Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería Mecánica Eléctrica

DISEÑO E IMPLEMENTACIÓN DE UNA MÁQUINA FRESADORA CNC PARA LA FABRICACIÓN DE PLACAS DE CIRCUITOS IMPRESOS

Rodrigo Rafael Chang Papa
Asesorado por el Ing. Byron Odilio Arrivillaga Méndez

Guatemala, julio de 2015

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

DISEÑO E IMPLEMENTACIÓN DE UNA MÁQUINA FRESADORA CNC PARA LA FABRICACIÓN DE PLACAS DE CIRCUITOS IMPRESOS

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

RODRIGO RAFAEL CHANG PAPA
ASESORADO POR EL ING. BYRON ODILIO ARRIVILLAGA MÉNDEZ

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN ELECTRÓNICA

GUATEMALA, JULIO DE 2015

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Pedro Antonio Aguilar Polanco
VOCAL I	Ing. Angel Roberto Sic García
VOCAL II	Ing. Pablo Christian de León Rodríguez
VOCAL III	Inga. Elvia Miriam Ruballos Samayoa
VOCAL IV	Br. Narda Lucía Pacay Barrientos
VOCAL V	Br. Walter Rafael Véliz Muñoz
SECRETARIA	Inga. Lesbia Magalí Herrera López

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

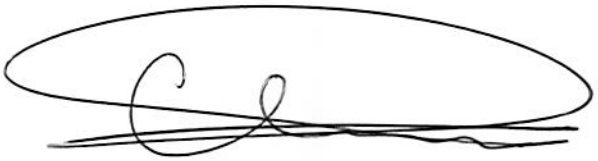
DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADOR	Ing. Carlos Eduardo Guzmán Salazar
EXAMINADOR	Ing. Otto Fernando Andrino González
EXAMINADOR	Ing. Byron Odilio Arrivillaga Méndez
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

DISEÑO E IMPLEMENTACIÓN DE UNA MÁQUINA FRESADORA CNC PARA LA FABRICACIÓN DE PLACAS DE CIRCUITOS IMPRESOS

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería Mecánica Eléctrica, con fecha agosto de 2014.



Rodrigo Rafael Chang Papa

Guatemala, 14 de mayo de 2015

Ingeniero Carlos Guzmán
Coordinador de Área de Electrónica
Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería

Señor Coordinador:

Por este medio tengo el gusto de informar a usted que he concluido con el asesoramiento del trabajo de graduación con título: "**DISEÑO E IMPLEMENTACIÓN DE UNA MÁQUINA FRESADORA CNC PARA LA FABRICACIÓN DE PLACAS DE CIRCUITOS IMPRESOS**", desarrollado por el estudiante **Rodrigo Rafael Chang Papa**, con carné **201020239**. Después de revisar su contenido final, considero que cumple con los requerimientos necesarios, y doy mi entera aprobación al mismo.

Atentamente,

Ing. Byron Arrivillaga Méndez
Col. 5217

Ing. Byron Odilio Arrivillaga Méndez
Colegiado No. 5217

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

Ref. EIME 28. 2015
Guatemala, 20 de MAYO 2015.

Señor Director
Ing. Guillermo Antonio Puente Romero
Escuela de Ingenierìa Mecànica Elèctrica
Facultad de Ingenierìa, USAC.

Señor Director:

Me permito dar aprobación al trabajo de Graduación titulado: DISEÑO E IMPLEMENTACIÓN DE UNA MÁQUINA FRESADORA CNC PARA LA FABRICACIÓN DE PLACAS DE CIRCUITOS IMPRESOS, del estudiante Rodrigo Rafael Chang Papa, que cumple con los requisitos establecidos para tal fin.

Sin otro particular, aprovecho la oportunidad para saludarle.

Atentamente,
ID Y ENSEÑAD A TODOS

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION ESCUELA DE INGENIERIA MECANICA ELECTRICA
FACULTAD DE INGENIERIA

Ing. Carlos Eduardo Guzmán Salazar
Coordinador Área Electrónica

sro

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

REF. EIME 28. 2015.

El Director de la Escuela de Ingeniería Mecánica Eléctrica, después de conocer el dictamen del Asesor, con el Visto Bueno del Coordinador de Área, al trabajo de Graduación del estudiante; RODRIGO RAFAEL CHANG PAPA titulado: DISEÑO E IMPLEMENTACIÓN DE UNA MÁQUINA FRESADORA CNC PARA LA FABRICACIÓN DE PLACAS DE CIRCUITOS IMPRESOS, procede a la autorización del mismo.

Ing. Guillermo Antonio Puente Romero

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION ESCUELA DE INGENIERIA MECANICA ELECTRICA
FACULTAD DE INGENIERIA

GUATEMALA, 15 DE JUNIO 2,015.

Universidad de San Carlos
de Guatemala

Facultad de Ingeniería
Decanato

DTG. 310.2015

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería Mecánica Eléctrica, al Trabajo de Graduación titulado: **DISEÑO E IMPLEMENTACIÓN DE UNA MÁQUINA FRESADORA CNC PARA LA FABRICACIÓN DE PLACAS DE CIRCUITOS IMPRESOS,** presentado por el estudiante universitario: **Rodrigo Rafael Chang Papa,** y después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, autoriza la impresión del mismo.

IMPRÍMASE:

Ing. Pedro Antonio Aguilar Polanco
Decano

Guatemala, 7 de julio de 2015

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERIA

/gdech

ACTO QUE DEDICO A:

Mis padres	Miguel Chang y Silvana Papa, por ser mis guías, ejemplo a seguir día a día, y por estar en todo momento junto a mí, brindándome su apoyo y amor.
Mi hermano	Carlos Chang, por apoyarme durante mi carrera, y por su cariño durante toda mi vida.
Mis abuelos	Rafael Chang y Yolanda Pappa, por compartir conmigo y transmitirme su sabiduría durante toda mi vida.

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala	Por ser la casa de estudios que me permitió formarme como profesional.
Facultad de Ingeniería	Por ser el lugar donde pude expandir mis conocimientos y alcanzar muchos objetivos personales.
Laboratorio de Electrotecnia	Por haberme brindado trabajo y experiencias que complementaron mi formación profesional.
Laboratorio de Electrónica	Por haberme brindado trabajo, amistades, y muchas experiencias que me formaron durante la carrera.
Ing. Byron Arrivillaga	Por asesorarme en este trabajo de graduación y su apoyo como tutor de laboratorio.
Esther Pineda	Por su apoyo en la realización de este trabajo de graduación.
Daniel Oxom	Por su apoyo en la realización del proyecto.
Amigos y compañeros de proyectos	Con quienes pasábamos días y noches enteras trabajando, por compartir conmigo muchas experiencias.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
A	Amperios
cm	Centímetros
GB	Gigabyte
ºC	Grados Celsius
º	Grados sexagesimales
kHz	Kilohertz
MB	Megabyte
mA	Miliamperios
mm/min	Milímetros por minuto
ns	Nanosegundos
in	Pulgadas
rev	Revoluciones
VDC	Voltios de corriente directa
VAC	Voltios eficaces
V	Voltios nominales
W	Watts

GLOSARIO

Anti-backlash	Mecanismo que permite reducir el *backlash*.
Arduino	Plataforma de hardware libre que utiliza un microcontrolador y un entorno de programación para el desarrollo de proyectos de electrónica.
Arduino *shield*	PCB que se conecta encima de la placa de Arduino para proveer funciones específicas.
Backlash	También llamado juego mecánico, es una holgura o pérdida de movimiento en un mecanismo causado por espaciamiento entre las partes.
Búfer	Compuerta lógica cuya salida es igual a la entrada.
C	Lenguaje de programación.
CAD	Software para diseño asistido por computadora (*Computer-aided design*, en inglés).
CAM	Software para fabricación asistida por computadora (*Computer-aided design*, en inglés).

Chopper	Tipo de control que utiliza conmutación a alta velocidad para mantener un voltaje y corriente promedio en un componente electrónico.
CI	Circuito integrado. Componente electrónico de pequeñas dimensiones que provee funciones específicas.
Conector IDC-10	Conector para placas de circuitos impresos con 10 terminales eléctricas en forma de pines distribuidos en 2 filas de 5 pines cada una.
Control de lazo abierto	Sistema de control que no utiliza realimentación de la señal de salida para controlarla.
Control de lazo cerrado	Sistema de control en el cual, para controlar la salida, se compara la señal de salida con otra de referencia.
DRC	Comprobación de reglas de diseño (Design Rule Check, en inglés). Herramienta de EAGLE para revisión de errores en el diseño del PCB.
EAGLE	*Software* para diseño de diagramas esquemáticos y PCB.
EMC2	Enhanced Machine Controller, en inglés. *Software* para control de máquinas herramienta para el sistema operativo Linux.

ERC	Comprobación de reglas eléctricas (Electrical Rule Check, en inglés). Herramienta de EAGLE para revisión de errores en el diagrama esquemático.
Flanco de bajada	Transición del nivel lógico alto al nivel lógico bajo.
Flanco de subida	Transición del nivel lógico bajo al nivel lógico alto.
Flip flop	Elemento de lógica digital de memoria capaz de almacenar un bit de información.
Fresadora	Máquina que utiliza una herramienta de corte para remover material de la superficie de una pieza de trabajo.
Fresadora CNC	Fresadora que utiliza control numérico computarizado.
GNU GPL	Licencia pública general del proyecto GNU. Utilizada para proyectos de *software* libre.
GRBL	*Software* intérprete de programas de control numérico computarizado para Arduino.
Hardware	Conjunto de componentes eléctricos, electrónicos y mecánicos de un sistema informático.
***Hardware* abierto**	Análogo a *software* libre para dispositivos electrónicos.

Interfaz de control	Circuito electrónico que sirve como intermediario entre la computadora y el resto de equipo electrónico.
Jumper	También llamado puente, permite interconectar dos terminales eléctricas de forma temporal.
LED	Diodo emisor de luz (*light emitting diode*, en inglés). Componente electrónico que emite luz y se utiliza como indicador en muchos dispositivos electrónicos.
Licencia *freeware*	*Software* distribuido sin costo.
Máquina herramienta	Máquina controlada a través de control numérico para mecanizar una pieza.
Motor paso a paso	Dispositivo electromecánico que convierte señales eléctricas en desplazamientos angulares discretos para controlar mecanismos con precisión.
Multiparadigma	Lenguaje de programación que soporta más de un paradigma o estilo de programación.
Pads	Porción de metal expuesto sobre la superficie de un PCB donde se suelda un componente.
PCB	Placa de circuito impreso (*printed circuit board*, en inglés).

Plywood	Hoja de material elaborado con finas chapas de madera pegadas una sobre la otra con resinas sintéticas mediante fuerte presión y calor.
Puente H	Circuito que sirve para cambiar la polaridad de voltaje aplicado a un componente de dos terminales.
PuTTY	Programa cliente emulador de terminal y consola serial.
PWM	Modulación por ancho de pulsos.
Python	Lenguaje de programación.
Resistencia *pull-down*	Resistencia conectada entre un pin de entrada digital y la referencia del voltaje de alimentación de un circuito.
Resistencia *pull-up*	Resistencia conectada entre un pin de entrada digital y el voltaje de alimentación de un circuito.
Sistema de tiempo real	Sistema que interactúa y responde a estímulos dentro de un plazo de tiempo determinado.
Software	Conjunto de programas que conforman la parte lógica de un sistema informático.
***Software* libre**	*Software* que puede ser utilizado, estudiado, modificado y redistribuido libremente.

Tabla de sacrificio	Área de trabajo de la máquina herramienta, donde se fijan las piezas a mecanizar.
Tap	Derivación central de un embobinado.
Tornillos M3	Tornillos milimétricos de 3 mm de diámetro.
Torque de retención	Torque requerido para girar el eje de un motor paso a paso cuando se encuentra energizado.
Torque dinámico	Los distintos tipos de torque generados en el eje de un motor paso a paso girando a distintas velocidades.
Transistor Darlington	Dispositivo semiconductor que combina dos transistores bipolares en un tándem para proporcionar una alta ganancia de corriente.
Tren de pulsos	Señal periódica conformada por pulsos digitales.
ULPs	User Language Program, en inglés. Programas definidos por el usuario para añadir funciones específicas a EAGLE.
Vías	Agujero en un PCB usado para pasar una señal de una capa a otra.
Wave-drive	Secuencia de un motor paso a paso en la que una bobina del motor es energizada a la vez.

RESUMEN

En el presente trabajo de graduación se describe el diseño y la construcción de una máquina fresadora y el equipo electrónico necesario para controlarla. Dicha máquina utiliza control numérico computarizado para fabricar placas de circuitos impresos.

En el primer capítulo se describen los conceptos fundamentales sobre control numérico computarizado y la operación de máquinas que lo utilizan para llevar a cabo distintas tareas de mecanización. Seguidamente, se describe la operación de fresado de placas de circuitos impresos, y se desarrolla la teoría de operación de motores paso a paso utilizados en este tipo de máquinas.

El segundo capítulo muestra el proceso de construcción de la máquina fresadora, listando los materiales necesarios y describiendo los mecanismos empleados en el movimiento de la máquina, así como las herramientas necesarias para llevar a cabo el fresado.

En el tercer capítulo se desarrolla el diseño de todo el equipo electrónico utilizado para controlar la máquina fresadora, explicando el funcionamiento de los circuitos electrónicos utilizados, y la interfaz de conexión con la computadora.

En el cuarto capítulo se describe la utilización y configuración de distintos programas en la computadora para llevar a cabo el diseño y fabricación de las placas de circuitos impresos.

OBJETIVOS

General

Diseñar e implementar una máquina fresadora CNC para la fabricación de placas de circuitos impresos.

Específicos

1. Presentar los conceptos fundamentales de control numérico computarizado, la operación de fresado de circuitos impresos y la teoría de operación de motores paso a paso.

2. Describir el proceso de construcción de una máquina fresadora de escritorio, mostrando el diseño de sus piezas y las herramientas de fresado.

3. Diseñar e implementar el equipo electrónico necesario para controlar la máquina fresadora, describiendo el funcionamiento de los circuitos y componentes electrónicos.

4. Describir el software utilizado para el diseño de placas de circuitos impresos y para el control de la máquina fresadora.

5. Proponer la utilización de dos programas de optimización de código G desarrollados para mejorar el proceso de fresado de circuitos impresos en máquinas fresadoras CNC.

INTRODUCCIÓN

Actualmente, todos los dispositivos electrónicos comerciales están compuestos por una placa de circuito impreso, que mantiene a todos los componentes electrónicos unidos e interconectados eléctricamente para realizar una tarea específica. En la fabricación de prototipos de circuitos electrónicos se puede utilizar un método de fabricación casera, utilizando placas de cobre virgen, una plancha de ropa, impresiones de tóner y una solución de cloruro férrico para eliminar el exceso de cobre de las placas vírgenes.

La desventaja de este proceso de fabricación es que es muy trabajoso, ya que todo el proceso es realizado a mano; además, los resultados dependen de la experiencia de la persona, y el desempeño varía a medida que se realizan varias placas de circuitos impresos. Finalmente este proceso requiere de forma inevitable la utilización de químicos, lo que hace más costoso el proceso de fabricación, y cuando no son manipulados correctamente son perjudiciales para la salud y causan daños al medio ambiente.

Por esto se presenta una solución al proceso de fabricación de placas de circuitos impresos, con la cual se pretende facilitar el proceso y obtener mejores resultados, mediante la construcción y puesta en marcha de una máquina fresadora que permita pasar directamente de la etapa del diseño del circuito impreso a la etapa de fabricación, utilizando solamente la placa de cobre virgen y la máquina con sus respectivas herramientas, ahorrando tiempo en el prototipado de dispositivos electrónicos y sin la manipulación de químicos nocivos.

1. FUNDAMENTOS DEL PROYECTO

Con el fin de establecer una base teórica sobre el proyecto, es importante describir primero los fundamentos sobre la operación de máquinas a través de control numérico computarizado y describir la teoría de operación de los motores paso a paso, ampliamente utilizados en este tipo de máquinas.

1.1. Control numérico computarizado (CNC)

El control numérico puede ser definido como las operaciones realizadas por máquinas herramienta a través de instrucciones específicamente codificadas al sistema de control de la máquina. Las instrucciones son combinaciones de letras del alfabeto, dígitos y símbolos seleccionados. Todas las instrucciones están escritas en un orden lógico y un formato predeterminado.

La colección de todas las instrucciones necesarias para trabajar una pieza se conoce como programa de control numérico o programa de control numérico computarizado. Dicho programa puede ser almacenado para ser usado en el futuro y obtener resultados idénticos sobre otra pieza en cualquier momento.

Un sistema de control numérico es diferente a un sistema de control numérico computarizado si se es muy estricto en la terminología. Ambos sistemas procesan instrucciones codificadas para trabajar una pieza, pero la diferencia principal radica en que un sistema de control numérico puro utiliza funciones lógicas fijas, que están incluidas de forma permanente en una unidad

de control, y no pueden ser cambiadas por el operador de la máquina. En cambio, un sistema computarizado utiliza un microprocesador y registros de memoria que almacenan una variedad de rutinas capaces de manipular las funciones lógicas, permitiendo al operador cambiar el programa en el control mismo con resultados instantáneos**.**

En la práctica se utiliza el término control numérico computarizado (CNC, por sus siglas en inglés) para referirse a las máquinas herramientas capaces de ejecutarlo.

1.2. Aplicaciones del control numérico computarizado

Los sistemas de control numérico computarizado son utilizados para máquinas que realizan una secuencia de movimientos y operaciones. Entre las aplicaciones más importantes se encuentran:

- Corte de piezas con láser: se utiliza una máquina con control numérico computarizado para manejar un láser de alta potencia, capaz de cortar madera y otros materiales con el fin de obtener piezas que de otra forma sería muy complicado realizar.

- Máquinas cortadoras de plasma: al igual que el proceso de corte con láser, se utiliza una cortadora de plasma para cortar planchas de metal muy gruesas y con alta precisión.

- Soldadoras: estas máquinas se utilizan para realizar con precisión el trabajo de soldadura de una serie de piezas.

- Tornos: los tornos que utilizan control numérico computarizado permiten desgastar con alta precisión una variedad de materiales mientras estos giran a alta velocidad.

- Troqueladoras: utilizan un cabezal de activación mecánica o hidráulica y consisten en una mesa amplia donde se coloca la pieza a mecanizar. La mesa se desplaza a lo largo y lo ancho a gran velocidad, produciendo piezas con rapidez y exactitud.

- Fresadoras: son máquinas que utilizan control numérico computarizado para desgastar o perforar madera y metales, utilizando una herramienta rotativa y una variedad de fresas y brocas.

1.3. Principio de funcionamiento

Para mecanizar una pieza se utiliza un sistema de coordenadas que especifican el movimiento de la herramienta de corte. El sistema se basa en el control de los movimientos de la herramienta de trabajo en relación con los ejes de coordenadas de la máquina, usando un programa ejecutado por computadora.

El programa que realiza los movimientos y controla la máquina utiliza un lenguaje especial llamado código G, que designa códigos y parámetros a la forma en que se realiza cada movimiento. De esta forma, es posible especificar una secuencia de comandos que debe ejecutar la máquina que implemente el control numérico computarizado.

Las fresadoras consisten generalmente en un área de trabajo en la cual se mueve verticalmente una herramienta rotatoria. En los modelos más modernos

de máquinas, la posición de la herramienta es controlada por motores de pasos o servomotores, que proveen movimientos altamente precisos.

El control de los motores es generalmente de lazo abierto, que funciona mientras las fuerzas se mantengan lo suficientemente pequeñas y las velocidades no sean muy altas. Sin embargo, en las máquinas comerciales para trabajo en metales, los sistemas de lazo cerrado son un estándar y son requeridos para alcanzar la precisión, velocidad y repetitividad necesaria.

1.4. Programación del control numérico

En el momento de evaluar los dibujos, materiales y equipo necesario para llevar a cabo un trabajo en una máquina, la complejidad de la tarea de programación se vuelve más clara. La realización de programas para máquinas CNC se puede realizar de dos formas: manualmente, o de forma automática.

Los trabajos de programación simples pueden ser asignados al operador de la máquina, ya que es una buena manera de ganar experiencia. En el caso de trabajos difíciles o complejos, se utilizan tecnologías de diseño asistido por computadora (CAD, por sus siglas en inglés) y tecnologías de fabricación asistida por computadora (CAM, por sus siglas en inglés), que han sido una parte esencial de los procesos de manufactura por muchos años.

1.4.1. Código G

Es un lenguaje de comandos de máquina basado en caracteres alfanuméricos que un controlador interpreta en movimientos discretos y en modos de trabajo. En términos generales, el lenguaje permite decirle a las

máquinas herramienta qué hacer y cómo hacerlo. No es el tipo de lenguaje de programación que requiere que se compile el programa para utilizarse.

El código G es considerado el estándar de la industria como lenguaje para control de máquinas herramienta y su sintaxis se adhiere al estándar RS-274. Hay un número finito de comandos base que son intrínsecos al lenguaje y dependiendo del proveedor del software de control puede haber distintos códigos o parámetros soportados para su implementación.

1.4.2. Programación manual

La programación manual ha sido el método más común para preparar un programa por muchos años. Consiste en escribir el programa para la máquina utilizando únicamente razonamientos y cálculos. Para mecanizados muy simples este proceso es el más rápido y confiable. Además, los programas pequeños pueden ser enviados directamente en la interfaz de control de la máquina.

La mayor desventaja de la programación manual es que se requiere mucho tiempo para desarrollar un programa de CNC totalmente funcional, además de que es muy fácil cometer errores en la programación y muy difícil hacer cambios en el programa. Sin embargo, la programación manual ofrece virtualmente libertad ilimitada en el desarrollo de la estructura de un programa, y obliga al programador a entender las técnicas de programación hasta el último detalle; es por ello que técnicas manuales son directamente aplicadas en los diseños asistidos por computadora. Contrario a lo que se piensa, se necesita un conocimiento profundo de los métodos de programación manual para un manejo eficiente de los programas asistidos por computadora.

1.4.3. Programación automática

La programación automática se realiza por medio de programas de computadora, que ayudan en el proceso de diseño de piezas a través de programas de CAD, y en el proceso de manufactura a través de programas de CAM, donde la programación de CNC es una pequeña parte. Todo el conjunto de programas de CAD/CAM cubre mucho más que solo el diseño y programación, forma parte de las tecnologías modernas conocidas como CIM (manufacturación integrada por computadora, por sus siglas en inglés).

El proceso de programación automática permite realizar diseños de mecanizado en programas de CAD y ajustar opciones relacionadas con la máquina herramienta, tales como profundidad de corte, rapidez de los movimientos y escala del trabajo; para producir un programa de la pieza o trabajo a mecanizar para ser cargado directamente en la máquina CNC.

1.5. Placa de circuito impreso

También conocida como PCB, por sus siglas en inglés (*printed circuit board*). Un circuito impreso soporta mecánicamente componentes electrónicos mientras los conecta eléctricamente utilizando pistas o caminos, *pads* y otras características grabadas a partir de una capa de cobre sobre una placa aislante de electricidad. La placa es generalmente fabricada de fibra de vidrio, plástico, teflón o baquelita. Los circuitos impresos pueden ser de una, dos, o múltiples capas de cobre de la placa.

Las placas de circuitos impresos son utilizadas en todo tipo de productos electrónicos. Como alternativa a su utilización, es posible usar placas perforadas con caminos predefinidos. El diseño de circuitos impresos requiere

mayor esfuerzo para seleccionar la disposición de los componentes, pero la manufactura es más rápida y económica que otros métodos de fabricación; además se eliminan los errores de cableado del operario que ensambla el circuito.

1.5.1. Fresado de circuitos impresos

Para fabricar PCBs es posible utilizar un método de grabado utilizando herramientas de corte con una máquina fresadora. Los circuitos impresos grabados con máquinas fresadoras pueden alcanzar un alto grado de calidad y servir como prototipos de rápido desarrollo para equipos electrónicos.

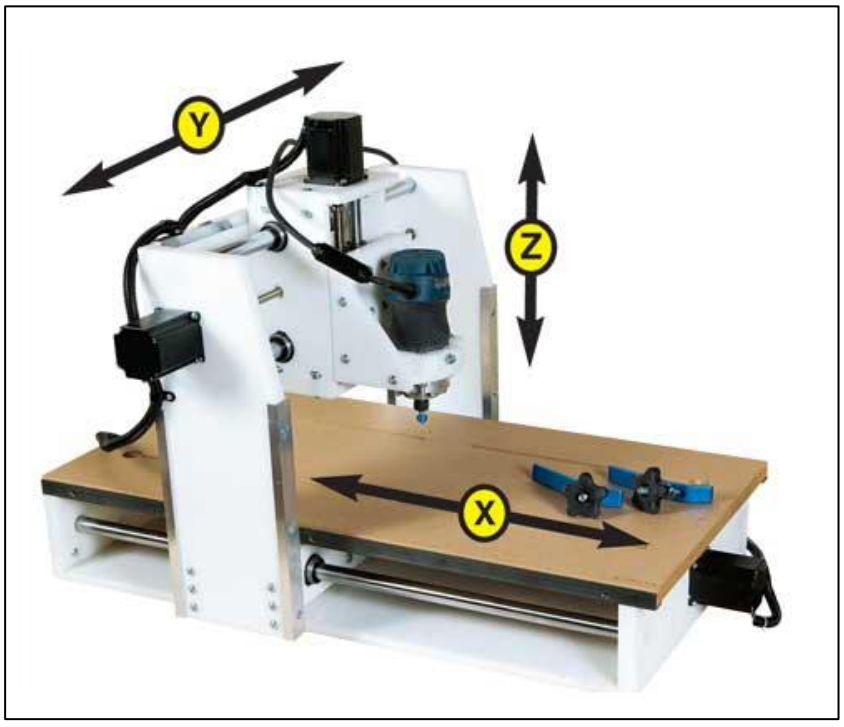
Las características de los PCBs fresados son más limitadas que las de circuitos impresos que utilizan métodos de fabricación industrial. Sin embargo, la ventaja de fabricar circuitos impresos utilizando una fresadora es que se elimina la necesidad de manejar químicos para remover el cobre excedente de las placas de cobre.

1.6. Máquina fresadora CNC

Una fresadora CNC es una máquina compuesta por mecanismos de movimiento en al menos tres ejes: “X”, “Y” y “Z”. Está compuesta por un cabezal dotado de un mecanismo de rotación, con una fresa (herramienta cortante) u otro tipo de herramienta, y de una mesa o área de trabajo, donde se fija la pieza. La herramienta de corte se puede mover para arriba y para abajo (para adentro y para afuera), mientras sigue físicamente una trayectoria dentro del área de trabajo.

Las máquinas fresadoras CNC son usualmente pequeñas, simples máquinas, sin cambiador de herramientas u otras características automáticas. Su rango de consumo de potencia es bajo. En la industria son utilizadas para producción de partes pequeñas o con propósitos de mantenimiento. Son diseñadas generalmente para contorneados, a diferencia de los taladros CNC. En la figura 1 se muestra una máquina fresadora CNC de tamaño convencional para uso en pequeños talleres y laboratorios.

Figura 1. **Máquina fresadora CNC**



Fuente: JOHNSTONE, Rob. *What is a CNC router?* http://www.woodworkersjournal.com/cnc-router/. Consulta: 30 de octubre de 2014.

1.7. Teoría de operación de motores paso a paso

Los motores paso a paso son dispositivos electromecánicos que convierten señales eléctricas en desplazamientos angulares discretos, con lo que permiten avanzar una determinada cantidad de grados (pasos) dependiendo de las entradas de control. La secuencia de los pulsos aplicados al motor está directamente relacionada con la dirección de rotación del eje del motor, y la frecuencia de los pulsos de entrada está directamente relacionada con la velocidad de rotación del eje del motor.

Este motor presenta alta precisión y repetitividad en cuanto al posicionamiento, los mejores motores tienen una precisión angular del tres al cinco por ciento (3 % - 5 %) del ángulo de un paso, y este error no es acumulativo de un paso al siguiente. Además, son muy confiables, ya que no hay escobillas de contacto dentro del motor, por lo que la vida útil está limitada por la vida del cojinete del eje. Es por esto que la mayor parte de las máquinas herramienta de aficionados y gama media disponibles en el mercado utilizan motores paso a paso.

Una de las ventajas más importantes de los motores paso a paso es su habilidad de ser controlados precisamente en un sistema de lazo abierto, es decir, un sistema en el que no se necesita información acerca de la posición. Este tipo de control elimina la necesidad de dispositivos sensores tales como codificadores ópticos, en cambio, la posición se conoce simplemente llevando la cuenta de los pulsos de entrada al motor.

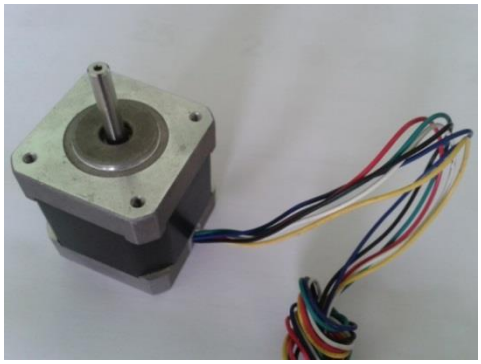
Generalmente los motores paso a paso tienen dos fases, pero también existen motores de tres y cinco fases. Un motor bipolar con dos fases tiene un embobinado por fase, mientras un motor unipolar tiene un embobinado con un

tap central por fase. El motor unipolar también es conocido como un motor de cuatro fases; sin embargo, posee solamente dos fases. También existen motores con embobinados separados por fase, los cuales pueden ser utilizados como motores bipolares o unipolares.

Los motores paso a paso pueden encontrarse utilizando una nomenclatura estándar para referirse al tamaño de la cara frontal, utilizando décimas de pulgada para referirse al tamaño. El estándar es conocido como NEMA ICS 16-2001, que es desarrollado por la Asociación Nacional de Fabricantes Eléctricos (NEMA, por sus siglas en inglés) con sede en Estados Unidos, que crea estándares desarrollados por fabricantes a través de un consenso voluntario para proveer descripciones de cómo deberían ser los productos eléctricos.

En la figura 2 se puede observar un motor paso a paso NEMA 17, es decir, que la cara frontal del motor mide 1.7 pulgadas de largo. Cuando se utiliza esta nomenclatura se sabe inmediatamente que otro motor NEMA 17 cabe en la superficie de montaje del motor sin rediseñar nada, haciendo posible cambiar los componentes más fácilmente.

Figura 2. **Motor paso a paso NEMA 17 de 6 cables**

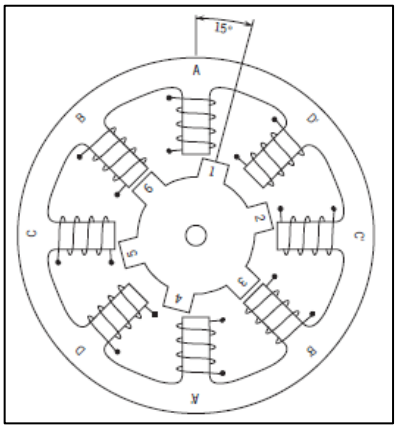


Fuente: elaboración propia.

1.7.1. Tipos de motores paso a paso

Existen tres tipos básicos de motores paso a paso: los motores de reluctancia variable, los de imán permanente y los híbridos. Los motores de reluctancia variable consisten en un rotor con múltiples dientes de hierro dulce y un estator con bobinas. Cuando las bobinas del estator se energizan con corriente, los polos se magnetizan y la rotación ocurre cuando los dientes del rotor son atraídos hacia los polos energizados del estator. En la figura 3 se muestra la sección transversal de un motor paso a paso de reluctancia variable.

Figura 3. **Motor paso a paso de reluctancia variable**

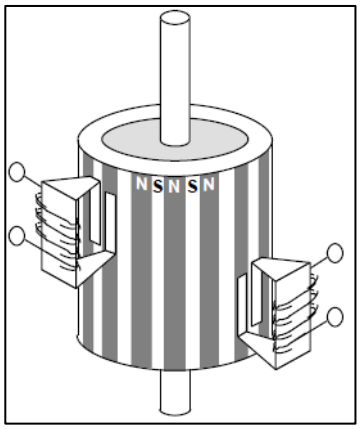


Fuente: *Stepper motor basics*.
http://users.ece.utexas.edu/~valvano/Datasheets/StepperBasic.pdf.
Consulta: 14 de noviembre de 2014.

El motor paso a paso de imán permanente es un motor de baja resolución y bajo costo con ángulos de paso típicos entre 7.5º a 15º por paso. Como su nombre implica, tienen un imán permanente en la estructura del motor. El rotor no posee dientes como el motor de reluctancia variable, en cambio el rotor está magnetizado con polos norte y sur, situados en una línea paralela al eje del

motor. Estos polos magnetizados del rotor proveen una intensidad de flujo magnético y es por esto que los motores de imán permanente exhiben características mejoradas de torque cuando se comparan con los de reluctancia variable. En la figura 4 se puede observar el principio de funcionamiento de un motor paso a paso de imán permanente.

Figura 4. **Motor de imán permanente**



Fuente: *Stepper motor basics*.
http://users.ece.utexas.edu/~valvano/Datasheets/StepperBasic.pdf.
Consulta: 14 de noviembre de 2014.

Los motores paso a paso híbridos son más costosos que los de imán permanente, pero proveen un mayor desempeño en resolución de pasos, torque y velocidad. Los ángulos de paso típicos para el motor híbrido varían de 0,9º a 3,6º por paso. Estos motores combinan las mejores características de los motores de imán permanente y los de reluctancia variable. El rotor tiene múltiples dientes y contiene imanes permanentes alrededor del eje. Los dientes en el rotor proveen una mejor trayectoria guía para el flujo magnético, lo que incrementa el torque de retención y el torque dinámico.

1.7.2. Funcionamiento de motores paso a paso

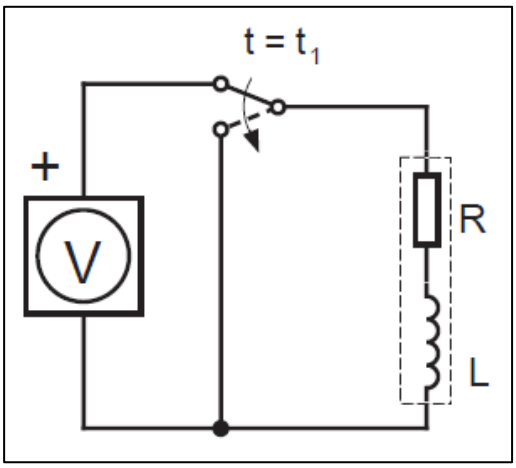
Los motores paso a paso están compuestos por bobinas, hechas a su vez de varias vueltas de cable de cobre. El cable es enrollado en una pieza plástica, lo que permite que se pueda manufacturar aparte el embobinado, el estator y otras partes mecánicas. En la etapa final de producción se montan las bobinas alrededor de los polos del estator.

Las bobinas presentan dos características físicas inherentes: la resistencia y la inductancia, que limitan el desempeño de cualquier motor paso a paso. La resistencia de las bobinas es responsable de la mayor parte de la pérdida de potencia y del calentamiento del motor.

El tamaño y las características térmicas del embobinado y del cuerpo del motor limitan la máxima disipación de potencia permitida en el embobinado, y para que un motor sea eficiente debe ser utilizado en el punto de mayor disipación de potencia, de lo contrario, el motor puede ser reemplazado con un motor de menor tamaño, lo que conlleva un costo menor en el diseño de un sistema.

La inductancia del motor hace que la bobina se oponga a los cambios de corriente, y por lo tanto, limita la velocidad de operación del motor. En la figura 5 se puede observar el circuito equivalente de una fase del motor paso a paso. Cuando se aplica un voltaje a la bobina del motor, el circuito equivalente del embobinado es una resistencia con un inductor en serie. Este circuito es conocido como circuito RL y posee un comportamiento exponencial en la corriente del circuito y en el voltaje de los componentes.

Figura 5. **Circuito equivalente de una fase del motor paso a paso**



Fuente: *Drive circuit basics*.
http://users.ece.utexas.edu/~valvano/Datasheets/StepperDriveBasic.pdf.
Consulta: 25 de enero de 2015.

Si en un tiempo inicial $t = 0$ se excita la bobina con el voltaje de alimentación del motor la corriente del circuito crece de acuerdo a la siguiente ecuación:

$$i(t) = \frac{V}{R}(1 - e^{-\frac{R}{L}t})$$

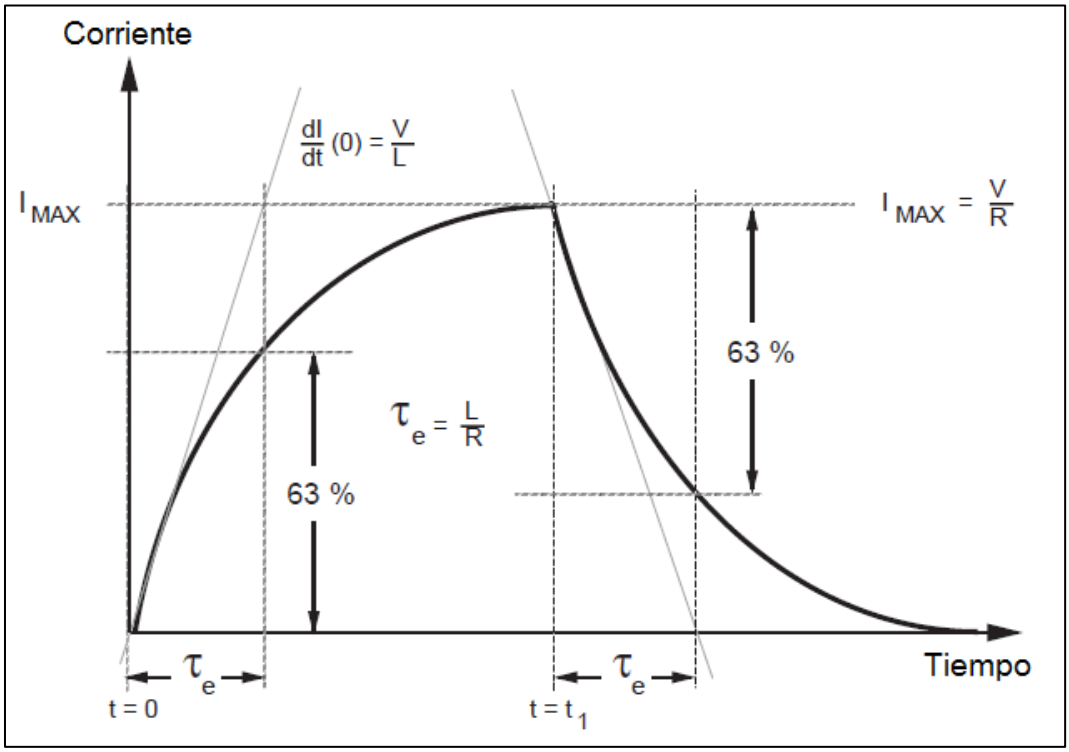
Donde L/R es una constante de tiempo dada por los parámetros de resistencia e inductancia de la bobina, y está definida como el tiempo necesario para que la corriente crezca o decaiga aproximadamente un 63 % de su valor final, tal como se muestra en la figura 6.

De la ecuación de corriente en la bobina se puede determinar que la máxima corriente que circulará por la bobina, después de un tiempo mucho

mayor a la constante de tiempo, está dada por $I_{max} = V/R$, y si se obtiene la razón de crecimiento en el tiempo inicial, se tiene que la corriente crece inicialmente de acuerdo con:

$$\frac{d}{dt} i(0) = \frac{V}{L}$$

Figura 6. **Corriente en una bobina del motor paso a paso**

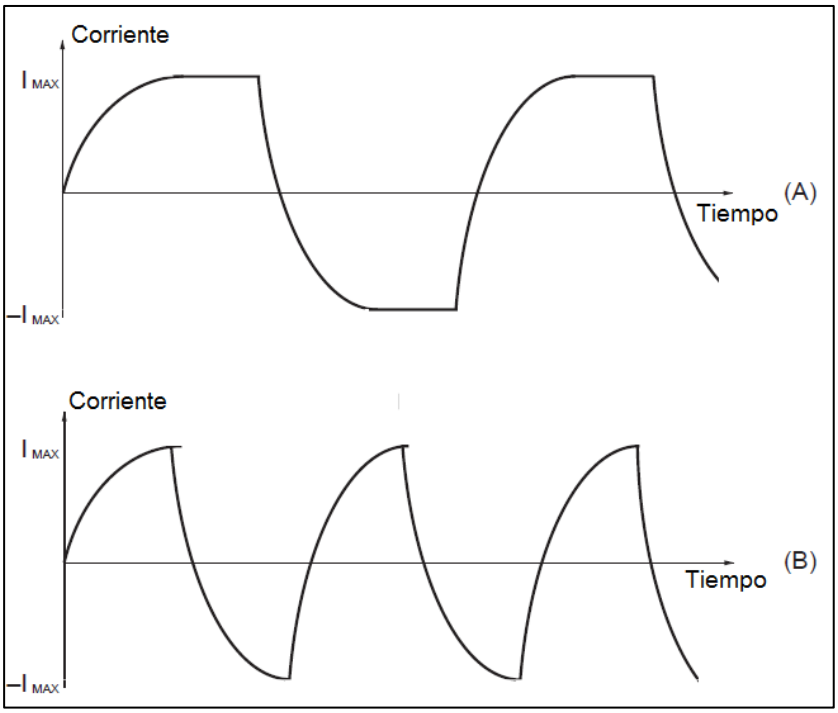


Fuente: *Drive circuit basics*.
http://users.ece.utexas.edu/~valvano/Datasheets/StepperDriveBasic.pdf.
Consulta: 25 de enero de 2015.

Cuando se aplica una señal cuadrada de voltaje de alimentación a la bobina del motor, que es el caso cuando se controla el motor utilizando la

secuencia de paso completo, la forma de onda de la corriente se ve suavizada, como se muestra en la figura 7. En el caso de la señal A, se observa que el tiempo de conmutación de la señal cuadrada de alimentación permite que la corriente en el motor llegue a su valor final y se mantenga energizado durante cierto tiempo.

Figura 7. **Corriente en una bobina con alimentación de onda cuadrada**



Fuente: *Drive circuit basics*.
http://users.ece.utexas.edu/~valvano/Datasheets/StepperDriveBasic.pdf.
Consulta: 25 de enero de 2015.

Conforme se aumenta la frecuencia de la señal cuadrada, la corriente deja de alcanzar su valor máximo, y ya que el torque es proporcional a la corriente, el máximo torque se ve reducido mientras la frecuencia de la señal de voltaje aumenta. Entonces, para superar el problema de la inductancia del motor y

obtener un desempeño de alta velocidad existen dos soluciones: aumentar la tasa de crecimiento o decrecimiento en la corriente del motor, y disminuir la constante de tiempo.

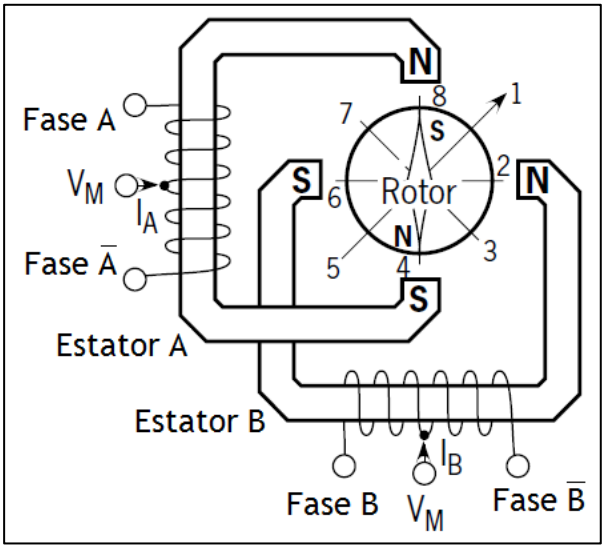
Para disminuir la constante de tiempo se puede disminuir la inductancia de la bobina o aumentar la resistencia. Mientras que aumentar la resistencia resulta en un aumento en la pérdida de potencia del motor, es preferible mantener la relación $\frac{V}{L}$ al máximo posible, ya que esta es la razón de cambio inicial en la corriente, lo que permite que la corriente alcance rápidamente niveles donde el torque del motor provee un buen desempeño.

Para manejar la corriente en la bobina entonces se debe utilizar un voltaje tan alto como sea posible manejar y mantener la inductancia tan baja como lo permita el proceso de fabricación. Generalmente, se manejan los motores paso a paso con voltajes de hasta 3 o 4 veces el voltaje nominal del motor (el voltaje que provoca la corriente máxima para la cual fueron diseñadas las bobinas del motor) y el controlador del motor se encarga de regular la corriente para no dañarlo.

1.7.3. Secuencias de movimiento

Los modos más comunes de operación de motores paso a paso son: la secuencia tipo *wave drive*, paso completo, medio paso y secuencias de micropasos. Para la explicación de las distintas secuencias de movimiento referirse a la figura 8.

Figura 8. **Representación de secuencias del motor paso a paso**



Fuente: *Stepper motor basics*.
http://users.ece.utexas.edu/~valvano/Datasheets/StepperBasic.pdf.
Consulta: 25 de enero de 2015.

1.7.3.1. Secuencia tipo *wave drive*

Esta secuencia de movimiento energiza solamente una de las bobinas del motor en cualquier momento. El estator es energizado de acuerdo con la secuencia $A \rightarrow B \rightarrow \bar{A} \rightarrow \bar{B}$, donde A representa una fase del motor energizada con una polaridad y $\bar{A}$ energizada con la polaridad opuesta. Con base en la figura 8, la secuencia de rotación consiste en la alineación de polos opuestos entre el rotor y los estatores, por lo que el motor se alinearía con los polos 2, 4, 6 y 8. Para motores unipolares y bipolares con los mismos parámetros de embobinado, este modo de excitación resultaría en la misma posición mecánica.

La desventaja de esta secuencia es que para motores unipolares se utiliza solamente un 25 % del embobinado disponible, y para motores bipolares un 50 % en cualquier momento, y con esto no se obtiene el máximo torque de salida del motor, por lo que el torque de paso y de retención es menor que con otras secuencias de movimiento.

1.7.3.2. Secuencia de paso completo

En esta secuencia están energizadas dos bobinas del motor en cualquier momento dado. El estator es energizado de acuerdo con la secuencia $AB \rightarrow \bar{A}B \rightarrow \overline{AB} \rightarrow A\bar{B}$. En la figura 8, la secuencia de rotación consiste en la alineación de polos opuestos entre el rotor y entre dos de los polos de los estatores, por lo que el motor se alinea con los polos 1, 3, 5 y 7. La secuencia de paso completo resulta en el mismo desplazamiento angular que la secuencia tipo *wave drive*, pero con un desfase en la posición angular de la mitad de un paso completo.

El torque de salida del motor unipolar es más bajo que el del motor bipolar para motores con los mismos parámetros de embobinado, ya que el motor unipolar utiliza solo un 50 % del embobinado disponible, mientras que el motor bipolar utiliza el 100 %.

1.7.3.3. Secuencia de medio paso

Esta secuencia combina la secuencia tipo *wave drive* y la secuencia de paso completo (1 y 2 fases energizadas). En un paso se energiza una fase, en el siguiente paso se mantiene energizada mientras se energiza la siguiente, y en el paso final se energiza solo la segunda, mientras que la primera se desenergiza.

El estator se energiza de acuerdo con la secuencia $AB \rightarrow B \rightarrow \bar{A}B \rightarrow \bar{A} \rightarrow \overline{AB} \rightarrow \bar{B} \rightarrow A\bar{B} \rightarrow A$; en la figura 8 el rotor se alinea con cada uno de los posibles polos en el estator. Esto resulta en un desplazamiento angular de la mitad de un paso completo.

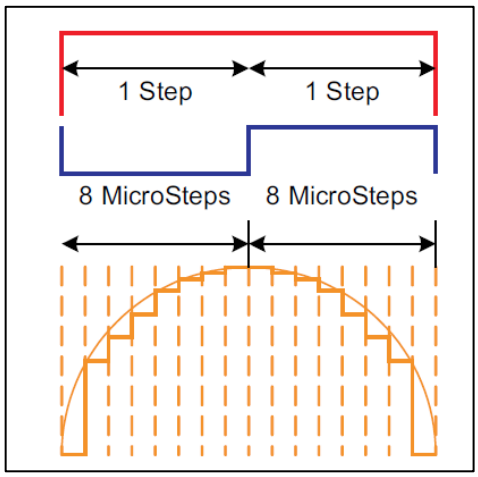
1.7.3.4. Secuencia de micropasos

Mover un motor paso a paso utilizando una secuencia de micropasos consiste en cambiar el flujo del estator más suavemente que en los modos de paso completo y medio paso. Por lo que un paso completo del motor puede ser dividido en múltiples pasos más pequeños, llamados micropasos, y se puede obtener observar una mejor calidad de movimiento del motor. La aplicación de la secuencia de micropasos a cualquier motor permite mejoras tales como: una mejor respuesta del torque, menor vibración del motor y menor incidencia de resonancia del motor.

La idea detrás de los micropasos es hacer que la corriente fluya en las bobinas del motor de forma distinta a la corriente máxima en una dirección y a la corriente máxima en la dirección opuesta. La utilización de micropasos se refiere comúnmente a la utilización de senos y cosenos para hacer que la corriente se aproxime a una forma de onda sinusoidal de corriente alterna.

En la figura 9 se puede observar una representación de los micropasos en la forma de onda de la corriente en la bobina de un motor. Por cada paso completo del motor, la técnica de control con micropasos divide el paso en 8 pasos más pequeños, y para cada uno se controla que la corriente en la bobina alcance cierto nivel predeterminado, para que en combinación con el resto de micropasos, la forma de onda en la corriente de la bobina se aproxime a la forma deseada, en este caso, el de una señal sinusoidal.

Figura 9. **Micropasos en la bobina del motor**



Fuente: *High Resolution Microstepping Driver*.
http://www.ti.com/lit/an/slva416/slva416.pdf.
Consulta: 21 de enero de 2015.

1.7.4. Esquemas de control de motores paso a paso

Un circuito manejador de motores paso a paso cumple dos tareas específicas: cambiar la dirección de la corriente y del flujo magnético en las bobinas de fase del motor, y manejar una cantidad controlable de corriente y permitir que los tiempos de crecimiento y decaimiento de la corriente del motor sean tan pequeños como sea posible para un buen desempeño de velocidad.

Los motores paso a paso requieren un cambio de dirección del flujo magnético independientemente en cada fase. El cambio de dirección se hace cambiando la dirección de la corriente, que puede hacerse de dos maneras: utilizando un controlador bipolar o un controlador unipolar.

1.7.4.1. Controlador bipolar

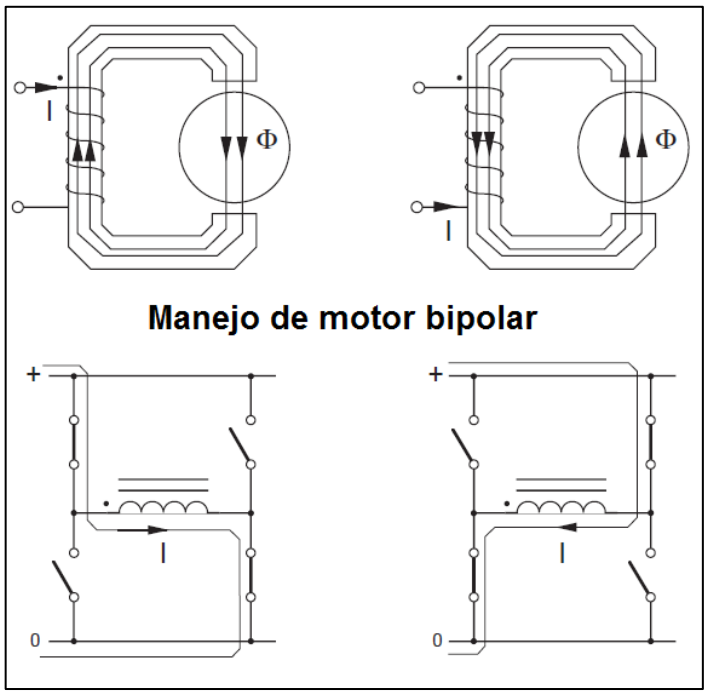
Un controlador bipolar utiliza el principio de cambiar la dirección de la corriente en un embobinado, cambiando la polaridad del voltaje aplicado a las terminales del embobinado.

