UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Construcción de una máquina fresadora de control numérico

TESIS PROFESIONAL

que para obtener el título de

INGENIERO MECÁNICO

PRESENTA

LUIS HUMBERTO SÁNCHEZ SÁNCHEZ

DIRECTOR DE TESIS

DR. ALEJANDRO C. RAMÍREZ REIVICH

Ciudad Universitaria, México, D.F., 2011

AGRADECIMIENTOS

A mi tío Luis, que me inculcó el gusto por la mecánica. Sus enseñanzas y ayuda han permitido lograr éste y otros proyectos.

A mi familia por su apoyo, paciencia, compañía y amor.

A la vida, por su maravilloso regalo diario durante todos estos años.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABLAS

INTRODUCCIÓN

El trabajo que aquí se presenta surgió y se desarrolló a partir de pláticas que se tuvieron con un tío, el señor Luis Sánchez Alva, trabajador y propietario de un pequeño taller mecánico industrial ubicado en el municipio de Ecatepec, Estado de México, el cual dispone, entre otras, de máquinas-herramienta como torno, fresadora y troqueladora.

Un problema en el taller es la falta de automatización de tareas y procesos, que permitan aumentar la producción y calidad de algunos de sus productos. En ese sentido, se decidió trabajar conjuntamente con el objetivo de construir una máquina de control numérico computarizado (CNC) similar a algunas que se habían visto en videos de *youtube*, la cual podría ser de gran ayuda para realizar trabajos que se realizan en el taller.

En el mercado el precio de estas máquinas varía según las características, desde 20 hasta 200 mil pesos, pero se tenía la certeza de que la inversión para este proyecto podía ser menor y de hecho así fue, pues el costo total fue de alrededor de 3 mil pesos debido a algunos de los siguientes factores:

- Para la construcción de la máquina se utilizaron algunas piezas y material de reuso.
- Los controladores electrónicos se elaboraron propiamente, lo cual representó un ahorro del 50 al 80%.
- Se utilizaron paqueterías gratuitas tanto para el control de la máquina como para la creación de los programas, demostrándose, en este aspecto, que es posible realizar una gran cantidad de operaciones sin tener que gastar en costosas paqueterías.
- Se contó con la generosa donación de una computadora por parte de otro familiar.

Procedimiento de trabajo

En principio se dividió el proyecto en dos tareas; por un lado la realización de los circuitos electrónicos y, por otro, la construcción de una fresadora de tres ejes, que correspondió al propietario del taller. Una vez cumplidas estas tareas y ya funcionando la máquina CNC, se procedió a investigar la manera de crear programas de código numérico para diversas operaciones de maquinado.

Dado el carácter eminentemente práctico del proyecto, no se siguió un método rígido para su desarrollo, pero se describe el procedimiento seguido en cada una de las partes del mismo y corresponde al orden de presentación de los capítulos:

Controladores:

- Se investigó el tipo de controladores que se requerían, para lo cual fueron de gran ayuda algunas páginas electrónicas, de las que se obtuvieron una gran cantidad de esquemas y diagramas.
- Se hicieron algunos primeros circuitos que no funcionaron. Se siguió probando y se logró un diseño que permitió comenzar a utilizar la máquina, sin embargo este modelo presentó deficiencias y llegó a fallar.
- Se requirió seguir investigando y se hicieron dos nuevos modelos que soportan más potencia y son los que actualmente se están utilizando.

Máquina:

- El propietario del taller revisó videos de máquinas similares en *youtube* especialmente para conocer los mecanismos de movimiento empleados.

- Lo anterior, aunado a su amplia experiencia, le permitió ir definiendo la manera en que construiría la máquina.
- Inició la construcción de la misma utilizando y adaptando piezas que tenía en el taller y comprando otras, básicamente en un tianguis de desderdicio industrial ubicado en la colonia *San Felipe*. Ahí mismo se compraron los motores a pasos utilizados.
- Una vez concluida, se acopló a los controladores electrónicos que se venían realizando paralelamente.

Programas en código numérico:

- Cuando se tuvo el primer modelo de controlador y la máquina ya funcionaba, se comenzó a investigar la parte de los códigos numéricos para realizar programas, ya que no se tenía conocimiento sobre ello.
- En un editor de texto se hicieron los primeros programas que realizaban operaciones sencillas como trazado de líneas y círculos.
- Se procedió a investigar lo relativo al diseño gráfico y su posterior conversión a código numérico para realizar tareas más complejas.

ANTECEDENTES

La tecnología del control numérico, como es conocida hoy día, surgió a mediados del siglo XX en el año 1952, asociada a la Fuerza Aérea de los EUA (USAF) y a los nombres de John Parsons y del Instituto Tecnológico de Massachusetts (MIT).

No fue aplicado a la manufactura de produccción sino hasta inicios de los 60's, pero el verdadero auge vino en la forma del control numérico computarizado (CNC) alrededor del año 1972 y una década después con la introducción de microcomputadoras asequibles.

La historia y desarrollo de esta tecnología ha sido bien documentada en varias publicaciones. En el campo de la manufactura y particularmente en el área del maquinado de metales, la tecnología del control numérico ha causado algo así como una revolución. Aun en los días anteriores a que las computadoras se volvieran dispositivos comunes en cualquier compañía y en muchos hogares, las máquinas-herramienta equipadas con sistemas de control numérico encontraron un lugar especial en el mercado de maquinaria. La evolución de la microelectrónica trajo cambios significativos en el sector de la manufactura en general y en la industria del maquinado de metales en particular.

Definición del Control Numérico.

El control numérico se puede definir como la operación de máquinas herramienta mediante instrucciones específicamente codificadas al sistema de control de la máquina. Las instruccciones son una combinación de letras, dígitos y símbolos. Todas las instrucciones están escritas en un orden lógico y de una forma predeterminada. El conjunto de instrucciones necesarias para maquinar una pieza se denomina programa de control

numérico (CN) o programa de control numérico computarizado (CNC) . Tal programa puede ser guardado para uso futuro y usado repetidamente para lograr los mismos resultados de maquinado en cualquier momento.

En estricta adhesión a la terminología hay una diferencia en el significado de las abreviaturas CN y CNC. El CN se refiere a la antigua y original tecnología del control numérico, mientras que la abreviatura CNC se refiere a la más moderna tecnología de control numérico computarizado, que es un sucesor del primero.

Los sistemas de CN usaban generalmente tarjetas perforadas o cintas magnéticas en las que estaba grabado el programa y cualquier modificación requerida implicaba el cambio de éstas; mientras que en los posteriores sistemas de CNC se utilizó una computadora en la cual se grababan los programas; lo cual dio una mayor flexibilidad para poder realizar cambios con resultados inmediatos ya que sólo se trataba de un archivo guardado en la memoria.

Tesis desarrolladas en la UNAM sobre el CN

En cuanto a las tesis realizadas sobre el control numérico, se revisó la mayoría de ellas y a continuación se describe brevemente las que se han considerado de mayor relevancia:

- En 1966 se elaboró "Tecnología del control numérico", en donde se explican los sistemas y elementos que integraban los equipos de control numérico. La aplicación de las computadoras en estos sistemas, algunos métodos matemáticos para la preparación de datos en programas de control numérico y un estudio relacionado con la justificación económica de estos equipos.
- En 1974 un grupo de estudiantes elaboró el trabajo denominado " Automatización por control numérico de máquinas-herramienta", en donde muestran el diseño de un

sistema electrónico para el control de una máquina de tres ejes, en el cual se ingresaban coordenadas mediante un teclado y éstas eran manejadas para producir movimiento en unos servomotores, que mediante un detector de aproximación y comparadores, reducían la velocidad de éstos cuando se acercaban a las coordenadas comandadas. La tesis muestra diagramas de todo lo anterior, aunque aparentemente no se implementó en una máquina-herramienta.

- En 1992 la tesis "Diseño y contrucción de un sistema electrónico para la operación y control de máquinas herramientas de control numérico", presentó la creación de un programa donde se ingresaban distancias a recorrer en ejes coordenados y se enviaban datos a través del puerto serial de una computadora a un microcontrolador 8751.
- En 1998 el trabajo "Diseño y construcción de un sistema automático de pirograbado asistido por computadora", muestra el diseño de un sistema con movimiento en tres ejes (*xyz*) controlado por un circuito basado en microcontroladores PIC's que recibían señales de un programa creado por los tesistas y que titularon PIROPLOT, que utilizaba archivos en formato PRN creados al momento de mandar a imprimir un archivo, pudiendo transformarse elementos de paqueterías como AutoCAD, CorelDraw o Paint. Se presenta el código fuente de PIROPLOT y de los microcontroladores, pero no resultados.
- En 2004 el trabajo "Automatización de una máquina herramienta convencional a máquina de control numérico mediante microcontroladores", muestra el diseño del controlador de una máquina utilizada para barrenar tarjetas de circuito impreso (PCB) en el laboratorio de mecatrónica de la Facultad de Ingeniería. Asímismo se elaboró un programa en Visual Basic que cargaba un código creado por la paquetería PROTEL, el cual contenía las coordenadas donde requería barrenarse. Se presentó el código fuente del programa y de los microcontroladores, pero no muestras de los resultados de maquinado.

Otros trabajos similares

En la actualidad existe sofisticada maquinaria que utiliza los sistemas de CNC en aplicaciones industriales. También existen proyectos de aplicación de control numérico en la construcción de máquinas del tipo *hágalo usted mismo*, que son de bajo costo para apoyo a talleres y empresas pequeñasy otros que se aplican para modernizar máquinas industriales que eran manuales originalmente o bien eran de CN, pero cuyos controladores ya son inservibles u obsoletos. Algunos de estos proyectos se presentan a continuación:

- El libro *CNC Robotics, Build Your Own Workshop Bot*[1], describe detalladamente la construcción de una fresadora de tres ejes así como de sus controladores de motores a pasos. Al final del libro se muestran pruebas y resultados.
- En el trabajo *Implementation of a Linux-Based CNC Open Control System*[2], los autores muestran la implementación del programa EMC en una pequeña fresadora de tres ejes que utiliza controladores comerciales y servomotores . Se detalla la arquitectura del programa (i.e. - módulos, interfaces de usuario, ejecutores de tareas, etc.) y se presentan los resultados en el maquinado de unos rostros. *Figura i*.

1 Williams, G., *CNC Robotics, Make your own workshop bot*. EUA: Mc.Graw Hill, 2003

2 Disponible en http://bib.irb.hr/datoteka/421305.209-Staroveski-Brezak-Udiljak-Majetic-CIM_2009.pdf

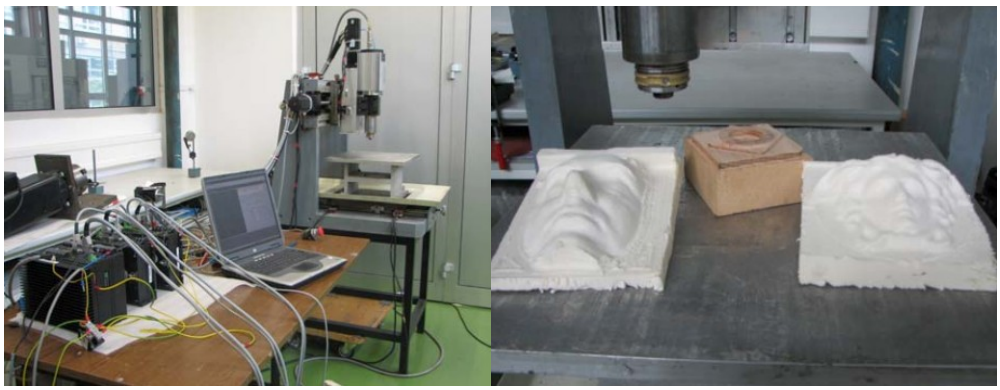


Figura i: Máquina y resultados de maquinado

- En la página http://www.stmental.net/~dfoster/emco se muestra el proyecto de dos jóvenes para modernizar un pequeño torno *EMCO Compact 5 CNC*. Hicieron la tarjeta para comunicación con el puerto paralelo de la PC y utilizaron el controlador de motores a pasos original de la máquina. *Figura ii*.

Figura ii: Modernización de torno EMCO

- En http://www.dalton.ax/stepper/ el autor muestra el diseño e implementación de un controlador de motor a pasos unipolar que trabaja a 10 A y 100 V para modernizar una fresadora industrial de CN, cuyos controladores originales se encontraban en pésimo estado. Figura iii.

Figura iii: Diseño de controlador para fresadora industrial

Además, existe un gran número de páginas electrónicas hechas por personas bien intencionadas que sin afán de lucro muestran información para realizar controladores y construir máquinas, sin contar, igualmente, con la enorme cantidad de foros relacionados con el CNC, en los cuales se pueden hacer preguntas y recibir respuesta por parte de los miembros. Esta es, en parte, la fuente de la cual se nutrió el proyecto que aquí se presenta.

CAPÍTULO 1 - DISEÑO ELECTRÓNICO

El funcionamiento general de la máquina comienza en una computadora que envía señales a través del puerto paralelo[3], desde el cual puede enviar y recibir señales. En este caso se ha utilizado para enviar las señales requeridas para activar los motores a pasos de cada eje y recibir la señal de switches límite y parada de emergencia. Las señales enviadas a los motores son a su vez recibidas por tarjetas que las interpretan y proveen la potencia que requieren los motores. Esto se muestra en la *Figura 1.1*

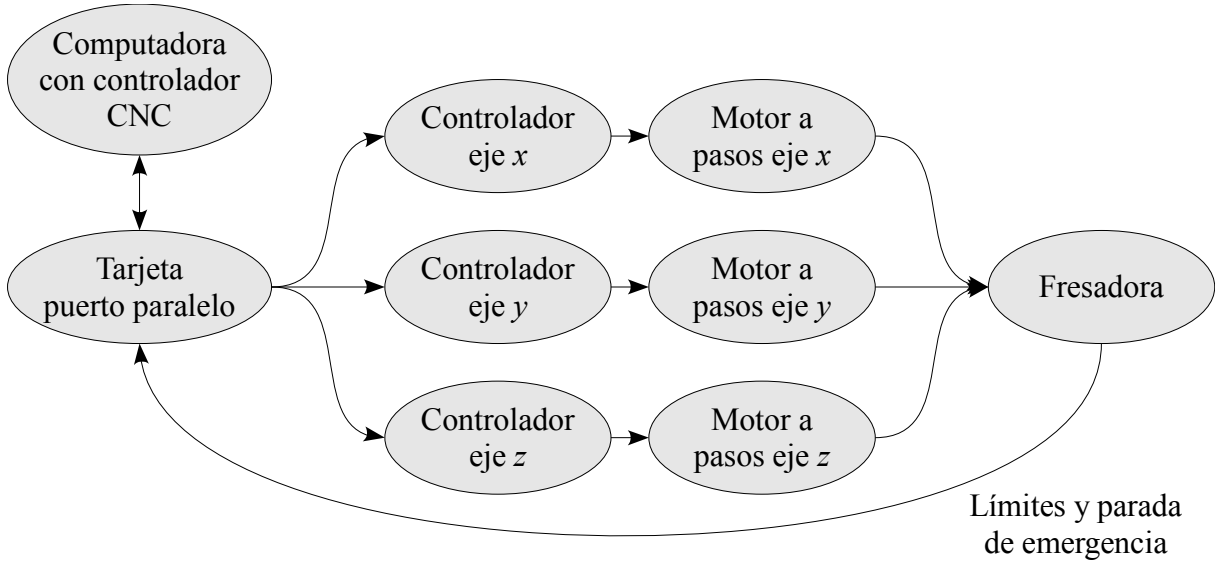


Figura 1.1: Componentes de la máquina CNC

En principio se aborda lo relativo al programa controlador de la máquina CNC, que es el *EMC2*, abordando aspectos como su obtención, instalación, requerimientos del equipo, etc., y después se describen los circuitos realizados para controlar la máquina. Para una explicación sobre el funcionamiento de los motores a pasos, ver ANEXO 1.

3 Aunque también hay algunos programas y controladores que utilizan el puerto USB como medio de comunicación.

El EMC (*Enhanced Machine Controller*) es una paquetería para controlar máquinas-herramienta, desarrollada por el Instituto Nacional de Estándares y Tecnología de los EUA (NIST). Puede controlar tanto servomotores (que se configuran mediante un control PID) como motores a pasos. Se utiliza uno o más puertos paralelos de conector DB-25 o tarjetas especializadas que se conectan a la tarjeta madre de la computadora en uno de sus puertos PCI y permiten controlar un mayor número de ejes coordenados, así como señales de entrada y salida. En este caso puede utilizarse el programa como un PLC que utiliza el lenguaje clásico y diagramas de lógica escalera.

Es un programa de software libre que trabaja en el sistema operativo Ubuntu en versiones de Soporte de Largo Plazo LTS (*Long Term Support*). Éstas han sido la 6.06, 8.04 y la 10.04 (que corresponden al año en que han sido lanzadas: 2006, 2008 y 2010, respectivamente).

En su sitio www.linuxcnc.org se encuentran, entre otros, los enlaces para descargarlo, la documentación (manuales para el usuario) y un foro para comunicarse con otros usuarios del programa.

Instalación – puede hacerse de las siguientes maneras, todas ellas explicadas a detalle en la parte correspondiente del sitio:

- Descargar una imagen de disco ISO que es copiada directamente en un CD. Ésta contiene el sistema operativo Ubuntu junto con el EMC2.
- Descargar un *script* que instala el programa automáticamente en el sistema operativo. Esta opción requiere de conexión a internet.
- Compilar el programa.

Documentación – se refiere a la información contenida en los manuales del usuario y que

indica a detalle el uso del programa en cada uno de sus módulos:

- Developer Manual (Manual del Desarrollador), 90 pp.
- Getting Started (Comenzando), 40 pp.
- Integrator Manual (Manual del Integrador), 270 pp.
- Manual Pages (Manual), 189 pp.
- User Manual (Manual de Usuario), 159 pp.
- HAL user manual (Manual de Usuario de HAL), 61 pp.

De especial ayuda para este trabajo fueron el *Getting Started* y el *User Manual*. Del primero, la información correspondiente a la instalación y configuración del programa y del segundo la introducción a la programación en código G, así como la referencia completa sobre los códigos soportados por el programa.

Requerimientos del equipo – en el manual *Getting Started* pueden verse los requerimientos mínimos del equipo para utilizar el programa:

- Procesador a 700 MHz x86 (se recomienda a 1.2 GHz x86).
- 384 MB de memoria RAM (se recomienda de 512 MB a 1 GB)
- Disco duro de 4 GB
- Tarjeta de gráficos capaz de soportar una resolución de 800x600 pixeles que no utilice los drivers de propietario NVidia o ATI, y la cual no sea un chip de video dentro de la tarjeta madre que comparte memoria con el CPU.

Interfaces de usuario – es el ambiente de trabajo en el cual se cargan y ejecutan los programas en código G. Son cuatro: AXIS, TkEMC, MINI y KEYSTICK. AXIS es la más popular y moderna de las cuatro y permite la visualización del código antes y durante la ejecución. TkEMC y MINI van dibujando la trayectoria del código mientras se ejecuta y KEYSTICK, la

más sencilla, no provee visualización ni antes ni durante la ejecución del código.

Modo de operación – es la manera en la cual se trabaja en cualquiera de las interfaces antes mencionadas y son tres:

- Automático – se ejecuta un programa en código G. Puede detenerse, pausarse y reiniciarse.
- MDI – es un modo en el cual se ingresan códigos línea por línea y son ejecutados de esa misma manera.
- Manual – permite mover los ejes coordenados (uno a la vez) al presionar una tecla o el cursor del mouse.

Una sesión típica en cualquiera de estas interfaces consiste en colocar los ejes en la posición cero o *home* mediante el modo Manual para poder ejecutar entonces un programa de control numérico en modo Automático o trabajar en el modo MDI.

Configuración – Para configurar una máquina que utiliza motores a pasos, se utiliza la opción *Stepconf Wizard*, en la cual se ingresa la función de los pines del puerto paralelo y parámetros de la máquina como medida de los ejes, paso del husillo, etc.

Límites, Desplazamiento y Home – cada eje tiene una cierta distancia de desplazamiento. El fin físico del recorrido es llamado *parada dura*.

Antes de la parada dura hay un *switch límite*. Si se encuentra un switch límite durante la operación, el EMC2 apaga el activador de los motores. La distancia entre el switch límite debe ser lo suficientemente larga como para permitirle al motor detenerse sin sobrepasarla.

Antes del switch límite hay un *límite suave*. Éste se pone en operación mediante software

después de hacer *home* durante la preparación de la máquina. Si un comando MDI o un programa en código G sobrepasara este límite suave, no se ejecuta y de estar en modo manual, el movimiento se termina en el límite suave.

El *switch de home* puede ser colocado en cualquier punto dentro del recorrido de las paradas duras. Dado el caso que las tarjetas (i.e. los circuitos) no desactiven la operación al tocar uno de los switches límite, puede utilizarse uno de ellos como switch de home.

La posición cero es la localización en un eje en donde es cero en el sistema coordenado de la máquina. Normalmente esta posición está entre los límites suaves.

Finalmente, la *posición de home* es la localización dentro del recorrido a donde se moverán los ejes después de realizar una rutina de home. Este valor debe estar dentro de los límites suaves y, en particular, la posición de home nunca debe ser igual a la del límite suave.

Habiendo visto la parte del software, ahora se analiza la del hardware, correspondiente a los circuitos con una breve explicación de su funcionamiento.

Tarjeta Paralelo

Es aquella que se conecta a la computadora a través de un cable paralelo con puerto DB-25 y recibe las señales enviadas por el EMC2. Para el control de cada motor a pasos pueden utilizarse dos pines; uno para la señal de paso y otro para la de dirección o bien tres, introduciendo el tercero una señal de activación/desactivación y que encuentra su uso en el caso de la parada de emergencia. En este caso se presiona el botón de emergencia y se ingresa la señal a través del pin *ESTOP In*, lo cual automáticamente emite la señal *ESTOP Out* para

desactivar los motores.

Hay distintas maneras de conectar las señales de salida de algún pin del conector DB-25 y dos de ellas se muestran en la *Figura 1.2*. En *I* se muestra una configuración que utiliza una resistencia de *pull-up* cuya función es definir claramente los estados alto o bajo de la señal (i.e.- 5 ó 0 volts). En *II* se tiene una configuración similar, pero con la adición de un búfer (el cual es un seguidor de la señal de entrada y se prende o apaga en correspondencia con ésta) que permite un aislamiento de la señal de salida con respecto al pin del puerto como una medida de seguridad además de permitir utilizar algunos miliamperes para alguna señal activadora de transistor, led, etc. Esta es la configuración utilizada en la tarjeta realizada y además se le agregó leds en cada señal dirigida a los motores para asegurarse que las señales estaban siendo recibidas por la tarjeta.

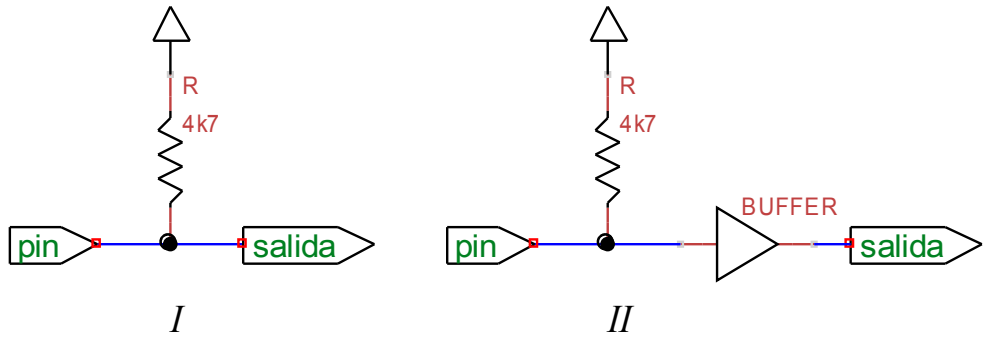


Figura 1.2: Conexión en pin de salida

Las señales de entrada pueden ser de switches límite, el botón de parada de emergencia, algún sensor, etc. Similar al caso de los pines de salidas, para las entradas puede utilizarse la resistencia de *pull-up* sola o con la adición de un búfer, tal como se muestra en la *Figura 1.3*. En este caso se utiliza un botón normalmente abierto y la tarjeta recibe una señal *alta* mientras éste no se presiona. Al hacerlo se cierra el circuito y el nodo de la resistencia conectado al pin

de entrada baja a tierra e ingresa entonces una señal *baja*.

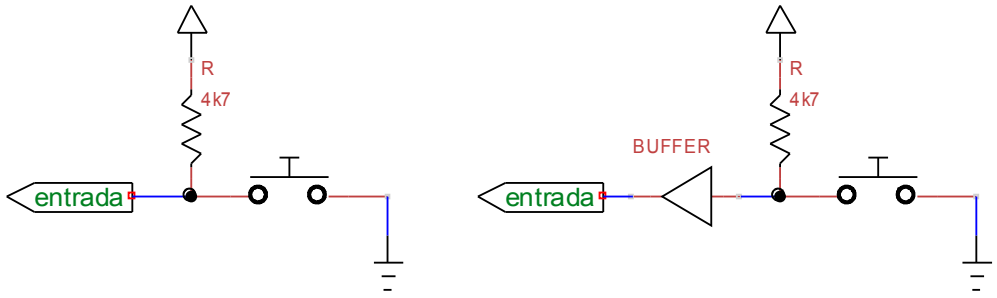


Figura 1.3: Configuración para pin de entrada

En el caso de los límites de los ejes pueden hacerse conexiones en serie o paralelo para tener en una sola entrada los límites mínimo y máximo del mismo e incluso el home. En la configuración paralelo se utilizan señales de Normalmente Abierto (NA) y en serie señales Normalmente Cerrado (NC). En la *Figura 1.4* se muestran ambas configuraciones para el caso de dos switches.

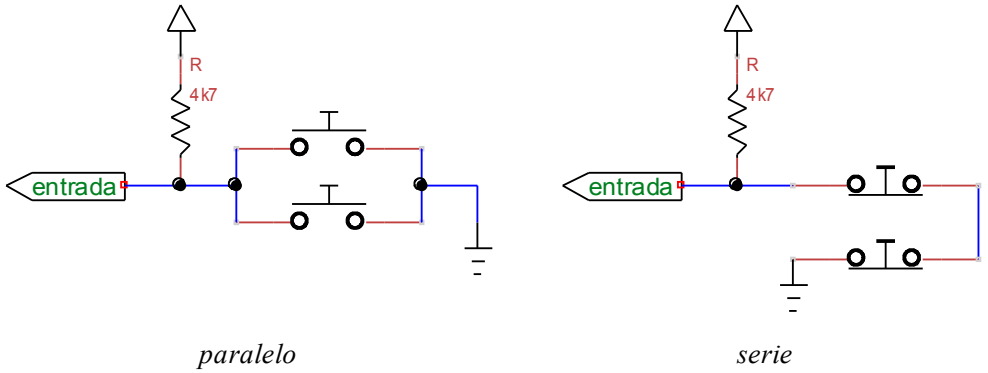


Figura 1.4: Configuraciones de conexiones de entrada

En la *Figura 1.5* se muestra la tarjeta paralelo realizada. Tiene salidas para tres ejes coordenados (*xyz*), cada uno con su correspondiente señal de paso, dirección y otra que

desactiva las tarjetas de los motores cuando se presiona el botón de parada de emergencia. Como entrada pueden utilizarse cuatro pines; tres para los límites de los ejes y el cuarto para la parada de emergencia. En el ANEXO 3 se presenta el diagrama de la misma.

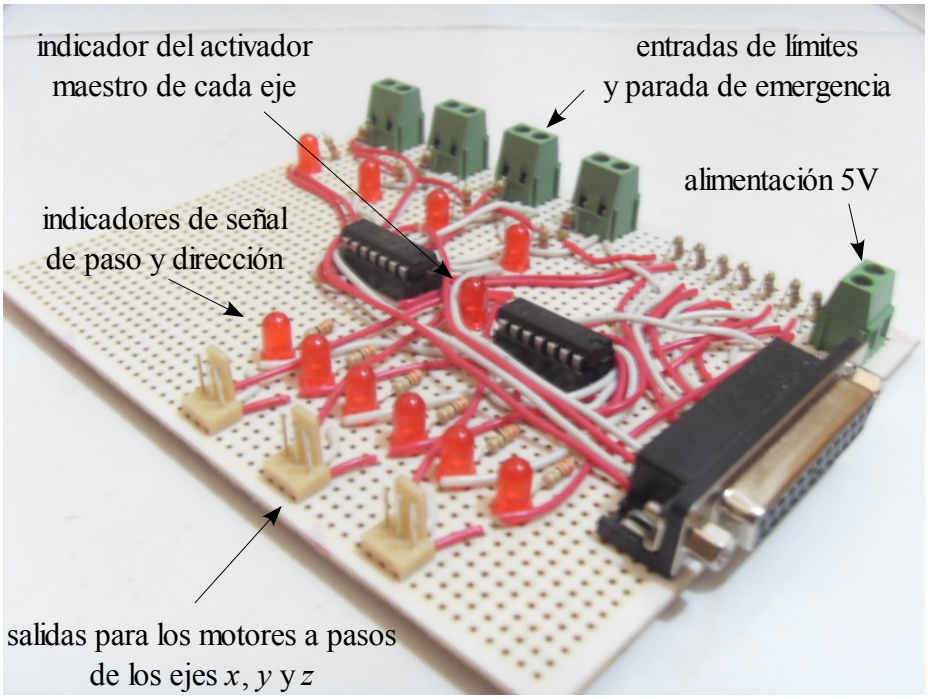


Figura 1.5: Tarjeta paralelo realizada

Tarjeta motor a pasos

Se han hecho tres tipos de tarjetas para el control de los motores a pasos que utilizan como base el Circuito Integrado (CI) L297, el cual recibe las señales de paso y dirección enviadas por el EMC y las convierte en la secuencia necesaria para controlar las fases de los motores a pasos, pero difieren en los elementos de la etapa de potencia. En principio se explicará brevemente el funcionamiento del CI L297 con información tomada de su hoja de datos (*datasheet*) y posteriormente se especificarán los detalles de cada tarjeta.

El CI L297

Es un controlador para motores a pasos que funciona como un traductor para generar las cuatros señales requeridas por un motor a pasos de dos fases utilizando fundamentalmente dos señales de entrada: paso y dirección. La señal de paso es un *tren de pulsos* que tiene como señal activa un bajo activo y la de dirección una señal del tipo encendido/apagado. En la *Figura 1.6* se muestra el diagrama de pines del CI y en la *Tabla 1.1* se detalla la función de cada uno de ellos.

