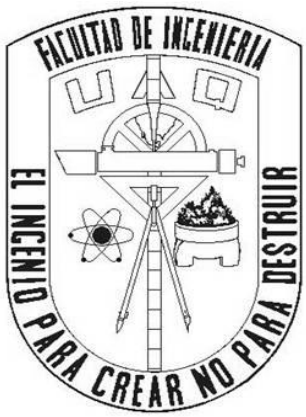
UNIVERSIDAD AUTÓNOMA DE QUERÉTARO

Campus San Juan del Río



FACULTAD DE INGENIERÍA

ELECTROMECÁNICA

GENERADOR MANUAL DE TRAYECTORIAS 3D PARA POSICIONAMIENTO DE ROBOT CARTESIANO

TESIS

Que como parte de los requisitos para obtener el título de

INGENIERO ELECTROMECÁNICO

PRESENTA:

JORGE MÉNDEZ CRUZ

DIRECTOR DE TESIS:

M. EN C. MANUEL GARCÍA QUIJADA

Presentación

El desarrollo de este proyecto se realizó en las instalaciones de la Universidad Autónoma de Querétaro Campus San Juan del Río, en la Facultad de Ingeniería Electromecánica al inicio del segundo semestre del año 2011 y está enfocado a la programación de robots soldadores de 3 grados de libertad -expandible a cuatro o más-, en donde la mayoría de los casos, se encuentran en espacios reducidos o ruidosos desde el punto de vista eléctrico. Bajo estas condiciones, es difícil implementar algún sistema de programación con cámaras que calculen las trayectorias a seguir por los robots soldadores. Cabe mencionar que la programación de estos robots es muy compleja, puesto que en la mayoría de los casos las trayectorias a seguir no son sencillas, es decir, constan de muchos arcos, segmentos de línea recta, cambios de plano de trabajo, etc., por este motivo para cada cambio que la trayectoria tenga, por más pequeño que este sea se debe ir programando el robot, lo que implica demasiado código de programación, tiempo y la implementación del programa.

Es este documento se presenta la metodología del proyecto con el que se pretende agilizar dicha programación del robot soldadores, sin la necesidad de la implementación de cámaras o sensores y reducir considerablemente el tiempo y la complejidad de dicha programación. Para ello solamente bastará mover el robot por el usuario u operador siguiendo una trayectoria por medio de un dispositivo espacial (*Mouse 3D*), guardar la trayectoria, para que posteriormente el robot la ejecute cuantas veces sea necesario, o modificarla en el momento que se requiera, de esta manera el robot seguirá la nueva trayectoria descrita. Con ello se agiliza la programación y tiempo empleados en este tipo de robots.

Agradecimientos

Este trabajo está dedicado especialmente a mi madre Juana Cruz Martínez, a mi padre Jorge Méndez Trejo, a mis hermanas y a toda mi familia porque ellos estuvieron conmigo dándome siempre todo lo mejor e impulsándome a seguir adelante.

Quiero agradecer a mis padres por todo lo que hasta ahora me han brindado; lo que soy ahora, sin ellos jamás hubiera podido ser, gracias por darme la oportunidad de realizar mi carrera profesional y poder así concluir esta etapa tan importante en mi vida, gracias por su amor, sus consejos, la confianza y todo el apoyo que me han dado desde siempre; este trabajo es por ustedes y para ustedes; siempre estaré completamente agradecido con ustedes por todo lo que han hecho por mí, los amo.

Gracias también a mi profesor Manuel García Quijada por la oportunidad que me dio de realizar este trabajo con él, por todo el conocimiento que compartió con mis compañeros de clase y conmigo durante la universidad, y por la amistad que desde entonces me ha brindado.

A mis sinodales les agradezco por el apoyo, la orientación y el tiempo que dedicaron durante la realización y conclusión de este trabajo, gracias.

También, quiero agradecer a todos mis profesores, porque cada uno de ellos fue una parte fundamental durante y para concluir mi carrera profesional.

Gracias a todos y cada uno de ustedes por darme esta oportunidad de seguir creciendo como persona y profesionalmente, por todo, mil gracias.

Resumen

El desarrollo e implementación de este proyecto se realizó en un robot cartesiano, el cual opera bajo la generación de códigos G, los cuales se mencionan en el contenido de este trabajo, dichos códigos fueron desarrollados e implementados con ayuda del programa realizado que en esencia es el presente proyecto. La base de estos códigos son las coordenadas cartesianas a partir de un punto para mover el robot dentro del espacio de trabajo, y para determinar las coordenadas de dichos códigos, estas fueron calculadas en base a geometría analítica para desarrollar líneas rectas y arcos atreves de puntos y radios para el caso de arcos, además del uso de transformaciones lineales para cambiar de plano de trabajo.

Otra parte del proyecto es la interfaz entre el usuario y el robot cartesiano, esta interfaz consta de una PC y un mouse 3D con los cuales, el usuario controla los movimientos del robot y captura la trayectoria a seguir.

El programa desarrollado permite mejorar el error que el usuario tiene al ir generando la trayectoria, es decir, si se realiza una línea recta, aunque el usuario genere arcos durante el movimiento del robot pero el comando seleccionado es el de línea recta, esta se hará cuando el programa de la trayectoria se ejecute.

Cada parte desarrollada de este proyecto se describe en el capítulo IV, donde se mencionan las pruebas realizadas, los herramentales necesarios para estas pruebas y su construcción, también la precisión de la trayectoria generada por el programa con respecto a la trayectoria original, entre otras características; además de las conclusiones y recomendaciones que se dan para un mejor manejo y una mayor eficiencia del programa.

CONTENIDO

CAPÍTULO I INTRODUCCIÓN

1.1 Estado del conocimiento

Querétaro, dinámico centro de desarrollo industrial, se encuentra en la región central de México, colindando con cinco estados: San Luis Potosí, Hidalgo, Michoacán, Estado de México y Guanajuato. El estado se destaca a nivel nacional por su gran infraestructura; la excelente localización y la calidad de sus parques industriales les permite tener acceso a las redes ferroviaria, carretera, telefónica, satelital y de energía eléctrica.

El objetivo general del sector industrial es impulsar la competitividad del estado y así fomentar una mayor inversión, con la cual, habrá una generación de más y mejores empleos.

La industria manufacturera del estado es dinámica, innovadora, productiva, competitiva y con un peso fundamental en la generación de empleos bien remunerados.

Este desarrollo de la economía diversifica y fortalece la estructura productiva de la entidad, es decir, dicho desarrollo es una constante instalación de innovadoras empresas en el sector industrial y oportunidades de más y mejores empleos para incrementar la calidad de vida de los queretanos.

La Encuesta Nacional de Ocupación y Empleo realizada por el INEGI en el último trimestre de 2008, muestra que el personal ocupado de la población en la entidad en el sector secundario es el 32.3%, del cual, el 65.9% se encuentra dentro del a industria manufacturera y de aquí, el 34.4% del Producto Interno Bruto (PIB) manufacturero es aportado por los productos metálicos, maquinaria y equipo.

De acuerdo a la última publicación del INEGI sobre el PIB por entidad federativa, la industria en Querétaro aporta el 36.7% del PIB total. De este porcentaje, la industria manufacturera aporta el 25.8%.

El DIME (Directorio Maestro Empresarial) es una herramienta de consultas para empresas que realizan alguna actividad industrial en Querétaro. La cobertura de este directorio abarca todo el estado de Querétaro y está conformado por un total aproximado de 1300 empresas; de este total aproximado el 39% de las empresas pertenecen al sector metal-mecánico.

Una de las actividades principales de una empresa metal-mecánica es adquirir por medio de una serie de procedimientos el cambio en la apariencia, forma o propiedades de cierto metal hasta llegar a los productos deseados como por ejemplo: herramientas, accesorios de máquinas, equipamientos, vehículos y materiales de transporte, ente otros.

El porcentaje de las empresas por municipio registradas en el DIME[1] se muestra en la Tabla 1.1. En la Tabla 1.2 se pueden observar los porcentajes del total de las empresas registradas entre los diferentes sectores industriales en el estado.

TABLA 1.1 Porcentaje de empresas registradas en el DIME por cada municipio del Estado de Querétaro. FUENTE: SEDESU

Municipio	Empresas registradas (%)
Amealco	0.4
Cadereyta	0.4
Colón	0.3
Corregidora	6.8
El Marqués	9.9
Ezequiel Montes	0.4
Huimilpan	0.1
Pedro Escobedo	0.6
Querétaro	72.1
San Juan del Río	8.3
Tequisquiapan	0.6
Tolimán	0.0
Total aproximado	1300

[1] http://www2.queretaro.gob.mx/sedesu/desemp/dime.html

TABLA 1.2 Porcentaje de empresas registradas en el DIME por sector industrial en el estado de Querétaro. FUENTE: SEDESU

Sector	Empresas registradas (%)
Agroindustria	0.5
Alimentos, bebidas y tabacos	4.1
Construcción	0.5
Eléctrica y electrónica	3.8
Madera	1.3
Metal básica	0.4
Metal mecánica	**39.0**
Minerales no metálicos	1.5
Papel e imprenta	3.6
Química, caucho y plástic	8.0
Servicios	15.6
Textiles	2.1
Otras industrias	19.6
Total aproximado	1300

El panorama presentado, permite intuir que el área metal mecánica está bastante extendida a nivel estatal y local. En este contexto, aunque al momento no es posible cuantificar de manera exacta el porcentaje de empresas que tengan entre sus procesos máquinas programables para la realización de actividades referentes al ensamble, maquinado y soldadura automatizada, si es posible intuir que el proceso de programación de este tipo de maquinaria es lento y requiere desde luego personal calificado y contratación externa de servicio para programar y/o poner en marcha las máquinas y líneas de producción.

En el área de la automatización, la principal aplicación de los robots tiene lugar en la industria, donde es habitual la repetición de tareas tales como la fabricación en serie de piezas y maquinaria. Esto obliga a realizar toda tarea exactamente igual. Un robot está programado para realizar los mismos movimientos y con la misma precisión, por lo que es ideal en aplicaciones industriales repetitivas.

El panorama actual de la robótica en Querétaro es muy variado, hay desde kits para armar un robot humanoide, hasta procedimientos de como armar robots caseros con componentes de computadora personal (PC*, personal computer*). La robótica ha tomado un auge creciente y vemos robots de todo tipo y para fines diversos, sin embargo, aún cuando el Estado de Querétaro tiene un desarrollo industrial importante, no se tienen fuentes de información confiables para cuantificar la distribución de robots y su impacto en las aplicaciones industriales.

El auge actual de la robótica se debe a varias circunstancias entre las que se pueden destacar:

- La alta velocidad de procesamiento de los microprocesadores actuales y el bajo costo de los componentes electrónicos.
- El uso de internet, ya que a través de esta gran red se pueden conocer y controlar robots ya creados así como adquirir piezas (motores, servomotores, etc.) que a nivel local son muy difíciles de conseguir.

Como ya se mencionó anteriormente, en Querétaro cada vez llegan al estado empresas nuevas e innovadoras tanto en procesos para realizar algún trabajo como en la tecnología que estas utilizan.

Entre las tecnologías se encuentran los robots que son ya muy empleados para la mayoría de los procesos de las industrias y principalmente en las manufactureras, de aquí que surge este proyecto.

1.1.1 La Robótica en la Industria

A pesar del uso extendido en diferentes industrias manufactureras, la completa automatización de algunos procesos como es la soldadura industrial no se ha alcanzado debido a que los parámetros del proceso no son completamente entendidos y cuantificados (Daeinabi y Teshnehlab, 2006).

En la industria, los robots son utilizados para la manufactura, ensamblar piezas, para soldar, para pintar, entre otras actividades.

Algunas actividades ocasionales en donde se emplean robots son dentro del entorno submarino, nuclear, para la desactivación de bombas, el espacio exterior, etc.

Dentro de los intentos que se están realizando para lograr el control total de diversos procesos en el que requiere calculo de trayectorias, existen sistemas para la automatización de robots basados en el concepto de Espacio de Trabajo Inteligente, el cual se genera con cámaras fotográficas digitales para describir las trayectorias que los robots deben seguir; empleando procesamiento de imágenes. Este concepto se basa en colocar las cámaras fotográficas alrededor del espacio de trabajo del robot como se muestra en la Figura 1.1a) para tener una mejor construcción de la trayectoria por donde el robot debe avanzar a diferencia del Concepto Convencional que consiste en colocar las cámaras fotográficas en el robot como es mostrado en la figura 1.1b). Con esto, el método de Espacio de Trabajo Inteligente ha tenido serios problemas para ser utilizado en la industria ya que se necesita adecuar el espacio de trabajo con buena iluminación para que se tenga un buen resultado en el procesamiento de imágenes.

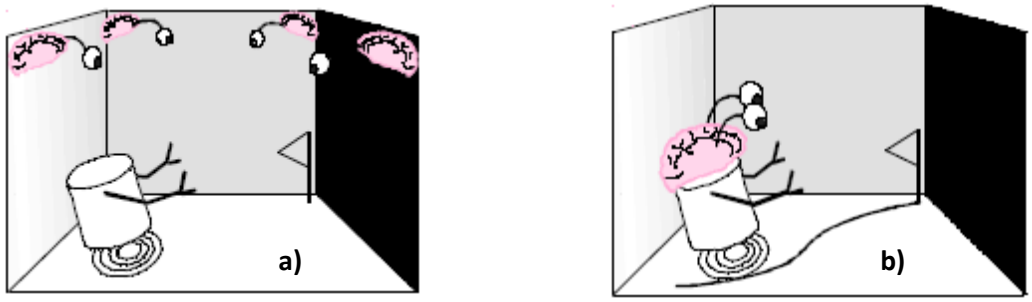


Figura 1.1 a) Concepto de espacio inteligente, b) Concepto convencional. (Takarics, et al, 2006)

A pesar de esto, el margen de error reportados para el seguimiento de las trayectorias es de 10 mm; este error es muy grande a nivel industrial y podría reducirse empleando cámaras digitales industriales de alto desempeño. (Takarics, $et\ al$, 2006)

Algunos otros sistemas enfocados hacia el control de robots son los empleados para soldar las bases de los barcos. Estos sistemas se programan por medio de la enseñanza del robot, es decir, estos son programados por algún operador que enseña al robot la trayectoria de la soldadura que debe seguir. El problema radica en que dichas bases son muy grandes y no son iguales exactamente entre sí. También, estos sistemas de robots pueden ser programados a través de un archivo electrónico de Diseño Asistido por Computadora (CAD, *Computer Aideed Design*) que describe el entorno donde el robot operará y las regiones de las piezas de trabajo que serán soldadas. Estos robots pueden ser programados fuera de línea a través del modelado en CAD y probar el programa antes de que este sea cargado al robot real. Una desventaja de este tipo de programación es que se debe describir exactamente la pieza de trabajo y una buena disponibilidad del espacio de trabajo ya que el robot es autónomo. Además, se requiere de un programador especializado para dichos sistemas o un entrenador para los robots. Aquí, el proceso de la soldadura debiera ser lo más exacto posible pero la colocación de los refuerzos y el pandeo de los paneles o bases del barco producen errores arriba de los 5 mm lo que implica volver a programar los robots. (Ang Jr., *et al,* 1999)

Otros sistemas robóticos automatizados para soldadura funcionan por medio del rastreo de sus trayectorias. Daeinabi y Teshnehlab en 2006 clasificaron que existen dos tipos de sistemas automatizados para el seguimiento de trayectorias, los cuales son: el tipo de ***no visión*** que para su funcionamiento emplea sensores basados en principios de sonido, magnetismo, capacitancia, etc. Y el segundo tipo es el de ***visión*** del cual, su forma más típica es un escaneo visual de un láser o alguna otra luz estructurada. Para el seguimiento de trayectorias basada en la visión, una cámara toma una fotografía de la junta a soldar y calcula la trayectoria para el futuro cordón de soldadura empleando procesamiento de imágenes. Esto se logra por medio de algoritmos que reconoce los bordes de la junta en la imagen de la pieza de trabajo.

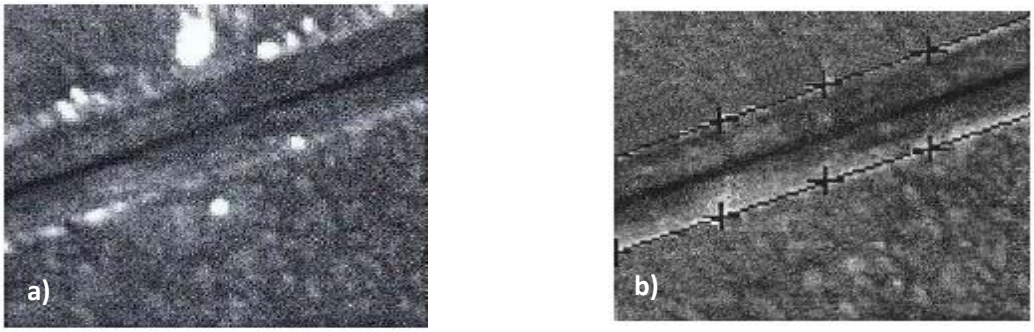


Figura 1.2 a) Sin bordes procesados, b) Con bordes procesados. (Daeinabi y Teshnehlab, 2006)

Algunos resultados de estos sistemas se ilustran en la Figura 1.2 a) donde se observa la imagen sin el reconocimiento de los bordes. En la Figura 1.2 b) se localizan estos bordes por medio de marcas en forma de cruz descritas por el algoritmo.

Estos son algunos ejemplos de los proyectos que se han desarrollado para la automatización de los robots en la industria, con esto puede deducirse que el acondicionamiento del lugar de trabajo y la automatización para estos robots es lenta y compleja; esta problemática proporciona el motivo suficiente para realizar el desarrollo de este proyecto, el cual se describe en los siguientes temas de este trabajo.

1.2 Objetivos

General.

Desarrollar, integrar e implementar un sistema basado en un dispositivo manual 3D (*mouse 3D*) que permita generar la trayectoria de la herramienta de un robot cartesiano de tres grados de libertad.

Específicos.

- Revisar la literatura referente a la programación de trayectorias de robots actuales.

- Implementar la interfaz gráfica entre un *mouse* 3D y la PC empleando el entorno de desarrollo *Microsoft Visual Studio 2008*.
- Realizar la interfaz entre la PC y el robot cartesiano empleando la comunicación serial RS232.
- Hacer la integración del *mouse*3D*,* la PC y el robot.
- Realizar pruebas de posicionamiento del robot en línea y fuera de línea.

1.3 Justificación e hipótesis

La automatización de los robots juega un papel muy importante en el mejoramiento de la productividad y la calidad de ésta, pero en muchos de estos procesos, la automatización se vuelve muy compleja por diferentes factores como por ejemplo, cuando se tienen que seguir trayectorias de corte o soldadura variantes, la programación de los robots que llevan a cabo estos trabajos deben controlarlos en cada cambio de su trayectoria, esto implica una lógica de mayor dificultad y un código más extenso en la programación de los robots.

Durante procesos industriales como la soldadura por arco eléctrico, se presentan muchas variables que pueden afectar este proceso y dar como resultado soldaduras insatisfactorias. Algunas de estas variables son la distorsión mecánica, la variación dimensional de la pieza de trabajo, una sujeción inapropiada o con obstáculos, además, si este proceso es hecho manualmente, el ser humano provoca otras variables desfavorables en la soldadura como la interrupción, una mala fusión de las piezas de trabajo en posiciones incómodas, provocación de porosidades dentro de los cordones de soldadura, etc. En contraste, las exigencias crecientes a la productividad y calidad empujan a la necesidad de robots en la industria, esto conlleva a automatizar este tipo de robots para que el aprovechamiento de estos sea favorable.

Con la soldadura automatizada se tienen las siguientes ventajas sobre la soldadura manual: se puede soldar en todas las posiciones, se tiene una buena apariencia y calidad en la soldadura, se puede soldar en condiciones extremas tales como altas temperaturas, ambientes de trabajo agresivos tales como aéreas radiactivas, entre otras. (Daeinabi y Teshnehlab, 2006)

Es así como surge la idea de emplear un método alternativo a los esquemas clásicos de generación de trayectorias para la automatización de maquinaria industrial con posibles aplicaciones de soldadura.

Algunas ventajas que engloba este proyecto de generación de trayectorias son las siguientes:

1. Se pretende realizar la tele-operación con componentes de fácil adquisición y bajo costo.
2. Para el posicionamiento del robot mediante este proyecto, no es necesaria la implementación de cámaras de video o fotográficas ni de programas de procesamiento de imágenes que reconozcan patrones para que el robot pueda llegar a su trayectoria.
3. El sistema permitirá que un operador, que no esté familiarizado con el robot, pueda posicionarlo con sólo mover el *mouse* 3D sin necesidad de realizarle una programación avanzada, actuadores, sensores e interfaces de elevados costos entre otros elementos necesarios para llevar a cabo la interacción con el robot.

1.4 Planteamiento General

El proyecto consiste en realizar un programa que interactúe entre el usuario, el dispositivo manual (mouse 3D) y el robot a controlar. En la Figura 1.3 se ilustra de forma esquemática la configuración general del sistema.

Para posicionar el robot mediante el método de tele-operación, es necesario contar con un dispositivo manual que permita al usuario mover al robot a una posición deseada. Para ello se pretende emplear un *mouse 3D* y obtener las coordenadas de los ejes cartesianos X, Y y/o Z mediante una interfaz programada en *Visual Studio 2008* y residente en la PC que genera las trayectorias que el robot debe seguir. Posteriormente, cuando ya se hayan capturado las coordenadas del *mouse 3D*, en el mismo programa se debe hacer la conversión a coordenadas absolutas para poder calcular virtualmente la posición del robot. Si se sobrepasaran las coordenadas propuestas del espacio de trabajo habría que evitar que el robot real salga del

espacio de trabajo, ya que podría dañarse mecánica o electrónicamente al provocar (intencional o accidentalmente) el operador un punto fuera de sus límites. Por lo tanto, el programa también deberá evitar que esto suceda con ayuda del registro de dichas coordenadas y comparándolas con las coordenadas máximas del espacio de trabajo real que el robot puede alcanzar en cada uno de sus ejes.

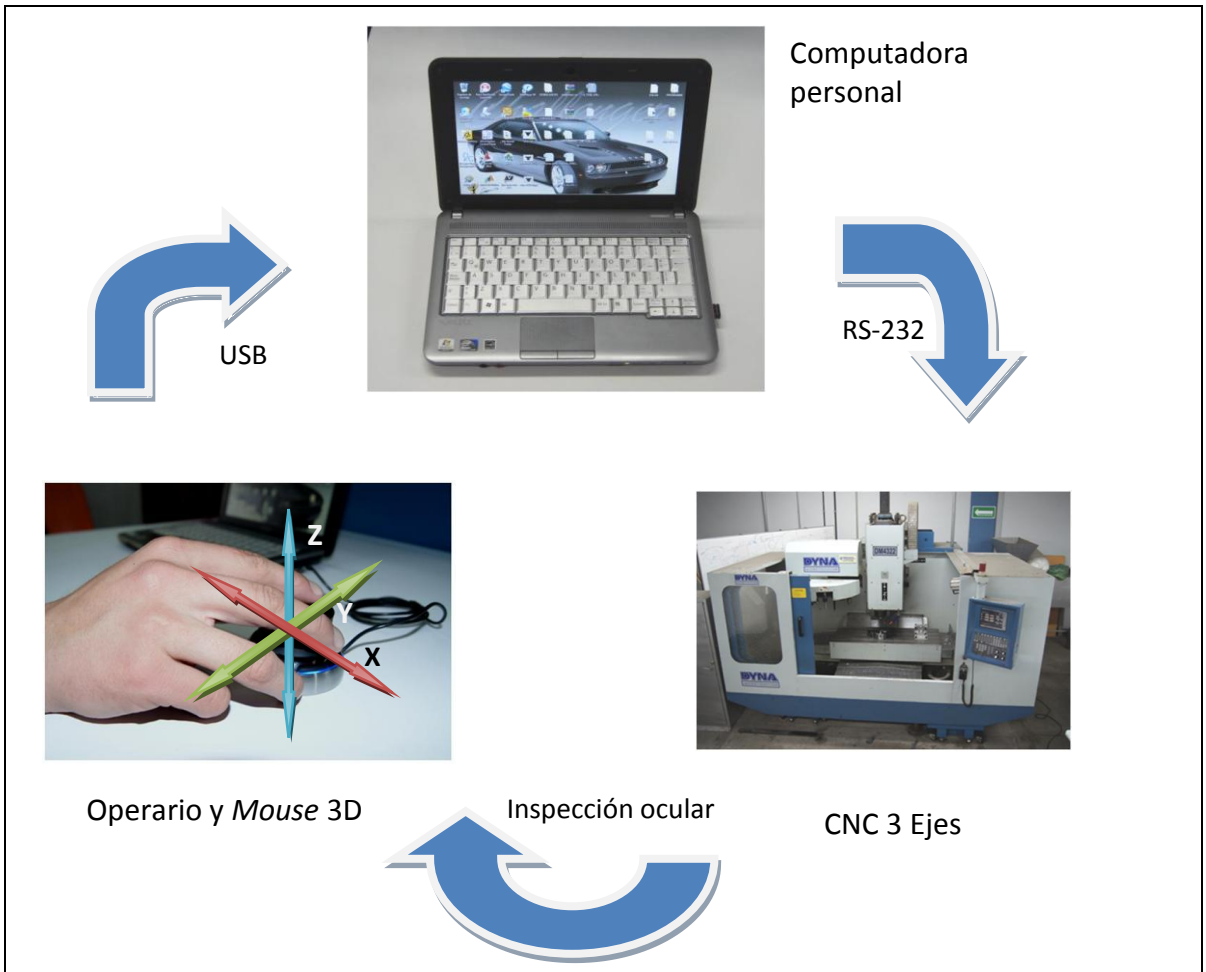


Figura 1.3 Planteamiento general del sistema generador de trayectorias.

Cuando las coordenadas sean las adecuadas, es decir, que ya se haya verificado que no sobrepasan los límites del robot, estas pasarán por un módulo de conversión para transformar las coordenadas del *mouse 3D* a códigos G, el cual, es el lenguaje que procesará el control del robot al que se le va a implementar este programa. Dichas pruebas se efectuarán en un centro de

fresado de Control Numérico Computarizado (CNC, *Computerized Numerical Control*) con tres grados de libertad basada en el principio de un robot cartesiano. Cuando la conversión a código G se finaliza, éste es enviado a través de una interfaz entre la PC y el robot empleando la comunicación serial RS-232 en modo de Control Numérico Distribuido (DNC, *Distributed Numerical Control*).

Cabe describir brevemente el modo DNC. Las entradas y salidas del puerto RS-232 serán utilizadas para enviar y recibir datos. Las fuentes externas normalmente son un disco duro. En múltiples aplicaciones, los programas son transferidos por medio de DNC. El control tiene características disponibles para hacer posible la transferencia de datos.

Para la comunicación entre una máquina CNC y una computadora usando el puerto RS-232, todo el equipo requerido es un cable serial entre los dos dispositivos y un software.

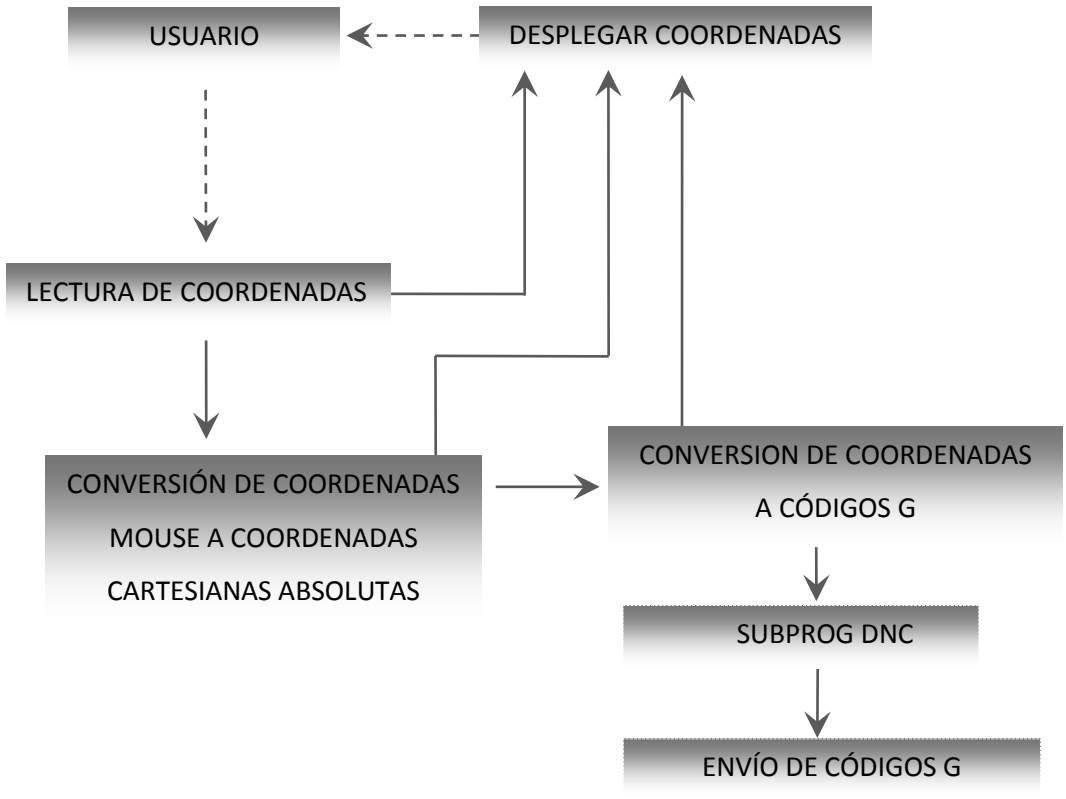


Figura 1.4 **Esquema general de programa generador de trayectorias 3D para posicionamiento de robot cartesiano**.

Para comunicarse con dos o más máquinas utilizando el mismo puerto RS-232, cada máquina debe ser unida a una caja de hendidura con un cable. La caja de hendidura está disponible con dos o más salidas, seleccionables por un botón seleccionador. Esta es la forma más simple de DNC. Requiere de procedimientos bien organizados para hacerlo trabajar de manera eficiente. El DNC no es una parte de la unidad de control.

Una vez descrito el DNC y el envío de código G realizado, el robot debe procesar el código y posicionarse en las coordenadas descritas en dicho código; con esto, el objetivo del proyecto se cumple.

Este programa además, debe desplegar en la pantalla de la PC las coordenadas que el *mouse*3D está proporcionando, el código G que se está ejecutando y las coordenadas del robot. Este planteamiento es esquematizado en la Figura 1.4 en donde se mencionan las cinco principales acciones que el programa debe realizar.

CAPÍTULO II REVISIÓN DE LITERATURA

2.1 Generalidades

Para el desarrollo y el buen funcionamiento de este proyecto se deben tener algunos conocimientos sobre transformaciones lineales, las cuales son el sustento matemático de éste, ya que se necesitaron para generar líneas en el espacio a través de proyecciones ortogonales por dar un ejemplo, ya que este es un caso en particular de transformaciones lineales, existen otros casos, los cuales también son mencionados más adelante en este capítulo.

Además de tener en cuenta las transformaciones lineales, es necesario tener conocimiento sobre geometría analítica, específicamente sobre la línea recta y el círculo, es decir, conocer y poder determinar las ecuaciones que describen estas dos regiones geométricas, ya que el programa desarrollado en esta investigación genera líneas rectas y arcos a través de sus ecuaciones para la descripción de las trayectorias por seguir; encontrando soluciones para la intersección de una línea recta con un círculo y un círculo que se ajuste a tres puntos dados. Estas soluciones son descritas más adelante.

También, es necesario tener conocimiento de lo que son los entrenadores *teach pendant* para tener una referencia de lo que en este proyecto es desarrollado, ya que dichos entrenadores y este proyecto tienen en común el seguimiento de una trayectoria cualesquiera.

Ahora bien, como este proyecto se desarrolló en un centro de fresado, se requiere tener conocimiento sobre el funcionamiento de estas máquinas, por ejemplo, deben de conocerse las partes principales de éstas máquinas y cuál es la función de cada una de estas, también, las diferentes formas de programar en éstas máquinas, cuales son las ventajas de los centros de fresado por control numérico, tales como la precisión y repetibilidad, entre otras más que son descritas con detalle en los próximos temas.

Este proyecto se comunica con el centro de fresado a través del puerto serial de la PC, por medio del protocolo de comunicación serial RS 232 (un cable USB-DB9), es por eso que debe conocerse el funcionamiento de este protocolo para evitar problemas de confusión y errores con el envío de señales mandadas por dicho protocolo al centro de fresado. Además, debe de comprenderse lo que es la comunicación DNC, que es con la cual este proyecto trabaja pero todo esto, está explicado a detalle más adelante en este capítulo.

El *mouse* 3D, es el dispositivo de entrada empleado en este proyecto, es decir, es el dispositivo con el cual el usuario introduce los datos al programa desarrollado, por tal motivo se debe tener conocimiento sobre el funcionamiento de dicho dispositivo. Algunas de las funciones descritas son: los modos de operación del dispositivo tales como el modo cámara y el modo objeto, además de la orientación de cada eje, es decir, en qué dirección está el eje X, Y y Z, hacia donde es la referencia positiva y negativa de cada uno de estos y las unidades máximas que tiene cada eje, entre otras.

Pasando al lenguaje de programación, se describe cómo funciona el entorno de Windows, ya que este es el sistema operativo en el cual es desarrollado este proyecto, en el compilador Microsoft Visual Studio 2008 y se describen algunas características de éste último tales como algunos comandos empleados para la programación y algunas otras características descritas posteriormente en este capítulo.

Después de esto, se mencionan y se definen lo que son códigos G y M, los cuales son necesarios para el funcionamiento del centro de fresado CNC, es decir, para la ejecución de funciones en dicho centro; por ende, estos códigos son generados en este proyecto, mas particularmente en el programa desarrollado, los cuales son mandados al centro de fresado para que éste ejecute la operación deseada. En este apartado del capítulo, se describen los códigos que dicho centro de fresado soporta y la estructura con la cual deben ser enviados.

Con estos conocimientos adquiridos se puede comprender el funcionamiento y desarrollo de este proyecto, además de una buena operación del programa desarrollado y con esto hacer de este trabajo una herramienta versátil para la solución de algún problema en torno a la finalidad de dicho proyecto.

2.2 Transformaciones Lineales

A finales del siglo XIX los matemáticos y los físicos comenzaron a darse cuenta de que no era necesario limitarse a las ternas $[a_1, a_2, a_3]$ para representar las coordenadas de posición de objetos en el espacio tridimensional. Reconocieron que las cuádruplas de números $[a_1, a_2, a_3, a_4]$ podían considerarse como puntos en el espacio "tetradimensional", las quíntuplas $[a_1, a_2, \ldots, a_5]$ como puntos en el espacio "pentadimensional" y así sucesivamente. (Anton, 2005)

Se puede observar que al estudiar el espacio tridimensional, la terna ordenada $[a_1, a_2, a_3]$ tiene dos interpretaciones geométricas: se puede interpretar como un punto, en cuyo caso a_1, a_2 y a_3 son las coordenadas como se muestra en la Figura 2.1a), o puede ser interpretado como un vector, donde a_1, a_2 y a_3 son las componentes como puede observarse en la Figura 2.1b). De aquí se considera que una *n*-ada ordenada $[a_1, a_2, \ldots, a_n]$ puede tomarse como un "punto generalizado" o como un "vector generalizado". (Anton, 2005)

Figura 2.1 a) Un punto en el espacio, b) Un vector en el espacio. (Anton, 2005)

El espacio físico tridimensional es un caso particular de un espacio vectorial. Un espacio vectorial es un conjunto de objetos, llamados vectores, que deben cumplir ciertas reglas algebraicas. Sean $\mathbf{v}$, $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$, y $\boldsymbol{w}$ elementos de un espacio vectorial $\mathcal{V}$, el cual esta definido sobre el conjunto de los reales, y sea α y β dos elementos de este conjunto, es decir, dos números reales. Las reglas mencionadas se resumen como sigue:

(i) La suma de $\mathbf{v}_1$ y $\mathbf{v}_2$, denotada como $\mathbf{v}_1 + \mathbf{v}_2$, es en si misma un elemento de $\mathcal{V}$ y es *conmutativa*, es decir, $\mathbf{v}_1 + \mathbf{v}_2 = \mathbf{v}_2 + \mathbf{v}_1$;

(ii) $\mathcal{V}$ contiene un elemento $\mathbf{0}$, llamado vector *cero* de $\mathcal{V}$,el cual, cuando se suma a cualquier otro elemento $\mathbf{v}$ de $\mathcal{V}$, permanece sin cambio, es decir, $\mathbf{v}+\mathbf{0}=\mathbf{v}$;

(iii) La suma definida en *(i)* es asociativa, es decir, $\mathbf{v}_1 + (\mathbf{v}_2 + \mathbf{v}_3) = (\mathbf{v}_1 + \mathbf{v}_2) + \mathbf{v}_3$;

(iv) Para cada elemento $\mathbf{v}$ de $\mathcal{V}$, existe un elemento correspondiente $\boldsymbol{w}$ también de $\mathcal{V}$, el cual cuando se suma a $\mathbf{v}$, produce el vector cero, es decir, $\mathbf{v}+\boldsymbol{w}=\mathbf{0}$. Además, $\boldsymbol{w}$ se representa como $-\mathbf{v}$;

(v) El producto $\alpha\mathbf{v}$, o $\mathbf{v}\alpha$, también es un elemento de $\mathcal{V}$, para cada $\mathbf{v}$ de $\mathcal{V}$ y cada número real α. Este producto es asociativo, es decir, $\alpha(\beta\mathbf{v}) = (\alpha\beta)\mathbf{v}$;

(vi) Si α es la unidad real, entonces $\alpha\mathbf{v}$ es idéntica a $\mathbf{v}$;

(vii) El producto definido en *(v)* es distributivo en el sentido que a) $(\alpha + \beta)\mathbf{v} = \alpha\mathbf{v} + \beta\mathbf{v}$ y b) $\alpha(\mathbf{v_1} + \mathbf{v_2}) = \alpha\mathbf{v_1} + \alpha\mathbf{v_2}$. (Angeles, 2007)

Aunque los espacios vectoriales pueden definirse sobre otros conjuntos, aquí se abordaran los espacios vectoriales sobre el conjunto de los números reales. Además, los espacios vectoriales pueden ser de dimensión finita o infinita, pero no se trataran a estos últimos. (Angeles, 2007)

Una transformación lineal representada como un operador $\mathbf{L}$ de un espacio vectorial $\mathcal{U}$ a un espacio vectorial $\mathcal{V}$, es una regla que asigna a cada vector $\mathbf{u}$ de $\mathcal{U}$, al menos a un vector $\mathbf{v}$ de $\mathcal{V}$ cuya representación es $\mathbf{v} = \mathbf{Lu}$. Tiene dos propiedades:

(i) Homogeneidad: $\mathbf{L}(\alpha\mathbf{u}) = \alpha\mathbf{v}$; y

(ii) Aditividad: $\mathbf{L}(\mathbf{u}_1 + \mathbf{u}_2) = \mathbf{v}_1 + \mathbf{v}_2$

Tipos particulares de transformaciones lineales del espacio euclidiano tridimensional $\mathcal{E}^3$ son proyecciones, reflexiones y rotaciones. (Angeles, 2007)

2.2.1 Proyección ortogonal

Una proyección ortogonal representada por la matriz $\mathbf{P}$ de $\mathcal{E}^3$ es en sí, una transformación lineal de dicho espacio en un plano Π que pasa a través del origen y posee un vector unitario normal $\mathbf{n}$, con las propiedades:

$$\mathbf{P^2} = \mathbf{P}, \mathbf{Pn} = \mathbf{0} \tag{2.1}$$

Cualquier matriz con la primera propiedad se le conoce como *idempotente*. La proyección $\mathbf{p}'$ de un vector de posición $\mathbf{p}$, en un plano Π de un vector unitario normal, es $\mathbf{p}$ menos la componente de $\mathbf{p}$ a lo largo del vector unitario $\mathbf{n}$ como se muestra en la Figura 2.2, es decir,

$$\mathbf{p}' = \mathbf{p} - \mathbf{n}(\mathbf{n}^T\mathbf{p}) \tag{2.2}$$

Donde el superíndice T denota la transpuesta de una matriz o un vector y $\mathbf{n}^T\mathbf{p}$ es equivalente al producto punto $\mathbf{n} \cdot \mathbf{p}$. (Angeles, 2007)

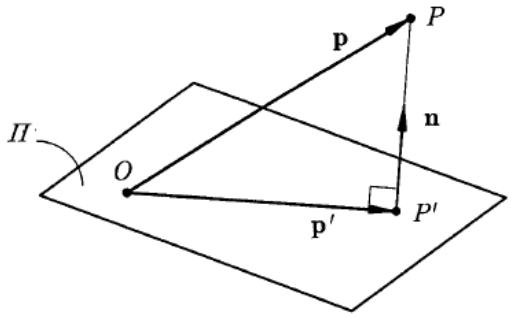


Figura 2.2 Proyección ortogonal en un plano. (Angeles, 2007)

Ahora la matriz identidad $\mathbf{1}$ se define como el mapeo de un espacio vectorial $\mathcal{V}$ en si mismo dejando sin cambio cada vector $\mathbf{v}$ de $\mathcal{V}$, es decir,

$$\mathbf{1v} = \mathbf{v} \tag{2.3}$$

Así, $\mathbf{p}'$ dada en la ec. (2.2), puede rescribirse como

$$\mathbf{p}' = \mathbf{1p} - \mathbf{nn}^T\mathbf{p} \equiv (\mathbf{1} - \mathbf{nn}^T)\mathbf{p} \tag{2.4}$$

Y así, la proyección ortogonal $\mathbf{P}$ en Π se representa como

$$\mathbf{P} = \mathbf{1} - \mathbf{n}\mathbf{n}^T \tag{2.5}$$

2.2.2 Reflexión

Una reflexión representada por la matriz $\mathbf{R}$ de $\mathcal{E}^3$ en un plano Π que pasa a través del origen, y que posee un vector unitario normal $\mathbf{n}$ es una transformación lineal dentro del mismo espacio, como se muestra en la Figura 2.3. Esto es, un vector $\mathbf{p}$ es mapeado por $\mathbf{R}$ dando como resultado un vector $\mathbf{p}'$ el cual, esta dado por

$$\mathbf{p}' = \mathbf{p} - 2\mathbf{n}\mathbf{n}^T\mathbf{p} \equiv (\mathbf{1} - 2\mathbf{n}\mathbf{n}^T)\mathbf{p} \tag{2.6}$$

Así, la reflexión $\mathbf{R}$ puede ser expresada como

$$\mathbf{R} = \mathbf{1} - 2\mathbf{n}\mathbf{n}^T \tag{2.7}$$

De la ec. (2.7) se observa que aparentemente una reflexión pura se representa por una transformación lineal simétrica y cuyo cuadrado es igual a la matriz identidad, es decir, $\mathbf{R}^2 = \mathbf{1}$. Ciertamente, la simetría, de la ec. (2.7) es evidente; la segunda propiedad es comprobada fácilmente como sigue

$$\begin{aligned} \mathbf{R^2} &= (\mathbf{1} - 2\mathbf{n}\mathbf{n}^T)(\mathbf{1} - 2\mathbf{n}\mathbf{n}^T) \\ &= \mathbf{1} - 2\mathbf{n}\mathbf{n}^T - 2\mathbf{n}\mathbf{n}^T + 4(\mathbf{n}\mathbf{n}^T)(\mathbf{n}\mathbf{n}^T) = \mathbf{1} \quad - 4\mathbf{n}\mathbf{n}^T + 4\mathbf{n}(\mathbf{n}^T\mathbf{n})\mathbf{n}^T \\ &= \mathbf{1} \quad - 4\mathbf{n}\mathbf{n}^T + 4\mathbf{n}\mathbf{n}^T = \mathbf{1} \end{aligned} \tag{2.8}$$

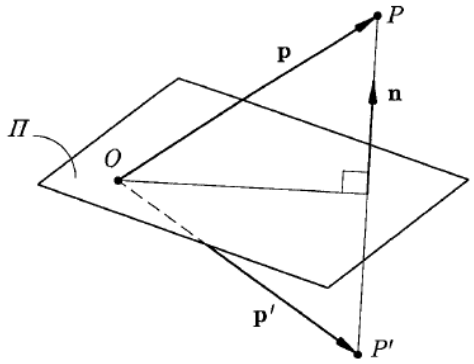


Figura 2.3 Reflexión en el espacio. (Angeles, 2007)

La cual aparentemente se reduce a $\mathbf{1}$ porque $\mathbf{n}$ es un vector unitario. Nótese que de la segunda propiedad se deduce que las reflexiones puras tienen una propiedad más

$$\mathbf{R^{-1}} = \mathbf{R} \tag{2.9}$$

Es decir, cada reflexión pura es igual a su inversa. Este resultado puede ser intuitivamente comprendido, notando que al reflejar dos veces una imagen usando dos espejos, la imagen original se conserva. Cualquier matriz cuadrada que sea igual a su inversa se le conoce como *auto-inversa*.

Ahora, derivamos la descomposición ortogonal de un vector $\mathbf{v}$ dado en dos componentes, una normal y una a lo largo de un vector unitario $\mathbf{e}$ como se muestra en la Figura 2.4. La componente de $\mathbf{v}$ a lo largo de $\mathbf{v}$, llamada componente axial $\mathbf{v}_{\parallel}$ es dada simplemente como

$$\mathbf{v}_{\parallel} = \mathbf{e}\mathbf{e}^{T}\mathbf{v} \tag{2.10}$$

Mientras que la componente normal correspondiente, $\mathbf{v}_{\perp}$ simplemente es la diferencia $\mathbf{v} - \mathbf{v}_{\parallel}$, es decir,

$$\mathbf{v}_{\perp} \equiv \mathbf{v} - \mathbf{v}_{\parallel} \equiv (\mathbf{1} - \mathbf{e}\mathbf{e}^{T})\mathbf{v} \tag{2.11}$$

La matriz dentro del paréntesis en la ecuación anterior, es muy frecuente en cinemática. (Angeles, 2007)

Otros conceptos que se necesitan son resumidos: la base de un espacio vectorial $\mathcal{V}$ es un conjunto de vectores linealmente independiente de $\mathcal{V}$, $\{\mathbf{v}_i\}_1^n$, en términos de cualquier vector $\mathbf{v}$ de $\mathcal{V}$ pueden expresarse como

$$\mathbf{v} = \alpha_1\mathbf{v_1} + \alpha_2\mathbf{v_2} + \cdots + \alpha_n\mathbf{v}_n \tag{2.12}$$

Donde los elementos del conjunto $\{\alpha_i\}_1^n$ son todos los elementos del conjunto sobre el que $\mathcal{V}$ se define, es decir, en este caso son números reales. El número n de elementos en el conjunto $\mathcal{B} = \{v_i\}_1^n$ se le nombra *dimensión* de $\mathcal{V}$. Observe que cualquier conjunto de n vectores linealmente independiente de $\mathcal{V}$ pueden ser la base de este espacio, pero una vez definida esta base, el conjunto de coeficientes reales $\{\alpha_i\}_1^n$ representan un vector v *único*. (Angeles, 2007)

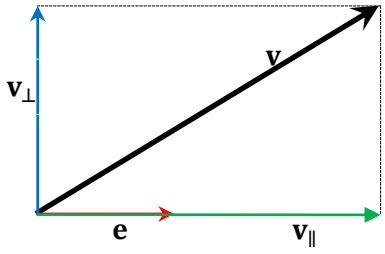


Figura 2.4 Descomposición ortogonal de un vector v dado.