Para cambiar la polaridad se necesita un total de 4 interruptores, formando un puente H. El método de control bipolar requiere un embobinado por fase, por lo que un motor de dos fases tendrá dos embobinados y por tanto cuatro terminales de conexión.

En la figura 10 se puede observar que la corriente circula a través del embobinado completo y dependiendo de la dirección de la corriente es el flujo magnético generado.

Este cambio en la dirección del flujo se logra cambiando de polaridad el voltaje aplicado en las terminales de la bobina (como se muestra en la parte inferior) a través de la conmutación de un par de interruptores a la vez, uno que permita la conexión del voltaje de alimentación del motor, y otro que permita la conexión del voltaje de referencia. Este cambio de dirección en el flujo magnético hace que el rotor gire hacia un lado o hacia el otro.

Figura 10. **Principio de funcionamiento de un controlador bipolar**

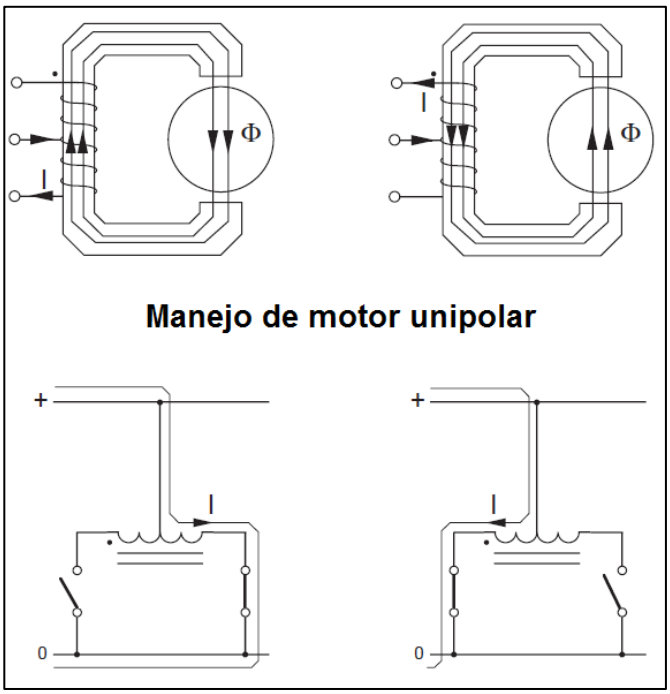


Fuente: *Drive circuit basics*.
http://users.ece.utexas.edu/~valvano/Datasheets/StepperDriveBasic.pdf.
Consulta: 21 de enero de 2015.

1.7.4.2. Controlador unipolar

El principio de funcionamiento de un controlador unipolar requiere que los embobinados del motor posean *tap* central, o dos embobinados separados para cada fase. La dirección del flujo magnético se logra cambiando la trayectoria de la corriente en la bobina, utilizando el *tap* central como una conexión común hacia los dos extremos de la bobina. En la figura 11 se puede observar que la alimentación de un extremo de la bobina permite la rotación en una dirección, y el otro extremo de la bobina con el *tap* central permite la rotación en la otra dirección.

Figura 11. **Principio de funcionamiento de un controlador unipolar**

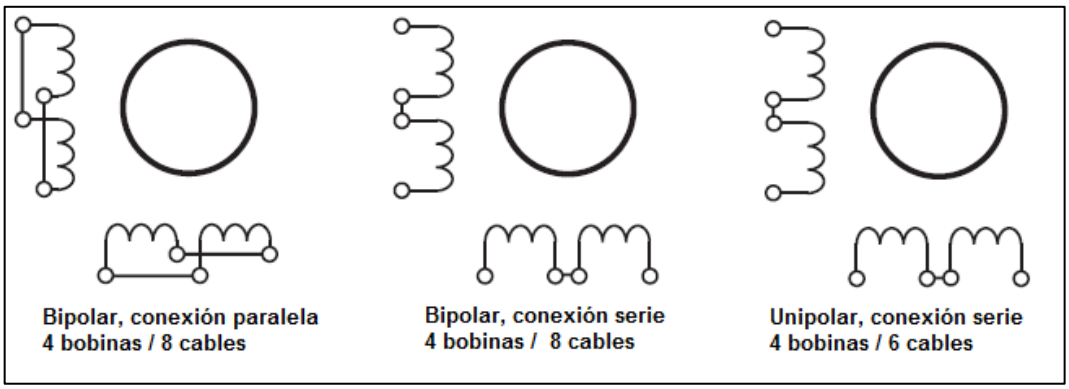


Fuente: *Drive circuit basics*.
http://users.ece.utexas.edu/~valvano/Datasheets/StepperDriveBasic.pdf.
Consulta: 21 de enero de 2015.

Este método requiere de dos interruptores por fase, pero utiliza solo la mitad del volumen disponible del embobinado. Por tanto, la pérdida de potencia en la bobina es el doble que la que provoca un controlador bipolar para una potencia de salida fija.

El motor unipolar con tap central tiene generalmente 3 cables por fase, haciendo un total de 6 cables para el motor. Un motor que tiene dos embobinados separados es generalmente conocido como un motor de 8 cables. Ambos tipos de motores (con 6 cables y con 8 cables) pueden ser utilizados con un controlador bipolar o unipolar, tal como se muestra en la figura 12.

Figura 12. **Esquemas de conexión de un motor paso a paso de 8 cables**



Fuente: *Drive circuit basics*.
http://users.ece.utexas.edu/~valvano/Datasheets/StepperDriveBasic.pdf.
Consulta: 21 de enero de 2015.

1.7.5. Control de corriente

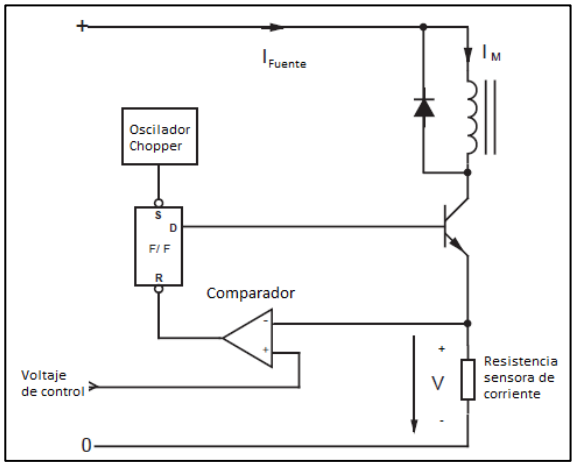
Para controlar el torque, así como el límite en la disipación de potencia en la resistencia de la bobina del motor, la corriente debe ser controlada o limitada. Además, cuando se utiliza la secuencia de medio paso se requiere un nivel de corriente cero, mientras que cuando se utilizan micropasos se requiere una corriente continuamente variable.

La técnica más utilizada es la de control *chopper*, que provee una solución óptima al problema de control de corriente y rápido crecimiento y reversa de la corriente. La idea básica es utilizar una fuente de alimentación que sea varias veces mayor que el voltaje nominal del motor y conmutar la alimentación del motor. La tasa de crecimiento inicial de la corriente está dada por V/L, donde V es el voltaje de alimentación del motor y L es la inductancia de la bobina, y por

tanto, es posible manipular el voltaje de alimentación para aumentar la tasa de crecimiento de la corriente.

Controlando el ciclo de trabajo del control *chopper* se puede obtener un voltaje y una corriente promedio que sean iguales a los valores nominales del motor. El control *chopper* se configura para una regulación constante de corriente, que se obtiene conmutando la corriente de las bobinas. El proceso requiere la utilización de una resistencia sensora de corriente para la corriente pico a través de la bobina, que se conecta en serie con la bobina del motor. Conforme la corriente aumenta, el voltaje, a través de la resistencia, es proporcional al valor de la corriente que pasa a través de ella ($V = IR$). Este voltaje se retroalimenta a un comparador de voltaje.

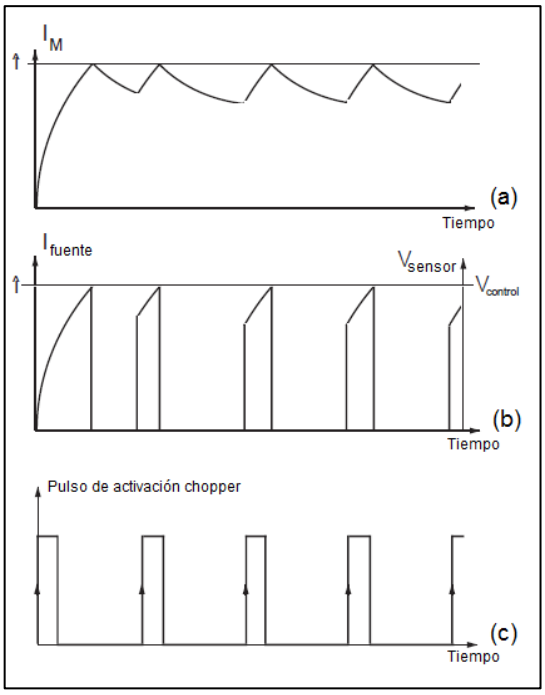
Figura 13. **Esquemático de un control de corriente tipo *chopper***



Fuente: *Drive circuit basics*.
http://users.ece.utexas.edu/~valvano/Datasheets/StepperDriveBasic.pdf.
Consulta: 28 de enero de 2015.

En la figura 13 se puede observar que el voltaje de retroalimentación se utiliza como entrada de un comparador con una referencia de voltaje constante. El comparador se encarga de reiniciar el *flip flop* de control que maneja un transistor de salida para controlar la bobina del motor. La corriente en la bobina decrece hasta que el reloj del oscilador del control *chopper* activa nuevamente el *flip flop*, que enciende el transistor nuevamente y el ciclo es repetido.

Figura 14. **Formas de onda de corriente en un controlador *chopper***



Fuente: *Drive circuit basics*.
http://users.ece.utexas.edu/~valvano/Datasheets/StepperDriveBasic.pdf.
Consulta: 28 de enero de 2015.

En la figura 14 (a) se puede observar las forma de onda de corriente en la bobina del motor, la corriente crece cuando está activo el transistor que maneja la bobina, y decrece cuando el transistor entra en estado de corte. En la figura

14 (b) se observa que la fuente de alimentación provee corriente en los intervalos de tiempo cuando la corriente del motor está aumentando. En la figura 14 (c) se observa el tren de pulsos de activación del circuito de control *chopper*. Estos permiten que se active el transistor que maneja la bobina y empiece a crecer la corriente en la bobina del motor.

La ventaja de un control de corriente constante es un control preciso del torque desarrollado, sin importar las variaciones en la fuente de alimentación. También permite un rápido desarrollo y reversa de la corriente, es decir, la corriente alcanza el valor necesario que permite que se desarrolle el torque necesario para dar un paso del motor más rápidamente, al contrario del caso en que se trabaja con el voltaje nominal del motor, ya que la relación V/L es más baja. La disipación de potencia es minimizada, así como la demanda de corriente.

La demanda de corriente no es la misma que la corriente nominal del motor para el control *chopper*, es la corriente nominal del motor multiplicada por el ciclo de trabajo del oscilador del control, como se describe por la siguiente ecuación:

$$I_{Fuente} = I_{Motor} \frac{V_{Motor}}{V_{Fuente}}$$

La figura 15 muestra el diagrama de un controlador general para una de las bobinas de un motor paso a paso. El controlador consiste en un puente H configurado como un controlador *chopper* de corriente constante. El bloque de lógica recibe la dirección con la cual debe polarizar la bobina del motor y maneja los transistores que controlan el puente H.

La corriente circula desde la terminal positiva de alimentación, indicada con (+) en el diagrama, pasando por uno de los transistores superiores del puente H, a través de la bobina y por uno de los transistores inferiores del puente H. Finalmente, la corriente pasa por la resistencia sensora hacia la referencia de la fuente de alimentación, indicada con (0). El recorrido de la corriente en el circuito se muestra como la trayectoria 1 en la figura 15.

Cuando la corriente pasa por la resistencia genera un voltaje que sirve de comparación contra un valor fijo de referencia y la salida del comparador señaliza al circuito de lógica para que realice la conmutación de los transistores a la frecuencia del oscilador *chopper*.

Dependiendo de cómo el puente H sea conmutado durante el período de apagado, la corriente puede tomar una de dos trayectorias posibles:

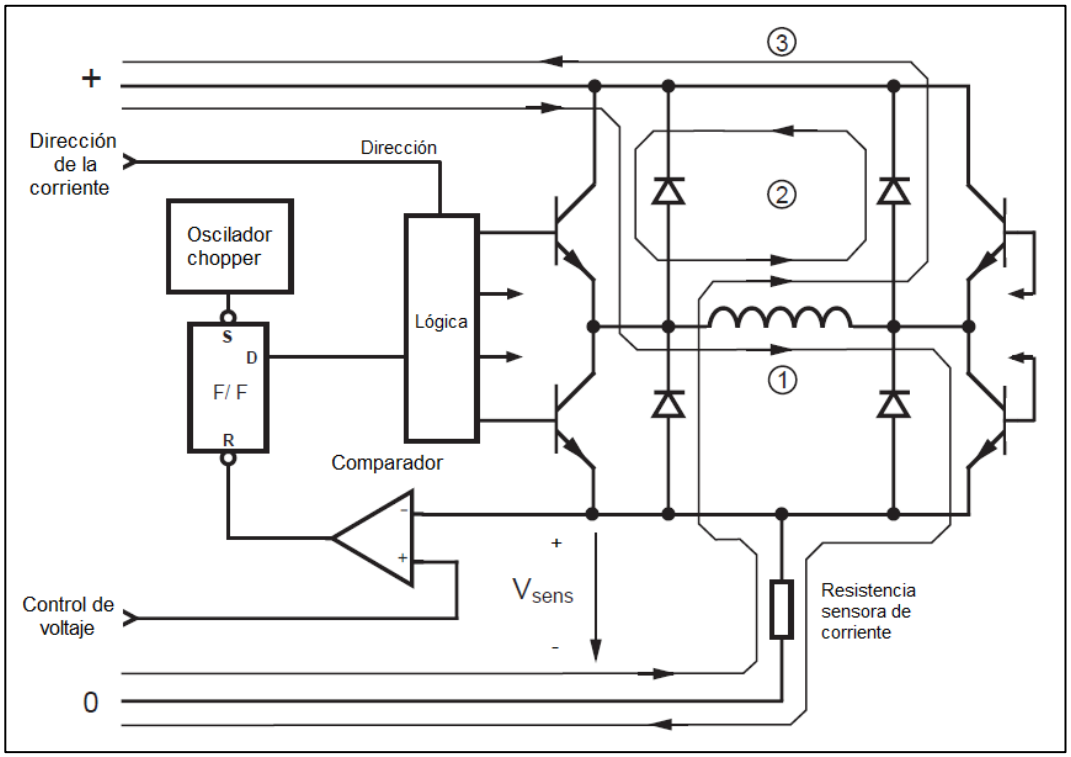
- La corriente recircula a través de un diodo y un transistor, como se muestra en la trayectoria 2 de la figura 15. Esto provoca que el decaimiento de la corriente sea lento.

- La corriente recircula de vuelta a través de la fuente de alimentación, como se muestra en la trayectoria 3 de la figura 15. La ventaja de este método es el rápido decaimiento de la corriente y la habilidad de reducir rápidamente a un nivel de corriente inferior.

Una aplicación importante de este método se da cuando se utilizan micropasos para manejar la corriente y esta tiene pendiente negativa, que sería imposible de seguir si el decaimiento en la misma es menor de lo que la pendiente demanda. La desventaja del rápido decaimiento de la corriente es que aumenta el rizado, lo que puede causar pérdidas en las bobinas del motor.

Algunos circuitos integrados de control de motores paso a paso, como el CI DRV8818 de Texas Instruments, incluyen un modo mixto de operación para el decaimiento en la corriente, el cual ayuda a disminuir el rizado y mantener aún así el rápido decaimiento en la corriente.

Figura 15. **Puente H con control de corriente *chopper***



Fuente: *Drive circuit basics*.
http://users.ece.utexas.edu/~valvano/Datasheets/StepperDriveBasic.pdf.
Consulta: 30 de enero de 2015.

1.8. Interfaz de control electrónico

Es una de las tarjetas electrónicas que conforman todo el equipo. Su función es recibir los datos provenientes de la computadora para control de los

motores de la máquina. En algunos casos, este equipo también puede realizar la interpretación del código G enviado por la computadora para manejar los motores. Actúa como un intermediario entre la computadora y las tarjetas electrónicas que controlan los motores; ya que cada motor requiere una secuencia de movimiento para posicionarse, se encarga de enviar las señales correspondientes a cada controlador individual de motor de forma simultánea para mover la máquina de acuerdo con el código G.

La interfaz de control se encarga también de manejar el encendido y apagado de las demás herramientas de la máquina, como líquidos refrigerantes, aspiradoras de viruta, y la herramienta rotativa. Asimismo, a esta se conectan las entradas de la máquina, como los interruptores de límite de los ejes de la máquina, la punta de prueba para superficies, y el botón de parada de emergencia de la máquina.

2. PROCESO DE CONSTRUCCIÓN DE MÁQUINA FRESADORA

Se muestra el proceso de construcción de la máquina fresadora, los materiales, las herramientas utilizadas, y otros aspectos importantes de la mecánica que conlleva el proyecto.

2.1. Diseño de la máquina fresadora de escritorio

La máquina que se construyó para el proyecto consiste en una fresadora de escritorio de 45 cm de ancho por 65 cm de largo, cuya estructura está hecha principalmente de madera, un marco metálico y partes de aluminio. La máquina consiste en una estructura con 3 ejes de movimiento, perpendiculares entre sí, que permiten el movimiento de una herramienta rotativa en un espacio de trabajo. Los ejes serán considerados por facilidad como X, Y, y Z, donde los ejes X y Y forman el plano de trabajo de la máquina y el eje Z corresponde al desplazamiento vertical de la herramienta.

2.1.1. Herramientas y equipo necesario

Para llevar a cabo la construcción de la máquina fue necesario contar con los siguientes materiales y herramientas:

- Un taller de mecánica y soldadura: para realizar el corte de las piezas de madera y la soldadura del marco que sirvió de base para la máquina y los acoples de transmisión de movimiento rotativo de los motores al movimiento lineal.

- Madera tipo *plywood*: se utilizaron planchas de 7/16 in y de 11/16 in para elaborar las piezas de la estructura.

- Perfiles de aluminio angular de 1/8 in: para el desplazamiento de los ejes se utilizó un sistema de rieles, sobre los cuales se desplazan carros fuertemente sujetados.

- Cojinetes: se utilizaron cojinetes de 19 mm de diámetro externo, y 6mm de diámetro interno, modelo SKF 626-2RS1, para realizar el desplazamiento de los carros en la máquina. Para la sujeción de las varillas roscadas en la estructura de la máquina se utilizaron cojinetes de 1/4 in de diámetro interno, 5/8 in de diámetro externo y 3/16 in de grosor.

- Varilla roscada: se utilizaron varillas roscadas de 1/4 in para la transmisión de movimiento y para mejorar la precisión en el posicionamiento.

- Juego de tornillos milimétricos: se utilizó un amplio juego de tornillos de 4 milímetros (M4) de varias longitudes para armar la máquina y sujetar las piezas.

- Equipo para corte de madera: se utilizó una sierra eléctrica para el corte de las piezas de la máquina.

- Prensa y equipo de sujeción: para mantener las piezas fijas mientras eran cortadas y se les abrían agujeros.

- Equipo de protección: se utilizaron gafas de protección para el corte de piezas y guantes para la manipulación de las piezas de madera.

2.1.2. Cojinetes y carriles

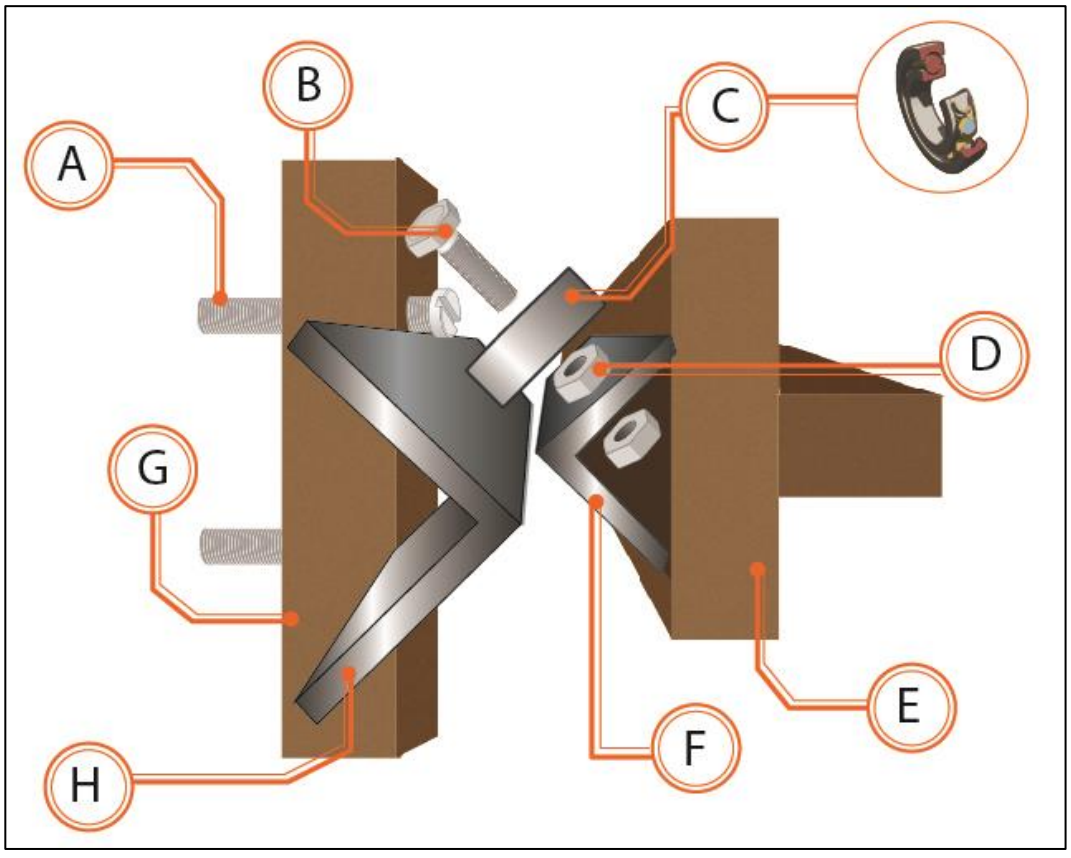
Para el movimiento de cada uno de los ejes de la máquina fresadora se armaron carriles utilizando perfiles de aluminio de 90º con un grosor de 1/8 in, sobre los cuales se desplazan los carros que conforman los distintos ejes de la máquina. Cada carro consta de un conjunto de cojinetes que sirven como llantas para moverse.

La armazón básica de cada uno de los carros se muestra en la figura 16. Cada uno de los carriles está montado sobre una parte fija en la máquina, como se indica en la pieza (G). Cada una de estas piezas fijas sostiene un perfil de aluminio, que sirve como carril para el desplazamiento de cada carro. Este perfil está indicado como la pieza (H), y está sujetado a la pieza fija de la máquina con tornillos. En la figura se muestra la armazón más general, utilizando el perfil para el eje X de la máquina, que es sujetado utilizando tornillos pasados a través de la pieza fija mientras sujetan el perfil de aluminio con la cabeza, estos tornillos se muestran como la pieza (A) en la figura. Por el otro lado de la pieza fija, son apretados utilizando para cada tornillo una roldana y una tuerca.

En la figura 16 se muestra la armazón del carro para el eje X, la pieza móvil consiste en una armazón de madera, indicada como la pieza (E), que se desplazará de acuerdo con los movimientos que realice el motor. El carro posee un perfil de aluminio igual al de la pieza fija, indicado como la pieza (F), sobre el cual se montan cojinetes (C) que sirven para el desplazamiento lineal de los carros. Estos cojinetes están sujetos al perfil de aluminio de la pieza móvil con tornillos (B) y tuercas (D).

Es importante que la pieza móvil (E) se ajuste contra la pieza fija (H), de tal forma que no exista una holgura mecánica que pueda llevar posteriormente a errores en la ejecución de los movimientos de trabajo de la máquina fresadora. De la misma forma, la sujeción del perfil de aluminio de la pieza fija debe estar fuertemente asegurada, ya que este soporta la fuerza de sujeción de la pieza móvil y el peso del carro que se mueve a lo largo del perfil.

Figura 16. **Armazón básica de los carriles para movimiento de los ejes**

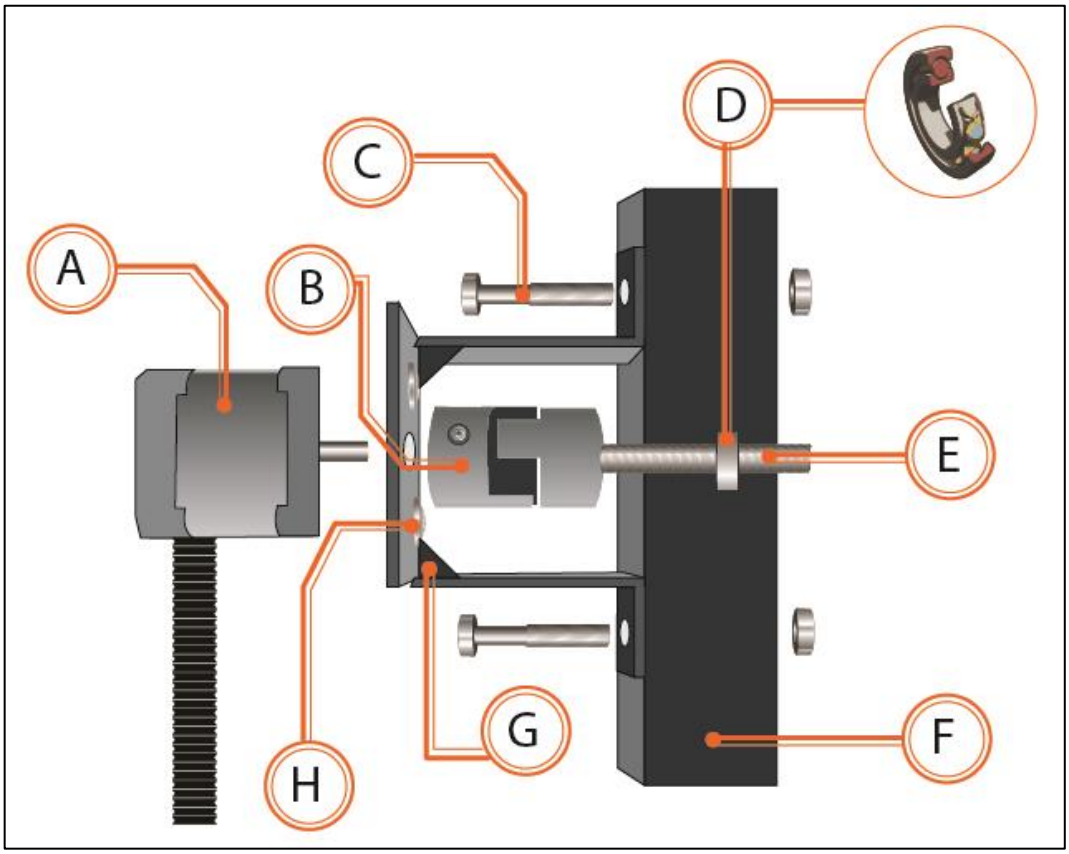


Fuente: elaboración propia, utilizando el programa Adobe Photoshop.

2.1.3. Transmisión del movimiento

El movimiento de los carros que componen la máquina es un desplazamiento lineal provocado por el desplazamiento de una tuerca fija en la pieza móvil. Cuando los motores generan el movimiento rotacional, este es transmitido a través de una varilla roscada, paralela al eje de desplazamiento del carro, y conforme la varilla gira, la tuerca se desplaza junto con el carro, obteniendo así el desplazamiento necesario para ejecutar los movimientos de la máquina.

Figura 17. **Transmisión del movimiento y acople del motor a los ejes**



Fuente: elaboración propia, utilizando el programa Adobe Photoshop.

En la figura 17 se muestra el diagrama general del acople del motor a la máquina fresadora utilizando el eje X como referencia. En la figura se observa que el motor paso a paso (A) es sujetado por la máquina a través de una base fija (G) que posee una disposición especial para los agujeros, indicados por (H), que permiten el paso de tornillos M3 de sujeción para el motor NEMA 17. El eje del motor es finalmente atornillado a un acople *anti-backlash* (B), que por un lado tiene el diámetro adecuado para el eje del motor de 5 mm, y por el otro lado está dimensionado para sostener la varilla roscada de 1/4 in. Más adelante se describirá la necesidad de utilizar este tipo de acople para la transmisión.

La varilla roscada es pasada a través de dos cojinetes, cuyo diámetro interno es de 1/4 in, que permiten sostener la varilla y sujetarla a la máquina. En la figura se muestra uno de los cojinetes (D), que en el caso del eje X se sujeta a la máquina en la base (F) de la misma, y la varilla roscada (E) pasa a través del cojinete para finalmente sujetarse junto con el motor en el acople (B).

De esta forma se logra que la varilla roscada y el eje del motor formen un solo cuerpo de rotación, así cuando el motor paso a paso gira mientras es controlado por las señales eléctricas de controlador, la varilla roscada gira junto con el eje del motor, haciendo que se desplace una tuerca sujeta al carro en uno de los ejes de la máquina, formando así el desplazamiento lineal.

En cuanto a la precisión del movimiento, se utilizaron varillas roscadas de 1/4 in, que poseen un paso de 20 vueltas por pulgada. Es decir, si se dan 20 vueltas a la varilla, una tuerca sobre la varilla se desplaza exactamente una pulgada. De modo que la precisión está relacionada con el paso de la varilla y con la precisión angular del motor (el mínimo ángulo de rotación del eje del motor cuando se da un paso).

Si se considera que el motor es de 200 pasos por revolución (1,8º por paso) y se está utilizando una secuencia de medio paso para manejar el motor, la precisión angular aumenta a 400 pasos por revolución (0,9º por paso). Con estos datos es posible calcular el mínimo desplazamiento lineal que se provoca en el carro desplazado por la varilla cuando se da un paso. El cálculo se muestra en la siguiente ecuación:

$$0{,}9° * \frac{1\ rev}{360\ °} * \frac{1\ in}{20\ rev} * \frac{25.4\ mm}{1\ in} = 0{,}003175\ mm$$

Es decir que la precisión en el desplazamiento lineal es de aproximadamente 3,2 micrones, siempre y cuando los carros se puedan mover libremente y la fuerza del motor sea suficiente para manejar la varilla y mover el carro. Cuando el motor no tiene fuerza suficiente para mover el carro ocurre un fenómeno conocido como “pérdida de pasos” donde el controlador envía las señales para hacer avanzar la máquina en algún eje y este no se desplaza exactamente las unidades indicadas debido a problemas en la mecánica de transmisión.

2.1.4. Acople de motores paso a paso

El acoplamiento del motor paso a paso hacia el elemento de transmisión mecánica consiste en una pieza que conecta el eje del motor con la varilla roscada para la transmisión del movimiento rotacional a un desplazamiento lineal. Este acople debe ser suficientemente flexible para absorber pequeñas desviaciones producidas durante el giro del motor, sin embargo, debe ser lo suficientemente rígido para garantizar una buena transmisión.

En la transmisión del movimiento puede darse un juego mecánico, conocido popularmente como *backlash*, que consiste en un espacio o pérdida de movimiento en un mecanismo, causado por los espacios entre las partes, en este caso, entre la varilla roscada y el eje del motor. Formalmente el *backlash*, puede ser descrito como el ángulo o la distancia máxima en que se puede mover un sistema mecánico en una dirección sin aplicar una fuerza apreciable en la siguiente parte de la secuencia mecánica.

Este desplazamiento puede traducirse en un paso realizado en el eje del motor, pero no transmitido hacia la varilla roscada, perdiendo un paso de precisión en el desplazamiento lineal. Es por esto que se utilizan acoples *anti-backlash*, como el que se muestra en la figura 17 indicado por (B). Este acople consiste en dos piezas metálicas que están unidas entre sí por una pieza de goma lo suficientemente rígida para mantener la torsión en el eje cuando se produce un paso en el motor. De esta forma el eje del motor paso a paso y la varilla roscada se vuelven un solo cuerpo de rotación, aumentando la precisión en el movimiento de la máquina.

Finalmente, es importante mencionar que el fenómeno de *backlash* no se da solo en el acople mecánico del motor a la varilla roscada, también existe entre los hilos de la varilla roscada y la tuerca que mueve los carros en los ejes. Es decir, existe un juego mecánico entre las roscas de la varilla y las de la tuerca de transmisión, que es una desventaja del tipo de transmisión utilizando una varilla roscada común de 1/4 in. Para minimizar el efecto del *backlash* producido en la varilla se utilizaron tuercas de transmisión largas (de 1/4 in por 22 mm de largo) que presentan un menor juego mecánico que las tuercas comunes de 5 mm.

2.1.5. Descripción del proceso de construcción

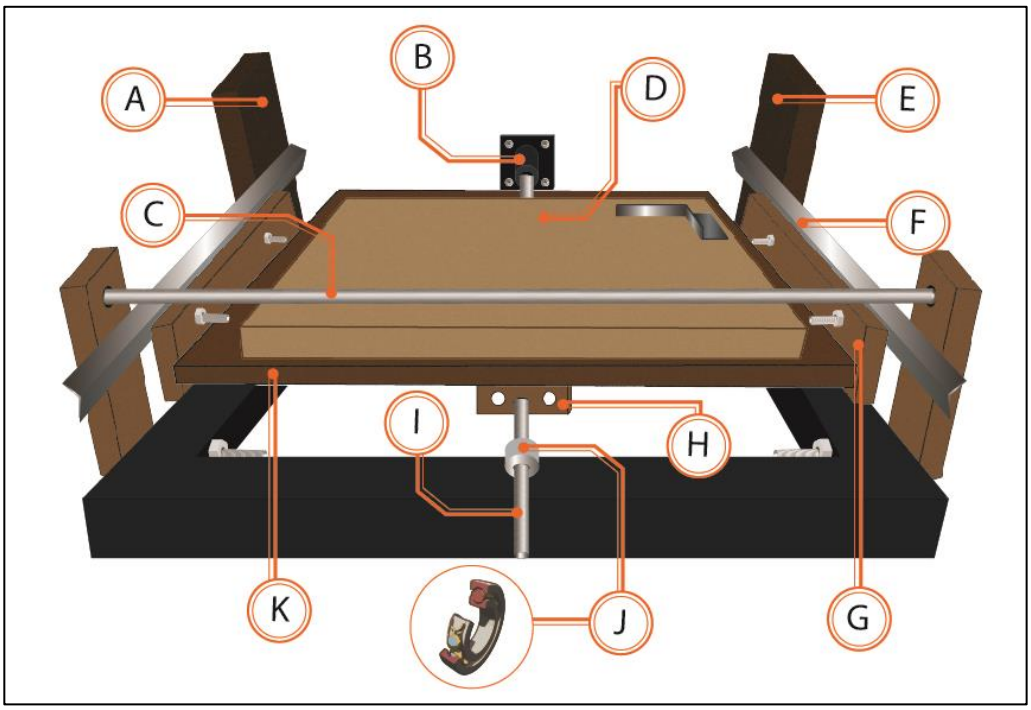
Se muestran los pasos de construcción general de la máquina utilizando diagramas con las piezas identificadas, sin embargo, los diseños completos de las piezas que componen la máquina fresadora se encuentran en la sección de anexos. Estos diseños están identificados por el eje al que pertenece la pieza y los nombres utilizados para las piezas en esta sección.

2.1.5.1. Construcción del eje X

La construcción del eje X empieza a partir de la base, que consiste en un marco metálico de 43 cm de ancho y 53 cm de largo, elaborado con un tubo cuadrado de acero al carbón de 1,5 in. Este marco sirve para sujetar las piezas de madera sobre las cuales se montan los perfiles de aluminio para los carros que conforman los ejes de la máquina.

En la figura 18 se observa un diagrama de la vista frontal de la máquina, donde se observan las piezas y componentes que conforman el eje X. Las piezas indicadas por (A) y (E) son elaboradas con madera tipo *plywood*; más adelante se hará referencia a estas piezas como los “brazos del puente”, ya que entre estas es sostenida una tercera pieza de madera conocida como “puente”, sobre el cual se desplaza el carro del eje Y y Z.

Figura 18. **Construcción del eje X en la máquina fresadora**



Fuente: elaboración propia, utilizando el programa Adobe Photoshop.

Las piezas (A) y (E) están sujetas al marco con tornillos de 1/4 in en ambos lados del marco de la base. Adicionalmente, se utilizan otras dos piezas de madera para sostener los perfiles de aluminio (F), sobre los cuales se desplaza el carro del eje X. Estas piezas están sujetas también al marco de la máquina utilizando tornillos de 1/4 in. La sujeción al marco con tornillos tiende a abrir las piezas, por lo que se utilizó otra varilla roscada de 1/4 in, indicada como la pieza (C), con el propósito de tensar las dos piezas entre sí y sostener firmemente los perfiles de aluminio.

Entre los perfiles de aluminio se inserta el carro para el eje X. Este carro está conformado por dos piezas exactamente iguales, identificadas por (G) que

se unen con otra pieza de madera identificada por (K), que sirve para sostener la tabla de sacrificio (D) sobre la cual se montan las piezas a trabajar en la máquina fresadora. El carro corre libremente sobre los perfiles de aluminio utilizando otros perfiles que ajustan perfectamente los cojinetes sobre el carril, tal como está indicado en la figura 16, que es una vista de perfil de la pieza (G) con las piezas de madera laterales.

Debajo del carro se encuentra otra pieza de madera, indicada por (H), que contiene una tuerca de 1/4 in por 22 mm, sobre la cual gira la varilla roscada (I) para permitir el desplazamiento del carro. Una vez insertado el carro del eje X sobre el carril formado por los perfiles (F) se inserta la varilla roscada de 60 cm de largo en la pieza (H) y se gira hasta la mitad de la varilla. Después se insertan cojinetes (J) de 1/4 in de diámetro interior en las varillas roscadas y sobre el marco de la máquina, que sirven para sostener la varilla en una posición fija.

Una vez el carro se mueva libremente con la varilla y se inserten los cojinetes de sujeción, estos se dejan fijos en una posición de la varilla utilizando tuercas de 1/4 in y se sujetan con tornillos al marco de la máquina. Finalmente uno de los extremos de la varilla se sujeta al acople del motor (B) para la transmisión del movimiento, lo que inmoviliza el carro del eje X, hasta que el motor gire provocando el desplazamiento de la tuerca en la pieza (H).

2.1.5.2. Construcción del eje Y

La construcción del eje Y empieza con el montaje del puente sobre los brazos. El puente es una pieza de madera tipo *plywood* de 43 cm de largo, que se monta entre las piezas (A) y (E) de la figura 18, utilizando tornillos de 1/4 in, pasados a través de las mismas, y tuercas en orificios del puente. El grosor de

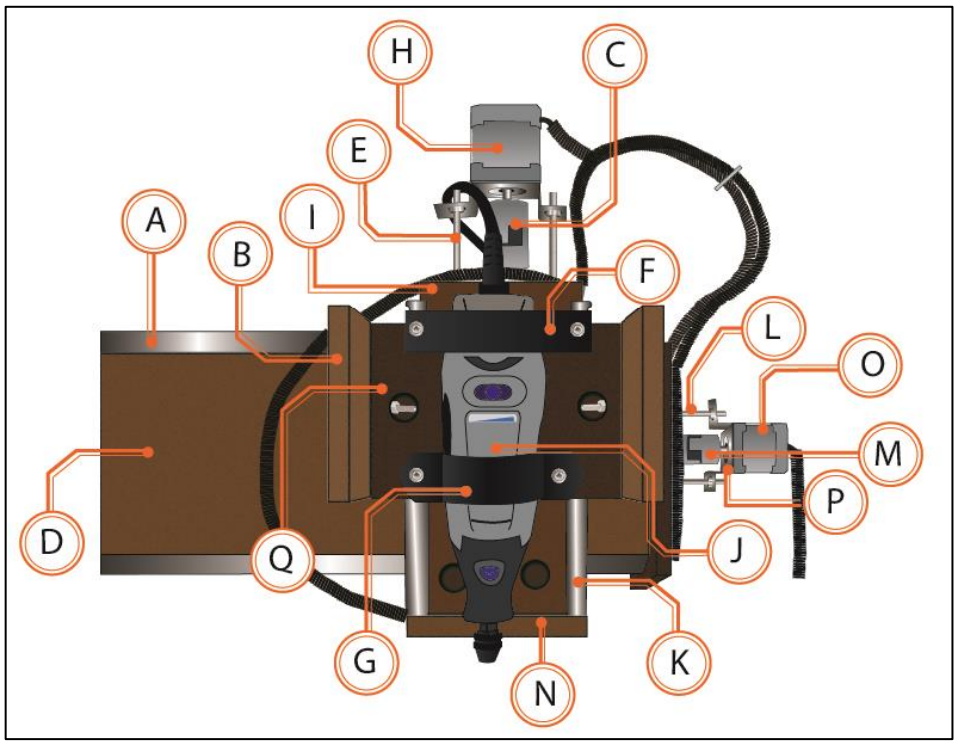
la pieza es de 11/16 in, que es mayor al de las otras piezas de madera, ya que sobre dicha pieza recae el peso del carro del eje Y y Z, y del motor del eje Z. Además, en esta pieza se montan los perfiles de aluminio que permiten el desplazamiento del carro del eje Y.

En la figura 19 se observa el diagrama de piezas que corresponde al eje Y, y al eje Z de la máquina fresadora. El puente está indicado como la pieza (D), sobre el cual se montan los perfiles de aluminio (A) que sirven como carriles para el carro del eje Y. El carro está conformado por las piezas (I), (N), (K) y una pieza trasera. Las piezas (I) e (N) contienen perfiles de aluminio internos, sobre los cuales están los cojinetes para sujetar a los perfiles de aluminio del puente.

Sobre uno de los brazos del puente, en el área derecha de la figura 19, se observa el montaje del motor paso a paso (O), sostenido al brazo del puente por una pieza de aluminio (P), diseñada específicamente para el marco del motor. Esta pieza está sujeta al brazo del puente utilizando tornillos M4 de 60 mm de largo y tuercas entre el brazo del puente y los tornillos, y en la pieza para el motor. El motor se une con la varilla roscada a través del acople *anti-backlash* (M), permitiendo la transmisión de movimiento y el desplazamiento del carro del eje Y.

El carro del eje Y debe ser ensamblado sobre el puente después de armar completamente el eje Z de la máquina, cuyo desplazamiento es sobre la pieza (K) del carro del eje Y. Para completar la construcción del carro del eje Y se utilizan tornillos M4 de 40 mm de largo, pasados en las piezas (I) e (N) y apretados con una tuerca sobre agujeros idénticos realizados en la pieza (K) y en la pieza trasera.

Figura 19. **Construcción del eje Y y Z de la máquina fresadora**



Fuente: elaboración propia, utilizando el programa Adobe Photoshop.

2.1.5.3. Construcción del eje Z

La construcción del eje Z se realiza sobre el carro del eje Y, utilizando la pieza (K) en el diagrama de la figura 19 como carril para el eje Z, sobre el cual se montan dos perfiles de aluminio, necesarios para el desplazamiento del carro del eje Z. Para el montaje de los carriles se utilizaron tornillos busca rosca de 20 mm de largo.

Sobre la pieza (I) en el diagrama la figura 19 se monta otra pieza de aluminio utilizando tornillos M4 (E) de 60 mm de largo. Esta pieza sirve para sostener al motor paso a paso (H) correspondiente al eje Z. El motor se une con

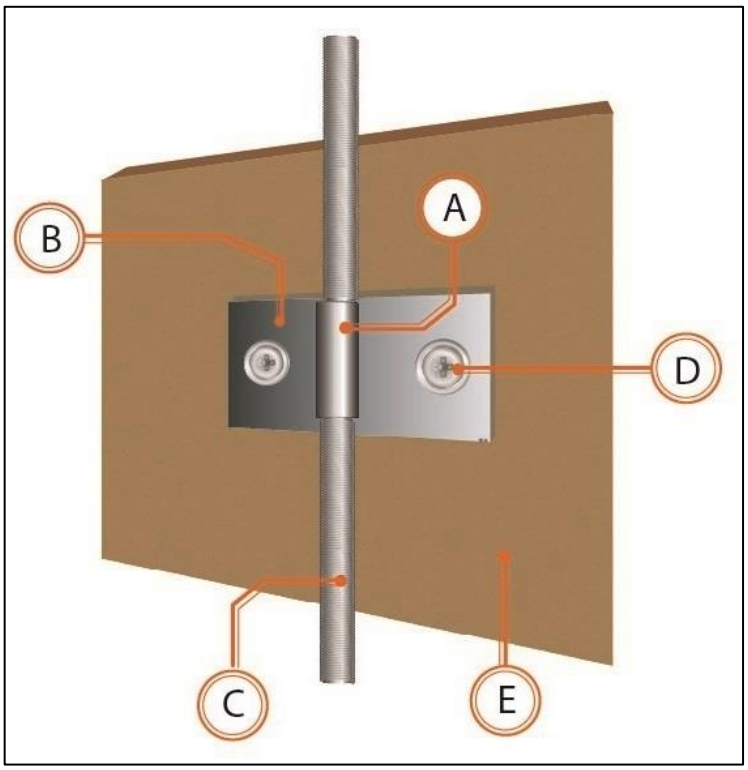
la varilla roscada a través de otro acople *anti-backlash* y la varilla roscada pasa a través del carro del eje Y por la parte de atrás, guiada por un cojinete de sujeción a la pieza (I), hasta llegar a la pieza (N). Es importante que la varilla se sujete a la pieza (I) de forma independiente, utilizando tuercas de 1/4 in, de lo contrario el peso del carro del eje Z caería sobre el eje del motor, lo que provocaría un mal funcionamiento del mismo por el esfuerzo mecánico.

El carro del eje Z consiste en la pieza lateral (B) y su respectiva pieza opuesta, que contiene perfiles de aluminio montados junto con cojinetes para sujetarse al carril del eje Z, sobre la pieza (K) del carro del eje Y. Estas piezas están unidas a través de dos piezas de madera intermedias con agujeros al centro, sobre los cuales se pasan tornillos M4 de 25 mm de largo y se aprietan con tuercas. Sobre la pieza frontal (Q) se realiza el montaje de la herramienta rotativa (J) para los trabajos a realizar en la máquina fresadora.

La herramienta rotativa es montada sobre la superficie de la pieza intermedia (Q) entre las piezas (B) y su opuesta, y es sujetada utilizando tornillos M4 de 80 mm de largo que pasan a través de dos piezas metálicas, (F) y (G) que mantienen a la herramienta en una posición fija respecto del carro del eje Z. Este diseño permite que se pueda retirar la herramienta rotativa para propósitos de mantenimiento o utilización en otras tareas.

En el diagrama de la figura 20 se muestra la pieza intermedia trasera (E), entre la pieza (B) del diagrama de la figura 19 y la pieza opuesta, donde se utiliza una pieza metálica (B), que se sujeta al carro del eje Z a través de tornillos M4 de 20 mm de largo y tuercas. Esta pieza metálica sujeta a la tuerca de 1/4 in de 22 mm de largo sobre la cual pasa la varilla roscada en el carro del eje Z.

Figura 20. **Vista trasera del carro del eje Z**



Fuente: elaboración propia, utilizando el programa Adobe Photoshop.

2.1.6. Posicionamiento de los interruptores de límite

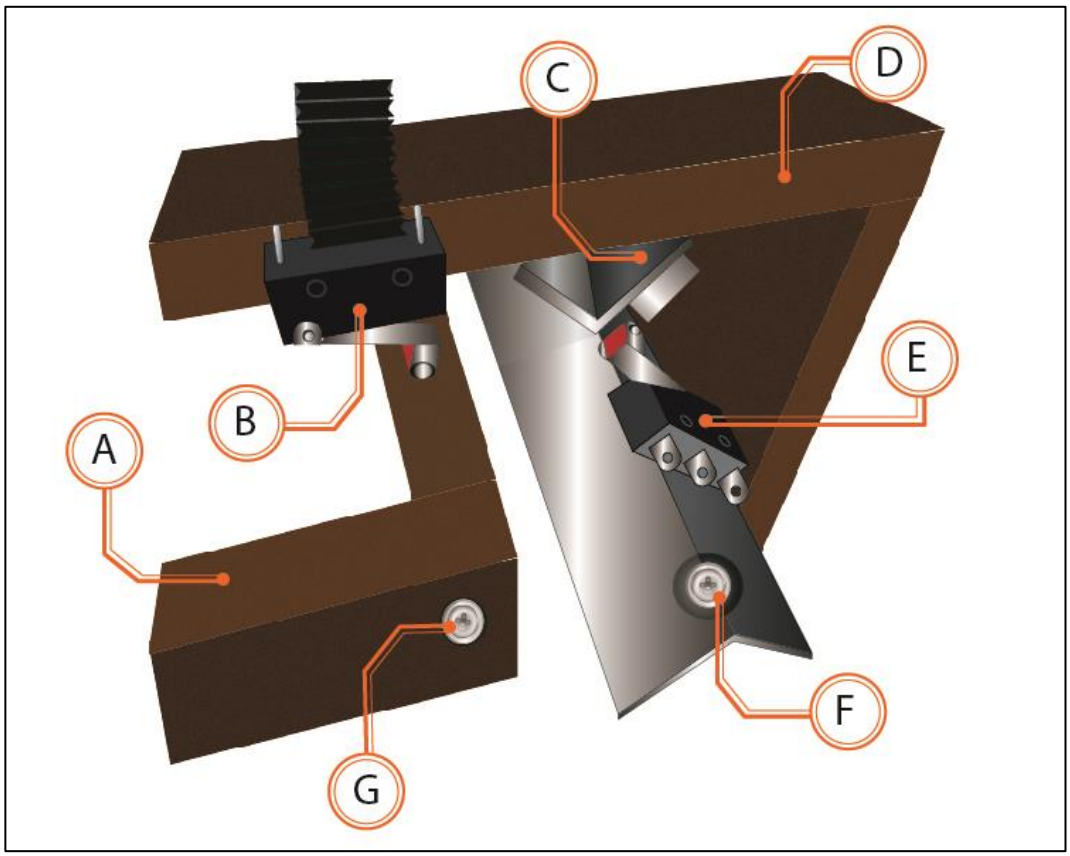
Los interruptores de límite son interruptores mecánicos que envían una señal al equipo electrónico de control, indicando que el carro en uno de los ejes de la máquina ha llegado a su máximo desplazamiento y debe detenerse, de lo contrario la máquina podría averiarse. Estos interruptores están situados en los extremos de los carriles para cada eje en la máquina fresadora.

Estos interruptores de límite son muy útiles para posicionar la máquina herramienta en una posición predeterminada. Son utilizados por las herramientas de software para encontrar un punto de referencia de la máquina. Los programas ejecutan secuencias de movimiento hasta que se obtengan las señales de los límites, indicando que se ha llegado al final, o al inicio del recorrido en uno, o en todos los ejes.

