UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño e implementación de un sistema CNC de dos ejes para uso docente

MEMORIA PRESENTADA POR:

Víctor Oltra Almiñana

Máster Universitario en Ingeniería Mecatrónica

DIRECTOR:

D. Juan José Serrano Martin

Valencia, Julio 2019

INDICE

INDICE 2
1. FINALIDAD TRABAJO 5
1.1. Objeto de estudio. 5
1.2. Justificación del trabajo. 5
2. INTRODUCCIÓN 6
2.1. Antecedentes. 6
2.2. Motivación. 6
2.3. Objetivos. 7
3. INTRODUCCIÓN A LAS MÁQUINAS CNC 8
3.1. Historia y evolución. 8
3.2. Estado actual. 9
3.3. Ventajas e inconvenientes. 11
3.4. Tipos de maquina CNC. 12
3.4.1. Control. 12
3.4.2. Estructura. 12
3.4.3. Material. 14
3.4.4. Número de ejes. 14
3.4.5. Función. 14
3.5. Implementación de una máquina CNC. 16
3.5.1. Elementos. 17
3.5.2. Control y movimiento de los motores. 21
3.5.3. Transmisiones. 28
4. DISEÑO DEL SISTEMA 31
4.1. Mecánica y estructura. 31
4.1.1. Base. 31
4.1.2. Ejes y transmisiones. 32
4.1.3. Elementos diseñados. 33
4.2. Electrónica. 34
4.2.1. Fuente de alimentación. 34
4.2.2. Microcontrolador. 36
4.2.3. Drivers. 37
4.2.4. Motores PAP. 39
4.2.5. Taladro. 42
4.2.6. Relé. 44
4.2.7. Finales de carrera y circuito anti rebote. 45

4.2.8. Convertidor USB/serie. 46
4.3. Software. 47
4.3.1. SolidWorks. 48
4.3.2. STM32-CubeMX. 48
4.3.3. arm Keil - µVision IDE. 49
4.3.4. Txapu-CNC. 50
5. DESCRIPCIÓN DEL SOFTWARE: DESARROLLO Y PROGRAMACIÓN 53
5.1. Arquitectura del programa. 53
5.2. Configuración STM32Cube. 55
5.2.1. Pines E/S. 55
5.2.2. Temporizadores y generación de PWM. 56
5.2.3. Comunicación UART y SPI. 57
5.2.4. Resumen. 59
5.3. Programación en Keil. 60
5.3.1. Organización proyecto. 60
5.3.2. Configuración periféricos. 61
5.3.3. Configuración motores. 62
5.3.4. Funciones driver. 63
5.3.5. Programa principal. 64
5.3.6. Funciones implementadas. 65
5.3.7. Interrupción final de carrera. 66
5.4. Flujogramas. 67
6. G-CODES EMPLEADOS 71
7. CONVERSIÓN DE MOVIMIENTO 73
7.1. Tipo de movimiento. 73
7.2. Calculo de pasos. 74
8. CONEXIONADO DEL SISTEMA 75
9. PRESUPUESTO 80
9.1. Introducción. 80
9.2. Desarrollo del presupuesto. 80
9.2.1. Materiales. 80
9.2.2. Licencias. 81
9.2.3. Mano de obra. 82
9.3. Presupuesto final. 82
10. MEJORAS DEL SISTEMA 83
11. CONCLUSIÓN 84
12. BIBLIOGRAFÍA 85
13. ANEXOS 86
13.1. ANEXO A: Programación. 86

13.2. ANEXO B: Documentación técnica. 108
13.3. ANEXO C: Plano conexión sistema. 143
13.4. ANEXO D: Planos piezas diseñadas. 144
13.5. ANEXO E: Imágenes del sistema. 150

1. FINALIDAD TRABAJO

1.1. Objeto de estudio.

¿Qué es una máquina de Control Numérico? ¿Cómo funciona? ¿Qué utilidad tiene?
Estas preguntas y muchas más se van a resolver en el siguiente trabajo, que consiste en la construcción, programación y puesta en marcha de una **máquina de control numérico** (CNC) de dos ejes, cuya función implementada es la de taladrar.

Durante el mismo, se va a proceder al análisis de los elementos necesarios para el diseño y control del sistema. Se tendrá en cuenta el listado de materiales, tanto electrónicos como estructurales, para su construcción.

Se va a llevar a cabo un sistema que cumpla eficazmente con los objetivos propuestos y finalmente se va a proceder al análisis de los elementos utilizados incorporando un presupuesto del coste total del sistema.

1.2. Justificación del trabajo.

El trabajo ha sido llevado a cabo en el Instituto Universitario de Tecnologías de la Información y Comunicaciones (I.U.I. ITACA) de la UPV, a partir de la oferta propuesta por D. Juan José Serrano Martín, para el diseño de una maquina CNC que englobe los conocimientos vistos en la asignatura de Sistemas Embebidos cursada en el Máster de Ingeniería Mecatrónica, con el fin de ser **objeto de estudio** para los alumnos que cursen dicha asignatura en los próximos años.
Los alumnos aprenderán acerca del uso de estas máquinas así como su funcionamiento, haciendo hincapié en el código, que hace la función de intérprete de los comandos.

Para ello se va a proceder a la construcción de un sistema nuevo, mediante la remodelación de una maquinas CNC propiedad del laboratorio, utilizando los conocimientos adquiridos durante el máster, de manera que sea lo más adecuada posible para su utilización en las prácticas de la asignatura de Sistemas Embebidos.

2. INTRODUCCIÓN

2.1. Antecedentes.

Desde los inicios de la industria las máquinas de control numérico han desempeñado un papel importante facilitando la labor a las personas.

Ante la necesidad de uso de herramientas más complejas y el afán de conseguir procesos automatizados, cada vez más rápidos y precisos, las maquinas CNC han ido evolucionando junto a la industria. Desde las primeras máquinas controladas por un ordenador central con electrónica de válvulas, relés y cableados, hasta las conocidas actualmente controladas por un microprocesador, cuyo reducido tamaño y baja potencia han hecho posible la utilización de dichas maquinas en formato doméstico, como son las impresoras 3D que hoy en día conocemos.

El mundo tecnológico avanza de forma muy rápida y las maquinas CNC tienen gran parte de culpa.

2.2. Motivación.

La principal motivación a la hora de realizar dicho trabajo es la de abarcar todos los conocimientos adquiridos a lo largo del master en un mismo objetivo.
Partiendo desde la electrónica necesaria para el control y uso de los motores paso a paso y terminando con la mecánica necesaria para el diseño de la estructura del sistema, haciendo del trabajo un buen ejemplo de sistema de ingeniería mecatrónica.

También es motivo de interés el aprendizaje del firmware de control de los motores paso a paso combinado con la interpretación de comandos, ya que es la base de todas las máquinas CNC. A este firmware se le conoce con el nombre de GRBL.

Otra motivación a tener en cuenta, pero no menos importante que la anterior, es la de crear un sistema simplificado, que sirva de apoyo al personal docente, para su uso en el estudio de las maquinas CNC en futuras generaciones de ingenieros mecatrónicos.