Sean $\mathcal{U}$ y $\mathcal{V}$ dos espacios vectoriales de dimensión m y n respectivamente, L una transformación lineal de $\mathcal{U}$ a $\mathcal{V}$, y sean $\mathcal{B}_\mathcal{U}$ y $\mathcal{B}_\mathcal{V}$ bases para $\mathcal{U}$ y $\mathcal{V}$ como se definen a continuación:

$$\mathcal{B}_\mathcal{U} = \{\mathbf{u}_j\}_1^m, \quad \mathcal{B}_\mathcal{V} = \{\mathbf{v}_i\}_1^n \tag{2.13}$$

Ya que cada $\mathbf{Lu}_j$ es un elemento de $\mathcal{V}$, pueden representarse de manera única en términos de los vectores de $\mathcal{B}_\mathcal{V}$, es decir,

$$\mathbf{Lu}_j = l_{1j}\mathbf{v}_1 + l_{2j}\mathbf{v}_2 + \cdots + l_{nj}\mathbf{v}_n, \quad j = 1, \ldots, m \tag{2.14}$$

En consecuencia, a fin de representar la imagen de los m vectores de $\mathcal{B}_{\mathcal{U}}$, llamados el conjunto $\{\mathbf{L}\mathbf{u}_j\}_1^m$, se necesitan n x m números reales l_{ij}, para $i = 1, \ldots, n$ y $j = 1, \ldots, m$. Estos números reales se ordenan en un arreglo n x m como sigue:

$$[\mathbf{L}]_{\mathcal{B}_{\mathcal{U}}}^{\mathcal{B}_{\mathcal{V}}} \equiv \begin{bmatrix} l_{11} & l_{12} & \cdots & l_{1m} \\ l_{21} & l_{22} & \cdots & l_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{nm} \end{bmatrix} \tag{2.15}$$

Al arreglo anterior se le conoce como la matriz de representación de $\mathbf{L}$ con respecto a $\mathcal{B}_{\mathcal{U}}$ y $\mathcal{B}_{\mathcal{V}}$. (Angeles, 2007)

2.2.3 Rotación de cuerpo rígido

Un *isomorfismo lineal*, es decir, una transformación lineal una a una que mapea un espacio $\mathcal{V}$ en sí mismo, es conocida como una *isometría* si este conserva la distancia entre dos puntos cuales quiera de $\mathcal{V}$. Si $\mathbf{u}$ y $\mathbf{v}$ se consideran como los vectores de posición de dos puntos, entonces la distancia d entre estos dos puntos está definida como:

$$d \equiv \sqrt{(\mathbf{u} - \mathbf{v})^T(\mathbf{u} - \mathbf{v})} \tag{2.16}$$

2.2.4 Matriz producto-cruz

Antes de introducir la matriz de representación de una rotación, se comenzará por definir la derivada parcial de un vector con respecto a otro vector. Esta derivada es una matriz que se describe como sigue: sean $\mathbf{u}$ y $\mathbf{v}$ vectores de los espacios $\mathcal{U}$ y $\mathcal{V}$, de dimensiones m y n, respectivamente. Además, sea t una variable real y f una función de valor real de t, $\mathbf{u} = \mathbf{u}(t)$ y $\mathbf{v} = \mathbf{v}(\mathbf{u}(t))$ siendo funciones vectoriales de t de dimensiones m y n, con $f = f(\mathbf{u}, \mathbf{v})$. La derivada de $\mathbf{u}$ con respecto a t, denotada por $\dot{\mathbf{u}}(t)$, es un vector m-dimensional cuya i-esima

componente es la derivada de la i-esima componente u_i de $\mathbf{u}$, en una base dada, con respecto a t. Para $\dot{\mathbf{v}}(t)$ se observa una definición similar. La derivada parcial de f con respecto a $\mathbf{u}$ es un vector m-dimensional cuya i-esima componente es la derivada parcial de f con respecto a u_i, con una definición correspondiente para la derivada parcial de f con respecto a $\mathbf{v}$. Las derivadas anteriores, se denotarán como arreglos en *columnas*, así,

$$\frac{\partial f}{\partial \mathbf{u}} \equiv \begin{bmatrix} \partial f/u_1 \\ \partial f/u_2 \\ \vdots \\ \partial f/u_m \end{bmatrix}, \quad \frac{\partial f}{\partial \mathbf{v}} \equiv \begin{bmatrix} \partial f/v_1 \\ \partial f/v_2 \\ \vdots \\ \partial f/v_n \end{bmatrix} \tag{2.17}$$

Además, la derivada parcial de $\mathbf{v}$ con respecto a $\mathbf{u}$ es un arreglo de n x m cuyos elementos (i, j) están definidos como $\partial v_i/\partial u_j$, es decir,

$$\frac{\partial \mathbf{v}}{\partial \mathbf{u}} \equiv \begin{bmatrix} \partial v_1/u_1 & \partial v_1/u_2 & \cdots & \partial v_1/u_m \\ \partial v_2/u_1 & \partial v_2/u_2 & \cdots & \partial v_2/u_m \\ \vdots & \vdots & \ddots & \vdots \\ \partial v_n/u_1 & \partial v_n/u_2 & \cdots & \partial v_n/u_m \end{bmatrix} \tag{2.18}$$

Así, la matriz producto cruz de dos vectores $\mathbf{v}$ y $\mathbf{x}$ en un cierto marco de referencia $\mathcal{F}$ cuyas componentes son

$$[\mathbf{v}]_{\mathcal{F}} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}, \quad [\mathbf{x}]_{\mathcal{F}} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{2.19}$$

Entonces,

$$[\mathbf{v} \times \mathbf{x}]_{\mathcal{F}} = \begin{bmatrix} v_2 x_3 - v_3 x_2 \\ v_3 x_1 - v_1 x_3 \\ v_1 x_2 - v_2 x_1 \end{bmatrix} \tag{2.20}$$

Por lo tanto,

$$\left[\frac{\partial(\mathbf{v}\ \mathrm{x}\ \mathbf{x})}{\partial \mathbf{x}}\right]_{\mathcal{F}} = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix} \tag{2.21}$$

De aquí en adelante, la derivada parcial de $(\mathbf{v}\ \mathrm{x}\ \mathbf{x})$ con respecto a $\mathbf{x}$ se denotará por la matriz $\mathbf{V}$ de 3x3. Por obvias razones, $\mathbf{V}$ se conoce como la *matriz producto-cruz* del vector $\mathbf{v}$. Así, el producto cruz anterior admite las representaciones alternativas:

$$\mathbf{v}\ \mathrm{x}\ \mathbf{x} = \mathbf{V}\mathbf{x} \tag{2.22}$$

Una propiedad de la matriz producto-cruz $\mathbf{A}$ de cualquier vector 3-dimensional $\mathbf{a}$ se debe a que es *anti-métrica*, es decir,

$$\mathbf{A}^T = -\mathbf{A} \tag{2.23}$$

Y por consecuencia,

$$\mathbf{a}\ \mathrm{x}\ (\mathbf{a}\ \mathrm{x}\ \mathbf{b}) = \mathbf{A}^2\mathbf{b} \tag{2.24}$$

Donde $\mathbf{A}^2$ puede comprobarse que es

$$\mathbf{A}^2 = -\|\mathbf{a}\|^2\mathbf{1} + \mathbf{a}\mathbf{a}^T \tag{2.25}$$

(Angeles, 2007)

2.2.5 Matriz de rotación

A fin de obtener la representación de la matriz de rotación, se considera la rotación mostrada en la Figura 2.5 con un ángulo ϕ sobre la línea $\mathcal{L}$. (Angeles, 2007)

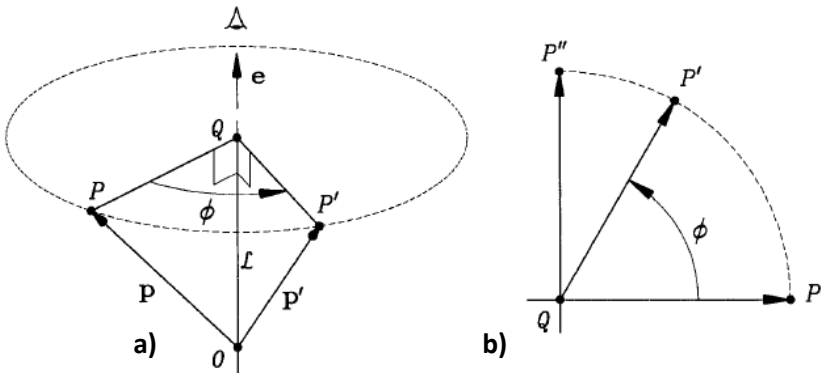


Figura 2.5 Rotación de un cuerpo rígido sobre una línea. (Angeles, 2007)

De la Figura 2.5a) se observa que uno puede escribir:

$$\mathbf{p}' = \overrightarrow{OQ} + \overrightarrow{QP'} \tag{2.26}$$

Donde $\overrightarrow{OQ}$ es la componente axial de $\mathbf{p}$ a lo largo del vector $\mathbf{e}$, el cual se obtuvo en la ec. (2.10),

Así,

$$\overrightarrow{OQ} = \mathbf{e}\mathbf{e}^T\mathbf{p} \tag{2.27}$$

Además, de la Figura 2.5b),

$$\overrightarrow{QP'} = (\cos\phi)\overrightarrow{QP} + (\sin\phi)\overrightarrow{QP''} \tag{2.28}$$

Donde $\overrightarrow{QP}$ es la *componente normal* de $\mathbf{p}$ con respecto a $\mathbf{e}$, como se mencionó en la ec. (2.11), es decir,

$$\overrightarrow{QP} = (\mathbf{1} - \mathbf{e}\mathbf{e}^T)\mathbf{p} \tag{2.29}$$

y $\overrightarrow{QP''}$ es

$$\overrightarrow{QP''} = \mathbf{e} \times \mathbf{p} \equiv \mathbf{E}\mathbf{p} \tag{2.30}$$

Con la sustitución de las ecs. (2.29) y (2.30) en la ec. (2.28) se tiene que

$$\overrightarrow{QP'} = \cos\phi\,(\mathbf{1} - \mathbf{e}\mathbf{e}^T)\mathbf{p} + \sin\phi\,\mathbf{E}\mathbf{p} \tag{2.31}$$

Si ahora las ecs. (2.27) y (2.31) son sustituidas en la ec. (2.26), uno obtiene

$$\mathbf{p}' = \mathbf{e}\mathbf{e}^T\mathbf{p} + \cos\phi(\mathbf{1} - \mathbf{e}\mathbf{e}^T)\mathbf{p} + \sin\phi\mathbf{E}\mathbf{p} \tag{2.32}$$

Así, la ec. (2.26) se reduce a

$$\mathbf{p}' = [\mathbf{e}\mathbf{e}^T + \cos\phi(\mathbf{1} - \mathbf{e}\mathbf{e}^T) + \sin\phi\mathbf{E}]\mathbf{p} \tag{2.33}$$

De la ec. (2.33) se deduce que $\mathbf{p}'$ es una transformación lineal de $\mathbf{p}$, aquí, la transformación está dada por la expresión dentro de los corchetes, la cual es la matriz de representación de la rotación $\mathbf{Q}$, es decir,

$$\mathbf{Q} = \mathbf{e}\mathbf{e}^T + \cos\phi(\mathbf{1} - \mathbf{e}\mathbf{e}^T) + \sin\phi\mathbf{E} \tag{2.34}$$

Un caso especial surge cuando $\phi = \pi$,

$$\mathbf{Q} = -\mathbf{1} + \mathbf{2}\mathbf{e}\mathbf{e}^T, \quad \text{para } \phi = \pi \tag{2.35}$$

Aquí se observa que $\mathbf{Q}$ es simétrica si $\phi = \pi$.

Otra forma de representación que se tiene de la matriz de rotación $\mathbf{Q}$ es:

$$\mathbf{Q} = \mathbf{1} + \sin\phi\,\mathbf{E} + (1 - \cos\phi)\mathbf{E}^2 \tag{2.36}$$

(Angeles, 2007)

2.2.6 Formas canónicas de la matriz de rotación

La matriz de rotación toma una forma simple si el eje de rotación coincide con uno de los ejes de coordenadas. Por ejemplo, si el eje X es paralelo al eje de rotación, es decir, paralelo al vector **e**, en un marco que se representará con la letra $\mathcal{X}$, entonces, se tiene:

$$[\mathbf{e}]_{\mathcal{X}} = \begin{bmatrix}1\\0\\0\end{bmatrix}, \quad [\mathbf{e}\mathbf{e}^T]_{\mathcal{X}} = \begin{bmatrix}1 & 0 & 0\\0 & 0 & 0\\0 & 0 & 0\end{bmatrix}, \quad [\mathbf{E}]_{\mathcal{X}} = \begin{bmatrix}0 & 0 & 0\\0 & 0 & -1\\0 & 1 & 0\end{bmatrix} \tag{2.37}$$

En el marco-$\mathcal{X}$, luego,

$$[\mathbf{Q}]_{\mathcal{X}} = \begin{bmatrix}1 & 0 & 0\\0 & \cos\phi & -\sin\phi\\0 & \sin\phi & \cos\phi\end{bmatrix} \tag{2.38}$$

Asimismo, si se definen los marcos de coordenadas $\mathcal{Y}$ y $\mathcal{Z}$ de tal manera que sus ejes Y y Z, respectivamente, coinciden con los ejes de rotación, entonces

$$[\mathbf{Q}]_{\mathcal{Y}} = \begin{bmatrix}\cos\phi & 0 & \sin\phi\\0 & 1 & 0\\-\sin\phi & 0 & \cos\phi\end{bmatrix} \tag{2.39}$$

Y

$$[\mathbf{Q}]_{\mathcal{Z}} = \begin{bmatrix}\cos\phi & -\sin\phi & 0\\\sin\phi & \cos\phi & 0\\0 & 0 & 1\end{bmatrix} \tag{2.40}$$

La representación de las ecs. (2.38 – 2.40) son conocidas como las *formas canónicas* X, Y y Z de la matriz de rotación. En muchos casos, una matriz rotación no puede obtenerse directamente con la información de orientación inicial y final de un cuerpo rígido, pero el movimiento total puede descomponerse en una secuencia de rotaciones simples tomando las formas canónicas anteriormente descritas. Una aplicación de las formas canónicas recae sobre la parametrización de rotaciones por medio de *ángulos de Euler*, consistentes en tres rotaciones sucesivas: ϕ, θ y ψ, sobre el eje de un marco de coordenadas. (Angeles, 2007)

2.2.7 Transformación de coordenadas con desplazamiento del origen

Cuando se está trabajando con coordenadas cuyo origen no coincide, la transformación de coordenadas se definen como sigue: sea $\mathbf{b}$ un vector de posición del origen $\mathcal{O}$ de $\mathcal{B}$, desde el origen de $\mathcal{A}$, como se muestra en la Figura 2.6. La transformación de coordenadas correspondiente de $\mathcal{A}$ a $\mathcal{B}$ de un vector de posición $\mathbf{p}$ de un punto P del espacio 3-dimensional Euclidiano entre dos marcos $\mathcal{A}$ y $\mathcal{B}$ se relacionan por

$$[\mathbf{p}]_{\mathcal{A}} = [\mathbf{b}]_{\mathcal{A}} + [\mathbf{Q}]_{\mathcal{A}}[\boldsymbol{\pi}]_{\mathcal{B}} \tag{2.41}$$

$$[\boldsymbol{\pi}]_{\mathcal{B}} = [\mathbf{Q}^T]_{\mathcal{B}}([-\mathbf{b}]_{\mathcal{A}} + [\mathbf{p}]_{\mathcal{A}}) \tag{2.42}$$

Con $\mathbf{b}$ definida como vector dirigido del origen de $\mathcal{A}$ hasta el origen de $\mathcal{B}$, y $\boldsymbol{\pi}$ el vector dirigido del origen de β a P como se observa en la Figura 2.6.

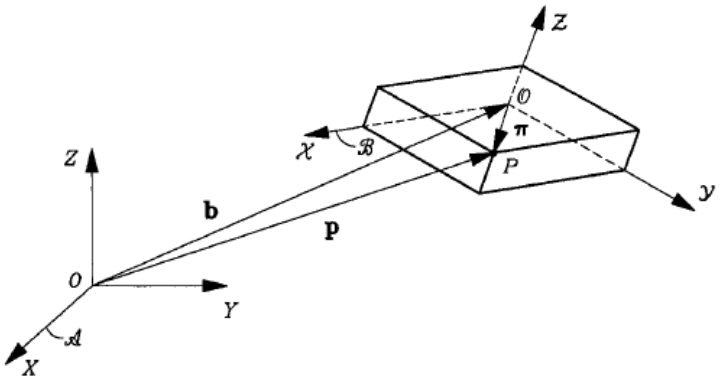


Figura 2.6 Marco de coordenadas con orígenes diferentes. (Angeles, 2007)

Prueba: de la Figura 2.6 se tiene que

$$\mathbf{p} = \mathbf{b} + \boldsymbol{\pi} \tag{2.43}$$

Si se expresa la ecuación anterior en el marco $\mathcal{A}$, se obtiene:

$$[\mathbf{p}]_{\mathcal{A}} = [\mathbf{b}]_{\mathcal{A}} + [\boldsymbol{\pi}]_{\mathcal{A}} \tag{2.44}$$

Donde se asume que $\boldsymbol{\pi}$ está disponible fácilmente en $\mathcal{B}$, y así, la ecuación anterior debe ser expresada como

$$[\mathbf{p}]_{\mathcal{A}} = [\mathbf{b}]_{\mathcal{A}} + [\mathbf{Q}]_{\mathcal{A}}[\boldsymbol{\pi}]_{\mathcal{B}}$$

Con lo cual se prueba que la ec. (2.41).

Ejemplo 2.1 *Si* $[\mathbf{b}]_{\mathcal{A}} = [-1,-1,-1]^T$*, además,* $\mathcal{A}$ *y* $\mathcal{B}$ *tienen las orientaciones relativas como se muestran en la* Figura 2.7*, encontrar el vector posición, en* $\mathcal{B}$*, de un punto* P*, de vector posición* $[\mathbf{p}]_{\mathcal{A}} = [1,1,1]^T$. (Angeles, 2007)

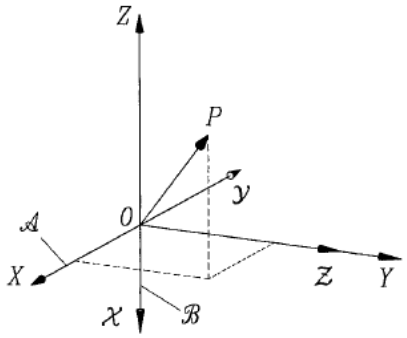


Figura 2.7 Marco de coordenadas $\mathcal{A}$ y $\mathcal{B}$ con un origen común. (Angeles, 2007)

Solución: Lo que obviamente se necesita es $[\boldsymbol{\pi}]_{\mathcal{B}}$, el cual está dado en la ec. (2.42). Así, primero se resuelve la suma dentro del paréntesis de esta ecuación, es decir,

$$[-\mathbf{b}]_{\mathcal{A}} + [\mathbf{p}]_{\mathcal{A}} = \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix}$$

Después, se necesita $[\mathbf{Q}^T]_{\mathcal{B}}$, el cual puede ser obtenido fácilmente de $[\mathbf{Q}]_{\mathcal{B}}$. Hasta ahora no se tiene esta matriz, pero se tiene $[\mathbf{Q}^T]_{\mathcal{A}}$, el cual es idéntico a $[\mathbf{Q}^T]_{\mathcal{B}}$, por consiguiente,

$$[\boldsymbol{\pi}]_{\mathcal{B}} = \begin{bmatrix} 0 & 0 & -1 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix} = \begin{bmatrix} -2 \\ -2 \\ 2 \end{bmatrix}$$

2.2.8 Coordenadas homogéneas

La transformación general de coordenadas, incluyendo el desplazamiento del origen, no es lineal, en general, puede ser fácilmente realizada por virtud de los términos *no-homogéneos* involucrados, es decir, el primer término del lado derecho de la ec. (2.41), el cual es independiente de $\mathbf{p}$. No obstante, tal transformación, puede ser representada en forma homogénea si las *coordenadas homogéneas* son introducidas. Estas son definidas como sigue: sea $[\mathbf{p}]_{\mathcal{M}}$ el conjunto de coordenadas de un punto *finito* P en referencia al marco $\mathcal{M}$. Lo que se quiere decir por un punto finito es aquel cuyas coordenadas todas son finitas. Así, se asume que el punto P no está en el *infinito*. Las coordenadas homogéneas de P en el espacio 4-dimensional $\{\mathbf{p}\}_{\mathcal{M}}$, son definidas como

$$\{\mathbf{p}\}_{\mathcal{M}} \equiv \begin{bmatrix} [\mathrm{P}]_{\mathcal{M}} \\ 1 \end{bmatrix} \tag{2.45}$$

La *transformación afín* de la ec. (2.41) ahora puede ser reescrita en forma de coordenadas homogéneas como

$$\{\mathbf{p}\}_{\mathcal{A}} = \{\mathbf{T}\}_{\mathcal{A}}[\boldsymbol{\pi}]_{\mathcal{B}} \tag{2.46}$$

Donde $\{\mathbf{T}\}_{\mathcal{A}}$ está definida como un espacio de 4x4, es decir,

$$\{\mathbf{T}\}_{\mathcal{A}} \equiv \begin{bmatrix} [\mathbf{Q}]_{\mathcal{A}} & [\mathbf{b}]_{\mathcal{A}} \\ [\mathbf{0}^T]_{\mathcal{A}} & 1 \end{bmatrix} \tag{2.47}$$

Además, se tiene que, las representaciones de llevar las coordenadas de $\{\mathrm{T}\}$ en el marco $\mathcal{B}$ a las coordenadas en el marco $\mathcal{A}$, en estos dos marcos, son idénticas:

$$\{\mathbf{T}\}_{\mathcal{A}} = \{\mathbf{T}\}_{\mathcal{B}} \tag{2.48}$$

La transformación inversa que se definió en la ec. (2.47) es derivada de las ecs. (2.41) y (2.42), es decir,

$$\{\mathbf{T}^{-1}\}_{\mathcal{B}} = \begin{bmatrix} [\mathbf{Q}^T]_{\mathcal{B}} & [-\mathbf{b}]_{\mathcal{B}} \\ [\mathbf{0}^T]_{\mathcal{B}} & 1 \end{bmatrix} \tag{2.49}$$

Además, las transformaciones homogéneas pueden ser concatenadas. Ciertamente, $\mathcal{F}_k$, para $k = \ i-1,\ i,\ i+1$, denota tres marcos de coordenadas con orígenes en O_k. También, sea $\mathbf{Q}_{i-1}$ la rotación que lleva a $\mathcal{F}_{i-1}$ a una orientación coincidiendo con la de $\mathcal{F}_i$. Si una definición similar para $\mathbf{Q}_i$ es adoptada, entonces $\mathbf{Q}_i$ denota la rotación que lleva a $\mathcal{F}_i$ a una orientación coincidiendo con la de $\mathcal{F}_{i+1}$. Primero, es considerado el caso en el que todos los tres orígenes coinciden. Claramente,

$$[\mathbf{p}]_i = [\mathbf{Q}_{i-1}^T]_{i-1}[\mathbf{p}]_{i-1} \tag{2.50}$$

$$[\mathbf{p}]_{i+1} = [\mathbf{Q}_i^T]_i[\mathbf{p}]_i = [\mathbf{Q}_i^T]_i[\mathbf{Q}_{i-1}^T]_{i-1}[\mathbf{p}]_{i-1} \tag{2.51}$$

La relación inversa de la ec. (2.51) es la siguiente:

$$[\mathbf{p}]_{i-1} = [\mathbf{Q}_{i-1}]_{i-1}[\mathbf{Q}_i]_i[\mathbf{p}]_{i+1} \tag{2.52}$$

Si ahora los orígenes no coinciden, sean $\mathbf{a}_{i-1}$ y $\mathbf{a}_i$ las denotaciones de los vectores $\overrightarrow{O_{i-1}O_i}$ y $\overrightarrow{O_iO_{i+1}}$, respectivamente. Las transformaciones de las coordenadas homogéneas $\{\mathbf{T}_{i-1}\}_{i-1}$ y $\{\mathbf{T}_i\}_i$ obviamente son

$$\{\mathbf{T}_{i-1}\}_{i-1} = \begin{bmatrix} [\mathbf{Q}_{i-1}]_{i-1} & [\mathbf{a}_{i-1}]_{i-1} \\ [\mathbf{0}^T]_{i-1} & 1 \end{bmatrix}, \quad \{\mathbf{T}_i\}_i = \begin{bmatrix} [\mathbf{Q}_i]_i & [\mathbf{a}_i]_i \\ [\mathbf{0}^T]_i & 1 \end{bmatrix} \tag{2.53}$$

Considerando que sus transformaciones inversas son

$$\left\{\mathbf{T}_{i-1}^{-1}\right\}_i = \begin{bmatrix} \left[\mathbf{Q}_{i-1}^T\right]_i & \left[\mathbf{Q}_{i-1}^T\right]_i[-\mathbf{a}_{i-1}]_{i-1} \\ [\mathbf{0}^T]_i & 1 \end{bmatrix} \tag{2.54}$$

$$\left\{\mathbf{T}_i^{-1}\right\}_{i+1} = \begin{bmatrix} \left[\mathbf{Q}_i^T\right]_{i+1} & \left[\mathbf{Q}_i^T\right]_{i+1}[-\mathbf{a}_i]_i \\ [\mathbf{0}^T]_{i+1} & 1 \end{bmatrix} \tag{2.55}$$

Por lo tanto, las transformaciones de coordenadas involucradas son

$$\{\mathbf{p}\}_{i-1} = \{\mathrm{T}_{i-1}\}_{i-1}\{\mathbf{p}\}_i \tag{2.56}$$

$$\{\mathbf{p}\}_{i-1} = \{\mathrm{T}_{i-1}\}_{i-1}\{\mathbf{T}_i\}_i\{\mathbf{p}\}_{i+1} \tag{2.57}$$

Las transformaciones inversas correspondientes son

$$\{\mathbf{p}\}_i = \left\{\mathrm{T}_{i-1}^{-1}\right\}_{i-1}\{\mathbf{p}\}_{i-1} \tag{2.58}$$

$$\{\mathbf{p}\}_{i+1} = \left\{\mathrm{T}_i^{-1}\right\}_i\{\mathbf{p}_i\}_i = \left\{\mathrm{T}_i^{-1}\right\}_i\left\{\mathrm{T}_{i-1}^{-1}\right\}_{i-1}\{\mathbf{p}\}_{i-1} \tag{2.59}$$

Ahora, si P se encuentra en el infinito, estas coordenadas homogéneas se pueden expresar en una forma más simple. Para este fin, se reescribe la ec. (2.45) en la forma

$$\{\mathbf{p}\}_{\mathcal{M}} \equiv \|\mathbf{p}\|\begin{bmatrix}[\mathrm{e}]_{\mathcal{M}} \\ 1/\|\mathbf{p}\|\end{bmatrix}$$

Y por lo tanto,

$$\lim_{\|\mathbf{p}\|\to\infty}\{\mathbf{p}\}_{\mathcal{M}} = \left(\lim_{\|\mathbf{p}\|\to\infty}\|\mathbf{p}\|\right)\left(\lim_{\|\mathbf{p}\|\to\infty}\begin{bmatrix}[\mathrm{e}]_{\mathcal{M}} \\ 1/\|\mathbf{p}\|\end{bmatrix}\right)$$

O

$$\lim_{\|\mathbf{p}\|\to\infty}\{\mathbf{p}\}_{\mathcal{M}} = \left(\lim_{\|\mathbf{p}\|\to\infty}\|\mathbf{p}\|\right)\begin{bmatrix}[\mathrm{e}]_{\mathcal{M}} \\ 0\end{bmatrix}$$

Ahora se definen las *coordenadas homogéneas de un punto P estando en el infinito* como apareció en la expresión anterior el espacio 4-dimensional, es decir,

$$\{\mathbf{p}_\infty\}_{\mathcal{M}} \equiv \begin{bmatrix}[\mathrm{e}]_{\mathcal{M}} \\ 0\end{bmatrix} \tag{2.60}$$

Lo cual quiere decir que es un punto en el infinito, en coordenadas homogéneas, tiene solo una dirección, dada por el vector unitario $\mathbf{e}$, pero una localización indefinida. Al trabajar con objetos dentro de la atmosfera de la Tierra, por ejemplo, las estrellas pueden ser tan supuestas en el infinito, y por lo tanto, su posición es completamente especificada simplemente por su longitud y su latitud, lo cual es suficiente para definir los cosenos de dirección de un vector unitario en coordenadas esféricas. (Angeles, 2007)

Por otra parte, puede suponerse que una matriz rotación está compuesta por tres columnas, es decir, cada una representando un vector unitario,

$$\mathbf{Q} = [\mathbf{e}_1 \quad \mathbf{e}_2 \quad \mathbf{e}_3]$$

Donde la triada $\{\mathbf{e}_k\}_1^3$ es orto-normal. Así se puede representar $\{\mathbf{T}\}_{\mathcal{A}}$ de la ec. (2.47) en la forma

$$\{\mathbf{T}\}_{\mathcal{A}} = \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 & \mathbf{b} \\ 0 & 0 & 0 & \mathbf{1} \end{bmatrix} \tag{2.61}$$

Por consiguiente, concluyendo que las columnas de matriz $\mathbf{T}$ de 4x4 representan las coordenadas homogéneas correspondientes a un conjunto de puntos, de las cuales, las primeras tres representan puntos en el infinito. (Angeles, 2007)

2.3 Otras herramientas matemáticas

2.3.1 La línea recta

Las propiedades fundamentales de la recta, de acuerdo a los axiomas de Euclides y tomando como referencia la Figura 2.8 son:

- Por dos puntos distintos pasa una y sólo una línea recta.
- Dos rectas distintas se cortan en un solo punto o son paralelas.

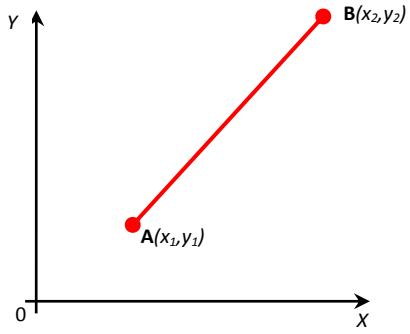


Figura 2.8 **Línea recta formada por los puntos A y B con coordenadas (x_1,y_1) y (x_2,y_2) respectivamente.**

La pendiente de una recta es el valor de la tangente de su ángulo de inclinación, es decir, la pendiente es igual a la tangente del ángulo que la línea recta forma con respecto al eje horizontal (por ejemplo, $\tan\alpha$) Además, la pendiente también puede calcularse si se tienen dos puntos cualesquiera que se encuentren sobre la línea recta de la manera siguiente:

$$m = \frac{y_2 - y_1}{x_2 - x_1} \tag{2.62}$$

Donde:

m representa la pendiente de la línea recta.

Las coordenadas del punto inicial y final de la línea respectivamente x_1, y_1 y x_2, y_2.

Para determinar la longitud de una línea recta mediante las coordenadas de sus dos puntos (inicial y final), suponiendo que P es el punto inicial y Q el punto final, con coordenadas x_1, y_1 y x_2, y_2 respectivamente, dicha distancia está representada por el segmento $\overline{\mathrm{PQ}}$ ó $d(\mathrm{P}, \mathrm{Q})$, la cual se calcula con la siguiente ecuación:

$$d(\mathrm{P}, \mathrm{Q}) = \left|\overline{\mathrm{PQ}}\right| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{2.63}$$

La ecuación de la recta puede determinarse conociendo un punto y la pendiente de ésta con la siguiente ecuación:

$$y - y_1 = m(x - x_1) \tag{2.64}$$

Podemos obtener la ecuación de la recta de varias maneras, dependiendo de los datos que sepamos de ella, y recíprocamente, si tenemos la ecuación de la recta, podemos escribirla en distintas formas y obtener de esas expresiones informaciones diversas acerca de la recta. Un caso importante es cuando conocemos la pendiente m y el punto P donde corta al eje Y; a su ordenada que usualmente se denota con la letra b, se le llama *ordenada al origen.* Tomamos el punto $\mathrm{P}(0, b)$ y la pendiente dada m; sustituimos en la ec. (2.64):

$$y = mx + b \tag{2.65}$$

La ecuación anterior se conoce como la forma *pendiente-ordenada al origen* de la ecuación de la recta. (Angeles, 2007)

Otra forma de encontrar la ecuación de la línea recta es conociendo dos puntos de ella, es decir, si conocemos las coordenadas de dos puntos de la recta, podemos encontrar su pendiente y tomando como punto fijo cualquiera de los dos que se conocen, se puede sustituir en la ec. (2.64) y obtener:

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) \tag{2.66}$$

Las ecuaciones anteriores sirven para determinar cualquier recta, excepto a las rectas verticales, ya que estas no tienen pendiente. Sin embargo, las ecuaciones para las rectas verticales son muy sencillas, ya que todos los puntos de ella tienen la misma primera coordenada. Así, la ecuación de la recta vertical que pasa por el punto (h, k) es $x = h$. (Angeles, 2007)

Una forma de la ecuación de la recta que cubre tanto a las rectas verticales como a las que no son es la *forma general* de la ecuación de la recta y se obtiene pasando todos los miembros de la ecuación a un miembro, de manera que éste quede igualado a cero, esta forma general queda como sigue:

$$Ax + By + C = 0 \tag{2.67}$$

2.3.2 El círculo

Un círculo es un lugar geométrico de los puntos que equidistan de un punto fijo llamado *centro.*

Muchos libros de geometría hacen la distinción entre los términos “circunferencia” y “círculo” para nombrar al borde y al interior de un disco, respectivamente. Algunos utilizan la palabra “circunferencia” como sinónimo de “perímetro” es decir, la longitud del borde del disco.

Para comprender mejor esta definición puede hacerse referencia a la Figura 2.9, donde se representa un círculo con coordenadas del centro en (*h,K*) y con un radio *r*; donde el radio *r* es una línea recta cualesquiera que tiene origen en las coordenadas del centro del circulo y termina en cualquier punto que se encuentre dentro del perímetro del círculo.

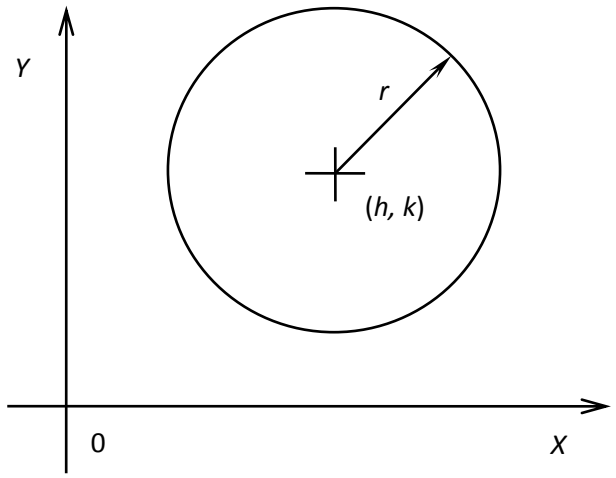


Figura 2.9 **Representación de un círculo con centro (*h,k*) de radio *r*.**

La tendencia moderna es utilizar la palabra "círculo" para referirse ya sea al borde solamente o al borde junto con el interior de la figura. De la misma manera se procede con los polígonos. (Oteyza, *et al*,2001)

En general, si se desea encontrar la ecuación del círculo que pasa por el punto $\mathrm{P}(x, y)$ cuya distancia al origen $\mathrm{O}(0,0)$ es igual a r, es decir, la ecuación de un círculo con centro en el origen, siendo r cualquier número no negativo, entonces observamos que dichos puntos deben satisfacer $d(\mathrm{P}, \mathrm{O}) = r$; sustituyendo las coordenadas de P y O en la fórmula de la distancia entre dos puntos, es decir, en la ec. (2.63), se obtiene $\sqrt{(x-0)^2 + (y-0)^2} = r$. Elevando al cuadrado ambos miembros de la ecuación, se llega finalmente a:

$$x^2 + y^2 = r^2 \tag{2.68}$$

Que es la ecuación del círculo de radio r con centro en el origen.

Para encontrar la ecuación de un círculo cuyo centro sea un punto $\mathrm{C}(a, b)$ distinto del origen, se utiliza la traslación de ejes. (Oteyza, *et al*,2001)

Se construye un nuevo sistema de coordenadas $X'Y'$ con centro en C. Con respecto a este nuevo sistema de coordenadas, el círculo con centro en C y radio r tiene por ecuación $(x')^2 + (y')^2 = r^2$. Para encontrar la ecuación del círculo con respecto al sistema de coordenadas original XY, se hace la sustitución $x' = x - a,\ y' = y - b$ y se obtiene

$$(x - a)^2 + (y - b)^2 = r^2 \tag{2.69}$$

Que es la *forma estándar* o *canónica* del círculo de radio r y con centro en $C(a, b)$.

Si se desarrollan los términos que están al cuadrado y se pasan todos los términos a un solo lado de la ec. (2.69) se obtiene $x^2 + y^2 + (-2a)x + (-2b)y + a^2 + b^2 - r^2 = 0$ y si $\mathrm{D} = -2a$, $\mathrm{E} = -2b$ y $\mathrm{F} = a^2 + b^2 - r^2$, se llega a la ecuación del tipo

$$x^2 + y^2 + \mathrm{D}x + \mathrm{E}y + \mathrm{F} = 0 \tag{2.70}$$

Que se llama la *forma general* de la ecuación del círculo. (Oteyza, *et al*,2001)

2.3.3 Intersección de un círculo con una recta

Para encontrar la solución de la intersección de un círculo con una recta, es decir, encontrar los puntos donde la línea recta toca al perímetro del círculo o viceversa, previamente de alguna manera debe conocerse la ecuación de la recta, ya sea con un punto y su pendiente o conociendo dos puntos de ella. Además, para esta solución se debe de contar con los datos suficientes para encontrar la ecuación del círculo. Una vez obtenidas estas dos ecuaciones, se deben de seguir los siguientes pasos:

- Debe de despejarse y de la ecuación de la recta, es decir, dejándola en función de x.
- Una vez despejada y de la ecuación de la recta, ésta se sustituye en la ecuación del círculo y se despeja x.

- Con el valor de x, puede encontrarse el valor de y sustituyendo a x en la ecuación de la recta. Estos dos valores de x e y son las coordenadas del punto donde la recta y el círculo dados se intersectan. (Oteyza, *et al*,2001)

Ahora bien, dados un círculo y una recta, puede suceder que:

- No se corten.
- La recta corte al círculo en un punto.
- La recta corte al círculo en dos puntos.

Estos tres diferentes casos se muestran gráficamente en la Figura 2.10, siendo representado el centro del circulo por las coordenadas *(h,k)* y las coordenadas del punto de la recta que corta al círculo son *(x,y)* para la recta que corta al círculo en un solo punto; para la recta que corta al círculo en dos puntos se tiene que las coordenadas son (x_1, y_1) y (x_2, y_2).

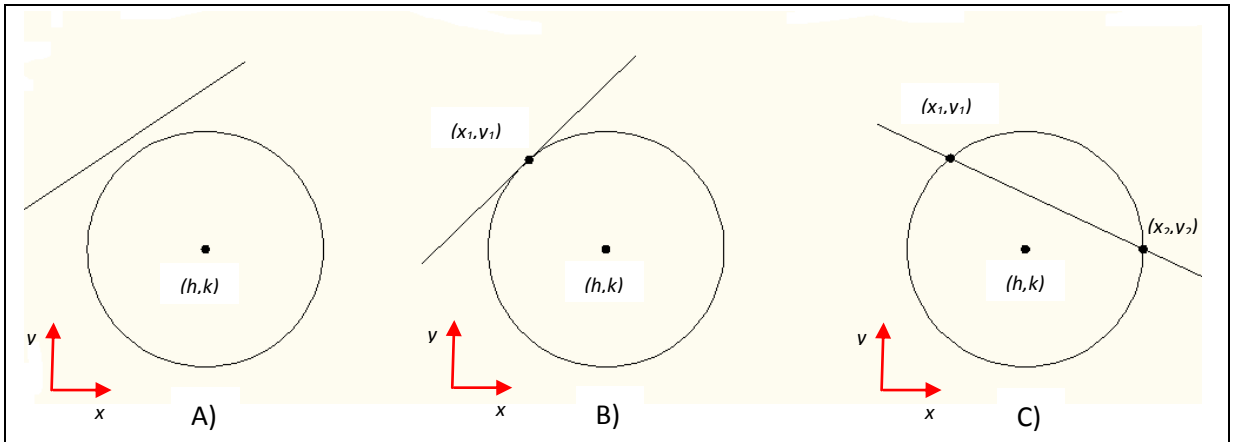


Figura 2.10 **A) La recta no corta al círculo en ningún punto. B) La recta corta al círculo en un punto. C) La recta corta al círculo en dos puntos.**

Estas situaciones en términos algebraicos se traducen de la siguiente manera: Si se consideran las ecuaciones de la recta y del círculo y se resuelven simultáneamente, resulta que:

- No hay solución (no se cortan).
- Hay una solución (se cortan en un solo punto).

- Hay dos soluciones (se cortan en dos puntos). (Oteyza, *et al*,2001)

2.3.4 El círculo que pasa por tres puntos

Un problema interesante es saber cuál es la mínima cantidad de puntos que determinan una curva. Por ejemplo, se sabe que por un punto pueden pasar una infinidad de rectas; en cambio, dados dos puntos, sólo hay una recta que pasa por ambos. En el caso del círculo, por uno y dos puntos pueden pasar una infinidad de círculos. En cambio, por tres puntos no alineados sólo puede pasar un círculo, ya que estos tres puntos determinan un triángulo y se sabe que el centro del *círculo circunscrito* a un triángulo es el punto donde se cortan las *mediatrices* de él; este punto se llama *circuncentro,* tal como se muestra en la Figura 2.11 en donde los puntos A, B y C forman el triangulo por el cual pasara el círculo, siendo las líneas en color rojo las mediatrices de los segmentos $\overline{\mathrm{AB}}$ y $\overline{\mathrm{AC}}$, cuya intersección es el centro del círculo. (Oteyza, *et al*,2001)

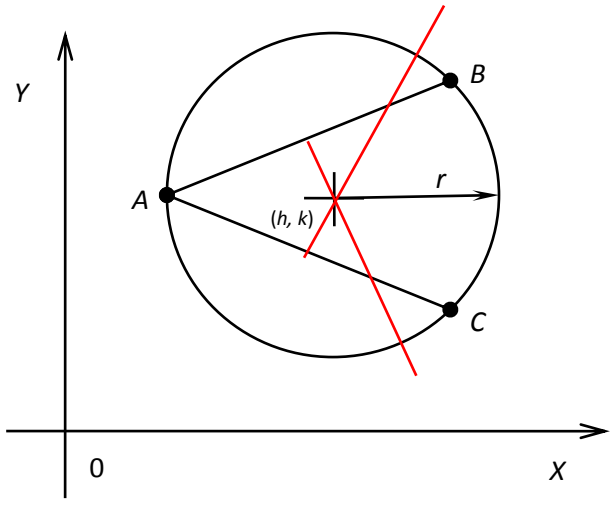


Figura 2.11 Imagen que muestra un circulo a partir de tres puntos no alineados (A, B y C).

Para encontrar la ecuación del círculo que se ajuste a tres puntos, de entrada deben conocerse las coordenadas de estos tres puntos. Para hacer más sencilla esta solución, se explicara mediante un ejemplo. (Oteyza, *et al*,2001)

Como ya se mencionó en el párrafo anterior, se deben conocer las coordenadas de los tres puntos por donde pasará la circunferencia, así, estos tres puntos son $\mathrm{P}(1,-1)$, $\mathrm{Q}(5,6)$ y $\mathrm{R}(-2,1)$ para explicar este ejemplo.

El centro $\mathrm{C}(x,y)$ del círculo que se busca equidista de los tres puntos dados. Es decir, $d(\mathrm{P},\mathrm{C}) = d(\mathrm{Q},\mathrm{C}) = d(\mathrm{R},\mathrm{C})$.

El siguiente paso es resolver la primera de las igualdades:

$$d(\mathrm{P},\mathrm{C}) = d(\mathrm{Q},\mathrm{C})$$

$$\sqrt{(x-1)^2+(y-(-1))^2} = \sqrt{(x-5)^2+(y-6)^2}$$

Elevando al cuadrado ambos miembros de la igualdad anterior y simplificando, se tiene:

$$8x + 14y - 59 = 0$$

Ahora, se resuelve la siguiente igualdad:

$$d(\mathrm{Q},\mathrm{C}) = d(\mathrm{R},\mathrm{C})$$

$$\sqrt{(x-5)^2+(y-6)^2} = \sqrt{(x-(-2))^2+(y-1)^2}$$

Elevando al cuadrado ambos miembros de la igualdad anterior y simplificando, se tiene:

$$-14x - 10y + 56 = 0$$

Como siguiente punto, se deben resolver simultáneamente las dos ecuaciones obtenidas; en este caso la última ecuación se multiplica por $\frac{1}{2}$ para simplificar:

$$8x + 14y - 59 = 0$$

$$-7x - 5y + 28 = 0$$

Para este ejemplo, se multiplica la primera ecuación por 7, la segunda por ocho y se suman, quedando como resultado de estas operaciones la siguiente ecuación:

$$58y - 189 = 0$$

De donde $y = \frac{189}{58}$.

Posteriormente, se sustituye el valor de y en la primera ecuación que se obtuvo resolviendo la primera igualdad para calcular el valor de x, el cual, resolviendo es $x = \frac{97}{58}$. Así, el centro del círculo es $\mathrm{C}\left(\frac{97}{58}, \frac{189}{58}\right)$.

Para encontrar el radio del círculo, calculamos la distancia de C a cualquiera de los puntos dados; por ejemplo,

$$d(\mathrm{R}, \mathrm{C}) = \sqrt{\left(-2 - \frac{97}{58}\right)^2 + \left(1 - \frac{189}{58}\right)^2} = \sqrt{\left(-\frac{213}{58}\right)^2 + \left(-\frac{131}{58}\right)^2} = \frac{13}{58}\sqrt{370}$$

Por lo tanto, la ecuación de círculo es:

$$\left(x - \frac{97}{58}\right)^2 + \left(y - \frac{189}{58}\right)^2 = \left(\frac{13}{58}\sqrt{370}\right)^2$$

De esta manera, se puede realizar una serie de pasos a seguir para encontrar la ecuación de un círculo que se ajuste a tres puntos dados, estos pasos pueden ser:

- Conocer las coordenadas de los tres puntos por donde el círculo pasará.
- Las coordenadas del centro del círculo, como no se conocen se representan como $\mathrm{C}(x, y)$.

