en obtener las dimensiones físicas en cada una de las piezas con ayuda de un vernier de carátula de reloj para obtener el error en función de las medidas requeridas por el plano de manufactura.

Se debe mencionar que el material empleado para realizar estas piezas fue *Nylamid* redondo de 1" en el caso del torno #OTMT-CO y aluminio con los mismos valores de diámetro en el caso del torno *Dyna Myte 3000*.

Fig. 6.2 Torno Dyna Myte 3000.

Fig. 6.3 Torno #OTMT-CO Automatizado.

Dentro de las consideraciones importantes en relación a estos dos tornos es que el torno CNC *Dyna Myte 3000* es un torno que está equipado con tornillos sin fin embalados lo que dota a esta herramienta de un juego mínimo entre las crestas del tornillo y la tuerca ensamblada al carro del buril en cada uno de sus dos ejes. Caso contrario al torno #OTMT-CO que cuenta con un tornillo de paso 0.056" convencional, por lo que se deduce que existe un desplazamiento angular del mismo sin que se genere algún movimiento del carro portaherramientas cada que éste experimenta algún cambio de dirección en sus movimientos y esta situación genera "pasos muertos" del motor a pasos que fue el tema tratado en el punto 3.2.1 de esta tesis.

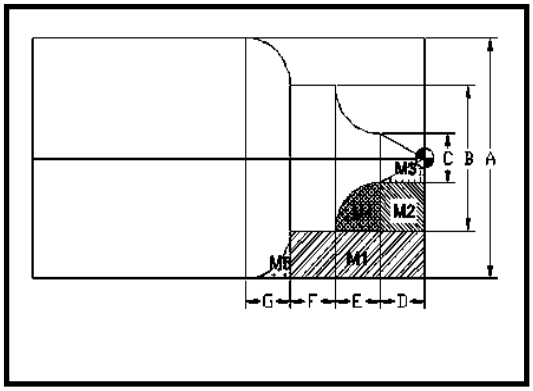


Fig. 6.4 Acotación de pieza en pulgadas.

La figura 6.4 muestra una pieza acotada en pulgadas, para lo cual ambos tornos fueron configurados para trabajar en estas unidades de medición. En la tabla comparativa 6.2 se muestran las dimensiones del plano, las obtenidas de las piezas maquinadas en el torno #OTMT-CO y las obtenidas de la pieza maquinada en el torno *Dyna Myte 3000*.

DIMENSIÓN.	MEDIDA EN EL PLANO (inch).	MEDIDA FÍSICA MICROTORNO #OTMT-CO AUTOMATIZADO (inch).	MEDIDA FÍSICA TORNO DYNA MYTE 3000 (inch).
A	0.980	0.978	0.988
B	0.600	0.597	0.605
C	0.200	0.197	0.205
D	0.200	0.196	0.203
E	0.200	0.197	0.203
F	0.200	0.197	0.203
G	0.200	0.196	0.203

Tabla 6.2 Datos de maquinados en pulgadas.

Como se puede observar en la tabla 6.2, el promedio en la precisión de los maquinados entre la medida requerida por el plano de manufactura y el microtorno #OTMT-CO automatizado es de 0.003” por debajo de la medida requerida, en tanto con el torno *Dyna Myte 3000* es de 0.005” por encima de la medida requerida en el plano de manufactura. Considerando que en la manufactura por torneado es preferible que las dimensiones finales de los maquinados tengan un superior al especificado por el plano, el torno *Dyna Myte 3000,* cumple satisfactoriamente esta característica. Una desventaja tangible es que en el caso del maquinado realizado en el microtorno #OTMT-CO automatizado es que le tomó 20 minutos realizar la pieza mencionada con anterioridad y el tiempo empleado en el torno *Dyna Myte 3000* fue de 10

minutos. Por otro lado, es preciso reportar que el tiempo requerido en la programación de la pieza en cuestión en el torno *Dyna Myte 3000,* es de aproximadamente de 15 minutos, mientras que en el simulador presentado en esta tesis, se invirtió un tiempo de 10 minutos.

La segunda pieza maquinada con el fin de estas comparaciones es la mostrada en la figura 6.5, con éstas, se pretende conocer el valor de la precisión de los maquinados en unidades milimétricas así como el desempeño de la programación del simulador para controlar al microtorno #OTMT-CO en esta modalidad.

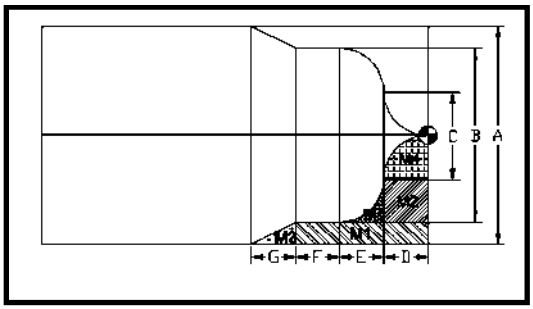


Fig. 6.5 Acotación de pieza en milímetros.

La modalidad entre maquinados en unidades métricas e inglesas se elige con el selector de unidades del simulador. En este caso el simulador deberá realizar la tarea de traducir la dimensión requería en el número de pasos necesarios para alcanzar dicha magnitud, de esta forma, podría decirse que el valor de la precisión en la modalidad de maquinados milimétricos deberá ser semejante a la equivalencia en milímetros de la precisión en los maquinados realizados en unidades inglesas.

Los datos obtenidos se muestran en la tabla 6.3 donde se observan las dimensiones que exige el plano de manufactura, las dimensiones obtenidas

luego de maquinar la pieza en el microtorno #OTMT-CO y las obtenidas en la pieza realizada en al torno *Dyna Myte 3000*.

DIMENSIÓN.	MEDIDA EN EL PLANO (mm).	MEDIDA FÍSICA MICRO TORNO #OTMT-CO AUTOMATIZADO (mm).	MEDIDA FÍSICA TORNO DYNA MYTE 3000 (mm).
A	25.146	24.917	25.150
B	20.066	19.761	20.100
C	9.906	9.000	9.966
D	5.080	4.975	5.140
E	5.080	4.975	5.140
F	5.080	4.897	5.140
G	5.080	4.890	5.140

Tabla 6.3 Datos de maquinados en milímetros.

Los datos obtenidos y mostrados en la tabla 6.3 muestran un valor promedio de 0.319mm (0.0125inch) por debajo de la medida requerida por el plano de manufactura en el caso de la pieza maquinada por el microtorno #OTMT-CO automatizado y un valor de 0.032mm (0.0012inch) en el caso del torno *Dyna Myte 3000*. Por lo anterior, se reporta una precisión superior del torno CNC *Dyna Myte 3000* por la presentada por el microtorno #OTMT-CO con sus modificaciones. El tiempo empleado en el maquinado realizado en el microtorno #OTMT-CO fue de 22 minutos mientras que en el torno *Dyna Myte 3000* fue de 10 minutos, lo que demuestra una mayor eficiencia del torno *Dyna Myte 3000* sobre el microtorno #OTMT-CO en cuestión del tiempo de maquinado.

6.1.4.- COMPARACIÓN DEL TERMINADO SUPERFICIAL ENTRE LAS PIEZAS MAQUINADAS EN EL MICROTORNO #OTMT-CO Y EL TORNO DYNA MYTE 3000

Otro factor comparativo es el terminado superficial de las piezas maquinadas por ambas herramientas, para lo cual, como se muestra en la tabla 6.4 los dos tornos muestran un terminado aceptable en los maquinados programados, con la diferencia de que las piezas maquinadas con el microtorno #OTMT-CO automatizado exhiben una pequeña discontinuidad en el término de un maquinado y el inicio del siguiente, esta discontinuidad puede ser justificada debido a la carencia de tornillos embalados en los dos ejes de movimiento del carro portaherramientas, por esta situación, dentro de las mejoras posibles del microtorno #OTMT-CO se considera la sustitución de los tornillos encargados del movimiento del carro portaherramientas por tornillos embalados.

	Torno #OTMT-CO Automatizado	Torno CNC Dyna Myte 300
Maquinado 1		
Maquinado 2		

Tabla 6.4 Comparación de terminados.

6.1.5.- COMPARACIÓN DEL COSTO DEL MICROTORNO #OTMT-CO Y EL TORNO DYNA MYTE 3000

El valor económico de las dos máquinas herramienta en comparación resultaría muy importante a la hora de adquirir alguna para su empleo en un taller de manufactura, claro está, que esta decisión tendría lugar luego de analizar que las capacidades de cada torno se ajustan a las tareas requeridas por el taller.

En este trabajo de tesis más que confrontar a los dos tornos mencionados, tiene la finalidad de exhibir una propuesta de modificación para el control de algún torno del tipo manual que finalmente tendrá características semejantes a las de un torno controlado numéricamente. Mencionado lo anterior, los datos de los costos son mostrados en la tabla 6.5 únicamente con un fin informativo.

TORNO.	COSTO DE TORNO.	COSTO DE LA MODIFICACIÓN.	COSTO TOTAL.
#OTMT-CO	$11,000.00m.n.	$10,000.00m.n.	$21,000.00m.n.
Dyna Myte 3000	$14,000.00 usd	N/R	$14,000.00 usd

Tabla 6.5 Comparación de costos entre los tornos comparados (costos aproximados).

6.2.- CONCLUSIONES

Después del diseño y construcción de la interfaz electrónica que tuviese comunicación con la computadora para el control de los motores a pasos, se logró la utomatización de las coordenadas del área de trabajo del microtorno manual #OTMT-CO, obteniendo características similares a las de un torno de control numérico. La forma de operación del microtorno y su desempeño en la precisión de los maquinados producidos en esta máquina herramienta, se asemejan con los del torno *Dyna Myte 3000*, de estas similitudes se pueden desglosar los siguientes resultados:

1. La morfología inicial del microtorno #OTMT-CO permite la adaptación de motores a pasos de tal manera que no interfiera con alguno de los movimientos del carro portaherramientas para el maquinado de piezas.
2. La disposición de las manivelas del microtorno permitieron el acoplamiento de los motores a pasos por medio de bridas a través de los soportes en acrílico asegurando un alineamiento aceptable en los ejes de los tornillos y de los motores a pasos.
3. La aplicación y la forma de planteamiento de las interpolaciones en el simulador habilitan al microtorno para desarrollar maquinados del tipo cónico y esférico, además de que se obtienen resultados favorables en la simplicidad del uso del microtorno.
4. Las comparaciones entre el microtorno #OTMT-CO modificado y el torno de control numérico *Dyna Myte 3000* muestran resultados de menor error por parte del torno CNC *Dyna Myte 3000* en cuestión de la precisión requerida por un plano de manufactura.

Por los puntos mencionados con anterioridad, se concluye que la adaptación de un sistema de automatización de coordenadas como el descrito en esta tesis, posibilita a un torno manual para maquinar piezas que requieran una precisión como la reportada en este trabajo.

También el microtorno #OTMT-CO modificado, queda restringido para maquinar piezas donde sea de vital importancia un maquinado por encima de las medidas requeridas por un plano de manufactura, luego de observar que en ambas pruebas la pieza maquinada presenta dimensiones inferiores a las programadas en el simulador.

En general, el montaje de un sistema de posicionamiento en el microtorno #OTMT-CO, es conveniente si se tiene como límite la parte económica para la compra de maquinaria para el equipamiento de un taller. El microtorno #OTMT-CO modificado, aumenta la producción en serie de piezas programadas y además queda habilitado para la realización de maquinados (cónicos y esféricos) que con su sistema original, sería imposible de conseguir y más difícilmente en una producción en serie homogénea de los mismos.

6.3.- APORTACIONES DEL TRABAJO

A lo largo del desarrollo de esta tesis los aportes en el campo de la automatización y desarrollo de sistemas de control en máquinas herramientas se enumeran como sigue:

1. Automatización del posicionado del microtorno #OTMT-CO.
2. Empleo del PIC 18F4550 en la automatización del torno OTMT-CO.
3. Propuesta para el aumento de productividad y eficiencia en un taller donde se trabaja con varios tornos #OTMT-CO, bajando costos de producción con la reducción de personal no muy calificado. Un obrero por muy calificado dificilmente podrá realizar maquinados esféricos de muy alta calidad, sin embargo, la automatización del torno #OTMT-CO brindará esas operaciones con la calidad deseada, al eliminar el factor humano que es la principal fuente de errores.
4. Propuesta de empleo de componentes electrónicos económicos para el desarrollo de una interfaz de bajo costo para el control de motores a pasos unipolares por medio del puerto USB.
5. A pesar de que el *software Visual Basic 6.0* no está concebido para dasarrollar tareas de automatización, se diseñó una aplicación basada en este *software* para la automatización del torno #OTMT-CO. Existen otros *software* como *LabView* que son dedicados a esos propósitos, pero su costo es más elevado que el *Visual Basic.* Para subsanar esta carencia de *Visual Basic* fue necesario apoyarse en el *software MicroCode Studio,* pero aún con este software adicional, el costo de la automatización se mantiene inferior comparado con la compra de un *software* diseñado específicamente para automatización.