En el diagrama de la figura 21 se muestra la disposición de uno de los interruptores de límite para el eje Y y el eje Z. El carro del eje Y, indicado como (D) sostiene a uno de los interruptores de límite (B) para el eje Z, mientras el carril del eje Y, indicado como (F) sostiene el otro interruptor de límite (E) para el eje Y. Estos interruptores fueron colocados en la máquina fresadora utilizando tornillos busca rosca y pegamento, ya que no son sometidos a grandes esfuerzos mecánicos.

Cuando el carro del eje Y se desplaza en dirección hacia el interruptor límite mostrado en la figura 21, conforme el perfil de aluminio (C) con los cojinetes se acerca al interruptor (E), llega a un punto donde el interruptor es empujado por el perfil, y finalmente activa el interruptor, lo que señaliza al control electrónico a detener el movimiento en el eje Y. De la misma forma, cuando el carro (A) del eje Z se desplaza hacia arriba, eventualmente presiona al interruptor (B), señalizando al control electrónico a detener el movimiento del eje Z.

Figura 21. **Posicionamiento de los interruptores de límite**



Fuente: elaboración propia, utilizando el programa Adobe Photoshop.

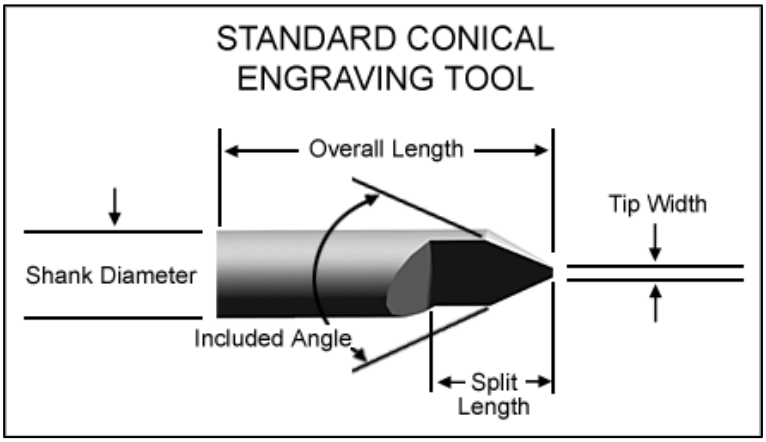
2.2. Herramientas de fresado y barrenado

Para la fabricación de PCBs utilizando la máquina fresadora es necesario colocar en la herramienta rotativa una fresa de grabado en V para remover el exceso de cobre del circuito. Para el barrenado de los circuitos impresos se utiliza una broca de 1/32 in de diámetro, generalmente de carburo de tungsteno.

En la figura 22 se muestran las características más generales de una fresa de grabado en V. Las fresas más utilizadas para grabado de PCBs son

de carburo de tungsteno 1/8 in de diámetro y de 30º, 45º y 60º, que es el ángulo de corte en la punta de la fresa.

Figura 22. **Fresa de corte para grabado en PCB**



Fuente: *Standard conical engraving tools*.
https://www.2linc.com/images/conical_layout.gif.
Consulta: 4 de abril de 2015.

El ancho de la punta es generalmente de 0,1 mm para las fresas mencionadas anteriormente. Este valor es útil en la configuración utilizada para el fresado de un PCB porque representa el grosor de la herramienta que remueve el cobre. Algunas fresas no presentan un ancho de la punta, y son completamente puntiagudas. Para calcular el ancho del grabado w es necesario realizar un cálculo de acuerdo con la siguiente ecuación:

$$w = 2\,z\,\tan\left(\frac{\theta}{2}\right)$$

Donde z es la profundidad de grabado y θ es el ángulo de apertura de la fresa de grabado.

3. DISEÑO E IMPLEMENTACIÓN DEL EQUIPO ELECTRÓNICO

Se muestra el diseño y los componentes del equipo electrónico del proyecto, conformado por una fuente de alimentación, una interfaz de control y tres controladores de motores paso a paso. Este equipo se encarga de controlar las señales eléctricas para mover la máquina herramienta de forma automatizada.

3.1. Diseño del controlador de motores paso a paso con CI L297 y L298

Para el control de cada uno de los motores de la máquina fresadora es necesario utilizar un controlador individual, que sirva como una interfaz más simple para controlar los motores, así el sistema intérprete de control numérico no debe preocuparse de las secuencias de los motores. Para esto se diseñó y construyó un controlador de motores paso a paso utilizando la pareja de circuitos integrados L297 y L298. El diseño completo puede consultarse en la sección de anexos. En esta sección se describirá el funcionamiento del controlador diseñado, refiriéndose a partes del diagrama presentado en los anexos.

El CI L297 integra toda la circuitería necesaria para controlar motores paso a paso bipolares y unipolares. Este integrado genera las señales de fase necesarias para manejar los motores, y estos pueden ser manejados en secuencia de medio paso, paso completo y con la secuencia *wave drive*.

Además, permite la utilización de un circuito controlador de corriente integrado, que funciona a través de PWM (modulación de ancho de pulso).

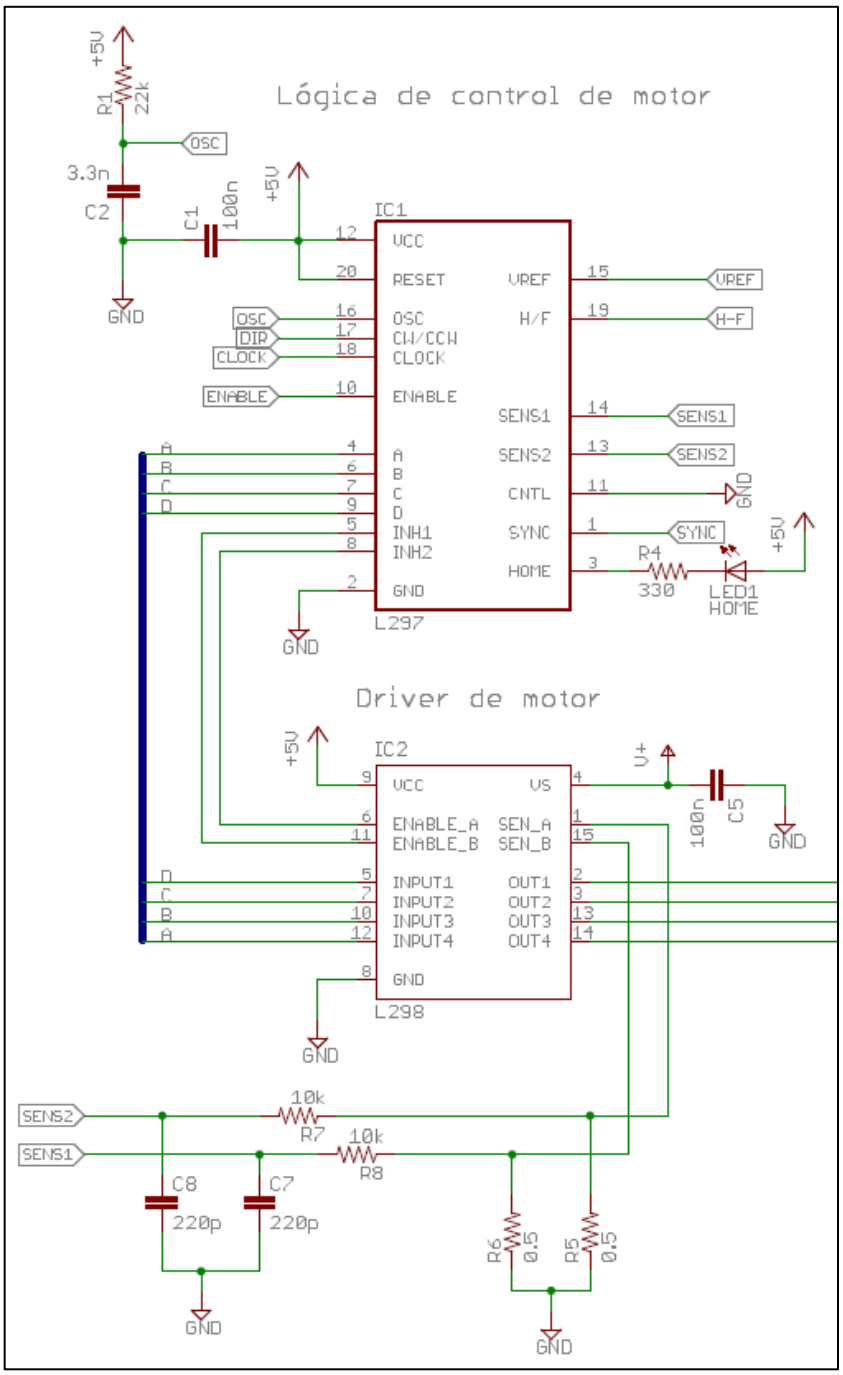
El CI L298 es un circuito integrado monolítico de 15 pines que sirve como controlador de puente completo (2 puentes H) de alto voltaje y un manejo de corriente de hasta 2A por puente H, diseñado para aceptar niveles lógicos de voltaje TTL y cargas inductivas como relés, solenoides, motores de corriente directa y motores paso a paso.

3.1.1. Diseño y explicación del circuito

El CI L297 es generalmente utilizado con un controlador con doble puente H (como el que provee el CI L298) para formar una interfaz completa de microprocesador a motor paso a paso. El controlador puede utilizarse con un motor unipolar a través de un arreglo de cuatro transistores Darlington. Para el diseño del controlador desarrollado se decidió utilizar la pareja L297 y L298, ya que esta combinación requiere muy pocos componentes adicionales, con lo que se reduce el costo y el espacio requerido para el controlador. Además, la pareja de circuitos integrados facilitan el desarrollo de *software*, ya que proveen una interfaz de *hardware* más simple.

En el diagrama esquemático de la figura 23 se muestra la conexión del diseño propuesto utilizando la pareja de circuitos integrados L297 y L298. Como se muestra en la figura, el CI L297 sirve como un elemento de lógica de control del motor paso a paso y el CI L298 el elemento que interactúa directamente con el motor.

Figura 23. **Diseño del controlador de motores, conexión de L297 y L298**



Fuente: elaboración propia, utilizando el programa EAGLE.

El CI L297 consta de 20 pines, algunos tienen funciones analógicas y otros pines desempeñan funciones digitales. A continuación se describe la función de cada uno de los pines y las conexiones realizadas en el diagrama.

- Pines 12 y 2 de alimentación VCC y GND: el circuito integrado trabaja con un voltaje de alimentación de 5V, conectados entre estos dos pines. Además se agregó el capacitor C1 para el desacople de señales de alta frecuencia, que aparecen como ruido en la señal de alimentación.

- Pin 20 de RESET: este pin es activo en bajo (realiza su función cuando el voltaje en el pin es 0V) y reinicia la secuencia aplicada en el motor a su punto inicial. Está directamente conectado a un nivel lógico alto, pues no es necesario reiniciar la secuencia del controlador del motor.

- Pin 16 de OSC: sirve como entrada para el punto intermedio de conexión de un circuito RC conformado por la resistencia R1 y el capacitor C2. La constante de tiempo de este circuito RC, dada por $\tau = RC$ determina la frecuencia del oscilador chopper para el control de corriente integrado. La ecuación que determina la frecuencia del oscilador es $f = \frac{1}{0{,}69RC}$ de acuerdo con la hoja de datos del fabricante.

- Pin 17 de CW/CCW: es una entrada digital que sirve como control de dirección de la rotación del motor, tanto en sentido horario (*clockwise*) como en sentido antihorario (*counterclockwise*). La dirección de rotación física del motor también depende de la conexión de las bobinas; este pin controla la dirección de la secuencia de rotación.

- Pin 18 de CLOCK: es una entrada digital activa en bajo. Cuando la señal que maneja presenta un flanco de bajada la secuencia del motor avanza un incremento. El paso ocurre en el flanco de subida de la señal.

- Pin 15 de VREF: este pin sirve como entrada analógica para configurar el voltaje de referencia del circuito controlador de corriente tipo *chopper*. Un voltaje aplicado en este pin determina la corriente pico del motor. El voltaje que debe aplicarse depende de dos valores: el valor de la corriente con la que se desea manejar el motor y del valor de las resistencias sensoras de corriente (R5 y R6). La ecuación para encontrar el voltaje a configurar en este pin es: $V_{ref} = I_{motor} R_{sensora}$

- Pin 19 de H/F: este pin es una entrada digital que sirve para controlar el tipo de secuencia de pasos aplicada al motor. Cuando la entrada es alta configura al CI L297 con la secuencia de medio paso, y cuando es baja lo configura con la secuencia de paso completo.

- Pines 13 y 14 (SENS1 y SENS2): estos pines sirven como entradas analógicas para los voltajes generados en las resistencias sensoras de corriente. Cuando la corriente retorna hacia tierra a través de las resistencias R5 y R6 genera un pequeño voltaje a través de estas. Estos voltajes son filtrados a través de una red RC conformada por las parejas R7 y C8 con R8 y C7. Finalmente, las señales de voltaje filtradas se conectan a los pines SENS1 y SENS2, donde son comparadas internamente con el voltaje de referencia para determinar la conmutación del control de corriente tipo *chopper*.

- Pin 11 de CNTL: este pin sirve como una entrada digital que define la acción del control *chopper*. Cuando la entrada es baja, el controlador

chopper actúa sobre los pines INH1 e INH2; cuando la entrada es alta, el controlador *chopper* actúa sobre las líneas de fase del motor (ABCD). Es por esto que el pin está directamente conectado a tierra, ya que se desea que el controlador *chopper* actúe sobre las líneas INH1 e INH2, que finalmente controlan la habilitación de los puentes H en el CI L298.

- Pin 1 de SYNC: es una salida del oscilador *chopper* en el CI L297. Este pin es utilizado cuando se quiere sincronizar la conmutación del oscilador *chopper* en varios circuitos integrados L297. En el diagrama de la figura 23 aparece conectado a un punto llamado SYNC en el diagrama completo; como se verá más adelante, este punto está conectado hacia el puerto de conexión del controlador con la interfaz.

- Pin 3 de HOME: es una salida de colector abierto que indica cuando el CI L297 se encuentra en el estado inicial de la secuencia de pasos. En el diagrama se le conectó la resistencia R4 y el LED1 como indicador visual de la secuencia aplicada al motor paso a paso.

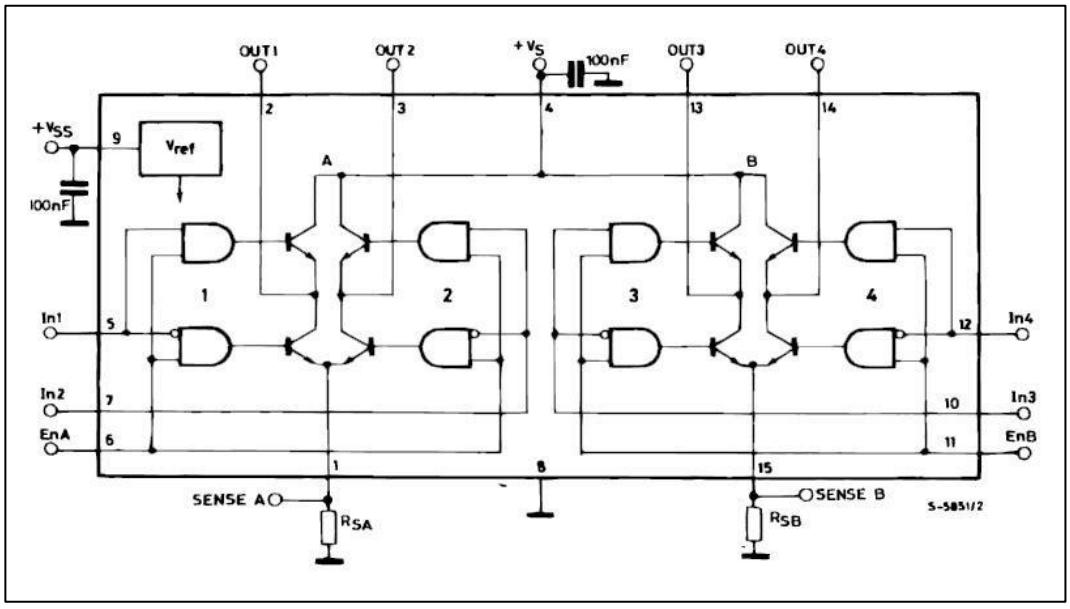
- Pines 4, 6, 7 y 9 (A, B, C, D): sirven como salidas digitales para controlar cada una de las fases del motor en la etapa de potencia. Están conectados a las entradas de control de los puentes H integrados en el CI L298 para manejar las polaridades y combinaciones con las que deben ser activadas las bobinas del motor.

- Pines 5 y 8 (INH1 e INH2): son salidas digitales activas en bajo que sirven para inhibir la etapa de control para las bobinas del motor. Estas salidas son controladas por el controlador *chopper* para regular la corriente, ya que la conexión de CNTL indica un estado lógico bajo. Están conectados directamente hacia el control de habilitación del CI

L298 para activar y desactivar los puentes H controladores de las bobinas del motor, y de esta forma regular la corriente que pasa a través de las mismas.

Para entender la conexión realizada en el diagrama de la figura 23 respecto del CI L298 es necesario revisar el diagrama de bloques interno del circuito integrado. Como se muestra en la figura 24, el CI L298 está conformado internamente por elementos de lógica digital que sirven para controlar los puentes H que manejan finalmente las bobinas del motor.

Figura 24. **Diagrama de bloques del CI L298**



Fuente: *L298 Dual full-bridge driver*.
http://www.st.com/web/en/resource/technical/document/datasheet/CD00000240.pdf.
Consulta: 9 de enero de 2015.

El CI L298 maneja dos fuentes de alimentación, una que sirve para la lógica digital del circuito integrado, indicada como Vss en el diagrama de la figura 24. El valor máximo de esta fuente puede ser de hasta 7 VDC, pero que se decidió utilizar un valor estándar de 5 VDC para la fuente de alimentación de la lógica, al igual que para el CI L297.

La otra fuente de alimentación, cuya referencia de voltaje es la misma que el voltaje de alimentación para la lógica, sirve para alimentar las bobinas del motor, es por esto que está internamente conectada a la unión de los colectores de cada pareja de puentes H en el diagrama de la figura 24, y se identifica como Vs. Esta fuente de alimentación debe ser desacoplada de señales de alta frecuencia que puedan interferir en la lógica de funcionamiento del CI L298, es por esto que se conectó un capacitor de 100 nF en el pin de alimentación en el diagrama de la figura 23. Además, se debe cuidar de que este capacitor esté ubicado físicamente lo más cerca posible del circuito integrado en el diseño de la placa de circuito impreso del controlador.

Las entradas identificadas en el diagrama de la figura 24 como A, B, C y D son las entradas digitales que sirven para conmutar de distintas formas los transistores que conforman los dos puentes H. Asimismo, estas entradas corresponden a las indicadas como INPUT1, INPUT2, INPUT3 e INPUT4 en el diagrama de la figura 23, y son controladas por las salidas del controlador lógico en el CI L297.

Las entradas de habilitación, indicadas como INH1 e INH2 en el diagrama de la figura 24 están indicadas como ENABLE_A y ENABLE_B en el diagrama de la figura 23. Como se explicó anteriormente, controlan la activación y desactivación de los puentes, y observando el diagrama de bloques del CI L298 es claro que la activación de los transistores no solo está sujeta a los valores de

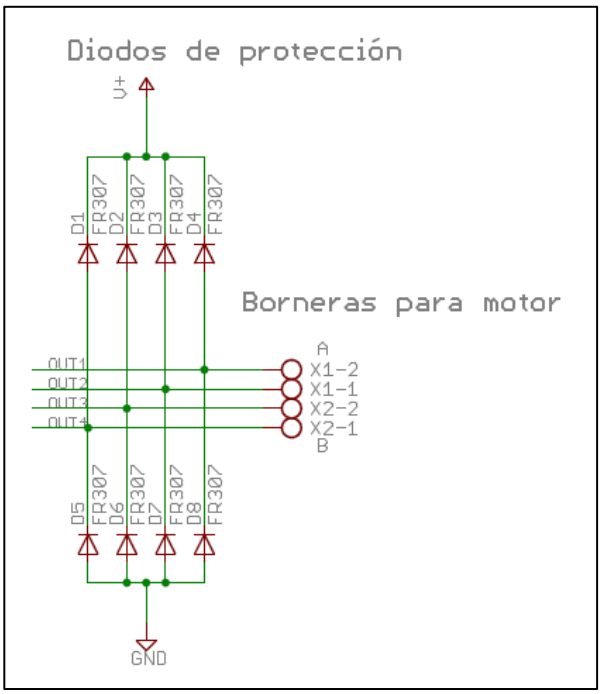
las entradas para la secuencia de los puentes H, sino que también depende de estos dos pines que son operados lógicamente con las entradas de secuencia.

Los dos pines de salida SENSE1 y SENSE2 indicados en la figura 24 están indicados en el diagrama de la figura 23 como SEN_A y SEN_B, y corresponden a la unión de los emisores de dos de los transistores que conforman cada puente H en el CI L298. Es por esto que estos pines están conectados a tierra a través de la resistencia sensora de bajo valor, indicada en el diagrama propuesto como R5 y R6.

En cuanto a las salidas del controlador para conectar el motor, las salidas de los dos puentes H, que corresponden a los pines indicados como OUT1, OUT2, OUT3 y OUT4 en los diagramas mostrados son conectados hacia una bornera en el diseño de la placa de circuito impreso, que servirá para atornillar los cables de los motores hacia el controlador, como se muestra en el diagrama de la figura 25.

Además, se agregó al diseño el conjunto de diodos de rápida recuperación que están polarizados normalmente de forma inversa, pero que permiten la recirculación de corriente hacia la fuente de alimentación, haciendo que la corriente a través de las bobinas del motor decaiga rápidamente.

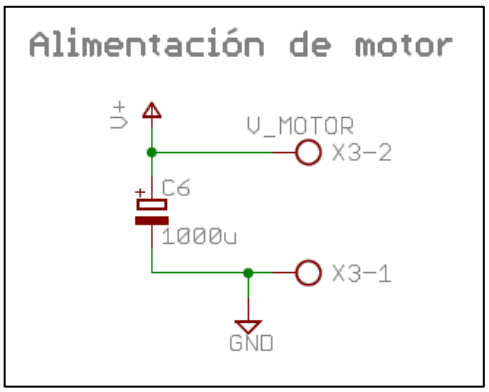
Figura 25. **Diseño del controlador de motores, conexión de salida**



Fuente: elaboración propia, utilizando el programa EAGLE.

Para la entrada de alimentación del motor del controlador también se utilizó una bornera, como se muestra en el diagrama de la figura 26. Esta conexión de entrada cuenta con un capacitor electrolítico, indicado en la figura como C6 de 1000 microfaradios. Este capacitor sirve como respaldo para la fuente de alimentación del motor, ayudando a mantener la señal de voltaje lo más constante posible, y desacoplando señales de alta frecuencia. Finalmente, este capacitor electrolítico está especificado para un voltaje máximo de 50 VDC, de acuerdo con las especificaciones del CI L298 para el voltaje máximo de alimentación del motor.

Figura 26. **Diseño del controlador de motores, alimentación del motor**



Fuente: elaboración propia, utilizando el programa EAGLE.

3.1.2. Ajuste de corriente del controlador

El diseño del controlador permite el ajuste de la corriente que se desea hacer circular por las bobinas del motor, esto se logra cambiando el voltaje de referencia para el controlador de corriente tipo *chopper* integrado en el CI L297, que se encuentra en el pin 15 del circuito integrado.

El ajuste de la corriente para el motor se hace cambiando el voltaje de referencia en el pin 15, el cual tiene conectado el pin común de un potenciómetro de precisión de 1, indicado como R3 en el diagrama de la figura 27. Este divisor de voltaje permite ajustar con precisión valores de voltaje en el rango de los milivoltios. Finalmente se agrega el capacitor C4 de 1 μF para mantener el voltaje constante en ese pin.

El valor de voltaje al cual se debe ajustar el potenciómetro puede medirse con un multímetro en el *jumper* JP1 en la placa de circuito impreso. Este depende del valor de las resistencias sensoras de corriente R5 y R6 en el

diagrama de la figura 23. También depende del valor de corriente que se desea hacer circular por los motores paso a paso. Para el caso específico de la máquina construida, los motores paso a paso utilizados fueron de un voltaje nominal de 24 V para una corriente nominal de 400 mA.

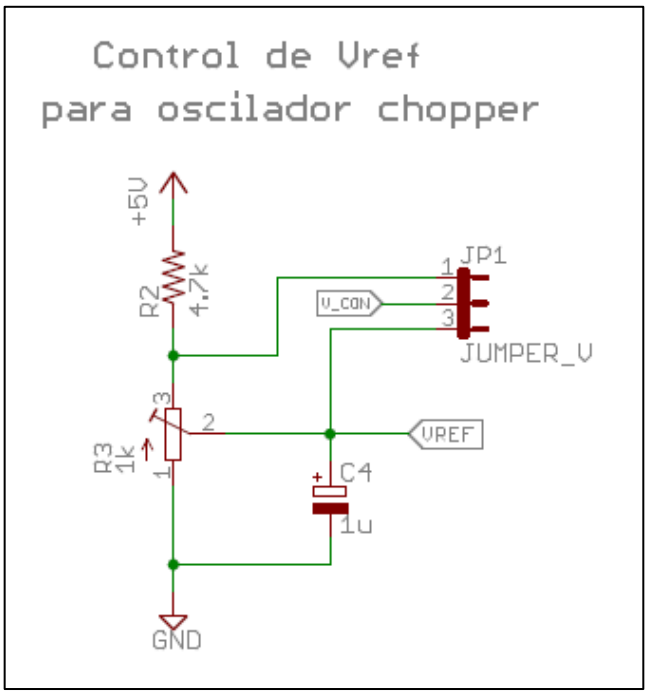
Sin embargo, no se trabajó utilizando la corriente máxima del motor, ya que el calentamiento se volvía excesivo a pesar de tener el máximo torque disponible, se trabajó utilizando un 70 % de la corriente máxima, que permitió un torque suficiente para el movimiento de la máquina y redujo notablemente el calentamiento a un 49 % (dado que la temperatura es proporcional a la potencia disipada en el motor, y la potencia disipada es proporcional al cuadrado de la corriente utilizada). Por lo tanto, se trabajó utilizando un valor de corriente de 280 mA por fase del motor paso a paso.

El cálculo del voltaje de referencia es como sigue: ya que la corriente genera un voltaje cuando pasa a través de las resistencias R5 y R6 en el diagrama de la figura 23, el voltaje máximo en estas resistencias se da cuando pasa la corriente máxima del motor, y está dado por la siguiente ecuación:

$$V_{ref} = R_{sensora}\, I_{motor}$$
$$V_{ref} = (0{,}5\Omega)(280mA) = 140\, mV$$

De esta forma se configuró cada uno de los motores de la máquina, haciendo un pequeño ajuste en la ecuación para el controlador del eje Z de la máquina, ya que es el eje que requiere mayor esfuerzo mecánico para moverse, por lo que se ajustó la corriente para trabajar a un 80 % del máximo, lo que aumenta el calentamiento pero también incrementa el torque del motor.

Figura 27. **Diseño del controlador de motores, ajuste de corriente**



Fuente: elaboración propia, utilizando el programa EAGLE.

3.1.3. Configuración de frecuencia del oscilador *chopper*

De acuerdo con la hoja de datos del CI L297 es posible ajustar la frecuencia del oscilador del control de corriente tipo *chopper*. Esto se hace escogiendo valores para una red RC dentro del circuito. La resistencia se conecta hacia el voltaje de alimentación y el capacitor hacia tierra. El punto entre la resistencia y el capacitor se conecta al pin 16 del CI L297.

La frecuencia del oscilador *chopper* está dada por la siguiente ecuación:

$$f_c = \frac{1}{0{,}69\ RC}$$

Utilizando los valores mostrados en el diagrama de la figura 23 se obtiene la siguiente frecuencia de conmutación para el circuito de control de corriente:

$$f_c = \frac{1}{(0{,}69)(22k\Omega)(3.3nF)} = 19{,}96kHz$$

Esta es la frecuencia de conmutación para los dos circuitos de control de corriente integrados en el CI L297 utilizando los valores recomendados por el fabricante para el circuito oscilador. Esta frecuencia de oscilación permite modular la corriente en las bobinas del motor paso a paso, con el fin de mantener una corriente promedio controlada por el voltaje de referencia del control *chopper*.

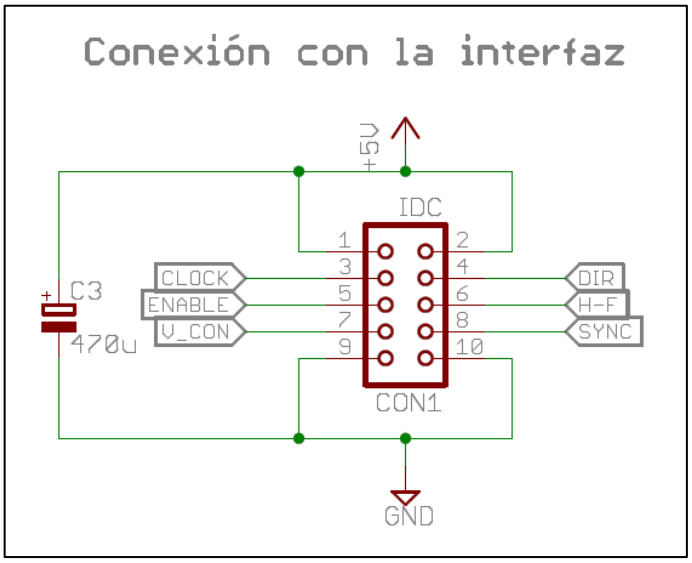
3.1.4. Conexiones del controlador

La conexión del controlador de motores paso a paso hacia la tarjeta de control de motores se realiza a través de un cable de tipo IDC-10, que recibe la alimentación para la lógica del controlador. Para la alimentación del motor, como se explicó en la sección de diseño del circuito, se utilizaron borneras para atornillar los cables de alimentación y los cables de los motores hacia la placa de circuito impreso del controlador.

La disposición de los pines del cable IDC-10 se muestra en el diagrama esquemático de la figura 28. Este conector está conformado por las señales más importantes para el control del motor paso a paso conectado al controlador. La interfaz de control, cuyo diseño y explicación se verá en la siguiente sección, se comunicará con los controladores individuales de motores utilizando las señales de CLOCK, DIR, ENABLE y H-F, como se muestran en la figura 28.

Con la señal de CLOCK (reloj), la interfaz de control señaliza al controlador en cada paso del motor por dar, con la señal DIR (dirección) la interfaz indica la dirección de rotación del motor y con la señal de ENABLE (habilitación) desactiva totalmente las bobinas del motor paso a paso para ahorrar energía o liberar el torque de retención. La señal H-F (*half/full)* sirve para seleccionar si se utilizará la secuencia de medio paso o la secuencia de paso completo. Finalmente, las señales de V_CON y SYNC sirven para realizar las mediciones correspondientes a cada controlador de motor sobre la interfaz de control.

Figura 28. **Diseño del controlador de motores, conector del controlador**



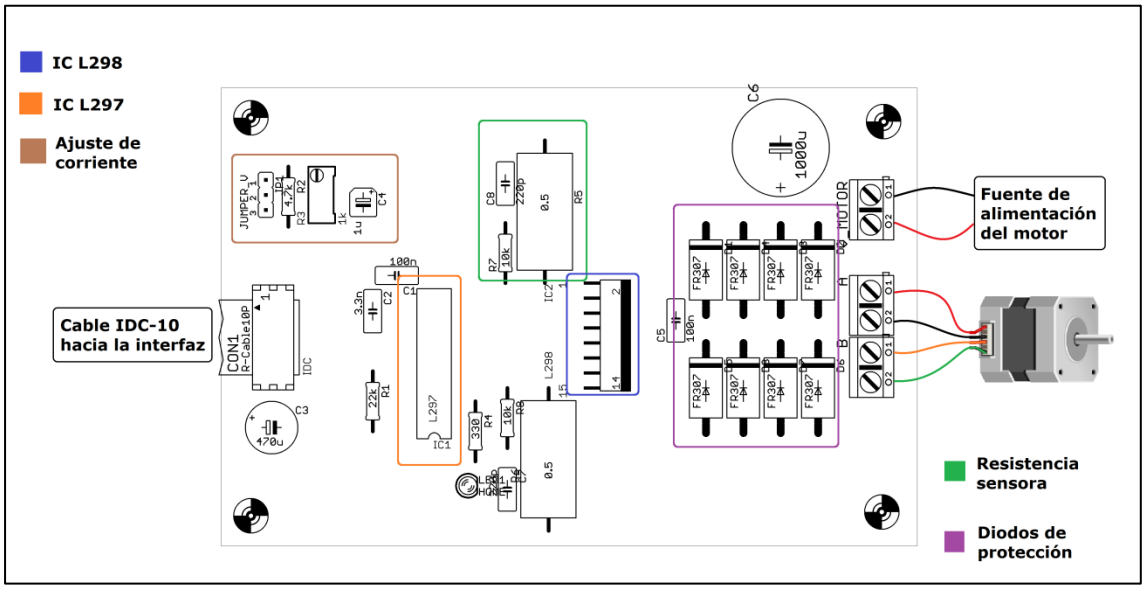
Fuente: elaboración propia, utilizando el programa EAGLE.

En la figura 29 se puede ver un diagrama de conexiones utilizando la disposición de la placa de circuito impreso que se diseñó para el controlador individual de motores paso a paso. Las señales de control se ubican del lado izquierdo de la figura, el conector IDC-10 va hacia la interfaz de control, y arriba

del conector de señales se encuentra el potenciómetro para ajustar la corriente del controlador. Del lado derecho de la figura se ubican las borneras de conexión para la alimentación del motor y para la conexión de las bobinas del motor.

Además, se muestran algunos de los componentes importantes que componen el controlador del motor, tales como los circuitos integrados que manejan la lógica y las señales de las bobinas, los diodos de rápida respuesta y las resistencias sensoras del controlador.

Figura 29. **Conexión del circuito controlador de motores paso a paso**



Fuente: elaboración propia, utilizando el programa EAGLE.

3.1.5. Potencia del controlador

El controlador está diseñado para un voltaje de alimentación del motor máximo de 50 V, y una disipación máxima de potencia de 25 W con una temperatura de la carcasa del circuito integrado de hasta 75 ºC, de acuerdo con la hoja de datos del CI L298.

Esto da como resultado corrientes máximas en las bobinas de hasta 2 A. Sin embargo, estos valores máximos están dados para corrientes pico no pulsantes, y debido a la utilización de la técnica de control de corriente, la disipación de potencia es menor. Además, para la implementación de cada controlador se instaló un disipador de calor de aluminio en el CI L298, lo que no permite que se eleve demasiado la temperatura en dicho controlador.

Finalmente, no se trabajó con la corriente máxima permisible del circuito integrado, ya que la corriente nominal de los motores paso a paso utilizados es de 400 mA, y con el control de corriente se trabajó con valores entre el 60 % y el 80 % de esta corriente máxima para los distintos controladores de motores implementados, dependiendo del eje que controlan en la máquina fresadora, ya que debido a la construcción, algunos motores necesitan una corriente mayor o menor para desplazar los ejes la máquina fresadora.

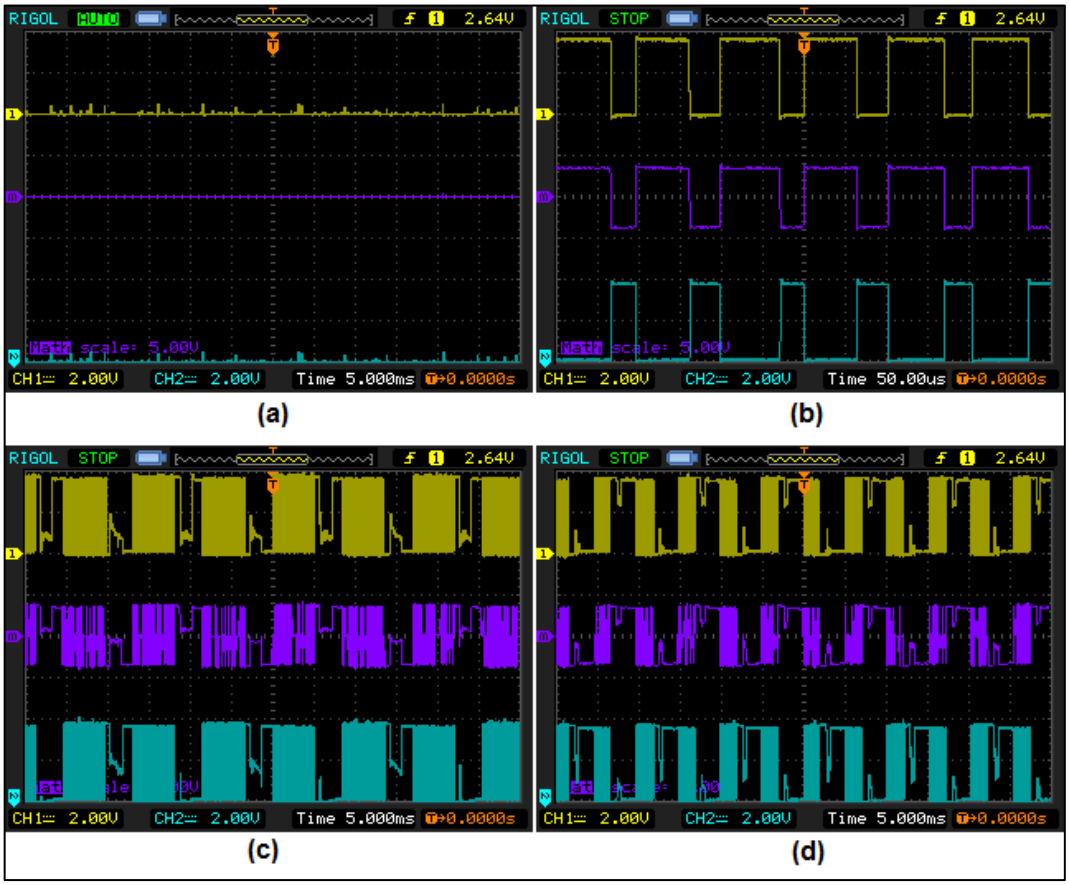
3.1.6. Formas de onda del controlador

Utilizando un osciloscopio Rigol DS1105E se observaron las formas de onda del controlador de motores bajo distintos esquemas de funcionamiento. Las formas de onda observadas fueron registradas y se muestran en la figura 30. En la figura se puede observar en amarillo la señal de voltaje en uno de los extremos de una de las bobinas del motor referida a tierra. La señal de voltaje

en el otro extremo se muestra en color celeste, mientras que la diferencia, que es el voltaje en la bobina del motor se muestra en color morado.

En la figura 30a se observa la forma de onda a través de la bobina del motor cuando este se encuentra detenido y el controlador desactiva el control de corriente para la bobina a través del pin de ENABLE, por lo que el motor se puede mover libremente. En la figura se puede observar que la corriente a través de la bobina es cero.

Figura 30. **Formas de onda del controlador de motores paso a paso**



Fuente: elaboración propia, utilizando osciloscopio Rigol DS1105E.

En la figura 30b se observa la acción del control de corriente *chopper*, el cual mantiene polarizada la bobina mientras el motor se mantiene detenido. En este caso la señal de voltaje a través del motor indica que este está siendo polarizado con una señal de voltaje bipolar, lo que hace que la corriente aumente y disminuya para mantener una corriente promedio en la bobina y generar el torque de retención del motor.

En la figura 30c se observa la señal de voltaje en la bobina del motor cuando este se encuentra moviéndose a una velocidad de 65 mm/min mientras desplaza uno de los ejes de la máquina. Asimismo, en la figura 30d se muestra la señal de voltaje mientras el motor se mueve a velocidad máxima.

3.2. Diseño de la interfaz de control

La interfaz de control se refiere al componente electrónico que se utiliza para intercomunicar la computadora y los controladores de motores paso a paso. Esta interfaz maneja todas las señales necesarias para cada controlador de motor (paso, dirección y habilitación), permitiendo escoger la secuencia del motor a aplicar (paso completo o medio paso). Además, maneja las señales correspondientes a los interruptores del límite de la máquina para indicar al *software* en caso de sobrepasar los límites de desplazamiento de alguno de los ejes. Por último, maneja las señales correspondientes a la activación de las herramientas, en este caso la activación de la herramienta rotativa para realizar los trabajos de fresado en la máquina.

En esta sección se describen y explican dos diseños diferentes de la interfaz de control, cada uno corresponde a un esquema distinto de utilización de la computadora en el proceso de mecanizado de las piezas. Los diseños completos de ambas interfaces de control se encuentran en la sección de

anexos, en esta sección se utilizarán partes de los diseños para una descripción y explicación más ordenada.

Uno de los diseños consiste en la utilización del puerto paralelo de la computadora. Este puerto utiliza un conector DB-25 y consiste en un bus de comunicación de 8 bits de datos y un conjunto de líneas de entrada.

La interfaz de control diseñada para este puerto utiliza un esquema de control donde la computadora es la encargada de interpretar el código G para el mecanizado de las piezas, y envía las señales procesadas necesarias para mover cada uno de los motores en la máquina y manejar las herramientas a través del puerto paralelo. Este es el esquema de conexión a máquinas CNC industriales y que requieran un control más sofisticado para el mecanizado de las piezas, ya que las herramientas de *software* disponible permiten al usuario interactuar con la máquina de una forma más flexible.

El otro diseño de la interfaz de control realizado consiste en la aplicación de una plataforma de *hardware* abierto basada en Arduino, que es una plataforma de desarrollo de proyectos electrónicos muy popular. Consiste en un microcontrolador de la familia Atmel, programado y depurado a través de una tarjeta de desarrollo donde es posible conectar más periféricos. La ventaja de esta plataforma es que la intercomunicación con la computadora es posible a través de una consola serial virtual, utilizando una conexión USB estándar.

Sobre esta plataforma se carga un proyecto de código abierto llamado GRBL, que es una alternativa de bajo costo a la utilización de un puerto paralelo para el control de movimiento de una máquina fresadora CNC.

GRBL consiste en un intérprete de código G de alto desempeño, que corre en una tarjeta de desarrollo Arduino, es decir, utilizando este esquema la computadora se encarga de enviar el código G a la tarjeta de desarrollo Arduino y esta se encarga de la interpretación y generación de las señales necesarias para mover los motores paso a paso y controlar las herramientas y los límites de la máquina. En el capítulo de *software* será descrita la utilización de GRBL con mayor detalle.

3.2.1. Diseño de la interfaz de control por puerto paralelo

La interfaz de control consiste en un circuito electrónico que maneja las señales provenientes del puerto paralelo de la computadora para manejar los controladores de motores individuales y las herramientas de la máquina fresadora. Entre las capacidades de esta plataforma se encuentran: la retransmisión de las señales de pasos, dirección y habilitación provenientes de la computadora hacia cada motor, la activación de herramientas a través de circuitos con relés, y el acople y envío de señales de interruptores de límites de la máquina hacia la computadora.

Esta interfaz utiliza el puerto paralelo de la computadora, mejor conocido como el puerto de impresora, que consiste en un conector DB-25 (25 pines de conexión) ordenados en dos filas, el cual está definido como conector tipo A de acuerdo con el estándar IEEE 1284 que norma el método de señalización y utilización del puerto paralelo. En la tarjeta de interfaz de control también se utiliza el conector DB-25, para lo cual es necesario utilizar un cable de extensión de la computadora hacia la interfaz.

3.2.1.1. Capacidades y requerimientos de la interfaz de control

La interfaz de control por puerto paralelo es capaz de retransmitir las señales provenientes de la computadora hacia los controladores de motores individuales para mover los motores de la máquina fresadora CNC, es posible manejar hasta un máximo de tres ejes con el diseño presentado. También permite la activación de hasta dos herramientas utilizando un relé para proveer aislamiento del circuito a cada herramienta. Finalmente la interfaz de control es capaz de conectarse a cinco interruptores de límite para detener cualquier operación si la máquina fresadora sobrepasa un límite de desplazamiento permitido.

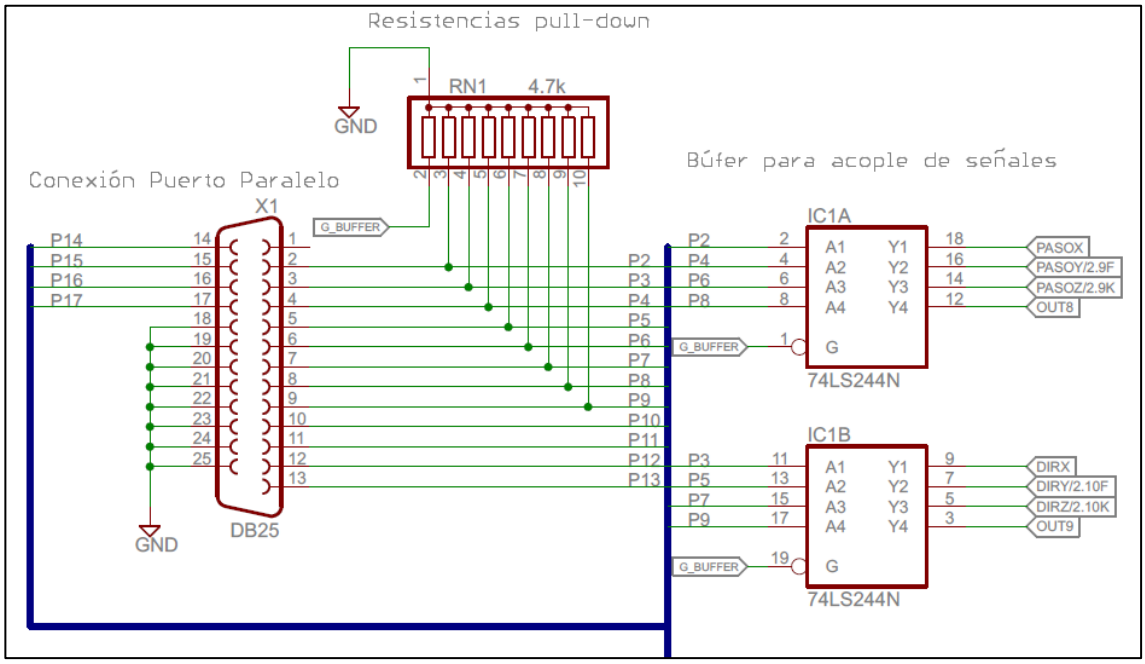
Con el diseño de interfaz se plantea la conexión de un controlador externo que permita manipular la máquina de forma manual, sin necesidad de la computadora. Sin embargo, como requerimiento para todas las demás operaciones, la interfaz de control necesita de una conexión hacia la computadora a través del puerto paralelo y de un *software* especializado para el manejo de las señales de control. Sin esta conexión, la interfaz de control no puede operar la máquina fresadora por sí misma.

La capacidad de procesamiento de códigos G para el control de la máquina fresadora depende directamente del *software* utilizado en la computadora para interpretar los programas de control numérico. Dependiendo del programa utilizado el estilo del código será uno u otro, pero en general, los programas permiten interpretar una amplia gama de códigos y realizar diversidad de movimientos.

3.2.1.2. Explicación del diseño

En el diseño de la interfaz de control se utiliza un conector DB-25 para enviar las señales de control hacia los motores paso a paso. La conexión de este conector hacia el circuito se muestra en la figura 31. El diseño del circuito incluye la utilización de un búfer de datos que permite el acople de señales hacia el resto del circuito, esto es para no cargar excesivamente el bus de datos del puerto paralelo, y para permitir un aislamiento de alta impedancia entre las señales de control hacia los motores y el puerto paralelo de la computadora.

Figura 31. **Conexión de puerto paralelo a interfaz de control**



Fuente: elaboración propia, utilizando el programa EAGLE.

El búfer utilizado es el CI 74LS244, que es un arreglo de 8 búfer triestado con un control de habilitación común hacia todos los búfer del circuito integrado.

Es generalmente utilizado como manejador de direcciones de memoria y como receptores y transmisores en sistemas basados en buses de datos. Este circuito integrado trabaja con un nivel lógico de voltaje de 5 V.

El conector DB-25 utiliza los pines 2 al 9 para la transmisión de datos necesarios para manejar los motores paso a paso; entre estas señales se encuentra la señal de paso, dirección y habilitación para cada uno de los motores. Como se observa en la figura 31, estas líneas del puerto paralelo son conectadas a las entradas del búfer y a una serie de resistencias *pull-down* hacia la tierra del circuito. Esto permite que las entradas de los búferes siempre tengan un nivel lógico definido, ya que de lo contrario, las señales que controlan los motores estarían flotantes (en un nivel indefinido) si el cable del puerto paralelo hacia la computadora está desconectado.

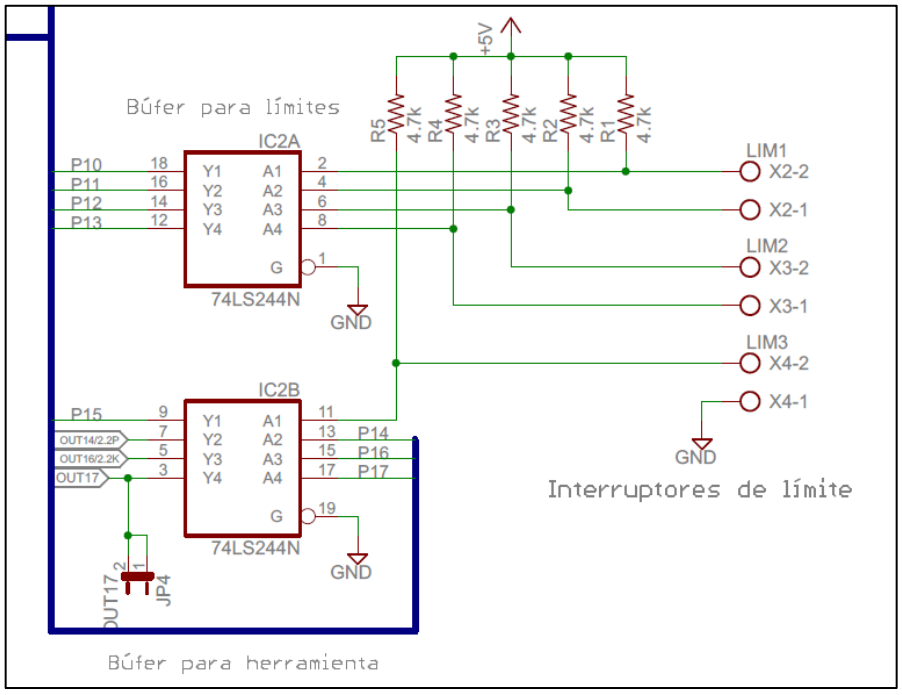
Los pines 10 al 17 del puerto paralelo son utilizados para el manejo de los interruptores de límite de la máquina y para la activación de la herramienta rotativa. Estos pines son conectados a un bus de datos, indicado con una línea gruesa en la figura 31. Y los pines 18 al 25 del conector son la referencia de voltaje del puerto paralelo, que se conecta a la tierra del circuito.

Como se muestra en la figura 32, los pines 10 al 13, y el pin 15 del puerto paralelo sirven como un bus de entrada hacia la computadora. En este bus se colocan los interruptores de límite, utilizando resistencias *pull-up* para definir el nível lógico por defecto para cada límite. Los finales de carrera deben utilizarse entre su terminal común y la terminal normalmente abierta, conectando ambas hacia los conectores en la tarjeta de control indicados por LIM1, LIM2 y LIM3.

Por defecto, el búfer de datos transmite un uno lógico hacia la computadora, indicando que el límite no está presionado. Cuando el límite es

presionado, este conecta una de sus terminales a tierra, y por tanto, conectando su respectiva resistencia *pull-up* a tierra, hace que sea transmitido un cero lógico hacia la computadora e indica que se ha excedido un límite físico en el recorrido de alguno de los ejes de la máquina.