- Ya que el centro equidista a cualquiera de los tres puntos del primer paso, se debe formar una igualdad con la distancia que hay de cada punto al centro (un término por cada distancia de cada punto al centro). Estas distancias se calculan con la ec. (2.63). (Oteyza, *et al*,2001)
- Se toma la primera parte de la igualdad (primeros dos términos) y se resuelve, dando como resultado una ecuación lineal con dos incógnitas (x, y), que son las coordenadas del centro.
- Ahora, se toma la segunda parte de la igualdad (segundo y tercer término) y también se resuelve, quedando como resultado otra ecuación lineal con dos incógnitas (x, y), que son las coordenadas del centro.
- Ahora, con las dos ecuaciones obtenidas en los dos pasos anteriores, se genera un sistema de dos ecuaciones con dos incógnitas. Estas ecuaciones se resuelven simultáneamente para encontrar una de las incógnitas, la cual, generalmente es y.
- Una vez obtenido el valor de la primera incógnita, este se sustituye en la primera ecuación obtenida de resolver la primera parte de la igualdad para obtener el valor de la segunda incógnita faltante que generalmente es x.
- Con esto, las coordenadas del centro del círculo ahora son conocidas.
- Posteriormente, puede conocerse el radio del círculo calculando la distancia que hay del centro a cualquiera de los tres puntos conocidos.
- Por último, las coordenadas del centro del círculo y el valor de su radio se sustituyen en la ec. (2.69) quedando así, la ecuación del círculo que pasa por los tres puntos dados. (Oteyza, *et al*,2001)

2.4 CNC's

2.4.1 Máquinas-herramienta

Las máquinas-herramienta por lo general son máquinas de potencia para corte o conformación de metales que se utilizan para dar forma a dichos metales, esto mediante:

- La eliminación de virutas.
- Prensado, estirado o corte.
- Procesos de maquinado eléctrico o controlados.

Cualquier máquina-herramienta por lo general es capaz de:

- Sujetar y apoyar la pieza de trabajo.
- Sujetar y apoyar una herramienta de corte.
- Impartir un movimiento adecuado (rotatorio o reciprocante) a la herramienta de corte o a la pieza de trabajo.
- Avanzar la herramienta de corte o la pieza de trabajo de forma que se logre la acción de corte y la precisión requerida.

Algunos ejemplos de las máquinas-herramienta pueden ser los taladros, tornos, sierras, fresadoras, esmeriladoras y rectificadoras, entre otras. Estas máquinas-herramienta pueden ser operadas manualmente o controladas por computadora, es decir, estas últimas son máquinas de control numérico por computadora. (Krar y Check, 2005)

2.4.2 Control Numérico por Computadora (CNC)

El control numérico (NC, *Numerical Control*) puede definirse como un método de controlar con precisión la operación de una máquina mediante una serie de instrucciones codificadas, formadas por números, letras del alfabeto y símbolos que la unidad de control de la maquina puede comprender. Estas instrucciones se convierten en pulsos eléctricos que los motores y controles de la maquina siguen para llevar a cabo las operaciones de maquinado sobre una pieza de trabajo. Los números, letras y símbolos son instrucciones codificadas que se refieren a distancias, posiciones, funciones o movimientos específicos que la máquina-herramienta puede comprender al maquinar una pieza. Los dispositivos de medición y de registro incorporados en las máquinas-herramienta de control numérico por computadora aseguran que la pieza que se está manufacturando sea exacta, además, estas máquinas-herramienta minimizan el error humano.

El control numérico por computadora y la PC han aportado cambios significativos a la industria metalmecánica. Nuevas máquinas-herramienta, en combinación con CNC, le permiten a la industria producir de manera consistente componentes y piezas con precisiones imposibles de imaginar hace solo unos cuantos años. Si el programa CNC ha sido apropiadamente preparado, y la máquina ha sido puesta a punto correctamente, se puede producir la misma pieza con el mismo grado de precisión cualquier cantidad de veces. Los comandos de operación que controlan la máquina-herramienta son ejecutados automáticamente con una velocidad, eficiencia, precisión y capacidad de repetición sombrosas.

La aceptación mundial de las máquinas CNC ha sido el resultado de su precisión, confiabilidad, capacidad de repetición y productividad como se muestra en la Figura 2.12:

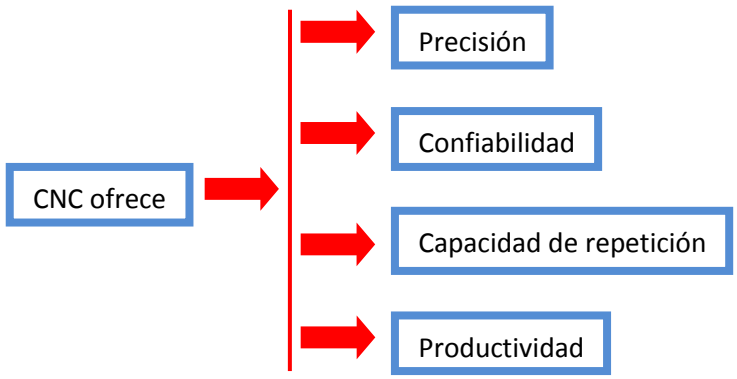


Figura 2.12 Rendimiento del CNC.

- **Precisión.** Un mecánico diestro es capaz de trabajar con tolerancias estrechas, como por ejemplo ±0.001 pulg (0.025 mm), o incluso menos en la mayor parte de las máquinas-herramienta, pero esta persona no puede ser capaz de trabajar con esa precisión todo el tiempo. Algún error humano significará desperdicio para la industria. Las máquinas-herramienta modernas CNC son capaces consistentemente de producir piezas que tienen una precisión con tolerancias de hasta 0.0001 a 0.0002 pulg (0.002 a 0.005 mm) durante el periodo de tiempo que la máquina herramienta haya sido programada con el mismo margen de precisión durante dicho tiempo, a esto se le conoce como precisión.

- **Confiabilidad.** Las máquinas herramienta CNC pueden maquinar a estrechas tolerancias, son capaces de repetir lo anterior una y otra vez. Las mejorías en las correderas, cojinetes, tornillos de bolas y mesas de dichas máquinas, ayudan a que las máquinas sean más precisas y robustas, así, es posible la producción de manera consistente de piezas precisas.
- **Capacidad de repetición.** La capacidad de repetición y la confiabilidad son muy difíciles de separar porque muchas de las mismas variables afectan a ambas. La capacidad de repetición de una máquina-herramienta involucra la comparación de cada una de las piezas producidas en dicha máquina en lo que se refiera a tamaño y precisión. La capacidad de repetición de una máquina CNC debe ser por lo menos la mitad de la tolerancia más pequeña de la pieza.
- **Productividad.** El empleo de las máquinas CNC y la automatización de las industrias para la manufactura genera productos de una muy alta calidad, y al mismo tiempo mejora el rendimiento sobre el capital invertido, reduce los costos de manufactura y de mano de obra. (Krar y Check, 2005)

2.4.3 Coordenadas cartesianas

Los movimientos de la máquina-herramienta que se utilizan para la producción de un producto son de dos tipos básicos: punto a punto (movimientos rectilíneos) y trayectoria continua (movimientos de contorneado), los cuales serán descritos más adelante.

El sistema de coordenadas cartesiano o rectangular permite que cualquier punto específico de un trabajo sea descrito en términos matemáticos en relación con cualquier otro punto a lo largo de tres ejes perpendiculares. Esto se adecua perfectamente a las máquinas-herramienta ya que su construcción se basa por lo general en tres ejes de movimiento (*X,Y,Z*) más un eje opcional de rotación.

El CNC se utiliza en todo tipo de máquinas-herramienta, desde la más simple hasta la más compleja. Las máquinas-herramienta más comunes son el centro de torneado y el centro de maquinado (máquina fresadora).

Los centros de torneado CNC son capaces de generar una mayor precisión y un ritmo más elevado de producción de lo que es posible en un torno convencional. El centro básico de torneado opera sobre dos ejes: el eje *X* controla el movimiento transversal de la torre portaherramientas mientras que el eje *Z* controla el movimiento longitudinal (acercándose o alejándose del cabezal) de la torre portaherramientas.

Los centros de maquinado permiten que se lleven a cabo mas operaciones sobre una pieza en una sola puesta a punto en vez de pasar la pieza de una máquina a otra para varias operaciones. Estas maquinas incrementan de manera importante la productividad porque el tiempo que antes se utilizaba para mover la pieza de una máquina a otra aquí se elimina. Los dos tipos principales de centros de maquinado son los modelos de husillo horizontal y de husillo vertical en donde este último es el modelo utilizado para el desarrollo de este proyecto. El centro de maquinado vertical opera sobre tres ejes: El eje *X* controla el movimiento hacia la izquierda o hacia la derecha de la mesa; el eje *Y* controla el movimiento de la mesa alejándose o hacia la columna de la máquina mientras que el eje *Z* controla el movimiento vertical, es decir, mueve el husillo hacia arriba o hacia abajo.

Los sistemas CNC se apoyan en el uso de coordenadas rectangulares porque el programador puede localizar con precisión cada punto de un trabajo.

Cuando están localizados los puntos de una pieza, se utilizan dos líneas rectas que se cruzan, una vertical y la otra horizontal. Estas líneas deben estar a 90° entre sí, el punto donde se cruzan se llama *origen,* o el *punto cero* como se muestra en la Figura 2.13.

Los planos coordenados en tres dimensiones se muestran en la Figura 2.14, los cuales se utilizan en el CNC.

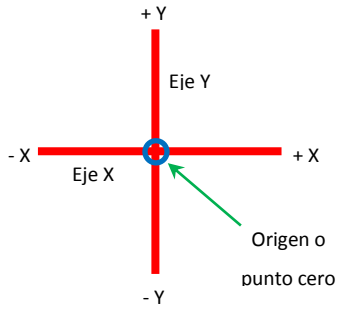


Figura 2.13 Origen o punto cero.

Como puede observarse en la Figura 2.14, los planos *X* e *Y* (ejes) son horizontales y representan los movimientos horizontales de la mesa de la máquina mientras que el plano o eje *Z* representa el movimiento vertical de la herramienta.

Los signos más (+) y menos (-) indican la dirección de movimiento desde el punto cero (origen) a lo largo del eje y los cuatro cuadrantes que se forman cuando se cruzan los ejes *X* y *Y*, están numerados en dirección contraria a las manecillas del reloj como se muestra en la Figura 2.15.

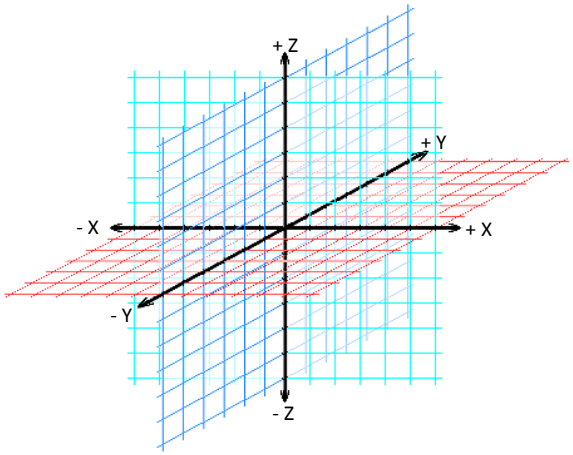


Figura 2.14 Planos coordenados tridimensionales.

Observe que de la Figura 2.15 todas las posiciones localizadas en el cuadrante 1 son de *X* positiva (+*X*) y de *Y* positiva (+*Y*); en el segundo cuadrante todas las posiciones tienen *X* negativa (-*X*) y *Y* positiva (+*Y*); en el tercer cuadrante todas las posiciones tienen *X* negativa (-*X*) y *Y* negativa (-*Y*), por último, en el cuarto cuadrante todas las posiciones son *X* positiva (+*X*) y *Y* negativa (-*Y*).

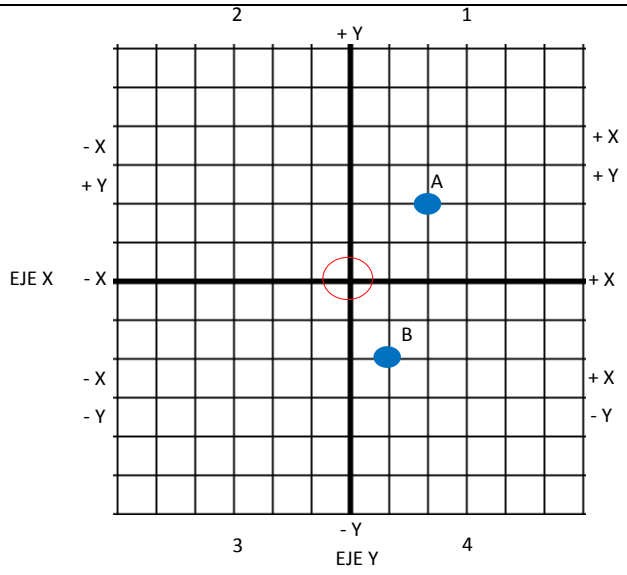


Figura 2.15 Cuadrantes formados por los planos de los ejes *X* y *Y*.

En la figura 2.15, el punto A esta a dos unidades a la derecha del eje *Y* y dos unidades por encima del eje *X*. Suponiendo que cada unidad es igual a 1 pulg. La localización del punto A es X + 2.000 y Y + 2.000. Para el punto B, la localización es X + 1.000 y Y – 2.000. En programación CNC, no es necesario indicar los valores positivos (+) ya que se suponen. Sin embargo, es necesario indicar los valores negativos (-), por ejemplo, tanto las localizaciones de A como de B se indican como sigue:

A	X2.000	Y2.000
B	X1.000	Y-2.000

En vista que el CNC depende considerablemente del sistema de coordenadas rectangulares, es fundamental seguir ciertas reglas:

- De ser posible se deben utilizar puntos de referencia sobre la pieza misma. Esto facilita la verificación de la precisión posterior de la pieza.
- Utilizar coordenadas cartesianas especificando planos *X, Y, Z* para definir todas las superficies de la pieza.
- Establecer planos de referencia a lo largo de las superficies de la pieza que sean paralelas a los ejes de la máquina.
- Establecer las tolerancias permisibles en la etapa de diseño.
- Describir la pieza de manera que sea fácil de determinar y de programar la trayectoria de la herramienta de corte.
- Dimensionar la pieza de manera que resulte fácil de determinar su forma sin cálculos ni estimaciones. (Krar y Check, 2005)

2.4.4 Sistemas de programación CNC

En el CNC se utilizan dos tipos de modos de programación, el sistema incremental y el sistema absoluto. Ambos sistemas cuentan con aplicaciones en la programación CNC, y ningún sistema es el más adecuado en toda ocasión. La mayor parte de los controles de las máquinas-herramienta son capaces de manejar la programación tanto incremental como absoluta mediante la modificación del código de programación.

El sistema incremental, las posiciones o dimensiones están dadas a partir del punto actual. Las dimensiones incrementales en un plano de trabajo se muestran en la Figura 2.16. Como puede observarse, las dimensiones de cada barreno están dadas a partir del barreno anterior. Una desventaja de la programación o posicionamiento incremental es que, si se ha cometido un error en cualquiera de las posiciones, este error es automáticamente arrastrado a las localizaciones siguientes.

Los códigos de comando que le indican a la máquina cómo mover la mesa y el husillo, para una máquina fresadora vertical son los siguientes:

- Un comando “más X” (+X) hace que la herramienta de corte se posicione a la derecha del último punto.

- Un comando “menos X” (-X) hace que la herramienta de corte se posicione a la izquierda del último punto.
- Un comando “más Y” (+Y) hace que la herramienta de corte se posicione hacia la columna.
- Un comando “menos Y” (-Y) hace que la herramienta de corte se aleje de la columna.
- Un comando “más Z” (+Z) hace que la herramienta de corte o el usillo se mueva hacia arriba o se aleje de la pieza de trabajo.
- Un comando “menos Z” (-Z) hace que la herramienta de corte se mueva hacia abajo o hacia adentro de la pieza de trabajo.

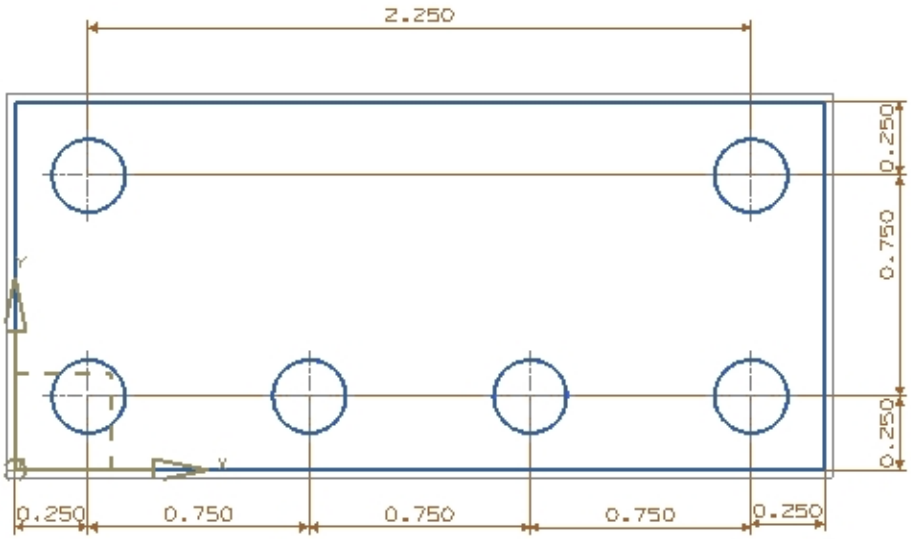


Figura 2.16 Sistema incremental mostrando las dimensiones de la pieza de trabajo.

En el sistema absoluto, todas las dimensiones o posiciones están dadas a partir de un punto de referencia sobre la pieza de trabajo o sobre la máquina. En la Figura 2.17 se utilizó la misma pieza que en la Figura 2.16, pero se dan todas las dimensiones a partir del cero o punto de referencia, que en este caso es la esquina inferior izquierda de la pieza. Por lo tanto, en el sistema absoluto de dimensionar o de programar, un error en cualquier dimensión sigue siendo un error, pero éste no es arrastrado a ninguna otra localización.

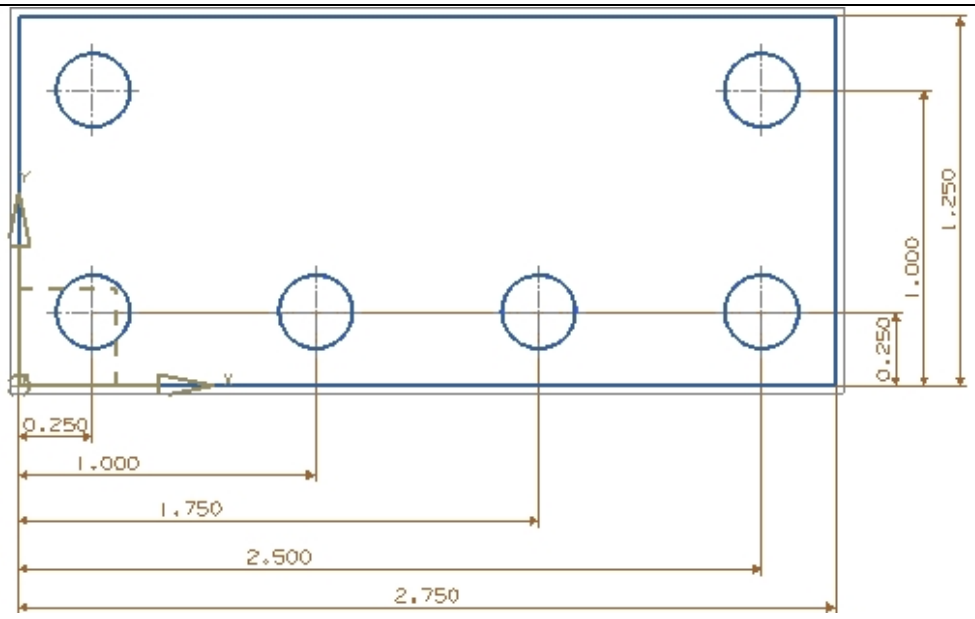


Figura 2.17 Sistema absoluto mostrando las dimensiones de la pieza de trabajo.

- Un comando “más X” (+X) hace que la herramienta de corte se posicione a la derecha del cero o punto de origen.
- Un comando “menos X” (-X) hace que la herramienta de corte se posicione a la izquierda del cero o punto de origen.
- Un comando “más Y” (+Y) hace que la herramienta de corte se posicione hacia la columna (por encima del cero o punto de origen).
- Un comando “menos Y” (-Y) hace que la herramienta de corte se aleje de la columna (por debajo del cero o punto de origen).
- Un comando “más Z” (+Z) hace que la herramienta de corte quede por encima del programa Z0 (por lo general la superficie superior de la pieza).
- Un comando “manos Z” (-Z) hace que la herramienta de corte se mueva por debajo del programa Z0. (Krar y Check, 2005)

2.4.5 Sistemas de posicionamiento CNC

Toda máquina CNC tiene ejes controlables deslizantes y/o giratorios. A fin de controlar estos ejes, se utilizan letras (llamadas direcciones) para identificar cada dirección de movimiento de la mesa o del husillo. En combinación con un número para formar una palabra, establece la distancia que se mueve el eje. Estas palabras son necesarias para que el programador pase la información respecto a la letra a las personas responsables de la puesta a punto y de la operación de la máquina CNC. Los constructores de máquinas-herramienta siguen estándares establecidos por la Asociación de Industrias de Electrónica (EIA, *Electronics Industries Association*), misma que asigna el sistema de codificación para los ejes de máquinas CNC. Los ejes principales son X, Y y Z, que se aplican a la mayor parte de las máquinas-herramienta con algunas excepciones. La norma EIA dice que el movimiento del eje horizontal más largo, que es paralelo a la mesa de trabajo es el eje X. El movimiento a lo largo del husillo es el eje Z y se le asigna el eje Y al movimiento perpendicular (en ángulo recto) tanto a los ejes X y Z.

Además de los ejes principales, existen ejes secundarios paralelos a los ejes X, Y y Z. Las direcciones (letras) A, B y C se refieren a ejes de movimiento rotativo alrededor de los ejes principales. I, J y K son letras también utilizadas para ejes rotativos en algunas máquinas cuando se utiliza interpolación circular para la programación de círculos o arcos parciales, en tanto que en otras máquinas, una letra R representa el radio de un círculo. Algunos centros de mandriles y de torneado también utilizan las letras U y W para movimientos incrementales paralelos a los ejes principales X y Z.

La programación CNC se clasifica en dos categorías diferentes, punto por punto y trayectoria continua, que pueden ser manejadas por la mayor parte de las unidades de control. Es necesario tener conocimiento de ambos métodos de programación para comprender qué aplicación tiene cada una de ellas en CNC.

El posicionamiento punto a punto está formando por cualquier cantidad de puntos programados unidos entre sí por líneas rectas. Este método se utiliza para localizar con precisión el husillo, o la pieza montada sobre la mesa de la máquina en una o más localizaciones especificas a fin de llevar a cabo operaciones como taladrado, rimado, mandrinado, machueleado y punzonado como se muestra en la Figura 2.18. El posicionamiento punto a punto es el proceso de

pasar de una posición o localización de coordenadas (X,Y) a otra, en la ejecución de la operación de maquinado es el retiro de la herramienta de trabajo y pasar a la siguiente localización hasta que todas las operaciones programadas hayan sido terminadas.

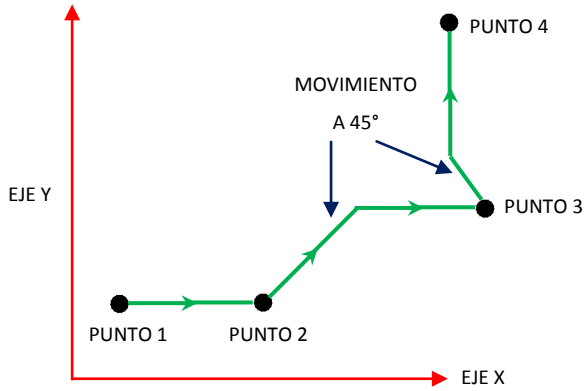


Figura 2.18 **Trayectoria seguida por el posicionamiento punto a punto entre varios puntos programados.**

Los taladros o máquinas de punto a punto, son idealmente adecuadas para el posicionamiento de la máquina-herramienta a una localización o punto exacto, la ejecución de la operación de maquinado (taladrar una perforación) y después pasar a la siguiente localización (donde se podría taladrar otra perforación). Siempre que esté identificado cada punto o localización de perforación dentro del programa, esta operación puede ser repetida tantas veces como se requiera.

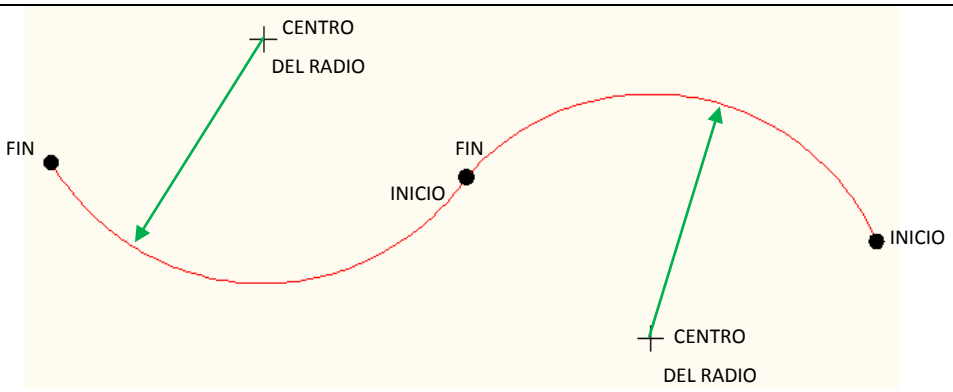


Figura 2.19 **Forma compleja en dos ejes utilizando interpolación circular.**

El maquinado en trayectoria continua o perfilada involucra trabajo producido en un torno o en una fresadora donde la herramienta de corte está por lo general en contacto con la pieza conforme se traslada de un punto programado al siguiente. El posicionamiento de trayectoria continua es la capacidad de controlar los movimientos de dos o más ejes de la máquina de manera simultánea, a fin de mantener una relación constante entre el cortador y la pieza. La información en el programa CNC debe posicionar con precisión la herramienta de corte desde un punto al siguiente y seguir una trayectoria precisa predefinida a la velocidad de avance programada a fin de producir la forma o perfil requerido como se muestra en la Figura 2.19. (Krar y Check, 2005)

2.4.6 Sistemas de control CNC

Los dos sistemas de control utilizados para las máquinas de control numérico por computadora son el de lazo abierto y el de lazo cerrado. La mayor parte de las máquinas-herramienta manufacturadas contienen un sistema de lazo cerrado porque es muy preciso y como resultado se puede producir un trabajo de mejor calidad.

En el sistema de lazo abierto, los datos de entrada son alimentados en la Unidad de Control de la Máquina (MCU, *Machine Control Unit*). Esta información decodificada en forma de programa, es puesto en orden hasta que el operador inicia el ciclo de maquinado de la máquina CNC. Los comandos del programa son convertidos de manera automática en pulsos o señales eléctricas que son enviados al MCU para energizar las unidades de servo control. Las unidades de servo control dirigen a los servomotores según la información suministrada por los datos de entrada del programa. La distancia que cada servomotor moverá al tornillo principal de la máquina dependerá del número de pulsos eléctricos que reciba de la unidad de servo control.

Este tipo de sistema es razonablemente sencillo; sin embargo, dado que no hay forma de verificar para determinar si el servo ha llevado a cabo su función correctamente, por lo general no es utilizado donde se requiera de una precisión superior a 0.001 pulg (0.025 mm).

El sistema de lazo cerrado (que es con el que cuenta la centro de maquinado CNC con la cual se llevará a cabo el desarrollo y las pruebas de este proyecto) es similar al sistema de lazo

abierto, con la excepción que se ha agregado una unidad de retroalimentación al circuito eléctrico como se observa en la Figura 2.20.

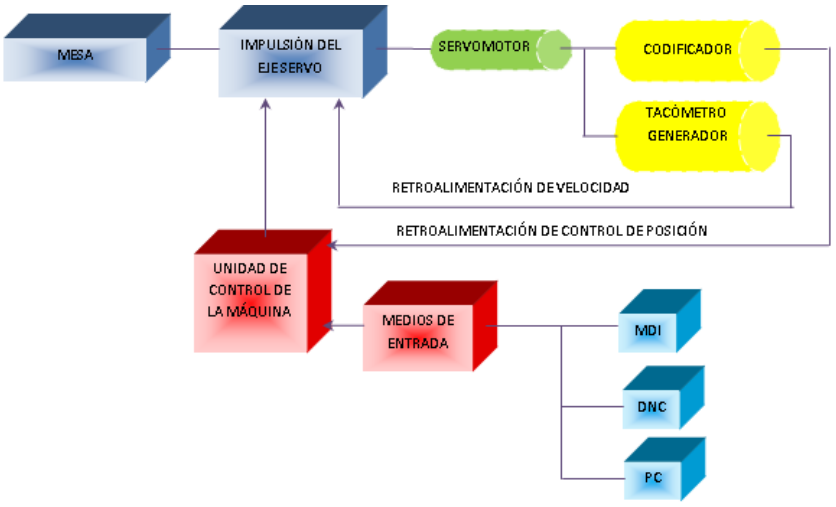


Figura 2.20 Sistema de lazo cerrado.

Esta unidad de retroalimentación, en forma de un *resolver* giratorio o codificador, puede estar acoplada en la parte trasera del eje del servomotor o en el extremo opuesto del tornillo principal. Se utiliza para un control absoluto de la posición y/o para retroalimentación de la velocidad. Otro tipo de sistema de retroalimentación es el codificador lineal, el cual consiste en una escala montada en una parte estacionaria de la máquina, como la rodilla de la máquina fresadora. El codificador lineal utiliza una cabeza o corredera lectora montada en la parte móvil de la máquina, como la mesa de la máquina. Ambas partes están conectadas magnéticamente u ópticamente, operando de igual forma a un *resolver* giratorio. Independientemente del sistema utilizado, la unidad de retroalimentación compara la cantidad que la mesa de la máquina ha sido movida por el servomotor con la señal enviada por la unidad de control. La unidad de control da instrucciones al servomotor para que efectúe los ajustes que resulten necesarios hasta que tanto la señal de la unidad de control como la señal de la unidad servo son iguales. En el sistema de lazo cerrado, se requieren de 10000 pulsos eléctricos para mover la corredera de la máquina 1 pulg (25.400 mm). Por lo tanto, en este tipo de sistema, un pulso originará un movimiento de 0.0001 pulg (0.002 mm) de la corredera de la máquina. Los sistemas de control numérico por

computadora de lazo cerrado son muy precisos porque se registra la precisión de la señal de comando y existe una compensación automática en caso de error. (Krar y Check, 2005)

2.4.7 Interpolación en el CNC

La interpolación, es decir, la generación de puntos de datos entre posiciones de coordenadas dadas de los ejes, es necesaria para cualquier tipo de programación. Dentro de la unidad de control de la máquina, un dispositivo conocido como un interpolador hace que los impulsores se muevan simultáneamente desde el principio del comando hasta su terminación. En las aplicaciones de programación CNC se utiliza con mayor frecuencia la interpolación lineal y la interpolación circular, sin embargo, existen también las interpolaciones helicoidales, parabólicas y cúbicas.

La interpolación lineal se utiliza para el maquinado en línea recta entre dos puntos; la interpolación circular se utiliza para círculos y arcos. La interpolación helicoidal, utilizada para roscas y formas helicoidales, está disponible en muchas máquinas CNC, también, se utiliza la interpolación parabólica y cúbica en industrias que manufacturan piezas de formas complejas como son los componentes aeroespaciales, moldes para carrocerías de automóviles, etc.

La interpolación se lleva siempre a cabo bajo velocidades de avance programadas.

La interpolación lineal consiste en cualquier número de puntos programados unidos entre sí mediante líneas rectas. Estas incluyen líneas horizontales, verticales o en ángulo donde los puntos pueden estar cercanos o alejados.

La interpolación circular facilita el proceso de programar arcos y círculos. En algunos sistemas CNC solamente se pueden programar a la vez un cuarto de círculo o un cuadrante (90°). Sin embargo, algunas otras unidades de control de máquinas CNC tienen capacidad de un círculo completo dentro del mismo comando, lo que ayuda a reducir la longitud del programa. También mejora la calidad de la pieza porque existe una transición suave en todo el círculo completo, sin interrupciones o descansos entre cuadrantes.

La información básica requerida para programar un círculo incluye la posición del centro del círculo, el inicio y final del arco que se va a maquinar, la dirección del corte, es decir, si se maquinará en sentido horario o anti horario y la velocidad de avance para la herramienta. (Krar y Check, 2005)

2.4.8 Planeación del programa CNC

La planeación del programa es una parte muy importante del maquinado CNC. Debe recolectarse, analizarse y calcularse información de importancia antes de escribir el programa. Se deben considerar las capacidades de la máquina consultando el manual de programación y de operación que lista la capacidad, requerimientos de herramental, formato de programación, etc.

Para programar CNC se deben tener antecedentes fundamentales en procedimientos y procesos de maquinado convencional, también se deben tomar en consideración todas las variables requeridas para la manufactura convencional de las piezas. Resulta ventajoso el referirse al plano de la pieza y encontrar las respuestas a las preguntas siguientes para la programación exitosa de una pieza:

- ¿Cuáles son las velocidades de corte apropiados para el tipo de material que se está maquinando?
- ¿Cómo se sujetara la pieza en una prensa simple o en un dispositivo especial?, ¿Interferirán las abrazaderas con el movimiento de los ejes?
- ¿Están disponibles las herramientas y sujetadores requeridos?
- ¿Se necesitará de un refrigerante especial, o es adecuado el tipo y concentración actual?
- ¿Cuál es la dirección de avance de la mesa? Recordando que es preferible el fresado ascendente y que esto no es un problema, ya que la mayor parte de las máquinas CNC tienen tornillos de bolas.
- ¿Con qué rapidez se puede mover la herramienta a su posición: traslación rápida o a la velocidad de avance?

- ¿Qué hará la herramienta cuando llegue a su posición, por ejemplo taladrar una perforación, fresar una cavidad, etc.?
- ¿Dónde estará localizado el cero u origen de la pieza, sobre la misma o sobre la máquina?

Es bueno recordar que el procedimiento para maquinar una pieza, ya sea mediante maquinado convencional o CNC, es básicamente el mismo. En el maquinado convencional, un operador diestro mueve manualmente las coordenadas de la máquina, en tanto que en el maquinado CNC las coordenadas de la máquina se mueven de manera automática a partir de la información suministrada por el programa CNC.

En conclusión, existen cuatro partes o elementos principales en un sistema de control numérico por computadora:

- Una computadora de uso general, que recolecta y almacena la información programada.
- Una unidad de control, que se comunica y dirige el flujo de información entre la computadora y la unidad de control de la máquina.
- La lógica de la máquina, que recibe información y que la pasa a la unidad de control de la máquina.
- La unidad de control de la máquina, que contiene las unidades servo, los controles de velocidad y de avance y las operaciones de la máquina como los movimientos del husillo y de la mesa y el cambiador automático de las herramientas (ATC) por sus siglas en inglés: *Automatic Tool Change*.

El sistema CNC como se muestra en la Figura 2.21, construido en base a una minicomputadora, contiene una gran capacidad de memoria y tiene muchas características de ayuda en la programación. Estas podrían incluir operaciones como edición de programas sobre la máquina, puesta a punto, operación y mantenimiento de la máquina. Muchas de estas características son juegos de instrucciones de máquina y de control, almacenados en la memoria que pueden ser extraídos para su uso en el programa de la pieza o por el operador de la máquina.

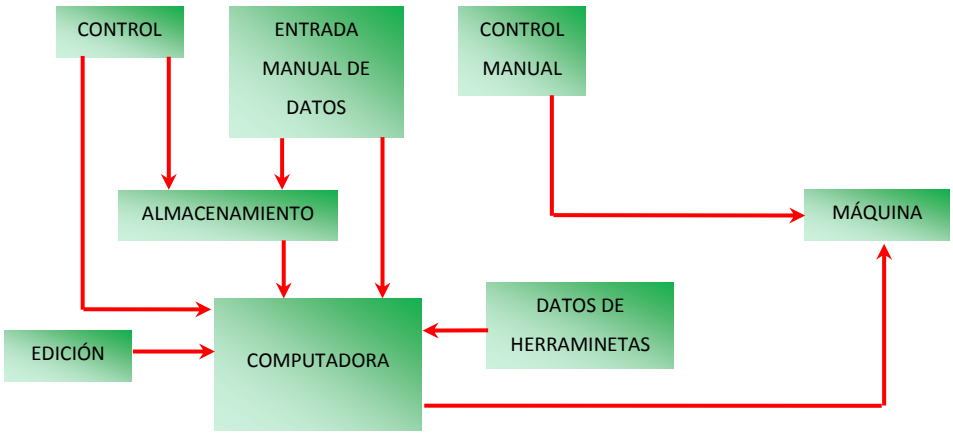


Figura 2.21 **Componentes del sistema de control de una máquina CNC.**

Sistemas antiguos CNC utilizaban lectores de cinta para leer el programa de la pieza que era preparado en una oficina de una unidad fuera de línea y entregado a la máquina en forma de una cinta perforada. En este sistema, la cinta se leía una vez y el programa de la pieza se almacenaba en la memoria para un maquinado repetitivo, es decir, el CNC no requería volver a leer la cinta para cada pieza, como era el caso en NC. Conforme evolucionaron las máquinas CNC, se incorporaron minicomputadoras y posteriormente microcomputadoras en sus controles y la cinta fue eliminada. Esto le permite al operador de la máquina la introducción manual del programa requerido para producir la pieza requerida. El programa queda almacenado en la memoria de la computadora para la producción de piezas adicionales. La ventaja principal de este sistema es su capacidad de operar en un modo vivo o conversacional, con comunicación directa entre la máquina y la computadora. Esta característica le permite al programador efectuar cambios en el programa sobre la máquina, o incluso desarrollar un programa sobre la máquina, y la entrada a la computadora es traducida inmediatamente en movimientos de la máquina. Por lo tanto, los cambios al programa se pueden observar de inmediato y efectuar las revisiones si es necesario. Esta idea del control de la máquina, permite que los programas sean probados, corregidos y revisados en una fracción del tiempo requerido por los sistemas de cinta. (Krar y Check, 2005)

2.5 Protocolos de Comunicación Serial

Existen dos formas de intercambiar información binaria: la paralela y la serial.

La comunicación paralela transmite todos los bits de un dato de manera simultánea, por lo tanto la velocidad de transferencia es rápida, sin embargo tiene la desventaja de utilizar una gran cantidad de líneas, por lo tanto se vuelve más costoso y tiene la desventaja de atenuarse a grandes distancias, por la capacitancia entre conductores así como sus parámetros distribuidos. Para este proyecto se emplea la comunicación serial, por este motivo, se tratará solamente la forma de comunicación serial.

La comunicación serial consiste en el envío de un bit de información de manera secuencial, es decir, un bit a la vez hasta completar un byte (arreglo conformado por ocho bits) de datos y a una velocidad de transferencia de datos igual entre el emisor y el receptor, que en este caso se trata de la PC y el centro de maquinado CNC respectivamente.

2.5.1 Comunicación RS232

La forma más común y sencilla de comunicar cualquier dispositivo con un ordenador es a través de su puerto serie, que es compatible con el denominado estándar RS232 (*o ELA232 Standard*). En un ordenador puede haber varios puertos serie, normalmente denominados COM1, COM2, etc.

Los puertos serie son accesibles mediante conectores. La norma RS232 establece dos tipos de conectores llamados DB-25 (de 25 pines) y DB-9 (de 9 pines) machos y hembras (de los cuales, este último es el empleado en este proyecto), como se muestra en la Figura 2.22. La norma RS232 se estableció para conectar un ordenador con un modem, por lo que aparecen muchos pines en los conectores DB-25 que en otro tipo de aplicaciones no se utilizan y en las que es más común utilizar el conector tipo DB-9. (Palacios, *et al,* 2004)

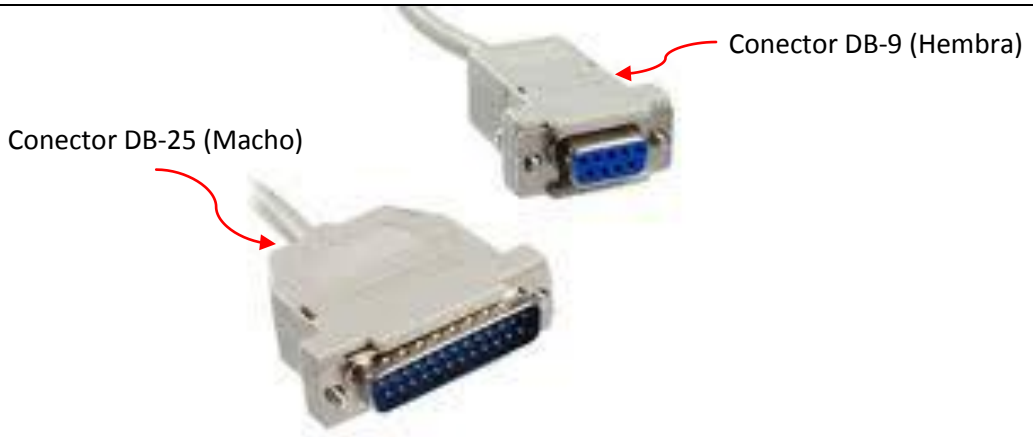


Figura 2.22 Conectores RS232.

Cada uno de los pines del conector RS232 tiene una función especificada por la norma. Hay unos terminales por los que se transmiten y reciben datos y otros que controlan el establecimiento, flujo y cierre de la comunicación. La Figura 2.23 describe los pines del DB-9.

PIN	SEÑAL
1	Data Carrier Detect (DCD)
2	Received Data (RxD)
3	Transmitted Data (TxD)
4	Data Terminal Ready (DTR)
5	Signal Ground (SG)
6	Data Set Ready (DSR)
7	Request to Send (RTS)
8	Clear to Send (CTS)
9	Ring Indicator (RI)

Figura 2.23 Identificación de los pines del conector DB-9.

Las especificaciones del puerto serie están contenidas en la norma RS232 (*o ELA Standard*).

Un dato importante a tener en cuenta en cualquier comunicación es la velocidad de transmisión, que es la cantidad de información enviada por la línea de transmisión en la unidad de tiempo. Hay distintas unidades para expresar esta medida, la más utilizada es el *Baudio*, que es proporcional a los Bits/segundo (bps), definidos como el número de bits de información enviados en un segundo.

La velocidad a la que pueden viajar los puertos COM de un ordenador está normalizada a 50, 75, 100, 110, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 76800, 115200, 230400, 460800, 921600 Baudios, etc. Esta gran variedad de valores normalizados son lo suficientemente rápidos para multitud de aplicaciones, de los cuales, la velocidad empleada para este proyecto es la de 115200 Baudios. (Palacios, *et al,* 2004)

En cuanto a niveles lógicos que debe cumplir una transmisión serie según la norma RS232, la Figura 2.24 muestra dichos niveles como sigue:

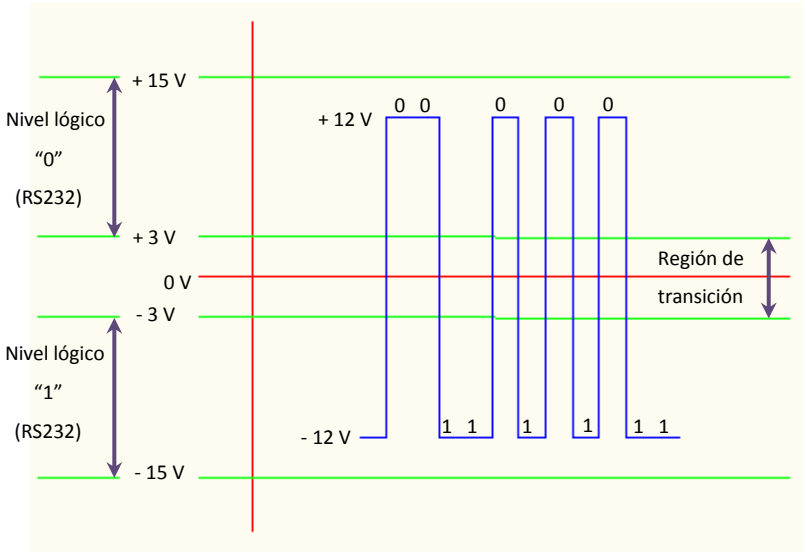


Figura 2.24 Niveles lógicos de tensión para RS232.

- Los datos se transmiten con lógica negativa, es decir, un voltaje positivo en la conexión representa un "0", mientras que un voltaje negativo representa un "1".
- Para garantizar un "0" lógico una línea debe mantener un voltaje entre +3 y +15 V.

- Del mismo modo, para un “1” lógico garantizado debe de estar entre -3 y -15 V.
- Los voltajes más usados son +12 V para el “0” y -12 V para el “1”.
- Es importante resaltar que cuando un puerto serie no está transmitiendo, mantiene el terminal de transmisión a “1” lógico a -12 V, normalmente.
- La banda muerta entre +3 V y -3 V se conoce como la región de transición donde los niveles lógicos no están definidos. Esto significa que cualquier valor entre +3 y -3 V puede interpretarse ambiguamente como “0” ó “1”.

Si se aumenta la velocidad de transmisión, las señales de datos se vuelven susceptibles a pérdidas de voltaje causadas por la capacitancia, resistencia e inductancia del cable. Estas pérdidas son conocidas como efectos de alta frecuencia y aumentan con la longitud del cable.

Estos valores de tensión proporcionan un amplio margen de seguridad que es de gran utilidad cuando los cables deben pasar por zonas cercanas a elementos que generan interferencias eléctricas: motores, transformadores, equipos de comunicación, etc. Estos elementos, sumados a la longitud del cable, pueden hacer disminuir la señal hasta en varios voltios, sin que afecte considerablemente al nivel lógico de la entrada.

La comunicación de datos en un puerto serie RS232 se puede llevar a cabo de dos formas diferentes, las cuales son: la comunicación síncrona, en la cual, el envío de los datos es sincronizado por el emisor a partir de un pulso constante de reloj (Clock), con cada pulso envía un nuevo dato. Y la comunicación asíncrona, que es la empleada para la realización de este proyecto, en la cual, la velocidad de envío de datos es acordada a priori entre el emisor y el receptor, es decir, la comunicación de datos en un puerto serie RS232 se usa normalmente para efectuar comunicaciones sin tiempo preestablecido para iniciarse. Los datos llegan en paquetes de información normalmente de 8 bits, pero para este proyecto solo se necesitarán 7 bits de datos. Algunos equipos envían caracter por caracter, otros guardan muchos caracteres en la memoria y cuando llega el momento para enviarlos, los envían uno tras otro.

El protocolo establecido por la norma RS232 envía la información estructurada en 4 partes, ilustradas en la Figura 2.25 con un ejemplo:

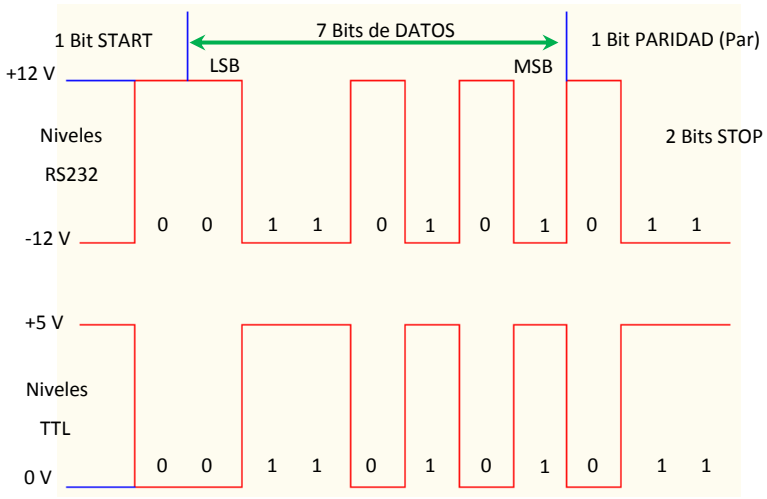


Figura 2.25 **Ejemplo de envío de un byte según la norma RS232 (Código ASCII de la letra "V": 1010110).**

- **Bit de inicio o arranque** (*Start*). Para evitar la pérdida de datos durante la comunicación, cada byte es precedido por un bit especial llamado start bit. Es un paso de -12 V a +12 V, es decir, de un "1" a un "0" lógico en la lógica negativa de la norma RS232. Cuando el receptor detecta el bit de inicio sabe que la transmisión ha comenzado y, a partir de entonces, debe leer las señales de la línea a distancias concretas de tiempo en función de la velocidad fijada por emisor y receptor.
- **Bits de datos** (*Datas*). Un bit es un acrónimo de ***B**inary digi**t*** (dígito binario) y es la unidad más pequeña que puede contener información en una computadora. Cada digito binario puede tener valores de uno (1) o cero (0). Además, un bit es algo como un switch que se puede encender o apagar dependiendo de lo que se requiera. En la computadora, cada letra, símbolos y dígitos usados en la programación CNC son representados por una serie de bits, ocho bits para ser precisos, que crean una unidad llamada byte. Los bits de datos son enviados al receptor después del bit de *Start*. El bit de menos peso LSB (*Least Significant Bit*)

es transmitido primero y el de mayor peso MSB (*Most Significant Bit*) el último. Un caracter de datos suele consistir en 7 u 8 bits. En el ejemplo de la Figura 2.25 trabaja con 7 bits.