6. Habilitación del microtorno manual #OTMT-CO para realizar maquinados cónicos y esféricos gracias a la interpolación matemática programada.
7. Propuesta de la modificación de un torno manual para obtener un control con características similares a las de un torno CNC a bajo costo.
8. Propuesta de automatización del posicionado del buril del microtorno #OTMT-CO, que puede ser extrapolada a cualquier torno manual.
9. Metodología para la eliminación de errores debidos a los pasos muertos producidos por las inperfecciones de los tornillos sin fin.

6.4.- TRABAJO A FUTURO

Existen ciertas modificaciones que pudieran realizarse en este proyecto con la finalidad de incrementar su funcionalidad y eficiencia, dentro de las cuales se sugieren las siguientes:

1. La sustitución de los materiales de los soportes de los motores a pasos para el empleo de maquinados en materiales de mayor dureza, considerando la sustitución del motor encargado del movimiento del chuck por otro con mayor potencia. Aunque para este punto, sería necesario la simulación del comportamiento de los materiales originales del microtorno #OTMT-CO con las fuerzas generadas por el maquinado de materiales mas duros.

2. Cálculo y adaptación de motores a pasos de mayor potencia para el maquinado de materiales con mayor dureza así como el cálculo de elementos electrónicos necesarios para el control de los mismos.

3. La sustitución de los tornillos para el movimiento del carro portaherramientas para alcanzar mayor precisión en los maquinados.

4. Desarrollar la programación necesaria para guardar los datos de algún maquinado específico para, en dado caso que se desee el mismo maquinado solo se abra al archivo y el simulador cargue los datos del maquinado elegido.

BIBLIOGRAFÍA

[1] "Máquinas y herramientas para la industria metalmecánica, uso y cuidado". American machinist magazine; Mc Graw Hill.

[2] Millán Gómez, Simón (2006). *Procedimientos de Mecanizado*. Editorial Paraninfo.

[3] *El mecanizado moderno – Manual práctico*. Sandvik Coromant.

[4] Cruz Teruel, Francisco. *Control numérico y programación 2.* MARCOMBO S.A, 2ª Edición.

[5] M. Simone C., Ángelo. *Automatización y control de un mini-torno paralelo.* Universidad de Carabobo.

[6] Marín Zapata, Edwin. *Diseño y construcción de un torno de control numérico.* Scientia et technica año XI, No. 29.

[7] Ramírez Cardona, Diego Alejandro y Manso Caraballo, Luis Edwin (2011). *Automatización de un torno paralelo, mediante un control numérico computarizado basado en PC.* UTP, Escuela de tecnología mecánica, Pereira.

[8] Londoño Ospina, Nelson; Molina P., William A. (2010). *Diseño de un sistema integrado para la conversión de un torno convencional a torno CNC.* Revista Politécnica ISSN 1900-2351, Año6, Número 10.

[9] *Guía Técnica de Mecanizado.* Sandvik Coromant (2006).

[10] *Catalogo de maquinaria industrial TRAVERS, OTMT TOOLS.*

[11] G. Prentice, Mikell (1997). *Fundamentos de manufactura moderna.* Hall.

[12] Porras Criado, Alejandro y Montanero Molina, Antonio Plácido (1990). *Autómatas Programables. Fundamento, Manejo, Instalación y Prácticas*. McGraw-Hill.

[13] Fuenlabrada de la Vega T., Samuel (2000). *Geometría analítica.* McGraw-Hill 2ª Edición.

[14] Piedrafita Moreno, Ramón. *Ingeniería de la automatización industrial.* Alfaomega, 2a Edición.

[15] Piedrafita Moreno, Ramón (2004). *Ingeniería de la automatización industrial.* Ra-Ma Editorial SA.

[16] Acebedo J., Mandado E. y Pérez S (1992). *Controladores lógicos y Autómatas programables.* Marcombo.

[17] H. Rashid, Muhammad (2002). *Circuitos microelectrónicos. Análisis y diseño.* Thomson.

[18] Paul Malvino, Albert. *Principios de Electrónica*. McGraw Hill.

[19] J.Millman&C., Halkias. *Electrónica integrada*. Hispano Europea.

[20] P. M. Martínez, J. Uceda, Aldana. *Electrónica Analógica.* E.T.S.I.I. de Madrid.

[21] R. Malik, Norbert (1996). *Circuitos electrónicos, análisis, simulación y diseño.* Prentice Hall.

[22] *Guía Técnica de Mecanizado.* Ed. AB-10 (2006), Sandvik Coromant.

[23] Balcells, Josep y Romeral, José Luis (1997). *Autómatas programables.* Marcombo, 1ª Edición.

[24] Grau Saldes, Antonio (2003). *Diseño y aplicaciones con autómatas programables.* Ed. Uoc.

[25] Angulo Usategui, José María y Angulo Martinez, Ignacio (1999). *Microcontroladores PIC. Diseño práctico de aplicaciones*. McGraw Hill, 2ª Edición.

[26] P. Groover, Mikell (1987). *Automation, Production Systems, and Computer Integrated Manufacturing*. Prentice-Hall.

[27] L. Wiener, Earl and E. Curry, Renwick (1980). *Flight-deck automation: Promises and problems vol. 23*.

[28] M. Swamidass, Paul and Kotha, Suresh (2000). *Strategy, advanced manufacturing technology and empirical evidence from U.S. manufacturing.* Journal of Operations anagement.

[29] Ola García, José Luis. *Prácticas de Electrónica: Circuitos Eléctricos.* Sin Editorial, 3ª Edición.

[30] Boylestad, Robert & Nashelky, Louis. *Teoría de Circuitos*, Thomson Editores, 8ª Edición.

[31] Ceballos, Fco. Javier (2000). *Enciclopedia de Microsoft Visual Basic$_{TM}$ 6.* Alfaomega.

[32] Mileaf, Harry (2004). *Electricidad.* Limusa, serie 6/7.

ANEXO 1. JERARQUÍA DE MAQUINADOS

Cuando el simulador tiene programados más de un maquinado éste utiliza la lógica del diagrama de flujo descrita en este anexo para determinar el orden en que se irán desarrollando los maquinados programados. Como ya se mencionó, el simulador tiene capacidad para la programación de 4 maquinados de cada tipo (4-cilíndrico, 4-cónico, 4-esférico).

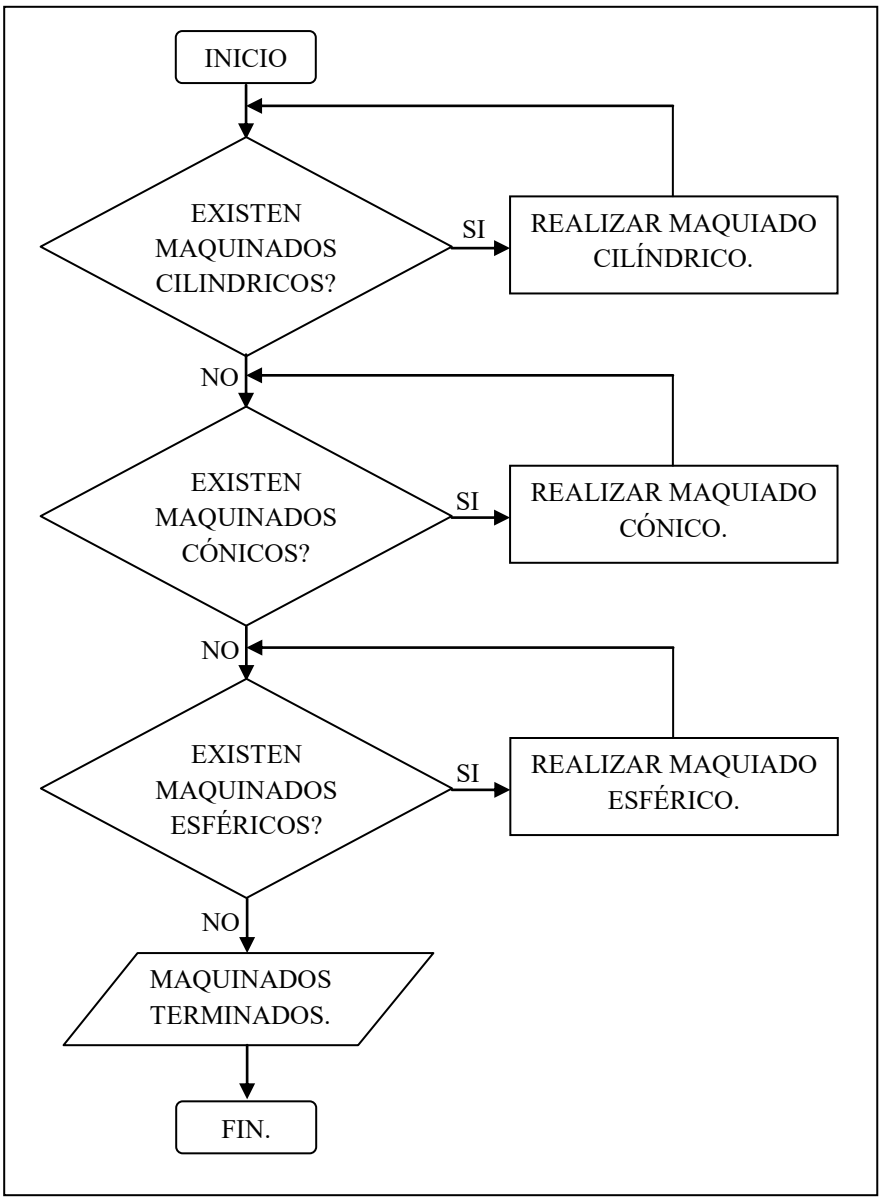


Diagrama 2.- Lógica para ejecución de maquinados.

ANEXO 2. LÍNEAS DE PROGRAMACIÓN INICIALES PARA COMUNICACIÓN POR USB PARA *VISUAL BASIC 6.0*

Las líneas mostradas en este anexo son las generadas por el *software Micro Code Studio 4.0.0.0* para trabajarlas en *Visual Basic 6.0.*

```
' vendor and product IDs
Private Const VendorID = 6017
Private Const ProductID = 2000

' read and write buffers
Private Const BufferInSize = 8
Private Const BufferOutSize = 8
Dim BufferIn(0 To BufferInSize) As Byte
Dim BufferOut(0 To BufferOutSize) As Byte

' ****************************************************************
' when the form loads, connect to the HID controller - pass
' the form window handle so that you can receive notification
' events...
'****************************************************************
Private Sub Form_Load()
   ' do not remove!
   ConnectToHID (Me.hwnd)
End Sub

'****************************************************************
' disconnect from the HID controller...
'****************************************************************
Private Sub Form_Unload(Cancel As Integer)
   DisconnectFromHID
End Sub

'****************************************************************
' a HID device has been plugged in...
'****************************************************************
Public Sub OnPlugged(ByVal pHandle As Long)
   If hidGetVendorID(pHandle) = VendorID And hidGetProductID(pHandle) = ProductID Then
      ' ** YOUR CODE HERE **
   End If
End Sub

'****************************************************************
' controller changed notification - called
' after ALL HID devices are plugged or unplugged
'****************************************************************
Public Sub OnChanged()
   Dim DeviceHandle As Long

   ' get the handle of the device we are interested in, then set
   ' its read notify flag to true - this ensures you get a read
   ' notification message when there is some data to read...
   DeviceHandle = hidGetHandle(VendorID, ProductID)
   hidSetReadNotify DeviceHandle, True
End Sub

'****************************************************************
' on read event...
'****************************************************************
Public Sub OnRead(ByVal pHandle As Long)

   ' read the data (don't forget, pass the whole array)...
   If hidRead(pHandle, BufferIn(0)) Then
      ' ** YOUR CODE HERE **
      ' first byte is the report ID, e.g. BufferIn(0)
      ' the other bytes are the data from the microcontrolller...
   End If
End Sub

'****************************************************************
' this is how you write some data...
'****************************************************************
Public Sub WriteSomeData()
   BufferOut(0) = 0   ' first by is always the report ID
   BufferOut(1) = 10  ' first data item, etc etc

   ' write the data (don't forget, pass the whole array)...
   hidWriteEx VendorID, ProductID, BufferOut(0)
End Sub
```

Figura A.- Programación inicial de PicBasic Pro.

ANEXO 3. LÓGICA DE MAQUINADOS

Este diagrama de flujo muestra la lógica para realizar cualquier maquinado en el simulador.