2.3. Objetivos.

En este trabajo se va a realizar el diseño y programación de una máquina de control numérico (CNC) de dos ejes, cuya funcionalidad sea de taladradora. La finalidad del trabajo es la del uso docente de la máquina, por tanto, debe tratarse de una maquina con una función simplificada que facilite su comprensión. Los objetivos principales son los siguientes:

- Selección de elementos electrónicos.
- Diseño de la estructura de la máquina.
- Programación del microcontrolador.
- Puesta en marcha y verificación del funcionamiento.

Además, debe de realizarse una valoración económica del gasto que supone la construcción de este sistema CNC y su viabilidad.

3. INTRODUCCIÓN A LAS MÁQUINAS CNC

3.1. Historia y evolución.

El uso de maquinaria para llevar a cabo diversos tipos de tareas se remonta varios siglos atrás, concretamente a la época comprendida entre finales del siglo XVIII y principios del XIX, conocida como revolución industrial, momento de la historia en el que se comenzaron a construir y utilizar diversos tipos de máquinas para los procesos de producción en las industrias.

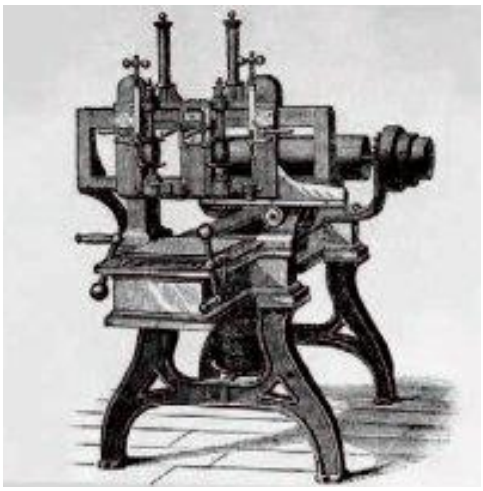


Ilustración 3.1.1 Máquina-Herramienta creada durante la revolución industrial.

Ya en el siglo XX, la industria requiere todo tipo de herramientas, aparatos y equipos que sirvan para agilizar y optimizar los procesos productivos de una empresa. Gracias al avance de la ciencia y la tecnología es posible crear más y mejores equipos que sirvan a estos propósitos, y que al mismo tiempo, cumplan con la demanda de la industria, que requiere herramientas con capacidades y funciones cada vez más específicas para su uso en procesos particulares.

Entre los equipos especializados más utilizados actualmente en algunas industrias están las máquinas CNC, equipos altamente funcionales gracias a las implementaciones tecnológicas con las que cuenta, que los convierten en herramientas ideales para las industrias con grandes volúmenes de producción.

¿Pero a qué se refieren estas siglas? CNC significa Control Numérico por Computadora. Este concepto fue desarrollado y estudiado a fondo de manera más seria y formal en el Instituto Tecnológico de Massachussets o MIT (Massachusetts Institute of Technology) en 1949 y el principal objetivo de estos estudios fue crear una fresadora experimental que tuviera integrado un sistema CNC que la hiciera mucho más precisa.

Finalmente fue construido el aparato, llamado Fresadora Cincinnati Hydrotel con Husillo-Vertical, el equipo que representó el punto de partida para las máquinas CNC actuales. Aunque el éxito en el desarrollo de este equipo no fue inmediato, pues tuvo que ser modificado varias veces, sufriendo cambios de piezas y componentes e incluso modificaciones en su diseño.

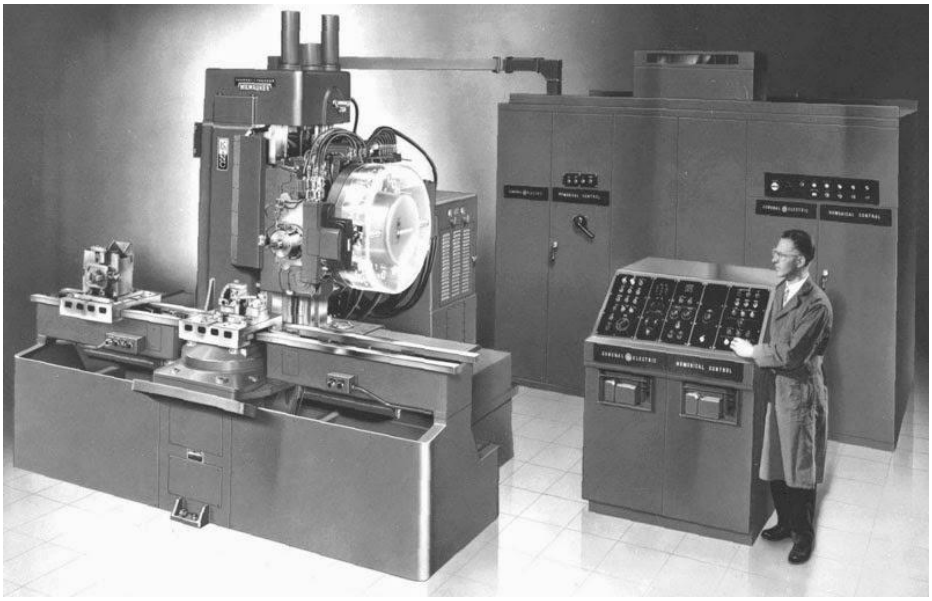


Ilustración 3.1.2 Ejemplo de una de las primeras máquinas CNC.

Como se puede suponer, la primera generación de estos equipos tenía que utilizar controladores de gran tamaño que consumían una gran cantidad de energía y que generaban una gran cantidad de calor. Los modelos de segunda generación tuvieron modificaciones que ayudaron a disminuir el consumo de energía, el tamaño de los equipos y la generación de calor, pero seguían siendo bastante grandes.

De la misma forma en que sucedió con muchos otros tipos de equipos, tecnologías e implementaciones, con el paso del tiempo y el avance tecnológico, el sistema se fue perfeccionando, se fueron agregando funciones, capacidades y características sumamente útiles, hasta conseguir los equipos que se tienen en la actualidad.

3.2. Estado actual.

Hoy en día los controles numéricos ofrecen funcionalidades que ni tan siquiera se podían imaginar en sus inicios, ya sean tornos, centros de mecanizado, máquinas de corte por láser, punzadoras o plegadoras entre otros siguen evolucionando cada día. Los diferentes fabricantes

de Máquinas-Herramienta, pugnan entre ellos por ofrecer a los clientes controles cada vez más modernos e intuitivos para el usuario.

Un tipo de equipos en los que el sistema de Control Numérico por Computadora ha resultado altamente eficiente, mejorado los procesos, optimizado las funciones y agilizado la producción, es en los sistemas de corte mediante el uso de tecnologías como el láser, pues gracias al CNC es posible programar el equipo para trabajar de manera automática o manual y de esta forma ser adaptado tanto a las necesidades de la empresa como a las del usuario de la máquina.

Ilustración 3.2.1 Seccionadora utilizada por empresa maderera.