- **Bit de paridad** (*Parity*). Dependiendo de la configuración de la transmisión un bit de paridad puede ser enviado después de los bits de datos. En aplicaciones sencillas no suele utilizarse. Con este bit se pueden descubrir errores en la transmisión. Se puede dar paridad par o impar, donde la paridad par es la más común para comunicaciones CNC, además, esta paridad también es acorde con el desarrollo de este proyecto. En la paridad par, por ejemplo, la palabra de datos a transmitir se completa con el bit de paridad de manera que el número de bits "1" enviados sea par, como el ejemplo de la Figura 2.25, donde la información de byte de datos más paridad es '01010110' (cuatro "1").
- **Bit de parada** (*Stop*). La línea queda a -12 V después del último bit enviado, es decir, queda a "1" en la lógica negativa de la norma RS232. Indica la finalización de la transmisión de una palabra de datos. El protocolo de transmisión de datos permite 1, 1.5, ó 2 bits de parada, que en este caso, se utilizarán dos bits.

En el ejemplo de la Figura 2.25 se envía un bit de inicio, una palabra de 7 bits (1010110), que corresponde a la letra "V" en código ASCII, un bit de paridad par y luego dos bits de paro.

Existen en la actualidad diferentes ejemplos de puertos que comunican información de manera serial. El conocido puerto serial ha sido gradualmente reemplazado por el puerto USB (Universal Serial Bus) que permite una mayor versatilidad en la conexión de múltiples dispositivos. Aunque en naturaleza serial, no suele referirse de esta manera ya que sigue sus propios estándares.

La mayoría de los centros de fresado CNC, incluyendo el centro de maquinado CNC empleado para este proyecto, poseen un puerto de comunicación serial. Para comunicarse con las PC´s actuales que poseen únicamente puerto USB requieren de un dispositivo "traductor", el cual es un convertidor USB-serial. A través de este "traductor", se pueden recibir y enviar datos a una PC de manera serial, que para este caso, la comunicación entre la PC y el centro de maquinado CNC será de esta manera, a través de un cable USB-RS232 y por tal motivo, este capítulo se enfocó en el protocolo de comunicación serial RS232. (Palacios, *et al,* 2004)

2.5.2 Control Numérico Distribuido (DNC)

La entrada/salida (I/O) del puerto RS232 en una máquina CNC son usadas para enviar y recibir datos. En muchas industrias, los programas son transferidos a través de lo que se conoce como Control Numérico Distribuido (DNC, *Distributed Numerical Control*). El control cuenta con características disponibles para hacer que la transferencia de datos sea posible. (Smid, 2003)

Las máquinas CNC utilizan un teclado de computadora con formato de acuerdo con la norma del Código Estándar Americano para Intercambio de Información (ASCII, *American Standard Code for Information Interchange*) para introducir información sobre programas directamente a la unidad de control de la máquina. Para una operación correcta, el uso del teclado requería algún tipo de software de comunicación y una conexión compatible entre el teclado de la computadora y la unidad de control de la máquina. El DNC que utiliza una microcomputadora junto con software de comunicación, está convirtiéndose en el método de entrada preferido. Con el DNC, los datos del programa pueden ser enviados a la CNC para el maquinado de piezas. Para la introducción manual de datos se necesita un teclado alfanumérico en el panel de control del operador. Si se hace edición al programa, esta nueva información también puede ser enviada de regreso a través del enlace DNC para que sea almacenado para uso futuro.

El control CNC se utiliza generalmente para controlar máquinas individuales, mientras que el control DNC se utiliza por lo general donde están involucradas seis o más máquinas CNC en un programa completo de manufactura, por ejemplo en un sistema de manufactura flexible.

Para la comunicación entre una máquina CNC y una PC usando el puerto RS232, todo el equipo requerido es un cable entre los dos dispositivos y un software.

En un sistema DNC como el que se muestra en la Figura 2.26, varias máquinas con CNC están controladas a partir de una computadora principal. Esto puede manejar la programación del trabajo y puede descargar un programa completo en la memoria de la máquina cuando se requieren nuevas piezas. En vista de que la mayor parte de las máquinas CNC están equipadas con su propia minicomputadora o microcomputadora, es posible operar cada máquina de manera individual mediante CNC en el caso que la computadora Principal fallara. En una instalación de

manufactura más reducida, se puede utilizar una microcomputadora para fines DNC. (Krar y Check, 2005)

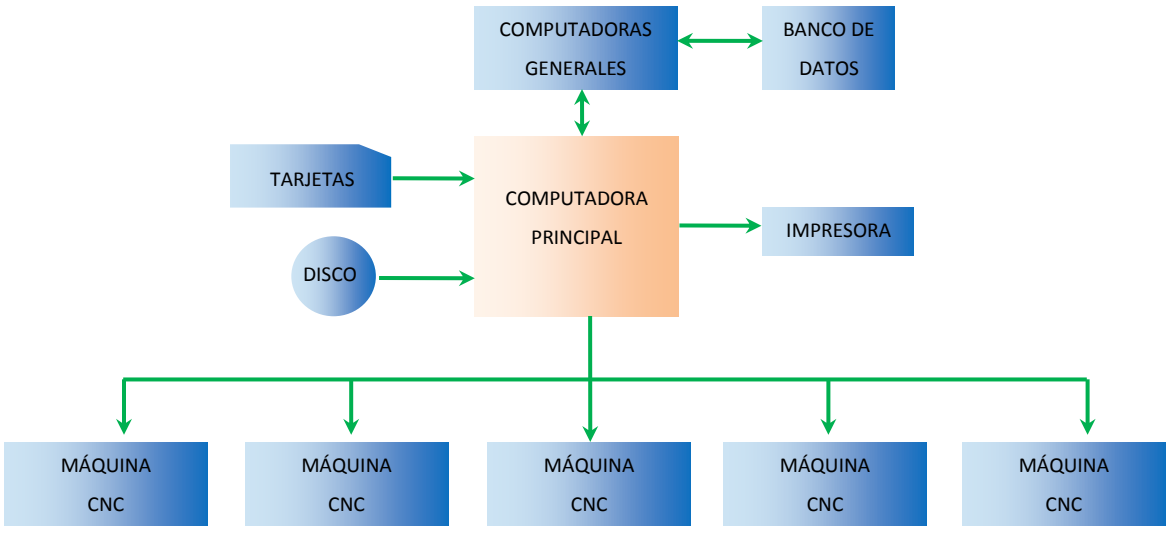


Figura 2.26 **Principales componentes de un sistema DNC.**

El DNC requiere entonces, procedimientos organizados para realizar un trabajo eficiente y de buena calidad. Además, cabe mencionar que el DNC no es parte de la unidad de control. (Krar y Check, 2005)

Para fines de este proyecto, la comunicación DNC será utilizada para comunicar al centro de maquinado CNC con la PC y que ésta última a su vez estará conectada al mouse 3D. Esto porque la máquina será controlada en línea para realizar diferentes trayectorias basadas en el movimiento que el operador imprima en el mouse 3D. Así, no se realizará un programa previo para cargarlo a la máquina, ya que cada trayectoria puede ser diferente porque puede el operador estar generando coordenadas continuamente. Ahora, si se desea que la máquina repita una o varias veces alguna trayectoria descrita por el mouse 3D, ésta se puede guardar en la memoria de la máquina y entonces podrá ser operada con CNC.

2.5.3 Mouse 3D

Este dispositivo 3Dconnexion permite a los programadores procesar para algún fin en específico los datos de entrada de tal dispositivo. Para que se tenga una mayor familiarización con este mouse 3D se muestra como es físicamente en la Figura 2.27 y a continuación se describen algunas de sus características y modo de operación.

Figura 2.27 Mouse 3D.

El mouse de navegación 3Dconnexion empleado para la realización de este proyecto es un dispositivo de entrada 3D que detecta el más mínimo movimiento de los dedos y al imprimir una presión en dirección al eje X, Y ó Z se generan traslaciones o rotaciones para mover modelos 3D de forma instantánea y simultánea; esto referido a programas dentro del ambiente de diseños 3D pero para este proyecto se capturan los datos proporcionados por el 3Dconnexion en cualquiera de los tres ejes anteriormente mencionados. Esto proporciona intuitivamente, la interacción de seis grados de libertad para el control de imágenes, gráficas, objetos en 3D, etc.

Hay dos formas básicas para mover objetos 3D con el dispositivo:

- Manipular el cabezal de mando del 3Dconnexion como si se estuviera sosteniendo el modelo 3D en la mano; esta forma se le conoce como **modo objeto**. Si se presiona hacia la izquierda, el modelo se mueve a la izquierda. Presionando a la

derecha, éste se mueve a la derecha. Si se levanta, es decir, jalarlo hacia arriba, el modelo se moverá hacia arriba y en caso contrario, si se presiona el 3Dconnexion el modelo se moverá hacia abajo. Además, alejar o acercar el cabezal de mando, el modelo responderá a alguno de estos movimientos respectivamente.

- Manipular el cabezal de mando del 3Dconnexion como si fuera una cámara se le conoce como **modo cámara**, es decir, si se tiene una escena (imagen) y el dispositivo se pulsa hacia adentro de ésta, la cámara se mueve hacia la escena. La escena parecerá moverse hacia el espectador. Si se presiona hacia la izquierda la cámara se moverá hacia este lado (la escena se mueve a la derecha). Si se presiona a la derecha, la cámara se moverá a la derecha (la escena se mueve a la izquierda). Si el cabezal de mando se jala, la cámara se moverá hacia arriba y si el cabezal se presiona, la cámara se moverá hacia abajo. La escena siempre se moverá en la dirección opuesta del dispositivo de entrada, es decir, el espectador entra en la escena y es como si caminara en ella; normalmente toma un poco de tiempo acostumbrarse a este modo.

Se pueden ejecutar cualquiera de estas acciones al mismo tiempo, haciendo que la imagen se mueva a medida que se mueve el dispositivo.

El objetivo de la interacción de un ratón 3D en una aplicación es proporcionar al usuario una sensación como si estuviera sosteniendo el objeto que se muestra o la cámara en la mano. Esta ilusión solo puede mantenerse si el objeto se mueve simultáneamente con el movimiento de la mano y en la dirección que el usuario espera.

La orientación de los ejes se tomará como sigue: el eje X será de derecha a izquierda; el eje Y se tomará acercando o alejando el cabezal de mando con referencia al usuario y el eje Z será presionando o jalando el mismo cabezal de mando del dispositivo.

Cuando se utiliza el *mouse 3D* para sostener y mover el objeto visto en 3D, la velocidad de encuadre en el plano de la pantalla debe ser ajustada en función de la distancia del punto de vista al objeto, es decir, cuando el usuario acerca el objeto, que pueda tener un mejor detalle de algún lugar en específico de dicho objeto y tener un control preciso del mouse 3D, en consecuencia la panorámica de velocidad debe reducirse.

Un algoritmo simple de velocidad que da cuenta de la distancia de visión es el que regula la panorámica de un punto o en la superficie del objeto, la cual queda independiente del tamaño del objeto, el tiempo que el punto toma atravesar la ventana de vista es constante para el mismo desplazamiento de la tapa del mouse 3D.

El dispositivo 3Dconnexion puede tener varios números de teclas, las cuales consisten en contar con funciones específicas pre asignadas, además de teclas que el usuario puede asignar a los comandos disponibles. Generalmente, las teclas están numeradas consecutivamente a partir del 1 hasta el número de teclas en el dispositivo. Cuando el usuario presiona una tecla, el comando asociado a ésta, no puede ser ejecutado directamente por el dispositivo, la aplicación notificará que debe ejecutar el comando requerido.

Hay una serie de comandos que están predefinidos y que deben ser reconocidos por todas las aplicaciones, estos comandos son:

- El comando de ajuste (*Fit*). La funcionalidad exacta asociada a "fit" debe interpretarse en el contexto de la aplicación. Por ejemplo, en un entorno de modelado que puede ser simplemente un objeto centrado en la pantalla, en un Sistema de Interfaz Gráfica (GIS, *Graphical Interface System*). Esta aplicación podría ser interpretada como el establecimiento de un determinado nivel de zoom y orientación.
- Activar el panel de configuración. Cuando este comando se recibe, la aplicación debe cambiar el estado de visibilidad de la configuración del mouse 3D o el panel de opciones.

El dispositivo *3Dconnexion* proporciona un movimiento completo, simultáneamente en seis ejes en todas y cada una de las direcciones. La Figura 2.28 muestra la orientación de los ejes de rotación y traslación en el sensor del dispositivo. A raíz de esta figura, se explican los datos del rango de cada eje, que "significa" el eje tanto en navegación 6D (modo de cámara) y control de objeto 6D (modo objeto) en los siguientes párrafos.

Figura 2.28 Orientación de los ejes de rotación y traslación del mouse 3D. (Bonk)

El rango normal de los ejes del dispositivo es de aproximadamente ± 500 unidades. Sin embargo, el usuario puede escalar hacia arriba o abajo de estos valores en la interfaz gráfica de usuario para que su aplicación pueda ser capaz de manejar los valores más grandes o más pequeños.

El dispositivo 3Dconnexion genera tanto una traslación y un vector de rotación al mismo tiempo que el usuario acciona, tironea o tuerce la tapa del dispositivo como se muestra en la Figura 2.29. El vector de traslación es proporcional al desplazamiento lineal que el usuario aplica al mango. Las rotaciones del dispositivo son proporcionales al vector sobre el cual el usuario está aplicando un desplazamiento rotatorio.

Figura 2.29 Aplicando una rotación al sensor 3D. (Bonk)

Es bastante fácil de interpretar el vector de traslación. Las tres componentes (X,Y y Z) de este vector se pueden aplicar de manera semejante como los datos del teclado o del mouse para la transformación de la visualización. El vector de rotación es un caso diferente. Aplicando el vector rotación como partes individuales no dará el mismo resultado que rotar sobre el vector (Figura 2.30). Si los datos se aplican individualmente, el usuario notará un “bambaleo” al realizar una rotación.

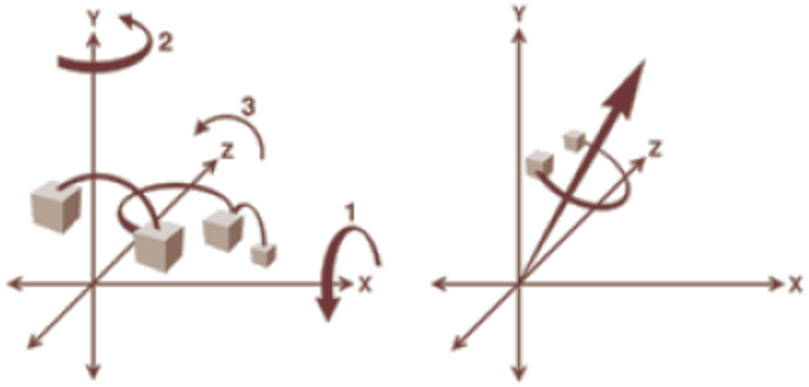


Figura 2.30 Rotación individual vs. Rotación sobre un vector. (Bonk)

Una de las preguntas más frecuentes en torno al dispositivo 3Dconnexion es la forma de asignar los ejes de este dispositivo para que coincida con la aplicación requerida. Esta es la regla general: la posición “home” para el dispositivo es con el eje negativo Z apuntando hacia la pantalla. Los dados en las Figuras 2.31 – 2.35 muestran la posición “home” o el resto de las posiciones del dispositivo entre los dos postes, ligeramente encima de la superficie como se muestra en dichas figuras.

Figura 2.31 Objeto en reposo. (Bonk)

Entrada del dispositivo	Modo cámara	Modo objeto
Traslación del eje Z de 0 a Max 	La cámara se mueve hacia atrás 	El objeto se acerca 
Traslación del eje Z de 0 a -Max 	La cámara se acerca 	El objeto se aleja 
Traslación del eje Y de 0 a Max 	La cámara se mueve hacia arriba o salta 	El objeto se mueve arriba 

Figura 2.32 Controles de traslación primera parte. (Bonk)

Entrada del dispositivo	Modo cámara	Modo objeto
Traslación del eje Y de 0 a -Max 	La cámara se baja 	El objeto se mueve abajo 
Traslación del eje X de 0 a Max 	La cámara se mueve a la derecha 	El objeto se mueve a la derecha 
Traslación del eje X de 0 a -Max 	La cámara se mueve a la izquierda 	El objeto se mueve a la izquierda 

Figura 2.33 Controles de traslación segunda parte. (Bonk)

Entrada del dispositivo	Modo cámara	Modo objeto
Rotación del eje Z de 0 a Max 	La cámara rueda a la izquierda 	El objeto baja del lado izq. 
Rotación del eje Z de 0 a -Max 	La cámara rueda a la derecha 	El objeto baja del lado derecho 
Rotación del eje Y de 0 a -Max 	La cámara gira a la derecha 	El objeto gira con las manecillas del reloj 

Figura 2.34 Controles de rotación primera parte. (Bonk)

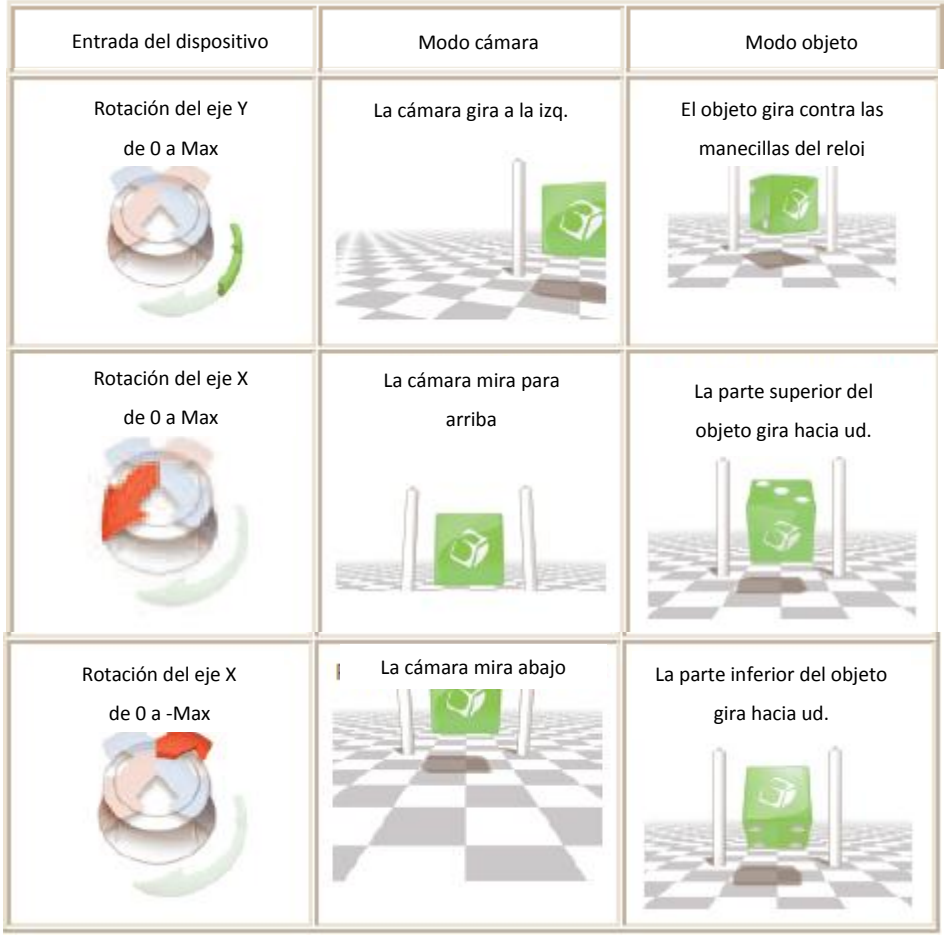


Figura 2.35 Controles de rotación segunda parte. (Bonk)

Es importante, desde el punto de vista de la experiencia del usuario, que la velocidad de respuesta de una aplicación sea compatible con la de las demás. El usuario se relaciona fácilmente con la fuerza necesaria para alcanzar una velocidad determinada (de rotación o traslación) en una aplicación y por lo tanto, espera el mismo comportamiento de otras aplicaciones.

Por default, el dispositivo está configurado con una baja velocidad en un inicio para "cualquier aplicación" para el uso sin experiencia. (Bonk)

2.6 Lenguajes de Programación

2.6.1 Microsoft Visual Studio 2008

Los programas Windows son independientes de la máquina en la que se ejecutan, el acceso a los dispositivos físicos se hace a través de interfaces, y nunca se accede directamente a ellos. Esta es una de las principales ventajas para el programador, ya que no hay que preocuparse por el modelo de tarjeta gráfica o de impresora, la aplicación funcionará con todas y será el sistema operativo el que se encargue de que así sea.

Un concepto importante es el de recurso. Desde el punto de vista de Windows, un recurso es todo aquello que puede ser usado por una o varias aplicaciones. Existen recursos físicos, que son los dispositivos que componen el ordenador, como la memoria, la impresora, el teclado o el ratón y recursos virtuales o lógicos, como los gráficos, los iconos o las cadenas de caracteres.

Por ejemplo, si la aplicación actual requiere el uso de un puerto serie, primero debe averiguar si está disponible, es decir, si existe y si no lo está usando otra aplicación; y después lo reservará para su uso. Esto es necesario porque este tipo de recurso no puede ser compartido.

Lo mismo pasa con la memoria o con la tarjeta de sonido, aunque son casos diferentes. Por ejemplo, la memoria puede ser compartida, pero de una forma general, cada porción de memoria no puede compartirse (al menos en los casos normales, aunque es posible hacer aplicaciones con memoria compartida) y se trata de un recurso finito. Las tarjetas de sonido, dependiendo del modelo, podrán o no ser compartidas por varias aplicaciones. Otros recursos como el ratón y el teclado también se comparten, pero se asigna su uso automáticamente a la aplicación activa, a la que normalmente nos referimos como la que tiene el “foco”, es decir, la que mantiene contacto con el usuario.

Desde el punto de vista como programador, también se consideran como recursos varios componentes como menús, los iconos, los cuadros de diálogo, las cadenas de caracteres, los mapas de bits, los cursores, etc. En sus programas, Windows almacena de forma separada el código y los recursos, dentro del mismo fichero, y estos últimos pueden ser editados por

separado, permitiendo por ejemplo, hacer versiones de los programas en distintos idiomas sin tener a los ficheros fuente de la aplicación.

Ahora, la forma en que se presentan las aplicaciones Windows (al menos las activas) ante el usuario, es la ventana (área rectangular de la pantalla que se usa de interfaz entre la aplicación y el usuario).

Cada aplicación tiene al menos una ventana, la ventana principal, y todas las comunicaciones entre usuario y aplicación se canalizan a través de una ventana. Cada ventana comparte el espacio de la pantalla con otras ventanas, incluso de otras aplicaciones, aunque solo una puede estar activa, es decir, solo una puede recibir información del usuario.

También, los programas en Windows están orientados a eventos, esto significa que normalmente los programas están esperando a que se produzca un acontecimiento que les incumba, y mientras tanto permanecen aletargados o dormidos.

Un evento puede ser por ejemplo, el movimiento del ratón, la activación del menú, la llegada de información desde el puerto serie, una pulsación de una tecla, etc.

Esto es así porque Windows es un sistema operativo multitarea, y el tiempo de microprocesador ha de repartirse entre todos los programas que se estén ejecutando. Si los programas fueran secuenciales puros, esto no sería posible, ya que hasta que una aplicación finalizara, el sistema no podría atender al resto.

Debido a la complejidad de los programas Windows, normalmente se dividen en ficheros fuente, que se compilan por separado y se enlazan juntos.

Cada compilador puede tener diferencias, más o menos grandes, a la hora de trabajar con proyectos. En esta investigación se trabajó con el compilador Microsoft Visual Studio 2008.

Para crear un proyecto Windows usando este compilador se elige el menú “File/New/Project…”.

Se abrirá un cuadro de diálogo donde se puede elegir el tipo de proyecto. En este caso se elige “Win32” y “Win32 project”. A continuación se pulsa “OK”.

El compilador crea un proyecto en un fichero C++, con el esqueleto de una aplicación para una ventana, a partir de ahí empieza el trabajo del programador.

En parte, para que no resulte muy difícil adaptarse a la terminología de Windows, y a la documentación existente, en la mayoría de los casos se refieren a componentes y propiedades de Windows con sus nombres en inglés. Por ejemplo, “button”, “check box”, “list box”, “combo box” o “property sheet”, y en algunas veces se mencionan en español, por ejemplo, “ dialog box” que significa “cuadro de diálogo”.

También, se hablan de ventanas “overlapped” o superponibles, que son las ventanas corrientes. Para el término “pop-up” no se tiene una traducción como tal.

Son también utilizados a menudo, con relación a “check boxes”, términos ingleses como checked, unchecked o grayed, en lugar de marcado, no marcado o gris.

Owner-draw, es un estilo que indica que una ventana o control no es la encargada de actualizarse en pantalla, esa responsabilidad es transferida a la ventana dueña del control o ventana.

Para “bitmap” se usa a menudo la expresión “mapa de bits”.

Ahora bien, los controles son la forma en que las aplicaciones Windows intercambian datos con el usuario. Normalmente se usan dentro de los cuadros de diálogo, pero en realidad pueden usarse en cualquier ventana. Existen bastantes, pero los más importantes y conocidos son:

- Control estatic: son etiquetas, marcos, iconos o dibujos.
- Control edit: permiten que el usuario introduzca datos alfanuméricos en la aplicación.
- Control list box: el usuario puede escoger entre varias opciones de una lista.
- Control combo box: es una combinación entre un edit y un list box.
- Control scroll bar: barras de desplazamiento, para la introducción de valores entre márgenes definidos.

- Control button: realizan acciones o comandos, de button se derivan otros dos controles muy comunes: control check box: permite leer variables de dos estados “checked” o “unchecked” y control radio button: se usa en grupos, dentro de cada grupo sólo uno puede ser activado.

Ahora, se explican brevemente los elementos que conforman una ventana (aunque no todos tienen por qué estar presentes en todas las ventanas, es decir, cada ventana debe contener solamente los componentes necesarios para que se cumpla adecuadamente el procedimiento de esta; los componentes de la ventana son (Figura 2.36):

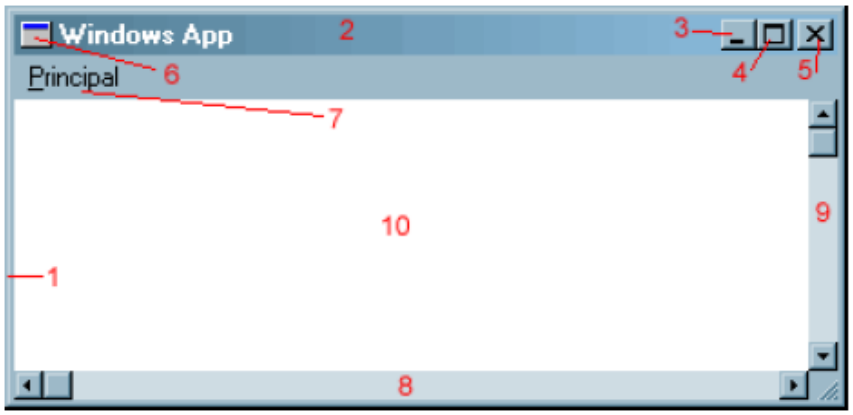


Figura 2.36 Componentes de una ventana. (Pozo, 2007)

- El borde de la ventana. Hay varios tipos, dependiendo de si están o no activas las opciones de cambiar el tamaño de la ventana. Se trata de un área estrecha alrededor de la ventana que permite cambiar su tamaño (1).
- La barra de título. Es la zona en la parte superior de la ventana que contiene el icono y el título de la ventana, esta zona también se usa para mover la ventana a través de la pantalla, y mediante doble clic, para cambiar entre el modo maximizado y tamaño normal (2).
- La caja de minimizar es una pequeña área cuadrada, situada en la parte derecha de la barra de título que sirve para disminuir el tamaño de la ventana (3).

- La caja de maximizar es una pequeña área cuadrada, situada en la parte derecha de la barra de título que sirve para agrandar la ventana para que ocupe toda la pantalla. Cuando la ventana está maximizada, se sustituye por la caja de restaurar (4).
- La caja de cerrar es una pequeña área cuadrada, situada en la parte derecha de la barra de título que sirve para cerrar la ventana (5).
- La caja de control de menú es una pequeña área cuadrada, situada en la parte izquierda de la barra de título, normalmente contiene el icono de la ventana, y sirve para desplegar el menú del sistema (6).
- Menú o menú del sistema. Se trata de una ventana especial que contiene las funciones comunes a todas las ventanas, también accesibles desde las cajas y el borde, como minimizar, restaurar, maximizar, mover, cambiar tamaño y cerrar. Este menú se despliega al pulsar sobre la caja de control de menú.
- La barra de menú es una zona distribuida debajo de la barra de título, contiene los menús de la aplicación (7).
- La barra de scroll horizontal, es la barra situada en la parte inferior de la ventana, permite desplazar horizontalmente la vista del área de cliente (8).
- La barra de scroll vertical, es la barra situada en la parte derecha de la ventana, permite desplazar verticalmente la vista del área de cliente (9).
- El área de cliente es la zona donde el programador sitúa los controles, y los datos para el usuario. En general, es toda la superficie de la ventana que no está ocupada por las zonas anteriores (10). (Pozo, 2007)

2.6.2 Código G y código M

El tipo más común de formato de programación utilizado para los sistemas de programación CNC, es el formato de dirección de palabra. Este formato contiene un gran número de códigos diferentes para transferir información de programa a los servos, relevadores, microinterruptores de la máquina, etc. a fin de ejecutar los movimientos necesarios para la fabricación de una pieza. Estos códigos, que cumplen con estándares establecidos, se reúnen en

una secuencia lógica conocida como un bloque de información. Cada bloque solamente debe contener la información suficiente para llevar a cabo un paso de una operación de maquinado.

Los programas para las piezas deben ponerse en un formato que pueda comprender la unidad de control de la máquina. El formato utilizado en un sistema CNC está determinado por el fabricante de la máquina-herramienta y se basa en la unidad de control de la máquina. Comúnmente se utiliza un formato de bloques variables que usa palabras (letras). Cada palabra de instrucción está formada por un caracter de dirección, como S, X, Y, Z, T, F ó M. Este caracter alfabético antecede datos numéricos utilizados para identificar una función específica de un grupo de palabras, para dar un valor de distancia, velocidad de avance o velocidad de la herramienta (rpm).

Los códigos más comunes utilizados para la programación CNC son los códigos G (comandos preparatorios) y los códigos M (funciones misceláneas) como se muestra en la Figura 2.37. Los códigos F, S, D, H, P y T se utilizan para representar funciones tales como avance, velocidad, excentricidad diametral del cortador, compensación de la longitud de la herramienta, llamada de subrutina, número de la herramienta, etc. Los códigos A (ángulo) y R (radio) se utilizan para localizar puntos sobre arcos y círculos que involucran ángulos y radios.

Los códigos G, llamados a veces códigos de ciclo, se refieren a alguna acción que ocurre en los ejes X, Y y/o Z de una máquina-herramienta. Estos códigos están agrupados en categorías, como el grupo número 01, que contiene los códigos G00, G01, G02 y G03. Estos códigos causan algún movimiento de la mesa o del cabezal de la máquina.

Un código G00 se utiliza para posicionar con rapidez la herramienta de corte o la pieza de trabajo de un punto de la misma a otro. Durante el rápido recorrido, se puede mover el eje X, el Y o ambos ejes simultáneamente. La velocidad de recorrido rápido puede variar de máquina a máquina y puede ir desde 200 hasta 800 pulg/min (5 a 20 m/min).

Los códigos G01, G02 y G03 mueven los ejes a una velocidad controlada de avance. El G01 se utiliza para interpolación lineal (movimiento en línea recta) mientras que el G02 (con las manecillas del reloj -CW-) y G03 (contra las manecillas del reloj -CCW-) se utilizan para interpolación circular (arcos y círculos). (Krar y Check, 2005)

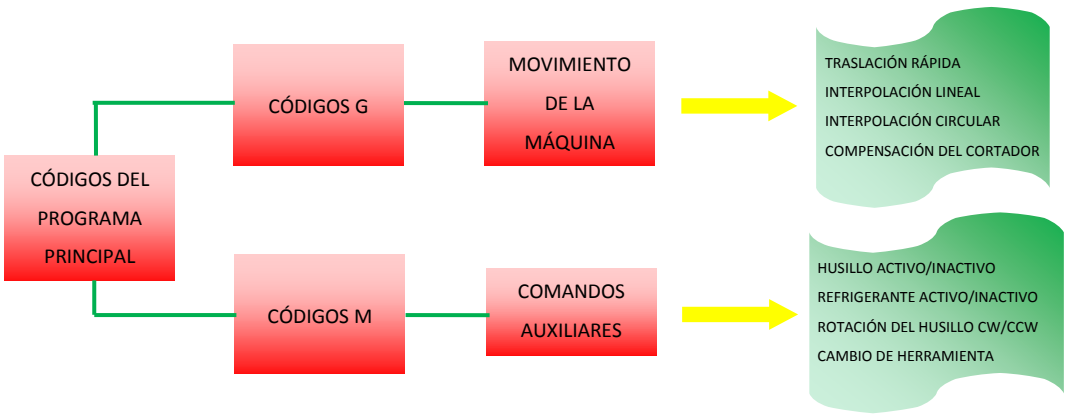


Figura 2.37 Tipos de códigos utilizados en programación CNC.

Algunos códigos G se clasifican como modales o no modales. Los códigos modales se mantienen en efecto en el programa hasta que son modificados por otro código del mismo grupo. Los códigos no modales se mantienen en efecto solo durante una operación y deben ser programados de nuevo siempre que se requieran. En el grupo 01, por ejemplo, solamente uno de los cuatro códigos de este grupo se puede utilizar en cualquier momento. Si un programa se inicia con un G00 y se escribe un G01 después, el G00 queda cancelado del programa hasta que se le vuelve a escribir. Si se introduce en el programa un código G02 o G03, el G01 quedará cancelado y así sucesivamente. La Tabla 2.1 muestra los códigos G referidos al centro de maquinado CNC Dyna; que es la máquina utilizada para este proyecto y que cumplen con los estándares EIA. (Krar y Check, 2005)

TABLA 2.1 Lista de códigos G para Dyna. FUENTE: Manual para Dyna M4, versión 2.2

Código G	Código Dyna	Descripción	Tipo
G00	GOF	Movimiento rápido	Modal
G01	GO	Interpolación lineal (Avance)	Modal
G02	ARCL	Interpolación circular (Manecillas del reloj CW)	Modal
G02.1	SPLL	Espiral (CW)	No modal
G03	ARCR	Interpolación circular (contra manecillas del reloj CCW)	Modal
G03.1	SPLR	Espiral (CCW)	No modal
G04	DWELL	Retardo o permanencia en el programa	No modal
G08	ARC	Arco (A través del punto medio)	No modal
G12	CIRL	Final del arco (CW)	No modal
G13	CIRR	Final del arco (CCW)	No modal
G16	Y_U	Conversión del cuarto eje al eje Y	Modal
G17	XY	Selección del plano XY	Modal
G18	XZ	Selección del plano XZ	Modal
G19	YZ	Selección del plano YZ	Modal
G20	IN	Selección del sistema en pulgadas (Sistema inglés)	Modal
G21	MM	Selección del sistema en milímetros (Sistema métrico)	Modal
G22	CONTOUR	Ciclo de fresado de contorno	No modal
G23	PKT	Ciclo de fresado de cajas universales (formas irregulares)	No modal
G24	RECT_PKT	Ciclo de caja rectangular	No modal
G25	CIR_PKT	Ciclo de caja circular	No modal
G26	DIE_F	Caja con paredes no perpendiculares a la base de ésta	No modal
G27	DIE_M	Escalón con paredes no perpendiculares a la base de éste	No modal
G28	GO_HOME	Manda los ejes a la posición Home de la máquina	No modal
G34	CIR_CYC	Arreglo circular (Barrenar sobre perímetro de un circulo)	No modal
G35	LINE_CYC	Ciclo de barrenos sobre una línea recta	No modal
G36	ARC_CYC	Ciclo de barrenos en un arco	No modal

Continuación de la Tabla 2.1

Código G	Código Dyna	Descripción	Tipo
G37	RECT_CYC	Ciclo de barrenos en una red	No modal
G40	OFF_COMP	Cancelar la compensación de corte en el plano XY	Modal
G41	COMP_L	Compensación del corte, herramienta del lado izquierdo	Modal
G42	COMP_R	Compensación de corte, herramienta del lado derecho	Modal
G43	COMP_TL	Compensación de corte, longitud de herramienta	Modal
G49	OFF_TL	Cancelar compensación de longitud de herramienta	Modal
G50	OFF_TRAN	Cancela la escala, rotación y funciones de espejo	Modal
G51	SCALE	Ciclo de escala (agrandar o reducir una forma específica)	Modal
G51.1	MIRROR	Ciclo de espejo (reflejar una forma específica)	Modal
G51.2	XYZ	Ciclo de traslación (Inclinación de plano)	Modal
G52	ZERO_AT	Establece un cero local en un punto dado	Modal
G53	COORD0	Selecciona las coordenadas máquina	Modal
G54	COORD1	Selecciona el primer sistema de coordenadas pieza	Modal
G55	COORD2	Selecciona el segundo sistema de coordenadas pieza	Modal
G56	COORD3	Selecciona el tercer sistema de coordenadas pieza	Modal
G57	COORD4	Selecciona el cuarto sistema de coordenadas pieza	Modal
G58	COORD5	Selecciona el quinto sistema de coordenadas pieza	Modal
G59	COORD6	Selecciona el sexto sistema de coordenadas pieza	Modal
G68	ROTATE	Rotar un programa alrededor de un punto dado	Modal
G73	STEP_CYC	Ciclo de barrenado	Modal
G74	TAP_REV	Atrás tocando una posición dada	Modal
G76	BORE_F	Mueve hacia abajo el eje Z hasta acabar el barreno	Modal
G80		Ciclo de cancelar barrenado	Modal
G81	DRILL	Ciclo de barrenado	Modal
G82	DRILL_P	Ciclo de barrenado con detención	Modal
G83	DRILL_Q	Ciclo de barrenado picoteando	Modal
G84	TAP	Ciclo de roscado	Modal
G85	BORE	Ciclo de mandrinado fino	Modal
G86	BORE_P	Ciclo de mandrinado	Modal

Continuación de la Tabla 2.1

Código G	Código Dyna	Descripción	Tipo
G87	BORE_B	Ciclo de mandrinado	Modal
G88	BORE_M	Ciclo de mandrinado	Modal
G89	BORE_S	Ciclo de mandrinado	Modal
G90	ABS	Modo absoluto	Modal
G91	INC	Modo incremental	Modal
G92	CURRENT	Genera una nueva posición de coordenadas	Modal
G94	F_MIN	Velocidad de alimentación en mm/min	Modal
G95	F_REV	Velocidad de alimentación en mm/rev	Modal
G98	END_Z0	Regresa al punto inicial	Modal
G99	END_R	Regresa al plano “R”	Modal
	SMOOTH=	Tasa de cambio suave	Modal
	ZFEED=	Cambia la velocidad de alimentación de Z	Modal

Los códigos M se utilizan para activar o desactivar diferentes funciones que controlan ciertas operaciones de la máquina-herramienta. Estos códigos por lo general no se agrupan por categorías, aunque varios códigos pueden controlar el mismo tipo de operación para ciertos componentes de la máquina. Por ejemplo, tres códigos, M03, M04 y M05, todos controlan alguna función del husillo de la máquina-herramienta: M03 hace girar el husillo de la máquina en sentido de las manecillas del reloj (CW), M04 hace girar al husillo de la máquina en sentido contrario a las manecillas del reloj (CCW), y el M05 desactiva el husillo. Los tres códigos se consideran modales porque se conservan válidos hasta que se introduce otro código que los reemplace.

La Tabla 2.2 muestra los códigos M utilizados para la programación de la centro de maquinado CNC Dyna. (Krar y Check, 2005)

TABLA 2.2 Lista de códigos M para Dyna. FUENTE: Manual para Dyna M4, versión 2.2

Código M	Descripción	Código M	Descripción
M00	Paro del programa	M03	Pone a girar el husillo en sentido CW
M01	Paro opcional	M04	Pone a girar el husillo en sentido CCW
M02	Fin del programa	M05	Apaga el husillo

Continuación de la Tabla 2.2

Código M	Descripción	Código M	Descripción
M06	Cambia la herramienta (opcional)	M65	Comando directo al manejador
M08	Enciende el refrigerante	M70	Llamar modo DNC
M09	Apaga el refrigerante	M71	Crea un salto condicional: Sí esto, entonces… (If)
M10	Cambio automático de herramienta de repuesto	M72	Salto para especificar el número de línea
M19	Alinea el husillo con el carrusel de herramientas	M73	Repetir programa n veces
M22	Envía señal de sincronización PLC	M74	Finalizar repetir programa
M23	Espera por la señal de sincronización PLC	M75	Fin del programa NC
M24	Apaga la señal de sincronización PLC	M76	Comenzar a ejecutar un nuevo programa NC
M25	Carga contador	M77	Marca iniciar bloque A
M26	Comenzar la cuenta regresiva	M78	Marca finalizar bloque A
M30	Fin de programa	M79	Llama un mensaje de error
M31	Encender la segunda bomba de refrigerante	M80	Calcula la velocidad del husillo
M32	Apaga la segunda bomba de refrigerante	M81	Cambio de los parámetros de la herramienta
M35	Transportador de rebaba hacia adelante	M82	Cambia los datos de la herramienta
M36	Transportador de rebaba hacia atrás	M84	Modo de control del husillo
M37	Paro del transportador de rebabas	M85	Ajuste de parámetros
M39	Apagado automático	M86	Escribir variable PLC
M40	Bloquear el eje U	M87	Impulsar el estado del sistema de pila
M41	Liberar le eje U	M88	Mantener el estado del sistema de pila
M42	Configuración automática de longitud de herramienta	M89	Mostrar un mensaje
M60	Indicador del estado de reset	M90	Inicio suave
M61	Estado del indicador	M98	Llama una subrutina
M62	Esperar a una señal	M99	Finaliza la subrutina
M64	Saltar		

En un centro de torneado, la función de algunos de los códigos G y M pueden ser diferentes de las funciones correspondientes a un centro de maquinado. Otros códigos no listados para centros de maquinado son de uso exclusivo de los centros de torneado. Varios códigos G y M

no están asignados y pueden ser utilizados por los fabricantes de CNC para funciones especiales de sus máquinas.

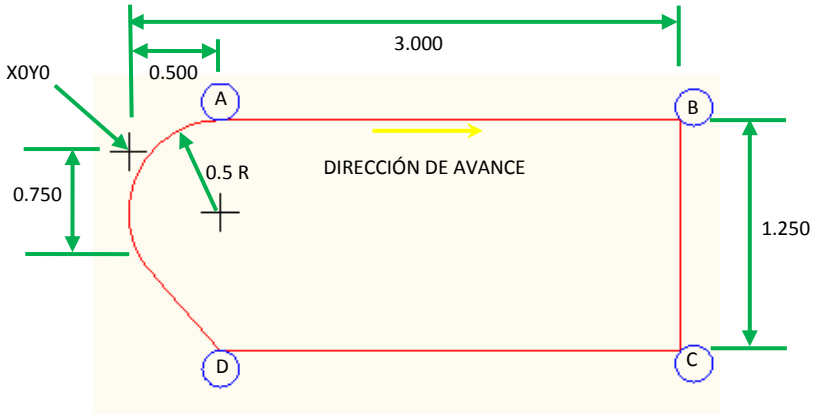


Figura 2.38 Una pieza de muestra para ilustrar procedimientos de programación CNC.

Cada bloque de información debe contener únicamente suficiente información para ejecutar un paso de una operación de maquinado. En la Figura 2.38 la herramienta se mueve primero del punto A al punto B. este bloque se debe escribir como G01 F8.0 X3.0; el movimiento de posición absoluta (G90) será de X.500 a X3.000, a una velocidad de avance de 8.0 pulg/min. El siguiente movimiento es del punto B al punto C, que debe escribirse como Y-1.250, para moverse de Y0 a Y-1.250. Estos dos bloques no pueden ser combinados de la forma G01 F8.0 X3.000 Y-1.250; la unidad de control de la máquina debe ser informada que debe efectuar cada movimiento por separado creando un bloque para cada movimiento. (Krar y Check, 2005)

CAPÍTULO III DESARROLLO DEL SOFTWARE

En el presente capítulo se describe la estructura general de la interfaz desarrollada. Se decidió nombrarlo “GenCoG_3D” de tal manera que el nombre describa su funcionalidad ya que es un Generador de Códigos G en 3D.

3.1 Definición de módulos

El programa o interfaz GenCoG_3D es una aplicación desarrollada dentro del entorno del compilador Visual Studio 2008. Tal aplicación funciona dentro del sistema operativo de *Windows* como se muestra en la Figura 3.1. Se puede apreciar que el sistema operativo puede trabajar simultáneamente con varias aplicaciones diferentes, es decir, es capaz de habilitar varias aplicaciones y estarlas operando en un mismo equipo (PC) controlando el tiempo que el procesador de la PC otorga a cada aplicación para un mayor rendimiento y organización de la PC. También se puede estar trabajando en un software y pasar a otro sin tener que deshabilitar la aplicación anterior, así, se pueden ejecutar diferentes tareas simultáneamente.

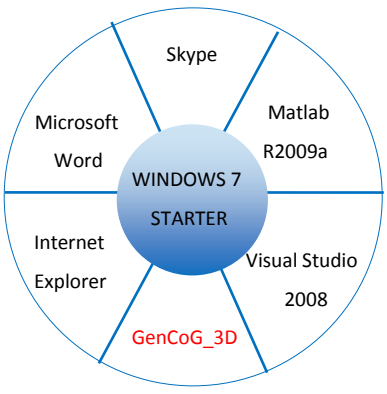


Figura 3.1 Ejecución de diversas aplicaciones en el sistema operativo Windows.

En la Figura 3.1 se muestran varias de las diferentes operaciones que el sistema operativo de Windows puede ejecutar, pero para fines de este proyecto solo se le tomará atención al programa GenCoG_3D que es desarrollado en el compilador de Visual Studio 2008 pero que es ejecutado por el sistema operativo de Windows como se explicó anteriormente.

Para ir adentrando en la forma de operación del programa GenCoG_3D se comenzará por explicar cada uno de los principales módulos que se han implementado para su operación haciendo referencia a la Figura 3.2. En esta figura puede apreciarse que son tres los principales módulos de este programa.

