Universidad Autónoma de Querétaro

Facultad de Ingeniería.

TESIS

Software para controlador de movimiento basado en FPGA.

Que como parte de los requisitos para obtener el grado de

Maestro en Ciencias con Línea Terminal en Instrumentación y Control Automático.

Presenta:

Javier Ugalde Estrella.

Dirigido por:

Dr. Roque Alfredo Osornio Ríos.

Dr. Luis Morales Velázquez.

Centro Universitario

San Juan del Río Qro.

Septiembre del 2011

México.

Universidad Autónoma de Querétaro

Facultad de Ingeniería

Maestría en Instrumentación y Control Automático

SOFTWARE PARA CONTROLADOR DE MOVIMIENTO BASADO EN FPGA

TESIS

Que como parte de los requisitos para obtener el grado de

Maestro en Ciencias en Instrumentación y Control Automático

Presenta:
Ing. Javier Ugalde Estrella

Dirigido por:
Dr. Roque Alfredo Osornio Ríos

SINODALES

Dr. Roque Alfredo Osornio Ríos Presidente	Firma
Dr. Luis Morales Velázquez Secretario	Firma
Dr. J. Jesús de Santiago Pérez Vocal	Firma
Dr. Miguel Trejo Hernández Suplente	Firma
M.I. Hayde Peregrina Barreto Suplente	Firma

Dr. Gilberto Herrera Ruíz
Director de la Facultad

Dr. Luis Gerardo Hernández Sandoval
Director de Investigación y Posgrado

Centro Universitario
Querétaro, Qro.
Agosto del 2011
México

RESUMEN

La utilización de las máquinas de control numérico computarizado (*Computer Numeric Control*, CNC) en la industria es de gran importancia, su aplicación abarca casi todos los procesos de manufactura, siendo necesario desarrollar investigación relacionada con el control de dichas máquinas. Este trabajo es un seguimiento de los desarrollos realizados en la Universidad Autónoma de Querétaro (UAQ) para los sistemas de control de movimiento, el principal trabajo en el cual se encuentra basada esta investigación es en el desarrollado por Morales (2010). En este trabajo se describe la metodología para el desarrollo de un software, bajo el sistema operativo Linux, para el sistema de control de movimientos MCUAQ4X desarrollado en la UAQ implementado en un arreglo de compuertas programables en campo (*Field Programmable Gate Array*, FPGA) y un sistema de control comercial (DMC 18x2 de Galil) aplicados a maquinaria CNC. Se presentan los procesos utilizados para la lectura e interpretación de códigos DMC y códigos G, estos últimos provenientes del código NC (*Numerical Control,* Control Numérico). Para el controlador MCUAQ4X se generan una serie de curvas NURBS las cuales son enviadas para su ejecución y utilizadas para la simulación en el software, implementándolo en torno y fresa mientras que el controlador DMC 18x2 únicamente fue implementado en un torno. El software desarrollado ha sido probado mediante la implementación de programas NC y DMC, presentando de esta manera un análisis en el seguimiento de las trayectorias y la medición de la rugosidad en las piezas maquinadas.

(**Palabras clave:** CNC, Código DMC, Código G, FPGA, Linux, NURBS, Software)

SUMMARY

The use of CNC (Computer Numerical Control) in the industry is of great importance, the application includes almost all manufacturing processes, making it necessary to carry out research related to the control of these machines. This work is monitoring the developments made in the Autonomous University of Querétaro (UAQ) for motion control systems, the main work in which this research is based is developed by Morales (2010). This paper describes the methodology for developing software under the Linux operating system for the motion control system developed at the UAQ MCUAQ4X implemented in FPGA (Field Programmable Gate Array) and commercial control system (DMC 18x2 of Galil) applied to CNC machinery. It presents the processes used to read and interpret DMC codes and G codes, latter code from the NC (Numerical Control). For the controller MCUAQ4X generated a series of NURBS curves are then sent for execution and used for simulation software. The developed software has been tested through the implementation of NC programs and DMC, thus presenting an analysis of the trajectories tracking and measuring the roughness of machined parts.

(**Keywords:** CNC, DMC Codes, FPGA, G Codes, Linux, NURBS, Software)

DEDICATORIAS

A mis padres y a mi amada novia Blanca Nalleli con todo mi amor y cariño, de quienes siempre recibí un apoyo incondicional y no existen palabras para describir el inmenso amor que les tengo.

AGRADECIMIENTOS

Principalmente a Dios quien nunca me ha desamparado a lo largo de mi vida. A mi familia por todo su apoyo y confianza a lo largo de toda mi carrera que con este trabajo culmina una etapa muy importante. A mi asesor Dr. Roque Alfredo Osornio Ríos por todos sus consejos y apoyo a lo largo de este trabajo. A mi coasesor Dr. Luis Morales Velázquez por su invaluable guía en este trabajo, de quien he aprendido muchas cosas que me han hecho crecer como persona e investigador. Al consejo Nacional de Ciencia y Tecnología por otorgarme una beca para realizar los estudios de posgrado en una de las instituciones de más alto nivel académico del país como lo es la Universidad Autónoma de Querétaro. A todos mis compañeros y amigos de quienes puedo presumir su amistad, los cuales he conocido a lo largo de mi vida y que a pesar de la distancia se que están allí para apoyarme y darme palabras de aliento como yo a ellos.

ÍNDICE

ÍNDICE DE FIGURAS

Figura Página

ÍNDICE DE TABLAS

ÍNDICE DE LISTADOS

Capítulo I.

INTRODUCCIÓN

La necesidad por mejorar la eficiencia y la precisión en un sistema de control numérico ha sido la mejor razón para el desarrollo de nuevas investigaciones y desarrollos tecnológicos tanto para el ámbito industrial como académico, logrando con esto, un gran avance en cuanto a la generación de nuevos métodos.

El proceso de manufactura de maquinados en Querétaro se encuentra relacionado principalmente con las PYMES en un 75%, con esto existe una importante necesidad tecnológica para actualizar maquinaria en este ramo. Creando así *software* de bajo costo y redituable aunado con una automatización bien definida.

Un software se refiere al equipamiento lógico o soporte lógico de una computadora digital, y comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de tareas específicas; en contraposición a los componentes físicos del sistema, llamados hardware. Tales componentes lógicos incluyen, entre muchos otros, aplicaciones informáticas como procesador de textos, que permite al usuario realizar todas las tareas concernientes a edición de textos; software de sistema, tal como un sistema operativo, que, básicamente, permite al resto de los programas funcionar adecuadamente, facilitando la interacción con los componentes físicos y el resto de las aplicaciones, también provee una interfaz para el usuario.

Por otra parte, el control numérico por computadora es un sistema que permite controlar en todo momento la posición de un elemento físico, normalmente una herramienta que está montada en una máquina. Esto quiere decir que mediante un software y un conjunto de órdenes, se controlan las coordenadas de posición de un punto (la herramienta) respecto a un origen.

El CNC controla todos los movimientos de la herramienta cuanto ésta se encuentra fabricando alguna pieza, donde controla la posición, velocidad, aceleración y desaceleración, lo cual permite obtener piezas con determinadas medidas y la creación de programas que permiten repetir con gran precisión piezas idénticas, también se utiliza para verificar las medidas de algo que ha sido que fabricado.

El software es una parte muy importante en el proceso de manufactura ya que este es el intérprete entre el usuario y el controlador, pues interpreta la información que el usuario escribe en él y le envía al controlador la información de tal manera que éste la entienda para que el controlador a su vez le indique a la máquina-herramienta lo que debe realizar.

1.1 Antecedentes

La utilización de las máquinas CNC en la industria es de gran importancia, su aplicación abarca casi todos los procesos de manufactura, siendo necesario desarrollar investigación relacionada con el control y automatización de dichas máquinas. A continuación se hace mención de algunos de los trabajos que han hecho estudio o actualmente estudian temas de distinta índole en el ámbito ingenieril, pero que a pesar de eso todos en conjunto tienen como mismo fin mejorar la calidad de producción en la rama industrial.

Osornio (2004), desarrolló una tarjeta de control de movimiento para tres ejes mediante el uso de lenguaje VHDL en un FPGA, en la tarjeta controladora están incluidos los módulos del PID (Proporcional-Integral-Derivativo), módulo de compensación de inercia y un módulo de control no lineal proporcionado por un generador de trayectoria. Alaniz (2005), desarrolló un sistema de control CNC para corte en el cual agrega módulos de monitoreo y optimización de la dinámica ajustando la velocidad de corte adecuada para cada proceso. Hernández (2006) presentó el diseño e implementación en hardware basado en FPGA de un controlador de movimiento PID para aplicación en servomotores. Colin (2006), realizó la descripción bajo VHDL para un controlador PID para aplicación en sistemas de control de posicionamiento con servomotores. En base al control también se han desarrollado trabajos internacionalmente como el de Osornio *et al.* (2007), quienes realizan el diseño de un sistema de control basado en FPGA para máquinas herramientas CNC de alta velocidad. Además, se establecen las bases para el control de la dinámica al proponer un método de parametrización de polinomios.

En lo que respecta a dinámica de movimiento de acuerdo con la literatura, la dinámica es un aspecto muy importante a considerar en los procesos de control relacionados con máquinas-herramienta CNC y Robots (Jönsson, 2005). Se han desarrollado trabajos como el de Erkorkmaz y Wong (2007), donde requirieron del

monitoreo de velocidad y posición para ejecutar su técnica de identificación basada en la introducción de códigos G de prueba; el objetivo de dicha técnica es establecer algunos parámetros de velocidad, aceleración y desaceleración máxima para el proceso de maquinado sin desconectar el lazo de control o interpolador. Otro trabajo reciente que muestra la importancia de medir estos parámetros en este caso el *jerk* es el desarrollado por Lin et al. (2007) en el que destacan la necesidad de una apropiada planeación de trayectorias en los sistemas CNC por lo que realiza la implementación de funciones interpoladoras NURBS con el objetivo de generar dinámicas de movimiento suaves mejorando los resultados de las interpolaciones lineales y circulares que son utilizadas comúnmente en los sistemas CAM. Su interpolador se basa en la creación de tres módulos: geométrico, dinámico y limitador de *jerk*. Este último módulo permite planear y aproximar los perfiles del *feedrate* de la curva de acuerdo a los segmentos de longitud y el límite de *jerk* dado. Sin embargo, no existe una medición o reconstrucción física de los parámetros de aceleración y *jerk* solo aproximaciones.

También se han realizado investigaciones para el análisis y monitoreo del desgaste de la herramienta de un CNC, Romero (2004) diseñó un sistema para el monitoreo y detección en línea de ruptura de herramientas en sistemas de manufactura CNC, la metodología fue basada en DWT (*Discrete Wavelet Transform*, Transformada Discreta de Wavelet) y la implementación fue en hardware mediante un FPGA. García (2006) presentó un sistema de monitoreo en un torno CNC para la detección de ruptura de herramientas basado en FFT (*Fast Fourier Transform*, Transformada Rápida de Fourier). Panahi *et al*. (2006) reportaron un método para la generación en línea de perfiles con dinámica controlada en velocidad y aceleración para sistemas de control. Trejo (2006) desarrolló un módulo basado en software para realizar procesos de monitoreo y maquinado en un proceso de torneado, estableciendo las bases para el monitoreo de señales (como la corriente) para poder analizar el desempeño de la máquinas CNC. Osornio *et al.* (2009) presentaron la implementación bajo FPGA de perfiles polinomiales de movimiento en el que se reportó control en los picos de *jerk* durante el proceso de maquinado.

En lo que respecta al desarrollo científico, diferentes técnicas de generación de trayectorias han sido propuestas enfocadas a mejorar la calidad de los acabados y reducción del tiempo de maquinado basados mediante curvas paramétricas como curvas pitagóricas,

de Bézier, B-spline y NURBS. Erkorkmaz y Altintas (2001) presentaron un algoritmo para la generación de trayectorias basado en funciones spline de quinto grado el cual produce curvas continuas para la aplicación en maquinaria de fresado. Tsai *et al.* (2003) propusieron una metodología basada en software para la interpolación de NURBS y su aplicación a maquinado CNC. Muller *et al.* (2004) desarrollaron una técnica para generar trayectorias basadas en interpolación spline cuya metodología incluye también un módulo de interpolación con curvas pitagóricas. Relacionado con la generación de trayectorias se encuentra el diseño de perfiles de movimiento que rigen la dinámica de movimiento de la máquina herramienta que se relaciona directamente con los niveles de vibración. Macfarlane y Croft (2003) desarrollaron un método para la obtención de trayectorias suaves y acotadas en *jerk* donde destacan que el control de la dinámica es importante en aplicaciones a maquinaria CNC y robots industriales debido a que se relaciona con un buen seguimiento de la trayectoria, niveles de vibración y en algunos casos un mal diseño de perfiles de movimiento puede llegar a saturar los servoamplificadores. Park *et al.* (2005) presentaron una metodología para interpolación de NURBS en el cual el error entre la curva ideal y la interpolada es acotado. De Santiago *et al*. (2008) propusieron un método para obtener los parámetros dinámicos del eje de una máquina-herramienta (velocidad, aceleración y *jerk*) de la posición dada por un encoder incremental óptico, realizando el cálculo de la primera, segunda y tercera derivada de la señal de posición discreta dada por el encoder utilizando diferencias finitas, además redujo el ruido producido por medio de la transformada wavelet discreta (DWT).

Por otro lado también se han realizado trabajos enfocados en el software como lo hizo Femat (2004) quien desarrolló un software que sirve de interfaz en un sistema de control CNC bajo tarjetas de posicionamiento comerciales. Mejía (2008) quien describe el diseño e implementación en software de un CNC aplicado a torno para la detección en línea de variaciones en las condiciones de corte, desarrolló la interfaz de comunicación entre el operario, la tarjeta de control de movimientos Galil y la tarjeta de adquisición de datos durante el corte, aunado con los estándares generales de códigos G y M (código NC). La arquitectura del software permite el manejo y la interacción con diversos módulos en desarrollos e investigaciones dentro del sistema siguiendo una interfaz bien definida. El software desarrollado fue probado mediante la implementación de programas CNC y su

correspondiente maquinado. Ugalde (2009) desarrolló una unidad para generación de trayectorias polinomiales a partir de código G aplicados a sistemas de posición FPGA, donde utilizó programación en C en un entorno de Linux para desarrollar un intérprete de códigos G mediante el cual se obtienen una serie de puntos a los cuales se les aplica el método de mínimos cuadrados para generar un conjunto de polinomios que posteriormente son enviados al sistema de posición FPGA.

Otro punto que también es importante es la interfaz de comunicación, por lo que se han realizado trabajos relacionados como Morales (2007) quien desarrolló un sistema con unidad USB de control de posición y generación de perfiles para un intercambiador automático de herramientas; el sistema se compone de tres bloques: la interfaz USB, el regulador PID y el generador de perfiles y su integración se realizó mediante un FPGA empleando lenguaje VHDL con una estructura de arquitectura abierta. Posteriormente Morales (2010) presentó una plataforma de arquitectura abierta basada en unidades multi-agente de hardware-software, mediante el desarrollo de un nuevo sistema de control distribuido multi-agente (*Multi-Agent Distributed CONtroller*, MADCON) para satisfacer los requerimientos de reconfigurabilidad para la siguiente generación de máquinas inteligentes.

1.2 Descripción del problema

Como se muestra en los antecedentes existe un gran número de investigaciones enfocadas en mejorar la dinámica en el movimiento de máquinas herramientas. Algunas investigaciones realizan el control del movimiento utilizando sistemas de control comerciales ya que son poco robustos, permiten minimizar tiempos de producción y mejorar parcialmente la calidad en la manufacturación de la pieza, esto representa una reducción en los costos de producción, sin embargo, tienen un alto costo de adquisición sin mencionar que se trata de sistemas cerrados, que en ocasiones requieren de recursos adicionales como apoyo del procesador de su arquitectura fija y cerrada, tal es el caso de PMAC fabricado por Delta Tau (2005) y Galil Control Motion (2010).

Los sistemas cerrados y de aplicación específica no permiten que sus algoritmos de control se adapten fácilmente a otras aplicaciones, o que la estructura del algoritmo sea modificada o se le adicionen elementos para adecuarse a necesidades más especificas de

algún usuario, además de que dichos algoritmos no son transparentes, es decir, son desconocidos para los usuarios finales, caso contrario es el de los sistemas con algoritmos de arquitectura abierta donde los desarrollos han sido en dos campos hardware y software sin embargo no hay trabajos que integren dichos desarrollos.

En los últimos años como se puede observar en los antecedentes, en la Facultad de Ingeniería de la Universidad Autónoma de Querétaro, se ha venido desarrollando tecnología con arquitectura abierta basada en FPGA para mejorar la dinámica que se produce en el movimiento de las máquinas herramientas durante el proceso de maquinado, observando el comportamiento de distintos parámetros como es la posición, velocidad, aceleración y el jaloneo. Esto con el fin de mejorar el proceso de alguna forma, ya sea mejorando el acabado, el tiempo de maquinado ó alargando el tiempo de vida de la herramienta de corte, sin embargo, los trabajos realizados hasta el momento únicamente se han enfocado principalmente en el monitoreo, control y mejoramiento de la dinámica en el movimiento para el sistema de control dejando como segundo término la parte de la comunicación del usuario.

Este trabajo se basa en el desarrollo de un software CNC principalmente en la parte de la interfaces gráficas de usuario (GUI) para el sistema de control de movimiento basado en FPGA desarrollado en la Universidad Autónoma de Querétaro. Uno de los últimos trabajos realizados para este sistema es el de Morales (2010) quien desarrolló una plataforma hardware-software basada en FPGA para aplicaciones industriales donde generó varios módulos de procesamiento digital especializados en procesos de monitoreo y control, en la Figura 1.1 se muestra el diagrama general del sistema de software para un controlador distribuido reconfigurable (*distributed reconfigurable controller*, DRC). La parte del software se encuentra conformado por tres hilos los cuales trabajan simultáneamente el primero se encarga de manejar la GUI la cual tienen como principal función interactuar con el usuario, manejar los objetos gráficos de la interfaz y enviar información, el segundo hilo se encarga de un Kernel en donde se realiza el manejo de la información y del servidor del hardware, por último el tercer hilo se encarga del agente DRC que es una unidad virtual del hardware DRC el cual contiene registros, paquetes y programas que permiten una conexión directa con la unidad DRC en el sistema..

En la industria unas de las normas para códigos NC utilizadas para el proceso de maquinado son el ISO 6983 y FANUC las cuales son seguidas por los controladores comerciales. El controlador comercial Galil DMC-1832 además de manejar el código NC maneja un código llamado DMC el cual es utilizado para capacitar a los usuarios con las herramientas que proporciona éste.

Por otra parte en la mayoría de los trabajos es necesario comparar el funcionamiento de la tecnología desarrollada con un producto comercial y en la Facultad de Ingeniería de la UAQ se compara con la tarjeta de control comercial Galil DMC-1832 de tal manera que también se requiere manejar este controlador.

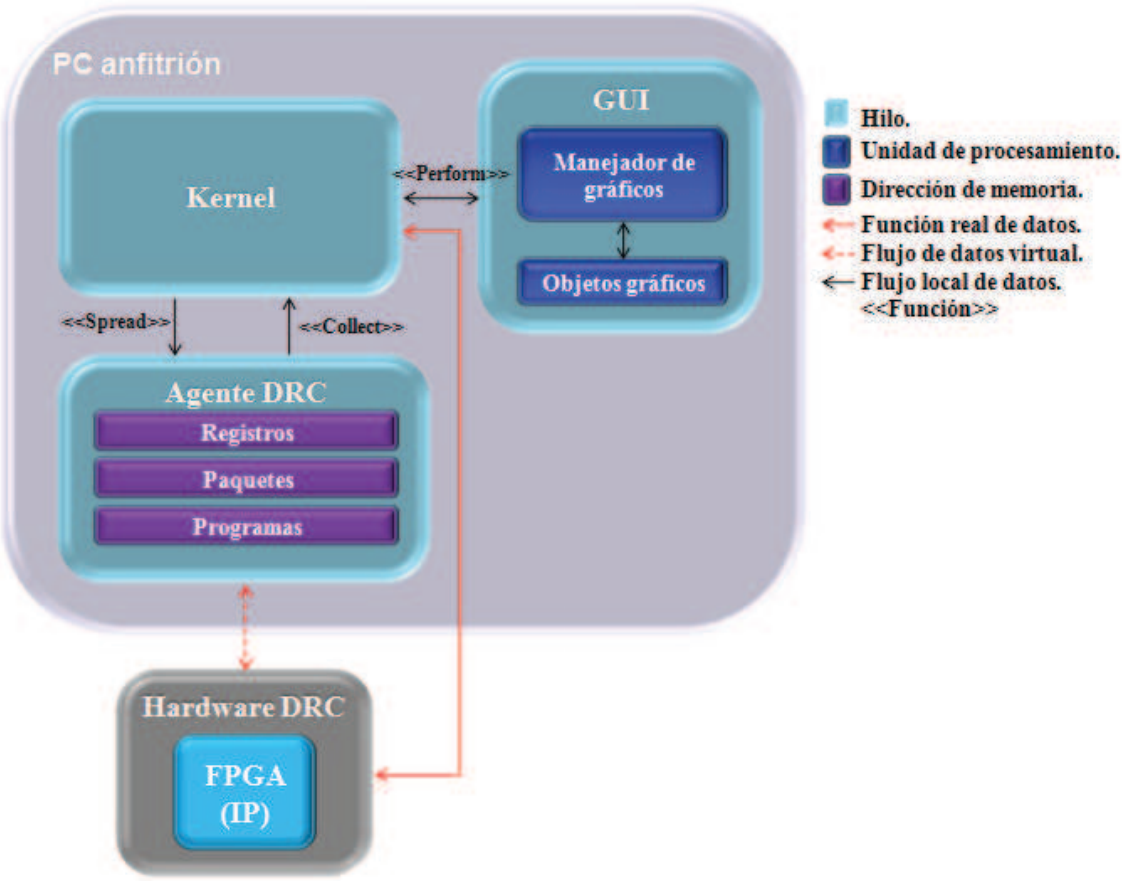

Figura 1.1 Diagrama general del sistema de software para un DRC.

De tal forma que se requiere agregar en la parte del software una serie de módulos que sean capaces de interpretar los comandos esenciales del código DMC y CN, con el fin de realizar la comunicación con el controlador propio y Galil. La interpretación del código

se requiere en la GUI ya que es en esta parte donde se tiene comunicación directamente con el usuario, también es necesario para el usuario un simulador el cual permita visualizar los movimientos que se realizarán y lograr de esta manera un entorno amigable con el usuario.

1.3 Objetivos

El objetivo principal de este trabajo es realizar un software que interprete la información dada por el usuario al controlador, ya sea el controlador basado en FPGA desarrollado en la UAQ o para una tarjeta controladora comercial como lo es la tarjeta de control Galil (DMC-1832), se pretende implementar únicamente algunos comandos de códigos G (ISO 6983) y DMC (18x2).

Para poder alcanzar el objetivo general de este trabajo y a la vez para que sea entendido de una mejor manera es necesario dividirlo en objetivos particulares los cuales son mencionados a continuación:

- Generar los *drivers* necesarios para la detección del controlador basado en FPGA.
- Desarrollar e implementar un compilador de algunos comandos DMC.
- Generar un intérprete de comandos DMC el cual proporcionará los parámetros necesarios a una tarjeta de control Galil.
- Programar un intérprete de comandos DMC el cual proporcionará los parámetros necesarios al controlador basado en FPGA.
- Desarrollar e implementar un compilador de algunos comandos NC.
- Generar de un intérprete de comandos NC el cual proporcionará los parámetros necesarios a una tarjeta de control Galil.
- Programar un intérprete de comandos NC el cual proporcionará los parámetros necesarios al controlador basado en FPGA.
- Diseñar un simulador de programas DMC.
- Diseñar un simulador de programas NC.
- Pruebas de maquinado en un torno con ambos controladores.
- Prueba de seguimiento en una fresadora con el controlador basado en FPGA.

1.4 Justificación

La industria manufacturera aporta, tanto en el estado como en el país, un importante porcentaje al Producto Interno Bruto (PIB), en su mayoría se encuentra conformada por micro, pequeñas y medianas empresas. La importancia de las pequeñas y medianas empresas, PYMES, en la economía y en la generación de empleos es fundamental, por lo que es necesario atender la problemática que enfrentan los empresarios que buscan iniciar un negocio o que requieren fortalecerlo para asegurar su supervivencia. Por lo cual el gobierno tiene como uno de sus objetivos fortalecer la industria para que mejore su calidad y productividad, y propiciar una mayor competitividad que favorezca su incursión en nuevos mercados nacionales e internacionales.

La utilización de las máquinas CNC en la industria es de gran importancia, su aplicación abarca casi todos los procesos de manufactura, siendo necesario desarrollar investigación relacionada con el control de dichas máquinas. En los últimos años han sido distintos los trabajos académicos, aportados por estudiantes de la Universidad Autónoma de Querétaro, relacionados con sistemas de control de movimiento. Estos trabajos utilizan tarjetas de control comerciales como lo es la tarjeta de control Galil o en algunos casos utilizan la tecnología desarrollada en la institución, un controlador basado en FPGA. Sin embargo, en la mayoría de los trabajos elaboran un programa provisional, para comunicarse ya sea con una tarjeta comercial ó con el controlador basado en FPGA, ó en el mejor de los casos, cuando utilizan tarjetas comerciales, implementan programas que fueron desarrollados con anterioridad.

Como se vio en los antecedentes, los trabajos de los estudiantes de la UAQ han generado diferentes tipos de software para controlar un CNC como es el caso de Mejía (2008) y Trejo (2007) pero manejando hardware comercial. También se han realizado trabajos los cuales su objetivo es integrar las tecnologías desarrolladas en la institución como lo hizo Gómez (2007) quien desarrolló e implementó un sistema de control en tiempo real de arquitectura abierta reconfigurable. Los trabajos anteriores se enfocan en el control de tarjetas comerciales, pero no en software que permita el manejo del controlador basado en FPGA con los mismos comandos con los que se maneja un controlador comercial, como lo pretende el software desarrollado en el presente trabajo. En consecuencia, para reducir

costos de desarrollo e investigación, es necesario implementar una plataforma que haga factible la integración de todas las invenciones realizadas de un mismo producto que se pueda usar como una herramienta más tanto para la industria como para la experimentación e investigación.

El presente trabajo desarrolla un software para el controlador de movimiento basado en FPGA mediante el cual podremos realizar tareas de desplazamiento de orden repetitivas o laboriosas, el software puede aplicarse en diferentes campos o sectores, pues su objetivo no es aplicarse únicamente para CNC sino que pueda ser empleado en todos los procesos donde se maneje una tarjeta de control de movimiento comercial para cuatro ejes, lo cual le da mucha versatilidad de uso en diferentes áreas.

Es de esta manera como se justifica el desarrollo del trabajo dentro de la UAQ. En la Facultad de Ingeniería, hace falta un software para el controlador de movimiento basado en FPGA con el cual no únicamente se pueda utilizar este controlador sino que también el controlador comercial Galil; además, se requiere un simulador con el cual se pueda observar el error de seguimiento para cada uno de los controladores, todo esto con la finalidad de poder mejorar la calidad de las tareas que se desempeñan en el área de manufactura de la institución. Este trabajo presenta una opción económica de automatización a la industria del país y región.

1.5 Planteamiento general

Este trabajo consiste en la realización de un software que permita comunicarse tanto con un controlador de movimiento comercial, en este caso se utilizará la tarjeta comercial Galil (DMC-1832), como con el sistema de control de posicionamiento basado en FPGA desarrollado en la UAQ. En la Figura 1.2 se plantea una vista general del proceso y el aporte que se realizará con el desarrollo de este trabajo.

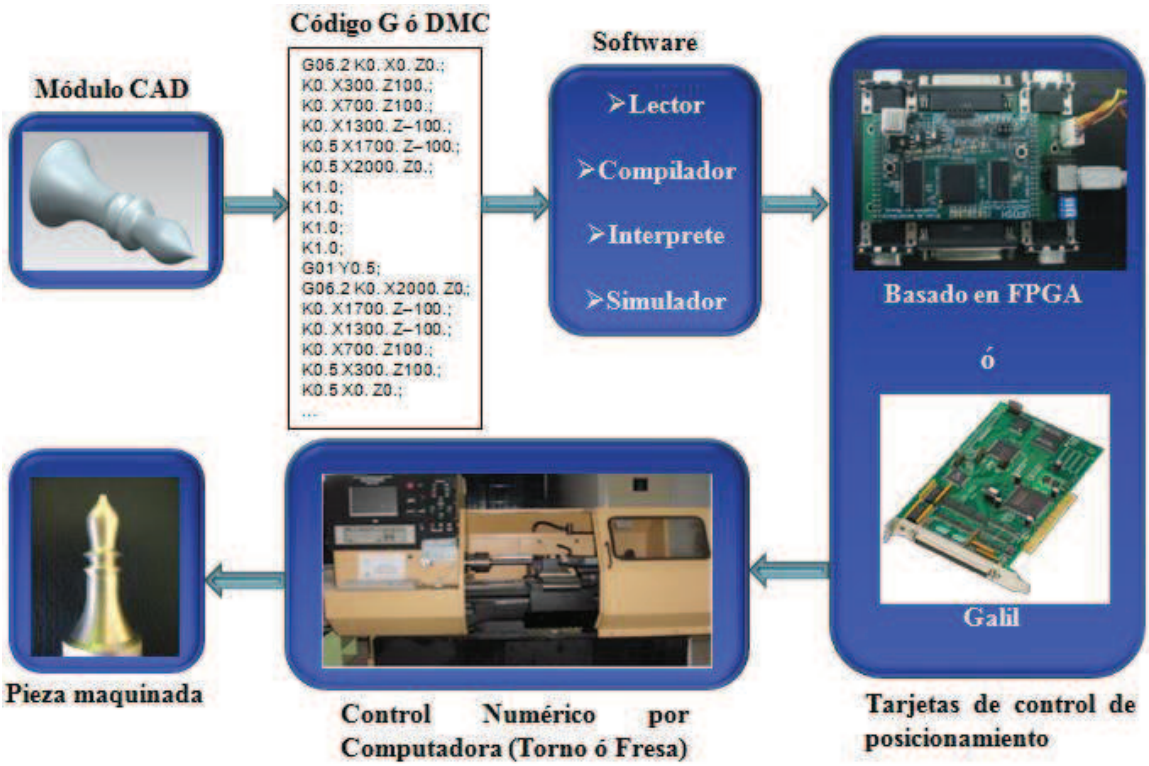

Figura 1.2 Diagrama general del sistema de software para un DRC.

Como se muestra en la Figura 1.2 el software a realizar tendrá la capacidad de compilar, interpretar y simular códigos G y DMC, de los cuales se obtendrán los parámetros para enviar a la tarjeta de control de posicionamiento. La tarjeta de posicionamiento, podrá ser una tarjeta comercial Galil ó una tarjeta de control de posicionamiento basado en FPGA, y permitirá controlar los movimientos que debe realizar el CNC para el maquinado de una pieza.

El software a desarrollar se encuentra dividido en cuatro partes fundamentales, la primera etapa el lector el cual realiza un preproceso al código el cual elimina los comentarios encontrados en el programa dejando únicamente los comandos que se desean para la interpretación. Sin embargo, no es suficiente enviar el código al intérprete con la eliminación de los comentarios, también se requiere una identificación de errores de escritura en los códigos esto se realiza en el compilador. La tercera etapa consiste en un intérprete donde se realizan los cálculos para obtener y mandar los parámetros necesarios al controlador que se encuentre en uso. Por último se encuentra la parte de simulación la cual realiza la gráfica del maquinado en base al código analizado.

Capítulo II.

FUNDAMENTACIÓN TEÓRICA

2.1 Control Numérico Computarizado (CNC)

Las primeras máquinas herramienta, como el torno, surgieron en la antigüedad; se tienen registros de tornos usados por los antiguos egipcios. Las máquinas herramienta como las conocemos hoy en día se desarrollaron en la revolución industrial. En 1775, John Wilkinson inventó en Inglaterra la máquina agujeradora de cañones (torno). En el año 1818 Eli Whitney inventó la máquina fresadora en New Haven Connecticut. Poco después el husillo de la máquina de Whitney se movió de ser horizontal a vertical, dando lugar al estilo Bridgeport de fresadora de rodilla.

En 1952, Mr. John T. Parsons colocó servomotores para controlar el movimiento en el eje *x* y el eje *y*, controlándolos mediante una computadora que leía tarjetas perforadas que le daban la posición a la bancada de la máquina. De esta forma, la máquina era capaz de realizar formas complejas como arcos que eran requeridos en la industria aeronáutica. Así nacieron las máquinas de control numérico (máquinas NC). Hoy en día la maquinaria moderna, tal como fresadoras y tornos, son de control numérico por computadora. Un microprocesador lee el programa en código G que el usuario crea de forma manual o usando un paquete de manufactura asistida por computadora (CAM, Computer Aided Manufacturing) (Worcester, 2000).

Se considera control numérico a todo dispositivo capaz de dirigir posicionamientos de un órgano mecánico móvil, en el que las órdenes relativas a los desplazamientos del móvil son elaboradas en forma totalmente automática a partir de informaciones numéricas definidas, bien manualmente o por medio de un programa. El término "control numérico" se debe a que las órdenes dadas a la máquina son indicadas mediante códigos numéricos (código G y M). El control numérico puede definirse como un funcionamiento de herramientas de la máquina por los medios de instrucciones específicamente codificado al sistema de control de máquina (Smid, 2003).

2.1.1 Torno

Un torno de CNC normalmente es una herramienta de la máquina con dos ejes, el vertical (eje X) y el horizontal (eje Z). El rasgo principal del torno que lo distingue de una fresadora es que la parte está rodando sobre la línea de centro de máquina. Además, la herramienta cortante es normalmente estacionaria, montada en un torreón corredizo. La herramienta cortante sigue el contorno del camino programado de la herramienta (Smid, 2003).

2.1.2 Fresadora

Una fresadora es una máquina herramienta utilizada para realizar mecanizados por arranque de viruta mediante el movimiento de una herramienta rotativa de varios filos de corte denominada fresa. En las fresadoras tradicionales, la pieza se desplaza acercando las zonas a maquinar a la herramienta, permitiendo obtener formas diversas, desde superficies planas a otras más complejas (Smid, 2003).

2.2 Sistemas CAD/CAM

El diseño y la fabricación asistidos por computadora (CAD/CAM) es una disciplina que estudia el uso de sistemas informáticos como herramienta de soporte en todos los procesos involucrados en el diseño y la fabricación de cualquier tipo de producto. Esta disciplina se ha convertido en un requisito indispensable para la industria actual que se enfrenta a la necesidad de mejorar la calidad, disminuir los costos y reducir los tiempos de diseño y producción (De Santiago, 2010).

En el diseño asistido por computadora (CAD, *computer-aided design*) la pieza que se desea maquinar se diseña en la computadora con herramientas de dibujo y modelado sólido. Posteriormente, el sistema CAM (manufactura asistida por computadora, *computer-aided manufacturing*) toma la información del diseño y genera la ruta de corte que tiene que seguir la herramienta para fabricar la pieza deseada; a partir de esta ruta de corte se crea automáticamente el programa de maquinado, el cual puede ser introducido a la máquina mediante un disco o enviado electrónicamente.

2.2.1 CAD

El modelo CAD consiste en el diseño de la pieza a maquinar como una concatenación de elementos geométricos que se puede realizar mediante una amplia gama de software CAD disponible para este propósito. En los sistemas modernos existe una gran variedad de formas para la realización de piezas y moldes que son representados usualmente mediante curvas y superficies paramétricas como curvas de Bézier, B-splines y NURBS (Non-Uniform Rotational B-splines) (De Santiago, 2010).

2.2.2 CAM

El proceso CAM está dedicado a calcular la trayectoria a partir del modelo CAD. En esta etapa, se generan segmentos de curvas que servirán de referencia para el controlador CNC. En general, los algoritmos CAM pueden ser divididos en tres etapas. La primera etapa es generar los puntos de contacto para el corte (puntos CC), los cuales son seleccionados en base a la geometría de la superficie de tallado. La segunda etapa es generar los datos de localización de corte (datos CL), el cual depende de ángulos específicos y las superficies normales a los puntos CC. La última etapa es convertir los datos CL a código de máquina, es decir, genera los comandos de movimiento de la máquina-herramienta; Tales comandos están enfocados primordialmente a describir la ruta de la herramienta, aunque también contiene otras instrucciones necesarias como velocidad de avance, velocidad de husillo, número de herramienta, etc. (De Santiago, 2010).

2.3 Sistemas de Control de Movimiento

El software y hardware de control de movimiento simplifican el desarrollo de todas sus aplicaciones de control de movimiento y proporcionan una fácil integración con productos de adquisición de datos y visión. Desde la automatización de equipos de pruebas y laboratorios de investigación hasta el control de máquinas biomédicas, de empaquetado y de producción, los ingenieros y los científicos utilizan movimiento para resolver una variedad de retos de aplicación, más rápido y a un menor costo.

Un sistema de control de movimiento consiste de cinco principales componentes: el dispositivo mecánico que se está moviendo, el motor (servo o por pasos) con retroalimentación y E/S de movimiento, el amplificador, el controlador inteligente y el

software de interfaz de programación/operación. En la Figura 2.1 se muestra un sistema de control común.

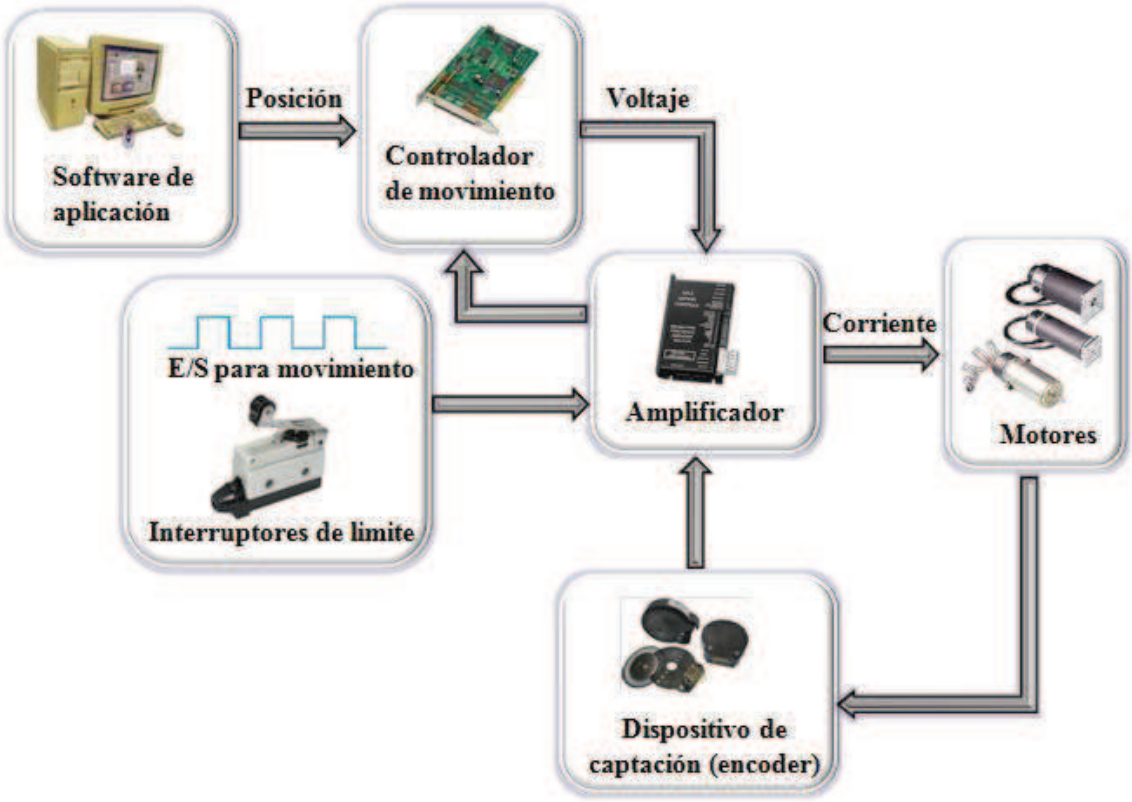

Figura 2.1 Sistema de control de movimiento básico.

Elementos mecánicos. Los motores están diseñados para proporcionar torque a algunos dispositivos mecánicos. Éstos incluyen deslizadores lineales, brazos robóticos y actuadores especiales.

Motor. Los motores convierten la energía eléctrica en energía mecánica y producen el torque requerido para moverse a la posición deseada.

La selección de motor y diseño mecánico es una parte crítica en el diseño de su sistema de control de movimiento. Muchas compañías de motores ofrecen asistencia para elegir el motor correcto. La Tabla 2.1 describe diferentes tecnologías de motores.

Tabla 2.1 Descripción de Motores (www.ni.com/motion/esa).

	Pros	Contras	Aplicaciones
Motores de Pasos.	Económico, puede ejecutar en lazo abierto, buen torque de eje inferior, limpio.	Ruidoso y resonante, torque de alta velocidad mediocre, no apto para ambientes calientes, no apto para cargas variables.	Posicionamiento, micro movimientos.
Servomotores de Escobillas DC.	Económico, velocidad moderada, buen torque de eje inferior, drives simples.	Requiere mantenimiento, no es limpio, las chispas de las escobillas causan EMI y peligro en ambientes explosivo.	Control de velocidad, control de posición a alta velocidad.
Servomotores sin Escobillas	No requiere mantenimiento, larga vida, sin chispas, altas velocidades, limpio, callado, ejecuta sin calentarse.	Drives caros y complicados.	Robótica, tomar y colocar, aplicaciones de alto torque.

Dispositivo de retroalimentación o sensor de posición. Algunas aplicaciones de control de movimiento (por ejemplo, control de motores de pasos) no requieren ningún dispositivo de retroalimentación de posición, pero para los servomotores sí es vital. El dispositivo de retroalimentación, generalmente un codificador de cuadratura, detecta la posición del motor y reporta el resultado al controlador, y de esa manera cierra el lazo con el controlador de movimiento

Los dispositivos de retroalimentación ayudan al controlador de movimiento a saber la ubicación del motor. El dispositivo de retroalimentación más común es el codificador de cuadratura, el cual proporciona posiciones en relación al punto de inicio. La mayoría de los controladores de movimiento están diseñados para funcionar con este tipo de codificadores. Otros dispositivos de retroalimentación incluyen potenciómetros que proporcionan retroalimentación de posición analógica, tacómetros que proporcionan retroalimentación de velocidad, codificadores absolutos para medidas de posición absoluta y captadores que también proporcionan medidas de posición absoluta.

Amplificador o drive. Los amplificadores (también llamados drives), es la parte del sistema que toma el comando desde el controlador de movimiento. El comando es tomado en forma de señales de voltaje analógico con baja corriente, y convertido a señales de alta corriente para dirigir o girar el motor. Existe una gran variedad de drives de motores; estos deben coincidir con el tipo de motor específico que dirigen. Por ejemplo, el drive de un motor de pasos se conecta a un motor de pasos y no a un servomotor. Aparte de coincidir con la tecnología del motor, el drive debe proporcionar la corriente de pico, corriente continua y voltaje correctos para dirigir el motor. Si el drive proporciona demasiada corriente, se corre el riesgo de dañar el motor. Si el drive proporciona muy poca corriente, el motor no alcanzará la capacidad de torque completa. Si el voltaje es demasiado bajo, el motor no podrá ejecutar a toda su velocidad.

Controlador de movimiento. El controlador de movimiento actúa como cerebro del sistema. Toma los perfiles de las posiciones y movimientos deseados y crea las trayectorias que deberán seguir los motores. Después entrega una señal de ±10 V a servomotores o pulsos de paso y dirección a motores de pasos.

Software de aplicación. Se puede utilizar software de aplicación para indicar posiciones deseadas y perfiles de control de movimiento.

Las tareas de movimiento generalmente son críticas y a menudo operan máquinas que pueden dañar a seres humanos. Por lo tanto, se requieren funciones de seguridad como conmutadores de límite y canales de E/S, para obtener información de estatus y ejecutar rutinas de apagado.

