Universidad del País Vasco
Euskal Herriko Unibertsitatea

ZIENTZIA ETA TEKNOLOGIA FAKULTATEA
FACULTAD DE CIENCIA Y TECNOLOGÍA

Gradu Amaierako Lana / Trabajo Fin de Grado
Ingenieritza elektronikoko Gradua / Grado en Ingeniería electrónica

Diseño e implementación de un sistema de control numérico

Egilea/Autor/a:
Asier Vidal Bartolomé
Zuzendaria/Director/a:
Iker Caballero O. de Zarate

Leioa, 2018ko Uztailaren 20a /Leioa, 20 de junio de 2018

Índice:

Introducción y objetivos

Desde hace años, la máquina herramienta ha jugado un papel fundamental en el desarrollo tecnológico mundial. Ahora, gracias a los avances de la electrónica ya es posible controlar este tipo de maquinaria de forma automatizada. De esta forma, se obtiene la MHCNC *(Máquina Herramienta Controlada por Computador).* Se trata de un sistema de automatización de máquinas basado en coordenadas y usado en diversos sectores. Con el auge de las impresoras 3D, los sistemas de control numéricos han provocado una revolución llevando la funcionalidad al máximo y los costes al mínimo para aplicaciones básicas.

En este proyecto, se ha diseñado un sistema de control numérico sencillo para la realización de diferentes tareas de manera offline y adhoc. Este tipo de desarrollos se realizan dentro de un proyecto de mayor envergadura, ya sea para una máquina herramienta de mayores prestaciones o de una cadena de fabricación, estableciendo así tareas más cortas y controles unitarios si fuera necesario.

Por tanto, los objetivos del proyecto son:

- Análisis de las aplicaciones actuales de máquinas de control numérico.
- Búsqueda de componentes habituales en el desarrollo de máquinas CNC.
- Diseño de una electrónica para el control de la máquina.
- Desarrollo software para una aplicación offline específica.
- Estudio de fabricabilidad y costes.

Se han aplicado conocimientos adquiridos a lo largo de la carrera así como otros aprendidos durante el desarrollo de este proyecto. A modo general, se presentan las áreas de conocimiento utilizadas:

- Diseño electrónico y prototipado.
- Desarrollo de firmware.
- Diseño de producto.
- Interfaces HMI.

Capítulo 1 . Estado del arte

1.1. La máquina herramienta CNC

En primer lugar, para dar una visión global de lo que se expondrá a continuación, se ha de entender qué es una máquina herramienta CNC (MHCNC). Las siglas CNC es un acrónimo, y se refiere a 'Control Numérico por Computador'. La CNC es una máquina herramienta automatizada sobre la que se ejecutan diversos programas de control numérico generados por software CAD/CAM **(véase apastados 1.2.1 y 1.2.2)** a partir de un dibujo en 2D o 3D, dependiendo del número de ejes a controlar. El control del movimiento de la herramienta se realiza mediante un código de programación que se genera desde un ordenador, y que un controlador interpreta. En este tipo de máquinas se controlan dos, tres o más ejes de movimiento mediante motores eléctricos, generalmente motores paso a paso, véase apartado **1.3.3.** [B1]

La configuración más común para este tipo de máquinas, es la disposición de tres ejes cartesianos XYZ. En el caso más simple, el eje Z sirve únicamente de anclaje para la herramienta de corte y se mecaniza la pieza capa a capa en planos bidimensionales descendentes. Es decir, una vez la herramienta de corte ha alcanzado la profundidad estipulada en la configuración previa al mecanizado, el eje Z permaneces estático, y los ejes XY son los encargados de realizar el trabajo. Un paso adelante en esta disposición, es la introducción de un control en el eje vertical e interpolar el movimiento de este junto con los ejes horizontales, para conseguir así realizar curvas tridimensionales en el espacio.

Las configuraciones anteriormente mencionadas son las más habituales, pero existen otras más complejas, introduciendo un mayor número de ejes, dependiendo de la complejidad de la pieza a mecanizar. En la figura inferior, se dan un par de ejemplos para MHCNC de tres y cuatro ejes respectivamente.

Figura 1.1. Fresadora de tres ejes (izquierda) y cuatro ejes (derecha).

1.1.1. Aplicaciones de la máquina herramientas CNC

La máquina herramienta CNC es la mejor solución para todo tipo de máquinas y sectores. Gracias a su alto rendimiento y precisión se obtiene una mayor calidad en el producto final.

Algunas máquinas que utilizan sistemas de CNC son las siguientes: centros de torneado o mecanizado, impresoras 3D, cortadora laser, máquina de corte por chorro de agua, fresadoras, prensas, dobladoras o cualquier otra aplicable al modelado de metales. En definitiva, cualquier máquina en la que pueda implementarse un movimiento automático y programado.

El ámbito de aplicaciones es muy extenso, tanto para grandes como para pequeñas series de producción. A priori se podría pensar, que la MHCNC están pensadas para un alto volumen de producción, pero en ocasiones este tipo de máquinas es la solución perfecta para realizar pequeñas tiradas, donde los beneficios totales no compensen la fabricación de moldes. Un ejemplo sería la creación de prototipos mediante impresoras 3D, con las que se consiguen muy buenos resultados mediante diferentes técnicas, como FDM (Modelado por Deposición fundida, del inglés, Fused Deposition Modeling) o SLS (Sinterizado Selectivo por Laser, del inglés, Selective Laser Sintering). A continuación, se muestran unas imágenes de dos piezas acabadas mediantes estas técnicas. [B23] [B24]

Figura 1.2. Pieza impresa por FDM

Figura 1.3. Pieza impresa por SLS

1.1.2. Ventajas

Este tipo de máquinas presentan ciertas ventajas, frente a las mesas XYZ con taladros verticales de acción manual, debido a la automatización de las mismas y la digitalización de los diseños.[B2] [B22]

- Mayor productividad, es posible usar estas máquinas las 24 horas del día sin necesidad de desconexión a excepción de las requeridas para su mantenimiento.

- Una única persona puede encargarse de la supervisión de varios centros de mecanizado.

- Reducción de la intervención del usuario reduciendo el error humano, y por tanto mejorando la precisión y velocidad.

- Aumenta la seguridad de los operarios en entornos peligrosos, debido a la ausencia de personal.

- Es posible simular la fabricación de una pieza, de este modo, en algunos casos, no será necesaria la creación de un prototipo. Lo que supone un ahorro de tiempo y dinero.

- Flexibilidad en cuanto al cambio de diseño.

1.1.3. Desventajas

Como contraparte, también existen una serie de desventajas que se tienen que tener en cuenta. [B2]

- Debido a la electrónica asociada y al software de diseño necesario, el desembolso inicial es considerablemente mayor.

- Aumento del coste de mantenimiento, debido a la complejidad de los sistemas de control.

- Desempleo. Estas máquinas no requieren de muchos trabajadores para su puesta en funcionamiento, generando desempleo y escasez de trabajo.

- Capacitación del operador. Al aumentar la complejidad de la maquinaria, se necesita de personal con capacitación y formación específica.

1.2. Control numérico

Es el encargado de dar la posición de uno o diversos componentes mecánicos, que constan de instrucciones relativas a su desplazamiento automático. Estas instrucciones constan de valores numéricos y simbólicos definidos en un código. Por lo tanto, el control numérico envía números y letras a la máquina herramienta, que interpreta a través de un controlador y los convierte en pulsos eléctricos para controlar los movimientos de los ejes a través del accionamiento de los motores, con el fin de llevar a cabo el proceso de mecanizado de la pieza de trabajo.

Los elementos que entran en juego a la hora de realizar el mecanizado son:

- **La máquina:** Se refiere a la estructura en sí misma. La cual está compuesta por la mesa, que delimita la superficie de trabajo, los rieles que componen cada eje coordenado, los motores y la herramienta de trabajo, como el taladro, laser u otro.

- **El programa:** Se refiere a las líneas de código generadas mediante el uso de software CAD/CAM. Este código abarca la información necesaria para el mecanizado del producto final. Este se almacena con un formato concreto (.nc, .cnc, .ccnc, etc.).

- **El controlador:** Es el conjunto de la parte electrónica de la MHCNC y el firmware cargado en el micro-controlador de este. Es el encargado de interpretar las instrucciones recogidas en el archivo anteriormente mencionado y a su vez las convierte en señales o pulsos eléctricos, generando las señales de control para los respectivos actuadores de la máquina. [B2]

1.2.1. Software CAD (Diseño Asistido por Computador)

Es el software utilizado para el diseño físico de la pieza. Existe una gran variedad de ellos, con diferentes enfoques y funcionalidades, dependiendo de las necesidades del trabajo y/o del usuario. Muchos de estos programas son de carácter gratuito, por lo que no solo están pensados para grandes empresas, sino que también un usuario particular es capaz de realizar diseños propios.

Dentro del ámbito electrónico, existen programas especializados en el diseño de placas de circuito impreso o PCB, algunos de los más utilizados son: Altium, OrCad, Eagle, entre otros. Por otra parte, para un diseño mecánico, existen diversas opciones: Fusion 360, FreeCAD, LibreCAD, etc. Los dos primeros especializados en diseños 2.5D y 3D mientras que el tercero se enfoca en el diseño de grabados en 2D. Una vez finalizado el diseño, estos programas brindan la posibilidad de generar archivos Gerber para poder enviarlos a un software CAM. [B26] [B27]

1.2.2. Software CAM (Manufactura Asistida por Computador)

El software CAM sirve de puente entre el CAD y el lenguaje de programación de las máquinas herramienta. Los archivos generados por este tipo de software se conocen como Gerbers, que incluyen el código numérico de programación de las MHCNC.

Una función importante en operaciones de mecanizado es la posibilidad de describir la trayectoria de la herramienta, el software CAM utiliza los modelos creados en el software CAD para generar dichas trayectorias, con esto se consigue evitar colisiones con los soportes de la máquina o con los posibles salientes de la pieza.

Cada vez se hace más difícil diferenciar entre CAD y CAM. Actualmente, la gran mayoría de software de diseño CAD vienen con un modulo CAM integrado, de esta forma se elimina la necesidad de transferencia de archivos entre diferentes programas, agilizando el proceso de diseño y fabricación. En la actualidad, existe una gran variedad de software CAD/CAM tanto de código libre como propietarios. [B25]

1.2.3. Código G

El código G o G-Code, es el nombre de un lenguaje de descripción de operaciones para máquinas de control numérico (CNC). La consideración de estandarizar este lenguaje de programación, tuvo lugar debido al gran aumento de máquinas CNC en el mercado. En un

principio se bautizó como ISO-6983, aunque existen versiones particulares de algunos fabricantes así como otros estándares, como la alternativa alemana DIN 66025. [B8]

El G-Code se almacena en formato texto, que puede leerse y modificarse con un editor de texto plano, aunque lo más habitual es que se genere mediante algún software de diseño y/o modelado.

Este código describe el movimiento y las diferentes operaciones que la máquina herramienta CNC debe realizar, como la puesta en marcha de los motores que generan el movimiento en los ejes o la activación de la herramienta de corte, taladro, laser, etc.

Los programas de fabricación asistida por ordenador (CAM) suelen generar directamente un documento con las instrucción G-code para controlar los sistemas de fabricación. Aunque en el modelado 3D ocasionalmente necesitan un paso intermedio, en este caso se genera un archivo alternativo, generalmente en formato .STL, que puede ser leído por aplicaciones diseñadas para generar G-Code. En este paso se divide la pieza en láminas paralelas de un espesor configurable. El eje Z desciende esa distancia mientras que los ejes XY mecanizan el plano, una vez terminado, se repite el proceso hasta completar la pieza, pasando por todas y cada una de las láminas. Configurando un menor espesor se obtendrá una mayor definición y por tanto un producto final de mayor calidad. Este proceso recibe el nombre de, laminado. [B17]

La sintaxis del código G es la siguiente; GXX, conde XX es un numero entero entre el 00 y el 99. En la siguiente tabla se enumeran y describen algunos de estos comandos.[B18]

Comando	Descripción
G00	Interpolación lineal rápida
G01	Interpolación lineal a la velocidad programada en el registro F*
G02	Movimiento circular en sentido horario
G03	Movimiento circular en sentido anti-horario
G04	Una pausa con un tiempo especifico
G17	Selección del plano XY
G18	Selección del plano XZ
G19	Selección del plano YZ
G40	Compensación anulada, o al centro de la línea de desplazamiento
G41	Compensación a la izquierda de la línea de desplazamiento
G42	Compensación a la derecha de la línea de desplazamiento
G70	Unidad de datos expresados en Pulgadas
G71	Unidad de datos expresados en Milímetros
G90	Desplazamiento en modo absoluto
G91	Desplazamiento en modo incremental o relativo

Tabla 1.1. Listado de comandos G-Code.

La mayoría de los comandos de la tabla anterior son sencillos de entender, en algún caso habrá que hacer un inciso, como por ejemplo en el caso de las interpolaciones. Para una mejor comprensión de la interpolación lineal y de la circular véanse apartado **1.4.3.**

1.3. Motores

El actuador principal de este tipo de proyectos es el motor. A continuación, se enumeran y describen brevemente los tipos de motores en máquinas herramienta CNC.

1.3.1. Motor de corriente continua con escobillas

Es el tipo de motor más sencillo. Como muestra la **Figura 1.4**, este tipo de motores se compone de un armazón giratorio (rotor) y el componente fijo (estator). El rotor, contiene una o más bobinas que están eléctricamente conectadas al conmutador, que es un cilindro compuesto de varios segmentos de contacto metálicos. El estator encierra el rotor y contiene imanes permanentes o electroimanes que genera un campo magnético. Las escobillas son los contactos eléctricos entre los segmentos del conmutador y la alimentación de motor, estas están hechas de un material blanco como el carbón.

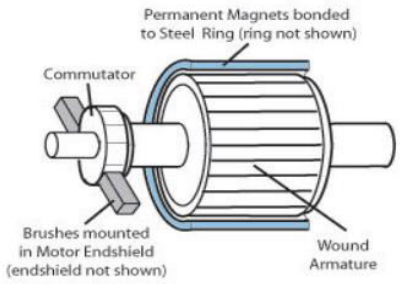


Figura 1.4. Construcción de un motor DC con escobillas.