El equipo industrial es fundamental para la cadena de producción de cualquier empresa, pues las necesidades de la industria actual se basan en la rapidez, la eficiencia y la calidad y para ello se requieren las mejores herramientas, la mejor tecnología y los mejores equipos.

A continuación, se puede observar la pantalla con la que el operador controla los movimientos indicados a la maquina CNC, este lenguaje es llamado “G-code” y más adelante se profundizará en este tema.

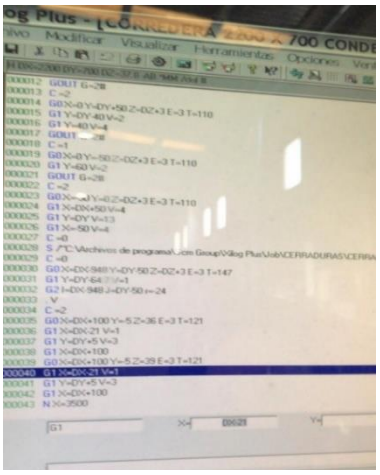


Ilustración 3.2.2 Puesto de control utilizado por el operador de la seccionadora anterior.

Además, gracias al avance tecnológico se han llegado a desarrollar maquinas CNC de carácter doméstico, es decir con unas prestaciones inferiores a las industriales y por tanto una menor potencia. Y es que, con el avance de la tecnología y la reducción de los microcontroladores, motores eléctricos etc… se ha logrado que se pueda incorporar dicha maquinaria en la vida cotidiana de las personas.

El ejemplo más conocido de maquina CNC doméstica es el de impresora 3D, que poco a poco va ganando mayor reconocimiento.

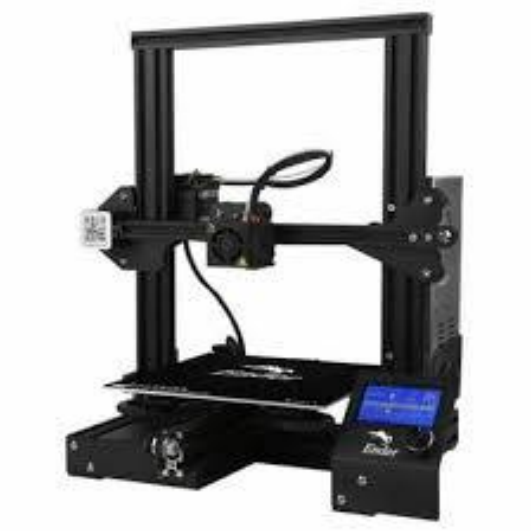


Ilustración 3.2.3 Ejemplo de impresora 3D actual.

3.3. Ventajas e inconvenientes.

La introducción de la maquinaria CNC ha cambiado la vida de las personas y estas son algunas de las ventajas e inconvenientes que han supuesto:

Ventajas:

- Permite una mejor planificación de las operaciones
- Se incrementa la flexibilidad de maquinado
- Reducción de los tiempos de mecanizado
- Mejor control del proceso y tiempos de maquinado
- Disminución en los costos por herramientas
- Se incrementa la seguridad para el usuario
- Reducción del tiempo de flujo de material
- Reducción del manejo de la pieza de trabajo
- Aumento de productividad

- Aumento en precisión
- Menor inventario
- Menor espacio físico

Desventajas:

- Mayor costo de inversión
- Mayor mantención
- Necesidad de entrenamiento del personal

3.4. Tipos de maquina CNC.

Existen muchos tipos de máquinas de CNC, las cuales comparten una misma base, es decir, el control de la maquina a través de una computadora, sin embargo, la finalidad o función de estas máquinas puede ser muy distinta.

A continuación, se muestran algunos ejemplos en los que se pueden agrupar dichas máquinas.

3.4.1. Control.

En función del tipo de control en el que están basadas las maquinas CNC, se pueden encontrar los siguientes grupos:

- **Punto a punto** → se mecanizan los puntos iniciales y finales, pero no la trayectoria. No se tiene control sobre parámetros como son la velocidad o el trazado.

- **Paraxial** → a diferencia de las punto a punto, si se puede programar la velocidad y los desplazamientos a lo largo de la trayectoria. Se debe tener en cuenta que la trayectoria es paralela a los ejes.

- **Interpolar** → ofrece la posibilidad de realizar mecanizados a lo largo de trayectorias de cualquier tipo. Son las más polivalentes en cuanto a mecanizado.

3.4.2. Estructura.

Existen muchos y diversos tipos de estructura para una maquina CNC, con diseños de todos los tamaños y manufacturados con diferentes materiales.

El tipo de estructura de la maquina va ligada a la función para la que ha sido creada. Algunos de los ejemplos de estructura más comunes son:

- **Pórtico de mesa móvil**: el puente en el que se encuentra el cabezal es fijo, mientras que es la mesa la que se desplaza a lo largo del eje X.

Ilustración 3.4.1 Ejemplo de pórtico de mesa móvil.

- **Pórtico de cabezal móvil**: el punto de apoyo o mesa en este caso es fijo y el puente donde se encentra el cabezal se desplaza en varias direcciones.

Ilustración 3.4.2 Ejemplo de pórtico de cabezal móvil.

- **Bancada multieje**: las máquinas multieje son capaces de realizar rotaciones alrededor de uno o de múltiples ejes

Ilustración 3.4.3 Ejemplo de bancada multi-eje.

3.4.3. Material.

Si se agrupa en función del producto a tratar, se puede agrupar en varios tipos de materiales, algunos de ellos son:

- **Madera** → son máquinas utilizadas en el mundo de la carpintería para el corte, limado o grabado de todo tipo de madera.

- **Acero** → suelen ser maquinas más rápidas y efectivas que las convencionales. Pueden realizar cortes o grabados mediante laser, tubos de CO2, plasma o agua.

- **Plástico** → la principal maquinaria que trabaja el plástico son las impresoras 3D, que crean piezas compactas.

3.4.4. Número de ejes.

Dependiendo del número de ejes disponibles, la función de la maquina puede cambiar. Es posible encontrar máquinas de 2 ejes, para movimientos lineales, de 3 ejes, para moverse en las tres direcciones del espacio e incluso máquinas de 5 o 6 ejes, los cuales permiten no solo el movimiento de la herramienta, sino que también permiten el movimiento de la bancada o la rotación del material a tratar.

3.4.5. Función.

Este grupo es el más diverso de todos, ya que existen muchas finalidades diferentes para un máquina de control numérico y con forme van evolucionando van surgiendo nuevas funciones.

Algunos ejemplos de máquinas CNC son:

- **Fresadora**: máquina-herramienta para realizar trabajos por arranque de viruta mediante el movimiento de una herramienta rotativa denominada fresa. Es usada para mecanizar diversos materiales como madera, acera o hierro.

Ilustración 3.4.4 Ejemplo de fresadora actual.

- **Torno**: máquina-herramienta que permite mecanizar, roscar, cortar, trapeciar, agujerear, cilindrar, desbastar y ranurar piezas de forma geométrica por revolución.