Figura 32. **Conexión de límites y herramientas a puerto paralelo**



Fuente: elaboración propia, utilizando el programa EAGLE.

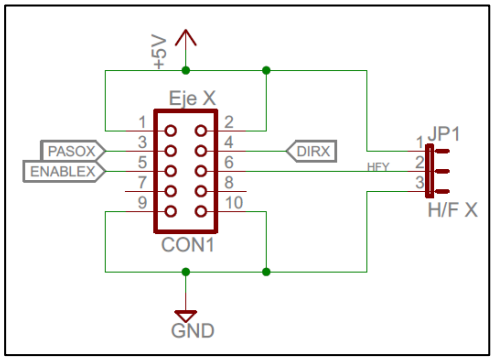
Los pines 14, 16 y 17 forman parte de un bus de salida de la computadora para el manejo de herramientas de la máquina CNC a través de la computadora. Estos pines son conectados también a un búfer para la retransmisión de las señales hacia los circuitos que manejan las herramientas. Para el control de habilitación de estos circuitos integrados búfer en relación con los límites y control de herramientas se conectaron directamente a tierra,

como se indica en la figura 32 a través de la conexión G de ambos bloques hacia GND.

La interfaz de control se conecta a cada uno de los controladores de motores paso a paso a través de una interfaz cableada con conectores IDC10. En la figura 33 se observa la conexión de salida para el controlador de motor del eje X. La interfaz de control posee tres de estos conectores, uno para cada controlador de motor. Este conector utiliza los pines 1 y 2 como la alimentación de la lógica del controlador de motor paso a paso, junto con los pines 9 y 10 para la referencia de voltaje. Los pines 3, 4 y 5 son las tres señales esenciales para el control de cada uno de los motores. El pin 6 es conectado a un *jumper* que permite escoger la secuencia de pasos que utilizará el controlador CI L297 del motor, ya sea de paso completo, o medio paso.

Los pines 7 y 8 se dejan desconectados del diseño de la interfaz, pero permiten realizar mediciones directamente sobre la interfaz y no sobre el controlador de motores, ya que están conectados a voltajes de referencia en el diseño del controlador de motor paso a paso individual.

Figura 33. **Conexión de salida hacia controladores de motores**



Fuente: elaboración propia, utilizando el programa EAGLE.

El circuito de alimentación de la interfaz de control se muestra en la figura 34. En este circuito se tienen dos etapas de regulación, una para obtener 5 VDC y otra para 12 VDC. El circuito fue diseñado para un voltaje de entrada de 26 VDC, ya que este voltaje de alimentación proviene del *tap* central del transformador utilizado para alimentar toda la circuitería de control de la máquina.

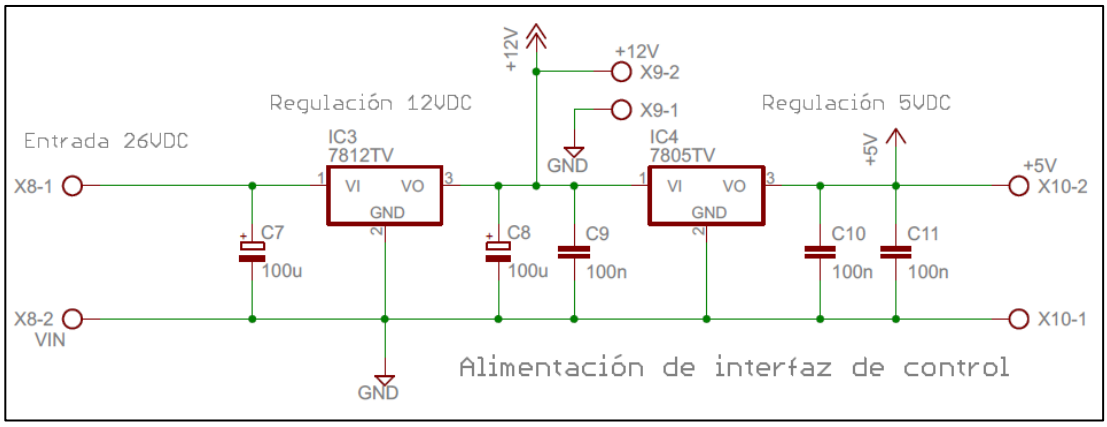
Para la regulación de 12 VDC se utilizó un regulador de voltaje de la serie CI LM7812, con un empaquetado TO-220 que posee tres terminales. Una terminal de voltaje de entrada a ser regulado, la terminal de conexión a tierra y la terminal de voltaje de salida regulado. En el diseño de la interfaz de control se incluyó una bornera de salida de voltaje como punto de conexión auxiliar para cualquier otro circuito que pudiera ser alimentado a futuro con 12VDC.

En serie al regulador de 12 VDC se conectó otro regulador de voltaje de la serie CI LM7805, también con un empaquetado TO-220, que permite la regulación de voltaje de 12 VDC a 5 VDC para la alimentación de la lógica de todos los circuitos de control. Ya que los reguladores utilizados en la interfaz de control presentan un empaquetado TO-220 se utilizó un disipador de calor de aluminio para montar ambos reguladores de voltaje, ya que estos, al ser de tipo lineal, disipan energía en el circuito interno con el fin de mantener constante el voltaje, y esto produce calentamiento de los empaquetados del circuito integrado.

El problema del calentamiento de los reguladores de voltaje es una desventaja que produce ineficiencia energética en la alimentación de todo el equipo electrónico. Sin embargo, la principal razón para utilizar este tipo de reguladores de voltaje es que son muy baratos y la circuitería externa a estos circuitos integrados es mínima.

En la siguiente sección, donde se describe el otro posible diseño para la interfaz de control se utilizará un regulador conmutado para la alimentación del circuito. Este tipo de diseño puede ser adaptado perfectamente al diseño de esta interfaz de control por puerto paralelo.

Figura 34. **Circuito de alimentación de interfaz de control por puerto paralelo**



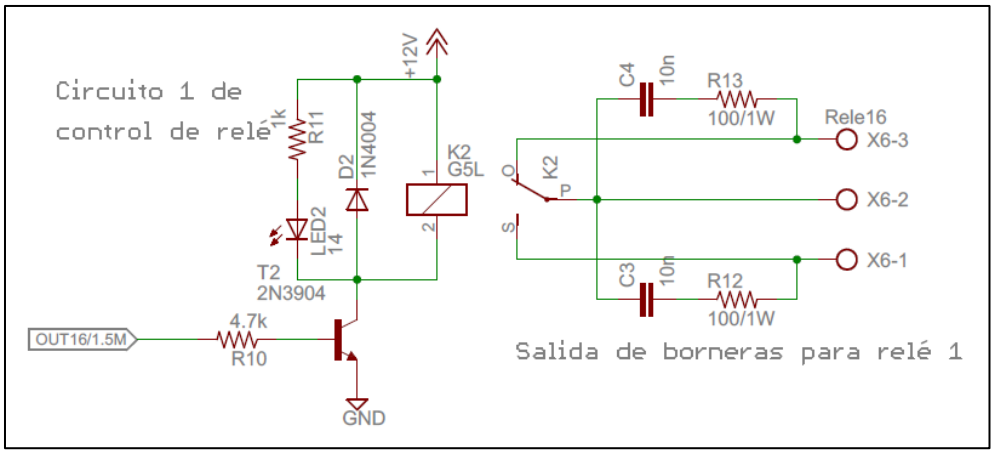
Fuente: elaboración propia, utilizando el programa EAGLE.

En cuanto a la activación de la herramienta rotativa, la interfaz de control utiliza un circuito de activación de un relé, controlado a través de los pines 14 y 16 del puerto paralelo. En la figura 35 se muestra la conexión de uno de los circuitos de relé, el mismo es activado a través de un transistor NPN 2N3904 en una configuración de colector abierto (donde la carga es conectada al colector, y el emisor del transistor a tierra).

En paralelo a la bobina del relé se conecta un diodo 1N4004 que normalmente estaría operando en polarización inversa cuando fluya corriente a través de la bobina. Este diodo permite descargar la energía almacenada en

forma de campo magnético en la bobina del relé, evitando así que se dañe el resto del circuito de activación del relé. Además se conecta una red de resistencia y diodo emisor de luz (led) en paralelo a la bobina del relé, de esta forma cuando se activa el relé hay un indicador visual de que la bobina permite la conexión de la terminal normalmente abierta hacia la terminal común.

Figura 35. **Circuito de activación de relé para control de herramienta**



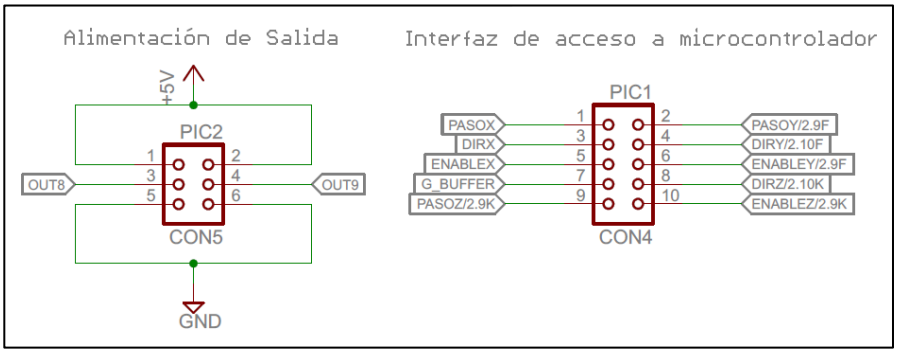
Fuente: elaboración propia, utilizando el programa EAGLE.

Del lado derecho de la figura 35 se muestra la conexión de los contactos del relé, el normalmente abierto (denotado como S), el normalmente cerrado (denotado como O), y el polo, o pivote (denotado como P). Estos contactos son conectados hacia una bornera de 3 terminales, donde es posible conectar de forma externa la herramienta rotativa para ser controlada por la interfaz de control. En paralelo a la conexión de los contactos del relé se observa la alimentación de una red RC con el objetivo de prevenir los chispazos producidos en los contactos del relé al momento de la conexión mecánica, alargando la vida del mismo.

La utilización de un relé para la tarea de activar la herramienta rotativa se debe a que la máquina fresadora fue pensada para utilizar herramientas de tipo comercial. Este tipo de herramientas utilizan generalmente una fuente de alimentación estándar de 120 VAC de la línea eléctrica. Es por esto que se decidió que las posibles herramientas activadas por la máquina fresadora, tales como una aspiradora, utilizaran un circuito de activación en la interfaz de control a través de un relé, y la alimentación de estos dispositivos se hiciera de forma externa.

Finalmente, la interfaz de control posee dos conectores IDC10 e IDC6 auxiliares, a través de los cuales es posible manejar las señales hacia los controladores de motores paso a paso individuales. En la figura 36 se observa la conexión de estos conectores auxiliares hacia las demás señales en el diseño. Esta salida auxiliar se agregó con el objetivo de permitir a futuro un dispositivo externo a la caja de control para manejar la máquina fresadora. Es decir, con esta salida auxiliar es posible la conexión de un dispositivo controlador manual a través de un panel de botones o palancas, con el fin de posicionar la máquina sin necesidad de utilizar la computadora.

Figura 36. **Conectores auxiliares en interfaz de control**



Fuente: elaboración propia, utilizando el programa EAGLE.

3.2.1.3. Mapeo de pines del puerto paralelo

Para la utilización de esta interfaz de control a través del puerto paralelo es necesario conocer la disposición precisa de cada uno de los pines. Esto permite configurar la interfaz de control en la computadora a través del *software* de control para máquinas CNC. La disposición de los pines con el diseño realizado se muestra en la tabla I, que es un resumen de las conexiones realizadas en el diseño de la interfaz de control presentado en la sección de anexos.

Tabla I. **Disposición de pines del puerto paralelo en la interfaz de control**

Número de pin del puerto paralelo	Utilización
1	No utilizado
2	Señal de paso eje X en CON1
3	Señal de dirección eje X en CON1
4	Señal de paso eje Y en CON2
5	Señal de dirección eje Y en CON2
6	Señal de paso eje Z en CON3
7	Señal de dirección eje Z en CON3
8	Salida auxiliar en CON5
9	Salida auxiliar en CON5
10	Entrada para interruptor límite en X2-2
11	Entrada para interruptor límite en X2-1
12	Entrada para interruptor límite en X3-2
13	Entrada auxiliar en X3-1
14	Activación de relé 2
15	Entrada auxiliar en X4-2
16	Activación de relé 1
17	Salida auxiliar en JP4
18-25	Referencia del puerto paralelo (GND)

Fuente: elaboración propia, con base en las figuras 30 y 31.

De igual forma, para cada uno de los conectores IDC10 para cada controlador de motor se presenta en la tabla II la disposición de pines para medición y depuración en las placas de circuitos impresos.

Tabla II. **Disposición de pines de salida para controladores de motores**

Número de pin del conector IDC10	Utilización
1 y 2	Alimentación de los circuitos lógicos (+5 VDC)
3	Señal de paso para el motor
4	Señal de dirección para el motor
5	Señal de habilitación del motor
6	Control de secuencia del motor
7	Voltaje de referencia del motor para ajuste de corriente
8	Señal de sincronización de controlador de motores
9 y 10	Voltaje de referencia a la alimentación (GND)

Fuente: elaboración propia.

3.2.2. Diseño de interfaz de control utilizando Arduino

Se presenta el diseño de una interfaz de control que se puede utilizar sobre la plataforma de *hardware* abierto Arduino, que es una placa de circuito impreso con un microcontrolador montado, pines de entradas y salidas de propósito general, y una conexión USB hacia la computadora para reprogramar el microcontrolador. La interfaz de control fue diseñada para montarse justo encima del Arduino, siendo utilizada como un *shield*, como muchas otras placas de circuito impreso que proveen de funcionalidades extra al mismo.

El modelo de Arduino utilizado para el diseño es el Arduino UNO R3, sobre el cual se monta el proyecto de código abierto GRBL, que permite el

control de máquinas CNC a través de una conexión USB con la computadora y emulación de un puerto serial. GRBL es un proyecto de *software* para el Arduino que está en constante desarrollo, a la fecha de redacción de este documento se encuentra en la versión 0.9 g.

3.2.2.1. Capacidades y requerimientos de la plataforma

El diseño de la interfaz para el control de la máquina fresadora se realizó con base en las capacidades y requerimientos de GRBL. GRBL permite el control completo de tres motores paso a paso simultáneamente, a través de las señales de paso, dirección, y habilitación para cada motor. El *software* permite el control de una única herramienta a través de una señal de habilitación, y permite la conexión de los tres interruptores de límite para cada uno de los ejes de la máquina fresadora. Como requerimiento, la interfaz de control debe proveer el voltaje de alimentación a la tarjeta Arduino. Además es necesario un cable de extensión USB macho tipo A hacia macho tipo B, para la entrada del Arduino.UNO R3.

La interfaz de control por puerto paralelo retransmite las señales provenientes del Arduino hacia los controladores de motores individuales para mover los motores de la máquina fresadora. Para esta interfaz es necesario utilizar una computadora para enviar los códigos G hacia el Arduino, pero la interpretación de los códigos y generación de las señales para los motores es realizada por el Arduino.

Finalmente, la capacidad de procesamiento de códigos G depende directamente de la versión utilizada del *software* de GRBL. El estilo del código G enviado al intérprete es único y la cantidad de códigos disponibles para

realizar movimientos es más limitada, a cambio de no utilizar una computadora para esta tarea. Sin embargo, con los códigos disponibles en GRBL es posible obtener los mismos resultados en el trabajo de fresado de placas de circuitos impresos, que utilizando una interfaz por puerto paralelo y un intérprete de código G en la computadora.

3.2.2.2. Portabilidad

Debido a la naturaleza de la plataforma de *hardware* abierto Arduino, es posible portar el diseño hacia una tarjeta que no sea el Arduino UNO R3, por ejemplo la tarjeta Arduino Mega 2560. Todo esto debe ser hecho utilizando el código fuente del proyecto GRBL en la página de desarrollo y cambiando la configuración de los pines que utiliza el software para que se adapte a los de la nueva tarjeta. Además, debe cambiarse el diseño de la interfaz de control para permitir que se monte sobre la nueva tarjeta Arduino. Para esto hay instrucciones en la página de desarrollo del proyecto.

Estos posibles cambios en la tarjeta de desarrollo permiten que el diseño de la interfaz de control presentado sea portable hacia otras plataformas, siempre y cuando se cambie la disposición de los componentes en la placa de circuito impreso, para que pueda ser montada sobre otras tarjetas de desarrollo, que no sean necesariamente Arduino.

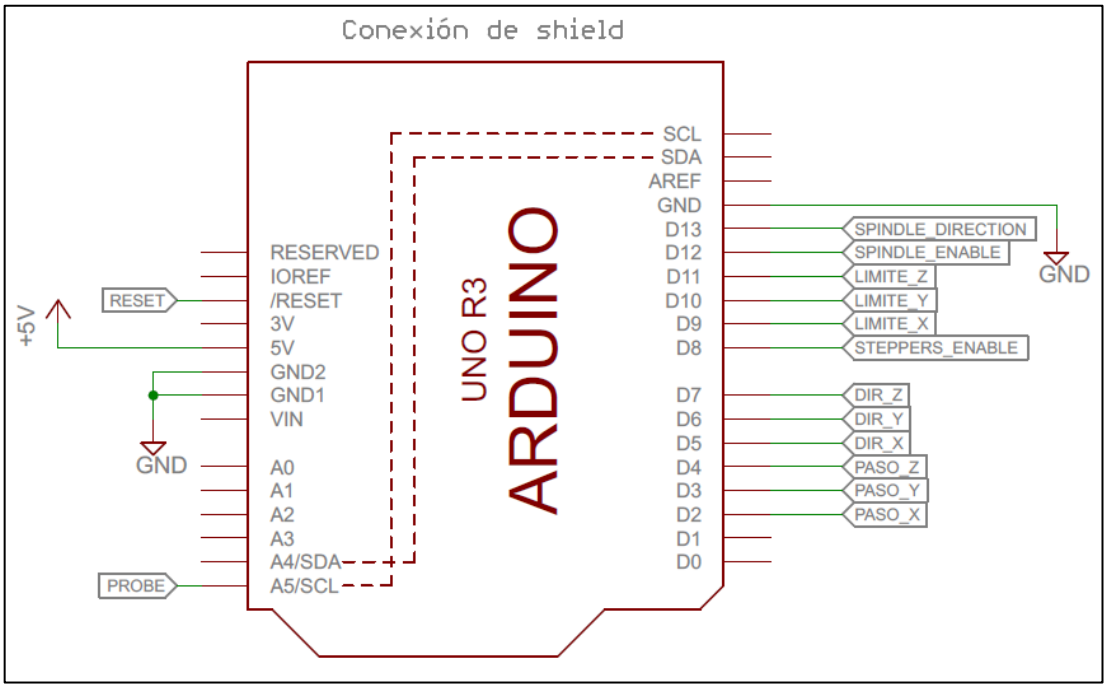
3.2.2.3. Explicación de la interfaz

El diseño de la interfaz de control está basado en la disposición de pines del Arduino UNO R3 para proveer las capacidades mínimas a la máquina fresadora y poder llevar a cabo la tarea de fresado de placas de circuitos impresos. El diseño completo de la interfaz de control puede ser consultado en

la sección de anexos; en esta sección se tomarán fragmentos de este diseño para explicar con mayor detalle el funcionamiento del circuito.

En la figura 37 se muestra la conexión de la tarjeta de desarrollo hacia el diseño de la interfaz de control. Del lado izquierdo del diagrama de la figura 37 se muestra la conexión de la alimentación hacia el Arduino, que será alimentado a través del circuito de regulación y alimentación en la interfaz de control, y no a través de la alimentación dada por la computadora a través del cable USB.

Figura 37. **Conexión de Arduino a interfaz de control**

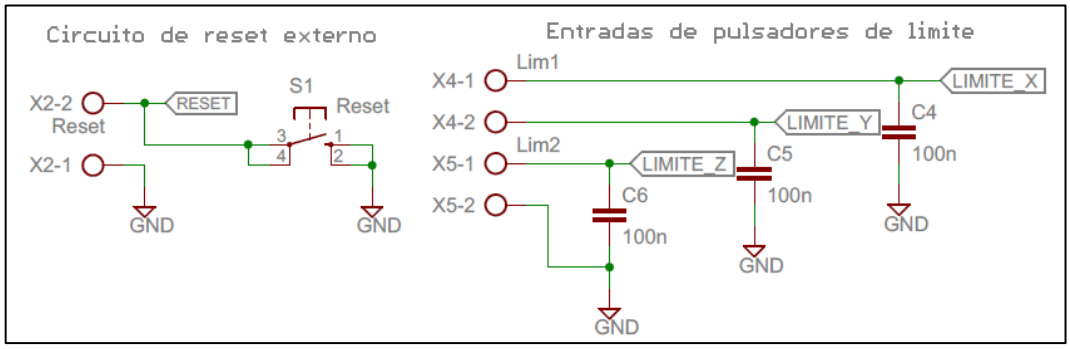


Fuente: elaboración propia, utilizando el programa EAGLE.

Las señales de control para los motores paso a paso se muestran del lado derecho en el diagrama de la figura 37. Estas señales son de paso, dirección y habilitación para todos los motores. Asimismo se muestra la conexión de los interruptores de límite para los tres ejes. Para este diseño es importante resaltar la conexión PROBE en el pin A5 del Arduino, que servirá para realizar el método de correción de alturas en la tarea de fresado de placas de circuitos impresos. Este procedimiento será explicado en el capítulo de *software* del proyecto.

En la figura 38 se muestra la conexión de los interruptores de límite hacia las borneras en la interfaz de control. Ya que el microcontrolador en la tarjeta de desarrollo posee resistencias *pull-up* internas solamente se agregaron capacitores conectados a tierra para que el voltaje en la entrada del pin no cambie bruscamente con cualquier señal de ruido externo. También se muestra la conexión del circuito de reinicio externo hacia una bornera, con el fin de implementar un botón de paro de emergencia si la máquina se encuentra en funcionamiento y es urgente detenerla. Asimismo, se agrega un botón para poder reiniciar el Arduino desde la interfaz de control.

Figura 38. **Conexión de interruptores de límite y reset externo**



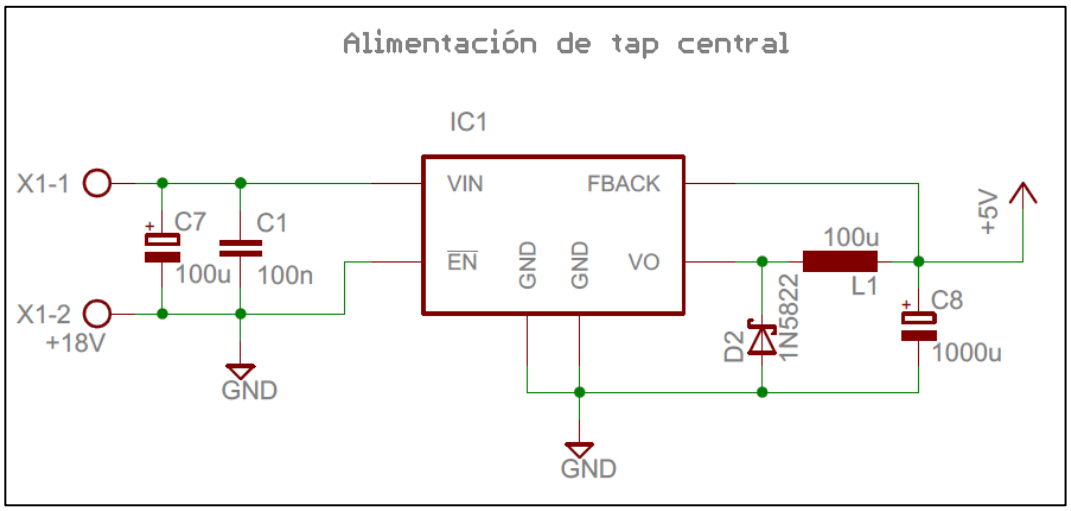
Fuente: elaboración propia, utilizando el programa EAGLE.

El circuito de alimentación para la interfaz de control se muestra en la figura 39. Este circuito es conectado hacia el tap central del transformador principal de la fuente de poder de todo el equipo electrónico. El voltaje de corriente directa con rizo es de 18 V, y para alimentar el resto de la interfaz se utilizó un voltaje de 5 VDC. La etapa de regulación está compuesta por un circuito integrado CI LM2576 de 5 VDC con un empaquetado D2T de 5 pines, que funciona como regulador conmutado para obtener el voltaje de alimentación de 5 VDC con hasta 3 A de corriente de salida.

El circuito de alimentación requiere que el voltaje de entrada tenga el menor rizo posible, para esto se conectan los capacitores C1 y C7 a la entrada del regulador. En la salida se conecta un diodo Schottky 1N5822 que provee una rápida respuesta al circuito de carga y descarga de corriente formado por la bobina y la inductancia.

Este tipo de regulador mantiene el voltaje constante gracias a la conmutación de la salida, permitiendo operar con mayor eficiencia, y por tanto, generando menos calor a disipar en el circuito. La eficiencia típica del circuito es del 88 % utilizando una fuente de alimentación de entrada de 18 V, de acuerdo con la hoja de datos del fabricante.

Figura 39. **Circuito de regulación conmutado**



Fuente: elaboración propia, utilizando el programa EAGLE.

Finalmente, las conexiones hacia los controladores de motores individuales se realiza utilizando conectores IDC10, al igual que en el diseño presentado en la sección anterior; los diagramas de conexión ya fueron presentados en la figura 33, y la disposición de los pines es la misma que la presentada en la tabla II de la sección anterior. Se siguió la misma disposición de pines para que exista una mayor flexibilidad en la plataforma que se desee utilizar para la interfaz de control.

Asimismo, el circuito de control de activación de la herramienta rotativa a través de relé es el mismo que el utilizado en el diseño de la interfaz de control por puerto paralelo. El circuito fue presentado en la figura 35 de la sección anterior. La única modificación al mismo para el diseño de esta interfaz de control es que se utilizó un relé con una bobina de 5 VDC en lugar del relé con bobina de 12 VDC utilizado en la interfaz paralela.

3.2.2.4. Mapeo de pines de la interfaz

En la tabla III se muestra la conexión de cada uno de los pines en la tarjeta de desarrollo Arduino hacia la interfaz de control. Este listado de conexiones es útil para mediciones directo en la placa de circuito impreso con el propósito de realizar pruebas y depuración del circuito.

Tabla III. **Disposición de pines del Arduino UNO R3 hacia interfaz de control**

Pin del Arduino	Funcionalidad
/RESET	Señal de reinicio del *software* GRBL
5V	Fuente de alimentación para microcontrolador
GND	Referencia de la fuente de alimentación
A5/SCL	Pin de entrada para punta de prueba
D0	Reservado para la consola serial
D1	Reservado para la consola serial
D2	Señal de paso para el eje X
D3	Señal de paso para el eje Y
D4	Señal de paso para el eje Z
D5	Señal de dirección para el eje X
D6	Señal de dirección para el eje Y
D7	Señal de dirección para el eje Z
D8	Señal de habilitación para los tres motores
D9	Entrada para interruptor límite del eje X
D10	Entrada para interruptor límite del eje Y
D11	Entrada para interruptor límite del eje Z
D12	Señal de activación de herramienta rotativa
D13	Señal de dirección de rotación para herramienta (no utilizado)

Fuente: elaboración propia.

3.2.2.5. Interacción con la computadora

Para interactuar con la interfaz de control es necesario contar con una computadora con un puerto USB. Para comunicarse con el *software* GRBL montado en el Arduino se deben instalar los controladores necesarios para que la computadora reconozca el puerto serial de la tarjeta de desarrollo Arduino. Después, simplemente debe conectarse la tarjeta de desarrollo Arduino hacia la computadora para que sea instalado un puerto serial de comunicación.

La interacción de la interfaz de control a través de GRBL hacia la computadora es realizada utilizando cadenas de caracteres en un programa de consola serial. Estas cadenas incluyen los códigos G a ser interpretados por GRBL y permiten conocer el estado de la interfaz de control. El proceso de interacción con GRBL será descrito con mayor detalle en el capítulo de *software* del proyecto.

3.2.3. Ventajas y desventajas de las plataformas

Los diseños presentados en las secciones anteriores permiten controlar la máquina fresadora CNC con suficiente flexibilidad y facilidad de uso para llevar a cabo la tarea de fresado de placas de circuitos impresos. Sin embargo, existen diferencias notables en cuanto al máximo desempeño que se puede obtener de cada plataforma, pues cada diseño presenta ventajas y desventajas en su utilización para otro tipo de tareas que se pueden llevar a cabo en una máquina herramienta.

Por una parte, el esquema de utilización de interfaz de control por puerto paralelo requiere utilizar la computadora para generar las señales que controlan los motores paso a paso para cada eje. Esto lleva a la necesidad de tener una

computadora optimizada para operar como sistema de tiempo real (que responde a una mínima latencia), que permita la interpretación de código G y la transmisión de las señales para la interfaz de control. Este tipo de equipo puede ser mucho más costoso que su contraparte, una plataforma de desarrollo con un microcontrolador.

Sin embargo, utilizar una computadora para la tarea de interpretación del código G permite utilizar herramientas de *software* mucho más complejas que las que se pueden montar sobre una tarjeta de desarrollo, y a su vez, permite a la máquina herramienta llevar a cabo tareas más complejas. Esto es, utilizando la plataforma donde se tiene una interfaz de control por puerto paralelo se tiene una mayor flexibilidad en cuanto al código G que se puede ejecutar en la máquina herramienta, con el costo que conlleva tener un equipo dedicado solo a esta tarea, además de que el puerto paralelo de las computadoras es cada vez menos común para computadoras comerciales que no son para este fin.

En cambio, al utilizar la plataforma donde se tiene un microcontrolador en una tarjeta de desarrollo permite utilizar cualquier computadora que posea un puerto de conexión USB y enviar el código G a ser ejecutado por la máquina herramienta a través de una consola serial, con la desventaja de que la cantidad de códigos y las opciones con las cuales se debe ejecutar este código son más limitadas. Sin embargo este esquema es preferido por muchos entusiastas y aficionados al mundo de las máquinas herramientas, ya que no requiere conocimientos especializados del *software* de control de dichas máquinas herramienta, y el costo de la tarjeta de desarrollo es mucho menor que el de una computadora.

En adelante, se prefiere el uso de la plataforma de hardware abierto Arduino, ya que es una plataforma de bajo costo, que no requiere tener una

computadora dedicada por completo a manejar la máquina herramienta. Además, permite reproducir y compartir más fácilmente el trabajo realizado hacia una comunidad de aficionados al trabajo con máquinas herramienta. Esta plataforma es suficiente para llevar a cabo el trabajo de fresado de placas de circuitos impresos, el grabado y corte de materiales, y la supervisión y realimentación a través de la computadora.

3.3. Diseño de la fuente de alimentación

La fuente de voltaje de la máquina fresadora se encarga de alimentar los motores y el resto de la lógica de control para que la máquina funcione en su totalidad. La fuente de voltaje es una parte muy importante, ya que el desempeño de los motores paso a paso depende del dimensionamiento en potencia para la misma. Los valores de sus componentes varían de acuerdo con las especificaciones de los motores, los controladores y la interfaz de control. En esta sección se describe la fuente de voltaje que se utiliza para el proyecto.

3.3.1. Requerimientos

Los requerimientos de la fuente de alimentación están basados en los requerimientos de voltaje y corriente para los motores paso a paso, y en menor medida de la corriente necesaria para alimentar el resto de circuitos de control. Típicamente, el voltaje de alimentación de los motores paso a paso es mayor que el voltaje nominal del motor, esto para desarrollar más rápidamente el torque que permite dar el paso del motor, y se utiliza un control de corriente en la lógica de control del motor para no dañar las bobinas del mismo.

Los motores utilizados en el proyecto son NEMA17 bipolares de 24 V nominales y 400 mA para cada bobina. La idea principal del proyecto era diseñar y construir todos los circuitos necesarios para su funcionamiento, por lo que se diseñó una fuente de alimentación lineal utilizando un transformador disponible en el mercado local, con un voltaje nominal de salida de 24 VAC y 5 A. Al medir el transformador, la salida fue de aproximadamente 25,5 VAC, que resulta en un voltaje pico de 36 V, de acuerdo con la relación entre el voltaje pico de una señal sinusoidal y su valor efectivo. Este valor supera el voltaje nominal de los motores paso a paso.

Una fuente de alimentación lineal utiliza un esquema donde se conectan una tras otra las etapas de: transformación, rectificación, filtrado, regulado y salida. Para el diseño de la fuente de alimentación no se utilizó la etapa de regulación, ya que la interfaz de control se encarga de la regulación a 5 VDC y a 12 VDC para el funcionamiento de los componentes lógicos. Para el funcionamiento de los motores paso a paso, ya que el voltaje de alimentación es más alto que el voltaje nominal y se utiliza un control de corriente, no es necesario regular el voltaje aplicado a los motores.

Sin embargo, es necesario utilizar una etapa de transformación, rectificación y filtrado adecuadas para las necesidades del proyecto. Ya que el transformador utilizado es de 25,5 VAC y 5 A, se adecuan las etapas de rectificación y filtrado a las características de este transformador.

3.3.2. Diseño de la fuente de alimentación

Dado que los valores del transformador de 25,5 VAC y 5 A son de voltaje en corriente alterna y la corriente efectiva, se determinan entonces los valores de voltaje en corriente directa y la corriente pico a ser suplida por el

transformador. La relación entre estos valores está regida por la raíz de dos, de acuerdo con la siguiente ecuación:

$$V_{pk} = \sqrt{2}\, V_{ef}$$

Donde V_{pk} es el voltaje pico de la señal sinusoidal presente en la bobina secundaria del transformador, y V_{ef} es el voltaje eficaz medido por el multímetro en la bobina secundaria del transformador. De forma similar, la relación entre la corriente efectiva y la corriente pico se rige a través de la ecuación:

$$I_{pk} = \frac{I_{ef}}{\sqrt{2}}$$

Donde I_{pk} es la corriente pico suplida por la bobina secundaria del transformador, y I_{ef} es la corriente eficaz medida a través de la bobina secundaria del transformador. Como se observa en las ecuaciones anteriores, mientras el voltaje pico del transformador aumenta respecto del voltaje eficaz, la corriente pico disminuye, esto es una consecuencia de que la potencia de salida del transformador es constante, es decir, se cumple que:

$$V_{pk}\, I_{pk} = V_{ef}\, I_{ef}$$

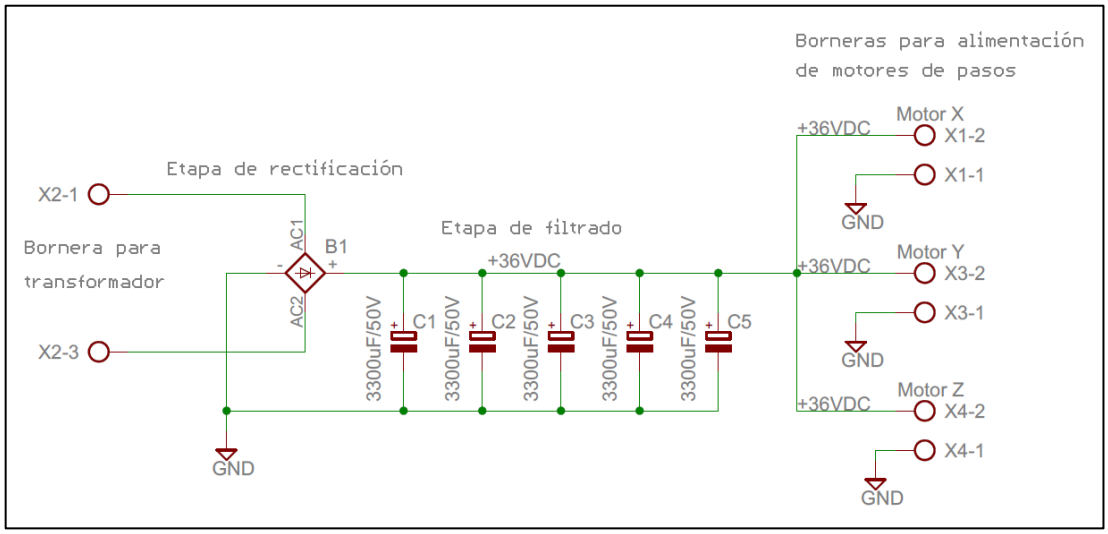
Por lo tanto, el voltaje pico del transformador en corriente directa sería de 36 V, y la corriente pico del transformador sería de aproximadamente 3,5 A. Cada motor paso a paso requiere una corriente de torque máxima de 400 mA, sumando la corriente para tres motores polarizados al mismo tiempo resulta un total de 1200 mA de corriente máxima para mantener el torque de retención de los tres motores al mismo tiempo. Con este cálculo, el transformador permite

alimentar la interfaz de control y el resto de circuitos que conforman la lógica de control.

En la figura 40 se observa la etapa de rectificación y filtrado de la fuente de alimentación. El transformador es conectado en una bornera hacia el puente de diodos, representado con el símbolo B1. Después de la etapa de rectificación, la señal de voltaje resulta una señal de corriente directa, pues ya no varía en polaridad, sin embargo es una señal pulsante, pues todavía presenta variaciones de voltaje cada medio ciclo de la señal sinusoidal.

Para convertir la señal en una de corriente directa, se aplica una etapa de filtrado, utilizando capacitores polarizados para mantener el voltaje constante y alimentar la carga, que en este caso son los motores que se conectan en las tres borneras mostradas la parte derecha del diagrama de la figura 40.

Figura 40. **Etapa de rectificación y filtrado de la fuente de voltaje**



Fuente: elaboración propia, utilizando el programa EAGLE.

Para determinar la capacitancia requerida es necesario conocer el voltaje de rizo que se desea. Ya que la carga se conecta hacia la red de capacitores, el transformador alimenta la red de capacitores, y estos a su vez la carga, sin embargo, cuando el voltaje de los capacitores es mayor al voltaje de la etapa de rectificación, no circula corriente de carga hacia los capacitores, es por esto que la señal en la salida de la etapa de filtrado presenta un pequeño rizo o variación de corriente alterna. La capacitancia C es determinada por la siguiente ecuación:

$$C = \frac{I}{2\,f\,Vr}$$

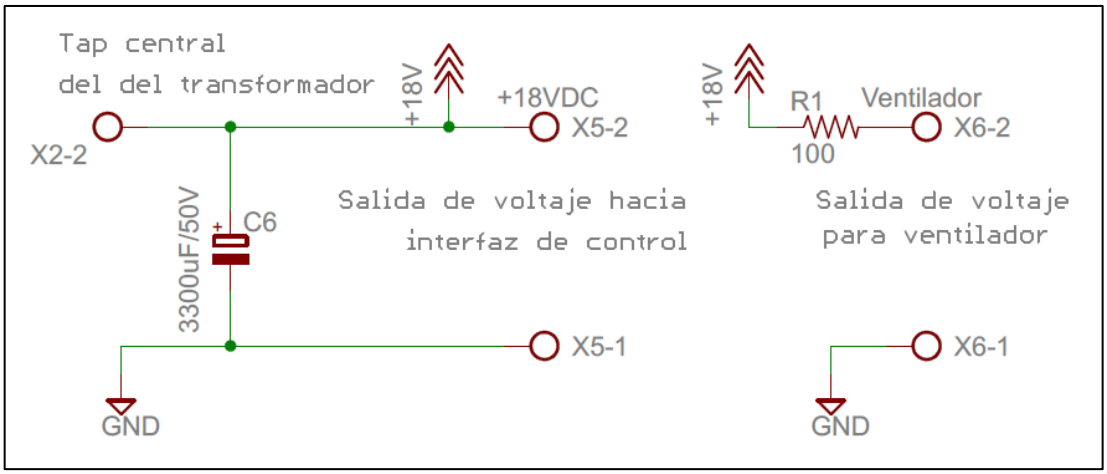
Donde I es la corriente media que se desea mantener, f la frecuencia de la señal sinusoidal en la bobina secundaria del transformador, y V_r el voltaje de rizo en la etapa de filtrado, que es generalmente un porcentaje del voltaje pico.

Utilizando la ecuación anterior, la frecuencia de 60 Hz, un voltaje de rizo del 5 % del voltaje pico, esto es 1,8 V, y la corriente pico a mantener como 3,5 A se obtiene una capacitancia resultante de 16203 μF. Por lo que en el diseño se utilizan 5 capacitores de 3300 μF, que dan una capacitancia resultante de 16500 μF, cuyo valor está lo cerca posible del valor calculado.

Finalmente, en la figura 41 se observa la conexión del *tap* central del transformador. Este es conectado hacia un capacitor a tierra, ya que la señal medida desde el punto utilizado como tierra hasta el *tap* central es sinusoidal rectificada, pero con la mitad del voltaje pico medido entre los extremos del transformador. Con esto se obtiene un voltaje pico de 18 V para la alimentación de la interfaz de control, donde se regulará este voltaje a 5 VDC y 12 VDC para los circuitos de lógica.

Además, se muestra la conexión del ventilador de la caja de control. El ventilador está compuesto de un motor DC sin escobillas de 12 V. Para alimentarlo se conecta a través de una resistencia hacia los 18 V provenientes del *tap* central del transformador. La resistencia limita la corriente al ventilador, y divide el voltaje de alimentación para que este tenga los 12 V nominales.

Figura 41. **Conexión de *tap* central del transformador**



Fuente: elaboración propia, utilizando el programa EAGLE.

3.3.3. Ventajas y desventajas de la implementación

La fuente de alimentación se diseñó e implementó utilizando componentes y un esquema de fuente de alimentación lineal, cuyo diseño es ineficiente respecto de una fuente de alimentación conmutada; sin embargo, es mucho más simple y menos costoso en tiempo de desarrollo. Además, las fuentes de alimentación lineal requieren de grandes transformadores, y generan más calor, mientras las fuentes conmutadas generan ruido eléctrico de alta frecuencia que puede interferir a los equipos y componentes próximos a estas fuentes, y son más propensas a averías, ya que su funcionamiento es más complejo.

Otra desventaja de la implementación fue la utilización de motores paso a paso con un voltaje nominal de 24 V, esto impone el requerimiento para la fuente de alimentación de los motores, que debe ser mayor este voltaje. Una solución alternativa más comercial hubiese sido conseguir motores paso a paso con un voltaje nominal de entre 3 V y 4 V, y utilizar una fuente de alimentación conmutada de 12 VDC para la alimentación de los motores y de la interfaz de control. Sin embargo, con el fin de gastar lo menos posible en las partes del proyecto, se decidió utilizar los motores y el transformador disponible para realizar la máquina herramienta.

3.4. Caja de control e interconexión

La caja de control permite montar las placas de circuitos impresos del proyecto para realizar las interconexiones necesarias y mantener todos los circuitos y cables en buen estado. Además, permite montar un panel de conexiones donde se puedan conectar los componentes de la máquina fácilmente.

3.4.1. Diseño de la caja de control

La caja de control consiste en una caja de madera de 5/8 in, con medidas de 41 cm de largo por 27 cm de ancho, y 18 cm de alto, unida con tornillos M4 pasados y apretados con tuercas. Esto permite que la caja sea desarmable para montar todas las placas de circuitos impresos sobre la base de la caja, el transformador para la fuente de alimentación, para realizar las conexiones entre las placas y finalmente armar las partes laterales de la caja para poder cerrarla.

En la parte frontal de la caja de control se instala un panel de conexiones hacia los componentes electrónicos montados en la máquina. El panel de conexiones incluye:

- Una conexión USB de la tarjeta de desarrollo Arduino.
- Tres conectores MIC334 para panel de 4 pines para las conexiones hacia los motores paso a paso.
- Un conector MIC334 para panel separado para los interruptores de límite.
- Un tomacorriente convencional para la conexión de la herramienta rotativa.
- Un interruptor de encendido.
- Un conector C14 para panel como alimentación de la caja de control.

La conexión USB de la caja de control es directamente la conexión presente en la tarjeta de desarrollo Arduino, y para que el conector pudiera ser alcanzable desde afuera, se dejó una abertura del ancho de la tarjeta en la parte frontal de la caja de control, con lo que se puede alcanzar el conector, el botón de reinicio de la tarjeta de desarrollo y ver el indicador de encendido de la tarjeta.

En la figura 42 se muestra un conector MIC334 para panel. Este conector es montado en la caja de control ya que posee 4 pines de conexión para los cuatro cables de los motores paso a paso. Además, se utiliza un conector igual para las tres señales de límite en cada eje de la máquina y la referencia o conexión a tierra.

Figura 42. **Conector MIC334 para panel de conexiones**

Fuente: *Socket MIC334*.
http://www.marelectronics.gr/photos/eshopItems/mic334.jpg.
Consulta: 1 de abril de 2015.

Los cables de la máquina fresadora son soldados a conectores hembra MIC324, para acoplarse rápidamente a los conectores instalados en el panel de conexiones. Estos conectores incluyen un seguro contra la tensión en los cables y una rosca de seguridad para mantener la conexión firmemente en el panel de conexiones. En la figura 43 se puede observar un conector MIC324 para los cables del motor y los interruptores de límite.

Figura 43. **Conector MIC324 para cables de la máquina fresadora**

Fuente: *Socket MIC324*.
http://www.marelectronics.gr/photos/eshopItems/mic324.jpg.
Consulta: 1 de abril de 2015.

En la figura 44 se muestra el conector C14 para panel utilizado para proveer la alimentación proveniente de la línea eléctrica de 120 VAC a la caja de control. Este conector, junto con el interruptor de encendido de la caja de control fueron reciclados de una vieja fuente de computadora.

Figura 44. **Conector C14 para panel de conexiones**

Fuente: *Single 3-pin chassis mount*. http://www.pouroutos.com/ac-dc-connectors.html. Consulta: 1 de abril de 2015.

Finalmente, para la conexión de la herramienta rotativa se utilizó un tomacorriente común. Este tomacorriente se conecta internamente hacia la línea eléctrica del conector C14 y se utilizan los contactos de relé en la interfaz de control para conmutar el encendido de la herramienta rotativa. Para montar este tomacorriente fue necesario abrir un agujero con el barreno para pasar los cables del tomacorriente.

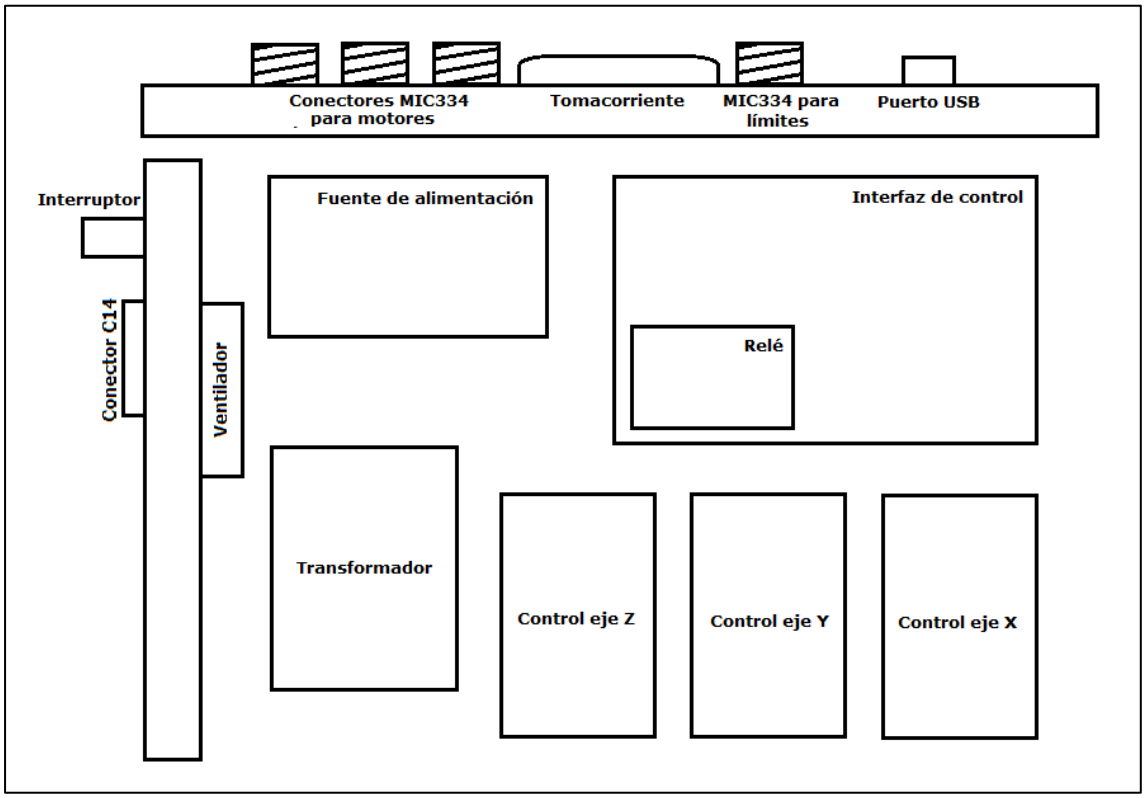
En la parte superior de la caja de control se monta una tapadera de madera de 41 cm de largo por 27 cm de ancho; esta se une a la caja con una bisagra y se limita la apertura con una cadena instalada en la parte izquierda de la caja. El ventilador de la caja de control se instala en una de las piezas

laterales cerca del transformador, y el flujo de aire es de los agujeros hechos para los tornillos de armazón de la caja hacia fuera a través del ventilador.

3.4.2. Interconexión de las placas de circuitos impresos

En la figura 45 se muestra un croquis del montaje de las placas de circuitos impresos, conectores y otros componentes en la caja de control. Por simplicidad, no se muestran las conexiones realizadas entre los distintos componentes, sin embargo, se listan a continuación.

Figura 45. **Montaje de placas de circuitos impresos**



Fuente: elaboración propia, utilizando el programa Dia.

- Conexiones del interruptor, el conector C14 y el transformador:

    - Los cables de vivo y neutro del conector C14 son conectados al transformador, pasando uno de estos a través del interruptor.
    - El transformador es conectado a la placa de fuente de alimentación.

- Conexión de la fuente de alimentación:

    - Los cables del transformador son conectados en las borneras de acuerdo con el diseño de la fuente de alimentación.
    - El ventilador es conectado en la bornera de salida para este propósito.
    - La salida de +18V de la fuente de alimentación es conectada a la interfaz de control.
    - La bornera de salida de cada motor es conectada hacia cada controlador de motor.

- Conexión de la interfaz de control:

    - La alimentación proviene de la salida de +18 V en la fuente de alimentación.
    - El puerto USB está conformado por el Arduino montado debajo de la interfaz de control.
    - Los cuatro cables del conector MIC334 de límites son conectados a las borneras para interruptores de límite.
    - El tomacorriente es conectado en paralelo a la conexión realizada con el transformador y el interruptor, pasando uno de los cables por circuito de relé.

- Los tres cables IDC10 de motores para cada eje son conectados cada uno hacia su controlador de motor.

- Conexión de cada controlador de motor:

  - La alimentación de cada motor proviene de una bornera en la placa de fuente de alimentación.
  - Los cuatro cables de cada motor se conectan hacia los conectores MIC334 en el panel de conexiones.
  - El cable IDC10 con las señales de control proviene de la interfaz de control.

3.4.3. Asignación de señales a conectores

Para referencia de las conexiones entre los motores paso a paso y los conectores MIC324 en la caja de control, se muestra en la tabla IV la disposición utilizada para las conexiones.

Tabla IV. **Disposición de conectores MIC324 hacia cables de motores paso a paso**

Número de pin del conector	Cable del motor
1	Bobina A+
2	Bobina A-
3	Bobina B+
4	Bobina B-

Fuente: elaboración propia.

Para referencia de las conexiones realizadas entre los cables de los interruptores de límite y el conector MIC324 en la caja de control, se muestra en la tabla V la disposición utilizada para las conexiones.