Cuando se conecta la fuente de alimentación al motor, las bobinas del rotor se energizan, convirtiéndolas en un electroimán, se genera una fuerza de atracción entre los polos opuestos del rotor y estator, que hace que el rotor gire sobre su eje libremente. A medida que el conmutador gira, las escobillas entran en contacto con las diferentes secciones de este haciendo que la polaridad de la corriente de las bobinas del rotor se invierta, manteniendo así la fuerza de atracción, y por lo tanto el movimiento del eje motor. (**Véase Figura 1.5)**

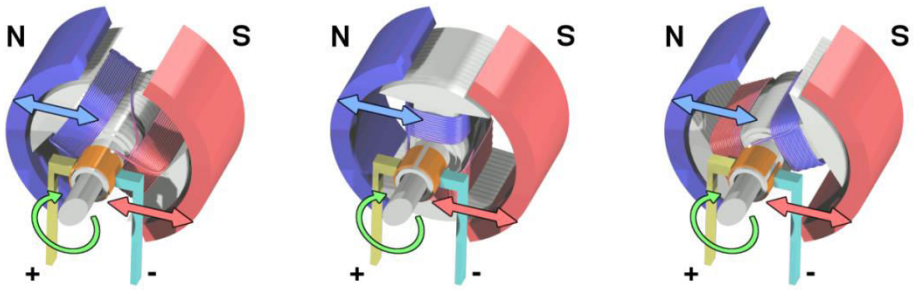


Figura 1.5. Diagrama de movimiento de un motor DC con escobillas.

El principal inconveniente de estos motores es el mantenimiento, costoso y laborioso, debido al desgaste sufrido por las escobillas al entrar en contacto con el conmutador. [B13] [B11]

1.3.2. Motor de corriente continua sin escobillas

El principio de funcionamiento de un motor DC sin escobillas es el mismo que el expuesto en el apartado anterior, salvo que en este caso, como su propio nombre indica, carece de escobillas y por lo tanto su construcción es muy distinta, como muestra la **Figura 1.6.**

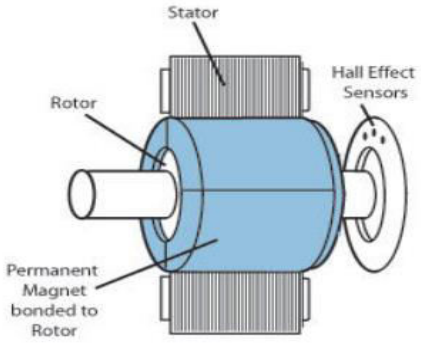


Figura 1.6. Construcción de un motor DC sin escobilla.

Al contrario que en el caso anterior, el imán permanente está colocado en el rotor y el estator es el que contiene los bobinados. En este caso, la conmutación se logra energizando sucesivamente las bobinas de alrededor de estator utilizando un controlador electrónico en conjunto con un sensor para conocer la posición del rotor, como por ejemplo un sensor de efecto Hall [B3] . Estos sensores se basan en el principio que lleva su nombre, por el cual se genera una caída de tensión entre los extremos de un conductor por el que circula una corriente, bajo la influencia de un campo magnético externo. De este modo es posible conocer la orientación del campo magnético y por ende la posición del rotor.[B11]

La principal desventaja de este tipo de motores es la complejidad en cuanto al control de cambio de polaridad. En el segundo de los casos, con la ausencia de sensores, cuando el motor gira a velocidades bajas en las cuales la intensidad tiene un valor muy pequeño, es difícil realizar un análisis de esta con exactitud.[B11] [B12]

1.3.3. Motor paso a paso (PAP)

Es un dispositivo electromecánico que convierte una serie de impulsos eléctricos en desplazamientos angulares discretos, es decir, es capaz de rotar una serie de grados (pasos) dependiendo de sus entradas de control. La cantidad de rotación es directamente proporcional al número de pulsos, y la velocidad de rotación es relativa a la frecuencia de los pulsos. El desplazamiento rotativo puede variar desde 90° (cuatro pasos por vuelta) hasta 1.8° (200 pasos por vuelta).[B14] [B15]

Poseen numerosas ventajas sobre los motores descritos anteriormente:

- Alta precisión en el posicionamiento, con un error que oscila entre un 3 y un 5% de la longitud del paso, no acumulable entre un paso y el siguiente.
- Posibilidad de aplicar par en situaciones estacionarias.
- Los únicos elementos de rozamiento son los rodamientos.
- Posibilidad de aplicar muy bajas velocidades de giro

Todas estas características los convierten en los motores ideales para aplicaciones donde se requieren movimientos precisos. Su principal inconveniente es que para su excitación requieren de una etapa lógica, denominada driver. [B4] [B10]

Existen diferentes tipos de motores PAP en función de su construcción, pero se pueden categorizar en tres grandes grupos. [B7]

❖ De reluctancia variable (VR)

Este tipo de motor está constituido por un rotor de material ferro-magnético no imanado, por este motivo no presentan par de retención, proporciona una buena respuesta dinámica, pero con un par motor relativamente bajo. Como muestra la **Figura 1.7**, el rotor está formando una serie de dientes (polos del rotor), estas ranuras conllevan una variación de la reluctancia en función de su posición angular. El estator se construye de forma similar, pero los dientes de este albergan las bobinas que forman los polos. Cada par de polos enfrentados se conoce como fase.[B9] [B19]

El número de dientes del rotor es menor que el del estator, de modo que solo un par de polos del estator y su correspondiente par de polos del rotor pueden estar alineados. Mediante el control de unos interruptores, se suministra la corriente a cada fase. Activando S1, la corriente fluye por la fase 1 (F1), generando un flujo magnético, en este instante los polos del rotor más próximos a los polos de la fase, son atraídos, generando así un movimiento de rotación. Alternando la activación y desactivación de los interruptores se consigue un movimiento circular continuo en ambos sentidos. **(Véase Figura 1.8)** [B9]

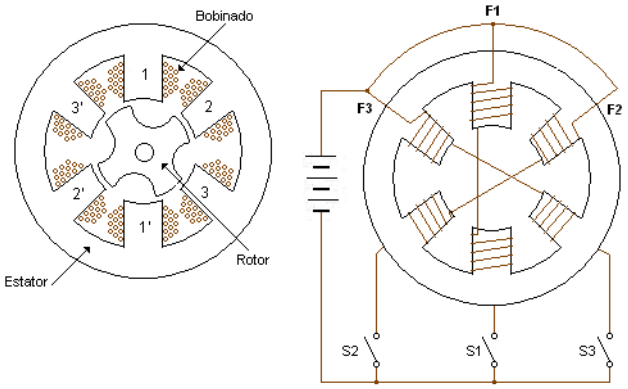


Figura 1.7. Construcción de un motor PAP de RV.

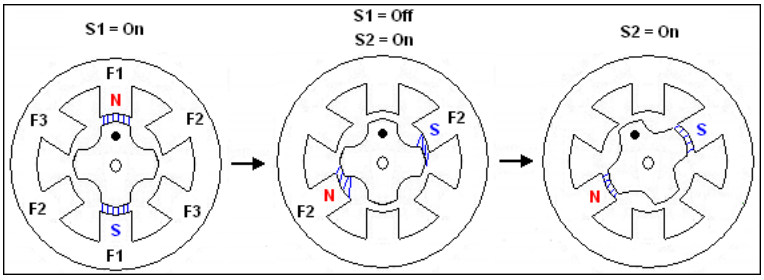


Figura 1.8. Secuencia de funcionamiento de un motor PAP de RV.

Este tipo de motores presentan un diseño más simple debido a que no requieren de un complejo rotor de imán permanente. La precisión de movimiento en relación a la cantidad de grados por paso, es relativamente baja, entre 5 y 15 grados por vuelta.[B10]

❖ De imán permanente

Este tipo de motores utiliza un imán permanente cilíndrico como rotor, el cual está magnetizado radialmente en una serie de polos, como muestra la **Figura 1.9**. El estator está construido con material ferro-magnético con el mismo número de polos que el rotor. [B19]

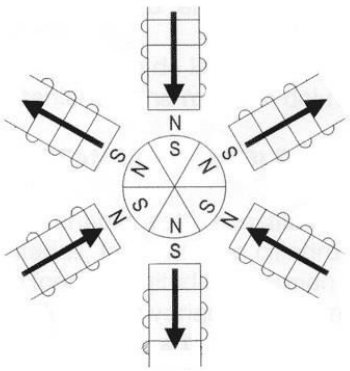


Figura 1.9. Construcción de un motor PAP de imán permanente.

Siguiendo la secuencia de la **Figura 1.10,** se energiza la fase A con un sentido determinado de la corriente, por lo tanto, el rotor se orienta de acuerdo al campo generado por dicha fase. Cambiando el sentido de la corriente de la fase B se invierte la polaridad de esta, haciendo que

el rotor gire un cuarto de vuelta. Realizando esto de forma sucesiva, se consigue mantener una rotación continua. [B19]

La cantidad de pasos por vuelta está limitada por el tipo de construcción, por lo tanto, los grados por paso son bastante elevados, desde 7.5° hasta 90°. [B9]

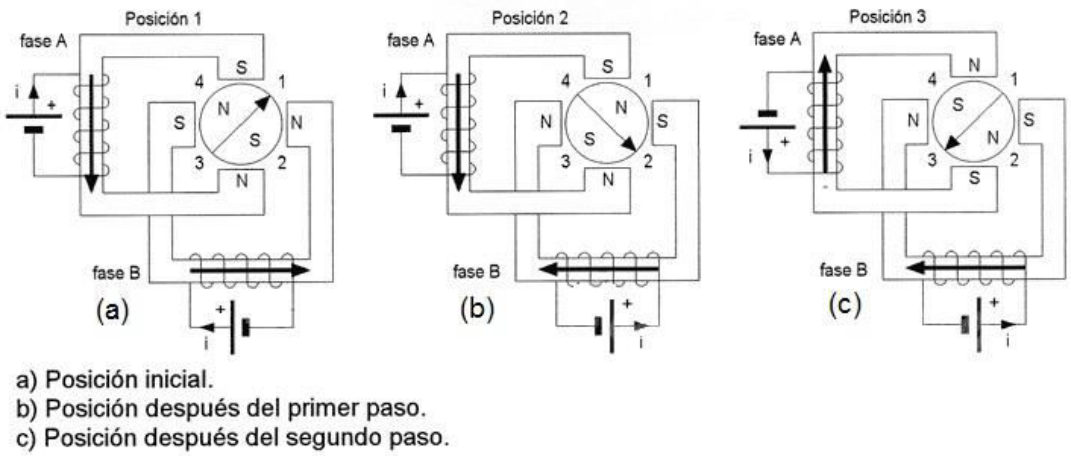


Figura 1.10. Secuencia de funcionamiento de un motor PAP de imán permanente.

❖ Híbridos

En este caso, se trata de una combinación de las dos tecnologías anteriores. Suelen estar construidos por anillos de acero dentado con un número de dientes ligeramente distinto al del estator, dichos anillos están montados sobre un imán permanente dispuesto axialmente. El rotor consta de tres partes (apilado simple), dos anillos de polos separados con los dientes desfasados una mitad de salto entre ellos (**Véase Figura 1.11 y Figura 1.12**), de esta forma se consigue una alta resolución, típicamente 1.8 grados, incluso de hasta 0.9 grados. [B20]

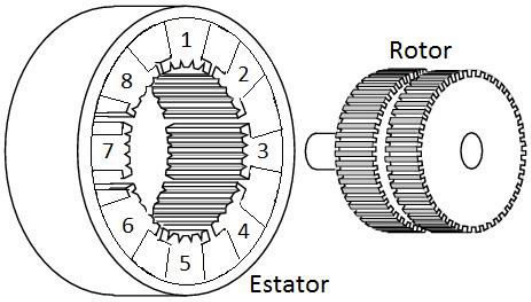


Figura 1.11. Construcción interna de un motor PAP híbrido.

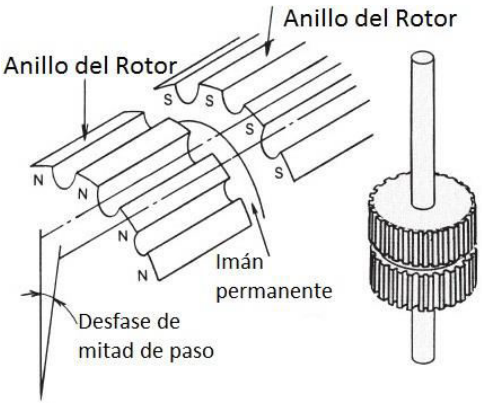


Figura 1.12. Detalle del desfase entre los dientes de los anillos del rotor.

Los motores de imán permanente o híbridos, a su vez, se pueden dividirse en dos subgrupos, unipolares y bipolares.

- ***Motor Unipolar***

Básicamente se componen de dos bobinas, cada una con una derivación en el centro. Estas derivaciones pueden estar unidas o separadas, de este modo, tendrán 5 o 6 cables que saldrán al exterior. Independientemente del número de cables, el cable de toma central (1 y 2 de la **Figura 1.13**) está conectado a una fuente de alimentación externa y los extremos de las bobinas serán llevados alternativamente a tierra. Así, la dirección de la corriente de las bobinas determinará el sentido de giro del motor. En los motores unipolares, únicamente la mitad de la bobina se energiza. [B21]

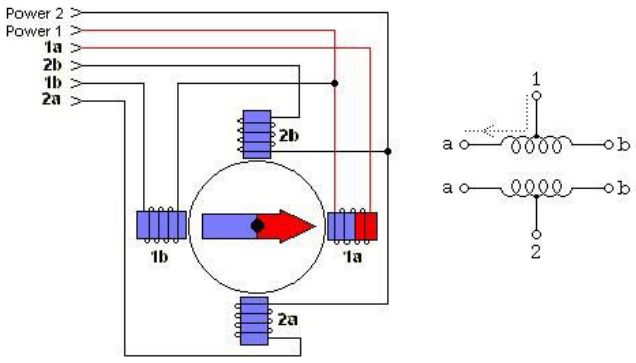


Figura 1.13. Esquema conceptual de un motor PAP unipolar.