Ilustración 3.4.5 Ejemplo de torno actual.

- **Cortadora FOAM**: máquina-herramienta que permite realizar cortes a través de un hilo caliente de Nicrom que se mueve entre dos ejes paralelos X-Y. El FOAM (espuma visco elástica) es el material más común cortado por estas máquinas.

Ilustración 3.4.6 Ejemplo de cortadora FOAM actual.

- **Rectificadora**: máquina-herramienta utilizada en el mecanizado de piezas por abrasión. Es un proceso que elimina material de una pieza con el fin de darle forma y modelarla.

Ilustración 3.4.7 Ejemplo de rectificadora actual.

- **Impresora 3D**: máquina capaz de realizar réplicas de diseños en 3D, creando piezas o maquetas volumétricas a partir de un diseño hecho por ordenador. Se puede observar una en la Ilustración 3.2.3 del apartado 3.2.

3.5. Implementación de una máquina CNC.

El significado de maquina CNC es máquina de control numérico por computadora, es decir, el controlador de las máquinas CNC recibe instrucciones de la computadora (en forma de códigos G y códigos M) y mediante su propio software interpreta esas instrucciones dando como resultado señales eléctricas destinadas a activar los motores que, a su vez, pondrán en marcha el sistema de accionamiento.

Para comprender en términos generales cómo funciona una máquina CNC se van a exponer algunos de los elementos que la componen y su funcionamiento.

3.5.1. Elementos.

A) *Motores PAP (paso a paso)*

Un motor paso a paso es un **dispositivo electromecánico** que convierte una serie de **pulsos eléctricos** en desplazamientos angulares (pasos). Estos motores son ideales para la construcción de mecanismos que requieran movimientos muy precisos. La característica principal que los define es que son capaces de moverse un paso por cada pulso aplicado, a diferencia del resto de motores de continua que giran libremente, y que pueden ser controlados y configurados, lo que les permite variar la velocidad de giro cambiando el tiempo transcurrido entre pasos.

El paso puede variar según el motor, desde pasos grandes de 90º hasta pasos pequeños de 1.8º. Además, estos motores poseen la habilidad de quedar enclavados en una posición, si una o más de sus bobinas están energizadas o bien quedando totalmente libres de corriente.

A continuación se exponen los distintos tipos de motores PAP que existen:

- **Motor PAP de imanes permanentes**: está formado por un estator de forma cilíndrica con un cierto número de bobinados alimentados en secuencia que crean un campo magnético giratorio de manera discontinua. El rotor concéntrico con el estator y situado sobre el eje contiene un imán fuertemente magnetizado, con dos polos magnéticos S-N, que a cada instante tenderá a alinearse con el campo magnético generado por la correspondiente bobina del estator. Su modo de operación es simple, se basa en la fuerza de atracción de dos imanes, uno permanente y otro temporal.

  Se caracterizan por tener una velocidad y un par bajo, con un ángulo de paso elevado (entre 45º y 90º) que dependerá del número de polos del estator y el rotor, por lo que su precio es más económico.

- **Motor PAP de reluctancia variable**: en este caso no se utiliza un campo magnético permanente, como resultado puede moverse sin limitaciones o sin un par de parada. El estator presenta la forma cilíndrica habitual conteniendo generalmente un total de tres devanados distribuidos de tal forma que estén separados 120º aproximadamente entre

dos de ellos. El rotor está formado por un núcleo de hierro dulce de estructura cilíndrica pero con una serie de dientes tallados longitudinalmente a lo largo de la superficie lateral. Cuando el primer devanado recibe la alimentación, atraerá al rotor hasta que el diente más cercano se alinee con el campo. Al llegar la excitación al siguiente devanado, otro diente será el más próximo, con lo que el rotor girará, de la misma forma, con el siguiente impulso avanzando un ángulo igual a los anteriores.

A diferencia del anterior, la polaridad o sentido de circulación de la corriente en cada devanado es diferente, ya que al no estar imantado el rotor, siempre se desplazará hasta la posición en que la reluctancia del circuito magnético del estator sea mínima.
La principal característica de estos motores de reluctancia variable es la elevada velocidad de accionamiento que permiten. Este motor se desarrolló con objeto de poder conseguir unos desplazamientos angulares más reducidos que en el caso anterior, sin que por este motivo haya de aumentarse considerablemente el número de bobinados.

- **Motor PAP hibrido**: operan combinando los principios de los motores de imán permanente y los de reluctancia variable, intentando obtener las características que destacan en cada uno de ellos. El motor híbrido consiste en un estator dentado y un rotor constituido por anillos de acero dulce dentado en un número ligeramente distinto al del estator y dichos anillos montados sobre un imán permanente dispuesto axialmente. Las líneas magnéticas que genera el imán son guiadas por dos cilindros acoplados a los extremos de cada uno de sus polos (norte y sur).

  En el motor de reluctancia variable, la excitación es producida únicamente por medio del bobinado, mientras que en el motor híbrido la excitación es conjunta entre el bobinado y el imán.

  La principal característica de estos motores es que se obtienen unos ángulos de paso pequeño y alto par con un tamaño menor a los otros motores.

En función de la forma de excitación y conexión de las bobinas del estator se puede clasificar a los motores PAP en dos grupos:

- **Motor PAP unipolar**: estos motores cuentan con dos bobinas con un punto medio de los cuales salen los cables hacia el exterior; estos cables se conectan a la fuente mientras que los extremos de las bobinas son aterrizadas para cerrar el circuito; dependiendo del tipo de motor, las líneas comunes pueden ser independientes o no.

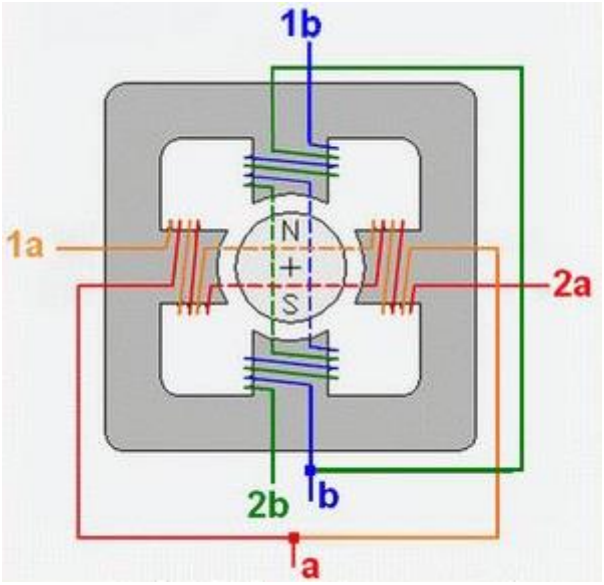


Ilustración 3.5.1 Distribución de los devanados de un motor PAP unipolar.

- **Motor PAP bipolar**: en este caso se cuentan con dos bobinas sin ningún punto medio donde salga un cable, por lo que se tienen cuatro cables y cada par corresponde a los terminales de una bobina.