2.4 Controlador de Movimiento

Para realizar movimientos de precisión en máquinas CNC dentro de un intervalo de tiempo definido es necesario el empleo de un controlador de posición que coordine la ejecución del movimiento. El controlador recibe instrucciones (por lo general desde una computadora) acerca del movimiento que se desea realizar. La mínima información necesaria para efectuar el movimiento es la posición final y tiempo en que se debe ejecutar el movimiento. Con estos datos el controlador actúa sobre la máquina por medio de servomotores acoplados a cada grado de libertad (Rivera, 2007).

Un controlador de movimiento actúa como el cerebro del sistema de control de movimiento, y calcula la trayectoria para cada movimiento especificado. Debido a que esta tarea es vital, a menudo se ejecuta en un procesador digital de señales (DSP) en la tarjeta misma del controlador para prevenir interferencias con la computadora principal (no se quiere que el movimiento se detuviera debido a que un software antivirus empezara a ejecutarse). El controlador de movimiento utiliza las trayectorias que calcula para determinar el comando de torque adecuado que debe enviar al amplificador del motor, y así causar movimiento.

El controlador de movimiento también debe cerrar el lazo de control PID. Debido a que esto requiere un alto nivel de determinismo y es vital para una operación consistente, el lazo de control generalmente se cierra en la tarjeta misma. Además de cerrar el lazo de control, el controlador de movimiento también ejecuta control de supervisión al monitorizar los límites y los detenimientos de emergencia para garantizar una operación segura. El hecho de que estas operaciones se efectúen en la tarjeta o en un sistema en tiempo real asegura la fiabilidad, el determinismo, la estabilidad y la seguridad de alto nivel necesarios para crear un sistema de control de movimiento funcional.

2.4.1 DMC 18x2

La serie DMC-18x2 de Galil Motion Control, Inc. son controladores de movimiento PCI bus para la aplicación de múltiples ejes. La DMC-18x2 incorpora un microprocesador de 32-bit y proporciona características avanzadas tales como la compensación PID con *feedforward* velocidad y la aceleración, la memoria con múltiples tareas al mismo tiempo para correr hasta ocho programas, y E/S para la sincronización del movimiento con acontecimientos externos. Los modos de movimiento incluyen la colocación de punto a punto, trotar, interpolación lineal y circular, contornear, engranaje electrónico, y ECAM (*Electronic Centralized Aircraft Monitoring,* Aviones de Vigilancia Electrónica Centralizada). Al igual que todos los reguladores de Galil, los controladores DMC-18x2 utilizar un lenguaje sencillo, de control intuitivo que hace que sean fáciles de programar. GalilTools software simplifica aún más la configuración del sistema con “un solo botón” puesta a punto del servo y en tiempo real visualiza la información de posición y velocidad. En la Figura 2.2 se muestra la tarjeta de control DMC-1842, la cual puede controlar hasta cuatro ejes.

Figura 2.2 Tarjeta de control DMC-1842 4-ejes PCI (Galil, 2010).

2.4.2 Controlador de Posición Basado en FPGA

En la Figura 2.3 se muestra un diagrama a bloques del controlador de posición implementado en la tarjeta Spartan-3, el cual se compone de un bloque Generador de perfiles que proporciona una referencia a seguir por el controlador PID, la señal de control se envía por medio de la interfaz DAC a un convertidor digital analógico de 12 bits para posteriormente ser amplificada por el Servoamplificador y así poder ser aplicada al Servomotor (Rivera, 2007).

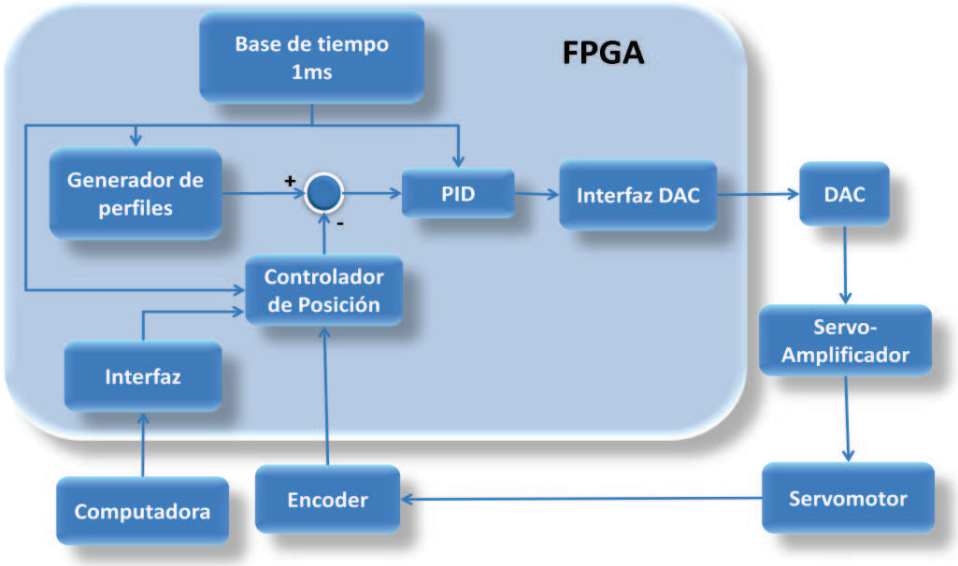

Figura 2.3 Controlador de movimiento basado en FPGA.

PID. El bloque controlador PID (Proporcional Integral Derivativo) se encarga básicamente de hacer cero el error entre la posición de referencia proporcionada por el generador de perfiles y la posición real de la máquina.

Sintonía. Se encarga de identificar la planta que se encuentra controlando, en este caso una máquina CNC. Para identificar la planta, se requiere medir la respuesta de la máquina a una perturbación controlada; cuando se identifica el comportamiento de la máquina se establecen los parámetros del PID en función de los valores entregados por la planta.

Generador de perfiles. Este bloque se encarga de proporcionar una referencia variable que debe seguir el controlador PID, esta referencia variable es el perfil de movimiento que contiene toda la información de la dinámica de movimiento con la que se desea que trabaje la máquina CNC.

Base de tiempo. El controlador de posición debe trabajar a una frecuencia de muestreo por lo menos de 1KSPS, las operaciones realizadas por cada bloque dentro del FPGA se realizan con una frecuencia mucho mayor por lo que es necesario contar con una base de tiempo que haga que el controlador actúe sobre la máquina CNC a una frecuencia de muestreo que esté dentro de los estándares industriales y además permita la interacción de cada bloque de forma sincronizada.

Servomotor. Trabajan de manera similar a los motores convencionales y permiten ejecutar los movimientos en una máquina CNC. Los servomotores convencionales utilizan un motor de corriente directa y un sistema de engranajes reductores que brinda al motor mayor fuerza y baja inercia. El controlador de movimiento actúa sobre el servomotor para ejecutar el correspondiente movimiento en la máquina CNC.

Encoder. Es un transductor rotativo que transforma un movimiento angular en una serie de impulsos. Un servomotor tienen acoplado un encoder y a partir de los pulsos generados por el movimiento es posible obtener velocidad o posición del mismo.

Contador de posición. A este bloque se le envían los pulsos obtenidos del encoder para encontrar la posición en la que se encuentra la máquina. Este valor se compara con la

posición de referencia que proporciona el generador de perfiles y el controlador PID se encarga de minimizarlo.

Interfaz DAC. Los controladores de posición actuales trabajan de manera digital, para realizar la conexión con el mundo real es necesario el uso de un DAC (*Digital to Analog Converter*, Convertidor Digital Analógico) la Interfaz DAC es una especie de traductor entre el controlador de posición y el DAC.

Servoamplificador. El DAC proporciona una señal de control analógica la cual es muy débil para ser conectada directamente al servomotor, el servoamplificador se encarga de reforzar esta señal para que tenga la suficiente potencia para mover al servomotor.

Interfaz PC. El controlador de posición debe recibir las órdenes de movimiento, se debe establecer la posición final, y el tiempo en que se realiza el movimiento. Es común que las órdenes sean dadas por una computadora que a la vez se apoya de software CAM para generar las órdenes necesarias para el controlador de movimiento, estas instrucciones son interpretadas por la interfaz PC para que el controlador de posición las ejecute.

En la Figura 2.4 se muestra los elementos para un sistema de control de posicionamiento basado en FPGA.

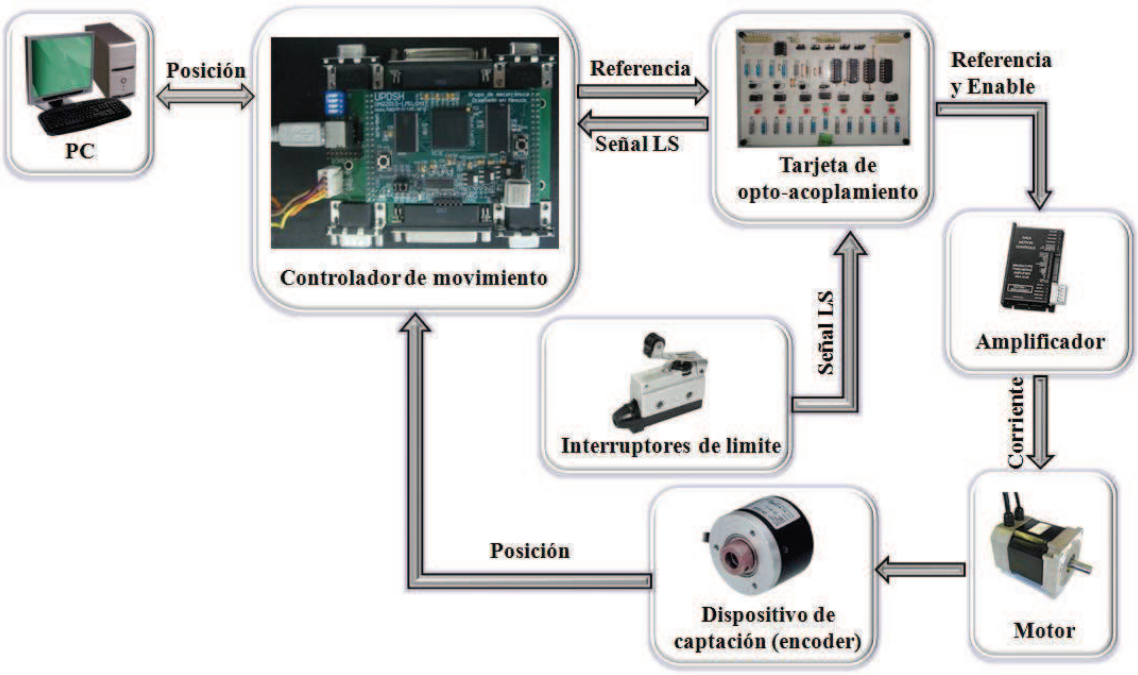

Figura 2.4 Elementos para un controlador de movimiento basado en FPGA.

2.5 Lenguaje de Programación NC

El lenguaje de programación de Control Numérico, a veces llamada código G, es el estándar más común usado a nivel industrial, pero en realidad, el código G es sólo una parte del programa NC que controla el lenguaje de programación NC y CNC máquinas herramienta. Actualmente, las máquinas convencionales CNC están diseñadas únicamente para la interpolación de curvas lineales y circulares bajo el estándar ISO 6983 (*International Estandarization Organization,* Organización Internacional de Estandarización) y la EIA RS274 (*Electronic Industries Association*, Asociación de Industrias Electronicas) conocidos como códigos G y M. Sin embargo, en equipos modernos se han introducido nuevos estándares de comunicación entre el CAD/CAM y el control CNC, que son basados en tareas enlatadas STEP-NC ISO 14649 junto con los códigos relacionados con la generación de curvas paramétricas Fanuc G06.2.

Los códigos G son funciones de movimiento de máquinas que incluyen movimientos rápidos, avances radiales, pausas y ciclos (Smid 2003). Se les conoce también como comandos preparatorios y requeridos para el maquinado de piezas.

Los códigos M son funciones misceláneas que se requieren para el maquinado de piezas pero que no son funciones de movimiento de la máquina. Algunas de ellos pueden ser el arranque, el paro del husillo, cambio de herramienta, encendido y apagado del refrigerante, paro de programa, etc. (Smid, 2003).

En la Figura 2.5 se muestra la estructura de un programa que es la llave del entendimiento de la tecnología CNC, esta debe tener la siguiente información: dirección, dato, palabra y programa.

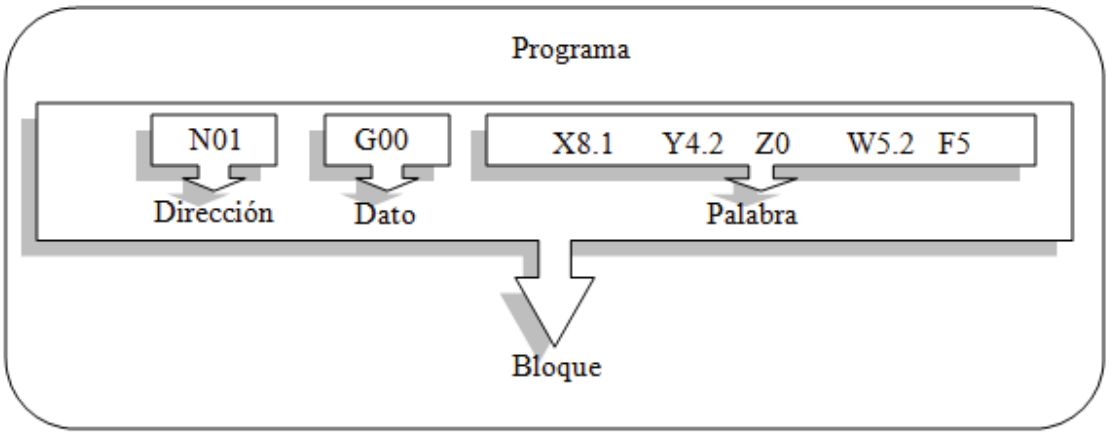

Figura 2.5 Formato de programa NC.

Dirección. Es la numeración de la instrucción, se usa al inicio con la letra mayúscula N seguida de dos dígitos 01 hasta 99999, si las líneas rebasan esta numeración es recomendable hacer dos o más programas separados.

Dato. Es una combinación de caracteres alfanuméricos que crean una instrucción simple. Generalmente empezando por una letra seguida de un número.

M es la dirección correspondiente a las funciones auxiliares o complementarias. La dirección M va seguida de un número de dos cifras.

G es la dirección correspondiente a las funciones preparatorias. Se utiliza para informar al control de las características de las funciones de mecanizado. La función G va seguida de un número de dos cifras.

Palabra. Muestra los movimientos de los ejes a realizar, cada eje inicia con la letra mayúscula seguida de un número entero o flotante que determina el desplazamiento de éste.

X, Y, Z, U, V, W son las direcciones correspondientes a las cotas según los ejes *X, Y, Z* de la máquina-herramienta. Dichas cotas se pueden programar en forma absoluta, es decir, con respecto al cero pieza o con respecto a la última cota respectivamente donde se utilizan las variables *X*, *Y* y *Z*. También se pueden programar en forma incremental, es decir, que las coordenadas se van incrementando para lo cual se utilizan las variables *U*, *V* y *W*.

F es la dirección correspondiente a la velocidad de avance. Va seguida de un número de cuatro cifras que indica la velocidad de avance en mm/min.

S es la dirección correspondiente a la velocidad de rotación del husillo principal. Se programa directamente en revoluciones por minuto, usando cuatro dígitos.

I, J, K son direcciones utilizadas para programación arcos de circunferencia. Cuando la interpolación se realiza en el plano X-Y, se utilizan las direcciones I y J. Análogamente, en el plano X-Z, se utilizan las direcciones I y K, y en el plano Y-Z, las direcciones J y K.

T es la dirección correspondiente al número de herramienta. Va seguido de un número de cuatro cifras en el cual los dos primeros indican el número de herramienta y los dos últimos el número de corrección de las mismas.

Bloque (renglón). Un bloque se encuentra compuesto de un grupo de palabras que forma una instrucción múltiple.

Programa. Es una lista de bloques que siguen un orden lógico.

2.5.1 Funciones Preparatorias

Se programan mediante la letra G seguida de dos cifras (G02) siempre al comienzo del bloque y sirven para determinar la geometría y condiciones de trabajo. Aunque existe

una estandarización (ISO 6983) pueden ser modificadas por los fabricantes y muchas de ellas no están determinadas.

Las funciones G se encuentran divididas en grupos. En una secuencia de programa sólo puede haber una función G de cada grupo. Las funciones G se activan de forma modal (auto mantenidas) o secuencialmente. Las que actúan modalmente son aquellas que siguen activas mientras no sean reemplazadas por una nueva función G del mismo grupo y las que actúan secuencialmente son aquellas que son activas sólo en la secuencia en la que se encuentran.

Las posiciones preferenciales se activan después de la conexión del control, tras el *reset* o tras el fin de programa. Estas no necesitan ser programadas. Son las asumidas siempre por defecto en ausencia de cualquier otra especificación (Smid, 2003).

Función G00. El trayecto de la herramienta en un bloque con G00 se realiza a la máxima velocidad posible por el control. Cuando acaba el bloque, el avance *F* anterior permanece.

Durante este movimiento no se mecaniza. El desplazamiento rápido se programa mediante la información de desplazamiento G00 y mediante la indicación del punto de destino. Este punto es alcanzado introduciendo cotas absolutas (G90) o cotas incrementales (G91). El desplazamiento rápido puede implicar una interpolación lineal o bien ser escalonado. La trayectoria programada con G00 se recorre con la máxima velocidad, el desplazamiento rápido, en una línea recta, sin maquinar la pieza (interpolación lineal). Para esto, el control supervisa la máxima velocidad permitida del eje. Esta velocidad se fija para cada eje como dato de máquina. El movimiento de desplazamiento se determina a través del menor valor de las velocidades del eje que han sido fijadas como dato de máquina (Smid, 2002).

Al programar G00, el valor para el avance programado bajo la dirección *F*, permanece en memoria y vuelve a ser activo, por ejemplo con G01.

Función G01. Interpolación lineal. Mientras no se especifique otro tipo de interpolación, los bloques siguientes realizarán los movimientos entre puntos siguiendo rectas.

Función G02, G03. La función G02 permite realizar interpolación en sentido horario (en dirección de las manecillas del reloj), mientras que G03 la interpolación en sentido antihorario (en dirección contraria de las manecillas del reloj).

Para la realización de las interpolaciones además del número de registro, las coordenadas del punto final y el avance se deben de otorgar los parámetros de *I, J* y *K* de pendiendo en el plano que se vaya a trabajar (Smid, 2003).

Función G06.2. Interpolación NURBS, ésta interpolación requiere una serie de palabras las cuales contienen los parámetros necesarios para desarrollar la interpolación. Los parámetros de la palabra son:

- P: Rango de la curva
- X, Y, Z: Punto de control
- R: El peso
- K: Nodo
- F: Velocidad de avance

Un rango de NURBS se puede especificar con la dirección de P. El ajuste rango, en su caso, se debe especificar en el primer bloque. Si el ajuste de rango se omite, rango de cuatro (grado de tres) se asume para los NURBS. El rango de datos válidos para P es de 2 a 4. Los valores de P tienen los siguientes significados:

- P2: NURBS tiene un rango de dos (grado de uno)
- P3: NURBS tiene un rango de tres (grado de dos)
- P4: NURBS tiene un rango de cuatro (grado de tres) (por defecto)

El peso de un punto de control se puede de finir en bloque programado. Cuando el ajuste del peso se omite, un peso de 1.0 es asumido.

El número de nodos especificado debe ser igual al número de puntos de control más el valor de rango. En los bloques se especifica del primero al último punto de control, cada punto de control y nodo son especificados en un bloque idéntico. La curva NURBS programada para la interpolación NURBS debe comenzar desde el primer punto de control y terminan en el último punto de control. Los primeros nodos *k* (donde *k* es el rango) deben tener los mismos valores que los nodos anteriores *k* (múltiples nodo).

Función G17. Selecciona el plano X-Y como el plano de trabajo para las interpolaciones circulares, en caso de que no sea especificado el plano este es el plano que se considera por default.

Función G18. Selecciona el plano X-Z como el plano de trabajo para las interpolaciones circulares.

Función G19. Selecciona el plano Y-Z como el plano de trabajo para las interpolaciones circulares.

Función G20. Selecciona los milímetros como unidades.

Función G21. Selecciona las pulgadas como unidades.

Función G90. Indica que el programa se encuentra en coordenadas absolutas, en caso de que no se indique el tipo de coordenadas de trabajo las coordenadas absolutas son seleccionadas por default.

Función G91. Indica que el programa se encuentra en coordenadas relativas.

2.6 Código DMC

Galil ha creado una colección de muestras DMC Código diseñado para enseñar a los principiantes y los usuarios más expertos acerca de la programación de los reguladores de Galil.

A continuación se muestra en la Tabla 2.2 los comandos DMC utilizados para este trabajo.

Tabla 2.2 Descripción de los comandos DMC.

COMANDO	PARÁMETROS	DESCRIPCIÓN
DP	X, Y, Z,W	Asigna valores a los ejes
PR	X, Y, Z,W	Movimiento especificando las coordenadas relativas

PA	X, Y, Z, W	Movimiento especificando las coordenadas absolutas
SP	X, Y, Z, W	Especifica la velocidad para cada eje
VS	S	Especifica la velocidad total
AC	X, Y, Z, W	Especifica la aceleración para cada eje
VA	S	Especifica la aceleración total
DC	X, Y, Z, W	Especifica la desaceleración para cada eje
VD	S	Especifica la desaceleración total
ST		Paro del movimiento
BG	XYZW	Comienza el movimiento de los ejes especificados
VM	XY	Inicio movimientos coordinados
VP	X,Y	Vector lineal
CR	R, θ, Δθ	Circunferencia
VE		Final de secuencia (VM)
LM	XYZW	Inicio de secuencia lineal
LI	X, Y, Z, W	Especifica valores absolutos de los ejes
LE		Final de secuencia (LM)
BGS		Iniciar secuencia
AM	XYZW	Espera a que termine el movimiento de los ejes especificados
AMS		Espera a que termine la secuencia
EN		Final de programa

2.7 Curvas paramétricas

Un vector $x \in \Re^n$ cuyas coordenadas dependen de un parámetro t recorre una curva paramétrica $x \in \left(x_1(t), x_2(t), ..., x_n(t)\right)$. En particular, si las funciones $x_k(t)$ son polinomios de grado menor o igual que n entonces $x(t)$ es una curva polinomial de grado n en t (Paluszny *et al.*, 2005).

El gráfico de una función $x(t)$ es una curva que tiene una forma especial $x(t) = \left(t, x(t)\right)$. Las curvas descritas por gráficos de funciones se denominan curvas funcionales.

2.7.1 Curvas de Bézier

El cálculo de la expansión binomial

$$1 = \left(u + (1-u)\right)^n = \sum_{i=0}^{n} \binom{n}{i} u^i (1-u)^{n-i} \tag{2.1}$$

Permite introducir los polinomios de Bernstein de grado n definidos en la ecuación (2.1) donde u es la variable independiente.

$$B_i^n(u) = \binom{n}{i} u^i (1-u)^{n-i} \quad i = 0,1,2,...,n \tag{2.2}$$

Algunas propiedades importantes para los polinomios de Bernstein de grado n son:

- Son linealmente independientes.
- Son simétricos.
- Las únicas raíces son en 0 y 1.
- Forman una partición de la unidad.
- Son positivos en (0,1).
- Satisfacen la relación de recurrencia.

$$B_i^{n+1}(u) = uB_{i-1}^n(u) + (1-u)B_{i-1}^n(u) \tag{2.3}$$

Los polinomios de Bernstein $B_i^n(u)$ de grado n forman una base para el espacio vectorial de polinomios de grado menor o igual que n. Por lo tanto toda curva polinomial $b(u)$ de grado menor o igual que n tiene una única representación de Bézier establecida en la ecuación (2.4).

$$b(u) = \sum_{i=0}^{n} c_i B_i^n(u) \tag{2.4}$$

La transformación afín $u = a(1-t) + bt$, deja invariante el grado de b, por lo tanto $b(u(t))$ también tiene una única representación de grado n, en términos de los $B_i^n(u)$.

$$b(u(t)) = \sum_{i=0}^{n} b_i B_i^n(t) \tag{2.5}$$

Los coeficientes b_i se denominan puntos de Bézier y son los vértices del polígono de Bézier de $b(u)$ sobre el intervalo $[a,b]$. Se refiere a t como el parámetro local y a u como el parámetro global de b.

2.7.2 B-Spline

De manera análoga a la representación de Bézier de curvas polinomiales también es conveniente expresar algunas curvas $s(u)$ como una combinación afín de ciertos puntos de control c_i, esto es:

$$s(u)=\sum_{i=0}^{n} c_i N_i^n(t) \tag{2.6}$$

Donde los $N_i^n(t)$ son funciones polinomiales por trozos con soporte finito y satisfacen ciertas condiciones de continuidad. Estas funciones reciben el nombre de B-splines.

Se puede introducir una relación de recurrencia para definir los B-splines, se considera por simplicidad una secuencia a_i doblemente infinita de nodos simples tales que $a_i < a_{i+1}$ para todo i. Entonces los B-splines N_i^n se definen a través de la siguiente relación de recurrencia.

$$N_i^0(u)=\begin{cases}1 & \text{si} \quad u\in[a_i,a_{i+1}]\\ 0 & \text{en caso contrario}\end{cases} \tag{2.7}$$

$$N_i^n(u)=\alpha_i^{n-1}N_i^{n-1}(u)+\left(1-\alpha_{i+1}^{n-1}\right)N_{i+1}^{n-1}(u) \tag{2.8}$$

Donde

$$\alpha_i^{n-1}=\frac{u-a_i}{a_{i+n}-a_i} \tag{2.9}$$

El cual es parámetro local con respecto al soporte de N_i^{n-1}.

La desventaja del modelo B-spline frente al de Bézier, aparte de la mayor complejidad matemática, consiste en que las funciones base no admiten una expresión explicita, y cambia al variar el vector de nodos. De hecho, muchos programas CAD, para procesar splines transforman cada segmento al modelo Bézier y así el procesado resulta más eficiente.

2.7.3 NURBS

NURBS (*Non Uniform Rational B-splines*) son el estándar para describir y modelar curvas y superficies sistemas CAD y gráficos por computadora. Son usadas para modelar cualquier cosa desde cuerpos de automóviles y esqueletos de barcos hasta personajes animados en películas. Una curva NURB se define como

$$s(t)=\frac{\sum_{i=0}^{n} N_i^p(t) w_i Q_i}{\sum_{i=0}^{n} N_i^p(t) w_i}=\frac{A(t)}{w(t)} \tag{2.10}$$

Donde Q_i son los puntos de control, w_i son los pesos correspondientes de Q_i. Además, $w(t)$ es la función de peso, $A(t)$ es la función B-spline de peso, $(n+1)$ es el número de puntos de control y p es el grado de la curva NURBS. La función base B-spline de grado p se define recursivamente como (2.8) y (2.9) para $i=0,1,2,...,n$.

2.8 GTK+

GTK+ o el conjunto de herramientas de desarrollo para el escritorio GNOME y el editor de imágenes GIMP (*General Image Manipulation Program*, Programa de Manipulación de Imágenes General) es un conjunto de bibliotecas multiplataforma para desarrollar interfaces gráficas de usuario, principalmente para los entornos gráficos GNOME, XFCR y ROX aunque también se puede usar en el escritorio Windows, MacOS y otros. Los desarrolladores eligen GTK+ por una variedad de razones, una de las principales es que GTK+ se encuentra bajo la licencia de GNU LGPL 2.1 (*Library General Public License*, Biblioteca de la Licencia Pública General) permitiendo el desarrollo tanto en software libre y propietario con GTK+ sin derechos de licencia o regalías, otros lo prefieren

por el rendimiento y el apoyo a la internacionalización. Algunos lo prefieren por el lenguaje de programación que utilizan. Ya que GTK + está escrito en C, pero tiene enlaces con muchos lenguajes de programación, tales como C++, Python y C# entre otros. (www.gtk.org).

GTK+ se basa en varias bibliotecas del equipo de GTK+ y GNOME:

- Glib: Biblioteca de bajo nivel estructura básica de GTK+ y GNOME. Proporciona manejo de estructura de datos para C, portabilidad, interfaces para funcionalidades de tiempo de ejecución como ciclos, hilos, carga dinámica o un sistema de objetos.

- GTK: Biblioteca la cual realmente contiene los objetos y funciones para crear la interfaz de usuario. Maneja *widgets* como ventanas, botones, menús, etiquetas, deslizadores, pestañas, etc.

- ADK: Biblioteca que actúa como intermediario entre gráficos de bajo nivel y gráficos de alto nivel.

- GDK: Biblioteca para crear interfaces con características de una gran accesibilidad muy importante para personas con alguna discapacidad. Pueden usarse utilerías como lupas de aumento, lectores de pantalla, o entradas de datos alternativas al clásico teclado o ratón.

- Pango: Biblioteca para el diseño y renderizado de texto, hace hincapié especialmente en la internacionalización. Es el núcleo para manejar las fuentes y el texto de GTK+2.

- Cairo: Biblioteca de renderizado avanzado de controles de aplicación.

2.8.1 Gtkmm

Gtkmm es la interfaz C++ oficial para la biblioteca de interfaz gráfica GTK+. Destacan las devoluciones de llamadas de tipo seguro, y un exhaustivo conjunto de *widgets* son fácilmente extensibles mediante herencia. Se pueden crear interfaces de usuario ya sea

en código o con el diseñador de la interfaces gráficas Glade, usando libglademm. Gtkmm es software libre distribuido bajo la GNU LGPL. (www.gtkmm.org).

Características:

- Usar la herencia para derivar *widgets* personalizados.
- Manejadores de señales de tipo seguro, en C++ estándar.
- Polimorfismo.
- El uso del estándar de C++ Biblioteca, incluidas las cadenas, los contenedores y los iteradores.
- Lleno de internacionalización con UTF8.
- Completa C++ de administración de memoria.
- Objeto de la composición.
- Desafectación automática de *widgets* dinámicamente asignada.
- Uso completo de C++ espacios de nombres.
- No macros.
- Compatibilidad con múltiples plataformas: Linux (gcc), FreeBSD (gcc), NetBSD (gcc), Solaris (gcc, Forte), Win32 (gcc, MSVC++. Net 2003), MacOS X (gcc), otros.
- Software libre y gratuito tanto para el desarrollo de código abierto y propietario.
- Comentado, diseñado y ejecutado en público.

2.9 Diseñador de Interfaz Glade

El diseñador de interfaz Glade es una herramienta de desarrollo visual de interfaces gráficas mediante GTK/GNOME. Es independiente del lenguaje de programación y predeterminadamente no genera código fuente sino un archivo XML. La posibilidad de generar automáticamente código fuente fue descontinuada desde Glade-3.

Aunque tradicionalmente se ha utilizado de forma independiente, está totalmente integrado en el recientemente liberado Anjuta 2. Cuenta con tres versiones, la primera para GTK+1 y las otras para GTK+2. Se encuentra bajo la licencia GPL.

El primer lanzamiento de Glade, la versión 0.1, se hizo el 18 de abril de 1998. Y Glade-3 se lanzó el 12 de agosto de 2006. Las diferencias más notorias para el usuario final son:

- "Deshacer" y "rehacer" disponible para todas las operaciones.
- Permite abrir varios proyectos simultáneamente.
- Remoción de la generación automática de código fuente.
- Ayuda contextual mediante DevHelp.

Sin embargo, la mayoría de las diferencias son internas. Glade-3 fue reescrito completamente, para poder tomar ventaja de las nuevas características de GTK+2 y el sistema GObject (Glade-3 comenzó a escribirse antes de que Glade-2 fuese portado a GTK+2). Por lo tanto el código principal de Glade-3 es más pequeño y permite nuevas cosas interesantes, incluyendo catálogo de *widgets* "enchufables" ("pluggable" *widgets*). Esto significa que bibliotecas externas pueden proveer su conjunto de *widgets* en tiempo de ejecución y Glade los detectará. De hecho, Glade-3 soporta sólo *widgets* estándar de GTK+; los *widgets* GNOME UI y DB son provistos por separado.

Las herramientas de Glade (paleta, editor, etc.) son implementadas como *widgets*. Esto permite una fácil integración con IDEs (*Integrated Development Environment*, Entorno de Desarrollo Integrado) como Anjuta o Scaffold, y hace que sea más fácil cambiar la interfaz.

GladeXML es un formato XML que Glade usa para almacenar los elementos de las interfaces diseñadas. Estos archivos pueden emplearse para construirla en tiempo de ejecución mediante la biblioteca libglade. Algunas versiones de Glade permitían generar automáticamente el código que generaría las interfaces; pero fue desaconsejado y descontinuado. En la Figura 2.6 se muestra la pantalla inicial del diseñador de interfaz grafica Glade.

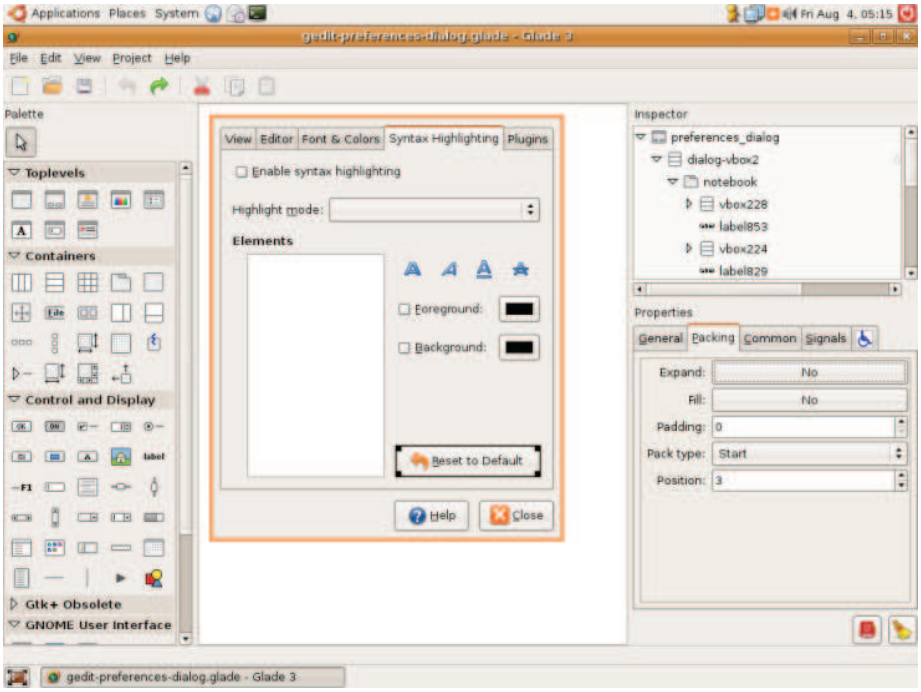

Figura 2.6 Diseñador de interfaz Glade.

Capítulo III.

METODOLOGÍA

En este trabajo requiere el diseño de una plataforma hardware-software la cual se encuentra integrada de varios módulos principales, el diseño de la plataforma es descrita por Morales (2010). La parte en software orientará al diseño de las interfaces gráficas para la comunicación con el usuario y de los controladores de puertos para la conexión de los módulos de procesamiento en hardware y la interfaz de usuario. Este trabajo se enfoca principalmente en la parte de GUI del software donde se requieren una serie de procedimientos jerárquicos para el manejador de gráficos ya que esté tiene una comunicación bidireccional con los objetos gráficos.

El planteamiento general propuesto de este trabajo se muestra en la Figura 3.1 donde se observa que el proceso de manufactura integrando la parte del software. La ventana principal de la GUI se encuentra dividida en cuatro secciones principales. La primera sección es un editor, donde el usuario puede escribir o modificar algún programa ya sea de código G ó DMC. En esta parte el usuario debe de seleccionar el tipo de código que va a implementar y su principal función es la compilación del programa. La siguiente sección es el simulador que se encuentra ampliamente ligada con la sección del editor pues al ser compilado el programa, se carga automáticamente el código funcional del programa. Este trabajo se enfoca principalmente en esta sección de tal forma que es donde se realiza la interpretación de los comandos tanto del código G como del DMC para la realización de una simulación ó una ejecución. La tercera sección es el *jog* la cual es una herramienta que se encuentra en la mayoría de los programas para CNC utilizada principalmente para ubicar la herramienta en una posición y a partir de esta ejecutar un programa para maquinar. La última sección se encuentra enfocada al código DMC ya que este es un código enfocado para enseñar al usuario la funcionalidad que se tiene al regular los parámetros de los perfiles como es la velocidad, aceleración y desaceleración.

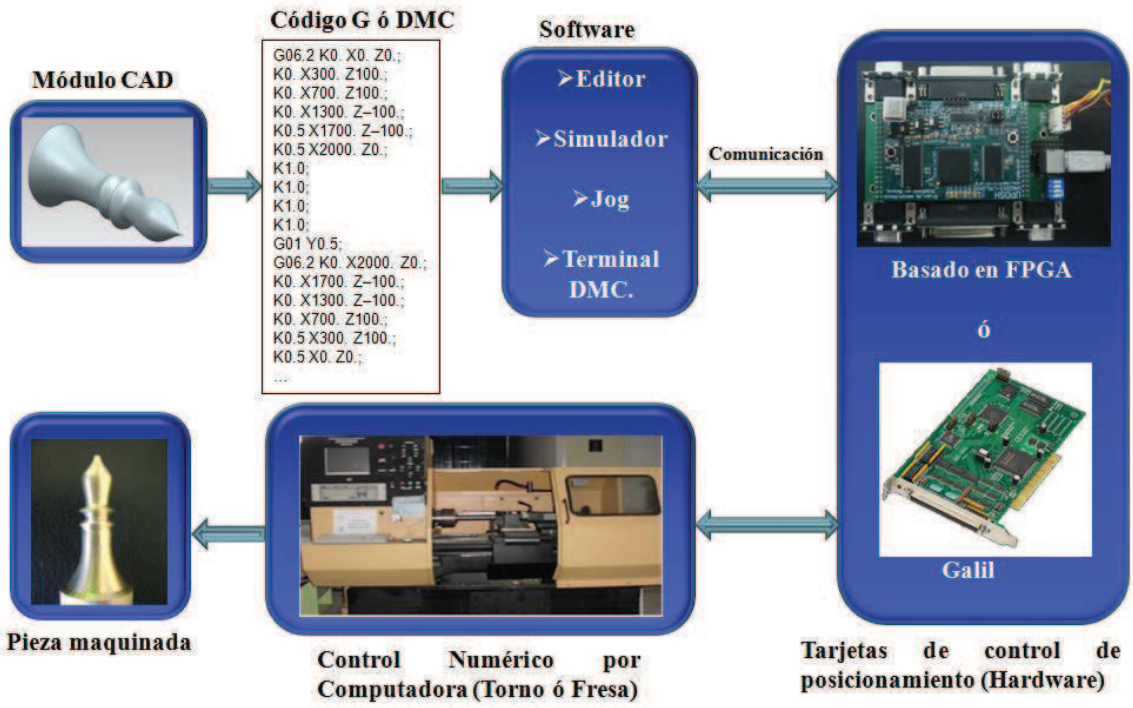

Figura 3.1 Planteamiento general propuesto para el proceso de manufactura.

3.1 Comunicación

En base a la plataforma hardware-software realizada por Morales (2010) se debe describir la unidad DRC en un archivo de formato XML, en el cual se encuentra representada la unidad DRC mediante un conjunto de registros, un conjunto de paquetes y subrutinas de operación. Dicho archivo debe tener la extensión .dcx para que sea apropiadamente identificado por el manejador de archivos desarrollado y descrito por Morales (2010).

La descripción de un DRC comienza por la entidad DRC que debe llevar cuatro propiedades:

- Nombre de la unidad DRC (name): MCUAQ4X
- Número de identificación del DRC (id): 1
- Versión del DRC (version): 0.0
- Archivo de mapa de fusibles (bitfile): No es requerido
- Dispositivo en el que está basado el DRC (device): Spartan3E-1600

- (eeprom): 32768

El puerto mediante el cual se comunica la PC con el DRC es el USB para el cual se utilizaron las siguientes propiedades:

- Nombre del dispositivo (name): USB
- Versión del USB (version): 1.1
- Número de identificación del fabricante (VID): 0X2603
- Número de identificación del protocolo (PID): 0X2401

Dentro de la estructura del puerto USB se colocan los conductos (pipes) de comunicación que soporta la unidad DRC, en este trabajo solo se ocupan dos pipes una de entrada y una de salida, la entidad Pipe de salida tiene las siguientes propiedades:

- Nombre del pipe (name): usbout
- Dirección del flujo de datos (dir): out
- Dirección física del pipe (addr): 0X01
- Tamaño del pipe (bytes): 64

La entidad Pipe de entrada tiene las siguientes propiedades:

- Nombre del pipe (name): usbin
- Dirección del flujo de datos (dir): in
- Dirección física del pipe (addr): 0X82
- Tamaño del pipe (bytes): 64

La siguiente parte de la descripción corresponde a los registros internos que tiene la unidad DRC. Todos los datos internos al DRC pueden ser organizados mediante una dirección única pero pueden variar en tamaño y tipo de dato que almacenan. Además cada registro puede manejarse separado en banderas que corresponden a bits individuales partiendo del menos significativo, cada registro puede tener un máximo de 32 banderas. Con el fin de reducir la cantidad de transferencias entre el dispositivo y la PC se puede realizar el envío de un paquete de datos que contenga varios registros. Dentro de la definición del paquete se colocan los registros en orden de envío que serán encapsulados en el paquete, solo es necesario hacer referencia a los registros previamente definidos. Fue

necesario definir rutinas automáticas que involucran el envío de varios paquetes y analizar registros o banderas. Todo el proceso de la construcción de registros, paquetes y rutinas de ejecución es descrito con mayor claridad por Morales (2010).

La comunicación entre el software y el hardware requiere la definición de un protocolo que regule las transferencias en el sistema. La Figura 3.2 muestra el protocolo de comunicación para el intercambio de paquetes entre el software y el hardware.

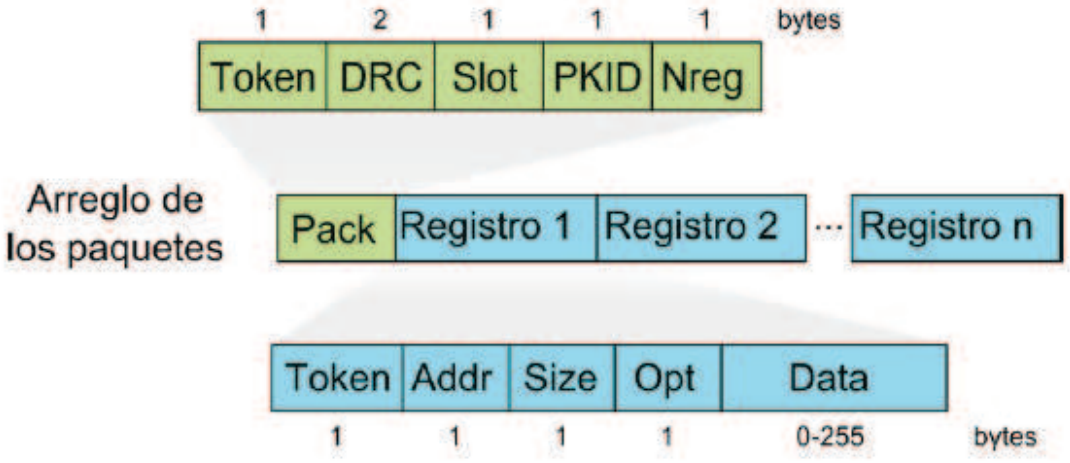

Figura 3.2 Arreglo de los paquetes en el protocolo de comunicación PC-DRC (Morales, 2010).

Mediante este protocolo los paquetes son intercambiados entre las unidades físicas DRC y las unidades virtuales agentes DRC, el formato para el envío y recepción de paquetes es el mismo para todas las transferencias y consiste en una región inicial con información del paquete seguido de la información de los registros. La región de información del paquete está dividida en varios campos: 1 byte para el tipo de sección (token), 2 bytes para el identificador del DRC involucrado en la transferencia, 1 byte del identificador único del DRC en el sistema (Slot), 1 byte para el identificador de paquete (PKID) y 1 byte para la cantidad de registros que contiene el paquete (Nreg). En el caso de la información del registro éste contiene: 1 byte para el tipo de sección (token), 1 byte para la dirección del registro involucrado (Addr), 1 byte con el tamaño en bytes del registro seleccionado (Size), 1 byte con opciones del registro (Opt) y termina con una región de datos de tamaño variable acorde al campo Size.