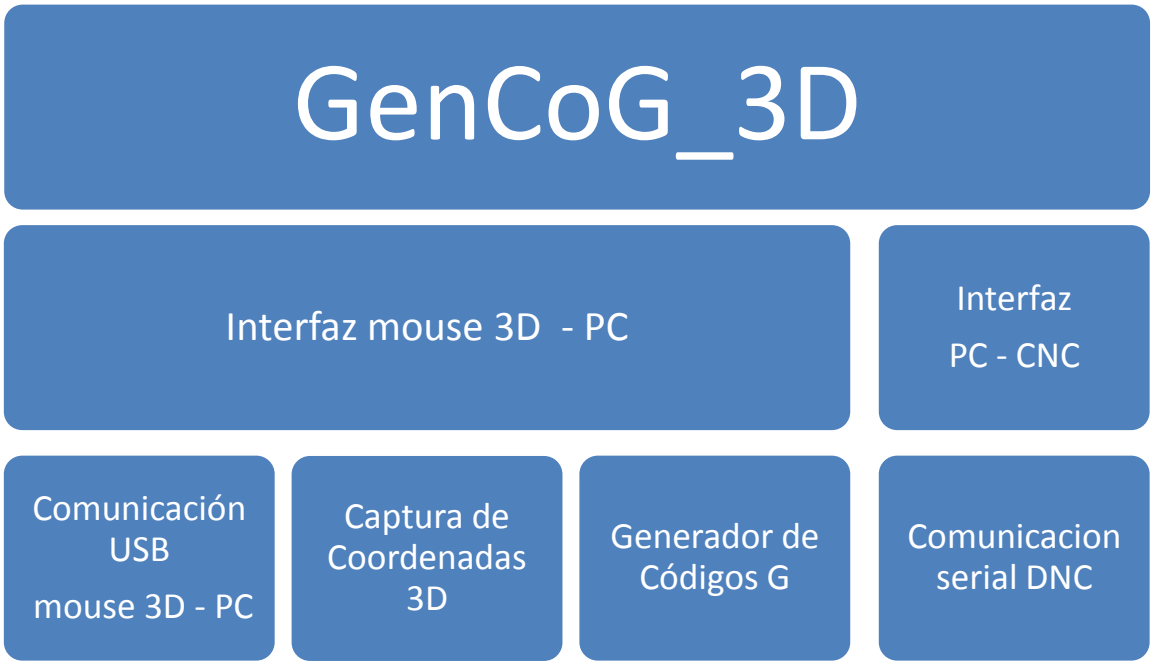


Figura 3.2 Principales módulos del programa GenCoG_3D.

Existen algunas otras funciones adicionales que no se detallarán, aunque no por ello son menos importantes.

El primer módulo es GenCoG_3D. Es el cuerpo principal de la interfaz donde se definen los datos, las estructuras y demás variables empleadas en éste y los otros módulos. Define también todas las funciones necesarias e invoca las bibliotecas y funciones que requiera para operar como

aplicación dentro del sistema operativo *Windows*. También, es la parte que se encarga de crear la ventana principal, que no es otra cosa más que la interfaz óptica, entre el programa y el usuario, para que éste pueda ver lo que está haciendo el programa.

El segundo módulo, llamado "Interfaz *mouse3D*-PC", es un función que se ocupa de establecer la comunicación entre el mouse 3D y la aplicación por medio de un puerto USB disponible en la computadora que hospede la aplicación. Captura las coordenadas, dadas por el usuario, en el dispositivo y hace la conversión necesaria para delimitar el punto por donde se quiere que la trayectoria de la maquina CNC se posicione A este módulo se le da especial importancia ya que es una parte clave del desarrollo de la aplicación (ver subcapítulo 3.2).

Dentro de este módulo, existe una subrutina llamada "Generador de Códigos G", está orientada a la generación de líneas rectas en el espacio o en el plano XY (mesa de la máquina CNC). Estas líneas rectas en lenguaje CNC son interpolaciones lineales G00 y G01. Su funcionamiento se basa en la transformación lineal llamada "Proyección ortogonal" descrita en el capítulo 2, la cual se genera activando o desactivando el botón del eje Z, es decir, si se quiere generar una línea recta en el espacio, se activa el eje Z en el *mouse* 3D (lógicamente los botones de los ejes X y Y están activados), aquí aparece la proyección ortogonal al plano XY. Por ende, si se requiere generar una línea recta solamente en el plano XY el botón del eje Z debe estar deshabilitado. También emplea otras transformaciones lineales como rotación, para sintetizar interpolaciones circulares (códigos G03 y G04) empleados en la generación de trayectorias continuas en conjunto con las interpolaciones lineales.

Finalmente, el tercer módulo es la "Interfaz PC-CNC". Esta componente se encarga que la aplicación GenCoG_3D se comunique con el centro de fresado CNC vía Puerto Serie (RS-232) para el envío de los códigos G y M sintetizados en el módulo anterior.

Cabe mencionar que el orden en que están descritos estos módulos no está relacionado con alguna jerarquía dentro del programa ya que todos son igual de importantes. También, es importante destacar que dicho orden no está de acuerdo a la programación de estos módulos, es decir, las coordenadas 3D no se capturan antes de realizar la transformación lineal.

Para explicar el funcionamiento del programa GenCoG_3D, se tomará como referencia el diagrama a bloques de la Figura 3.6. En esta figura, el primer bloque es llamado "USUARIO", aquí

empieza el diagrama a bloques ya que el usuario ejecuta o cancela el programa. El segundo bloque lleva el nombre de "programa", esto porque para la ejecución de este, se necesita seleccionar dicho programa para que Windows pueda ejecutarlo.

3.2 Desarrollo del módulo *Mouse 3D*-PC

Como ya se definió en la sección anterior, esta componente de la aplicación verifica que los dispositivos USB (*mouse* 3D y cable serial USB-RS232) estén conectados, si este es el caso, el programa continua, de lo contrario, si no están conectados o que estén dañados, el programa debe desplegar un aviso de que el o los dispositivos no están conectados correctamente.

Para establecer la comunicación serial de la PC con la máquina de CNC, se configuraron los siguientes parámetros del puerto RS-232:

TABLA 3.1 Parámetros configurados para la comunicación serial de la PC con la máquina CNC.

Parámetro	Valor
Velocidad de transferencia (*Baudrate*)	115200 bytes/seg.
Tamaño del dato transmitido	7 bits
Paridad	Par
Bits de detención	2 bits

Dicha configuración debe ser la misma en la computadora del CNC a fin de evitar problemas de comunicación.

Si los dispositivos están listos para operar, el programa en la pantalla principal crea botones para el control de los ejes del *mouse* 3D (eje X, Y y Z). Es decir, estos botones tienen como tarea activar o desactivar los ejes X, Y o Z del *mouse* 3D para capturar u omitir las coordenadas de dichos ejes. Permite la captura de datos de entrada de dichos ejes o cancela el funcionamiento de los mismos. Esto es con el fin de facilitar los movimientos en línea recta, por ejemplo, si se desea hacer una línea recta sobre el eje X, sin cambiar de altura, entonces se desactivan los ejes Y y Z, así, se sigue esta misma metodología para cualquiera de las

combinaciones de ejes restantes. Ahora, si se requiere trabajar con los tres ejes simultáneamente, entonces se habilitan todos para operar como sea requerido. Además, esta componente es la encargada de invocar los botones en la pantalla principal, uno de ellos, es el de “Mover Hta”, el cual, al ser activado, crea una ventana en donde pide que se ingresen las coordenadas de referencia de la máquina (conocidas como Coordenadas del Cero Máquina) como se muestre en la Figura 3.3. La información requerida se emplea para calcular el espacio de trabajo aceptable del que dispone la máquina CNC cuando se ejecute alguna interpolación lineal o circular. Cabe mencionar que esta ventana no es desplegada si previamente se introducen dichas coordenadas en el menú de la pantalla principal llamado “Coord. Maq”.

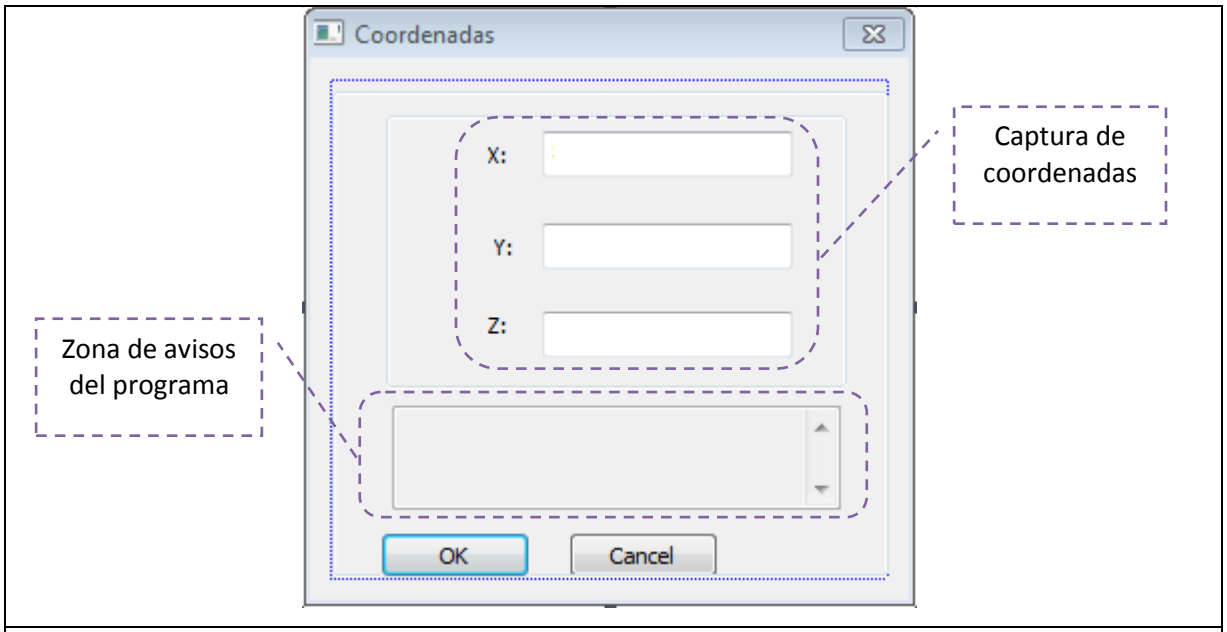


Figura 3.3 **Introducción de coordenadas máquina para el espacio de trabajo disponible.**

En la Figura 3.3 puede apreciarse que el primer campo (*Sample edit box*) es una ventanilla en la que se introduce un valor numérico. Este valor será la coordenada máquina en el eje X, la segunda ventanilla está destinada para el eje Y y la tercera ventanilla está referida al eje Z. La cuarta ventanilla solamente notifica si alguna o varias coordenadas están fuera del espacio de trabajo al momento de presionar el botón “OK”, si esto ocurre, hay que corregir dicha coordenada y volver a presionar el botón “OK” para que las coordenadas sean capturadas y empleadas por el software GenCoG_3D.

3.2.1 Generador de Códigos G

Esta subrutina se ideo para sintetizar los códigos G a partir de las instrucciones y coordenadas del mouse 3D. Si se detecta movimiento en el *mouse* 3D, se comienzan a generar códigos G01 (movimientos rápidos de un punto a otro) que incluyen las coordenadas de X, Y y Z en caso de que estos tres botones estén habilitados, de no ser así, los códigos solo mandarán las coordenadas de los ejes habilitados por DNC a la fresadora (por ende, ésta debe estar en modalidad DNC para que el envío del código surta efecto). Si estas coordenadas están fuera del espacio de trabajo de la fresadora, el programa manda un mensaje de que alguno o varios de los ejes esta fuera del límite y simplemente los códigos para ese o esos ejes no se deben enviar.

Otro botón que se habilita cuando se encuentra activado el botón anteriormente mencionado es “Coord. Iniciales”, el cual, si se presiona aparece una ventana en donde se piden las coordenadas iniciales de la trayectoria, es decir, son las coordenadas relativas de la máquina o Cero Pieza (si es que la fresadora no se ha puesto a cero en la posición actual, la cual debe ser el inicio de la trayectoria). Esto con el fin de que, al iniciar el programa, el seguimiento de la trayectoria no haga un movimiento brusco hasta el cero pieza que pudiera tener guardado la última memoria de la máquina CNC. En caso contrario, es decir, que la fresadora se haya puesto a cero en la posición inicial de la trayectoria, no es necesario introducir estas coordenadas ya que por omisión el programa las considerará como cero.

Siguiendo con la funcionalidad de esta componente, existe otro botón que lleva por nombre “Capturar Tray.” que es el encargado de ir calculando una línea recta, un arco positivo o un arco negativo (que son las interpolaciones lineales y circulares que el programa GenCoG_3D soporta) mediante las coordenadas correspondientes para cada caso, es decir, hay otros tres botones para generar una línea recta (“Línea”), un arco positivo (“Arco +”), el cual es generado en sentido contrario a las manecillas del reloj (referido al sentido en el que se miden los ángulos), o un arco negativo(“Arco -”), el cual es generado al mismo sentido que las manecillas del reloj (con la misma referencia al arco positivo). Solamente puede seleccionarse uno de estos tres botones a la vez, es decir, si se selecciona el botón “Línea” los otros dos botones quedan deshabilitados (“Arco +” y “Arco -”) ya que por lógica en la trayectoria solo se hace una interpolación a la vez. Entonces, para generar una línea, ya sea de trabajo (velocidad de avance controlada), la cual se puede seleccionar con el botón “Trabajo” o de posición (velocidad de avance predeterminada en

el programa), la cual puede seleccionarse con el botón “Posición” el programa pide la coordenada inicial de esta (esta coordenada se captura presionando el botón izquierdo del *mouse* 3D), posteriormente a través del *mouse* 3D se lleva la mesa de la fresadora a la posición requerida y ahí se introduce la coordenada final de la línea recta, con estas dos coordenadas el programa genera el código G01 (el cual corresponde a la creación de una línea recta de trabajo) o la misma estructura para una línea recta pero con el código G00 (el cual corresponde a la creación de una línea recta de posicionamiento rápido). Una vez introducidas las dos coordenadas necesarias (Figura 3.4). En esta figura, el punto P1 representa el punto inicial de la línea y el punto P2 es el punto final de la misma. Para generar el código G de la línea, el botón “Línea” es liberado y los botones “Arco +” y “Arco -” vuelven a quedar disponibles para una nueva tarea. Aquí, si el botón “Trabajo” es habilitado se deshabilita el botón “Posición” y viceversa (estos dos botones son habilitados mediante el botón derecho del *mouse* 3D). Todas estas instrucciones las va almacenando en un buffer para emplearlas posteriormente en la ejecución de la trayectoria. En este punto, si se desea finalizar la trayectoria, se debe presionar el botón “Cargar”, el cual, carga la trayectoria descrita en el portapapeles de la PC y puede ser desplegado por ejemplo, en un documento de texto para posteriormente mandarlo al centro de maquinado CNC mediante una parte del programa GenCoG_3D que lleva por nombre DNC. Aquí, se genera una ventana con un apartado en donde se pega el código G que fue guardado en el portapapeles.

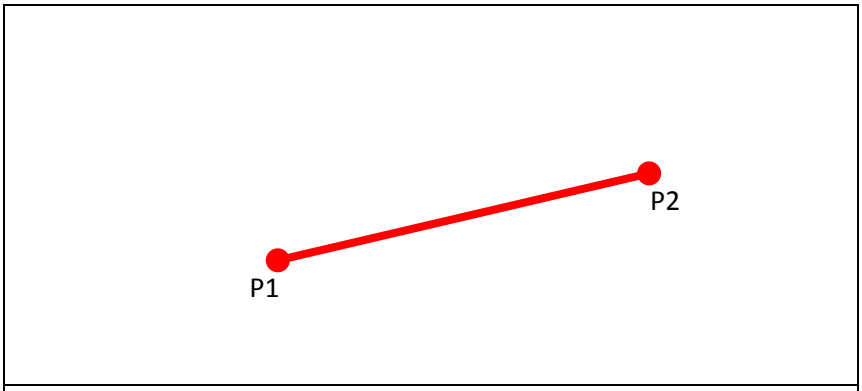


Figura 3.4 Ejemplo de la generación de una línea recta.

Ahora, también puede describirse un arco positivo o negativo habilitando cualquiera de los botones correspondientes a estas dos funciones en lugar de presionar el de línea. Para la

generación del código G (G03 para arco positivo y G02 para arco negativo) del arco, el programa pide tres puntos por los cuales va a pasar el arco (tres coordenadas de entrada). Estos tres puntos se cargan de la misma manera como se mencionó para la línea. Aquí, el programa a través de una función interna calcula el centro de dicho arco ya que este es necesario para el código G, además de la coordenada inicial (primera coordenada dada) y final (última coordenada dada) de dicho arco. Este procedimiento lleva el principio que se menciona en el capítulo 2 en el subtema 2.3.4. Para una mejor noción de cómo se generan los arcos, en la Figura 3.5 se tiene lo que es un arco positivo y un arco negativo, en donde se indica cual es el punto inicial, medio y final para cada caso.

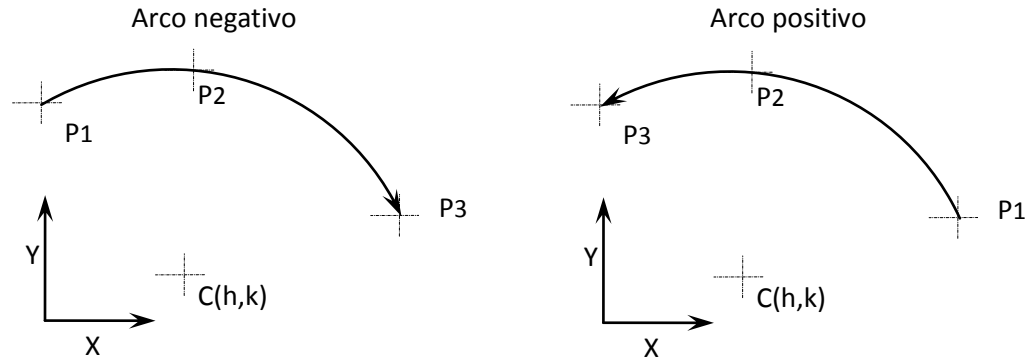


Figura 3.5 **Ejemplos de la generación de un arco positivo y un arco negativo donde P1 es el punto inicial del arco y P3 es el punto final del arco (esto aplica para ambos casos).**

Si se requiere de un arco de trabajo o de posición, se habilitan los mismos botones mencionados para la misma situación de la línea, ya sea para un arco positivo o negativo. En ambos casos, el punto P1 indica el punto inicial del arco, el punto P2 es el punto medio, el punto P3 es el punto final del arco y el punto C es el centro del arco.

Para que el centro de maquinado CNC genere el arco requerido, se sigue el mismo procedimiento descrito para la línea, es decir, se presiona el botón “Cargar” y se guarda el código

en el portapapeles de la PC. Cabe mencionar que los arcos solo se ejecutan en el plano XY ya que el centro de maquinado CNC no soporta interpolaciones helicoidales. También, cabe señalar que si se selecciona el botón de arco positivo, este deshabilita los botones de línea y de arco negativo. Lo mismo pasa para un arco negativo, deshabilita los otros dos botones hasta concluir con dicha operación.

Una vez presionado el botón "Cargar" éste deshabilita el botón "Mover Hta." Para liberar el puerto serial de envío de información para no saturar dicho puerto y que a la hora de mandar los códigos a través del DNCManager.exe no arroje basura o se tenga un problema de comunicación.

Si se desea generar otra trayectoria, se puede volver a habilitar el botón "Mover Hta." y realizar el procedimiento antes mencionado.

Ahora bien, si la trayectoria requerida es más compleja, es decir, si se requieren líneas de trabajo o de posición y arcos de trabajo o de posición, además de ir intercalando líneas con arcos positivos o negativos y líneas en el espacio o en el plano es posible realizarla, ya que los códigos G, como ya se mencionó anteriormente, no se ejecutan hasta que se presiona el botón "Cargar". Por ejemplo, deben estar habilitados los botones de los ejes con los que se va a trabajar, se debe presionar el botón de mover herramienta, y mover la mesa del centro de maquinado CNC a la posición requerida, estando ahí, se presiona el botón de capturar trayectoria y posteriormente se presiona el botón correspondiente a la operación deseada, para este ejemplo se escogió una línea de trabajo. Entonces, se presiona el botón de línea, se presiona el botón de trabajo y se presiona el botón derecho del *mouse* 3D para capturar la primera coordenada de la línea, ya sea en el plano (solamente habilitados los ejes X y Y) o en el espacio (habilitando los tres ejes) posteriormente se mueve la mesa de la fresadora a la coordenada final de la línea, independientemente de estar siguiendo una línea recta o una curva, ya que el programa al final tiene que hacer una línea recta por el comando seleccionado, lo mismo pasa para cualquiera de las opciones de arco. Una vez que se llego al punto final de la línea recta, se introduce esta coordenada con el botón de *mouse* 3D. Los botones de arco y línea quedan de nueva cuenta disponibles, ahora se selecciona un arco negativo de posición. Aquí cabe mencionar que el programa toma como primera coordenada del siguiente objeto (línea, arco positivo o arco negativo) la última del objeto anterior a partir del segundo objeto y de aquí en adelante hasta que

se concluya la trayectoria será así, pero aun así, debe ser confirmada la primera coordenada del objeto seleccionado con el botón derecho de *mouse* 3D, por lo tanto, en este caso para el arco negativo se necesitan tres coordenadas por lo que debe pulsarse tres veces el botón del *mouse* 3D que carga dichas coordenadas, recordando que el último punto o coordenada introducida independientemente de por donde se halla llevado la mesa de la centro de maquinado CNC representa el final del arco. Si una vez introducidas las coordenadas del arco se presiona el botón de cargar, aquí finaliza la captura de la trayectoria y se libera el botón de mover herramienta, por ende, quedan deshabilitados el resto de los botones hasta que se vuelve a presionar el botón "Mover Hta.". Se guarda en un archivo de texto el código generado y se ejecuta el DNCManager.exe para posteriormente mandar dicho código al centro de maquinado CNC sin ningún problema.

Otra característica importante del programa es que por default, independientemente del tipo de trayectoria que sea generada, al principio del código se envían tres líneas de código, las cuales levantan el eje Z 5cm de la altura final con el propósito de librar obstáculos que puedan existir en el trayecto de regreso al inicio de la trayectoria (recuérdese que la posición actual que se tiene de la mesa de la fresadora es el punto final de la trayectoria a realizar), posteriormente se mueve la mesa en el plano XY hasta llegar a las coordenadas iniciales de la trayectoria y finalmente baja el eje Z a la altura inicial de la trayectoria para comenzar con el código de dicha trayectoria. Los movimientos anteriores en el eje Z se hacen con códigos G00, es decir, son movimientos rápidos ó de posición y los movimientos en el plano XY se hacen a una velocidad de 500 m/min.

Entonces, siguiendo con el ejemplo, al momento de mandar el código al centro de maquinado CNC el eje Z se levantará, la mesa se moverá a las coordenadas iniciales de dicha trayectoria y el eje Z bajará a la altura inicial de la trayectoria rápidamente. Posteriormente, se generará una línea recta del primer punto al segundo punto dado, con una velocidad moderada y el arco lo hará con una velocidad default (arco de posición), aquí acaba el código G generado para este ejemplo.

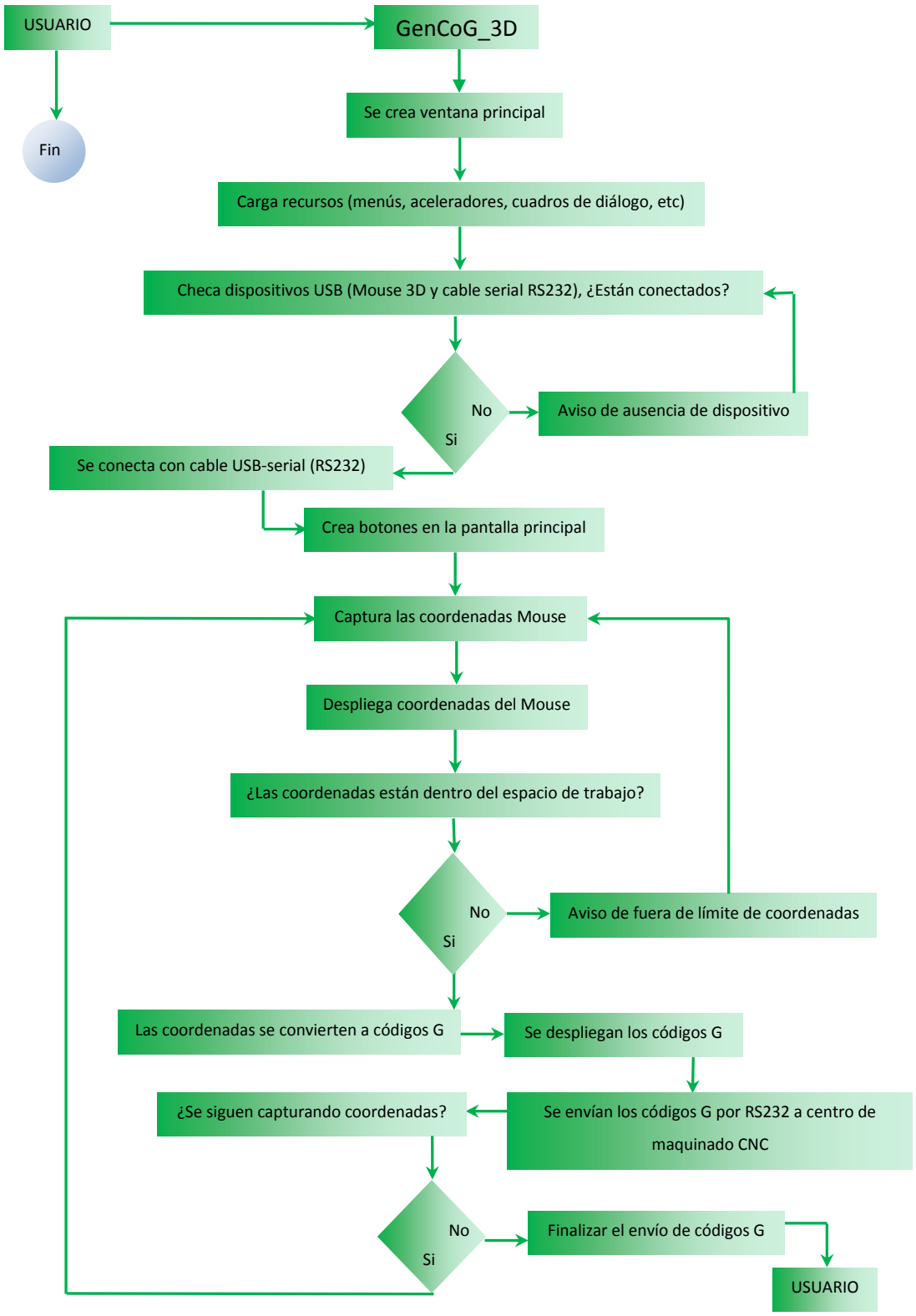


Figura 3.6 **Diagrama a bloques del programa GenCoG_3D.**

Después de haber creado los botones en la pantalla principal, el usuario, puede comenzar a trabajar con el *mouse 3D* y es aquí cuando las coordenadas del *mouse 3D* son desplegadas en la pantalla principal del programa para que se tenga noción de hasta qué punto se moverá la mesa, y/o husillo del centro de maquinado CNC. Por otra parte, el programa realiza la detección y configuración del puerto serial, de no ser así, envía un aviso de error en la comunicación serial por lo que el usuario debe ir a uno de los menús de la pantalla principal llamado "Dispositivo" en el cual, habrá un submenú llamado "RS232", el cual desplegará una ventana en donde se configurará el puerto serial, es decir, checa los puertos COM disponibles y se puede seleccionar alguno de ellos para el cable serial RS232, y el cable es conectado al programa, y posteriormente, se definen los parámetros, tales como el número de bits de inicio de transmisión (bit de *start*), el número de bit de datos, la paridad, el número de bits de paro y la velocidad de transferencia de datos (estas configuraciones deben de ser idénticas entre la máquina CNC y la PC para una buena comunicación). Cuando se concluye con estas configuraciones, se guardan estos cambios con un botón de aceptar que se encuentra en esta ventana o cancelar si no se desean guardar los cambios. Después de esto, el puerto serial está listo para operar, todo esto en caso de que no lo haga automáticamente el programa en un principio, es decir, cuando se ejecuta debe conectar el puerto serial.

3.3 Desarrollo del módulo PC-CNC

Una vez que se han generado o sintetizado los códigos G, se envían a través del cable serial USB-RS232 a la máquina CNC. Esta máquina debe estar en modo DNC para la recepción de los datos enviados por el programa y así, poder controlar dicha máquina desde el programa GenCoG_3D, de otro modo, aunque el programa este enviando datos, la máquina CNC simplemente no hará nada. Para poner en modalidad DNC a la máquina CNC, el usuario debe hacerlo manualmente desde *Manual Data Input* (MDI, por sus siglas en inglés) desde la máquina CNC. El programa GenCoG_3D no puede hacerlo por sí mismo porque si no se encuentra previamente la máquina CNC en modalidad DNC ningún dispositivo externo puede tener control sobre la misma.

Posteriormente, el programa GenCoG_3D si sigue recibiendo coordenadas, es decir, si sigue detectando movimiento en el *mouse* 3D vuelve a checar que las coordenadas capturadas de éste estén dentro del límite de trabajo de la máquina CNC y vuelve a hacer el procedimiento descrito anteriormente desde este punto, obviamente sin volver a configurar el puerto serial. Si ya no captura más coordenadas, el programa se mantendrá en la última posición descrita por el *mouse* 3D y mandará finalizar la comunicación DNC de la máquina CNC para que ésta no se quede esperando datos en esta modalidad y evitar que marque algún error por falta de entrada de datos. Por último, el programa GenCoG_3D se mantendrá esperando a que el usuario vuelva a enviar datos a la máquina CNC o finalice la ejecución del programa.

3.4 Integración y caracterización

El *mouse* 3D, como ya se mencionó en el subtema 2.5.3 tiene por cada eje un movimiento máximo de 500 unidades a partir de su punto neutro u origen, es decir, desde cero hasta su desplazamiento máximo positivo o negativo en el eje X, Y ó Z es de 500 unidades o cuentas. Cada una de estas cuentas en la centro de maquinado CNC, dependiendo en qué sistema se encuentre operando dicha máquina, ya sea en el sistema inglés (pulgadas) o en el sistema métrico (milímetros) representan una pulgada o un milímetro respectivamente. Para las pruebas realizadas y descritas en el siguiente capítulo, el centro de maquinado CNC siempre estuvo operando bajo el sistema métrico, por lo que en este caso, cada cuenta del *mouse* 3D valdría 1 mm pero, como el *mouse* 3D tiene una sensibilidad muy alta, es decir, detecta el más mínimo movimiento y si el usuario no está muy familiarizado con este dispositivo, lo más seguro es que pueda sacar del espacio de trabajo al centro de maquinado CNC en cuestión de segundos ya que las coordenadas enviadas al centro de maquinado CNC son incrementales, es decir, si el usuario mueve el *mouse* 3D hasta llegar a tope alguno o varios de los ejes estando en modalidad de mover herramienta va a aumentar el desplazamiento de la mesa de dicha fresadora 500 mm a partir de la última coordenada que haya capturado, y para tener una mejor referencia de esto, el CNC en la que se realizaron las pruebas para GenCoG_3D tiene a partir del cero máquina (*home*) en el eje X una longitud de carrera de 1100 mm, en el eje Y tiene por carrera una longitud de -560mm y en el eje Z su carrera tiene -590mm de longitud (los signos negativos son porque a partir de la posición *home* de la máquina, la convención de sus ejes es negativa). Y a pesar de que esta

máquina tiene sensores de límite de carrera, puede dañarse por la fuerza del impacto. Es por eso que para prevenir este tipo de accidentes, en el programa GenCoG_3D se caracterizan las unidades del *mouse* 3D, es decir, cada cuenta recibida por dicho dispositivo en el eje X y Y es dividida entre una constante, la cual tiene un valor de 25, ya que con las pruebas realizadas y descritas en el siguiente capítulo se fue adquiriendo experiencia sobre dicha caracterización quedando como mejor valor el de 25; con esto entonces, cada cuenta recibida ya no vale numéricamente 1 mm sino que ahora su valor numérico es de 1/25 mm, es decir, 0.040 mm, con esto, se logra un mayor control sobre la mesa del centro de maquinado CNC en los ejes X y Y, esto representa la reducción de la sensibilidad del *mouse* 3D.

Ahora bien, el eje Z también está caracterizado para evitar los mismos accidentes que pueden ocurrir en el plano XY y adicionalmente evitar impactos de la herramienta en la mesa del centro de maquinado CNC. Para la caracterización de este eje no se dividieron las cuentas del *mouse* 3D entre la misma constante que se empleó en los ejes X y Y, esto porque con forme se realizaban las pruebas descritas en el siguiente capítulo se fue observando que este eje es el más difícil de controlar, ya que como se mencionó con anterioridad, si no se controla totalmente este eje, se puede llegar a impactar la herramienta con la mesa del centro de maquinado CNC y hasta se puede llegar a dañar el husillo, por tal motivo, y a través de la experiencia adquirida, se llegó a la conclusión de tomar una constante de 100 para dividir cada cuenta del *mouse* 3D y tener controlado de una forma más segura el eje Z; con esto, cada cuenta de dicho dispositivo de entrada para el eje Z ya no es de 1mm sino que ahora, con esta caracterización cada cuenta del *mouse* 3D para el centro de maquinado CNC tiene un valor numérico de 1/100 mm, es decir, cada cuenta representa 0.010 mm, con esto, se reduce la sensibilidad del *mouse* 3D y así, quedan completamente caracterizadas las coordenadas de entrada del *mouse* 3D.

En la Tabla 3.2 se resumen las constantes de conversión obtenidas por la caracterización de las cuentas del Mouse 3D a unidades del sistema métrico (mm).

TABLA 3.2 Constantes de conversión para la caracterización del Mouse 3D.

Eje de movimiento	Unidades (mm)
Eje X	1/25
Eje Y	1/25
Eje Z	1/100

Con estas caracterizaciones para los tres ejes, se llegó a la conclusión de que si las constantes divisoras eran muy grandes, se saturaba de información el centro de maquinado CNC, porque para llegar a un punto muy próximo al último, se necesitaban de muchas líneas de código, por lo que se perdían líneas de dichos códigos y además, se avanzaba muy poco con bastantes líneas generadas, y si estas constantes eran muy chicas, se perdía el control de la centro de maquinado CNC por la sensibilidad muy elevada del *mouse* 3D.

Otra caracterización para el programa GenCoG_3D fue el tiempo del temporizador que envía los códigos G cuando se está moviendo la herramienta, es decir, por medio de un temporizador se envían datos cada cierto tiempo. Para estos movimientos, como ya se mencionó, se envían códigos G00 para cada cambio que se tiene del *mouse* 3D, por tal motivo el centro de maquinado CNC cada que recibe una nueva coordenada, acelera inmediatamente hasta alcanzar la velocidad máxima dada por este comando hasta llegar a la coordenada final, y ahí desacelera bruscamente hasta quedar en reposo, pero como sigue recibiendo datos, vuelve a acelerar y a frenar hasta que el usuario deja de imprimir movimiento en el *mouse* 3D. Esto provoca que se sienta un golpeteo y jaloneo en la mesa del centro de maquinado CNC, a través de las pruebas realizadas y descritas en el capítulo siguiente, se mejoró el tiempo del temporizador del envío de estos datos, es decir, se fue mejorando el intervalo de tiempo con el que se mandan los datos (códigos G) a la fresadora. Este tiempo se fue reduciendo, ya que entre más grande era el intervalo de tiempo de espera del programa para enviar los datos, el incremento de las coordenadas actuales del *mouse* 3D era muy grande en comparación con las anteriores; por tal motivo, se perdía el control de movimiento de la mesa del centro de maquinado CNC, ya que se movía bruscamente intentando alcanzar las últimas coordenadas recibidas. Esto se debía a que las velocidades de avance del CNC son relativamente bajas a comparación de la velocidad de respuesta del mouse 3D por que dicho dispositivo no toma en cuenta la inercia de los ejes de la máquina CNC. Conforme se fue reduciendo el tiempo de espera del programa, se fue notando que dichos golpeteos y jaloneos se iban reduciendo, pero si se bajaba mucho este tiempo del temporizador, el centro de maquinado CNC perdía también líneas de código por el poco tiempo que se le daba para que las ejecutara y con esto, también era saturada la memoria del centro de maquinado CNC; por lo que al final se concluyó que el tiempo de espera para el envío de códigos en la modalidad de mover herramienta debía de ser de 180 milisegundos.

CAPÍTULO IV PRUEBAS Y ANÁLISIS DE RESULTADOS

4.1 Análisis Comparativo y Discusión

Para confirmar la efectividad del sistema desarrollado se implementaron una serie de experimentos para evaluar el desempeño del mismo.

Algunas de las primeras pruebas realizadas se diseñaron con líneas rectas, ya que el usuario en un principio no está familiarizado con el programa GenCoG_3D y con la máquina, le resultaría difícil seguir con un margen de error mínimo líneas curvas, es por eso que se opto por comenzar las pruebas con líneas rectas.

En los subcapítulos siguientes se muestra la experimentación efectuada.

4.1.1 Problemas con la comunicación serial

Además de las pruebas que se realizaron, con la ayuda de un osciloscopio se hicieron otras tantas pruebas para determinar las señales faltantes para la ejecución de los códigos G en el centro de maquinado CNC, es decir, una vez que se logró generar el código G en el programa GenCoG_3D, se investigó en el manual de control de la máquina y programación *DYNA M4*, versión 2.2 que para la comunicación entre el centro de maquinado CNC y una PC externa se necesitaba de una conexión especial conocida como *RTS/CTS* (*hardware handshaking*) que significa intercambio de hardware, la cual consiste en la conexión que se muestra en la Figura 4.1 para que exista la correcta comunicación entre la PC y el centro de maquinado CNC en modalidad DNC.

Figura 4.1 **Conexión para la comunicación con el centro de maquinado CNC.**

En la figura anterior se puede observar la conexión necesaria para que exista una comunicación DNC del centro de maquinado CNC con una PC; se aprecia el número de pines que deben utilizarse en un DB-9 para ambos extremos (DNC y PC) y como deben ir conectados; las siglas de éstos pines fueron mencionadas y descritas en la Figura 2.23. Una vez conocida la conexión faltante, se prosiguió a realizar la manufactura de esta conexión mediante dos terminales DB-9 hembras; este cable se conecto entre la terminal serial del cable serial USB y el centro de maquinado CNC, esto porque no se lograba el envío de los códigos; una vez teniendo este cable y conectándolo como ya se mencionó, se conecto la PC con la centro de maquinado CNC pero aún así la comunicación no se logró. Posteriormente, se opto por analizar las señales mandadas por el DNCManager.exe para determinar si hacía falta algún pulso en las señales mandadas u optar por otra opción causante del problema. Esta prueba se realizó con la ayuda de un osciloscopio conectándolo al cable manufacturado del lado de la PC, analizando la señal del pin 3 (TD) y el pin 2 (RD). Entonces, fue cuando se mando una línea de código G a través del ejecutable mencionado anteriormente y después de analizar los pulsos de la señal mandada por dicho ejecutable al centro de maquinado CNC se determinó que el código binario bajo el protocolo RS232 que se mandaba después de cada línea pertenecía al número 13 en código ASCII (*American Standard Code for Information Interchange*) representando un "*enter*" o retorno de carro, seguido del número 10 que también en código ASCII representa un salto de línea. Después de esta investigación, se añadió a cada fin de línea de código del programa GenCoG_3D un 13 y un 10, fue así como el centro de maquinado CNC comenzó a ejecutar el código G enviado por este

programa. En la Figura 4.2 se muestra una imagen con la PC conectada al centro de maquinado CNC generando los pulsos faltantes antes mencionados para que la ejecución de los códigos G se llevara a cabo.

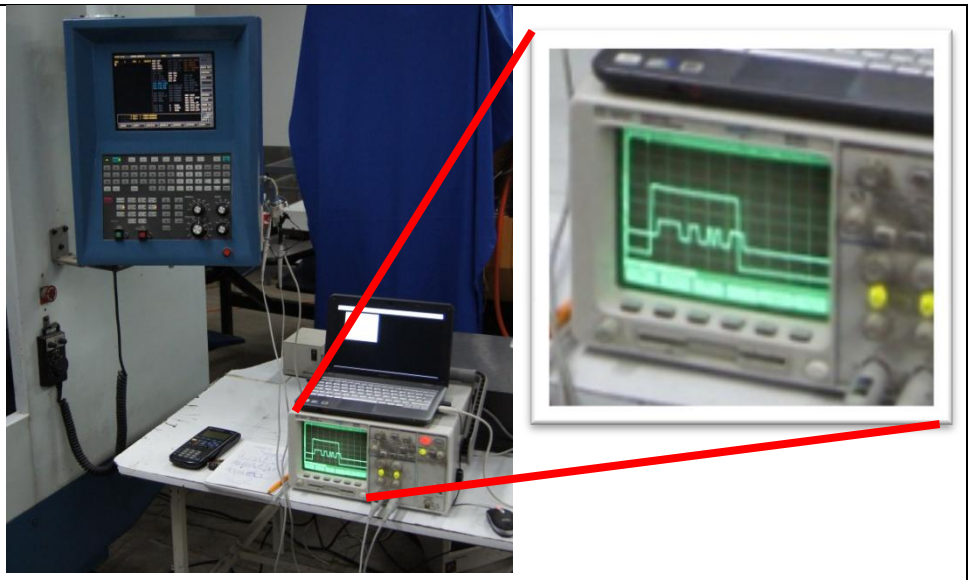


Figura 4.2 Análisis del código G enviado al centro de maquinado CNC.

Como se aprecia en la figura anterior, en el osciloscopio se muestran dos señales, la que contiene varios pulsos es la señal de envío de datos (pin 3, TD) y la señal en la que se ve solamente un pulso cuadrado es la señal que se recibió del centro de maquinado CNC (pin 2, RD). En la primera señal se observan los pulsos que representan parte del código G enviado a la fresadora. Con este análisis se pudo enviar y ejecutar el código G en el centro de maquinado CNC.

4.1.2 Diseño del trazador

Una vez resuelto el problema de la comunicación, se siguió mejorando y resolviendo problemas de diseño del programa GenCoG_3D. Es decir, se integraron los botones para configurar la trayectoria a seguir, las ventanas de configuración, los cálculos para generar los arcos y líneas, entre otras tantas cosas; cuando se concluyó con esta parte, se prosiguió a la

realización de pruebas en el programa. Las pruebas realizadas se hicieron con figuras impresas en hojas de papel, puestas y pegadas sobre la mesa de la centro de maquinado CNC (las primeras pruebas se realizaron con rectángulos y posteriormente se hicieron con círculos). El procedimiento a seguir fue hacer un trazador para una plumilla, esto para poder sujetar la plumilla al husillo de la fresadora de tal manera que estuviera firme y que no sufriera algún daño durante la prueba. Se presenta el plano de diseño en la Figura 4.3 y una imagen de cuando se estaba manufacturando en la Figura 4.4.

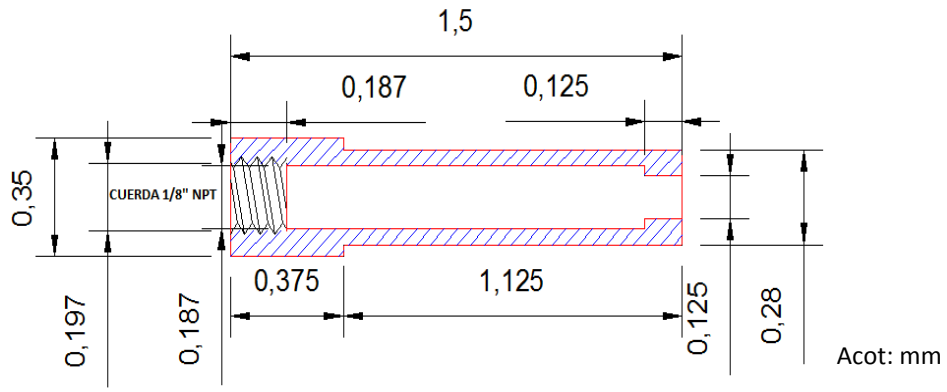


Figura 4.3 Plano del trazador de una plumilla (corte transversal).

En la Figura 4.5 se muestra el porta-herramientas con el popote de tinta adentro (plumilla). Este sujetador se colocó en el husillo del centro de maquinado CNC como se muestra en la Figura 4.6 con una boquilla y un porta herramienta empleados para la manufactura en dicha centro de maquinado CNC. Con el sujetador colocado en el husillo del centro de maquinado CNC y la hoja de prueba fija en la mesa del centro de maquinado CNC se fue marcando la trayectoria seguida con el *mouse* 3D en línea (rojo) con la modalidad mover herramienta y capturar trayectoria del programa GenCoG_3D para ir capturando los puntos clave tanto para líneas como para arcos.

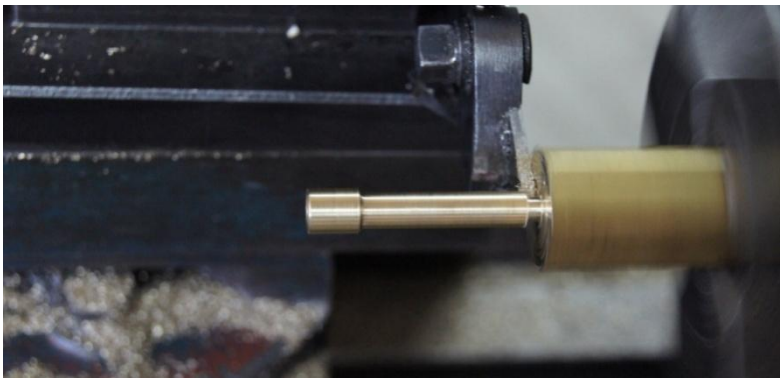


Figura 4.4 Manufactura del trazador.

Posteriormente, cambiando el color de la tinta (azul) se trazó la trayectoria programada a través de los puntos dados cargados por medio del ejecutable DNCManager.exe, es decir, el código G generado por GenCoG_3D se pasó al ejecutable en curso y de ahí se mando al centro de maquinado CNC, la cual siguió la trayectoria descrita.

Figura 4.5 Porta herramientas con trazador y plumilla.

En los subcapítulos siguientes se presentan las pruebas con sus respectivas mediciones. Todas las pruebas fueron realizadas sin ser alineadas con los ejes de la máquina (X y Y), esto para comprobar que se puede seguir cualquier trayectoria con el programa GenCoG_3D sin tener que

ser preparada previamente, es decir, el usuario puede llegar inmediatamente a colocar, en este caso las hojas de prueba, en cualquier posición y comenzar a generar la trayectoria solicitada.

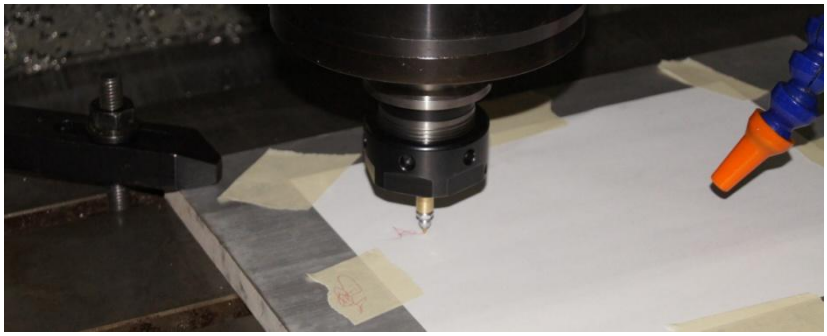


Figura 4.6 Trazador colocado en el husillo del centro de maquinado.