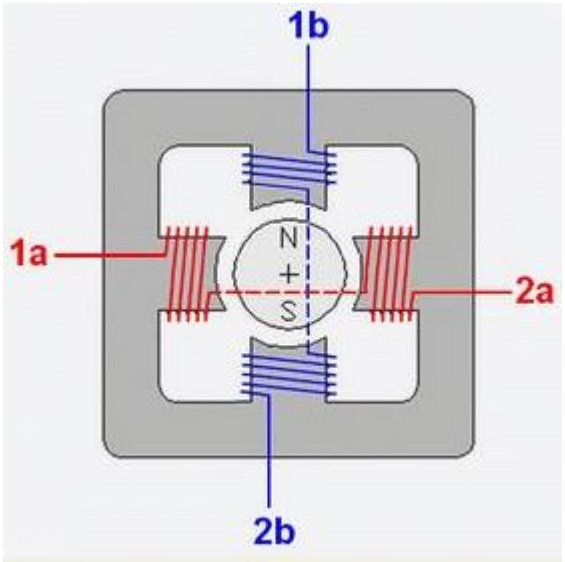


Ilustración 3.5.2 Distribución de los devanados de un motor PAP bipolar.

Los motores bipolares en comparación con los unipolares presentan mayor par, mayor anclaje, menor tamaño, precio más económico y un control más complejo.

B) *Drivers control motores PAP*

Un driver para motores paso a paso es un dispositivo electrónico diseñado para el control de la señal eléctrica o pulsos de dichos motores. Su función es la de recibir las señales enviadas desde el microcontrolador, que definen el tipo de paso del motor (micro-step, half-step, full-step), el sentido de giro, la velocidad, el par…

Mediante el uso de una fuente de alimentación externa y a través de una serie de componentes y puentes en H, el driver modifica a valores nominales la tensión y corriente del motor necesarias para su movimiento, siguiendo las órdenes enviadas desde el microcontrolador.

C) *Microcontrolador*

Un microcontrolador es un circuito integrado programable capaz de ejecutar un programa previamente diseñado, verificado y compilado que se encuentra almacenado en su memoria. Están compuestos de varios bloques funcionales, que cumplen una tarea específica. Algunos de estos bloques son:

- **CPU**: es la unidad central de proceso que interpreta las instrucciones de un programa informático mediante la realización de las operaciones básicas aritméticas, lógicas y de entrada/salida del sistema.

- **Memorias (RAM y Flash)**: La memoria RAM está destinada al almacenamiento de información temporal que será utilizada por el procesador para realizar cálculos u otro tipo de operaciones lógicas.
  Mientras que la memoria flash permite la lectura y escritura de múltiples posiciones de memoria en la misma operación. Gracias a ello, la tecnología flash, mediante impulsos eléctricos, permite velocidades de funcionamiento muy elevadas.

- **Periféricos**: Se considera periférico al conjunto de dispositivos que sin pertenecer al núcleo fundamental de la computadora, formado por la unidad central de procesamiento (CPU) y la memoria central, permitan realizar operaciones de entrada/salida (E/S) complementarias al proceso de datos que realiza la CPU.

Estas tres unidades básicas en un computador, CPU, memoria central y el subsistema de E/S, están comunicadas entre sí por tres buses o canales de comunicación:

- **Direcciones**, para seleccionar la dirección del dato o del periférico al que se quiere acceder.

- **Control**, básicamente para seleccionar la operación a realizar sobre el dato (principalmente lectura, escritura o modificación).

- **Datos**, por donde circulan los datos.

Un microcontrolador tiene los mismos bloques de funcionamiento básicos de una computadora lo que nos permite tratarlo como un pequeño dispositivo de cómputo.
Por tanto, su función es la de núcleo central del sistema, ya que es responsable de almacenar y ejecutar el programa que configura los drivers para el movimiento del motor y además recibe señales externas que informan del estado del sistema.

D) *Final de carrera*

Se trata de un dispositivo electrónico, mecánico o neumático, situado al final de un recorrido o de un elemento móvil, que indica la posición final de este mediante un accionamiento mecánico y a su vez envía una señal para identificar el estado del sistema.

Son sensores electromecánicos, es decir, combinan una parte mecánica con una eléctrica. También son conocidos como sensores de contacto y por tanto necesitan estar en contacto con un objeto para detectar la llegada de un elemento móvil.

En la actualidad se están desarrollando sensores ópticos para realizar la función de los finales de carrera, evitando así la parte mecánica y el contacto con el objeto.
Internamente están compuestos de interruptores normalmente abiertos (NA), cerrados (NC) o conmutadores dependiendo de la función que vayan a cumplir.

- **Circuito anti rebote**: los finales de carrera electrónicos envían una señal eléctrica al ser accionada su parte mecánica. Debido a que la entrada del microcontrolador es muy sensible y para evitar el retorno de la señal, ruido en la misma o un mal contacto del sensor, se puede optar por incorporar un circuito anti rebote a la señal del final de carrera. Una posición del sensor enviará un voltaje que equivaldrá a un 1 lógico mientras que la otra posición enviará otro voltaje distinto que equivaldrá a un 0 lógico.

3.5.2. Control y movimiento de los motores.

Como se ha comentado en el apartado anterior el movimiento de los motores PAP se realiza a través de un driver, que configura los impulsos eléctricos (señal PWM) que excitan las bobinas

del motor para realizar su movimiento. La excitación de las bobinas se puede realizar siguiendo distintos patrones:

- **Full-step**: el movimiento del rotor se consigue con el cambio de polaridad en la corriente que circula por los bobinados de las fases. Existen dos métodos de excitación de los devanados en el modo full-step, excitando una fase cada vez o dos fases de forma simultánea.

  El primer modo requiere menos cantidad de energía por parte del driver que cualquier otro modo. En el segundo el par que se obtiene en esta secuencia es el máximo que puede entregar el motor, ya que en todo momento ambas fases están activas con la intensidad nominal de trabajo, sin embargo consume más energía que el modo anterior.

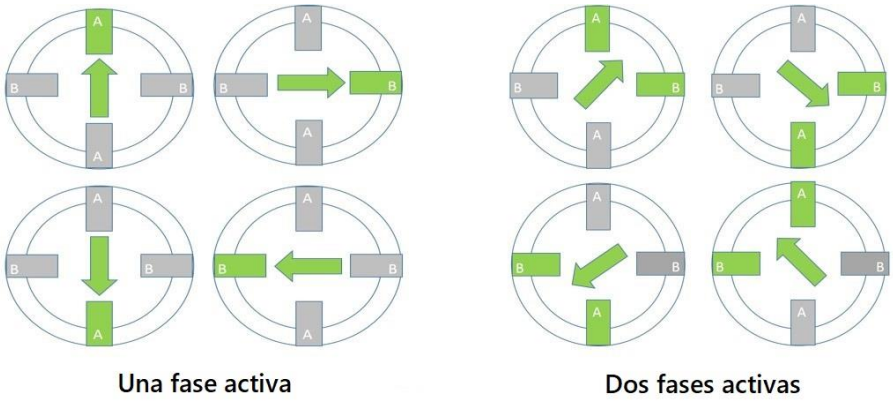


Ilustración 3.5.3 Métodos de excitación de los devanados con un patrón full-step.