3.2 Editor

Esta sección se encuentra en una pestaña de la ventana principal de la interfaz GUI, se muestra en la Figura 3.3. Esta es la primera pantalla que aparece al ejecutar el programa, donde lo primero que debe realizar el usuario si desea ejecutar ó simular un programa NC ó DMC es seleccionar el tipo de código de tal manera que a partir de dicha selección se podrá abrir archivos de texto (.txt para programas NC) ó archivos DMC (.dmc para programas DMC) que se podrán modificar únicamente en la parte del editor. También podrán escribirse programas directamente en el editor, esto no quiere decir que no intervenga la selección del tipo de código puesto que la compilación del programa depende directamente de esta selección. De igual forma interviene en la ayuda que se puede observar en la interfaz la cual se encuentra basa en las descripciones de los comandos dada en este trabajo en el Capítulo II.

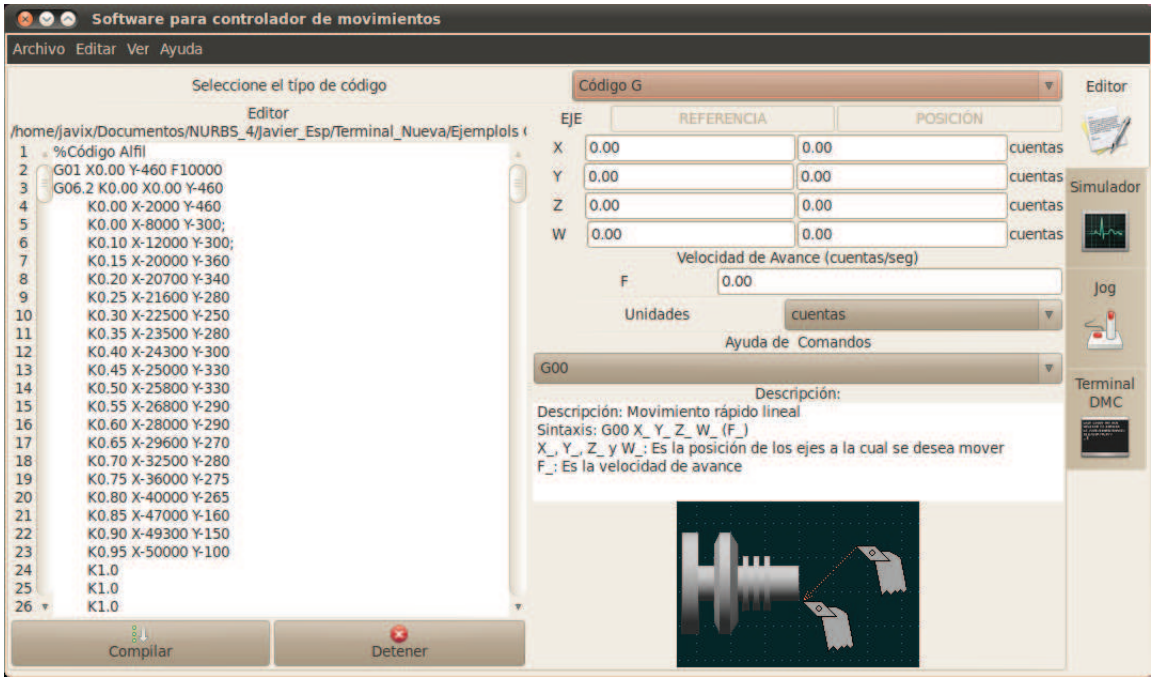

Figura 3.3 Pestaña del editor de la GUI.

En esta sección también se elije el tipo de unidades que desea observar el usuario durante la ejecución del programa, estas unidades pueden ser cuentas, milímetros, centímetros y pulgadas.

Una vez que se tiene el código en el editor de texto del software se procede a realizar la compilación del programa, sin embargo este proceso se encuentra dentro de un pre-proceso llamado lector, donde se elimina los caracteres que no se requieren como dobles espacios, dobles tabulaciones e incluso se eliminan los comentarios para cada línea de comandos. Ya eliminados estos caracteres, se realiza la compilación de tal manera que no pueden ser enviados comandos inválidos, comandos con parámetros inexistentes o simplemente comandos con errores de sintaxis.

3.2.1 Lector

En los programas NC y DMC al igual que en la mayoría de los lenguajes de programación se requiere realizar comentarios para identificar lo que realiza dicho programa. Los comentarios son especificados con el carácter '%' para el código NC y un apostrofe (') para el código DMC. El lector además de cargar el programa a una terminal gráfica, realiza un pre-proceso el cual elimina todos los espacios, tabulaciones y renglones de más que se encuentren en el programa así como los comentarios realizados por el usuario para que el programa cargado se encuentre limpio de basura y pueda realizarse una compilación de alguna forma más sencilla y rápida

3.2.1.1 Lector de código NC

En la Figura 3.4 se muestra un diagrama de flujo en el cual se muestran las comparaciones necesarias para quitar los caracteres innecesarios en un bloque NC (Figura 2.5). Si el carácter es un espacio ó una tabulación entra a un proceso de agregar espacio, este proceso compara si el buffer se encuentra vacio ó si el último carácter en el buffer es un carácter de fin de línea, si resultan verdaderas ambas condiciones el espacio no se agrega ya que estas comparaciones indican que el espacio se encuentra al inicio del bloque por lo que es un espacio innecesario, de lo contrario agrega un espacio al buffer y entra al proceso de avanzar este proceso avanza hasta encontrar un carácter distinto de espacio y tabulación, con el fin de evitar duplicar los espacios y tabulaciones. En caso de que el carácter sea '%' indica el inicio de un comentario por lo cual entra a el proceso de avanza que en esta ocasión realiza el avance hasta encontrar el fin de línea ó fin de archivo. Por último si el carácter que se encuentra en comparación indica el fin de línea ó fin de una instrucción (;),

entra al proceso de agregar fin de línea, en este proceso compara si el archivo se encuentra vacio o si el último carácter en el buffer es un fin de línea, de ser así no se agrega el fin de línea puesto que estas comparaciones indican que son líneas vacías en el programa, de lo contrario se agrega un fin de línea al buffer y se manda a compilar únicamente el bloque de comando donde se genera un buffer de errores. Si el carácter no entra a ninguna de las condiciones anteriores se agrega al buffer ya que este sería un carácter alfanumérico y no se encuentra dentro de los comentarios. Al terminar de analizar el buffer del programa se agrega el fin de archivo al buffer que fue llenado con los comandos considerando que el buffer de error siempre se debe de vaciar al iniciar la compilación, se compara si el buffer de error se encuentra vacio despliega el buffer de error, el cual contendrá la lista de errores que se detectaron indicando la línea y el error, en la sección del editor de lo contrario se carga el buffer en la sección del simulador (Figura 3.11).

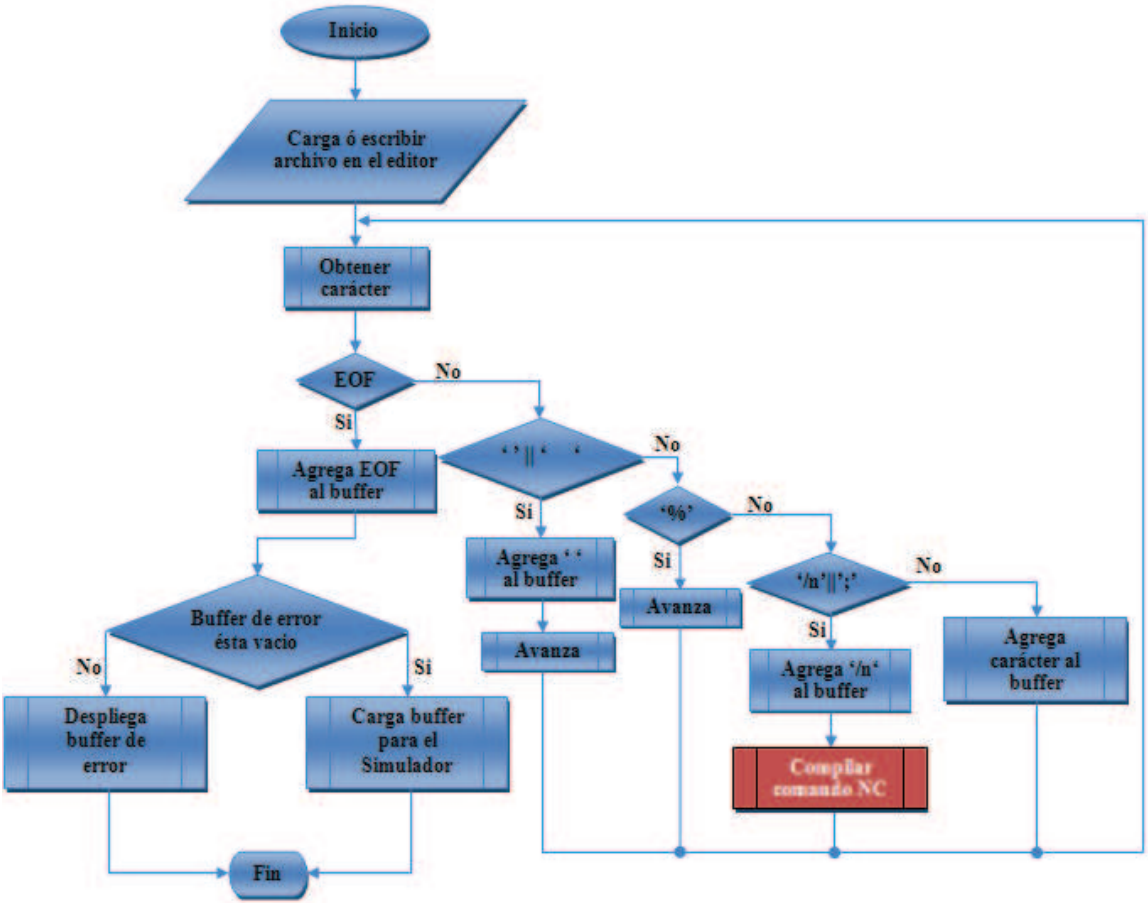

Figura 3.4 Diagrama de flujo del lector de código NC.

3.2.1.2 *Lector de código DMC*

Como se puede observar en la Figura 3.5, se realiza el mismo procedimiento del lector de código NC, las únicas diferencias es la forma de realizar los comentarios pues para este código los comentarios se identifican con un apostrofe (‘) y el proceso de compilación es diferente ya que estos comandos tienen una diferente forma de sintaxis.

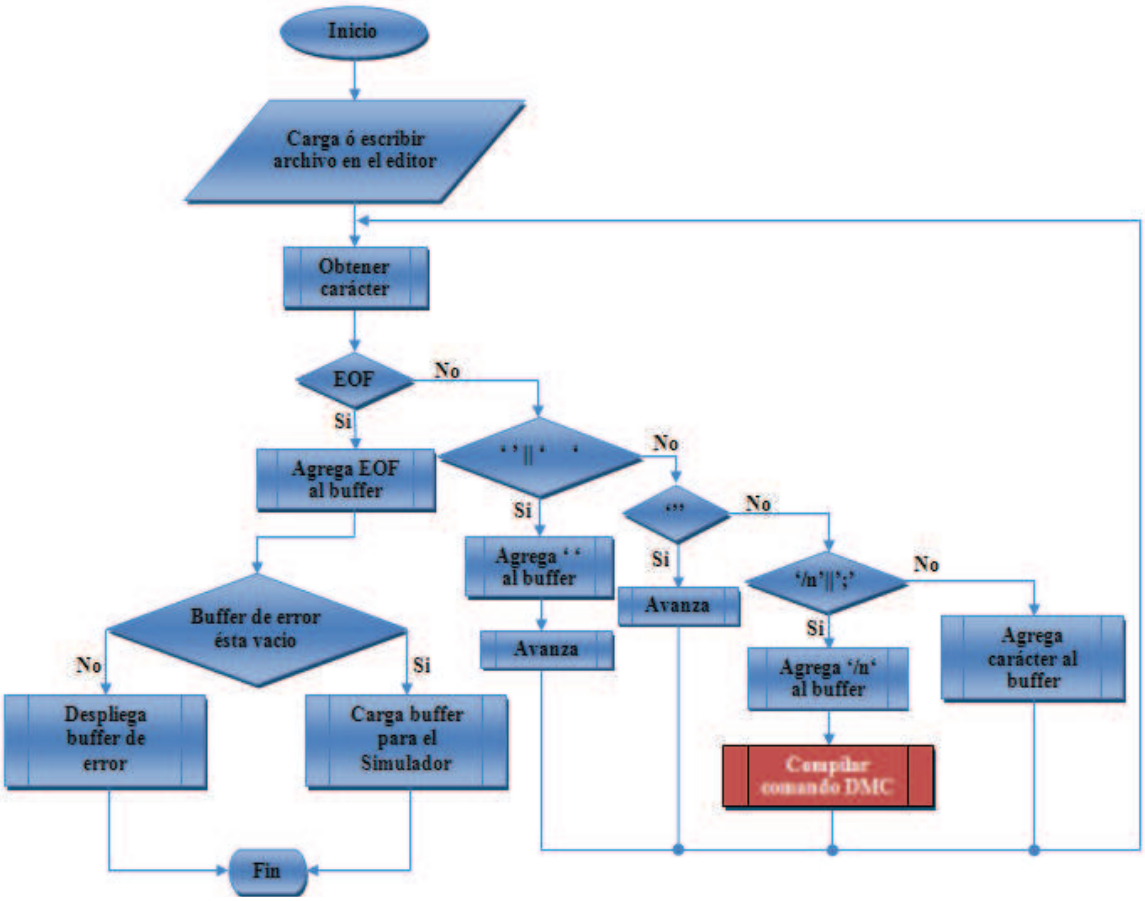

Figura 3.5 Diagrama de flujo del módulo del lector de código DMC.

3.2.2 Compilador

Es requerido un compilador tanto para los códigos NC y DMC ya que al generar manualmente un programa es normal que existan errores de sintaxis, esto podría ser muy perjudicial en la realización de un maquinado ya que la máquina-herramienta no realizaría los movimientos deseados. El compilador analiza comando por comandado como se muestra en los diagramas de flujo de la Figura 3.4 y Figura 3.5.

3.2.2.1 Compilación de comandos NC

En este trabajo se implementaron los comandos para interpolaciones lineales (G00 y G01), interpolaciones circulares (G02 y G03), interpolación con NURBS (G06.2), comandos para la selección de coordenadas absolutas (G90) ó relativas (G91), comandos para la selección de las unidades de trabajo milímetros (G20) ó pulgadas (G21) y por último la selección del plano de trabajo para la realización de las circunferencias X-Y (G17), X-Z (G18), Y-Z (G19). Los parámetros que requiere cada uno de los comandos se describen en el Capítulo II.

El diagrama de flujo principal de la compilación se muestra en la Figura 3.6 el cual comienza con el comando proporcionado por el lector, donde el primer paso es obtener el primer carácter, este podrá ser 'N' que indica el numero de bloque, 'G' que indica el tipo de bloque ó dirección, o en su defecto un fin de archivo. En caso que comience el bloque con 'N' se compila la serie de caracteres que se encuentre delante de dicho carácter, este proceso se indica con el nombre "Compilar valor N", estos caracteres deben de indicar un valor entero positivo y únicamente debe incluir caracteres numéricos de lo contrario se agregara un error en el buffer de error, ya terminada la compilación del valor de 'N' se avanza hasta el carácter que se encuentre delante de un espacio. Por lo general debe ser 'G' puesto que indica el tipo de comando que se va a utilizar sin embargo, en algunas ocasiones se indica un comando de interpolación para emplear con sus parámetros únicamente en el primer bloque para después solo escribir los parámetros de la misma interpolación en los siguientes bloques, es decir, sin especificar nuevamente el tipo de comando. De tal forma que si no se encuentra el carácter 'G' se busca la ultima interpolación indicada para saber que parámetros son los que se deben especificar encaso que no exista alguna interpolación previa se escribe en el buffer de error la falta del tipo de interpolación. Si se encuentra 'G' se obtiene el tipo de comando. Por último se realiza una serie de comparaciones para comprobar que el tipo de comando es se encuentra implementado en el software de lo contrario se indica en el buffer de error. Si el comando es aceptado se manda a un proceso de compilación dependiendo del tipo de comando ya que cada comando tiene diferentes parámetros.

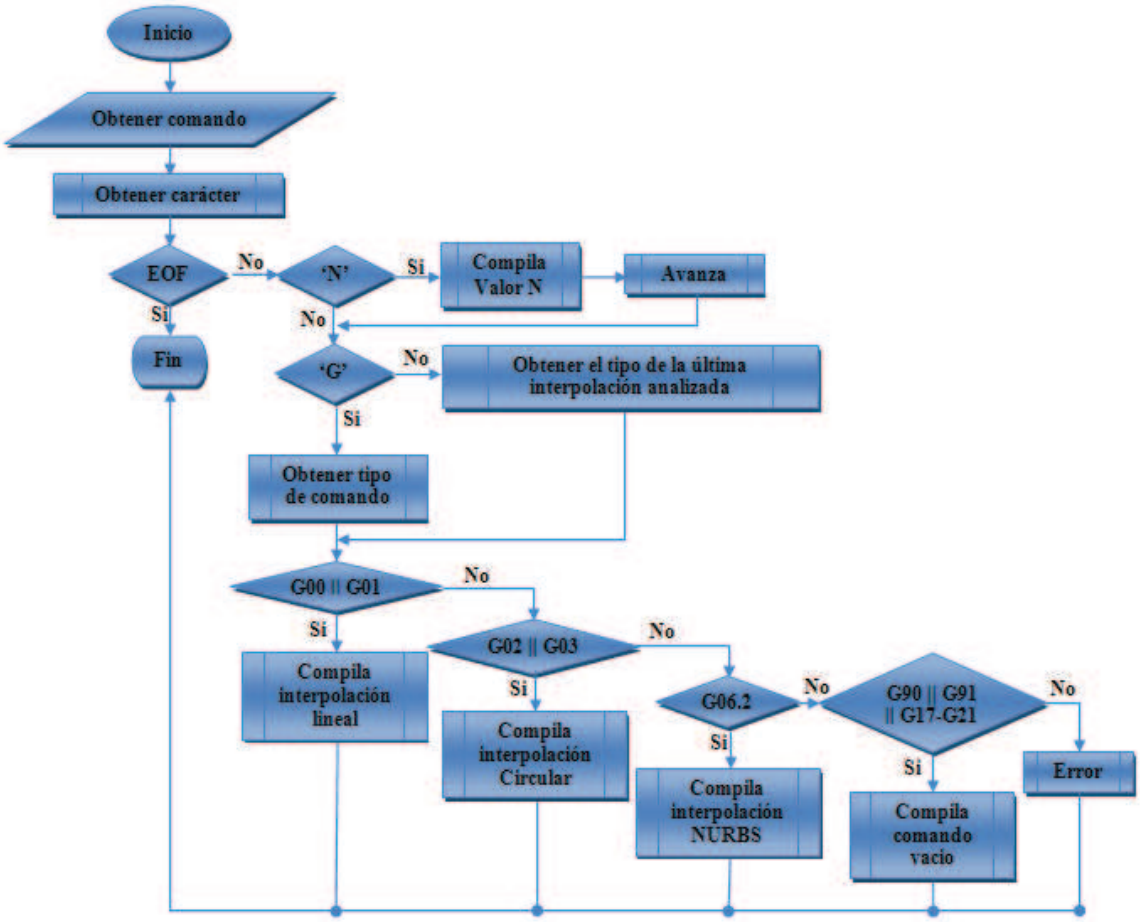

Figura 3.6 Diagrama de flujo principal del módulo de compilación de código NC.

El diagrama de flujo de la compilación para la interpolación lineal se muestra en la Figura 3.7. La compilación de una interpolación lineal se realiza recorriendo carácter por carácter el bloque del comando, de tal forma que se obtiene el primer carácter el cual podrá ser 'N' ó 'G', direcciones que ya fueron compiladas con anterioridad por lo tanto, únicamente se realiza un avance hasta un carácter adelante del siguiente espacio para obtener un nuevo carácter o fin de archivo. Los parámetros que contiene una interpolación lineal son velocidad de avance (*F*) y el valor de los ejes (*X*, *Y*, *Z* y *W*) de tal forma que se realizaron un procesos de compilación para los valores de la velocidad y otro para los valores de los ejes. La velocidad de avance debe ser indicada con un número positivo por lo que sí es un número negativo o si contiene un carácter diferente a los caracteres numéricos se indica en el buffer de error, de igual manera para los ejes sin embargo, los parámetros de los ejes si pueden tomar valores negativos.

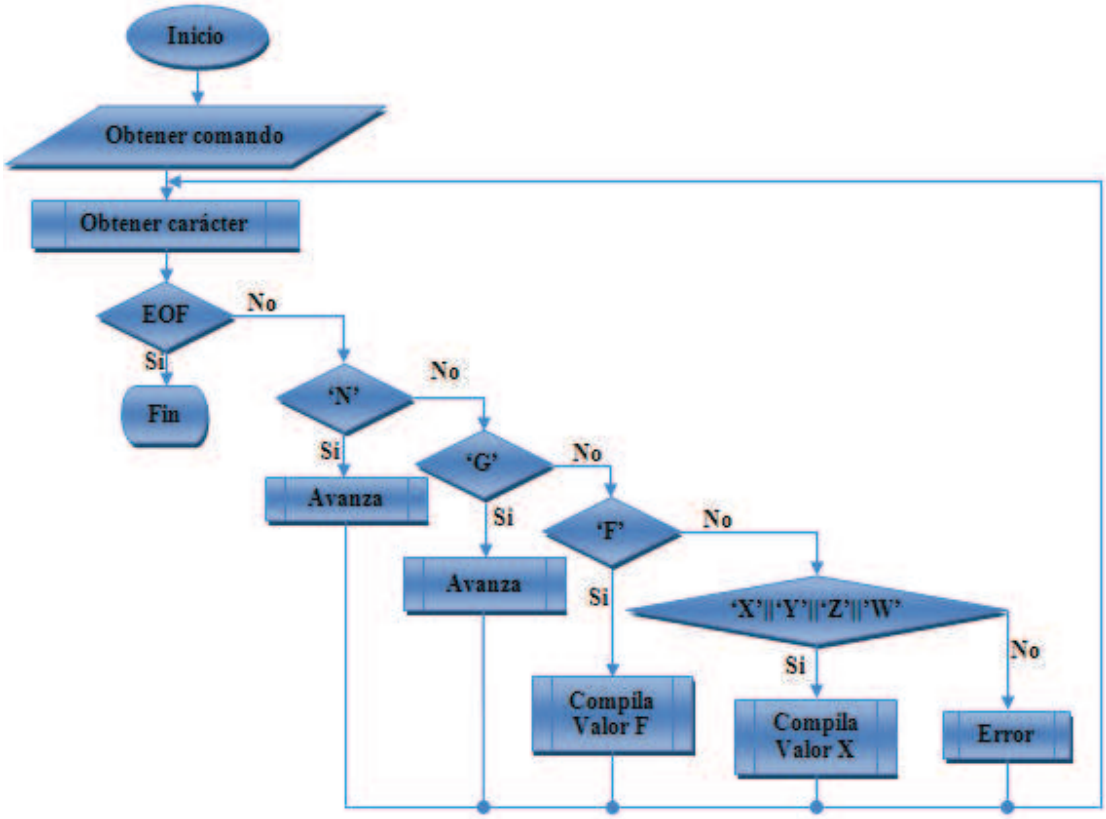

Figura 3.7 Diagrama de flujo para la compilación de las interpolaciones lineales en código NC.

Al igual que para la interpolación lineal se realizo una compilación dedicada a las interpolaciones circulares. El diagrama de flujo se muestra en la Figura 3.8, se realiza un procedimiento semejante al de la interpolación lineal la única diferencia es que el proceso de compilación "Compilar valor *X*" es utilizado para los parámetros *X*, *Y*, *Z*, *I*, *J* y *K*. El eje *W* no es posible especificarlo pues no se encontró algún comando NC que utilice un cuarto eje para las interpolaciones circulares.

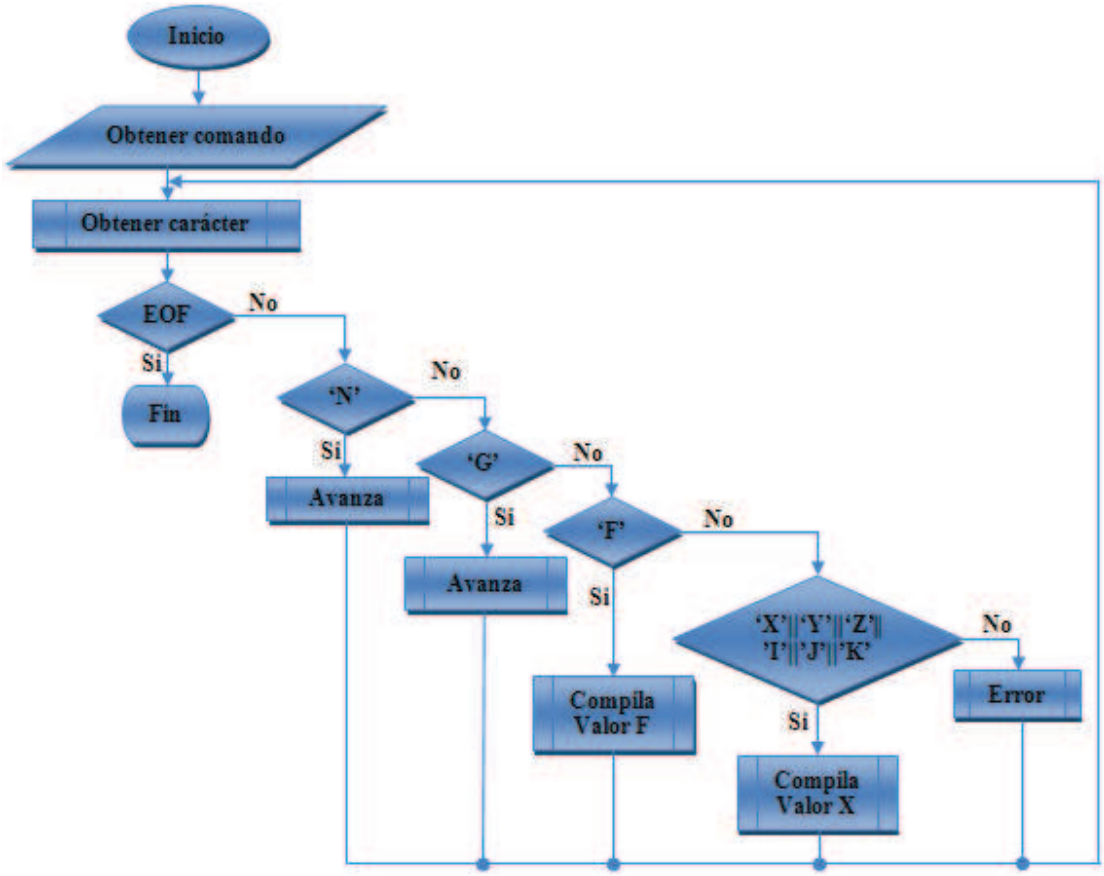

Figura 3.8 Diagrama de flujo para la compilación de las interpolaciones circulares en código NC.

La interpolación NURBS contiene como parámetros los ejes (*X*, *Y*, *Z* y *W*), la velocidad de avance (*F*), el peso para cada punto de control (*R*), el grado (*P*) y el valor para cada nodo (*K*). El diagrama de flujo de la Figura 3.9 muestra que se realizan una serie de compilaciones diferentes para cada parámetro. El valor de los nodos debe de ser un valor positivo y debe encontrarse entre el rango de 0 a 1. El valor del peso debe ser un valor entero positivo. Los valores de peso y de la velocidad de avance deben de ser positivos. Por último los valores de los ejes podrán tener cualquier valor. Encaso de que no cumpla algún parámetro con lo especificado anterior mente se agrega el dicho error al buffer de error.

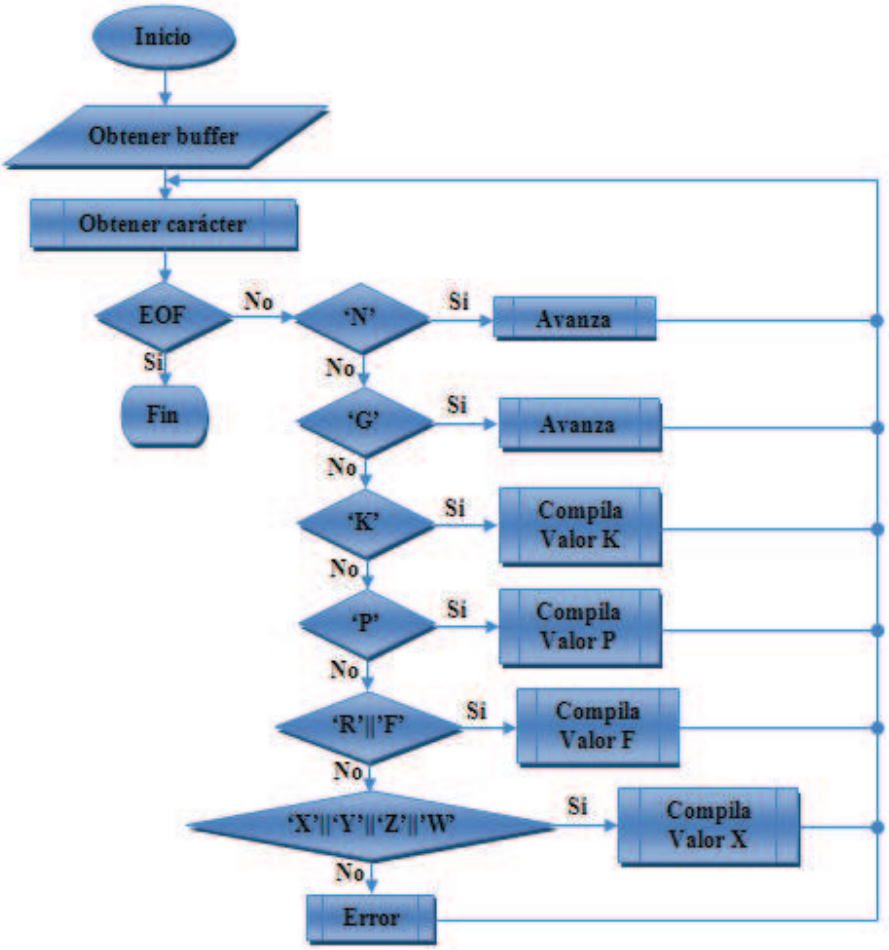

Figura 3.9 Diagrama de flujo para la compilación de las interpolaciones NURBS en código CN.

En caso de que el comando no sea ninguna de las interpolaciones ya mencionadas podrá ser un comando utilizado para especificar el tipo de coordenadas de trabajo, plano de trabajo o unidades de trabajo. Estos comandos no requieren ningún parámetro de tal forma que si se especifica algún parámetro se indicara el error al igual que en las interpolaciones.

3.2.2.2 Compilación de comandos DMC

En este trabajo se implementaron comandos para posición independiente de los ejes (PR, PA, SP, AC y DC), movimientos coordinados para 2, 3 o 4 ejes (LM, LI, LE, VS, VA y VD), movimiento coordinado en 2-D (VM, VP, CR, VE, VS, VA y VD), comandos para iniciar movimientos (BG y BGS) y comandos de espera (AM y AMS). Se muestra una breve descripción de cada uno de los comandos en la Tabla 2.2.

La compilación es realizada comando por comando, es decir, recibe un bloque de instrucción del lector de código DMC el cual contiene el comando a compilar. La primera parte de la compilación es identificar el tipo de comando el cual es indicado con los primeros caracteres, esto es obtener los caracteres del bloque hasta encontrar un espacio, una coma ó un final de archivo. Una vez obtenido el comando se realiza el proceso mostrado en el diagrama de flujo de la Figura 3.10 donde se realiza una serie de comparaciones para verificar si el comando es alguno de los comandos implementados y dependiendo del tipo de comando entra a una compilación de parámetros, en caso de no sea un comando que no esté implementado se añade un error al buffer de error para ser desplegado en la pantalla y pueda ser visto por el usuario.

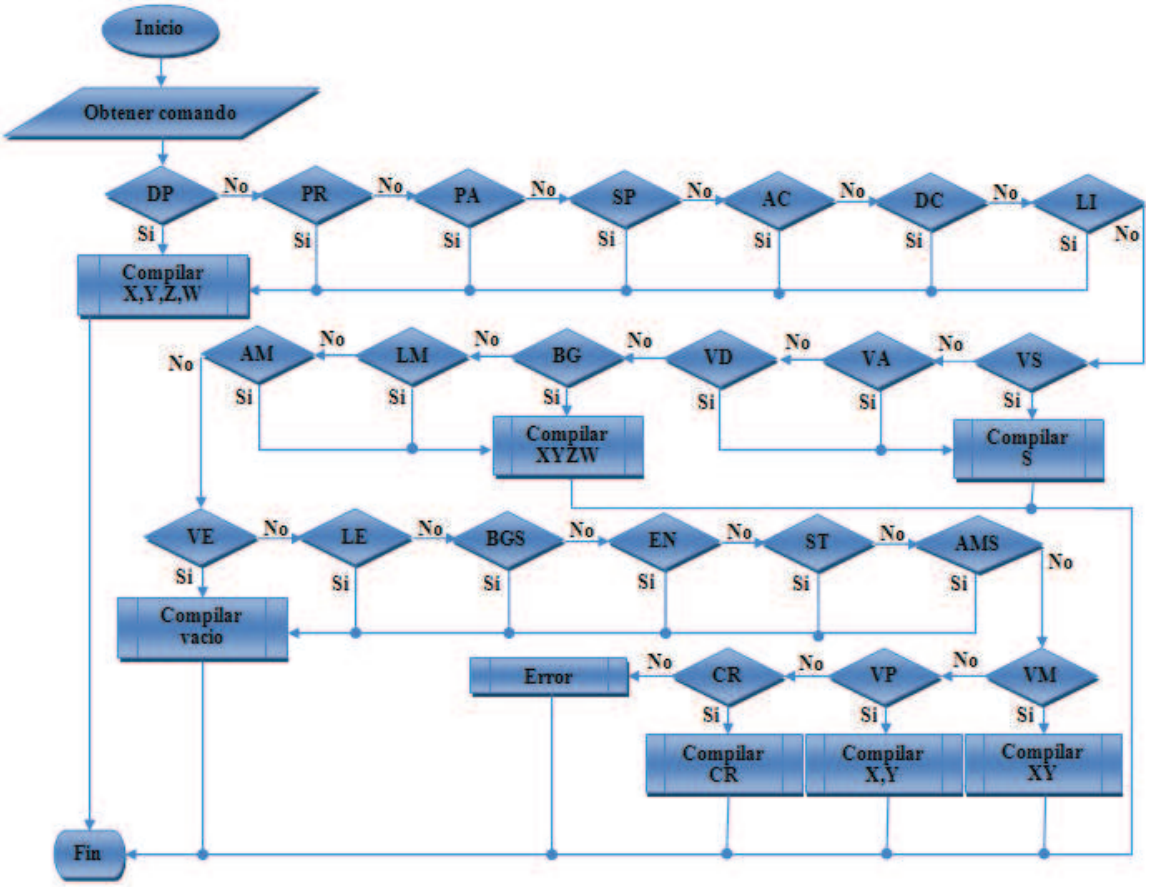

Figura 3.10 Diagrama de flujo principal del módulo de compilación de código DMC.

En la Tabla 3.1 se muestran los comandos que entran al proceso de “Compilar X, Y, Z, W” en el cual debe contener de uno a cuatro para metros, en caso de que no contenga

ningún parámetro ó que contenga más de los que se pueden especificar carga un error en el buffer de error, para estos comandos los parámetros son separados por una coma ó un espacio. El comando LI debe de contener antes el comando LM ya que este indica el inicio de una secuencia vectorial y además debe tener el comando LE que indique el final de la secuencia.

Tabla 3.1 Comandos para Compilar X, Y, Z, W.

COMANDO	PARÁMETROS	DESCRIPCIÓN
DP	X, Y, Z,W	Asigna valores a los ejes
PR	X, Y, Z,W	Movimiento especificando las coordenadas relativas
PA	X, Y, Z, W	Movimiento especificando las coordenadas absolutas
SP	X, Y, Z, W	Especifica la velocidad para cada eje
AC	X, Y, Z, W	Especifica la aceleración para cada eje
DC	X, Y, Z, W	Especifica la desaceleración para cada eje
LI	X, Y, Z, W	Especifica valores absolutos de los ejes

Los comandos que entran en el proceso de "Compilar S" son mostrados en la Tabla 3.2, estos comandos deben tener únicamente un parámetro para los comandos VS y VA el valor debe ser positivo mientras que para VD debe ser un valor negativo.

Tabla 3.2 Descripción para Compilar S.

COMANDO	PARÁMETROS	DESCRIPCIÓN
VS	S	Especifica la velocidad total
VA	S	Especifica la aceleración total
VD	S	Especifica la desaceleración total

Los comandos que requieren el proceso de "Compilar XYZW" se muestran en la Tabla 3.3 donde se especifican los ejes a utilizar. El comando BG requiere que exista un movimiento para realizarse, en caso de que exista un comando BG ó BGS anterior debe existir un AM ó AMS enseguida del comando BG ó BGS esto para que la simulación que se genera sea el movimiento que se realizara.

Tabla 3.3 Descripción para Compilar XYZW.

COMANDO	PARÁMETROS	DESCRIPCIÓN
BG	XYZW	Comienza el movimiento de los ejes especificados
LM	XYZW	Inicio de secuencia lineal
AM	XYZW	Espera a que termine el movimiento de los ejes especificados

El proceso de “Compilar vacio” es empleado para los comandos mostrados en la Tabla 3.4 estos comandos no requieren ningún parámetro. Los comandos VE y LE requieren sus respectivos comandos para iniciar la secuencia en caso que no haya sido especificada se carga el error. El comando BGS requiere las mismas condiciones que el comando BG. El programa siempre requiere el fin de programa (EN) en caso de no tenerlo se carga un error.

Tabla 3.4 Descripción para Compilar vacio.

COMANDO	PARÁMETROS	DESCRIPCIÓN
ST		Paro del movimiento
VE		Final de secuencia (VM)
LE		Final de secuencia (LM)
BGS		Iniciar secuencia
AMS		Espera a que termine la secuencia
EN		Final de programa

Los comandos para movimientos coordinados se muestran en la Tabla 3.5 donde VP indica una interpolación lineal y CR una interpolación circular únicamente en dos ejes. Los parámetros que se deben de especificar en el comando VP deben coincidir con los especificados en el comando VM, la circunferencia que se realiza con CR es en el plano especificado con VM. La secuencia de movimientos coordinados debe tener también un final de secuencia que es especificado con VE.

Tabla 3.5 Comandos para movimientos coordinados.

COMANDO	PARÁMETROS	DESCRIPCIÓN
VM	XY	Inicio movimientos coordinados
VP	X,Y	Vector lineal
CR	R, θ, Δθ	Circunferencia
VE		Final de secuencia (VM)

3.3 Simulador

La sección de simulación se encuentra bloqueada hasta que sea compilado un programa en la sección del editor, dicho programa deberá estar sin errores y se cargara automáticamente el buffer que contiene únicamente los comandos ya sean de código NC ó DMC, es decir, se carga el programa sin comentarios y sin espacios ó tabulaciones dobles. Este trabajo se encuentra enfocado principalmente en esta sección de tal forma que se

realiza la interpretación de los comandos para ser enviados y ejecutados al sistema de control de posicionamiento basado en FPGA. Además es posible graficar en tiempo de ejecución la posición obtenida del sistema, la referencia que se da y los datos obtenidos por el software, es decir los datos calculados para la simulación. En la Figura 3.11 se muestra la pestaña del simulador desarrollada para la interfaz gráfica de usuario donde al igual que en el editor se despliega tanto la referencia que se da como la posición censada.

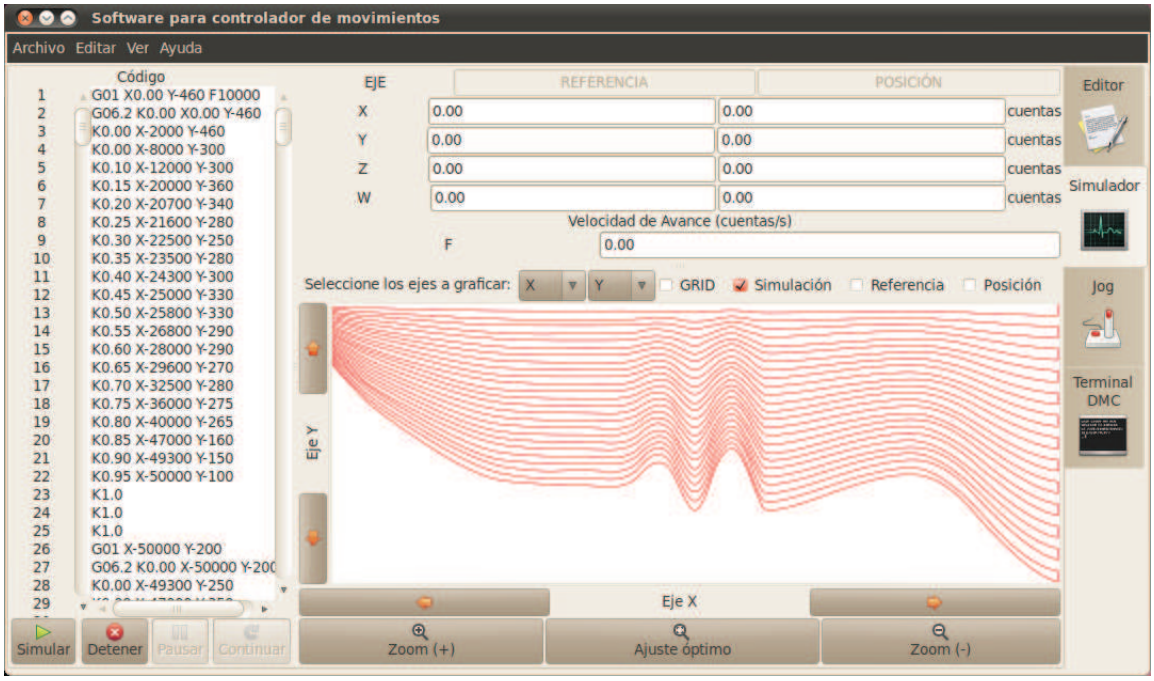

Figura 3.11 Pestaña del simulador de la GUI.

Cabe mencionar que el sistema de control de posicionamiento basado en FPGA se encuentra desarrollado para interpretar curvas NURBS. Las curvas se almacenan en una base de datos que contiene una lista de los puntos de control, una lista de los nodos y valores para la velocidad de avance, aceleración y desaceleración, estos últimos tres son necesarios para la construcción de los perfiles. Los puntos de control son conformados por los valores de los ejes (*X*, *Y*, *Z* y *W)* especificados en cuentas y el peso que tiene el punto de control (*P*). Los valores de los nodos (*K*) deben encontrarse en el intervalo de 0 a 1. La velocidad de avance debe ser especificada en cuentas/segundo, la aceleración y la desaceleración en cuentas^2/segundos.

3.3.1 Adquisición de parámetros

Antes de realizar la interpretación de los comandos es necesario adquirir los parámetros de cada uno de los comandos para la construcción de las curvas. Dependiendo del tipo de código se realiza una base de datos provisional donde contenga los parámetros que ofrece cada uno de los comandos y a partir de los parámetros almacenados se realizara la generación de las curvas.

3.3.1.1 Adquisición de parámetros de los códigos NC

Los programas con código NC al igual que los códigos DMC se realizan de manera secuencial de tal manera que se generan banderas para identificar el tipo de coordenadas en las que se trabaja (*C*), el plano de trabajo (*P*) y las unidades de trabajo (*U*). Las coordenadas por default son absolutas, el plano por default es *XY* y las unidades por default para este trabajo se tomaron las cuentas, sin embargo no existe comando para especificar que se trabajara bajo estas unidades.