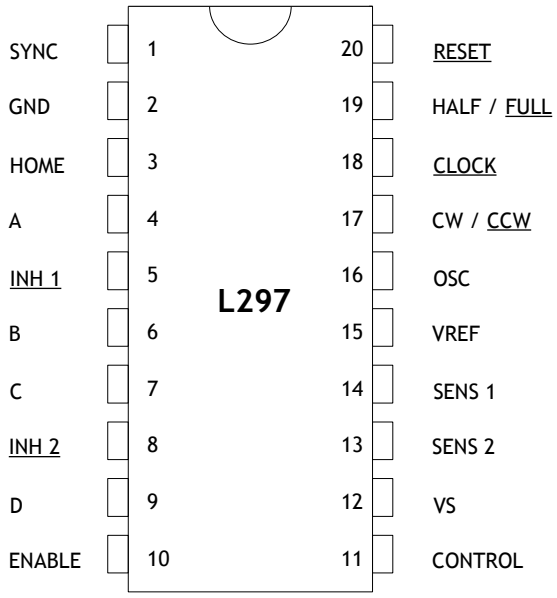


Figura 1.6: Diagrama de pines del CI L297

En los pines 17 y 18 se ingresan las señales fundamentales para el funcionamiento, que son las de dirección y paso respectivamente. En el 19 se determina el modo de operación: medio-paso o paso-completo. En los 13 y 14 se ingresan las señales de sensado provenientes de la etapa de potencia (que se verá más adelante) y que son comparadas con un voltaje ajustable ingresado en el pin 15 a una frecuencia determinada por el oscilador 16; que en su conjunto forman un regulador de corriente. El pin 2 es la conexión a tierra y en el pin 12 se ingresa la

señal de la de alimentación (5 volts). El 10 activa o desactiva el CI y de hecho es en este pin donde puede ingresarse la señal del puerto paralelo que es la de la parada de emergencia de salida (ESTOP Out). Del 4 al 9 se envían las señales con que se controlan los elementos de potencia. A, B e INH1 permiten controlar una de las fases del motor y C, D e INH2 la otra. El 11 determina la manera en que actúa el CI a la hora de desactivar las fases del motor. Con el 1 pueden sincronizarse varios L297's y así utilizar un solo oscilador.

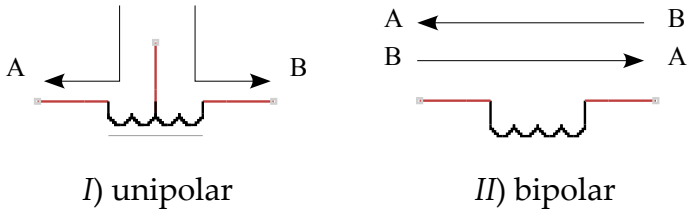
Tabla 1.1: Función de los pines del CI L297

Pin #	Nombre	Función
1	SYNC	Salida del oscilador del *chopper*. Las salidas SYNC de todos los L297 que serán sincronizados se conectan juntos y el oscilador que requeriría cada uno es sustituido por uno solo.
2	GND	Conexión a tierra.
3	HOME	Salida abierta de colector que indica cuando el L297 está en estado inicial (ABCD = 0101). El transistor está abierto cuando la señal está activa.
4	A	Señal directora para la etapa de potencia de la fase A del motor
5	INH1	Control inhibidor activamente bajo para el driver de las fases A y B. Cuando un puente bipolar es utilizado, esta señal puede ser utilizada para asegurar una caída rápida de la corriente de carga cuando una bobina es desenergizada. También es usado por el Chopper para regular la corriente de carga si la entrada de CONTROL está inactiva.
6	B	Señal directora para la etapa de potencia de la fase B del motor.
7	C	Señal directora para la etapa de potencia de la fase C del motor.
8	INH2	Control inhibidor activamente bajo para el driver de las fases C y D. Tiene la misma función que INH1.
9	D	Señal directora para la etapa de potencia de la fase D del motor.
10	ENABLE	Pin para activar el CI. Cuando está inactivo INH1, INH2, así como A, B, C y D se desactivan.
11	CONTROL	Entrada de control que maneja la manera en que funciona el chopper. Cuando está inactiva el chopper actúa en INH1 e INH2, cuando está activa trabaja en las líneas de fase A, B, C y D.

12	V_S	Alimentación de 5 [V_{CD}] .
13	$SENS_1$	Entrada para sensado de la corriente de carga de las etapas de potencia de las fases C y D.
14	$SENS_2$	Entrada para sensado de la corriente de carga de las etapas de potencia de las fases A y B.
15	V_{REF}	Voltaje de referencia para el circuito del chopper. El voltaje aplicado en este pin determina la corriente de carga pico
16	OSC	Un circuito RC conectado a esta terminal determina la frecuencia del chopper. Esta terminal es conectada a tierra en todos los dispositivos excepto uno para sincronizar configuraciones de L297's. La frecuencia es *f=1/0.69 RC.*
17	CW / CCW	Entrada de control para dirección horaria o antihoraria. La dirección física real del motor también depende de cómo se conectan las bobinas. Al ser sincronizado internamente, la dirección puede ser cambiada en cualquier momento.
18	CLOCK	Reloj de pasos. Un pulso activamente bajo en esta entrada avanza al motor un paso. El paso ocurre en el impulso de cada señal, es decir, cuando ésta comienza.
19	HALF / FULL	Entrada para selección de paso medio o completo. Cuando está activado, se activa la operación en medio paso y al estar desactivado, a paso completo. El modo One-phase-on full step es obtenido al seleccionar FULL cuando el traductor del L297 está en numeración par. El modo Two phase on full step es activado al seleccionar FULL cuando el traductor está en posición impar. (La posición de home es designada estado 1).
20	RESET	Entrada de reset. Un pulso activamente bajo en esta entrada reinicia el traductor a su posición inicial (estado 1, ABCD = 0101)

Las señales emitidas por el L297 pueden utilizarse para controlar un motor a pasos en modo unipolar o bipolar. Los motores unipolares tienen una bobina con tap central para cada una de sus fases y se necesita alternar el extremo hacia el cual fluye la corriente para activar los polos como se requiere, tal como se muestra en la *Figura 1.7 I*. En el caso de los bipolares se tienen bobinas en las que debe invertirse el sentido de flujo de corriente, tal como se muestra

en la misma figura , *II*.



I) unipolar *II*) bipolar

Figura 1.7: Tipo de fase de los motores a pasos

Para cambiar el sentido de circulación de corriente a través de las bobinas del motor en su modo de control bipolar se utiliza un *puente* H, tal como el mostrado en la *Figura 1.8*. En el puente de la izquierda la corriente circula de derecha a izquierda y en el de la derecha en sentido contrario. En un circuito estos *switches* son normalmente transistores o mosfets.

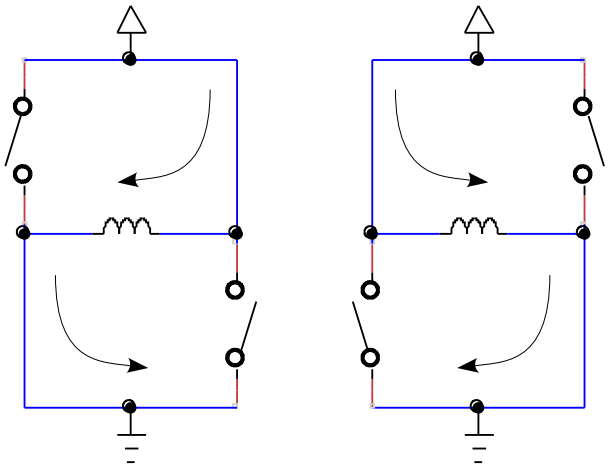


Figura 1.8: Estados de un Puente H

Como veremos a continuación, las señales *A* y *B* junto con o independientes de *INH1* emitidas por el L297 pueden utilizarse para controlar los dos estados de un puente H y así una de las fases del motor a pasos. Las señales *C*, *D* e *INH2* controlan otra fase.

El CI L298

Es un dispositivo que contiene dos puentes H y que es comúnmente utilizado junto con el L297 para controlar un motor a pasos de dos fases en modo bipolar. Igual que con el L297, la información presentada ha sido tomada de su hoja de datos. El diagrama de pines del L298 se muestra en la *Figura 1.9* y en la *Tabla 1.2* se muestra la función de cada uno de ellos

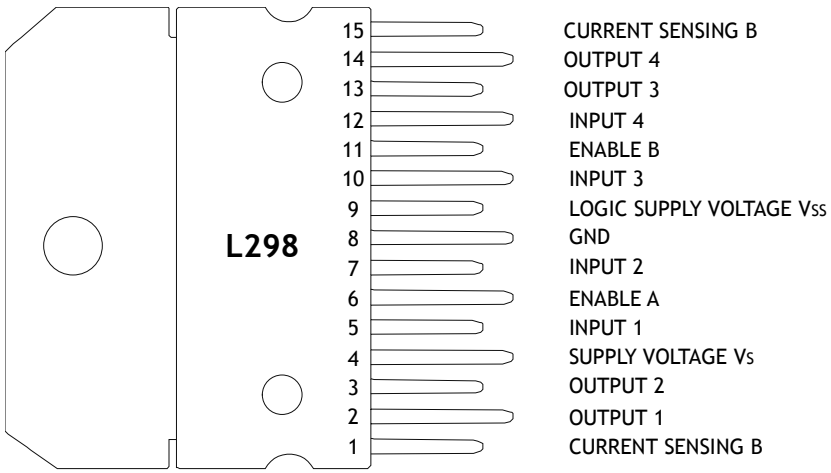


Figura 1.9: Diagrama de pines del CI L298

La función de los pines 5–7 es ingresar las señales activadoras de un puente H, cuyas salidas son 2 y 3. Lo mismo sucede con las entradas 10–12 con sus respectivas salidas 13 y 14 para el segundo puente H. En el 9 se ingresa el voltaje de lógico con que trabajan las señales de entrada (normalmente 5 volts) y en el 4 el voltaje de alimentación del motor, que puede ser de 5 a 36 volts. En los pines 1 y 15 salen señales que se conectan a resistores de sensado y que se utilizan para regular la corriente en las fases del motor.

Tabla 1.2: Función de los pines del CI L298		
Pin #	Nombre	Función
1; 15	Sense A; Sense B	Entre este pin y tierra es conectado el resistor de censado para controlar la corriente de la carga.
2; 3	Out1; Out 2	Salidas del puente A; la corriente que fluye a a través de la carga conectada entre estos dos pines es monitoreada en el pin 1
4	V_S	Alimentación de voltaje para la etapa de potencia. Un capacitor no inductivo de 100nF debe colocarse entre este pin y tierra
5; 7	Input 1; Input 2	Entrada compatible con voltajes TTL para el puente A.
6, 11	Enable A; Enable B	Entrada compatible con voltajes TTL. En estado bajo se deshabilita el puente A (enable A) y/o puente B (enable B).
8	GND	Tierra.
9	V_{SS}	Alimentación de voltaje para los bloques lógicos. Un capacitor de 100nF debe ser colocado entre este pin y tierra.
10, 12	Input 3; Input 4	Entrada compatible con voltajes TTL para el puente B.
13; 14	Out 3; Out 4	Salidas del puente B; la corriente que fluye a a través de la carga conectada entre estos dos pines es monitoreada en el pin 15.

Para analizar el funcionamiento del mismo más a fondo se muestra su diagrama interior en la *Figura 1.10.*

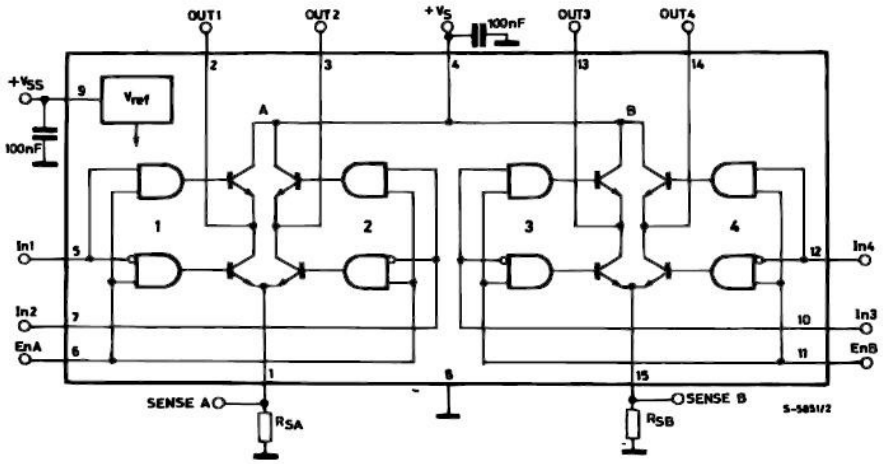


Figura 1.10: Diagrama interior del L298

Consta de 8 compuertas AND (que son activadas cuando sus dos señales de entrada están activadas) y el mismo número de transistores NPN que son activados mediante la señal emitida por esas compuertas.

Las entradas In1, In2 y EnA corresponden a las señales A, B e INH1 emitidas por el L297 para controlar una de las fases del motor y, por otro lado, In3, In4 y EnB corresponden a C, D e INH2 para la segunda fase. Las señales INH1 e INH2 producen una señal baja activa que sirve para *inhibir* el funcionamiento de su fase correspondiente.

Los estados del puente de la izquierda se generan de la siguiente manera:

- Las cuatro compuertas tienen utilizada una entrada con la señal de activación EnA. Las compuertas de la izquierda utilizan su segunda entrada con la señal In1, pero con la diferencia de que en una de ellas ésta se encuentra negada (representado por el punto blanco en la compuerta inferior). Lo mismo sucede en las compuertas de la derecha con la señal In2.

- Para el estado In1=1, In2=0, EnA=1 la compuerta izquierda superior recibe dos señales activas y se activa. La izquierda inferior recibe la señal In1 negada (cero lógico) y se desactiva. La compuerta derecha superior recibe la señal inactiva de *In2* y se mantiene apagada, pero la inferior, al recibir esta misma señal negada recibe entonces dos señales activas y se enciende. Este estado es igual al del puente de la derecha de la *Figura 27*.
- En el estado In1=0, In2=1, EnA=1 se activa la compuerta derecha superior e izquierda inferior, que corresponde al del puente de la izquierda de la *Figura 27*.

En la parte baja de cada puente H antes de la conexión a tierra se coloca una resistencia de sensado en la cual se genera un voltaje cuando está fluyendo corriente a través de ella. Este pin SENSE X registra este valor de voltaje y es conectado al CI L297 a su pin SENS X, que compara este valor de voltaje con un voltaje de referencia regulable. Estos voltajes son ingresados en un comparador que se desactiva cuando el voltaje de sensado supera a aquel de referencia y activa las señales para inhibir las fases, sea INH1 ó INH2. Esto permite controlar la corriente a través de las bobinas del motor.

Este CI puede soportar una corriente máxima de 2 amperes por puente, pero puede utilizarse una conexión en paralelo, en la cual se utilizan los dos puentes como si fuera uno solo y de esta manera puede soportar 3 amperes. Esta conexión (que es la que se ha utilizado para las tarjetas realizadas) se muestra en la *Figura 1.11*.

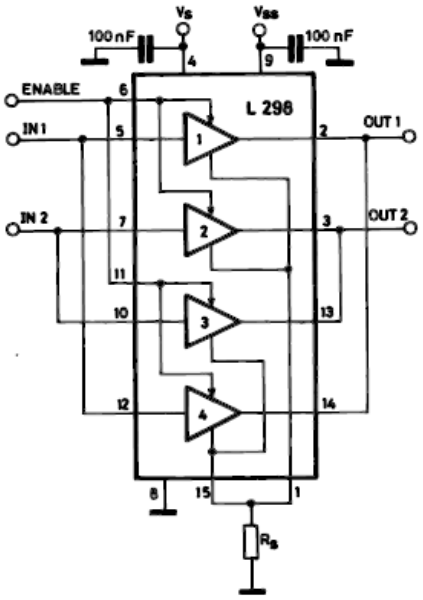


Figura 1.11: Configuración de conexión en paralelo del L298

La primer tarjeta realizada se hizo utilizando este CI y se muestra en la *Figura 1.12.* El circuito se basa en el presentado

en la hoja de datos tanto del L297 como del L298. En el ANEXO 3 se muestra el diagrama del circuito.

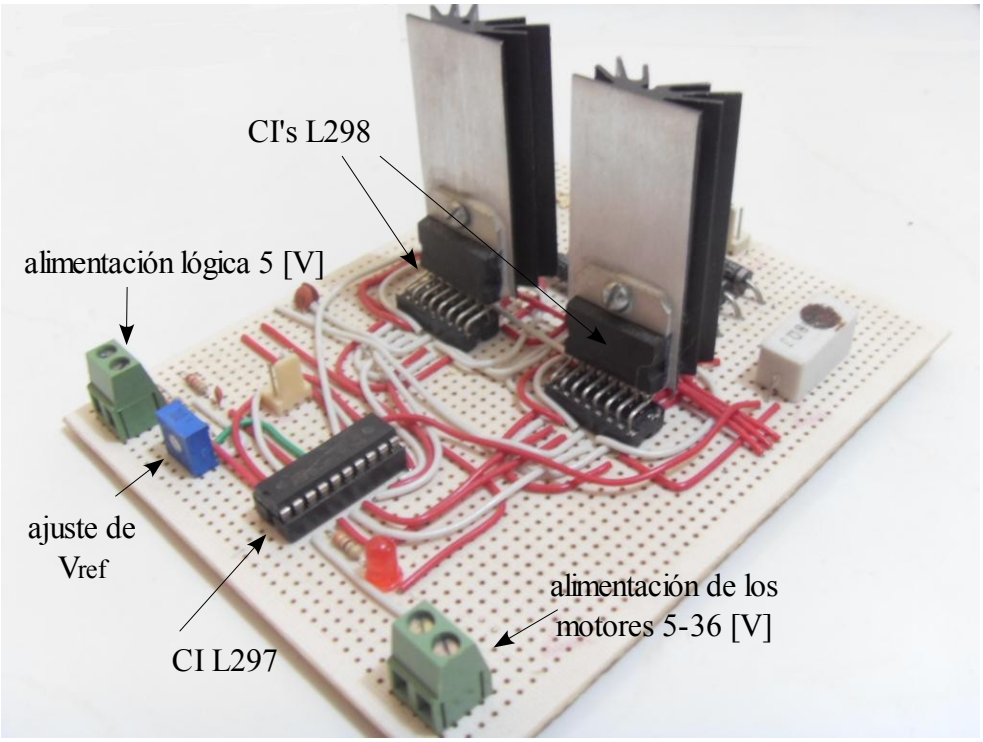


Figura 1.12: Tarjeta realizada con los L297 y L298

Puentes H Discretos

Si bien las tarjetas que utilizan el L298 han funcionado y de hecho se siguen ocupando, la idea de hacer otra tarjeta surgió debido a que en algún momento de las pruebas se conectó esa tarjeta con una fuente de laptop de 19.5V / 5 A y uno de los L298's se quemó. Se cambió el quemado y al volver a probarla, aun reduciendo la corriente lo más posible e incrementándola poco a poco, el otro corrió la misma suerte.

Se consideró necesario realizar una tarjeta que tuviera una mayor capacidad de corriente y para ello se realizaron los puentes H con componentes independientes. Después de buscar y

encontrar varios modelos, se decidió utilizar uno que utiliza MOSFETS de canal N y P que son activados mediante transistores NPN y PNP[4], tal como el mostrado en la *Figura 1.13*.

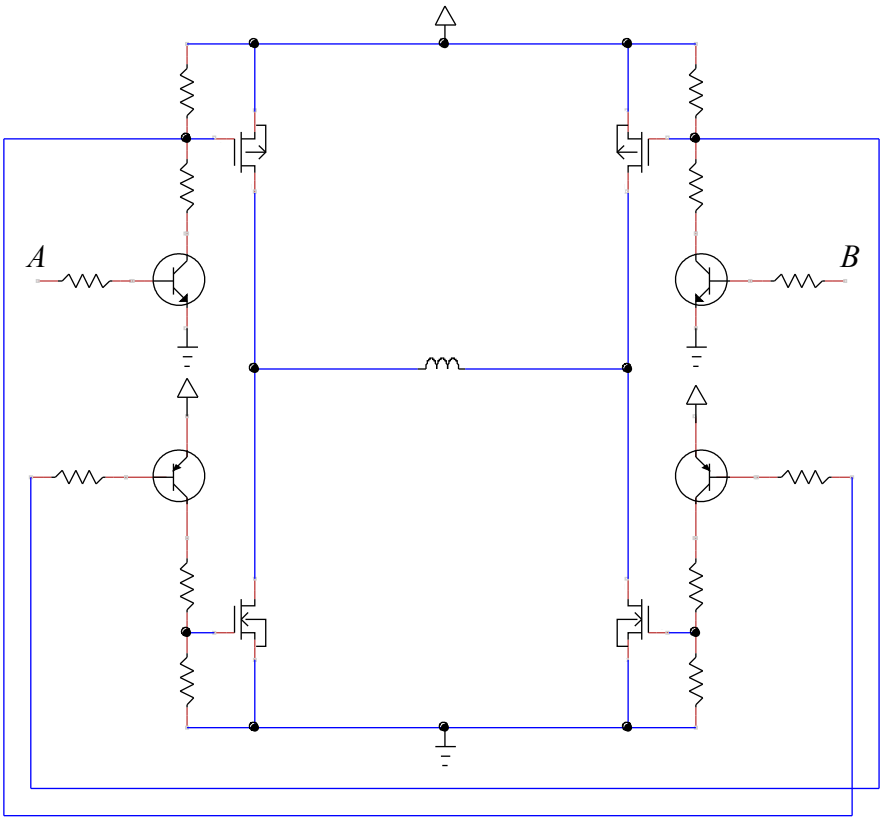


Figura 1.13: Puente H utilizado

Cuando se activa la señal A (uno lógico) se ingresa una señal en la base del transistor NPN (lado izquierdo superior) y el voltaje en el nodo donde se encuentran las dos resistencias *baja* a tierra, emitiendo una señal baja (cero lógico) que activa el mosfet de canal P. Este mismo nodo está conectado a la base del transistor PNP del lado derecho inferior, que al recibir esa señal baja-activa se enciende, provocando que el voltaje en el nodo de las resistencias *suba* a la alimentación (uno lógico) y enciende el mosfet de canal N conectado a dicho nodo, logrando finalmente el estado mostrado en el puente del lado derecho en la *Figura 1.8*. El estado en el cual B está activado produce el puente izquierdo de esa misma figura.

4 Basado en el circuito presentado en: http://www.cadvision.com/blanchas/hexfet/

De manera semejante al L298, las señales activadoras de los puentes están precedidas por compuertas AND que tienen por entradas A, B e INH1 para un puente y C, D e INH2 para el otro. La configuración para el primer caso se muestra en la *Figura 1.14*. Se ocupan dos compuertas por puente en vez de las cuatro del L298 debido a que se requierem úinicamente dos señales para controlarlo.

Figura 1.14: Configuración de las compuertas

La tarjeta realizada se muestra en la *Figura 1.15* y el diagrama del circuito en el ANEXO 3.

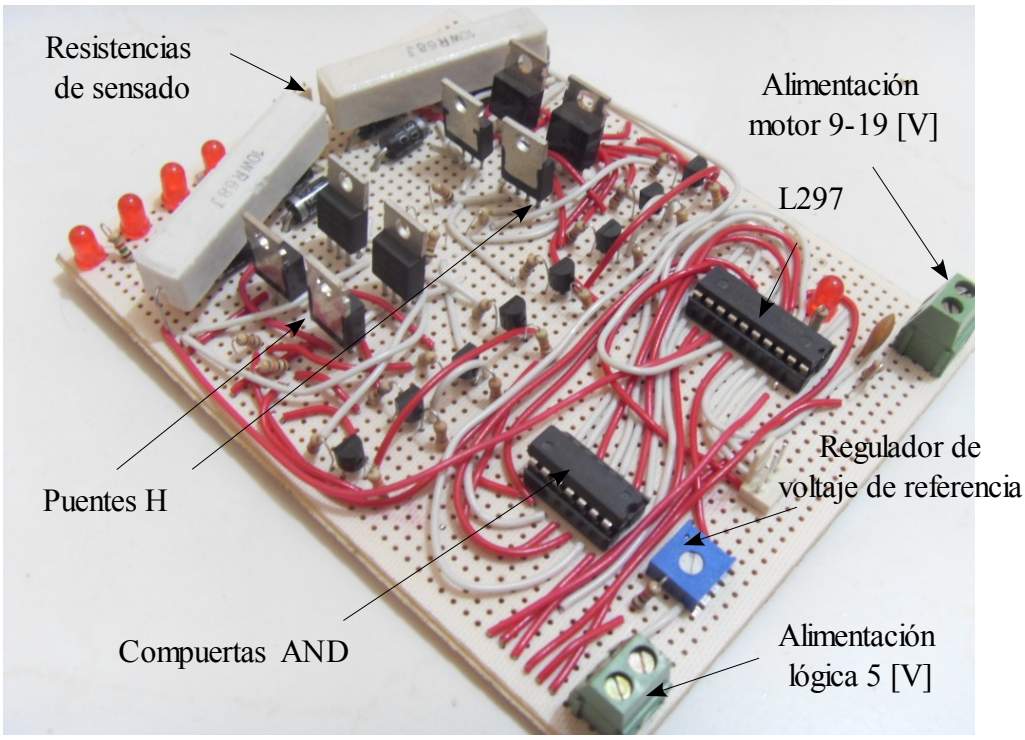


Figura 1.15: Tarjeta realizada con puentes discretos

Una tarjeta similar a ésta pero con un solo puente H fue realizada para controlar el sentido de giro de un motor de corriente directa y fue probada con la misma fuente de laptop mencionada anteriormente de 19.5 V / 5 A y tuvo un buen rendimiento. En treinta minutos de operación los mosfets se mantuvieron casi a temperatura ambiente.

Controlador Unipolar[5]

Este es el último modelo de tarjeta realizado y se ha hecho con el fin de poder utilizar tanto altas corrientes como un mayor voltaje en relación con la de los puentes H discretos, que no soporta voltajes mayores a 20; ya que este valor es el ingresado directamente en las compuertas de los mosfets.

Muy sucintamente puede decirse que la velocidad del motor a pasos está dada por el voltaje y la fuerza por el amperaje, por supuesto hasta un cierto límite. Además de lo anterior, el motor controlado en modo unipolar tiene un mejor rendimiento en altas velocidades debido a la menor inductancia de las bobinas.

En la Figura 1.16 se muestra la manera en que se controla una fase. El tap central de la bobina se conecta a la alimentación y se va alternando el extremo hacia el cual fluye la corriente al activar el mosfet correspondiente. Los extremos inferiores se conectan a la resistencia de sensado.

5 Basado en el circuito presentado en: http://www.dalton.ax/stepper/

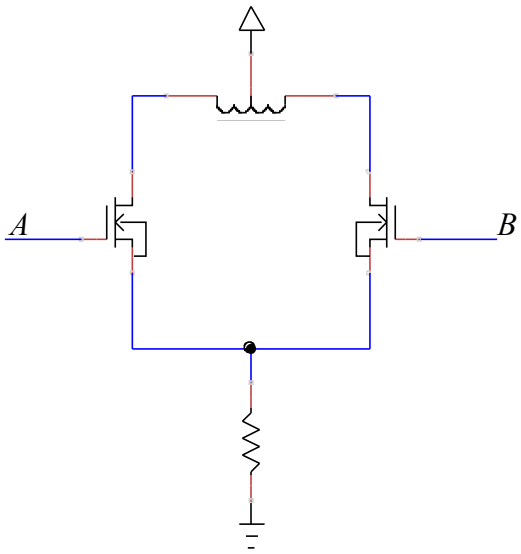


Figura 1.16: Control unipolar

La manera de activar los mosfets de canal N al ingresar un uno-lógico en su compuerta se muestra en la *Figura 1.17.* Las señales emitidas por el L297 se ingresan esta ocasión en compuertas NAND, cuyas salidas se ingresan en un activador de mosfets que funciona como un búfer inversor. Para activar convenientemente un mosfet se requiere regularmente al menos 7 V y el activador utilizado soporta hasta 15 V, representado en la figura como V_G.

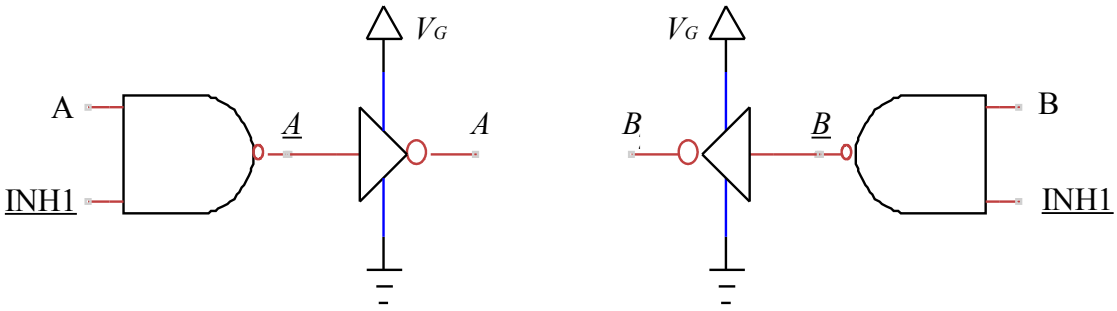


Figura 1.17: Activación de las compuertas

Esta tarjeta se muestra en la *Figura 1.18* y el diagrama del circuito en el ANEXO 3.