- **Half-step**: En este modo, las secuencias de la onda y el control del paso completo están entremezclados, de manera que se permita que el rotor esté alineado en la mitad de cada paso.

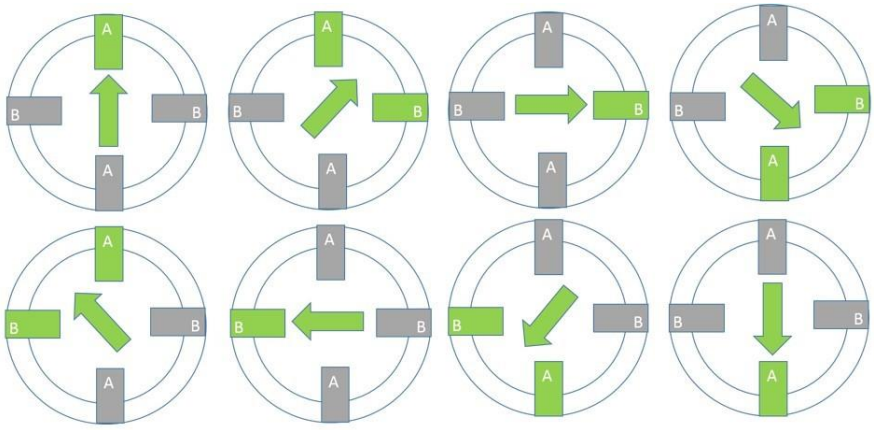


Ilustración 3.5.4 Excitación de los devanados con un patrón half-step.

- **Micro-step**: el “microstepping” básicamente consiste en dividir 1 paso del motor en varios micro pasos. Al controlar la dirección y la amplitud del flujo de corriente en cada devanado, la resolución aumenta y las características del motor mejoran, dando menos vibración y un funcionamiento más suave. Debido a que las ondas sinusoidales trabajan juntas, hay una transición suave de un devanado a otro. Cuando la corriente aumenta en uno, disminuye en el otro, dando como resultado una progresión de paso suave y una salida de par mantenida.

No hay que excederse en el “microsteping”, ya que si bien se puede estar añadiendo precisión al CNC, al usar un porcentaje de la corriente también se está perdiendo fuerza en el motor.

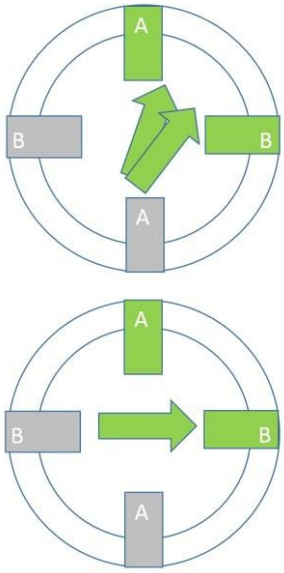
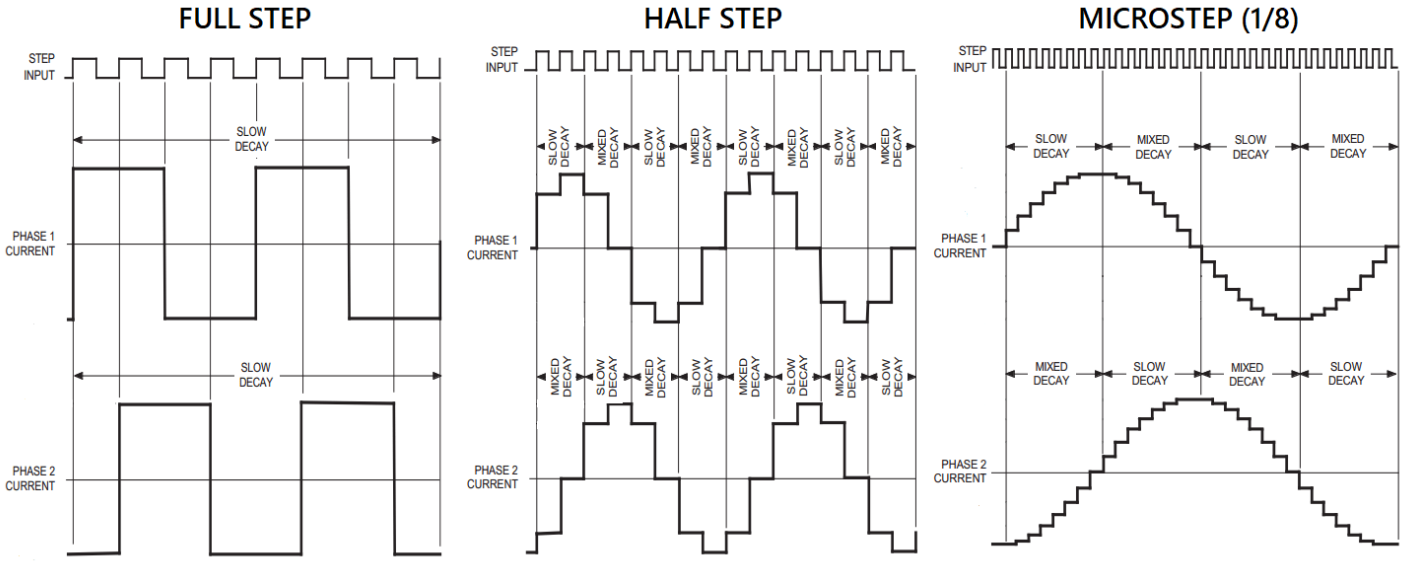


Ilustración 3.5.5 Excitación de los devanados con un patrón micro-step.

A continuación se puede observar un gráfico con los distintos patrones de señal con los que excitar los devanados de un motor PAP:



Gráfica 3.5.1 Distintos patrones de señal para la excitación de los devanados de un motor PAP.

Algunos de los parámetros característicos de los motores PAP a tener en cuenta son:

- **Ángulo de paso**: ángulo de giro del eje del motor para pasar de una posición estable a otra.

- **Número de pasos por vuelta**: número total de pasos que debe realizar el motor para que su eje de una vuelta completa.

$$N^{o}{}_{pasos} = \frac{360}{Ángulo_{paso}}$$

- **Precisión de paso (step accuracy)**: hace referencia al error de posición que puede tener el motor entre la posición real alcanzada y la posición teórica requerida. Se trata de un valor constante.

- **Par de mantenimiento (holding torque)**: par resistente que ejerce un motor detenido en una posición estable, sin energizar los devanados.

- **Par de retención (detent torque)**: cantidad de torsión necesaria para mover el motor un paso completo cuando los devanados están energizados pero el rotor está parado.