En la Figura 3.12 se muestra el diagrama de flujo principal para la obtención de los parámetros necesarios para construir una curva paramétrica a partir de los comandos NC. Este proceso se realiza comando por comando, obteniendo los comandos del buffer cargado en el simulador (Figura 3.11). El primer proceso mostrado en el diagrama de flujo es para obtener el primer carácter, si el primer carácter es 'N' el proceso avanza hasta encontrar el siguiente carácter alfabético puesto que el código ya fue previamente compilado, el carácter indicara un comando ó un parámetro de algún comando NC. En caso de que sea algún parámetro, el tipo de comando conserva el valor del último comando de interpolación, si encuentra el carácter 'G' se obtiene el tipo de comando. Obtenido el tipo de comando se realiza una serie de comparaciones para identificar el tipo de comando. Las interpolaciones requieren un almacenamiento de sus parámetros de tal motivo que se realizo un almacenamiento para cada una de las interpolaciones. Los demás comandos activan banderas para identificar las condiciones de trabajo.

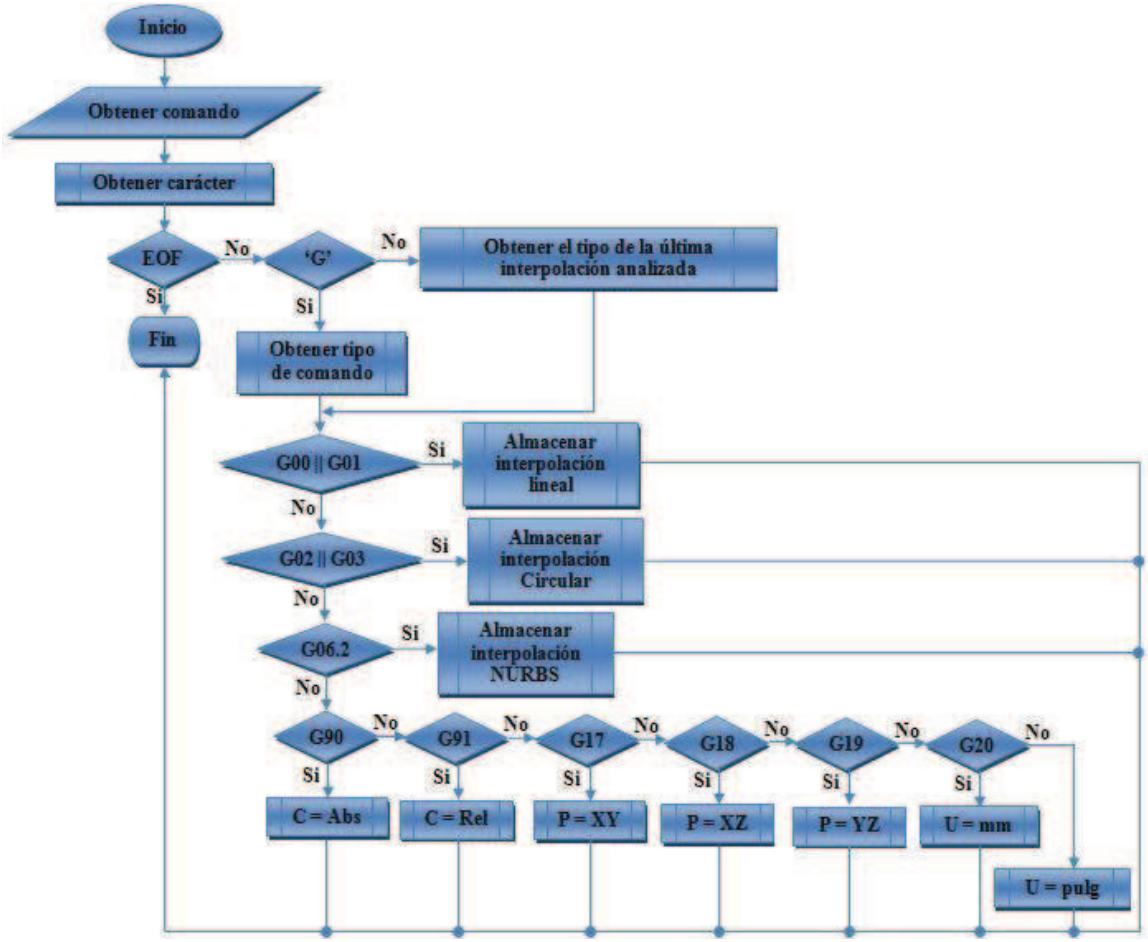

Figura 3.12 Diagrama de flujo principal del módulo de adquisición de parámetros de comandos CN.

Una interpolación lineal consta de los parámetros *X*, *Y*, *Z* y *W*, para especificar los valores de los ejes y *F* para la velocidad de avance por lo cual en un comando de interpolación lineal se buscan la especificación de estos parámetros como se muestra en la Figura 3.13, en caso de que sean especificados se obtiene el valor de lo contrario se mantiene el valor anterior de la lista. Para cada valor de parámetro especificado se realiza la conversión a cuentas y en el caso de que la especificación sea para alguno de los ejes también se realiza la conversión a valores absolutos, esto es requerido ya que el sistema de control de posicionamiento basado en FPGA requiere que los puntos de control estén especificados en cuentas y coordenadas absolutas.

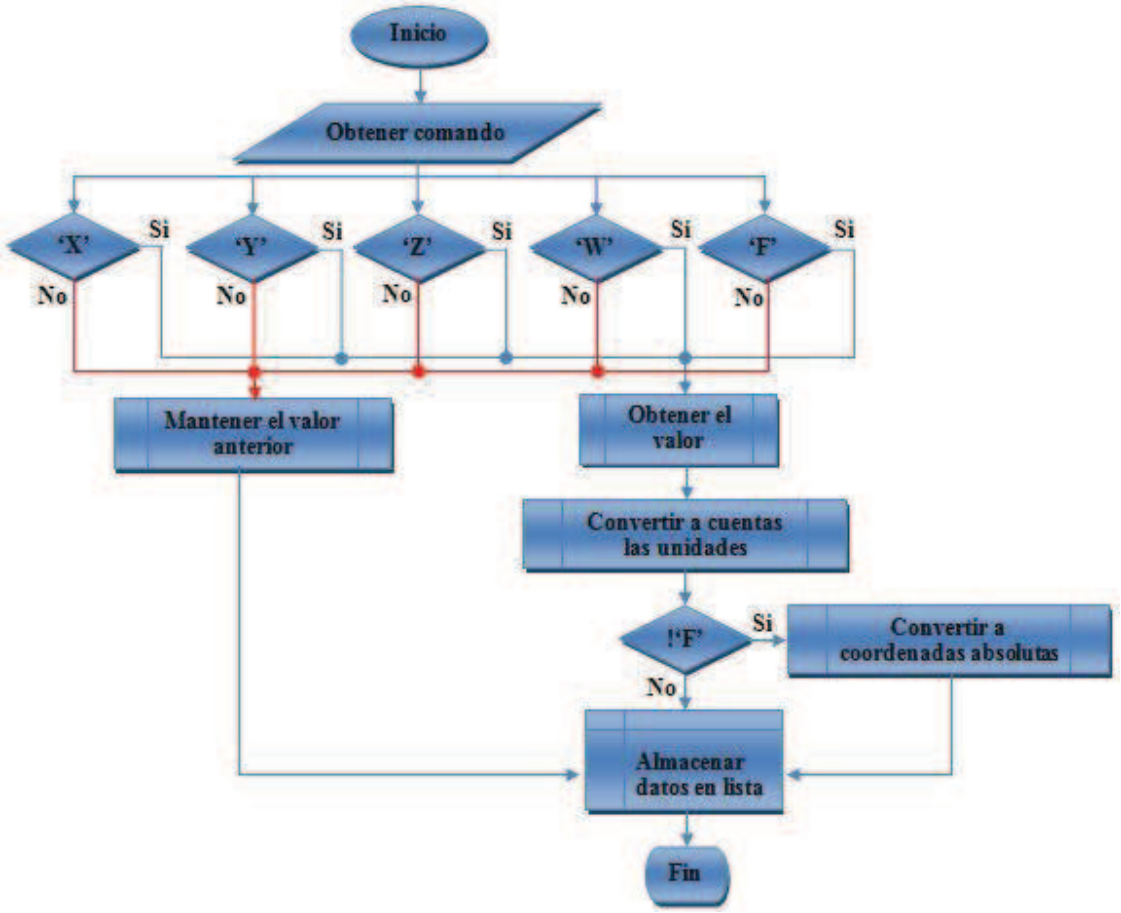

Figura 3.13 Diagrama de flujo para el almacenamiento de parámetros de las interpolaciones lineales en código CN.

La interpolación circular además de los parámetros de una interpolación lineal requiere los parámetros *I*, *J* y *K* los cuales especifican el centro de la circunferencia. Estos parámetros siempre son dados en coordenadas relativas y como se muestra en la Figura 3.14 estos valores son guardados en forma de coordenadas absolutas, es decir, el valor asignado en *I* más el valor de *X* anterior, el valor asignado en *J* más el valor de *Y* anterior y el valor asignado en *K* más el valor de *Z* anterior.

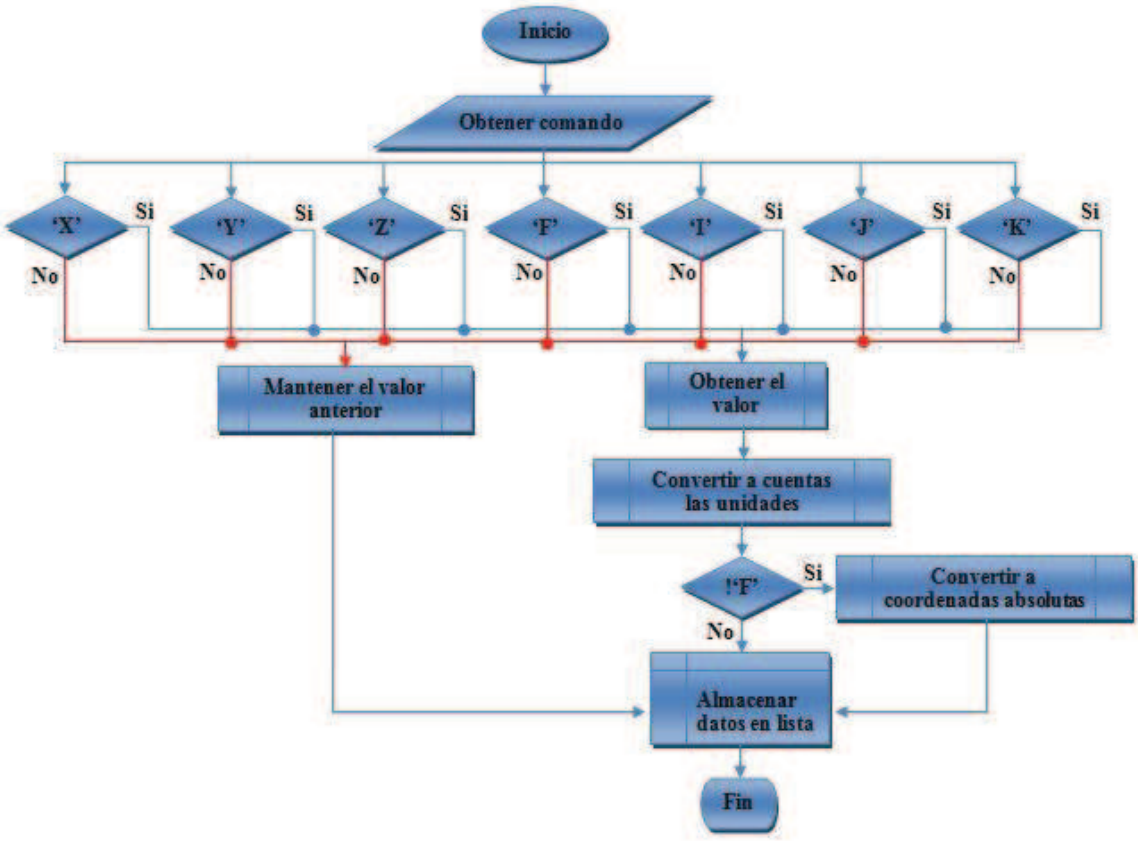

Figura 3.14 Diagrama de flujo para el almacenamiento de parámetros de las interpolaciones NURBS de los comandos CN.

En las interpolaciones NURBS puede especificar los valores de los ejes, la velocidad de avance, el peso, el valor del nodo y el grado. Sin embargo el grado no es requerido ya que el grado puede ser calculado restando el número de puntos de control al número de nodos menos uno. En la Figura 3.15 se muestra el diagrama de flujo para el almacenamiento de los parámetros de un comando NURBS. Al igual que en las interpolaciones anteriores se obtienen los parámetros que fueron asignados. En caso de que sea el peso ó el valor de nodo solo se guarda el dato en la base de datos. Si es la velocidad de avance se realiza la conversión a cuentas. Si la especificación es de un eje además de realizar la conversión a cuentas convierte la coordenada a coordenadas absolutas.

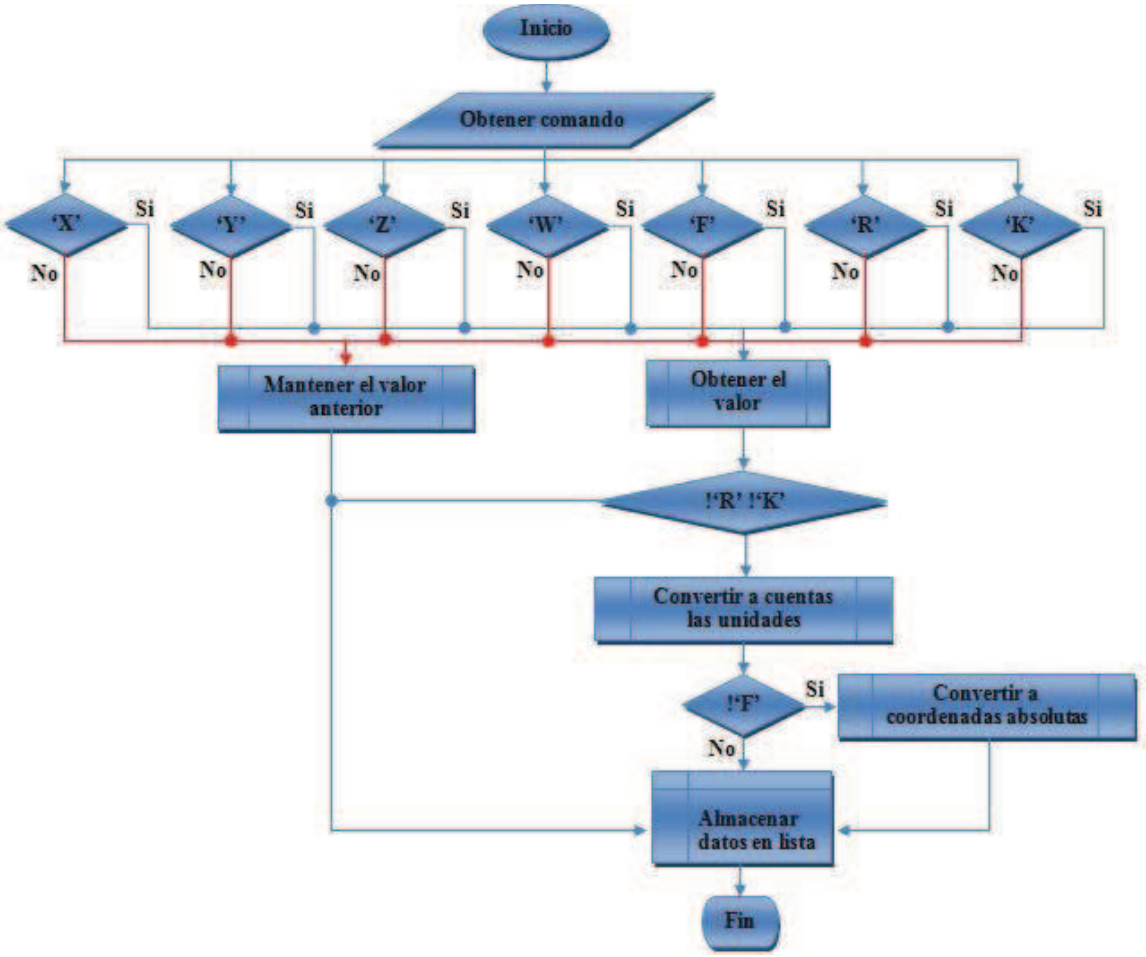

Figura 3.15 Diagrama de flujo para el almacenamiento de parámetros de las interpolaciones circulares de los comandos CN.

Para ejemplificar con mayor claridad el proceso de la adquisición de datos se utiliza el Listado 3.1 donde se muestra un programa NC después de haber pasado el proceso de compilación por de tal forma que no contiene comentarios y sin errores.

Listado 3.1 Ejemplo de código NC.

```
N01 G00 X0 Y0 Z0 W0 F200
N02 G01 Y10
N03 G02 X10 Y0 I0 J-10
N04 G00 X8 Y12
N05 G06.2 K0.0 X8.0 Y12.0 Z0.0
N06 K0.0 X5.0 Y8.0;
N07 K0.0 X0.0;
N08 K0.111 X4.0 Y4.0;
N09 K0.222 X3.0 Y0.0;
N10 K0.333 X8.0 Y3.0;
N11 K0.444 X13.0 Y0.0;
N12 K0.555 X12.0 Y4.0;
N13 K0.666 X16.0 Y8.0;
N14 K0.777 X11.0;
```

```
N15 K0.888 X8.0 Y12.0;
N16 K1.0
N17 K1.0
N18 K1.0
```

Cada línea de comando que indique algún tipo de interpolación ya sea lineal, circular ó NURBS Listado 3.1 genera una línea de almacenamiento en una lista enlazada, la lista generada con el listado anterior se muestra en la Tabla 2.1 donde se puede observar que para las interpolaciones lineal se guarda los valores de los ejes (*X*, *Y*, *Z* y *W*) y la velocidad de avance (*F*), para las interpolaciones circulares además de los parámetros de las interpolaciones lineales requiere los valores de *I*, *J* y *K* que serán utilizados para obtener el centro de la circunferencia y el plano en el que se lleva a cabo la circunferencia. Por último para las interpolaciones NURBS las cuales se encuentran compuesta por una serie de puntos de control que son indicados cada punto de control con una línea de comando por lo tanto para cada punto de control se almacenan las coordenadas, el valor de la velocidad, el valor del nodo y el valor del peso (*R*). Cabe mencionar que si el plano no es especificado toma el valor 17 que corresponde al plano *XY*, si no es especificado el peso toma un valor por default de uno y que el valor de la velocidad de avance se mantiene hasta ser modificada en alguna línea de comando.

Tabla 3.6 Lista de almacenamiento de parámetros de los comandos G.

G	X	Y	Z	W	F	I	J	K	R	Nodo	Plano
0	0	0	0	0	200	0	0	0	1	0	XY
1	0	10	0	0	200	0	0	0	1	0	XY
2	10	0	0	0	200	0	-10	0	1	0	XY
0	8	12	0	0	200	0	-10	0	1	0	XY
6.2	8	12	0	0	200	0	-10	0	1	0	XY
6.2	5	8	0	0	200	0	-10	0	1	0	XY
6.2	0	8	0	0	200	0	-10	0	1	0	XY
6.2	4	4	0	0	200	0	-10	0	1	0.111	XY
6.2	3	0	0	0	200	0	-10	0	1	0.222	XY
6.2	8	3	0	0	200	0	-10	0	1	0.333	XY
6.2	13	0	0	0	200	0	-10	0	1	0.444	XY
6.2	12	4	0	0	200	0	-10	0	1	0.555	XY
6.2	16	8	0	0	200	0	-10	0	1	0.666	XY
6.2	11	8	0	0	200	0	-10	0	1	0.777	XY
6.2	8	12	0	0	200	0	-10	0	1	0.888	XY
6.2	8	12	0	0	200	0	-10	0	1	1	XY
6.2	8	12	0	0	200	0	-10	0	1	1	XY
6.2	8	12	0	0	200	0	-10	0	1	1	XY

3.3.1.2 Adquisición de parámetros de los códigos DMC

La compilación del código da la certeza de que no existen errores por lo que en la parte de adquisición únicamente nos enfocamos a la adquisición para el almacenamiento de los datos. El diagrama principal para el almacenamiento de los datos se muestra en la Figura 1.2 donde se observan los comandos que ofrecen información necesaria para la realización del movimiento.

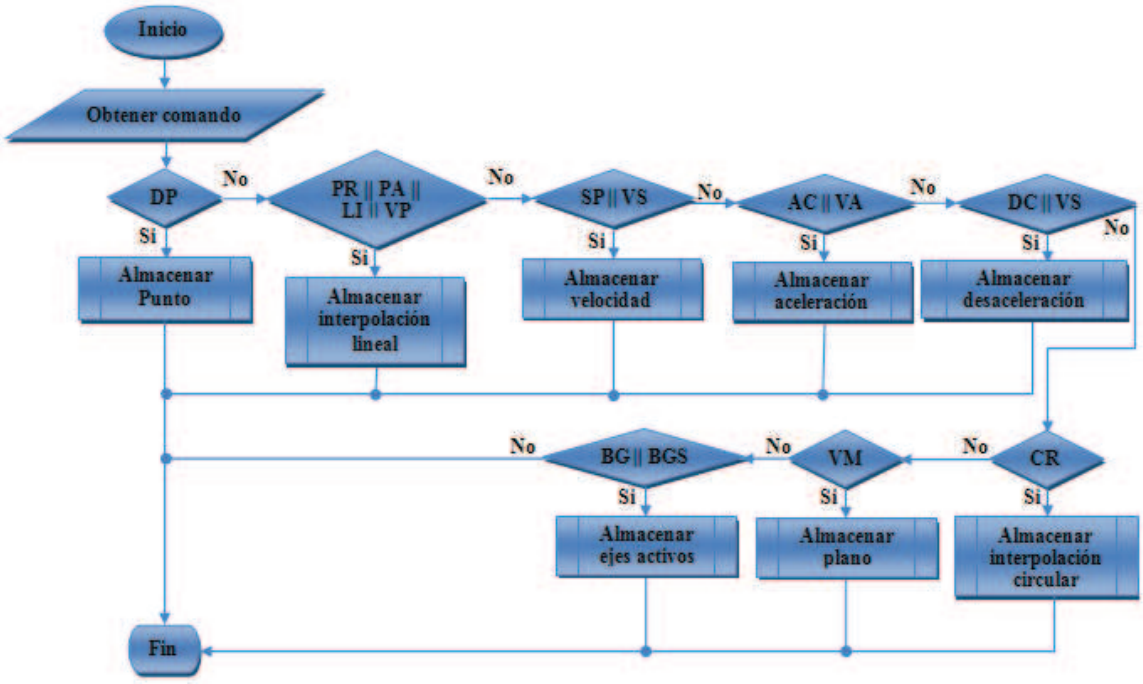

Figura 3.16 Diagrama de flujo principal para el almacenamiento de datos del código DMC.

El proceso de "Almacenar punto" únicamente almacena las coordenadas obtenidas del comando DP. El proceso de "Almacenar interpolación lineal" convierte las coordenadas a coordenadas absolutas si es necesario y las almacena, los procesos de "Almacenar velocidad", "Almacenar aceleración" y "Almacenar desaceleración" pueden especificar sus respectivos parámetros para cada eje o para todo el movimiento de tal forma que es necesario almacenar parámetros de velocidad, aceleración y desaceleración para cada uno de los ejes y la velocidad total. El comando CR proporciona tres parámetros para la construcción de la interpolación circular los cuales son almacenados en el proceso de "Almacenar interpolación circular". El comando VM actualiza una bandera en la cual se almacena el plano de trabajo dependiendo los ejes especificados en el comando esto es

realizado en el proceso de “Almacenar plano”. Por último el comando BG se especifican los ejes de los cuales se desea realizar el movimiento por lo cual activa los comandos que se desean mover y BGS activa todos los ejes esto se realiza en el proceso de “Almacenar ejes activos”.

Para ejemplificar lo que realizan los procesos descritos anteriormente se presenta el Listado 3.2 donde se muestra un programa con código DMC donde se especifican interpolaciones lineales y circulares además de los valores de velocidad, aceleración y desaceleración.

Listado 3.2 Ejemplo de código DMC.

```
DP 0,0
VM XY
VP -4000,0
CR 1500,270,-180
VS 2000
VA 10000
VD 10000
VP 0,3000
CR 1500,90,-180
VE
BGS
EN
```

En la Tabla 3.7 se muestra la forma de almacenamiento de los parámetros necesarios para la interpretación de los comandos. En la Tabla 3.7 DMC indica el tipo de comando; *X* y *Y* indican los valores de las coordenadas ya que se muestra un ejemplo únicamente dos ejes sin embargo el trabajo se encuentra desarrollado para 4 ejes; *Xs* y *Ys* indican la velocidad para cada eje en caso de especificar con SP; *Ts* indica la velocidad de una secuencia esto se indica cuando se especifica la velocidad con VS; *Xa* y *Ya* indican la aceleración para cada eje si es especificado con AC; Ta indica la aceleración de una secuencia indicada con VA; *Xd* y *Yd* indican la desaceleración cuando se indica con DC; Td indica la desaceleración cuando se utiliza VD; *x* y *y* indican la activación de los ejes cero indica desactivo y uno activo estos son utilizados con los comandos BG y BGS; R, θ y $\Delta\theta$ son los parámetros dados por el comando CR para interpolaciones circulares.

A continuación se realiza una breve descripción del proceso de almacenamiento que se realiza en cada uno de los comandos del Listado 3.2. En el comando DP estable una posición por lo cual se requiere guardar las coordenadas; VM especifica el plano de trabajo por lo que guarda el plano de trabajo para ponerlo en las interpolaciones lineales y circulares que se realicen; VP indica una interpolación lineal por lo que se guardan las

coordenadas; CR indica una interpolación circular por lo que se guarda los parámetros R, θ y $\Delta\theta$, además que se mantiene las coordenadas del punto de inicio; VS establece la velocidad para los siguientes comandos y actualiza la velocidad para todos los comandos almacenados anteriormente hasta llegar al primer comando ó hasta encontrar un BG; VA y VD establecen la aceleración y desaceleración respectivamente para los comandos siguientes y al igual que VS actualizan sus respectivos parámetros para los comandos almacenados anteriormente; se almacena una interpolación lineal (VP); se almacena una interpolación circular (CR); VE indica el final de la secuencia; BGS activa todos los ejes y finalmente EN indica el final del programa.

Tabla 3.7 Lista de almacenamiento de parámetros de los comandos DMC.

DMC	X	Y	Xs	Ys	Ts	Xa	Ya	Ta	Xd	Yd	Td	x	y	R	θ	Δθ	Plano
DP	0	0	0	0	2000	0	0	10000	0	0	10000	0	0	0	0	0	XY
VP	-4000	0	0	0	2000	0	0	10000	0	0	10000	0	0	0	0	0	XY
CR	-4000	0	0	0	2000	0	0	10000	0	0	10000	0	0	1500	270	-180	XY
VP	0	3000	0	0	2000	0	0	10000	0	0	10000	0	0	0	0	0	XY
CR	0	3000	0	0	2000	0	0	10000	0	0	10000	0	0	1500	90	-180	XY
BGS	0	3000	0	0	2000	0	0	10000	0	0	10000	1	1	0	0	0	XY

3.3.2 Generación de NURBS

El controlador de movimiento basado en FPGA recibe la información de los movimientos en forma de curvas NURBS de tal manera que es necesario convertir las interpolaciones lineales y circulares en forma de curvas NURBS.

Las curvas NURBS están compuestas por los puntos de control (N_p), el peso para cada punto de control (W_p) y una serie de nodos. El numero de nodos (N_k) que se requiere para cada curva depende del grado (g) que se desea dicha curva, la ecuación (3.1) se muestra como obtener el grado de la curva a partir de del número de puntos de control que y del numero de nodos que se tengan, cabe mencionar que siempre deben de existir más nodos que puntos de control.

$$g = N_k - N_p - 1 \tag{3.1}$$

3.3.2.1 Curvas NURBS a partir de una interpolación lineal

Una interpolación lineal está compuesta por dos puntos como se muestra en Figura 3.17 donde P_0 es el punto donde inicia la interpolación y P_1 es el punto final de la

interpolación. Una interpolación lineal como en los comandos DMC y G sus principales parámetros son las coordenadas del punto que al que se desea llegar teniendo como punto inicial el punto anterior ó el punto en el que se encuentra posicionada la máquina.

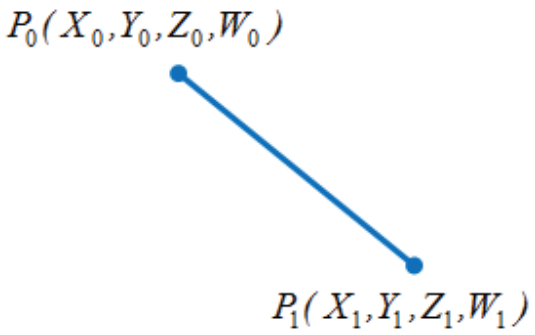

Figura 3.17 Interpolación lineal.

La conversión de una interpolación lineal a NURBS es muy sencillo (Tabla 3.8) ya que se tienen los puntos de control (P_0 y P_1) y de acuerdo a la ecuación (3.1) se requieren cuatro nodos, ya que el grado de la curva es uno, con valores de 0, 0, 1 y 1; el peso para cada punto de control es de uno para que comience en el punto inicial y termine en el punto final.

Tabla 3.8 Curva NURBS para una interpolación lineal.

Nodos	Puntos de control	Pesos
0	X_0, Y_0, Z_0, W_0	**1**
0	X_1, Y_1, Z_1, W_1	**1**
1	-	-
1	-	-

3.3.2.2 *Curvas NURBS a partir de una interpolación circular*

Para dibujar una sección canoníca se puede utilizar curvas NURBS de tal manera que una curva con tres puntos de control (P_0, P_1 y P_2) de pesos 1, w y 1, respectivamente y como se requiere una curva de segundo grado son necesarios seis nodos como se muestra en la Tabla 3.9.

Tabla 3.9 Curva NURBS para una sección canoníca.

Nodos	Puntos de control	Pesos
0	X_0, Y_0, Z_0, W_0	**1**
0	X_1, Y_1, Z_1, W_1	***w***

0	X_2, Y_2, Z_2, W_2	1
1	-	-
1	-	-
1	-	-

En la Figura 3.18 se muestran cuatro curvas NURBS con los mismos puntos de control y nodos pero con un peso *w* en el punto P_1. Cuando *w* es mayor a uno se genera una curva hiperbólica, si *w* es igual a uno una curva parabólica y cuando *w* es menor a uno se forma una curva elíptica. Esto sucede siempre y cuando P_1 se encuentre en el punto de la intersección de las dos rectas perpendiculares a los puntos P_0 y P_2.

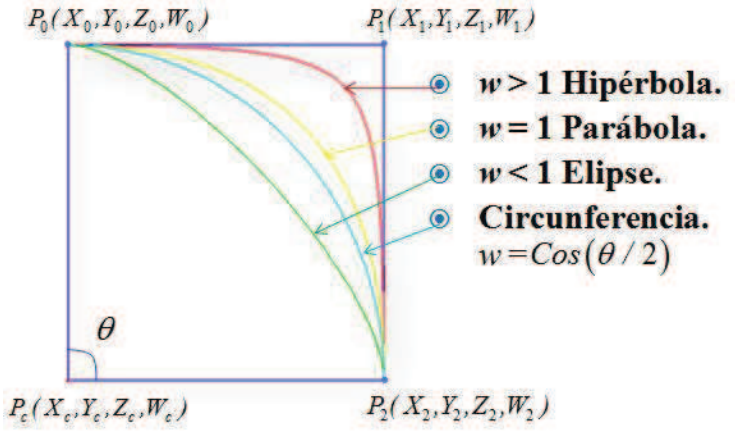

Figura 3.18 Curvas NURBS de tres puntos con diferentes pesos.

Según la Figura 3.18 para trazar un arco circular (elipse), el módulo de $P_0 P_1$ tiene que ser igual al módulo de $P_1 P_2$, el peso del punto P_1 tiene que coincidir con el seno de $\theta/2$ siendo θ el ángulo formado por los vectores $P_0 P_1$ y $P_1 P_2$. Esta manera de trazar un arco circular no funciona cuando θ es igual a 180º pues los módulos de $P_0 P_1$ y $P_1 P_2$ tienden a infinito por lo que en este trabajo se traba con ángulos menores o iguales a 90º, esto no quiere decir que no se puedan realizar circunferencias con ángulos mayor a 90º si no que cuando un ángulo es mayor de 90º se divide en dos o más secciones según lo requiera como se muestra en la Figura 3.19.

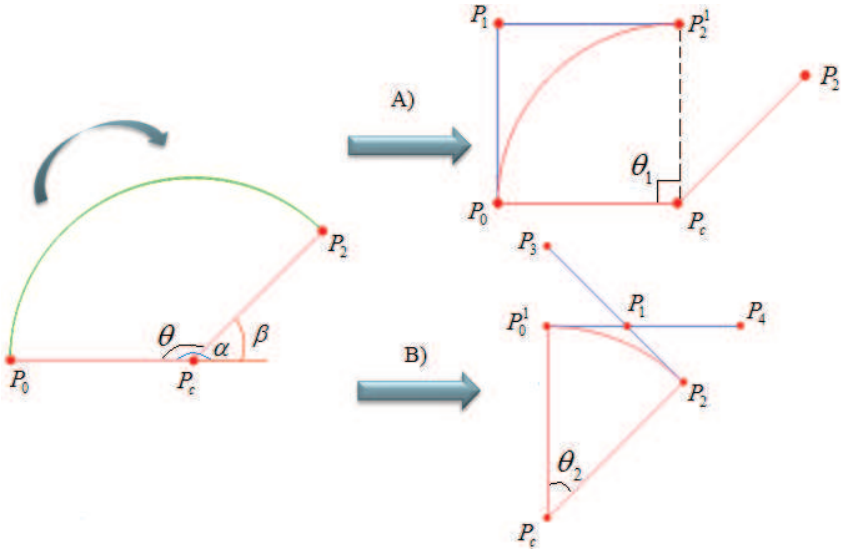

Figura 3.19 División de una interpolación circular con ángulo mayor a 90º.

A partir de cualquier comando de interpolación circular ya sea de código G ó DMC es posible obtener los puntos P_0, P_2 y P_c mostrados en la Figura 3.19. El primer paso es encontrar el ángulo θ y para encontrarlo es necesario calcular primero los ángulos formados por las rectas $P_c P_0$ (α) y $P_c P_2$ (β). Considerando que las interpolaciones circulares únicamente se realizan en dos ejes dependiendo del plano de trabajo se realizo una función para calcular el ángulo formado por dos puntos, la función realiza se encuentra basada en las siguientes condiciones ejemplificadas con los puntos P_c y P_1:

- Si $X_0 - X_c = 0$ y $Y_0 - Y_c > 0$; $\alpha = 90$
- Si $X_0 - X_c = 0$ y $Y_0 - Y_c < 0$; $\alpha = 270$
- Si $X_0 - X_c > 0$ y $Y_0 - Y_c = 0$; $\alpha = 0$
- Si $X_0 - X_c < 0$ y $Y_0 - Y_c = 0$; $\alpha = 180$
- Si $X_0 - X_c > 0$ y $Y_0 - Y_c > 0$; $\alpha = a\tan\left(\frac{Y_0 - Y_c}{X_0 - X_c}\right)$
- Si $X_0 - X_c < 0$ y $Y_0 - Y_c > 0$; $\alpha = a\tan\left(\frac{Y_0 - Y_c}{X_0 - X_c}\right)$
- Si $X_0 - X_c < 0$ y $Y_0 - Y_c > 0$; $\alpha = a\tan\left(\frac{Y_0 - Y_c}{X_0 - X_c}\right) + 180$
- Si $X_0 - X_c < 0$ y $Y_0 - Y_c < 0$; $\alpha = a\tan\left(\frac{Y_0 - Y_c}{X_0 - X_c}\right) + 180$

- Si $X_0 - X_c > 0$ y $Y_0 - Y_c > 0$; $\alpha = a\tan\left(\frac{Y_0 - Y_c}{X_0 - X_c}\right) + 360$

El valor de θ depende del sentido y de los valores de α y β por lo tanto se realizaron las siguientes consideraciones:

- Si $\alpha > \beta$ y el sentido es CW; $\theta = \alpha - \beta$
- Si $\alpha < \beta$ y el sentido es CW; $\theta = 360 - (\alpha - \beta)$
- Si $\alpha > \beta$ y el sentido es CCW; $\theta = 360 - (\beta - \alpha)$
- Si $\alpha < \beta$ y el sentido es CCW; $\theta = \beta - \alpha$
- Si $\alpha - \beta = 0$; $\theta = 360$

Hasta este punto de Figura 3.19 podemos conocer los ángulos α, β y θ, y los puntos P_0, P_2 y P_c. Como θ es mayor a 90º la el arco se divide en los cuadrantes A) y B) como se muestra en la Figura 3.19. Para el cuadrante A) es necesario obtener los puntos P_1 y P_2^1 para ello se debe obtener el radio de la circunferencia (*r*) como se muestra en la ecuación (3.2).

$$r = \sqrt{(X_0 - X_c)^2 + (Y_0 - Y_c)^2} \tag{3.2}$$

Una vez que se tiene el radio es posible encontrar las coordenadas para los puntos P_1 y P_2^1 pero se utiliza una variable para el sentido *v* la cual toma un valor de 90º si es en el sentido CW y -90º si es en el sentido CCW. Las ecuaciones para las coordenadas de P_1 y P_2^1 se muestran a continuación:

$$X_1 = X_c + r(\cos\alpha) + r\left[\cos(\alpha + v)\right] \tag{3.3}$$

$$Y_1 = Y_c + r(\sin\alpha) + r\left[\sin(\alpha + v)\right] \tag{3.4}$$

$$X_2^1 = X_c + r\left[\cos(\alpha + v)\right] \tag{3.5}$$

$$Y_2^1 = Y_c + r\left[\cos(\alpha + v)\right] \tag{3.6}$$

Por último el peso se obtiene con la ecuación (3.7).

$$w = \cos\left(\frac{\theta_1}{2}\right) \tag{3.7}$$

Para el cuadrante B) el ángulo que forma el arco de la circunferencia es menor a 90º y únicamente se conocen los puntos P_0^1, P_2 y P_c ya que $P_0^1 = P_2^1$. Para tener una mayor claridad del proceso que se realiza con los ángulos menores a 90º se utiliza la Figura 3.20 donde $P_0 = P_0^1$, $P_2 = P_2$, $P_c = P_c$, $\theta_2 = \beta$ y $\theta_1 = \alpha + v$ con respecto a la Figura 3.19. El ángulo $\alpha_1 = \theta_1 - \theta_2$, los ángulos $\alpha_2 = 90 - \alpha_1$ y $h = r / \sin(\alpha_2)$ que es el valor de la hipotenusa para obtener con las siguientes ecuaciones las coordenadas de los puntos P_3 y P_4:

$$X_4 = X_c + h(\cos\theta_2) \tag{3.8}$$

$$Y_4 = Y_c + h(\sin\theta_2) \tag{3.9}$$

$$X_3 = X_c + h(\cos\theta_1) \tag{3.10}$$

$$Y_3 = Y_c + h(\sin\theta_1) \tag{3.11}$$

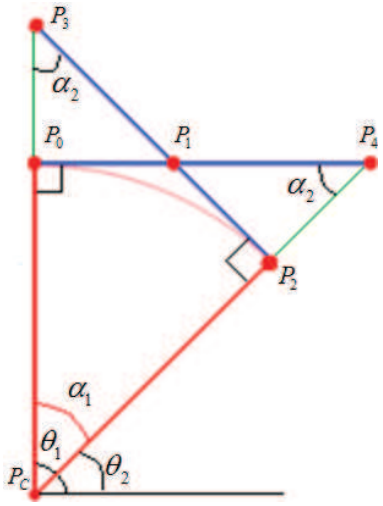

Figura 3.20 Arco circular con un ángulo menor a 90º.

Ya obtenidos los puntos P_3 y P_4, se calcula las pendientes para las rectas formadas por los puntos P_0P_4 y P_2P_3 como se muestra en las ecuaciones (3.12) y (3.13). Utilizando la ecuación de la recta ($y = mx + b$) y despejando la constante *b* se tienen las ecuaciones (3.14) y (3.15).

$$m_{04} = \frac{Y_4 - Y_0}{X_4 - X_0} \tag{3.12}$$

$$m_{23} = \frac{Y_3 - Y_2}{X_3 - X_2} \tag{3.13}$$

$$b_{04} = Y_0 - m_{04}X_0 \tag{3.14}$$

$$b_{23} = Y_2 - m_{23}X_2 \tag{3.15}$$

Por último se despeja la coordenada *x* de las dos ecuaciones de la recta para obtener el valor de intersección en el eje *x*, esto mismo se realiza para la coordenada *y* obteniendo las ecuaciones (3.16) y (3.17).

$$\frac{Y_1 - b_{04}}{m_{04}} = \frac{Y_1 - b_{23}}{m_{23}} \tag{3.16}$$

$$m_{04}X_1 + b_{04} = m_{23}X_1 + b_{23} \tag{3.17}$$

Despejando los valores de Y_1 y X_1 de las ecuaciones (3.16) y (3.17) se obtienen las ecuaciones (3.18) y (3.19) que son utilizadas para obtener los valores de las coordenadas del punto P_1 el cual es el punto faltante para generar la curva NURB.

$$Y_1 = \frac{b_{04}m_{23} - b_{23}m_{04}}{m_{23} - m_{04}} \tag{3.18}$$

$$X_1 = \frac{b_{23} - b_{04}}{m_{04} - m_{23}} \tag{3.19}$$

3.3.3 Intérprete

La parte del intérprete es parte fundamental de este trabajo puesto que genera una base de datos de curvas NURBS en listas enlazadas, generando una lista de todas las curvas requeridas para la generación de la trayectoria. Esto es necesario ya que el controlador propio realiza movimientos de curvas NURBS por lo que requiere los parámetros, los cuales son una serie de puntos de control, nodos y pesos, para realizar el movimiento de la curva.

La lista de curvas es necesaria ya que una trayectoria puede estar compuesta por varias curvas ya sean lineales, circulares ó NURBS y el controlador propio ejecuta solo una curva de tal forma que al realizar una trayectoria se envía la primer curva y se espera a que finalice su ejecución para enviar la siguiente curva y así sucesivamente hasta terminar la lista de curvas que conforman la trayectoria.

En este punto del trabajo se tiene una lista de comandos ya sean de código G ó DMC, estos comandos son interpolaciones lineales, circulares ó NURBS sin embargo, es necesario convertir las interpolaciones lineales y circulares en interpolaciones NURBS puesto que el controlador propio únicamente interpreta curvas NURBS. La conversión de interpolaciones lineales y circulares se muestra en la sección 3.3.2.

3.3.3.1 Intérprete de código CN

La interpretación de comandos no es más que convertir todas las interpolaciones en curvas NURBS y guardarlas en una lista para su ejecución ó simulación. En la Figura 3.21 se muestra un diagrama de flujo en donde el primer paso es obtener la lista de comandos G estos comandos únicamente contendrán interpolaciones lineales, circulares y NURBS como se observa en la Tabla 3.6 las interpolaciones lineales generaran interpolaciones NURBS el procedimiento se encuentra descrito en la sección 3.3.2.1 y las interpolaciones circulares generaran una serie de curvas NURBS como se explica en la sección 3.3.2.2.