Tabla V. **Disposición de conectores MIC324 hacia cables de límites**

Número de pin del conector	Cable asignado
1	Interruptores límite para eje X
2	Interruptores límite para eje Y
3	Interruptores límite para eje Z
4	Señal de referencia (GND)

Fuente: elaboración propia.

3.4.4. Puntos de medición y sus valores

Se da una lista de puntos importantes de medición para determinar el correcto funcionamiento de cada una de las placas de circuito impreso en la caja de control. Las mediciones deben ser hechas con mucho cuidado cuando se hacen sobre pines de los conectores IDC10. Asimismo, las mediciones no deben realizarse sobre los conectores MIC334 en el panel de conexiones, sino sobre las borneras en las placas de circuitos impresos.

- Mediciones del transformador y la línea eléctrica:
    - Se debe medir un voltaje eficaz de alrededor de 120 VAC en la bobina primaria del transformador.
    - En la bobina secundaria debe medirse un voltaje eficaz de alrededor de 24 VAC. La medición en esta bobina debe hacerse

con el interruptor encendido y apagado, para determinar la conmutación de este.

- Mediciones en la fuente de alimentación:

  - En la bornera de conexión del transformador, entre los extremos se debe medir un voltaje eficaz de alrededor de 24 VAC provenientes del transformador. Y entre el *tap* central y un extremo del transformador alrededor de 12 VAC.
  - En la salida de +18 V se debe medir alrededor de 18 VDC (voltaje en corriente directa).
  - En la salida para cada motor debe medirse alrededor de 36 VDC para la alimentación de los motores.

- Medición del voltaje del ventilador:

  - El voltaje para el ventilador debe ser de alrededor de 12 VDC.

- Mediciones la interfaz de control:

  - En la bornera de alimentación, alrededor de 18 VDC.
  - Siguiendo el diseño de la placa de circuito impreso, medir en los pines de los conectores IDC10 el voltaje de alimentación 5VDC, las señales de control para los motores y el voltaje de referencia para el control de corriente de cada motor.
  - En la bornera de relé, medir un circuito abierto cuando el relé está desactivado; cuando se encuentra activo debe medirse un voltaje de 0 V entre la terminal normalmente abierta y el pivote del relé.

- Mediciones en los controladores de motores:

    - Voltaje de alimentación 5 VDC y tierra entre los pines 1 y 10, o 2 y 9 del conector IDC.
    - Señales de control de motores y voltaje de alimentación en el resto de pines, según el diseño del controlador de motor.
    - Voltaje de alimentación del motor en la bornera, alrededor de 36 VDC.

Un correcto funcionamiento de las placas de circuitos impresos mantiene los motores paso a paso polarizados, y se puede escuchar un chirrido de alta frecuencia de la conmutación de las bobinas debido al control de corriente de los controladores. Asimismo, el led de funcionamiento en la tarjeta de desarrollo Arduino enciende y se mantiene en verde, la herramienta rotativa está apagada y el ventilador funciona normalmente.

4. DESCRIPCIÓN DEL SOFTWARE PARA UTILIZAR LA MÁQUINA FRESADORA

Se muestran todas las herramientas de *software* para utilización de la máquina herramienta y generación del código G para fresado de placas de circuitos impresos. Las herramientas de software más importantes son:

- El programa de diseño de circuitos impresos.
- El programa de conversión del diseño a código G.
- El programa intérprete del código G y controlador de la máquina fresadora, y las alternativas de este.
- Programas de manipulación y optimización del código G para agregar funcionalidad extra en el fresado de placas de circuitos impresos, como son un optimizador de trayectorias y un método de medición y corrección de alturas.

4.1. Software para diseño de placas de circuitos impresos

Se describe el programa para el diseño de placas de circuitos impresos y la configuración del mismo para la generación de código G.

4.1.1. Proceso de diseño de placas de circuitos impresos

El *software* utilizado para el diseño computarizado de circuitos impresos es CadSoft EAGLE (cuyas siglas en inglés son Easy Applicable Graphical Layout Editor). EAGLE es un programa de diseño de diagramas esquemáticos y circuitos impresos con enrutador automático. Este programa es muy popular en

el mundo de los proyectos “Hazlo tu mismo” (*DIY* por sus siglas en inglés), ya que muchas versiones del programa tienen una licencia *freeware*.

Entre las virtudes del programa se puede resaltar que, a pesar de no ser de código abierto, se pueden realizar diseños extremadamente completos en la versión gratuita, tal como en la versión de pago, con la única restricción de mantener un tamaño máximo de la placa de circuito impreso de 10 cm por 8 cm, y utilizar solamente dos capas de cobre para el diseño del circuito impreso. Además, existe una gran cantidad de librerías de componentes circulando por la red de forma gratuita.

Finalmente, en EAGLE es posible agregar programas escritos por el usuario, denominados ULPs (User Language Programs). Estos programas son escritos en un lenguaje parecido a C, que permiten añadir funciones personalizadas a EAGLE, tales como abrir y exportar archivos desconocidos para el programa. Al igual que con las librerías de componentes, las ULPs de EAGLE son distribuidas por la red en forma gratuita, permitiendo a los usuarios de EAGLE mejorar su experiencia personal con el software.

El programa consta de tres partes esenciales:

- Un panel de control, donde es posible crear y seleccionar proyectos, y dentro de estos crear y seleccionar archivos esquemáticos y circuitos impresos. Además permite manejar directorios de librerías y ULPs y proyectos.

- Un editor de diagramas esquemáticos, donde es posible colocar componentes y conectarlos a través de cables o etiquetas para representar las conexiones eléctricas.

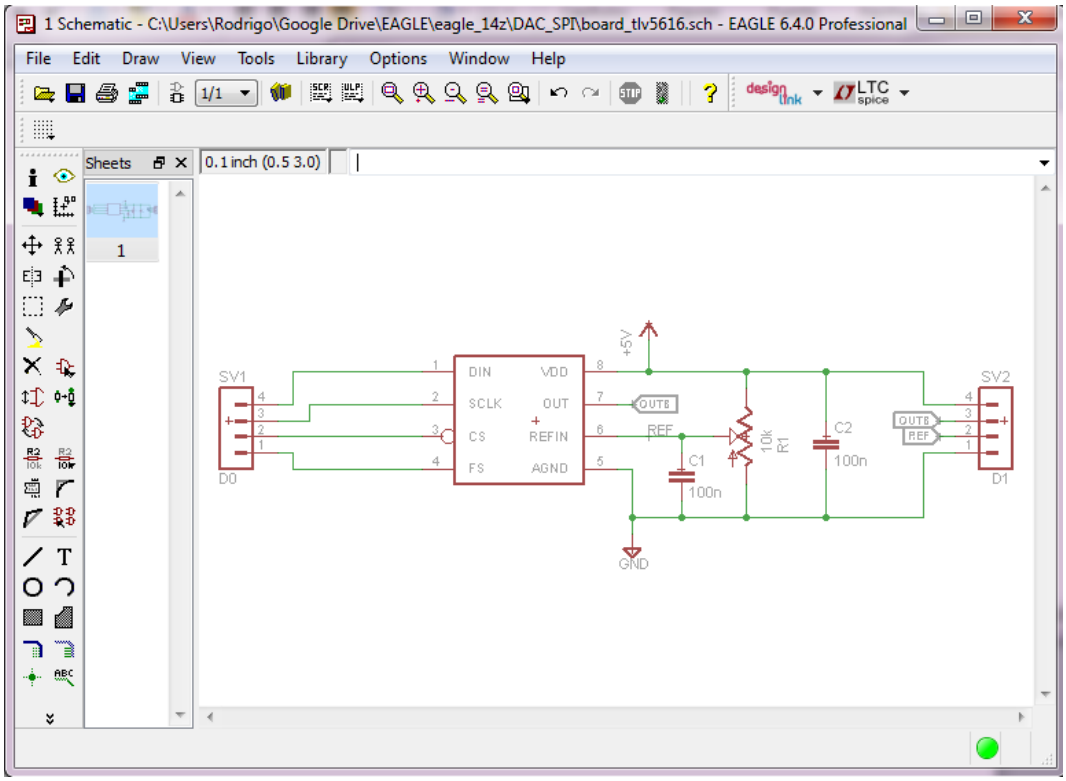
- Un editor de circuitos impresos con enrutador automático, que a partir del diseño esquemático permite posicionar los componentes electrónicos y dibujar las pistas que forman los caminos eléctricos en la placa de circuito impreso.

En la figura 46 se muestra el editor de diagramas esquemáticos de EAGLE. Del lado izquierdo de la figura se puede observar la barra de herramientas; para aplicar una herramienta al diseño basta con dar clic sobre esta. También es posible escribir el nombre del comando correspondiente a la herramienta para aplicar la función. Esto es especialmente útil para manejar muchas herramientas y cambiar entre estas rápidamente, ya que se puede asignar una tecla rápida a comandos especificados por el usuario.

El editor de esquemáticos permite añadir todos los componentes necesarios para desarrollar una placa de circuito impreso. Además, si es necesario utilizar componentes que no estén en las librerías de EAGLE, es posible crear los componentes con su representación esquemática y dimensiones del encapsulado.

Una característica importante del editor de esquemáticos es que EAGLE posee una herramienta llamada Electrical Rule Check (ERC), que es un comando utilizado para encontrar errores eléctricos en el diagrama esquemático. La herramienta muestra mensajes de errores y advertencias que deben ser tomadas en cuenta antes de la producción del circuito impreso. Asimismo, el comando realiza un chequeo de consistencia, para determinar que todas las conexiones eléctricas del diagrama esquemático estén presentes en el diseño del circuito impreso.

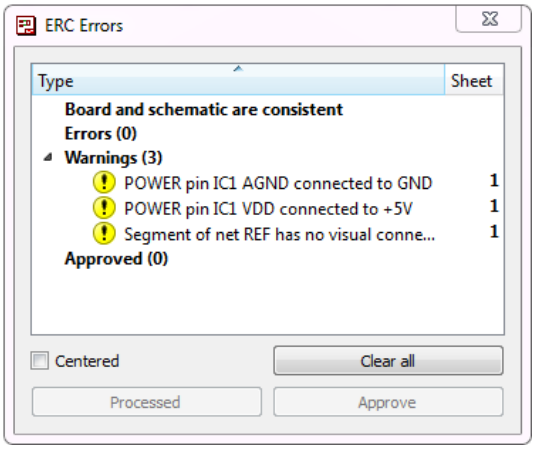
Figura 46. **Editor de diagramas esquemáticos de EAGLE**



Fuente: elaboración propia, utilizando el programa EAGLE.

En la figura 47 se muestra la ventana de ERC para el diagrama esquemático de la figura 46. En este caso, la herramienta no indica ningún error eléctrico en el diagrama, sin embargo, es importante revisar las advertencias mostradas y corregir las que provoquen una inconsistencia en el diseño del circuito.

Figura 47. **Ventana de ERC de EAGLE**



Fuente: elaboración propia, utilizando el programa EAGLE.

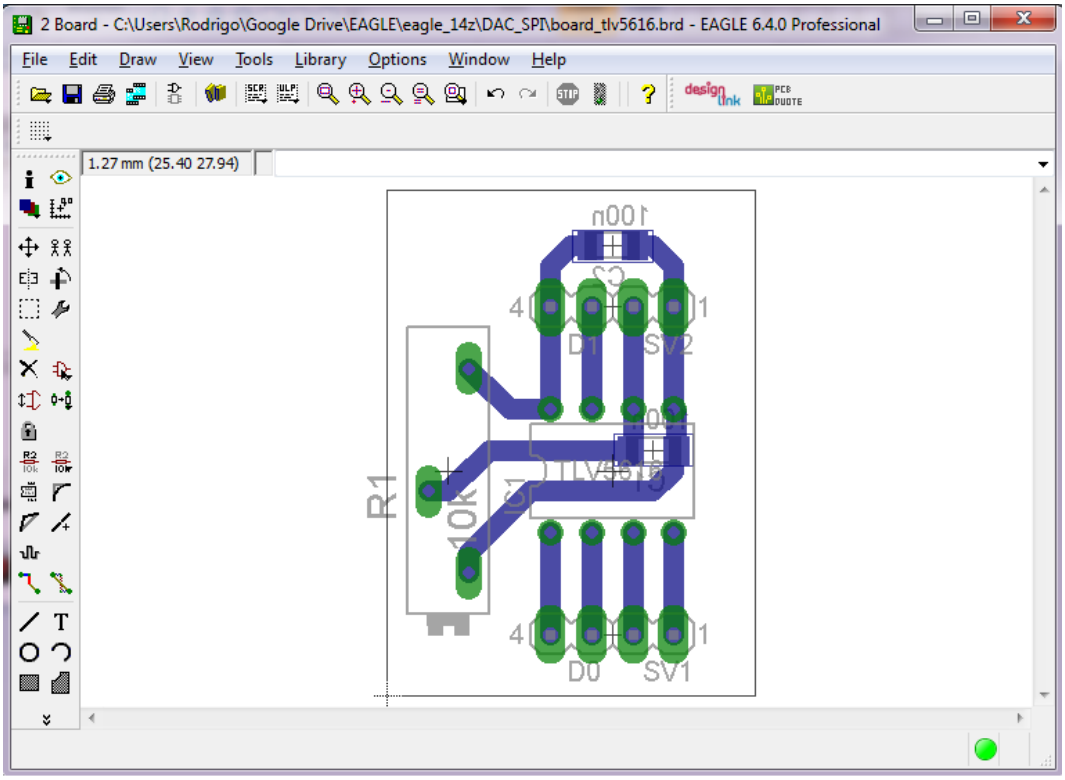
A partir del diagrama esquemático es posible crear el diagrama de circuito impreso. Para esto debe seleccionarse la opción *File -> Switch to board* en la ventana de edición del circuito esquemático. Una vez creado el diagrama de circuito impreso se posicionan los componentes y se dibujan las pistas que conforman las conexiones eléctricas en la placa de circuito impreso.

En la figura 48 se muestra la ventana de edición de circuitos impresos de EAGLE, que corresponde al diagrama esquemático de la figura 46. Del lado izquierdo de la figura se observan todas las herramientas disponibles para el diseño del circuito impreso. Nuevamente, resulta extremadamente útil asignar teclas rápidas a los comandos ejecutados por las herramientas.

Las pistas que unen a los componentes son conectadas a terminales especiales llamadas *pads*, que son los puntos del circuito donde va un pin del componente electrónico y además debe hacerse un agujero para que pase el pin del componente de un lado de la placa de circuito impreso al otro.

Existen también las terminales especiales llamadas vías, que a diferencia de los *pads* no conectan componentes; en cambio, sirven para realizar una conexión de un lado del circuito al otro a través de un agujero.

Figura 48. **Editor de circuitos impresos de EAGLE**

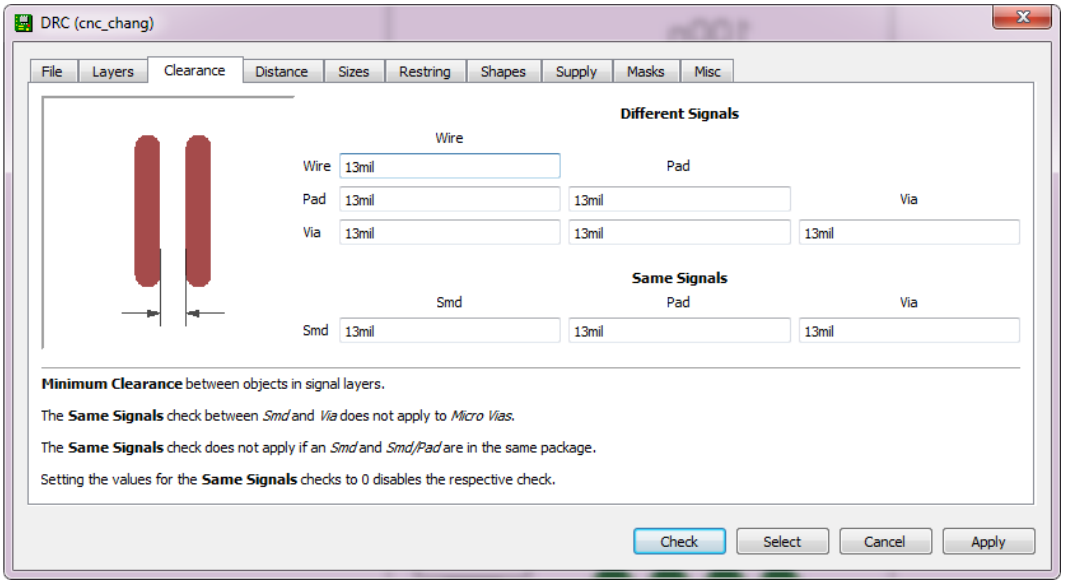


Fuente: elaboración propia, con base en la figura 46.

Una característica importante del editor de circuitos impresos de EAGLE es que permite revisar de acuerdo con un archivo de configuración si el diseño cumple especificaciones de tamaños de pistas, separación entre dichas pistas, *pads* y vías, distancias entre estas hacia la placa de circuito impreso, y muchas más configuraciones. Para revisar estas configuraciones se utiliza la herramienta Design Rule Check (DRC).

Esta es una función muy útil para configurar las especificaciones de producción para la placa de circuito impreso. En el caso de la fabricación a través de la máquina fresadora, la herramienta determina si es posible realizar la placa de circuito impreso, o de lo contrario, si se viola alguna de las restricciones impuestas en el proceso de fabricación. Muchos fabricantes de placas de circuitos impresos proveen un archivo de configuración, con extensión *.dru, para cargarlo en la herramienta de chequeo.

Figura 49. **Ventana de DRC de EAGLE**



Fuente: elaboración propia, utilizando el programa EAGLE.

Finalmente, cuando se tiene el diseño completo y revisado a través de la herramienta de DRC de EAGLE, el diseño está listo para ser convertido y exportado a código G para ser enviado a la máquina fresadora.

4.1.2. Configuración de DRC de EAGLE para la máquina fresadora

En la tabla VI se muestra la configuración de la herramienta DRC de EAGLE para chequeo de los diseños de circuitos impresos a realizar en la máquina fresadora. La tabla muestra la configuración por pestañas, nombres de los campos y los valores correspondientes a cada uno. La configuración mostrada está hecha a partir de la configuración por defecto del programa, modificando solamente los campos detallados en la tabla VI.

Tabla VI. **Configuración de DRC de EAGLE**

Pestaña de DRC	Nombre del campo	Valor asignado
Layers	*Copper 1*	0,035 mm
	Copper 2	0,035 mm
Clearance	*Todos los campos*	13 mil
Distance	*Copper / dimension*	30 mil
	Drill / hole	8 mil
Sizes	*Minimum width*	10 mil
	Minimum drill	31 mil
Restring	*Pads top min*	21 mil
	Pads top %	25
	Pads top max	30 mil
	Pads bottom min	21 mil
	Pads bottom %	30
	Pads bottom max	30 mil
	Vias outer min	21 mil
	Vias outer %	25
	Vias outer max	30 mil

Fuente: elaboración propia.

4.1.3. Generación de código G

Para generar el archivo de código G necesario para el fresado a partir del diseño se utiliza un ULP de EAGLE que realiza dicha tarea, llamado *pcb-gcode*. El ULP obtiene a partir del diseño del circuito impreso y de la configuración utilizada el código G para fresar o cortar alrededor de las pistas de los circuitos en la placa de cobre virgen.

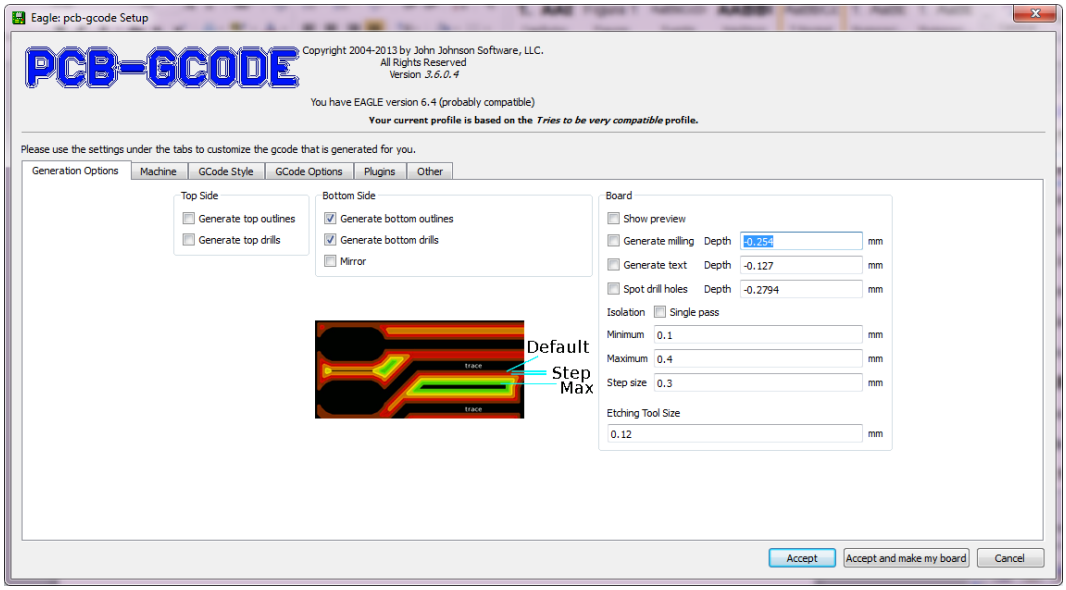
El ULP *pcb-gcode* también permite crear los archivos para barrenado de la placa de circuito impreso, archivos de corte para cortar la placa de circuito impreso y grabar texto en la superficie de la placa. El programa es gratuito y altamente configurable para obtener resultados óptimos. El ULP puede ser descargado de la página oficial de su autor y las instrucciones de instalación se encuentran dentro del archivo con el programa y el resto de la documentación.

Una vez instalado el ULP *pcb-gcode* se puede abrir la ventana de configuración del programa utilizando el comando *run pcb-gcode-setup* en la línea de comandos de la ventana de edición de circuitos impresos. Esta ventana de configuración permite seleccionar distintas opciones y formas de generar el código G para el fresado de la placa de circuito impreso. Es necesario realizar esta configuración al menos una vez, ya que las opciones de generación son específicas para cada máquina herramienta. Para generar el código G sin lanzar la ventana de configuración se puede utilizar el comando *run pcb-gcode* en la línea de comandos; para esto es necesario tener configurado el programa.

En la figura 50 se observa la ventana de configuración del ULP *pcb-gcode*. En esta ventana es posible configurar los archivos que se desean generar, los valores utilizados para separación de las pistas en la generación, tamaño de las herramientas, brocas disponibles para barrenado, distancias por defecto,

velocidades de corte, y opciones y estilos del código G generado. El ULP *pcb-gcode* incluye una extensa documentación en el directorio de instalación, donde es posible consultar el significado de cada uno de los posibles campos de configuración y escoger el valor óptimo a ser utilizado en la máquina fresadora.

Figura 50. **Ventana de configuración de la ULP pcb-gcode de EAGLE**



Fuente: elaboración propia, utilizando el programa EAGLE.

Los archivos de código G generado para la cara de atrás de las placas de circuitos impresos utilizan coordenadas negativas, es decir, el diseño de circuito impreso original es reflejado en el eje X. Esto es para que al fresar la placa y darle vuelta coincida con el diseño realizado.

4.1.3.1. Configuración de ULP pcb-gcode de EAGLE para máquina fresadora

En la tabla VII se muestra la configuración utilizada en la ULP *pcb-gcode* de EAGLE para la generación de archivos de código G para el fresado de placas de circuitos impresos en la máquina fresadora. La tabla muestra la configuración por pestañas, nombres de los campos y los valores correspondientes a cada uno.

Tabla VII. **Configuración de ULP *pcb-gcode* de EAGLE**

Pestaña de pcb-gcode	Nombre del campo	Valor asignado
Generation options	*Generate bottom outlines*	Casilla marcada
	Generate bottom drills	Casilla marcada
	Isolation single pass	Casilla marcada
	Isolation minimum	0,1 mm
	Isolation maximum	0,4 mm
	Isolation step size	0,3 mm
	Etching tool size	0,12 mm
Machine	*Units*	Milimeters
	Spindle spin up time	4 segundos
	Feed rates XY	85 mm/min
	Feed rates Z	60 mm/min
	Z High	5 mm
	Z Up	2 mm
	Z Down	-0,1 mm
	Drill depth	-1,5 mm
	Drill dwell	3
	Tool change position X	0 mm
	Tool change position X	0 mm
	Tool change position Z	10 mm
	Epsilon	0,00635 mm
	Drill rack file	default.drl

Continuación de la tabla VII.

GCode Style	*GCode Style*	generic.pp
GCode Options	*NC file comments*	Casillas desmarcadas
	Use user gcode	Casilla desmarcada
	Debug flag	Casilla desmarcada
	Do tool change with zero step	Casilla desmarcada
	Use line numbers	Casilla desmarcada
	Use simple drill code	Casilla marcada
	Campos de *File naming*	Valores por defecto

Fuente: elaboración propia.

En la tabla VII el campo *Isolation single pass* controla si el programa genera una sola trayectoria de corte para las pistas del circuito impreso. Si esta casilla se encuentra desmarcada el programa genera varias trayectorias de corte con el objetivo de separar más los cortes de las pistas entre sí y minimizar la posibilidad de corto circuito. Para esto toma las configuraciones de *Isolation mínimum*, *Isolation maximum* y *Isolation step size.* El campo *Etching tool size* corresponde al cálculo realizado en la sección 2.2 para una fresa en V de 60º de grabado.

El campo *Z Down* controla la profundidad de fresado del circuito impreso. Todo el código G generado para fresado del circuito está basado en la suposición de que la altura $Z = 0$ en el programa de control numérico corresponde a la superficie de la placa de cobre virgen. Por lo tanto, una altura $Z = -0{,}1$ corresponde a todos los puntos con una profundidad de 0,1 mm sobre la placa de cobre virgen, lo que permite el fresado para formar las pistas del circuito impreso.

Las configuraciones de *Drill rack file* y *GCode Style* son archivos de configuración que utiliza el programa de generación de código G para informarse de las brocas disponibles para el barrenado y el estilo del código G generado. El archivo *default.drl* es cargado en la ventana de configuración del ULP *pcb-gcode*. Earchivo *generic.pp* debe ser copiado en el directorio de instalación del ULP *pcb-gcode* para que la configuración tome efecto. Ambos archivos se muestran completos en la sección de anexos para su utilización.

El tamaño de la herramienta es configurado en el campo *Etching tool size*, y corresponde al diámetro de la herramienta de corte, que es utilizado por el programa para generar el espaciamiento de las pistas que conforman los circuitos impresos. La de medida de 0,12 mm corresponde al cálculo realizado para una profundidad de fresado de 0,1 mm utilizando una fresa de corte en V de 60º.

4.2. Software para control de la máquina fresadora

Se describen los programas para el control de la máquina herramienta y las configuraciones utilizadas.

4.2.1. Alternativas de *software* libre

Para el control de la máquina fresadora existen varios programas de control que permiten posicionar la máquina, cargar archivos de código G e interpretarlos, y proveer al usuario con una interfaz de control que permita conocer el estado de la máquina en todo momento. En esta sección se muestran los distintos programas que pueden ser utilizados para el control de la máquina fresadora, utilizando los diseños electrónicos mostrados en la sección de electrónica del proyecto.

Las programas utilizados son alternativas de *software* libre y de código abierto, es decir, que el software puede ser utilizado, copiado, estudiado, modificado y redistribuido libremente, ya que es posible acceder al código fuente de los programas. Esto es bastante útil para conocer cómo funciona el programa y agregar funcionalidad especial a un proyecto.

4.2.2. Linux CNC EMC2

Linux CNC es un sistema de *software* libre para el control computarizado de máquinas herramienta como fresadoras, tornos y cortadoras de plasma. Está liberado bajo una licencia GNU GPLv2. Linux CNC está conformado por el programa EMC2 (Enhanced Machine Control, actualmente en la versión 2.6.7) que incluye varias interfaces gráficas, un intérprete de código G de acuerdo con el estándar RS-274, un sistema de planeamiento de movimientos en tiempo real y operación de electrónica de bajo nivel como sensores y controladores de motores. Es precompilado y distribuido como una versión especial de Ubuntu para facilidad de instalación.

4.2.2.1. Capacidades del software

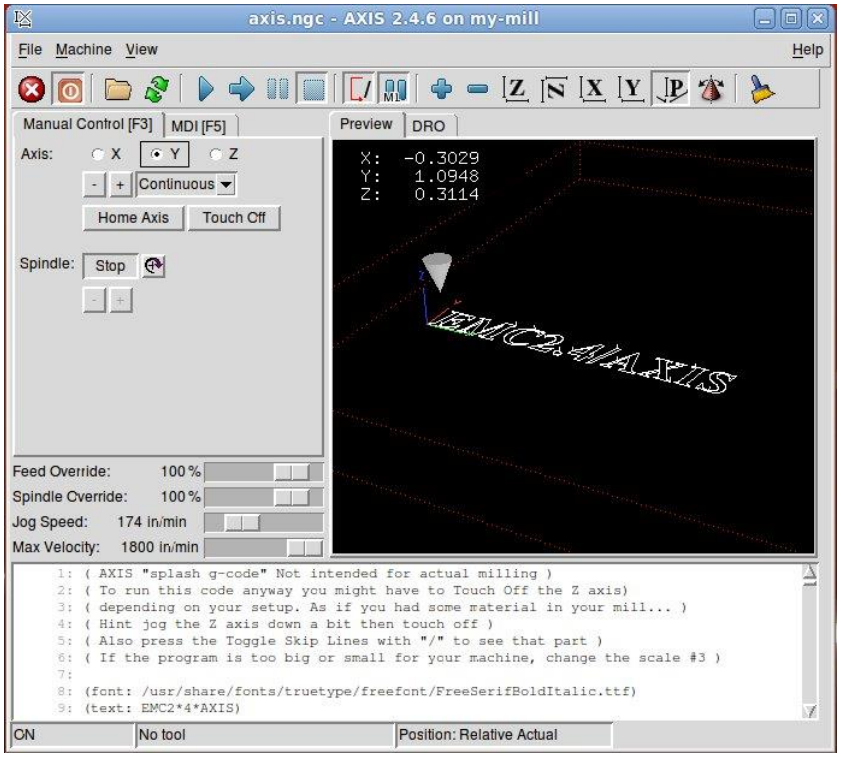
En la figura 51 se puede observar la ventana principal del programa de control de la máquina fresadora, llamado AXIS. En esta ventana es posible realizar las siguientes operaciones:

- Visualizar las trayectorias y la ubicación de la punta de la herramienta.
- La pesataña de DRO (*Digital read out*) permite conocer la ubicación de la herramienta de acuerdo al sistema de coordenadas cartesianas.
- Los controles manuales permiten controlar cada eje independientemente, activar o desactivar la herramienta rotativa, llevar cualquier eje a la

posición de *home*. Además es posible modificar la velocidad de rotación de la herramienta.

- La parte inferior de la ventana muestra una vista del código G a ser ejecutado por la máquina.
- Los iconos de acceso rápido en la parte superior de la ventana permiten apagar y encender la máquina, y reanudar o pausar la ejecución de código G.
- Es posible cargar archivos de código G utilizando la opción de *File -> Open.*

Figura 51. **Ventana principal de Linux CNC EMC2**



Fuente: *Linux CNC EMC2*.
https://www.buildyourcnc.com/item/control-SOFTWARE-linuxcnc.
Consulta: 2 de abril de 2015.

4.2.2.2. Requerimientos

Para que Linux CNC EMC2 funcione correctamente para controlar la máquina fresadora deben cumplirse los siguientes requerimientos de *hardware*:

- Se requiere una arquitectura de computadora x86.
- Un tamaño recomendado de memoria RAM de 512 MB.
- Un procesador mínimo Pentium II de 400 MHz como mínimo, sin embargo el desempeño de sistemas con motores paso a paso puede verse afectado, se recomienda utilizar procesadores más nuevos y rápidos.
- Un mínimo de 4 GB de disco duro para la instalación de EMC2.
- La tarjeta madre debe poseer un puerto paralelo con un conector DB-25.

Los requerimientos de *hardware* para correr EMC2 son relativamente bajos para los estándares de hoy en día, sin embargo existen problemas de incompatibilidad con computadoras portátiles, y algunas tarjetas de video integradas en computadoras que provocan un mal desempeño del sistema de tiempo real implementado por EMC2 para el control de la electrónica de la máquina herramienta.

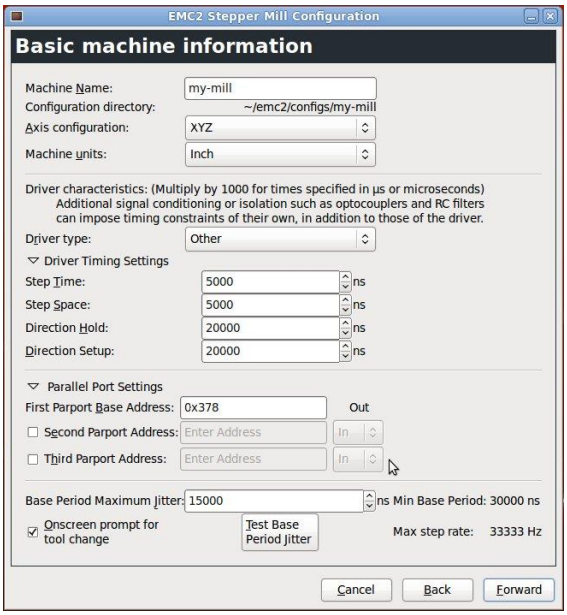
Para utilizar EMC2 con la interfaz de control para motores paso a paso se debe correr primero una prueba de latencia (*latency-test*) integrada en EMC2. El propósito de esta prueba es determinar si la máquina es apta para realizar tareas de tiempo real. Utilizando los resultados de la prueba de latencia puede determinarse que una computadora es muy buena para correr EMC2 cuando la prueba de latencia indica un tiempo máximo entre 15 y 20 microsegundos.

4.2.2.3. Configuración de la interfaz de control por puerto paralelo

Para utilizar EMC2 con la interfaz de control por puerto paralelo desarrollada en el capítulo 3 es necesario configurar el programa para que utilice la disposición de pines en la tarjeta de circuito impreso. Por simplicidad, no se muestra el proceso de instalación de EMC2, para esto puede consultarse la documentación oficial del programa en la página del proyecto.

Una vez la distribución de EMC2 en la computadora, la configuración se realiza en el programa *EMC2 Stepconf Wizard*. Si es la primera vez que se configura la máquina fresadora debe seleccionarse la opción *Create a new configuration* para crear un nuevo perfil. En la figura 52 se muestra la ventana de configuración de perfil para la máquina fresadora. En esta ventana se configura la información esencial acerca de la máquina, del puerto paralelo, y de los controladores de motores. En la tabla VIII se muestra la configuración utilizada para completar los campos en la ventana de configuración.

Figura 52. **Configuración de perfil para fresadora en EMC2**



Fuente: *Linux CNC EMC2*.
https://www.buildyourcnc.com/item/control-SOFTWARE-linuxcnc.
Consulta: 2 de abril de 2015.

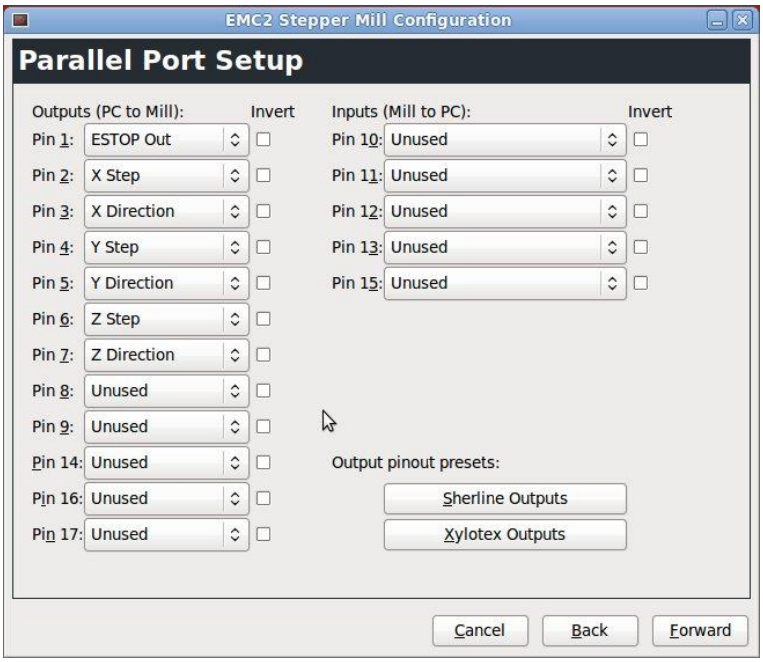
Tabla VIII. **Configuración de perfil para fresadora en EMC2**

Campo	Valor asignado
Axis configuration	XYZ
Machine units	mm
Driver type	Other
Step time	5000 ns
Step space	5000 ns
Direction hold	20000 ns
Direction setup	20000 ns
First parport base address	0x378
Base period maximum jitter	Valor resultante de la prueba de latencia

Fuente: elaboración propia, con base en la figura 52.

La siguiente ventana de configuración consiste en la disposición de pines del puerto paralelo hacia la interfaz de control. Esta ventana de configuración se muestra en la figura 53, donde debe seleccionarse la función de cada pin del puerto paralelo. Para completar la información requerida se puede consultar la tabla I en el capítulo 3, donde se muestra la disposición de pines del puerto paralelo en la interfaz de control.

Figura 53. **Configuración de puerto paralelo en EMC2**

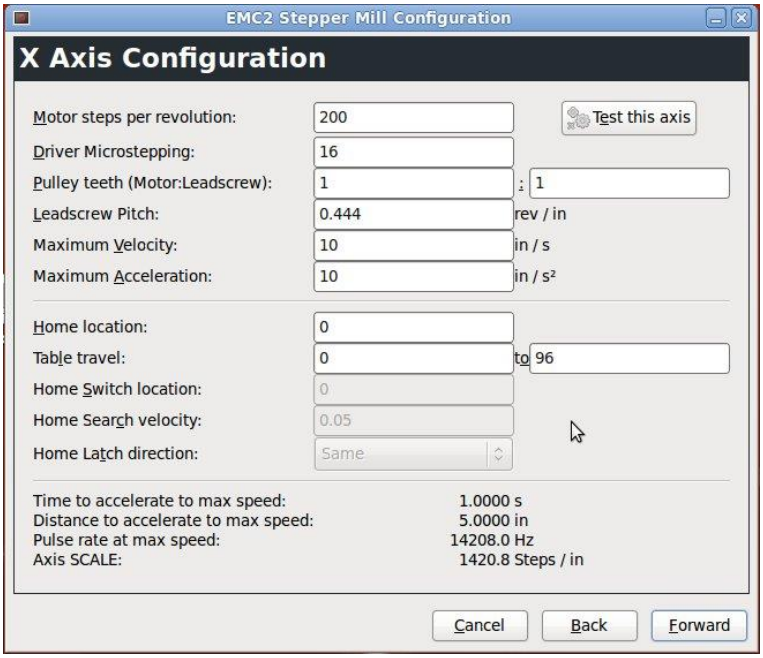


Fuente: *Linux CNC EMC2*.
https://www.buildyourcnc.com/item/control-SOFTWARE-linuxcnc.
Consulta: 2 de abril de 2015.

Finalmente, las siguientes ventanas de configuración se refieren a la configuración individual de cada eje de la máquina fresadora. En la figura 54 se observa la ventana de configuración del eje X, sin embargo, existe una ventana de configuración igual para los ejes Y y Z. En esta ventana es posible configurar

toda la información relacionada con la mecánica del eje, como el número de pasos por vuelta, máxima velocidad de desplazamiento, máxima aceleración y máximo recorrido del eje en la máquina fresadora.

Figura 54. **Ventana de configuración de motor en EMC2**



Fuente: *Linux CNC EMC2*.
https://www.buildyourcnc.com/item/control-SOFTWARE-linuxcnc.
Consulta: 2 de abril de 2015.

En la tabla IX se muestra la configuración común a los tres ejes de la máquina fresadora. El campo *Motor steps per revolution* es el número de pasos completos que necesita el motor para dar una vuelta, el valor asignado corresponde al motor paso a paso utilizado para la máquina fresadora. De forma independiente se configura el campo *Driver microstepping* para indicar que se está utilizando una secuencia de medio paso en el controlador del motor.

Tabla IX. **Configuración de ejes de la máquina en EMC2**

Nombre del campo	Valor asignado
Motor steps per revolution	200
Driver microstepping	2 (controlador configurado con medio paso)
Pulley teeth	1:1
Leadscrew Pitch	1,27 mm/rev
Home location	0
Table travel	-200 *to* 200

Fuente: elaboración propia.

El valor del campo *Leadscrew pitch* indica la cantidad de milímetros de desplazamiento con una vuelta del motor. Si se utiliza una configuración en pulgadas debe ingresarse el valor de vueltas por pulgada, ya que se utilizó una varilla roscada de 1/4 in, para la transmisión, con 20 vueltas por pulgada, el cálculo de este número para una configuración en milímetros corresponde a la siguiente conversión:

$$\frac{1\ in}{20\ rev} * \frac{25{,}4\ mm}{1\ in} = 1{,}27\ mm/rev$$

Los valores de velocidad máxima y aceleración máxima deben encontrarse por experimentación. Utilizando la opción de *Test this axis*, en la ventana de configuración es posible mover la máquina con una configuración de prueba que permite determinar, tras varias corridas experimentales, la velocidad y aceleración máxima para configurar la máquina.

Una vez configurados los tres ejes de la máquina, el programa de configuración guarda el perfil creado. Y con esto ya es posible abrir la ventana

de principal de Linux CNC EMC2 para el control de la máquina fresadora a través del puerto paralelo.

4.2.3. GRBL

GRBL es un proyecto de software libre que proporciona una alternativa de bajo costo a los sistemas de máquinas fresadoras controladas por puerto paralelo. La versión 0.9 y posterior está liberada bajo una licencia de software libre GPLv3. El proyecto ha sido adaptado para proyectos de máquinas fresadoras, cortadoras láser y troqueladoras. Debido a sus requerimientos de *hardware* el proyecto se ha vuelto de código abierto, mantenido por una comunidad en Internet.

4.2.3.1. Capacidades del software

El controlador está escrito en C altamente optimizado para utilizar inteligentemente cada característica de los chips AVR y alcanzar temporización precisa y operación asíncrona. Es capaz de mantener hasta 30 kHz de operación estable. GRBL incluye el completo manejo de aceleración con planeamiento de movimientos. Esto significa que el controlador busca hasta 18 movimientos en el futuro y planea las velocidades para entregar aceleración suave y sin tirones en las curvas.

GRBL acepta código G de acuerdo con el estándar RS-274, sin embargo no es posible utilizar todos los códigos G, como funciones macro, variables y ciclos fijos de mecanizado. Sin embargo, es posible utilizar programas externos a GRBL para procesar el código G requerido y traducirlo a código G básico para ser ejecutado por GRBL. Para más información de los códigos G disponibles en GRBL se debe visitar la página oficial del proyecto en internet.

GRBL es un intérprete de código G y controlador de la máquina fresadora, sin embargo no posee una interfaz gráfica para su utilización; para esto se debe utilizar una computadora con *software* diferente instalado. El acceso a GRBL es a través de una consola serial.

4.2.3.2. Requerimientos

Para instalar GRBL es necesaria una tarjeta de desarrollo Arduino con un microcontrolador ATmega328. También es posible instalar GRBL en otras tarjetas de desarrollo con microcontroladores AVR (como en la tarjeta Arduino Mega 2560), que son una familia de microcontroladores del fabricante Atmel. En la página del proyecto está disponible un archivo binario precompilado para cargarse en la tarjeta de desarrollo, pero también es posible compilar el código fuente del proyecto, modificando el código fuente y cargándolo al Arduino.

Para la interacción con GRBL es necesaria una computadora con puerto USB, con los controladores para la tarjeta de desarrollo Arduino instalados para tener acceso al puerto serial del microcontrolador. Finalmente, para la configuración de GRBL es necesario un programa de consola serial, como PuTTY, para enviar los comandos de configuración hacia el controlador.

4.2.3.3. Configuración de GRBL para la máquina fresadora

Para la máquina fresadora se utilizó GRBL en su versión 0.9g. Para conectarse a la consola serial se utiliza PuTTY en el puerto serial instalado utilizando una tasa de baudios de 115200. La configuración de GRBL se realiza enviando un comando de configuración a través de la consola serial.

El formato de los comandos de configuración es el siguiente:

$$\$x = val$$

Donde $\$x$ es el número de configuración, indicado en la página de documentación GRBL, y val es el valor asignado a la configuración. En la tabla X se muestra la configuración aplicada a GRBL para el control de la máquina fresadora. Los comandos de configuración pueden ser enviados uno a uno o cargando un archivo de texto con la configuración en la consola serial conectada a GRBL.

Tabla X. **Configuración de GRBL para máquina fresadora**

Nombre del campo	Número de configuración	Valor asignado
Step pulse, usec	$ 0	80
Step idle delay, ms	$ 1	255
Step port invert mask	$ 2	0
Dir port invert mask	$ 3	4
Step enable invert, bool	$ 4	1
Limit pins invert, bool	$ 5	0
Probe pin invert, bool	$ 6	0
Status report mask	$ 10	3
Junction deviation, mm	$ 11	0,05
Arc tolerance, mm	$ 12	0,1
Report inches, bool	$ 13	0
Auto start, bool	$ 14	1
Soft limits, bool	$ 20	0
Hard limits, bool	$ 21	1
Homing cycle, bool	$ 22	1
Homing dir invert mask	$ 23	2
Homing feed, mm/min	$ 24	180
Homing seek, mm/min	$ 25	180
Homing debounce, msec	$ 26	100

Continuación de la tabla X.

Homing pull-off, mm	$ 27	7
X, step/mm	$ 100	314,96
Y, step/mm	$ 101	314,96
Z, step/mm	$ 102	314,96
X max rate, mm/min	$ 110	300
Y max rate, mm/min	$ 111	150
Z max rate, mm/min	$ 112	150
X accel, mm/sec^2	$ 120	20
Y accel, mm/sec^2	$ 121	15
Z accel, mm/sec^2	$ 122	10
X max travel, mm	$ 130	200
Y max travel, mm	$ 131	200
Z max travel, mm	$ 132	200

Fuente: elaboración propia.

Las configuraciones mostradas en la tabla X no corresponden a una asignación de pines entre el software y la interfaz de control, porque esto es predeterminado en GRBL. En cambio las configuraciones corresponden únicamente a los valores de velocidad, aceleración y configuración del controlador de motor para cada eje de la máquina fresadora.

Las configuraciones que utilizan una máscara indican al software si los pines utilizados para lectura son leídos en estado lógico alto o bajo. Por ejemplo, el campo *Dir port invert mask* tiene un valor de 4, cuya representación binaria con 8 bits corresponde a 00000100, donde el 1 en la tercera posición de derecha a izquierda indica que la dirección positiva del eje X está invertida respecto de las direcciones positivas de los ejes Y y Z. Para el cálculo del resto de las máscaras se puede consultar la documentación oficial de GRBL, donde está explicado con mayor detalle.

El número de pasos por milímetro utilizado para todos los ejes es de 314,96 pasos/mm. Este número le indica a GRBL cuántos pasos debe enviar a los controladores de motores para desplazar un eje de la máquina un milímetro. Este número resulta de la utilización de una varilla roscada de 1/4 in (con 20 vueltas por pulgada) para la transmisión de la máquina fresadora, y de un motor paso a paso con 200 pasos por vuelta, utilizando la secuencia de medio paso. Puede obtenerse realizando la siguiente conversión:

$$\frac{400\ pasos}{rev} * \frac{20\ rev}{1\ in} * \frac{1\ in}{25{,}4\ mm} = 314{,}96\ pasos/mm$$

4.2.3.4. Software para envío de código G

Una vez obtenidos los archivos de código G para el fresado de un circuito impreso se deben enviar a GRBL para controlar la máquina fresadora. Ya que la interfaz hacia GRBL es una consola serial, los comandos deberían ser enviados uno por uno. Es por esto que se han desarrollado distintos programas que se encargan de gestionar el funcionamiento de GRBL, permitiendo enviar archivos de código G hacia GRBL, dando una vista previa del trabajo de mecanizado y de las coordenadas de la herramienta. Estos programas circulan por Internet y permiten que la comunicación y control de GRBL desde la computadora sea más fácil.

Uno de estos programas es GRBL Controller, que es un proyecto de *software* diseñado para enviar código G a máquinas CNC. El programa puede ser descargado gratuitamente, y permite interactuar directamente con GRBL a través de una línea de comandos, aunque también incluye un panel de control donde es posible controlar el movimiento de la máquina para llevarla a una posición inicial.

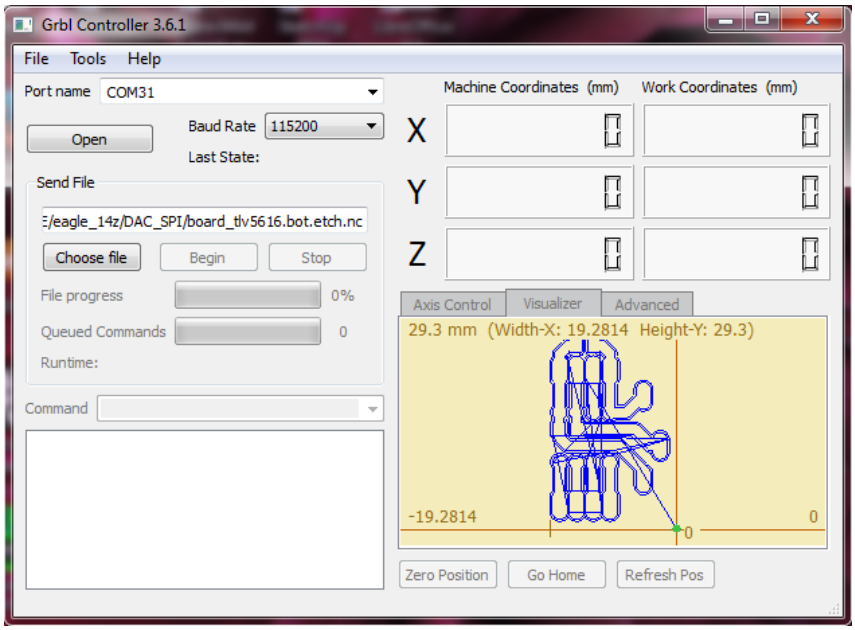
También es posible cargar archivos de código G para ser enviados a GRBL, y muestra una vista previa de la trayectoria a seguir por la máquina fresadora.

En la figura 55 se muestra la ventana principal de GRBL Controller, donde se carga el archivo de código G generado utilizando el diseño mostrado en el editor de circuitos impresos de EAGLE de la figura 48. Una vez cargado el código G se inicia la conexión con GRBL presionando el botón *Open* en la parte superior de la ventana. La máquina debe ser posicionada de forma manual utilizando la línea de comandos incluida, y para fresar el circuito impreso se debe colocar la fresa en la herramienta rotativa.

Finalmente, la máquina se posiciona en un punto sobre la placa de cobre virgen que sirva como origen del sistema de coordenadas. Utilizando el botón *Zero Position* se envía el código G correspondiente a poner la posición física de la máquina como el origen del sistema de coordenadas. Una vez la máquina esté lista, se procede con el envío del programa de código G hacia GRBL, que se encarga de interpretar y mover la máquina para realizar el trabajo de mecanizado.

Existen muchos otros programas para la tarea de envío de código G a GRBL, desde simples programas de consola hasta programas que montan un servicio web para acceder desde cualquier dispositivo conectado a la red del servidor que maneja la máquina herramienta. Entre este tipo de proyectos puede destacarse el proyecto Octoprint para GRBL.

Figura 55. **Ventana principal de GRBL Controller**



Fuente: elaboración propia, con base en la figura 48.