- ***Motor Bipolar***

En este caso, las bobinas del motor, carecen de derivación central, es decir, únicamente se exponen cuatro cables al exterior. Al no tener la toma central, se energiza la totalidad de la bobina, por este motivo se consiguen torques un 30% superiores aproximadamente, en comparación con los unipolares. Para lograr este incremento, estos motores requieren unos circuitos de control más complejos, lo que tendrá un impacto importante en el coste de la aplicación. [B21]

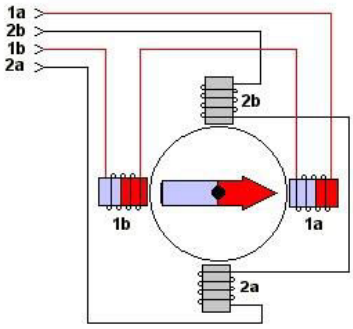


Figura 1.14. Esquema conceptual de un motor PAP bipolar.

Teniendo en cuenta lo redactado en los párrafos anteriores, se realiza una tabla con las principales características de cada uno de los tres tipos de motores paso a paso.

Reluctancia variable	Imán permanente	Híbridos
• Sin par de retención • Alta respuesta dinámica • Bajo par motor • Entre 5° y 15° por paso	• Buen par de retención y sostenimiento • Velocidad de paso baja • Entre 7.5° y 90° por vuelta	• Buen par de retención y sostenimiento • Velocidad de paso alta • Menos de 1.8° por vuelta

Tabla 1.2. Características principales de los motores PAP.

1.4. Tipos de control

El control de los ejes de una máquina herramienta CNC puede clasificarse, dependiendo de las funciones que realizará, en tres grandes grupos: control punto a punto, control paraxial y control continuo. En los siguientes párrafos se describen cada uno de ellos.

1.4.1. Control punto a punto

Es el tipo de control más simple de todos ellos. Este sistema de control causa que la herramienta se mueva de un punto a otro de la pieza, atendiendo a lo establecido por la programación. Solo se mecaniza en ese punto en concreto, la herramienta no actúa de manera permanente. Este tipo de control es el más sencillo de todos, y su uso queda restringido a trabajos de taladrado, troquelado, soldadura por puntos, entre otros. [B2]

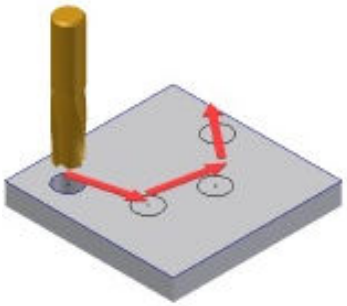


Figura 1.15. Control punto a punto.

Los factores clave en este tipo de controladores, es el tiempo transcurrido para mover el sistema de un punto a otro y la precisión con la que lo logra, sin considerar la trayectoria a seguir y haciendo caso omiso del error cometido durante el movimiento de los ejes.

1.4.2. Control paraxial

En este caso, el recorrido de la herramienta se realiza de manera controlada pero solo paralela u ortogonalmente a los ejes correspondientes. Con este control es posible el mecanizado de contornos, sin embargo, por la ausencia de interpolación, únicamente es posible el control de un motor a la vez, por lo tanto, en caso de realizar líneas diagonales, serán de forma escalonada, lo que genera un error a lo largo de toda la trayectoria. Este normalmente será asumible dependiendo de la precisión mecánica de la MHCNC y del tamaño de la herramienta de mecanizado, una fresa por ejemplo, o hará que este tipo de control quede en desuso para ciertas tareas. [B2]

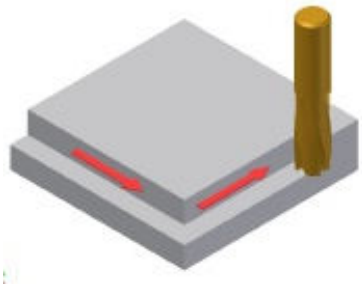


Figura 1.16. Control paraxial.

El control paraxial queda restringido generalmente para el fresado de ranuras, biseles rectos, u otros trabajos similares. El factor más importante, es minimizar el error de seguimiento, siendo este la desviación de la trayectoria de referencia. [B5]

1.4.3. Control continuo

Este control resulta mucho más preciso y conveniente que los controles anteriores, en especial para el mecanizado de contornos curvos, por la capacidad de controlar más de un motor de manera simultánea. De esta forma, la máquina herramienta es capaz de realizar cualquier

recorrido en su área de trabajo, eliminando el escalonado anteriormente mencionado. Esta capacidad de mover dos o más motores de forma simultánea, se denomina interpolación. Por razones obvias, este control es el más completo, y utilizado, y por ende el más complejo. [B2]

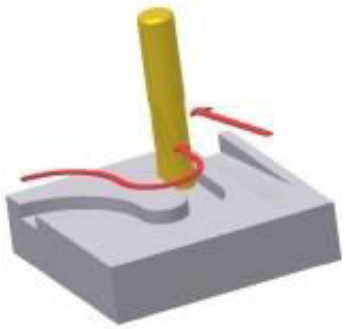


Figura 1.17. Control continuo.

Este tipo de tecnología se emplea para mecanizados más complejos, en especial para fresado 3D de superficies irregulares, fresado de NURBS[(1)], entre otras. [B5] [B16]

En este punto, se puede dividir el control continuo, en dos grandes subgrupos, dependiendo de la interpolación a realizar en función de las características de la pieza.

❖ Interpolación lineal

Como su propio nombre indica, consiste en realizar un desplazamiento lineal, siguiendo una trayectoria rectilínea. Para lograrlo, se calcula el punto medio entre dos puntos de referencia. Mientras la fresa avanza, se van realizando correcciones de deriva si existiesen, en cualquiera de los ejes. Este tipo de movimiento puede hacerse tanto para dos como para tres ejes. [B2]

❖ Interpolación circular

Esta interpolación consiste en realizar movimientos circulares, en sentido horario o anti-horario, calculando los puntos intermedios entres dos puntos, inicial y final, siguiendo una trayectoria curvilínea. A medida que la fresa avanza, se realizan las correcciones de deriva si las hubiese. [B2]

Existen dos casos para este tipo de interpolaciones dependiendo de los datos conocidos. En el primero de ellos es necesario conocer los puntos inicial, final y el radio del trazado, mediante los cuales se realiza el cálculo para obtener el punto central de la curva. Mientras que en el segundo caso se deben conocer los puntos inicial, final y central.

---

(1) **NURBS:** Del inglés, Non-uniform rational B-splines (B-splines racionales no uniformes). Son representaciones matemáticas de geometrías en 3D capaces de describir cualquier forma con precisión. Gracias a su precisión y flexibilidad se pueden utilizar modelos NURBS en cualquier proceso, desde la ilustración hasta la fabricación.[B16] [B6]

Capítulo 2 . Diseño

En los siguientes párrafos se describen las especificaciones del proyecto a un nivel general, entrar en detalle en el desarrollo. Para ello, se describen los elementos que se utilizarán y los objetivos de diseño en cada caso. Para poder establecer un orden claro, se dividirá el proceso de diseño en tres grandes grupos. Mecánica, electrónica y firmware.

2.1. Mecánica

Se adquiere una fresadora de tres ejes XYZ comercial completa **(véase Figura 2.1)**, con la intención de aprovechar la mecánica de esta para la creación del proyecto. Se trata de una mesa de estructura de aluminio de 26cm de largo, 24cm de ancho y 22cm de altura. La superficie de trabajo es de 17cm de largo, 10cm de ancho y 5cm de altura. Dada la reducida superficie, esta fresadora está pensada mara pequeños mecanizados o grabados.

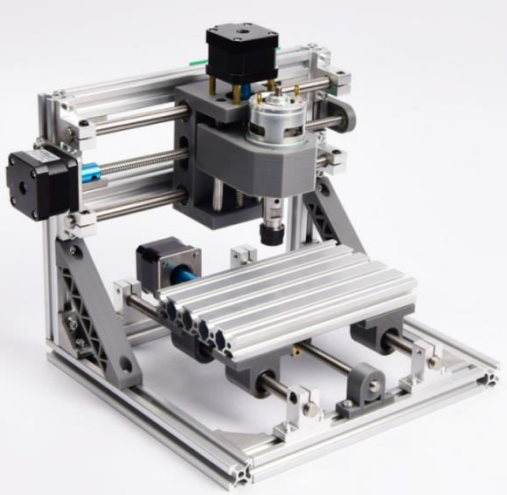


Figura 2.1. Fresadora CNC de tres ejes

La trasmisión del movimiento se realiza mediante raíles con dos guías solidas de 10mm de diámetro y varilla roscada de 1mm de paso y 8mm de diámetro, las cuales se anclan a los rotores de cada motor, gracias a unos acoples cilíndricos sujetos con la ayuda de dos tornillos.

Los motores, con Nº de referencia 17HS1352-P4130, son motores paso a paso de tipo híbridos de dos fases, bipolares de apilado simple. **(Véase apartado 1.3.3)**

2.2. Electrónica

La electrónica que acompaña a la mecánica, es una pequeña placa basada en el micro-controlador ATMEGA328/P del fabricante Microchip y tres drivers basados en el chip A4988 del fabricante Allegro MicroSystems, que sirven para controlar el paso y dirección de los motores PAP. [B30]

El objetivo en este punto es sustituir la placa principal por un Arduino Uno y realizar mediante software CAD el diseño de una PCB que pueda albergar dichos drivers. A esta PCB se le añadirán además una serie de componentes que no están presentes en la versión comercial. Un botón de paro de emergencia, dos conectores para agregar un pequeño teclado junto a un LCD de 16X2 caracteres y dos interruptores finales de carrera, uno para el eje X y otro para eje Y.

❖ Placa para los drivers

Esta placa se compone de la red de interconexiones entre el Arduino Uno y los periféricos asociados a este. Este diseño se compondrá principalmente de una serie de conectores macho para poder encajar las PCBs una encima de la otra, a modo de "sándwich". Se le añadirán además dos conectores para conectar la placa de interfaz de usuario y los finales de carrera.

❖ Placa de interfaz de usuario

La idea de añadir esta interfaz surge a partir de la realización de una serie de pruebas con la fresadora comercial. Esta depende de un software que hay que instalar previamente de un PC, con el se pueden realizar los ajustes previos al mecanizado de un pieza o incluso, manejarla de forma manual, es decir, sin la necesidad de cargar código G. Al añadir esta placa, se pueden realizar dichas tareas sin la necesidad de enviar la información a través de un PC, por lo que permite la utilización de la fresadora sin depender de un software externo y adecuando el programa para la función concreta dentro de un sistema más complejo.

❖ Finales de carrera

Otra de las carencias de la versión comercial, es la ausencia de finales de carrera. Estos se colocan como sistema de seguridad para evitar la colisión de las guías de rodamientos de los brazos de la fresadora con la mecánica de la misma, evitando tanto el desgaste de la parte mecánica como el de la electrónica al evitar movimiento de motor innecesario. Este tipo de soluciones aportan un valor añadido al producto.

A continuación, se muestra un diagrama explicativo de las placas del sistema implementado.

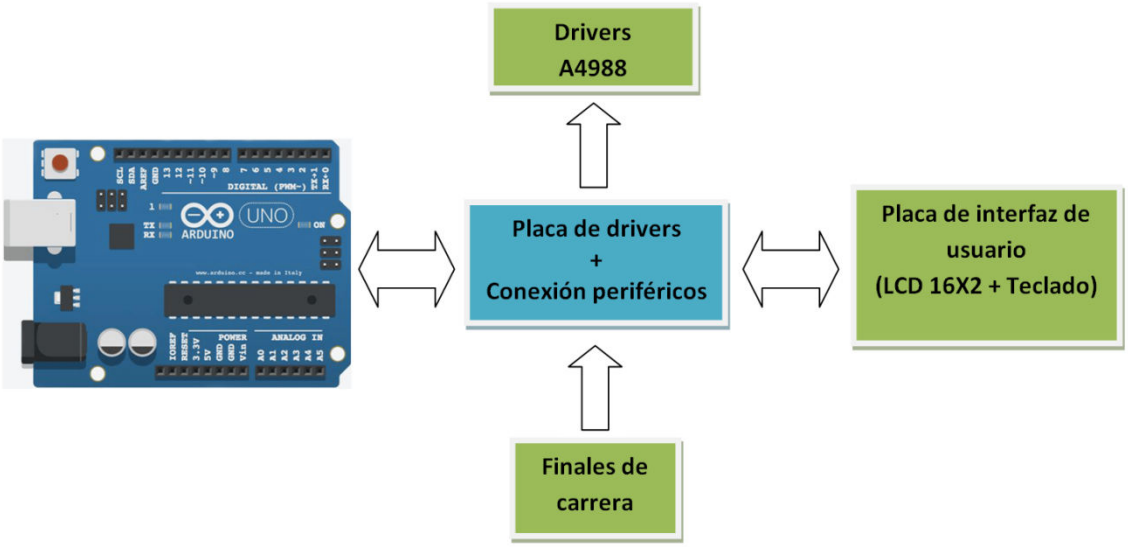


Figura 2.2. Diagrama de montaje de las diferentes placas de la fresadora CNC.

2.3. Firmware

Esta es quizá la parte más compleja y crítica del proyecto. Se trata del programa que controla los movimientos de la fresadora, interpreta las órdenes que recibe de los periféricos y se encarga de representar los datos pertinentes por la pantalla LCD.

Este programa constará de dos partes principales. En la primera de ellas, se albergarán las funciones principales para su ejecución, y en la otra el programa que estará en constante ejecución a la espera de la interacción del usuario.

Las opciones que esta máquina puede realizar, se presentarán al usuario en forma de un pequeño menú explicativo, por el que se puede navegar gracias a los pulsadores de la placa de interfaz de usuario anteriormente mencionada.

Capítulo 3 . Desarrollo

A partir de este punto, se describirá más detalladamente los pasos que se han seguido para la elaboración de este proyecto. Esta sección se dividirá en los siguientes apartados: electrónica, fabricación y firmware.