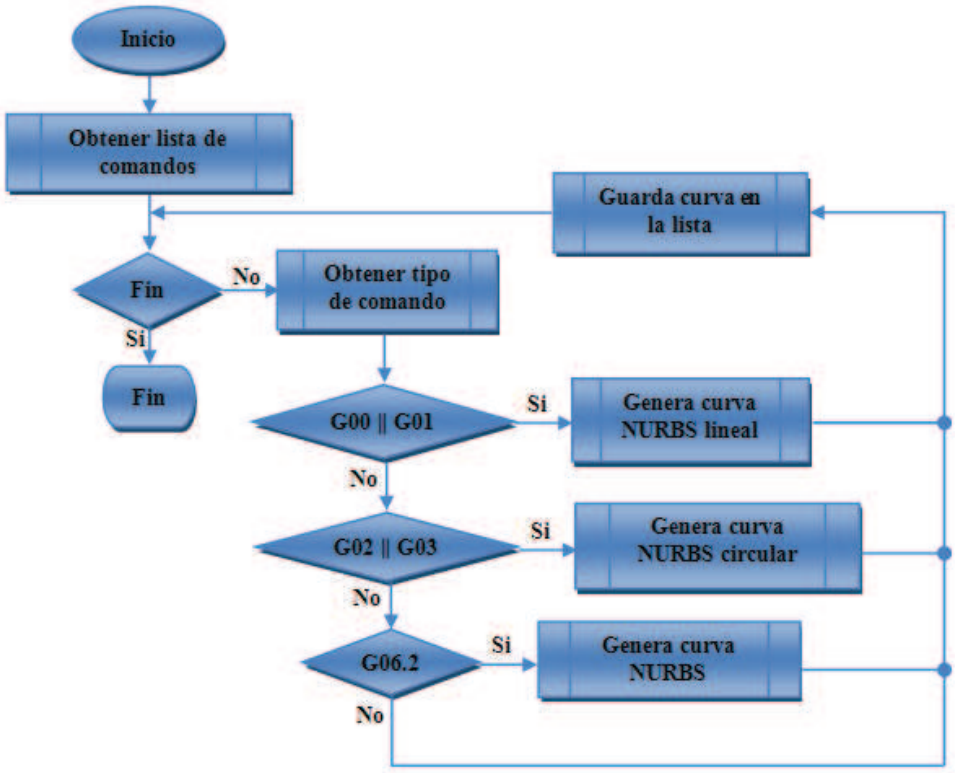

Figura 3.21 Diagrama de flujo del almacenamiento de curvas NURBS a partir de comandos G.

Para interpolaciones NURBS únicamente se almacenan los puntos de control que contengan un nodo distinto de uno con su respectivo peso y también se almacenan todos los nodos especificados para la curva. Cada una de las curvas se guarda en una lista la cual contendrá todas las curvas para construir la trayectoria ya sea para que la simule el software ó la ejecute el controlador propio.

3.3.3.2 Intérprete de código DMC

En la Figura 3.22 se muestra un diagrama de flujo en el cual se observa el procedimiento que se realiza para generar una lista de curvas NURBS de un programa DMC para su simulación ó para su ejecución en el sistema de control propio. El primer paso es obtener una lista de comandos como la que se muestra en la Tabla 3.7, a continuación se realiza una lista de los pasos que se realizan en el diagrama de flujo de la Figura 3.22 considerando que nos encontramos en el inicio de la lista de comandos.

1. Se busca el siguiente comando de inicio de movimiento (BG ó BGS) para encontrar los ejes que se encuentran activos. Si no existe ningún comando de inicio de movimiento no se realiza ni simula ningún movimiento y finaliza el proceso.
2. Si se encuentra un comando de inicio de movimientos se obtienen los ejes activados y dependiendo de los ejes que estén activos se obtiene la velocidad, aceleración y desaceleración.
3. Se regresa hasta el inicio de la lista de comandos ó hasta encontrar un comando de inicio de movimiento.
4. En caso de que se haya encontrado un comando de inicio de movimiento en el paso anterior se avanza un comando en la lista y se obtiene el tipo de comando.
5. En caso de que el comando indique una interpolación lineal ó circular se genera en curva NURBS como se describe en la sección 3.3.2 y se guardan en una lista de curvas las cual será utilizada para la simulación y ejecución del programa, en caso que se encuentre un comando DP se coloca esa posición como la posición para iniciar el movimiento. Para estos tres casos se obtiene el siguiente comando y se vuelve a realizar el paso 5.
6. Si el tipo de comando no entra en ninguno de los casos especificados en el paso 5, esto indica que es un comando de inicio de movimiento por lo que se regresa al paso 1.

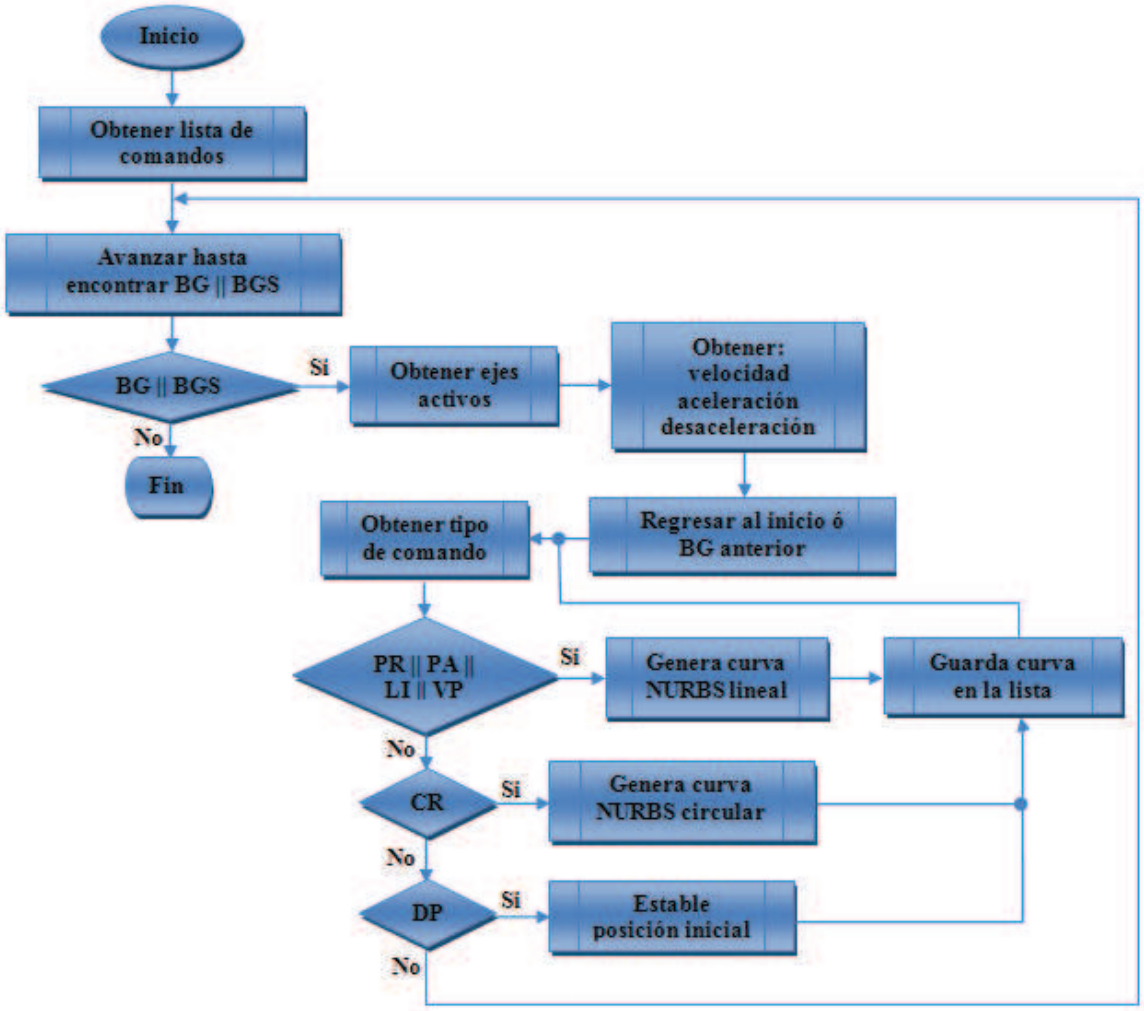

Figura 3.22 Diagrama de flujo del almacenamiento de curvas NURBS a partir de comandos DMC.

3.3.4 Evaluación de curvas NURBS

Una vez que ya se tienen todas las curvas almacenadas el último proceso faltante es la simulación pues la ejecución ya se encuentra integrada en los módulos de software realizados por Morales (2010). Dichos módulos únicamente requieren la curva que se desea ejecutar por el sistema de movimiento sin embargo solo reciben una curva a la vez por lo que se agrego un hilo el cual espera a que termine la ejecución de una curva para enviar la siguiente, realizando este proceso hasta finalizar la lista de curvas. Durante la ejecución se almacenan los puntos cada vez que se detecta algún cambio en los puntos que regresa el controlador de movimiento esto se realiza con el fin de simular la ejecución entiempo real.

Regresando a la parte de la simulación se implemento un algoritmo basado en las ecuaciones (2.7), (2.8), (2.9) y (2.10) para la evaluación de curvas NURBS. El diagrama de flujo mostrado en la Figura 3.23 muestra el proceso para la evaluación de una curva donde los datos de entrada son el número de puntos que se desean para la reconstrucción de la curva (*N*); los puntos de control (Q_i), cada punto de control es conformado por las coordenadas (X, Y, Z y W) y su respectivo peso; y los nodos (tk_i). Es necesario obtener el grado de la curva (*p*) el cual es definido por la ecuación (3.20) ya que es requerido para los ciclos de la evaluación. Las curvas se evalúan en un intervalo de 0 a 1 de tal forma que para obtener *N* puntos es necesario tener un incremento de $1/N$ entre cada evaluación. El primer paso para la obtención de un punto para la reconstrucción es encontrar el índice *r*, este índice no es más que una posición del vector de nodos y debe cumplir que $tk_{r-1} \le t < tk_r$ el siguiente paso es obtener los puntos de control a evaluar estos se consiguen en un ciclo el cual va de 0 a *p* guardando los puntos de control en la matriz *PP* de tamaño $p \times p$, la posición de los puntos de control se encuentra dada por la ecuación (3.21). Una vez que se tienen *p+1* puntos de control comienza la evaluación recursiva que consiste en ir generando una matriz triangular en *PP* calculando el valor de α como se muestra en la ecuación (3.22) para encontrar el punto $PP_{j,i-1}$ con la ecuación (3.23). Al finalizar los ciclos se genera el punto $PP_{0,0}$ al cual únicamente le falta dividir el valor de las coordenadas obtenidas entre el peso como se muestra en la ecuación (3.24) para obtener un punto para la reconstrucción de la curva. Este proceso se realiza para toda la lista de curvas generada en la parte del intérprete.

$$p = num[tk_i] - num[Q_i] - 1 \tag{3.20}$$

$$jp = r - ip - 1 \tag{3.21}$$

$$\alpha = \frac{t - tk_{j+r-i}}{tk_{j+r} - tk_{j+r-i}} \tag{3.22}$$

$$jp = r - ip - 1 \tag{3.23}$$

$$PP_{0,0}[X,Y,Z,W] = \frac{PP_{0,0}[X,Y,Z,W]}{PP_{0,0}[R]} \tag{3.24}$$

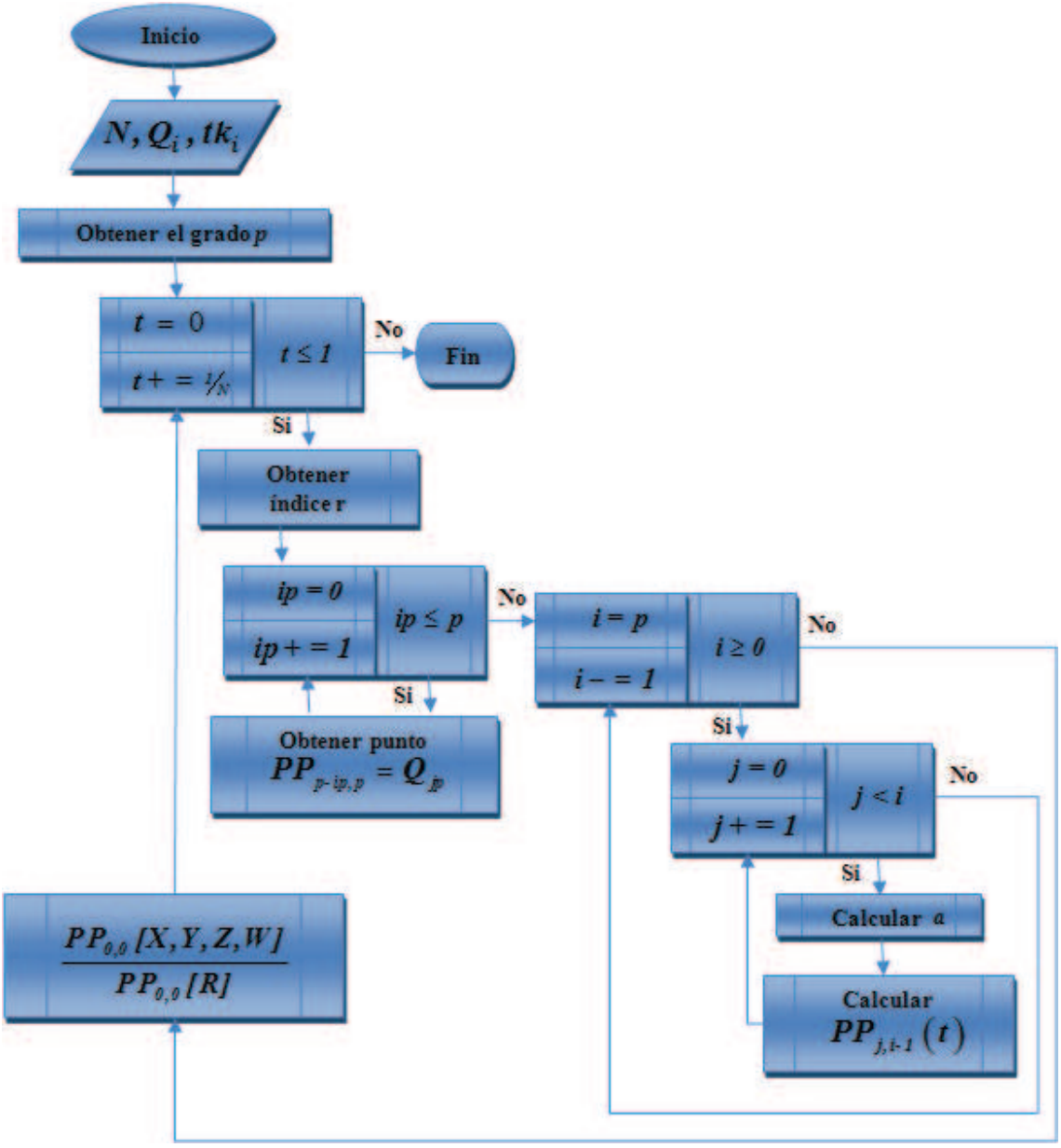

Figura 3.23 Diagrama de flujo para la simulación de NURBS.

3.4 Jog

En todos los software de sistemas de control de posicionamiento existe una herramienta llamada jog en donde se realizan movimientos individuales para cada uno de

los ejes. Esta herramienta es muy importante ya que permite posicionar el cortador de la máquina-herramienta en la posición donde debe iniciar el maquinado, indicar el inicio y moverse al inicio indicado. También es utilizado para verificar el funcionamiento individual de cada uno de los ejes. En la Figura 3.24 se muestra el jog realizado para GUI del software donde se debe indicar cuanto se desea mover eligiendo la medida esta puede ser cuentas, milímetros, centímetros y pulgadas. Para realizar el movimiento se puede seleccionar el eje en el combo box de "Eje" y después dar click en "Mover" indicando el sentido en la caja de texto, la otra forma de realizar el movimiento es simplemente dar click en uno de los botones del jog para este caso no importa el signo de la caja de texto ya que se toma el valor absoluto y se mueve en el sentido indicado mandando una interpolación lineal al controlador. Se agrego en la parte del jog un combo box donde se elige el número de ejes con los que se desea trabajar el cual puede variar de uno a cuatro.

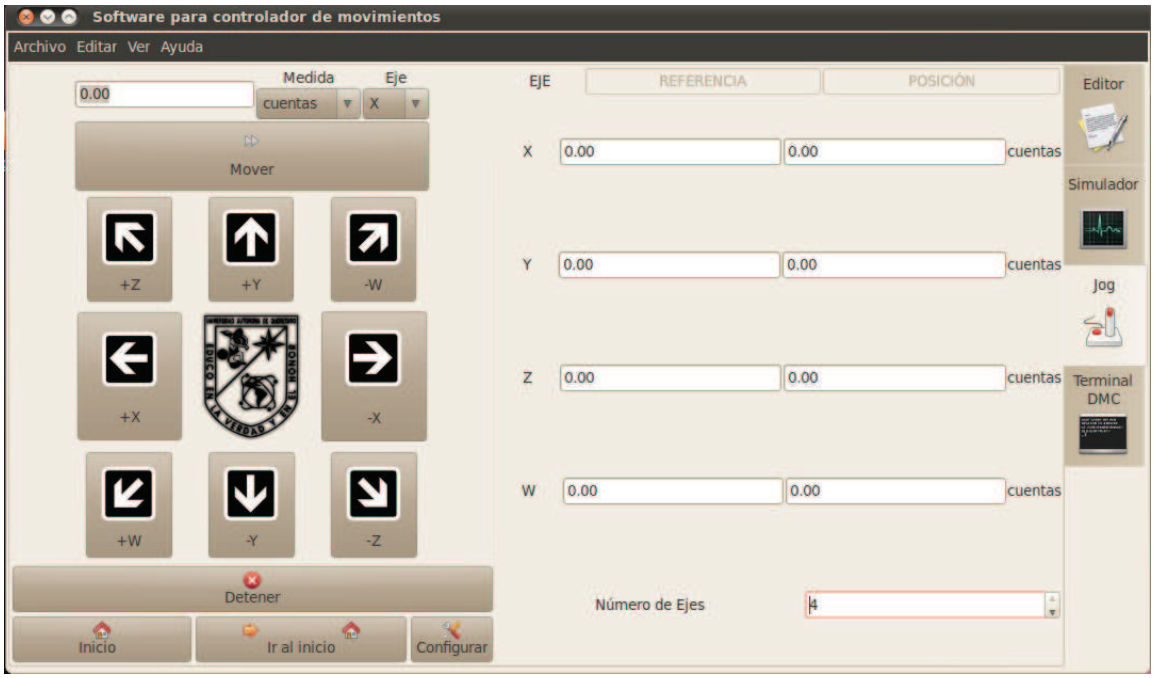

Figura 3.24 Pestaña del jog de la GUI.

En la Figura 3.24 se observa un botón de configuración este botón abre una ventana secundaria en la cual se introducen los parámetros del controlador y de movimiento. En la Figura 3.25 se muestra la pestaña de los parámetros del controlador en esta pestaña se introducen los valores de la ganancia proporcional (Kp), integral (Ki) y derivativa (Kd)

para cada uno de los ejes. Cabe mencionar que estas ganancias son calculadas dependiendo del controlador que se vaya a utilizar.

Figura 3.25 Pestaña de los parámetros del controlador.

En la Figura 3.26 se muestra la pestaña para los parámetros de movimiento, el primer parámetros es la frecuencia de muestreo (Fs) este parámetro únicamente es utilizado por el controlador MCUAQ4X pues almacena los datos en una memoria RAM (*Random Access Memory*, Memoria de acceso aleatorio), el segundo parámetro es la velocidad de avance (Vf) y es dado en cuentas por segundo, el tercer y cuarto parámetro es la aceleración (Am) y des aceleración (Dm) y son dados en cuentas por segundo cuadrado. Estos tres últimos parámetros son utilizados para la generación de perfiles. El cuarto parámetro es la equivalencia en cuentas de un centímetro, este parámetro es utilizado solo por el software para la conversión de unidades. Por último se tiene el límite de error este parámetro es utilizado por los controladores para la seguridad del sistema.

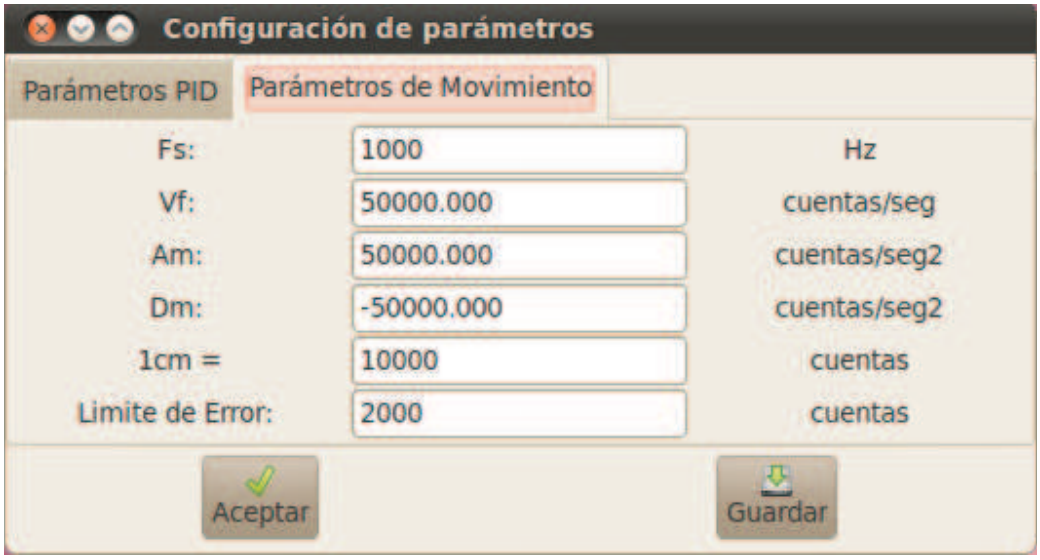

Figura 3.26 Pestaña de los parámetros de movimiento.

Cabe mencionar que el controlador MCUAQ4X tiene la capacidad de almacenar los parámetros Fs, Vf, Am, Dm y limite de error ya que contiene una EEPROM (*Electrically-Erasable Programmable Read-Only Memory*, Memoria de solo lectura programable y borrable eléctricamente).

3.5 Terminal DMC

Basados en la terminal desarrollado por Galil se diseño una terminal para comandos DMC la cual se muestra en la Figura 3.27 en donde se escribe el comando que se desea enviar al controlador dicho comando es compilado antes de ser cargado y se ejecuta la rutina hasta enviar un comando de inicio de movimiento (BG ó BGX), además se le agrego una sección donde se puede observar el movimiento que se encuentra realizando ó fue realizado y al igual que en el simulador es posible observar la simulación, la referencia y la posición. Sin embargo debido a las propiedades que tiene los comandos DMC no es posible simular antes de la ejecución, es decir, todas las gráficas son realizadas durante ó después de la ejecución.

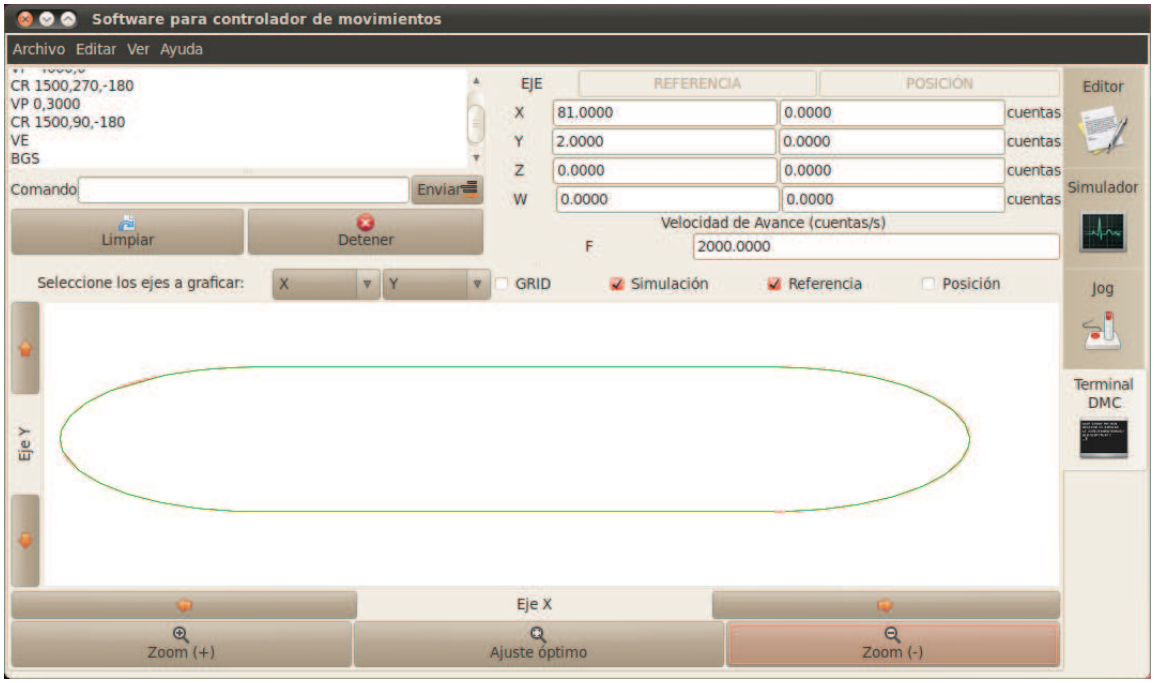

Figura 3.27 Pestaña de la terminal DMC de la GUI.

El proceso de ejecución y simulación es el mismo realizado en el simulador.

3.6 Experimentación

La experimentación fue dividida en dos partes principales, la primera experimentación se encuentra basada en el maquinado de una pieza la cual fue realizada en un torno CNC de procedencia checa, instrumentado por la Universidad Autónoma de Querétaro (Figura 3.28), para esta experimentación se utilizaron ambos controladores para realizar la misma pieza. El torno tiene como unidades motrices de los ejes X y Z un servo EZG705 de motor 0-101, un encoder radial de 6000 cuentas/rev, y un servoamplificador Copley-403 a 25 Vdc por cada eje y el husillo está dotado de un motor trifásico de corriente alterna configurado a 220 volts en conexión delta. El control se hace a través de una computadora con Windows 2000 la cual aloja una tarjeta de control Galil de 2 ejes. Dicha tarjeta cuenta con software para aplicaciones (Galil, 1994) y se efectúa a través de ella y el sistema de control de posicionamiento basado en FPGA la aplicación del control numérico.

Figura 3.28 Torno CNC utilizado en la experimentación.

La herramienta empleada en la experimentación fue un inserto de carburo de tungsteno de la empresa Travers con el número de inserto TNMG 432 (Figura 3.29). Su número de catálogo es 22-003-094 con operación de desbaste y con rompevirutas. El material utilizado para la experimentación fue una barra de aluminio de una pulgada de diámetro.

Figura 3.29 Nomenclatura del inserto.

Para la medición de la rugosidad en las piezas maquinadas se utilizo un rugosímetro modelo SJ-201 de la marca Mitutoyo el cual se muestra en la Figura 3.30.

Figura 3.30 Rugosímetro modelo SJ-201 de la marca Mitutoyo.

En la segunda parte de la experimentación se realizó en una fresadora instrumentada en paralelo con este trabajo en la Universidad Autónoma de Querétaro (Figura 3.31), y únicamente se utilizo el controlador propio ya que no se cuenta con un controlador GALIL para tres ejes. La fresadora tiene como unidades motrices de los ejes X, Y y Z un servomotor Brushless IB34003, un encoder radial de 4000 cuentas/rev, y un servoamplificador Brushless BX25A20 a 190 Vdc por cada eje y aun falta de instrumentar la parte del husillo, por lo que no se realizó maquinado únicamente se realizaron varias trayectorias para el movimiento de los ejes. El control se hace a través de una computadora personal y el controlador de movimientos MCUAQ4X y de esta manera comprobar el funcionamiento del software realizado.

Figura 3.31 Fresadora CNC utilizada en la experimentación.

Para la implementación de las curvas NURBS fue utilizado el controlador de movimiento de 4 ejes MCUAQ4X (Figura 3.32), el controlador está basado en un FPGA de bajo costo y alta capacidad. El sistema está diseñado para que todos los cálculos de tiempo real sean realizados por el hardware y la interfaz con el usuario sea mediante una PC que se comunica mediante puerto serial RS232 o USB. En este trabajo se utilizó el puerto USB para la comunicación entre el hardware y software.

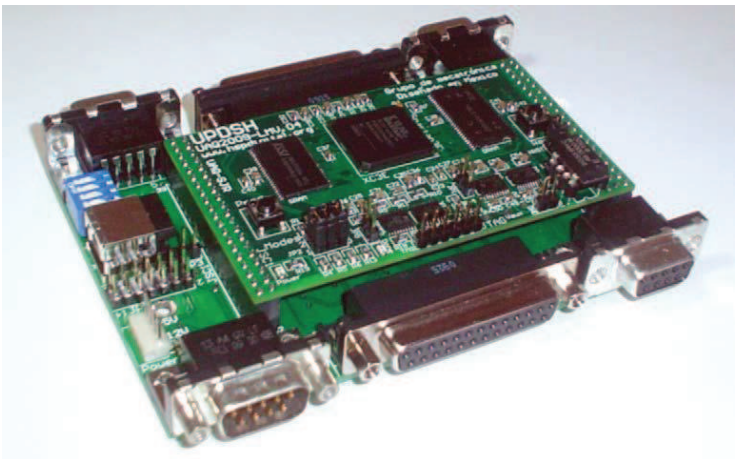

Figura 3.32 Controlador de movimiento para 4 ejes MCUAQ4X.

3.6.1 Maquinado de una trayectoria

Este experimento consiste en realizar un maquinado en una barra de aluminio la cual tiene un diámetro de 2.54cm. El modelo CAD se muestra en la Figura 3.33c el cual fue generado con Unigraphics 6 (NX6) que es un software estándar de CAD/CAM, la pieza tiene una altura de 5.00cm.

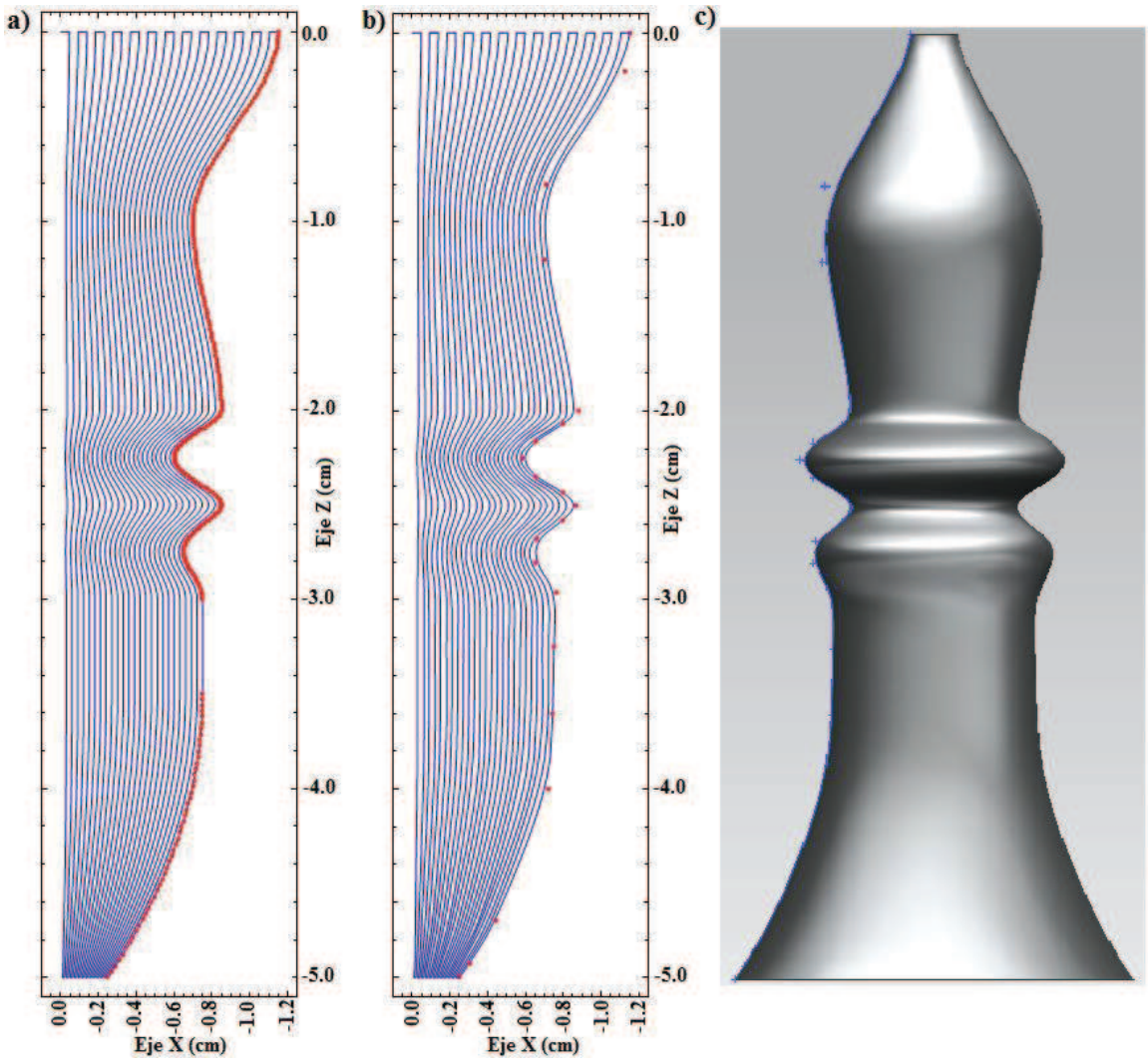

Figura 3.33 Pieza de trabajo, (a) trayectoria basada en interpolaciones lineales, (b) trayectoria basa en interpolaciones NURBS, (c) modelo CAD.

La trayectoria de maquinado basada en interpolaciones lineales es presentada en la Figura 3.33a la cual consta de 6176 interpolaciones lineales G01, velocidad de avance constante 300 mm/min y velocidad de husillo de 900 rev/min. Los nodos al final del maquinado son resaltados con puntos en la Figura 3.33a y muestra la segmentación hecha

por el código G donde se puede apreciar el excesivo número de trazos lineales introducidos en ciertas regiones.

La trayectoria de maquinado basada en interpolaciones NURBS es presentada en la Figura 3.33b la cual consta de 25 interpolaciones NURBS G06.2 y 26 interpolaciones lineales G01. Cada una de las interpolaciones NURBS consiste de 21 puntos de control, velocidad de avance constante 5000 mm/min y velocidad de husillo de 900 rev/min. Los puntos de control de la ultima interpolación NURBS son resaltados en la Figura 3.33b.

3.6.2 Seguimiento de una trayectoria

Esta experimentación consiste en analizar el seguimiento de los ejes de la fresadora al realizar el desbaste para obtener una pirámide como la mostrada en la Figura 3.34b a partir de la pieza mostrada en la Figura 3.34a la cual tiene un grosor de 0.762mm. El grosor de cada una de las estrellas de la pieza final es de 0.254mm.

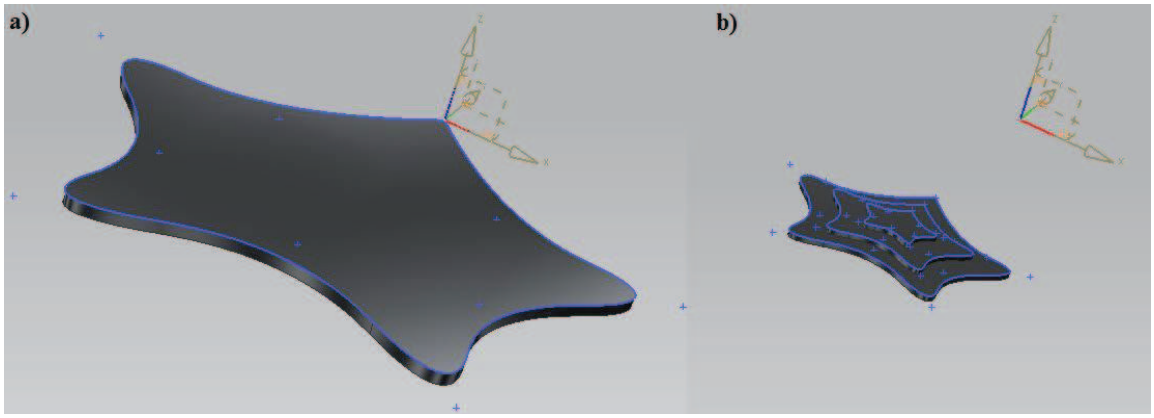

Figura 3.34 Pieza de trabajo, (a) pieza inicial, (b) pieza final.

La trayectoria del maquinado se muestra en la Figura 3.35, las trayectorias se muestra solo en los ejes X y Y, sin embargo, existen movimientos en el eje Z. El primer movimiento se realiza en el eje Z, es un movimiento lineal de -0.254mm, al terminar el movimiento se comienza la trayectoria mostrada en la Figura 3.35 hasta realizar la estrella más pequeña. Una vez finalizada se mueve al punto (0.0, 0.0, -0.254) para realizar otro movimiento lineal en el eje Z y realizar la trayectoria hasta las líneas verdes obteniendo de este modo la segunda estrella. Se regresa al punto (0.0, 0.0, -508) para realizar el último movimiento lineal en el eje Z y realizar la última trayectoria esta llega hasta la línea negra

obteniendo de esta forma la pieza final mostrada en la Figura 3.34b. El maquinado consiste en 51 interpolaciones NURBS y 53 interpolaciones lineales, cada una de las interpolaciones NURBS se encuentra conformada con 10 puntos de control, la velocidad de avance utilizada es de 100000cuentas/seg para los desbastes intermedios y de 30000cuentas/seg en los desbastes finales para cada una de las estrellas. Los puntos de control de la estrella más pequeña son resaltados en la Figura 3.35.

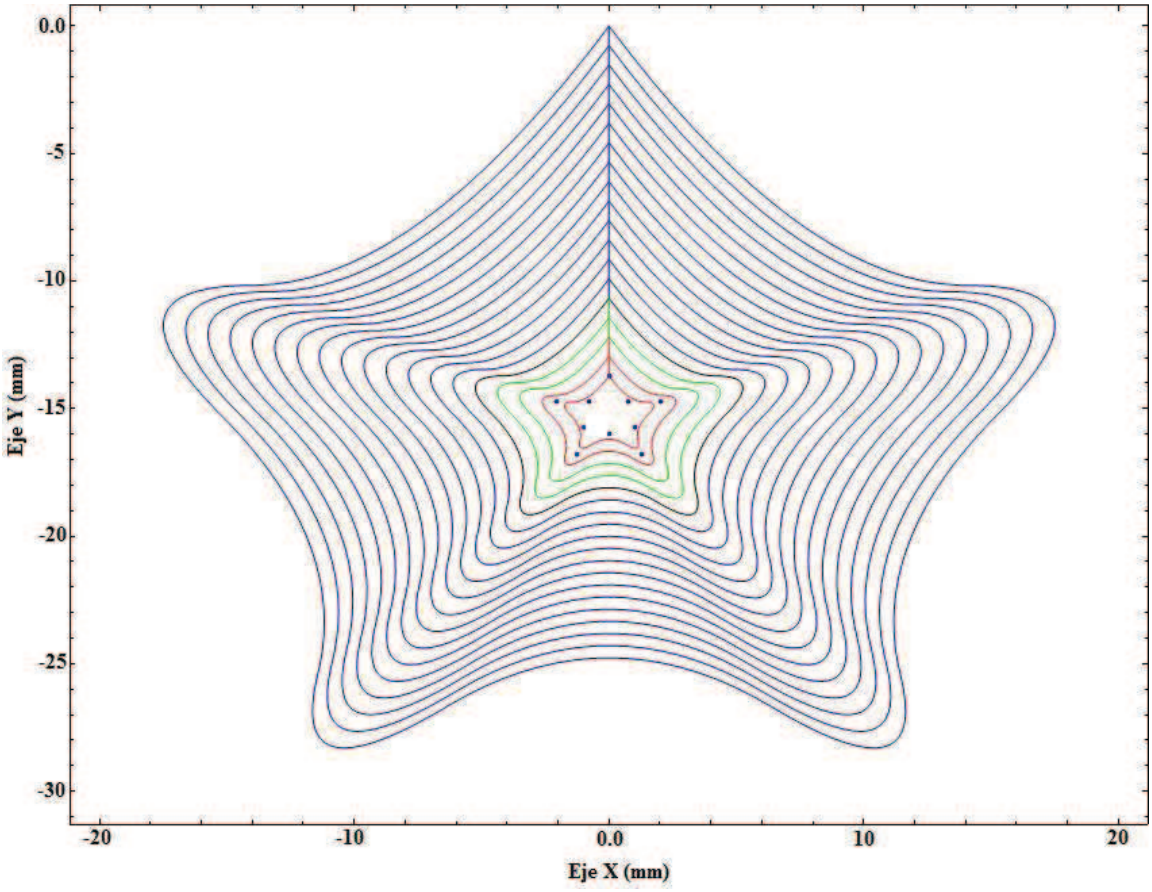

Figura 3.35 Trayectoria para el maquinado.

Capítulo IV.

RESULTADOS

En este capítulo se presentan los resultados obtenidos de simulación, experimentación y el funcionamiento del software realizado para distintas trayectorias utilizadas tanto para maquinado en torno como para únicamente los movimientos de los ejes en una fresadora.

La metodología propuesta para el funcionamiento del software comienza después de generar un código G ó DMC, este código puede ser cargado en el software ó escrito directamente en el editor del software. Como fue mencionado no todos los comandos son interpretados por lo que el primer paso es realizar una compilación la cual muestra los comandos que no son aceptados para la interpretación y deben ser modificados ó eliminados del programa. Una vez que se tiene el código correcto, es decir, sin errores es posible interpretar los comandos contenidos en el programa para ser simulados y si se desea realizar la ejecución. Durante la ejecución es posible monitorear el movimiento que van realizando los ejes de la máquina-herramienta así como la referencia que se le envía al controlador, este monitoreo se realiza cada 150 milisegundos en el software, sin embargo en el hardware es posible monitorear cada microsegundo, es decir, a una frecuencia de 1KHz, estos datos son almacenados en la memoria RAM los cuales al finalizar la ejecución del programa pueden ser descargados para el análisis de seguimiento en programas como MATLAB ó Mathematica.

4.1 Maquinado de una trayectoria

Se realizaron dos maquinados, el primero se baso en interpolaciones lineales y el segundo en interpolaciones NURBS.

El primer programa para el maquinado contiene 6176 interpolaciones lineales, las coordenadas de dicho código se encuentran dadas en milímetros y tiene una velocidad de avance de 5000mm/min. El código fue cargado en el editor del software como se muestra en la Figura 4.1 donde se realizo la compilación del programa.

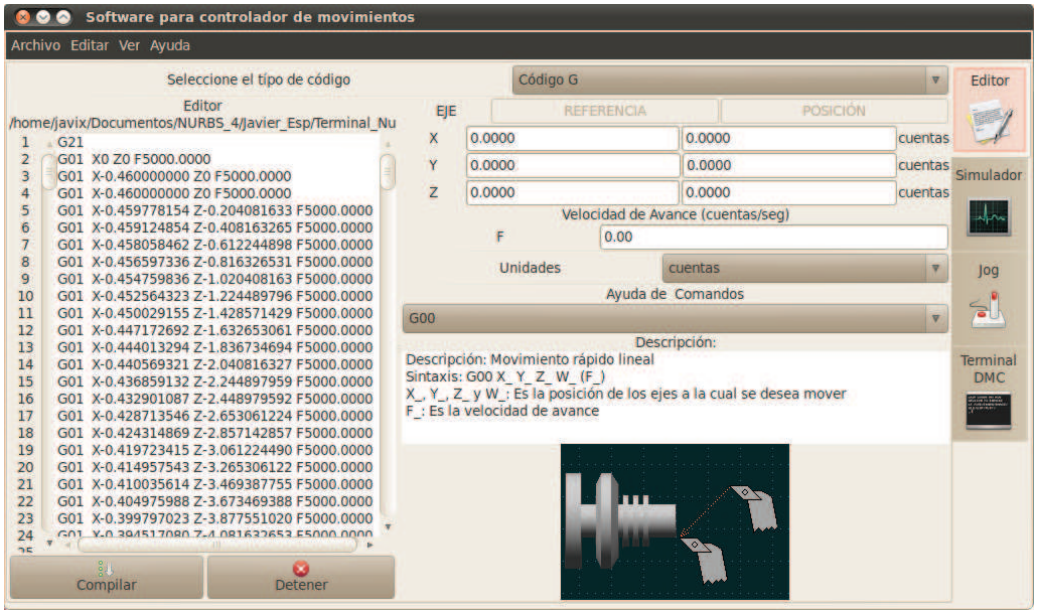

Figura 4.1 Editor del software con el programa basado en interpolaciones lineales.

Una vez compilado el programa se cargo automáticamente el programa en la pestaña del simulador en donde se realizo la simulación del maquinado como se muestra en la Figura 4.2 para esto se convirtieron todas las interpolaciones en curvas NURBS, también se muestra un mallado en el cual las unidades son cuentas.