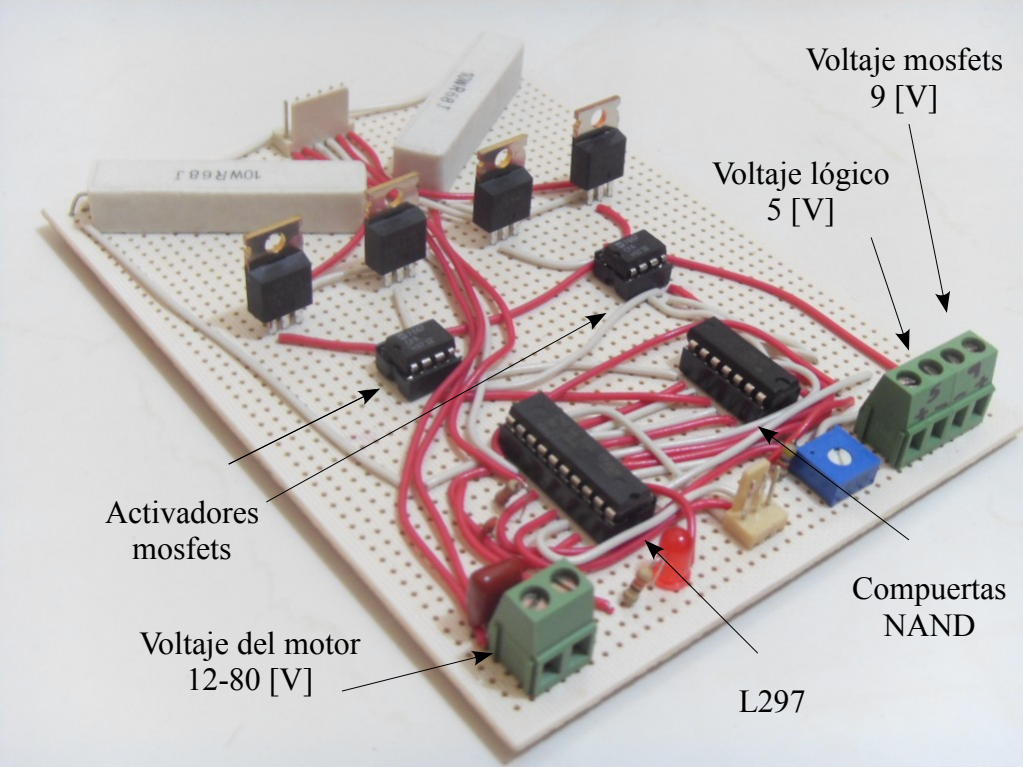


Figura 1.18: Controlador unipolar realizado

CAPÍTULO 2 - DESCRIPCIÓN DE LA MÁQUINA

La máquina fue hecha por el propietario del taller Luis Sánchez y se muestra en la *Figura 2.1*.

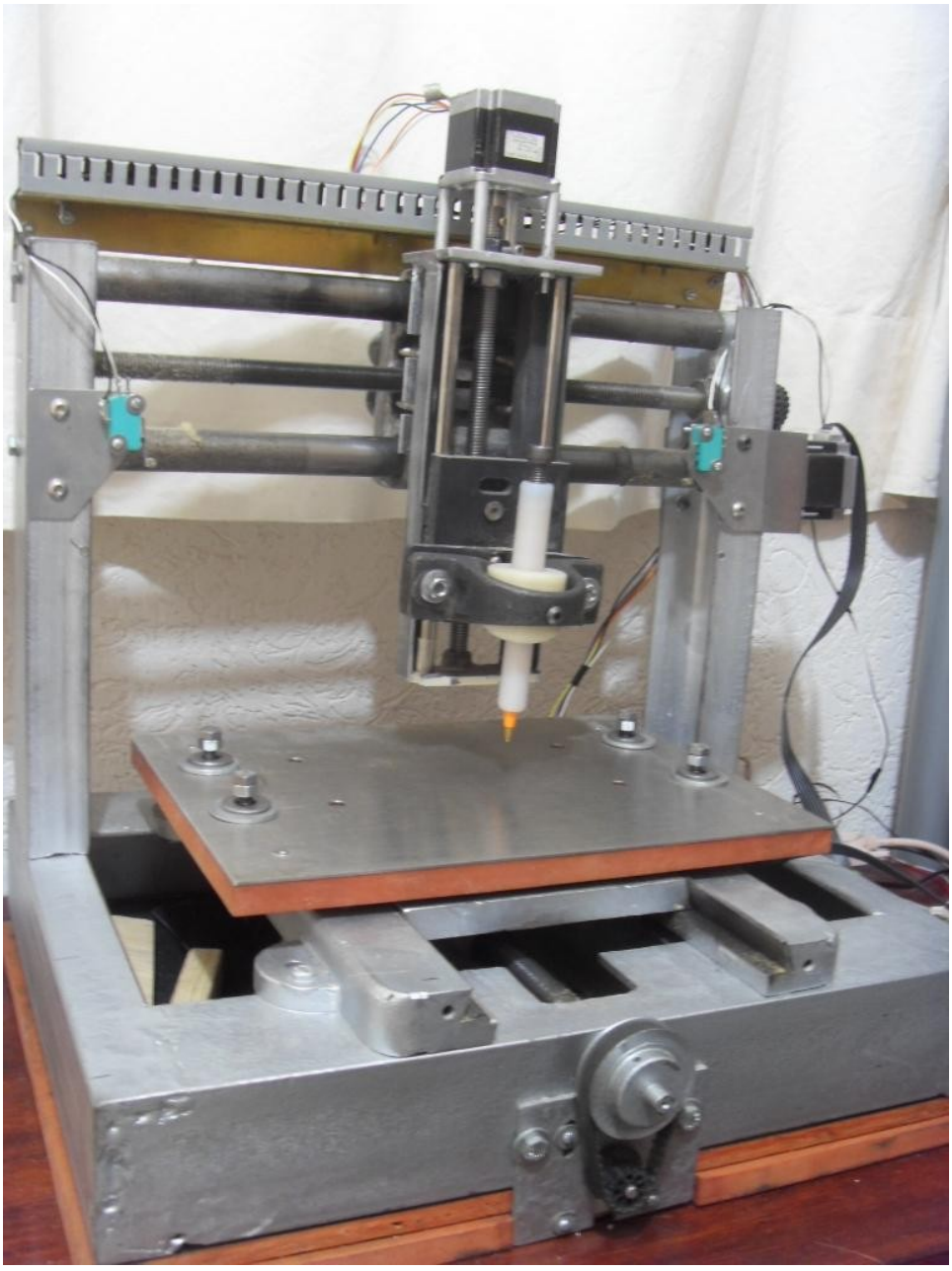


Figura 2.1: Fresadora construida

Algunos detalles al respecto son:

- De especial ayuda fueron los husillos utilizados para los ejes *x* y *y*, que eran originalmente de otra máquina.
- El marco ha sido hecho con canal de hierro de tres pulgadas.
- Los engranes (excepto uno) para acoplar los motores a pasos con los husillos fueron tomados de una impresora vieja.
- Los motores a pasos se compraron de medio uso en el tianguis de desperdicio mencionado a $50 pesos c/u.

Para las primeras pruebas fue de gran ayuda el dibujar los programas en una hoja de papel con una pluma, para lo cual se realizó el artilugio que se ve en la *Figura 2.1* y suple al *spindle*. El material es silicón y la pieza de enmedio y de mayor radio es insertada en una chumacera sin balero. Para el trabajado en madera se utilizó un router de 120V y 30 000 rpm que fue comprado igualmente de medio uso. Se coloca ahí mismo pero detenido con una pieza de solera deformada para acoplarse al mismo, mostrado en la *Figura 2.2*.

Figura 2.2: Router utilizado

Las especificaciones de la máquina se muestran en la *Tabla 2.1*.

Tabla 2.1: Especificaciones de la máquina

Parámetro	**Valor**
Número de ejes	*3 (xyz)*
Número de herramientas	*1*
Dimensión de la base	*350x250 mm*
Desplazamiento máximo del eje *x*	*250 mm - 10 pulg*
Desplazamiento máximo del eje *y*	*100 mm - 4 pulg*
Desplazamiento máximo del eje *z*	*50 mm - 2 pulg*
Velocidad del Spindle	*30 000 rpm*
Velocidad de avance	*750 mm/min - 30 pulg/min*

Se ha utilizado una laptop con un procesador Athlon a 2500 Mhz, 512 MB de memoria RAM y con salida de puerto paralelo DB-25.

La configuración de los ejes arroja algunos datos como los mostrados a continuación, que corresponden a los del eje *x*:

- Tiempo para acelerar a máxima velocidad: 0.0067 s
- Distancia para acelerar a máxima velocidad: 0.0007 pulg
- Frecuencia del paso a máxima velocidad: 1257.1 Hz
- Escala del eje: 6285.7 pasos / pulg

Puede verse que si se realizan 6285 pasos por pulgada entonces el avance mínimo por paso es de 0.00016 pulg (0.00404 mm), que sería la distancia mínima que puede desplazarse la máquina; pero esto depende a su vez de factores mecánicos como fricción, juego mecánico en

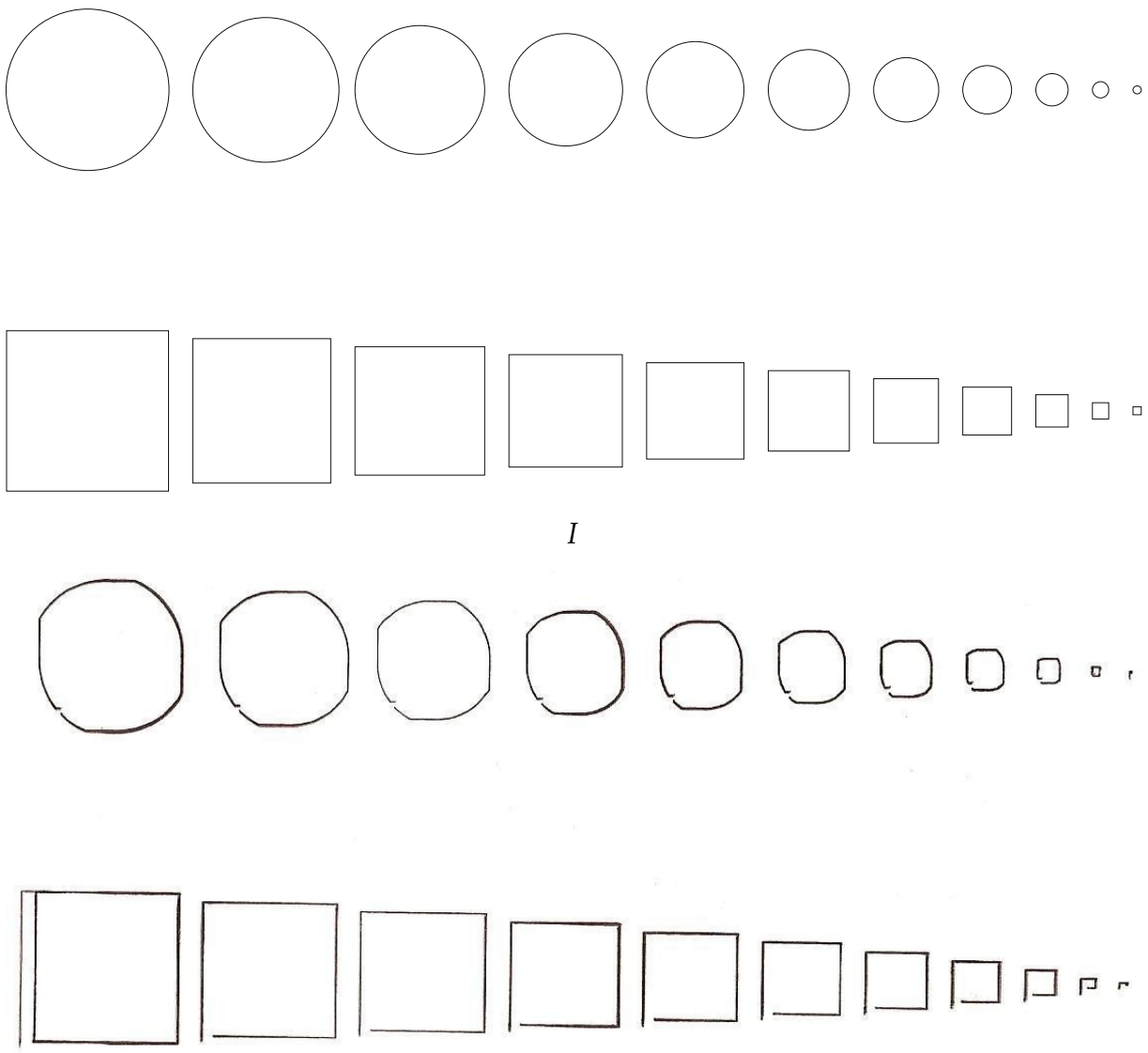
los elementos de tracción, etc.

En este sentido se consideran comúnmente tres factores para tener una referencia del rendimiento real de un equipo, que son:

- Precisión – es la diferencia entre una medida comandada y la real lograda por el sistema.
- Repetibilidad – es el rango de posiciones logradas por un sistema cuando es comandado repetidamente bajo las mismas condiciones para que se mueva a una misma posición. La repetibilidad puede ser a su vez medida de manera unidireccional o bidireccional. En el primero caso se ignora la respuesta del sistema ante un cambio en el sentido de giro y en el segundo se mide la capacidad para colocarse en la misma posición por ambos lados.
- Resolución – es el mínimo movimiento posible que puede lograr el sistema.

Se hicieron dos prueba para conocer estos parámetros.

La primera consistió en dibujar círculos y cuadrados comenzando con un diámetro y lado, respectivamente, de 20mm y decrementándolos sucesivamente en 2mm hasta tener figuras de éste tamaño. Finalmente un último dibujo de 1mm de cada uno de ellos. El dibujo fuente y el resultado del *maquinado* se muestran en la *Figura 2.3 I* y *II*, respectivamente.



I

II

Figura 2.3: Prueba en círculos y cuadrados

El maquinado comienza en la parte izquierda inferior tanto en los círculos como en los cuadrados. En los cuadrados se inicia con el lado vertical con un movimiento de abajo hacia arriba y termina con el movimiento horizontal de derecha a izquierda. Puede verse que el punto final no es el mismo que el de inicio. Esta diferencia en distancias en lo cuadrados para los movimientos horizontales y verticales han sido medidas con un vernier y se han obtenido

los resultados mostrados en la *Tabla 2.2.*

Tabla 2.2: Medidas de los cuadrados [mm]

Cuadrado patrón	*x*		*y*		Δ *x*	Δ *y*
	longitud ida	longitud regreso	altura ida	altura regreso		
20	20	18.4	19.7	19.4	1.6	0.3
18	17	15.5	18.2	17.5	1.5	0.7
16	16	14.9	16	15.6	1.1	0.4
14	14	12.9	14.1	13.5	1.1	0.6
12	12.1	10.9	12.1	11.5	1.2	0.6
10	10	9	10	9.6	1	0.4
8	8	7	8.2	7.5	1	0.7
6	6	5	6	5.5	1	0.5
4	4.2	3.3	4.1	3.6	0.9	0.5
2	2.2	1.4	2.3	1.6	0.8	0.7
				promedio	1.12	0.54

Los *movimientos de ida* significan el primer movimiento en una cierta dirección hecho por la máquina y es en éste donde puede verse que la *precisión* para ambos ejes varía en el rango de los ±0.3mm. En los *movimientos de regreso* puede verse una deficiencia en la *repetibilidad bidireccional*, donde hay un error absoluto promedio de 1.12 y 0.54mm para los ejes *x* y *y*, respectivamente.

Además de lo anterior, la operación que se muestra se repitió en cinco ocasiones y puede verse que todas la figuras, exceptuando el cuadrado más grande, fueron dibujados sobre la misma línea. Esto representa una buena repetibilidad del equipo, aun cuando en cada una de ellas los cuadrados y círculos no se cerraran.

Finalmente para la *resolución* se ha hecho un barrido con incrementos de ¼ de milímetro con fines ilustrativos y también para evitar confundir éste parámetro con el de la repetibilidad bidireccional mencionado anteriormente. Esto se muestra en la *Figura 2.4*.

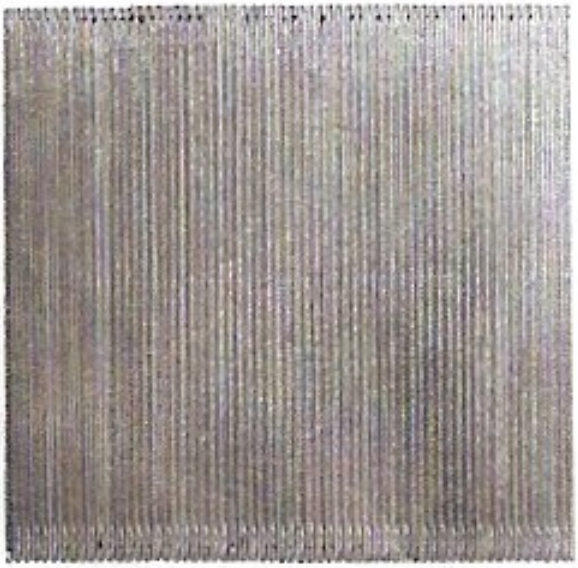


Figura 2.4: Barrido con avance de 0.25mm

CAPÍTULO 3 - DISEÑO DE PROGRAMAS

El Código G es el lenguaje entendido por las máquinas CNC las cuales son comandadas por una computadora en la cual hay un programa encargado de traducir este código a las señales eléctricas requeridas.

El capítulo se desarrolla mostrando algunos ejemplos de programas en el lenguaje de programación en código G y posteriormente algunas operaciones realizables con paqueterías gratuitas en las cuales se diseñan gráficamente las partes deseadas y son posteriormente transformadas a código numérico, lo cual representa los dos primeros pasos del diagrama mostrado en la *Figura 3.1*. El tercer paso consiste en cargar este código en la paquetería que controla la máquina CNC, representada en el tercer paso de la misma figura.

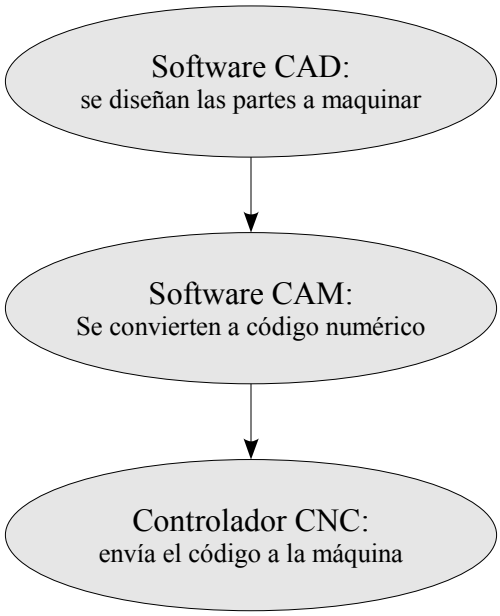


Figura 3.1: Procedimiento de diseño CAD-CAM-CNC

En la etapa para el proceso de CAD/CAM existen una gran cantidad de paqueterías y no es el propósito mencionarlas todas, sin embargo, para tener una noción sobre ellas se presenta el resultado de una encuesta realizada en el sitio *www.cnccookbook.com*, en la cual participaron alrededor de 200 personas respondiendo a la pregunta de cuál paquetería de CAD/CAM utilizan.

La persona que organizó los resultados decidió que sería conveniente colocar aquellas paqueterías con un precio superior a los $1000 dólares en un grupo llamado *PRO* y las de

precio menor o igual en otro grupo llamado *HOBBY*. Cabe aclarar que esto no implica que las personas que utilizan las paqueterías del segundo grupo lo hagan necesariamente como pasatiempo.

Los resultados se muestran en las F*iguras 3.2* y *3.3*.

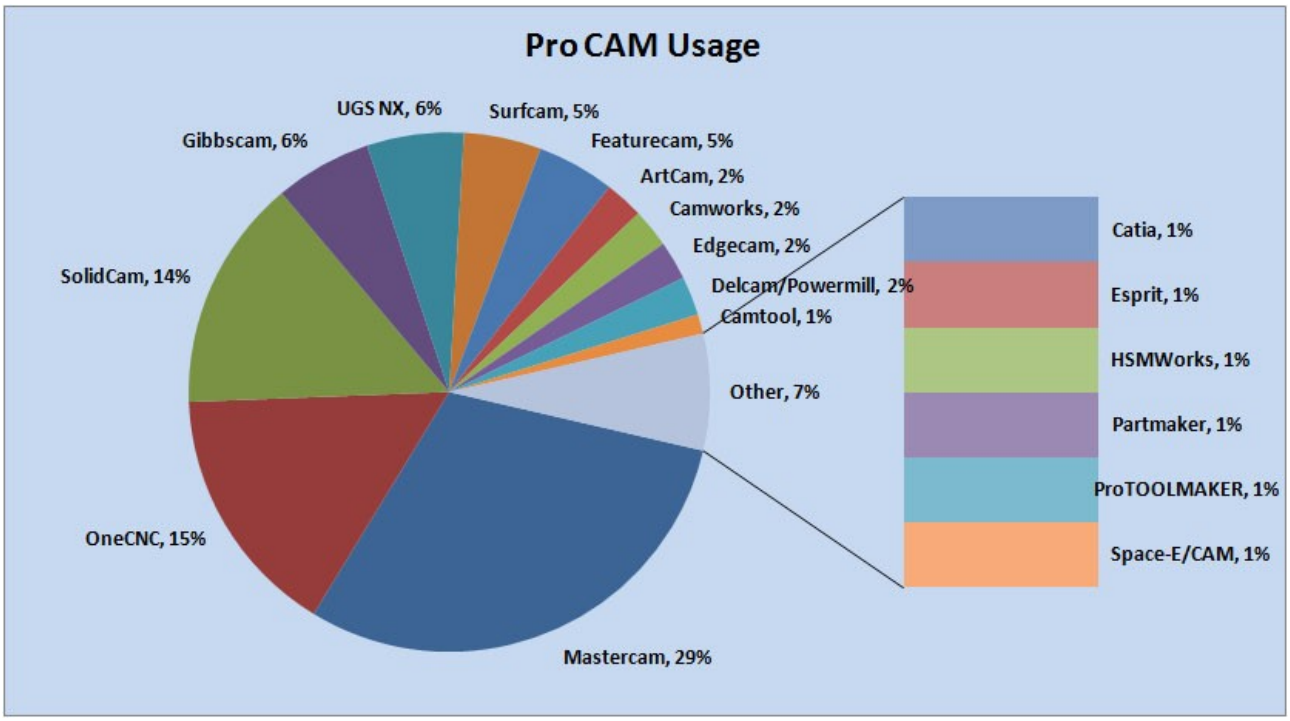


Figura 3.2: Resultados de programas PRO

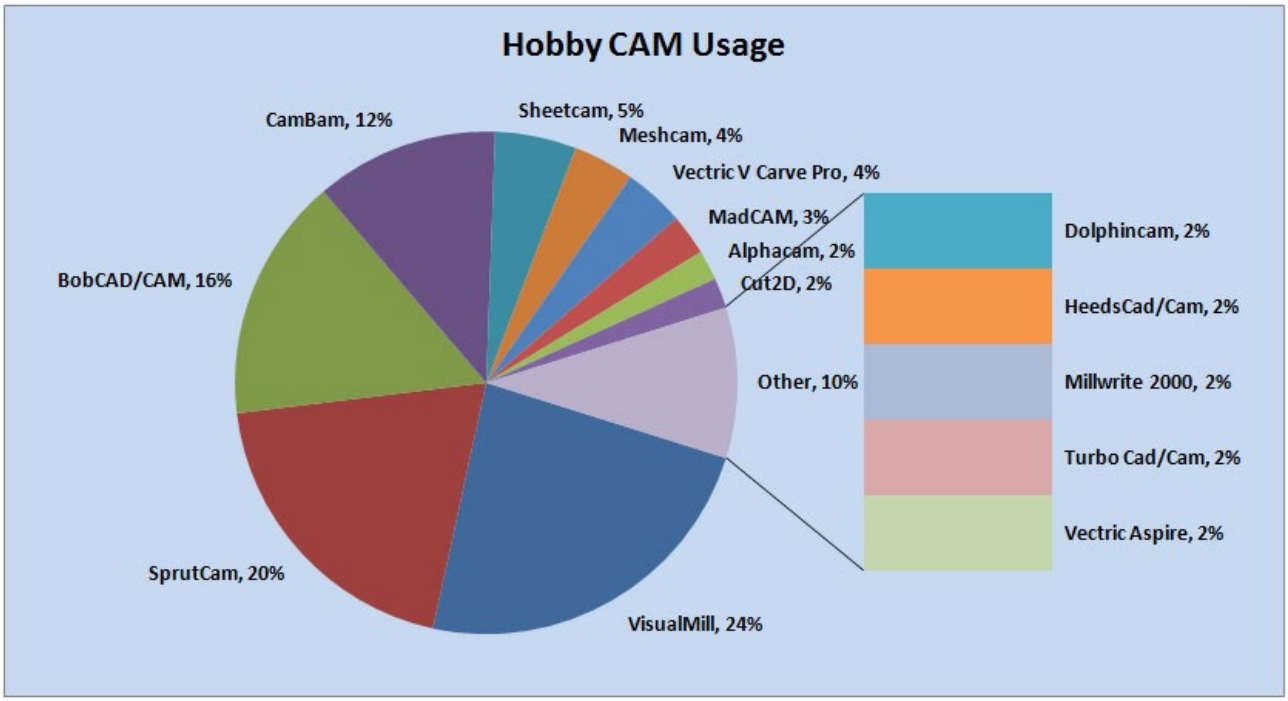


Figura 3.3: Resultados de programas hobby

Estos resultados, como los de cualquier encuesta, deben tomarse con cautela y se podría suponer que muchas personas que trabajan en áreas de manufactura de distintas compañías no supieron sobre la encuesta llevada a cabo en este sitio. Pero sí nos da una idea de la gran cantidad de paqueterías disponibles para este propósito y quizá una referencia de selección en algún momento. Finalmente, cabe mencionar que de las paqueterías utilizadas en este trabajo y que serán vistas más adelante, sólo CamBam aparece en estos resultados.

Para los controladores CNC existen igualmente una gran cantidad de paqueterías, de las cuales se hace una breve reseña que no pretende ser exhaustiva ni jerarquizante, sino simplemente mostrar algunas opciones:

- EMC2 – corre bajo el sistema operativo Ubuntu. Es software libre y su sitio oficial para

documentación, descargas, foro, etc. es *http://ww.linuxcnc.com*. Es la paquetería utilizada en este trabajo y ya se trató a detalle sobre ella en el capítulo anterior.

- Mach3 – es un programa que corre bajo Windows y que es relativamente popular a juzgar por lo visto en una gran cantidad de videos. Algunas tarjetas recientes que utilizan el puerto USB en vez del puerto paralelo están diseñadas específicamente para usarse con este programa. Su sitio es *http://www.machsupport.com*, desde el cual puede descargarse una versión de prueba y comprarlo por $175 dólares.
- KCAM – es otro programa que corre en Windows. Su costo es de $95 dólares y su sitio *http://www.kellyware.com*.
- TurboCNC – es un programa que corre bajo el sistema operativo DOS y controla hasta 8 ejes coordenados. Su distribución es *shareware*, que permite descargarlo y utilizarlo con todas sus características y sin tiempo limitado, esperando el autor a que si se decide utilizarlo se adquiera la licencia por $60 dólares. Su sitio es *http://www.dakeng.com*.
- CNC Pro – corre igualmente en DOS y recientemente se convirtió en un programa gratuito de fuente libre. Para descargarlo necesita uno registrarse en su grupo de yahoo *http://tech.groups.yahoo.com/group/CNCPro*.
- USBCNC – es un programa diseñado para utilizarse mediante el puerto USB con unas tarjetas producidas por la misma compañía. El sitio es *http://www.edingcnc.com*.

Ahora se pasará a la parte medular del capítulo. En la *Figura 3.4* se muestra un esbozo de algunas de las operaciones y procedimientos tratados a lo largo del capítulo para crear programas en código numérico.

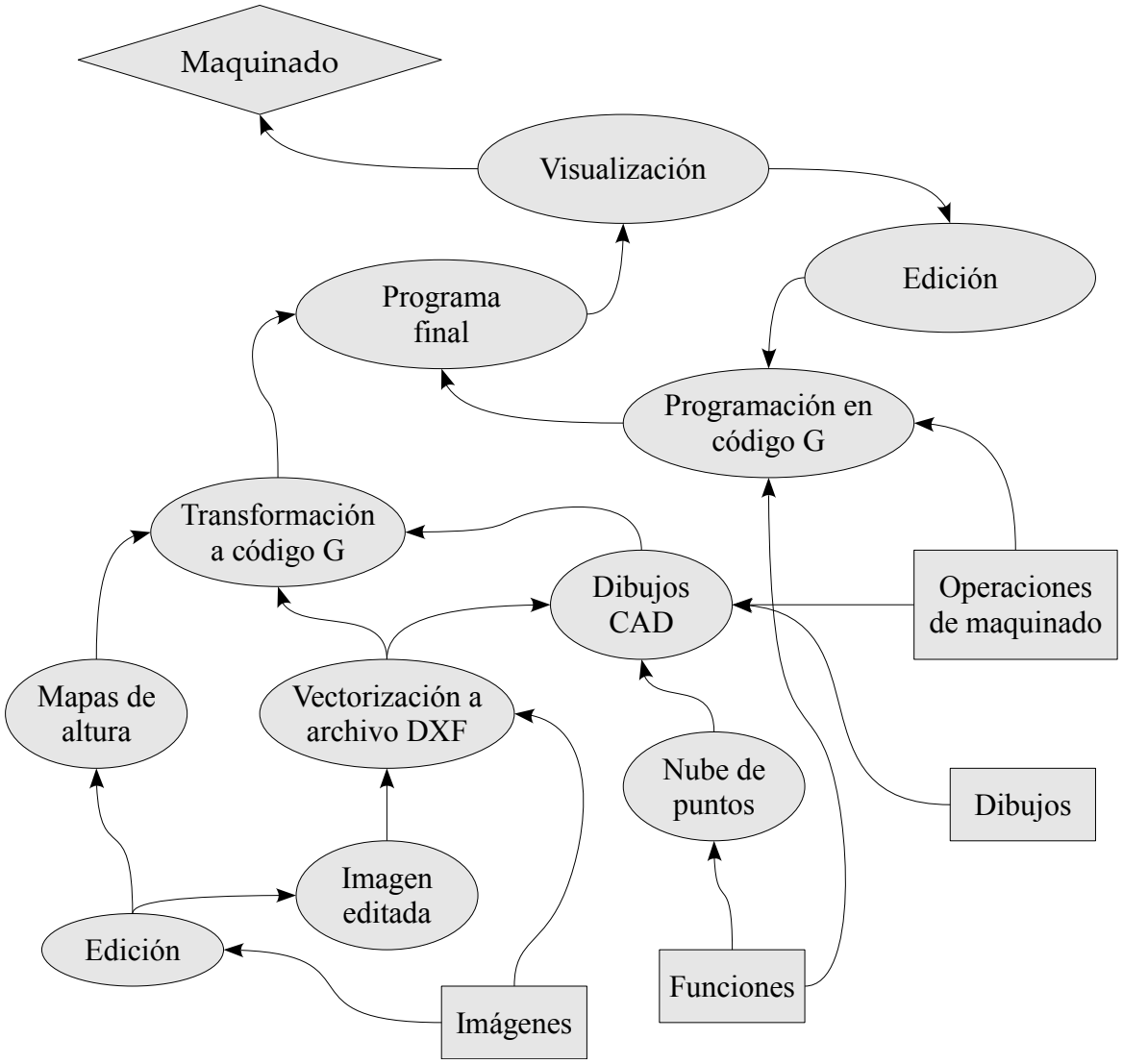


Figura 3.4: Bosquejo de procedimiento para creación de programas

En principio se mostrará cómo funciones matemáticas pueden ser fácilmente programables directamente en código G utilizando únicamente un editor de textos o bien ser obtenidas en una hoja de cálculo y utilizarse como una nube de puntos.