4.1.3 Trazo y seguimiento de trayectorias lineales

Una de las primeras pruebas que se realizaron para el análisis de la operación del programa GenCog_3D es la que se muestra en la Figura 4.7, la cual es una figura rectangular seccionada cada diez milímetros para hacer las mediciones de error en estas distancias. El color rojo representa la trayectoria hecha en línea (tiempo real ó trayectoria manual) y la trayectoria azul es la generada con el programa GenCoG_3D, es decir, esta última trayectoria ya es realizada con el código G generado siguiendo dicha trayectoria con el *mouse* 3D (fuera de línea ó trayectoria CN).

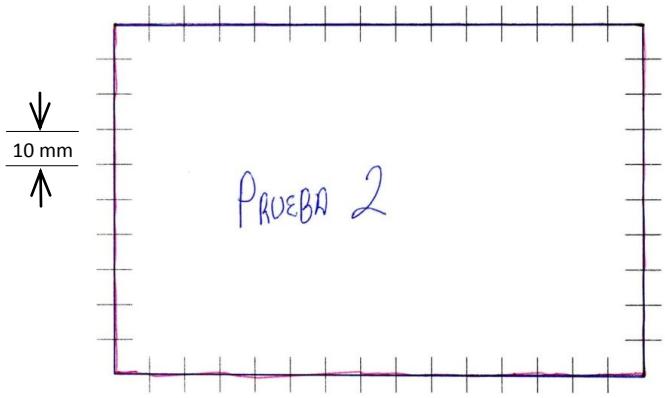


Figura 4.7 **Ejemplo de seguimiento de una trayectoria rectangular.**

En la Figura 4.8 se muestra el código G sintetizado generado por GenCoG_3D para generar la trayectoria CN descrita en la Figura 4.7 comenzando en la esquina superior izquierda; recuerde que al principio de cada código G sintetizado se generan líneas que mueven al eje Z hacia arriba para librar obstáculos de la trayectoria descrita y posteriormente regresar a las coordenadas de inicio de la trayectoria para que en este punto se baje el husillo hasta completar las coordenadas iniciales en los ejes X, Y y Z de la trayectoria a realizar.

Archivo Edición Formato Ver Ayuda

```
PRUEBA 2 RECTANGULO
G00 Z50.000;
G00 X0.000 Y0.000;
G01 Z0.000 F1000;
G01 X-5.840 Y-99.320 Z0.000 F1000;
 X144.320 Y-109.360  ;
 X149.960 Y-8.880  ;
 X-0.000 Y0.040  ;
```

Figura 4.8 **Ejemplo del código G creado para el seguimiento de una trayectoria rectangular (tome como referencia la Figura 4.7).**

Como puede observarse en la figura anterior existen siete líneas de código G generado por GenCoG_3D para realizar la trayectoria sintetizada (CN) de la prueba rectangular mostrada en la Figura 4.7; a continuación se describen cada una de estas líneas (vea Figura 4.7):

En la línea 1 se usa un movimiento rápido sobre el eje Z, esto, como ya se mencionó, es para librar los obstáculos que se llegaran a encontrar en el recorrido de la trayectoria; por lo que el husillo de la fresadora se levanta 50 milímetros para evitar algún impacto con dichos obstáculos.

La línea 2 del código es otro movimiento rápido para mover el husillo de la fresadora a las coordenadas iniciales de la trayectoria que previamente fue guardada como se explicó en capítulos anteriores. Este movimiento se hace sobre el plano XY, es decir, se conserva la altura generada por la línea de código anterior.

La línea 3 es para bajar el husillo de la fresadora, pero en este caso se realiza este movimiento con un G01, que es un movimiento rectilíneo con una velocidad controlada; con este movimiento el husillo se posiciona en la altura inicial de la trayectoria.

Con la línea 4 del código se mueve le husillo hasta la esquina inferior izquierda de la prueba con una velocidad controlada, cabe mencionar que el valor de la coordenada X difiere de cero puesto que la hoja de la prueba fue colocada sobre la mesa de la fresadora sin ser alineada para demostrar que con el programa GenCoG_3D puede generarse la trayectoria deseada sin que se esté alineada con la mesa de la fresadora.

Ahora el husillo se mueve de la esquina inferior izquierda a la esquina inferior de recha con la línea 5 del código, nótese que en esta línea solamente aparecen las coordenadas del punto al que llegara el husillo y que la velocidad de avance junto con el tipo de código no aparecen, esto porque para llegar al siguiente punto se utiliza un movimiento similar al utilizado en la línea anterior, se dice similar porque aunque se use el mismo comando y velocidad de avance, las coordenadas son diferentes. Por tal motivo es posible omitir la escritura del tipo de código y la velocidad de avance siempre y cuando esta ultima sea la misma que la anterior escrita. Este tipo de código como ya se mencionó en capítulos anteriores es posible omitir su escritura hasta que se utilice otro comando diferente, lo que hace mas rápido y ligero (menos información enviada de la PC al centro de maquinado CNC) al código generado para realizar la trayectoria.

La línea 6 es similar a la línea 5 y es generada para mover el husillo a la esquina superior derecha de la prueba.

Por último la línea 7 se genera para llegar a la esquina superior izquierda y completar así la trayectoria cargada al programa GenCoG_3D.

120			PRUEBA 2: RECTÁNGULO		
			Relación milímetros/píxeles: 10 mm = 120 píxeles		
	TRAYECTORIA MANUAL			TRAYECTORIA CN	
	píxeles		Milímetros	Píxeles	Milímetros
Medición 1:	7		0.583	0	0.000
Medición 2:	5		0.417	0	0.000
Medición 3:	5		0.417	0	0.000
Medición 4:	8		0.667	0	0.000
Medición 5:	3		0.250	0	0.000
Medición 6:	5		0.417	0	0.000
Medición 7:	7		0.583	3	0.250
Medición 8:	3		0.250	5	0.417
Medición 9:	5		0.417	7	0.583
Medición 10:	0		0.000	2	0.167
Medición 11:	3		0.250	3	0.250
Medición 12:	3		0.250	1	0.083
Medición 13:	3		0.250	3	0.250
Medición 14:	2		0.167	4	0.333
Medición 15:	2		0.167	4	0.333
Medición 16:	5		0.417	5	0.417
Medición 17:	2		0.167	5	0.417
Promedio:			0.333		0.206

Figura 4.9 Mediciones de error de la segunda prueba rectangular.

Posteriormente, en la Figura 4.9 se muestra la medición de los errores (manual y CN) de dicha prueba; esta medición se hizo escaneando la hoja de la prueba realizada y se hizo la relación de los pixeles equivalentes a diez milímetros (véase Figura 4.7), de aquí se derivan el resto de las mediciones, las cuales fueron realizadas cada 30mm a partir del punto de inicio de las trayectorias. Para la medición de cada error (pixeles) de cada punto se cuenta el número de pixeles desde el centro de la trayectoria original (línea negra) perpendicularmente siempre a dicha trayectoria hasta el centro de la línea de la trayectoria manual o CN, ya sea la línea roja o azul respectivamente. El punto de inicio es el punto superior izquierdo de la figura prueba. En la figura anterior, también se observa un número en la parte superior izquierda, el cual representa el número de pixeles que hay por cada 10mm en este caso; cada prueba tiene su propia relación pixeles/milímetros. También, se aprecia en letra roja una leyenda que dice: “TRAYECTORIA MANUAL”, estas dos columnas debajo de este texto representan el valor del error manual, es

decir, el error generado en línea por el usuario a través del software GenCoG_3D y las dos columnas que están debajo de la leyenda en color azul que dice: “TRAYECTORIA CN” representan el error generado por los códigos G del software GencCoG_3D, es decir, es el error de control numérico. Estos errores se muestran en pixeles y milímetros, de los cuales, las mediciones en milímetros son las utilizadas para las gráficas posteriores.

En la Figura 4.10 se muestran las gráficas de cada una de las pruebas realizadas con este rectángulo donde se aprecian dos líneas por cada prueba.

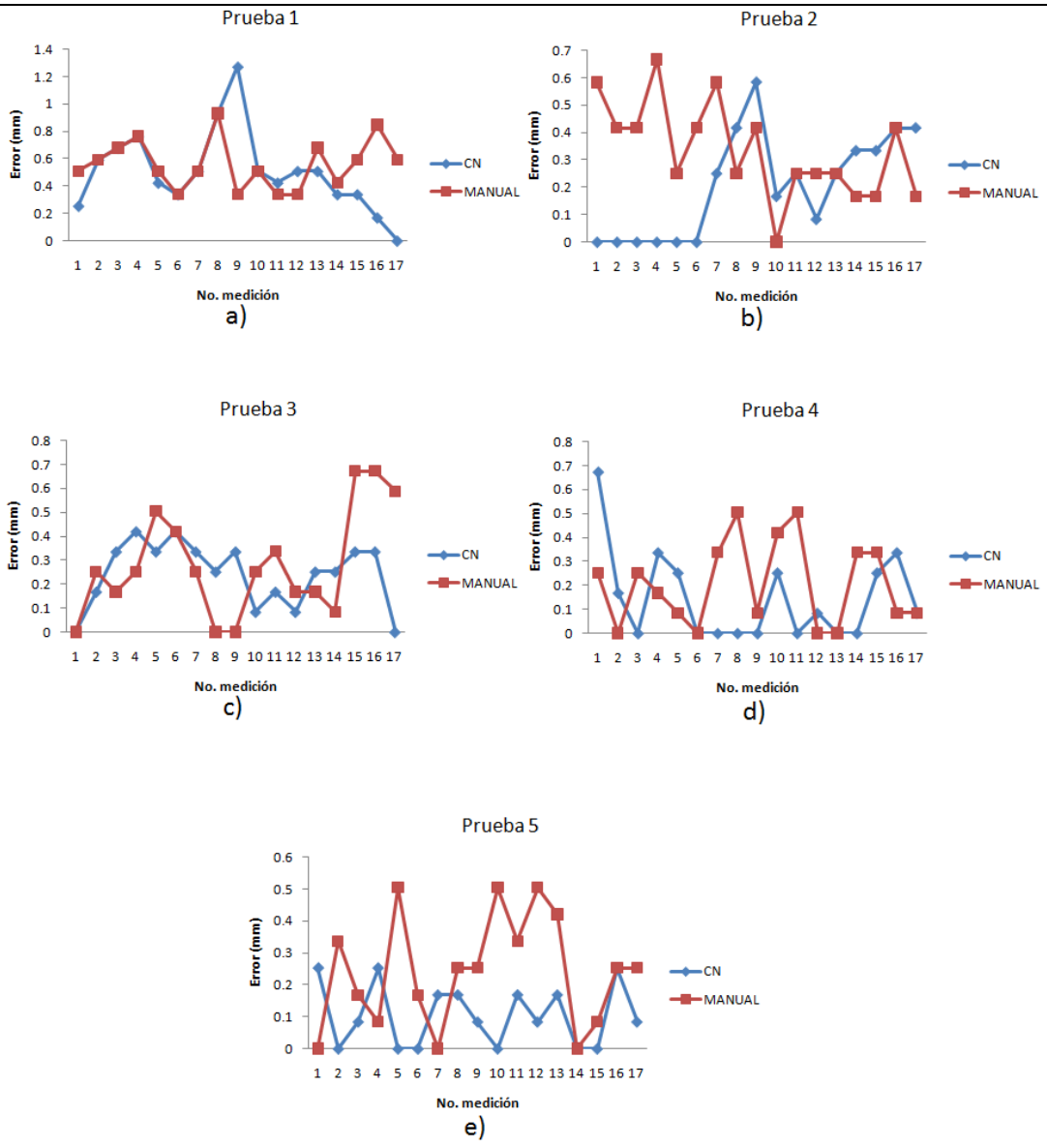


Figura 4.10 Gráfica que muestra el error CN y el error manual de cada prueba rectangular.

En la Figura 4.10a se aprecia la gráfica en color rojo que representa el error manual de cada una de las mediciones en la primera prueba, es decir, en el primer rectángulo; al igual que la gráfica en color azul que representa el error CN también para cada medición del primer rectángulo. Es decir, para cada figura rectangular en la que se hicieron estas pruebas (Figura 4.7, la cual representa la prueba No. 2 de 5) se obtuvieron 17 mediciones (eje horizontal en las graficas de la Figura 4.10) cada 30 mm a partir del punto inicial de la prueba, las cuales se grafican contra la magnitud del error calculado (eje vertical en las graficas de la Figura 4.10). También puede observarse que el error manual y el error CN son muy similares, es decir, el error entre ellos es mínimo y en la mayoría de las pruebas el error manual se mantiene por debajo de los 0.8mm y para el error CN en ciertos puntos es cero.

Para la segunda prueba rectangular, referida a la Figura 4.10b la analogía es similar a la descrita para la Figura 4.10a. El eje horizontal representa lo mismo que en la figura anterior al igual que el eje vertical, y estas analogías serán las mismas para las tres pruebas rectangulares restantes (Figuras 4.10c – 4.10e), esto incluye el color de las líneas, el cual, el color rojo representa el error manual y el color azul representa el error CN.

En la segunda prueba (Figura 4.10b), se aprecia que el error manual aumentó un poco referido a la primera prueba, mientras que el error NC bajó considerablemente en las primeras mediciones también tomando como referencia la primera prueba; pero entre estos dos errores, al final casi se empatan.

Para la tercera prueba (Figura 4.10c), el error CN y el error manual están muy parecidos y bajando en ambos casos la magnitud del error.

En la prueba número cuatro (Figura 4.10d), se puede observar que el error NC ya tiene más mediciones en cero, al igual que el error manual cada vez disminuye mas su magnitud, es decir, también tiende a acercarse a cero.

En la prueba número cinco (Figura 4.10e), ambos errores volvieron a aumentar en referencia a la prueba anterior, esto es debido a que quizá el usuario en ocasiones pierde un poco la concentración al estar generando la trayectoria o llegue a confundirse en un movimiento con el *mouse* 3D, es decir, que mueva el dispositivo hacia atrás en vez de moverlo hacia adelante, por dar un ejemplo.

De cinco pruebas realizadas con la imagen de la Figura 4.7, se hizo el promedio de los errores manual y NC; los valores medidos al respecto se muestran en la Figura 4.11, es decir, de cada prueba realizada se calculó el promedio del error manual y NC, los cuales están indicados en dicha figura.

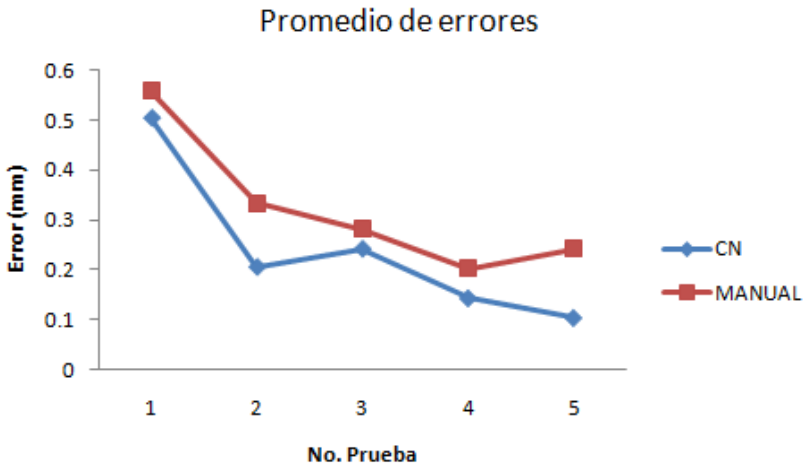


Figura 4.11 Gráfica del error de CN en pruebas de trayectoria rectangular.

En la figura anterior se puede apreciar que en las pruebas rectangulares el error manual, y por ende el error CN fue bajando conforme el usuario se familiarizaba con el programa GenCoG_3D. El error CN promedio de cada prueba está representado con la línea de color azul. El error manual promedio de cada prueba está representado por la línea roja.

El eje horizontal en la gráfica de la Figura 4.11 muestra el número de prueba referente a la Figura 4.7, es decir, el número 1 representa la primera prueba, el número 2 representa la segunda prueba y así sucesivamente. Mientras que el eje vertical representa el valor del error promedio en cada prueba, es decir, para la primera prueba rectangular el error promedio manual fue de 0.503 mm (línea roja), en la segunda prueba, el error manual promedio es de 0.205 mm y así sucesivamente hasta completar las cinco pruebas realizadas.

Ahora bien, para el error CN promedio en la primera prueba es de 0.503 mm (línea azul), la segunda prueba presenta un error CN promedio de 0.205 mm y así sucesivamente.

4.1.4 Trazo y seguimiento de trayectorias circulares

Además de las pruebas en figuras rectangulares, se realizaron otras cinco pruebas más sobre un circulo dibujado en una hoja de papel tal como se muestra en la Figura 4.12 (prueba circular No. 2), en donde se puede apreciar que la figura circular también esta seccionada cada 10mm respecto a la longitud de arco, esto para poder llevar a cabo las mediciones de error de las trayectorias. La trayectoria en color rojo es la que se siguió en línea (trayectoria manual) y la trayectoria azul es la generada por GenCoG_3D (trayectoria CN). La técnica para determinar el error en cada una de las mediciones de éstas pruebas fue generar triángulos rectángulos en donde se hacía cada medición (cada 30mm sobre el perímetro del círculo) y así poder calcular la hipotenusa, la cual es el valor del error en cada medición.

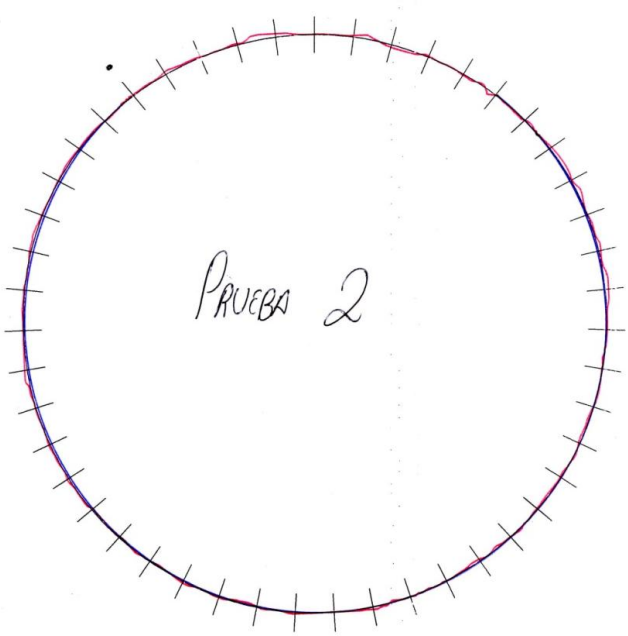


Figura 4.12 Ejemplo de seguimiento de una trayectoria circular.

La siguiente figura muestra un segmento de la prueba donde se realizaron estas mediciones para poder comprender un poco más la formación de dicho triangulo rectángulo.

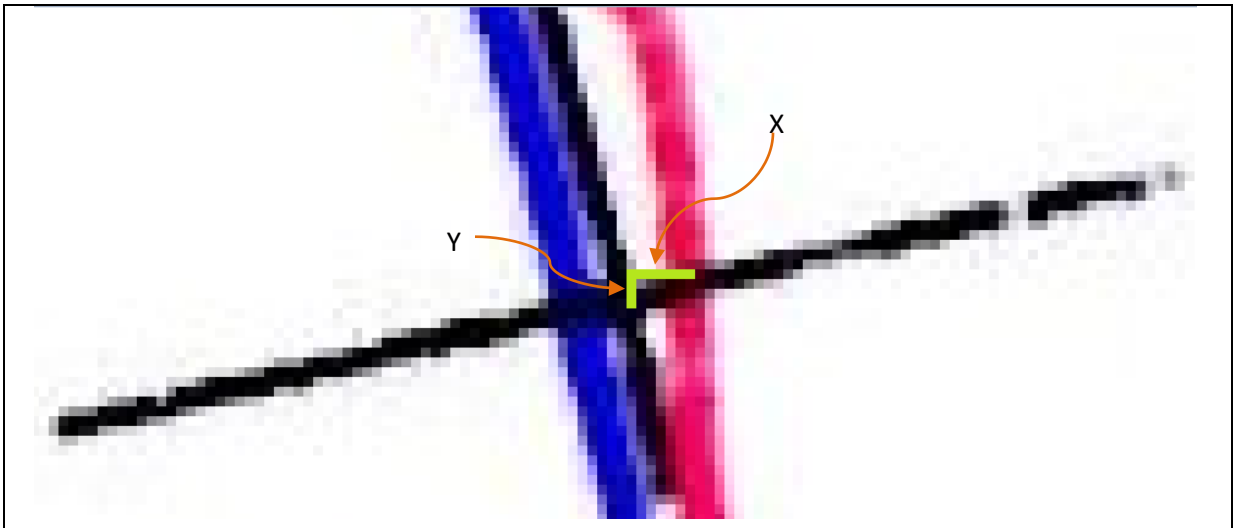


Figura 4.13 **Triangulo rectángulo (cateto opuesto y adyacente en color verde) formado en un segmento de la prueba circular para determinar el error manual (trayectoria roja) donde la hipotenusa del triangulo está representada por el segmento de la línea negra (trayectoria original), mientras que la línea azul representa la trayectoria CN.**

En esta figura puede observarse como es que se determina el error manual y de igual manera el error CN en las pruebas circulares; como puede observarse, se generan dos líneas rectas en color verde para formar los catetos del triángulo rectángulo, el cateto opuesto, que en este caso es la línea vertical tiene por origen el centro de la trayectoria original (línea en color negro), y el cateto adyacente (línea horizontal) se genera a partir del centro de la línea roja justo donde intercepta la línea de trayectoria original, terminando por empatar con la línea vertical, así, se cuentan los números de pixeles que conforman a cada una de estas líneas, teniendo estos valores se aplica el teorema de Pitágoras, por el cual es posible determinar la hipotenusa del triangulo, es decir, la hipotenusa será el numero de pixeles que conforman el error generado en ese punto de la trayectoria.

La Figura 4.13 solo muestra el ejemplo para determinar el error en la trayectoria generada manualmente, pero es el mismo procedimiento para determinar el error en la trayectoria generada automáticamente (línea de color azul), es decir, la trayectoria CN.

Una vez comprendida la técnica de cómo se calcula el error de las trayectorias (manual y CN) en este tipo de pruebas, se presenta en la siguiente figura el código G sintetizado para la trayectoria CN de la Figura 4.12, en donde la línea 1 al igual que en el subcapítulo anterior es generada para levantar el husillo de la fresadora 50mm sobre el punto final de la trayectoria a realizar para librar obstáculos en su recorrido de regreso al punto inicial de dicha trayectoria, el cual es sobre la intersección que existe entre la línea de la trayectoria original y una de las líneas que seccionan al círculo, esta línea puede identificarse por un punto negro que se localiza fuera de la figura pero que está alineado con una de las líneas seccionantes, tal como se muestra en la Figura 4.12.

La línea 2 se genera para mover el husillo de la fresadora a las coordenadas X y Y del punto de inicio de la trayectoria, en este caso es la trayectoria circular de la Figura 4.12.

La línea 3 es generada para bajar el husillo de la fresadora hasta compensar la altura que ganó en la línea 1 para librar obstáculos y posicionarse finalmente en el punto de inicio de la trayectoria a realizar, además, este movimiento lo hace con una velocidad controlada. Puede notarse también que las coordenadas del eje X y Y no aparecen puesto que el movimiento solo es en el eje Z.

Por último, la línea 4 es generada para hacer el arco, cabe mencionar que el código G03 es utilizado en esta fresadora en específico para realizar un arco sobre el plano XY en el sentido contrario de las manecillas del reloj (puede consultarse la Tabla 2.1); este código en su estructura lleva las coordenadas finales del arco (X,Y), las coordenadas del centro de dicho arco (I,J) que son calculadas por el programa GenCoG_3D y la velocidad de avance de la interpolación. Las coordenadas iniciales no se piden porque se entiende que la posición actual del husillo de la fresadora será el punto inicial del arco.

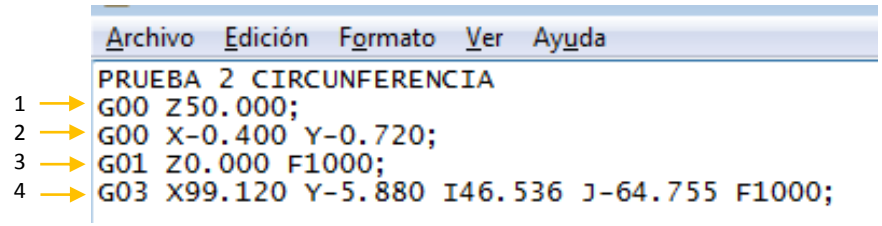


Figura 4.14 **Ejemplo del código G creado para el seguimiento de una trayectoria circular (tome como referencia la Figura 4.12).**

Con esto se conoce y se comprende el código generado para la trayectoria circular de la Figura 4.12, recuérdese también que el error de esta trayectoria fue medido como ya se mencionó en párrafos anteriores con referencia a la trayectoria original (línea negra); y también capturado cada 30mm sobre el perímetro de la misma trayectoria original, estos datos fueron capturados en la Figura 4.15 como sigue:

119

PRUEBA 2: CÍRCULO

Relación milímetros/píxeles: 10 mm = 119 píxeles

	No. De pixeles por cateto		TRAYECTORIA MANUAL		No. De pixeles por cateto		TRAYECTORIA CN	
	X	Y	píxeles	Milímetros	X	Y	Píxeles	Milímetros
Medición 1:	5	3	5.831	0.490	0	0	0.000	0.000
Medición 2:	3	3	4.243	0.357	6	8	10.000	0.840
Medición 3:	2	4	4.472	0.376	2	8	8.246	0.693
Medición 4:	2	3	3.606	0.303	4	8	8.944	0.752
Medición 5:	5	5	7.071	0.594	5	5	7.071	0.594
Medición 6:	6	4	7.211	0.606	6	2	6.325	0.531
Medición 7:	7	2	7.280	0.612	3	2	3.606	0.303
Medición 8:	0	0	0.000	0.000	0	0	0.000	0.000
Medición 9:	2	2	2.828	0.238	2	3	3.606	0.303
Medición 10:	3	5	5.831	0.490	0	0	0.000	0.000
Medición 11:	directo	directo	4.000	0.336	directo	directo	3.000	0.252
Medición 12:	3	4	5.000	0.420	3	5	5.831	0.490
Medición 13:	4	3	5.000	0.420	0	0	0.000	0.000
Medición 14:	5	2	5.385	0.453	0	0	0.000	0.000
Medición 15:	directo	directo	2.000	0.168	0	0	0.000	0.000
Medición 16:	3	2	3.606	0.303	0	0	0.000	0.000
Promedio:				0.385				0.297

Figura 4.15 **Errores registrados en la segunda prueba circular.**

Entonces, en la Figura 4.15 se muestra una tabla con los valores del error de las dos trayectorias hechas en esta prueba (Figura 4.12), es decir, la trayectoria manual y la trayectoria CN. También en la Figura 4.15 se observa un número en la parte superior izquierda, este número representa la cantidad de pixeles que existen por cada 10mm sobre el perímetro del círculo; las columnas encabezadas por “No. de píxeles por cateto” representan los catetos de los triángulos rectángulos formados en cada medición para poder calcular el error en cada una de estas, siendo el error la hipotenusa de este triángulo rectángulo resultante. Las columnas con encabezado rojo (TRAYECTORIA MANUAL) son referentes al error descrito por la trayectoria manual y las columnas con encabezados azules (TRAYECTORIA CN) representan el error de dichas mediciones obtenidas de la trayectoria descrita por GenCoG_3D. Además, al igual que en el subcapítulo anterior, para las siguientes gráficas, las líneas rojas representan el error manual y las líneas azules representan el error CN.

También, en la Figura 4.15, puede observarse que en las columnas “No. de pixeles por cateto” aparecen las variables “X” y “Y”, las cuales están referidas a los catetos del triángulo rectángulo descrito en el párrafo anterior; posteriormente hacia la derecha de estas columnas de datos, en la columna “Trayectoria manual” aparece el encabezado “píxeles”, aquí, cada valor representa la hipotenusa según los valores de la columna anterior (catetos del triángulo rectángulo), es decir, ese valor es el error total de la prueba en ese punto, posteriormente convertido a milímetros como anteriormente ya se mencionó.

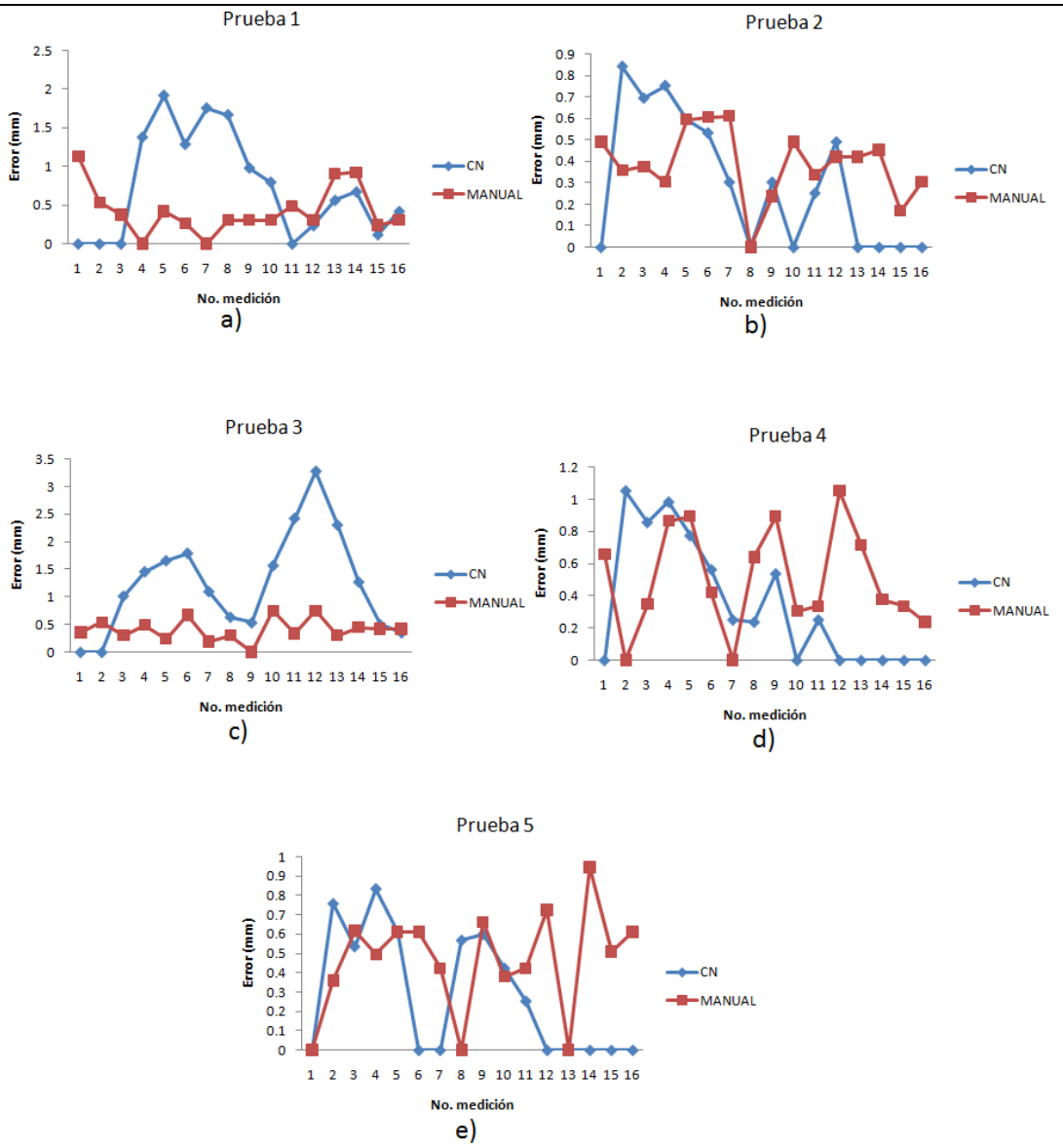


Figura 4.16 Gráfica que muestra el error CN y el error manual de cada prueba circular.

En la Figura 4.16a se observa que el error manual (línea roja y de aquí en adelante para las siguientes cuatro pruebas) tiene una magnitud de error menor a la magnitud del error CN, esto debido a que el usuario todavía no se familiariza por completo con el software GenCoG_3D para trayectorias circulares y es que los puntos necesarios para la generación del arco se introdujeron muy cercanos uno de otro.

En la Figura 4.16a y hasta la Figura 4.16e, el eje horizontal representa el número de medición de la prueba en curso y el eje vertical indica el error en milímetros de cada una de estas mediciones.

En la Figura 4.16b, que es la segunda prueba circular, se aprecia que el error manual se mantiene similar al error manual de la primera prueba pero, el error CN bajó un poco más que el error CN de la primera prueba, esto debido a que el usuario comenzó a familiarizarse con el software desarrollado y los tres puntos necesarios para generar el arco fueron introducidos mas equidistantes entre sí.

Para la tercera prueba circular, el error manual y el error CN se aprecian en la Figura 4.16c, en donde se observa que el error CN volvió a aumentar considerablemente con respecto a las pruebas anteriores, esto es debido a que el usuario vuelve a poner los puntos para generar el arco muy cerca uno de otro, porque también se observa que el error manual con respecto a la prueba anterior se mantiene muy similar, es decir, no mejoró como se esperaba pero tampoco empeoró.

Posteriormente, en la cuarta prueba circular, que es observada en la Figura 4.16d, se aprecia que el error CN redujo su magnitud considerablemente con respecto a la prueba anterior, esto porque el usuario introdujo los puntos para la generación del arco equidistantes entre sí, mientras que el error manual se mantuvo similar al error manual de la prueba anterior, ya que al seguir una trayectoria circular, tiene un grado de dificultad mayor que el seguir una trayectoria en línea recta, por tal motivo, al usuario le cuesta un poco mas de trabajo mover dos ejes simultáneamente que uno solo a la vez.

Finalmente, en la quinta y última prueba circular (Figura 4.16e), se observa que el error CN bajó un poco más referido a la prueba anterior, es decir, se mejoró la equidistancia entre los puntos necesarios para la generación del arco, mientras que el error manual también se mejoró un poco mas con referencia a la prueba anterior. Esto refleja que el usuario ha adquirido experiencia con el software GenCoG_3D.

En la Figura 4.17 se muestra la gráfica de los promedios de los errores de cada una de las pruebas circulares realizadas (cinco pruebas hechas).

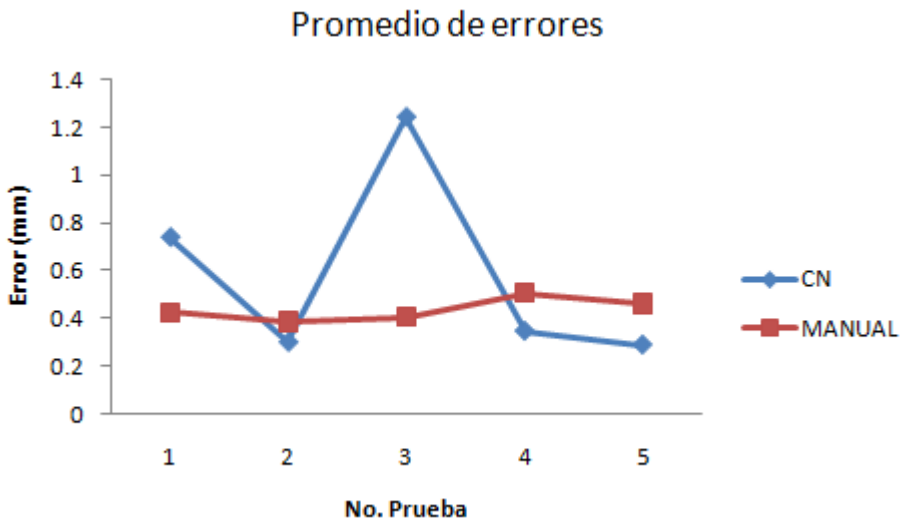


Figura 4.17 **Gráfica que muestra el error promedio CN (azul) y el error promedio manual (rojo) en cada prueba circular.**

En la figura anterior se aprecia la línea de color azul que representa el error promedio CN en cada prueba circular realizada, es decir, sobre el eje horizontal el número uno representa la primera prueba, el dos, la segunda prueba y así sucesivamente hasta completar las cinco pruebas realizadas. Mientras que el eje vertical representa el error de cada prueba medido en milímetros. Además, se observa que en la prueba número 3, el error CN medido sobresale de las otras cuatro pruebas, esto es debido a que en ésta prueba, los tres puntos necesarios para generar el código G del arco se introdujeron muy cerca uno de otro, esto ocasiona que el arco generado por GenCoG_3D no siga fielmente la circunferencia original.

Además, se observa que el error CN fue disminuyendo conforme el usuario se fue familiarizando con el software GenCoG_3D.

Después, en la Figura 4.17 también puede observarse que la línea en color rojo representa el error manual de las cinco pruebas de arco realizadas, aquí se observa además que el error estuvo muy variado y no disminuyó como se hubiera querido, ya que la magnitud del error depende, incluso, del estado de ánimo y concentración del usuario al momento de operar el software GenCoG_3D.

Así, se puede observar que los errores registrados en estas pruebas son mucho menores que los errores evaluados en otros proyectos como los descritos en el primer capítulo de este proyecto.

Ahora, para hacer la comparación del error entre ambas pruebas (circular y rectangular), nos basaremos en la Figura 4.18 que contiene las graficas de error CN de las pruebas rectangulares y circulares al igual que los errores manuales de estas dos pruebas.

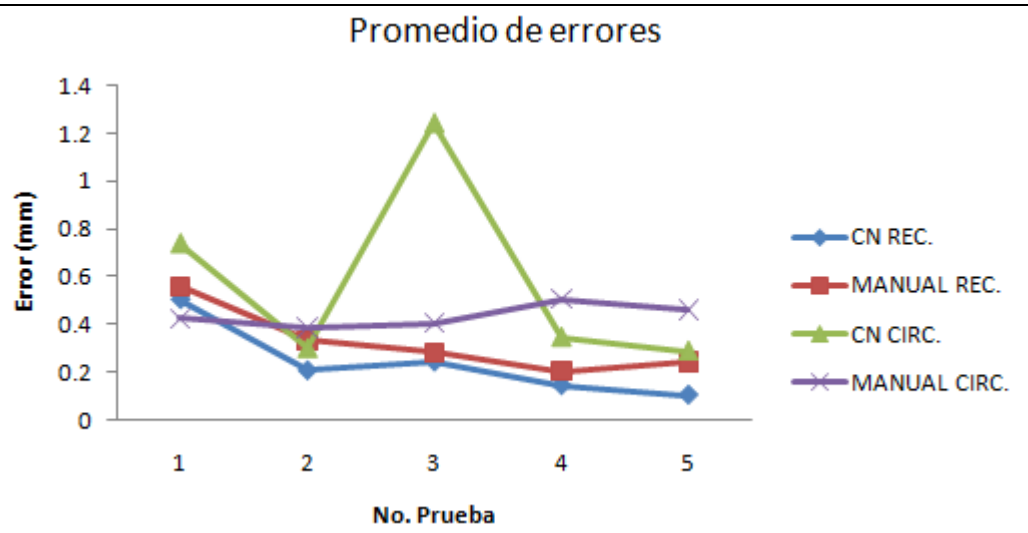


Figura 4.18 **Gráfica que muestra los errores promedio CN y los errores promedio manuales en cada prueba circular y rectangular.**

En la figura anterior se aprecian cuatro gráficas, las cuales son descritas como sigue: la línea azul representa el error CN promedio de cada prueba rectangular que se realizó; la línea roja representa el error manual promedio de estas mismas pruebas rectangulares. La línea verde representa el valor del error CN promedio de cada prueba circular realizada y para estas mismas pruebas, la línea morada representa el error manual promedio de cada una de estas.

El eje horizontal de la gráfica en la figura anterior representa el número de prueba realizada tanto para las rectangulares como para las circulares. Mientras que el eje vertical, representa la magnitud del error promedio para cada una de estas pruebas.

Como puede observarse, en la Figura 4.18, comparando los errores CN de las pruebas rectangulares y circulares, se ve notablemente que el error medido en las pruebas rectangulares es menor que el de las pruebas circulares, esto es lógico, ya que seguir una línea recta tiene menor dificultad que seguir una trayectoria circular, lo mismo pasa con los errores manuales de estas pruebas; puede observarse que el error medido en las pruebas rectangulares es menor que el que se obtuvo de las pruebas circulares, esto por la misma razón descrita anteriormente.

4.2 Conclusiones y Recomendaciones

El desarrollo de este proyecto involucró aspectos de conocimientos especializados sobre Programación media-avanzada bajo *Windows*, Procesos de Manufactura y Programación de Control Numérico, Protocolos de Comunicación Serial, Herramientas matemáticas sobre transformaciones lineales, Geometría Analítica y Análisis de Señales para así lograr el *software* Generador de Trayectorias 3D: GenCoG_3D.

La integración de los conocimientos mencionados anteriormente, para el desarrollo del proyecto, supuso mucho trabajo para la puesta en marcha del *software*, así mismo hubo que realizar ingeniería inversa en la comunicación serial entre un programa comercial de CAM y la máquina CNC para implementar la comunicación de nuestro *software* con el centro de maquinado.

En cuanto al desempeño para la puesta en marcha del software GenCoG_3D, se logró observar que se debe de tener cuidado o ganar experiencia en el manejo del *mouse* 3D por parte del usuario u operador, esto para lograr una trayectoria más fiel a la original, principalmente cuando se está manejando "en línea" el centro de maquinado CNC con dicho dispositivo. Esto con el fin de evitar colisiones entre el husillo y la mesa o los accesorios de sujeción, incluso con la misma pieza y la herramienta. En consecuencia, este especial cuidado deriva en que la trayectoria generada de forma manual sea todavía más precisa en referencia a la trayectoria original.

Una ventaja de este proyecto es que para seguir una trayectoria cualquiera, no necesita estar alineada con los ejes de la mesa de la máquina, ya que el *mouse* 3D permite la manipulación de ésta hasta en los tres ejes a la vez. Tampoco es necesario poner el cero pieza, puesto que con el *mouse* 3D se puede llevar la herramienta hasta el punto inicial donde la trayectoria se comience a generar.

Una ventaja sumamente importante es que el usuario no necesita ser un experto programador de CNC, es decir, si el usuario sabe lo básico de la programación CNC como por ejemplo el conocimiento de los códigos G, puede ser capaz de operar con facilidad el programa GenCoG_3D para generar la trayectoria deseada. Esto se debe a que el programa estructura y

genera los códigos de la máquina de forma automática con solo presionar un botón que indica al programa el inicio de la trayectoria a voluntad del usuario. Cabe mencionar que las interpolaciones que se emplean son solamente líneas (sobre los planos formados por los ejes X, Y y Z ó en el espacio) y arcos (solamente en el plano X,Y, ya que el centro de maquinado CNC en donde se realizó este proyecto está limitado a esta interpolación).

En la síntesis de trayectorias circulares, para las cuales se requieren tres puntos, se aconseja, que estos puntos sean lo más equidistantes entre sí a lo largo de la trayectoria del arco. Así, la síntesis del arco descrita por el programa GenCoG_3D resulta más precisa y, por ende, la ejecución del movimiento será más fiel a la trayectoria original. En caso contrario, si dos de los tres puntos están muy cercanos unos de otros, la trayectoria sintetizada dejará de ser semejante a la original ya que los errores inducidos por el usuario se ven amplificados.

Por otra parte, se observó también que el error de seguimiento en trayectorias circulares sigue una relación inversamente proporcional al diámetro de la trayectoria.; es decir, que entre más cerrado sea el diámetro, el error resulta ser mayor. Esto es atribuible a que el usuario es menos diestro al seguir trayectorias circulares pequeñas.

En los resultados de los experimentos, se puede observar que en el seguimiento de las trayectorias de forma manual, el error fue disminuyendo a medida que el usuario va adquiriendo experiencia con el uso del *Mouse 3D* y la respuesta de la máquina de CNC. Esto refleja que el sistema es sensible a la experiencia y destreza del usuario.

Otro punto no menos importante es referido a la precisión, la cual es sensible también a la perspectiva de observación y agudeza visual del usuario, ya que si no existe una buena iluminación del entorno durante la síntesis o programación de la trayectoria, los resultados conducen a mayores errores. Además, el empleo del trazador, en algunas ocasiones bloquea la visibilidad entre el usuario y la trayectoria. Cabe mencionar que esto pudiera disminuirse si en un trabajo futuro se empleara un trazador láser u óptico en vez de emplear un trazador mecánico de contacto.

En general, el proyecto en general demuestra que el proceso puede ser aplicable a procesos de soldadura automatizada o robotizada, ya que en términos generales la precisión del sistema es bastante aceptable para estos procesos de unión. Por otro lado, en las pruebas de

funcionamiento, se observó que el tiempo empleado para la generación de las trayectorias es corto; puesto que no se requieren actividades de modelado y análisis del entorno para la programación de robots, tampoco se requiere del modelado de las piezas que se desean unir e incluso se puede prescindir de la calibración y alineación precisa de las piezas, del robot y del punto de inicio de la trayectoria; todo ello en el entendido que la trayectoria sintetizada se encuentra dentro del espacio de trabajo del robot.