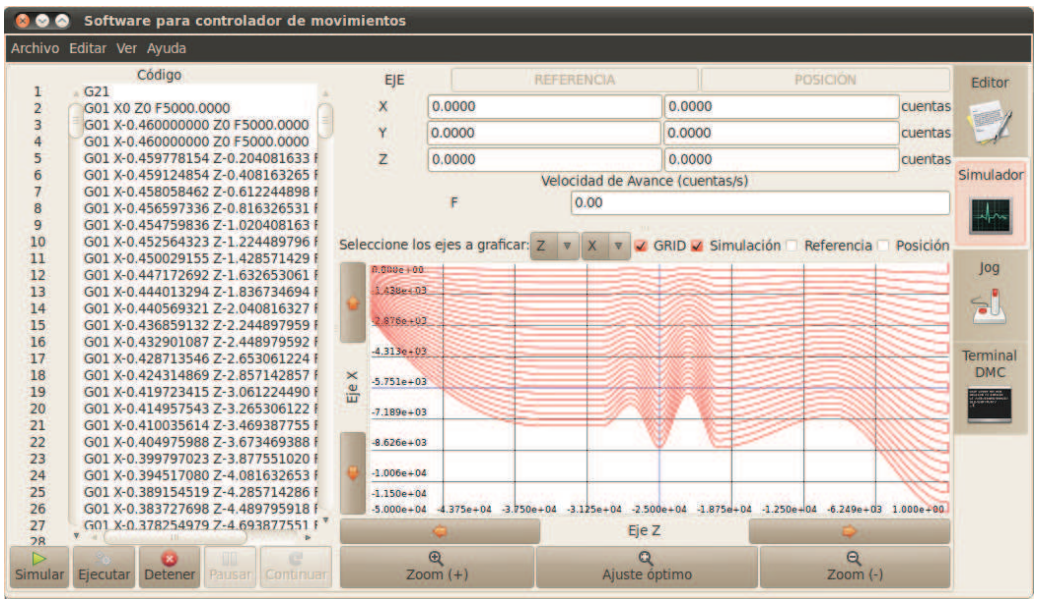

Figura 4.2 Simulación del programa basado en interpolaciones lineales.

En esta misma pestaña se realiza la ejecución del maquinado y como se muestra en la Figura 4.3 se monitorea la referencia enviada por el controlador y la posición de la máquina herramienta y se puede observar gráficamente en el software. Durante la ejecución se inhabilitan varios botones para que el usuario no interrumpa la comunicación entre el software y el hardware. El maquinado tuvo una duración aproximada de 41 minutos.

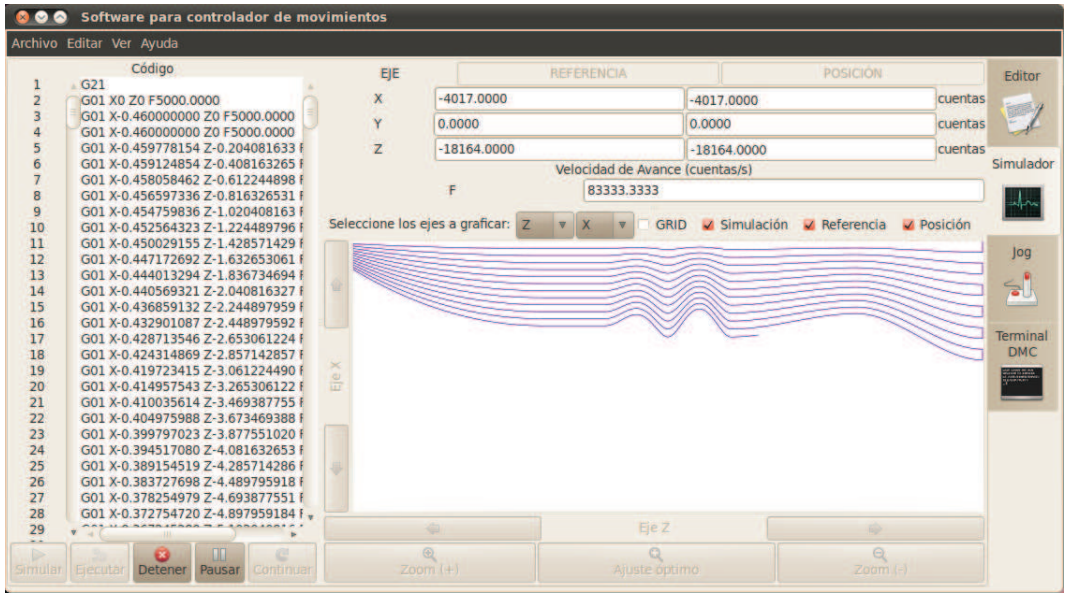

Figura 4.3 Ejecución del programa basado en interpolaciones lineales.

El segundo programa consta de 25 interpolaciones NURBS y 26 interpolaciones lineales y las unidades de trabajo utilizadas fueron cuentas. Cada una de las interpolaciones NURBS consiste de 21 puntos de control, la velocidad de avance aplicada fue de 5000 mm/min. El programa cargado en el software se muestra en la Figura 4.4 el cual fue compilado en esta sección para ser cargado en la pestaña del simulador.

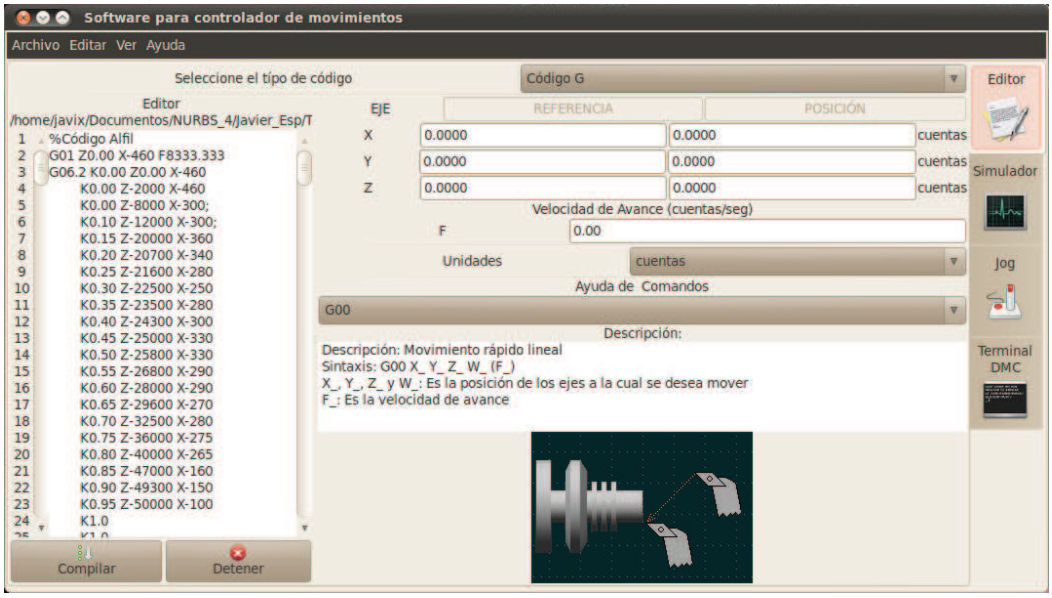

Figura 4.4 Editor del software con el programa basado en interpolaciones NURBS.

Cargado el programa en la parte del simulador se realizo la simulación y la ejecución del programa. La ejecución se muestra en la Figura 4.5, el maquinado tuvo una duración aproximada de 3 minutos.

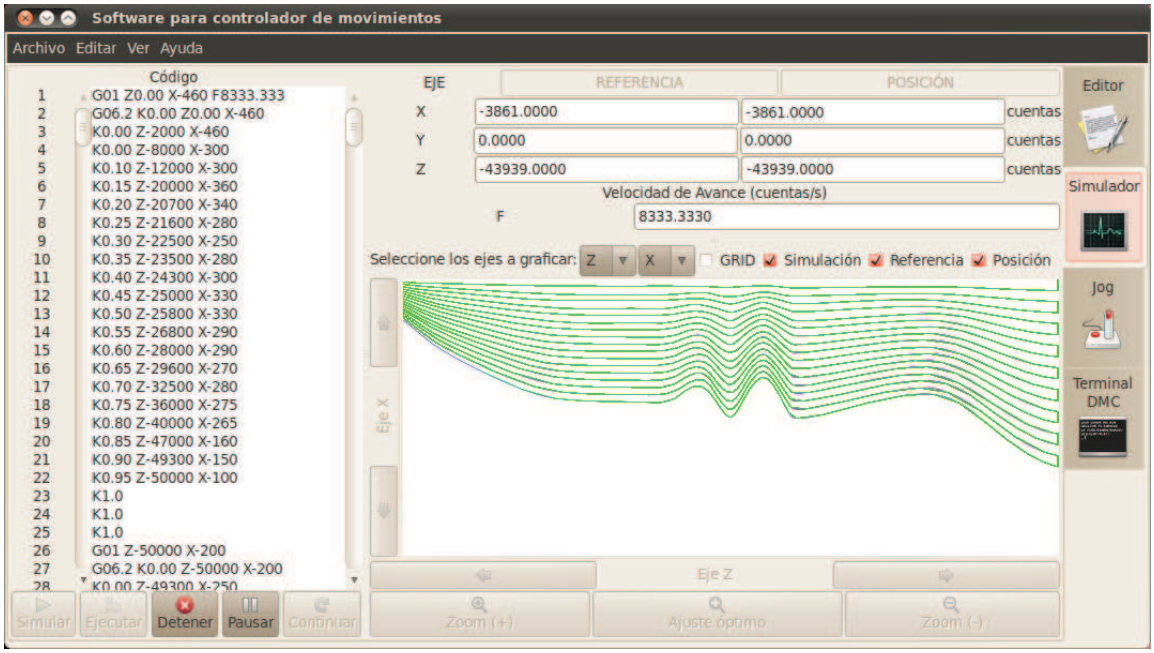

Figura 4.5 Ejecución del programa basado en interpolaciones NURBS.

Cabe mencionar que antes de realizar la ejecución se establecieron los paremos de control y de movimiento desde el software. Los parámetros de control utilizados para el torno se muestran en la Figura 4.6.

Figura 4.6 Parámetros de control para torno.

En la Figura 4.7 se muestran los parámetros de movimiento utilizados para el maquinado en el torno.

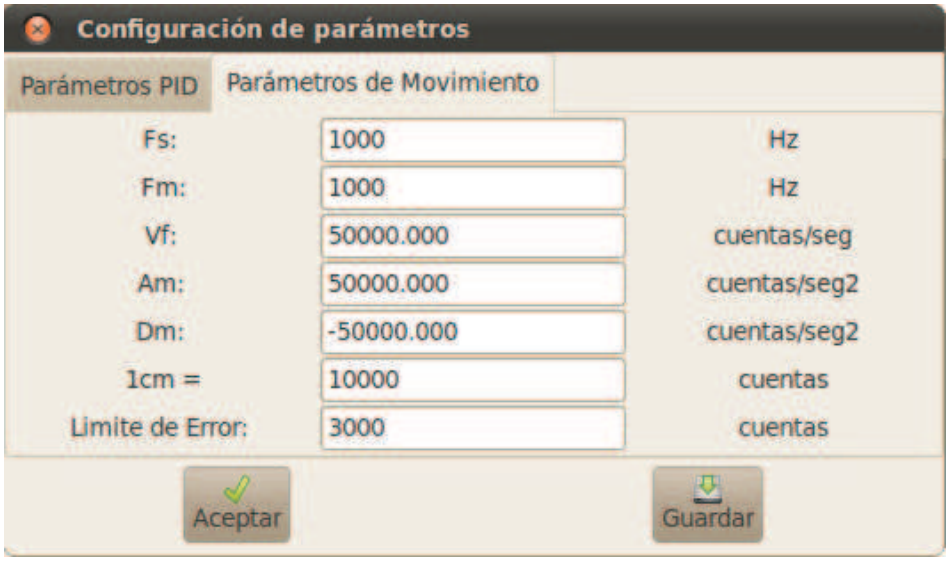

Figura 4.7 Parámetros de movimiento para torno.

Para el programa basado en interpolaciones lineales se utilizo la tarjeta de control Galil mientras que para el programa basado en interpolaciones NURBS se utilizo el controlador MCUAQ4X. Los resultados de los maquinados se muestran en la Figura 4.8.

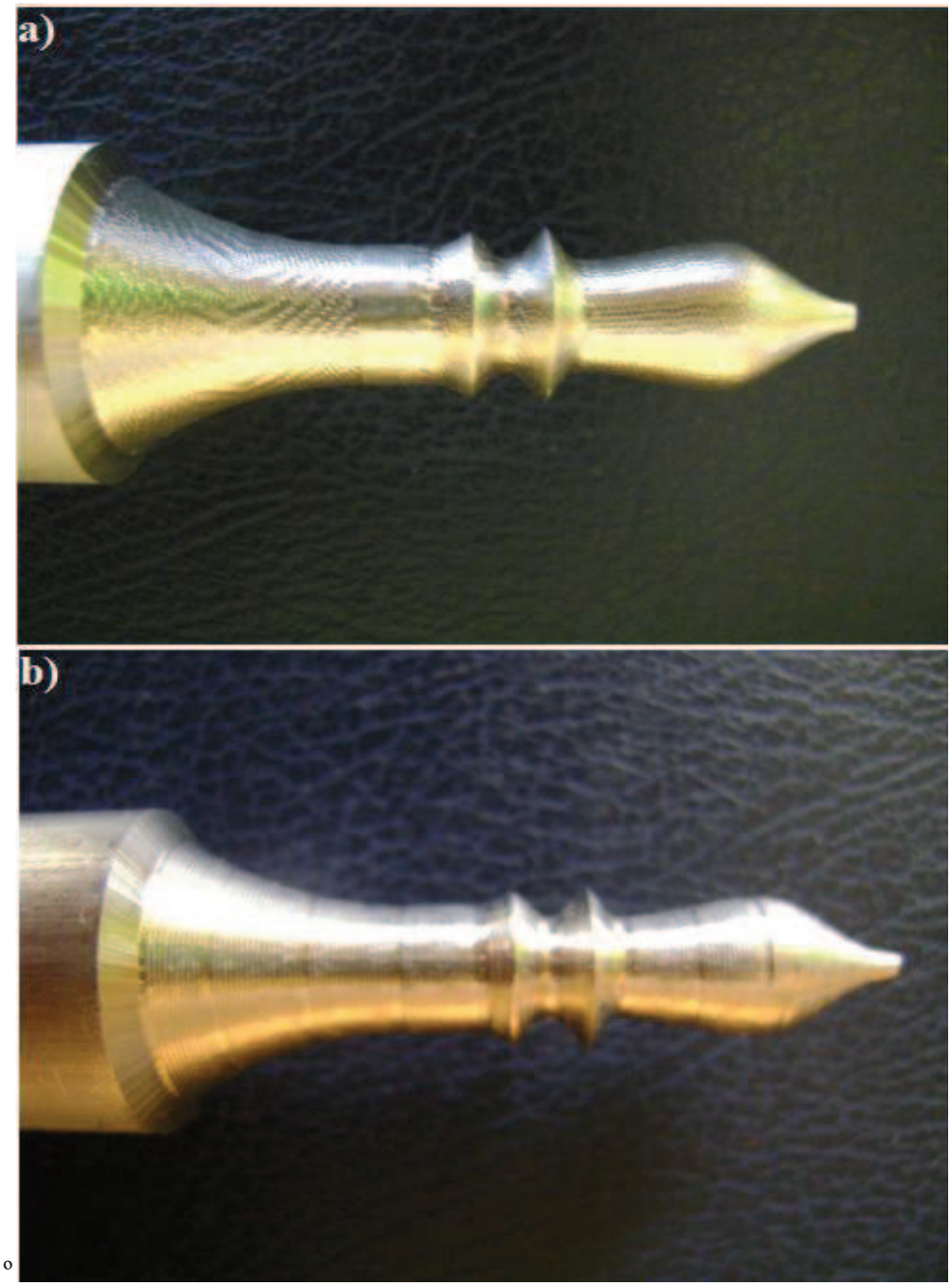

Figura 4.8 Resultado de los maquinados, (a) realizado con el controlador Galil, (b) realizado con el controlador MCUAQ4X.

En cuanto a la calidad del maquinado se realizaron varias mediciones de rugosidad a las piezas en cuatro regiones estratégicas mostradas en la Figura 4.9 con un rugosímetro modelo SJ-201 de la marca Mitutoyo.

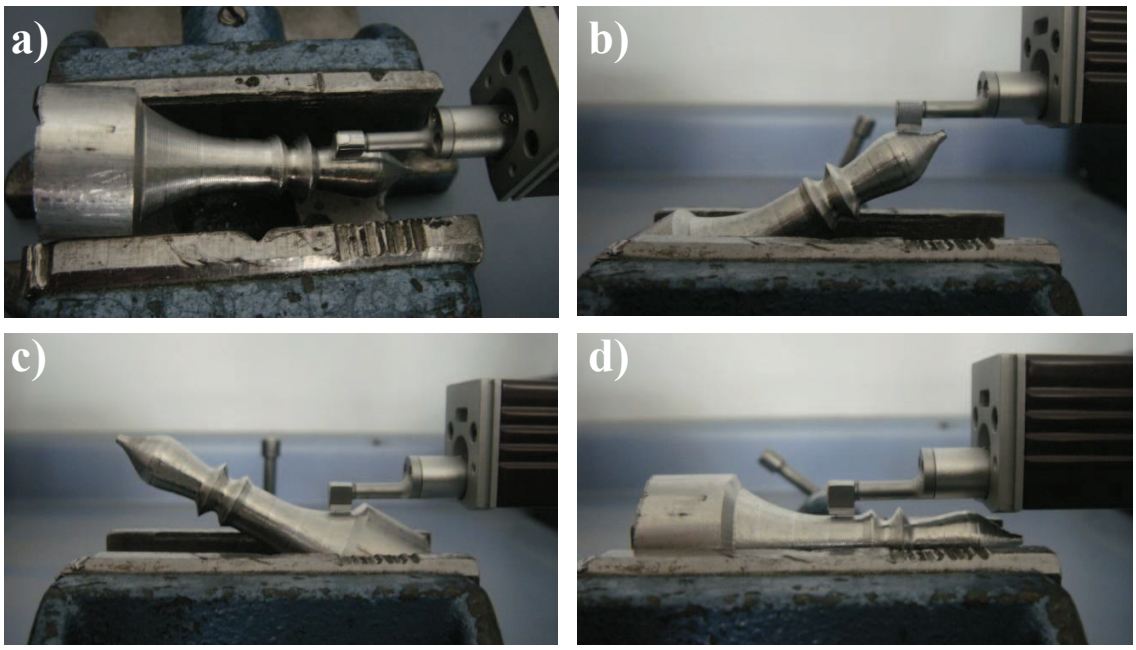

Figura 4.9 Regiones de medición.

Las mediciones fueron de 0.25 mm x 5 pasadas. Se hicieron las comparaciones de los valores promedio obtenidos donde se mostro un valor menor de rugosidad en las mediciones para la pieza con interpolaciones lineales, sin embargo, la diferencia no es muy considerable. Los resultados se muestran en la Tabla 4.1.

Tabla 4.1 Medición de rugosidad.

Región	Rugosidad (µm) Interpolaciones lineales	Rugosidad (µm) Interpolaciones NURBS
A	**1.42**	**1.66**
B	**1.07**	**1.34**
C	**0.77**	**0.93**
D	**1.08**	**1.15**

4.2 Seguimiento de una trayectoria

El programa para la realización de la trayectoria se encuentra constituido por 51 interpolaciones NURBS y 53 interpolaciones lineales, cada una de las interpolaciones

NURBS se encuentra conformada con 10 puntos de control, la velocidad de avance utilizada es de 100000cuentas/seg para los desbastes intermedios y de 30000cuentas/seg en los desbastes finales para cada una de las interpolaciones finales. El programa fue cargado y compilado en la parte del editor del software, el cual se muestra en la Figura 4.10.

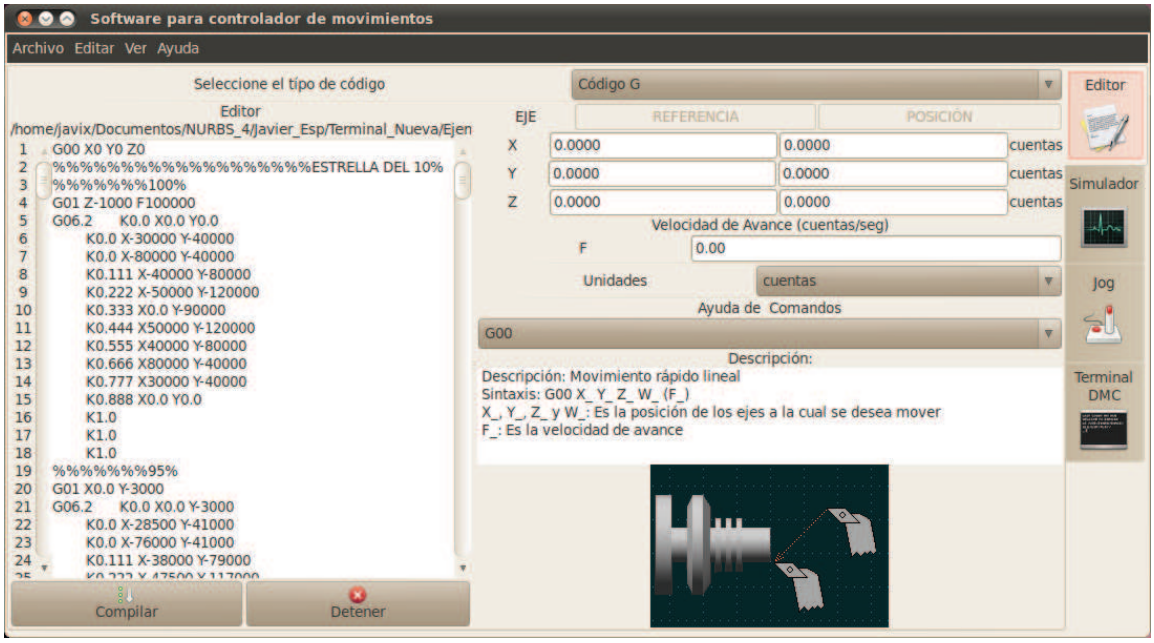

Figura 4.10 Editor del software con el programa.

Una vez que se realiza la compilación se carga el programa en la sección del simulador donde se puede observar trayectoria que se desea realizar en forma de isométricos seleccionando los ejes que se desean graficar para la simulación ó en la ejecución del programa. En la Figura 4.11 se muestra el seguimiento de la trayectoria en el plano XY durante la ejecución del programa.

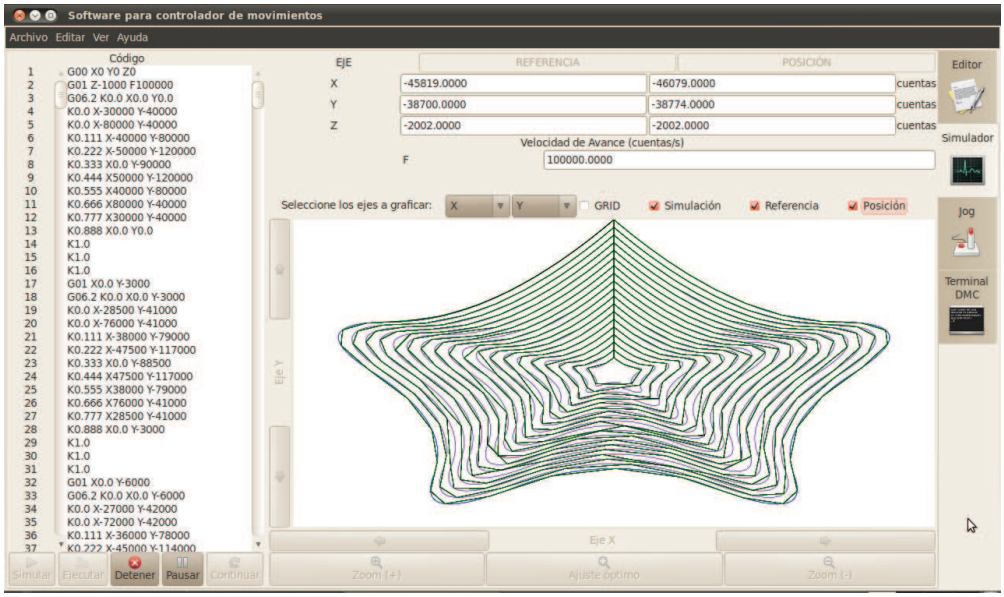

Figura 4.11 Simulador del software en el plano XY durante la ejecución del programa.

El software fue realizado para poder monitorear el cualquier plano durante la ejecución, en la Figura 4.12 se muestra el plano XZ durante la ejecución del programa. Como se muestra en la Figura 4.11 la velocidad de monitoreo del software es lenta a comparación de la velocidad con la que trabaja el hardware, sin embargo, para el análisis del seguimiento de la trayectoria se utilizaron los datos que se almacenan en la memoria del hardware. Estos datos pueden ser descargados al finalizar la ejecución de algún programa en el software, y puede modificarse la frecuencia de muestreo del hardware desde el software en los parámetros de movimiento.

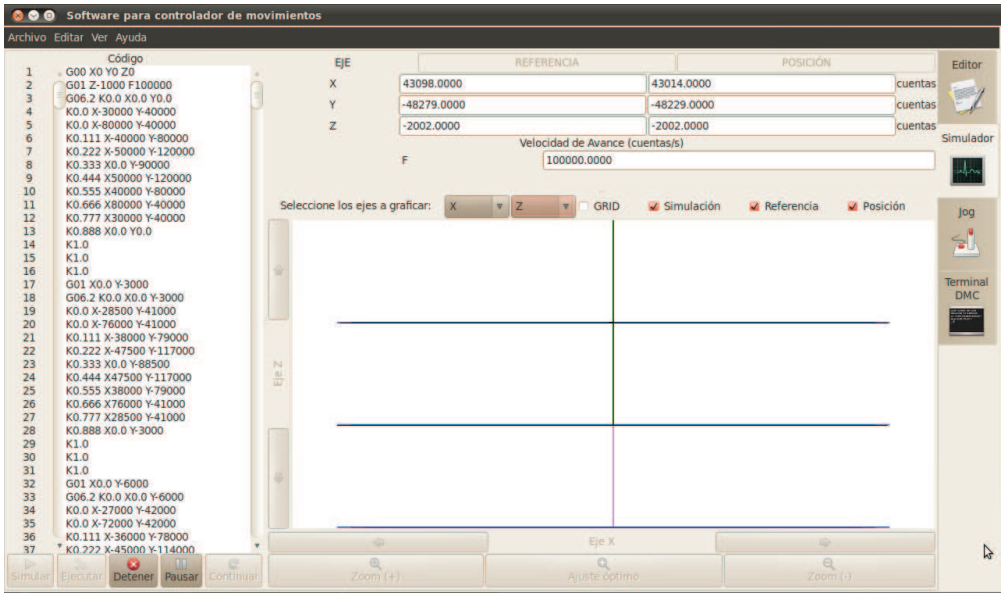

Figura 4.12 Simulador del software en el plano XZ durante la ejecución del programa.

Antes de realizar la ejecución se establecieron los parámetros de control y de movimiento desde el software. Los parámetros de control utilizados para la fresadora se muestran en la Figura 4.13.

Figura 4.13 Parámetros de control para fresadora.

En la Figura 4.14 se muestran los parámetros de movimiento utilizados para el maquinado en el torno.

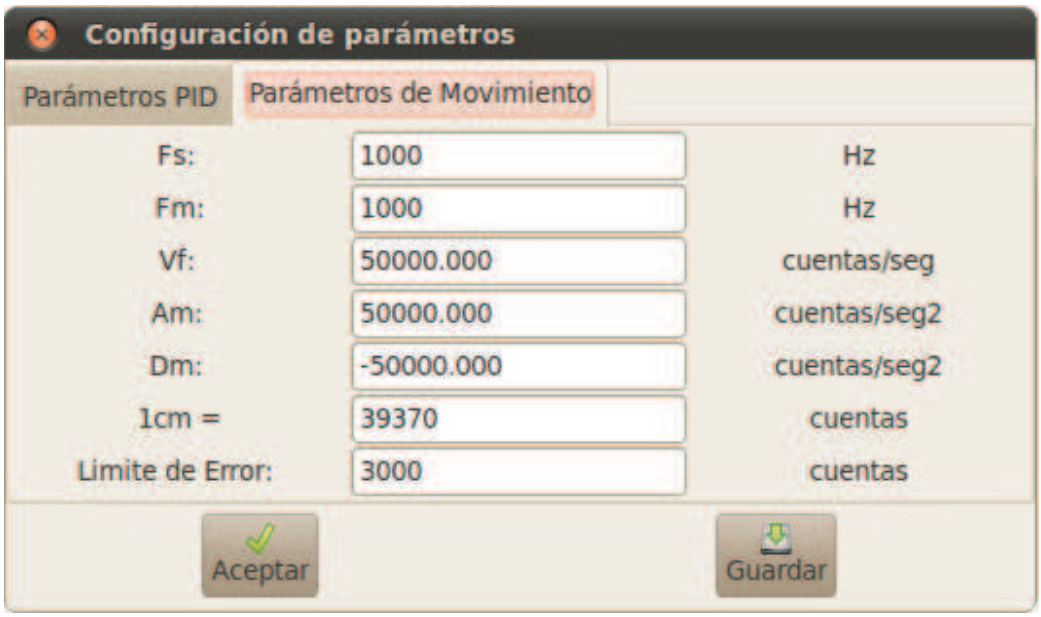

Figura 4.14 Parámetros de movimiento para fresadora.

La ejecución de la trayectoria tuvo una duración aproximada de 15 minutos. Al finalizar la ejecución se descargaron los datos del hardware y con ayuda de MATLAB se graficaron los datos de referencia y de posición obteniendo de esta manera la grafica de la Figura 4.15.

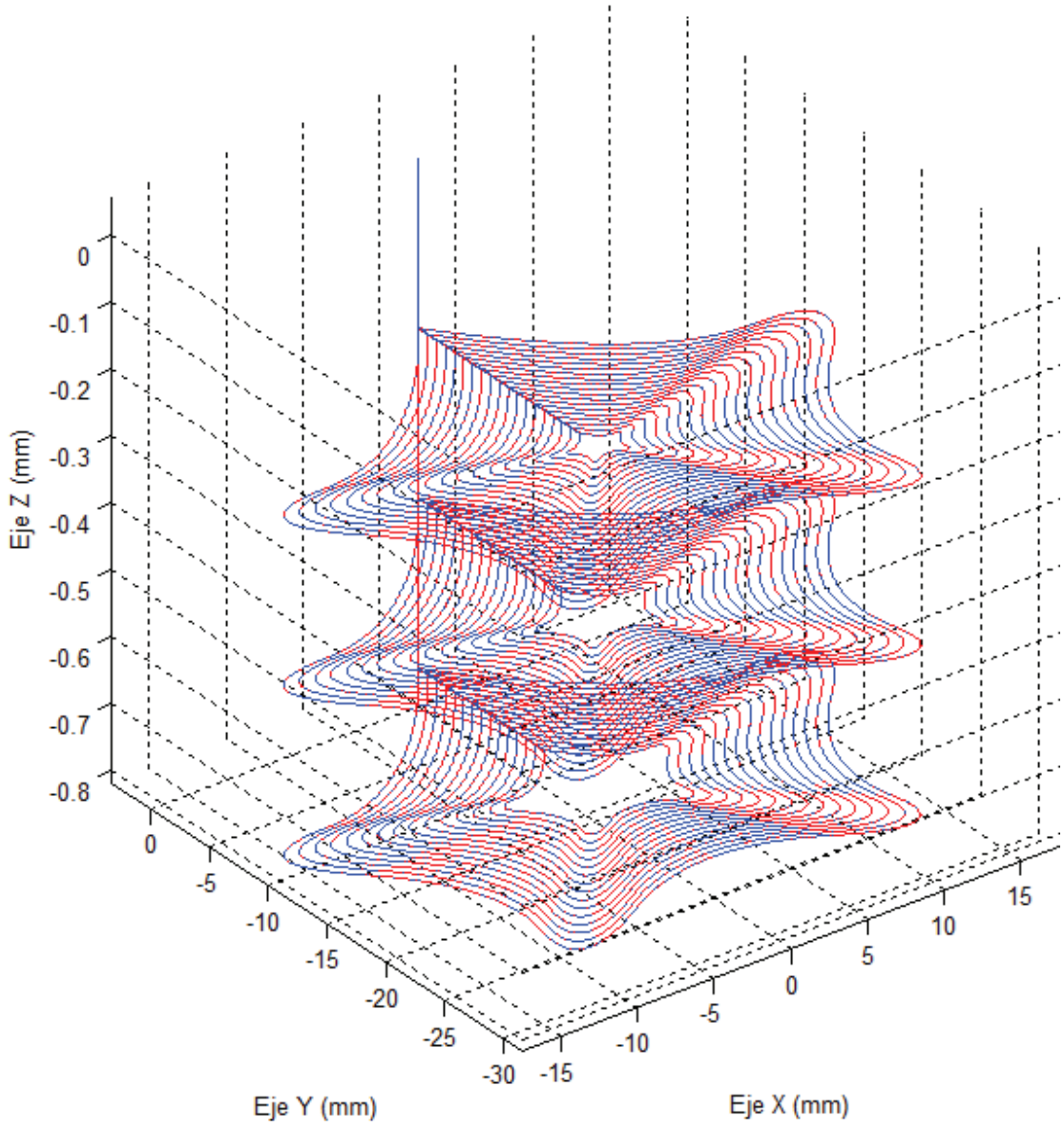

Figura 4.15 Gráfica de los datos de referencia y posición obtenidos del hardware.

En la Figura 4.15 se muestra el seguimiento de la trayectoria, sin embargo, no se aprecia el error que existe entre la referencia y la posición de tal forma que se muestra en la Figura 4.16 el error entre la referencia y la posición de cada uno de los puntos obtenidos, donde se obtuvo un error máximo de 23.4µm y un error promedio de 0.3175 µm.

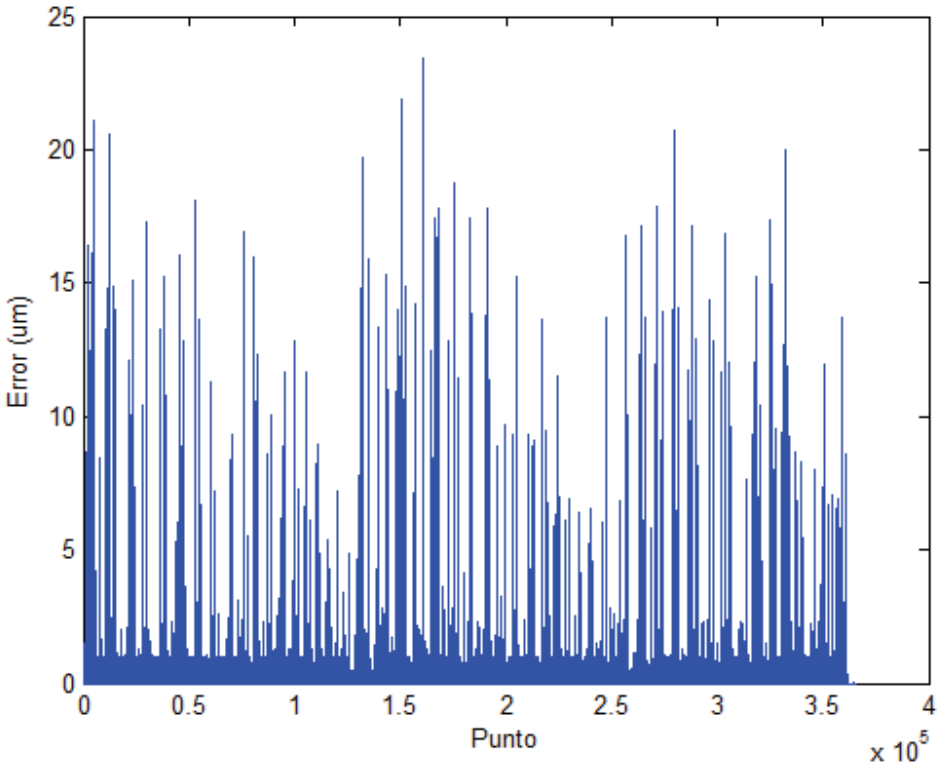

Figura 4.16 Error entre referencia y posición de cada uno de los puntos obtenidos.

Capítulo V.

CONCLUSIONES Y PROSPECTIVAS

En este trabajo se presentan el desarrollo e implementación de un software para sistemas de control de movimiento, enfocado principalmente al sistema MCUAQ4X el cual es un sistema de control desarrollado en la Universidad Autónoma de Querétaro, además se le da seguimiento al principal trabajo realizado por Morales (2010) que es una plataforma de co-diseño de sistemas embebidos y en este trabajo se orienta la plataforma a los sistemas de control de movimiento.

El software realizado fue capaz de cargar códigos G y DMC los cuales fueron compilados, ya que solo se trabajo con los comandos más utilizados, estos comandos son interpretados para generar una serie de curvas NURBS las cuales son enviadas al controlador MCUAQ4X para su ejecución en la máquina-herramienta y su simulación en el software. En caso de que se desee utilizar una tarjeta Galil se manda todo el código después de ser compilado para su ejecución y para la simulación en el software se siguen utilizando curvas NURBS.

En los programas CAD se pueden generar trayectorias suaves basadas en curvas NURBS, sin embargo, en la parte del CAM se genera una gran segmentación en la trayectoria, esto se debe a que el pre-procesador se encuentra programado para segmentar la trayectoria en códigos que puedan ser interpretados por el controlador que se desea utilizar y si el controlador no puede interpretar curvas paramétricas como B-spline ó NURBS el preprocesador los segmenta en una gran cantidad de interpolaciones lineales, lo cual se convierte en un problema por la gran cantidad de datos que se deben interpretar y transmitir al controlador.

En base a los resultados se muestra que el error de rugosidad entre códigos basados en interpolaciones lineales y códigos basados en interpolaciones NURBS no es muy significativo, sin embargo, el tiempo de maquinado es grande con las interpolaciones lineales comparado con el tiempo de maquinado con interpolaciones NURBS, esto es por la gran cantidad de datos que se deben de interpretar y transmitir.

En este trabajo también se muestra el buen funcionamiento que tiene el controlador MCUA4X, pues el error entre la referencia que se le envía al sistema y la posición a la que se llega es aceptable, sin embargo, el error puede disminuir disminuyendo la velocidad de avance del CNC.

Una de las principales aportaciones de este trabajo al sistema de control de movimiento desarrollado en la UAQ es la utilización de los códigos estándar más utilizados y que se encuentran regidos por las normas ISO6983 y FANUC G06.2 que son utilizados industrialmente. Mostrando de esta manera que los trabajos desarrollados son dirigidos a aplicaciones industriales con el fin de mejorar los procesos y disminuir costos. Sin embargo, el software también se encuentra dirigido hacia el estudio ya que es posible obtener los datos de referencia y posición para el análisis de la dinámica de los sistemas.

El software fue generado en una plataforma de Linux con una interfaz gráfica realizada en Glade generando de esta manera un software de arquitectura abierta. Además al trabajar bajo Linux se utiliza de una licencia de código abierto (GNU GLP) la cual tienen las siguientes características:

- Permite la copia, modificación y redistribución del software.
- Proporciona garantía de los derechos del usuario a la copia, modificación y redistribución del software.
- Como no tiene costo, tampoco ofrece garantías.
- Puede ser vendido y se puede cobrar por los servicios sobre el software.
- Cualquier patente sobre el mismo debe ser licenciada para el beneficio de todos.
- El software modificado no debe tener costo por la licencia.
- Tiene que incluir el código fuente.
- Los cambios en la licencia deben mantener ciertos términos generales.

Este trabajo dio fruto a dos artículos de congreso internacional, el primer artículo presenta una unidad de generación de trayectorias polinomiales a partir de código G aplicadas a sistemas de control de posición FPGA. El segundo artículo presenta un intérprete de curvas B-spline a partir del comando G06.2 para controlador de movimiento basado en FPGA aplicado a un torno CNC.

El presente trabajo presenta un software de arquitectura abierta sobre el cual se pretende que sigan creciendo sus módulos de aplicación un ejemplo podía ser en agregar un módulo para la sintonización e implementar más comandos para su interpretación.

REFERENCIAS

Alaniz Lumbreras Pedro Daniel. 2005. *Unidad de CNC con sistema de corte adaptable para la optimización de maquinado*. Tesis de Doctorado. Universidad Autónoma de Querétaro, Facultad de Ingeniería.

Colin Robles J. Ángel. 2006. *Descripción VHDL de los bloques funcionales de un controlador digital PID.* Tesis de Maestría. Universidad Autónoma de Querétaro, Facultad de Ingeniería.

Delta Tau. 2005. *Data system Corpotration. Turno PMAC-Lite PCI Data Sheet* . Delta Tau Data System.

De Santiago Pérez J Jesús, Osornio-Ríos Roque Alfredo, Romero-Troncoso René de Jesús, Herrera-Ruiz Gilberto, Delgado-Rosas Manuel. 2008. *DSP Algorithm for the Extraction of Dynamics Parameters in CNC Machine Tool Servomechanisms from an Optical Incremental Encoder*. International Journal of Machine Tools & Manufacture. 48: 1328-1334.

De Santiago Pérez J Jesús. 2010. *Algoritmos para la interpolación de trayectorias con optimización de dinámica de movimiento en los servomotores.* Tesis de Doctorado. Universidad Autónoma de Querétaro. Facultad de Ingeniería.

Erkorkmaz Kaan, Altintas Yusuf. 2001. *High speed CNC system design. Part I: jerk limited trajectory generation and quintic spline interpolation*. International Journal of Machine Tools & Manufacture. 41:1323-1345.

Erkorkmaz K, Wong W. 2007. *Rapid identification technique for virtual CNC drives*. International Journal of Machine Tools and Manufacture. 47:1381-1392.

Femat Díaz Aurora. 2004. *Desarrollo de la ingeniería de software para la implementación de un CNC*. Tesis de Maestría. Universidad Autónoma de Querétaro, Facultad de Ingeniería.

Galil Motion Control. 2010. DMC-18x2 Series. PCI Bus Econo Series, 1-4 axes.

García Quijada Manuel. 2006. *Sistema de supervisión de ruptura en herramientas de corte para tornos de control numérico computarizado*. Tesis de grado Maestría. Universidad Autónoma de Querétaro, Facultad de Ingeniería.

Gómez Loenzo Roberto Augusto. 2007. Sistema de control numérico en tiempo real de arquitectura abierta reconfigurable. Tesis de grado Maestría. Universidad Autónoma de Querétaro, Facultad de Ingeniería.

Hernández Vargas Mariano. 2006. *Diseño para implementación en hardware de un control digital PID*. Tesis de Maestría. Universidad Autónoma de Querétaro, Facultad de Ingeniería.

Jönsson A, Wall J, Broman G. 2005. *A virtual machine concept for real-time simulation of machine tool dynamics*. International Journal of Machine Tools and Manufacture. 45: 795-801.

Lin Ming-Tzong, Tsai Meng-Shiun, Yau Hong-Tzong. 2007. *Development of a dynamics-based NURBS interpolator with real-time look-ahead algorithm*. International Journal of Machine Tools and Manufacture. 47: 2246-2262.

Macfarlane Sonja, Croft Elizabeth A. 2003. *Jerk-bounded manipulator trajectory planning: design for real time applications*. IEEE Transactions on Robotics and Automation. 19: 42-52.

Mechanical Engineering - Worcester Polytechnic Institute. 2000. Institute Road, Worcester, MA 01609-2280. http://www.me.wpi.edu

Mejía Ugalde Mario. 2008. *Desarrollo de un control numérico computarizado para un torno*. Tesis de Maestría. Universidad Autónoma de Querétaro. Facultad de Ingeniería.

Muller Matthias, Erdos Gabor, Xirouchakis Paul. 2004. *High accuracy spline interpolation for 5-axis machining*. Computer Aided Design. 36:1379-1393.

Morales V. Luis. 2007. *Unidad de USB de Control de Posición y Generación de Perfiles para un Intercambiador Automático de Herramientas*. Tesis de Maestría. Universidad Autónoma de Querétaro.