Posteriormente se muestran algunas técnicas de edición para que imágenes puedan ser utilizadas como base para un programa de control numérico. Un tratamiento permite obtener

mapas de altura que generan relieves y otro que éstas puedan ser vectorizadas, lo cual genera un archivo tipo CAD que puede ser un paso previo a su transformación en código G o bien para ser utilizadas en operaciones de maquinado más complejas.

Operaciones de maquinado en las cuales se requieren trayectorias de herramienta que cubran ciertas áreas o perfiles requieren de paqueterías especializadas.

Algunos de los métodos aquí mostrados se explican a detalle en el trabajo de Yohudi, disponible en su sitio *http://www.cnc4free.org*. Es un *ebook* gratuito de cerca de 400 pp. en donde detalla paso a paso la manipulación de imágenes y uso de la paquetería CNC ToolKit aplicada para operaciones en máquinas de 3 y 4 ejes.

Ejemplos de programas en código G

Al iniciar las pruebas de la máquina se escribieron programas en código G para realizar tareas simples. Para ello se recurrió al manual de usuario del EMC2, en el cual hay una parte que trata los aspectos más elementales de la programación en este lenguaje y posteriormente se exponen todos los códigos soportados. En el ANEXO 2 se presenta un resumen con los fundamentos de este lenguaje y se explican algunos códigos.

A continuación se presentan algunos de los programas realizados, especificando que un programa se escribe en un editor de textos, llamado justamente *Editor de textos* en Ubuntu o el *Bloc de notas* en Windows. Este archivo se guarda con la extensión NGC (Ej. Programa.ngc), que es la utilizada por el EMC2.

La mayoría de las paqueterías utilizadas en Windows para creación de código G (y que se verán más adelante) producen programas en código G con la extensión NC, la cual no es

reconocida por el EMC2 y por lo tanto es necesario cambiarles la extensión de NC a NGC.

Ejemplo 1 – función seno

```
---
%

G20
G90
F6

#1=0
#2=[#1*2*3.1416/360]
#3=[SIN[#1]]

G1
X[#2] Y[#3] Z[0]

O1 WHILE [#1 LE 360]
#1=[#1+1]
#2=[#1*2*3.1416/360]
#3=[SIN[#1]]
X[#2] Y[#3]
O1 ENDWHILE

%
---
```

- El programa comienza y termina con los símbolos de porcentaje (%).
- Se trabaja con *unidades* en *pulgadas* (*G20*) y en *modo de distancia absoluto* (*G90*). La velocidad de avance será de 6 [pulg/min].
- Se definen 3 variables (o *parámetros*), que servirán para guardar los valores de la función. #1 es una variable para el conteo de grados, #2 convertirá el valor de grados (almacenado en #1) a radianes y #3 evalúa el seno del valor en grados de #1.

- Se define un *avance lineal* con el comando *G1* y se dirige al primer punto coordenado evaluado de la función (y una altura Z=0).
- Comienza un *ciclo* con el comando *while*, que se repetirá *mientras* #1 sea *menor o igual* a 360 (grados).
- Se aumenta la variable #1 en 1 (grado) y actualiza los valores de la función en #2 y #3, poniéndolos finalmente en los *ejes*.
- Todos los parámetros que sean función de aquel que cambia dentro del ciclo (en este caso #1) deben ser introducidos dentro de la subrutina para que puedan ser actualizados.

En la *Figura 3.5* se muestra el resultado de este programa

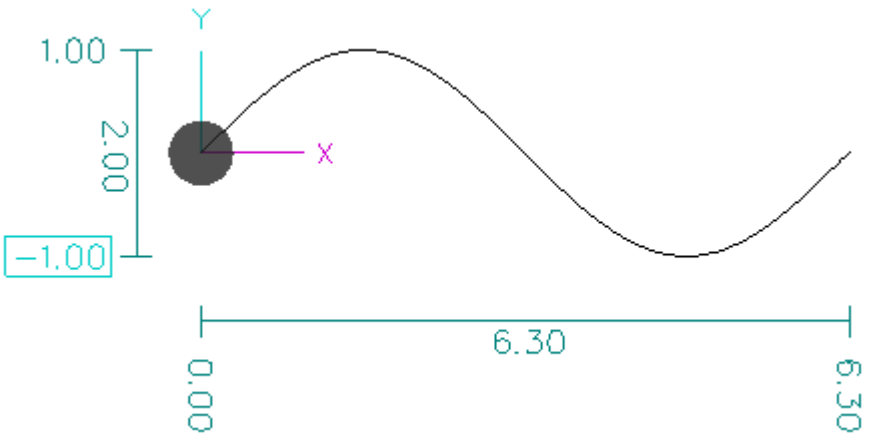


Figura 3.5: Visualización del programa de función seno

Ejemplo 2 – función polar

El siguiente programa, muy similar al anterior, introduce algunos aspectos no considerados en el primero tales como:

- posicionamiento en la primera coordenada con cambio en la altura de *z*. Esto evita que se *trabaje* sobre el material durante el posicionamiento.
- Se hace uso de una ecuación de transformación a coordenadas cartesianas, ya que se evalúa una ecuación polar.
- Se traslada el centro de la función a un punto de manera que no se sobrepasen los límites de la máquina y se evita lo sucedido en el ejemplo anterior, donde la función evalúa valores negativos en *y*. En ese caso el programa no puede ejecutarse. (Véase el valor mínimo en *y* de la *Figura 3.5* , el cual indica que se ha sobrepasado el límite de la máquina).

```
---
%

G20
G90
F6

#1=0; conteo de grados
#2=2; radio
#3=8; pétalos
#4=[#2*SIN[#3*#1]]; función
#5=[#4*COS[#1]]
#6=[#4*SIN[#1]]
#7=2; abscisa del centro de la flor
#8=2; ordenada del centro de la flor
#9=[#7+#5]
#10=[#8+#6]

G1
Z0.1
X[#9] Y[#10]
Z0

O1 WHILE [#1 LE 360]
```

```
X[#9] Y[#10]
#1=[#1+1]
#4=[#2*SIN[#3*#1]]
#5=[#4*COS[#1]]
#6=[#4*SIN[#1]]
#9=[#7+#5]
#10=[#8+#6]
X[#9] Y[#10]

O1 ENDWHILE

Z0.1

%
---
```

El resultado se muestra en la *Figura 3.6*.

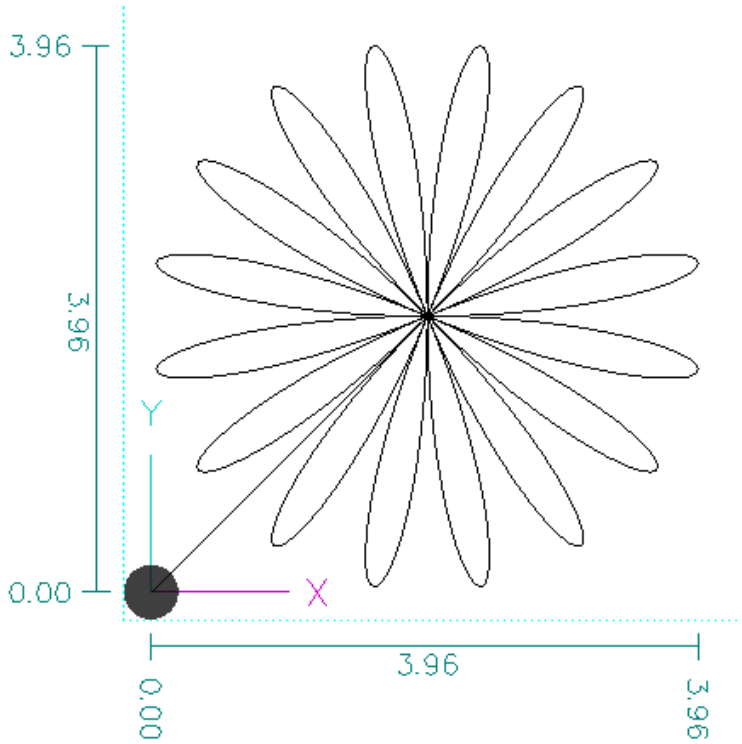


Figura 3.6: Visualización del programa de función polar

Ejemplo 3 – barrido rectangular

Este programa realiza una secuencia que puede ser utilizada para rebajar un material en una sección rectangular, ingresándose las coordenadas inferiores y superiores, así como el avance por paso. Al final del programa se hacen algunos comentarios.

```
---
%

g20 g90
f6

#1=0.5; punta inferior izquierda x
#2=0.5; punta inferior izquierda y
#3=3; punta superior derecha x
#4=2.55; punta superior derecha y
#5=0.05; avance en y
#6=0; variable binaria
#7=[#2+#5]; contador de avance en y

g1
z0.1
x[#1] y[#2]
z0

o1 while [#7 le #4]

o2 if [#6 eq 0]
x[#3]
#6=1
o2 else
x[#1]
#6=0
o2 endif

o3 if [#7 eq #4]
o1 break
```

```
o3 else
o3 endif

y[#7]
#7=[#7+#5]

o1 endwhile

%
---
```

- Se ha utilizado la secuencia if/else/endif, la cual ha servido para mover la coordenada x alternativamente de la coordenada superior a la inferior basándose en el valor de un parámetro (#6). Se asignó el valor de cero y *se pregunta* si tiene este valor para la primera evaluación, iniciándose el comportamiento deseado. Nótese que en cada evaluación se intercambia el valor del parámetro de cero a uno y viceversa.
- La subrutina o3 se asegura que al haber llegado al valor máximo deseado (en *y*) la subrutina principal (o1) se termine o rompa mediante el comando *break*; ya que de no tenerla, el ciclo terminaría con un avance vertical final (i.e. *rompiendo* el rectángulo).

El resultado se muestra en la *Figura 3.7*.

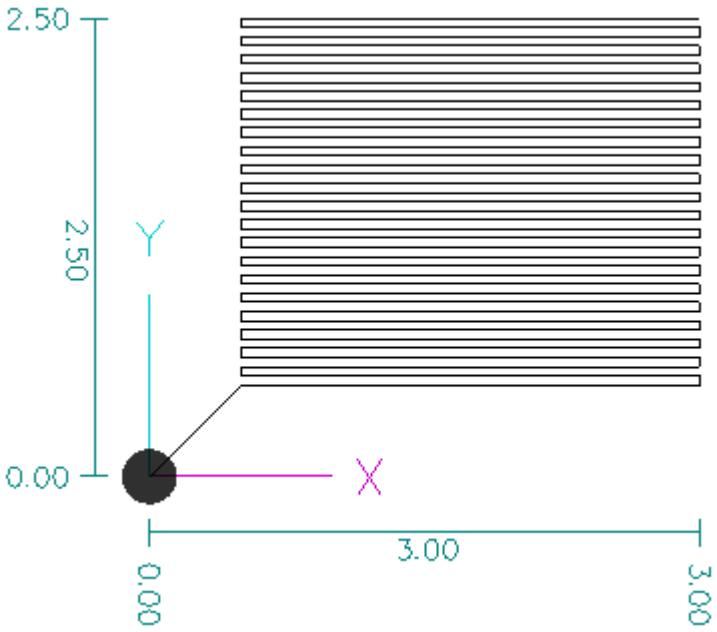


Figura 3.7: Visualización del programa de barrido rectangular

Ejemplo 4 – barrido circular

Este programa realiza una secuencia similar a la anterior, pero en forma circular. Las variables de entrada son la posición, el diámetro menor y mayor, así como el avance por vuelta. Los parámetros que no son para editarse tienen colocado “ne” en el comentario.

```
---
%

g20 g90
f6

#1=0.05; radio inicial
#2=0.05; incremento de radio por vuelta
#3=1.5; radio final
```

```
#4=1 (contador de grados; ne)
#5=1; incremento en contador de grados
#6=[#2/360] (incremento radial por paso; ne)
#7=1.5; centro círculo x
#8=1.5; centro círculo y
#9=[#7+[#1*cos[#4]]] (coordenada x; ne)
#10=[#8+[#1*sin[#4]]] (coordenada y; ne)

g1
z0.1
x[#9] y[#10]
z0

o1 while [#1 le #3]

#4=1

o2 while [#4 le 360]

#4=[#4+#5]
#1=[#1+#6]
#9=[#7+[#1*cos[#4]]]
#10=[#8+[#1*sin[#4]]]
x[#9] y[#10]

o2 endwhile

o1 endwhile

%
```

El resultado se muestra en la *Figura 3.8*

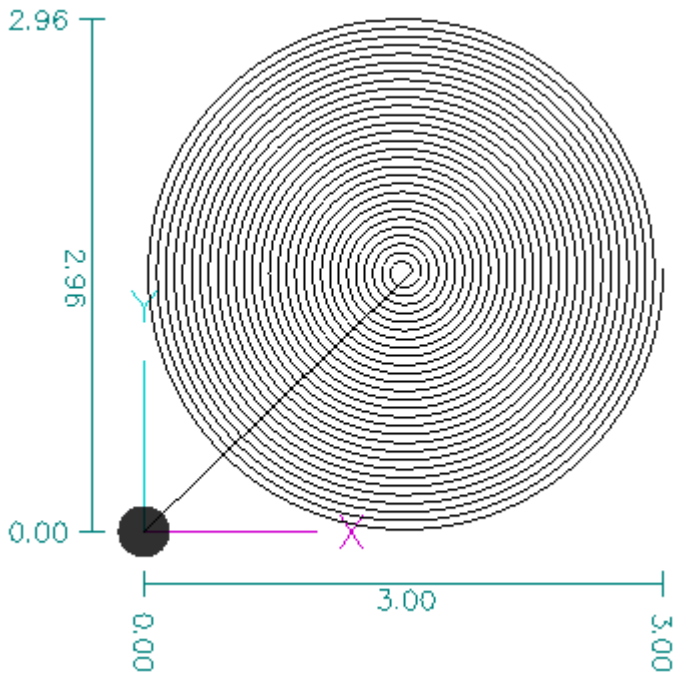


Figura 3.8: Visualización del programa de barrido circular

Habiendo visto esta parte correspondiente a la creación de código en forma de texto, se pasará ahora al diseño gráfico y su transformación a código numérico.

Diseño de dibujos en paqueterías CAD

El término CAD (por sus siglas en inglés) se refiere al Diseño Asistido por Computadora. Para nuestro propósito, se trata de crear *dibujos* que definirán las áreas y trayectorias de la herramienta en las operaciones de maquinado que se deseen y son un paso casi imprescindible en las tareas que se irán explicando a lo largo de este capítulo. Este tipo de archivos tienen normalmente las extensiones DXF o DWG. La primera es la que más interesa para el propósito que aquí nos compete debido a que es la reconocida por la mayoría de las paqueterías que se mostrarán para generar programas de código numérico.

A continuación se definen algunos conceptos comunes a este tipo de paqueterías

- Los objetos básicos que generalmente pueden crearse son líneas, arcos de circunferencia y círculos que se colocan en puntos coordenados, además de las *polilíneas* , que consisten de dos o más líneas y/o arcos de circunferencia que forman un solo objeto, en donde cada punto de unión de los mismos es conocido como *nodo*. Algunas paqueterías permiten agregar objetos más complejos como elipses, polígonos, estrellas, etc.
- Las *splines* o *curvas de bezier* son curvas con bordes suaves que tienen dos puntos de control por cada nodo que permiten controlar la curvatura en ese punto.
- La rejilla (*grid*) es un mallado con un espaciamiento constante definido por el usuario que es de ayuda para colocar los objetos deseados en el espacio.
- La opción *Ortho* permite colocar objetos como líneas solamente de manera vertical u horizontal.
- El *Snap* es un modo en el cual el cursor del ratón se aproxima automáticamente a otros objetos que pueden ser los nodos de la rejilla, puntos finales o intermedios de líneas, centros de círculos, etc.

La paquetería CAD casi por antonomasia es AutoCAD, cuya versión 2011 cuesta cerca de $900 dólares. Como se ha dicho, en este trabajo se presentan aplicaciones gratuitas y para este caso se recomiendan las siguientes:

- A9CAD[6] – es una aplicación CAD para diseño en dos dimensiones. Es la más sencilla de las aquí listadas de acuerdo al número de funciones y herramientas.

6 Disponible en: *http://www.a9tech.com*

- HYCAD[7] – es de las citadas quizá la más recomendable debido a la gran cantidad de herramientas que posee, tales como:
  - Vectorizar imágenes (File-Vectorize) para obtener un archivo dxf. Hay un punto dedicado posteriormente a la vectorización.
  - Dibujar funciones tales como parábolas, elipses, senoides, espirales, cardioides, etc. mediante la función *User Curve*.
  - Las función *Join* es de gran ayuda para optimizar dibujos. Su función es crear una polilínea con la trayectoria más larga posible basada en líneas que si bien pueden estar unidas visiblemente, no son parte del mismo objeto.
- CADuntu[8] - antes llamado LibreCAD posee una gran cantidad de herramientas para ayuda durante el dibujo de figuras y contiene algo que los anteriores no tienen, que son las fuentes de un solo paso (explicadas posteriormente en "Texto y tipos de fuente").
- FreeCAD es una aplicación que permite trabajar tanto en dos como en tres dimensiones. Sus funciones de *Extrusión* y *Revolución* permite generar rápidamente objetos en 3D.

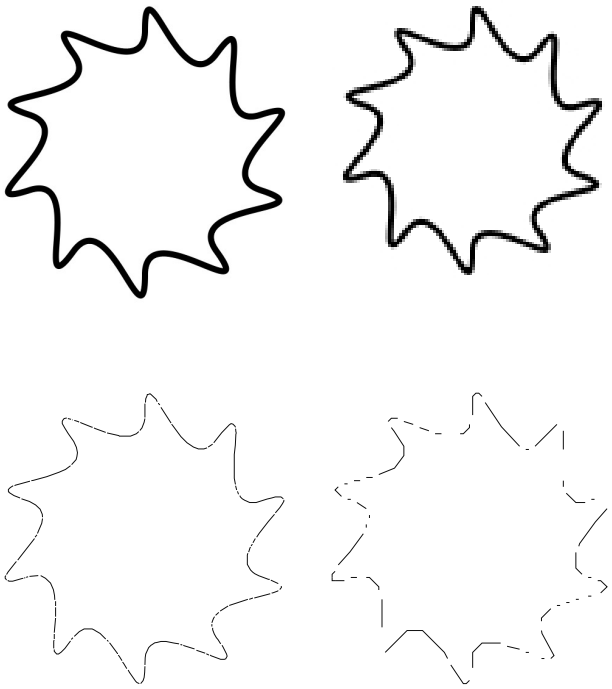
Manipulación de imágenes

A continuación se expondrán algunos métodos para el tratamiento de imágenes que sirven para su posterior vectorización o para usarse como mapas de altura. Para lo anterior se ha utilizado el programa GIMP[9], que es una aplicación para manejo y edición de imágenes.

Tamaño y resolución

7 Disponible en: *http://www.drawease.com*
8 Disponible en: *http://caduntu.org*
9 Disponible en: *http://www.gimp.org*

Para la creación de un nuevo archivo, también llamado plantilla, se ingresan dos variables: tamaño y resolución. El tamaño puede ingresarse en milímetros o pulgadas y la resolución en pixeles por unidad (i.e.- pixeles/mm o pixeles/pulgada), pudiendo verse como la cantidad de *puntitos* por unidad. En la *Figura 3.9* se muestran dos imágenes del mismo tamaño pero con distinta resolución, la de la izquierda siendo de 10 y la derecha de 2 pix/mm. Si bien más adelante se abundará sobre vectorización, por el momento basta con mostrar el resultado de ésta en las imágenes inferiores y ver cómo el resultado es deficiente en el de la imagen de menor resolución, donde se han formado líneas discontinuas que alteran la forma original y requieren de trabajo adicional para su unión. En la imagen vectorizada basada en la imagen de mayor resolución aún se encuentran algunas discontinuidades pero son de menor tamaño y cantidad. Para una resolución de 20 pix/mm la imagen vectorizada es continua, sin embargo, aquí comienza a involucrarse el factor de velocidad de procesamiento y recursos del sistema, que para imágenes de mayor complejidad es de particular relevancia.



50x50 mm;10 pix/mm 50x50 mm;2 pix/mm

Figura 3.9: Imágenes con distinta resolución (arriba) y sus correspondientes resultados de vectorización (abajo)

Simplificando contenido

Pueden importarse imágenes de las extensiones más comunes como jpg, png, bmp y también archivos pdf. De hecho se reconocen archivos de más de treinta extensiones.

En la *Figura 3.10* se muestra un ejemplo sobre el tratamiento de una imagen para su posterior vectorización. En *I* se muestra la figura original. En *II* se ha aplicado un filtro para detección de bordes (Filtros-Detectar bordes-Diferencia de gaussianas) y finalmente en *III* se ha utilizado la opción de Umbral (Colores-Umbral), que convierte la imagen a dos colores (blanco y negro) a partir de un valor seleccionable mínimo desde el cual se *toman* los colores de *II*. Ésta última imagen, tal como se verá más adelante podrá ser transformada a código G.

I *II* *III*

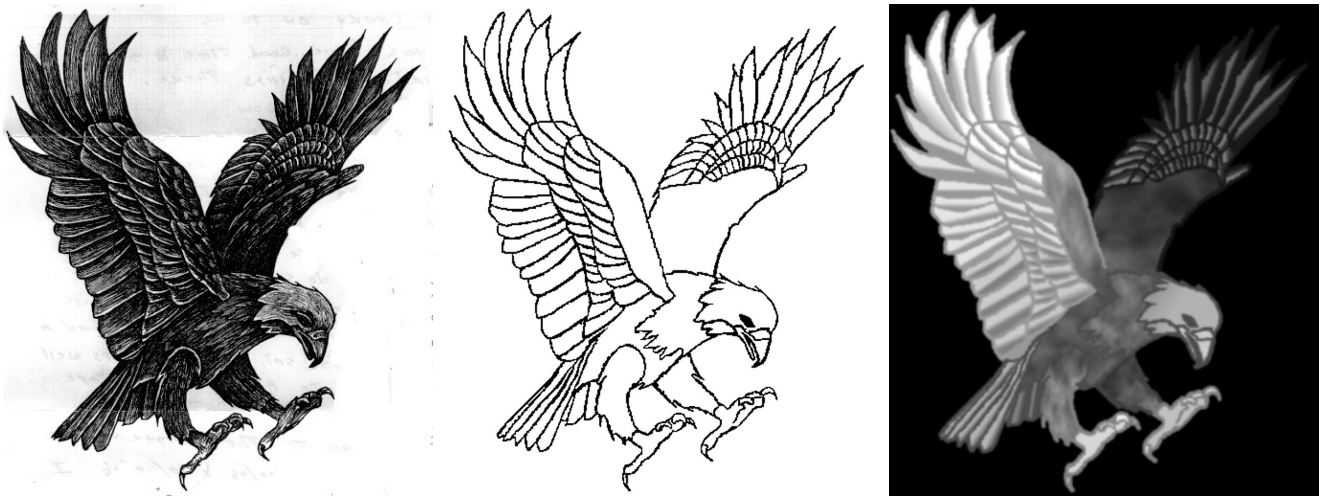
Figura 3.10: Tratamiento de una imagen

Dibujado de mapas de altura.

Son imágenes que utilizan el espectro de grises formados desde el blanco hasta el negro y, como se verá más adelante, permiten generar relieves. Los colores claros corresponden a una mayor altura y los obscuros a una menor.

En la *Figura 3.11* se muestra una imagen (*I*) a la cual se le aplicó el proceso de la *Figura 3.10,* además de una limpieza extra para dejarúnicamente los bordes de la misma tal como se muestra en *II* y un último proceso para pintar áreas de la misma y obtener el resultado de *III*. Esto involucra básicamente:

- seleccionar áreas cerradas, tales como las *plumas* de este ejemplo y utilizar la herramienta de *Selección difusa* (Herramientas-Herramientas de selección-Selección difusa), conocida como varita mágica en otras aplicaciones.
- las áreas seleccionadas son coloreadas con la *Herramienta de relleno* (Herramientas-Herramientas de pintura-Relleno) teniendo en mente el tipo de perfil final deseado, esto es, aquellas zonas de mayor y menor altura serán pintadas con colores más claros u obsuros, respectivamente.



I *II* *III*

Figura 3.11: Tratamiento para mapa de altura

Escritura de programas en Hojas de Cálculo

Una manera de obtener las coordenadas para un programa en código G o para una opción conocida como *nube de puntos* en algunas paqueterías de CAD, consiste en elaborarlas en una hoja de cálculo como *Calc* de OpenOffice o *Excel* de MSOffice. Para este trabajo se ha utilizado Calc. Esto requiere obtener una columna de valores como resultado de una concatenación de elementos de distintas celdas. En Calc, la función que realiza esta operación es *CONCATENAR*. Posteriormente se copia y pega esta columna en un editor de textos como el *Bloc de notas.*

En la *Figura 3.12* se muestra el resultado de este proceso para la función seno en las últimas tres columnas: *Código G* y *Nube de puntos txt* y *asc*. Las columnas con valores *xyz* y *XYZ* difieren en cuanto que las segundas tienen aplicada la función *REDONDEAR* a 3 decimales, ya que de no hacerlo se concatenan números con 16 cifras decimales.

x [º]	*x[rad]*	*y=seno(x)*	*z*	*X*	*Y*	*Z*	*","*	*" "*	*Código G*	*Nube de puntos (txt)*	*Nube de puntos (asc)*
0	0.00	0.00	0	0.000	0.000	0.000	,		X0 Y0 Z0	0,0,0	0 0 0
1	0.02	0.02	0	0.017	0.017	0.000	,		X0.017 Y0.017 Z0	0.017,0.017,0	0.017 0.017 0
2	0.03	0.03	0	0.035	0.035	0.000	,		X0.035 Y0.035 Z0	0.035,0.035,0	0.035 0.035 0
3	0.05	0.05	0	0.052	0.052	0.000	,		X0.052 Y0.052 Z0	0.052,0.052,0	0.052 0.052 0

Figura 3.12: Muestra de la hoja de cálculo con valores concatenados en distintos formatos

El formato en cada una de las tres últimas columnas es el siguiente:

- Código G – contiene las coordenadas de cada punto directamente para ser insertadas en un programa en código G. Este archivo requeriría de los comandos necesarios al inicio y final del programa tales como modo de distancia, tipo de unidades, fin de programa, etc. La forma es la siguiente (siendo opcional el espacio entre palabras):

  X[valor X] Y[valor Y] Z[valor Z]

- Nube de puntos (*txt*) – Este tipo de archivo puede ser usado en CamBam (aplicación vista más adelante) con uno de sus *plugins* (Plugins-PointsCloud). Este archivo se guarda con la extensión txt y tiene el siguiente fomato (sin espacios):

  [valor X],[valor Y],[valor Z]

- Nube de puntos (*asc*) – Este formato puede ser usado en FreeCAD en una de sus opciones llamada Points (View-Workbench-Points), donde se importa un archivo con la extensión ASC. Aquí se utiliza el siguiente patrón (con espacio entre valores):

  [valor X] [valor Y] [valor Z]

En la *Figura 3.13* se muestra el resultado de esta importación en el programa CamBam y del cual se hablará detalladamente más adelante.

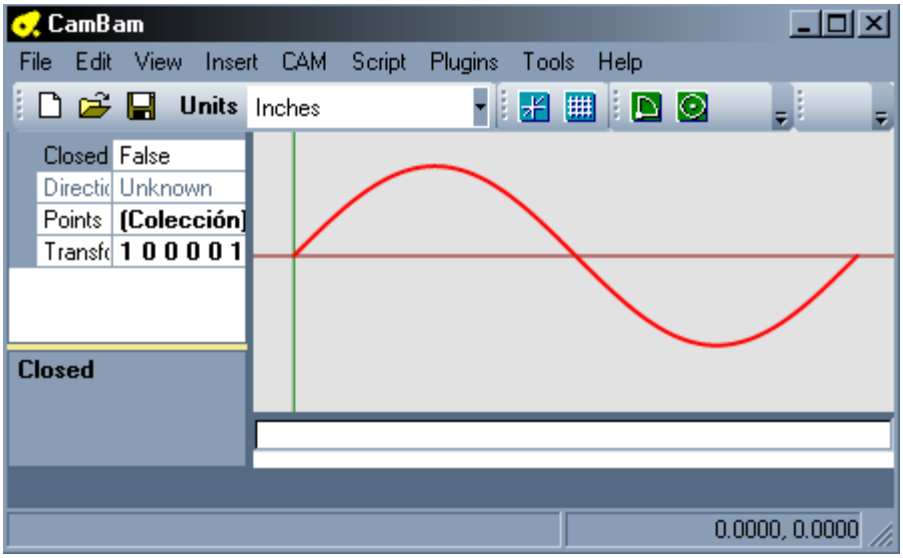


Figura 3.13: Importación de nube de puntos en CamBam

Texto y tipos de fuente

Casi cualquier paquetería de CAD permite insertar un texto con opciones tales como tipo de letra y tamaño. Estos son tomados de un *banco de datos* contenido (para el caso de Windows) en la carpeta Fonts, dentro de la carpeta Windows y son archivos del tipo TTF (e.g. - Arial.ttf, Times New Roman.ttf, etc.). Este método produce polilíneas que siguen el contorno del tipo de letra y se genera algo similar a lo mostrado en la *Figura 3.14*.

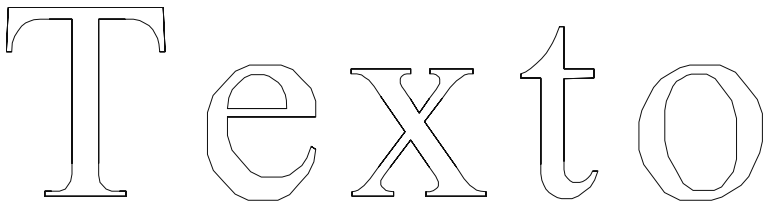


Figura 3.14: Letras de dos pasos

Para ciertas aplicaciones de maquinado es preferible utilizar letras de un solo paso. Para ello puede ocuparse tanto un programa específico para ello, tal como *StickFont*[10] o una aplicación CAD que viene con tipos de fuente de este tipo como CADuntu. En ambas puede exportarse el texto generado como un archivo DXF y para el caso de Stickfont, un programa en código G directamente. En la *Figura 3.15* se muestra este tipo de letras.