3.1. Electrónica

En este apartado se explica el funcionamiento de los diferentes circuitos electrónicos que toman parte, tanto los comerciales como las placas diseñadas específicamente para este proyecto. Se adjuntan además una serie de esquemáticos para ayudar a la comprensión del funcionamiento argumentando el por qué de cada uno de las decisiones tomadas.

3.1.1. Arduino Uno

Es una plataforma computacional física open-source basada en una tarjeta de E/S y un entorno de desarrollo que implementa un lenguaje basado en C. Es una de las plataformas hardware más conocidas y versátiles del movimiento “maker”. Presenta una serie de características que lo hacen único para este tipo de trabajos, como la facilidad de conexionado, la cantidad de documentación, tutoriales, proyectos realizados a disposición de los usuarios y lo más importante, un lenguaje de programación sencillo e intuitivo. [B29]

En la imagen inferior, se muestra la apariencia de esta plataforma señalando sus principales componentes y su patillaje o pin-out. [B29]

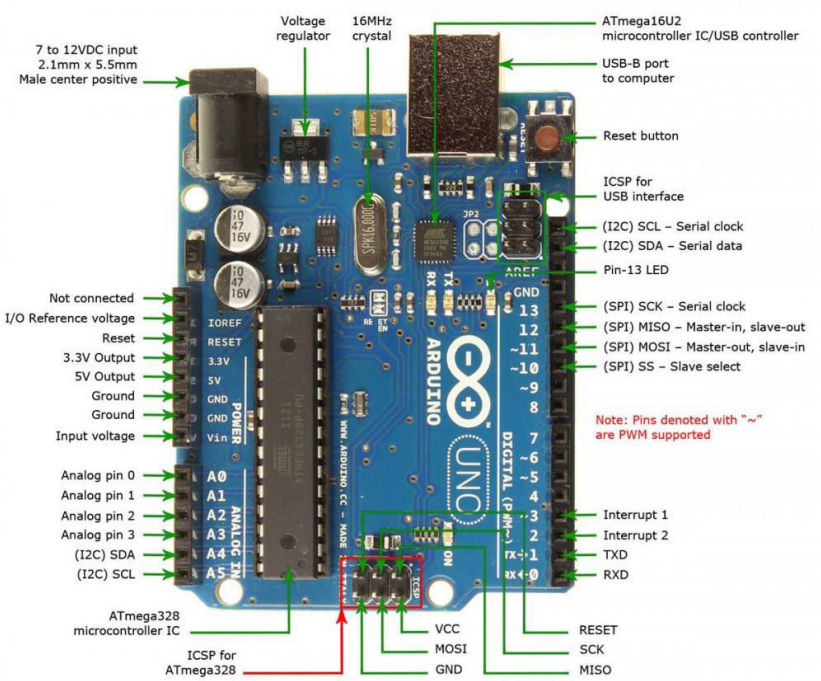


Figura 3.1. Pin-out Arduino Uno.

Existen multitud de modelos basados en los chips de la serie ATmega, pero se hará un pequeño resumen con las características principales del modelo utilizado; en este caso, el Arduino Uno. **(Véase Tabla 3.1)**

Micro-controlador		Atmega328P
Voltaje de operación		5V
Voltaje de entrada	**Conector**	7V – 12V
	USB	5V
E/S digitales		14 (6 PWM)
Entradas analógicas		6
Corriente por E/S		40mA
Memoria Flash		32KB
Memoria SRAM		2KB
Memoria EEPROM		1KB
Frecuencia de reloj		16MHz

Tabla 3.1. Características principales Arduino Uno.

3.1.2. Drivers A4988

Estos drivers son esenciales para el funcionamiento de este proyecto, sin estos u otros similares, sería imposible realizar un control sobre los motores paso a paso. Se trata de unos drivers diseñados para operar con motores paso a paso bipolares en los modos de operación de, paso completo, 1/2 de paso, 1/4 de paso, 1/8 de paso y 1/16 de paso. Es decir, con la simple configuración de una serie de pines **(véase Tabla 3.3)** el avance del motores se verá reducido en dichas proporciones, con esto se conseguirá un aumento sustancial en la precisión del movimiento de la herramienta. A continuación, se adjunta una imagen del driver con su pin-out y una tabla donde se explica la función de cada uno de sus pines. [B30]

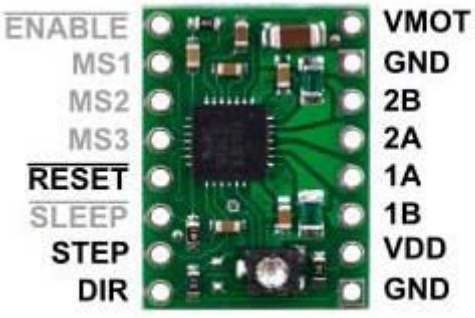


Figura 3.2. Placa de Drivers A4988

PIN	Descripción
$\overline{\text{ENABLE}}$	Activo en nivel bajo, habilita las entradas y salidas
MS1	Pin de selección de micro-paso
MS2	Pin de selección de micro-paso
MS3	Pin de selección de micro-paso
$\overline{\text{RESET}}$	Activo en nivel bajo, pone el chip en modo reset.
$\overline{\text{SLEEP}}$	Activo en nivel bajo, habilita el modo "SLEEP" de bajo consumo.
STEP	Entrada de la señal de STEP. Se activa mediante flaco ascendente
DIR	Selección la dirección de giro del motor
VMOT	Voltaje de alimentación del motor
GND	Tierra
2B	Conexión con la fase 2 del motor
2A	Conexión con la fase 2 del motor
1A	Conexión con la fase 1 del motor
1B	Conexión con la fase 1 del motor
VDD	Voltaje lógico

Tabla 3.2. Función de los pines del Driver a 4988

La siguiente tabla muestra la configuración de los pines MS1, MS2 y MS3 para utilizar los diferentes modos de micro-pasos, donde L representa un nivel bajo de tensión (un cero lógico) y H un nivel alto de tensión (un uno lógico). [B30]

MS1	MS2	MS3	Resolución de micro-paso
L	L	L	Paso completo
H	L	L	1/2 de paso
L	H	L	1/4 de paso
H	H	L	1/8 de paso
H	H	H	1/16 de paso

Tabla 3.3. Configuración de pines para los diferentes modos de micro-pasos

Como se ha comentado, se requiere de tres unidades de estos drivers, una para cada motor que controla el movimiento de los ejes XYZ. Al igual que el Arduino, estas placas disponen de unos conectores que hacen posible que se puedan conectar directamente encima de una placa que los pueda albergar. Estos drivers están basados en el chip A4988 del fabricante Allegro, en cuya hoja de especificaciones se pueden encontrar las siguientes características.

Símbolo	Características	Valor	Unidades
V_{MOT}	Voltaje de operación	8 - 35	V
I_{OUT}	Corriente continua por fase	1	A
	Corriente máxima por fase[(2)]	2	A
V_{DD}	Voltaje lógico de alimentación	-0.3 – 5.5	V
V_{IN}	Voltaje lógico de entrada	-0.3 – 5.5	V

Tabla 3.4. Características de los Drivers A4988.

[(2)] Con disipador térmico o refrigeración por aire. La restricción de corriente no se muestra en hoja de especificaciones del fabricante. Es decir, el chip soporta los 2A de corriente, pero la placa donde está montado no es capaz de disipar tanto calor por sí misma. Por ese motivo requiere de refrigeración adicional.

3.1.3. Placa de Drivers

Como se ha comentado anteriormente, esta placa se encarga de albergar los drivers que controlan los motores paso a paso y de conectar el Arduino Uno con los periféricos asociados (LCD, teclado y talado).

❖ Esquemático

El diseño consta de tres drivers A4988, uno para cada motor. Las conexiones de estos se realizan de acuerdo a los establecido en la hoja del fabricante **(véase Figura 3.4.).** En este se especifica la colocación de unos condensadores de 100µF, para filtrar posibles picos de tensión en la alimentación. La placa donde está montado el driver A4988, ya dispone de estos condensadores al lado del pin V_{MOT} **(véase Figura 3.2),** además se añaden otros en el diseño realizado, como segunda medida de seguridad. Se coloca también, una tira de cuatro pines macho con un espaciado de 2.54mm que harán de conectores para los motores.

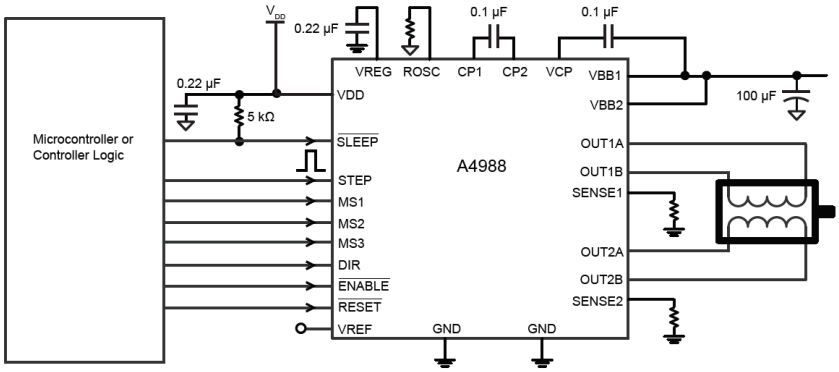


Figura 3.3. Típico diagrama de aplicación.

Sin olvidar que este diseño se trata de un prototipo, se colocan en paralelo a los pines "STEP" unos LEDs que servirán de ayuda a la hora de escribir el código del programa principal, de esta forma, a modo de debug, se identificará de un simple vistazo cuáles de los motores están recibiendo señal.

Además de las conexiones para los drivers, se han incluido una serie de pines macho para conectar directamente la placa de Arduino. Estas conexiones, son las entradas y salidas del propio micro-controlador, que se conectan a su vez al resto de placas.

Una de las carencias de la placa comercial era la ausencia de un botón de paro emergencia o similar, de esto se encarga el componte SW1. Se trata de interruptor de codillo de vástago metálico que soporta una corriente de hasta 6A, más que suficiente teniendo en cuenta las características expuestas en la **Tabla 3.4.** Mediante este switch se podrá interrumpir el paso de la corriente principal, en caso de un fallo en la ejecución del programa.

Por último, se colocan unos conectores para conectan a los periféricos mencionados: la placa de interfaz (LCD y teclado de pulsadores), los interruptores finales de carrera y el taladro. En caso de querer profundizar, el esquemático se encuentra en el **Anexo A**

❖ Rutado

Una vez realizado el diseño del esquemático, se procede con el rutado de la PCB. Antes de realizar ninguna conexión es imprescindible la creación de las huellas o footprints[3] de los componentes **(véase Figura 3.4 y Figura 3.5).** Para realizar este proceso de una manera cómoda y ordenada, se realiza una librería con todos los componentes del proyecto y sus footprints correspondientes, donde se podrán ir añadiendo más a medida que se vallan necesitando.

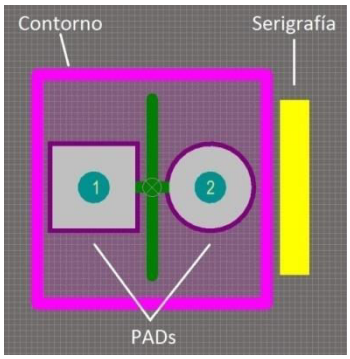


Figura 3.4. Footprint de un condensador electrolítico.

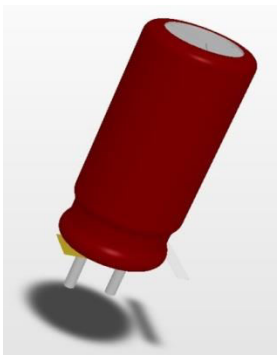


Figura 3.5. 3D de un condensador electrolítico.

Una vez terminado este trabajo, es cuestión de importar los componentes del diseño esquemático al archivo PCB, realizar el posicionamiento de estos, de tal forma que cumplan una serie de requisitos (posibilidad de encajar las diferentes placas entre sí, respetar la orientación de los conectores, evitar que los componentes tengan colisiones, etc.), y por último rutar las diferentes pistas del circuito. Las normas de rutado que se han seguido, se enumeran en la siguiente lista.

- Dejar en la medida de lo posible, las líneas que salen de los componentes que se desean soldar, por la capa inferior.
- Realizar los cambios de dirección de las pistas en ángulos de 45° como máximo.
- Colocar un plano completo de GND para favorecer el retorno de corriente hasta la fuente.
- Dejar una separación de 0.5 mm entre el plano de masa, el borde de la PCB y las pistas.
- Grosor mínimo de pista de 0.5mm.
- Grosor de pista de alimentación principal (V_{CC}) de 1.5mm.

[3] Del inglés huella. Se refiere al patrón de impresión de un componente en una placa de circuito impreso. Se compone principalmente de los PADs, dimensiones del contorno, serigrafía e incluso un 3D donde se podrá ver la apariencia del componente.

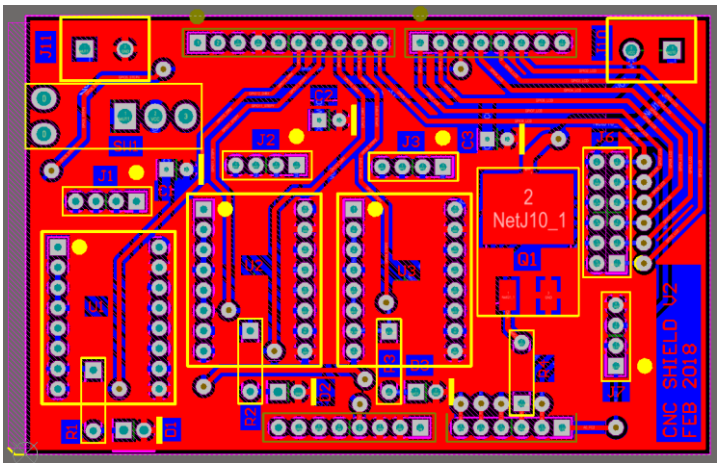


Figura 3.6. Rutado de placa de drivers.