Morales Velázquez Luis. 2010. *Diseño de plataforma hardware-software para el desarrollo de aplicaciones industriales basadas en FPGA*, Tesis de Doctorado. Universidad Autónoma de Querétaro. Facultad de Ingeniería.

Osornio Ríos Roque Alfredo. 2004. *Diseño y construcción de una tarjeta controladora de 3 ejes*. Tesis de Maestría. Universidad Autónoma de Querétaro. Facultad de Ingeniería.

Osornio-Ríos R A, Romero Troncoso R J, Herrera-Ruiz G, Castaneda-Miranda R. 2007 *Computationally efficient parametric analysis of discrete-time polynomial based acceleration-deceleration profile generation for industrial robotics and CNC machinery*, Mechatronics, 511-523.

Osornio-Ríos R A, Romero-Troncoso R J, Herrera-Ruiz G and Castañeda-Miranda R. 2009. *FPGA implementation of higher degree polynomial acceleration profiles for peak jerk reduction in servomotors*. Robot Cim-Int Manufac, 25:379-392.

Paluszny Marco, Prautzsch Hartmut, Boehm Wolfgang. 2005. *Metodos de Bezier y B-splines*. Universitatsverlag Karlsruhe. Karlsruhe.

Panahi Issa M. S,Toliyat Hamid A. 2006. *An efficient processor-based online generation of time-optimal trajectories*. IEEE Transactions on Control Systems Technology. 14: 966-973.

Park Jinho, Nam Sungho, Yang Minyang. 2005. *Development of a real-time trajectory generator for NURBS interpolation based on two-stage interpolation method*. International Journal of Advanced Manufacturing Technology. 26: 359-365.

Rivera Guillen Jesús Rooney. 2007. *Perfiles Polinomiales de Movimiento para Máquinas CNC*. Universidad de Guanajuato. FIMEE.

Rivera Guillen Jesús Rooney. 2010. *Algoritmos óptimos en FPGA para controlar la dinámica de movimiento en maquinaria CNC*. Tesis de Doctorado. Universidad Autónoma de Querétaro. Facultad de Ingeniería.

Romero Troncoso René de Jesús, 2004. *Procesamiento de señales para la detección de ruptura de herramientas en sistemas de manufactura por control numérico computarizado.* Tesis de Maestría. Universidad Autónoma de Querétaro, Facultad de Ingeniería.

Smid Peter. 2003. *CNC Programming Handbook: A Comprehensive Guide to Practical CNC Programming* (2nd Ed.). Industrial Press Inc., New York.

Trejo Hernández Miguel. 2006. *Módulo de maquinado y monitoreo, aplicando control difuso en proceso de torneado*. Tesis de Maestría. Universidad Autónoma de Querétaro. Facultad de Ingeniería.

Tsai M-C, Cheng C-W, Cheng M-Y. 2003. *A real-time NURBS surface interpolator for precision three-axis CNC machining*. International Journal of Machine Tools & Manufacture. 43:1217-1227.

Ugalde Estrella Javier. 2009. *Unidad de trayectorias polinomiales a partir de código G aplicados a sistemas de posición FPGA*. Tesis de Licenciatura. Universidad Autónoma de Querétaro. Facultad de Ingeniería.

http://www.inegi.org.mx/inegi/contenidos/espanol/prensa/comunicados

http://www.gtkmm.org, fecha de consulta (Abril, 2010).

http://www.gtk.org, fecha de consulta (Abril, 2010).

http://www.ni.com/motion/esa, fecha de consulta (Abril, 2010).

APÉNDICE

CIINDET 2010

VIII Congreso Internacional sobre Innovación y Desarrollo Tecnológico,
24 al 26 de octubre de 2010, Cuernavaca Morelos, México

Unidad para generación de trayectorias polinomiales a partir de código G aplicadas a sistemas de posición FPGA.

J. Ugalde Estrella, student member, IEEE, J. Benitez Rangel, L. Morales Velázquez, student member, IEEE.

Resumen: La utilización de las máquinas CNC en la industria es de gran importancia, su aplicación abarca casi todos los procesos de manufactura, siendo necesario desarrollar investigación relacionada con el control de dichas máquinas. En este artículo se describe la metodología para el desarrollo de una unidad de generación de trayectorias para un controlador de posicionamiento basado en FPGA aplicado a maquinaria CNC. Se presentan los algoritmos para la construcción de dicha unidad y su implementación en una plataforma libre. El objetivo del trabajo es crear una estandarización de los códigos G con el controlador en hardware propio basado en FPGA. Se presentan pruebas de la unidad en conjunto con el controlador de posición FPGA y su comparativa con un controlador comercial aplicados a maquinado en un torno reconvertido a CNC mostrándose las ventajas de la investigación mediante los resultados obtenidos.

Abstract: The use of CNC machine tools in industry is very important. CNC machines are involved in most of manufacturing processes, being necessary to develop related research in controlling these machines. This work describes the methodology to develop one trajectory generation unit as reference for a FPGA-based position controller applied to CNC machinery. The algorithms to build the developed unit and its implementation are presented in a free platform. The objective of the work is to create a G code standardization using the developed proprietary FPGA hardware. The system is tested by using the developed trajectory unit and the FPGA position controller and compared with a commercial controller in a retrofitted to CNC lathe, this shows the advantages of this investigation through the obtained results.

Keywords: CNC, FPGA, trayectorias polinomiales, controlador.

Javier Ugalde Estrella. (javi_star67@hotmail.com)
Juan P. Benitez Rangel. (benitez@uaq.mx)
Universidad Autónoma de Querétaro, Centro universitario, Cerro de las Campanas s/n, Querétaro, Querétaro, México. www.uaq.mx
Luis Morales Velázquez (lmorales@ieee.org)
Universidad Autónoma de Querétaro, Centro universitario, Cerro de las Campanas s/n, Querétaro, Querétaro, México. www.uaq.mx

Introducción

Las máquinas herramientas de CNC (Control Numérico Computarizado) se encuentran extensamente propagadas en aplicaciones industriales, muestra de ello es que de acuerdo al censo del INEGI del 2009 un 12% del total de empresas existentes en el país se dedican a la manufactura y requieren en sus procesos este tipo de maquinaria. Actualmente existen diversas alternativas en máquinas de Control Numérico que pueden ser adquiridas por la industria mexicana y que son proporcionadas por empresas como Fagor, Siemens, Fanuc, Num, Dina, Mitsubishi, Heidenhain, entre muchas otras, todas ellas extranjeras con tecnología completamente propietaria lo cual genera costos excesivos imposibles de pagar por el 99.8% de las empresas que constituyen la categoría de las Pequeñas y Medianas empresas (PyMES) complicando su supervivencia y competencia en el mercado.

La dependencia tecnológica es un problema no solo nacional, pues éste afecta a la industria mundial, por tal motivo se han desarrollado varias investigaciones buscando solucionar los problemas de control en maquinaria CNC. Las investigaciones se han enfocado a la generación de metodología para control, generación de referencias y dinámica, que en muy pocos casos se han implementado en una máquina. Por otro lado los controladores comerciales accesibles en costo generan pequeñas discontinuidades sobre la dinámica debido al generador de perfil para posición del controlador el cual está basado generalmente sobre un perfil trapezoidal de velocidad. El parámetro para medir la discontinuidad y suavidad dinámica es conocido como jerk y es definido como la razón de cambio de la aceleración. Erkorkmaz y Altintas [1] afirman que niveles de Jerk altos generan mayor cantidad de vibraciones. Por lo anterior, uno de los puntos que se ha considerado en la dinámica de las máquinas es alimentar al controlador con perfiles de posición suficientemente suaves, de tal manera que reduzcan los niveles de jerk, minimizando discontinuidades y evitando la generación de vibraciones, esto se ha logrado mediante diversas técnicas tales como: filtros digitales, técnicas de

interpolación, curvas S, NURBS (*Non-Uniform Rational B-Spline*; spline-base rotacionales no uniformes), polinomios de orden superior, polinomios a trazos, etc. Por otro lado también se ha estudiado diversas técnicas para la limitar el jerk como la propuesta de Yih [2] quien presentó el diseño de un filtro lineal de tercer orden para limitación de jerk en máquinas CNC para mejorar el perfil de posición del servomotor. Gasparetto y Zanotto [3] propusieron una nueva metodología para planificar trayectorias suaves basada sobre un perfil polinomial flexible de quinto grado para un jerk cuadrático. Liu [4], propuso un método para la generación de trayectoria de referencia de movimiento suave y su aplicación en tiempo real, tomó en cuenta las restricciones máximas dadas en velocidad, aceleración y jerk. A pesar de existir un gran número de técnicas para control de dinámica, la primera limitante con la que se han enfrentado en su implementación son las altas cargas computacionales y su poca reconfigurabilidad en el hardware para lograr su integración. La solución a este problema ha sido implementar los algoritmos en dispositivos FPGA (*Field Programmable Gate Array*, arreglo de compuertas programables en campo), que permiten incrementar la capacidad de cómputo, velocidad de procesamiento y la integración de módulos para desarrollar el control en tiempo real. Ejemplo de dichas aplicaciones es el trabajo de Jeon y Kim [5] quienes desarrollaron un generador de perfiles basado en aceleración y desaceleración trapezoidal implementado en un FPGA para aplicaciones de robóticas industriales. Girau y Boumaza [6] presentaron una arquitectura embebida para resolver el problema de navegación en robótica basada en FPGA que computa trayectorias. Los trabajos más relacionados con el control y la dinámica de maquinaria CNC basados en FPGA son los de Osornio et al. [7] quienes presentan un controlador Proporcional-Integral-Derivativo (PID) basado en un FPGA aplicado a maquinas CNC de alta velocidad para lograr obtener los tiempos de actualización de control requeridos. Posteriormente Osornio et al. [8] en otro trabajo integran el módulo PID con un nuevo desarrollo de un generador de perfiles basado en polinomios de alto grado limitando la dinámica del jerk de máquinas CNC y robótica para mejorar los procesos de maquinados. La novedad de su sistema fue el desarrollo de un algoritmo recursivo multigrado para la evaluación polinomial, computacionalmente eficiente, en generación de perfiles y la implementación a bajo costo de la estructura digital en FPGA.

A pesar de haber investigaciones de este tipo de leyes de control y generadores de referencias en hardware, los trabajos hasta ahora reportados en FPGA no han sido estandarizados para trabajar con códigos estándar que manejan la mayoría del software CAD (Diseño Asistido por Computadora)/CAM (Maquinado Asistido por Computadora) comercial como códigos G (estándar ISO 6983), es decir, se han realizado solo pruebas funcionales de dichos módulos FPGA.

En este trabajo se presenta una unidad en software para estandarizar el controlador y generador de perfiles en FPGA basados en las metodologías [7-8] con los códigos G, buscando minimizar los problemas ocasionados por la segmentación de trayectorias en dichos códigos. Para la realización de esta unidad se parte del CAD realizando la representación de curvas las cuales son proporcionadas al CAM que realiza la segmentación de curvas dando como resultado el código NC de acuerdo al ISO 6983, el cual proporciona la segmentación de líneas y arcos que debe realizar el maquinado. Este código se leerá y analizará de tal forma que se pueda realizar la segmentación y rotación de coordenadas, que se interpola mediante el método de mínimos cuadrados, para finalmente obtener una función polinomial que se transmite al controlador CNC basado en FPGA. Las pruebas de la unidad se llevan a cabo en un torno CNC y se demuestra la eficacia de la metodología mediante la experimentación.

Materiales y Métodos

Ming et al. [9] afirma que los CNC convencionales solo proporcionan las interpolaciones lineales y redondas, por lo que para la fabricación los sistemas CAM (Manufactura Asistida por Computadora) tienen que *segmentar* las curvas paramétricas o aparecer en muchos bloques pequeños de NC lineales y redondos. Este acercamiento padece los siguientes problemas: (1) la fluctuación y discontinuidad de velocidad de avance, (2) la discontinuidad de aceleración que produce grandes jerk, que podría causar la vibración del sistema y reducir la calidad del mecanizado, (3) no podrían planearse la velocidad de avance para los segmentos pequeños, y así el mecanizado de gran velocidad no puede lograrse, (4) resultan confinando pequeños errores de cordón en pequeños segmentos y causa un volumen grande de transmisión de los datos que podría

cargar excesivamente el sistema de CNC. Por lo que en este trabajo únicamente se utilizan movimientos rápidos (G00), interpolaciones lineales (G01) e interpolaciones circulares (G02 y G03).

El proceso general para la comunicación con el sistema de control de posicionamiento FPGA basado en [7-8] se muestra en la **Figura 1** donde se observa que la unidad para generación de trayectorias polinomiales consta de cuatro componentes esenciales las cuales son el *lector y codificador*, el *convertidor*, la *segmentación y rotación,* y el *método de mínimos cuadrados*.

El controlador requiere 14 parámetros, los primeros seis son los valores de las constantes de los polinomios, son seis porque el controlador está diseñado para procesar polinomios hasta de quinto grado, el último valor de la coordenada X rotada del polinomio, la velocidad de avance mínima dividida entre 60, aceleración, desaceleración, una variable la cual depende del ángulo de rotación, el valor de la continuidad con el siguiente polinomio (1 si existe y 0 si no existe), el valor del punto donde inicia el polinomio sin rotar, es decir, el valor de la coordenada X y el valor de la coordenada Z donde comienza el polinomio. En el primer renglón requiere el número de polinomios totales y la frecuencia de muestreo, los demás valores se ponen en cero.

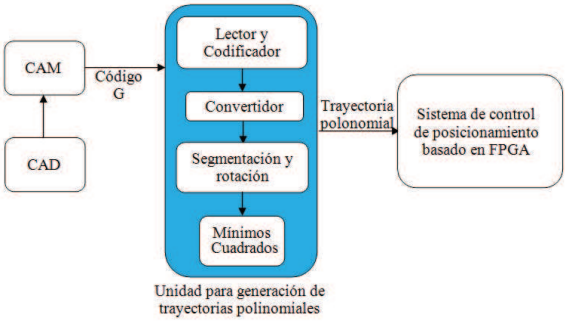

Fig. 1 Diagrama General.

Lector y codificador

La función de lector permite identificar el tipo de comando que se encuentra leyendo, los parámetros que ofrece depende del tipo de comando ya sea una interpolación lineal ó un movimiento rápido, que única mente brinda como información las coordenadas (*X, Y y Z*) y la velocidad de avance (*F*), ó interpolación circular la cual además de las coordenadas finales de la circunferencia (*X, Y y Z*) también brinda las coordenadas del centro de la circunferencia (*I, J y K*). Para la realización de este módulo se desarrollaron dos subrutinas *buscar* la cual proporciona la posición del carácter que se desea buscar, en caso de que el carácter no se encuentre manda el valor de -1. Esta posición es enviada a *sacarnum* que devuelve el número que se encuentra enseguida del carácter buscado, en caso de que el valor sea -1 se queda con el valor que tenía anteriormente y de esta manera se sabe cuál es el comando que se está leyendo. En la **Figura 2** se muestra el diagrama de flujo del *lector y codificador* donde se muestra las rutinas que se deben realizar para leer y codificar el código CN. Los datos de entrada son el nombre del archivo que se va a leer, el nombre del archivo que se va a crear, así como el número de cuentas que equivale a 1mm.

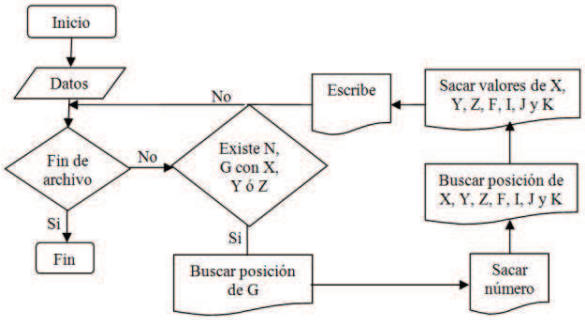

Fig. 2 Diagrama de flujo del *lector y codificador*.

El proceso busca renglón por renglón un comando (G00, G01, G02 y G03). Se busca la posición en la cual se encuentra ubicada la letra G, esta posición es enviada a *sacarnum* que devuelve el número que se encuentra enseguida del carácter buscado, obteniendo de esta forma el tipo de comando. Una vez obtenido el comando se procede a adquirir los parámetros necesarios para cada uno de los comandos, se busca la posición de *X, Y, Z, F, I ó K* dependiendo del comando y se obtiene el valor numérico. En caso que no se encuentre escrito un valor de éstos se registra el último valor que se obtuvo.

Convertidor

En caso de que sea una interpolación circular se realiza el proceso del *convertidor*, en donde se convertirá la interpolación circular a una serie de interpolaciones lineales. Pero para esta unidad no se convertirán todas las interpolaciones directamente, se convierte una por una, es decir, se realiza una interpolación lineal y entra

al siguiente proceso como si se estuviera leyendo una serie de interpolaciones lineales. Los comandos G02 y G03 son interpolaciones circulares, donde es necesario ubicar el punto central del círculo el cual está dado por los valores que se encuentran en *I* y *K*, estos valores son desplazados a partir del punto anterior, es decir, el punto que fue dado por la instrucción anterior ya que en la instrucción leída se encuentra el punto al que se desea desplazar. Para realizar el proceso de la conversión es necesario realizar una serie de pasos los cuales son especificados en el diagrama de flujo de la **Figura 3**.

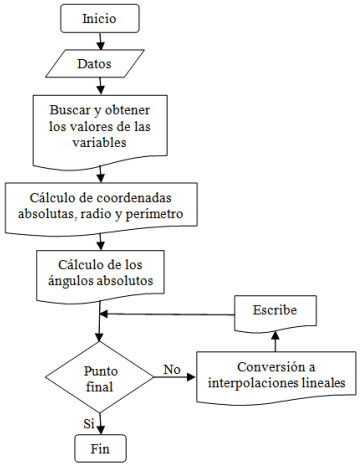

Fig. 3 Diagrama de flujo del *convertidor*.

En la **Figura 4** se muestra una interpolación circular G02 donde las coordenadas absolutas de los puntos *P* y P_1 ya se conocen pues P_1 es el punto de inicio de donde parte la circunferencia el cual está dado por las coordenadas X_1 y Z_1 que son las coordenadas en las que se encuentra la herramienta antes de la instrucción de interpolación circular y *P* esta dado por las coordenadas *X* y *Z* que son las coordenadas a las que se desea llegar y estas se encuentran en la instrucción de interpolación circular. Para el punto P_c esta dado por las coordenadas *I* y *K* a partir del punto de inicio de la interpolación, de tal forma que para encontrar sus coordenadas absolutas en el plano *X*-*Z* es necesario sumarle a *I* el valor de X_1 y a *K* el valor de Z_1.

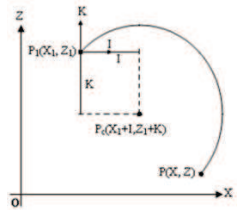

Fig. 4 Coordenadas absolutas.

Una vez obtenidas las coordenadas de los puntos en coordenadas absolutas, se calcula el radio r_c de la circunferencia mediante la ecuación 1.

$$r_c = \sqrt{K^2 + I^2} \quad (1)$$

Se calculan los ángulos de los puntos con respecto a P_c para calcular el perímetro de la trayectoria que se va a realizar. En la **Tabla 1** se muestran las ecuaciones para cada caso donde θ_1 es el ángulo.de P_1 con respecto a P_c y θ_2 es el ángulo.de *P* con respecto a P_c.

Tabla 1 Ecuaciones para el cálculo del perímetro.

Sentido	Horario	Antihorario
$\theta_1 > \theta_2$	$r_c(\theta_1 - \theta_2)$	$r_c(2\pi + \theta_2 - \theta_1)$
$\theta_2 > \theta_1$	$r_c(2\pi + \theta_1 - \theta_2)$	$r_c(\theta_2 - \theta_1)$

Para realizar la conversión en sentido horario ó antihorario, se calcula un ángulo de desplazamiento $(\Delta\theta)$ que ayuda a tener una distancia fija entre puntos, el cual se obtiene a partir del radio de la circunferencia (r_c) y una distancia (d) que se propuso de 100 cuentas, por lo tanto la ecuación es la siguiente:

$$\Delta\theta = \frac{d}{r_c} \quad (2)$$

En la **Figura 5** se muestra en forma gráfica el proceso que se realiza para la transformación de la interpolación circular en sentido horario y antihorario, donde P_i es el punto inicial de la circunferencia, P_f es el punto final, P_c es el punto central de la circunferencia, θ_1 es ángulo absoluto del punto P_i con respecto de los ejes del punto P_c, *S* es la distancia de la circunferencia que existe entre el punto P_i y el punto P_f, r_c es el radio de la circunferencia, d es la distancia del ángulo de desplazamiento y $\Delta\theta$ es el ángulo de desplazamiento.

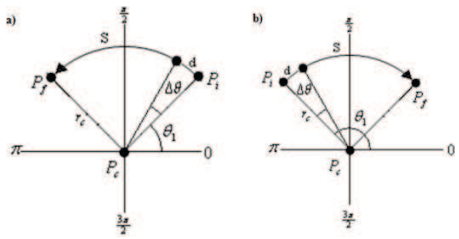

Fig. 5 Convertidor a) Antihorario, b) Horario.

El ángulo de avance aumentará hasta llegar al punto final y el cálculo de las coordenadas se realiza a partir del punto central de la circunferencia que tiene como coordenadas *Xc* y *Zc*, la ecuación para obtener los puntos de interpolación lineal es la siguiente:

Para sentido horario

$$X = X_c + r_c \cos(\theta_1 - \Delta\theta)$$
$$Z = Z_c + r_c \sin(\theta_1 - \Delta\theta) \quad (3)$$

Para sentido antihorario

$$X = X_c + r_c \cos(\theta_1 + \Delta\theta)$$
$$Z = Z_c + r_c \sin(\theta_1 + \Delta\theta) \quad (4)$$

Segmentación y rotación

Una trayectoria está compuesta por una serie de pequeños trazos, éstos pueden formar una serie de polinomios, pero un solo polinomio no podría realizar toda una trayectoria ya que ésta puede llegar a tener dos valores de *Z* para un solo valor de *X*. Con los módulos del *lector* y el *convertidor* solo se está reconstruyendo la trayectoria del maquinado, utilizando únicamente interpolaciones lineales, es decir, se tienen una serie de puntos con los cuales se puede recontruir el maquinado. Es necesario identificar los puntos que pueden ser contenidos en un mismo polinomio, por lo que se requiere de un módulo de *segmentación* el cual proporciona un ángulo de rotación para cada trazo del maquinado.

El ángulo de rotación de cada trazo depende del ángulo que forma dicho trazo, el ángulo tangencial se calcula a partir del segundo punto. Para ejemplificar esta función se muestra en la **Figura 6** dos puntos que forman una recta, además se muestra el ángulo de inclinación.

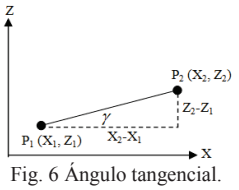

Fig. 6 Ángulo tangencial.

En la **Figura 6** se tienen dos puntos, P_1 que es el punto inicial de la recta y P_2 que es el punto final, gamma (γ) es el ángulo formado por la recta el cual es calculado de la siguiente forma

$$\gamma = \tan^{-1}\left(\frac{Z_2 - Z_1}{X_2 - X_1}\right) \quad (5)$$

El ángulo de rotación que le corresponde al trazo depende de la **Tabla 2**. Para pertenecer a un mismo polinomio una serie de trazos éstos deben de tener el mismo ángulo de rotación, esto garantiza que el polinomio *P(x)* tenga un solo valor para *x*

Tabla 2 Ángulo de rotación

Ángulo de inclinación	Ángulo de rotación	Valor para el controlador
$\frac{-\pi}{4} < \gamma \leq \frac{\pi}{4}$	**0**	0
$\frac{\pi}{4} < \gamma \leq \frac{3\pi}{4}$	$\frac{-\pi}{2}$	3
$\frac{3\pi}{4} < \gamma \leq \pi$	π	2
$\frac{-3\pi}{4} < \gamma \leq \frac{-\pi}{4}$	$\frac{\pi}{2}$	1
$\frac{-3\pi}{4} \leq \gamma \leq -\pi$	π	2

El ángulo de rotación es el ángulo necesario para rotar todos los trazos que pueden componer un polinomio de tal manera que queden entre el primer y cuarto cuadrante, además de que la división debe ser simétrica. Para la parte de rotación se realizan una serie de puntos los cuales pueden conformar un solo polinomio, aplicándole a la serie de puntos el método de mínimos cuadrados, para ello se requiere que el polinomio comience en el punto (0,0), por lo tanto las ecuaciones de rotación que se aplican a las coordenadas *X* y *Z* de todos los puntos son las siguientes:

$$X_r = (X_p - X_i)\cos\theta + (Z_p - Z_i)\sin\theta \quad (6)$$
$$Z_r = (Z_p - Z_i)\cos\theta - (X_p - X_i)\sin\theta \quad (7)$$

Donde X_r y Z_r son las coordenadas rotadas, X_p y Z_p son las coordenadas del punto que se desea rotar, X_i y Z_i son las coordenadas donde inicia el polinomio y θ es el ángulo de rotación que le corresponde a la serie de puntos del polinomio. El primer punto es (0,0) puesto que X_i es igual a X_p y Z_i es igual a Z_p.

Lo que se realiza con este proceso se puede representar gráficamente en la **Figura 7**, considerando una parte del maquinado, es decir, una serie de trazos los cuales se encuentra de la forma de la **Figura 7 a)**. A esta

trayectoria se le podría aplicar directamente un polinomio, sin embargo, si la trayectoria sigue creciendo podría tener dos valores del eje *Z* para un solo valor del eje *X*, por lo que se le aplica el proceso de *rotación y segmentación*, de tal forma que es posible reconstruir la trayectoria en la forma como se muestra en la **Figura 7 b)**. El proceso de rotación facilita la programación del algoritmo de mínimos cuadrados debido a que todos los puntos se ubican en el semiplano positivo.

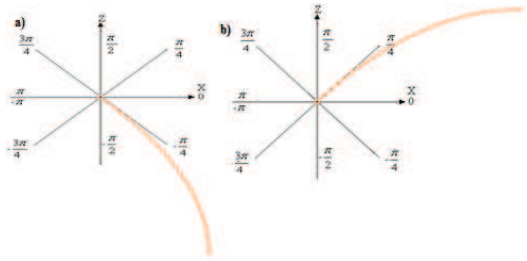

Fig. 7 a) Trayectoria propuesta, b) Trayectoria rotada.

Mínimos cuadrados.

El método de mínimos cuadrados esta realizado considerando que en ocasiones la información tiene errores como cuando proviene de medidas físicas, se recomienda pasar un polinomio de aproximación por los puntos dados o cerca de ellos, donde se considera un *error mínimo*. Para el desarrollo de este trabajo se utilizó el método de mínimos cuadrados de donde se obtiene la ecuación siguiente.

$$\begin{bmatrix} n & \sum_{i=1}^{n} x_i & \cdots & \sum_{i=1}^{n} x_i^{\,k} \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^{\,2} & \cdots & \sum_{i=1}^{n} x_i^{\,k+1} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{n} x_i^{\,k} & \sum_{i=1}^{n} x_i^{\,k+1} & \cdots & \sum_{i=1}^{n} x_i^{\,2k} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_k \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \\ \vdots \\ \sum_{i=1}^{n} x_i^{\,k} y_i \end{bmatrix} \quad (8)$$

Donde n es el número de puntos; k el grado del polinomio; x y y son las coordenadas correspondientes de cada uno de los puntos y las α son las constantes del polinomio.

Para forzar que un polinomio comience en cero la constante debe tener un valor de cero, esto es:

$$P(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_k x^k$$
$$P(0) = a_0 = 0 \quad (9)$$

Mientras que para suavizar el cambio de los polinomios, la derivada del polinomio evaluada en cero se obliga a que tome el valor de la derivada del polinomio anterior ó el valor de la pendiente de la última recta, esto expresado matemáticamente es:

$$P'(x) = a_1 + 2a_2 x + \cdots + k a_k x^{k-1}$$
$$P'(0) = a_1 = \alpha \quad (10)$$

Donde α toma el valor de la derivada del polinomio anterior ó el valor de la pendiente de la última recta. Por lo tanto la matriz del método de mínimos cuadrados queda conformada de la siguiente manera:

$$\begin{bmatrix} \sum_{i=1}^{n} x_i^4 & \sum_{i=1}^{n} x_i^5 & \cdots & \sum_{i=1}^{n} x_i^{k+2} \\ \sum_{i=1}^{n} x_i^5 & \sum_{i=1}^{n} x_i^6 & \cdots & \sum_{i=1}^{n} x_i^{k+3} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{n} x_i^{k+2} & \sum_{i=1}^{n} x_i^{k+3} & \cdots & \sum_{i=1}^{n} x_i^{2k} \end{bmatrix} \begin{bmatrix} a_2 \\ a_3 \\ \vdots \\ a_k \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i x_i^2 - a_0 \sum_{i=1}^{n} y_i x_i^2 - a_1 \sum_{i=1}^{n} y_i x_i^3 \\ \sum_{i=1}^{n} y_i x_i^3 - a_0 \sum_{i=1}^{n} y_i x_i^3 - a_1 \sum_{i=1}^{n} y_i x_i^4 \\ \vdots \\ \sum_{i=1}^{n} y_i x_i^k - a_0 \sum_{i=1}^{n} y_i x_i^k - a_1 \sum_{i=1}^{n} y_i x_i^{k+1} \end{bmatrix} \quad (11)$$

Medición de error de cordón.

Para cada uno de los puntos ó interpolación lineal se requiere realizar el proceso de la **Figura 8**, donde se van agregando a un polinomio. Una vez que es obtenido el polinomio se calculan tres errores de cordón los cuales no deben sobrepasar el valor del error máximo. En la **Figura 9** se muestra un trazo conformado por el punto inicial del trazo (P_a) y el punto final (P_f), en la **Figura 9 a)** solo se enfoca en medir el error en los puntos de inicio y final del trazo y en la **Figura 9 b)** a parte de estos dos puntos se miden otros tres puntos los cuales se denominan errores de cordón.

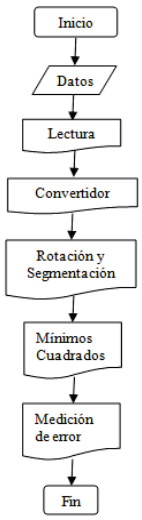

Fig. 8 Diagrama del proceso general.

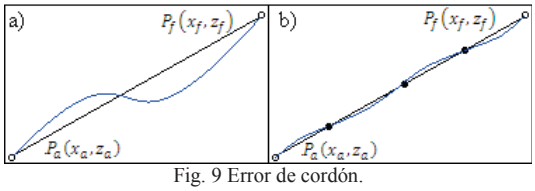

Fig. 9 Error de cordón.

Las ecuaciones para medir los errores son las siguientes:

$$Error_1 = P(x_f) - z_f \quad (12)$$

$$Error_2 = P\left(x_a + \frac{x_f - x_a}{2}\right) - \left(z_a + \frac{z_f - z_a}{2}\right) \quad (13)$$

$$Error_3 = P\left(x_a + \frac{x_f - x_a}{3}\right) - \left(z_a + \frac{z_f - z_a}{3}\right) \quad (14)$$

$$Error_4 = P\left(x_a + \frac{2(x_f - x_a)}{3}\right) - \left(z_a + \frac{2(z_f - z_a)}{3}\right) \quad (15)$$

Donde P es un polinomio de la forma $P(x) = a_1 x + a_2 x^2 + \cdots + a_k x^k$ en el cual se evalúan diferentes valores para obtener los diferentes errores de cordón, el error que se mida no debe exceder el valor del error máximo permitido. En caso de que alguno de los errores medidos exceda el error máximo, el polinomio calculado no es aceptado y se procede a generar un nuevo polinomio.

Experimentación

El experimento en el cual se probó la funcionalidad de la unidad propuesta se muestra en la **Figura 10,** donde se obtiene un código NC para un proceso de torneado de dos ejes utilizando el ISO 6983, el código se obtuvo a partir de un programa de CAD. El código es introducido en la unidad para generación de trayectorias polinomiales de donde se obtiene un archivo con nombre FPGA.txt. Este archivo es el que recibe el sistema de control de posición FPGA. La unidad se ejecutó en un sistema operativo de OpenSUSE versión 10.3 de Linux.

Es importante destacar que la experimentación se llevó a cabo comparando el controlador FPGA más la unidad, con un controlador Galil [10] en ambos casos los códigos G se obtuvieron de un CAD/CAM comercial. La trayectoria a maquinar se muestra en la **Figura 11,** el código NC contiene 6176 instrucciones que se convierten en 6175 trazos, que son el número de movimientos que realiza la tarjeta Galil para realizar el maquinado**.** Para utilizar el controlador basado en FPGA al código NC se le aplicó un polinomio de tercer grado, un error máximo permisible de 3 cuentas, una aceleración de 0.02, una desaceleración de -0.2 y una constante de 1000 cuentas por milímetro.

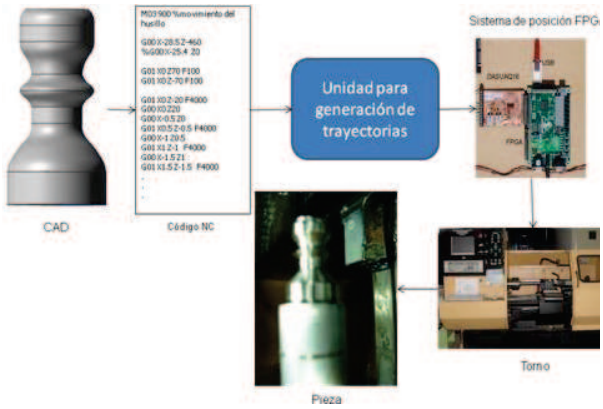

Fig. 10 Proceso general para obtener un maquinado con la unidad para generación de trayectorias.

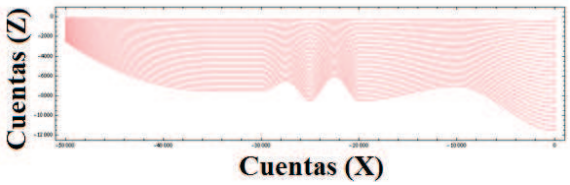

Fig. 11 Trayectoria de pieza diseñada en CAD/CAM.

Resultados

El código NC para la trayectoria de la pieza maquinada contiene 6176 instrucciones que se convierten en 6175 trazos, que son el número de movimientos que realiza la tarjeta Galil, estos puntos se muestran en la **Figura 12**. El tiempo de maquinado de esta pieza fue de aproximadamente 40 minutos con un error de seguimiento de 33.92 cuentas

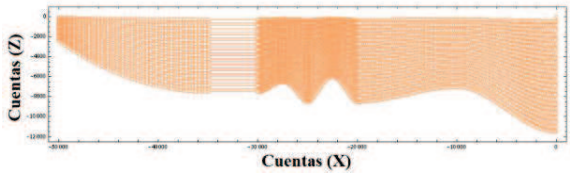

Fig. 12 Trayectoria para el controlador Galil.

Al aplicar la unidad generadora de trayectoria polinomiales a la trayectoria se generaron 804 polinomios los cuales son graficados en la **Figura 13**.

El tiempo de maquinado con el controlador de posición FPGA fue de 10 minutos con un error de seguimiento de 14.54 cuentas.

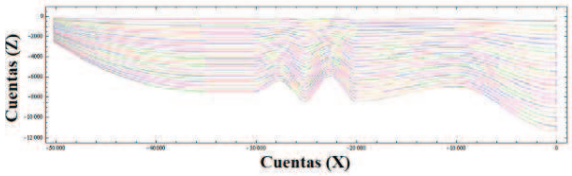

Fig. 13 Trayectoria para el sistema de control de posición FPGA.

El error máximo de cada polinomio que conforma la trayectoria se encuentra graficado en la **Figura 14** donde se puede observar que no sobrepasa el error máximo permisible de 3 cuentas.

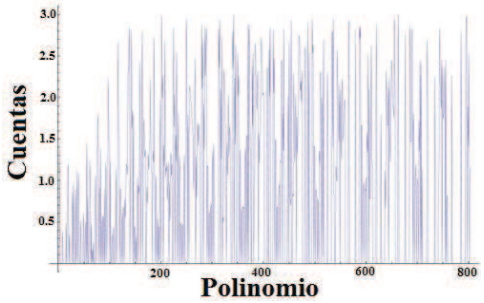

Fig. 14 Error de cordón máximo calculado en los polinomios de la trayectoria.

Conclusiones

En este artículo se presenta el desarrollo de una unidad para la generación de trayectorias para un controlador de posición FPGA. Se realizaron pruebas de maquinado comparando la propuesta con un controlador comercial Galil. Las pruebas mostraron que la unidad generó los parámetros necesarios para el controlador basado en FPGA satisfactoriamente puesto que el maquinado y la simulación se realizaron correctamente. Además la comparativa de los controladores demostró que el controlador FPGA en conjunto con la unidad propuesta fue superior al comercial en los parámetros de tiempo de maquinado y el error de seguimiento. Este trabajo a través de la unidad desarrollada permite estandarizar los códigos utilizados universalmente en la manufactura con un controlador propio desarrollado en hardware FPGA, presentando con ello una opción de automatización a bajo costo, mediante tecnología reconfigurable y que puede ser escalada.

Referencias

[1] K. Erkorkmaz and Y. Altintas, "High Speed CNC system design. Part I: jerk limited trajectory generation and quintic spline interpolation," *International Journal of Machine tools & Manufacture.*, vol. 41, pp.~1323-1345, July 2001.

[2] C. Yih, "Design and implementation of a linear jerk filter for a computerized numerical controller," *J Control Engineering Practice*, vol. 13, pp.~567-576. 2005.

[3] Gasparetto A. and Zanotto V., "A new method for smooth trajectory planning of robot manipulators," *Mechanism and Machine Theory*, vol. 42, pp.~455-471. Apr. 2007.

[4] S. Liu, "An On-line Reference-Trajectory Generator for Smooth Motion of Impulse-Controlled Industrial Manipulators". *IEEE/Advanced Motion Contro*". ISBN: 0-7803-7479-7.

[5] J. W. Jeon and Y. K. Kim, "FPGA based acceleration and deceleration circuit for industrial robots and CNC machine tools," *Mechatronics*, vol 12, pp.~635-642. 2002.

[6] B. Girau and A. Boumaza, "Embedded harmonic control for dynamic trajectory planning on FPGA," in *artificial intelligence and applications*, in IASTED conference proceedings, (Innsbruck, Austria), pp.~244-249, 2007.

[7] R. A. Osornio, R. de J. Romero, G. Herrera, R. Castañeda, "The application of reconfigurable logic to high speed CNC milling machines controllers", *Control Engineering Practice*, vol. 16, pp.~674-684. 2008.

[8] R. A. Osornio, R. de J. Romero, G. Herrera, R. Castañeda, "FPGA implementation of higher degree polynomial acceleration profiles for Peak jerk reduction in servomotors", *Robotics and Computer-Integrated Manufacturing*, vol. 25, pp.~379-392. Apr. 2009.

[9] L. Ming, T. Meng, Y. Hong, "Development of a dynamics-based NURBS interpolator with real time look-ahead algorithm", *International Journal of Machine Tools &Manufacture*. 2007.

[10] Galil Motion Corporation. DMC-18x0 Series Data Sheet. <www.galilmc.com>, 2004.

Currículo corto de los autores

J. Ugalde-Estrella obtuvo el grado de Ingeniero Eléctrico Mecánico en la Universidad Autónoma de Querétaro en 2009. Actualmente se encuentra estudiando la Maestría en Instrumentación y Control Automático en la Universidad Autónoma de Querétaro.
Luis Morales-Velazquez recibió el grado de Ingeniero en Electrónica de la Universidad de Guanajuato, Guanajuato, México, y el grado de Maestro en Ciencias en Instrumentación y Control de la Universidad Autónoma de Querétaro, Querétaro, México. Actualmente es estudiante del Doctorado en Ingeniería en la Universidad Autónoma de Querétaro participando en el Grupo de investigación HSPdigital. Sus intereses

de investigación incluyen procesamiento de señales en hardware, mecatrónica y desarrollo de plataformas de procesamiento en tiempo real basadas en FPGA.

J. P. Benitez-Rangel obtuvo el título de Ingeniero Electromecánico en 1999, el grado de Maestro en Ciencias (instrumentación y control) en 2002 y el grado de Doctor en Ingeniería en 2009 todos por parte de la Universidad Autónoma de Querétaro. Actualmente es profesor e investigador en la Facultad de Ingeniería de la Universidad Autónoma de Querétaro, donde realiza trabajos en el área de manufactura. Pertenece al Sistema Nacional de Investigadores (SIN) y cuenta con el perfil PROMEP.

Interprete de curvas B-spline a partir del comando G06.2 para controlador de movimiento basado en FPGA aplicado a un torno CNC.

Interpreter B-spline curves from the G06.2 command for motion controller based on FPGA applied to CNC lathe.

Javier Ugalde Estrella[2], Roque Alfredo Osornio Rios[1], Luis Morales Velázquez[1], René de Jesús Romero Troncoso[1], José Antonio Romero Navarrete[1], Olguín Rodríguez Francisco[3]

[1]Profesores de la Maestría en Mecatrónica de Ingeniería de la Universidad Autónoma de Querétaro, [2]Estudiante de la Maestría en Instrumentación y Control Automático de la Universidad Autónoma de Querétaro. [3]Estudiante de la Maestría en Mecatrónica de la Universidad Autónoma de Querétaro

RESUMEN. La utilización de las máquinas CNC en la industria es de gran importancia, su aplicación abarca casi todos los procesos de manufactura, siendo necesario desarrollar investigaciones relacionadas con el control de dichas máquinas. En este trabajo se describe la metodología para el desarrollo de un intérprete de comandos G06.2, el cual es un comando que proviene de los códigos G utilizado para generación de curvas NURBS y B-spline, en software para controlar los movimientos de una máquina-herramienta CNC utilizando un sistema de control de posicionamiento propio basado en FPGA. Se presentan los algoritmos utilizados para la generación de dicho intérprete y su implementación en una plataforma libre. El objetivo del trabajo es seguir una estandarización de los códigos G basados en Fanuc G06.2 con el controlador en hardware basado en FPGA desarrollado en la Universidad Autónoma de Querétaro.

Palabras clave: CNC, comando G06.2, Curvas B-spline, Controlador basado en FPGA.

1. INTRODUCCIÓN

La importancia de las pequeñas y medianas empresas (PYMES), en la economía y en la generación de empleos es fundamental para el país, de tal forma que es necesario atender la problemática que enfrentan los empresarios que buscan iniciar un negocio o que requieren fortalecerlo para asegurar su supervivencia. El gobierno tiene como uno de sus objetivos apoyar la rama industrial para propiciar que las empresas tengan una mayor competitividad favoreciendo su incursión en nuevos mercados nacionales e internacionales. Las máquinas herramientas de CNC (Control Numérico Computarizado) se encuentran extensamente propagadas en aplicaciones industriales, ya que con éstas el proceso de la industria manufacturera incrementa radicalmente pues generan una mayor producción, reduciendo los tiempos del mismo de tal forma que reduce costos e incrementa las ganancias ya sea a corto o a largo plazo. Actualmente existen diversas alternativas en máquinas de Control Numérico que pueden ser adquiridas por la industria mexicana y que son proporcionadas por empresas como Fagor, Siemens, Fanuc, Num, Dina, Mitsubishi, Heidenhain, entre muchas otras, todas ellas extranjeras con tecnología completamente propietaria lo cual genera costos excesivos imposibles de pagar por la mayoría de las PYMES complicando su supervivencia y su competitividad en el mercado. Las máquinas convencionales están diseñadas únicamente para interpolaciones lineales y circulares bajo el estándar ISO 6983, este estándar maneja los conocidos códigos G y M. Sin embargo, en equipos modernos se han introducido nuevos estándares de comunicación entre el CAD/CAM y el control CNC que son basados en tareas enlatadas STEP-NC ISO 14649 junto con los códigos relacionados con la generación de curvas paramétrica Fanuc G06.2.