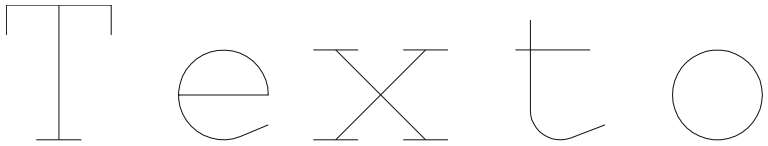
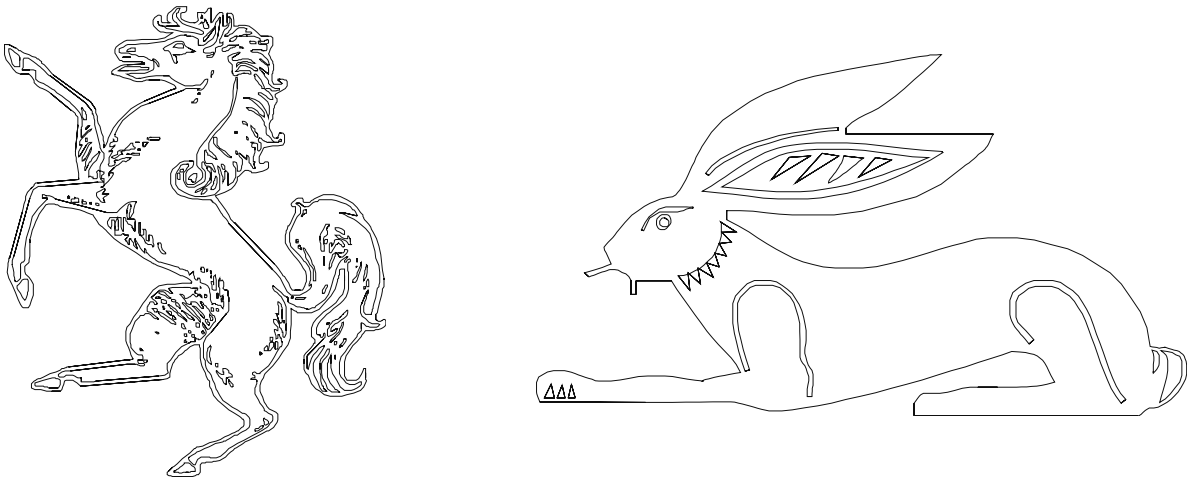


Figura 3.15: Letras de un paso

Relacionado con lo anterior están los tipos de letra llamados *Dingbats,* que más que una letra son una figura y de los cuales cada letra del abecedario corresponde a una figura distinta. Son archivos igualmente de extensión TTF de los cuales hay decenas que pueden descargarse de páginas como *urbanfonts.com* o *dafont.com*, algunos gratuitos y otros con cierto costo. En la *Figura 3.16 I* se muestra la letra Y del dingbat *Intellecta Heraldics* y en *II* la letra K del dingbat *Old Egypt Glyphs*. Estas figuras se pueden utilizar directamente en un programa CAD como HYCAD y ser guardadas como un archivo DXF para ser fácilmente convertibles a código G.

10 Disponible en: *http://www.ncplot.com/stickfont/stickfont.htm*



I *II*

Figura 3.16: Ejemplos de Dingbats

Dibujo de engranes

Una gran cantidad de ejemplos mostrados en internet consisten en utilizar este tipo de máquinas para cortar engranes, generalmente en madera. En este sentido hay proyectos en los cuales se hacen relojes que utilizan estos engranes.

La aplicación *GearDXF*[11] permite dibujar engranes ingresando parámetros como paso diametral, número de dientes y ángulo de presión. Al tener el engrane deseado se exporta como un archivo DXF. La interfaz de usuario de este programa se muestra en la *Figura 3.17*.

11 Disponible en: *http://www.forestmoon.com/Software/GearDXF*

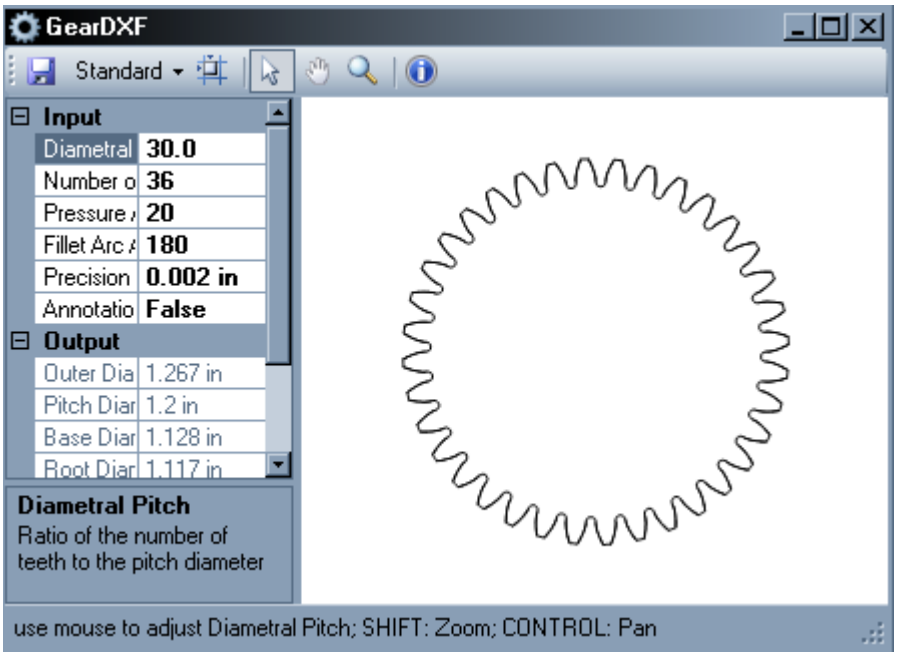


Figura 3.17: Interfaz del programa GearDXF

Mapas de altura

Una imagen en escala de grises puede ser utilizada como un mapa de altura. Los colores claros tendientes al blanco representan zonas de mayor altura y las oscuras cercanas al negro serán profundas. Algunas aplicaciones permiten generar programas en código G que maquinan este relieve. Esto se realiza en pasos horizontales, verticales o con una combinación de ambos. A continuación se muestran tres maneras de realizar esto:

1. Utilizando la interfaz de usuario *AXIS* del *EMC2* puede abrirse un archivo de imagen que es transformado en su correspondiente relieve en código G.

   Una desventaja que se ha visto en este procedimiento es el tiempo de procesamiento. En una ocasión se dispuso la transformación a código de una imagen que más de doce horas después seguía en proceso.

2. Utilizando el *plugin* de *CamBam* llamado *HeightMap Generator* (Plugins-HeightMap Generator) se han obtenido los mejores resultados. La conversión es casi inmediata y la calidad del relieve bastante fiel.
3. En GMAX puede utilizarse el modificador *Displace* sobre un plano que contiene la imagen deseada. Posteriormente se aplica un modificador del tipo *Meshsmoothing* a este plano, que es el que dará el detalle al relieve. Finalmente, utilizando CNCToolkit se proyecta una polilínea sobre la superficie donde está el relieve, que puede ser una secuencia de barrido rectangular o circular que puede ser creada por el mismo CNCToolkit. Esta línea proyectada es la que será transformada a código G por el CNCToolkit.

   Esta opción es la que requiere de más pasos, además de bastantes recursos del sistema. Se ha trabajado con un equipo a 1.6 Ghz y 1 GB en memoria RAM y en los últimos pasos el sistema regularmente se pasma.

Este tipo de operaciones requiere generalmente de más de una operación de maquinado; en las que se utiliza inicialmente un cortador de plano para remover la mayor cantidad de material con avances de 30 a 40 % y pasos posteriores de detalle con un cortador esférico.

En la *Figura 3.18* se muestra la visualización del relieve producido en CamBam del mapa de altura que se mostró en la *Figura 3.11* (p. 67).

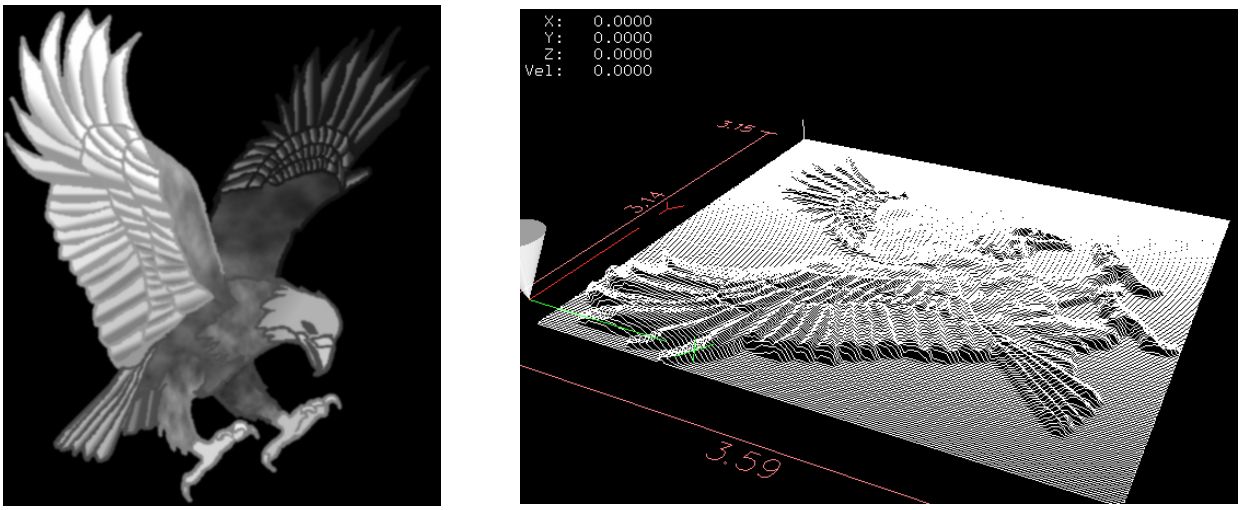


Figura 3.18: Muestra del relieve generado con base al mapa de altura correspondiente

Vectorización

Consiste en la conversión de una imagen que puede estar en algunos de los tipos de archivo de imágenes más comunes tales como JPG, PNG o BMP, a un archivo del tipo de DXF. Este formato es el más común y reconocido por distintas paqueterías para ser transformado directamente en código G.

Algunos programas gratuitos que permiten realizar esta operación son: *HYCAD*, *Inkscape*[12] y *WinTopo*[13].

Puede hacerse una detección de bordes o de línea central, tal como se muestra en la *Figura 3.19*. En *I* se muestra una imagen, en *II* su correspondiente vectorización basada en detección de bordes y en *III* con el tipo de línea central promedio.

12 Disponible en: *http://inkscape.org*
13 Disponible en: *http://wintopo.com*

Figura 3.19: Modos de detección de frontera y línea media en vectorización

En la *Figura 3.20 I* se muestra una imagen que fue manipulada con los métodos enunciados anteriormente para tener el resultado de *II* y su vectorización en un archivo DXF con el uso de la paquetería WinTopo, mostrada en *III*.

I *II* *III*

Figura 3.20: Tratamiento de una imagen y su posterior vectorización

En general, al vectorizar una imagen se obtiene un archivo en el cual las polilíneas están tanto separadas como desunidas. Con HYCAD pueden juntarse las separadas ayudándose con el modo OSNAP, que *jala* el cursor al punto final de la línea a la cual se desea juntar y posteriormente usar la opción Join (MoreModify-Join) que las une y las hace una polilínea. Con ello se obtiene un archivo optimizado, con un menor número de polilíneas, lo que a su vez redunda en un menor tiempo de maquinado. En el ejemplo de la *Figura 3.20*, al aplicar esta opción de Join se redujo el número de polilíneas de cerca de 3000 a 700.

Sin embargo, no en todos los casos es posible vectorizar una imagen utilizando alguno de los programas mencionados. Una amiga necesitaba vectorizar algunos mapas que contenían líneas de carreteras, ríos, divisiones geográficas, etc. y la vectorización automática resultó deficiente y muy defectuosa. En este caso, se tuvo que cargar la imagen escaneada y dibujar las líneas *manualmente* en un programa CAD. Un ejemplo de lo anterior, aunque de menor

complejidad, se ha realizado con el mapa mostrado en la *Figura 3.21 I*, sobre el cual se han dibujado las fronteras y contornos con curvas de bezier. Para ello se ha utilizado HYCAD y su función *Insert Image* (Draw-Insert Image), que carga una imagen en la ventana de trabajo. El mapa vectorizado se muestra en *II.*

I

II

Figura 3.21: Mapa vectorizado manualmente

Transformación de archivos DXF a código G

Al tener un archivo DXF puede procederse a su transformación a código G con algunas de las siguientes aplicaciones

- DXF2GCODE[14] es una aplicación dedicada exclusivamente a realizar esta operación de transformación. Sencilla y recomendable.
- CamBam – Al seleccionar las polilíneas que desean transformarse se utiliza la opción de insertar la operación de maquinado Engrave (CAM-Engrave). Después de ingresar algunos parámetros de maquinado como radio de cortador, velocidad de avance, etc. se utiliza la opción de crear archivo en código G (CAM-Create GCode File).
- GMAX/CNC Toolkit – el uso de estas paqueterías permite realizar esta operación de una manera igualmente sencilla. Más adelante se dedicará una parte exclusiva para estas aplicaciones.

Transformación de dibujos a código G

Otra alternativa para generar programas en código G consiste en utilizar dibujos que pueden hacerse, por ejemplo, en Office. Se guardan como archivo SVG y éste es tranformado al tipo DXF, lo cual puede realizarse con la aplicación *Inkscape*, que por sí sola es una poderosa paquetería de diseño gráfico.

En la *Figura 3.22* se muestra la visualzación del código G que se ha obtenido con base en la *Figura 3.4*, la cual fue hecha en Draw de OpenOffice y fue transformada usando el procedimiento descrito anteriormente.

14 Disponible en: *http://www.christian-kohloeffel.homepage.t-online.de/dxf2gocde.html*

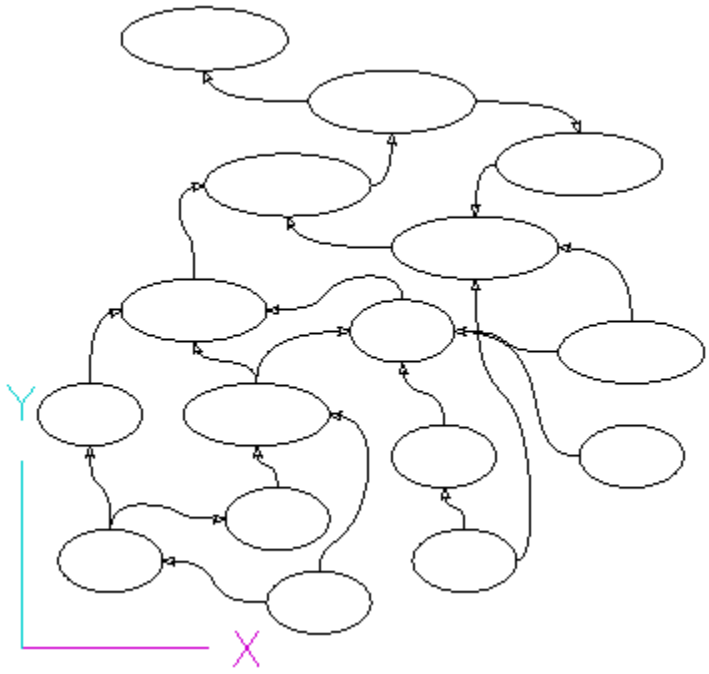


Figura 3.22: Visualización del código generado con base a la Figura 3.4

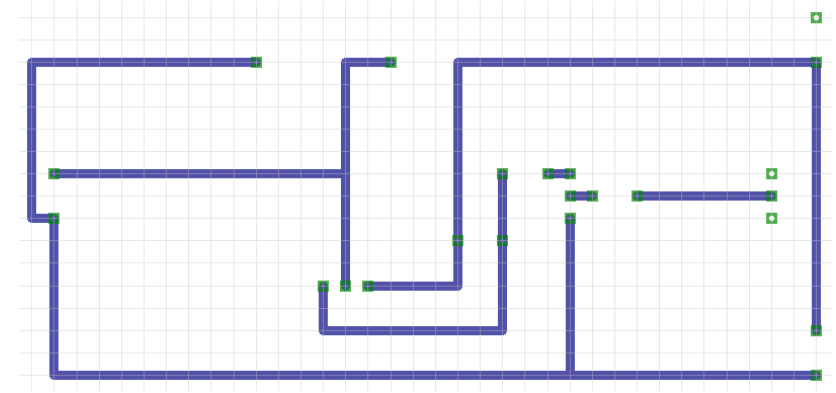
Grabado de circuitos impresos

Esta aplicación consiste en fresar el contorno de las pistas para una tarjeta de circuito, logrando el aislamiento requerido, así como el barrenado para las piezas del circuito.

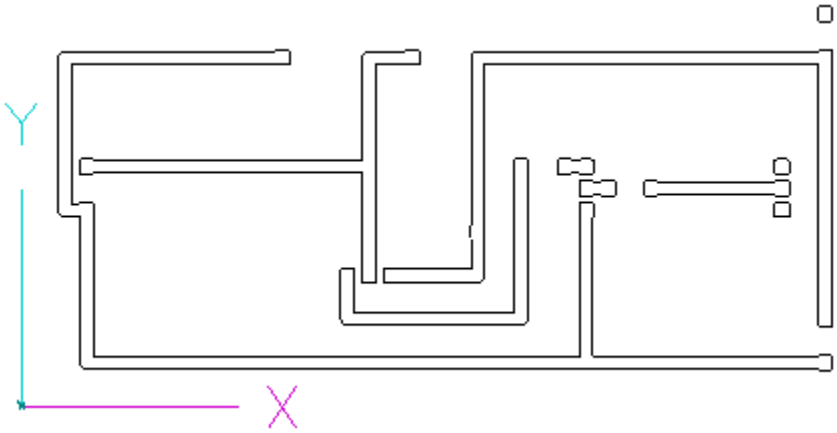
Hay varias opciones para lo anterior y una buena guía que cubre bastantes de ellas es la del sitio RepRap[15]. Una opción relativamente común y popular es la de utilizar PCB-GCode, que es un *script* para la paquetería EAGLE. PCB-GCode es gratuito y puede ser descargado después de registrarse en el grupo de yahoo del mismo[16]; por otro lado EAGLE es gratuito en su versión más sencilla siempre y cuando no sea con fines lucrativos. En caso contrario debe registrarse el producto, cuya licencia más barata cuesta $49 dólares.

15 Disponible en: *http://reprap.org/wiki/PCB_Milling#pcb2gcode*
16 Disponible en: *http://groups.yahoo.com/group/pcb-gcode/*

Sólo con fines demostrativos se ha utilizado EAGLE para diseñar el circuito mostrado en la *Figura 3.23 I* y posteriormente PCB-GCode para generar el código G tanto para el aislamiento de las pistas del circuito como para los barrenos requeridos, tal como se muestra en *II*.



I



II

Figura 3.23: Circuito diseñado y visualización de su código G correspondiente

CamBam[17]

Es una aplicación que puede utilizarse para hacer dibujos CAD, aunque sin todas las características que ofrecen las paqueterías especializadas como las mencionadas anteriormente, pero su principal ventaja es la de crear el código numérico de varias operaciones de maquinado. Tiene una versión gratuita y otra profesional con un costo de $149 dólares. En este trabajo se ha utilizado la versión gratuita.

A continuación se presentan algunas operaciones que pueden realizarse con esta paquetería:

- Importación de nube de puntos – En la parte: "Escritura de programas en hojas de cálculo" (p. *67*) se ha explicado la generación de una columna de datos concatenados que contienen las coordenadas *x,y,z* de algo que puede ser una función, aunque no necesariamente, y que en conjunto forman una *nube de puntos*. Este archivo con extensión TXT puede ser leído en CamBam mediante uno de sus *plugins* (Plugins-Point Cloud Reader). Esto permite tener una polilínea que puede ser entonces exportada directamente como un archivo vectorizado DXF o utilizarse como trayectoria para una operación de maquinado.
- Generación de mapas de altura – En "Mapas de altura" (p. *72*) se explica el uso de CamBam para generar el código G de un relieve basado en una imagen.
- Operaciones de maquinado – Una de las principales y más útiles herramientas de CamBam es la que permite generar las trayectorias de herramienta para distintas operaciones de maquinado, que se insertan mediante la opción CAM del menú, y las cuales serán explicadas a continuación.

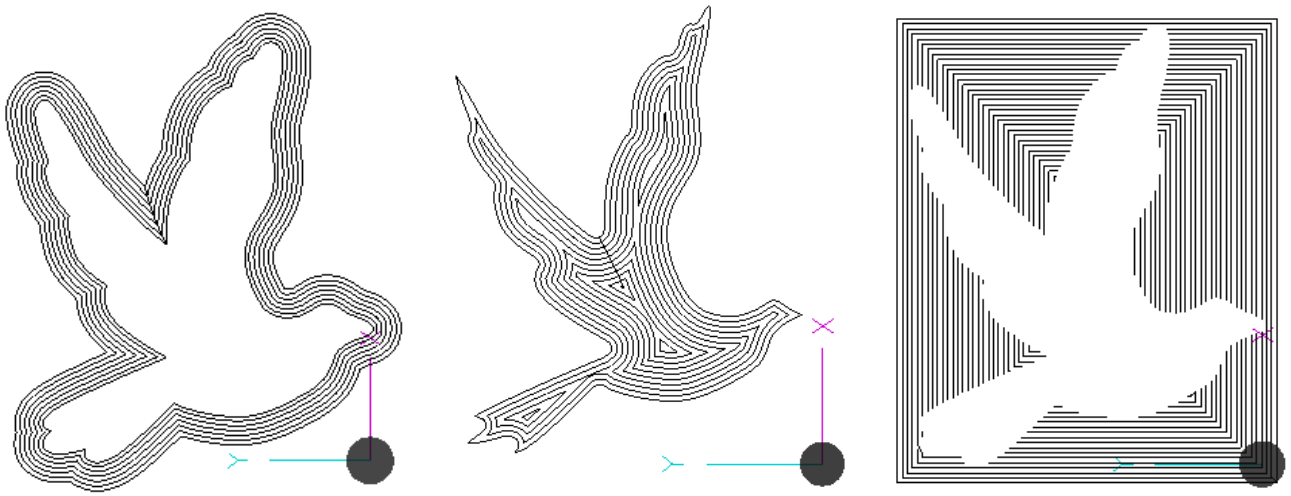
    - 2.5D Profile – genera una trayectoria que circula alrededor del exterior de una

17 Disponible en: *http://www.cambam.info*

polilínea cerrada, tal como se muestra en la *Figura 3.24 I.*

- Pocket – genera una trayectoria dentro de una polilínea cerrada que cubre toda el área de ésta. *Figura 3.24 II.*
- Drill – genera una secuencia de barrenado tomando como base un punto coordenado.
- Engrave – genera una trayectoria que sigue la de una polilínea cerrada o abierta.
- Bas Relief – prepara la superficie del material en un área rectangular sobre la cual se encuentra una polilínea con una secuencia igual a la de la *Figura 3.7.*
- Region – Si bien no es una operación de maquinado directa, se obtiene mediante la selección de las polilíneas cerradas que se desea formen una *región* y después seleccionando la opción de insertar Region. *Figura 3.24 III*.

En cada operación, con ciertas diferencias específicas de cada una, se ingresan parámetros tales como diámetro de la herramienta, profundidad final, avance por paso de maquinado, etc.



I *II* *III*

Figura 3.24: Trayectorias generadas para distintas operaciones en CamBam

Además de lo anterior, se ha visto que CamBam puede *estandarizar* archivos DXF, ya que algunos archivos exportados en este formato por ciertas paqueterías no son reconocidos por otras. En este sentido se ha visto que todos estos archivos han sido reconocidos con este programa y al exportarlos desde el mismo, son reconocidos por aplicaciones que antes no lo hacían.

GMAX / CNC Toolkit.

GMAX es una aplicación gratuita para el diseño de modelos en 3 dimensiones, principalmente encaminado al desarrollo de contenido para juegos de video. Es una versión reducida del programa profesional *3DS MAX*, desarrollado por la misma compañía. Se descarga[18] y posteriormente hay que hacer un registro para obtener (vía correo electrónico) el número serial que permite utilizarlo.

Por otro lado, *CNC Toolkit* es un *script* para programas tales como 3DS Max o GMAX y es prácticamente como una subaplicación de éstos. Permite generar código numérico para máquinas CNC de 3, 4 ó 5 ejes. Para obtenerlo necesita uno registrarse en el grupo de yahoo[19] del mismo y así tener acceso a la descarga de los archivos.

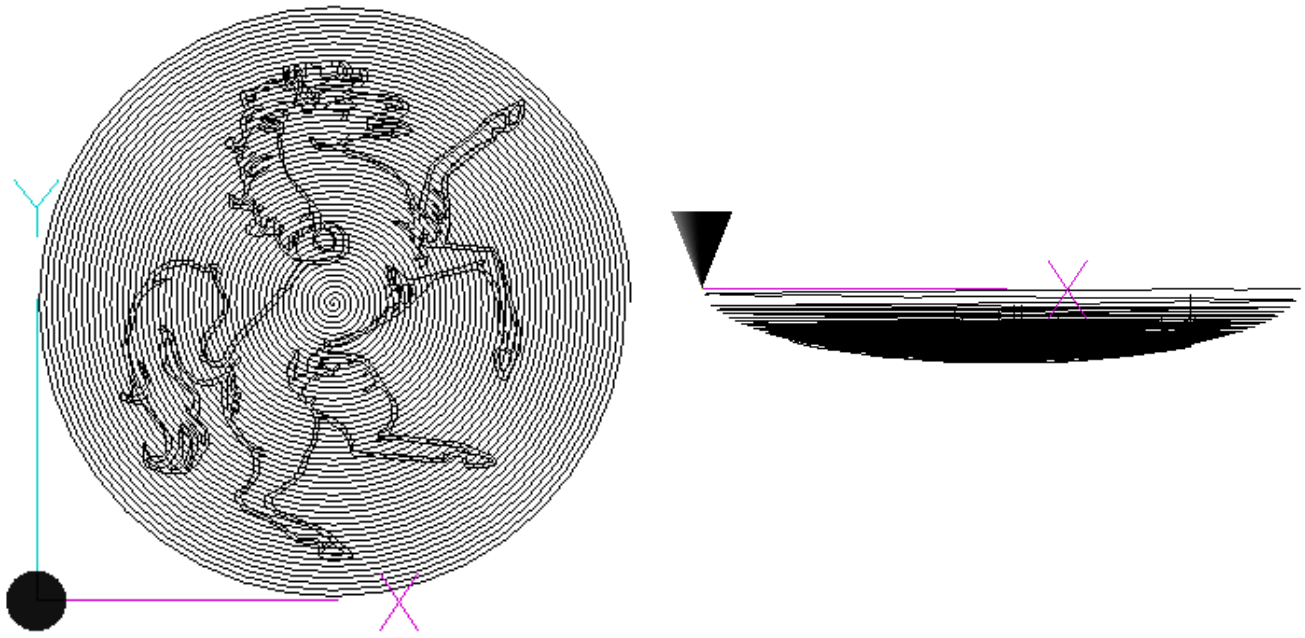
El archivo del programa con extensión MS es colocado en la carpeta llamada *scripts*, que se encuentra en la carpeta de instalación (en este caso del GMAX). Entonces puede llamarse mediante la opción *Run Script*, que está en la pestaña de *MAXScript*.

A continuación se presentan algunos ejemplos en los cuales se han utilizado estas aplicaciones.

18 Disponible en: *http://www.turbosquid.com/gmax*
19 Disponible en: *http://groups.yahoo.com/group/CNC_Toolkit*

Proyección de *Splines*

CNC Toolkit puede generar trayectorias siguiendo una polilínea / spline para producir el código G correspondiente, además de poder proyectar éstas sobre una superficie. En la *Figura 3.25* se muestra el código G generado con la proyección del *dingbat* mostrado en la *Figura 3.16* sobre una superficie en forma de plato, la cual se maquina con una espiral incremental similar a la mostrada en la *Figura 3.8*, que igualmente se proyecta sobre esta superficie. La espiral incremental es generada por CNC Toolkit y se ingresan parámetros como radio y paso por vuelta.



Vista Superior Vista Frontal

Figura 3.25: Proyección de splines sobre superficies

Cambiador de forma

Uno de los modificadores de GMAX llamado *Morpher* permite cambiar la forma de una figura en otra gradualmente siempre y cuando tengan el mismo número de nodos. Si las figuras no tienen la misma cantidad de nodos puede utilizarse la opción de *CNC Toolkit* para subdividir una polilínea y obtener más nodos en la misma, de manera que se iguale el número con los de otra figura. En la *Figura 3.26* se muestra la visualización del programa generado por CNCToolkit para un conjunto de curvas generadas con este modificador y que requirió de la subdivisión de nodos en el círculo, ya que era menor en número a los del polígono.

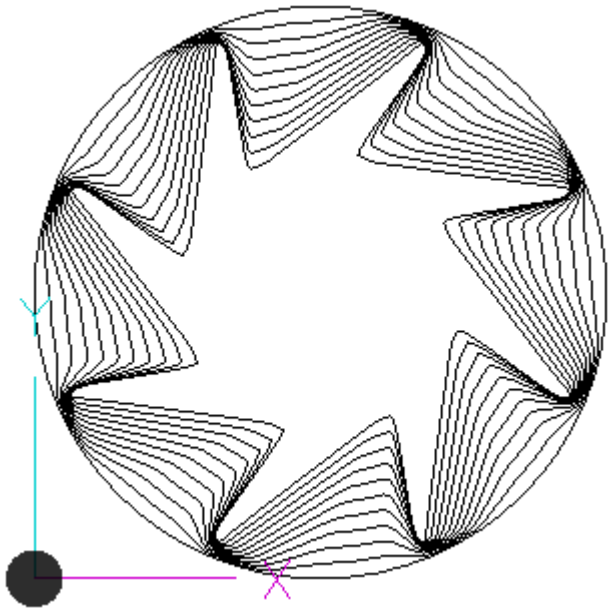


Figura 3.26: Visualización de código generado en CNC Toolkit basado en el modificador cambiador de forma de GMAX

El código generado por CNCToolkit se guarda en una ventana similar al Bloc de notas de Windows propia del programa llamada *MAXScript Listener Window* y la cual,

desafortunadamente, tiene la restricción de no permitir guardar más de 1000 líneas de código. Para ello, sin embargo, puede utilizarse un *Grabber*[20], que es un pequeño programa que guarda lo que se encuentra en esa ventana sin restricciones.