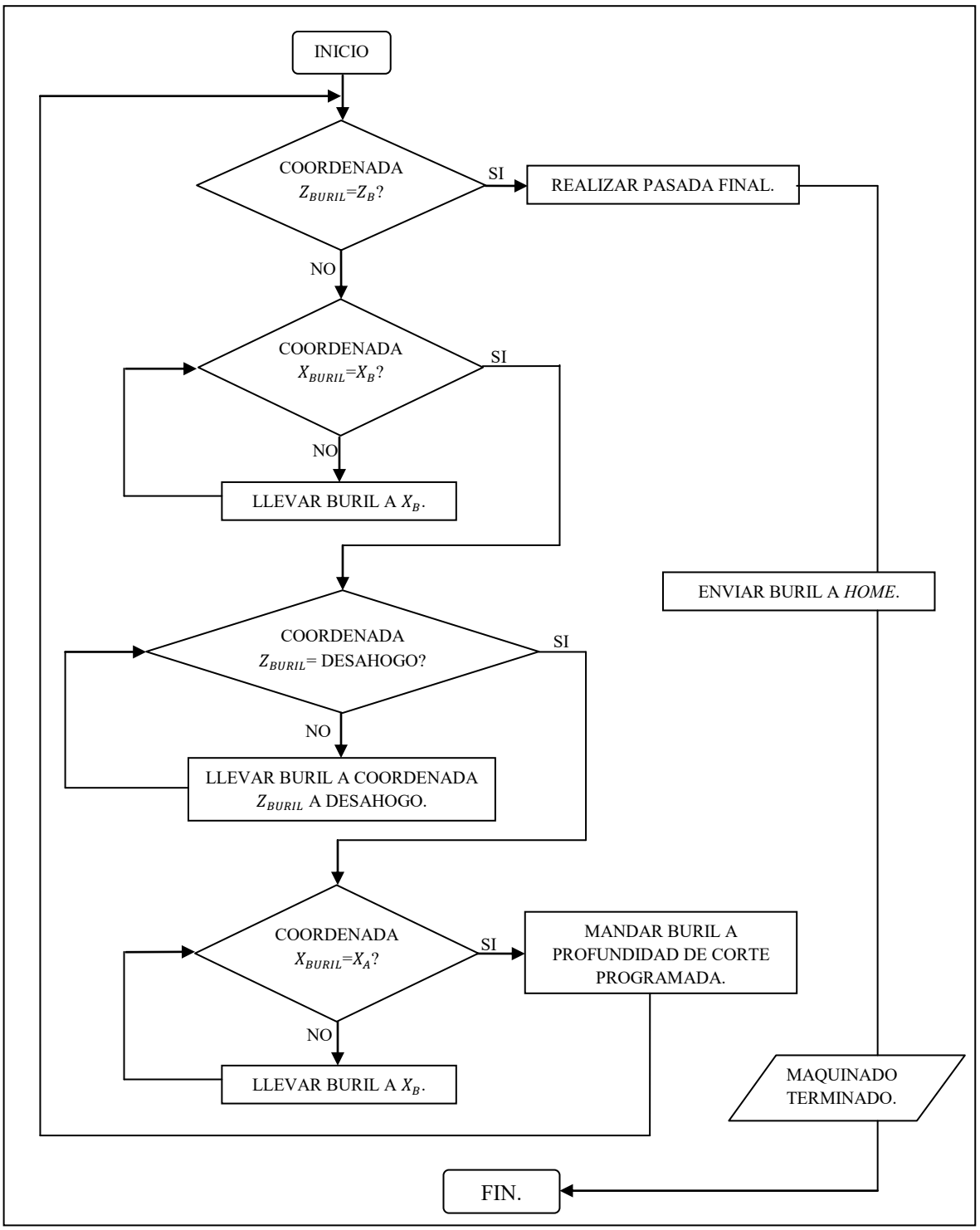


Diagrama 3.- Lógica de maquinados.

ANEXO 4. DIAGRAMA INTERPOLACIÓN DEL MAQUINADO CÓNICO

Diagrama para la obtención de "coordenadas meta" en maquinados cónicos y cilíndricos.

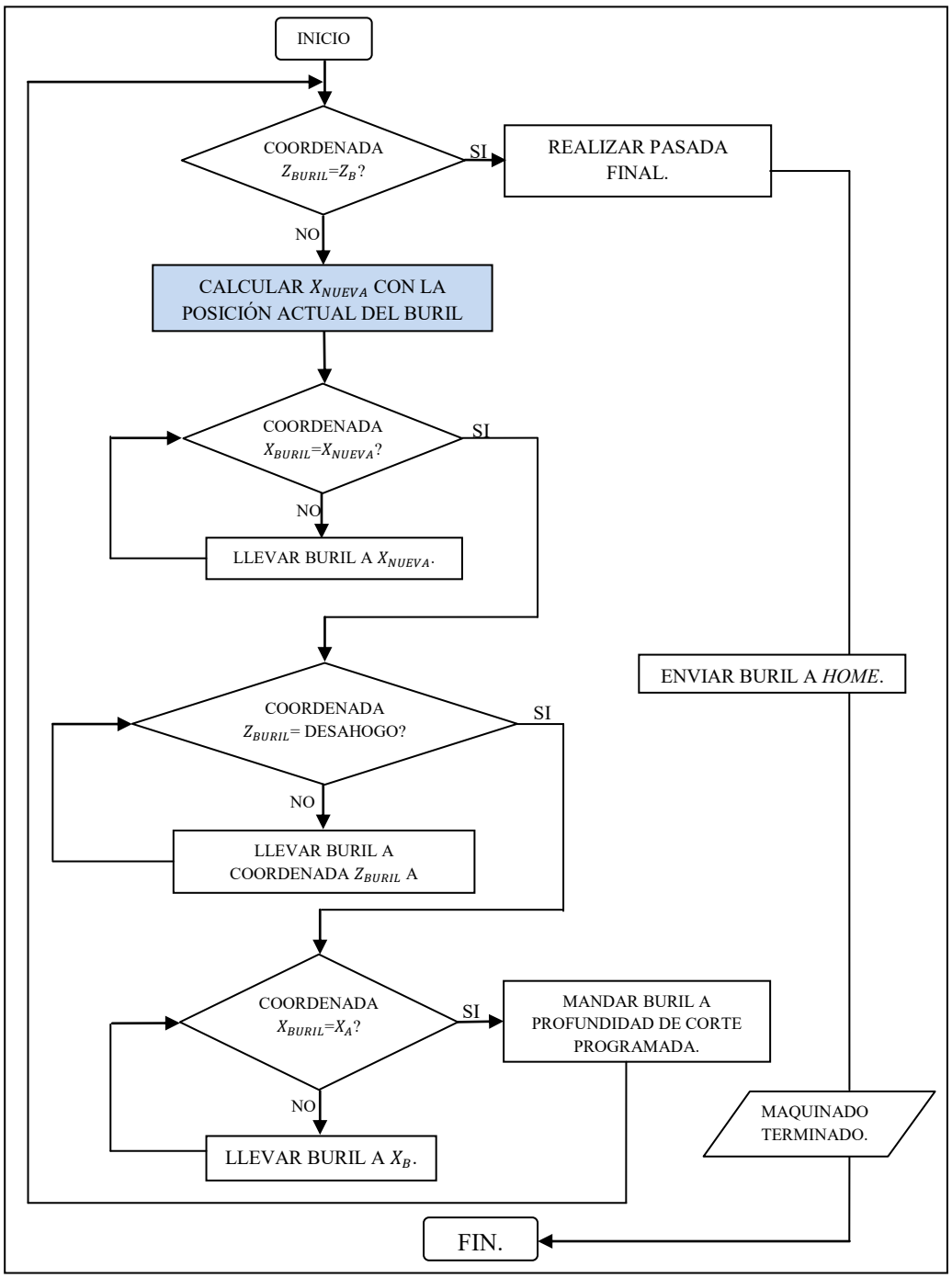


Diagrama 4.- Lógica para obtención de coordenadas de maquinados cónicos y esféricos..

ANEXO 5. LÍNEAS DE PROGRAMACIÓN INICIALES PARA COMUNICACIÓN POR USBPARA *MICRO CODE STUDIO 4.0.0.0*

Estas líneas de programación son la inicialmente arrojadas por la herramienta *EasyHID Wizard* de *PicBasic Pro* para ser cargadas en el PIC 18F4550.

```
' *************************************************************
' * Auto generated EasyHID file. PBP 2.60 and above           *
' *************************************************************

' include the HID descriptor
INCLUDE "DESCUSBProject.bas"

DEFINE OSC 48
DEFINE LOADER_USED 1

USBBufferSizeMax   CON 8   ' maximum buffer size
USBBufferSizeTX    CON 8   ' input
USBBufferSizeRX    CON 8   ' output

' the USB buffer...
USBBuffer        VAR BYTE[USBBufferSizeMax]
USBBufferCount   VAR BYTE

' *************************************************************
' * main program loop - remember, you must keep the USB       *
' * connection alive with a call to USBService every couple   *
' * of milliseconds or so...                                  *
' *************************************************************
USBINIT ' initialise USB...
ProgramStart:
   GOSUB DoUSBIn
   GOSUB DoUSBOut
   GOTO ProgramStart

' *************************************************************
' * receive data from the USB bus                             *
' *************************************************************
DoUSBIn:
   USBBufferCount = USBBufferSizeRX              ' RX buffer size
   USBSERVICE                                    ' keep connection alive
   USBIN 1, USBBuffer, USBBufferCount, DoUSBIn   ' read data, if available
   RETURN

' *************************************************************
' * wait for USB interface to attach                          *
' *************************************************************
DoUSBOut:
   USBBufferCount = USBBufferSizeTX              ' TX buffer size
   USBSERVICE                                    ' keep connection alive
   USBOUT 1, USBBuffer, USBBufferCount, DoUSBOut ' if bus available, transmit data
   RETURN
```

Figura B.- Programación inicial de EasyHID Wizard de PicBasic Pro para microcontroladores PIC.

ANEXO 6. PROGRAMACIÓN COMPLETA DEL MICROCONTROLADOR PIC 18F4550.

```
' ************************************************************
' * Auto generated EasyHID file. PBP 2.60 and above          *
' ************************************************************

' include the HID descriptor
INCLUDE "DESCUSBOTMTCO.bas"

DEFINE OSC 20
'DEFINE LOADER_USED 1

USBBufferSizeMax    CON 8   ' maximum buffer size
USBBufferSizeTX     CON 8   ' input
USBBufferSizeRX     CON 8   ' output

' the USB buffer...
USBBuffer         VAR BYTE[USBBufferSizeMax]
USBBufferCount    VAR BYTE

'VARIABLES POR DAVID
ACUMULADO VAR WORD
SOUT VAR BIT
SEIN VAR BIT
DIN1 VAR BYTE
DIN2 VAR BYTE
'DECLARAR PUERTOS
ADCON1 = %00001111
TRISB = %00110000
TRISD = 0
TRISC.0 = 0
PORTB = 0
PORTD = 0
PORTC.0 = 0

' ************************************************************
' * main program loop - remember, you must keep the USB      *
' * connection alive with a call to USBService every couple  *
' * of milliseconds or so...                                 *
' ************************************************************
USBINIT ' initialise USB...
PORTC.0 = 1
PAUSE 2000
PORTC.0 = 0
ProgramStart:
   GOSUB DoUSBIn
   DIN1 = USBBuffer[0]
   PORTB = DIN1
   USBSERVICE
   PAUSE 1
   DIN2 = USBBuffer[1]
   PORTD = DIN2
   USBSERVICE
   PAUSE 1
   'gosub DoUSBOut
IF PORTB.4=1 THEN
    USBBuffer[2]=1
    GOSUB DoUSBOut
    USBSERVICE
    PAUSE 1
ELSE
USBBuffer[2]=0
GOSUB DoUSBOut
USBSERVICE
PAUSE 1
IF PORTB.5=1 THEN
    USBBuffer[3]=1
    GOSUB DoUSBOut
    USBSERVICE
    PAUSE 1
ELSE
USBBuffer[3]=0
GOSUB DoUSBOut
USBSERVICE
PAUSE 1
ENDIF
ENDIF
   GOTO ProgramStart

' ************************************************************
' * receive data from the USB bus                            *
' ************************************************************
DoUSBIn:
   USBBufferCount = USBBufferSizeRX              ' RX buffer size
   USBSERVICE                                    ' keep connection alive
   USBIN 1, USBBuffer, USBBufferCount, DoUSBIn   ' read data, if available
   RETURN

' ************************************************************
' * wait for USB interface to attach                         *
' ************************************************************
DoUSBOut:
   USBBufferCount = USBBufferSizeTX              ' TX buffer size
   USBSERVICE                                    ' keep connection alive
   USBOUT 1, USBBuffer, USBBufferCount, DoUSBOut ' if bus available, transmit dat
   RETURN
```