Octoprint es un proyecto de software libre para manejo de impresoras 3D a través de una interfaz web. La singularidad del proyecto es que puede ser montado en una computadora Raspberry Pi, que es una computadora de bajo costo, y del tamaño de una tarjeta de crédito, que puede ser conectada a la red y corre distribuciones optimizadas de GNU/Linux. Existe una versión modificada de Octoprint para controlar GRBL, y es posible instalar este programa en la computadora Raspberry Pi para controlar la máquina fresadora virtualmente, desde cualquier dispositivo con acceso a la red.

Este esquema de utilización de la máquina fresadora puede resultar particularmente útil cuando se considera que muchos trabajos de mecanizado pueden llevar horas en completarse. Utilizando una Raspberry Pi, el consumo energético del sistema se ve reducido drásticamente, además, es posible

controlar la máquina herramienta desde la computadora o un dispositivo móvil, como una tableta o un teléfono inteligente, y monitorear o documentar el funcionamiento utilizando una cámara web, que envíe la información de la cámara a través de la red hacia el dispositivo móvil.

Este esquema de utilización y monitoreo de la máquina herramienta con la Raspberry Pi está fuera de los objetivos del proyecto, sin embargo, se deja planteado al lector debido al crecimiento en la tendencia del “internet de las cosas” (IoT por sus siglas en inglés), que se refiere a la interconexión digital de objetos de uso cotidiano a través de internet.

4.3. Propuesta de programas de optimización de código G

Se muestra el desarrollo y utilización de dos programas de optimización escritos en Python para mejorar el desempeño de la máquina mientras ejecuta el código G para el fresado de circuitos impresos. Python es un lenguaje de programación interpretado y de código abierto. Entre sus principales ventajas se encuentran: Python utiliza una sintaxis que favorece el código legible, es multiparadigma y multiplataforma, es decir, los programas pueden ejecutarse en distintos sistemas operativos.

Los dos programas utilizan como entrada un archivo de código G y generan un archivo de salida con el código G mejorado. Los dos programas de optimización desarrollados son:

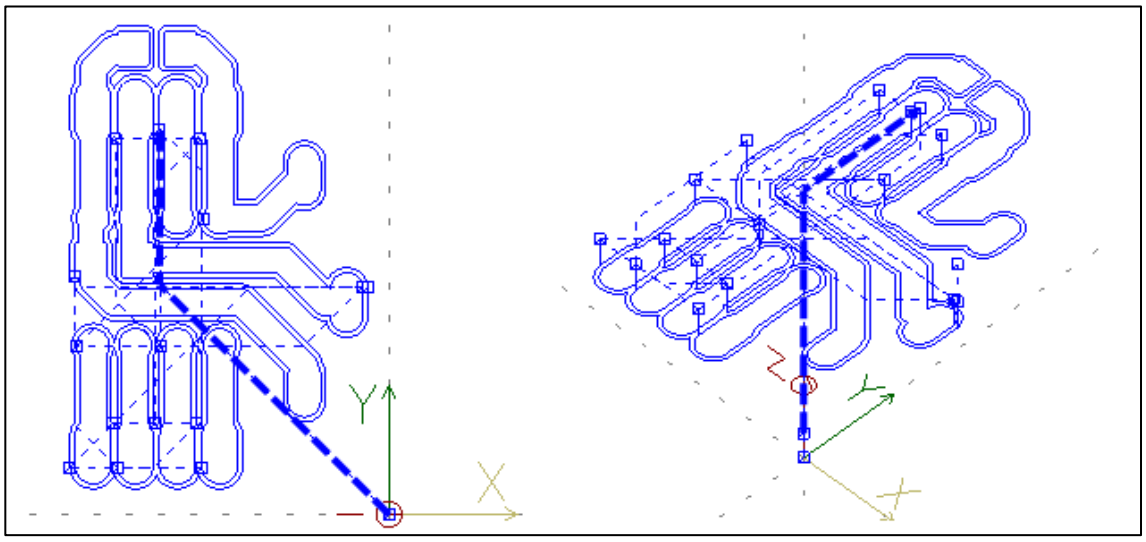
- Un programa de optimización de trayectorias para mejorar el tiempo de fresado de las placas de circuitos impresos.

- Un programa que realiza mediciones sobre la placa de cobre virgen para determinar la superficie que la conforma, y en base a esta, corregir el código G para fresar a una altura adecuada.

4.3.1. Programa de optimización de trayectorias

Cuando se genera el código G a partir del diseño de circuito impreso, el ULP *pcb-gcode* de EAGLE toma cada una de las líneas que conforman las pistas y genera trayectorias en un sistema de coordenadas para convertir a código G. Al convertir el diseño de circuito impreso mostrado en la figura 48 se obtiene como resultado un archivo de código G con un conjunto de trayectorias de mecanizado, mostrado en la figura 56.

Figura 56. **Diseño de circuito impreso convertido a código G**



Fuente: elaboración propia, con base en la figura 48.

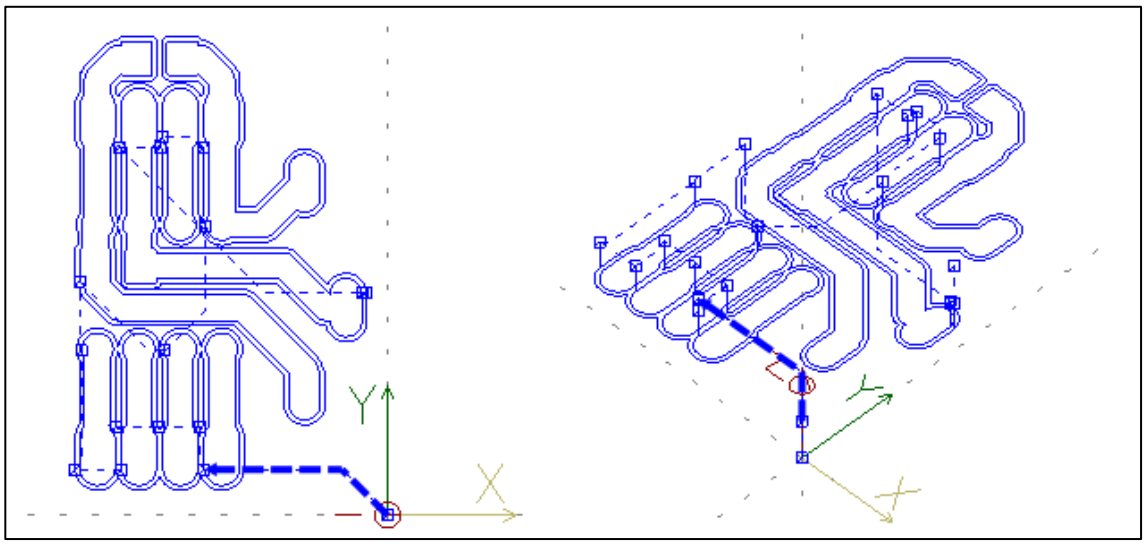
Como se muestra en la figura 56, cada una de las trayectorias consiste en el movimiento de la herramienta hacia un punto de inserción en la placa de circuito impreso, luego la herramienta baja al plano de trabajo, sobre la placa de circuito impreso y se mueve cortando mientras forma la pista. Cuando la herramienta vuelve al punto de inserción, la pista queda terminada y la herramienta se levanta para buscar un nuevo punto de inserción. Este es el proceso para separar cada una de las pistas de la placa de circuito impreso.

En la figura 56 se muestra resaltada la primera trayectoria a ser realizada por la máquina. Asimismo, es posible observar que esta no es la trayectoria cuyo punto de inserción está más cerca del origen. Y así, con las demás trayectorias, el código G generado no recorre los puntos de inserción de las trayectorias utilizando los puntos más cercanos. Esto provoca que la máquina herramienta tarde mucho más realizando el trabajo de mecanizado, que si estuviera utilizando los puntos de inserción más cercanos entre sí.

Para solucionar este problema se desarrolló un programa en Python que utiliza el archivo de código G generado por el ULP *pcb-gcode* para buscar las trayectorias y sus puntos de inserción.

El programa utiliza la lista de puntos y parte del origen como punto de inserción para determinar el siguiente punto de inserción más cercano en todo momento. El código fuente del programa desarrollado puede consultarse en la sección de anexos. En la figura 57 se muestra el resultado de aplicar el programa al archivo de código G mostrado en la figura 56.

Figura 57. **Código G para circuito impreso optimizado**



Fuente: elaboración propia, con base en la figura 48.

Como se puede observar en la figura 57, la primera trayectoria se encuentra resaltada nuevamente, y esta utiliza el primer punto de inserción más cercano al origen. La segunda trayectoria utiliza el punto de inserción más cercano al primero, y así en adelante. El código G se encuentra optimizado en distancia para el fresado de la placa de circuito impreso, y ya que no se modifican parámetros de velocidad en el programa, el tiempo debe reducirse.

Para utilizar el programa se utiliza el siguiente comando en la consola del sistema:

```
python cncopt.py <archivo>
```

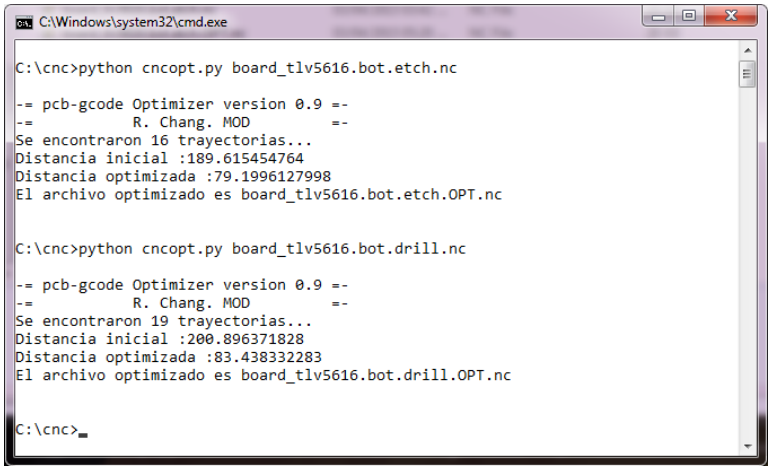
Donde se llama al intérprete de Python y se ejecuta el programa `cncopt.py`, dando como argumento del programa el nombre del archivo de

código G a ser optimizado. El programa genera las nuevas trayectorias y guarda un nuevo archivo en el mismo directorio del archivo original, agregando una extensión *.OPT al archivo original.

En la figura 58 se observa la utilización del programa en la consola de Microsoft Windows 7. Como se muestra, se carga el archivo de código G generado con el ULP pcb-gcode para el fresado y barrenado de la placa de circuito impreso, y el programa genera los nuevos archivos optimizados. Una vez obtenido el archivo de código G optimizado, está listo para ser enviado al intérprete de código G y trabajar la placa de circuito impreso.

Para el código G de fresado de la placa de circuito impreso el programa contabiliza un recorrido de 189,61 mm para que la máquina pueda fresar el circuito, mientras que con el código G optimizado necesita recorrer 79,19 mm para el fresado de la placa. Esto representa una disminución en tiempo de fresado del 58 %, es decir, menos de la mitad del tiempo de fresado original.

Figura 58. **Salida estándar del programa de optimización**



Fuente: elaboración propia.

4.3.2. Programa de medición y corrección de alturas

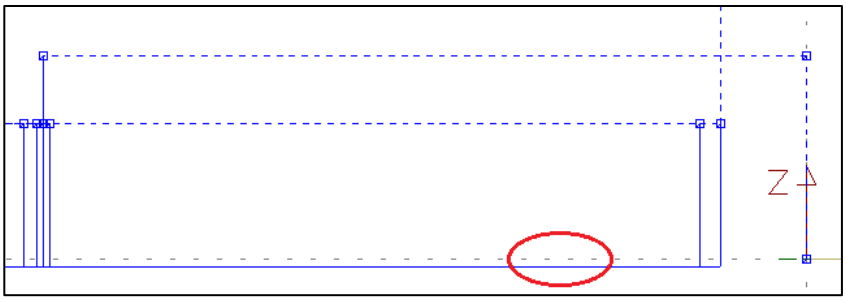
El fresado de placas de circuitos impresos consiste en que la herramienta de corte entre la superficie de la placa de cobre virgen y se mueva a lo largo de la placa para cortar las pistas y separarlas entre sí. Esta es la forma en que el código G es generado a partir del diseño del circuito impreso.

Utilizando las coordenadas cartesianas de la máquina fresadora, esto es equivalente a que la herramienta se desplace sobre la superficie de un plano modelado matemáticamente por:

$$z = -p$$

Donde z corresponde a la coordenada de altura de la herramienta de corte y p es una constante positiva que representa la profundidad de fresado del circuito impreso sobre la placa de cobre virgen. En la figura 59 se observa el código G generado para el diseño de la figura 48. La línea punteada del origen representa el plano $z = 0$, que físicamente sería la superficie de cobre de la placa. La línea inferior en azul representa el plano de fresado utilizado para realizar separar las pistas del circuito impreso.

Figura 59. **Plano de fresado de placa de circuito impreso**



Fuente: elaboración propia, con base en la figura 48.

La distancia entre la superficie de cobre y el plano de fresado es generalmente muy pequeña, del orden de decenas de milímetros. Debido a la construcción de las máquinas herramienta, o a la utilización de tablas de sacrificio para el fresado de circuitos impresos, pueden existir pequeños desniveles sobre la superficie de trabajo de la máquina. Al fijar la placa de cobre virgen a la máquina; estos desniveles provocan que la superficie de la placa no sea paralela a los planos formados por el desplazamiento de la herramienta.

En otras palabras, la placa de cobre puede presentar una superficie irregular, no representada precisamente por un plano $z = -p$ con una única profundidad de fresado. Esto provoca que en el fresado de circuitos impresos, ciertas regiones sobre la placa no sean alcanzadas por la herramienta cuando esta baja, y por lo tanto las pistas dibujadas sobre la placa no son realmente cortadas. También puede darse el caso contrario, donde la profundidad de fresado excede la profundidad necesaria y se corta demasiado de la placa de cobre.

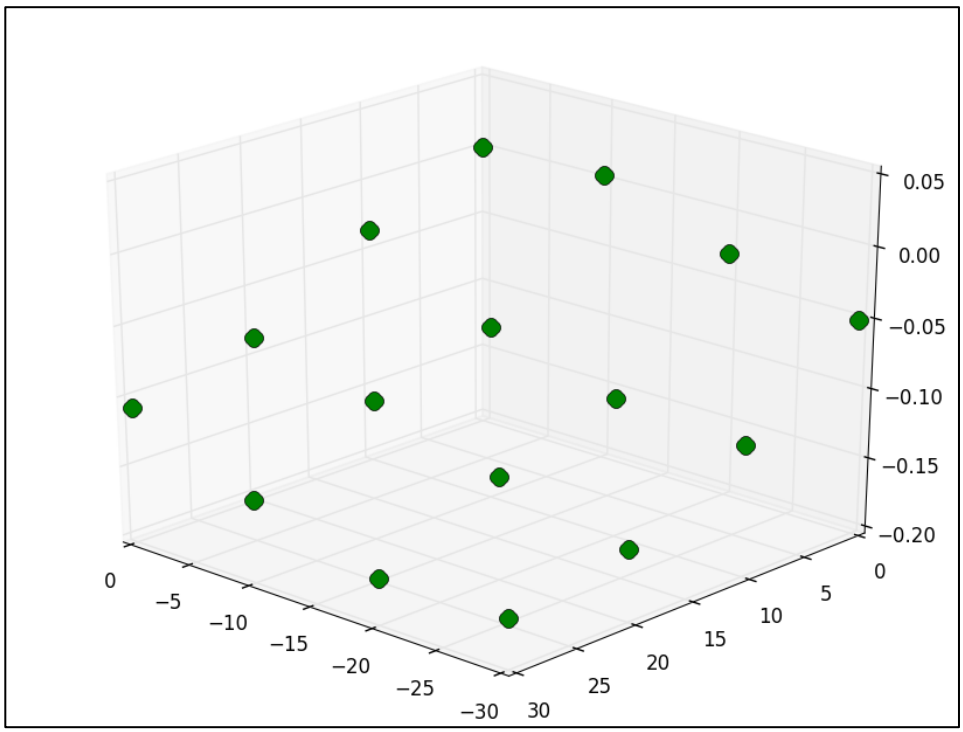
Esto es claramente un problema, ya que requiere ajustar manualmente el código G sobre ciertas trayectorias para asegurarse que la herramienta corte a la profundidad justa. De acuerdo con la experimentación, es muy difícil lograr que la placa de cobre virgen sea sujetada de forma exactamente paralela, aun si la máquina no presenta un desnivel, los elementos de sujeción de la placa de cobre siempre terminan por curvarla.

Utilizando el pin de prueba en la interfaz de control descrita en el capítulo 3, y un par de lagartos de conexión, se coloca uno a la placa de cobre y otro a la fresa de corte. Con la ayuda del código G38.2 es posible realizar una medición de altura. El código G38.2 mueve la fresa con el lagarto hacia la placa de cobre virgen, y cuando la toca se cierra un circuito eléctrico formado por los

lagartos, señalizando a la interfaz de control para detener el movimiento. Ya que la máquina herramienta tiene una gran precisión de movimiento, es posible medir la diferencia de alturas en una placa de cobre virgen.

El resultado de este procedimiento se ilustra en la figura 60, donde se realiza una medición de alturas sobre un área de 30 mm por 30 mm utilizando una cuadrícula de 10 mm. Las dimensionales de los ejes son milímetros. El punto de referencia es el punto (0,0), cuya altura es definida como cero, y a partir de esta altura se mide el resto de alturas. Se puede observar claramente que los puntos distribuidos en el área de medición no forman un plano de la forma $z = -p$.

Figura 60. **Medición de alturas en placa de cobre virgen**



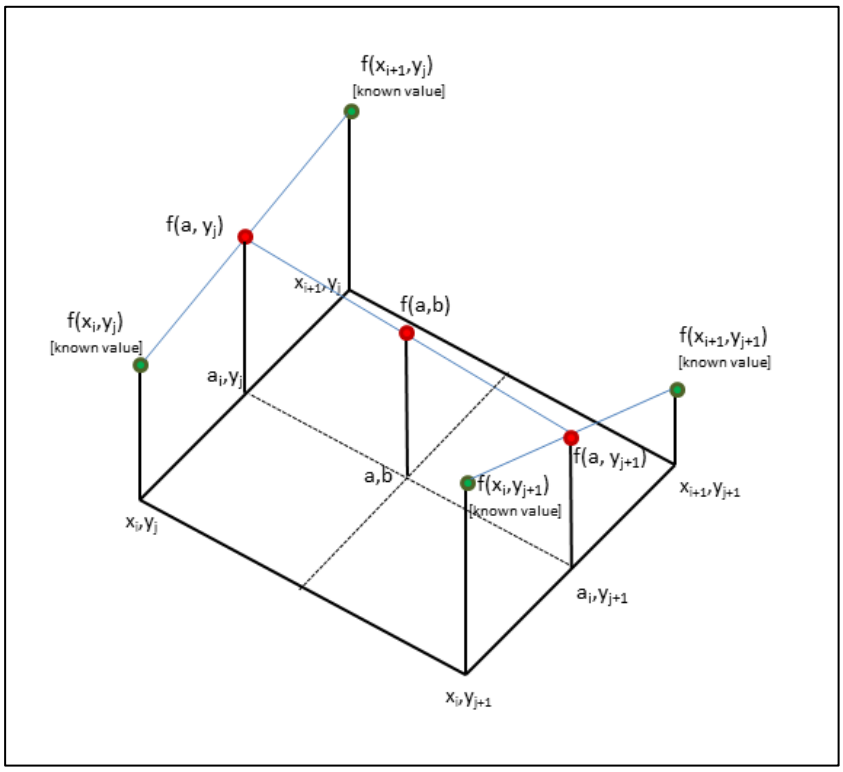
Fuente: elaboración propia, utilizando el lenguaje Python.

A modo de ejemplo, si se utilizara un archivo de código G para fresar una placa sobre la superficie de 30 mm por 30 mm a una profundidad constante de 0,1 mm, entonces el fresado no sería inefectivo en los puntos debajo del plano z =0,1 mm, por ejemplo, el punto (-30,30).

Para corregir este problema en el fresado de placas de circuitos impresos se desarrolló un programa en Python que realiza mediciones en una cuadrícula regular sobre la superficie de la placa de cobre y a partir de los puntos obtenidos, aplica un método de interpolación bilineal para generar la ecuación que representa a la superficie en cualquier punto, y con la ecuación es posible corregir cualquier coordenada especificada en un archivo de código G.

El método de interpolación bilineal es una extensión de las interpolaciones lineales para funciones de dos variables en una cuadrícula regular. En la figura 61 se ilustra el funcionamiento del método. La idea principal es realizar dos interpolaciones lineales en una dirección, y una interpolación más en la otra dirección con los cuatro puntos conocidos. Cada uno de los pasos resulta en una interpolación lineal, pero la interpolación completa para cualquier punto dentro de la cuadrícula es cuadrática.

Figura 61. **Método de interpolación bilineal**



Fuente: *Bilinear interpolation*. http://pixshark.com/bilinear-interpolation.htm. Consulta: 3 de abril de 2015.

El programa puede encontrarse en la sección de anexos. Para utilizarlo es necesario tener instalados los módulos Numpy, Scipy y Matplotlib de Python para los cálculos de transformación bilineal y la gráfica de puntos medidos. Para acceder al puerto serial de la interfaz de control se requiere el módulo Serial de Python.

Para utilizar el programa se puede correr a través de la consola del sistema, y se debe especificar el archivo de código G a corregir, las dimensiones del área de medición y el tamaño de la cuadrícula, y la profundidad

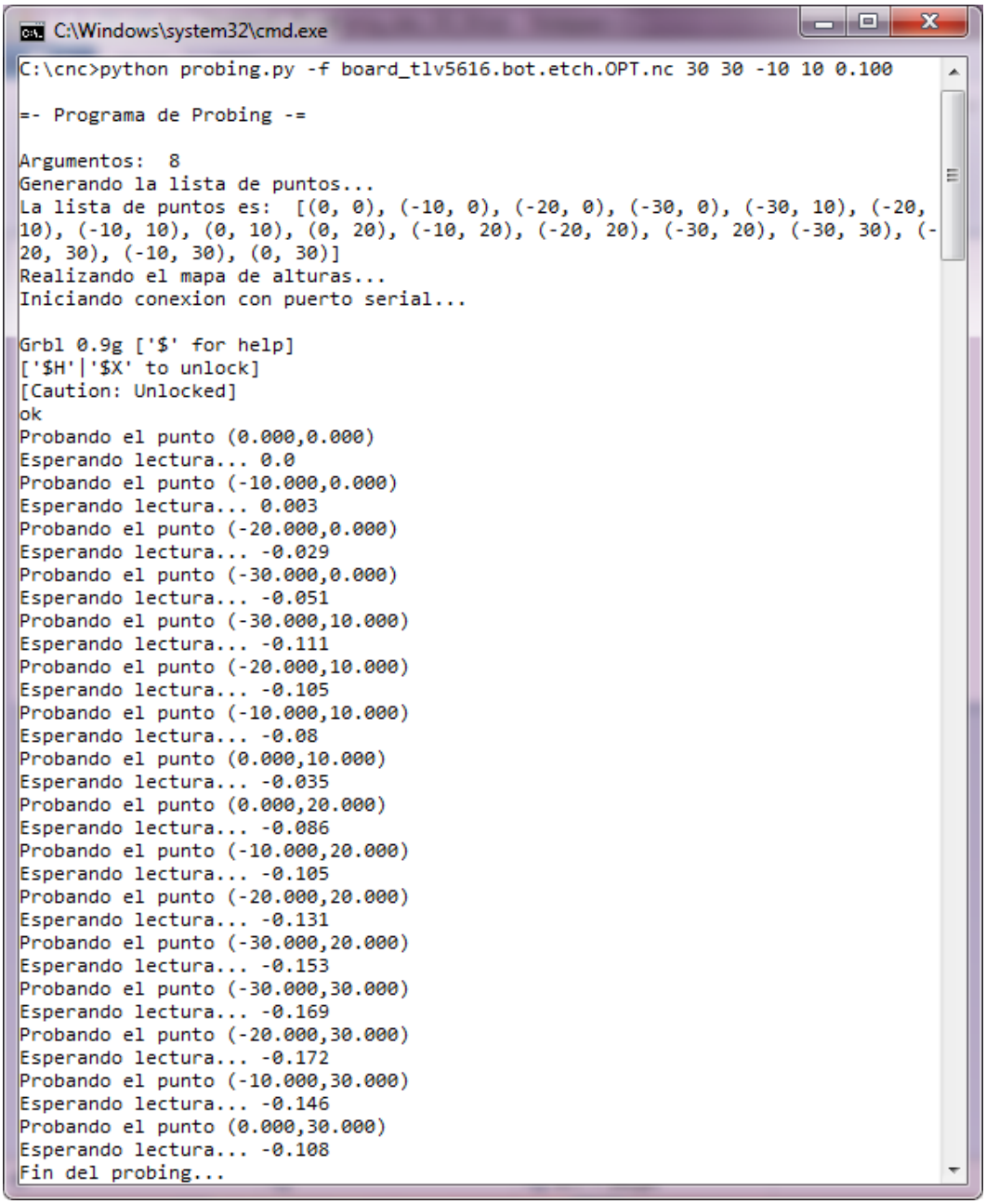
de fresado a partir de la superficie. El formato de la línea de comandos es el siguiente:

```
python probing.py -f <archivo> <x> <y> <dx> <dy> <prof_z>
```

Donde `<archivo>` indica el nombre del archivo con el código G a corregir, `<x>` y `<y>` especifican el área sobre la cual se quiere medir, `<dx>` y `<dy>` especifican el tamaño horizontal y vertical de la cuadrícula respectivamente, y `<prof_z>` especifica la profundidad de fresado a ser reemplazada por el programa en el archivo de código G.

El programa toma los argumentos de la línea de comandos y genera una lista de puntos para realizar mediciones. El área especificada por los valores `<x>` y `<y>` debe contener a todos los puntos del archivo de código G, ya que de lo contrario se producirá un error en la corrección del código G. Además, los valores `<dx>` y `<dy>` pueden ser valores negativos que representan el avance de la máquina herramienta mientras realiza las mediciones. El valor `<dx>` es generalmente un valor negativo, ya que el fresado de placas de circuitos impresos de una cara utiliza la cara de cobre para realizar las mediciones, y esta se ubica en el reverso de la placa.

Figura 62. **Salida estándar del programa de corrección de alturas**



Fuente: elaboración propia.

En la figura 62 se muestra la salida estándar en la consola del sistema cuando se ejecuta el programa de medición y corrección de alturas. El programa corrige el código G para el diseño de la figura 48. Con un área de 30 mm por 30 mm se pueden cubrir todos los puntos del código G del archivo. El tamaño de la cuadrícula se especifica como -10 y 10, que indica un avance negativo de 10 mm sobre el eje X, y positivo de 10 mm sobre el eje Y.

El programa genera la lista de puntos y empieza a controlar la máquina para realizar las mediciones. Cuando completa las mediciones, calcula y corrige todas las líneas de código G del archivo especificado. Como resultado, se crea un nuevo archivo con el mismo nombre del archivo de entrada, agregando el prefijo *.LEV. Este archivo está listo para ser enviado al intérprete del código G.

CONCLUSIONES

1. Se determinó que un sistema de control numérico computarizado es una forma eficiente de controlar máquinas herramienta para fresar placas de circuitos impresos.

2. Utilizando varillas de acero galvanizado de 1/4 in, con un paso de 20 vueltas por pulgada se obtuvo una precisión de movimiento de la máquina fresadora de 3 micrones.

3. Empleando una fresa de grabado en V de carburo de tungsteno de 30º se obtiene una precisión de corte de 0,1 mm en el fresado de placas de circuitos impresos.

4. Aplicando la técnica de control de corriente para motores paso a paso se controló el torque dinámico de los motores, manteniendo el calentamiento y el consumo de energía al mínimo.

5. Usando la interfaz de control por puerto paralelo se pueden manejar máquinas herramienta utilizadas en la industria.

6. La interfaz de control para Arduino es utilizada en máquinas fresadoras utilizadas en laboratorios de mecánica eléctrica para prototipado de placas de circuitos impresos y mecanizado de piezas.

7. Para reducir el tiempo de mecanizado de la máquina herramienta se debe emplear el programa de optimización de trayectorias de código G para fresado y barrenado de placas de circuitos impresos.

8. Para eliminar el desnivel en la mecánica de sujeción de la placa hacia la máquina herramienta durante el fresado de placas de circuitos impresos se debe utilizar el programa de medición y corrección de alturas.

RECOMENDACIONES

1. Para la persona que construye la máquina fresadora: durante la construcción de la máquina herramienta tomar en cuenta que no exista holgura en el mecanismo de desplazamiento de los ejes de la máquina, como se describe en la sección 2.1.2, para garantizar la precisión de los trabajos de mecanizado.

2. Para el operario de la máquina: tomar en cuenta la colocación de forma periódica de aceite en las varillas roscadas y carros móviles de la máquina fresadora mostrados en la sección 2.1.2 y 2.1.3, para evitar el sobrecalentamiento de la máquina durante su funcionamiento.

3. Para la persona que configura el equipo electrónico: durante la primera utilización del equipo electrónico es importante realizar el ajuste de corriente para cada uno de los motores paso a paso a un 70 % de la corriente nominal del motor para evitar el sobrecalentamiento. El procedimiento para realizar este ajuste se puede encontrar en la sección 3.1.2.

4. Para el operario de la máquina: durante la configuración de la máquina herramienta en la computadora es importante utilizar una configuración de velocidad de 100 mm/min, y una aceleración de 5 mm/s^2 para evitar daños en los mecanismos de desplazamiento de la máquina. Para configurar estos valores se puede referir a la sección 4.2.2.3 para la configuración utilizando una interfaz de puerto paralelo, o a la sección 4.2.3.3 para la configuración, utilizando una interfaz con Arduino.

5. Para el diseñador de placas de circuitos impresos: durante la etapa de diseño de placas de circuitos impresos tomar en cuenta, con ayuda de la herramienta de reglas de diseño (DRC), que la mecanización del circuito impreso sea posible, como se muestra en la sección 4.1.1 y 4.1.2.

6. Para la persona que configura la máquina: es importante revisar que los códigos de desplazamiento enviados desde la computadora para una distancia predefinida correspondan a un desplazamiento real de las unidades especificadas para cualquiera de los ejes. Esto puede hacerse utilizando la ventana principal de Linux CNC EMC2 en el caso de utilizar una interfaz de puerto paralelo; y para la interfaz de control con Arduino considerar el *software* de envío de código G descrito en la sección 4.2.3.4.

7. Para el operario de la máquina: es importante fijar fuertemente la placa de cobre virgen a la superficie de trabajo al utilizar la máquina herramienta, para evitar que esta se desprenda mientras se realiza el trabajo de mecanizado.

8. Para el operario de la máquina: tomar las precauciones y medidas de seguridad necesarias al utilizar la máquina herramienta, y no intervenir directamente cuando esté en funcionamiento.

BIBLIOGRAFÍA

1. *Asistente de configuración Stepconf*. [en línea]. <http://linuxcnc.org/docs/html/config/stepconf_es.html>. [Consulta: 2 de abril de 2015].

2. *Drive circuit basics*. [en línea]. <http://users.ece.utexas.edu/~valvano/Datasheets/StepperDriveBasic.pdf>. [Consulta: 13 de noviembre de 2014].

3. *EAGLE ERC*. [en línea]. <http://web.mit.edu/xavid/arch/i386_rhel4/help/44.htm>. [Consulta: 2 de abril de 2015].

4. *GRBL*. [en línea]. <https://github.com/grbl/grbl>. [Consulta: 3 de abril de 2015].

5. _____. [en línea]. <http://zapmaker.org/projects/grbl-controller-3-0/>. [Consulta: 3 de abril de 2015].

6. _____. [en línea]. <https://github.com/grbl/grbl/wiki>. [Consulta: 3 de abril de 2015].

7. *High resolution microstepping driver with the DRV88xx series.* [en línea]. <http://www.ti.com/lit/an/slva416/slva416.pdf>. [Consulta: 25 de noviembre de 2014].

8. *Hybrid stepper motors*. [en línea]. <http://users.ece.utexas.edu/~valvano/Datasheets/Stepper.pdf>. [Consulta: 13 de noviembre de 2014].

9. KELLY, James. HOOD-Daniel, Patrick. *Build Your Own CNC Machine*. Estados Unidos: Apress, 2009. 240 p. ISBN: 978-1430224891.

10. *L297 Stepper motor controller*. [en línea]. <http://malatok.at.ua/_fr/0/l297l298.pdf>. [Consulta: 8 de febrero de 2015].

11. *L297 Stepper motor controllers.* [en línea]. <http://www.st.com/web/en/resource/technical/document/datasheet/CD00000063.pdf>. [Consulta: 8 de febrero de 2015].

12. *L298 Dual Full-Bridge Driver*. [en línea]. <http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/CD00000240.pdf>. [Consulta: 8 de febrero de 2015].

13. *LinuxCNC (EMC2)*. [en línea]. <https://www.buildyourcnc.com/item/control-SOFTWARE-linuxcnc>. [Consulta: 2 de abril de 2015].

14. *LinuxCNC Hardware Requirements*. [en línea]. <http://wiki.linuxcnc.org/cgi-bin/wiki.pl?Hardware_Requirements>. [Consulta: 2 de abril de 2015].

15. *NEMA Motor*. [en línea]. <http://reprap.org/wiki/NEMA_Motor>. [Consulta: 25 de noviembre de 2014].

16. OVERBY, Alan. *CNC Machining Handbook*. Estados Unidos: McGraw-Hill, 2010. 272 p. ISBN: 978-0071623018.

17. *PCB basics*. [en línea]. <https://learn.sparkfun.com/tutorials/pcb-basics>. [Consulta: 4 de abril de 2015].

18. *PCB-GCode*. [en línea]. <http://www.pcbgcode.org/>. [Consulta: 2 de abril de 2015].

19. SEDRA, Adel S. SMITH, Kenneth C. *Microelectronic circuits*. 6a ed. Oxford University Press, 2009. 1456 p. ISBN: 978-0195323030.

20. SMID, Peter. *Numerical control*. *CNC Programming Handbook*. 2a ed. Estados Unidos: Industrial Press Inc., 2002. 497p. ISBN: 978-0831131586.

21. *Stepper motor basics*. [en línea]. <http://users.ece.utexas.edu/~valvano/Datasheets/StepperBasic.pdf>. [Consulta: 13 de noviembre de 2014].

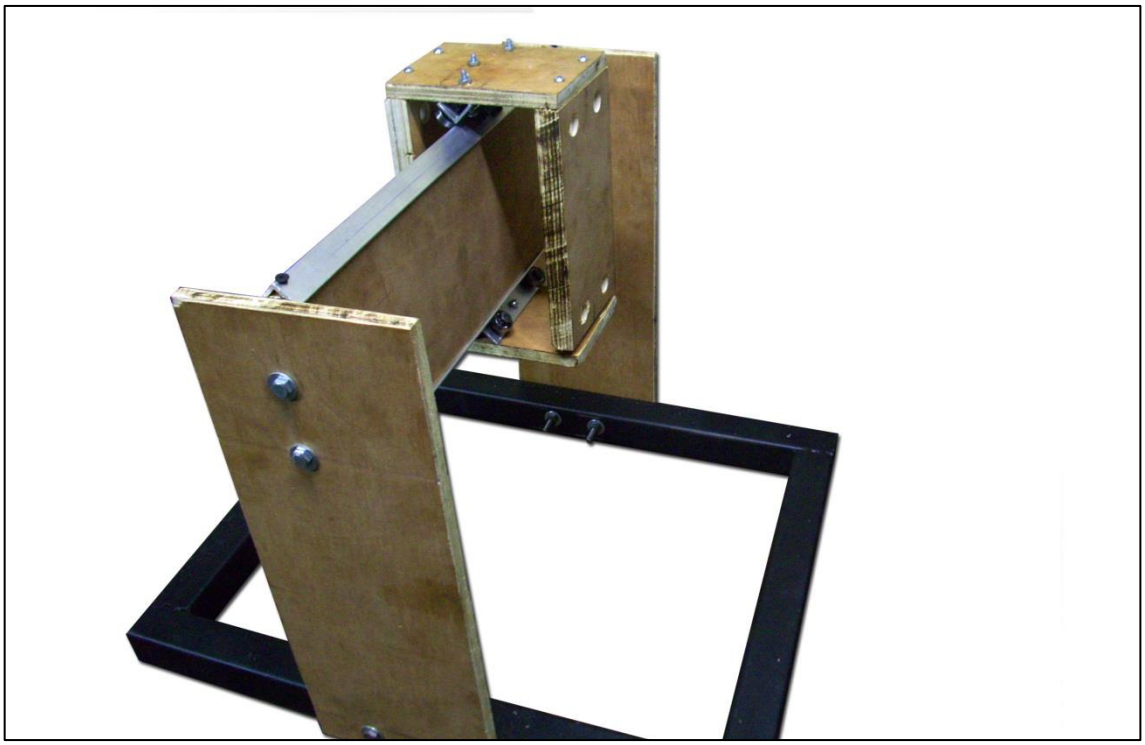
22. VALVANO, Jonathan. *Stepper Motor Interface*. *Embedded Systems: Real-Time Interfacing to ARM® Cortex™-M Microcontrollers*. 3a ed. Estados Unidos: 2013. 584 p. ISBN: 978-1463590154.

ANEXOS

Anexo 1. **Fotografías del proyecto**

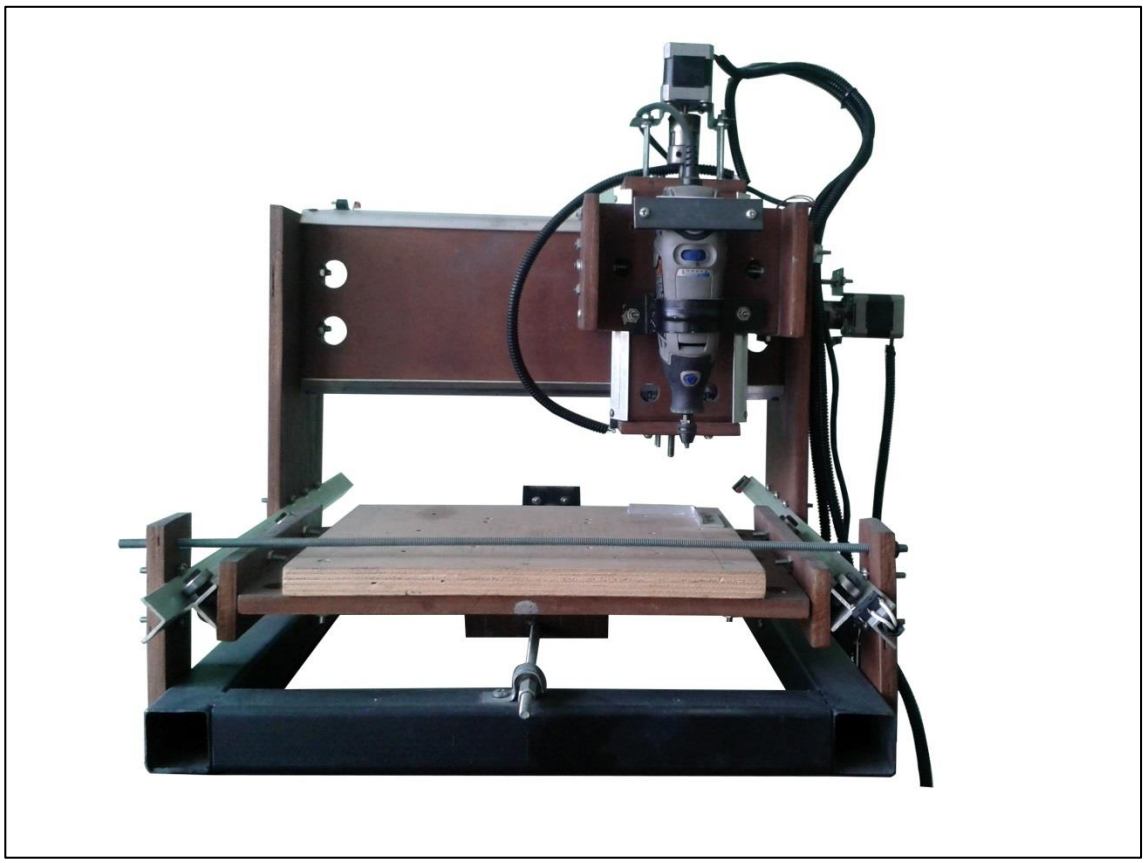
Se muestran las fotografías del proceso de construcción de la máquina fresadora, el equipo electrónico terminado y el acabado final de los trabajos de mecanización.

Anexo 1a. **Construcción de la máquina fresadora**



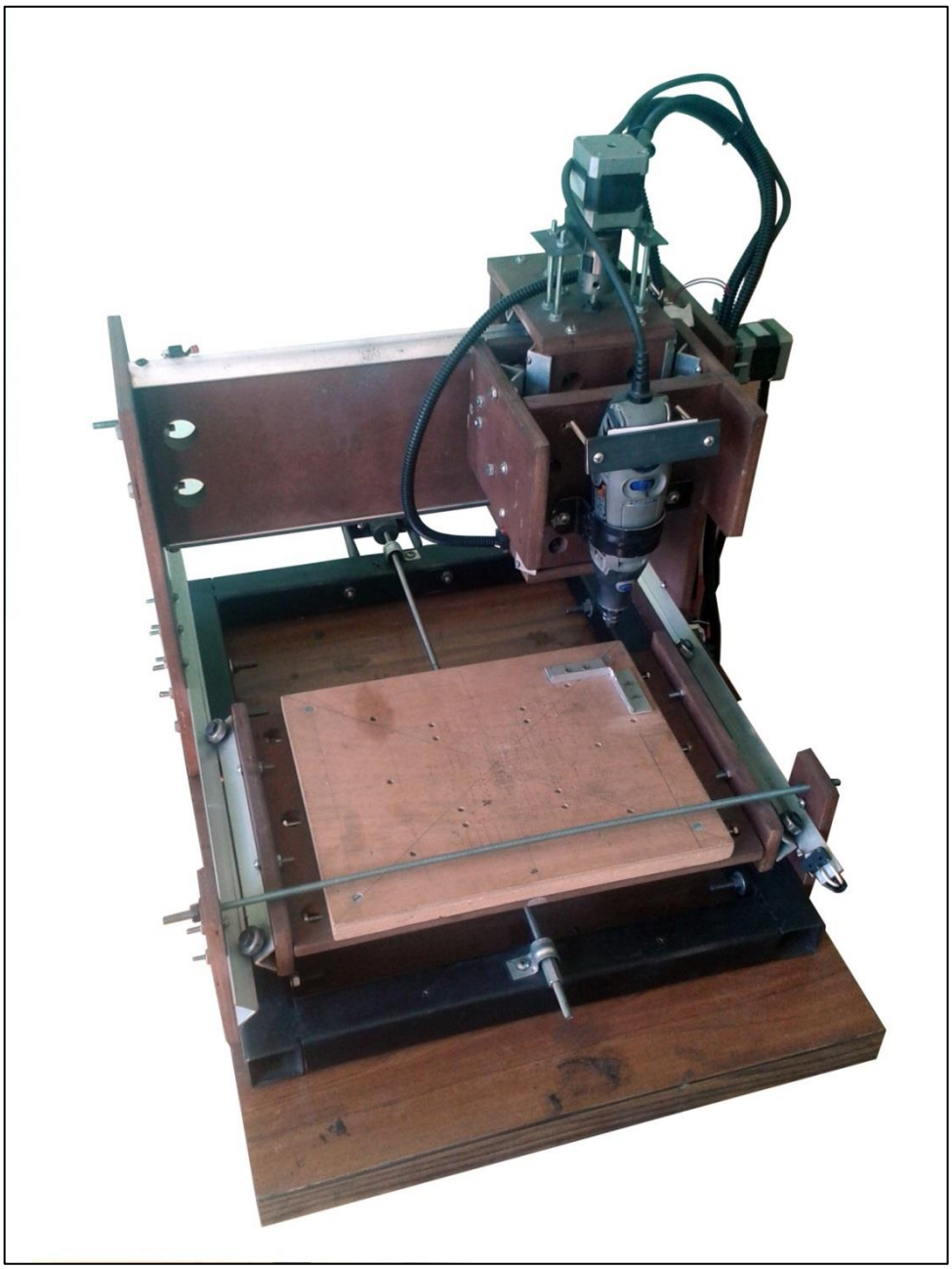
Fuente: domicilio particular.

Anexo 1b. **Máquina fresadora terminada. Vista frontal**



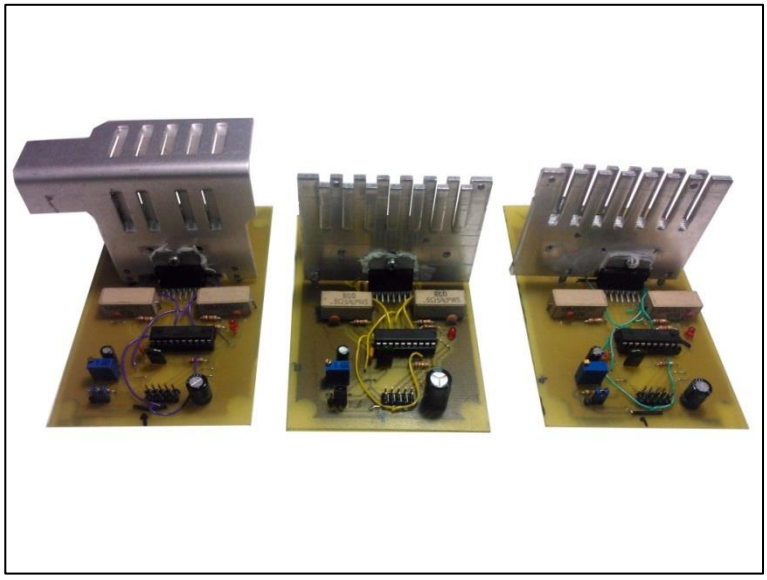
Fuente: domicilio particular.

Anexo 1c. **Máquina fresadora terminada. Vista superior**



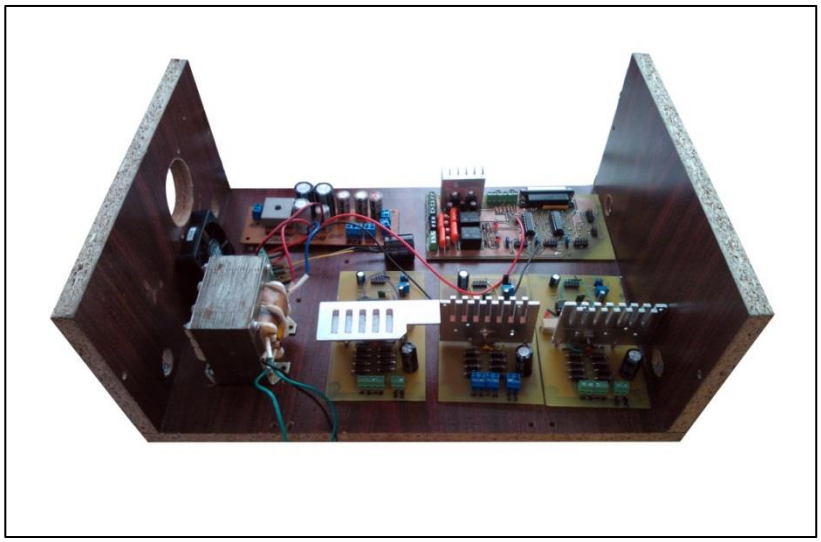
Fuente: domicilio particular.

Anexo 1d. **Controladores de motores paso a paso**



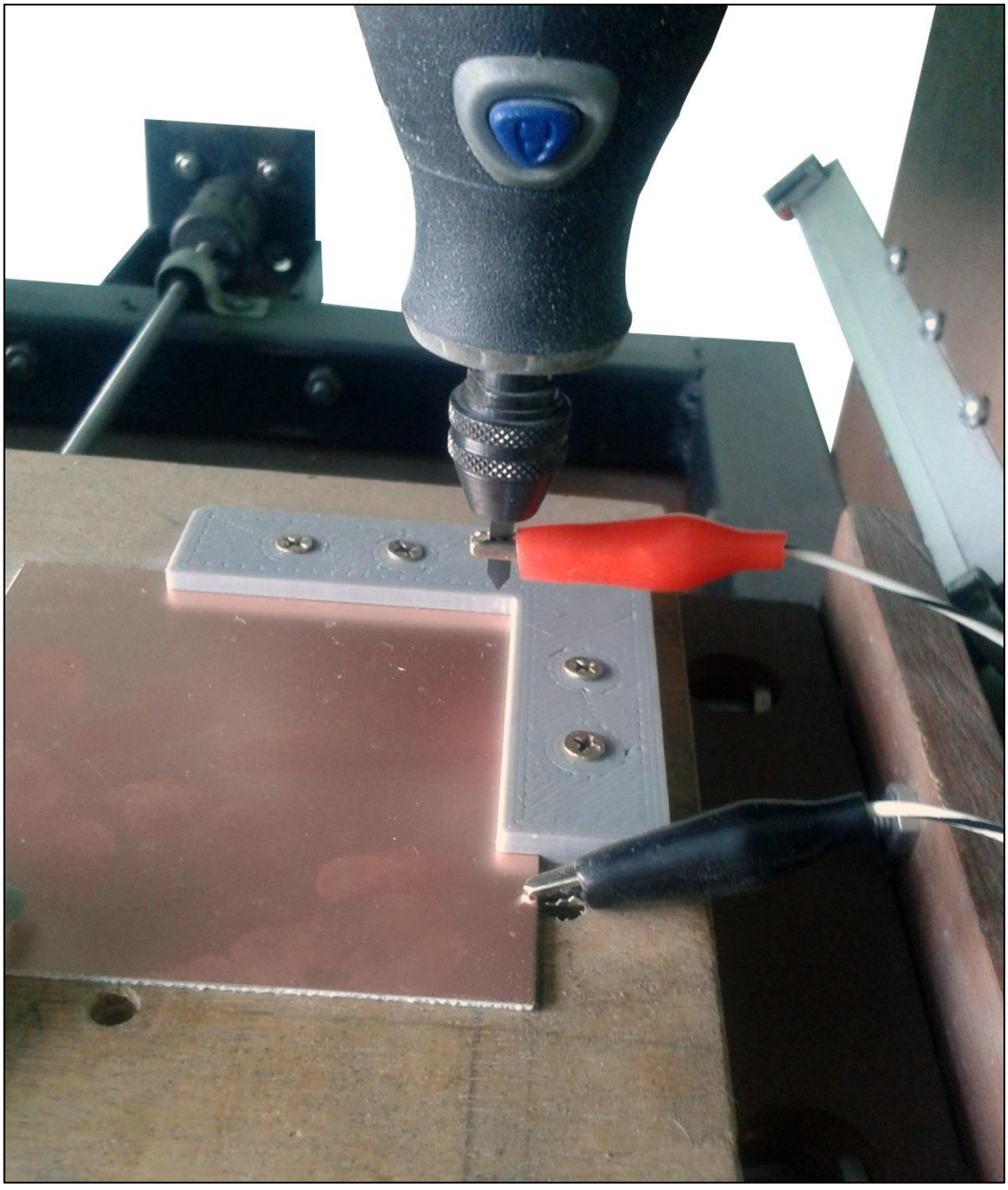
Fuente: domicilio particular.