Manejo de objetos en tres dimensiones

Aplicaciones como GMAX y FreeCAD pueden utilizarse para generar este tipo de objetos, aunque GMAX tiene un abanico de posibilidades mucho mayor al de FreeCAD, debido a la cantidad de funciones y modificadores con que cuenta.

Cuando se ha creado un objeto y ha sido guardado con el tipo de extensión debida, puede ser utilizado en aplicaciones como FreeMill o CNCToolKit para que generen las trayectorias de herramienta correspondientes.

Originalmente GMAX puede exportar objetos en el formato *Plasma 3d* (P3D), pero en varias paqueterías utilizadas se ha visto que no reconocen este tipo de archivo. Sin embargo, puede utilizarse un *script* para exportar objetos en formato MD3[21], el cual es más fácil de ser reconocido. Estos archivos pueden ser a su vez transformados en otros más comunes tales como OBJ, STL ó 3DS con el uso de otras paqueterías tales como LithUnwrap y MeshLab[22], siguiendo el orden mostrado en la *Figura 3.27*. FreeCAD puede exportar objetos directamente en archivos STL y OBJ, entre otros.

20 Hay varios, pero en esta página se encuentran enlaces para dos de ellos: *http://www.turbosquid.com/Forum/Index.cfm/stgAct/PostList/intThreadID/26799*

21 Disponible en: *http://pages.videotron.com/browser*

22 LithUnwrap: *http://www.geeks3d.com/20090303/lithunwrap-free-uv-mapper-for-windows*
Meshlab: *http://meshlab.sourceforge.net*

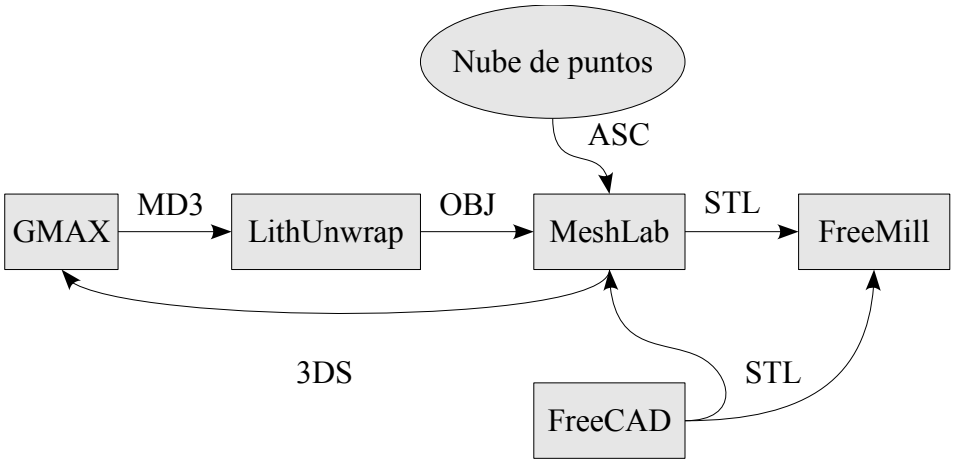
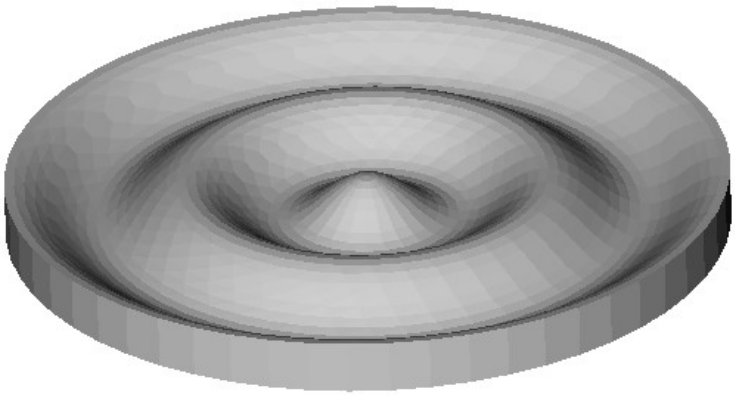


Figura 3.27: Procedimiento para manejo de objetos en 3D

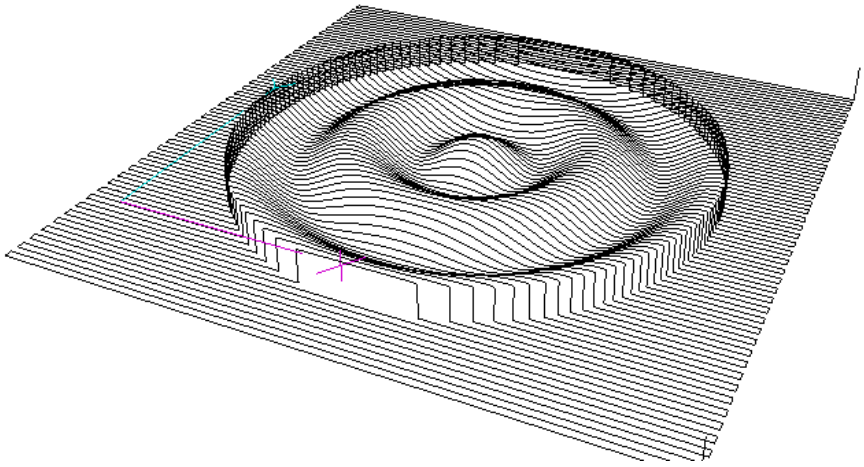
FreeMill[23] es una aplicación que permite generar la trayectoria de la herramienta de una fresadora de tres ejes para el maquinado de objetos en tres dimensiones. Si bien soporta varios tipos de archivo, el que especialmente nos interesa es el *STL*. La trayectoria consiste en un barrido rectangular como el de la *Figura 3.7* que se proyecta sobre la superficie deseada.

En la *Figura 3.28 I* se muestra un objeto generado en FreeCAD mediante la opción de sólido de revolución, el cual es exportado como un objeto STL. Este archivo es utilizado en FreeMill para generar la trayectoria mostrada en *II* y es asimismo abierto en MeshLab para cambiarle el formato y convertirlo al tipo 3DS, que es el soportado por GMAX. Con CNCToolKit pueden generarse trayectorias como las de las *Figuras 3.7* y *3.8*. Como se ha visto, estas polilíneas o splines se pueden proyectar en una superficie y tomar la forma de ésta. En *III* se muestra la proyección de una de estas espirales sobre la superficie tratada.

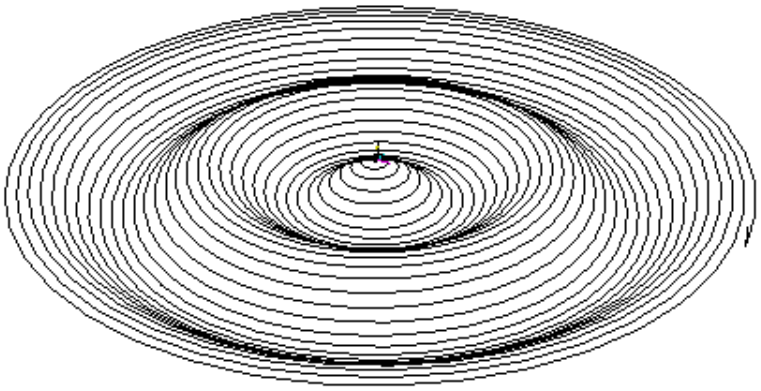
23 Disponible en: *http://www.mecsoft.com/freemill.shtm*



I



II



III

Figura 3.28: Superficie y visualización de códigos con distintos métodos de barrido

Ahora supóngase el siguiente ejemplo, en el cual se tiene una función para generar una superficie que es programada en un cierto rango y se obtiene la tríada de coordenadas (*xyz*) de esta función como una matriz columna. Estos valores pueden exportarse a un editor de textos y ser utilizados como una nube de puntos. La *Figura 3.29* muestra el programa realizado en *Euler Math Toolbox*[24] que realiza lo explicado anteriormente para la función de un paraboloide hiperbólico.

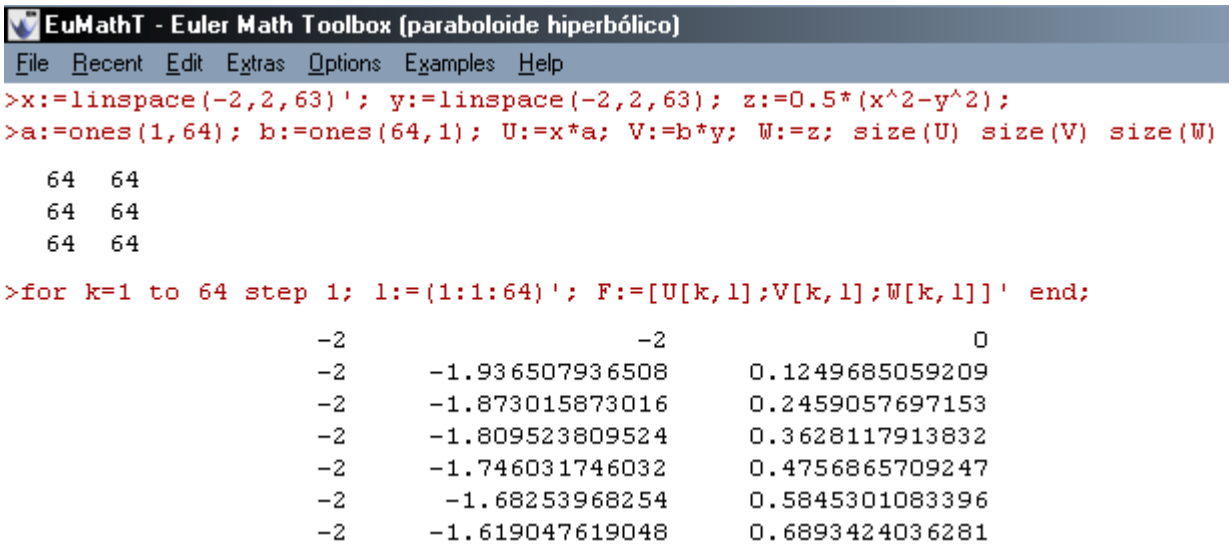
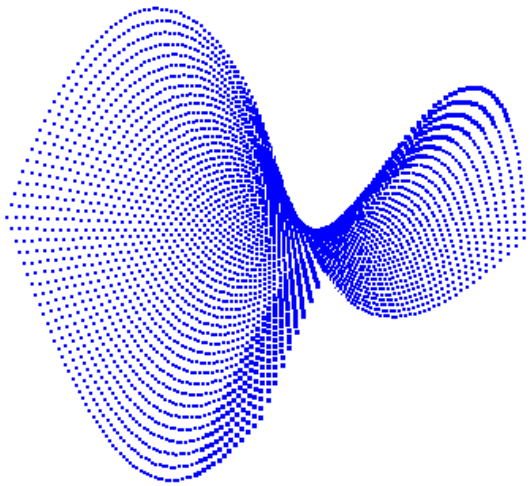


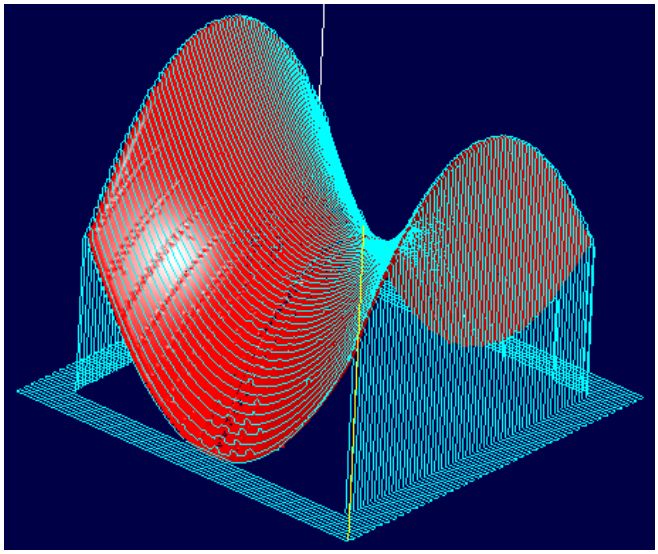
Figura 3.29: Programa en Euler para obtención de coordenadas de funciones de dos variables

Esta serie de valores copiada en el Bloc de notas y guardada con extensión ASC, puede ser importada en MeshLab, tal como se mostró en la *Figura 3.27*. En esta paquetería se utilizó el filtro para conjunto de puntos *Marching Cubes RIMLS* (Filters-Point Set-Marching Cubes), el cual genera una superficie continua y que para este ejemplo ha sido guardada como archivo STL, mismo que puede ser utilizado en FreeMill. En la *Figura 3.30* se muestra la nube de puntos de la superficie (*I*) y la trayectoria de herramienta generada en FreeMill (*II*).

24 Disponible en: *http://eumat.sourceforge.net*



I



II

Figura 3.30: Superficie como nube de puntos y su posterior uso para generación de código al haber sido transformada a superficie continua

La superficie generada mediante este filtro no resulta completamente lisa y su calidad depende en parte de la densidad de los puntos utilizados. La nube de puntos mostrada en *I* utiliza 64 valores tanto en *x* como en *y*, pero la superficie mostrada en *II* se formó con una nube de 128 valores en cada eje, ya que con la cantidad de 64 su calidad era menor.

Visualización de programas en código G

Es conveniente revisar siempre un programa antes de ejecutarlo y para ello se utiliza un visualizador de código G, que puede ser alguno de los siguientes:

- AXIS es la interfaz gráfica de usuario del EMC2 que permite visualizar un programa antes y durante su ejecución. Como cualquier programa en código G del EMC2

requiere que su extensión sea NGC. Cuando un programa contiene códigos O específicos del EMC2 (RS274NGC) sólo pueden ser vistos con esta aplicación y no en las que se mencionan a continuación, que sólo soportan códigos G

- NCSim[25] es una aplicación de Windows que permite visualizar y simular programas en código G de una máquina de 3 ejes. Algunos programas que han sido generados en CamBam al ser vistos con este programa muestran unas figuras de círculos que no corresponden al programa y que al ser visualizados con otras de las aplicaciones son presentados tal cual son, es decir, sin esos círculos. A pesar de ello, es una de las aplicaciones que se usó con más frecuencia debido a la rapidez y facilidad de uso.

- CNCSimulator[26] es una aplicación que puede simular tanto un torno como una fresadora y es un tanto más profesional que el NCSim.

- XinCNC[27] es una aplicación muy sencilla para visualizar programas.

25 Disponible en: *http://www.cs.technion.ac.il/~gershon/NCSim*
26 Disponible en: *http://www.cncsimulator.com*
27 Disponible en: *http://sourceforge.net/projects/xincnc*

CAPÍTULO 4 - RESULTADOS

Hasta el momento se ha utilizado la máquina para realizar las operaciones mostradas en el capítulo IV a efecto de conocer su rendimiento, tiempos, calidad, etc. Sin embargo, todavía no se le ha dado una aplicación para trabajos propios del taller, lo cual se hará en el corto plazo. Además, se está construyendo una máquina de mayor tamaño y capacidad para la cual ya se tienen sus componentes (i.e. tarjetas, computadora, motores, etc.).

A continuación se muestran resultados de algunas operaciones realizadas en la máquina.

Dibujos con pluma

Las operaciones de las *Figuras 4.1* a *4.5* fueron grabadas con una pluma como accesorio.

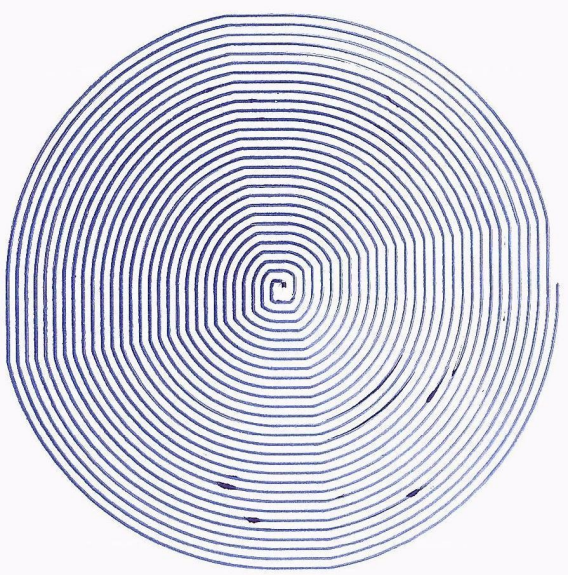


Figura 4.1: Barrido circular con avance de media pulgada

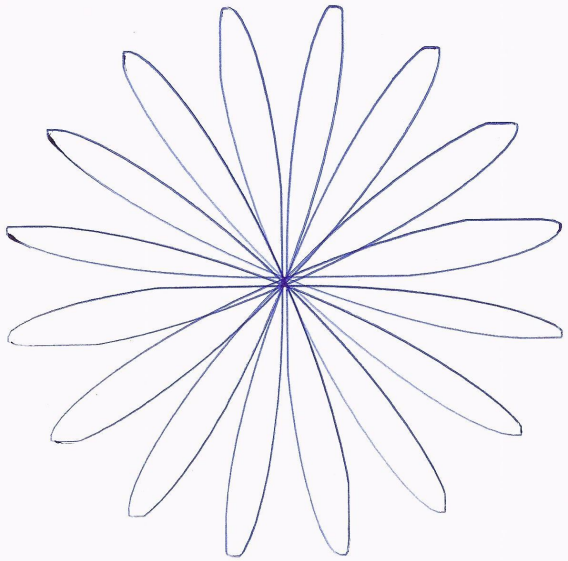


Figura 4.2: Función polar con pétalos de dos pulgadas

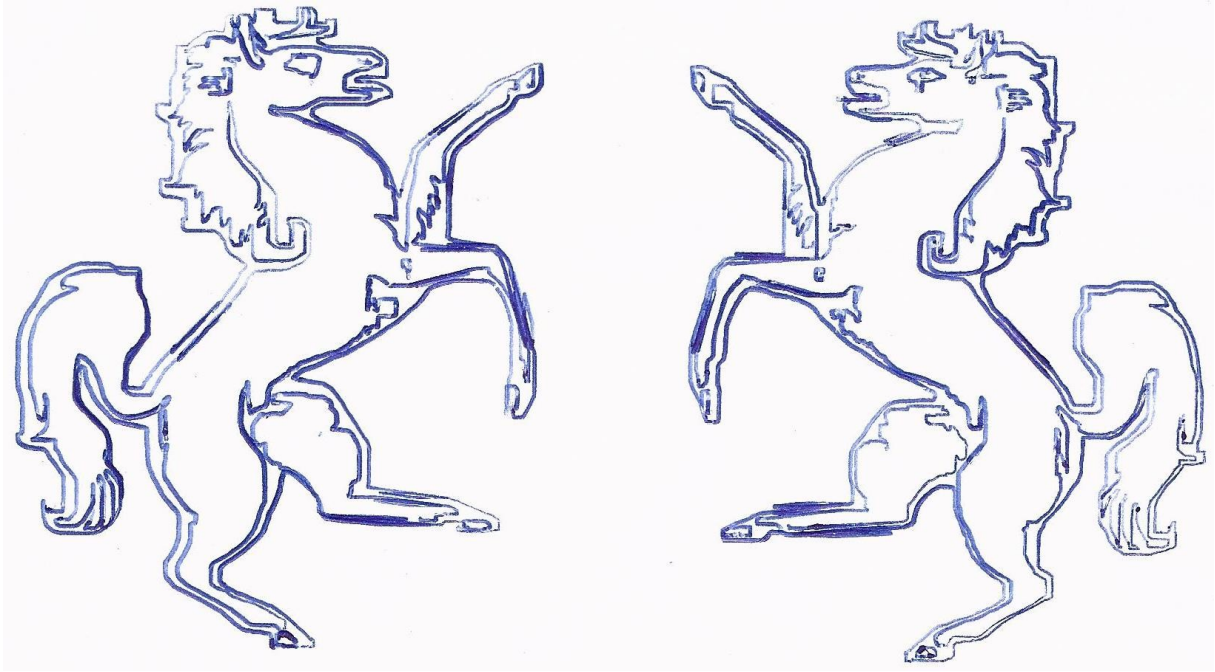


Figura 4.3: Dingbats

Figura 4.4: Paloma

Figura 4.5: Un grabado de Gustav Doré de El Quijote

En las *Figuras 4.6* y *4.7* se muestra el fresado y barrenado en una tarjeta fenólica. Algunas pistas han sido grabadas sobre la misma línea tanto de ida como de regreso y no en el rectángulo aislante requerido. Los barrenos sí corresponden con lo esperado. Aquí puede apreciarse claramente el error causado por la pérdida de pasos mientras se reacomoda la tuerca al cambio de sentido de giro en el husillo.

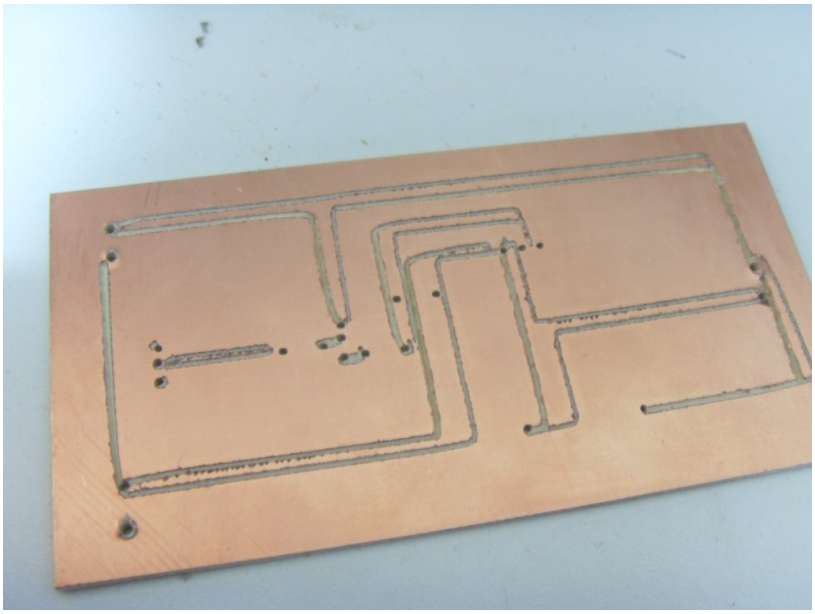


Figura 4.6: Fresado de pistas

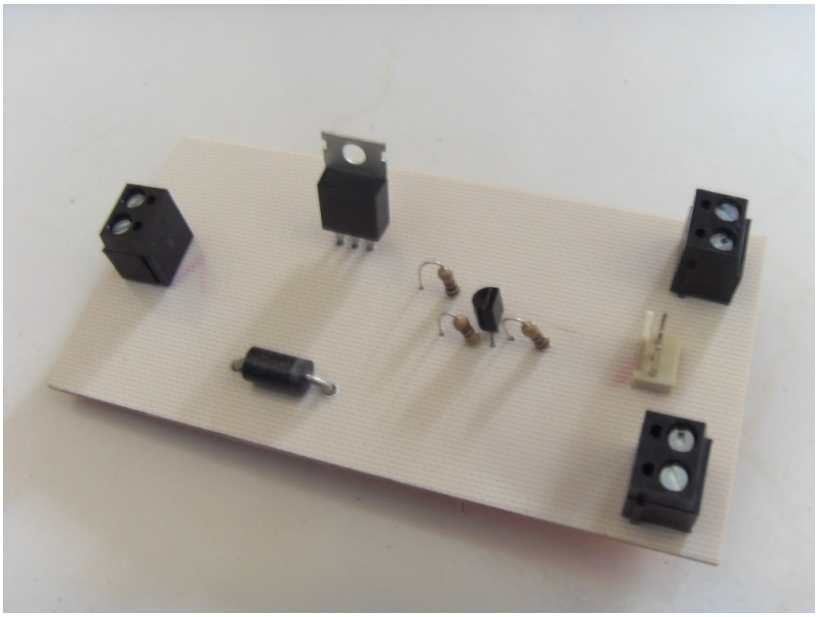


Figura 4.7: Barrenado para colocación de piezas

En la *Figura 4.8* se muestra el maquinado de letras. Las de arriba son letras de dos pasos y las de abajo de uno

Figura 4.8: Maquinado de letras

En la *Figura 4.9* puede verse el maquinado de una superficie mediante el código generado en FreeMill, consistente en un barrido rectangular en zigzag.

Figura 4.9: Maquinado de superficie

En la *Figura 4.10* se muestra el maquinado de una superficie en forma de plato y el posterior grabado de una figura sobre ésta

Figura 4.10: Proyección sobre superficies

En la *Figura 4.11* se presenta el maquinado de un relieve basado en un mapa de altura, el cual se hizo en tres pasos. El primero con un cortador plano de 3 mm de diámetro y después con cortadores esféricos de 4 y 3 mm, respectivamente. El tiempo de maquinado fue cercano a 6 horas

Figura 4.11: Mapa de altura

CONCLUSIONES

El objetivo del trabajo presentado se cumplió dado que se dispone de la máquina planeada, los controladores requeridos y de un gran abanico de posibilidades para aplicaciones de maquinado. Sin embargo, la expectativa más personal de apoyo al señor Luis Sánchez, se concretará hasta que la máquina realizada y la que actualmente se está construyendo operen en el taller y se le haya transmitido a él de manera sencilla y en un lenguaje simple lo contenido en este trabajo, para que pueda realizar sus programas de código numérico y, eventualmente, replicar los controladores para otros proyectos. Se mantendrá contacto con él.

Sobre las deficiencias y errores en los componentes se presentan algunos comentarios y sugerencias y finalmente algunas conclusiones de carácter general.

Sobre el controlador

- Se ha visto que la tarjeta basada en los CI's L297 y L298 es sencilla y funcional para valores de corriente menores a los 3 amperes, propia de motores pequeños, pero para aplicación en motores de mayor tamaño (NEMA ≥23), como seguramente se requerirán en el taller, resulta insuficiente.
- El controlador unipolar ha permitido utilizar la máquina a una velocidad de avance de 30 pulg/min, en contraste con los controladores bipolares, que logran sólo 12 pulg/min.
- El controlador de puentes H discretos basados en mosfets tiene la desventaja de poder utilizarse con un valor máximo de 20 volts, ya que este valor es el ingresado directamente en las compuertas. En este sentido es conveniente diseñar un puente H basado únicamente en mosfets de canal N, en el cual se pueda regular independientemente el voltaje de activación y de la bobina del motor. Para ello se

usarán CIs tales como el IR2101 o IR2104, que permiten activar la parte alta y baja de medio puente H basado en mosfets de canal N.

- Un aspecto que no se ha determinado de manera certera es si las tarjetas que utilizan mosfets requieren o no de los diodos que se conectan en parelelo con las bobinas del motor, debido a que tienen un diodo interno que funciona como un zener. Esto tiene que ver con el parámetro de la energía en avalancha y se continuará analizando hasta conocer la respuesta.
- Prácticamente todos los controladores comerciales tienen integrada la función de micropaso que permite obtener resoluciones que varían entre los 1000 y 50000 pasos por revolución. Si bien en el ANEXO A se muestra que este modo de control trae consigo importantes desventajas, se hubiera deseado conocer el rendimiento de la máquina bajo estas condiciones. Hay algunos CIs que tienen integrada la función de micropaso junto con métodos de regulación de corriente más sofisticados al del L297, tal como el A3967, que sin embargo, al ser de contacto superficial, requieren una tarjeta de circuito impreso más compleja a las hechas en este trabajo.
- El costo total de las tarjetas realizadas (i.e. una para cada eje y la de puerto parelelo) es de alrededor de $1000 pesos. A modo de comparación, véase que un controlador de motor a pasos comercial promedio ronda los $1500.

Sobre la máquina

Las deficiencias vistas en las pruebas de repetibilidad bidireccional se mostraron y comentaron con el dueño del taller, quien propuso algunas sugerencias para resolverlas.

- Eventualmente, se piensa cambiar el movimiento hecho hasta ahora mediante los husillos por un movimiento realizado mediante banda, con el cual se podrá tener una mayor velocidad y una respuesta inmediata ante el cambio de giro en los motores. Con

los husillos ésto no se logra debido a un reacomodo en la tuerca, debido al cual se pierde una cierta cantidad de pasos en el proceso.

- Si bien existe cierto tipo de acoplamientos (*zero backlash couplings*) con los cuales puede evitarse la pérdida de pasos al momento de cambio de giro, la implementación de este sistema puede costar 4 ó 5 veces más que el movimiento por banda.

Finalmente, se piensa ocupar la máquina para diversos trabajos en madera, aunque no se descarta su eventual uso en el maquinado de metales, ya sea en el fresado o con el uso de un equipo de plasma.

Sobre los programas

La paquetería CNC Toolkit es una poderosa herramienta que permite crear código para máquinas de 3, 4 y 5 ejes coordenados. En este trabajo no se ha entrado a detalle en ello, pero el lector interesado puede encontrar más información en el sitio del programa y en el tutorial de *cnc4free.org*.

Por último, tal como se ha mostrado, existe una gran cantidad de recursos gratuitos para realizar operaciones de maquinado, los cuales resultarán seguramente suficientes para lo que se tiene planeado realizar en el taller.

De carácter general

Los sistemas de control numérico del tipo *hágalo usted mismo* son cada vez más comunes, a juzgar por la enorme cantidad de foros y videos que muestran proyectos en la red. Sin embargo, su uso aún no se extiende ampliamente, en especial entre operarios de pequeños talleres y carpinteros, a quienes les facilitaría su trabajo y les permitiría mayor eficiencia y

competitividad.

Por otro lado, la necesidad del propietario del taller permitió involucrarse en este proyecto que a él le resultará benéfico para la modernización de procesos en su negocio, y al que presenta este trabajo, le dio la oportunidad de aprender y resolver de manera autodidacta lo que fue requiriendo su desarrollo.

Se ha visto que existe una gran cantidad de necesidades de orden técnico en éste y seguramente también en otros talleres, las cuales pueden constituírse, entre los estudiantes, en oportunidades para desarrollar proyectos similares al aquí realizado, que además tienen un sentido de apoyo y labor social.

ANEXO A - ASPECTOS TEÓRICO-PRÁCTICOS DE LOS MOTORES A PASOS

Las máquinas de control numérico requieren de un sistema de posicionamiento que sea lo más preciso posible. Para ello suelen utilizarse motores a pasos o servomotores, aunque no son los únicos medios para este propósito.