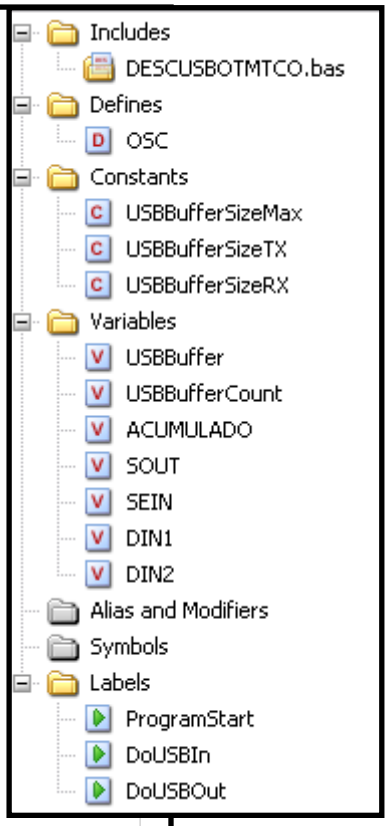
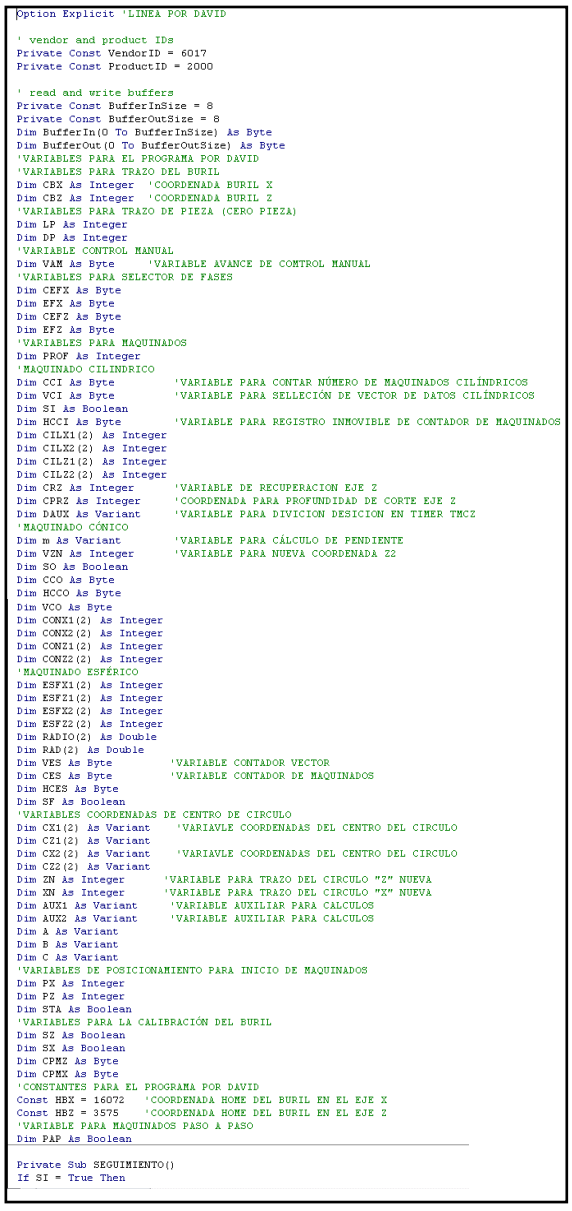


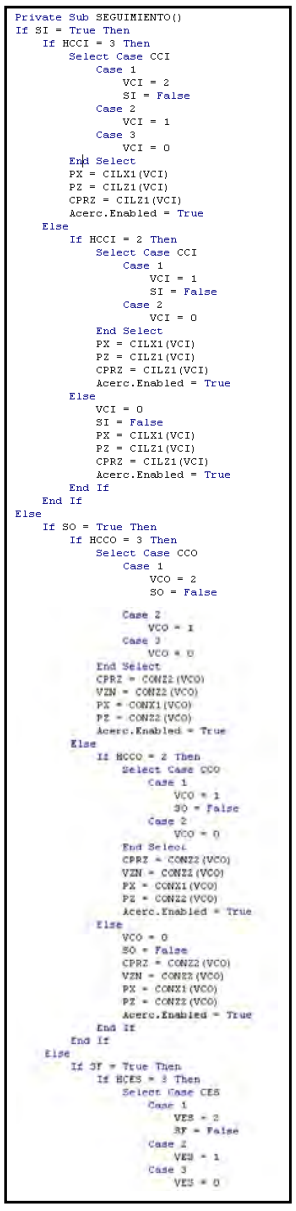
Figura C.- Programación total del PIC 18F4550 de este proyecto de tesis.

ANEXO 7. PROGRAMACIÓN COMPLETA EN EL *SOFTWARE VISUAL BASIC 6.0.*

Éstas son las líneas de programación integras para *Visual Basic 6.0,* algunas página contienen tres columnas. Están ordenadas en columnas, al terminar cada una de ellas le sigue la consecutiva a la derecha.

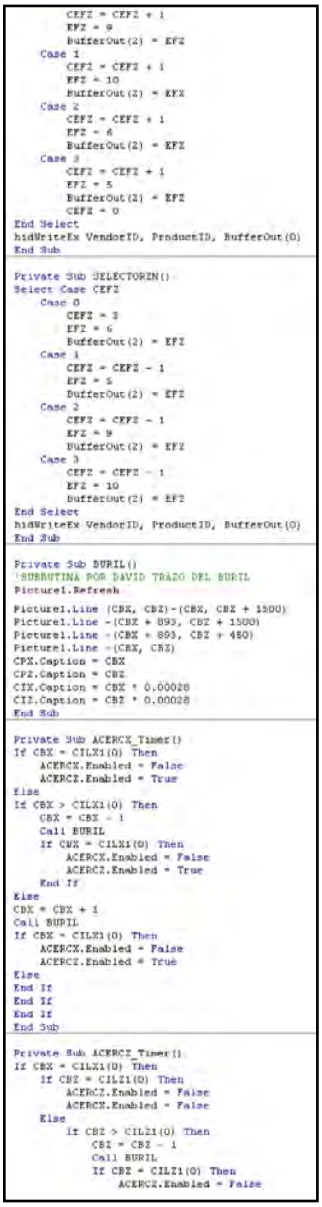


```
Option Explicit 'LINEA POR DAVID

' vendor and product IDs
Private Const VendorID = 6017
Private Const ProductID = 2000

' read and write buffers
Private Const BufferInSize = 8
Private Const BufferOutSize = 8
Dim BufferIn(0 To BufferInSize) As Byte
Dim BufferOut(0 To BufferOutSize) As Byte
'VARIABLES PARA EL PROGRAMA POR DAVID
'VARIABLES PARA TRAZO DEL BURIL
Dim CBX As Integer  'COORDENADA BURIL X
Dim CBZ As Integer  'COORDENADA BURIL Z
'VARIABLES PARA TRAZO DE PIEZA (CERO PIEZA)
Dim LP As Integer
Dim DP As Integer
'VARIABLE CONTROL MANUAL
Dim VAM As Byte     'VARIABLE AVANCE DE COMTROL MANUAL
'VARIABLES PARA SELECTOR DE FASES
Dim CEFX As Byte
Dim EFX As Byte
Dim CEFZ As Byte
Dim EFZ As Byte
'VARIABLES PARA MAQUINADOS
Dim PROF As Integer
'MAQUINADO CILINDRICO
Dim CCI As Byte         'VARIABLE PARA CONTAR NÚMERO DE MAQUINADOS CILÍNDRICOS
Dim VCI As Byte         'VARIABLE PARA SELLECIÓN DE VECTOR DE DATOS CILÍNDRICOS
Dim SI As Boolean
Dim HCCI As Byte        'VARIABLE PARA REGISTRO INMOVIBLE DE CONTADOR DE MAQUINADOS
Dim CILX1(2) As Integer
Dim CILX2(2) As Integer
Dim CILZ1(2) As Integer
Dim CILZ2(2) As Integer
Dim CRZ As Integer      'VARIABLE DE RECUPERACION EJE Z
Dim CPRZ As Integer     'COORDENADA PARA PROFUNDIDAD DE CORTE EJE Z
Dim DAUX As Variant     'VARIABLE PARA DIVICION DESICION EN TIMER TMCZ
'MAQUINADO CÓNICO
Dim m As Variant        'VARIABLE PARA CÁLCULO DE PENDIENTE
Dim VZN As Integer      'VARIABLE PARA NUEVA COORDENADA Z2
Dim SO As Boolean
Dim CCO As Byte
Dim HCCO As Byte
Dim VCO As Byte
Dim CONX1(2) As Integer
Dim CONX2(2) As Integer
Dim CONZ1(2) As Integer
Dim CONZ2(2) As Integer
'MAQUINADO ESFÉRICO
Dim ESFX1(2) As Integer
Dim ESFZ1(2) As Integer
Dim ESFX2(2) As Integer
Dim ESFZ2(2) As Integer
Dim RADIO(2) As Double
Dim RAD(2) As Double
Dim VES As Byte         'VARIABLE CONTADOR VECTOR
Dim CES As Byte         'VARIABLE CONTADOR DE MAQUINADOS
Dim HCES As Byte
Dim SF As Boolean
'VARIABLES COORDENADAS DE CENTRO DE CIRCULO
Dim CX1(2) As Variant   'VARIAVLE COORDENADAS DEL CENTRO DEL CIRCULO
Dim CZ1(2) As Variant
Dim CX2(2) As Variant   'VARIAVLE COORDENADAS DEL CENTRO DEL CIRCULO
Dim CZ2(2) As Variant
Dim ZN As Integer       'VARIABLE PARA TRAZO DEL CIRCULO "Z" NUEVA
Dim XN As Integer       'VARIABLE PARA TRAZO DEL CIRCULO "X" NUEVA
Dim AUX1 As Variant     'VARIABLE AUXILIAR PARA CALCULOS
Dim AUX2 As Variant     'VARIABLE AUXILIAR PARA CALCULOS
Dim A As Variant
Dim B As Variant
Dim C As Variant
'VARIABLES DE POSICIONAMIENTO PARA INICIO DE MAQUINADOS
Dim PX As Integer
Dim PZ As Integer
Dim STA As Boolean
'VARIABLES PARA LA CALIBRACIÓN DEL BURIL
Dim SZ As Boolean
Dim SX As Boolean
Dim CPMZ As Byte
Dim CPMX As Byte
'CONSTANTES PARA EL PROGRAMA POR DAVID
Const HBX = 16072   'COORDENADA HOME DEL BURIL EN EL EJE X
Const HBZ = 3575    'COORDENADA HOME DEL BURIL EN EL EJE Z
'VARIABLE PARA MAQUINADOS PASO A PASO
Dim PAP As Boolean

Private Sub SEGUIMIENTO()
If SI = True Then
```

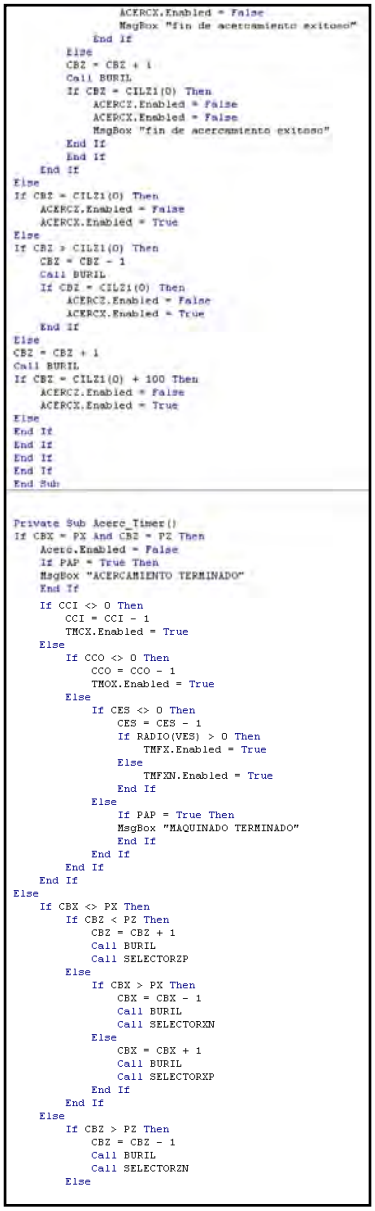


```
Private Sub SEGUIMIENTO()
If SI = True Then
    If HCCI = 3 Then
        Select Case CCI
            Case 1
                VCI = 2
                SI = False
            Case 2
                VCI = 1
            Case 3
                VCI = 0
        End Select
        PX = CILX1(VCI)
        PZ = CILZ1(VCI)
        CPRZ = CILZ1(VCI)
        Acerc.Enabled = True
    Else
        If HCCI = 2 Then
            Select Case CCI
                Case 1
                    VCI = 1
                    SI = False
                Case 2
                    VCI = 0
            End Select
            PX = CILX1(VCI)
            PZ = CILZ1(VCI)
            CPRZ = CILZ1(VCI)
            Acerc.Enabled = True
        Else
            VCI = 0
            SI = False
            PX = CILX1(VCI)
            PZ = CILZ1(VCI)
            CPRZ = CILZ1(VCI)
            Acerc.Enabled = True
        End If
    End If
Else
    If SO = True Then
        If HCCO = 3 Then
            Select Case CCO
                Case 1
                    VCO = 2
                    SO = False

                Case 2
                    VCO = 1
                Case 3
                    VCO = 0
            End Select
            CPRZ = CONZ2(VCO)
            VZN = CONZ2(VCO)
            PX = CONX1(VCO)
            PZ = CONZ2(VCO)
            Acerc.Enabled = True
        Else
            If HCCO = 2 Then
                Select Case CCO
                    Case 1
                        VCO = 1
                        SO = False
                    Case 2
                        VCO = 0
                End Select
                CPRZ = CONZ2(VCO)
                VZN = CONZ2(VCO)
                PX = CONX1(VCO)
                PZ = CONZ2(VCO)
                Acerc.Enabled = True
            Else
                VCO = 0
                SO = False
                CPRZ = CONZ2(VCO)
                VZN = CONZ2(VCO)
                PX = CONX1(VCO)
                PZ = CONZ2(VCO)
                Acerc.Enabled = True
            End If
        End If
    Else
        If SF = True Then
            If HCES = 3 Then
                Select Case CES
                    Case 1
                        VES = 2
                        SF = False
                    Case 2
                        VES = 1
                    Case 3
                        VES = 0
```