Gracias a los avances de los programas de diseño, se pueden obtener imágenes en 3D del resultado final del circuito sin la necesidad de realizar prototipos. Este hecho ayuda a realizar los análisis de interferencias con la mecánica, para evitar colisiones entre esta y la PCB o los elementos que la conforman. A continuación, se muestran dos imágenes de la placa de drivers en 3D para tener una idea de cómo quedará una vez terminada y soldados sus componentes.

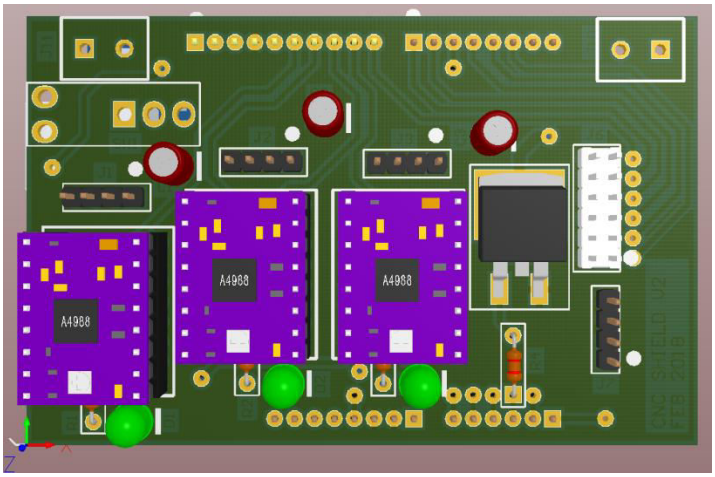


Figura 3.7. Imagen 3D de la placa de drivers.

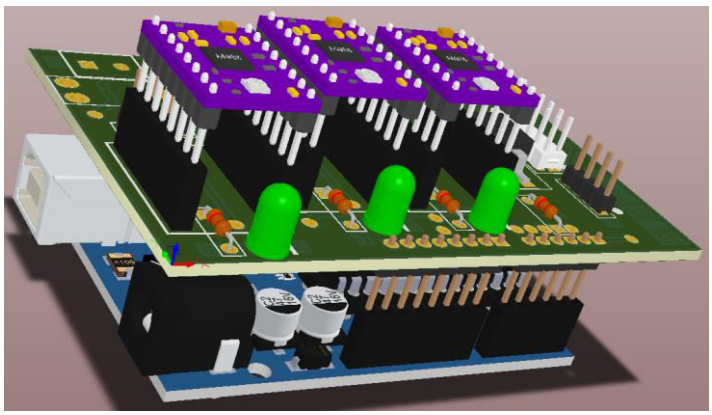


Figura 3.8. Imagen 3D de la placa de drivers, sobre Arduino Uno.

3.1.4. Placa de interfaz de usuario

El interfaz de usuario es el encargado de albergar el teclado, compuesto de pulsadores, y el LCD de 16X2 caracteres. Esta placa se conecta a la anterior mediante un conector de 12 pines.

❖ Esquemático

La conexión de los pulsadores se realiza mediante divisores de tensión a una entrada analógica, con la intención de ahorrar pines de entrada y salida digitales. De esta forma, se consiguen conectar hasta cinco pulsadores a un solo pin. La idea es dividir el rango de tensión de entrada en cinco partes iguales, de forma que al accionar uno de estos pulsadores, llegue a la entrada analógica un nivel diferente de tensión, y poder reconocer así cuál de ellos es el que se está pulsando. A continuación, se exponen los cálculos de las resistencias que se han de colocar para obtener dichos niveles de voltaje de entrada.

$$\frac{V_{CC}}{N+1} = \frac{5V}{6} \approx 0.83 \qquad (3.1)$$

V_{CC}: Voltaje de alimentación.
N: Número de pulsadores

Teniendo en cuenta estos rangos de voltaje, la ecuación de un divisor de tensión (2) y fijando la resistencia R_{BOT}, en 3.3kΩ, por ejemplo, se obtienen los resultados que aparecen en la **Tabla 3.5**. Para el valor de V_{OUT} se tomará el valor medio en cada rango. Es decir, en el primero de los casos para un rango de entre 0.83V y 1.66V se tomará el valor 1.245V. En caso de no existir un valor comercial de la resistencia obtenida, se sustituirá por un valor cercano.

$$Vout = V_{CC}\frac{R_{TOP}}{R_{TOP}+R_{BOT}} \rightarrow R_{TOP} = R_{BOT}\left(\frac{V_{CC}-V_{OUT}}{V_{OUT}}\right) \qquad (3.2)$$

Resistencia	Valor [kΩ]	Valor comercial [kΩ]
R1	9.95	10
R2	4.65	4.7
R3	2.38	2.2
R4	1.11	1.0
R5	0.31	0.33

Tabla 3.5. Valores de las resistencias de los divisores de tensión de los pulsadores

R_1, R_2, R_3, R_4, R_5, representa la resistencia R_{TOP} para cada pulsador y la R_{10} corresponde a la resistencia R_{BOT} de la ecuación **(3.2).**

En paralelo al pin de entrada analógico se añade un condensador cerámico con el fin de filtrar los picos de tensión que se generan al accionar los pulsadores, de este modo, se evitará que el micro-controlador pueda obtener como entrada varios cambios de nivel, detectando diferentes pulsaciones. Además, a la hora de realizar la lectura del valor de entrada de estos

pines en el código programado, se agregará un delay de unos pocos milisegundos, como medida de prevención añadida.

En cuanto a la conexión del LCD, se toma como referencia el datasheet del controlador HD44780U del fabricante HITACHI. La conexión de este tipo de pantallas es un bus paralelo de 4-bits y dos pines de control. Uno para tener acceso a los registros del controlador y el segundo para la activación del modo lectura o escritura. Todas estas conexiones se realizan a los pines de E/S del micro-controlador. Por último, posee una conexión reservada para variar la iluminación del fondo de la pantalla y otras dos para la alimentación. [B31]

Por último, esta placa también alberga la conexión de 12 pines con la placa de drivers mencionada en el apartado anterior y unos LEDs que una vez realizado el código de programación, servirán como indicadores del estado de la ejecución del programa.

❖ Rutado

El proceso de diseño, reglas y consideraciones de rutado de la placa de circuito impreso, es el mismo que en el caso anterior, con la excepción de que la PCB no ha de estar posicionada sobre otra, por lo tanto el posicionamiento de los componentes se ha realizado en base a la experiencia de usuario (UX), estableciendo un posicionamiento de cursores para el control manual y el desplazamiento entre menús.

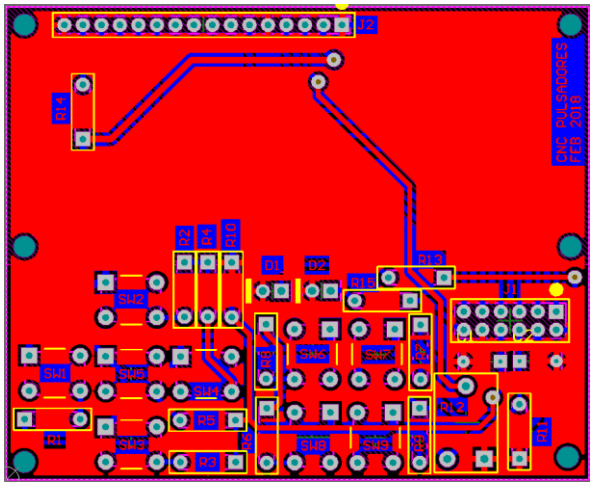


Figura 3.9. Rutado de la PCB interfaz de usuario.

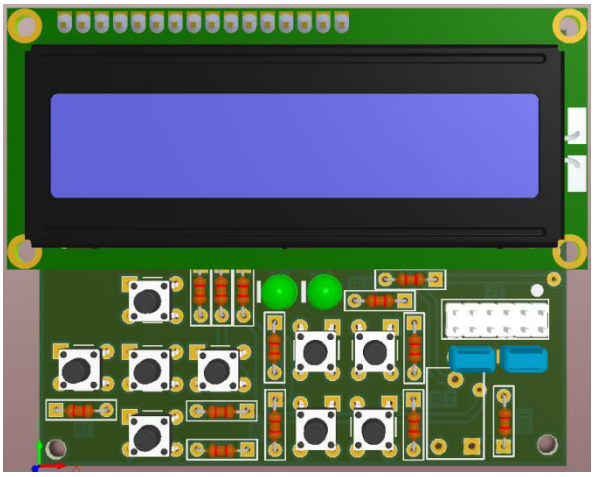


Figura 3.10. 3D de la PCB interfaz de usuario.

3.2. Fabricación

Para la realización de las PCBs se ha optado por la técnica de fotolitografía y atacado al acido. Este proceso se basa en la transmisión mediante luz ultra violeta (UV), de un patrón impreso sobre un material transparente o translucido, a una placa de fibra de vidrio con una capa de cobre en ambas caras, recubiertas con resina fotosensible. Esta técnica consta de cinco pasos bien diferenciados.

1. **Creación de los fotolitos:**

En este caso se ha usado papel de cebolla, que es un material translúcido, para la impresión de los fotolitos. El software Altium dispone de un panel para dividir la PCB en capas e imprimir únicamente las que se vayan a utilizar, en este caso la capa superior (TOP) e inferior (BOTTOM).

Una vez impresos, se adhieren entre sí dejando una de las caras libres, por donde se introducirá la placa virgen a modo de sándwich. Este paso debe ser rápido puesto que la resina comienza a reaccionar con la luz ambiente y podría estropear el trabajo.

2. **Insolación:**

En este segundo paso, se introducen las placas de cobre con los fotolitos colocados en la insoladora. Esta máquina tipo caja, se compone de unas lámparas de luz UV en ambas caras, un temporizador y un sistema que realiza un pequeño vacio que ayuda en el posicionamiento del fotolito. **(Véase Figura 3.11)**

Figura 3.11. Insoladora.

Trascurrido el tiempo de insolación, cinco minutos aproximadamente, la resina que recubre la capa de cobre que ha sido expuesta a la luz UV, es decir, la parte que el fotolito no estaba cubriendo, ha reaccionado químicamente y esta lista para el siguiente paso.

3. Revelado:

El proceso de revelado se lleva a cabo sumergiendo la placa ya expuesta en una solución de sosa caustica y agua. El revelador elimina la resina expuesta dejando únicamente el cobre al descubierto. Este paso dura aproximadamente 30 segundos, transcurridos los cuales se deberá aclarar la placa con abundante agua, antes de proceder con el siguiente paso. La **Figura 3.12** muestra el aspecto de una placa de cobre después del proceso de revelado.

4. Atacado:

El objetivo del atacado es eliminar todo el cobre expuesto, no protegido por la resina. Esta reacción química se consigue mediante una solución a partes iguales de: Agua, Acido clorhídrico estabilizado y agua oxigenada.

Es imprescindible la realización de este proceso y el anterior, en un entorno bien ventilado y hacer uso de guantes, mascarilla y gafas, respetando así unas mínimas protecciones de prevención de riesgos laborales. Una vez preparada la mezcla, se introduce la placa de cobre en la cubeta ejerciendo cierta presión sobre ella para que no salga a la superficie. Trascurridos 60 segundos escasos, la placa esta lista para realizar el segundo aclarado. El resultado final se muestra en la **Figura 3.13**.

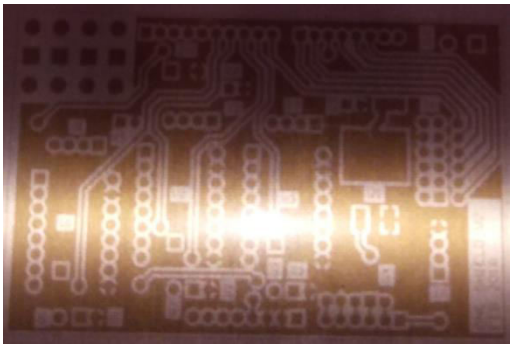


Figura 3.12. Placa después del proceso de revelado

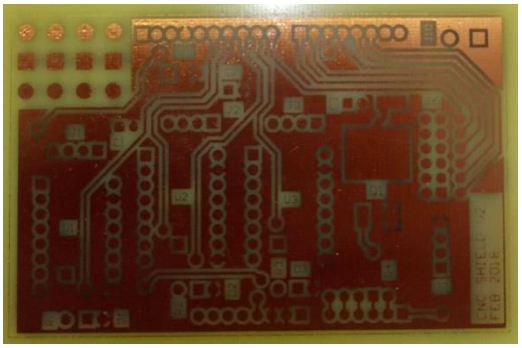


Figura 3.13. Placa después del proceso de atacado.

5. Taladrado:

Terminados los cuatro pasos anteriores la PCB está terminada, únicamente precisa del taladrado de los PADs y la soldadura de los componentes. Antes de realizar los taladros, es conveniente marcar el centro del PAD con un punzón o similar, de este modo se evitará que la broca tome cierta deriva y pueda llegar a partirse. Un punto a tener en cuenta en la soldadura, es empezar por los componentes que tenga un perfil menor, para ayudar a un correcto posicionamiento del resto de componentes.

A continuación se adjuntado las imágenes de las placas una vez taladradas y soldados sus componentes.

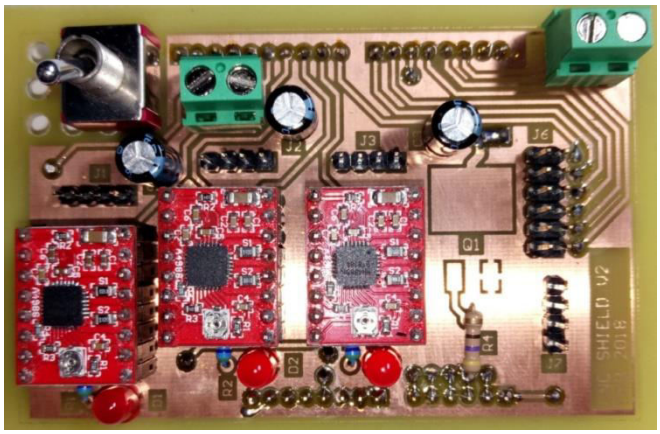


Figura 3.14. Resultado final de la placa de drivers.