Los equipos industriales utilizan por lo general servomotores, cuyo control se realiza mediante un sistema en lazo cerrado en el cual se retroalimenta la posición del rotor mediante un *encoder* que envía una cierta cantidad de pulsos por revolución y son variables en cuanto a su resolución y costo. Algunos valores que se han visto son 3 mil, 260 mil, 1 millón y 16 millones de pulsos por revolución.

Los motores a pasos generalmente utilizan un sistema en lazo abierto que es más sencillo de diseñar y operar, por lo que son los preferidos para proyectos como el que aquí se presenta.

Funcionamiento

Los motores a pasos son un tipo de motores eléctricos que convierten un pulso eléctrico en un paso y pueden ser controlados precisamente en términos de cuánto habrán de rotar y a qué velocidad, para lo cual requieren de un controlador que se encarga de enviar esos pulsos.

El número de pasos que con que se mueve el motor es el mismo que envía el controlador y se mueve a una velocidad igual a la frecuencia con que son enviados los pulsos.

Una de las cualidades es su habilidad para posicionarse con precisión, además de que su

rotor, de una inercia pequeña, les permite acelerar rápidamente y los hace ideales para movimientos cortos y rápidos.

Hay tres tipos de motores a pasos, que son de *reluctancia variable*, *imán permanente* e *híbridos*. Los de reluctancia variable tienen tanto un rotor como estator endentados, pero carecen de imán. Los de imán permanente tienen un imán como rotor que carece de dientes. Finalmente, los híbridos combinan el imán y los dientes de los anteriores.

A continuación se describe la construcción de un motor híbrido relativamente común que realiza 200 pasos por revolución, tal como el mostrado en la *Figura A.1* y que es además del tipo utilizado en la máquina construida.

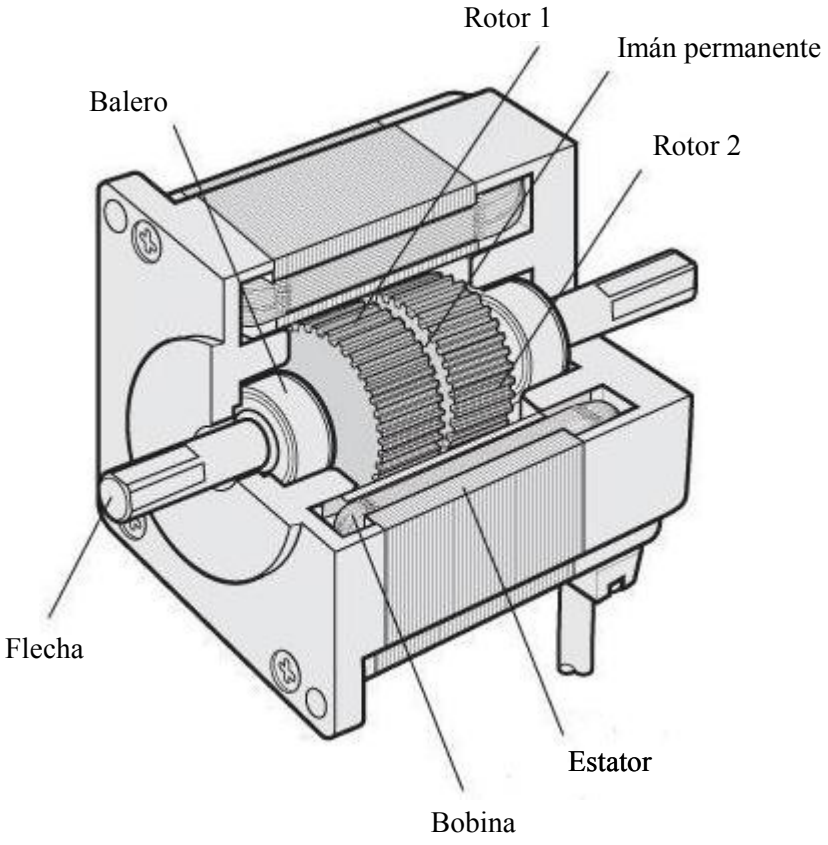


Figura A.1: Construcción de un motor a pasos híbrido típico

El imán está magnetizado axialmente, de manera que en el rotor, la mitad superior es polo norte y la inferior sur. En el imán hay dos ruedas dentadas con 50 dientes cada una que están desfasadas entre sí 3.6 º. Donde hay un espacio entre dientes de una rueda se encuentra un diente de la otra.

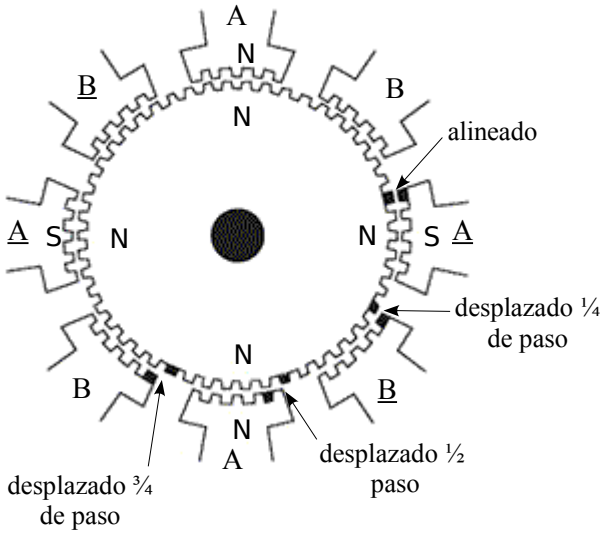


Figura A.2: Bobinas de un motor a pasos híbrido de dos fases.

Este tipo de motores son regularmente de dos, tres o cinco fases[28], pero aquí se tratará únicamente el caso de los de dos fases.

Tienen cuatro polos por fase que están separados cada 90º. Están enrollados de manera que los separados 180º son de la misma polaridad, mientras que los separados 90º son de polaridad opuesta. Si la corriente en una fase se invierte, cambia su polaridad, de manera que cualquier polo del estator puede ser un polo norte o sur.

28 Para información de motores de cinco fases puede verse lo publicado por Oriental Motor, que construye este tipo de motores. *http://www.orientalmotor.com/MotionControl101/2phase-v-5phase.html*

En la *Figura A.2*, supóngase que los polos a las 12 y a las 6 son norte y los polos a las 3 y 9 son sur. Al energizarlos, los polos de las 12 y 6 atraen al polo sur del rotor magnético y 3 y 9 atraen el polo norte de éste. Visto desde el lado opuesto, se verían los dientes alineándose a las 12 y 6. Dependiendo de en qué dirección se desee girar, tendrían que energizarse los polos de las 2 y 7 o los de las 11 y 5. Es aquí donde se requiere del controlador para determinar la secuencia necesaria.

El rotor tiene 50 dientes y un paso entre dientes de 7.2°. Al moverse, algunos dientes están desalineados en relación con los dientes del estator ¾, ½ ó ¼ de paso. Cuando se realiza un paso, se toma la ruta más fácil y, dado que ¼ de 7.2 es 1.8, el motor se mueve 1.8° cada paso, tal como se muestra en la *Figura A.2*.

Hay motores con 100 dientes en el rotor que permiten 400 pasos por revolución. Por otro lado, los motores de cinco fases igualmente con rotores de 50 ó 100 dientes permiten resoluciones mayores de 500 ó 1000 pasos por revolución, respectivamente. El controlador requerido por un motor de dos fases es distinto a otro de motor de cinco fases.

Zonas de operación

Un parámetro para la selección de un motor a pasos son las curvas Torque-Velocidad, donde la Velocidad está representada en las abscisas y Torque en las ordenadas, la cuales tienen una forma similar a la mostrada en la *Figura A.3*. En ellas hay una serie de términos que necesitan ser definidos.

- Torque de contención – es la cantidad de torque que produce el motor en reposo cuando está energizado.
- Zona de arranque-parada – En esta zona el motor puede arrancar y detenerse

inmediatamente.

- Rango de torcedura – Esta zona se encuentra entre el *Torque de llegada* y *Torque de salida*, y para operar en ella debe iniciarse el movimiento en la *Zona de arranque-parada* e incrementarse la frecuencia de los pulsos de alimentación para ingresar en ella o decrementarla para salir de ella. De lo contrario, el motor pierde el seguimiento de los pulsos y pierde pasos, por ejemplo, al cambiar la dirección de giro.
- Velocidad máxima de arranque – es la frecuencia máxima a la cual puede comenzar a girar el motor, medida sin carga.
- Velocidad máxima – Es el máximo valor a que puede girar el motor sin perder sincronicidad con los pulsos de alimentación, medida sin carga.
- Torque de llegada (*pullin torque*) – son los valores de torque máximos en los cuales el motor puede arrancar, detenerse o invertir el sentido de giro (bajo carga), en sincronicidad con la frecuencia de los pulsos de alimentación.
- Torque de salida (*pullout torque*) – Son los valores máximos en que un motor (bajo carga) puede girar sin pararse o atascarse.
- Zona de arranque-parada – en esta área el motor puede arrancar, detenerse o cambiar de giro en sincronizidad con los pulsos de entrada.

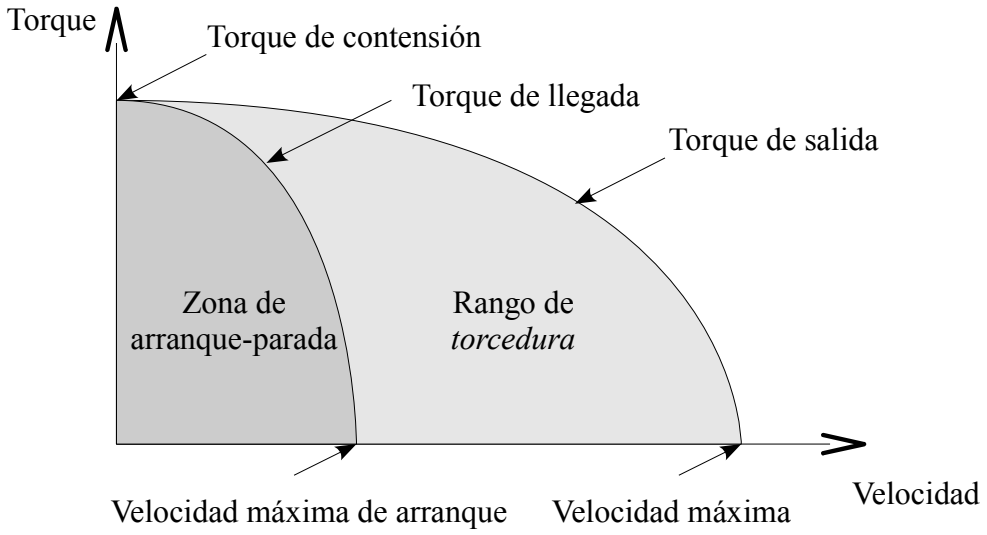


Figura A.3: Zonas de operación de un motor a pasos

La curva real de un motor a pasos controlado en modo bipolar y unipolar (términos que serán explicados más adelante) es mostrada en la *Figura A.4* En el modo bipolar se muestran las curvas para dos voltajes.

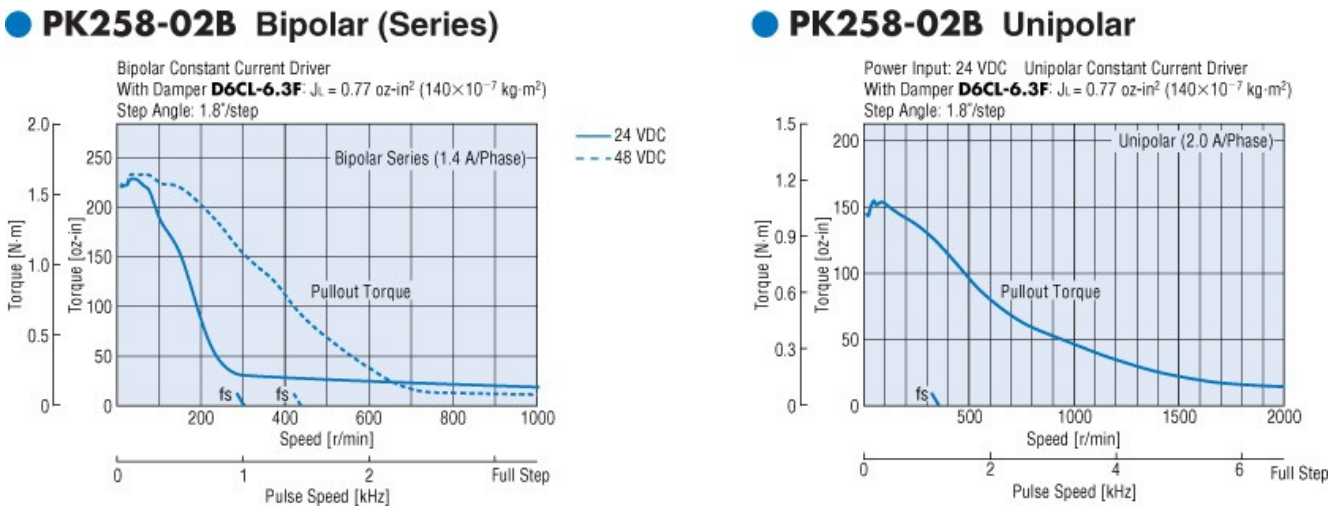


Figura A.4: Curva real de un motor a pasos

Hay varios factores que afectan el torque producido por estos motores, tales como la frecuencia de los pasos, la corriente que circula en las bobinas y el modo en que son controlados.

La inductancia afecta el torque en altas velocidades. Cada motor tiene un cierto valor de inductancia y resistencia. La inductancia está en henrys, que dividida por resistencia (ohms) resulta un valor en segundos. Este cantidad es el tiempo que le toma a la bobina cargarse al 63% de su valor nominal. Si un motor tiene un valor nominal de 1 ampere, después de una constante de tiempo la bobina estará en 0.63 amperes y en casi 1 ampere después de 4 ó 5 constantes de tiempo. Dado que el torque es proporcional a la corriente, si el motor ha sido cargado con 63% de su corriente entregará sólo 63% de su torque.

En bajas velocidades la corriente puede entrar y salir sin problema, pero en altas no puede entrar lo suficientemente rápido antes de que se encienda la siguiente fase y por lo tanto el torque se ve reducido.

En este sentido, el voltaje de alimentación juega un papel importante en el desempeño del motor en altas velocidades, ya que permite una energización más rápida de la bobina.

Modos de control

Hay cuatro maneras de controlar este tipo de motores:

1. Paso completo con
    a) una fase activa
    b) dos fases activas
2. Medio paso
3. Micropaso

1.a) Paso completo con una fase activa

Consta de una secuencia eléctrica de cuatro pasos para rotar el motor y en cada uno se energiza sólo una fase. En la *Figura A.5* se muestra esta secuencia para un hipotético motor de 4 pasos por revolución. Cada paso de 90° en este motor corresponde al paso de 1.8° del motor antes visto.

Cuando se energiza un polo, por ejemplo, como polo sur, atrae al polo norte del rotor

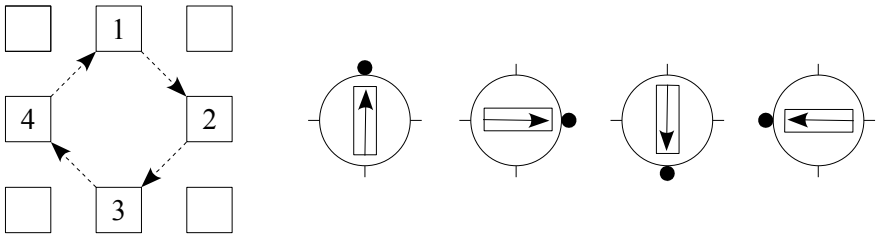


Figura A.5: Control de una fase activa

1.b) Paso completo con dos fase activas

En este modo se energizan las dos fases en cada paso y el rotor es atraído a dos polos que tienen la misma polaridad, alineándose finalmente el polo opuesto de éste a la mitad de ambos. *Figura A.6*.

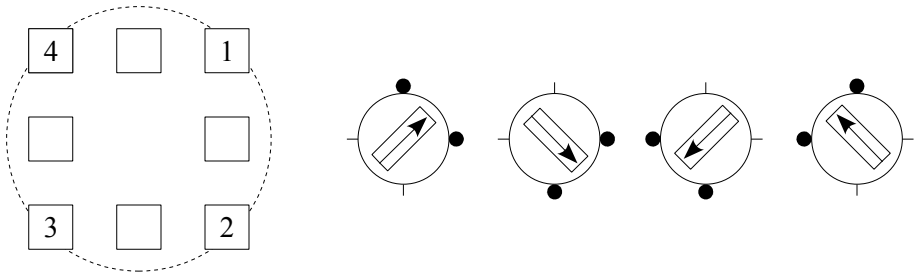


Figura A.6: Control de dos fases activas

Este modo produce más torque, ya que en el de una fase activa actúa una unidad de torque sobre el rotor, mientras que en el de dos fases activas actúan dos unidades (una a las 12 y otra a las 3), que sumadas vectorialmente resultan en un vector con módulo de 1.41 (i.e.- 40% más torque).

2) Medio paso

Este modo se basa en los dos anteriores y permite reducir el ángulo de paso a la mitad. En este caso hipotético de 90 a 45º y en el motor real explicado anteriormente de 1.8 a 0.9º. *Figura A.7*.

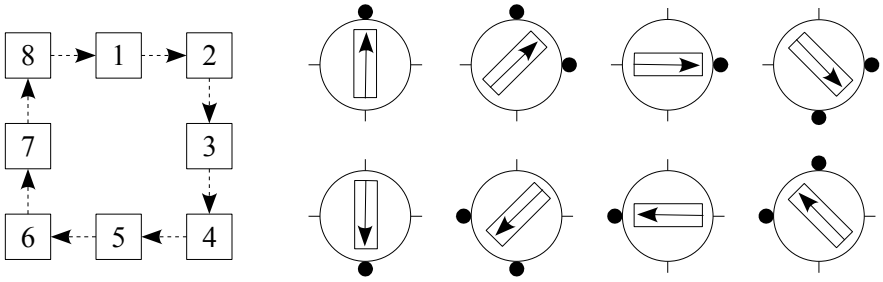


Figura A.7: Control de medio paso

3) Micropaso

Es un modo para reducir el ángulo de paso y con ello aumentar la resolución del motor. En este modo las fases no están totalmente prendidas o apagadas, sino que se genera una señal cuasisenoidal por la que se va incrementando (o decrementando) la corriente a través de las fases. De esta manera el rotor se aproxima en pequeños pasos de una fase a otra. La división de pasos (dependiendo del controlador) varía normalmente de 4 a 256.

Sin embargo, mientras se incrementa la resolución (i.e. la cantidad de pasos) el *torque incremental* por micropaso decrementa drásticamente. La resolución se incrementa pero la precisión se reduce.

La expresión del torque incremental para un micropaso es:

$$T_{inc} = T_{cont} \cdot sen(90/\mu_{ppc})$$

donde:

T_{inc}: Torque incremental por micropaso

T_{cont}: Torque de contensión en paso completo

m_{ppc}: número de micropasos por paso completo

La *Figura A.8* muestra el descenso súbito del torque incremental como función de los micropasos. Para 16 micropasos el torque incremental es ya 10% del torque de contensión.

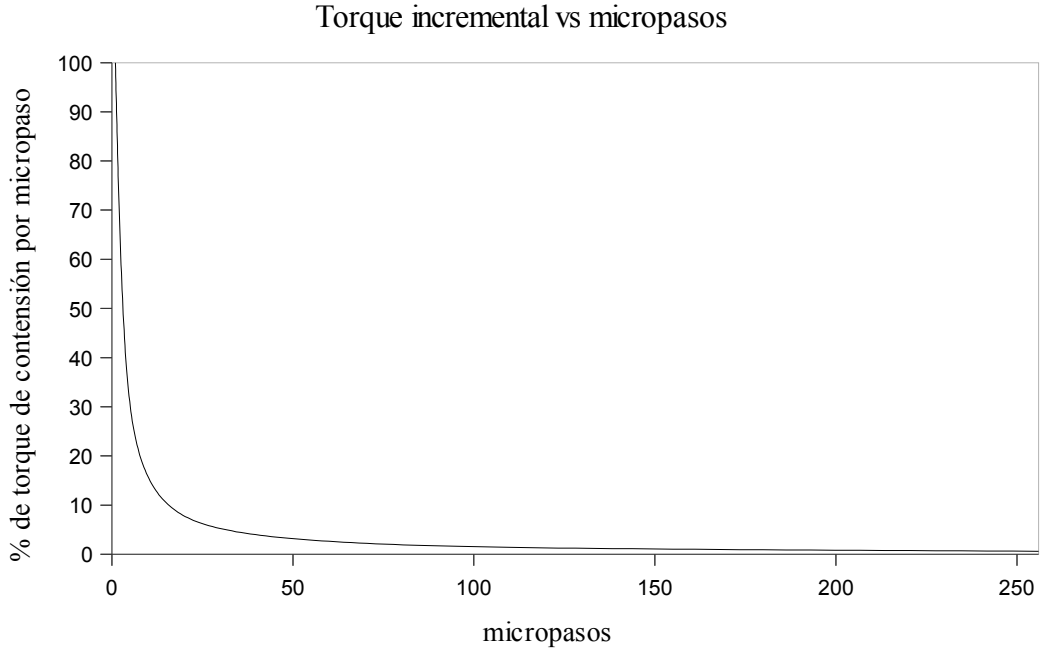


Figura A.8: Curva de torque incremental vs. micropasos

A pesar de ello, hay otras razones para elegir el micropaso más allá de la resolución, tales como un rendimiento más suave, menor ruido mecánico y menor vibración.

Motores unipolares y bipolares

Los motores de dos fases pueden ser a su vez unipolares o bipolares. Esto tiene que ver con la forma en que están conectadas y distribuidas las fases en el motor y de manera indirecta con el modo en que son controlados. En el caso unipolar cada fase tiene un tap central que normalmente se conecta a la alimentación y se va alternando el extremo hacia el cual circula la corriente, tal como se muestra en la *Figura A.9* a. En el caso bipolar se tiene una bobina en la cual se necesita invertir el sentido de circulación de la corriente a través de ésta para tener el cambio de polaridad en los polos del motor, como se muestra en la *Figura A.9* b.

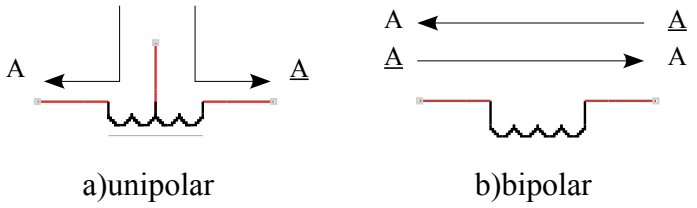


Figura A.9: Tipo de fase de los motores a pasos

Los motores unipolares producen menos torque debido a que trabajan solamente con la mitad de la bobina encendida, pero tienen un mejor rendimiento en altas velocidades debido a que su inductancia es menor. Los motores bipolares producen más torque en velocidades pequeñas pero en altas su mayor inductancia impide que las bobinas se energizen/desenergizen correctamente y se reduce su rendimiento.

Un motor unipolar puede ser controlado en modo bipolar si se dejan desconectados los taps centrales de las fases, pero lo contrario no se cumple; un motor bipolar no puede ser controlado en modo unipolar.

Los motores de dos fases pueden ser de cuatro, seis u ocho cables, tal como se muestra en la *Figura A.10*. Hay algunos que tienen cinco cables que bien pueden ser unipolares en los cuales los taps centrales se han conectado en un solo cable o ser de cinco fases, en cuyo caso todos los cables presentan continuidad entre sí.

Los motores a pasos de ocho cables suelen ser los de mayor tamaño y pueden conectarse de distintas maneras dependiendo de la necesidad o posibilidad. Puede realizarse una conexión en modo unipolar o bipolar en serie y paralelo, tal como se muestra en la *Figura A.11*.

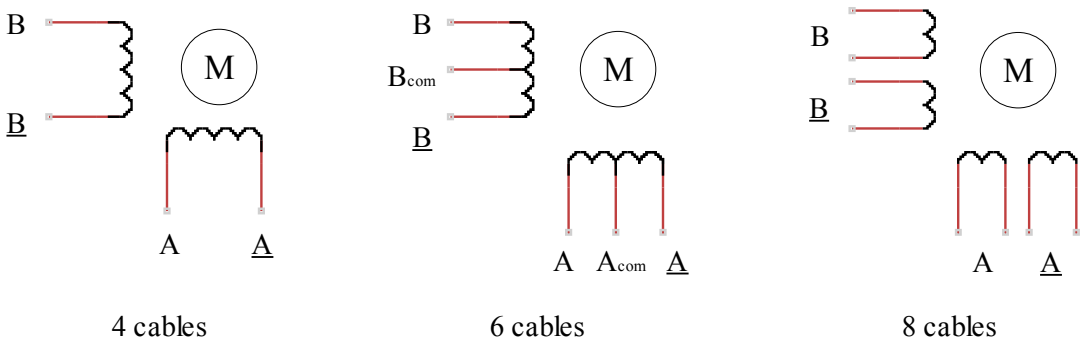


Figura A.10: Configuración de cableado en los motores

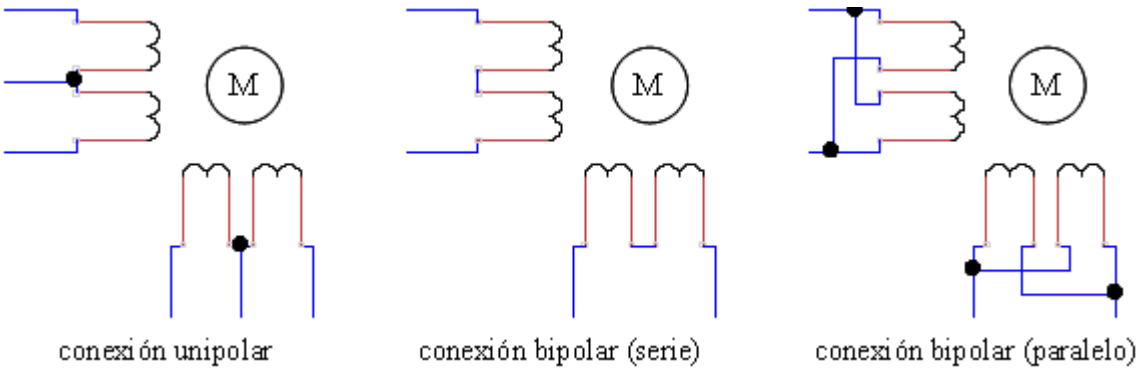


Figura A.11: Configuraciones de conexionado en motores de ocho cables

ANEXO B - PROGRAMACIÓN EN CÓDIGO G

A continuación se presentan los fundamentos de la programación en código G y algunos de sus comandos para el lenguaje *RS274/NGC* soportado por el EMC2 y el cual, si bien tiene comandos específicos del mismo, la mayoría son estándar dentro de este lenguaje de programación[29].

Fundamentos

Un programa está basado en líneas de código, también llamadas bloques y éstos a su vez incluyen comandos para realizar distintas operaciones.

Un bloque típico consta de un número de línea opcional seguida por una o más *palabras* siendo éstas una letra seguida de un número (o algo que evalúa un número). Una palabra puede dar directamente un comando o proveer un argumento para éste. La mayoría de los comandos comienzan con G o M (generales o misceláneos)

Un programa puede estar contenido en uno o varios archivos y éstos pueden comenzar y terminar con un signo de porcentaje (%) para facilitarle la tarea al interpretador. Esto es opcional si el archivo tiene los comandos M2 o M30, pero es necesario si no lo tiene. Se marcará un error si se utiliza un signo de porcentaje al inicio y no al final.

Formato de una línea

Una línea consiste en el orden marcado que se muestra a continuación, con la restricción de que sea menor a 256 caracteres.

29 Extractos del Manual de Usuario (pp. 59-102)

- Un carácter (opcional) supresor del bloque, que es una diagonal (/).
- Un número de línea (opcional).
- Cualquier número de palabras, configuración de palabras y comentarios.

Número de línea

Es la letra N seguida por número entero positivo no mayor a 5 dígitos. Puede o no ser utilizado, aunque se recomienda no hacerlo ya que no tiene sentido.

Palabras

Una *palabra* es una letra (excepto N) seguida de un número real y puede comenzar con las letras mostradas en la *Tabla B.1*

Tabla B.1: Letras utilizadas en los códigos numéricos.

Letras y sus significados	
Letra	**Significado**
A	Eje *A* de la máquina
B	Eje *B* de la máquina
C	Eje *C* de la máquina
D	Número de compensación del radio de la herramienta
F	Velocidad de avance
G	Función general
H	Índice de compensación de longitud de herramienta
I	Compensación para arcos en *X* y ciclos de barrenado
J	Compensación para arcos en *Y* y ciclos de barrenado
K	Compensación para arcos en *Z* y ciclos de barrenado
	Razón de movimiento-spindle para movimientos sincronizados G33
M	Función miscelánea
N	Número de línea
P	Tiempo de espera en los ciclos de barrenado y con G4
	Llave utilizada con G10
Q	Incremento de avance en los ciclos G73 y G83
R	Radio del arco o plano
S	Velocidad del spindle
T	Selección de herramienta
U	eje *U* de la máquina
V	eje *V* de la máquina
W	eje *W* de la máquina
X	eje *X* de la máquina
Y	eje *Y* de la máquina
Z	eje *Z* de la máquina

Números

- Un número consta de un signo opcional de más (+) o menos (-) seguido por cero o varios dígitos y continuado posiblemente por un punto decimal seguido por cero o varios dígitos (Ej. 987.6545).
- Pueden ser enteros o decimales.
- Un número distinto de cero sin signo es asumido como positivo

Parámetros numerados

Un parámetro numerado es el símbolo de gato (#) seguido por un número entero entre 1 y 5399. Este número determina el *número de parámetro*. Un valor es almacenado en un parámetro mediante el operador igual (=) (Ej. "#3=15" asigna el valor de 15 al parámetro 3).

Parámetros nombrados

Funcionan igual a los numerados pero son más fáciles de leer. Los nombres del parámetro son convertidos a letras minúsculas y se suprimen espacios y tabuladores. Estos deben ser encerrados con los símbolos < > .

#<parámetro> Es un parámetro nombrado local. Tienen validez en el contexto en que es asignado, de tal manera que no puede accederse a él fuera de la subrutina en que se asigna. Por esto puede utilizarse el mismo parámetro en subrutinas distintas sin temor a que se cambien sus valores entre una y otra.