Anexo 1e. **Caja de control y fuente de alimentación**



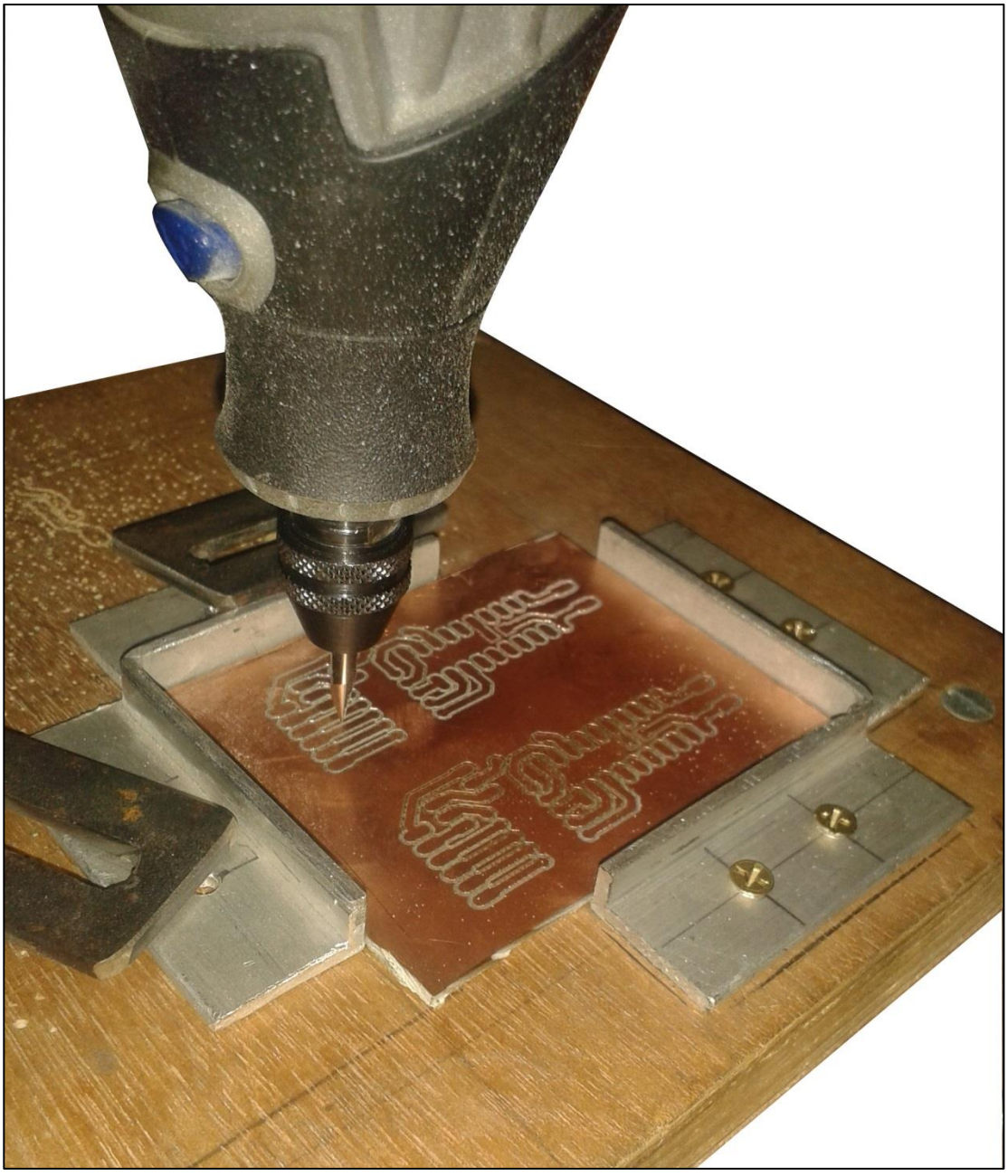
Fuente: domicilio particular.

Anexo 1f. **Máquina fresadora realizando medición y corrección de alturas**



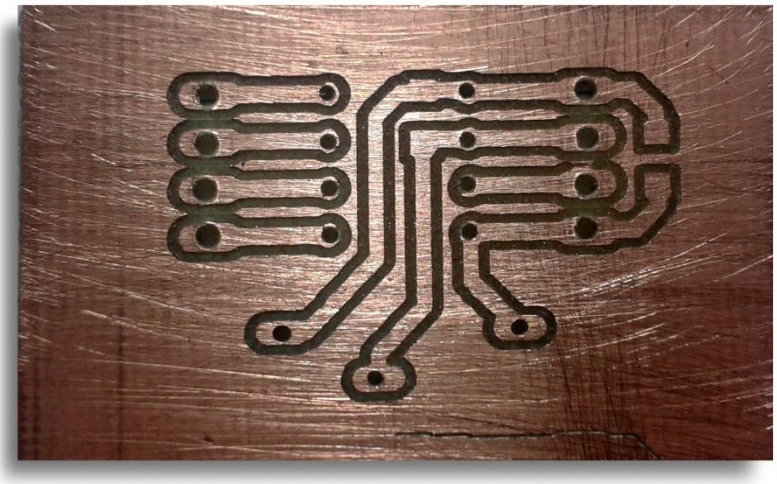
Fuente: domicilio particular.

Anexo 1g. **Máquina fresadora realizando el fresado sobre una placa de cobre virgen**



Fuente: domicilio particular.

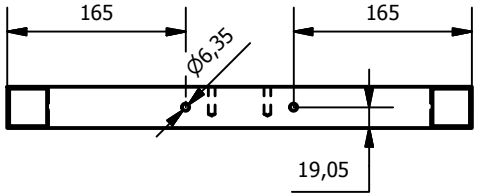
Anexo 1h. **Fresado y barrenado de una placa de circuito impreso**

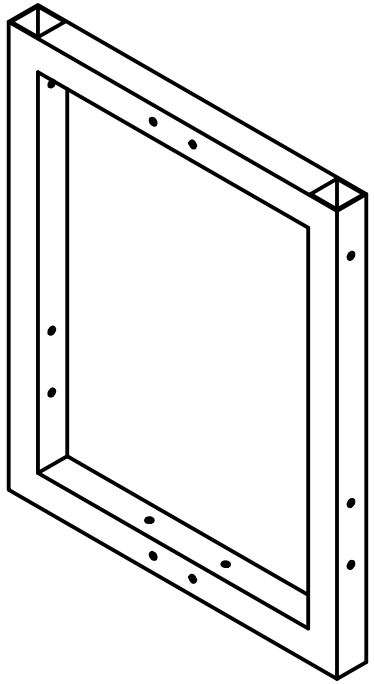


Fuente: domicilio particular.

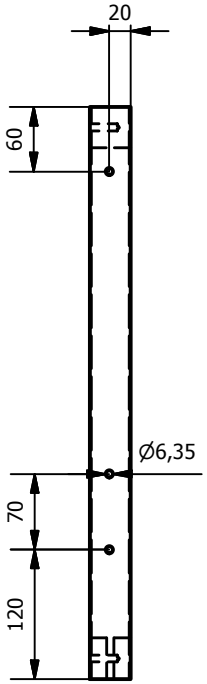
DISEÑOS DE LA MÁQUINA FRESADORA

VISTA FRONTAL
ESCALA (1 : 7)

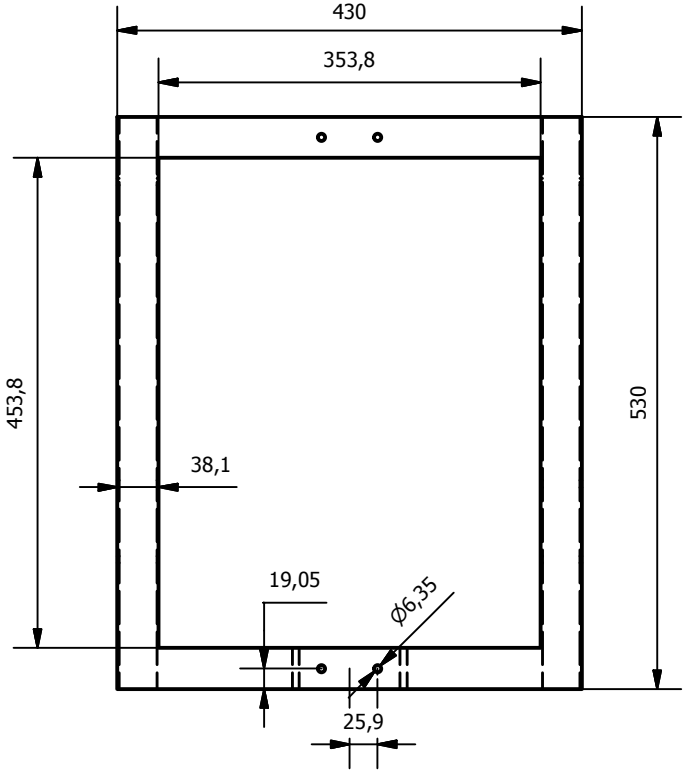




BASE PARA MÁQUINA
ISOMÉTRICO
ESCALA (1 : 7)



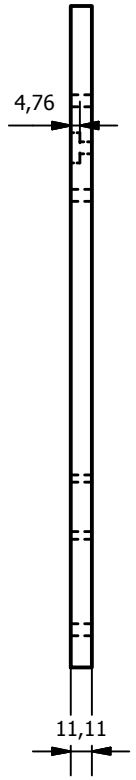
VISTA LATERAL
ESCALA (1 : 7)



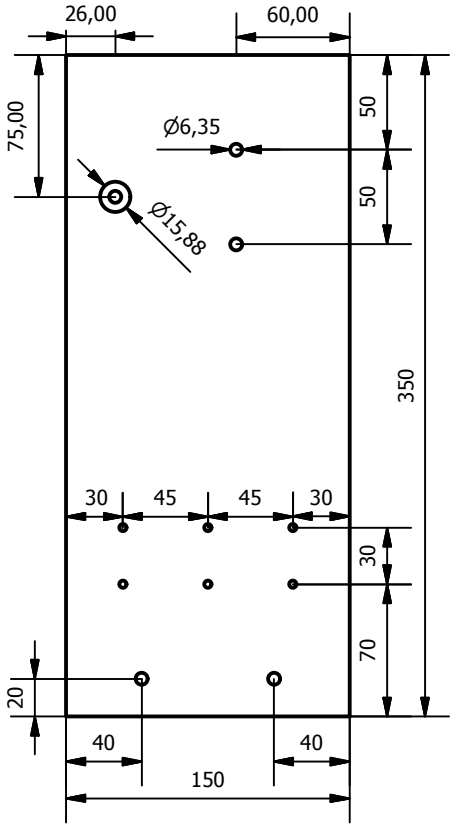
VISTA SUPERIOR
ESCALA (1 : 7)

Nota: Tubo cuadrado de 1 ½ in de acero al carbón.
Nota: Todas las medidas en mm.

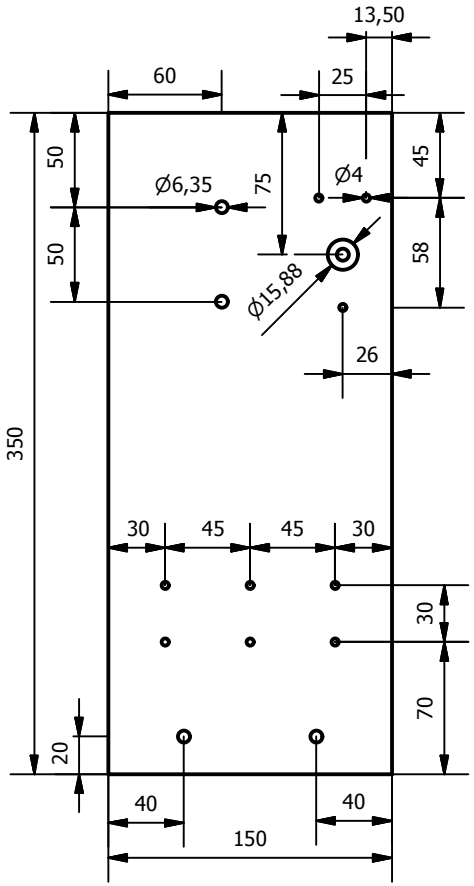
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA	DISEÑO DE MÁQUINA FRESADORA CNC	AUTOR: RODRIGO CHANG
FACULTAD DE INGENIERÍA		ABRIL DE 2015



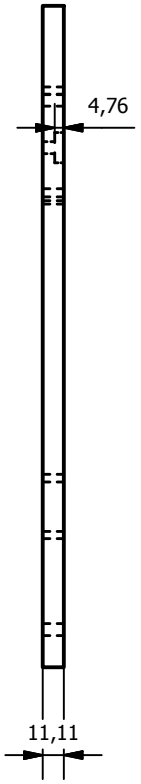
VISTA LATERAL
ESCALA (1 : 4)



BRAZO IZQUIERDO PUENTE
ESCALA (1 : 4)



BRAZO DERECHO PUENTE
ESCALA (1 : 4)

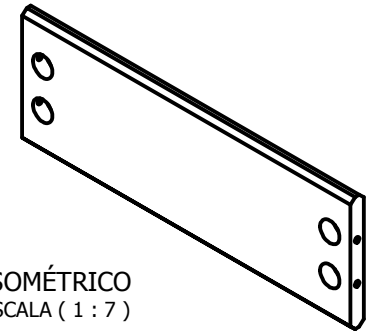


VISTA LATERAL
ESCALA (1 : 4)

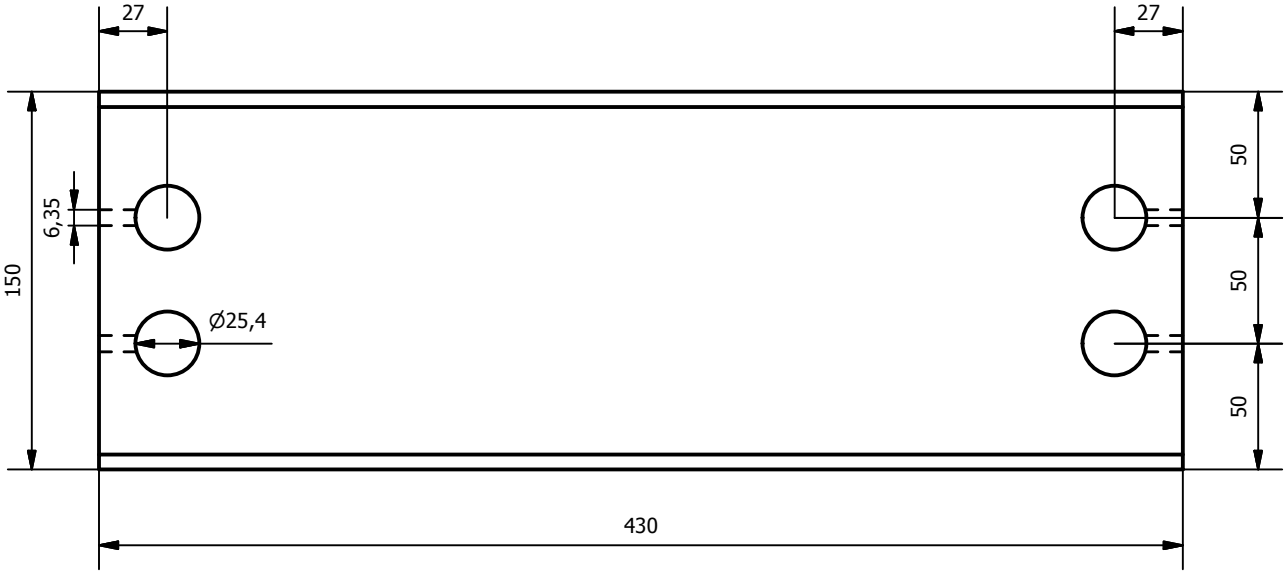
Nota: Todas las medidas en mm.

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA	DISEÑO DE MÁQUINA FRESADORA CNC	AUTOR: RODRIGO CHANG
FACULTAD DE INGENIERÍA		ABRIL DE 2015

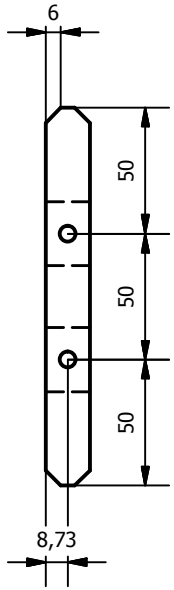
VISTA SUPERIOR
ESCALA (1 : 3)



ISOMÉTRICO
ESCALA (1 : 7)



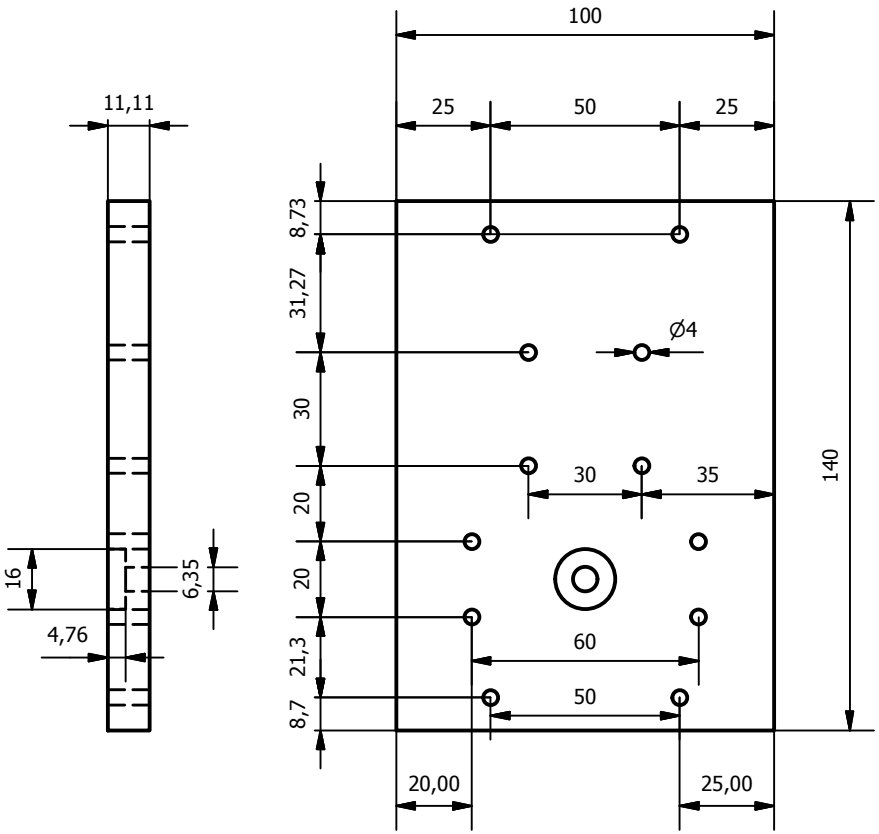
PUENTE PARA EJE Y
ESCALA (1 : 3)



VISTA LATERAL
ESCALA (1 : 3)

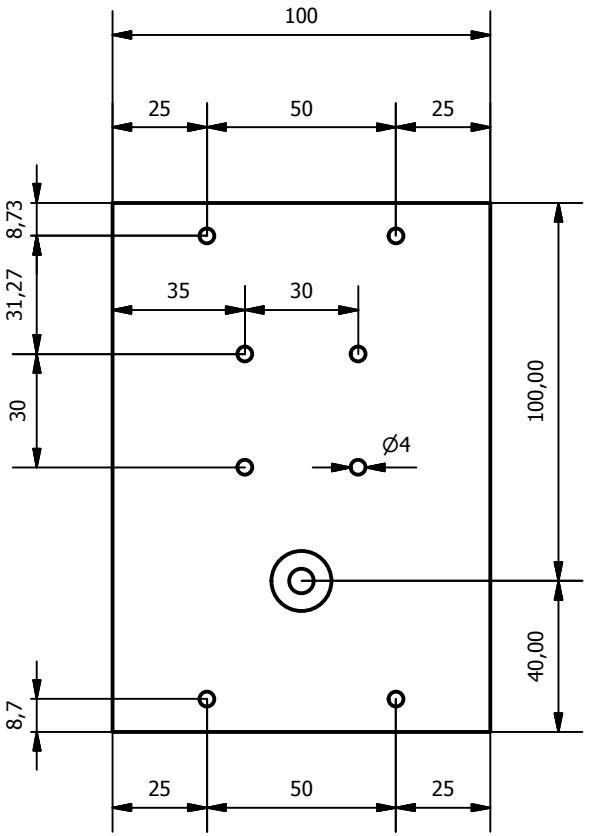
Nota: Todas las medidas en mm.

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA	DISEÑO DE MÁQUINA FRESADORA CNC	AUTOR: RODRIGO CHANG
FACULTAD DE INGENIERÍA		ABRIL DE 2015

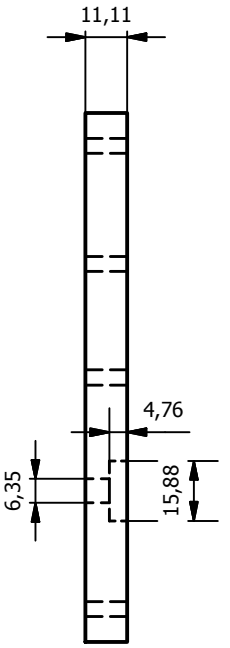


VISTA LATERAL
ESCALA (1 : 2)

CARRO EJE Y
PIEZA SUPERIOR
ESCALA (1 : 2)



CARRO EJE Y
PIEZA INFERIOR
ESCALA (1 : 2)

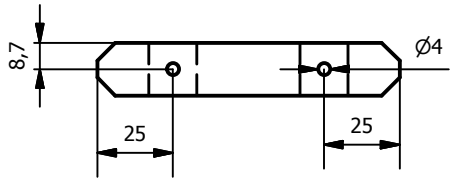


VISTA LATERAL
ESCALA (1 : 2)

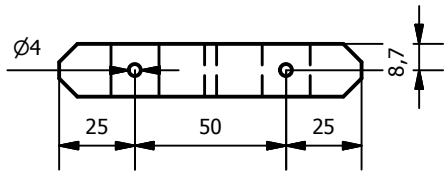
Nota: Todas las medidas en mm.

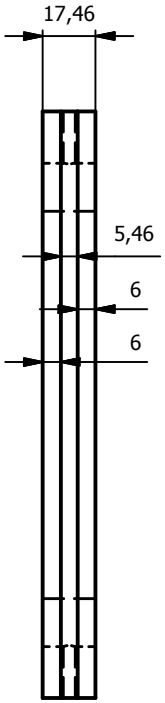
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA	DISEÑO DE MÁQUINA FRESADORA CNC	AUTOR: RODRIGO CHANG
FACULTAD DE INGENIERÍA		ABRIL DE 2015

VISTA SUPERIOR
ESCALA (1 : 2.5)

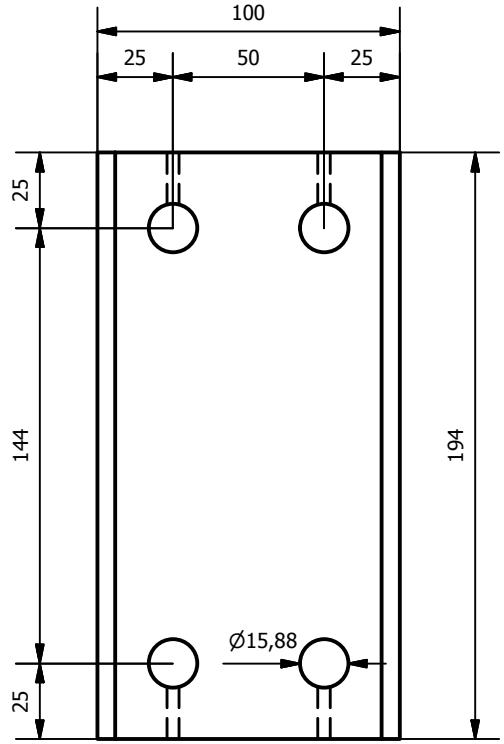


VISTA SUPERIOR
ESCALA (1 : 2.5)

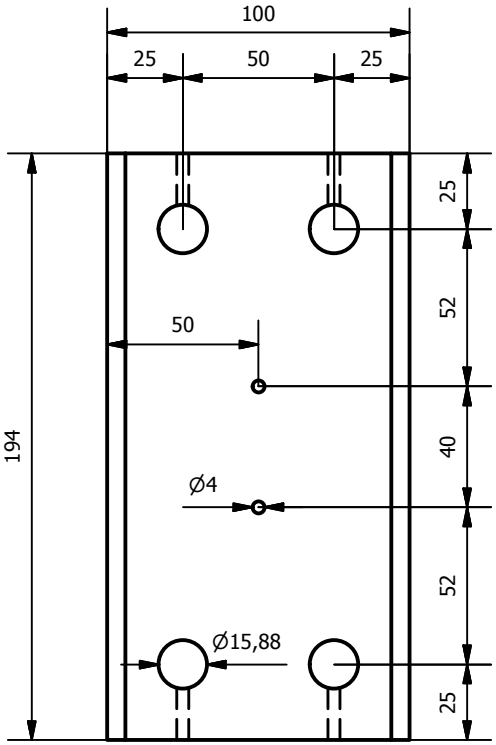




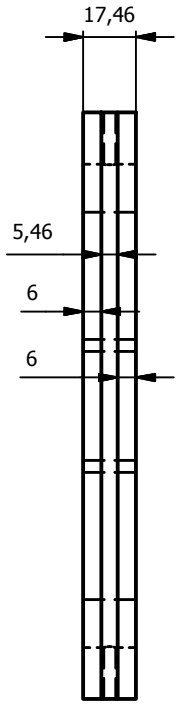
VISTA LATERAL
ESCALA (1 : 2.5)



CARRO EJE Y
PIEZA FRONTAL
ESCALA (1 : 2.5)



CARRO EJE Y
PIEZA TRASERA
ESCALA (1 : 2.5)

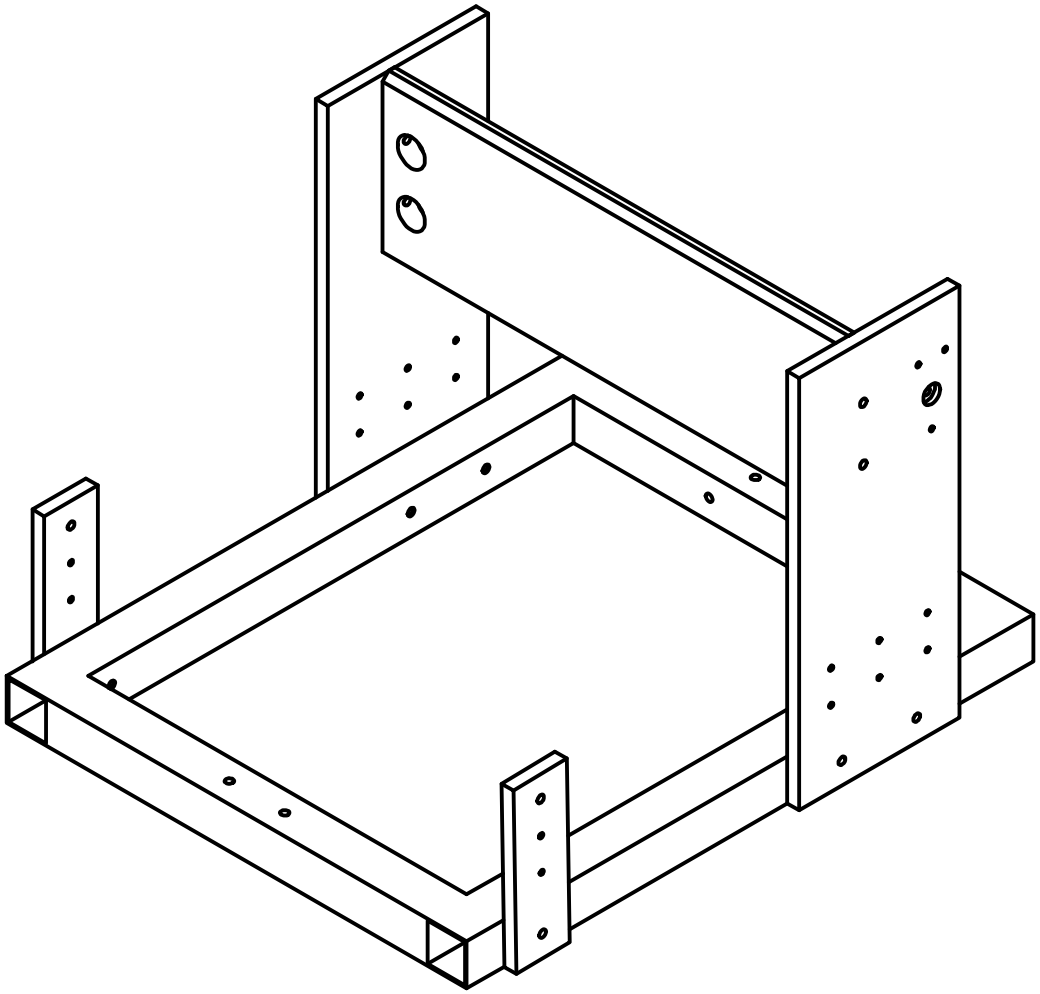


VISTA LATERAL
ESCALA (1 : 2.5)

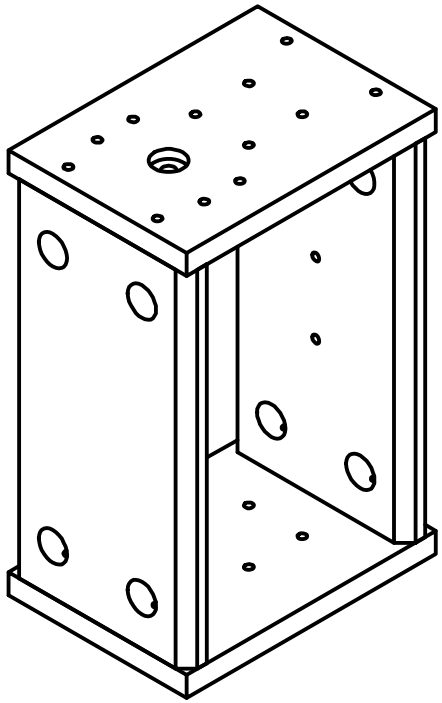
Nota: Todas las medidas en mm.

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA	DISEÑO DE MÁQUINA FRESADORA CNC	AUTOR: RODRIGO CHANG
FACULTAD DE INGENIERÍA		ABRIL DE 2015

ENSAMBLE DE MÁQUINA FRESADORA
BRAZOS Y PUENTE
ESCALA (0.20 : 1)



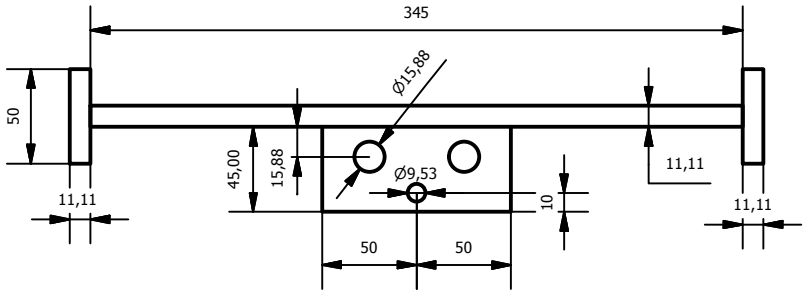
ENSAMBLE CARRO EJE Y
ISOMÉTRICO SUPERIOR
ESCALA (1 : 3)

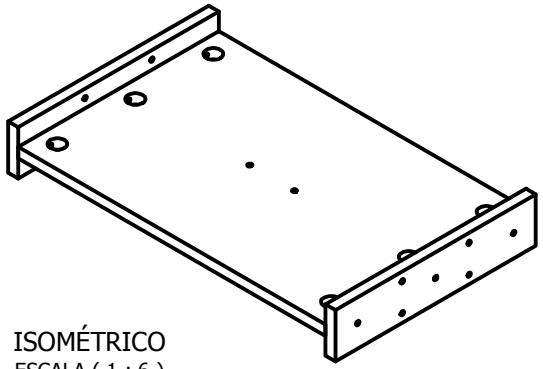


Nota: Todas las medidas en mm.

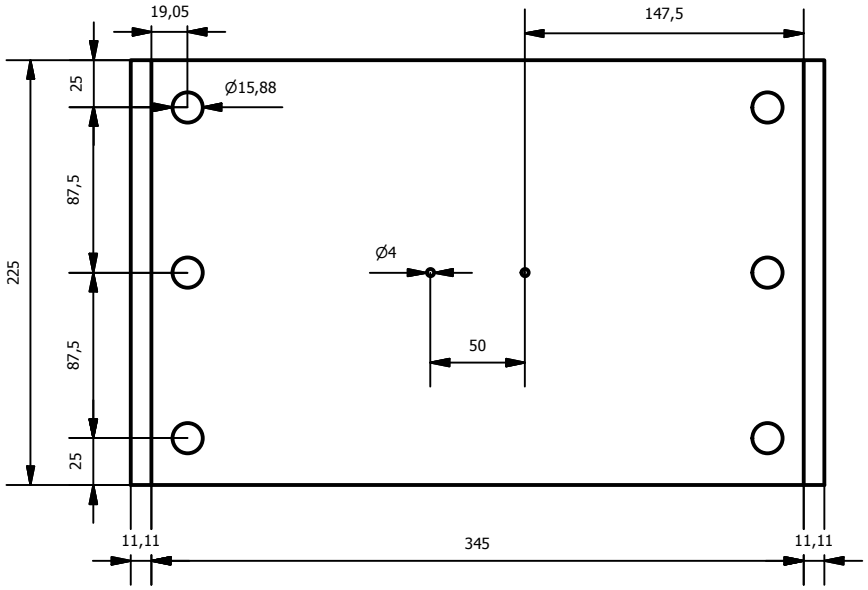
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA	DISEÑO DE MÁQUINA FRESADORA CNC	AUTOR: RODRIGO CHANG
FACULTAD DE INGENIERÍA		ABRIL DE 2015

VISTA FRONTAL
ESCALA (1 : 4)

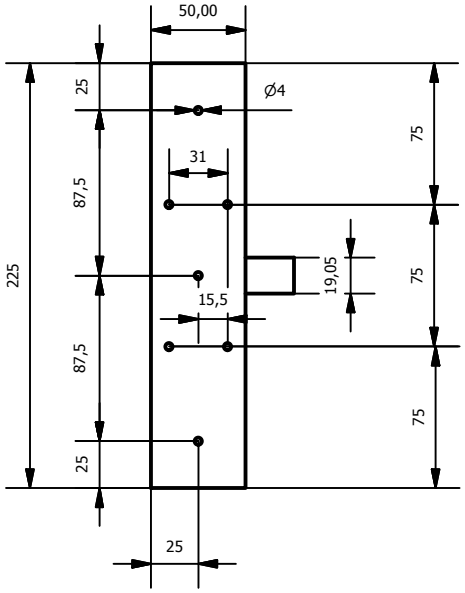




ISOMÉTRICO
ESCALA (1 : 6)



CARRO EJE X, VISTA SUPERIOR
ESCALA (1 : 4)

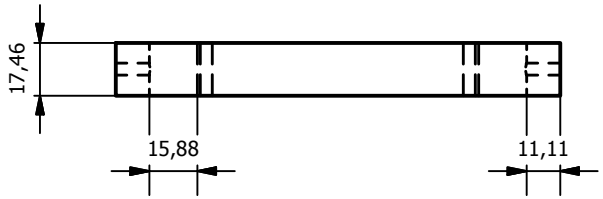


VISTA LATERAL
ESCALA (1 : 4)

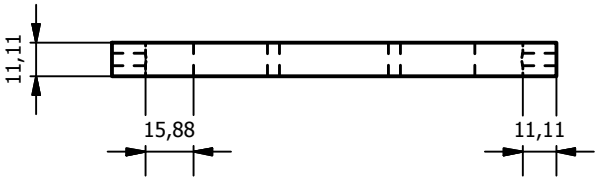
Nota: Todas las medidas en mm.

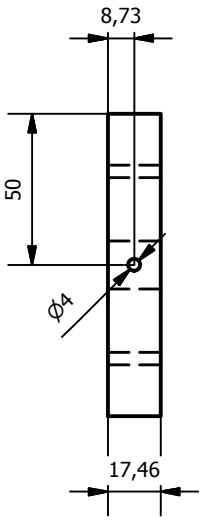
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA	DISEÑO DE MÁQUINA FRESADORA CNC	AUTOR: RODRIGO CHANG
FACULTAD DE INGENIERÍA		ABRIL DE 2015

VISTA SUPERIOR
ESCALA (1 : 2.5)

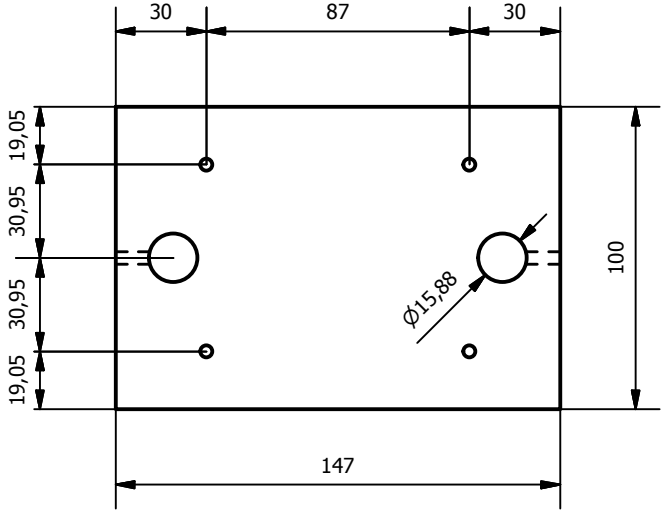


VISTA SUPERIOR
ESCALA (1 : 2.5)

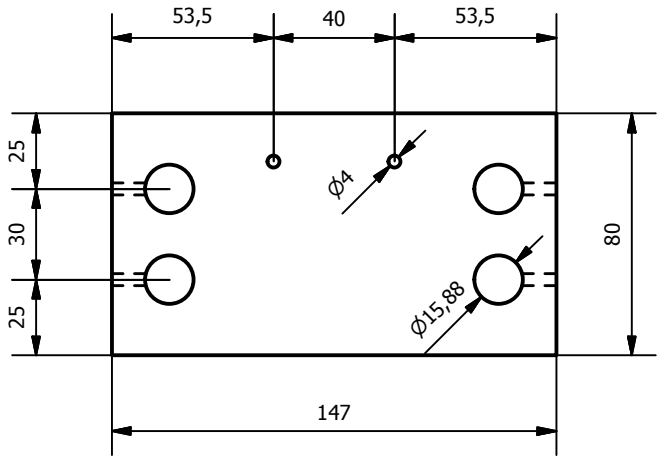




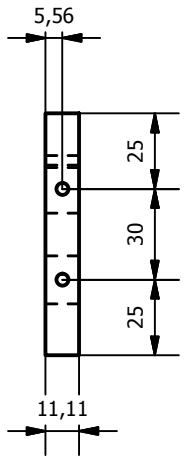
VISTA LATERAL
ESCALA (1 : 2.5)



CARRO EJE Z
PIEZA FRONTAL
ESCALA (1 : 2.5)



CARRO EJE Z
PIEZA TRASERA
ESCALA (1 : 2.5)

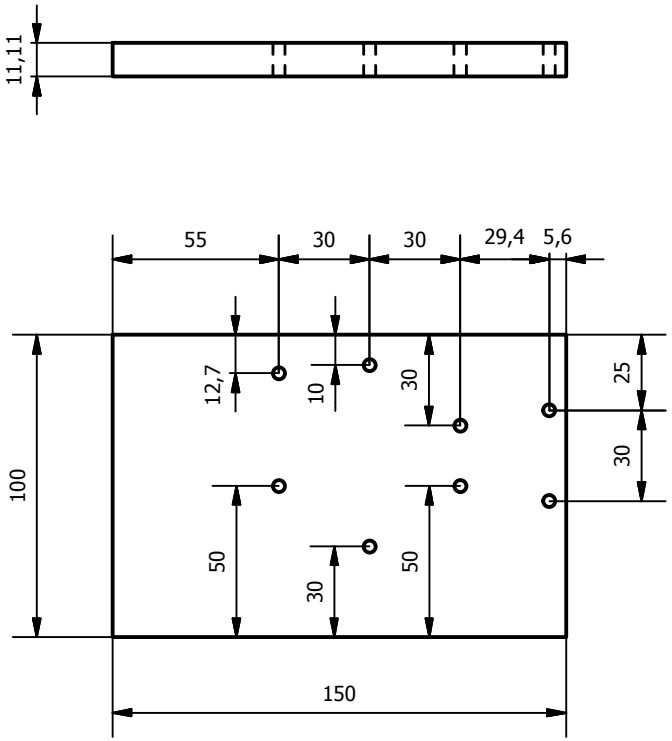


VISTA LATERAL
ESCALA (1 : 2.5)

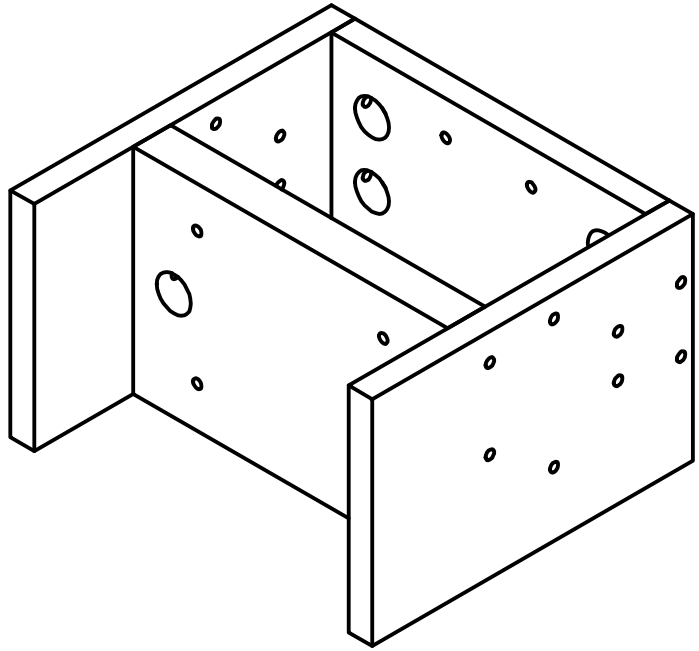
Nota: Todas las medidas en mm.

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA	DISEÑO DE MÁQUINA FRESADORA CNC	AUTOR: RODRIGO CHANG
FACULTAD DE INGENIERÍA		ABRIL DE 2015

VISTA SUPERIOR
ESCALA (1 : 2.5)



CARRO EJE Z
PIEZA LATERAL
ESCALA (1 : 2.5)

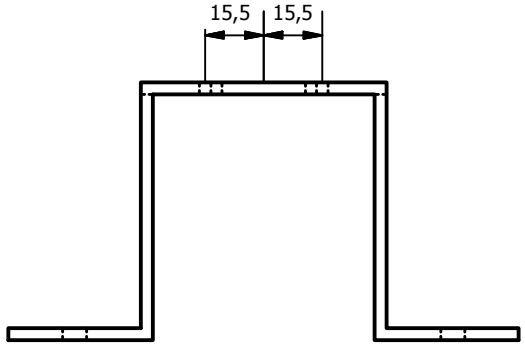


ENSAMBLE CARRO EJE Z
ISOMÉTRICO
ESCALA (1 : 2.5)

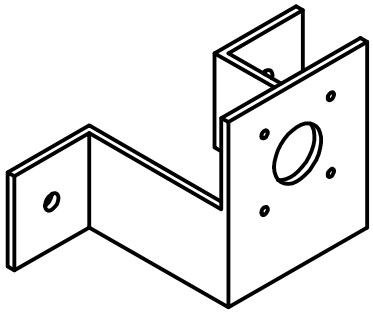
Nota: Todas las medidas en mm.

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA	DISEÑO DE MÁQUINA FRESADORA CNC	AUTOR: RODRIGO CHANG
FACULTAD DE INGENIERÍA		ABRIL DE 2015

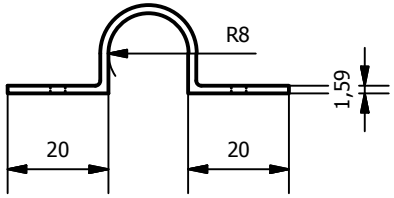
VISTA SUPERIOR
ESCALA (1 : 2)

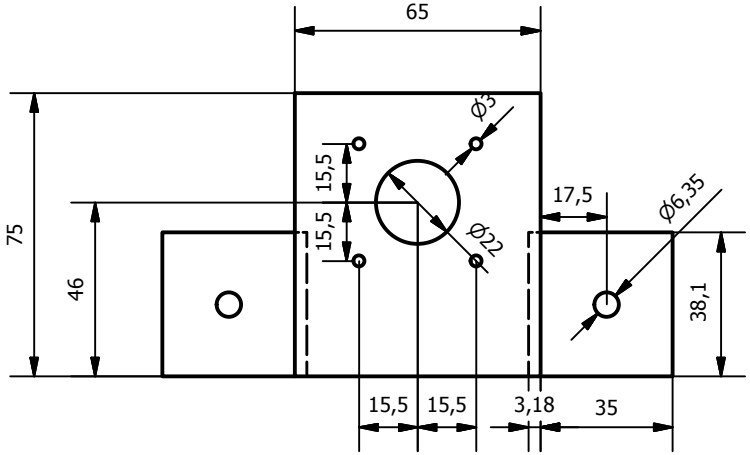


ISOMÉTRICO
ESCALA (1 : 2.5)

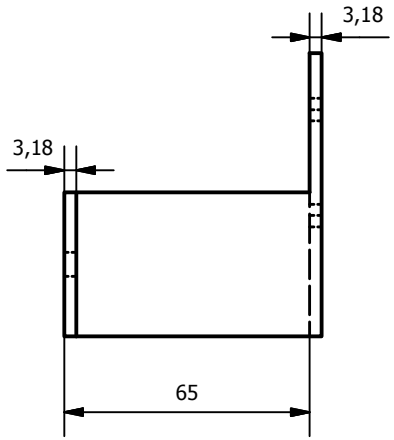


VISTA FRONTAL
ESCALA (1 : 1.5)

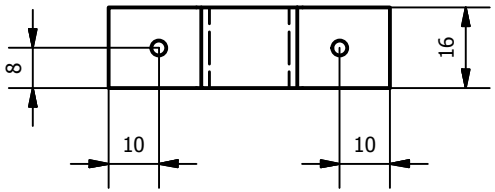




SUJETADOR DE MOTOR EJE X
ESCALA(1 : 2)



VISTA LATERAL
ESCALA (1 : 2)

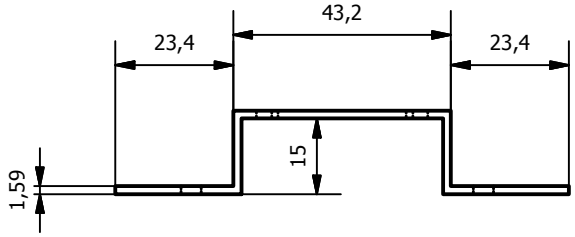


SUJETADOR DE COJINETE EJE X
VISTA SUPERIOR
ESCALA (1 : 1.5)

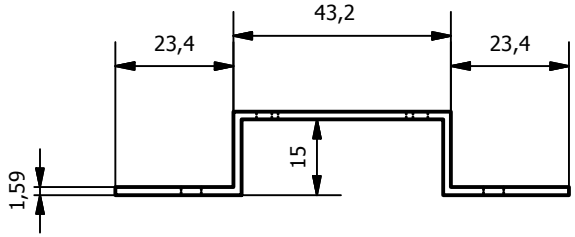
Nota: Todas las medidas en mm.

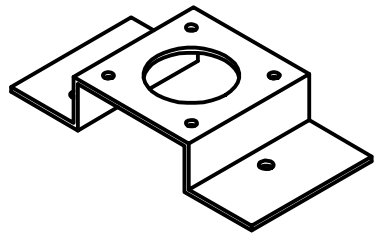
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA	DISEÑO DE MÁQUINA FRESADORA CNC	AUTOR: RODRIGO CHANG
FACULTAD DE INGENIERÍA		ABRIL DE 2015

SUJETADOR DE MOTOR EJE Y
VISTA FRONTAL
ESCALA (1 : 1.5)



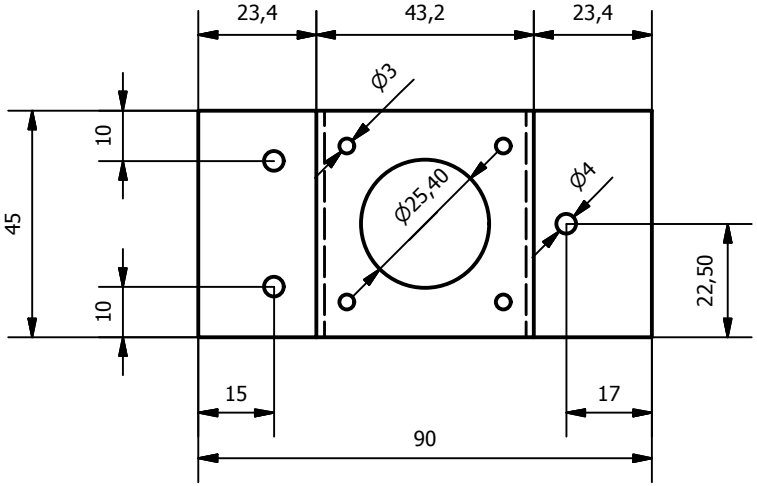
SUJETADOR DE MOTOR EJE Z
VISTA FRONTAL
ESCALA (1 : 1.5)



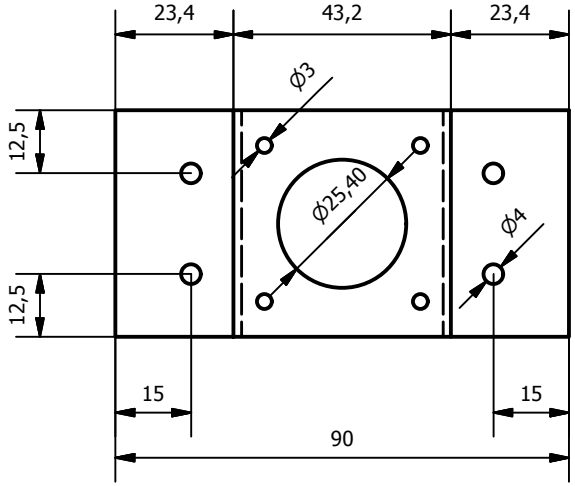


ISOMÉTRICO
ESCALA (1 : 2)

VISTA SUPERIOR
ESCALA (1 : 1.5)



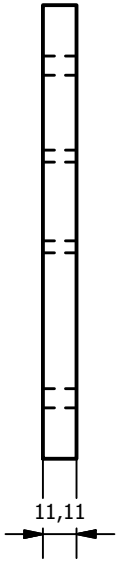
VISTA SUPERIOR
ESCALA (1 : 1.5)



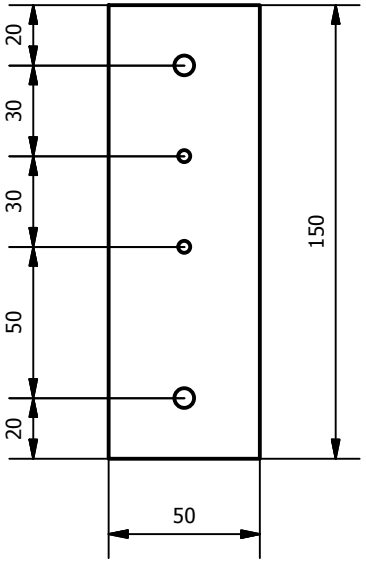
Nota: Todas las medidas en mm.