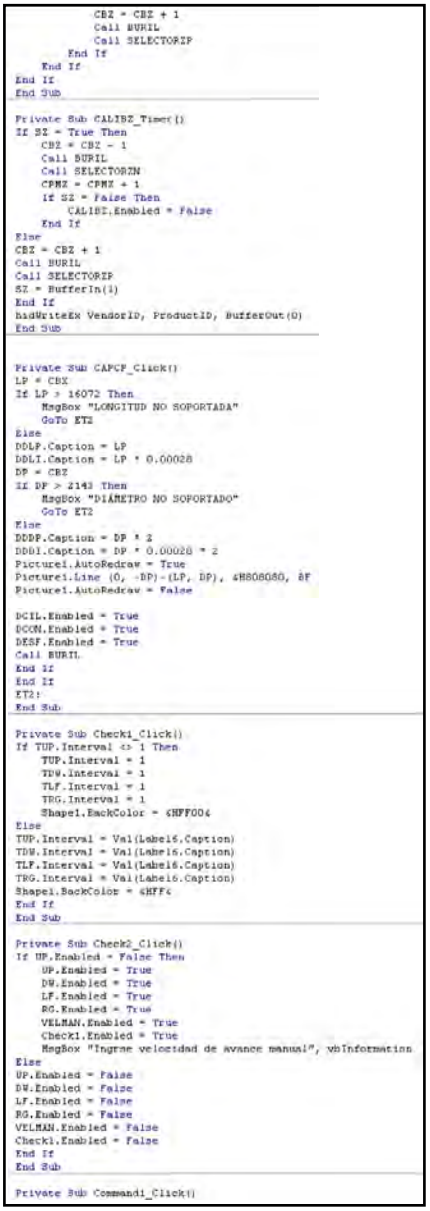
```
                End Select
                CPRZ = ESFZ2(VES)
                ZN = ESFZ2(VES)
                VZN = ESFZ2(VES)
                PX = ESFX1(VES)
                PZ = ESFZ2(VES)
                Acerc.Enabled = True
            Else
                If HCES = 2 Then
                    Select Case CES
                        Case 1
                            VES = 1
                            SF = False
                        Case 2
                            VES = 0
                    End Select
                    CPRZ = ESFZ2(VES)
                    ZN = ESFZ2(VES)
                    VZN = ESFZ2(VES)
                    PX = ESFX1(VES)
                    PZ = ESFZ2(VES)
                    Acerc.Enabled = True
                Else
                    VES = 0
                    SF = False
                    CPRZ = ESFZ2(VES)
                    ZN = ESFZ2(VES)
                    VZN = ESFZ2(VES)
                    PX = ESFX1(VES)
                    PZ = ESFZ2(VES)
                    Acerc.Enabled = True
                End If
            End If
        Else
        MsgBox "MAQUINADO TERMINADO"
        TMZ.Enabled = True
        End If
    End If
End If
End Sub


Private Sub SELECTORXN()
Select Case CEFX
    Case 0

        CEFX = 3
        EFX = 6
        BufferOut(1) = EFX
    Case 1
        CEFX = CEFX - 1
        EFX = 5
        BufferOut(1) = EFX
    Case 2
        CEFX = CEFX - 1
        EFX = 9
        BufferOut(1) = EFX
    Case 3
        CEFX = CEFX - 1
        EFX = 10
        BufferOut(1) = EFX
End Select
hidWriteEx VendorID, ProductID, BufferOut(0)
End Sub

Private Sub SELECTORXP()
Select Case CEFX
    Case 0
        CEFX = CEFX + 1
        EFX = 9
        BufferOut(1) = EFX
    Case 1
        CEFX = CEFX + 1
        EFX = 10
        BufferOut(1) = EFX
    Case 2
        CEFX = CEFX + 1
        EFX = 6
        BufferOut(1) = EFX
    Case 3
        CEFX = CEFX + 1
        EFX = 5
        BufferOut(1) = EFX
        CEFX = 0
End Select
hidWriteEx VendorID, ProductID, BufferOut(0)
End Sub

Private Sub SELECTORZP()
Select Case CEFZ
    Case 0
```



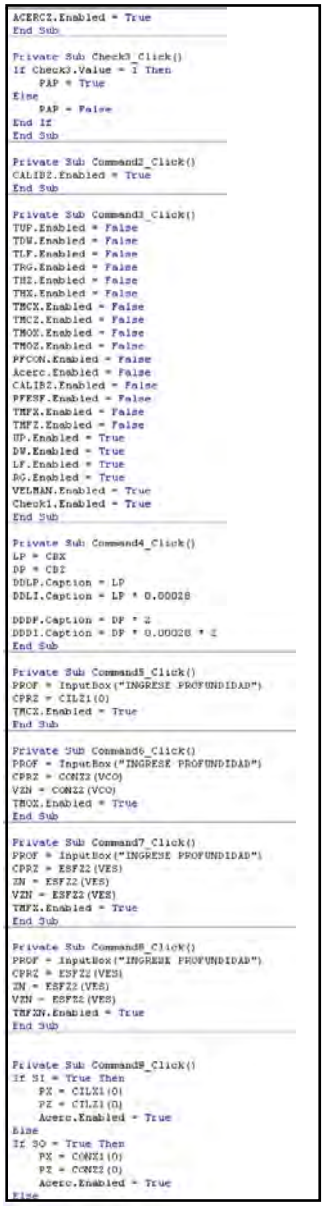
```
        CEFZ = CEFZ + 1
        EFZ = 9
        BufferOut(2) = EFZ
    Case 1
        CEFZ = CEFZ + 1
        EFZ = 10
        BufferOut(2) = EFZ
    Case 2
        CEFZ = CEFZ + 1
        EFZ = 6
        BufferOut(2) = EFZ
    Case 3
        CEFZ = CEFZ + 1
        EFZ = 5
        BufferOut(2) = EFZ
        CEFZ = 0
End Select
hidWriteEx VendorID, ProductID, BufferOut(0)
End Sub

Private Sub SELECTORZN()
Select Case CEFZ
    Case 0
        CEFZ = 3
        EFZ = 6
        BufferOut(2) = EFZ
    Case 1
        CEFZ = CEFZ - 1
        EFZ = 5
        BufferOut(2) = EFZ
    Case 2
        CEFZ = CEFZ - 1
        EFZ = 9
        BufferOut(2) = EFZ
    Case 3
        CEFZ = CEFZ - 1
        EFZ = 10
        BufferOut(2) = EFZ
End Select
hidWriteEx VendorID, ProductID, BufferOut(0)
End Sub

Private Sub BURIL()
'SUBRUTINA POR DAVID TRAZO DEL BURIL
Picture1.Refresh
Picture1.Line (CBX, CBZ)-(CBX, CBZ + 1500)
Picture1.Line -(CBX + 893, CBZ + 1500)
Picture1.Line -(CBX + 893, CBZ + 450)
Picture1.Line -(CBX, CBZ)
CPX.Caption = CBX
CPZ.Caption = CBZ
CIX.Caption = CBX * 0.00028
CIZ.Caption = CBZ * 0.00028
End Sub

Private Sub ACERCX_Timer()
If CBX = CILX1(0) Then
    ACERCX.Enabled = False
    ACERCZ.Enabled = True
Else
If CBX > CILX1(0) Then
    CBX = CBX - 1
    Call BURIL
    If CBX = CILX1(0) Then
        ACERCX.Enabled = False
        ACERCZ.Enabled = True
    End If
Else
CBX = CBX + 1
Call BURIL
If CBX = CILX1(0) Then
    ACERCX.Enabled = False
    ACERCZ.Enabled = True
Else
End If
End If
End If
End Sub

Private Sub ACERCZ_Timer()
If CBX = CILX1(0) Then
    If CBZ = CILZ1(0) Then
        ACERCZ.Enabled = False
        ACERCX.Enabled = False
    Else
        If CBZ > CILZ1(0) Then
            CBZ = CBZ - 1
            Call BURIL
            If CBZ = CILZ1(0) Then
                ACERCZ.Enabled = False
```