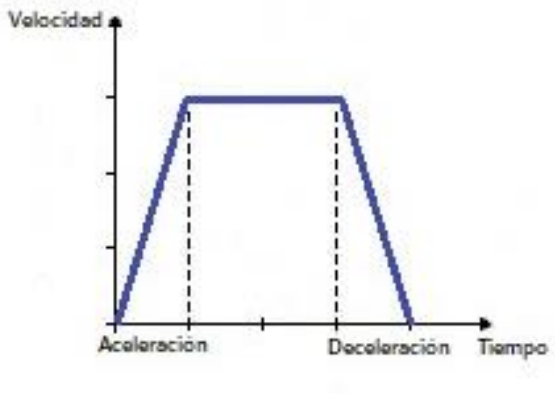
- **Par pull-in**: Relación del par que es capaz de entregar el motor a máxima velocidad sin pérdida de pasos.

- **Par pull-out**: Relación del par que es capaz de entregar el motor con la velocidad sin pérdida de pasos durante el arranque y la parada.

Una vez visto cómo funciona el movimiento de los motores se pasa a la coordinación de sus movimientos con el fin de poder controlarlos.

Para ello, se creó un firmware de control de los motores PAP combinado con un lenguaje específico de los sistemas de control numérico, a este firmware se le conoce más comúnmente como GRBL.

El firmware de control de los motores PAP convierte cada movimiento en una lista de trapecios. Esto es debido a que el motor para moverse de un punto de inicio a otro final, debe pasar de un estado de reposo a otro de movimiento entre el cual existe una aceleración, hasta alcanzar la velocidad deseada, y una deceleración antes de la llegada al punto final.



Gráfica 3.5.2 Velocidad de un motor PAP con el paso del tiempo.

Este es el núcleo del algoritmo, despedazar los movimientos en trapecios para cada motor, dividiendo los movimientos en una lista de líneas rectas y calcular la aceleración y frenada de cada segmento, en cada eje.

La lista de trapecios se envía a la parte más importante del programa, la interrupción de “timer”, donde se transforman esos trapecios en los pulsos anteriormente vistos, a una velocidad perfectamente definida para que los motores se muevan de forma coordinada. Todo esto se tiene que ejecutar de forma perfectamente sincronizada.

En cuanto al lenguaje de programación, es denominado “**G-code**” y posee múltiples implementaciones dependiendo del fabricante. Es el lenguaje más usado para el control numérico asistido por computador, consiste en una lista de comandos que realizan una serie de funciones específicas, indicándole a la máquina mediante coordenadas, parámetros de velocidad y comandos de movimiento, que trayectoria debe seguir la herramienta. Se puede encontrar detallado cada uno de los comandos en el manual “**The Nist RS274NGC Interpreter**”.

Los comandos están divididos en grupos modales, en los cuales solo puede actuar un comando a la vez, dado que es lógicamente imposible que actúen dos comandos del mismo grupo simultáneamente.

The modal groups for G codes are:

group 1 = {G0, G1, G2, G3, G38.2, G80, G81, G82, G83, G84, G85, G86, G87, G88, G89} motion
group 2 = {G17, G18, G19} plane selection
group 3 = {G90, G91} distance mode
group 5 = {G93, G94} feed rate mode
group 6 = {G20, G21} units
group 7 = {G40, G41, G42} cutter radius compensation
group 8 = {G43, G49} tool length offset
group 10 = {G98, G99} return mode in canned cycles
group 12 = {G54, G55, G56, G57, G58, G59, G59.1, G59.2, G59.3} coordinate system selection
group 13 = {G61, G61.1, G64} path control mode

The modal groups for M codes are:

group 4 = {M0, M1, M2, M30, M60} stopping
group 6 = {M6} tool change
group 7 = {M3, M4, M5} spindle turning
group 8 = {M7, M8, M9} coolant (special case: M7 and M8 may be active at the same time)
group 9 = {M48, M49} enable/disable feed and speed override switches

In addition to the above modal groups, there is a group for non-modal G codes:

group 0 = {G4, G10, G28, G30, G53, G92, G92.1, G92.2, G92.3}

Tabla 3.5.1 Grupos modales para G-codes.

El código G se ocupa de la geometría del hardware, por ejemplo, desarrollos de movimiento directo, tareas de penetración y determinación de las unidades de estimación.

G Code	Meaning
G0	rapid positioning
G1	linear interpolation
G2	circular/helical interpolation (clockwise)
G3	circular/helical interpolation (counterclockwise)
G4	dwell
G10	coordinate system origin setting
G17	XY-plane selection
G18	XZ-plane selection
G19	YZ-plane selection
G20	inch system selection
G21	millimeter system selection
G28	return to home
G30	return to secondary home
G38.2	straight probe
G40	cancel cutter radius compensation
G41	start cutter radius compensation left
G42	start cutter radius compensation right
G43	tool length offset (plus)
G49	cancel tool length offset
G53	motion in machine coordinate system
G54	use preset work coordinate system 1
G55	use preset work coordinate system 2
G56	use preset work coordinate system 3
G57	use preset work coordinate system 4
G58	use preset work coordinate system 5
G59	use preset work coordinate system 6
G59.1	use preset work coordinate system 7
G59.2	use preset work coordinate system 8
G59.3	use preset work coordinate system 9
G61	set path control mode: exact path
G61.1	set path control mode: exact stop
G64	set path control mode: continuous
G80	cancel motion mode (including any canned cycle)
G81	canned cycle: drilling
G82	canned cycle: drilling with dwell
G83	canned cycle: peck drilling
G84	canned cycle: right hand tapping
G85	canned cycle: boring, no dwell, feed out
G86	canned cycle: boring, spindle stop, rapid out
G87	canned cycle: back boring
G88	canned cycle: boring, spindle stop, manual out
G89	canned cycle: boring, dwell, feed out
G90	absolute distance mode
G91	incremental distance mode
G92	offset coordinate systems and set parameters
G92.1	cancel offset coordinate systems and set parameters to zero
G92.2	cancel offset coordinate systems but do not reset parameters
G92.3	apply parameters to offset coordinate systems
G93	inverse time feed rate mode
G94	units per minute feed rate mode
G98	initial level return in canned cycles
G99	R-point level return in canned cycles

Tabla 3.5.2 Comandos G.

Mientras que los códigos M se ocupan de la configuración de las herramientas de la máquina, tales como los comandos de encendido/apagado y el retorno de la máquina al origen o al punto de corte.

M Code	Meaning
M0	program stop
M1	optional program stop
M2	program end
M3	turn spindle clockwise
M4	turn spindle counterclockwise
M5	stop spindle turning
M6	tool change
M7	mist coolant on
M8	flood coolant on
M9	mist and flood coolant off
M30	program end, pallet shuttle, and reset
M48	enable speed and feed overrides
M49	disable speed and feed overrides
M60	pallet shuttle and program stop

Table 7. M Codes

Tabla 3.5.3 Comandos M.

3.5.3. Transmisiones.

Las transmisiones son mecanismos que sirven para transmitir, comunicar energía o movimiento desde un punto de una máquina a otro. Por ello son necesarias para transmitir el movimiento de los motores a cada uno de los elementos, que permiten movilidad, de la máquina. Algunos de los elementos más comunes y utilizados son:

- **Engranajes**: los engranajes son juegos de ruedas que disponen de unos elementos salientes denominados “dientes”, que encajan entre sí, de manera que transmiten movimiento circular de una rueda a otra. No necesitan correa de transmisión ni otro medio de unión, simplemente están unidas por sus dientes.