#<_parámetro> Es un parámetro nombrado global. Puede accederse a ellos dentro de subrutinas llamadas y asignar valores dentro de ellas.

Expresiones

Una expresión es un conjunto de caracteres que comienza con el corchete izquierdo y terminan con el derecho. Dentro de ellos hay números, valores de parámetros, operaciones matemáticas y otras expresiones. Una expresión es evaluada para producir un número (Ej. [5+COS[90]-[3*EXP[3]]]).

Operadores binarios

Aparecen solamente dentro de expresiones. Hay 4 operaciones matemáticas básicas: adición(+) sustracción (-) multiplicación (*) y división(/). Hay tres operaciones lógicas (AND), (OR) y (XOR).
La octava operación es la de módulo (MOD). La novena operación es potencia (**), que eleva el número de la izquierda a la potencia de la derecha (Ej. 5**3 eleva 5 al cubo).
Los operadores relacionales igual (EQ) no igual (NE), mayor que (GT), mayor o igual que (GE), menor que(LT) y menor o igual que(LE).
Las operaciones lógicas son realizadas en cualquier número real. El número cero es equivalente a un *falso lógico* y cualquier número distinto de cero equivale a *verdadero lógico*.

Funciones

Las funciones se muestran se muestran en la *Tabla B.2*.

Los argumentos para las operaciones que toman medidas en ángulos (seno, coseno y tangente) están en grados.

La operación FIX redondea a la *izquierda* (menos positivo o más negativo) (Ej. FIX[2.8] =2 y

FIX[-2.8]= -3). La operación FUP redondea a la *derecha* (Ej. FIX[2.8] =3 y FIX[-2.8]= -2).

Tabla B.2: Funciones soportadas en el código RS274 / NGC.

Función	Resultado
ATAN[Y]/[X]	Tangente inversa de cuatro cuadrantes
ABS[arg]	Valor absoluto
ACOS[arg]	Coseno inverso
ASIN[arg]	Seno inverso
COS[arg]	Coseno inverso
EXP[arg]	*e* elevado a la potencia indicada
FIX[arg]	Redondear hacia abajo a un entero
FUP[arg]	Redondear hacia arriba a un entero
ROUND[arg]	Redondear al entero más próximo
LN[arg]	Logaritmo base e
SIN[arg]	Seno inverso
SQRT[arg]	Raíz cuadrada
TAN[arg]	Tangente

Comentarios

Estos pueden ser agregados a las líneas de código para clarificar la intención del programador. Estos están encerrados entre paréntesis “( )” o al final de la línea usando punto y coma (;).

Ej. G0 (movimiento rápido) X1 Y1

G0 X1 Y1; movimiento rápido

Códigos G

G0 – movimiento lineal rápido

G0 *ejes*

Se utiliza para un movimiento rápido en línea recta, generalmente cuando no se está realizando corte.

G1 – movimiento lineal

G1 *ejes*

Se utiliza para un movimiento lineal a la velocidad de avance programada, generalmente cuando se realiza corte. Se marcará error si no se ha especificado velocidad de avance.

G2, G3 – arcos de circunferencia

Un arco circular o helicoidal se especifica con los comandos G2 (horario) o G3 (antihorario). El eje del arco debe ser paralelo a los ejes *x*, *y* o *z*. El eje (o equivalentemente, el plano perpendicular al eje) se selecciona con G17 (eje *z*, plano *xy*), G18 (eje *y*, plano *xz*) o G19 (eje *x*, plano *yz*). Los planos 17.1, 18.1 y 19.1 no están soportados actualmente.

Hay dos formatos para especificar un arco: formato de centro y de radio.

Es un error si no se ha especificado velocidad de avance

Formato de centro (formato preferido)

En este formato se especifican las coordenadas del punto final del arco así como la de desplazamiento del centro del arco en relación a la posición actual. En este formato el punto final puede ser el mismo que la posición actual (en cuyo caso se tendría un círculo completo).

Es un error si el radio del arco varía en el punto inicial y final por una distancia mayor a 0.0002 [pulg] ó 0.002 [mm] (dependiendo de las unidades de trabajo).

En el plano *xy* se programa

G2 ó G3 *ejes* I- J-

Las palabras de ejes son opcionales excepto que al menos X o Y deben ser utilizados si se desea programar un arco menor a 360 grados. I (distancia en x) y J (distancia en y) es la coordenada del centro del arco desplazada con respecto a la posición actual. Si se incluye algún valor en Z, se hará una espiral. Es un error si se omiten *I* y *J*.

En el plano *xz* se programa

G2 ó G3 *ejes* I- K-

En el plano *yz* se programa

G2 ó G3 *ejes* J- K-

Ejemplo

Si se está trabajando en el plano *xy* y la coordenada de inicio es el punto (0,0), para formar un arco trazado en sentido horario de radio uno (1) que finaliza en el punto (1,1), tendría que utilizarse el código mostrado para formar el trazo mostrado en la *Figura B.1*. En este caso sólo se ha utilizado el desplazamiento en *x* dado por I debido a que al ser cero el desplazamiento en *y* (J) puede omitirse.

G2 X1 Y1 I1

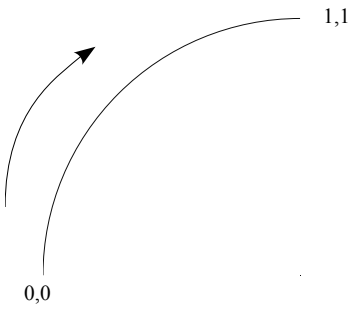


Figura B.1: Trazado de arco de circunferencia

Círculos completos

Para programar círculos completos se utilizan sólo los desplazamientos I, J o K.

G2 ó G3 I- J- K-

G4 – demora

G4 P[segundos]

Este comando mantendrá los ejes sin moverse durante el periodo de tiempo en segundos especificado por el número P.

Es un error si el número P es negativo.

G17, G18, G19, G17.1, G18.1, G19.1 – selección de plano

Estos códigos establecen el plano de trabajo, tal como se muestra en la *Tabla B.3*.

Tabla B.3: Códigos de los ejes coordenados.

Código	Plano
G17	XY
G18	XZ
G19	YZ
G17.1	UV
G18.1	UW
G19.1	VW

G20, G21 - unidades

G20 se utiliza para utilizar unidades en pulgadas y G21 para unidades en milímetros.

Es conveniente el programar estos comandos cerca del inicio del programa y no cambiarlo por otro durante el programa.

G40 – detiene la compensación radial

G40 detiene la compensación radial del cortador. El siguiente movimiento debe ser un movimiento lineal.

Es un error si se programa un movimiento de arco G2/3 después de G40.

G41, G42 – compensación radial del cortador

G41 ó G42 D[herramienta]

G41 comienza la compensación radial a la izquierda de la línea programada.

G42 comienza la compensación radial a la derecha de la línea programada.

El movimiento debe ser por lo menos tan largo como el radio de la herramienta y puede ser un movimiento rápido.

La compensación sólo puede ser realizada si está activo el plano XY.

Compensación radial del cortador de la tabla de herramientas.

La palabra D es opcional. Si no aparece, se usará la herramienta actual en el spindle. Si se utiliza, el número D es normalmente el número de ranura (*slot)* de la herramienta del spindle en uso, aunque no es necesario.

G90, G91 – modo de distancia

G90 establece el modo de distancia absoluto (las coordenadas tienen valor en referencia a un punto fijo, por ejemplo, el origen) y G91 el modo incremental (las coordenadas representan incrementos respecto a la posición actual/última de la máquina).

Códigos M

M0, M1, M2, M30, M60 – Detención y fin de programa

Para pausar un programa temporalmente (indepedientemente del botón de detención opcional), programe M0. EMC2 se mantiene en modo automático, de manera que el modo MDI y otras acciones manuales no están activadas.

Para pausar un programa temporalmente (pero sólo si está activado el botón de detención opcional), programe M1. EMC2 se mantiene en modo automático, de manera que el modo MDI y otras acciones manuales no están activadas.

Para intercambiar las *pallet shuttles* y detener un programa temporalmente (independientemente de la configuración del botón de detención opcional), programe M60

Si un programa es detenido con M0, M1 ó M60, al presionar el botón de inicio de ciclo se reanuda el programa a partir de la siguiente línea.

Para terminar un programa, programe M2. Para intercambiar las *pallet shuttles* y terminar un programa, programe M30. No se ejecutan líneas de código después de estos comandos y al presionar el botón de inicio de ciclo, el programa comienza al inicio del archivo.

M3, M4, M5 – control del spindle

Para comenzar a mover el spindle en sentido horario a la velocidad programada, programe M3.

Para comenzar a mover el spindle en sentido antihorario a la velocidad programada, programe M4.

Para detener el spindle, programe M5.
Si la velocidad programada es cero y se programa M3 ó M4, el spindle no se moverá. Si después se programa una velocidad distinta de cero, aquél comenzará a moverse.

M7, M8, M9 – control de refrigerante

Para encender el refrigerante tipo *mist*, programe M7.

Para encender el refrigerante tipo *flood*, programe M8.

Para apagar todo refrigerante, programe M9.

M62 a M65 – control de salidas

Para controlar un bit de salida digital, programe M- P-, donde la palabra M está en el rango 62-65 y P desde 0 a 3. De ser necesario, este número puede ser incrementado usando el parámetro *num_dio* al cargar el controlador de movimiento. A este respecto puede verse el Manual del integrador en la sección de configuración de la sección EMCy HAL para mayor información.

M62 – enciende una salida digital sincronizada con movimiento.

M63 – apaga una salida digital sincronizada con movimiento.

M64 – enciende una salida digital inmediatamente.

M65 – apaga una salida digital inmediatamente.

Los comando M62 y M63 serán puestos en espera y entonces comandos subsecuentes que se refieran al mismo número de salida sobreescribirán las instrucciones precedentes. Más de un cambio en las salidas puede ser especificado al programar más de un comando M62/63.

El cambio efectivo de las salidas sucederá al inicio del siguiente comando de movimiento. Si no hay un comando subsecuente de movimiento, el cambio en las salidas no sucederá. Es conveniente utilizar un comando de movimiento (i.e. G0, G1, etc.) justo después de un M62/63.

M64 y M65 suceden inmediatamente que son recibidos por el controlador y no están sincronizados con movimiento.

M66 – control de entradas

Para leer el valor de un pin de entrada digital o analógico, programe P- E- L- Q-, donde la palabra P y la E varían de 0 a 3. De ser necesario el número de entradas y salidas puede ser incrementado al usar el parámetro *num_dio* o el *num_aio* al cargar el controlador de movimiento. A este respecto puede verse el Manual del integrador en la sección de

configuración de la sección EMCy HAL para mayor información. Sólo una de las palabras P o E pueden estar presentes. Es un error si ambas están ausentes.

M66 espera de una entrada

- La palabra P- especifica el número de entrada digital.
- La palabra E- especifica el número de entrada analógica.
- La palabra L- especifica el tipo de espera:

  0 Modo de espera inmediato – no espera, regresa inmediatamente. El valor actual de la entrada es guardado en el parámetro #5399.

  1 Modo de espera de incremento – espera a que la entrada realice un evento de incremento.

  2 Modo de espera de decremento – espera a que la entrada realice un evento de decremento.

  3 Modo de espera alto – espera a que la entrada seleccionada vaya a un estado alto.

  4 Modo de espera bajo – espera a que la entrada seleccionada vaya a un estado bajo.
- La palabra Q especifica el tiempo de lapso de espera. Si este lapso es sobrepasado, la espera es interrumpida y la variable #5399 tendrá el valor de -1. El valor de Q es ignorado si la palabra L- es cero (inmediato). Un valor de Q igual a cero es un error si la palabra L- es diferente de cero.
- Modo 0 es el único permitido para una entrada analógica.

La espera de una entrada por M66 detiene la ejecución posterior del programa hasta que el evento seleccionado (o el lapso de espera) ocurra.

Es un error el programar M66 tanto con una palabra P- como E- (i.e. una entrada tanto digital como analógica). En EMC2 éstas entradas no son monitoreadas en tiempo real y por lo mismo

no deben usarse para aplicaciones de tiempo críticas.

Códigos O

Permiten el control de flujo en los programas. Cada bloque tiene un número asociado, el cual es usado después de la O.

El comportamiento es incierto si:

- otras palabras son utilizadas en la misma línea de la palabra O
- se hacen comentarios en la misma línea

Subrutinas: sub, endsub, return, call

Las subrutinas se extienden desde "O- sub" hasta "O- endsub". Las líneas dentro de la subrutina (el cuerpo) no son ejecutadas tal como aparecen en el programa, sino cada vez que la subrutina es llamada con "O- call".

Ej.

...

(subrutina que envía al origen)

O100 sub

G0 X0 Y0 Z0

O100 endsub

O100 call

...

Dentro de una subrutina puede ejecutarse “O- return”, que regresa a la línea de llamado tal como si se hubiera encontrado “O- endsub”.

Ciclos (Loops): do, while, endwhile, break, continue

El ciclo while tiene dos estructuras: *while/endwhile y do/while*. En cada caso, el ciclo se termina cuando la condición que se evalúa es falsa.

Ej.

```
...
(dibuja una recta a 45º)
#1 = 0
O101 while [#1 lt 10]
G1 X[#1] Y[#1]
#1 = [#1+1]
O101 endwhile
...
```

Dentro de un ciclo while, “O- break” termina inmediatamente el ciclo y “O- continue” salta inmediatamente a la próxima evaluación de la condición while.

Condicional: if, else, endif

El condicional if ejecuta un grupo de declaraciones si una condición es verdadera y otro grupo si es falsa

Ej.

...

(llama a una u otra subrutina)

O102 if [#2 eq 0]

O130 call

O102 else

O148 call

O102 endif

...

Repeat

Este comando hará que se ejecuten las declaraciones dentro del repeat/endrepeat el número especificado de veces.

Ej.

...

(llama cinco veces una subrutina)

O103 repeat [5]

O42 call

O103 endrepeat

...

Direccionamiento indirecto

El número del código puede ser dado por un parámetro o cálculo.

Ej.

...

O[#101+2] call

...

Los códigos O son especialmente útiles para el cálculo de parámetros, expresiones, operadores binarios y funciones.

Otros códigos

F – establece velocidad de avance

Para programar la velocidad de avance se utiliza el comando F <n>, donde <n> es un número y dependiendo si las unidades están en milímetros o pulgadas, las unidades serán mm/min o pulg/min, respectivamente.

S – establece velocidad del spindle

Para determinar la velocidad del spindle en unidades de revoluciones por minuto (rpm), se programa S <n>. El spindle comenzará a girar a esa velocidad cuando se programe que comience a moverse (M3 ó M4). Puede programarse M0, en cuyo caso el spindle no se moverá y se marcará un error si <n> es un número negativo.

ANEXO C - DIAGRAMAS DE LOS CONTROLADORES

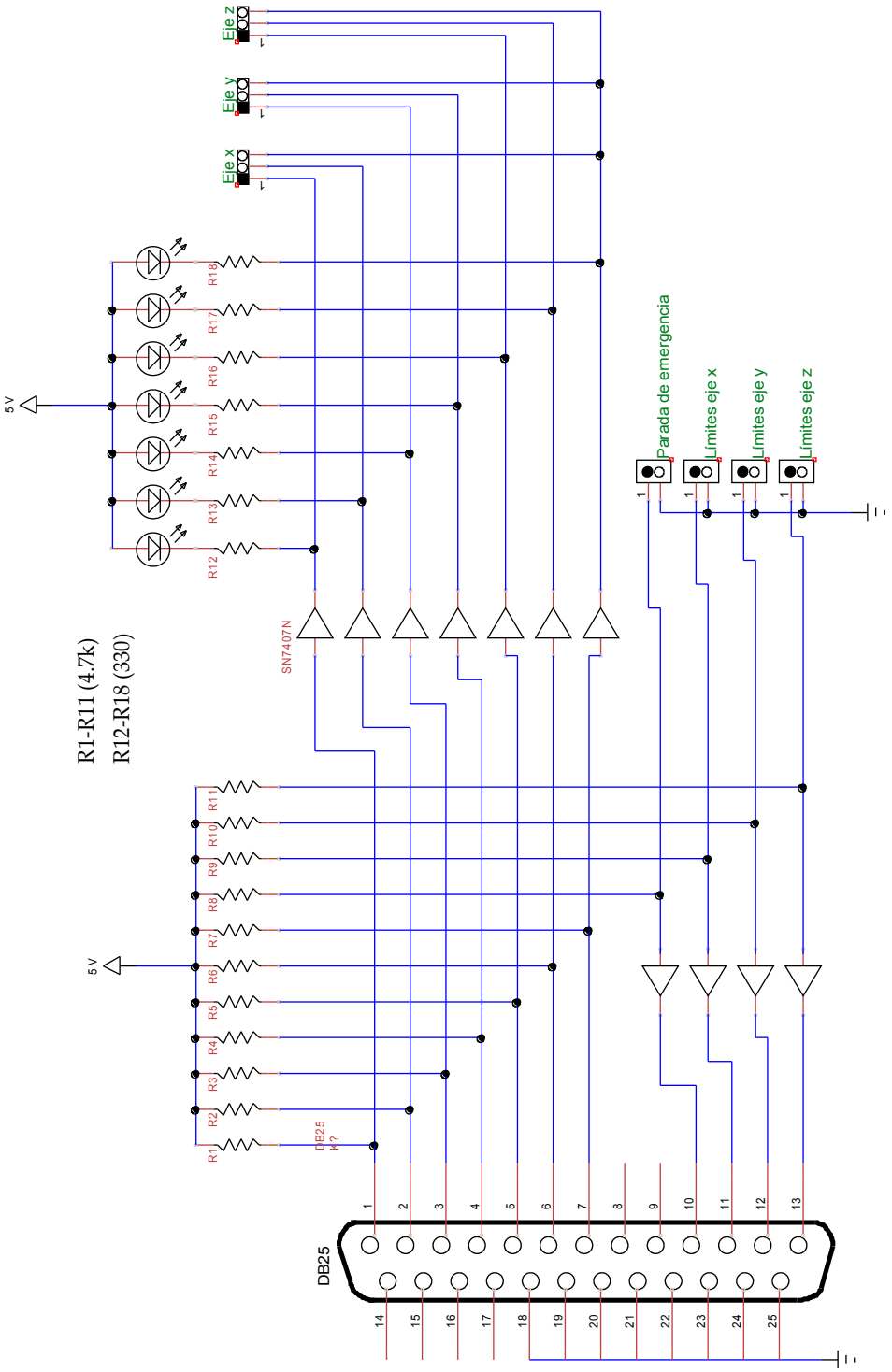


Figura C.1: Diagrama de la tarjeta paralelo

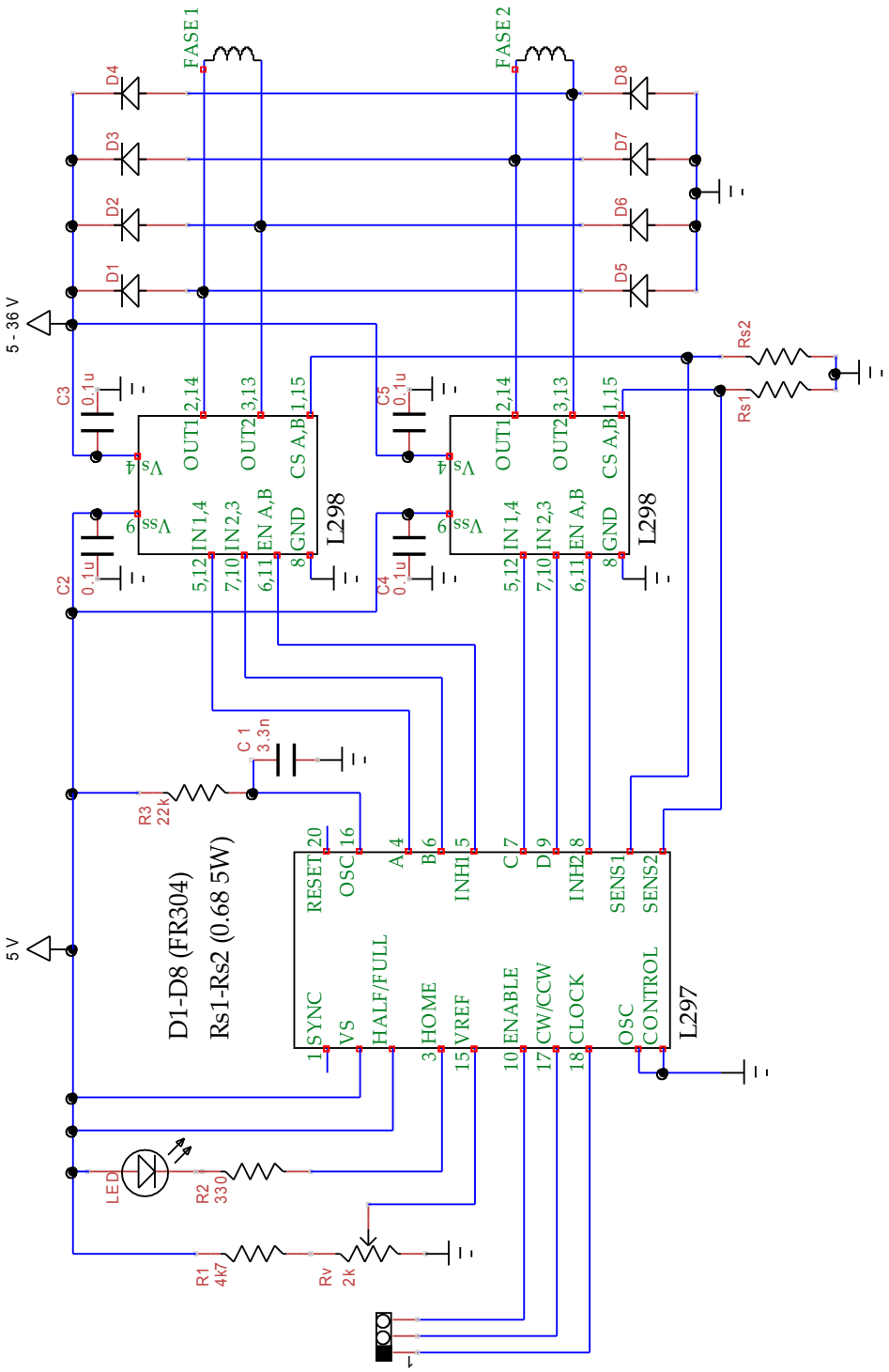


Figura C.2: Diagrama de la tarjeta con L297 y L298

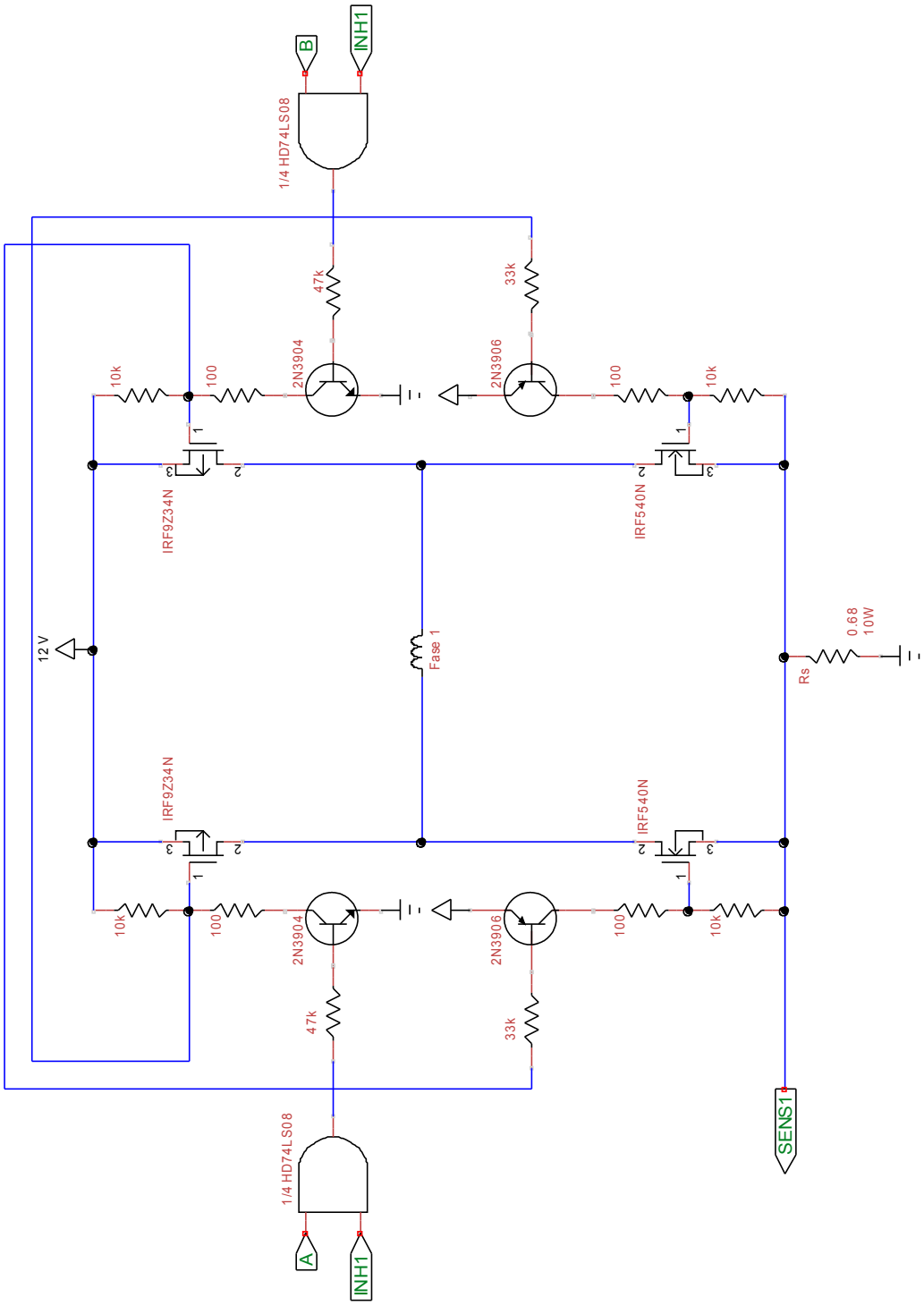


Figura C.3: Diagrama de puente H discreto

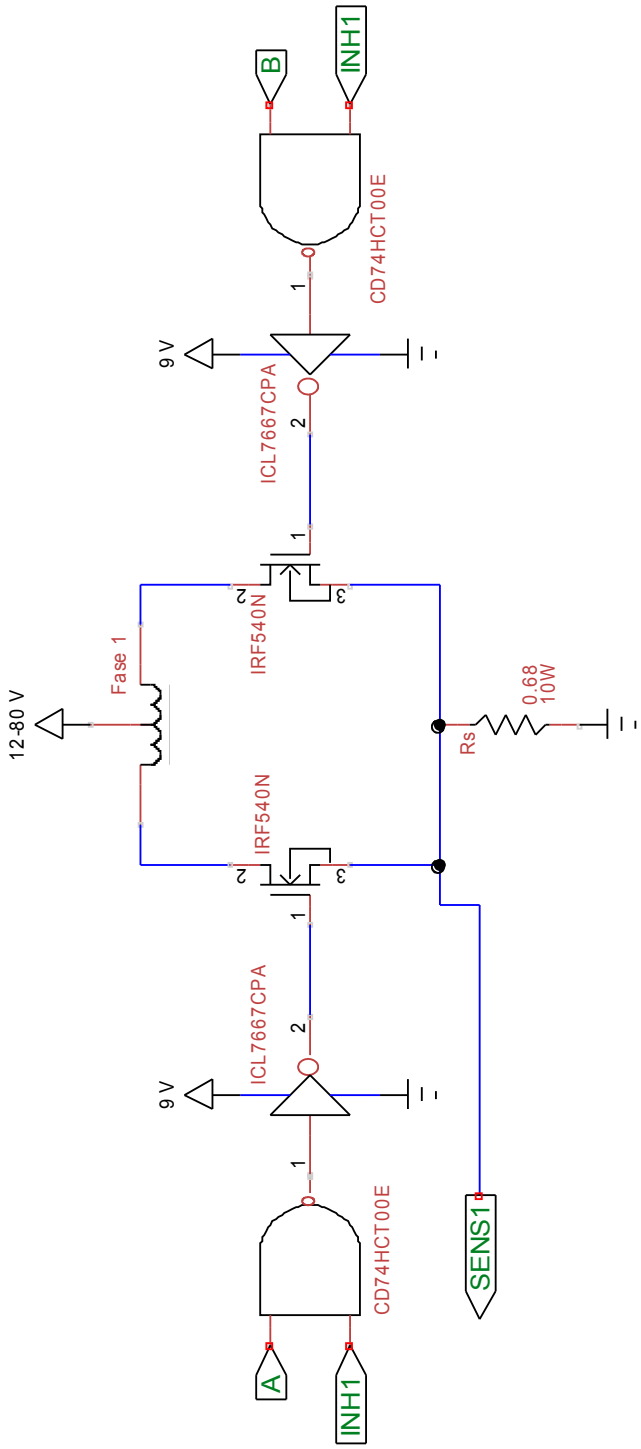


Figura C.4: Diagrama de control unipolar

BIBLIOGRAFÍA

- Micro Motion Solutions, 2011. *Microstepping: Myths and Realities* [en línea]
  Disponible en: http://www.micromo.com/microstepping-myths-and-realities.aspx
  [accesado 11-mayo-2011]
- Oriental Motor, 2010. *Basics of Step Motors* [en línea]
  Disponible en : http://www.orientalmotor.com/technology/articles/step-motor-basics.html
  [accesado 11-mayo-2011]
- Oriental Motor, 2010. *Stepping Motor Overview* [en línea]
  Disponible en: http://www.orientalmotor.com/technology/articles/stepping-motor-overview.html
  [accesado 11-mayo-2011]
- Smid, P., 2003. *CNC Programming Handbook*. 2ª edición.
  EUA: Industrial Press Inc
- STMicroelectronics, 2001. *L297 Stepper Motor Controllers*
  Disponible en: http://www.datasheetcatalog.com/datasheets_pdf/L/2/9/7/L297.shtml
  [accesado 11-mayo-2011]
- STMicroelectronics, 2000. *L298 Dual Full-bridge Driver*
  Disponible en: http://www.datasheetcatalog.com/datasheets_pdf/L/2/9/8/L298.shtml
  [accesado 11-mayo-2011]
- Williams, G., 2003. *CNC Robotics, Build your own workshop bot*. 1ª edición
  EUA: McGraw-Hill
- Yohudi, 2010. The FREE Creative CNC Solution [ebook]
  Disponible en: http://cnc4free.org
  [accesado 11-mayo-2011]