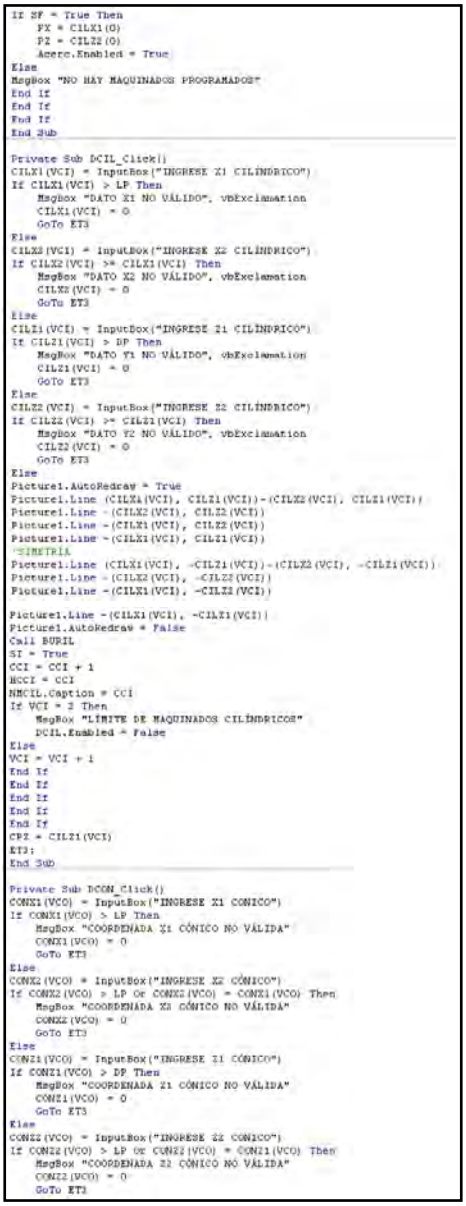
```
                ACERCX.Enabled = False
                MsgBox "fin de acercamiento exitoso"
            End If
        Else
        CBZ = CBZ + 1
        Call BURIL
        If CBZ = CILZ1(0) Then
            ACERCZ.Enabled = False
            ACERCX.Enabled = False
            MsgBox "fin de acercamiento exitoso"
        End If
        End If
    End If
Else
If CBZ = CILZ1(0) Then
    ACERCZ.Enabled = False
    ACERCX.Enabled = True
Else
If CBZ > CILZ1(0) Then
    CBZ = CBZ - 1
    Call BURIL
    If CBZ = CILZ1(0) Then
        ACERCZ.Enabled = False
        ACERCX.Enabled = True
    End If
Else
CBZ = CBZ + 1
Call BURIL
If CBZ = CILZ1(0) + 100 Then
    ACERCZ.Enabled = False
    ACERCX.Enabled = True
Else
End If
End If
End If
End If
End Sub

Private Sub Acerc_Timer()
If CBX = PX And CBZ = PZ Then
    Acerc.Enabled = False
    If PAP = True Then
    MsgBox "ACERCAMIENTO TERMINADO"
    End If
    If CCI <> 0 Then
        CCI = CCI - 1
        TMCX.Enabled = True
    Else
        If CCO <> 0 Then
            CCO = CCO - 1
            TMOX.Enabled = True
        Else
            If CES <> 0 Then
                CES = CES - 1
                If RADIO(VES) > 0 Then
                    TMFX.Enabled = True
                Else
                    TMFXN.Enabled = True
                End If
            Else
                If PAP = True Then
                MsgBox "MAQUINADO TERMINADO"
                End If
            End If
        End If
    End If
Else
    If CBX <> PX Then
        If CBZ < PZ Then
            CBZ = CBZ + 1
            Call BURIL
            Call SELECTORZP
        Else
            If CBX > PX Then
                CBX = CBX - 1
                Call BURIL
                Call SELECTORXN
            Else
                CBX = CBX + 1
                Call BURIL
                Call SELECTORXP
            End If
        End If
    Else
        If CBZ > PZ Then
            CBZ = CBZ - 1
            Call BURIL
            Call SELECTORZN
        Else
```



```
            CBZ = CBZ + 1
            Call BURIL
            Call SELECTORZP
        End If
    End If
End If
End Sub

Private Sub CALIBZ_Timer()
If SZ = True Then
    CBZ = CBZ - 1
    Call BURIL
    Call SELECTORZN
    CPMZ = CPMZ + 1
    If SZ = False Then
        CALIBZ.Enabled = False
    End If
Else
CBZ = CBZ + 1
Call BURIL
Call SELECTORZP
SZ = BufferIn(1)
End If
hidWriteEx VendorID, ProductID, BufferOut(0)
End Sub


Private Sub CAPCP_Click()
LP = CBX
If LP > 16072 Then
    MsgBox "LONGITUD NO SOPORTADA"
    GoTo ET2
Else
DDLP.Caption = LP
DDLI.Caption = LP * 0.00028
DP = CBZ
If DP > 2143 Then
    MsgBox "DIÁMETRO NO SOPORTADO"
    GoTo ET2
Else
DDDP.Caption = DP * 2
DDDI.Caption = DP * 0.00028 * 2
Picture1.AutoRedraw = True
Picture1.Line (0, -DP)-(LP, DP), &H808080, BF
Picture1.AutoRedraw = False

DCIL.Enabled = True
DCCN.Enabled = True
DESF.Enabled = True
Call BURIL
End If
End If
ET2:
End Sub

Private Sub Check1_Click()
If TUP.Interval <> 1 Then
    TUP.Interval = 1
    TDW.Interval = 1
    TLF.Interval = 1
    TRG.Interval = 1
    Shape1.BackColor = &HFF00&
Else
TUP.Interval = Val(Label16.Caption)
TDW.Interval = Val(Label16.Caption)
TLF.Interval = Val(Label16.Caption)
TRG.Interval = Val(Label16.Caption)
Shape1.BackColor = &HFF&
End If
End Sub

Private Sub Check2_Click()
If UP.Enabled = False Then
    UP.Enabled = True
    DW.Enabled = True
    LF.Enabled = True
    RG.Enabled = True
    VELMAN.Enabled = True
    Check1.Enabled = True
    MsgBox "Ingrse velocidad de avance manual", vbInformation
Else
UP.Enabled = False
DW.Enabled = False
LF.Enabled = False
RG.Enabled = False
VELMAN.Enabled = False
Check1.Enabled = False
End If
End Sub

Private Sub Command1_Click()
```



```
ACERCZ.Enabled = True
End Sub

Private Sub Check3_Click()
If Check3.Value = 1 Then
    PAP = True
Else
    PAP = False
End If
End Sub

Private Sub Command2_Click()
CALIBZ.Enabled = True
End Sub

Private Sub Command3_Click()
TUP.Enabled = False
TDW.Enabled = False
TLF.Enabled = False
TRG.Enabled = False
THZ.Enabled = False
THX.Enabled = False
TMCX.Enabled = False
TMCZ.Enabled = False
TMOX.Enabled = False
TMOZ.Enabled = False
PYCON.Enabled = False
Acerc.Enabled = False
CALIBZ.Enabled = False
PYESF.Enabled = False
TMFX.Enabled = False
TMFZ.Enabled = False
UP.Enabled = True
DW.Enabled = True
LF.Enabled = True
RG.Enabled = True
VELMAN.Enabled = True
Check1.Enabled = True
End Sub

Private Sub Command4_Click()
LP = CBX
DP = CBZ
DDLP.Caption = LP
DDLI.Caption = LP * 0.00028

DDDP.Caption = DP * 2
DDD1.Caption = DP * 0.00028 * 2
End Sub

Private Sub Command5_Click()
PROF = InputBox("INGRESE PROFUNDIDAD")
CPRZ = CILZ1(0)
TMCX.Enabled = True
End Sub

Private Sub Command6_Click()
PROF = InputBox("INGRESE PROFUNDIDAD")
CPRZ = CONZ2(VCO)
VZN = CONZ2(VCO)
TMOX.Enabled = True
End Sub

Private Sub Command7_Click()
PROF = InputBox("INGRESE PROFUNDIDAD")
CPRZ = ESFZ2(VES)
ZN = ESFZ2(VES)
VZN = ESFZ2(VES)
TMFX.Enabled = True
End Sub

Private Sub Command8_Click()
PROF = InputBox("INGRESE PROFUNDIDAD")
CPRZ = ESFZ2(VES)
ZN = ESFZ2(VES)
VZN = ESFZ2(VES)
TMFZN.Enabled = True
End Sub


Private Sub Command9_Click()
If SI = True Then
    PX = CILX1(0)
    PZ = CILZ1(0)
    Acerc.Enabled = True
Else
If SO = True Then
    PX = CONX1(0)
    PZ = CONZ2(0)
    Acerc.Enabled = True
Else
```



```
If SF = True Then
    PX = CILX1(0)
    PZ = CILZ2(0)
    Acerc.Enabled = True
Else
MsgBox "NO HAY MAQUINADOS PROGRAMADOS"
End If
End If
End If
End Sub

Private Sub DCIL_Click()
CILX1(VCI) = InputBox("INGRESE X1 CILÍNDRICO")
If CILX1(VCI) > LP Then
    MsgBox "DATO X1 NO VÁLIDO", vbExclamation
    CILX1(VCI) = 0
    GoTo ET3
Else
CILX2(VCI) = InputBox("INGRESE X2 CILÍNDRICO")
If CILX2(VCI) >= CILX1(VCI) Then
    MsgBox "DATO X2 NO VÁLIDO", vbExclamation
    CILX2(VCI) = 0
    GoTo ET3
Else
CILZ1(VCI) = InputBox("INGRESE Z1 CILÍNDRICO")
If CILZ1(VCI) > DP Then
    MsgBox "DATO Y1 NO VÁLIDO", vbExclamation
    CILZ1(VCI) = 0
    GoTo ET3
Else
CILZ2(VCI) = InputBox("INGRESE Z2 CILÍNDRICO")
If CILZ2(VCI) >= CILZ1(VCI) Then
    MsgBox "DATO Y2 NO VÁLIDO", vbExclamation
    CILZ2(VCI) = 0
    GoTo ET3
Else
Picture1.AutoRedraw = True
Picture1.Line (CILX1(VCI), CILZ1(VCI))-(CILX2(VCI), CILZ1(VCI))
Picture1.Line -(CILX2(VCI), CILZ2(VCI))
Picture1.Line -(CILX1(VCI), CILZ2(VCI))
Picture1.Line -(CILX1(VCI), CILZ1(VCI))
'SIMETRÍA
Picture1.Line (CILX1(VCI), -CILZ1(VCI))-(CILX2(VCI), -CILZ1(VCI))
Picture1.Line -(CILX2(VCI), -CILZ2(VCI))
Picture1.Line -(CILX1(VCI), -CILZ2(VCI))

Picture1.Line -(CILX1(VCI), -CILZ1(VCI))
Picture1.AutoRedraw = False
Call BURIL
SI = True
CCI = CCI + 1
NCCI = CCI
NMCIL.Caption = CCI
If VCI = 2 Then
    MsgBox "LÍMITE DE MAQUINADOS CILÍNDRICOS"
    DCIL.Enabled = False
Else
VCI = VCI + 1
End If
End If
End If
End If
End If
CPZ = CILZ1(VCI)
ET3:
End Sub

Private Sub DCON_Click()
CONX1(VCO) = InputBox("INGRESE X1 CONICO")
If CONX1(VCO) > LP Then
    MsgBox "COORDENADA X1 CÓNICO NO VÁLIDA"
    CONX1(VCO) = 0
    GoTo ET3
Else
CONX2(VCO) = InputBox("INGRESE X2 CÓNICO")
If CONX2(VCO) > LP Or CONX2(VCO) = CONX1(VCO) Then
    MsgBox "COORDENADA X2 CÓNICO NO VÁLIDA"
    CONX2(VCO) = 0
    GoTo ET3
Else
CONZ1(VCO) = InputBox("INGRESE Z1 CÓNICO")
If CONZ1(VCO) > DP Then
    MsgBox "COORDENADA Z1 CÓNICO NO VÁLIDA"
    CONZ1(VCO) = 0
    GoTo ET3
Else
CONZ2(VCO) = InputBox("INGRESE Z2 CONICO")
If CONZ2(VCO) > LP Or CONZ2(VCO) = CONZ1(VCO) Then
    MsgBox "COORDENADA Z2 CÓNICO NO VÁLIDA"
    CONZ2(VCO) = 0
    GoTo ET3
```

```
Else
If CONZ1(VCO) >= CONZ2(VCO) Then
    MsgBox "COORDENADAS Z1 Y Z2 NO VALIDAS"
    MsgBox "CONDICIÓN SIEMPRE Z2 > Z1--INGRESE COORDENADAS DE NUEVO", vbExclamation
    CONX1(VCO) = 0
    CONX2(VCO) = 0
    CONZ1(VCO) = 0
    CONZ2(VCO) = 0
    GoTo ET3
Else
Picture1.AutoRedraw = True
Picture1.Line (CONX1(VCO), CONZ1(VCO))-(CONX2(VCO), CONZ2(VCO))
Picture1.Line (CONX1(VCO), -CONZ1(VCO))-(CONX2(VCO), -CONZ2(VCO))
Picture1.AutoRedraw = False
Call BURIL
SO = True
CCO = CCO + 1
NCCO = CCO
NMCON.Caption = CCO
If VCO = 2 Then
    MsgBox "LIMITE DE MAQUINADOS CÓNICOS"
    BCON.Enabled = False
Else
VCO = VCO + 1
End If
End If
End If
End If
End If
End If
ET3:
End Sub

Private Sub DESF_Click()
ESFX1(VES) = InputBox("INGRESE X1 ESFÉRICO")
If ESFX1(VES) > LP Then
    MsgBox "COORDENADA X1 ESFÉRICO NO VALIDA"
    ESFX1(VES) = 0
    GoTo ET4
Else
ESFZ1(VES) = InputBox("INGRESE Z1 ESFÉRICO")
If ESFZ1(VES) > DP Then
    MsgBox "COORDENADA Z1 ESFÉRICO NO VALIDA"
    ESFZ1(VES) = 0
    GoTo ET4
Else
ESFX2(VES) = InputBox("INGRESE X2 ESFÉRICO")
If ESFX2(VES) > LP Or ESFX2(VES) = ESFX1(VES) Then
    MsgBox "COORDENADA X2 ESFÉRICO NO VALIDA"
    ESFX2(VES) = 0
    GoTo ET4
Else
ESFZ2(VES) = InputBox("INGRESE Z2 ESFÉRICO")
If ESFZ2(VES) > DP Then
    MsgBox "COORDENADA Z1 ESFÉRICO NO VALIDA"
    ESFZ2(VES) = 0
    GoTo ET4
Else
RADIO(VES) = InputBox("INGRESE RADIO")
'RESTRINGIR RADIO''''''''''''''''''''''''''''''''''''''<<<<<<<<<<<<<<<<<<<<<<<<<<JOJOJOJOJOJOJOJOJOJOJOJOJOJOJOJOJOJOJO
If RADIO(VES) > 61340 Then
    MsgBox "RECORDAR RESTRINGIR EL RADIO"
    MsgBox "RADIO MAXIMO 61340"
Else
AUX1 = (ESFZ1(VES) - ESFZ2(VES)) / (ESFX2(VES) - ESFX1(VES))
AUX2 = (ESFX2(VES) ^ 2 - ESFX1(VES) ^ 2 + ESFZ2(VES) ^ 2 - ESFZ1(VES) ^ 2) / (2 * (ESFX2(VES) - ESFX1(VES)))
A = AUX1 ^ 2 + 1
B = -2 * ESFX1(VES) * AUX1 + 2 * AUX1 * AUX2 - 2 * ESFZ1(VES)
C = ESFX1(VES) ^ 2 + ESFZ1(VES) ^ 2 - 2 * ESFX1(VES) * AUX2 + AUX2 ^ 2 - RADIO(VES) ^ 2
CZ1(VES) = (-B + Sqr(B ^ 2 - 4 * A * C)) / (2 * A)
CX1(VES) = AUX1 * CZ1(VES) + AUX2
CZ2(VES) = (-B - Sqr(B ^ 2 - 4 * A * C)) / (2 * A)
CX2(VES) = AUX1 * CZ2(VES) + AUX2
If RADIO(VES) > 0 Then
    RAD(VES) = RADIO(VES)
    Picture1.AutoRedraw = True
    For ZN = ESFZ2(VES) To ESFZ1(VES) Step -1
        XN = (2 * CX2(VES) + Sqr((-2 * CX2(VES)) ^ 2 - 4 * (CX2(VES) ^ 2 + (ZN) ^ 2 - 2 * ZN * CZ2(VES) + (CZ2(VES)) ^ 2 - (RAD(VES)) ^ 2)
        Picture1.PSet (XN, ZN)
        Picture1.PSet (XN, -ZN)
    Next ZN
    Picture1.AutoRedraw = False
    GoTo ET5
Else
    RAD(VES) = Abs(RADIO(VES))
    Picture1.AutoRedraw = True
    For ZN = ESFZ2(VES) To ESFZ1(VES) Step -1
        XN = (2 * CX1(VES) - Sqr((-2 * CX1(VES)) ^ 2 - 4 * (CX1(VES) ^ 2 + (ZN) ^ 2 - 2 * ZN * CZ1(VES) + (CZ1(VES)) ^ 2 - (RAD(VES)) ^ 2)
        Picture1.PSet (XN, ZN)
        Picture1.PSet (XN, -ZN)
```

```
    Next ZN
    Picture1.AutoRedraw = False
    GoTo ET5
End If
End If
End If
End If
End If
End If
ET5:
CES = CES + 1
HCES = CES
SF = True
NMESF.Caption = CES
If CES = 3 Then
    MsgBox "LIMITE DE MAQUINADOS ESFÉRICO ALCANSADO"
    DESF.Enabled = False
Else
    VES = VES + 1
End If
ET4:
CILX1(VCI) = 0
CILZ1(VCI) = 0
CILX2(VCI) = 0
CILZ2(VCI) = 0
End Sub
```

```
Private Sub DW_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
If UP.Enabled = False Then
    UP.Enabled = True
End If
If Label16.Caption = 0 Then
    MsgBox "INGRESE VALOR DE AVANCE <> 0"
Else
TDW.Enabled = True
End If
End Sub
```

```
Private Sub DW_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
TDW.Enabled = False
End Sub
```

```
' ********************************************************************
' when the form loads, connect to the HID controller - pass
' the form window handle so that you can receive notification
' events...
'********************************************************************
Private Sub Form_Load()
    ' do not remove!
    ConnectToHID (Me.hwnd)
'LINEAS POR DAVID
'CBX = HBX - 2000
CBX = 16072
'CBZ = HBZ - 1000
CBZ = 2143
Call BURIL
End Sub
```

```
'********************************************************************
' disconnect from the HID controller...
'********************************************************************
Private Sub Form_Unload(Cancel As Integer)
    DisconnectFromHID
End Sub
```

```
'********************************************************************
' a HID device has been plugged in...
'********************************************************************
Public Sub OnPlugged(ByVal pHandle As Long)
    If hidGetVendorID(pHandle) = VendorID And hidGetProductID(pHandle) = ProductID Then
        ' ** YOUR CODE HERE **
    End If
End Sub
```

```
'********************************************************************
' a HID device has been unplugged...
'********************************************************************
Public Sub OnUnplugged(ByVal pHandle As Long)
    If hidGetVendorID(pHandle) = VendorID And hidGetProductID(pHandle) = ProductID Then
        ' ** YOUR CODE HERE **
    End If
End Sub
```

```
'********************************************************************
' controller changed notification - called
' after ALL HID devices are plugged or unplugged
'********************************************************************
Public Sub OnChanged()
    Dim DeviceHandle As Long
    ' get the handle of the device we are interested in, then set
    ' its read notify flag to true - this ensures you get a read
    ' notification message when there is some data to read...
    DeviceHandle = hidGetHandle(VendorID, ProductID)
    hidSetReadNotify DeviceHandle, True
End Sub
```

```
'********************************************************************
' on read event...
'********************************************************************
Public Sub OnRead(ByVal pHandle As Long)

    ' read the data (don't forget, pass the whole array)...
    If hidRead(pHandle, BufferIn(0)) Then
        ' ** YOUR CODE HERE **
        ' first byte is the report ID, e.g. BufferIn(0)
        ' the other bytes are the data from the microcontrolller...
    End If
End Sub
```

```
'********************************************************************
' this is how you write some data...
'********************************************************************
Public Sub WriteSomeData()
    BufferOut(0) = 0    ' first by is always the report ID
    BufferOut(1) = 10   ' first data item, etc etc

    ' write the data (don't forget, pass the whole array)...
    hidWriteEx VendorID, ProductID, BufferOut(0)
End Sub
```

```
Private Sub HOME_Click()
If CBZ < HBZ Then
    UP.Enabled = False
    DW.Enabled = False
    LF.Enabled = False
    RG.Enabled = False
    CAPCP.Enabled = False
    VELMAN.Enabled = False
    Check1.Enabled = False
    TH2.Enabled = True
End If
End Sub
```

```
Private Sub LF_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
If Label16.Caption = 0 Then
    MsgBox "INGRESE VALOR DE AVANCE <> 0"
Else
TLF.Enabled = True
End If
If RG.Enabled = False Then
    RG.Enabled = True
End If
End Sub
```

```
Private Sub LF_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
TLF.Enabled = False
End Sub
```

```
Private Sub MANDOR_Click()
PX = Val(MRX.Text)
PZ = Val(MRZ.Text)
Acero.Enabled = True
End Sub
```

```
Private Sub MAQUINAR_Click()
PROF = InputBox("INGRESE PROFUNDIDAD DE CORTE")
Call SEGUIMIENTO
End Sub
```

```
Private Sub PFCON_Timer()
If CBZ >= CONZ2(VCO) Then
    XN = (((CPRZ - Z) - CONZ1(VCO)) / m + CONX1(VCO))
    If CBX = XN Then
        If CBZ = DP + SO Then
            PFCON.Enabled = False
            If PAP = True Then
            MsgBox "maquinado terminado"
            End If
            Call SEGUIMIENTO
        Else
        CBZ = CBZ + 1
        Call BURIL
        Call SELECTORZP
        End If
    Else
    CBX = CBX - 1
    Call BURIL
    Call SELECTORXN
```

```
    End If
Else
If CBZ = CPRZ Then
    XN = (((CPRZ - 2) - CONZ1(VCO)) / m + CONX1(VCO))
    If CBX <= XN Then
        CPRZ = CPRZ + 1
    Else
    CBX = CBX - 1
    Call BURIL
    Call SELECTORXN
    End If
Else
CBZ = CBZ + 1
Call BURIL
Call SELECTORZP
End If
End If
End Sub

Private Sub PFESF_Timer()
If CBZ >= ESFZ2(VES) Then
    XN = (2 * CX2(VES) + Sqr((-2 * CX2(VES)) ^ 2 - 4 * (CX2(VES) ^ 2 + (CPRZ) ^ 2 - 2 * CPRZ * CZ2(VES) + (CZ2(VES)) ^ 2 - (RAD(VES)) ^ 2)
    If CBX = XN Then
        If CBZ = DP + 50 Then
            PFESF.Enabled = False
            If PAP = True Then
            MsgBox "maquinado terminado"
            End If
            Call SEGUIMIENTO
        Else
        CBZ = CBZ + 1
        Call BURIL
        Call SELECTORZP
        End If
    Else
    CBX = CBX - 1
    Call BURIL
    Call SELECTORXN
    End If
Else
If CBZ = CPRZ Then
    XN = (2 * CX2(VES) + Sqr((-2 * CX2(VES)) ^ 2 - 4 * (CX2(VES) ^ 2 + (CPRZ) ^ 2 - 2 * CPRZ * CZ2(VES) + (CZ2(VES)) ^ 2 - (RAD(VES)) ^ 2)
    If CBX <= XN Then
        CPRZ = CPRZ + 1
    Else
    CBX = CBX - 1
    Call BURIL
    Call SELECTORXN
    End If
Else
CBZ = CBZ + 1
Call BURIL
Call SELECTORZP
End If
End If
End Sub
Private Sub PFESFN_Timer()
If CBZ >= ESFZ2(VES) Then
    XN = (2 * CX1(VES) - Sqr((-2 * CX1(VES)) ^ 2 - 4 * (CX1(VES) ^ 2 + (CPRZ) ^ 2 - 2 * CPRZ * CZ1(VES) + (CZ1(VES)) ^ 2 - (RAD(VES)) ^ 2)
    If CBX = XN Then
        If CBZ = DP + 50 Then
            PFESFN.Enabled = False
            If PAP = True Then
            MsgBox "maquinado terminado"
            End If
            Call SEGUIMIENTO
        Else
        CBZ = CBZ + 1
        Call BURIL
        Call SELECTORZP
        End If
    Else
    CBX = CBX - 1
    Call BURIL
    Call SELECTORXN
    End If
Else
If CBZ = CPRZ Then
    XN = (2 * CX1(VES) - Sqr((-2 * CX1(VES)) ^ 2 - 4 * (CX1(VES) ^ 2 + (CPRZ) ^ 2 - 2 * CPRZ * CZ1(VES) + (CZ1(VES)) ^ 2 - (RAD(VES)) ^ 2)
    If CBX <= XN Then
        CPRZ = CPRZ + 1
    Else
    CBX = CBX - 1
    Call BURIL
    Call SELECTORXN
    End If
Else
CBZ = CBZ + 1
Call BURIL
Call SELECTORZP
```

```
End If
End If
End Sub

Private Sub RG_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
If Label1.Caption = 0 Then
    MsgBox "INGRESE VALOR DE AVANCE <> 0"
Else
TRG.Enabled = True
End If
If LF.Enabled = False Then
    LF.Enabled = True
End If
End Sub

Private Sub RG_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
TRG.Enabled = False
End Sub

Private Sub SH_Timer()

End Sub

Private Sub TDW_Timer()
CBZ = CBZ + 1
Call BURIL
Call SELECTORZP
If CBZ = 3575 Then
    MsgBox "LÍMITE SUPERIOR EJE Z", vbExclamation
    TDW.Enabled = False
    DW.Enabled = False
End If
End Sub

Private Sub THX_Timer()
CBX = CBX + 1
Call BURIL
Call SELECTORXP
If CBX = HBX Then
    THX.Enabled = False
    UP.Enabled = True
    DW.Enabled = True
    LF.Enabled = True
    RG.Enabled = True
    CAPCP.Enabled = True
    VELMAN.Enabled = True
    Check1.Enabled = True
End If
End Sub

Private Sub TH2_Timer()
CBZ = CBZ + 1
Call BURIL
Call SELECTORZP
If CBZ = HBZ Then
    TH2.Enabled = False
    If CBX <> HBX Then
        THX.Enabled = True
    End If
End If
End Sub

Private Sub Timer1_Timer()
If CBZ = CPRZ Then

    XN = (2 * CX1(VES) - Sqr((-2 * CX1(VES)) ^ 2 - 4 * (CX1(VES) ^ 2 + (CPRZ) ^ 2 - 2 * CPRZ * CZ1(VES) + (CZ1(VES)) ^ 2 - (RAD(VES)) ^ 2)
    If CBX = XN Then
        THFX.Enabled = False
        THFZ.Enabled = True
    Else
    CBX = CBX - 1
    Call BURIL
    Call SELECTORXN
    End If
Else
If CBX = ESFX1(0) Then
    THFX.Enabled = False
    THF2.Enabled = True
Else
CBX = CBX + 1
Call BURIL
Call SELECTORXP
End If
End If
End Sub

Private Sub Timer2_Timer()
If CBX = XN Then
    If CPRZ = ESFZ1(0) - 2 Then
```

```
        If CBZ = DP + 50 Then
            MsgBox "fin de rutina"
            TMFZ.Enabled = False
            CPRZ = 0
            CBZ = 0
            PROF = 0
        Else
        CBZ = CBZ + 1
        Call BURIL
        Call SELECTORZP
        End If
    Else
    CPRZ = CPRZ + 10
    If CBZ = CBZ Then
        CPRZ = CPRZ - PROF
        TMFZ.Enabled = False
        TMFX.Enabled = True
    Else
    CBZ = CBZ + 1
    Call BURIL
    Call SELECTORZP
    End If
    End If
Else
DAUX = CPRZ / ESFZ1(0)
If DAUX > 1 Then
    If CBZ = CPRZ Then
        TMFZ.Enabled = False
        TMFX.Enabled = True
    Else
    CBZ = CBZ - 1
    Call BURIL
    Call SELECTORZN
    End If
Else
CPRZ = ESFZ1(0) - 2
ZN = CPRZ
If CBZ = CPRZ Then
    TMFZ.Enabled = False
    PFESF.Enabled = True
    'TMCX.Enabled = True
Else
CBZ = CBZ - 1
Call BURIL
Call SELECTORZN
```

```
        If CBZ = DP + 50 Then
            MsgBox "fin de rutina"
            TMFZ.Enabled = False
            CPRZ = 0
            CBZ = 0
            PROF = 0
        Else
        CBZ = CBZ + 1
        Call BURIL
        Call SELECTORZP
        End If
    Else
    CPRZ = CPRZ + 10
    If CBZ = CPRZ Then
        CPRZ = CPRZ - PROF
        TMFZ.Enabled = False
        TMFX.Enabled = True
    Else
    CBZ = CBZ + 1
    Call BURIL
    Call SELECTORZP
    End If
    End If
Else
DAUX = CPRZ / ESFZ1(0)
If DAUX > 1 Then
    If CBZ = CPRZ Then
        TMFZ.Enabled = False
        TMFX.Enabled = True
    Else
    CBZ = CBZ - 1
    Call BURIL
    Call SELECTORZN
    End If
Else
CPRZ = ESFZ1(0) - 2
ZN = CPRZ
If CBZ = CPRZ Then
    TMFZ.Enabled = False
    PFESF.Enabled = True
    'TMCX.Enabled = True
Else
CBZ = CBZ - 1
Call BURIL
Call SELECTORZN
```