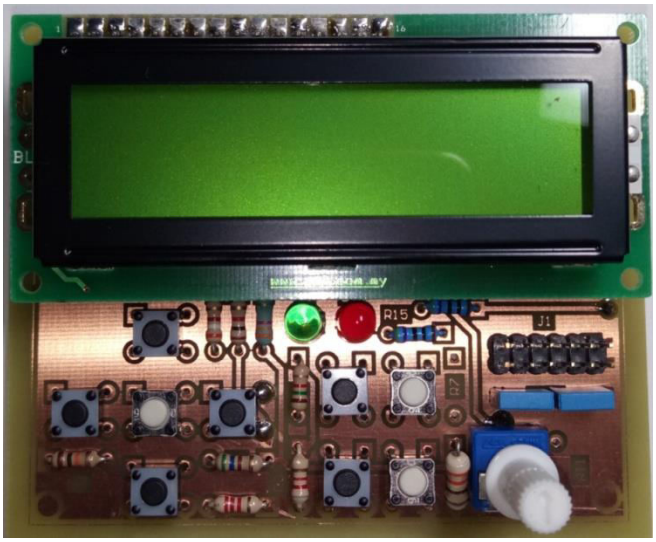


Figura 3.15. Resultado final de la placa de interfaz de usuario.

3.3. Firmware

En este apartado se explica todo el proceso de la desarrollo del código que controla la fresadora CNC. De cara a ofrecer una lectura amena y que pueda entenderse sin grandes conocimientos de firmware, se irá exponiendo el desarrollo junto con diagramas y explicaciones. El código puede dividirse en tres grandes grupos: **Funciones de control, Funciones de movimiento** y **funciones de menú.**

3.3.1. Funciones de control

Antes de realizar ninguna acción es recomendable establecer un origen de coordenadas mediante la opción del menú correspondiente, en caso contrario, se tomará como origen el punto donde se encuentre la herramienta al encenderla.

Para comprender mejor las siguientes explicaciones, se ha de saber de qué forma se introducen los datos de la pieza que se desea realizar. Estos se agrupan en una lista de puntos

clave, con unidades en milímetros. Es decir, por cada cambio de segmento de la pieza, se deberá introducir un nuevo punto. A continuación, se adjunta un fragmento de código como ejemplo, en el que se realiza un cuadrado de 10mm de lado.

```
PuntoFin[0] = 10; PuntoFin[1] = 0;
MOVIMIENTO_LINEAL(PuntoIni, PuntoFin);
PuntoFin[0] = 10; PuntoFin[1] = -10;
MOVIMIENTO_LINEAL(PuntoIni, PuntoFin);
PuntoFin[0] = 0; PuntoFin[1] = -10;
MOVIMIENTO_LINEAL(PuntoIni, PuntoFin);
PuntoFin[0] = 0; PuntoFin[1] = 0;
MOVIMIENTO_LINEAL(PuntoIni, PuntoFin);
```

Figura 3.16. Código de ejemplo de un cuadrado de 10mm de lado.

Como se ha comentado, las unidades de los segmentos son en milímetros, pero el código que realiza el control de los movimientos es en pasos. La conversión se realiza mediante una función que calcula cuantos pasos han de moverse los motores para una determinada distancia. Para hacer este cálculo es necesario conocer una serie de datos, como el avance de la varilla roscada por cada vuelta completa, la cantidad de pasos del motor para una vuelta completa y el modo de micro-pasos configurado en los pines del driver. Sabiendo esto, el resultado se reduce al siguiente cálculo

- *Cantidad de pasos por vuelta en cada motor: 200 pasos*
- *Modo de micro-paso configurado: 1/16 de paso*
- *Avance de la varilla roscada por vuelta completa: 4mm*
- *Distancia lineal a desplazar: D*

$$Cantidad\ de\ micropasos = \frac{200 \cdot 16 \cdot D}{4} \qquad (3.3)$$

La función que realiza la conversión se muestra a la derecha. Es una función no accesible por el usuario que retorna el número de pasos para una distancian en milímetros dada. Todas las funciones de movimiento realizan llamadas a esta función de forma interna.

```
float CONVERSION_PASOS(float milimetros){
  float pasos = 0;
  pasos = (16*200*milimetros)/4;
  return pasos;
  }
```

Figura 3.17. Función de conversión entre milímetros y pasos.

Otro de los parámetros a tener en cuenta es la dirección de giro de los motores. Esta se reconoce mediante el signo del valor introducido por el usuario. Existen dos funciones idénticas, una para determinar la dirección de giro de motor X y otra para la dirección del motor Y y Z.

```
void DIRECCION_X(float x){
  if (x < 0){
    digitalWrite(DirX, HIGH);
  }
  if (x >= 0){
    digitalWrite(DirX, LOW);
  }
}
```

Figura 3.18. Función de elección de dirección.

3.3.2. Funciones de movimiento

❖ Realización de rectas

En cuanto a las rectas, se pueden dividir en tres tipos: paralelas al eje OX, paralelas al eje OY y diagonales.

En primer lugar, la función de realización de rectas (**véase Figura 3.18**), determina la dirección en la que se deberán mover los motores y cambia el valor lógico de los pines de dirección en consecuencia. Para identificar el tipo de recta, la función se divide en tres sentencias condicionales. El primero de los casos, para una recta paralela al eje OY, se utiliza la función "tone()", la cual genera una señal con una frecuencia a elección del programador. Sabiendo el número de pasos y la frecuencia, es posible determinar el tiempo que esta señal debe estar activa para alcanzar la distancia requerida. El segundo de los casos, corresponde a las rectas paralelas al eje OX. En este caso, la señal se genera mediante la función "analogWrite()", la cual genera una señal de frecuencia fija y ciclo de trabajo programable.

El motivo de utilizar dos funciones diferentes queda patente después de la expiación de las rectas diagonales. La idea es controlar los motores X e Y de forma simultánea, para ello se debe conocer la proporción (ratio) de movimiento de uno con respecto al otro. Es decir, para una recta de pendiente 0.5, el movimiento del motor X será el doble que el del motor Y. Para conseguir esto, se utiliza una función de frecuencia fija para el motor X (analogWrite()) y una con frecuencia variable para el motor Y (tone()). Siguiendo con el ejemplo anterior, la frecuencia de Y deberá ser la mitad que la de X. El motivo de no utilizar ambas señales con frecuencia programable, es que Arduino no permite generar dos señales del tipo tone() por dos pines diferentes de manera simultánea.

```
//MOOVIMIENTO LINEAL DIAGONAL XY.....................
void MOVIMIENTO_LINEAL(float Inicio[2],float Final[2]){
  float ratio = 1;
  float freqX = 976.5625;
  float freqY = 976.5625;
  float tX = 0;
  float tY = 0;
  float t_activo = 0;
  float x = 0;
  float y = 0;
  float pasosX = 0;
  float pasosY = 0;

  x = Final[0] - Inicio[0]; //mm
  y = Final[1] - Inicio[1]; //mm

  DIRECCION_X(x);
  DIRECCION_Y(y);
  pasosX = CONVERSION_PASOS(x); //pasos
  pasosY = CONVERSION_PASOS(y); //pasos

  float pasosXX = abs(pasosX);
  float pasosYY = abs(pasosY);
```

```
if (x == 0 && y != 0){ // SOLO SE MUEVE Y
  Serial.println("SOLO SE MUEVE Y");
  tY = 1/freqY;
  t_activo = tY * pasosYY * 1000;
  tone(StepY, freqY);
  delay(t_activo);
  noTone(StepY);
}
if (y == 0 && x != 0){ // SOLO SE MUEVE X
  Serial.println("SOLO SE MUEVE X");
  tX = 1/freqX;
  t_activo = tX * pasosXX * 1000;
  analogWrite(StepX, 127); // Señal PWM a 50%
  delay(t_activo);
  analogWrite(StepX, 0);
}
if (x != 0 && y != 0){ //SE MUEVEN AMBOS MOTORES
  Serial.println("SE MUEVEN X e Y");
  ratio = pasosXX/pasosYY;
  freqY = freqX/ratio;
  tX = 1/freqX;
  t_activo = tX * pasosXX * 1000;
  analogWrite(StepX, 127); // Señal PWM a 50%
  tone(StepY, freqY);
  delay(t_activo);
  analogWrite(StepX, 0);
  noTone(StepY);
}
PuntoIni[0] = PuntoFin[0];
PuntoIni[1] = PuntoFin[1];
PuntoAct[0] = PuntoAct[0] + pasosX;
PuntoAct[1] = PuntoAct[1] + pasosY;
```

Figura 3.19. Función para la realización de rectas.

Teniendo en cuenta la **Figura 3.16**, para la realización de un segmente rectilíneo únicamente es necesario el parámetro “Punto Final”, mientras que el valor “Punto Inicial” es un parámetro interno, el cual se determina en primera instancia por el punto en el que se encuentra la herramienta. Cuando el programa está en ejecución, el punto final del segmento pasa a ser el punto inicial del siguiente.

❖ Realización de curvas

Para llevar a cabo esta tarea, se hace uso de la función anterior y de la aproximación poligonal. Con este método se aproxima una curva mediante la realización de rectas cuyos puntos de inicio y fin están muy próximos entre sí. Cabe destacar que esta función necesita de un segundo parámetro para poder funcionar, el centro.

Esta función está basada en calcular el ángulo que forma el vector director del punto inicial y del vector director del punto final con respecto al centro de la curva, para determinar el recorrido que debe efectuar la herramienta. Para realizar estos cálculos se hace uso de la siguiente ecuación.

$$cos\Theta = \frac{u \cdot v}{|u||v|} = \frac{|u_1 v_1 + u_2 v_2|}{\sqrt{u_1^2 u_2^2} \cdot \sqrt{v_1^2 v_2^2}} \qquad (3.4)$$

Mediante esta función únicamente se puede obtener el ángulo relativo entre dos vectores. Por lo tanto, será necesario conocer además el cuadrante de la circunferencia donde se encuentren ambos puntos. A continuación, se expone un ejemplo en el que se realizan los cálculos iníciales para efectuar un trazo de 90° en el segundo cuadrante.

$$Punto\ central = (-5,0)$$

$$Punto\ inicial = (0,0) \rightarrow \vec{u} = (5,0)$$

$$Punto\ final = (-5,-5) \rightarrow \vec{v} = (0,-5)$$

$$cos\Theta = \frac{u \cdot v}{|u||v|} = \frac{|5 \cdot 0 + 0 \cdot (-5)|}{\sqrt{u_1^2 u_2^2} \cdot \sqrt{v_1^2 v_2^2}} = 0 \rightarrow \Theta = 90°$$

Como se puede ver, el ángulo obtenido es positivo, pero el punto de destino se encuentra al final del segundo cuadrante, por lo tanto, este ángulo debería de ser de -90° o de 270°. Es decir, dependiendo del cuadrante donde el punto se encuentra habrá que realizar una corrección u otra.

El fragmento de código dedicado a la realización de curvas, está compuesto por tres funciones: una función para calcular el ángulo entre dos vectores (véase **Figura 3.20**), otra para determinar el cuadrante donde se encuentran los puntos (véase **Figura 3.21**), y por último la función que efectúa el cálculo de los puntos de la circunferencia. Esta última realiza llamadas sucesivas dentro de un ciclo "While" que tiene como inicio el ángulo de partida y como final el ángulo de destino (**véase Figura 3.22**). Una vez calculado el ángulo, es simple averiguar las distancias X e Y que deben desplazarse los ejes, mediante trigonometría.

```
float ANGULO(float Punto1[2], float Punto2[0], float Centro[2]){
  float vec1[2] = {0,0};
  float vec2[2] = {0,0};
  float ang_rad = 0;
  float ang_grad = 0;
  vec1[0] = Punto1[0] - Centro[0];
  vec1[1] = Punto1[1] - Centro[1];
  vec2[0] = Punto2[0] - Centro[0];
  vec2[1] = Punto2[1] - Centro[1];
  float numerador = abs(vec1[0]*vec2[0] + vec1[1]*vec2[1]);
  float denominador = sqrt(pow(vec1[0],2)+pow(vec1[1],2)) * sqrt(pow(vec2[0],2)+pow(vec2[1],2));
  float aux = numerador/denominador;
  ang_rad = acos(aux);
  ang_grad = (ang_rad * 180)/pi;
  if(aux == 1 ){
    if(vec1[0] == vec2[0] && vec1[1] == vec2[2]){
      ang_grad = 0;}
    else{
      ang_grad = 180;}
}
  return ang_grad;
}
```

Figura 3.20. Función para calcular el ángulo entre dos vectores.

```
float CUADRANTE(float Punto[2], float Centro[2]){//
  int cuadrante = 0;
  if (Punto[0] >= Centro[0] && Punto[1] > Centro[1]){
    cuadrante = 1;
    }
  if (Punto[0] > Centro[0] && Punto[1] <= Centro[1]){
    cuadrante = 2;
    }
  if (Punto[0] <= Centro[0] && Punto[1] < Centro[1]){
    cuadrante = 3;
    }
  if (Punto[0] < Centro[0] && Punto[1] >= Centro[1]){
    cuadrante = 4;
    }
  return cuadrante;
  }
```

Figura 3.21. Función para determinar el cuadrante.

```
void MOVIMIENTO_CURVA(float Inicio[2],float Final[2],float Centro[2])
float x = 0;
float y = 0;
float R = 0;

float ang_rad = 0;
float ang_destino = 0;

x = Centro[0] - Inicio[0]; // Calculo del radio de la curva
y = Centro[1] - Inicio[1]; //
R = sqrt(pow(x,2) + pow(y,2));

float vecOX[2] = {Centro[0]+R,Centro[1]}; //Vector del eje OX
float ang_inic_grad = ANGULO(Inicio,vecOX,Centro);
float ang_relat_grad = ANGULO(Inicio,Final,Centro);

int cuad_ini = CUADRANTE(Inicio, Centro);
int cuad_fin = CUADRANTE(Final, Centro);
```

```
switch (cuad_ini)
  case 1:
  ang_inic_grad = ang_inic_grad;
  break;
  case 2:
  ang_inic_grad = ang_inic_grad * (-1);
  break;
  case 3:
  ang_inic_grad = -180 +ang_inic_grad;
  break;
  case 4:
  ang_inic_grad = ang_inic_grad;
  break;}
float i = ang_inic_grad;
while (i > ang_inic_grad - ang_relat_grad){
  i--;
  ang_rad = (i*pi)/180;
  PuntoFin[0] = Centro[0] + R*cos(ang_rad);
  PuntoFin[1] = Centro[1] + R*sin(ang_rad);
  MOVIMIENTO_LINEAL(PuntoIni, PuntoFin);
  }
```

Figura 3.22. Función para realizar círculos.