ANEXO A

Archivos de código fuente

GeneradorTray1p0.cpp

```
// GeneradorTray1p0.cpp : Defines the entry point for the application.
#include "stdafx.h"
#include "GeneradorTray1p0.h"
#define MAX_LOADSTRING 100
#define dpi 3.141592654
//longitud del espacio de trabajo de la DM4322 (Dyna mechtronics)
#define DMX  1100.0    //mm
#define DMY  -560.0    //mm
#define DMZ  -590.0    //mm
double CoAbCePi[4];
float fctemp[4];
double crel[4], cabsol[4];
//Variables globales:
HINSTANCE hInst;                // current instance
TCHAR szTitle[MAX_LOADSTRING];          // The title bar text
TCHAR szWindowClass[MAX_LOADSTRING];      // the main window class name
HWND hWndMain, hWndDlg ;
HDC         hdc ;
SiHdl       devHdl ;
// Forward declarations of functions included in this code module:
ATOM        MyRegisterClass(HINSTANCE hInstance);
BOOL        InitInstance(HINSTANCE, int);
LRESULT CALLBACK  WndProc(HWND, UINT, WPARAM, LPARAM);
INT_PTR CALLBACK  About(HWND, UINT, WPARAM, LPARAM);
INT_PTR CALLBACK  TresDConnexion(HWND, UINT, WPARAM, LPARAM);
INT_PTR CALLBACK  ProcDlgCommSettings(HWND, UINT, WPARAM, LPARAM);
INT_PTR CALLBACK  ProcDlgDNC(HWND, UINT, WPARAM, LPARAM);
void CALLBACK    CapturarCoordenadas3D(HWND,UINT, UINT,DWORD);
void CALLBACK    EnvioDatos(HWND,UINT, UINT,DWORD);
INT_PTR CALLBACK  ProcDlgCircun(HWND, UINT, WPARAM, LPARAM);
INT_PTR CALLBACK  ProcDlgCircyLinea(HWND, UINT, WPARAM, LPARAM);
INT_PTR CALLBACK    InsertarCoordenadas(HWND, UINT, WPARAM, LPARAM);
INT_PTR CALLBACK    InsertarCoordenadasMaq(HWND, UINT, WPARAM, LPARAM);
INT_PTR CALLBACK  ProcDlgConfiguracion(HWND, UINT, WPARAM, LPARAM);
BOOL bEcTresPunts  (double Px, double Py, double Qx, double Qy, double Rx, double Ry);
BOOL bSolSis    (double Aa, double Bb, double Cc, double Dd, double Ee, double Ff);
BOOL bP2C      (double *mag, double *ang, double *Cx, double *Cy);
BOOL bC2P      (double *Cx, double *Cy, double *mag, double *ang);
BOOL InterCircToLine (double *p1x, double *p1y, double *p2x, double *p2y, double *h,
            double *k, double *radio,
            double *s1x, double *s1y, double *s2x, double *s2y);
BOOL ConectarPuertoSerial(HWND);
void Copy2Clipboard  (HWND hwnd, char *pszData);
typedef struct coordenadas
{
  double x;
  double y;
  double z;
} c;
struct
{
  c cd[3];
  int ncoord;
  int tipo;
  int movto;
}
objeto[10000];
int num_objetos = 0, num_cord = 0;
char codigosG[100000], portpapeles[100000];
BOOL bBotonI = FALSE, bBotonD = FALSE, bPuertoConectado = FALSE;
char Trayectoria[10000][500];
int nInstruc = 0;
#define TR 0
#define PS 1
int opcion = TR;
BOOL bCoordMaq = FALSE ;
#define Vel 1000.0
               //feed rate, rapid feed, in line feed, unused
double dVel[4] = {Vel,      7000.0,   9000.0,    0.0};
// define los eventos de operacion de la botonera
// Define el tipo de botón de la ventana principal
struct
{
  long estilo ; // Estilo del boton
  char *texto ; // Texto del boton
  BOOL bChecarBoton ;  // Bandera de habilitación
  BOOL bApEntrada ;  // Bandera de habilitación
  BOOL bApMovHta ;  // Bandera de habilitación
  BOOL bApCapTray ;  // Bandera de habilitación
  int iCoordraton;
}
boton[] = //apariencia boton inicial
```

```
{
    BS_RADIOBUTTON,  "Eje X",          FALSE,  TRUE,  TRUE,  TRUE,  0,
    BS_RADIOBUTTON,  "Eje Y" ,          FALSE,  TRUE,  TRUE,  TRUE,  0,
    BS_RADIOBUTTON,  "Eje Z" ,          FALSE,  TRUE,  TRUE,  TRUE,  0,
    BS_RADIOBUTTON,  "Mover Hta." ,        FALSE,  TRUE,  TRUE,  TRUE,  0,
    BS_RADIOBUTTON,  "Coord. Iniciales",      FALSE,  FALSE,  TRUE,  TRUE,  0,
    BS_RADIOBUTTON,  "Capturar Tray.",      FALSE,  FALSE,  TRUE,  TRUE,  0,
    BS_RADIOBUTTON,  "Línea",          FALSE,  FALSE,  FALSE,  TRUE,  0,
    BS_RADIOBUTTON,  "Arco +",          FALSE,  FALSE,  FALSE,  TRUE,  0,
    BS_RADIOBUTTON,  "Arco -",          FALSE,  FALSE,  FALSE,  TRUE,  0,
    BS_RADIOBUTTON,  "Trabajo",          FALSE,  FALSE,  FALSE,  FALSE,  0,
    BS_RADIOBUTTON,  "Posición",          FALSE,  FALSE,  FALSE,  FALSE,  0,
    BS_RADIOBUTTON,  "Cargar",          FALSE,  FALSE,  FALSE,  TRUE,  0,
    BS_RADIOBUTTON,  "P. de Emergencia" ,    TRUE,  TRUE,  TRUE,  TRUE,  0
} ;
#define NUMBOTONES (sizeof boton / sizeof boton[0])
HWND  hwndBoton[NUMBOTONES] ;
//Definicion de constantes y variables para el puerto RS 232
#define PORT1 0
#define PORT2 1
#define PORT3 2
#define PORT4 3
#define PORT5 4
#define PORT6 5
#define PORT7 6
#define PORT8 7
#define PORT9 8
#define PORT10 9
#define PORT11 10
#define PORT12 11
#define PORTCOMNUMBER PORT12 + 1
unsigned char sPortName[PORTCOMNUMBER][80] = {
  {"COM1"},
  {"COM2"},
  {"COM3"},
  {"COM4"},
  {"COM5"},
  {"COM6"},
  {"COM7"},
  {"COM8"},
  {"COM9"},
  {"COM10"},
  {"COM11"},
  {"COM12"}};
int iIndexPort = 0;
HANDLE hComm;
BOOL m_bPortReady;
DCB  m_dcb;
COMMTIMEOUTS m_CommTimeouts;
DWORD iBytesWritten;
int gle;
char dsend[50000]="";//sendb); letra w 7 bits, dato a enviar
DWORD dwRead;
char ptrrx[200]="";  // variable donde se vuelcan los datos leidos
char ptrrx2[200]="";
int iszptrx = 0 ;
//double fabsX,fabsY,fabsZ;
char sdatosleidos[300]="" ;
#define TIMERMOUSE3D    1  //define identificador del temporizador
#define DELTATIME      180  //define el periodo del temporizador en mSeg. cada que mover hta.
#define TIMERDNC      2  //define identificador del temporizador
#define DELTATIMEDNC    500  //define el periodo del temporizador en mSeg. dar tiempo a la fresadora
int inVeces = 0;
double AA, BB, CC, DD, EE, FF, CX, CY, PX, PY, QX, QY, RX, RY;
BOOL ConfigurePortCOM(DWORD BaudRate, BYTE ByteSize, DWORD fParity, BYTE Parity, BYTE StopBits)
{
  if((m_bPortReady = GetCommState(hComm, &m_dcb))==0)
  {
    MessageBox(NULL, "GetCommState Error", "ERROR", MB_OK | MB_ICONEXCLAMATION) ;
    CloseHandle(hComm);
    return FALSE;
  }
  m_dcb.BaudRate =BaudRate;
  m_dcb.ByteSize = ByteSize;
  m_dcb.Parity =Parity ;
  m_dcb.StopBits =StopBits;
  m_dcb.fBinary=TRUE;
  m_dcb.fParity=fParity;
  m_bPortReady = SetCommState(hComm, &m_dcb);
  if(m_bPortReady ==0)
  {
    MessageBox(NULL, "SetCommState Error", "ERROR", MB_OK | MB_ICONEXCLAMATION) ;
    CloseHandle(hComm);
    return FALSE;
  }
  return TRUE;
}
BOOL SetCommunicationTimeouts(DWORD ReadIntervalTimeout, DWORD ReadTotalTimeoutMultiplier, DWORD ReadTotalTimeoutConstant,
DWORD WriteTotalTimeoutMultiplier, DWORD WriteTotalTimeoutConstant)
{
  if((m_bPortReady = GetCommTimeouts (hComm, &m_CommTimeouts))==0)
    return FALSE;
  m_CommTimeouts.ReadIntervalTimeout =ReadIntervalTimeout;
  m_CommTimeouts.ReadTotalTimeoutConstant =ReadTotalTimeoutConstant;
  m_CommTimeouts.ReadTotalTimeoutMultiplier =ReadTotalTimeoutMultiplier;
```

```
  m_CommTimeouts.WriteTotalTimeoutConstant = WriteTotalTimeoutConstant;
  m_CommTimeouts.WriteTotalTimeoutMultiplier =WriteTotalTimeoutMultiplier;
  m_bPortReady = SetCommTimeouts (hComm, &m_CommTimeouts);
  if(m_bPortReady ==0)
  {
    MessageBox(NULL, "StCommTimeouts function failed", "ERROR", MB_OK | MB_ICONEXCLAMATION) ;
    CloseHandle(hComm);
    return FALSE;
  }
  return TRUE;
}
BOOL WriteABuffer(void)
{
  char mssg[5000] ;
  OVERLAPPED osWrite = {0};
  DWORD dwWritten;
  BOOL fRes;
  int i ;
   // Create this writes OVERLAPPED structure hEvent.
   osWrite.hEvent = CreateEvent(NULL, TRUE, FALSE, NULL);
   if (osWrite.hEvent == NULL)
   {
     MessageBox(NULL, "Error creating overlapped event handle", "ERROR", MB_OK | MB_ICONEXCLAMATION) ;
      // Error creating overlapped event handle.
      return FALSE;
   }
   // Issue write.
   if (!WriteFile(hComm, dsend, strlen(dsend), &dwWritten, &osWrite)) // Cantidad de bytes a escribir 3er parametro
   {
      if (GetLastError() != ERROR_IO_PENDING)
    {
        // WriteFile failed, but it isn't delayed. Report error and abort.
    MessageBox(NULL, "WriteFile failed, but it isn't delayed. Report error and abort.", "ERROR", MB_OK |
MB_ICONEXCLAMATION) ;
        fRes = FALSE;
      }
      else
    {
        // Write is pending.
        if (!GetOverlappedResult(hComm, &osWrite, &dwWritten, TRUE))
     {
      MessageBox(NULL, "Write is pending is pending.", "Aviso", MB_OK | MB_ICONEXCLAMATION) ;
       fRes = FALSE;
     }
        else
     {
       for (i = 0; i < dwWritten; i++) // Convesion de ascii a hexadecimal
       {
         itoa((int)dsend[i], ptrrx2, 10) ;
         strcat(sdatosleidos, ptrrx2) ;
         strcat(sdatosleidos, ", ") ;
       }
       sprintf(mssg, "Escritura exitosa:  %s,  %i",sdatosleidos,strlen(dsend)) ; // Ventana q imprime los caracteres q se
muestran
       strcpy(sdatosleidos, "") ;
       fRes = TRUE;
     }
      }
   }
   else
   {
          sprintf(mssg, "La operacion de escritura se ha realizado inmediatamente.   %s",sdatosleidos) ;
      fRes = TRUE;
   }
   CloseHandle(osWrite.hEvent);
   return fRes;
}
int   SbInit();
void  SbMotionEvent(SiSpwEvent *pEvent);
void  SbZeroEvent();
void  SbButtonPressEvent(int buttonnumber);
void  SbButtonReleaseEvent(int buttonnumber);
int WINAPI WinMain(HINSTANCE hInstance,
                   HINSTANCE hPrevInstance,
                   LPTSTR    lpCmdLine,
                   int       nCmdShow)
{
  UNREFERENCED_PARAMETER(hPrevInstance);
  UNREFERENCED_PARAMETER(lpCmdLine);
   // TODO: Place code here.
////////////////////////////////////////////////////////////////////////////////////////////////
   int            num;      // number of button pressed
   MSG            msg;      // incoming message to be evaluated
   BOOL           handled;  // is message handled yet
   SiSpwEvent     Event;    // SpaceWare Event
   SiGetEventData EData;    // SpaceWare Event Data
   int  res;            // SbInits result..if>0 it worked, if=0 it didnt work
   handled = SPW_FALSE;     // init handled
////////////////////////////////////////////////////////////////////////////////////////////////
//  MSG msg;
  HACCEL hAccelTable;
  // Initialize global strings
  LoadString(hInstance, IDS_APP_TITLE, szTitle, MAX_LOADSTRING);
  LoadString(hInstance, IDC_GENERADORTRAY1P0, szWindowClass, MAX_LOADSTRING);
  MyRegisterClass(hInstance);
```

```
  // Perform application initialization:
  if (!InitInstance (hInstance, nCmdShow))
  {
    return FALSE;
  }
  res = SbInit();

  if (res < 1)
  {
    MessageBox(hWndMain,
      "No se ha detectado el dispositivo SpaceBall.\nEl programa se cerrará.", "AVISO",
      MB_OK | MB_ICONEXCLAMATION);
    if (hWndMain != NULL)
    {
      DestroyWindow(hWndMain);    /* destroy window */
    }
    ExitProcess(1);                /* exit program */
  }
   /* start message loop */
   while ( GetMessage( &msg, NULL, 0, 0 ) )
      {
      handled = SPW_FALSE;
       /* init Window platform specific data for a call to SiGetEvent */
       SiGetEventWinInit(&EData, msg.message, msg.wParam, msg.lParam);
       /* check whether msg was a Spaceball event and process it */
       if (SiGetEvent (devHdl, 0, &EData, &Event) == SI_IS_EVENT)
          {
          if (Event.type == SI_MOTION_EVENT)
             {
             SbMotionEvent(&Event);        /* process Spaceball motion event */
          }
          if (Event.type == SI_ZERO_EVENT)
             {
             SbZeroEvent();                /* process Spaceball zero event */
          }
          if (Event.type == SI_BUTTON_EVENT)
             {
             if ((num = SiButtonPressed (&Event)) != SI_NO_BUTTON)
                {
                SbButtonPressEvent(num);        /* process Spaceball button event */
                }
             if ((num = SiButtonReleased (&Event)) != SI_NO_BUTTON)
                {
                SbButtonReleaseEvent(num);        /* process Spaceball button event */
                }
          }
         handled = SPW_TRUE;              /* spaceball event handled */
          }
        /* not a Spaceball event, let windows handle it */
     if (handled == SPW_FALSE)
         {
         TranslateMessage( &msg );
          DispatchMessage( &msg );
          }
       }
    return( (int) msg.wParam );
ATOM MyRegisterClass(HINSTANCE hInstance)
{
  WNDCLASSEX wcex;
  wcex.cbSize = sizeof(WNDCLASSEX);
  wcex.style      = CS_HREDRAW | CS_VREDRAW;
  wcex.lpfnWndProc  = WndProc;
  wcex.cbClsExtra    = 0;
  wcex.cbWndExtra    = 0;
  wcex.hInstance    = hInstance;
  wcex.hIcon      = LoadIcon(hInstance, MAKEINTRESOURCE(IDI_GENERADORTRAY1P0));
  wcex.hCursor    = LoadCursor(NULL, IDC_ARROW);
  wcex.hbrBackground  = (HBRUSH)(COLOR_WINDOW+1);
  wcex.lpszMenuName  = MAKEINTRESOURCE(IDC_GENERADORTRAY1P0);
  wcex.lpszClassName  = szWindowClass;
  wcex.hIconSm    = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI_SMALL));
  return RegisterClassEx(&wcex);
}
BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
{
  hInst = hInstance; // Store instance handle in our global variable
  hWndMain = CreateWindow(szWindowClass, szTitle, WS_OVERLAPPEDWINDOW,
    CW_USEDEFAULT, 0, CW_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);
  if (!hWndMain)
  {
    return FALSE;
  }
  ShowWindow(hWndMain, SW_SHOWMAXIMIZED);
  UpdateWindow(hWndMain);
  return TRUE;
}
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
  int wmId, wmEvent, longitud;
  char messg[50000], messg2[50000], messg3[50000];
  PAINTSTRUCT ps;
  HDC hdc;
  static int cxChar, cyChar, cxClient, cyClient, iMaxWidth,
    iVscrollPos, iVscrollMaxm, iHscrollPos, iHscrollMax;
  static int iMsg ;
```

```
static int  iNup, iNdown, iNreturn, iNstop;
unsigned char szBuffer[50] ;
static char szTop[]  = "iMsg          wParam          lParam",
  szUnd[]       = "_____       ______          ______",
  szFormat[]    = "%-16s%04X-%04X        %04X-%04X";
static RECT  rect ;
int      i,j;
TEXTMETRIC  tm ;
switch (message)
{
case WM_CREATE:
  hdc = GetDC (hWnd) ;
  SelectObject (hdc, GetStockObject (SYSTEM_FIXED_FONT)) ;
  GetTextMetrics (hdc, &tm) ;
  cxChar = tm.tmAveCharWidth ;
  cyChar = tm.tmHeight + tm.tmExternalLeading - 2;
  ReleaseDC (hWnd, hdc);

  for (i = 0 ; i < NUMBOTONES ; i++)
  {
    hwndBoton[i] = CreateWindow ("Button", boton[i].texto,
    WS_CHILD | WS_VISIBLE | boton[i].estilo | BS_PUSHLIKE,
    cxChar, cyChar * (1 + 3 * i), //coordenada superior izquierda del boton
    20 * cxChar, 10 * cyChar / 4,//coordenada inferior derecha del boton
    hWnd, (HMENU) (ID_EJEX + i),
    ((LPCREATESTRUCT) lParam) -> hInstance, NULL) ;
    EnableWindow(hwndBoton[i], boton[i].bApEntrada) ;
  }
  bPuertoConectado = ConectarPuertoSerial(NULL);
  if(bPuertoConectado == FALSE)
  {
    SendMessage (hWnd, WM_COMMAND, ID_DISPOSITIVO_RS232,0L);

  }
  return 0;
case WM_ACTIVATEAPP:
  break;
case WM_KEYDOWN:
//case WM_KEYUP:
  // user hit a key to close program
  if (wParam == VK_SPACE)
  {
    MessageBox(NULL,"%&/$#%","VENTANA",MB_ICONEXCLAMATION | MB_YESNOCANCEL);
    SendMessage (hWnd, WM_COMMAND, ID_P_EMERG,0L);
  }
  return (INT_PTR)FALSE ;
case WM_COMMAND:
  wmId    = LOWORD(wParam);
  wmEvent = HIWORD(wParam);
  // Parse the menu selections:
  switch (wmId)
  {
  case ID_CONFIGURACION_SISTEMA:
    DialogBox(hInst, MAKEINTRESOURCE(IDD_DIALOG6), hWnd, ProcDlgConfiguracion);
    break;
  case ID_UTILS_CIRCUNFERENCIA:
    DialogBox(hInst, MAKEINTRESOURCE(IDD_DIALOG3), hWnd, ProcDlgCircun);
    break;
  case ID_UTILS_CIRCYLINEA:
    DialogBox(hInst, MAKEINTRESOURCE(IDD_DIALOG4), hWnd, ProcDlgCircyLinea);
    break;
  case ID_DISPOSITIVO_RS232:
    DialogBox(hInst, MAKEINTRESOURCE(IDD_DIALOG1), hWnd, ProcDlgCommSettings);
    break;
  case ID_CONFIGURACION_DNC:
    DialogBox(hInst, MAKEINTRESOURCE(IDD_DIALOG2), hWnd, ProcDlgDNC);
    break;
  case ID_EJEX:
    if (BST_CHECKED == IsDlgButtonChecked (hWnd, ID_EJEX))
    {
      boton [ID_EJEX-ID_EJEX].bChecarBoton = FALSE;
    SendDlgItemMessage (hWnd,ID_EJEX,BM_SETCHECK, (WPARAM) BST_UNCHECKED,0);
    }
    else
    {
      boton [ID_EJEX-ID_EJEX].bChecarBoton = TRUE;
      SendDlgItemMessage (hWnd,ID_EJEX,BM_SETCHECK, (WPARAM) BST_CHECKED,0);
    }
    break;
  case ID_EJEY:
    if (BST_CHECKED == IsDlgButtonChecked (hWnd, ID_EJEY))
    {
      boton [ID_EJEY-ID_EJEX].bChecarBoton = FALSE;
    SendDlgItemMessage (hWnd,ID_EJEY,BM_SETCHECK, (WPARAM) BST_UNCHECKED,0);
    }
    else
    {
      boton [ID_EJEY-ID_EJEX].bChecarBoton = TRUE;
      SendDlgItemMessage (hWnd,ID_EJEY,BM_SETCHECK, (WPARAM) BST_CHECKED,0);
    }
    break;
  case ID_EJEZ:
    if (BST_CHECKED == IsDlgButtonChecked (hWnd, ID_EJEZ))
    {
      boton [ID_EJEZ-ID_EJEX].bChecarBoton = FALSE;
```

```
  SendDlgItemMessage (hWnd,ID_EJEZ,BM_SETCHECK, (WPARAM) BST_UNCHECKED,0);
  }
  else
  {
    boton [ID_EJEZ-ID_EJEX].bChecarBoton = TRUE;
    SendDlgItemMessage (hWnd,ID_EJEZ,BM_SETCHECK, (WPARAM) BST_CHECKED,0);
  }
  break;
case ID_MOVER_HTA:
  if (bCoordMaq)
  if (BST_CHECKED == IsDlgButtonChecked (hWnd, ID_MOVER_HTA))
  {
    boton [ID_MOVER_HTA-ID_EJEX].bChecarBoton = FALSE;
    KillTimer(hWndMain,TIMERMOUSE3D);
    SendDlgItemMessage (hWnd,ID_MOVER_HTA,BM_SETCHECK, (WPARAM) BST_UNCHECKED,0);
    for (i = 0 ; i < NUMBOTONES ; i++)
      EnableWindow(hwndBoton[i], boton[i].bApEntrada);
  }
  else
  {
    for (i = 0 ; i < NUMBOTONES ; i++)
      EnableWindow(hwndBoton[i], boton[i].bApMovHta) ;
    boton [ID_MOVER_HTA-ID_EJEX].bChecarBoton = TRUE;
    SendDlgItemMessage (hWnd,ID_MOVER_HTA,BM_SETCHECK, (WPARAM) BST_CHECKED,0);
    boton [ID_CAPTURAR_TRAY-ID_EJEX].bChecarBoton = FALSE;
    SendDlgItemMessage (hWnd,ID_CAPTURAR_TRAY,BM_SETCHECK, (WPARAM) BST_UNCHECKED,0);
    nInstruc = 0;
    if(hComm == INVALID_HANDLE_VALUE)
    {
      gle = GetLastError();
      sprintf(messg, "error no.%d\n", gle);
      MessageBox(NULL, messg, "Cannot open Communication Port..", MB_OK | MB_ICONEXCLAMATION) ;
    }
    else
    {
      if(!ConfigurePortCOM(115200, 7, 0,EVENPARITY, TWOSTOPBITS))
      {
        MessageBox(NULL, "Cannot Configure Communication Port", "Error", MB_OK | MB_ICONEXCLAMATION) ;
        CloseHandle(hComm);
      }
      else
      {
        if(!SetCommunicationTimeouts(1000,1000,1000,1000,1000))
        {
          MessageBox(NULL, "Cannot Configure Communication Timeouts", "Error", MB_OK | MB_ICONEXCLAMATION) ;
          CloseHandle(hComm);
        }
      }
    }
    SetTimer(hWndMain,TIMERMOUSE3D,DELTATIME,CapturarCoordenadas3D);
  }
  else
    SendMessage(hWndMain,WM_COMMAND,ID_COORD,0L);
  break;
  case ID_CAPTURAR_TRAY:
  if (BST_CHECKED == IsDlgButtonChecked (hWnd, ID_CAPTURAR_TRAY))
  {
    for (i = 0 ; i < NUMBOTONES ; i++)
      EnableWindow(hwndBoton[i], boton[i].bApMovHta) ;
    boton [ID_CAPTURAR_TRAY-ID_EJEX].bChecarBoton = FALSE;
  SendDlgItemMessage (hWnd,ID_CAPTURAR_TRAY,BM_SETCHECK, (WPARAM) BST_UNCHECKED,0);
  }
  else
  {
    for (i = 0 ; i < NUMBOTONES ; i++)
      EnableWindow(hwndBoton[i], boton[i].bApCapTray) ;
    boton [ID_CAPTURAR_TRAY-ID_EJEX].bChecarBoton = TRUE;
    SendDlgItemMessage (hWnd,ID_CAPTURAR_TRAY,BM_SETCHECK, (WPARAM) BST_CHECKED,0);
  }
  break;
  case ID_COORD:
    DialogBox(hInst, MAKEINTRESOURCE(IDD_DIALOG5), hWnd, InsertarCoordenadasMaq);
    break;
  case ID_COORD_INICIALES:
    DialogBox(hInst, MAKEINTRESOURCE(IDD_DIALOG5), hWnd, InsertarCoordenadas);
    break;
  case ID_LINEA:
  if (BST_CHECKED == IsDlgButtonChecked (hWnd, ID_LINEA))
  {
    num_cord = 0;
    boton [ID_LINEA-ID_EJEX].bChecarBoton = FALSE;
    EnableWindow(hwndBoton[ID_ARCO_POS - ID_EJEX], TRUE);
    EnableWindow(hwndBoton[ID_ARCO_NEG - ID_EJEX], TRUE);
    EnableWindow(hwndBoton[ID_TRABAJO - ID_EJEX], FALSE);
    EnableWindow(hwndBoton[ID_POSICION - ID_EJEX], FALSE);
  SendDlgItemMessage (hWnd,ID_LINEA,BM_SETCHECK, (WPARAM) BST_UNCHECKED,0);
  }
  else
  {
    num_cord = 0;
    boton [ID_LINEA-ID_EJEX].bChecarBoton = TRUE;
    SendDlgItemMessage (hWnd,ID_LINEA,BM_SETCHECK, (WPARAM) BST_CHECKED,0);
    objeto[num_objetos].ncoord = 2;
    objeto[num_objetos].tipo = 1;
    EnableWindow(hwndBoton[ID_ARCO_POS - ID_EJEX], FALSE);
```

```
  EnableWindow(hwndBoton[ID_ARCO_NEG - ID_EJEX], FALSE);
  EnableWindow(hwndBoton[ID_TRABAJO - ID_EJEX], TRUE);
  EnableWindow(hwndBoton[ID_POSICION - ID_EJEX], TRUE);
}
SendDlgItemMessage (hWnd,ID_TRABAJO,BM_SETCHECK, (WPARAM) BST_CHECKED - opcion,0);
break;
case ID_ARCO_POS:
if (BST_CHECKED == IsDlgButtonChecked (hWnd, ID_ARCO_POS))
{
  num_cord = 0;
  boton [ID_ARCO_POS-ID_EJEX].bChecarBoton = FALSE;
  EnableWindow(hwndBoton[ID_LINEA - ID_EJEX], TRUE);
  EnableWindow(hwndBoton[ID_ARCO_NEG - ID_EJEX], TRUE);
  EnableWindow(hwndBoton[ID_TRABAJO - ID_EJEX], FALSE);
  EnableWindow(hwndBoton[ID_POSICION - ID_EJEX], FALSE);
SendDlgItemMessage (hWnd,ID_ARCO_POS,BM_SETCHECK, (WPARAM) BST_UNCHECKED,0);
}
else
{
  num_cord = 0;
  boton [ID_ARCO_POS-ID_EJEX].bChecarBoton = TRUE;
  SendDlgItemMessage (hWnd,ID_ARCO_POS,BM_SETCHECK, (WPARAM) BST_CHECKED,0);
  objeto[num_objetos].ncoord = 3;
  objeto[num_objetos].tipo = 2;
  EnableWindow(hwndBoton[ID_LINEA - ID_EJEX], FALSE);
  EnableWindow(hwndBoton[ID_ARCO_NEG - ID_EJEX], FALSE);
  EnableWindow(hwndBoton[ID_TRABAJO - ID_EJEX], TRUE);
  EnableWindow(hwndBoton[ID_POSICION - ID_EJEX], TRUE);
}
SendDlgItemMessage (hWnd,ID_TRABAJO,BM_SETCHECK, (WPARAM) BST_CHECKED - opcion,0);
break;
case ID_ARCO_NEG:
if (BST_CHECKED == IsDlgButtonChecked (hWnd, ID_ARCO_NEG))
{
  num_cord = 0;
  boton [ID_ARCO_NEG-ID_EJEX].bChecarBoton = FALSE;
  EnableWindow(hwndBoton[ID_LINEA - ID_EJEX], TRUE);
  EnableWindow(hwndBoton[ID_ARCO_POS - ID_EJEX], TRUE);
  EnableWindow(hwndBoton[ID_TRABAJO - ID_EJEX], FALSE);
  EnableWindow(hwndBoton[ID_POSICION - ID_EJEX], FALSE);
SendDlgItemMessage (hWnd,ID_ARCO_NEG,BM_SETCHECK, (WPARAM) BST_UNCHECKED,0);
}
else
{
  num_cord = 0;
  boton [ID_ARCO_NEG-ID_EJEX].bChecarBoton = TRUE;
  SendDlgItemMessage (hWnd,ID_ARCO_NEG,BM_SETCHECK, (WPARAM) BST_CHECKED,0);
  objeto[num_objetos].ncoord = 3;
  objeto[num_objetos].tipo = 3;
  EnableWindow(hwndBoton[ID_ARCO_POS - ID_EJEX], FALSE);
  EnableWindow(hwndBoton[ID_LINEA - ID_EJEX], FALSE);
  EnableWindow(hwndBoton[ID_TRABAJO - ID_EJEX], TRUE);
  EnableWindow(hwndBoton[ID_POSICION - ID_EJEX], TRUE);
}
SendDlgItemMessage (hWnd,ID_TRABAJO,BM_SETCHECK, (WPARAM) BST_CHECKED - opcion,0);
break;
case ID_TRABAJO:
  if (BST_CHECKED == IsDlgButtonChecked (hWnd, ID_TRABAJO))
{
  opcion = PS;
  boton [ID_TRABAJO-ID_EJEX].bChecarBoton = FALSE;
  boton [ID_POSICION-ID_EJEX].bChecarBoton = TRUE;
}
else
{
  opcion = TR;
  boton [ID_TRABAJO-ID_EJEX].bChecarBoton = TRUE;
  boton [ID_POSICION-ID_EJEX].bChecarBoton = FALSE;
}
  SendDlgItemMessage (hWnd,ID_TRABAJO,BM_SETCHECK, (WPARAM) BST_CHECKED - opcion,0);
  SendDlgItemMessage (hWnd,ID_POSICION,BM_SETCHECK, (WPARAM) BST_UNCHECKED + opcion,0);
  break;
case ID_CARGAR:
  num_cord = 0;
  sprintf(codigosG,"");
  for(i = 0; i<num_objetos; i++)
  {
    if (i == 0)
    {
    sprintf(codigosG,"G00 Z%5.3f;",objeto[i].cd[0].z + 50.0);
    longitud = strlen(codigosG);
    codigosG[longitud] = 13;
    codigosG[longitud + 1] = 10;
    codigosG[longitud + 2] = '\0';
    sprintf(messg,"G00 X%5.3f Y%5.3f;",objeto[i].cd[0].x,objeto[i].cd[0].y);
    longitud = strlen(messg);
    messg[longitud] = 13;
    messg[longitud + 1] = 10;
    messg[longitud + 2] = '\0';
    sprintf(messg3,"G01 Z%5.3f F%i;",objeto[i].cd[0].z,(int)dVel[0]);
    longitud = strlen(messg3);
    messg3[longitud] = 13;
    messg3[longitud + 1] = 10;
    messg3[longitud + 2] = '\0';
    strcat(messg,messg3);
```

```
          strcat(codigosG,messg);
          }
          if(objeto[i].tipo == 1)
          {
            messg[0]= '\0' ;
            messg2[0]= '\0' ;
            if (i == 0)
              sprintf(messg,"G01 X%5.3f Y%5.3f Z%5.3f
F%i;",objeto[i].cd[1].x,objeto[i].cd[1].y,objeto[i].cd[1].z,(int)dVel[objeto[i].movto]);
            else
            {
              if (objeto[i].tipo != objeto[i-1].tipo)
                sprintf(messg,"G01 ");
              else
                sprintf(messg," ");
              strcat(messg,messg2) ;
              if (objeto[i].cd[1].x != objeto[i - 1].cd[ objeto[i - 1].ncoord - 1 ].x)
                sprintf(messg2,"X%5.3f ",objeto[i].cd[1].x);
              else
                sprintf(messg2," ");
              strcat(messg,messg2) ;
              if (objeto[i].cd[1].y != objeto[i - 1].cd[ objeto[i - 1].ncoord - 1].y)
                sprintf(messg2,"Y%5.3f ",objeto[i].cd[1].y);
              else
                sprintf(messg2," ");
              strcat(messg,messg2) ;
              if (objeto[i].cd[1].z != objeto[i - 1].cd[ objeto[i - 1].ncoord - 1 ].z)
                sprintf(messg2,"Z%5.3f ",objeto[i].cd[1].z);
              else
                sprintf(messg2," ");
              strcat(messg,messg2) ;
              if ((int)dVel[objeto[i].movto] != (int)dVel[objeto[ i - 1].movto])
                sprintf(messg2,"F%i;",(int)dVel[objeto[i].movto]);
              else
                sprintf(messg2,";");
              strcat(messg, messg2) ;
            }
            longitud = strlen(messg);
            messg[longitud] = 13;
            messg[longitud + 1] = 10;
            messg[longitud + 2] = '\0';
          }
          if(objeto[i].tipo == 2)
          {
            bEcTresPunts (objeto[i].cd[0].x, objeto[i].cd[0].y, objeto[i].cd[1].x, objeto[i].cd[1].y, objeto[i].cd[2].x,
objeto[i].cd[2].y);
            bSolSis (AA, BB, CC, DD, EE, FF);
            messg[0]= '\0' ;
            messg2[0]= '\0' ;
            if (i == 0)
              sprintf(messg,"G03 X%5.3f Y%5.3f I%5.3f J%5.3f F%i;",objeto[i].cd[2].x,objeto[i].cd[2].y,CX -
objeto[i].cd[0].x,CY - objeto[i].cd[0].y,(int)dVel[objeto[i].movto]);
            else
            {
              sprintf(messg,"G03 X%5.3f Y%5.3f I%5.3f J%5.3f",objeto[i].cd[2].x,objeto[i].cd[2].y,CX -
objeto[i].cd[0].x,CY - objeto[i].cd[0].y);
              if ((int)dVel[objeto[i].movto] != (int)dVel[objeto[i-1].movto])
                sprintf(messg2," F%i;",(int)dVel[objeto[i].movto]);
              else
                sprintf(messg2,";");
                strcat(messg, messg2) ;
            }
            longitud = strlen(messg);
            messg[longitud] = 13;
            messg[longitud + 1] = 10;
            messg[longitud + 2] = '\0';
          }
          if(objeto[i].tipo == 3)
          {
            bEcTresPunts (objeto[i].cd[0].x, objeto[i].cd[0].y, objeto[i].cd[1].x, objeto[i].cd[1].y, objeto[i].cd[2].x,
objeto[i].cd[2].y);
            bSolSis (AA, BB, CC, DD, EE, FF);
            messg[0]= '\0' ;
            messg2[0]= '\0' ;
            if (i == 0)
              sprintf(messg,"G02 X%5.3f Y%5.3f I%5.3f J%5.3f F%i;",objeto[i].cd[2].x,objeto[i].cd[2].y,CX -
objeto[i].cd[0].x,CY - objeto[i].cd[0].y,(int)dVel[objeto[i].movto]);
            else
            {
              sprintf(messg,"G02 X%5.3f Y%5.3f I%5.3f J%5.3f",objeto[i].cd[2].x,objeto[i].cd[2].y,CX -
objeto[i].cd[0].x,CY - objeto[i].cd[0].y);
              if ((int)dVel[objeto[i].movto] != (int)dVel[objeto[i-1].movto])
                sprintf(messg2," F%i;",(int)dVel[objeto[i].movto]);
              else
                sprintf(messg2,";");
              strcat(messg, messg2) ;
            }
            longitud = strlen(messg);
            messg[longitud] = 13;
            messg[longitud + 1] = 10;
            messg[longitud + 2] = '\0';
          }
          strcat(codigosG,messg);
        }
        Copy2Clipboard(hWnd, codigosG);
```

```
        SendMessage(hWndMain,WM_COMMAND,ID_MOVER_HTA,0L);
      break;
    case ID_P_EMERG:
      if (BST_CHECKED == IsDlgButtonChecked (hWnd, ID_P_EMERG))
      {
        boton [ID_P_EMERG-ID_EJEX].bChecarBoton = TRUE;
        SendDlgItemMessage (hWnd,ID_P_EMERG,BM_SETCHECK, (WPARAM) BST_UNCHECKED,0);
      }
      else
      {
        boton [ID_P_EMERG-ID_EJEX].bChecarBoton = FALSE;
        SendDlgItemMessage (hWnd,ID_P_EMERG,BM_SETCHECK, (WPARAM) BST_CHECKED,0);
      }
      break;
    case IDM_ABOUT:
      DialogBox(hInst, MAKEINTRESOURCE(IDD_ABOUTBOX), hWnd, About);
      break;
    case IDM_EXIT:
      KillTimer(hWndMain,TIMERMOUSE3D);
      DestroyWindow(hWnd);
      break;
    default:
      return DefWindowProc(hWnd, message, wParam, lParam);
    }
    break;
   case WM_PAINT:
     hdc = BeginPaint(hWnd, &ps);
     // TODO: Add any drawing code here...
     EndPaint(hWnd, &ps);
     break;
   case WM_DESTROY:
    PostQuitMessage(0);
    break;
  default:
    return DefWindowProc(hWnd, message, wParam, lParam);
  }
  return 0;
}
INT_PTR CALLBACK ProcDlgDNC(HWND hDlg, UINT nMsg, WPARAM wParam, LPARAM lParam)
{
  hWndDlg = hDlg;
  unsigned char uctrama[100]= "0000", uctrhi[100]= "0",uctrlow[100]= "0";
  double dvolts = 0.0;
  int idecimal= 0, itrhi= 0, itrlow= 0;
  unsigned int data=0;
  unsigned char a[2], b[1],buffer[200],sendb[14], *vptr;
  char *ptr;
  char err[80];
  int slen;
  char x[1];
  char bw;
  int i ;
  int longitud;
  b[0] = 0x31;
  vptr = (unsigned char*)&b;
  switch (nMsg)
  {
  case WM_INITDIALOG:
    return FALSE;
  case WM_COMMAND:
    switch (LOWORD(wParam))
    {
    case IDC_BUTTON2:
      inVeces = 0;
      GetDlgItemText(hDlg, IDC_EDIT1, codigosG, 10000);
      SetTimer(hWndDlg, TIMERDNC, DELTATIMEDNC, EnvioDatos);
      EnableWindow(GetDlgItem(hDlg, IDC_BUTTON2), FALSE);
      return FALSE;
    case IDC_BUTTON4:
      KillTimer(hWndDlg, TIMERDNC);
      EnableWindow(GetDlgItem(hDlg, IDC_BUTTON2), TRUE);
      return FALSE;
    case IDOK: //
      EndDialog(hDlg, 0);
      return TRUE;
    case IDCANCEL:
      EndDialog(hDlg, 0);
      return TRUE;
    }
    break;
  }
  return FALSE;
}
BOOL ConectarPuertoSerial(HWND hWnd)
{
  char messg[1000];
  BOOL bresultado = FALSE;
      hComm = CreateFile((LPCSTR)sPortName[0],
      GENERIC_READ | GENERIC_WRITE,
      0,
      0,
      OPEN_EXISTING,
      FILE_FLAG_OVERLAPPED,
      0);
      if(hComm == INVALID_HANDLE_VALUE)
```

```
      {
        gle = GetLastError();
        sprintf(messg, "error no.%d\n", gle);
        MessageBox(hWnd, messg, "Cannot open Communication Port..", MB_OK | MB_ICONEXCLAMATION) ;
      }
      else
      {
        if(!ConfigurePortCOM(115200, 7, 0,EVENPARITY, ONESTOPBIT))
        {
          MessageBox(hWnd, "Cannot Configure Communication Port", "Error", MB_OK | MB_ICONEXCLAMATION) ;
          CloseHandle(hComm);
        }
        else
        {
          bresultado = TRUE;
          if(!SetCommunicationTimeouts(1000,1000,1000,1000,1000))
          {
            MessageBox(hWnd, "Cannot Configure Communication Timeouts", "Error", MB_OK | MB_ICONEXCLAMATION) ;
            CloseHandle(hComm);
            bresultado = FALSE;
          }

        }
      }
      return bresultado;
}

INT_PTR CALLBACK ProcDlgCommSettings(HWND hDlg, UINT nMsg, WPARAM wParam, LPARAM lParam)
{
  char messg[2000];
  unsigned int data=0;
  unsigned char a[2], b[1],buffer[200],sendb[14], *vptr;
  char *ptr;
  char err[80];
  int slen;
  char x[1];
  char bw;
  int i ;
  b[0] = 0x31;
  vptr = (unsigned char*)&b;
  switch (nMsg)
  {
  case WM_INITDIALOG:
    SendDlgItemMessage (hDlg, IDC_LIST1, LB_ADDSTRING, 0, (LPARAM)sPortName[PORT1]);
    SendDlgItemMessage (hDlg, IDC_LIST1, LB_ADDSTRING, 0, (LPARAM)sPortName[PORT2]);
    SendDlgItemMessage (hDlg, IDC_LIST1, LB_ADDSTRING, 0, (LPARAM)sPortName[PORT3]);
    SendDlgItemMessage (hDlg, IDC_LIST1, LB_ADDSTRING, 0, (LPARAM)sPortName[PORT4]);
    SendDlgItemMessage (hDlg, IDC_LIST1, LB_ADDSTRING, 0, (LPARAM)sPortName[PORT5]);
    SendDlgItemMessage (hDlg, IDC_LIST1, LB_ADDSTRING, 0, (LPARAM)sPortName[PORT6]);
    SendDlgItemMessage (hDlg, IDC_LIST1, LB_ADDSTRING, 0, (LPARAM)sPortName[PORT7]);
    SendDlgItemMessage (hDlg, IDC_LIST1, LB_ADDSTRING, 0, (LPARAM)sPortName[PORT8]);
    SendDlgItemMessage (hDlg, IDC_LIST1, LB_ADDSTRING, 0, (LPARAM)sPortName[PORT9]);
    SendDlgItemMessage (hDlg, IDC_LIST1, LB_ADDSTRING, 0, (LPARAM)sPortName[PORT10]);
    SendDlgItemMessage (hDlg, IDC_LIST1, LB_ADDSTRING, 0, (LPARAM)sPortName[PORT11]);
    SendDlgItemMessage (hDlg, IDC_LIST1, LB_ADDSTRING, 0, (LPARAM)sPortName[PORT12]);
    SendDlgItemMessage (hDlg, IDC_LIST1, LB_SELECTSTRING, (UINT)-1, (LPARAM)sPortName[PORT1]);
    return (INT_PTR)FALSE;
  case WM_COMMAND:
    switch (LOWORD(wParam))
    {
    case IDC_BUTTON1:
      iIndexPort = SendDlgItemMessage(hDlg, IDC_LIST1, LB_GETCURSEL, 0, 0);
      hComm = CreateFile((LPCSTR)sPortName[iIndexPort],
      GENERIC_READ | GENERIC_WRITE,
      0,
      0,
      OPEN_EXISTING,
      FILE_FLAG_OVERLAPPED,
      0);

      if(hComm == INVALID_HANDLE_VALUE)
      {
        gle = GetLastError();
        sprintf(messg, "error no.%d\n", gle);
        MessageBox(hDlg, messg, "Cannot open Communication Port..", MB_OK | MB_ICONEXCLAMATION) ;
      }
      else
      {
        EnableWindow( GetDlgItem( hDlg, IDC_LIST1 ), FALSE);
        EnableWindow( GetDlgItem( hDlg, IDC_BUTTON1 ), FALSE);
        if(!ConfigurePortCOM(115200, 7, 0,EVENPARITY, ONESTOPBIT))
        {
          MessageBox(hDlg, "Cannot Configure Communication Port", "Error", MB_OK | MB_ICONEXCLAMATION) ;
          CloseHandle(hComm);
        }
        else
        {

          if(!SetCommunicationTimeouts(1000,1000,1000,1000,1000))
          {
            MessageBox(hDlg, "Cannot Configure Communication Timeouts", "Error", MB_OK | MB_ICONEXCLAMATION) ;
            CloseHandle(hComm);
          }
        }
      }
```

```
      return (INT_PTR)FALSE;
    case IDOK: //Obtener datos del cuadro de diálogo

//      EnableMenuItem(hmMenus[MENUPRINCIPAL], ID_GRPHICAS, MF_BYCOMMAND | MF_ENABLED);
      EnableWindow( GetDlgItem( hDlg, IDC_LIST1 ), TRUE);
      EnableWindow( GetDlgItem( hDlg, IDC_BUTTON1 ), TRUE);
//      CloseHandle(hComm);
      EndDialog(hDlg, 0);
      return (INT_PTR)TRUE;

    case IDCANCEL:
      EndDialog(hDlg, 0);
      return (INT_PTR)TRUE;

    }
    break;
  }
  return (INT_PTR)FALSE;
}
// Message handler for about box.
INT_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
  UNREFERENCED_PARAMETER(lParam);
  switch (message)
  {
  case WM_INITDIALOG:
    return (INT_PTR)TRUE;

    case WM_COMMAND:
    switch (LOWORD(wParam))
    {
    case IDOK:


    case IDCANCEL:

      EndDialog(hDlg, LOWORD(wParam));

      return (INT_PTR)TRUE;
    }
    break;
  }
  return (INT_PTR)FALSE;
}
//IDD_DIALOG5
INT_PTR CALLBACK InsertarCoordenadas(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
  UNREFERENCED_PARAMETER(lParam);
  char c_inicial[80];
  switch (message)
  {
  case WM_INITDIALOG:
    SetWindowText(GetDlgItem(hDlg,IDC_STATIC1),"Insertar coordenadas pieza");
    return (INT_PTR)TRUE;

  case WM_COMMAND:
    switch(LOWORD (wParam))
    {
    case IDOK:
      GetDlgItemText(hDlg,IDC_EDIT1,c_inicial,12);
      if(strlen(c_inicial) != 0)
      fctemp[0] = atof(c_inicial);
      GetDlgItemText(hDlg,IDC_EDIT2,c_inicial,12);
      if(strlen(c_inicial) != 0)
      fctemp[1] = atof(c_inicial);
      GetDlgItemText(hDlg,IDC_EDIT3,c_inicial,12);
      if(strlen(c_inicial) != 0)
      fctemp[2] = atof(c_inicial);
    case IDCANCEL:
      EndDialog(hDlg, LOWORD(wParam));
      return (INT_PTR)TRUE;
    }
    break;
  }
  return (INT_PTR)FALSE;
}
INT_PTR CALLBACK InsertarCoordenadasMaq(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
  UNREFERENCED_PARAMETER(lParam);
  char c_inicial[80], cadena1[100], cadena2[100], cadena3[100];
  BOOL berror;
  switch (message)
  {
  case WM_INITDIALOG:
    SetWindowText(GetDlgItem(hDlg,IDC_STATIC1),"Insertar coordenadas máquina");
    SetDlgItemText(hDlg,IDC_EDIT1,"636.397");
    SetDlgItemText(hDlg,IDC_EDIT2,"-271.548");
    SetDlgItemText(hDlg,IDC_EDIT3,"-582.981");
    return (INT_PTR)TRUE;
  case WM_COMMAND:
    switch(LOWORD (wParam))
    {
    case IDOK:
      bCoordMaq = TRUE ;
      berror = FALSE;
```

```
      sprintf(cadena1,"La coordenada ");
      GetDlgItemText(hDlg,IDC_EDIT1,c_inicial,12);
      CoAbCePi[0] = atof(c_inicial);
      cabsol[0] = CoAbCePi[0];
      if(CoAbCePi[0] > DMX || CoAbCePi[0] < 0)
      {
        sprintf(cadena2,"X ");
        strcat(cadena1,cadena2);
        berror = TRUE;
      }

      GetDlgItemText(hDlg,IDC_EDIT2,c_inicial,12);
      CoAbCePi[1] = atof(c_inicial);
      cabsol[1] = CoAbCePi[1];
      if(CoAbCePi[1] < DMY || CoAbCePi[1] > 0)
      {
        sprintf(cadena2,"Y ");
        strcat(cadena1,cadena2);
        berror = TRUE;
      }

      GetDlgItemText(hDlg,IDC_EDIT3,c_inicial,12);
      CoAbCePi[2] = atof(c_inicial);
      cabsol[2] = CoAbCePi[2];
      if(CoAbCePi[2] < DMZ || CoAbCePi[2] > 0)
      {
        sprintf(cadena2,"Z ");
        strcat(cadena1,cadena2);
        berror = TRUE;
      }
      if(berror)
      {
        strcat(cadena1," esta fuera del espacio de trabajo");
        SetDlgItemText(hDlg,IDC_EDIT4,cadena1);
      }
      else
      {
        SetDlgItemText(hDlg,IDC_EDIT4,"");
        EndDialog(hDlg, LOWORD(wParam));
      }
      return (INT_PTR)FALSE;

    case IDCANCEL:
      CoAbCePi[0] = 0.0;
      CoAbCePi[1] = 0.0;
      CoAbCePi[2] = 0.0;
      EndDialog(hDlg, LOWORD(wParam));
      return (INT_PTR)TRUE;
    }
    break;
  }
  return (INT_PTR)FALSE;
}
INT_PTR CALLBACK TresDConnexion(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
  UNREFERENCED_PARAMETER(lParam);
  switch (message)
  {
  case WM_INITDIALOG:
    return (INT_PTR)TRUE;

  case WM_COMMAND:
    switch(LOWORD (wParam))
    {
    case IDOK:
    case IDCANCEL:
      EndDialog(hDlg, LOWORD(wParam));
      return (INT_PTR)TRUE;
    }
    break;
  }
  return (INT_PTR)FALSE;
}
INT_PTR CALLBACK ProcDlgCircun(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
  UNREFERENCED_PARAMETER(lParam);
  char cadena[100];
  switch (message)
  {
  case WM_INITDIALOG:
    return (INT_PTR)TRUE;

  case WM_COMMAND:
    switch(LOWORD (wParam))
    {
    case IDOK:
      GetDlgItemText (hDlg, IDC_EDIT1,cadena,20);
      PX=atof(cadena);
      GetDlgItemText (hDlg, IDC_EDIT2,cadena,20);
      PY=atof(cadena);
      GetDlgItemText (hDlg, IDC_EDIT3,cadena,20);
      QX=atof(cadena);
      GetDlgItemText (hDlg, IDC_EDIT4,cadena,20);
      QY=atof(cadena);
      GetDlgItemText (hDlg, IDC_EDIT5,cadena,20);
```

```
      RX=atof(cadena);
      GetDlgItemText (hDlg, IDC_EDIT6,cadena,20);
      RY=atof(cadena);
      bEcTresPunts (PX, PY, QX, QY, RX, RY);
      bSolSis (AA, BB, CC, DD, EE, FF);
      sprintf (cadena, "X=%5.5f, Y=%5.5f", CX,CY);
      SetDlgItemText(hDlg, IDC_EDIT7, cadena);
    break;
    case IDCANCEL:
      EndDialog(hDlg, LOWORD(wParam));
      return (INT_PTR)TRUE;
    }
    break;
  }
  return (INT_PTR)FALSE;
}
INT_PTR CALLBACK ProcDlgCircyLinea(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
  UNREFERENCED_PARAMETER(lParam);
  char cadena[100];
  double l1x, l1y, l2x, l2y, h, k, r, p1x, p1y, p2x, p2y;
  switch (message)
  {
  case WM_INITDIALOG:
    return (INT_PTR)TRUE;
  case WM_COMMAND:
    switch(LOWORD (wParam))
    {
    case IDOK:
      GetDlgItemText (hDlg, IDC_EDIT1,cadena,20);
      l1x = atof(cadena);
      GetDlgItemText (hDlg, IDC_EDIT2,cadena,20);
      l1y = atof(cadena);
      GetDlgItemText (hDlg, IDC_EDIT3,cadena,20);
      l2x = atof(cadena);
      GetDlgItemText (hDlg, IDC_EDIT4,cadena,20);
      l2y = atof(cadena);
      GetDlgItemText (hDlg, IDC_EDIT5,cadena,20);
      h = atof(cadena);
      GetDlgItemText (hDlg, IDC_EDIT6,cadena,20);
      k = atof(cadena);
      GetDlgItemText (hDlg, IDC_EDIT7,cadena,20);
      r = atof(cadena);
      InterCircToLine (&l1x, &l1y, &l2x, &l2y, &h, &k, &r, &p1x, &p1y, &p2x, &p2y);
      sprintf (cadena, "X=%5.5f, Y=%5.5f", p1x,p1y);
      SetDlgItemText(hDlg, IDC_EDIT8, cadena);
      sprintf (cadena, "X=%5.5f, Y=%5.5f", p2x,p2y);
      SetDlgItemText(hDlg, IDC_EDIT9, cadena);
    break;
    case IDCANCEL:
      EndDialog(hDlg, LOWORD(wParam));
      return (INT_PTR)TRUE;
    }
    break;
  }
  return (INT_PTR)FALSE;
}
BOOL bEcTresPunts (double Px, double Py, double Qx, double Qy, double Rx, double Ry)
{
  AA= - 2.0 * Px + 2.0 * Qx;
  BB= - 2.0 * Py + 2.0 * Qy;
  DD= - 2.0 * Qx + 2.0 * Rx;
  EE= - 2.0 * Qy + 2.0 * Ry;
  CC= - pow(Px,2.0) - pow(Py,2.0) + pow(Qx,2.0) + pow(Qy,2.0);
  FF= - pow(Qx,2.0) - pow(Qy,2.0) + pow(Rx,2.0) + pow(Ry,2.0);
  return FALSE;
}
BOOL bSolSis (double Aa, double Bb, double Cc, double Dd, double Ee, double Ff)
{
  double det;
  BOOL bCond = FALSE;
  det=(Aa * Ee - Bb * Dd);
  if (det != 0.0)
  {
  CX= (- (Bb * Ff) + (Cc * Ee)) / det;
  CY= ( (Aa * Ff) - (Dd * Cc)) / det;
  bCond = TRUE;
  }
  else
  {
  CX = 1.0E20;
  CY = 1.0E20;
  }
  return bCond;
}
BOOL bP2C (double *mag, double *ang, double *Cx, double *Cy)
{
  BOOL bCond = FALSE;
  *Cx = (*mag) * cos(*ang);
  *Cy = (*mag) * sin(*ang);
  return bCond;
}
BOOL bC2P (double *Cx, double *Cy, double *mag, double *ang)
{
  BOOL bCond = FALSE;
```

```
  double signo, den;
  if (*Cy != 0.0)
  {
  signo = *Cy / fabs(*Cy);
  }
  else
  {
    signo = 1.0;
  }
  *mag = sqrt(pow(*Cx,2.0) + pow(*Cy,2.0));
  den = *Cy;
  if (den == 0.0)
  {
    den = 1E-10;
  }
  *ang = (dpi / 2.0) * signo - atan(*Cx / den);
  return bCond;
}
BOOL InterCircToLine (double *p1x, double *p1y, double *p2x, double *p2y, double *h,
            double *k, double *radio,
            double *s1x, double *s1y, double *s2x, double *s2y)
{
  BOOL bCond = FALSE;
  double angline, angcentro, magcentro, difcx, difcy, magline, temp1, temp2, temp3, temp4;
  difcx = (*h) - (*p1x);
  difcy = (*k) - (*p1y);
  bC2P (&difcx, &difcy, &magcentro, &angcentro);
  difcx = (*p2x) - (*p1x);
  difcy = (*p2y) - (*p1y);
  bC2P (&difcx, &difcy, &magline, &angline);
  temp1 = (angline) - asin((magcentro * sin(angline - angcentro)) / (*radio));
  temp2 = (angline) + asin((magcentro * sin(angline - angcentro)) / (*radio)) + dpi;
  temp3 = magcentro * cos(angline - angcentro) - (*radio) * cos(angline - temp1);
  temp4 = magcentro * cos(angline - angcentro) - (*radio) * cos(angline - temp2);
  bP2C (&temp3, &angline, &difcx, &difcy);
  *s1x = (*p1x) + difcx;
  *s1y = (*p1y) + difcy;
  bP2C (&temp4, &angline, &difcx, &difcy);
  *s2x = (*p1x) + difcx;
  *s2y = (*p1y) + difcy;
  return bCond;
}
int SbInit()
{
  int res=0;                            // result of SiOpen, to be returned
  SiOpenData oData;                    // OS Independent data to open ball
  //init the SpaceWare input library
  if (SiInitialize() == SPW_DLL_LOAD_ERROR)
     {
    MessageBox(NULL,"Error: No se pueden cargar los archivos dll SiAppDll",
              NULL, MB_ICONEXCLAMATION);
   }
  SiOpenWinInit (&oData, hWndMain);    // init Win. platform specific data
  SiSetUiMode(devHdl, SI_UI_ALL_CONTROLS); // Config SoftButton Win Display
  // open data, which will check for device type and return the device handle
   //  to be used by this function
  if ( (devHdl = SiOpen ("3DxTest32", SI_ANY_DEVICE, SI_NO_MASK,
                         SI_EVENT, &oData)) == NULL)
     {
     SiTerminate();  // called to shut down the SpaceWare input library
     res = 0;        // could not open device
     return res;
     }
  else
     {
     res = 1;        // opened device succesfully //
     return res;
     }
  //      return res;
}
void SbMotionEvent(SiSpwEvent *pEvent)
{
  char buff0[80];                              // text buffer for TX
  char buff1[80];                              // text buffer for TY
  char buff2[80];                              // text buffer for TZ
  char buff3[20];                              // text buffer for RX
  char buff4[20];                              // text buffer for RY
  char buff5[20];                              // text buffer for RZ
  char buff6[100];
  //char buff6[100];            //buffer de texto para mostrar pixeles
   int len0,len1,len2,len3,len4,len5,len6;        // length of each buffer
   //RECT rtam;
   // put the actual ball data into the buffers
  if (boton[0].bChecarBoton == TRUE)
  {
  boton [0].iCoordraton = (int)pEvent->u.spwData.mData[SI_TX];
  }
  else
  {
    boton [0].iCoordraton = 0;
  }

  if (boton[1].bChecarBoton == TRUE)
  {
  boton [1].iCoordraton = (int)pEvent->u.spwData.mData[SI_TY];
```

```
  }
  else
  {
    boton [1].iCoordraton = 0;
  }

  if (boton[2].bChecarBoton == TRUE)
  {
  boton [2].iCoordraton = (int)-pEvent->u.spwData.mData[SI_TZ];
  }
  else
  {
    boton [2].iCoordraton = 0;
  }
  if (boton[ID_P_EMERG - ID_EJEX].bChecarBoton == TRUE)
  {
  len0= sprintf( buff0, "Desplazamiento X: %d         ", boton [0].iCoordraton );
  len1= sprintf( buff1, "Desplazamiento Y: %d         ", boton[1].iCoordraton );
  len2= sprintf( buff2, "Desplazamiento Z: %d         ", boton[2].iCoordraton );
  len3= sprintf( buff3, "X: %5.3f            ", crel[0]);
  len4= sprintf( buff4, "Y: %5.3f            ", crel[1]);
  len5= sprintf( buff5, "Z: %5.3f            ", crel[2]);
  }
  else
  {
    SbZeroEvent();
  }
   // get handle of our window to draw on
  hdc = GetDC(hWndMain);
    // print buffers
   TextOut(hdc, 400 , 80, "COORDENADAS MOUSE ", 18);
   TextOut(hdc, 410 , 105, buff0, len0);
  TextOut(hdc, 410 , 125, buff1, len1);
  TextOut(hdc, 410 , 145, buff2, len2);
  TextOut(hdc, 400 , 170, "COORDENADAS MÁQUINA ", 20);
  TextOut(hdc, 410 , 195, buff3, len3);
  TextOut(hdc, 410 , 215, buff4, len4);
  TextOut(hdc, 410 , 235, buff5, len5);
  TextOut(hdc, 400 , 260, "CÓDIGO MÁQUINA ", 15);
  ReleaseDC(hWndMain,hdc);
}
void CALLBACK    CapturarCoordenadas3D(HWND hWnd,UINT ms, UINT id,DWORD time)
{
  //boton [0].iCoordraton
  char messg[5000], messg2[5000];
  int longitud;
  char err[80],cadena1[1000], cadena2[1000];
  BOOL blimite[4] = {FALSE,FALSE,FALSE,FALSE};
  double ddivisorxy = 25.0,  ddivisorz = 100.0;
  hdc = GetDC(hWndMain);
  if(boton [ID_LINEA-ID_EJEX].bChecarBoton || boton [ID_ARCO_POS - ID_EJEX].bChecarBoton || boton [ID_ARCO_NEG -
ID_EJEX].bChecarBoton)
  {
    sprintf(messg,"inserte coord :[%i]      ",num_cord + 1);
    TextOut(hdc, 300 , 0, messg, 20);
  }
  else
    TextOut(hdc, 300 , 0, "                                                    ", 30);
  sprintf(cadena2,"");
  if((CoAbCePi[0] + fctemp[0] + (boton [0].iCoordraton / ddivisorxy)) >= DMX || (CoAbCePi[0] + fctemp[0] + (boton
[0].iCoordraton / ddivisorxy)) < 0.0)
  {
    strcat(cadena2,"X ");
    blimite[0] = blimite[3] = TRUE;
  }
  else
  {
    fctemp[0] = fctemp[0] + (boton [0].iCoordraton / ddivisorxy);
    crel[0] = fctemp[0];
  }
  if((CoAbCePi[1] + fctemp[1] + (boton [1].iCoordraton / ddivisorxy)) <= DMY || (CoAbCePi[1] + fctemp[1] + (boton
[1].iCoordraton / ddivisorxy)) > 0.0)
  {
    strcat(cadena2,"Y ");
    blimite[1] = blimite[3] = TRUE;
  }
  else
  {
    fctemp[1] = fctemp[1] + (boton [1].iCoordraton / ddivisorxy);
    crel[1] = fctemp[1];
  }
  if((CoAbCePi[2] + fctemp[2] + (boton [2].iCoordraton / ddivisorz)) <= DMZ || (CoAbCePi[2] + fctemp[2] + (boton
[2].iCoordraton / ddivisorz)) > 0.0)
  {
    strcat(cadena2,"Z ");
    blimite[2] = blimite[3] = TRUE;
  }
  else
  {
    fctemp[2] = fctemp[2] + (boton [2].iCoordraton / ddivisorz);
    crel[2] = fctemp[2];
  }
  cabsol[0] = CoAbCePi[0] + fctemp[0];
  cabsol[1] = CoAbCePi[1] + fctemp[1];
  cabsol[2] = CoAbCePi[2] + fctemp[2];
```

```
  sprintf(cadena1,"                                ");
  if(blimite[3] == TRUE)
  {
    sprintf(cadena1,"Límite ");
    strcat(cadena1,cadena2);
  }
  TextOut(hdc,400,310,cadena1,strlen(cadena1));
  sprintf(cadena1,"-----------------------------------------------------------------");
  if (boton [0].iCoordraton != 0 || boton [1].iCoordraton != 0 || boton [2].iCoordraton != 0)
  {
    strcpy(dsend,"G01");
    if (boton [ID_EJEX-ID_EJEX].iCoordraton != 0 && blimite[0] == FALSE)
    {
      sprintf(messg," X%4.3f",crel[0]);
      strcat(dsend,messg);
    }
    if (boton [ID_EJEY-ID_EJEX].iCoordraton != 0 && blimite[1] == FALSE)
    {
      sprintf(messg," Y%4.3f",crel[1]);
      strcat(dsend,messg);
    }
    if (boton [ID_EJEZ-ID_EJEX].iCoordraton != 0 && blimite[2] == FALSE)
    {
      sprintf(messg," Z%4.3f",crel[2]);
      strcat(dsend,messg);
    }
    //sprintf(messg," F8000;");
    sprintf(messg," F%i;", (int)dVel[2]) ;
    if (boton [0].iCoordraton != 0 || boton [1].iCoordraton != 0 || boton [2].iCoordraton != 0)
      strcat(dsend,messg);
    longitud = strlen(dsend);
    dsend[longitud] = 13;
    dsend[longitud+1] = '\0';
    sprintf(messg2,"GenCoG 3D       (%s)", dsend);
    SetWindowText(hWndMain,messg2);
    strcpy(cadena1,dsend) ;
    if(!WriteABuffer())
    {
      int gle = GetLastError();
      sprintf(err, "Cannot Write %c to %s\n%x", dsend, sPortName[iIndexPort], gle);
      MessageBox(NULL, err, "Error", MB_OK | MB_ICONEXCLAMATION) ;
      CloseHandle(hComm);
    }
    strcpy(messg,dsend);
    longitud = strlen(messg);
    messg[longitud ] = 10;
    messg[longitud + 1] = '\0';
    strcpy(Trayectoria[nInstruc],messg);
    nInstruc++;
  }
  TextOut(hdc, 410 , 285, cadena1, strlen(cadena1));
}
void
SbZeroEvent()
{
   // get handle of our window to draw on
   hdc = GetDC(hWndMain);
   boton [0].iCoordraton=0;
   boton [1].iCoordraton=0;
   boton [2].iCoordraton=0;
   TextOut(hdc, 410 , 80, "S M          ", 20);
   //release our window handle
   ReleaseDC(hWndMain,hdc);
}
void
SbButtonPressEvent(int buttonnumber)
{
   char mensaje[1000] = "", mensaje2[1000] = "";
   int i;
  // get handle of our window to draw on
   hdc = GetDC(hWndMain);
   // print button pressed(does not include rezero button)
   switch (buttonnumber)
      {
      case SI_APP_FIT_BUTTON:           // #31 defined in si.h
        //TextOut(hdc, 0 , 0, "Fit Button Pressed L ", 20);
     if (boton[ID_LINEA - ID_EJEX].bChecarBoton == TRUE)
     {
       if (objeto[num_objetos].ncoord > (num_cord - 1))
       {
         if(num_cord == 0 && num_objetos > 0)
         {
          objeto[num_objetos].cd[num_cord].x = objeto[num_objetos - 1].cd[objeto[num_objetos - 1].ncoord - 1].x;
          objeto[num_objetos].cd[num_cord].y = objeto[num_objetos - 1].cd[objeto[num_objetos - 1].ncoord - 1].y;
          objeto[num_objetos].cd[num_cord].z = objeto[num_objetos - 1].cd[objeto[num_objetos - 1].ncoord - 1].z;
         }
         else
         {
           objeto[num_objetos].cd[num_cord].x = crel[0];
           objeto[num_objetos].cd[num_cord].y = crel[1];
           objeto[num_objetos].cd[num_cord].z = crel[2];
         }
        num_cord ++;
       }
       if (objeto[num_objetos].ncoord == num_cord)
```

```
        {
         objeto[num_objetos].movto = opcion;
         num_objetos ++;
         SendMessage(hWndMain,WM_COMMAND,ID_LINEA,0L);
        }
      }
      if (boton[ID_ARCO_POS - ID_EJEX].bChecarBoton == TRUE)
      {
        if (objeto[num_objetos].ncoord > (num_cord - 1))
        {
          if(num_cord == 0 && num_objetos > 0)
          {
           objeto[num_objetos].cd[num_cord].x = objeto[num_objetos - 1].cd[objeto[num_objetos - 1].ncoord - 1].x;
           objeto[num_objetos].cd[num_cord].y = objeto[num_objetos - 1].cd[objeto[num_objetos - 1].ncoord - 1].y;
           objeto[num_objetos].cd[num_cord].z = objeto[num_objetos - 1].cd[objeto[num_objetos - 1].ncoord - 1].z;
          }
          else
          {
           objeto[num_objetos].cd[num_cord].x = crel[0];
           objeto[num_objetos].cd[num_cord].y = crel[1];
           objeto[num_objetos].cd[num_cord].z = crel[2];
          }
         num_cord ++;
        }
        if (objeto[num_objetos].ncoord == num_cord)
        {
         objeto[num_objetos].movto = opcion;
         num_objetos ++;
         SendMessage(hWndMain,WM_COMMAND,ID_ARCO_POS,0L);
        }
      }
      if (boton[ID_ARCO_NEG - ID_EJEX].bChecarBoton == TRUE)
      {
        if (objeto[num_objetos].ncoord > (num_cord - 1))
        {
          if(num_cord == 0 && num_objetos > 0)
          {
           objeto[num_objetos].cd[num_cord].x = objeto[num_objetos - 1].cd[objeto[num_objetos - 1].ncoord - 1].x;
           objeto[num_objetos].cd[num_cord].y = objeto[num_objetos - 1].cd[objeto[num_objetos - 1].ncoord - 1].y;
           objeto[num_objetos].cd[num_cord].z = objeto[num_objetos - 1].cd[objeto[num_objetos - 1].ncoord - 1].z;
          }
          else
          {
          //MessageBox(NULL,"","",NULL);
           objeto[num_objetos].cd[num_cord].x = crel[0];
           objeto[num_objetos].cd[num_cord].y = crel[1];
           objeto[num_objetos].cd[num_cord].z = crel[2];
          }
         num_cord ++;
        }
        if (objeto[num_objetos].ncoord == num_cord)
        {
         objeto[num_objetos].movto = opcion;
         num_objetos ++;
         SendMessage(hWndMain,WM_COMMAND,ID_ARCO_NEG,0L);
        }
      }
         break;
       case 1:
         //TextOut(hdc, 0 , 0, "Button 1 Pressed ", 17);
       if (boton[ID_LINEA - ID_EJEX].bChecarBoton == TRUE || boton[ID_ARCO_POS - ID_EJEX].bChecarBoton == TRUE ||
boton[ID_ARCO_NEG - ID_EJEX].bChecarBoton == TRUE)
       {
         SendMessage(hWndMain,WM_COMMAND,ID_TRABAJO,0L);
       }
         break;
       default:
         // TextOut(hdc, 0 , 0, "Button ? Pressed ", 17);
       MessageBox (NULL,"Favor de configurar la aplicación en el programa 3Dconnexion","AVISO",NULL);
         break;
       }
     //release our window handle
     ReleaseDC(hWndMain,hdc);
}
void
SbButtonReleaseEvent(int buttonnumber)
{
   // get handle of our window to draw on
   hdc = GetDC(hWndMain);

   // print button pressed(does not include rezero button)
   switch (buttonnumber)
      {
      case SI_APP_FIT_BUTTON:           // #31 defined in si.h
        // TextOut(hdc, 0 , 0, "Fit Button Released", 20);
         break;
      case 1:
         TextOut(hdc, 0 , 0, "Button 1 Released", 17);
         break;
      case 2:
         TextOut(hdc, 0 , 0, "Button 2 Released", 17);
         break;
      case 3:
         TextOut(hdc, 0 , 0, "Button 3 Released", 17);
         break;
```

```
      case 4:
         TextOut(hdc, 0 , 0, "Button 4 Released", 17);
         break;
      case 5:
         TextOut(hdc, 0 , 0, "Button 5 Released", 17);
         break;
      case 6:
         TextOut(hdc, 0 , 0, "Button 6 Released", 17);
         break;
      case 7:
         TextOut(hdc, 0 , 0, "Button 7 Released", 17);
         break;
      case 8:
         TextOut(hdc, 0 , 0, "Button 8 Released", 17);
         break;
       default:
         TextOut(hdc, 0 , 0, "Button ? Released", 17);
         break;
       }
    //release our window handle
    ReleaseDC(hWndMain,hdc);
}
//char CaracteresPCb[10000];
void Copy2Clipboard(HWND hwnd, char *pszData)
{
  HGLOBAL hData;
  LPVOID pData;
  OpenClipboard(hwnd);
  EmptyClipboard();
  hData = GlobalAlloc(GMEM_DDESHARE | GMEM_MOVEABLE, strlen(pszData) + 1);
  pData = GlobalLock(hData);
  strcpy((LPSTR)pData, pszData);
  GlobalUnlock(hData);
  SetClipboardData(CF_TEXT, hData);
  CloseClipboard();
}
INT_PTR CALLBACK ProcDlgConfiguracion(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
  UNREFERENCED_PARAMETER(lParam);
  static char mssg[4][80];
  static BOOL berror[3] = {FALSE, FALSE, FALSE};
  switch (message)
  {
  case WM_INITDIALOG:
    sprintf(mssg[1],"%5.3f",dVel[0]);
    SetDlgItemText(hDlg, IDC_EDIT1, mssg[1]);
    sprintf(mssg[2],"%5.3f",dVel[2]);
    SetDlgItemText(hDlg, IDC_EDIT2, mssg[2]);
    return (INT_PTR)TRUE;
    case WM_COMMAND:
    switch (LOWORD(wParam))
    {
    case IDOK:
      GetDlgItemText(hDlg, IDC_EDIT1, mssg[1], 12) ;
      GetDlgItemText(hDlg, IDC_EDIT2, mssg[2], 12) ;
      if(atof(mssg[1]) < 1.0 || atof(mssg[1])> 14000.0)
        berror[1] = TRUE ;
      else
        berror[1] = FALSE ;
      if(atof(mssg[2]) < 1.0 || atof(mssg[2])> 14000.0)
        berror[2] = TRUE ;
      else
        berror[2] = FALSE ;
      if(berror[1] || berror[2])
        MessageBox(hDlg, "La velocidad de avance esta fuera del rango 1-14000 mm/min", "AVISO", NULL);
      else
      {
        dVel[0] = atof(mssg[1]) ;
        dVel[2] = atof(mssg[2]) ;
        EndDialog(hDlg, LOWORD(wParam));
      }
      return (INT_PTR)TRUE;
    case IDCANCEL:
      EndDialog(hDlg, LOWORD(wParam));
      return (INT_PTR)TRUE;
    }
    break;
  }
  return (INT_PTR)FALSE;
}
void CALLBACK  EnvioDatos(HWND hWnd,UINT ms, UINT id,DWORD time)
{
  int ii, inbloques;
  char cadena1[1000]="", cadena2[1000];
  char err[80];
  int longitud;
  dsend[0] = '\0';
  inbloques = (int)ceil(strlen(codigosG) / 255.0) ;
  for(ii=0; ii < 255; ii++)
  {
    cadena1[ii] = codigosG[ii + (inVeces * 255)] ;
  }
  strcpy(dsend,cadena1);
  longitud = strlen(dsend);
  SetDlgItemText(hWndDlg, IDC_EDIT2, dsend) ;
```

```
  if(!WriteABuffer())
  {
    int gle = GetLastError();
    sprintf(err, "Cannot Write %c to %s\n%x", dsend, sPortName[iIndexPort], gle);
    MessageBox(NULL, err, "Error", MB_OK | MB_ICONEXCLAMATION) ;
    CloseHandle(hComm);
  }
  if (inbloques < inVeces)
  {
    SendMessage (hWndDlg, WM_COMMAND, IDC_BUTTON4,0L);
  }
  inVeces++;
}
```