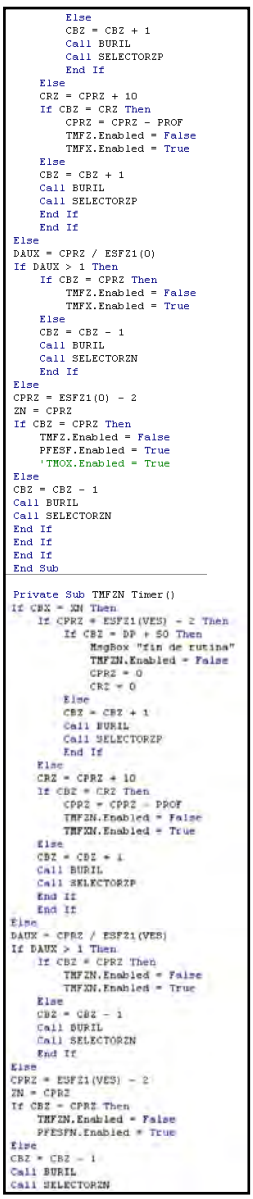
```
End If
End If
End If
End Sub

Private Sub Timer3_Timer()
If CBZ >= ESFZ2(VES) Then
    ZN = (2 * CX1(VES) - Sqr((-2 * CX2(VES)) ^ 2 - 4 * (CX1(VES) ^ 2 + (CPRZ) ^ 2 - 2 * CPRZ * CZ1(VES) + (CZ1(VES)) ^ 2 - (RAD(VES)) ^ 2)
    If CBZ = ZN Then
        If CBZ = DP + 50 Then
            PFESF.Enabled = False
            MsgBox "maquinado terminado"
        Else
        CBZ = CBZ + 1
        Call BURIL
        Call SELECTORZP
        End If
    Else
    CBX = CBX - 1
    Call BURIL
    Call SELECTORXN
    End If
Else
If CBZ = CPRZ Then
    ZN = (2 * CX1(VES) - Sqr((-2 * CX2(VES)) ^ 2 - 4 * (CX1(VES) ^ 2 + (CPRZ) ^ 2 - 2 * CPRZ * CZ1(VES) + (CZ1(VES)) ^ 2 - (RAD(VES)) ^ 2)
    If CBZ <= ZN Then
        CPRZ = CPRZ + 1
    Else
    CBX = CBX - 1
    Call BURIL
    Call SELECTORXN
    End If
Else
CBZ = CBZ + 1
Call BURIL
Call SELECTORZP
End If
End If
End Sub

Private Sub TLF_Timer()
CBX = CBX - 1
Call BURIL
Call SELECTORXN
If CBX = 0 Then
    MsgBox "LIMITE INFERIOR EJE X", vbExclamation
    TLF.Enabled = False
    LF.Enabled = False
End If
End Sub


Private Sub TMCX_Timer()
If CBZ = CPRZ Then
    If CBX = CILX2(VCI) Then
        TMCX.Enabled = False
        TMCZ.Enabled = True
    Else
    CBX = CBX - 1
    Call BURIL
    Call SELECTORXN
    End If
Else
If CBX = CILX1(VCI) Then
    TMCX.Enabled = False
    TMCZ.Enabled = True
Else
CBX = CBX + 1
Call BURIL
Call SELECTORXP
End If
End If
End Sub

Private Sub TMCZ_Timer()
If CBX = CILX2(VCI) Then
    If CPRZ = CIL22(VCI) - 2 Then
        If CBZ = DP + 50 Then
            If PAP = True Then
            MsgBox "MAQUINADO TERMINADO"
            End If
            TMCZ.Enabled = False
            Call SEGUIMIENTO
        Else
        CBZ = CBZ + 1
        Call BURIL
        Call SELECTORZP
        End If
    Else
    CRZ = CPRZ + 10
```

```
    If CBZ = CRZ Then
        CPRZ = CPRZ - PROF
        TMCZ.Enabled = False
        TMCX.Enabled = True
    Else
    CBZ = CBZ + 1
    Call BURIL
    Call SELECTORZP
    End If
    End If
Else
DAUX = CPRZ / CILZ2(VCI)
If DAUX > 1 Then
    If CBZ = CPRZ Then
        TMCZ.Enabled = False
        TMCX.Enabled = True
    Else
    CBZ = CBZ - 1
    Call BURIL
    Call SELECTORZN
    End If
Else
CPRZ = CILZ2(VCI) - 2
If CBZ = CPRZ Then
    TMCZ.Enabled = False
    TMCX.Enabled = True
Else
CBZ = CBZ - 1
Call BURIL
Call SELECTORZN
End If
End If
End If
End Sub
```