❖ Control manual

Otra de las funciones clave es la que controla el movimiento del eje OZ. Esta es sencilla de implementar, únicamente es necesario como parámetro de entrada, una distancia en milímetros. Se determina el número de pasos que se desea desplazar y la dirección de desplazamiento, mediante las funciones de control comentadas en los párrafos anteriores. A partir de estos datos se construye un ciclo "While" con el número de pasos totales, generando una señal cuadrada en el pin STEP del driver que controla el motor Z. De forma análoga, se realizan las funciones para el control manual de los motores X e Y.

```
void MOVIMIENTO_Z(float milimetros){
  float pasosZ = 0;
  int z = 0;
  pasosZ = CONVERSION_PASOS(milimetros);
  DIRECCION_Y(milimetros);
  while (z < abs(pasosZ)){
    z++;
    digitalWrite(StepZ, HIGH);
    delayMicroseconds(t);
    digitalWrite(StepZ, LOW);
    delayMicroseconds(t);
  }
```

Figura 3.23. Función de movimiento del eje OZ.

❖ Retorno

Para finalizar con este apartado, se necesita una función que realice el retorno al origen de coordenadas previamente guardado. Cabe mencionar que al final de cada función de movimiento, se añaden unas variables en las que se almacena la posición actual de cada motor con respecto al origen establecido, de este modo la función "VUELTA_ORIGEN" únicamente tendrá que recorrer ese número de pasos en sentido contrario al valor almacenado.

Como no es necesario seguir el mismo recorrido, primero se realiza el movimiento en la dirección X, después en la dirección Y y por último la de Z.

3.3.3. Funciones de menú

El menú permite al usuario navegar por las diferentes opciones de la máquina CNC. Para poder alternar entre menús o para seleccionar una opción concreta, se utilizan los pulsadores del teclado. Cada uno de estos pulsadores cumple una función específica dependiendo del menú o submenú en el que se encuentre el usuario. La disposición de los pulsadores se expone en la **Figura 3.26.** La siguiente imagen muestra un diagrama de bloques con las opciones que aparecen en el menú.

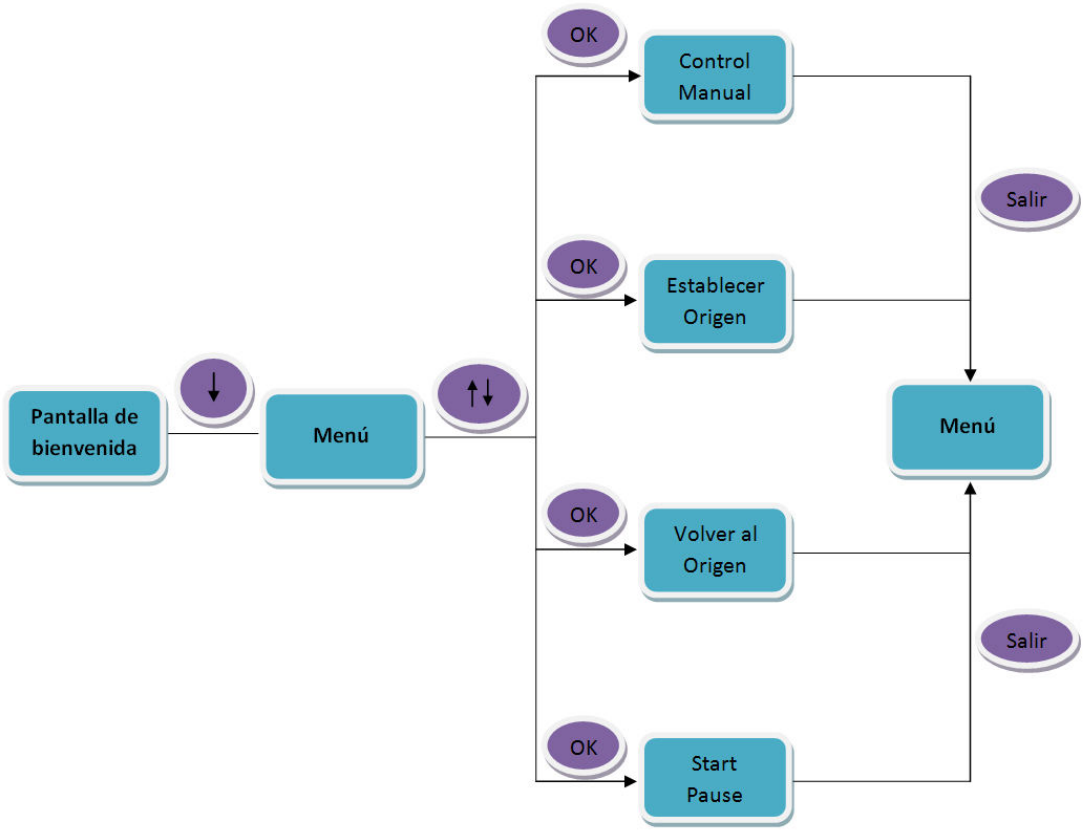


Figura 3.24. Diagrama de bloques del menú

Para manejar el LCD mediante el entorno de desarrollo de Arduino se utilizan las funciones que se recogen en la siguiente tabla.

Función	Descripción
#include <LiquidCrystal.h>	Añade la librería que contiene las funciones de control del LCD.
LiquidCrystal lcd(1,2, 3, 4, 5, 6)	Se inicializan los pines que se utilizan en la conexión del LCD.
lcd.clear()	Deja en blanco el LCD.
lcd.setCursor(Fila, Columna)	Pone el cursor en la posición correspondiente.
lcd.print()	Muestra por pantalla el contenido del paréntesis, sobrescribiendo la fila.
lcd.write()	Escribe el contenido del paréntesis en la posición que marca la función "lcd.setCursor()", sin sobrescribir toda la fila.

Tabla 3.6. Funciones principales para el manejo del LCD.

Siguiendo el diagrama de bloques anterior, al encender la máquina, el LCD mostrará una pantalla de bienvenida, la cual dará acceso a las opciones del menú al pulsar la tecla de dirección inferior (DOWN), **véase Figura 3.26.**

Puesto que el menú principal consta de cuatro opciones y el LCD solamente es capaz de mostrar dos de forma simultánea, se utiliza un contador cíclico para saber sobre que opción se está "apuntando" y mostrar el resultado en consecuencia. Es decir, si el contador tiene un valor igual a tres, el LCD mostrará las dos últimas filas y señalará mediante una flecha el tercer submenú. **(Véase Figura 3.25)**

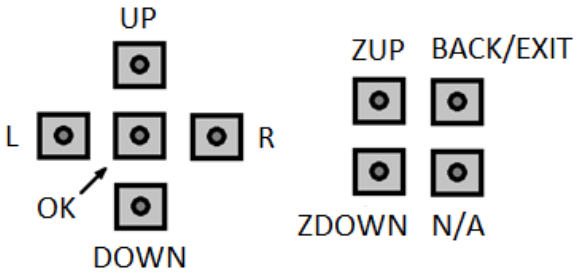
Figura 3.25. LCD con función de menú activa.

La opción de control manual, no requiere una introducción de datos previa al manejo de la máquina, ni de establecer un origen de coordenadas. El control de los motores se efectúa mediante las teclas de dirección "UP", "DOWN", "R", "L", "ZUP" Y "ZDOWN". Para poner en marcha el taladro se utiliza el botón "OK".

Al igual que en el caso anterior, para establecer un origen coordenado, se realiza un movimiento manual de los ejes de la herramienta. Una vez alcanzado el punto deseado, se pulsa "OK" para almacenar en memoria el punto en cuestión.

Para regresar al origen de coordenadas establecido, se accederá al submenú correspondiente a esta opción y se efectuará una pulsación sobre el botón "OK".

Por último, una vez realizados los ajustes previos, será necesario seleccionar la opción Start / Pause. Ambas funciones se realizan con el botón “OK”, dependiendo de si la máquina está en ejecución o en parada.



UP - Dirección superior
DOWN- Dirección inferior
R - Dirección derecha
L - Dirección izquierda
OK - “OK”

ZUP - Movimiento de Z ascendente
ZDOWN- Movimiento de Z descendente
BACK/EXIT- Retroceder o salir
N/A - Sin uso

Figura 3.26. Distribución de los pulsadores en el teclado.

Para ganar un poco de estética, se han creado una serie de caracteres especiales, como por ejemplo lo símbolos de “Start” y “Pause”. Esta es una tarea sencilla que se basa en rellenar cada renglón del LCD con unos (negro) y ceros (blanco), como si de una matriz se tratase. La siguiente imagen muestra la sección del código donde se realiza esta tarea.

```
byte arrowU[8] = {B00000, B00100, B01110, B11111, B01110, B01110, B00000};
byte arrowB[8] = {B00000, B01110, B01110, B11111, B01110, B00100, B00000};
byte arrowL[8] = {B00000, B00100, B01111, B11111, B01111, B00100, B00000};
byte arrowR[8] = {B00000, B00100, B11110, B11111, B11110, B00100, B00000};
byte PLAY[8] = {B10000, B11000, B11100, B11110, B11100, B11000, B10000};
byte PAUSE[8] = {B00000, B11011, B11011, B11011, B11011, B11011, B11011};
```

Figura 3.27. Definición de caracteres especiales.

Para evitar que los botones tomen funciones no deseadas se utiliza un variable “estado” que adquiere un número de 1 al 4 (dependiendo del submenú en el que el usuario se encuentre), de este modo, los pulsadores solo realizarán la acción que tengan asignada en ese “estado”.

Por último, para volver al menú principal se utiliza el botón “Salir”, **véase Figura 3.26**. Mediante esta acción se reinicia la variable “estado” pero se conserva el valor del contador mencionado anteriormente, de esta forma, el menú apuntará a la última opción seleccionada.

Presupuesto

Antes de la realización de este proyecto, se realizó una estimación de horas que habría que dedicar, teniendo en cuenta los diferentes aspectos clave de este trabajo. A lo largo de la realización de trabajo, se han ido recopilando los horas que realmente se han dedicado, como muestra la siguiente tabla, existe una diferencia bastante notable. El aumento de las horas dedicadas está motivado por el hecho de haber realizado dos prototipos en total. En un principio, se decidió realizar el montaje sobre una placa de prototipado perforada, pero la cantidad de conexiones realizadas hacía imposible determinar si los errores producidos se debían a una mala programación o las conexiones en cuestión.

TAREA	HORAS PRESUPUESTADAS	HORAS DEDICADAS	COSTE HORARIO (€/h)	MATERIALES (€)	SUBTOTAL (€)
Especificaciones del sistema	8	10	40	0	400
Compras	12	16	40	250	890
Diseño esquemático	24	30	50	0	1500
Diseño PCB	32	28	50	0	1400
Prototipado	8	10	40	100	500
Firmware	90	110	50	0	5500
Test	40	44	50	0	2200
Documentación	40	48	40	60	1980
TOTAL	254	296		410	14370

Tabla. Horas presupuestadas y horas dedicadas al trabajo de fin de grado.

Dando un valor aproximado en euros a las horas de ingeniería, administración y compras, es posible estimar el coste total del proyecto.

Conclusiones

Como se ha comentado anteriormente este montaje se trata de un prototipo y por ello es fácilmente mejorable una vez vistas las limitaciones que posee. Para realizar un examen ordenado, se irán comentado las líneas futuras que se podrían tomar en cada una de las áreas de conocimiento: mecánica, electrónica y programación, respectivamente.

Una de las limitaciones de las MHCNC de tamaño de escritorio, pero no solo de la utilizada en este proyecto, es que únicamente se encuentran en formato de tres ejes. Existe una solución relativamente sencilla, mecánicamente hablando, para añadir un cuarto eje. Esta se basa en añadir un módulo, junto con su motor PAP correspondiente, directamente en la bancada de la herramienta. Con este accesorio sería posible utilizar esta máquina como un pequeño torno, con la desventaja de la disminución de la superficie de trabajo. Para adoptar esta solución sería necesario añadir un cuarto driver A4988 y modificar gran parte del código implementado.

En cuanto a la electrónica, el modelo de Arduino escogido para este proyecto, es suficiente para las funcionalidades que se querían conseguir, pero si se desea añadir un cuarto eje habría que pensar en pasar a un modelo con mejores prestaciones, como por ejemplo el Arduino Mega. Este modelo presenta una serie de mejoras con respecto a su hermano menor, como el aumento de E/S digitales a 54, entre otras.

El añadir los interruptores de final de carrera a este proyecto se ha realizado como una prueba de concepto para determinar la precisión de la máquina en cuanto a la posición inicial. La precisión testada no fue la idónea, por lo que se decidió controlar dicha posición mediante el guardado de dicha posición “zero” a través de los pasos del motor.

Si se desea profundizar más con estos actuadores se podrían utilizar sensores ópticos infrarrojos. Estos sensores se componen de un diodo emisor IR y un fototransistor. Proporcionan como salida una señal de tensión variable, dependiendo de la luz que el fototransistor recibe. De este modo, no solo se podría detener la herramienta cuando se produce una colisión, sino también avisar cuando la distancia disminuya de un cierto valor, por ejemplo.

Por último, en cuanto a la introducción de los datos de mecanizado de la pieza, se hace de forma manual introduciéndolos en el archivo de programación de Arduino. Es decir, si se desean introducir nuevos datos es necesario reprogramar el dispositivo. Este hecho proporciona un nivel de seguridad mayor que una conexión física o dispositivo extraíble, pero hace más complicada la actualización del mismo. Esta lista de ejecución se podría introducir en un archivo de texto plano y almacenarlo en una tarjeta SD. Posteriormente, habría que modificar el código del controlador para poder interpretar cada una de las líneas de ejecución y actuar en consecuencia.