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA	DISEÑO DE MÁQUINA FRESADORA CNC	AUTOR: RODRIGO CHANG
FACULTAD DE INGENIERÍA		ABRIL DE 2015

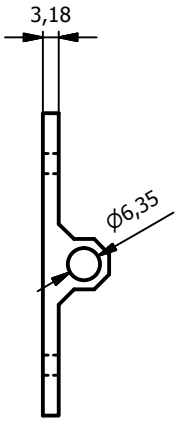
VISTA LATERAL
ESCALA (1 : 2.5)

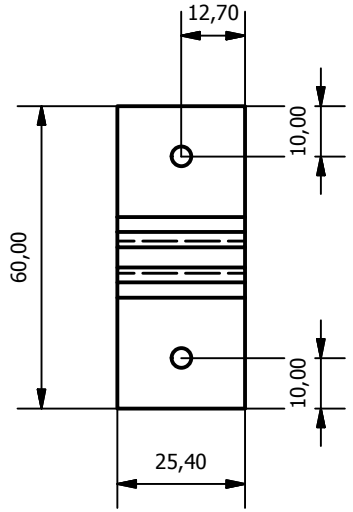


SUJETADOR DE PERFIL ANGULAR
CARRO EJE X
VISTA FRONTAL
ESCALA (1 : 2.5)



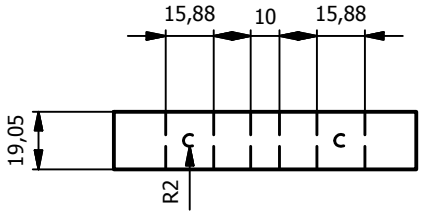
VISTA LATERAL
ESCALA (1:1.5)

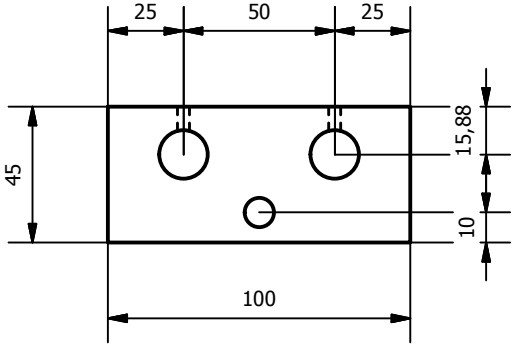




SUJETADOR DE TUERCA DE TRANSMISIÓN
CARROS EJES Y, Z
ESCALA (1:1.5)

VISTA SUPERIOR
ESCALA (1 : 2.5)



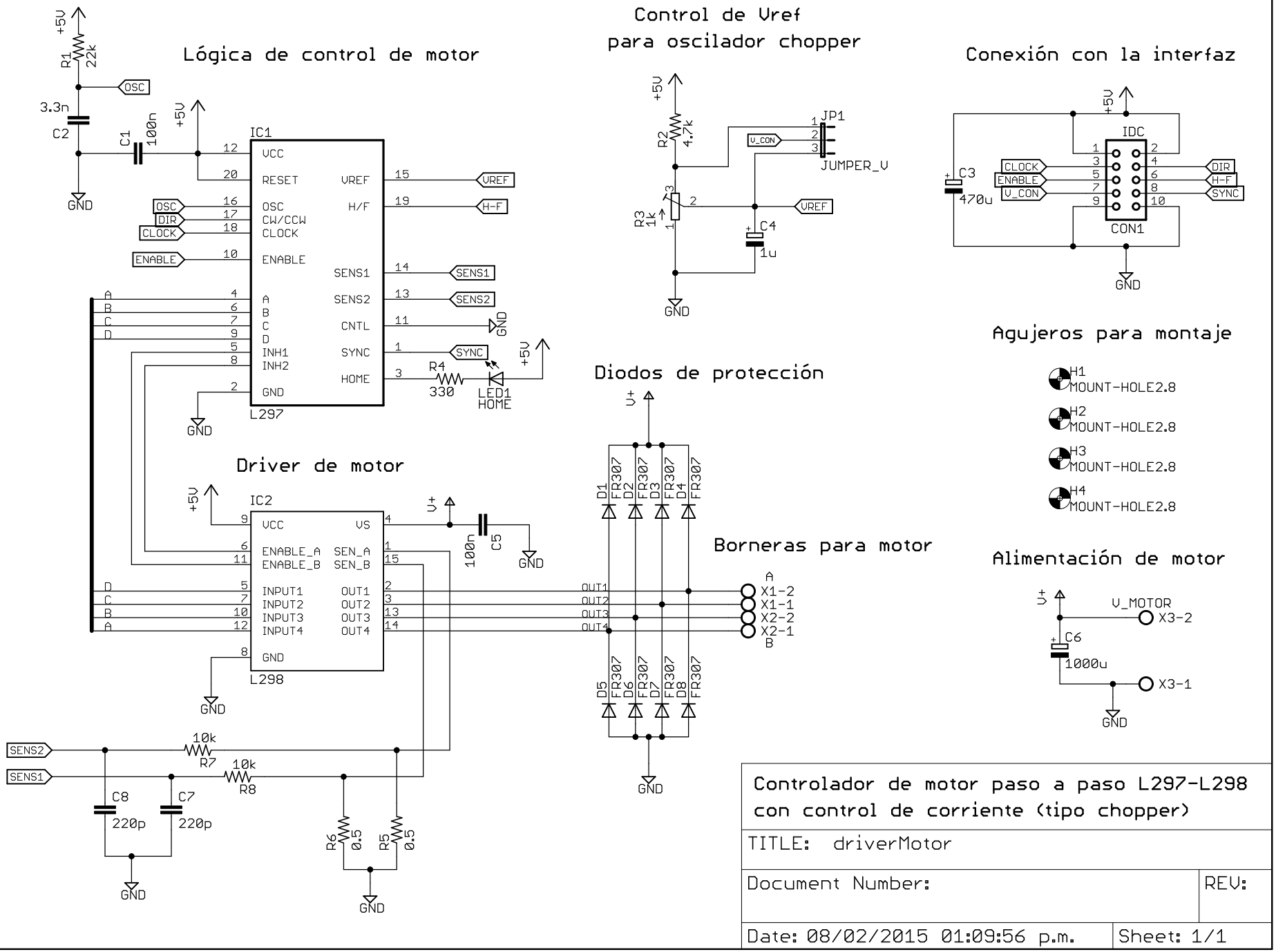


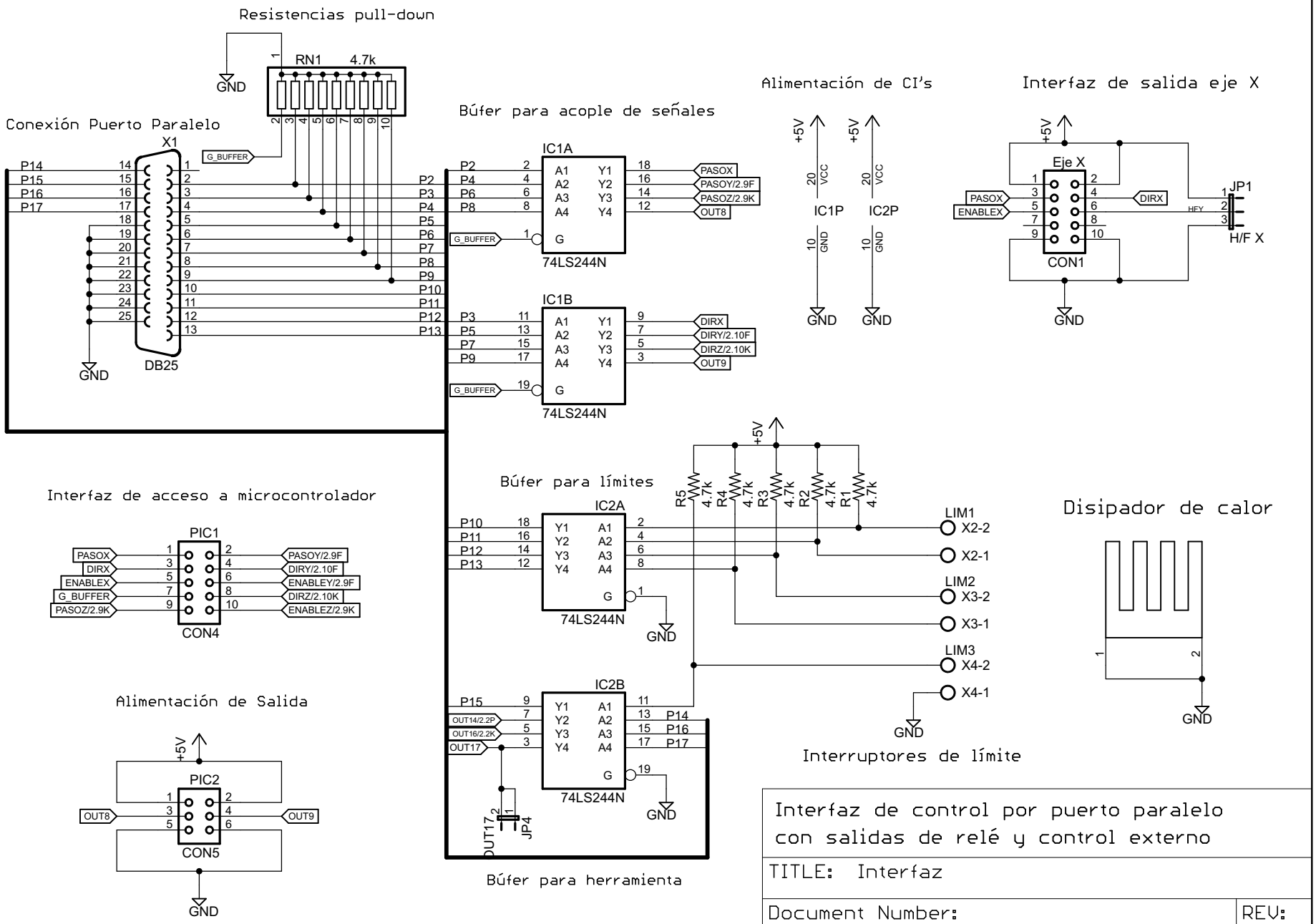
SUJETADOR DE TUERCA DE TRANSMISIÓN
CARRO EJE X
ESCALA (1 : 2.5)

Nota: Todas las medidas en mm.

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA	DISEÑO DE MÁQUINA FRESADORA CNC	AUTOR: RODRIGO CHANG
FACULTAD DE INGENIERÍA		ABRIL DE 2015

DIAGRAMAS DE CIRCUITOS ELECTRÓNICOS


Lógica de control de motor
Control de Vref
para oscilador chopper
Conexión con la interfaz
Agujeros para montaje
Diodos de protección
Driver de motor
Borneras para motor
Alimentación de motor
IC1
L297
IC2
L298
IDC
CON1
JP1
JUMPER_V
H1 MOUNT-HOLE2.8
H2 MOUNT-HOLE2.8
H3 MOUNT-HOLE2.8
H4 MOUNT-HOLE2.8
Controlador de motor paso a paso L297-L298
con control de corriente (tipo chopper)
TITLE: driverMotor
Document Number:
REV:
Date: 08/02/2015 01:09:56 p.m.
Sheet: 1/1



Interfaz de control por puerto paralelo con salidas de relé y control externo

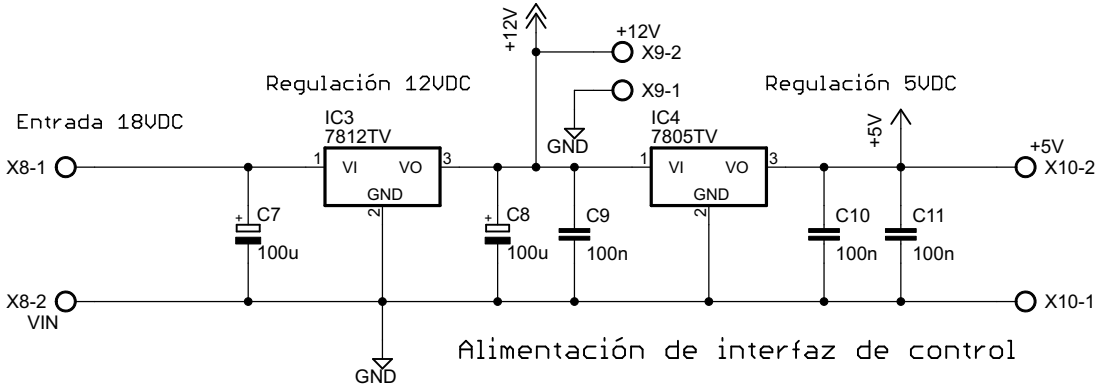
TITLE: Interfaz

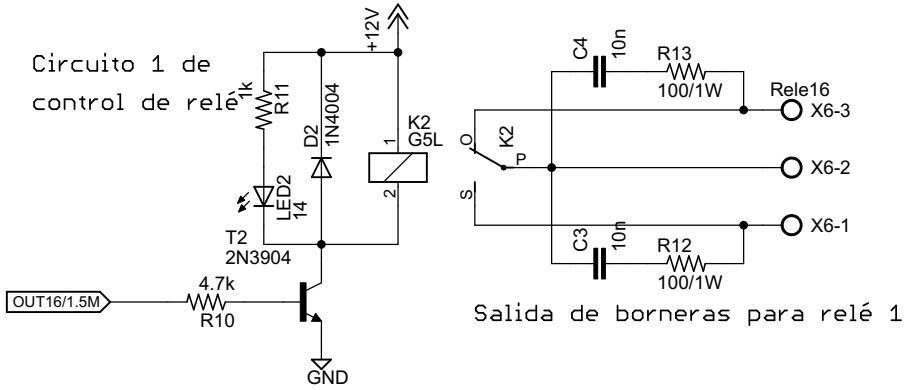
Document Number:

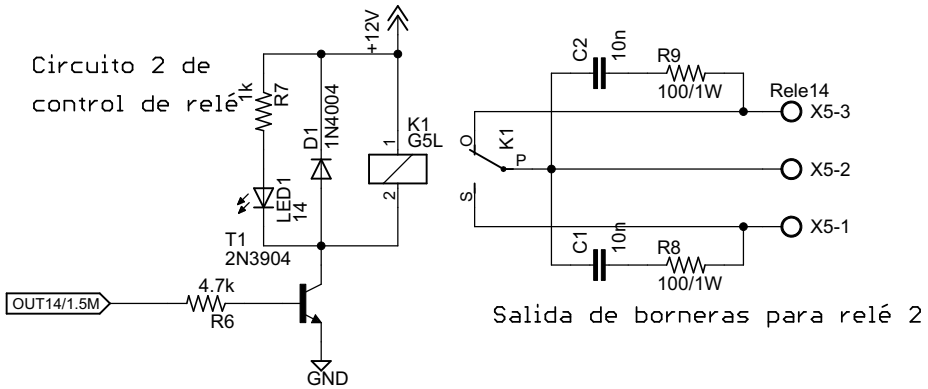
REV:

Date: 30/03/2015 01:05:12 p.m.

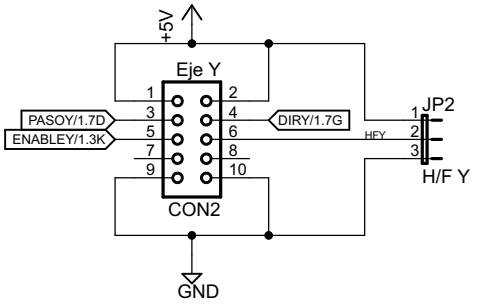
Sheet: 1/2



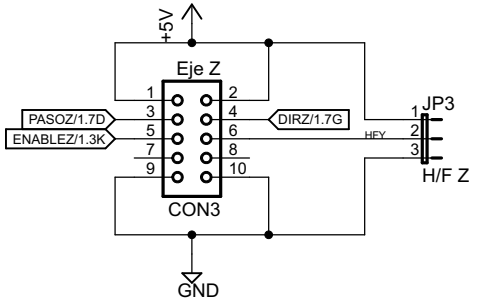




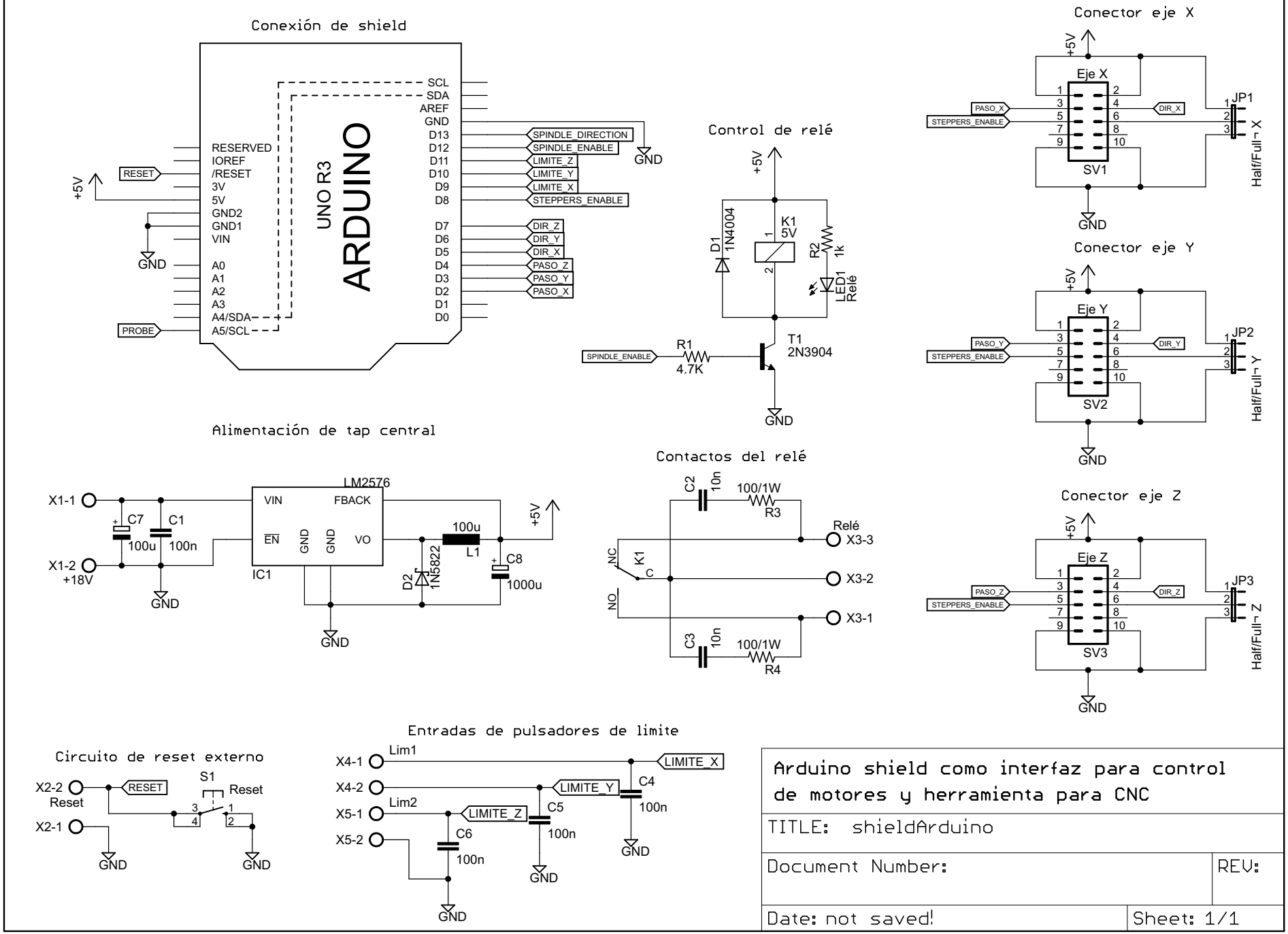
Interfaz de salida eje Y



Interfaz de salida eje Z



Interfaz de control por puerto paralelo con salidas de relé y control externo	
TITLE: Interfaz	
Document Number:	REV:
Date: 30/03/2015 01:05:12 p.m.	Sheet: 2/2



Arduino shield como interfaz para control de motores y herramienta para CNC

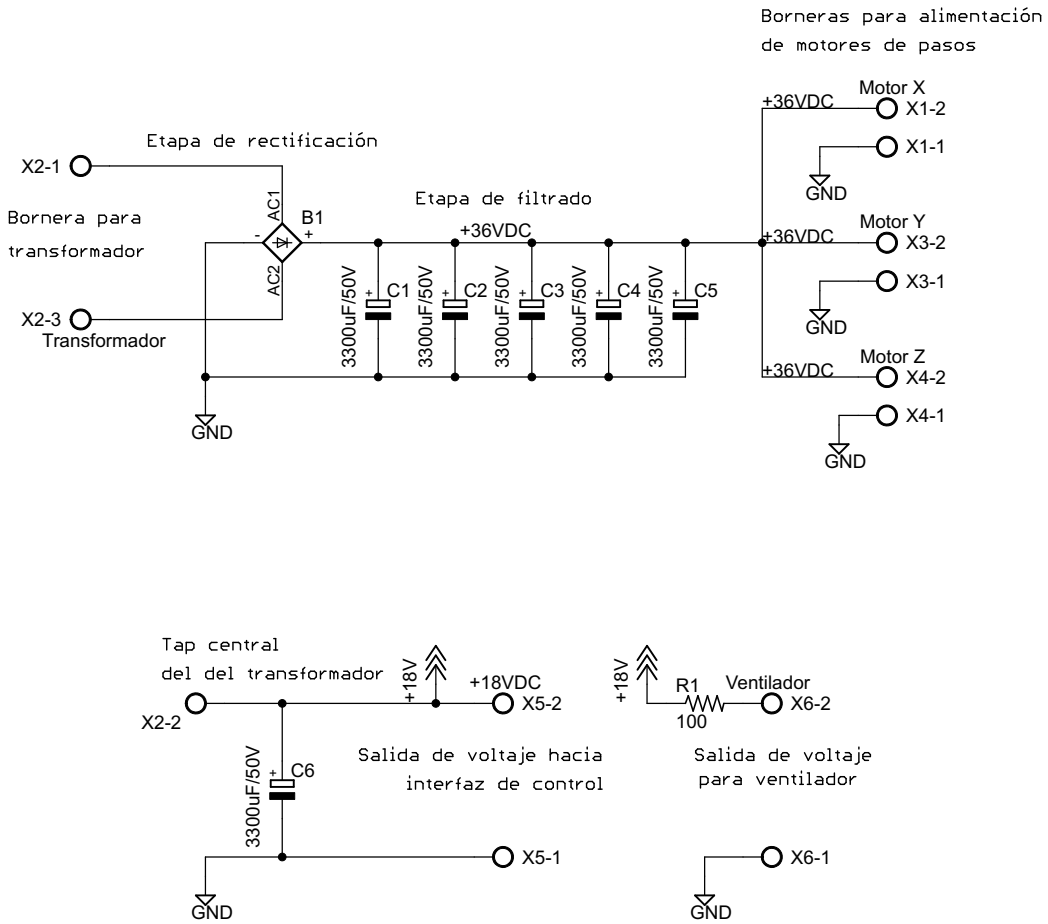
TITLE: shieldArduino

Document Number:

REV:

Date: not saved!

Sheet: 1/1



H1
MOUNT-HOLE4.1

H2
MOUNT-HOLE4.1

H3
MOUNT-HOLE4.1

H4
MOUNT-HOLE4.1

Fuente de alimentación de 36VDC para control de máquina fresadora CNC	
TITLE: fuenteCNC	
Document Number:	REV:
Date: 01/04/2015 12:29:59 a.m.	Sheet: 1/1

PROGRAMAS Y ARCHIVOS DE CONFIGURACIÓN

Programa de optimización de trayectorias de código G

En el código 1 se muestra el código del programa desarrollado en Python para optimizar trayectorias en el código G de fresado y barrenado.

Código 1: **Programa de optimización en Python**

```
#-= Rodrigo Chang =-'
#-= Noviembre 2014 =-'

import re
import math
import sys

HEADER_PATTERN = '^([^~,]+?G0?0 Z[0-9.\s]+)'
FOOTER_PATTERN = '(G0?0 Z[0-9.\sM]+)$'
INTERESTING_PATTERN = 'G0?0 X[-0-9.]+ Y[-0-9.]+\nG0?1 Z-[0-9.]+[^~,]+?G0?0 Z[^-].+?\n'

# Seleccionar si utiliza el header por defecto, optimo para fresado y perforado de placas
# con ULP pcb-gcode de EAGLE
CUSTOM_HEADER='(Optimizado en Python)\n(Metrico)\nG21\n(Absolute coordinates)\nG90\nG0 Z3.0000\nM3\nG4 P4.0000\n'
USE_CUSTOM_HEADER = True

'''
Clase nodo:
  Mantiene las coordenadas de inicio y de final,
  y la correspondiente cadena de codigo G
'''
class mynode(object):
    def __init__(self, nodegcode):
        self.nodegcode = nodegcode
        result = re.findall("G0?0 X(.*?) Y(.*?)[ ]*?\n",nodegcode)
        self.start=result[0]
        # Intenta encontrar coordenadas de final con el patron
        # para los archivos de agujeros no encuentra el patron
        # asi que el punto final es el mismo que el inicial
        try:
            result = re.findall("G0?1 X(.*?) Y(.*?)[ ]?\nG0?0 Z[^-]",nodegcode)
            self.finish=result[0]
        except IndexError:
            self.finish=self.start

    def disp(self):
        print self.nodegcode

    def __str__(self):
        return 'Inicio: ' + str(self.start) + ' --- Fin: ' + str(self.finish)

# Funcion para calcular la distancia euclidiana entre dos puntos(X1,Y1),(X2, Y2)
def edist(point1,point2):
    distance = math.sqrt( math.pow(float(point1[1])-float(point2[1]),2) + math.pow(float(point1[0])-float(point2[0]),2) )
    return distance

# Devuelve el nombre del archivo con '.OPT' antes de la ultima extension
def opt_name(s, prefix):
    assert '.' in s
    i = -1
```

```
    while s[i] != '.':
        i = i - 1
    old_ext = s[i:]
    return s[-len(s): i] + prefix + old_ext

'''
    Programa principal
'''
print ''
print '-= pcb-gcode Optimizer =-'
print '-=     R. Chang        =-'

# Revisar si el archivo fuente se da en la linea de comandos.
if (len(sys.argv) < 2):
  print 'Indique el archivo original... (por ejemplo "python cncopt.py myfile.tap")'
  sys.exit("Archivo no definido. Saliendo...")

filename = sys.argv[1]
gcode = open(filename, 'r').read();
gcode = re.sub("[ ]{2,}"," ",gcode);
#Limpiar el codigo G de uno o mas espacios
gcode = re.sub(" \n", "\n", gcode)

# Obtener encabezado y pie del codigo G de interes
result = re.findall(HEADER_PATTERN, gcode, re.IGNORECASE)
header = result[0]
result = re.findall(FOOTER_PATTERN, gcode, re.IGNORECASE)
footer = result[0]

# Encontrar todas las trayectorias
m = re.findall(INTERESTING_PATTERN, gcode, re.IGNORECASE)
nodes=[]
proc_nodes=[]
print "Se encontraron "+str(len(m))+" trayectorias..."

# Crear un nodo para cada trayectoria
for l in m:
    nodes.append(mynode(l))

# Evaluar la distancia total antes de optimizar
total_cost=0;
total_cost=total_cost + edist(('0','0'),nodes[0].start)
for l in range(0, len(nodes)-1):
    total_cost = total_cost + edist(nodes[l].finish,nodes[l+1].start)
print "Distancia inicial :"+str(total_cost)

# Algoritmo de optimizacion principal
# Encontrar el primer punto mas cercano al origen (0,0)
next_node_idx = 0
min_dist=1000
for l in nodes:
    dist = edist(('0','0'),l.start)
    if dist<min_dist:
        min_dist = dist
        next_node_idx = nodes.index(l)

# Procesar el resto de nodos
while(nodes):
    next_node = nodes.pop(next_node_idx)
    proc_nodes.append(next_node)
    min_dist=1000
    for l in nodes:
        dist = edist(next_node.finish,l.start)
```

```
114         if dist<min_dist:
115             min_dist = dist
116             next_node_idx = nodes.index(l)
117
118 # Evaluar la distancia total despues de optimizar
119 total_cost=0;
120 total_cost=total_cost + edist(('0','0'),proc_nodes[0].start)
121 for l in range(0, len(proc_nodes)-1):
122     total_cost = total_cost + edist(proc_nodes[l].finish,proc_nodes[l+1].start)
123 print "Distancia optimizada :" + str(total_cost)
124
125 # Escribir el nuevo archivo optimizado
126 filename = opt_name(filename, '.OPT')
127 print 'El archivo optimizado es ' + filename
128 print ''
129 wf = open(filename,'w')
130 if (USE_CUSTOM_HEADER):
131     wf.write(CUSTOM_HEADER)
132 else:
133     wf.write(header)
134 for l in proc_nodes:
135     wf.write(l.nodegcode)
136 wf.write(footer)
137 wf.close()
```

Programa de medición y corrección de alturas

En el código 2 se muestra el código del programa desarrollado en Python para medir alturas sobre la placa de cobre y corregir código G de fresado.

Código 2: **Programa de medición y corrección de alturas en Python**

```
1  # Height probing
2  # Autor: Rodrigo Chang
3  # Ultima fecha de modificacion: 9 de noviembre de 2014
4
5  # Programa para hacer probing en PCB utilizando G38.2 de GRBL y comunicacion por puerto
       serial 8N1@115200
6
7  import serial
8  import re
9  import sys
10 import os
11 import numpy as np
12 import matplotlib.pyplot as plt
13 from mpl_toolkits.mplot3d import Axes3D
14 from matplotlib import cm
15 from scipy import interpolate
16
17 # Datos por defecto para la cuadricula
18 DELTA_X = -10 # 1cm
19 DELTA_Y = 10
20 MILL_DEPTH = 0.100
21
22 # Patron de captura de alturas devuelto por GRBL
23 PROBE_PATTERN = '\[PRB:[-\d.]+,[-\d.]+,([-\d.]+)\]'
24
```

```
# Patron de reemplazo para archivos de codigo G
# Encuentra los codigos G1 que van hacia un punto X, Y
G1_PATTERN = 'G0?1 X([-0-9.]+) Y([-0-9.]+)'
# Encuentra los codigos G0 X# Y# y luego G1 Z# (rapido hacia coord. (X,Y) y baja la herramienta)
G0Z_PATTERN = 'G0?0 X([-0-9.]+) Y([-0-9.]+)\nG1 Z([-0-9.]+)'

# Archivo de salida por defecto para puntos X, Y, Z del mapa de alturas.
OUTPUT_FILE = 'mapa_alturas.txt'

# Puerto serial para GRBL
SERIAL_PORT = 'COM31'
BAUDRATE = 115200

'''
    Devuelve la lista de puntos utilizando una trayectoria en zig-zag
    l (int > 0) -> unidades de DELTA_X a utilizar para generar la lista
    h (int > 0) -> unidades de DELTA_Y a utilizar para generar la lista
'''
def listaPuntos(l, h, dx=DELTA_X, dy=DELTA_Y):
    # Iterar sobre el eje Y y recorrer X en zigzag
    direccion = 0 # direccion inicial X en la direccion de DELTA_X
    puntos = [] # lista de puntos para devolver
    for j in range(h + 1):
        if (direccion == 0):
            for i in range(l + 1):
                puntos.append((i*dx, j*dy))
            direccion = 1
        else:
            for i in range(l, -1, -1):
                puntos.append((i*dx, j*dy))
            direccion = 0
    return puntos

'''
    Realiza el probing controlando el puerto serial especificado.
    Escribe en el archivo 'file' los puntos y devuelve un
    diccionario utilizando las tuplas de puntos (x,y) como llaves
    y las alturas como valores.

    CUIDADO: Esta funcion envia comandos para mover la maquina,
    asegurarse que el area de puntos este despejada. Mover la maquina de forma
    manual al punto que servira como (0,0) y a cualquier altura.

    puntos -> lista de tuplas, e.g. [(x0, y0), (x1, y1), ..., (xi, yj)]
    port, baudrate -> nombre y velocidad del puerto serial
    filename -> Archivo de salida para escribir los puntos (x,y,z) separados por '\t'
    pattern -> Patron para detectar la lectura obtenida por GRBL para la funcion G38.2
'''
def realizarProbing(puntos, port=SERIAL_PORT, baudrate=BAUDRATE, pattern=PROBE_PATTERN, filename=OUTPUT_FILE):
    # Abrir el puerto serial
    print 'Iniciando conexion con puerto serial...'
    puerto = serial.Serial(port, baudrate, timeout=2)
    resp = puerto.readlines()
    for l in resp:
        print l,
    # Desbloquear
    puerto.write('$X\n')
    resp = puerto.readlines()
    for l in resp:
        print l,

```

```
86      # Abrir un archivo y guardar los puntos
87      probemap = {}
88      f = open(filename, 'w')
89
90      # Hacer probing para cada punto en la lista
91      puerto.write('G90\n')
92      puerto.write('G21\n')
93      for punto in puntos:
94          print 'Probando el punto (%-4.3f,%-4.3f)' % (punto[0], punto[1])
95          puerto.write('G1 Z1.000 F95.00\n')
96          puerto.write('G0 X%-4.3f Y%-4.3f\n' % (punto[0], punto[1]))
97          puerto.write('G38.2 Z-20.000 F30.00\n')
98          print 'Esperando lectura...',
99          # Leer lineas hasta obtener el patron esperado
100         while True:
101             # Lee la linea recibida por GRBL
102             l = puerto.readline()
103             # Ver si el patron coincide con el formato dado por G38.2
104             result = re.match(pattern, l, re.IGNORECASE)
105             # Si el patron coincide
106             if result != None:
107                 # Leer la profundidad recibida
108                 depth = float(result.groups()[0])
109                 # Si el punto es (0,0) utilizarlo como referencia z=0
110                 if punto == (0,0):
111                     ref = depth
112
113                 # Obtener cada profundidad a partir de la referencia
114                 depth = depth - ref
115                 # Guardar la altura del punto en el diccionario e imprimirla en pantalla
    y al archivo
116                 probemap[punto] = depth
117                 print depth
118                 f.write('%-4.3f\t%-4.3f\t%-4.3f\n' % (punto[0], punto[1], depth))
119                 # Sale del ciclo infinito
120                 break
121
122     # Fin del probing, cerrar el archivo
123     f.close()
124     print 'Fin del probing...'
125     # Regresar al origen
126     puerto.write('G1 Z1.000 F95.00\n')
127     puerto.write('G0 X0 Y0\n')
128     # Cerrar el puerto serial
129     puerto.close()
130
131     # Devolver el mapa de alturas
132     return probemap
133
134 '''
135     Recibe el mapa de alturas y devuelve las listas X, Y, Z correspondientes.
136 '''
137 def probeMapToList(probeMap):
138     xm = []
139     ym = []
140     zm = []
141     for p in probeMap.keys():
142         xm.append(p[0])
143         ym.append(p[1])
144         zm.append(probeMap[p])
145
146     return xm, ym, zm
147
```

```
'''
    Recibe el diccionario de puntos y alturas y grafica los puntos en una vista
    tridimensional, si recibe una funcion grafica tambien la superficie de interpolacion.
'''
def graficarMapa(probeMap, function=None):
    # Obtener la lista de valores x,y,z
    xm, ym, zm = probeMapToList(probeMap)

    # Crea la ventana
    fig = plt.figure(1)
    ax = Axes3D(fig)
    # Graficar los puntos en 3D
    ax.plot(xm, ym, zm, 'go', linewidth=2, markersize=11)

    # Si se especifica la funcion, graficar la version interpolada
    if (function != None):
        xnew = np.arange(min(xm), max(xm), 0.1)
        ynew = np.arange(min(ym), max(ym), 0.1)
        znew = function(xnew, ynew)

        xx, yy = np.meshgrid(xnew, ynew)
        ax.plot_surface(xx, yy, znew, cmap=cm.coolwarm, rstride=20, cstride=20)

    # Muestra la grafica
    plt.show()
    #plt.savefig('mapaAlturas.pdf')

'''
    Obtiene la funcion de interpolacion utilizando TODOS los puntos de probeMap
    Utiliza la funcion interpolate.interp2d con 'cubic'

    Devuelve la funcion de interpolacion
'''
def interpolarMapa(probeMap):
    # Obtener la lista de valores x,y,z
    xm, ym, zm = probeMapToList(probeMap)
    # Interpolar la funcion
    f = interpolate.interp2d(xm, ym, zm, kind='cubic')
    # Devuelve el objeto de funcion
    return f

'''
    Define una clase para almacenar la matriz de funciones de interpolacion.
    Utiliza el metodo de llamada __call__ para valuar un punto (x,y)
    Para calcular la funcion a utilizar, requiere dx y dy
'''
class BilinearMatrix(object):
    def __init__(self, matrixF, dx=DELTA_X, dy=DELTA_Y):
        self.matrixF = matrixF
        self.dx = dx
        self.dy = dy

    def __call__(self, x, y):
        i = int(x / self.dx)
        j = int(y / self.dy)
        f = self.matrixF[i][j]
        return f(x,y)

'''
    Obtiene la funcion de interpolacion dividendo en areas de dx*dy y utilizando
    los 4 puntos mas cercanos.
'''
def interporlarMapa2(probeMap, l, h, dx=DELTA_X, dy=DELTA_Y):
```

```
    listaF = []
    for i in range(l):
        rowF = []
        for j in range(h):
            xi = i*dx
            xii = (i+1)*dx
            yj = j*dy
            yjj = (j+1)*dy
            xm = [xi, xii, xi, xii]
            ym = [yj, yj, yjj, yjj]
            zm = [probeMap[(xi,yj)], probeMap[(xii,yj)], probeMap[(xi,yjj)], probeMap[(
    xii,yjj)]]

            f = interpolate.interp2d(xm, ym, zm, kind='linear')
            rowF.append(f)

        listaF.append(rowF)
    return listaF

'''
    Modifica el archivo de codigo G con la funcion especificada
    filename -> str: La ruta hacia el archivo a modificar
    f -> Es el objeto de funcion devuelto por interp2d para modificar los puntos X y Y
    prof_fresado -> profunidad de fresado a partir de z=0 (valor positivo)
'''
def modificarArchivo(filename, f, prof_fresado):
    '''
        Funcion que recibe el nombre original del archivo y el prefijo a agregar.
        Devuelve el nuevo nombre del archivo
    '''
    def opt_name(s, prefix):
        assert '.' in s
        i = -1
        while s[i] != '.':
            i = i - 1
        old_ext = s[i:]
        return s[-len(s): i] + prefix + old_ext

    '''
        Reemplaza las ocurrencias utilizando el patron G1_PATTERN, para cada punto (x,y)
        obtiene la profundidad z = f(x,y) y la agrega en codigo G
    '''
    def f_repl_g1(f):
        def repl_g1(match):
            x = float(match.group(1))
            y = float(match.group(2))
            z = f(x, y)[0] - prof_fresado
            return ('G1 X%-4.4f Y%-4.4f Z%-4.3f' % (x, y, z))
        return repl_g1

    '''
        Reemplaza las ocurrencias utilizando el patron G0Z_PATTERN, para cada punto (x,y)
        obtiene la profundidad z = f(x,y) y la agrega en codigo G
    '''
    def f_repl_g0z(f):
        def repl_g0z(match):
            x = float(match.group(1))
            y = float(match.group(2))
            z = f(x, y)[0] - prof_fresado
            return ('G0 X%-4.4f Y%-4.4f\nG1 Z%-4.3f' % (x, y, z))
        return repl_g0z

    # Abre el archivo especificado y obtiene el codigo G
```

```
273     gcode = open(filename, 'r').read()
274     # Limpiar el codigo G de espacios dobles o mas
275     gcode = re.sub('[ ]{2,}', ' ', gcode)
276     gcode = re.sub(' \n', '\n', gcode)
277 
278     # Reemplazar las ocurrencias de G1 X# Y# enviando la funcion de superficie
279     gcode = re.sub(G1_PATTERN, f_repl_g1(f), gcode)
280     # Reemplazar las ocurrencias de G0 X# Y#\nG1 Z# enviando la funcion de superficie
281     gcode = re.sub(G0Z_PATTERN, f_repl_g0z(f), gcode)
282 
283     # Guardar el archivo modificado
284     wf = open(opt_name(filename, '.LEV'), 'w')
285     wf.write(gcode)
286     wf.close()
287 
288 '''
289     Imprime las instrucciones para iniciar el programa correctamente
290     desde la linea de comandos
291 '''
292 def imprimeInstrucciones():
293     #print '=- Programa de Probing -=\n'
294     print 'Utilizacion correcta:'
295     print '\tpython probing.py -f <archivo> <x> <y> <dx> <dy> <prof_z>'
296     print '\tpython probing.py -f <archivo> <x> <y>'
297     print '\tpython probing.py -p <x> <y> <dx> <dy>'
298     print '\tpython probing.py -p <x> <y>\n'
299     print 'x,y\t:\tDimensiones del area a probar'
300     print 'dx, dy\t:\tCambios en ambos ejes, por defecto dx=-10mm, dy=10mm'
301     print 'prof_z\t:\tProfundidad de fresado a partir de la superficie de contacto'
302 
303 '''
304     Realiza la rutina general de probing utilizando el tamano de la placa.
305     Opcionalmente modifica el archivo de codigo G especificado.
306     length_x -> tamano de la placa en el eje X (en mm)
307     length_y -> tamano de la placa en el eje Y (en mm)
308     prof_z -> profundidad de fresado a partir de la superficie de contacto de la placa
309     filename -> Nombre del archivo para modificar el codigo G
310 '''
311 def rutinaGeneral(length_x, length_y, dx = DELTA_X, dy = DELTA_Y, prof_z = MILL_DEPTH,
    filename=None):
312     # Obtener la lista de puntos
313     print 'Generando la lista de puntos...'
314     l = abs(length_x / dx)
315     h = abs(length_y / dy)
316     puntos = listaPuntos(l, h, dx, dy)
317     print 'La lista de puntos es: ', puntos
318 
319     # Obtener las alturas de los puntos
320     print 'Realizando el mapa de alturas...'
321     probemap = realizarProbing(puntos)
322     print 'Mapa de alturas: ', probemap
323     print 'Se ha terminado el mapa, generando el modelo...'
324 
325     # Obtener la funcion de interpolacion por areas
326     listaFunciones = interporlarMapa2(probemap, l, h, dx, dy)
327     bmatrix = BilinearMatrix(listaFunciones, dx, dy)
328 
329     # Graficar el mapa de alturas
330     graficarMapa(probemap)
331 
332     # Si se especifico, modificar el archivo original con el mapa obtenido.
333     if (filename != None):
334         print 'Modificando el archivo original...'
```

```
335         modificarArchivo(filename, bmatrix, prof_z)
336
337 '''
338     Programa principal
339 '''
340 print '\n=- Programa de Probing -=\n'
341 args = len(sys.argv)
342
343 # Si solo esta el nombre del programa, pedir el tamano de la placa
344 if (args == 1):
345     length_x = int(raw_input('Ingrese largo en X [mm]: '))
346     length_y = int(raw_input('Ingrese alto en Y [mm]: '))
347     # Realizar procedimiento general
348     rutinaGeneral(length_x, length_y)
349
350 # Si hay mas argumentos, revisar si es probing o modificacion de archivo
351 else:
352     opcion = sys.argv[1]
353     print 'Argumentos: ', args
354
355     # Si la opcion es solamente mapeo de alturas
356     if (opcion == '-p') and (args in [4,6]):
357         # Obtener los argumentos
358         length_x = int(sys.argv[2])
359         length_y = int(sys.argv[3])
360         # Si se especifican todos los argumentos
361         if (args == 6):
362             delta_x = int(sys.argv[4])
363             delta_y = int(sys.argv[5])
364             # Realizar el procedimiento general
365             rutinaGeneral(length_x, length_y, delta_x, delta_y)
366         # Sino, tomar los valores especificados por defecto
367         else:
368             # Realizar el procedimiento general
369             rutinaGeneral(length_x, length_y)
370
371     # Si la opcion es modificacion de archivo
372     elif (opcion == '-f') and (args in [5,8]):
373         # Obtener los argumentos
374         filename = sys.argv[2]
375
376         # Revisar si el archivo existe
377         if (os.path.isfile(filename)):
378             length_x = int(sys.argv[3])
379             length_y = int(sys.argv[4])
380             # Si se especifican todos los argumentos
381             if (args == 8):
382                 dx = int(sys.argv[5])
383                 dy = int(sys.argv[6])
384                 prof_z = float(sys.argv[7])
385                 # Realizar la rutina general especificando todos los parametros
386                 rutinaGeneral(length_x, length_y, dx, dy, prof_z, filename)
387             # Sino, tomar los valores especificados por defecto
388             else:
389                 # Realizar la rutina general con valores por defecto
390                 rutinaGeneral(length_x, length_y, filename=filename)
391
392         # Si no existe el archivo especificado
393         else:
394             print 'El archivo especificado no existe...'
395
396     # Si la opcion indicada es incorrecta
397     else:
```

```
398         imprimeInstrucciones()
```

Archivo de configuración generic.pp de pcb-gcode

En el código 3 se muestra la configuración de estilo del código G generado por el ULP pcb-gcode para fresado de placas de circuitos impresos. Dicho archivo debe ser copiado en la carpeta de instalación del ULP.

Código 3: **Configuración de perfil de generación de código G de pcb-gcode**

```
1  //
2  // Options for pcb-gcode.ulp.
3  // Often used options are at the top of the file.
4  // Copied to gcode-defaults.h by the setup program.
5  //
6  // author=John Johnson
7  // description=Tries to be very compatible
8  //
9
10 int FILENAMES_8_CHARACTERS = NO;
11
12 //
13 // Comments.
14 //
15 string COMMENT_BEGIN  = "(";
16 string COMMENT_END    = ")";
17
18 //
19 // Format strings for coordinates, etc.
20 //
21 string FORMAT     = "%-6.4f ";        /* coordinate format */
22 string FR_FORMAT  = "F%-5.2f ";       /* feedrate format */
23 string IJ_FORMAT  = "I" + FORMAT + "J" + FORMAT;
24 string EOL        = "\n";            /* standard line ending */
25 string PARAM      = "P";             /* some use P, some # for parameters */
26 //
27 // Modes
28 //
29 string INCH_MODE            = "G20" + EOL;
30 string INCH_MODE_COMMENT    = COMMENT_BEGIN + "Inch Mode" + COMMENT_END + EOL;
31 string METRIC_MODE          = "G21" + EOL;
32 string METRIC_MODE_COMMENT  = COMMENT_BEGIN + "Metric Mode" + COMMENT_END + EOL;
33 string MIL_MODE             = "M02 (Please setup MIL_MODE in gcode-defaults.h)" + EOL;
34 string MICRON_MODE          = "M02 (Please setup MICRON_MODE in gcode-defaults.h)" + EOL;
35 string ABSOLUTE_MODE        = COMMENT_BEGIN + "Absolute Coordinates" + COMMENT_END + EOL
       + "G90" + EOL;
36
37 //
38 // G codes
39 //
40 string RAPID    = "G0 ";
41 string FEED     = "G1 ";
42 string ARC_CW   = "G2 ";
43 string ARC_CCW  = "G3 ";
44 //string DWELL    = "G4 " + PARAM + "%f" + EOL;
45 string DWELL    = "G4 " + PARAM + FORMAT + EOL;
46
```

```
//
// M codes
//
string SPINDLE_ON     = "M3" + EOL + DWELL;
string SPINDLE_OFF    = "M5" + EOL;
string END_PROGRAM    = "M2" + EOL;
string OPERATOR_PAUSE = "M6 ";

//
// Coordinates
//
string MOVE_X   = "X" + FORMAT;
string MOVE_Y   = "Y" + FORMAT;
string MOVE_XY  = "X" + FORMAT + "Y" + FORMAT;
string MOVE_Z   = "Z" + FORMAT;
string MOVE_XYZ = MOVE_XY + MOVE_Z;

//
// Rapids
//
string RAPID_MOVE_X       = RAPID + MOVE_X;
string RAPID_MOVE_Y       = RAPID + MOVE_Y;
string RAPID_MOVE_XY      = RAPID + MOVE_XY;
string RAPID_MOVE_XY_HOME = RAPID + "X0 Y0";
string RAPID_MOVE_Z       = RAPID + MOVE_Z;
string RAPID_MOVE_XYZ     = RAPID + MOVE_XYZ;

//
// Feeds
//
string FEED_MOVE_X            = FEED + MOVE_X;
string FEED_MOVE_Y            = FEED + MOVE_Y;
string FEED_MOVE_XY           = FEED + MOVE_XY;
string FEED_MOVE_XY_WITH_RATE = FEED + MOVE_XY + FR_FORMAT;
string FEED_MOVE_Z            = FEED + MOVE_Z;
string FEED_MOVE_Z_WITH_RATE  = FEED + MOVE_Z   + FR_FORMAT;
string FEED_MOVE_XYZ          = FEED + MOVE_XYZ;

//
// Drilling holes
//
// Not using G82 so it will be very generic.
//
string DRILL_CODE       = ";( G82 not used )";
string RELEASE_PLANE    = "R" + FORMAT;
string DWELL_TIME       = PARAM + "%f";

string DRILL_FIRST_HOLE = RAPID + "Z" + real_to_string(DEFAULT_Z_UP) + EOL
                                + RAPID + MOVE_XY + EOL
                                + FEED + MOVE_Z + FR_FORMAT + EOL
                                + FEED + "Z" + real_to_string(DEFAULT_Z_UP) + EOL
                                + COMMENT_BEGIN + RELEASE_PLANE + " " + DWELL_TIME +
    COMMENT_END + EOL;

string DRILL_HOLE = COMMENT_BEGIN + RAPID + "Z" + real_to_string(DEFAULT_Z_UP) +
    COMMENT_END + EOL
                                  + RAPID + MOVE_XY + EOL
                                  + FEED + "Z" + real_to_string(DRILL_DEPTH) + EOL
                                  + FEED + "Z" + real_to_string(DEFAULT_Z_UP) + EOL;

//
// Tool change
//
```

```
string TOOL_CODE        = "T%02d ";
string TOOL_MM_FORMAT   = "%1.3fmm";
string TOOL_INCH_FORMAT = "%1.4fin";
string TOOL_CHANGE      = OPERATOR_PAUSE + TOOL_CODE + " ; " + FORMAT + EOL;

string TOOL_CHANGE_TABLE_HEADER = COMMENT_BEGIN +
  " Tool|       Size       |  Min Sub |  Max Sub |   Count " + COMMENT_END + EOL;

string TOOL_CHANGE_TABLE_FORMAT(int tool_number, real size_mm, real size_inch, real
    min_drill, real max_drill, int count)
{
  string formatted;

  sprintf(formatted, COMMENT_BEGIN + " " + TOOL_CODE + " " + TOOL_MM_FORMAT + " " +
    TOOL_INCH_FORMAT + " " + TOOL_INCH_FORMAT + " " + TOOL_INCH_FORMAT + " " +
    COMMENT_END + EOL,
    tool_number, size_mm, size_inch, min_drill, max_drill);
  return(formatted);
}

//
// Circles / Arcs
//
string CIRCLE_TOP     = ARC_CW + MOVE_XY + IJ_FORMAT + EOL;
string CIRCLE_BOTTOM  = ARC_CCW + MOVE_XY + IJ_FORMAT + EOL;
```

Archivo de configuración de barrenado default.drl (Drill Rack File)

En el código 4 se muestra la configuración del archivo de brocas disponibles para generación del código G de barrenado.

Código 4: **Configuración de brocas disponibles para generación de código G**

```
# -*- Mode: Eagle -*-
#
# Sample drill rack file.
#
# drill_size is the size of the drill bit.
# minimum is the smallest hole the bit will be used for.
# maximum is the largest hole the bit will be used for.
#
# Each value can have a "unit of measure" added to it,
# such as 0.500mm . If you do not provide the unit of measure,
# values over 0.250 are considered millimeters, and
# values under 0.250 are considered inches. For example:
# 0.400 would be considered 0.400 mm.
# 0.125 would be considered 0.125 inches.
#
# Please note that you must use a TAB character
# between each setting on a line.
#
# Tip: Set the TAB size of your editor to 12 characters.
#
tool  drill_size  minimum maximum length
T01 0.032in 0.000in 0.125in 1.5in
```