La necesidad por mejorar la eficiencia y la precisión en un sistema de control numérico ha sido la mejor razón para el desarrollo de nuevas investigaciones y desarrollos tecnológicos tanto para el ámbito industrial como académico, logrando con esto, un gran avance en cuanto a la generación de nuevos métodos. Existe en la Universidad Autónoma de Querétaro una gran cantidad de trabajos desarrollados para mejorar y/ó analizar los procesos de manufactura. Los procesos de manufactura consisten principalmente de cuatro etapas: 1) Definición del modelo CAD (Computer Aided Design, Diseño Asistido por Computadora), 2) Procesamiento CAM (Computer

Aided Manufacuring, Manufactura Asistida por Computadora), 3) Sistema de control CNC y 4) Maquinado (Yeng et al., 2005). Las investigaciones se han enfocado a la generación de metodología para control, generación de referencias y dinámica, que en muy pocos casos se han implementado en una máquina. Las máquinas CNC utilizan un interpolador el cual calcula con su respectivo periodo de muestreo la siguiente ubicación de la herramienta en tiempo real de acuerdo a la velocidad de avance especificada, en los sistemas convencionales la interpolación para la trayectoria se realiza en base a códigos G y M que se basan en segmentación mediante trazos lineales y circulares. Sin embargo, recientemente se han reportado serias desventajas con este método de interpolación en aplicaciones reales como son: 1) Errores de transmisión entre CAD/CAM y CNC debido a la gran cantidad de comandos generados al segmentar una curva de alta precisión en pequeños trazos lineales. 2) Bajo precisión en los acabados debido a la discontinuidad de la segmentación. 3) La herramienta necesita acelerar y desacelerar en cada segmento, lo que provoca velocidad discontinua en la unión de dos segmentos y puede causar daños al mecanismo y a la pieza a maquinar. 4) Comportamiento dinámico discontinuo, especialmente aceleración, desaceleración y jerk. Erkorkmaz y Altintas (2001) afirman que niveles de jerk altos generan mayor cantidad de vibraciones. Ramesh et al. (2005) destacan que parte elemental para lograr alta precisión en las máquinas-herramienta es el desempeño de la dinámica de los ejes, menciona además que en los sistemas CNC se utiliza primordialmente lazos de control en velocidad y posición para controlar los ejes de la máquina-herramienta en el cual se emplean tacogeneradores y encoders digitales para retroalimentarlos. Gao et al. (2005) afirman que un problema inherente en los sistemas de posicionamiento inerciales es el crecimiento, con el tiempo, de los errores de medición en la velocidad y posición. En lo que respecta al desarrollo científico, diferentes técnicas de generación de trayectorias han sido propuestas enfocadas a mejorar la calidad de los acabados y reducción del tiempo de maquinado basados mediante curvas paramétricas como curvas pitagóricas, de Bézier, B-spline y NURBS (Non-Uniform Rational B-spline; spline-base rotacionales no uniformes). Macfarlane y Croft (2003) desarrollaron un método para la obtención de trayectorias suaves y acotadas en jerk donde destacan que el control de la dinámica es importante en aplicaciones a maquinaria CNC y robots industriales. Los trabajos más relacionados con el control y la dinámica de maquinaria CNC basados en FPGA son los de Osornio et al. (2008) quienes presentan un controlador Proporcional-Integral-Derivativo (PID) basado en un FPGA aplicado a maquinas CNC de alta velocidad para lograr obtener los tiempos de actualización de control requeridos. Posteriormente Osornio et al. (2009) en otro trabajo integran el módulo PID con un nuevo desarrollo de un generador de perfiles basado en polinomios de alto grado limitando la dinámica del jerk de máquinas CNC y robótica para mejorar los procesos de maquinados. La novedad de su sistema fue el desarrollo de un algoritmo recursivo multigrado para la evaluación polinomial, computacionalmente eficiente, en generación de perfiles y la implementación a bajo costo de la estructura digital en FPGA. Como se puede observar existen diferentes investigaciones enfocadas en mejorar la dinámica de los sistemas de control de posicionamiento en máquinas herramientas, sin embargo no realizan una unidad en software que permita la comunicación entre el usuario y el hardware. Morales (2010) presenta una plataforma de arquitectura abierta basada en unidades multi-agente de hardware-software, mediante el desarrollo de un nuevo sistema de control distribuido multi-agente (*Multi-Agent Distributed CONtroller*, MADCON) para satisfacer los requerimientos de reconfigurabilidad para la siguiente generación de máquinas inteligentes.

En este trabajo se presenta un intérprete en software para estandarizar el controlador y generador de perfiles implementados en un FPGA, basados en las metodologías de Osornio y Mórales, buscando minimizar los problemas ocasionados por la segmentación de trayectorias en los programas NC. Para la realización de esta unidad se parte del CAD realizando la representación de curvas las cuales son proporcionadas al CAM que realiza la segmentación de curvas dando como resultado el código NC de acuerdo al ISO 6983 y al códigos relacionado con la generación de curvas paramétrica Fanuc G06.2, el cual proporciona la segmentación de líneas y arcos que deben realizarse en el maquinado. Este código se leerá, compilará y analizará de tal forma que se pueda interpretar para obtener los parámetros necesarios que requiere el controlador de posición basado en FPGA. Las pruebas de la unidad se llevan a cabo en un torno CNC y se demuestra la eficacia de la metodología mediante la experimentación.

2. METODOLOGÍA

Este trabajo consiste en la elaboración de un intérprete de comando G06.2, el cual se encuentra estandarizado por Fanuc y es utilizado para la realización de las curvas paramétricas como NURBS y B-spline, dicho intérprete

contiene un módulo de lectura del programa NC, un módulo para la compilación de este y además un simulador el cual permite observar los movimientos que realizarán en el torno durante el maquinado. La programación utilizada es C++ en un entorno de Linux. El proceso general se muestra en un diagrama de bloques en la **Figura 1** donde el programa NC con comandos G06.2 puede ser obtenido desde un módulo CAD o ser introducido por el usuario directamente en el intérprete, el intérprete como ya fue mencionado contiene tres módulos fundamentales el lector, el compilador y el simulador. El lector y el compilador son los módulos que se encargar de obtener los parámetros requeridos por el sistema de control de posicionamiento basado en FPGA. El sistema de posicionamiento se encuentra en comunicación con el CNC que a su vez controla los movimientos de la máquina-herramienta (torno).

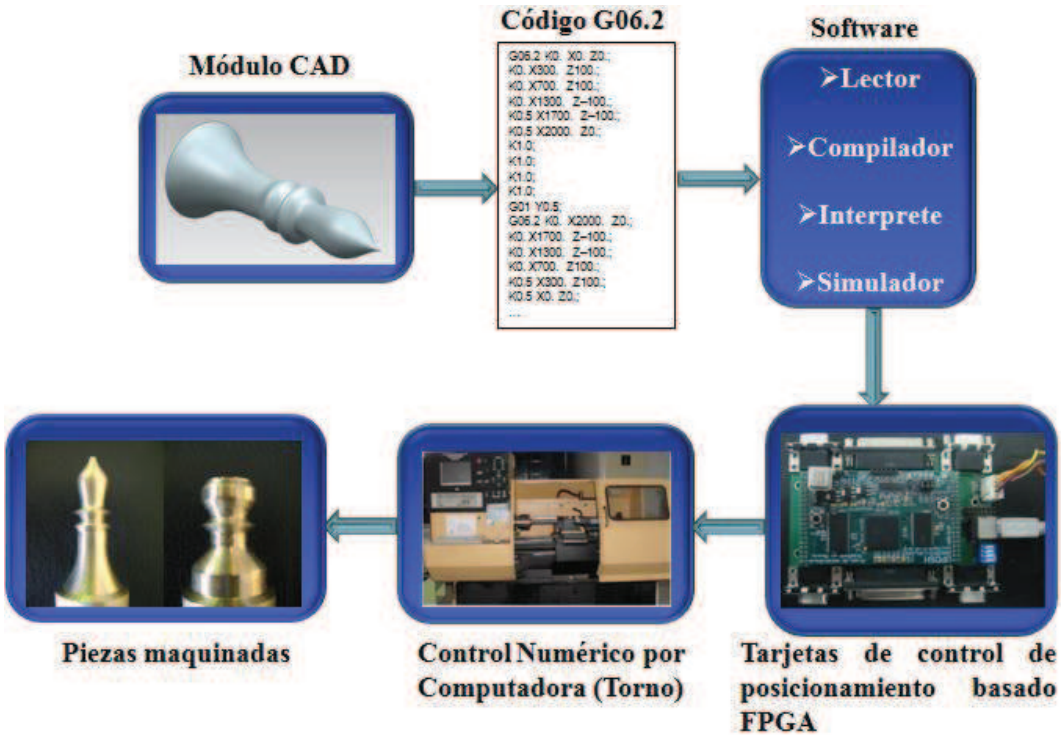

Figura 1. Proceso general de un maquinado.

Un programa de maquinado NC se encuentra compuesto por una serie de bloques como el mostrado en la **Figura 2**, cada uno de estos bloques está compuesto por una dirección, un dato y una palabra. La dirección es el número del bloque, el dato es el comando que se realizará y la palabra es compuesta por los parámetros que requiere cada uno de los comandos. En este trabajo se utiliza el comando G06.2, el cual se muestra en la **Figura 2**, donde la palabra se compone por el rango el cual es especificado por la dirección **P**, el peso se especifica con la dirección **R**, la velocidad de avance con **F**, los valores de los puntos de control con **K** y las coordenadas de los puntos de control con **X**, **Y** y **Z**.

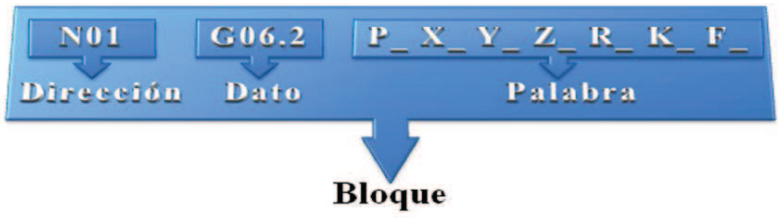

Figura 2. Bloque de un programa CNC.

2.1. Lector

En los programas NC al igual que en la mayoría de los lenguajes de programación se requiere realizar comentarios para identificar lo que realiza dicho programa, los comentarios son especificados con el carácter '%'. El lector además de cargar el programa a una terminal gráfica, realiza un pre-proceso el cual elimina todos los espacios, tabulaciones y renglones de más que se encuentren en el programa así como los comentarios realizados por el usuario para que el programa cargado se encuentre limpio de basura y pueda realizarse una compilación de alguna forma más sencilla y rápida.

En la **Figura 3** se muestra un diagrama de flujo en el cual se muestran las comparaciones necesarias para quitar los caracteres innecesarios en un bloque. Si el carácter es un espacio ó una tabulación entra a un proceso de agregar espacio este proceso compara si el buffer se encuentra vacio ó si el último carácter en el buffer es un carácter de fin de línea, el espacio no se agrega ya que estas comparaciones indican que el espacio es al inicio del bloque, de lo contrario agrega un espacio al buffer y entra al proceso de avanzar este proceso avanza hasta encontrar un carácter distinto de espacio y tabulación. En caso de que el carácter sea '%' indica el inicio de un comentario por lo cual entra a el proceso de avanza que en esta ocasión se realiza el avance hasta encontrar el fin de línea. Por último si el carácter que se encuentra en comparación indica el fin de línea ó fin de una instrucción (;), entra al proceso de agregar fin de línea, en este proceso se compara si el archivo se encuentra vacio ó si el último carácter en el buffer es un fin de línea no se agrega el fin de línea puesto que estas comparaciones indican que son líneas vacías en el programa, de lo contrario se agrega un fin de línea al buffer. Si el carácter no entra a ninguna de las condiciones anteriores se agrega al buffer ya que este sería un carácter alfanumérico y no se encuentra dentro de los comentarios.

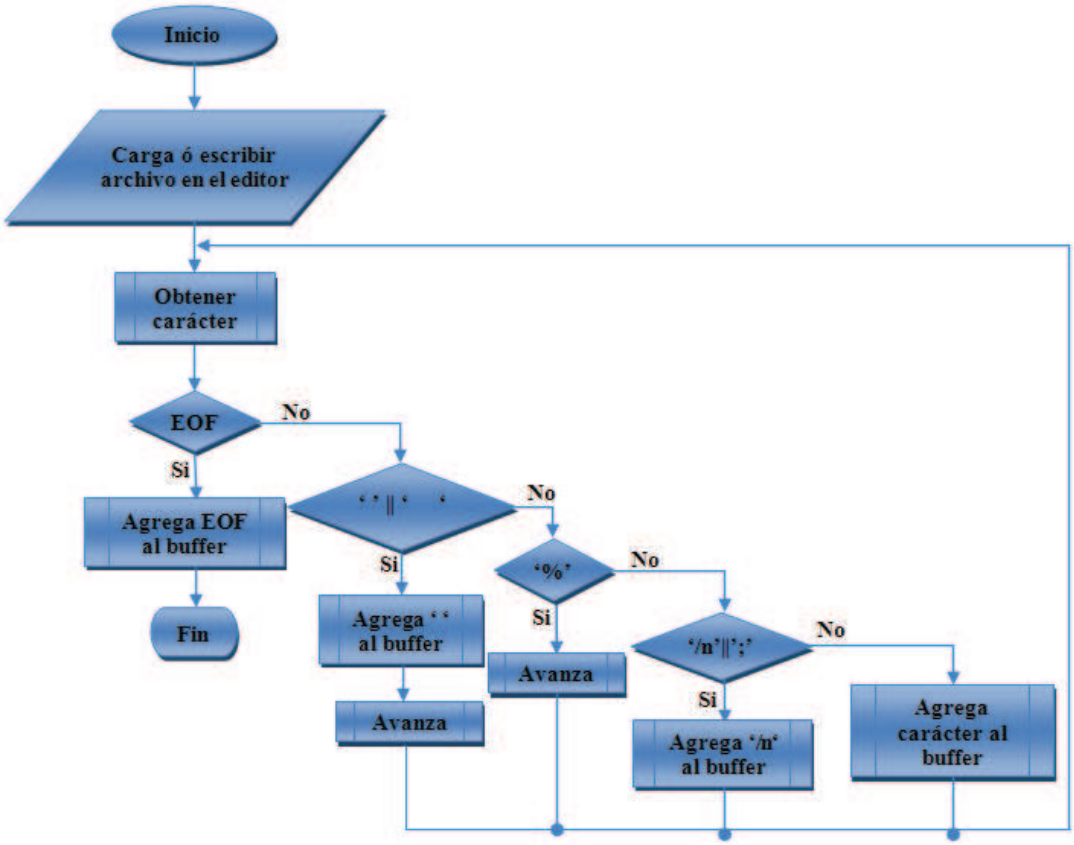

Figura 3. Diagrama de flujo del módulo del lector.

2.2. Compilador

Es requerido un compilador para el comando G06.2 ya que al generar manualmente un programa NC es normal que existan errores de sintaxis, esto podría ser muy perjudicante al realizar un maquinado ya que la máquina-herramienta no realizaría los movimientos deseados. El compilador recibe el buffer generado por el lector que únicamente contiene las líneas con bloques de datos. Las direcciones para cada uno de los parámetros se muestran en la **Figura 2**, las cuales deben de ser identificadas para obtener los valores que contiene cada una de estas direcciones.

En la **Figura 4** se muestra el diagrama de flujo del proceso del compilador, las comparaciones que se muestran son las comparaciones necesarias para identificar las direcciones que pueden existir en un comando G06.2. La primera dirección que se analiza es **N** y entra a un proceso donde se compila el valor que contiene, dicho valor debe de ser un número entero positivo. El segundo valor a compilar es el valor que se obtiene de la dirección **G** el cual debe contener el comando que se desea analizar, en este trabajo únicamente se analizan los comandos G06.2, G90 y G91, de tal forma que solo puede tener como valor 06.2, 90 ó 91. La tercera dirección que se analiza es **K**, esta dirección únicamente puede contener valores de 0 a 1 ya que las curvas paramétricas NURBS y B-spline son evaluadas en ese intervalo. Las direcciones **F** y **R** son introducidas a un mismo proceso de compilación ya que tanto la velocidad de avance como el peso deben tener un valor positivo. Por último las direcciones de los distintos ejes (**X**, **Y** y **Z**) entran a un mismo compilado ya que su valor puede ser cualquier número real. Si existe algún otro carácter alfabético extra para este comando se detecta un error de sintaxis ya que no hay mas direcciones disponibles.

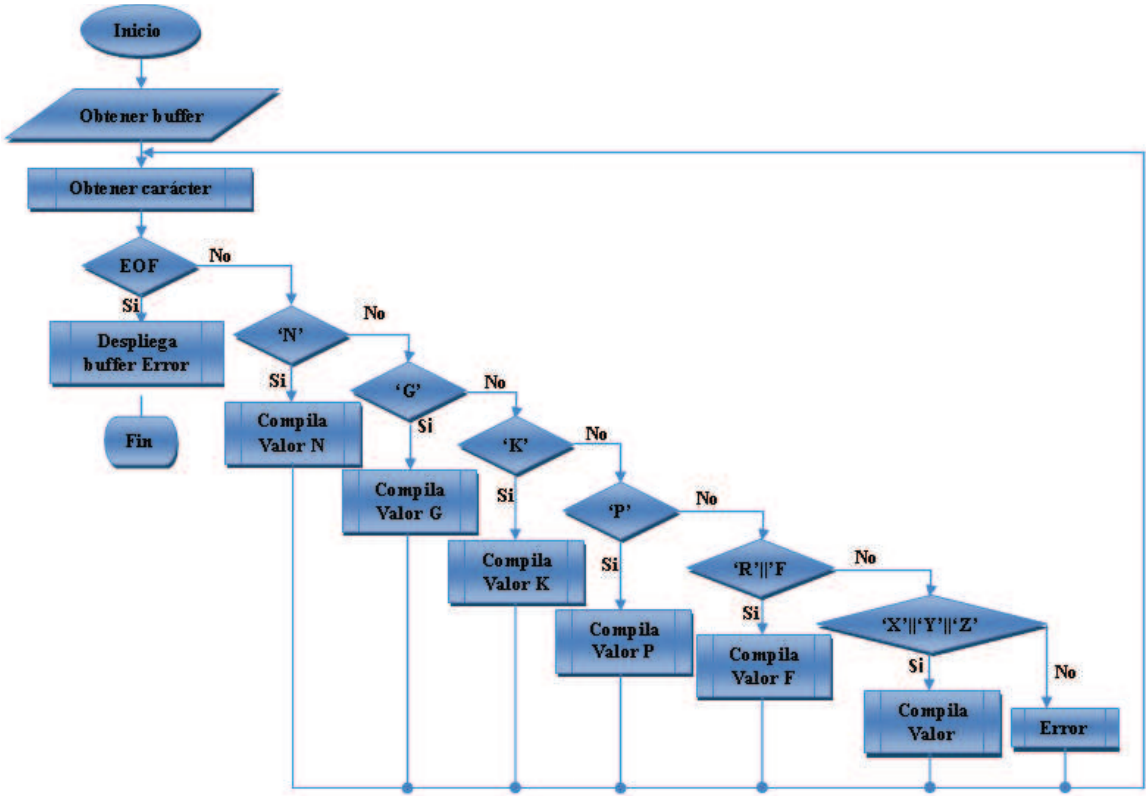

Figura 4. Diagrama de flujo del módulo del compilador.

En caso de que en alguna dirección se esté introduciendo un valor que no pueda ser contenido por esta dirección el proceso de compilar introduce dicho error al buffer de error. Todos los caracteres de los valores de las

direcciones deben ser numéricos, en caso que exista un carácter alfabético en los caracteres de algún valor se detecta un error de sintaxis este error es introducido al buffer de error.

Los comandos G90 y G91 hacen referencia al tipo de coordenadas G90 se refiere a coordenadas absolutas y G91 a coordenadas relativas. Si no se hace referencia al tipo de coordenadas que se desea trabajar se considera que el programa se encuentra trabajando en coordenadas absolutas. Estos comandos no requieren ningún parámetro por lo tanto si se identifica algún parámetro hacia este comando se detecta un error de sintaxis y es introducido en el buffer de error.

2.3. Interprete

Al realizar un programa NC no siempre se requiere especificar todas las direcciones, un ejemplo de un programa NC con código G06.2 se muestra en la **Figura 5**, como se puede observar no es especificada la dirección **P**, si no es especificada toma un valor por default de 4, la dirección del peso **R** tampoco es especificada y toma un valor de uno en los puntos de control que no sea especificada, sin embargo, para la utilización de B-spline este parámetro no es requerido ya que el peso únicamente es requerido para curvas NURBS. En el código se observa que solo se declara una vez la dirección de la velocidad de avance **F** al igual que la dirección de la coordenada del eje **Z**, de tal forma que para los siguientes bloques estos valores se mantienen, esto hasta que sea especificado un nuevo valor para las direcciones.

```
%Estrella
N01 G06.2 K0.0 X8.0 Y12.0 Z0.0 F1000;
N02 K0.0 X5.0 Y8.0;
N03 K0.0 X0.0;
N04 K0.111 X4.0 Y4.0;
N05 K0.222 X3.0 Y0.0;
N06 K0.333 X8.0 Y3.0;
N07 K0.444 X13.0 Y0.0;
N08 K0.555 X12.0 Y4.0;
N09 K0.666 X16.0 Y8.0;
N10 K0.777 X11.0;
N11 K0.888 X8.0 Y12.0;
N12 K1.0
N13 K1.0
N14 K1.0
```

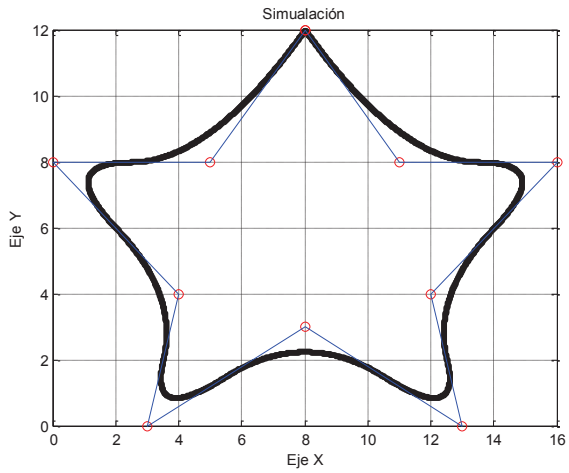

Figura 5. Programa NC con código G06.2 y simulación.

Se realizó un módulo que es capaz de interpretar un programa NC, con comandos G06.2, G90 y G91, y obtiene los parámetros necesarios de dicho programa para poder aplicar el método de las curvas B-spline. Este módulo recibe el archivo del lector siempre y cuando haya pasado con éxito el proceso de compilación. El intérprete cuenta con una lista enlazada utilizada como una FIFO en la cual se almacenan los valores de las direcciones de la siguiente forma **G**, **X**, **Y**, **Z**, **F**, **K**, **P** y **R** esto se muestra en la **Tabla 1**. Los valores de los ejes siempre son almacenados en coordenadas absolutas ya que el controlador de posición basado en FPGA únicamente recibe estos los valores de las los ejes en coordenadas absolutas. El controlador siempre requiere el envió del valor de todos los ejes por lo tanto el programa de la **Figura 5** queda almacenado como se muestra en la **Tabla 1**.

Tabla 1. Forma de almacenamiento de los datos.

G	X	Y	Z	F	K	P	R
6.2	8.0	12.0	0.0	1000	0.0	4.0	1.0
6.2	5.0	8.0	0.0	1000	0.0	4.0	1.0
6.2	0.0	8.0	0.0	1000	0.0	4.0	1.0
6.2	4.0	4.0	0.0	1000	0.111	4.0	1.0
6.2	3.0	0.0	0.0	1000	0.222	4.0	1.0
6.2	8.0	3.0	0.0	1000	0.333	4.0	1.0
6.2	13.0	0.0	0.0	1000	0.444	4.0	1.0
6.2	12.0	4.0	0.0	1000	0.555	4.0	1.0
6.2	16.0	8.0	0.0	1000	0.666	4.0	1.0
6.2	11.0	8.0	0.0	1000	0.777	4.0	1.0
6.2	8.0	12.0	0.0	1000	0.888	4.0	1.0
6.2	8.0	12.0	0.0	1000	1.0	4.0	1.0
6.2	8.0	12.0	0.0	1000	1.0	4.0	1.0
6.2	8.0	12.0	0.0	1000	1.0	4.0	1.0

2.4. Simulación

Hasta este esté punto ya es posible enviar los datos al sistema de posicionamiento basado en FPGA, sin embargo, se realizó un módulo de simulación el cual reciba los mismos parámetros que utiliza el sistema de control de posicionamiento basado en FPGA para la generación de las curvas B-spline a partir de los valores almacenados en el intérprete.

El sistema de control basado en FPGA para generar las curvas B-spline requiere únicamente del programa NC todos los valores de los puntos de control y los valores de los ejes pero solo los indicados en el programa, en la **Tabla 2** se muestran los valores que requiere el sistema de posición del programa NC mostrado en la **Figura 2**.

Tabla 2. Datos requeridos por el sistema de posicionamiento basado en FPGA.

X	Y	Z	K
8.0	12.0	0.0	0.0
5.0	8.0	0.0	0.0
0.0	8.0	0.0	0.0
4.0	4.0	0.0	0.111
3.0	0.0	0.0	0.222
8.0	3.0	0.0	0.333
13.0	0.0	0.0	0.444
12.0	4.0	0.0	0.555
16.0	8.0	0.0	0.666
11.0	8.0	0.0	0.777
8.0	12.0	0.0	0.888
No requerido	No requerido	No requerido	1.0
No requerido	No requerido	No requerido	1.0
No requerido	No requerido	No requerido	1.0

Como se observar en la **Tabla 2** no son requeridos todos los valores de los ejes de los puntos de control pero si todos los valores de los puntos de control de tal forma que no se pueden enviar directamente todos los dato. Por lo tanto se considero que finalizada un curva B-spline cuando el valor del punto siguiente es menor que el punto

actual y se realiza el cálculo para la generación de los puntos de la curva almacenada y se simula al terminar el proceso se inicia el almacenamiento de los datos de la siguiente curva.

La implementación para la obtención de las curvas B-spline al igual que en el sistema de control fue realizada de acuerdo al algoritmo de Boor basado en la inserción de nodos múltiples. La **Figura 6** muestra proceso propuesto en un diagrama de flujo para esta técnica paramétrica donde los puntos de control $\{Q_i\}$, el vector de nodos $\{tk_i\}$ y el parámetro t son las entradas, y el dato B-spline es la salida. Se muestra el diseño para un eje, pero este puede ser duplicado para más ejes. En este caso, el diseño se basa en el algoritmo de Boor descrito por Yau et al. (2006) el cual se logra mediante la evaluación recursiva de las ecuaciones **(1)** y **(2)**.

$$Q_i^j(t) = \left(1-\alpha_i^j\right)Q_i^{j-1}(t) + \alpha_i^j Q_{i+1}^{j-1}(t) \quad \textbf{(1)}$$

$$\alpha_i^j = \frac{t - tk_{r-j+1}}{tk_{r+i} - tk_{r-j+i}} \quad \textbf{(2)}$$

$$p = num[tk] - num[Q_i] - 1 \quad \textbf{(3)}$$

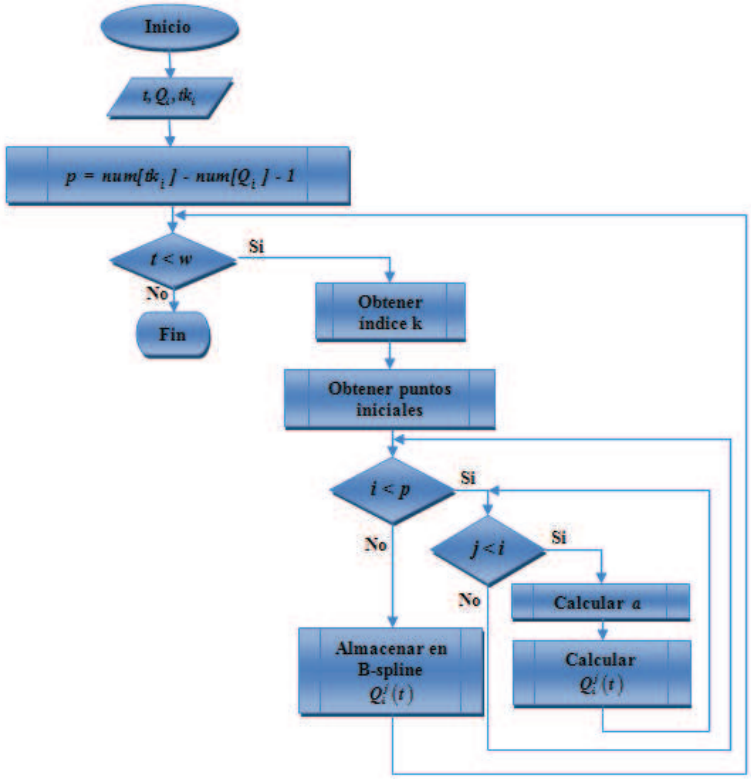

Figura 6. Diagrama de flujo para las curvas B-spline.

El grado del B-spline se encuentra definido por la cantidad de nodos y la cantidad de puntos de control como se muestra en la ecuación **(3)**, el proceso se repite dependiendo la cantidad de puntos $\{w\}$ que sean necesarios para la reconstrucción de la curva, el valor de t incrementa $1/w$ de 0 hasta 1, se obtiene el valor del índice tal que $t \epsilon [tk, tk_{r+1})$ para obtener los puntos de control iniciales, la cantidad de puntos iniciales debe de ser igual al grado de la

curva. Una vez obtenido esto se procede al cálculo recursivo de las ecuaciones **(1)** y **(2)**, obteniendo al final un punto correspondiente a la curva. Todos los puntos son guardados en B-spline para al final poder reconstruir la curva

2.5. Interfaz gráfica

Como ya fue mencionado este trabajo es desarrollado bajo un entorno de Linux, para generación de la interfaz gráfica se utilizó Glede el cual es una herramienta de desarrollo visual de interfaces gráficas mediante GTK/GNOME. Haciendo referencia a los *widgets* generados en Glade con las librerías de GTK y gtkmm.

En la interfaz gráfica generada se puede cargar el archivo del programa NC, editar y guardar además de compilar, simular y ejecutar los programas NC con comandos G06.2, G00 y G01. Cabe mencionar que si el archivo no pasa por la compilación no es posible ni simular ni ejecutar el programa, la simulación es opcional sin embargo es recomendable realizarla ya que se puede observar los movimientos que realizara la máquina-herramienta durante el maquinado.

La ventana principal se encuentra dividida en tres secciones el editor, el simulador y el jog. En el editor (ver **Figura 7**) donde se carga el programa NC y se compila. Además cuenta con ayuda para la sintaxis que se debe utilizar para cada uno de los comandos (G00, G01 y G06.2). Una vez compilado el programa en el editor y sin errores se activan los controladores del simulador (ver **Figura 9**), donde se carga el programa NC limpio, es decir, sin comentarios y sin espacios extras, el texto contenido no puede ser editado únicamente es utilizado para tener un seguimiento en la simulación y ejecución. Los ejes mostrados en esta sección muestran el posicionamiento de los ejes con respecto al home dado. Por último la interfaz del jog es utilizada para posicionar el portaherramientas en la posición inicial del maquinado y establecer el home de la máquina-herramienta, esto se realiza seleccionando la medida en que se desea mover (cuentas, mm, cm ó pulgadas) y el eje (X, Y ó Z). En esta sección también se configura el controlador ya que el sistema de posicionamiento utilizado requiere ciertos parámetros que no se pueden dar con código NC.

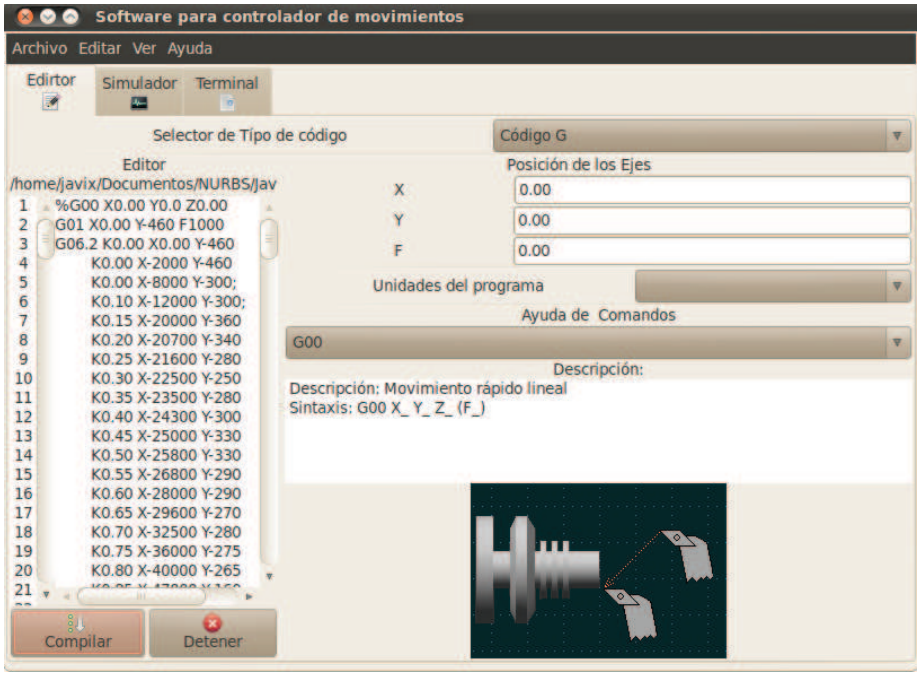

Figura 7. Interfaz del editor de programa NC.

La configuración del controlador consta de dos secciones, la primera sección requiere las ganancias de control, proporcional (Kp), integral (Ki) y derivativa (Kd) las cuales son obtenidas al identificar la planta de los servomotores utilizados en la máquina herramienta. En la segunda sección se introducen los parámetros de movimiento para el controlador los cuales son la frecuencia de muestreo (Fs), velocidad de avance máxima (Vf), aceleración máxima (Am), desaceleración máxima (Dm), el límite de error máximo y el equivalente de 1 cm a cuentas este valor depende de los servomotores utilizados y es necesario para realizar las conversiones. Los primeros cuatro parámetros son requeridos para la construcción de los perfiles de posición, velocidad y aceleración en el sistema de control de posicionamiento, el límite de error es un parámetro de seguridad ya que si es mayor el error por seguridad se detiene el proceso de maquinado.

3. PRUEBAS

El proceso general se muestra en la Figura 1, la trayectoria realizada para esta prueba corresponde al desbaste de un alfil con curvas B-spline, En **Figura 8** se muestra la trayectoria que se deberá realizar, los puntos marcados son los puntos requeridos para generar la trayectoria con B-spline. El material utilizado para el maquinado es una barra de aluminio de 1 pulgada de diámetro, un inserto de carburo de tungsteno de la empresa Travers con el número de inserto TNMG 432 y un torno CNC de procedencia checa, instrumentado por la Universidad Autónoma de Querétaro.

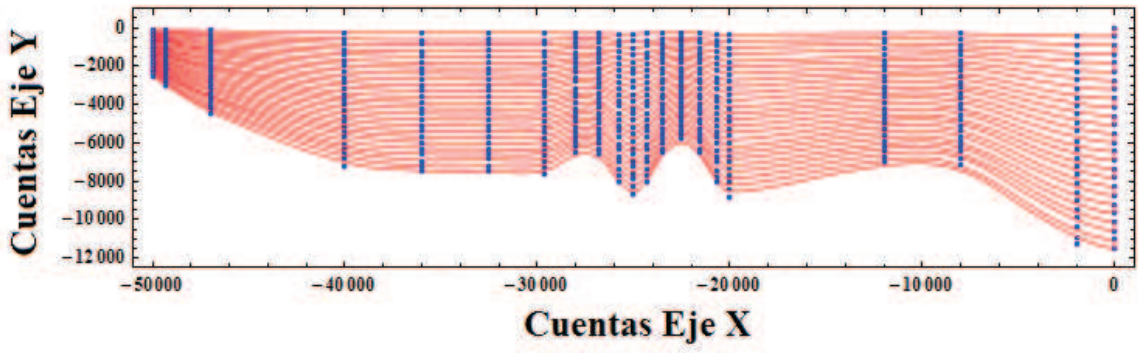

Figura 8. Puntos de los B-spline y trayectoria del código de prueba.

Las ganancias obtenidas para el control de los servomotores se muestran en la **Tabla 3** y los parámetros de movimiento utilizados para esta prueba se muestran en la **Tabla 4**.

Tabla 3. Ganancias para el control de los servomotores.

	Kp	Ki	Kd
Servomotor X	**250.4291**	**7.6583**	**1.7157**
Servomotor Y	**250.4291**	**7.6583**	**1.7157**

Tabla 4. Parámetros de movimiento utilizados para la prueba.

Frecuencia de Muestreo	**1000 Hz**
Velocidad de Avance Maxima	**10000 cuentas/seg**
Aceleración Máxima	**100000 cuentas/seg^2**
Desaceleración Máxima	**-100000 cuentas/seg^2**
Limite de Error Máximo	**3000 cuentas**
Equivalente a 1cm	**10000 cuentas**

4. RESULTADOS

Se cargo el programa en la interfaz realizada para el intérprete, se compilo y se simulo, la simulación se muestra en la **Figura 9** donde se puede observar que la trayectoria simulada en la interfaz es muy similar a la trayectoria deseada mostrada en la **Figura 8**.

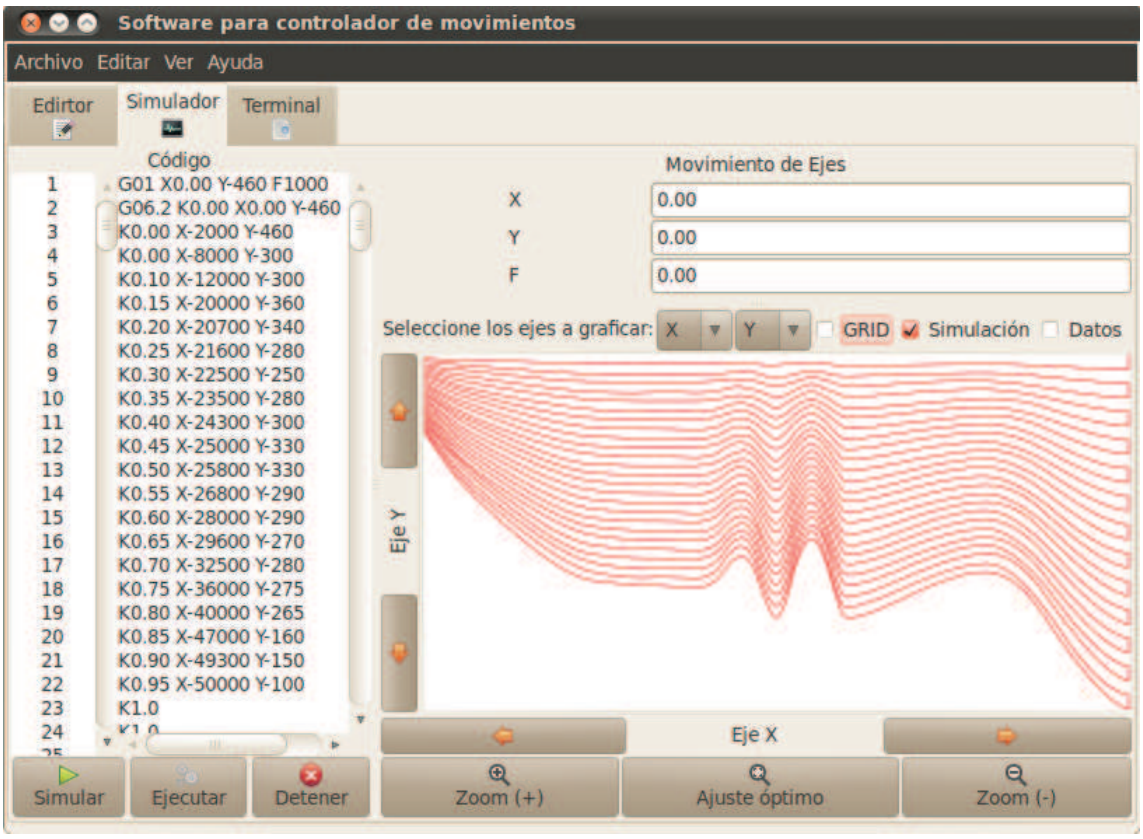

Figura 9. Interfaz de la simulación de la prueba realizada.

El tiempo realizado para el desgaste fue de 9 minutos con 30 segundos, la pieza resultante se muestra en la **Figura 10**, cabe mencionar que el límite de error para esta prueba fue de 3000 cuentas.

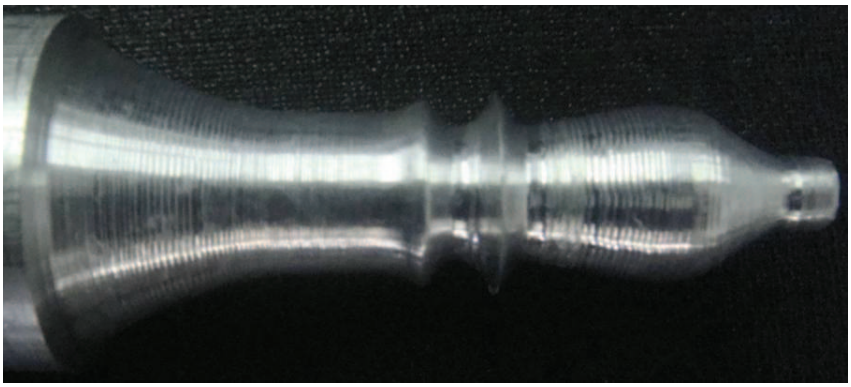

Figura 10. Pieza de prueba realizada.

5. CONCLUSIONES

Este trabajo no solo se realiza una comunicación con el sistema de control propio basado en FPGA, también se diseña una unidad en software con una interfaz gráfica que sea fácil de utilizar para todo usuario que la requiera y es una aportación al sistema de control de posicionamiento basado en FPGA desarrollado en la Universidad Autónoma de Querétaro por un grupo de investigadores enfocados a reconfigurar los sistemas de control, además que se sigue un estándar en los comandos G06.2 (FANUC), G00 y G01 (ISO 6983).

El sistema se probo desarrollando el maquinado de un alfil en una barra de aluminio de 1 pulgada de diámetro utilizando un inserto de carburo de tungsteno. Los resultados mostraron una comunicación satisfactoria entre el intérprete y el controlador de posicionamiento siguiendo una estandarización de los códigos (ISO 6983 y FANUC).

6. REFERENCIAS

Erkorkmaz Kaan & Altintas Yusuf. 2001. High speed CNC system design. Part I: jerk limited trajectory generation and quintic spline interpolation. *International Journal of Machine Tools & Manufacture*. 41:1323-1345.

Gao J., Webb P., Gindy N. 2005. Research on an inertial positioning system for a parallel kinematic machine. Mechatronics. 15:1-22.

Macfarlane Sonja, Croft Elizabeth A. 2003. Jerk-bounded manipulator trajectory planning: design for real time applications. IEEE Transactions on Robotics and Automation. 19: 42-52.

Morales Velázquez Luis. 2010. Diseño de plataforma hardware-software para el desarrollo de aplicaciones industriales basadas en FPGA. Tesis de Doctorado. Universidad Autónoma de Querétaro. Facultad de Ingeniería.

Osornio R. A., Romero R. de J., Herrera G., Castañeda R. 2008. The application of reconfigurable logic to high speed CNC milling machines controllers, *Control Engineering Practice*, vol. 16, pp.~674-684.

Osornio R. A., Romero R. de J., Herrera G., Castañeda R. 2009. FPGA implementation of higher degree polynomial acceleration profiles for Peak jerk reduction in servomotors, *Robotics and Computer-Integrated Manufacturing*, vol. 25, pp.~379-392.

Paluszny Marco, Prautzsch Hartmut, Boehm Wolfgang. 2005. Metodos de Bezier y B-splines. Universitatsverlag Karlsruhe. Karlsruhe

Ramesh R., Mannan M. A., Poo A. N. 2005. Tracking and contour error control in CNC servo systems. International Journal of Machine Tools and Manufacture. 45: 301-326.

Yau H-T, Lin M-T, Tsai M-S. 2006. Real-time NURBS interpolation using FPGA forhigh speed motion control. Comput Aided Design. 38:1123-1133.

Yeng C H, Altintas Y & Erforkmaz K. 2005. Virtual CNC system. Part I. *System architecture*, 1107-1123.