Bibliografía

[B1] Ángel Andrés López López, Plinnio Roberto Parra Santos, “Diseño de una Fresadora Router CNC”, Escuela Superior Politécnica del Litoral, 2016.

[B2] José Enrique Gaibor Puente, Cristian Jhonny Carrión Paladines, “Diseño e implementación de una máquina CNC para la fabricación de placas de circuito impreso para componentes SMD”, Escuela Superior Politécnica de Chimborazo, 2015.

[B3] Prof. Roberto Martín Murdocca, “Sensores de efecto Hall”, Universidad Nacional de San Luis.

[B4] Miguel Riquelme García, “Diseño y fabricación de una fresadora CNC de 3 ejes para el mecanizado de PCB con plataformas de desarrollo abiertas”, Universidad Politécnica de Cartagena, 2014.

[B5] J. Guillermo Tello Albarrán, Miguel Ramírez-Cadena, Arturo Molina, Roberto Pérez, “Estudio de controladores de movimiento CNC para micro máquinas herramienta”, Convención Científica de Ingeniería y Arquitectura, 2012.

[B6] Les Piegl, “On NURBS”, University of South Florida, 1991.

[B7] Steve Jennings, “MOTORES PASO A PASO”, Disponible en: http://revistas.sena.edu.co/index.php/inf_tec/article/view/899 [Visitado 21 de Junio de 2018].

[B8] José Marton Carbonel, “Smulador de mecanizados en código ISO-6983”, Universidad Politécnica de Valencia, 2011.

[B9] Carlos Canto, “Motores de paso o stepper motors”, Universidad Autónoma de San Luis Potosí.

[B10] Alfonso de Lara Rubio, “Diseño e implementación en FPGA para fresadora de tres ejes”, Universidad Rey Juan Carlos, 2011.

[B11] Juan Alberto Moreno Mangas, Josep Fiol Ramon, “Entrenador digital para convertidores en aplicaciones docentes”, Universitat politécnica de Catalunya, 2011.

[B12] “¿Con o sin escobillas? ¿Qué motor de CC debería elegir?”. Disponible en: https://www.arrow.com/es-mx/research-and-events/articles/which-dc-motor-is-best-for-your-application, [Visitado el 5 de abril de 2018]

[B13] “Así funciona el motor de corriente continua o directa”. Disponible en: http://www.asifunciona.com/electrotecnia/af_motor_cd/af_motor_cd_6.htm [Visitado el 8 de abril de 2018]

[B14] “Motor eléctrico Brushless: Funcionamiento y características”. Disponible en: http://www.cochesrc.com/motor-electrico-brushless-funcionamiento-y-caracteristicas-a3607.html [Visitado el 6 de abril de 2018]

[B15] “Brushless sensored y brushless sensorless”. Disponible en: http://www.cochesrc.com/brushless-sensored-y-brushless-sensorless-a3618.html#axzz2Cwl6VtcV [Visitado el 7 de abril de 2018]

[B16] “NURBS”, Disponible en: http://www.3dcadportal.com/nurbs.html [Visitado el 14 de abril de 2018]

“3 DIC 2014

[B17] “¿Qué es G-Code?”. Disponible en: https://polaridad.es/que-es-g-code/, [Visitado el 22 de marzo de 2018]

[B18] “Códigos para CNC”. Disponible en: http://r-luis.xbot.es/cnc/codes03.html, [Visitado el 21 de marzo de 2018]

[B19] “Plotter Router Fresadora CNC”, Disponible en: http://www.alciro.org/alciro/Plotter-Router-Fresadora-CNC_1/Motores-Paso-a-Paso-Step-Motor_37.htm, [Visitado el 2 de junio de 2018]

[B20] “Motores paso a paso”, Disponible en: http://www.monografias.com/trabajos17/motor-paso-a-paso/motor-paso-a-paso.shtml [Visitado el 15 de junio de 2018]

[B21] “Motores paso a paso unipolares”, Disponible en: https://www.diarioelectronicohoy.com/blog/motores-pap-unipolares [Visitado el 21 de mayo de 2018]

[B22] “Las 6 principales ventajas de utilizar maquinaria CNC”. Disponible en: http://kuzudecoletaje.es/las-6-principales-ventajas-de-utilizar-maquinaria-cnc/ [Visitado el 22 de mayo de 2018]

[B23] “Fused deposition modeling”. Disponible en: http://www.materialise.com/en/manufacturing/3d-printing-technology/fused-deposition-modeling [Visitado el 21 de junio de 2018]

[B24] “What is selective laser Sintering?”. Disponible en: https://www.livescience.com/38862-selective-laser-sintering.html [Visitado el 21 de junio de 2018]

[B25] “Que es CAD/CAM?”. Disponible en: https://www.autodesk.es/solutions/cad-cam, [Visitado el 11 de marzo de 2018]

[B26] “CAM Welcome”. Disponible en: https://help.autodesk.com/view/fusion360/ENU/, [Visitado el 11 de marzo de 2018]

[B27] “LibreCAD Open Source 2D-CAD”. Disponible en: https://librecad.org/, [Visitado el 14 de marzo de 2018]

[B28] “8-bit AVR Microcontrollers ATmega328/P DATASHEET SUMMARY”. Disponible en: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Summary.pdf, [Visitado el 13 de mayo de 2018]

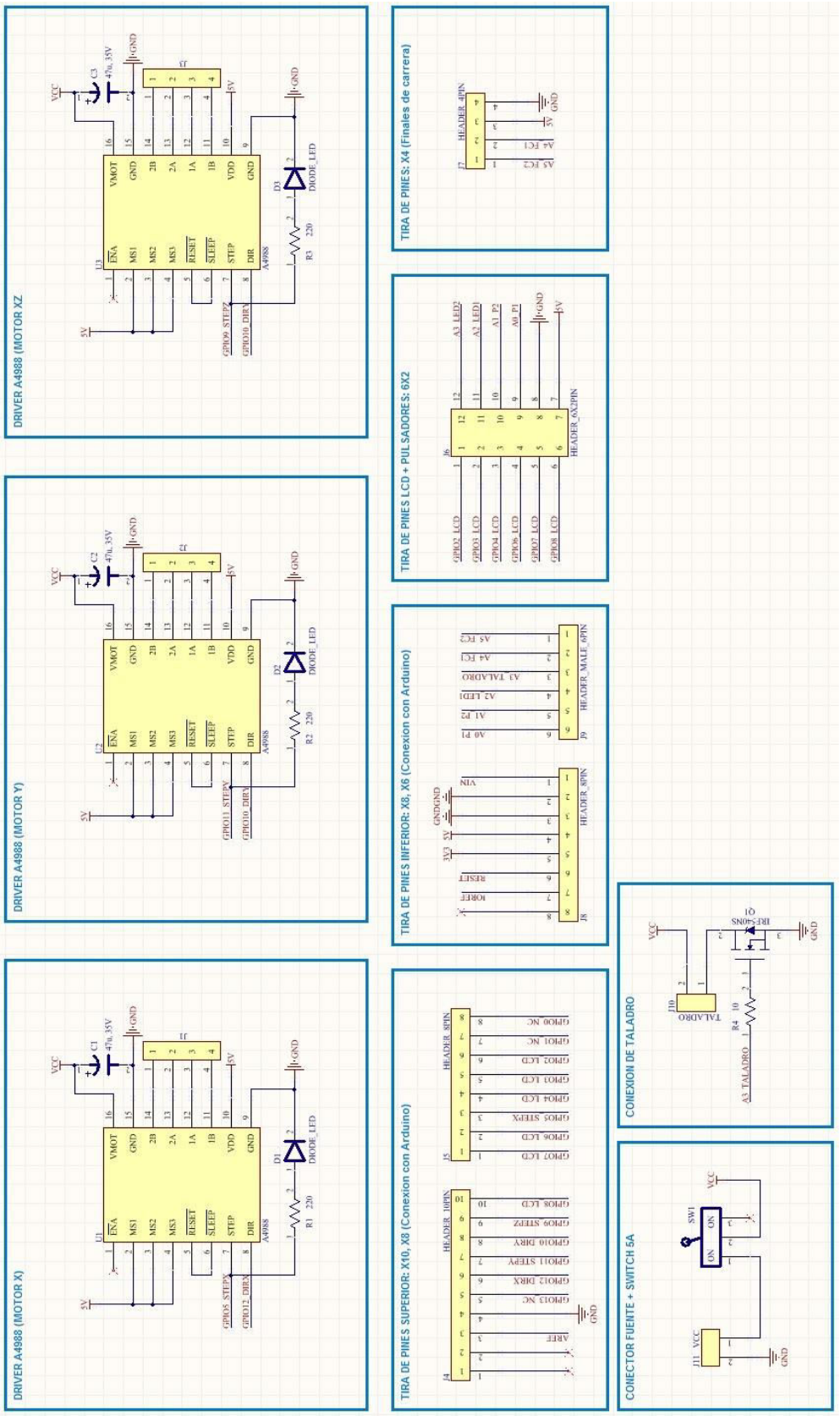
[B29] “Arduino UNO R3”. Disponible en: http://arduino.cl/arduino-uno/, [Visitado el14 de junio de 2018].

[B30] “A4988: DMOS Microstepping Driver with Translator and Overcurrent Protection”. Disponible en: https://www.allegromicro.com/es-ES/Products/Motor-Driver-And-Interface-ICs/Bipolar-Stepper-Motor-Drivers/A4988.aspx [Visitado el 21 de abril de 2018]

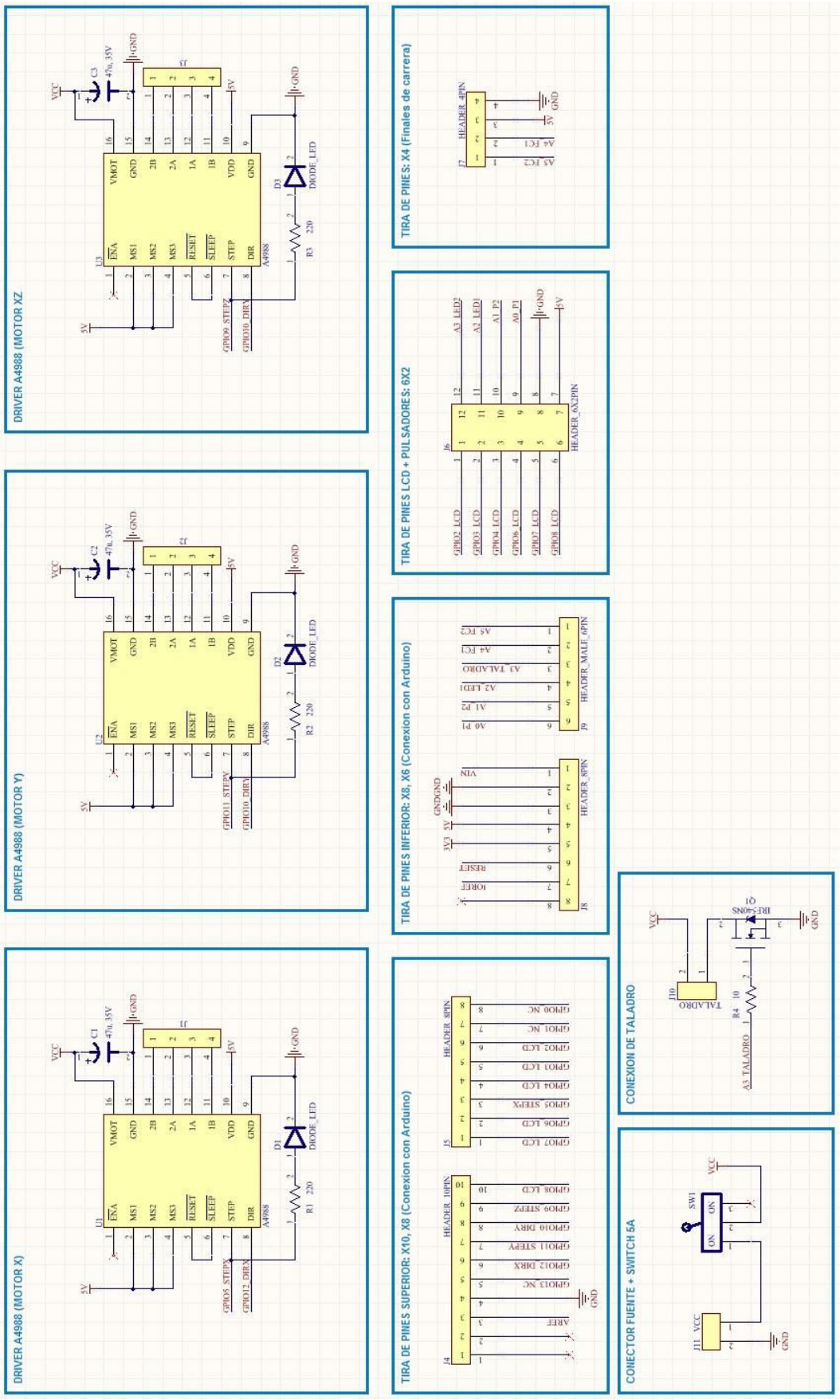
[B31] “Dot Matrix Liquid Crystal Display Controller/Driver”. Disponible en: https://circuitdigest.com/sites/default/files/HD44780U.pdf, [Visitado el 21 de junio de 2018]

Anexos

Anexo A



Anexo B


DRIVER A4988 (MOTOR XZ)
DRIVER A4988 (MOTOR Y)
DRIVER A4988 (MOTOR X)
TIRA DE PINES: X4 (Finales de carrera)
TIRA DE PINES LCD + PULSADORES: 6X2
TIRA DE PINES INFERIOR: X8, X6 (Conexion con Arduino)
TIRA DE PINES SUPERIOR: X10, X8 (Conexion con Arduino)
CONEXION DE TALADRO
CONECTOR FUENTE + SWITCH 5A