stdafx.cpp

```
// stdafx.cpp : source file that includes just the standard includes
// GeneradorTray1p0.pch will be the pre-compiled header
// stdafx.obj will contain the pre-compiled type information
#include "stdafx.h"
// TODO: reference any additional headers you need in STDAFX.H
// and not in this file
```

Archivos de encabezado

GeneradorTray1p0.h

```
#pragma once
#include "resource.h"
```

Resource.h

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by GeneradorTray1p0.rc
//
#define IDC_MYICON                      2
#define IDD_GENERADORTRAY1P0_DIALOG     102
#define IDS_APP_TITLE                   103
#define IDD_ABOUTBOX                    103
#define IDM_ABOUT                       104
#define IDM_EXIT                        105
#define IDI_GENERADORTRAY1P0            107
#define IDI_SMALL                       108
#define IDC_GENERADORTRAY1P0            109
#define IDR_MAINFRAME                   128
#define ID_EJEX                         129
#define ID_EJEY                         130
#define ID_EJEZ                         131
#define ID_MOVER_HTA                    132
#define ID_COORD_INICIALES              133
#define ID_CAPTURAR_TRAY                134
#define ID_LINEA                        135
#define ID_ARCO_POS                     136
#define ID_ARCO_NEG                     137
#define ID_TRABAJO                      138
#define ID_POSICION                     139
#define ID_CARGAR                       140
#define ID_P_EMERG                      141
#define IDD_DIALOG1                     142
#define IDD_DIALOG2                     143
#define IDD_DIALOG3                     144
#define IDD_DIALOG4                     145
#define IDD_DIALOG5                     146
#define IDD_DIALOG6                     147
#define IDC_EDIT1                       1000
#define IDC_EDIT2                       1001
#define IDC_EDIT3                       1002
#define IDC_EDIT4                       1003
#define IDC_EDIT5                       1004
#define IDC_EDIT6                       1005
#define IDC_BUTTON1                     1006
#define IDC_LIST1                       1007
#define IDC_BUTTON3                     1009
```

```
#define IDC_BUTTON2                     1010
#define IDC_EDIT7                       1011
#define IDC_BUTTON4                     1011
#define IDC_EDIT8                       1012
#define IDC_EDIT9                       1013
#define IDC_STATIC1                     1014
#define IDC_CHECK1                      1017
#define ID_3DCONNEXION                  32771
#define ID_Menu                         32772
#define ID_ARCHIVO_GUARDAR              32773
#define ID_ARCHIVO_GUARDARCOMO          32774
#define ID_CONFIGURACI32775             32775
#define ID_CONFIGURACI32776             32776
#define ID_CONFIGURACI32777             32777
#define ID_CONFIGURACI32778             32778
#define ID_ACCELERATOR32779             32779
#define ID_ACCELERATOR32783             32783
#define ID_DISPOSITIVO_RS232            32785
#define ID_CONFIGURACION_DNC            32786
#define ID_UTILS_CIRCUNFERENCIA         32787
#define ID_UTILS_CIRCYLINEA             32788
#define ID_COORD                        32789
#define ID_CONFIGURACI32790             32790
#define ID_CONFIGURACION_SISTEMA        32791
#define IDC_STATIC                      -1
// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NO_MFC                     1
#define _APS_NEXT_RESOURCE_VALUE        149
#define _APS_NEXT_COMMAND_VALUE         32792
#define _APS_NEXT_CONTROL_VALUE         1021
#define _APS_NEXT_SYMED_VALUE           110
#endif
#endif
```

Stdafx.h

```
// stdafx.h : include file for standard system include files,
// or project specific include files that are used frequently, but
// are changed infrequently
//
#pragma once
#include "targetver.h"
#define WIN32_LEAN_AND_MEAN             // Exclude rarely-used stuff from Windows headers
// Windows Header Files:
#include <windows.h>
// C RunTime Header Files
#include <stdlib.h>
#include <malloc.h>
#include <memory.h>
#include <tchar.h>
#include <stdio.h>
#include <math.h>
#include <float.h>
#include <windowsx.h>
#include <string.h>
#include <winuser.h>
#include <winbase.h>
// TODO: reference additional headers your program requires here
#include "spwmacro.h"  /* Common macros used by SpaceWare functions. */
#include "si.h"        /* Required for any SpaceWare support within an app.*/
#include "siapp.h"     /* Required for siapp.lib symbols */
```

Targetver.h

```
#pragma once
#ifndef WINVER                          // Specifies that the minimum required platform is Windows Vista.
#define WINVER 0x0600           // Change this to the appropriate value to target other versions of Windows.
#endif
#ifndef _WIN32_WINNT            // Specifies that the minimum required platform is Windows Vista.
#define _WIN32_WINNT 0x0600     // Change this to the appropriate value to target other versions of Windows.
#endif
#ifndef _WIN32_WINDOWS          // Specifies that the minimum required platform is Windows 98.
#define _WIN32_WINDOWS 0x0410 // Change this to the appropriate value to target Windows Me or later.
#endif
#ifndef _WIN32_IE                       // Specifies that the minimum required platform is Internet Explorer
7.0.
#define _WIN32_IE 0x0700        // Change this to the appropriate value to target other versions of IE.
#endif
```

Archivos de recursos

GeneradorTray1p0.rc

```
// Microsoft Visual C++ generated resource script.
//
#include "resource.h"

#define APSTUDIO_READONLY_SYMBOLS
//////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#ifndef APSTUDIO_INVOKED
#include "targetver.h"
#endif
#define APSTUDIO_HIDDEN_SYMBOLS
#include "windows.h"
#undef APSTUDIO_HIDDEN_SYMBOLS

//////////////////////////////////////////////////////////////////////////////
#undef APSTUDIO_READONLY_SYMBOLS

//////////////////////////////////////////////////////////////////////////////
// English (U.S.) resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#pragma code_page(1252)
#endif //_WIN32

//////////////////////////////////////////////////////////////////////////////
//
// Icon
//

// Icon with lowest ID value placed first to ensure application icon
// remains consistent on all systems.
IDI_GENERADORTRAY1P0     ICON                    "GeneradorTray1p0.ico"
IDI_SMALL                ICON                    "small.ico"

//////////////////////////////////////////////////////////////////////////////
//
// Menu
//

IDC_GENERADORTRAY1P0 MENU
BEGIN
    POPUP "&Archivo"
    BEGIN
        MENUITEM "A&brir",                      ID_Menu
        MENUITEM "&Guardar",                    ID_ARCHIVO_GUARDAR
        MENUITEM "Guardar &Como",               ID_ARCHIVO_GUARDARCOMO
        MENUITEM "&Salir",                      IDM_EXIT
    END
    POPUP "&Dispositivo"
    BEGIN
        MENUITEM "&RS 232",                     ID_DISPOSITIVO_RS232
    END
    POPUP "Con&figuración"
    BEGIN
        MENUITEM "&Mouse",                      ID_CONFIGURACI32775
        MENUITEM "&RS-232",                     ID_CONFIGURACI32776
        MENUITEM "&DNC",                        ID_CONFIGURACION_DNC
        MENUITEM "Códigos &G",                  ID_CONFIGURACI32778
        MENUITEM SEPARATOR
        MENUITEM "Sistema",                     ID_CONFIGURACION_SISTEMA
    END
    POPUP "A&yuda"
    BEGIN
        MENUITEM "Acerca de...",                IDM_ABOUT
    END
    POPUP "Utils"
    BEGIN
        MENUITEM "Circunferencia",              ID_UTILS_CIRCUNFERENCIA
        MENUITEM "Circ y Línea",                ID_UTILS_CIRCYLINEA
    END
    MENUITEM "Coord. Maq.",                 ID_COORD
END


//////////////////////////////////////////////////////////////////////////////
//
// Accelerator
//

IDC_GENERADORTRAY1P0 ACCELERATORS
BEGIN
    "/",            IDM_ABOUT,              ASCII,  ALT, NOINVERT
    "?",            IDM_ABOUT,              ASCII,  NOINVERT
```

```
END


/////////////////////////////////////////////////////////////////////////////
//
// Dialog
//

IDD_ABOUTBOX DIALOGEX 0, 0, 170, 62
STYLE DS_SETFONT | DS_MODALFRAME | DS_FIXEDSYS | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "About GeneradorTray1p0"
FONT 8, "MS Shell Dlg", 0, 0, 0x1
BEGIN
    ICON            128,IDC_STATIC,14,14,21,20
    LTEXT           "GeneradorTray1p0, Version 1.0",IDC_STATIC,42,14,114,8,SS_NOPREFIX
    LTEXT           "Copyright (C) 2010",IDC_STATIC,42,26,114,8
    DEFPUSHBUTTON   "OK",IDOK,113,41,50,14,WS_GROUP
END


/////////////////////////////////////////////////////////////////////////////
//
// DESIGNINFO
//

#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO
BEGIN
    IDD_ABOUTBOX, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 163
        TOPMARGIN, 7
        BOTTOMMARGIN, 55
    END
END
#endif    // APSTUDIO_INVOKED


#ifdef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
//
// TEXTINCLUDE
//

1 TEXTINCLUDE
BEGIN
    "resource.h\0"
END

2 TEXTINCLUDE
BEGIN
    "#ifndef APSTUDIO_INVOKED\r\n"
    "#include ""targetver.h""\r\n"
    "#endif\r\n"
    "#define APSTUDIO_HIDDEN_SYMBOLS\r\n"
    "#include ""windows.h""\r\n"
    "#undef APSTUDIO_HIDDEN_SYMBOLS\r\n"
    "\0"
END

3 TEXTINCLUDE
BEGIN
    "\r\n"
    "\0"
END

#endif    // APSTUDIO_INVOKED


/////////////////////////////////////////////////////////////////////////////
//
// String Table
//

STRINGTABLE
BEGIN
    IDS_APP_TITLE           "GeneradorTray1p0"
    IDC_GENERADORTRAY1P0    "GENERADORTRAY1P0"
END

#endif    // English (U.S.) resources
/////////////////////////////////////////////////////////////////////////////


/////////////////////////////////////////////////////////////////////////////
// Spanish (Mexican) resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ESM)
#ifdef _WIN32
LANGUAGE LANG_SPANISH, SUBLANG_SPANISH_MEXICAN
#pragma code_page(1252)
#endif //_WIN32

/////////////////////////////////////////////////////////////////////////////
```

```
//
// Dialog
//

IDD_DIALOG1 DIALOGEX 0, 0, 215, 125
STYLE DS_SETFONT | DS_MODALFRAME | DS_FIXEDSYS | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Conectar RS 232"
FONT 8, "MS Shell Dlg", 400, 0, 0x1
BEGIN
    DEFPUSHBUTTON   "&Aceptar",IDOK,110,42,50,14
    PUSHBUTTON      "Cancelar",IDCANCEL,109,66,50,14
    GROUPBOX        "",IDC_STATIC,7,7,201,111
    PUSHBUTTON      "Conectar",IDC_BUTTON1,109,18,50,14
    LISTBOX         IDC_LIST1,17,15,48,40,LBS_NOINTEGRALHEIGHT | WS_VSCROLL | WS_TABSTOP
END

IDD_DIALOG2 DIALOGEX 0, 0, 316, 196
STYLE DS_SETFONT | DS_MODALFRAME | DS_FIXEDSYS | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Interfaz DNC"
FONT 8, "MS Shell Dlg", 400, 0, 0x1
BEGIN
    DEFPUSHBUTTON   "Aceptar",IDOK,145,175,50,14
    PUSHBUTTON      "Cancelar",IDCANCEL,217,175,50,14
    GROUPBOX        "Códigos G",IDC_STATIC,7,7,302,182
    EDITTEXT        IDC_EDIT1,14,21,288,99,ES_MULTILINE | ES_AUTOHSCROLL | WS_VSCROLL | WS_HSCROLL
    PUSHBUTTON      "Enviar",IDC_BUTTON2,95,125,50,14
    PUSHBUTTON      "Cancelar Envío",IDC_BUTTON4,163,124,65,14
    EDITTEXT        IDC_EDIT2,24,142,277,30,ES_MULTILINE | ES_AUTOHSCROLL | WS_VSCROLL
END

IDD_DIALOG3 DIALOGEX 0, 0, 316, 183
STYLE DS_SETFONT | DS_MODALFRAME | DS_FIXEDSYS | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Circunferencia"
FONT 8, "MS Shell Dlg", 400, 0, 0x1
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,205,162,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,259,162,50,14
    GROUPBOX        "Static",IDC_STATIC,7,7,302,176
    GROUPBOX        "Static",IDC_STATIC,31,25,120,111
    EDITTEXT        IDC_EDIT1,36,39,40,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_EDIT2,89,36,40,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_EDIT3,39,69,40,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_EDIT4,93,68,40,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_EDIT5,41,97,40,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_EDIT6,93,97,40,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_EDIT7,189,53,79,14,ES_AUTOHSCROLL | WS_DISABLED
END

IDD_DIALOG4 DIALOGEX 0, 0, 316, 183
STYLE DS_SETFONT | DS_MODALFRAME | DS_FIXEDSYS | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Dialog"
FONT 8, "MS Shell Dlg", 400, 0, 0x1
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,205,162,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,259,162,50,14
    GROUPBOX        "Static",IDC_STATIC,20,7,142,169
    EDITTEXT        IDC_EDIT1,45,19,40,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_EDIT2,44,37,40,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_EDIT3,43,57,40,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_EDIT4,42,77,40,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_EDIT5,39,102,40,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_EDIT6,39,120,40,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_EDIT7,39,145,40,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_EDIT8,186,25,112,14,ES_AUTOHSCROLL | WS_DISABLED
    EDITTEXT        IDC_EDIT9,187,63,111,14,ES_AUTOHSCROLL | WS_DISABLED
END

IDD_DIALOG5 DIALOGEX 0, 0, 199, 167
STYLE DS_SETFONT | DS_MODALFRAME | DS_FIXEDSYS | DS_CENTER | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Coordenadas"
FONT 8, "MS Shell Dlg", 400, 0, 0x1
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,23,146,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,86,146,50,14
    GROUPBOX        "",IDC_STATIC1,7,7,185,153
    GROUPBOX        "",IDC_STATIC,25,14,153,88
    LTEXT           "X:",IDC_STATIC,59,26,8,8
    LTEXT           "Y:",IDC_STATIC,61,54,8,8
    LTEXT           "Z:",IDC_STATIC,59,78,8,8
    EDITTEXT        IDC_EDIT1,77,23,83,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_EDIT2,77,51,83,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_EDIT3,77,78,83,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_EDIT4,27,107,149,32,ES_MULTILINE | ES_AUTOHSCROLL | WS_DISABLED | WS_VSCROLL
END

IDD_DIALOG6 DIALOGEX 0, 0, 316, 183
STYLE DS_SETFONT | DS_MODALFRAME | DS_FIXEDSYS | DS_CENTER | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Configuración general"
FONT 8, "MS Shell Dlg", 400, 0, 0x1
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,205,162,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,259,162,50,14
    GROUPBOX        "",IDC_STATIC,13,7,302,169
    GROUPBOX        "Velocidad de avance en repeticion",IDC_STATIC,164,22,117,39
    EDITTEXT        IDC_EDIT1,168,34,68,17,ES_AUTOHSCROLL
```

```
    LTEXT           "mm / min",IDC_STATIC,248,38,29,8
    GROUPBOX        "Velocidad de avance en línea",IDC_STATIC,28,21,117,39
    LTEXT           "mm / min",IDC_STATIC,112,38,29,8
    EDITTEXT        IDC_EDIT2,35,35,68,17,ES_AUTOHSCROLL
END


/////////////////////////////////////////////////////////////////////////////
//
// DESIGNINFO
//

#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO
BEGIN
    IDD_DIALOG1, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 208
        TOPMARGIN, 7
        BOTTOMMARGIN, 118
    END

    IDD_DIALOG2, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 309
        TOPMARGIN, 7
        BOTTOMMARGIN, 189
    END

    IDD_DIALOG3, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 309
        TOPMARGIN, 7
        BOTTOMMARGIN, 176
    END

    IDD_DIALOG4, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 309
        TOPMARGIN, 7
        BOTTOMMARGIN, 176
    END

    IDD_DIALOG5, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 192
        TOPMARGIN, 7
        BOTTOMMARGIN, 160
    END

    IDD_DIALOG6, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 309
        TOPMARGIN, 7
        BOTTOMMARGIN, 176
    END
END
#endif    // APSTUDIO_INVOKED

#endif    // Spanish (Mexican) resources
/////////////////////////////////////////////////////////////////////////////


#ifndef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 3 resource.
//


/////////////////////////////////////////////////////////////////////////////
#endif    // not APSTUDIO_INVOKED
```

REFERENCIAS

- Angeles Jorge, ***Fundamentals of Robotic Mechanical Systems,*** Third edition, Springer, 2007.

- De Oteyza de Oteyza Elena, Lam Osnaya Emma, Hernández Garciadiego Carlos, Carrillo Hoyo Ángel Manuel, Ramírez Flores Arturo, ***Geometría Analítica y trigonometría,*** Primera edición, Pearson Educación, 2001.

- Bonk Markus, ***3DxInput API,*** 3Dconnexion public.

- ***Dyna M4 Machine Control & Programming Manual,*** version 2.2, Dyna Mechtronics Inc. 2170 Martin Avenue Santa Clara, Ca. 95050 USA.

- F. Krar Steve, F. Check A. ***Tecnología de las Máquinas-Herramienta,*** Quinta edición, Alfaomega, Segunda Reimpresión, México, Abril 2005.

- Howard Anton. ***Introducción al Álgebra Lineal,*** Tercera edición, Limusa Wiley, 2005.
- http://galaxi0.wordpress.com/el-puerto-serial.

- http://www2.queretaro.gob.mx/sedesu/desemp/dime.html.

- http://www2.queretaro.gob.mx/sedesu/proyest/p_interno/2009/CAP08.pdf.

- Khadijeh Daeinabi, Mohammad Teshnehlab. ***Seam Tracking of Intelligent Arc Welding Robot,*** Proceedings of the 6th WSEAS Int. Conf. on Systems Theory & Scientific Computation, Elounda, Greece, August 21-23, 2006 (pp161-166).

- Marcelo H. Ang Jr., Wei Lin y Ser-Yong Lim., ***A Walk-Through Programmer Robot for Welding in Shipyards***, Industrial robot, Vol. 26, No. 5, 1999, pp. 377-388.

- Palacios Enrique, Remiro F., López L. J. ***Microcontrolador PIC16F84 Desarrollo de Proyectos***, Primera Edición, Alfaomega Grupo Editor, 2004.

- Pozo Coronado Salvador, ***Win API con Clase Aplicaciones con API 32,*** Con Clase: http:\winapi.conclase.net, marzo 2007.

- Smid P. ***CNC Programming Handbook***, Second Edition, Industrial Press Inc. 2003.
- Stanley I. Grossman. ***Álgebra Lineal,*** Quinta edición, Mc Graw Hill.

- Takarics B., Szemes P. T. ***Superflexible Welding Robot Based on the Intelligent Space Concept***, 7th International Symposium of Hungarian Researchers on Computational Intelligence, 2006.

Universidad Autónoma de Querétaro
Facultad de Ingeniería
Dirección

ACUERDO 481/11

C. U. 12 de septiembre de 2011

C. JORGE MÉNDEZ CRUZ
Pasante de Ingeniería Electromecánica
Presente.

Con relación a su oficio enviado al H. Consejo Académico de la Facultad en el que solicita titularse bajo la opción de tesis individual, me permito informarle que en la sesión ordinaria del 12 de septiembre del año en curso, este cuerpo colegiado acordó aceptar la opción de titulación por lo que deberá trabajar en el tema **"GENERADOR MANUAL DE TRAYECTORIAS 3D PARA POSICIONAMIENTO DE ROBOT CARTESIANO"**, bajo la dirección del M. en C. Manuel García Quijada.

El Contenido aceptado por el H. Consejo Académico es el siguiente:

Capítulo I Introducción

1.1 Estado del conocimiento
1.2 Objetivos
1.3 Justificación e hipótesis
1.4 Planteamiento general

Capítulo II Revisión de Literatura

2.1 Generalidades
2.2 Transformaciones lineales
2.3 Otras herramientas matemáticas
2.4 CNC´s
2.5 Protocolos de comunicación serial
2.6 Lenguajes de programación

Capítulo III Desarrollo del software

3.1 Definición de módulos
3.2 Desarrollo del módulo *Mouse 3D*-PC
3.3 Desarrollo del módulo PC-CNC
3.3 Integración y caracterización

Centro Universitario
Cerro de las Campanas
Santiago de Querétaro
Qro., México C.P. 76010
Tel. 01 (442) 192 12 00
Ext. 6016
Fax 6006

Universidad Autónoma de Querétaro
Facultad de Ingeniería
Dirección

Capítulo IV Pruebas y Análisis de Resultados

4.1 Análisis comparativo y discusión
4.2 Conclusiones y recomendaciones

Anexo A

Referencias

También hago de su conocimiento las disposiciones de nuestra Facultad, en el sentido que antes del Examen profesional deberá cumplir con los requisitos de nuestra legislación y deberá imprimir el presente oficio en todos los ejemplares de su tesis.

Atentamente
"EL INGENIO PARA CREAR NO PARA DESTRUIR"

DR. GILBERTO HERRERA RUIZ
Director
c.c.p. Archivo

*GHR/DHM.

Centro Universitario
Cerro de las Campanas
Santiago de Querétaro
Qro., México C.P. 76010
Tel. 01 (442) 192 12 00
Ext. 6016
Fax 6006