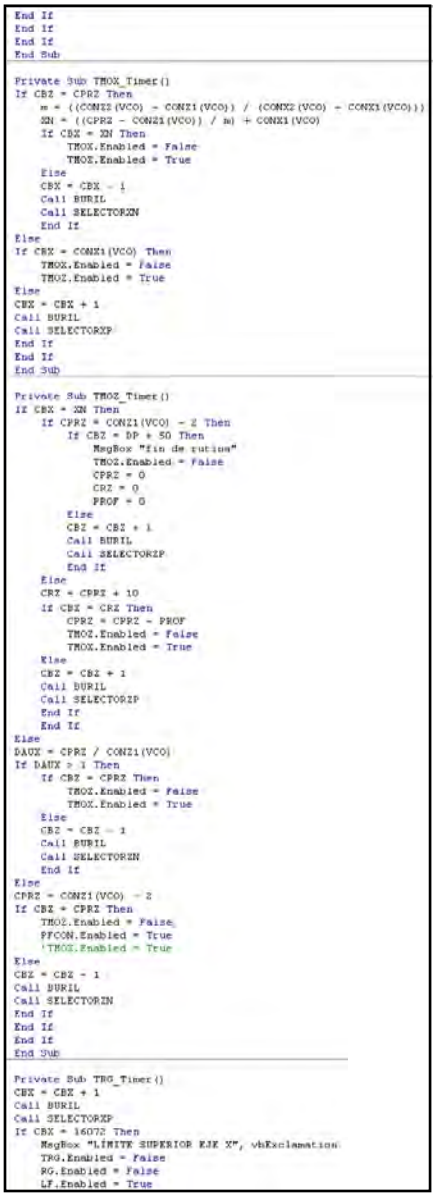
```
Private Sub TMFX_Timer()
If CBZ = CPRZ Then
    XN = (2 * CX2(VES) + Sqr((-2 * CX2(VES)) ^ 2 - 4 * (CX2(VES) ^ 2 + (CPRZ) ^ 2 - 2 * CPRZ * CZ2(VES) + (CZ2(VES)) ^ 2 - (RAD(VES)) ^ 2)
    If CBX = XN Then
        TMFX.Enabled = False
        TMFZ.Enabled = True
    Else
    CBX = CBX - 1
    Call BURIL
    Call SELECTORXN
    End If
Else
If CBX = ESFZ1(VES) Then
    TMFX.Enabled = False
    TMFZ.Enabled = True
Else
CBX = CBX + 1
Call BURIL
Call SELECTORXP
End If
End If
End Sub
```

```
Private Sub TMFXN_Timer()
If CBZ = CPRZ Then
    XN = (2 * CX1(VES) - Sqr((-2 * CX1(VES)) ^ 2 - 4 * (CX1(VES) ^ 2 + (CPRZ) ^ 2 - 2 * CPRZ * CZ1(VES) + (CZ1(VES)) ^ 2 - (RAD(VES)) ^ 2)
    If CBX = XN Then
        TMFXN.Enabled = False
        TMFZN.Enabled = True
    Else
    CBX = CBX - 1
    Call BURIL
    Call SELECTORXN
    End If
Else
If CBX = ESFX1(VES) Then
    TMFXN.Enabled = False
    TMFZN.Enabled = True
Else
CBX = CBX + 1
Call BURIL
Call SELECTORXP
End If
End If
End Sub
```

```
Private Sub TMFZ_Timer()
If CBX = XN Then
    If CPRZ = ESFZ1(0) - 2 Then
        If CBZ = DP + 50 Then
            MsgBox "fin de rutina"
            TMFZ.Enabled = False
            CPRZ = 0
            CRZ = 0
            PROF = 0
```



```
        Else
        CBZ = CBZ + 1
        Call BURIL
        Call SELECTORZP
        End If
    Else
    CRZ = CPRZ + 10
    If CBZ = CRZ Then
        CPRZ = CPRZ - PROF
        TMFZ.Enabled = False
        TMFX.Enabled = True
    Else
    CBZ = CBZ + 1
    Call BURIL
    Call SELECTORZP
    End If
    End If
Else
DAUX = CPRZ / ESFZ1(O)
If DAUX > 1 Then
    If CBZ = CPRZ Then
        TMFZ.Enabled = False
        TMFX.Enabled = True
    Else
    CBZ = CBZ - 1
    Call BURIL
    Call SELECTORZN
    End If
Else
CPRZ = ESFZ1(O) - 2
ZN = CPRZ
If CBZ = CPRZ Then
    TMFZ.Enabled = False
    PFESF.Enabled = True
    'TMOX.Enabled = True
Else
CBZ = CBZ - 1
Call BURIL
Call SELECTORZN
End If
End If
End If
End Sub

Private Sub TMFZN_Timer()
If CBX = XN Then
    If CPRZ = ESFZ1(VES) - 2 Then
        If CBZ = DP + 50 Then
            MsgBox "fin de rutina"
            TMFZN.Enabled = False
            CPRZ = 0
            CRZ = 0
        Else
        CBZ = CBZ + 1
        Call BURIL
        Call SELECTORZP
        End If
    Else
    CRZ = CPRZ + 10
    If CBZ = CRZ Then
        CPRZ = CPRZ - PROF
        TMFZN.Enabled = False
        TMFXN.Enabled = True
    Else
    CBZ = CBZ + 1
    Call BURIL
    Call SELECTORZP
    End If
    End If
Else
DAUX = CPRZ / ESFZ1(VES)
If DAUX > 1 Then
    If CBZ = CPRZ Then
        TMFZN.Enabled = False
        TMFXN.Enabled = True
    Else
    CBZ = CBZ - 1
    Call BURIL
    Call SELECTORZN
    End If
Else
CPRZ = ESFZ1(VES) - 2
ZN = CPRZ
If CBZ = CPRZ Then
    TMFZN.Enabled = False
    PFESFN.Enabled = True
Else
CBZ = CBZ - 1
Call BURIL
Call SELECTORZN
```



```
End If
End If
End If
End Sub

Private Sub TMOX_Timer()
If CBZ = CPRZ Then
    m = ((CONX2(VCO) - CONX1(VCO)) / (CONX2(VCO) - CONX1(VCO)))
    XN = ((CPRZ - CONZ1(VCO)) / m) + CONX1(VCO)
    If CBX = XN Then
        TMOX.Enabled = False
        TMOZ.Enabled = True
    Else
    CBX = CBX - 1
    Call BURIL
    Call SELECTORXN
    End If
Else
If CBX = CONX1(VCO) Then
    TMOX.Enabled = False
    TMOZ.Enabled = True
Else
CBX = CBX + 1
Call BURIL
Call SELECTORXP
End If
End If
End Sub

Private Sub TMOZ_Timer()
If CBX = XN Then
    If CPRZ = CONZ1(VCO) - 2 Then
        If CBZ = DP + 50 Then
            MsgBox "fin de rutina"
            TMOZ.Enabled = False
            CPRZ = 0
            CRZ = 0
            PROF = 0
        Else
        CBZ = CBZ + 1
        Call BURIL
        Call SELECTORZP
        End If
    Else
    CRZ = CPRZ + 10
    If CBZ = CRZ Then
        CPRZ = CPRZ - PROF
        TMOZ.Enabled = False
        TMOX.Enabled = True
    Else
    CBZ = CBZ + 1
    Call BURIL
    Call SELECTORZP
    End If
    End If
Else
DAUX = CPRZ / CONZ1(VCO)
If DAUX > 1 Then
    If CBZ = CPRZ Then
        TMOZ.Enabled = False
        TMOX.Enabled = True
    Else
    CBZ = CBZ - 1
    Call BURIL
    Call SELECTORZN
    End If
Else
CPRZ = CONZ1(VCO) - 2
If CBZ = CPRZ Then
    TMOZ.Enabled = False
    PFCON.Enabled = True
    'TMOX.Enabled = True
Else
CBZ = CBZ - 1
Call BURIL
Call SELECTORZN
End If
End If
End If
End Sub

Private Sub TRG_Timer()
CBX = CBX + 1
Call BURIL
Call SELECTORXP
If CBX = 16072 Then
    MsgBox "LÍMITE SUPERIOR EJE X", vbExclamation
    TRG.Enabled = False
    RG.Enabled = False
    LF.Enabled = True
```

```
End If
End Sub

Private Sub TUP_Timer()
CBI = CBI - 1
Call BURIL
Call SELECTORZN
If CBI = 0 Then
    MsgBox "LaMITE INFERIOR EJE Z", vbExclamation
    TUP.Enabled = False
    UP.Enabled = False
End If
End Sub

Private Sub UP_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
If DW.Enabled = False Then
    DW.Enabled = True
End If
If Label16.Caption = 0 Then
    MsgBox "INGRESE VALOR DE AVANCE <> 0"
Else
TUP.Enabled = True
End If
End Sub

Private Sub UP_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
TUP.Enabled = False
End Sub

Private Sub VELMAN_Click()
ET1:
VAM = InputBox("INGRESE VALOR ENTRE 1 Y 255")
If VAM <= 0 Or VAM > 255 Then
    MsgBox "VALOR NO VALIDO"
    GoTo ET1
Else
Label16.Caption = VAM
TUP.Interval = VAM
TDW.Interval = VAM
TRG.Interval = VAM
TLF.Interval = VAM
End If
End Sub
```

Fin de la programación.