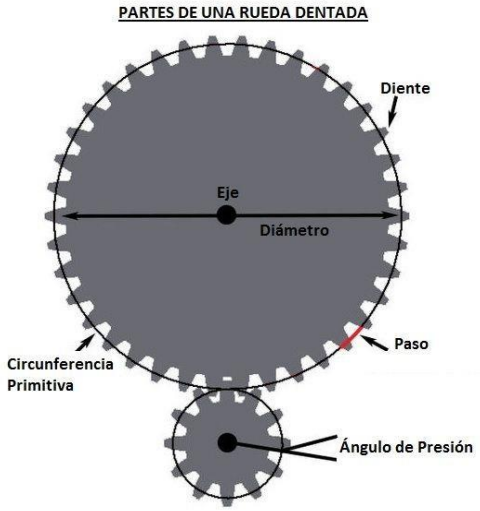


Ilustración 3.5.6 Disposición de las partes de una rueda dentada.

Para que dos ruedas dentadas formen un engranaje deben tener el mismo tipo de dientes, es decir deben ser del mismo tipo de rueda dentada. Lo que varía es el número de dientes de una u otra rueda dentada. Además como ya vimos también deben tener el mismo paso y módulo.

Algunos tipos de engranaje son:

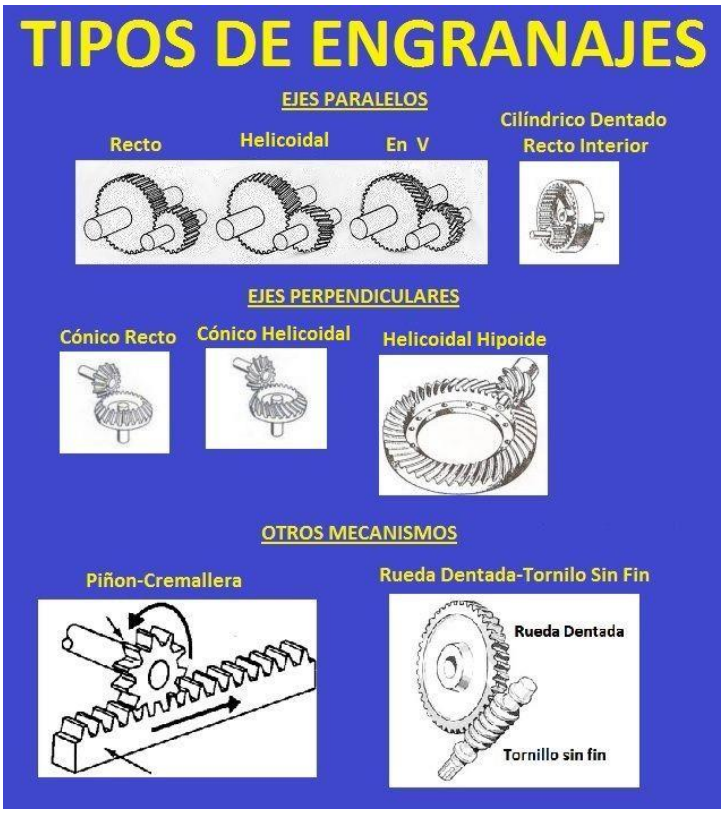


Ilustración 3.5.7 Tipos de engranajes.

- **Correas dentadas**: las correas dentadas o correas de distribución son uno de los métodos más comunes de transmisión de energía mecánica entre dos ejes. Mediante un sistema dentado, que comparte con los ejes, permiten la transmisión del movimiento circular de un eje a otro perpendicular sin necesidad de contacto entre ellos.

Ilustración 3.5.8 Ejemplo de correa dentada.

- **Husillos**: los husillos son un tipo de tornillo largo y de gran diámetro que se utilizan para producir el desplazamiento lineal de la tuerca. Al aplicarse rotación al tornillo se produce el desplazamiento de la tuerca. También son conocidos como tornillo sin fin.

Ilustración 3.5.9 Ejemplo de husillo.

4. DISEÑO DEL SISTEMA

4.1. Mecánica y estructura.

El sistema a tratar está constituido por dos ejes perpendiculares entre sí, que permiten el desplazamiento longitudinal y transversal de la herramienta.

Para el diseño del sistema se ha a reutilizado materiales y piezas que han caído en desuso en el laboratorio. De esta manera se consigue un ahorro de costes en material y su reciclaje. Sin embargo no ha sido posible un reciclaje al 100%, también se han tenido que diseñar algunas de las piezas del sistema.

4.1.1. Base.

El sistema se encuentra anclado a una base cuadrangular de 50x50 mm de metacrilato (PMMA) transparente, que permite el visionado de las conexiones electrónicas así como su disposición. Proporciona un buen soporte y robustez, pero su principal ventaja es su reducido peso, fundamental para permitir el transporte del sistema sin mucho esfuerzo. Presenta una serie de almohadillas en la parte inferior para adherirse a la superficie sobre la que se coloque.

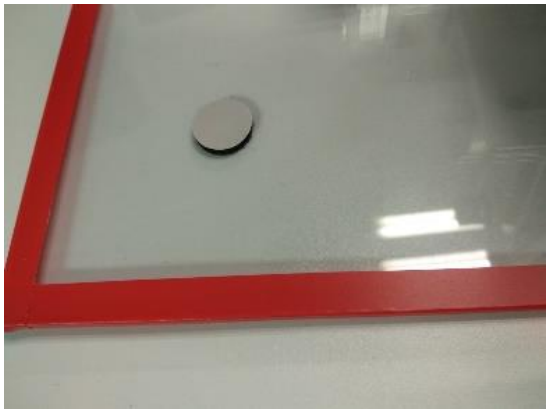


Ilustración 4.1.1 Base de metacrilato que sostiene el sistema.

La base de la estructura principal del sistema está compuesta por láminas de acero inoxidable de forma rectangular. Presentan una serie de agujeros que sirven para la sujeción de las mismas y para la combinación con el resto de elementos.

Ilustración 4.1.2 Láminas de acero que forman la estructura del sistema.

4.1.2. Ejes y transmisiones.

Las guías de desplazamiento están compuestas por varillas lisas calibradas, de acero inoxidable, de 10 mm de diámetro. Para conseguir una mayor precisión y con el fin de evitar vibraciones, deformidades y flexiones, cada eje contará con dos varillas cuyos extremos van sujetos a los salientes de la estructura principal.

Ilustración 4.1.3 Guías de acero del sistema.

Se han empleado una serie de correas y ruedas dentadas que transmitirán el movimiento del eje del motor al resto de elementos. Las ruedas dentadas están formas de acero inoxidable de 20 mm de diámetro, mientras que las correas dentadas están formadas por una combinación de caucho y nylon para los dientes, con hilos metálicos en su interior.

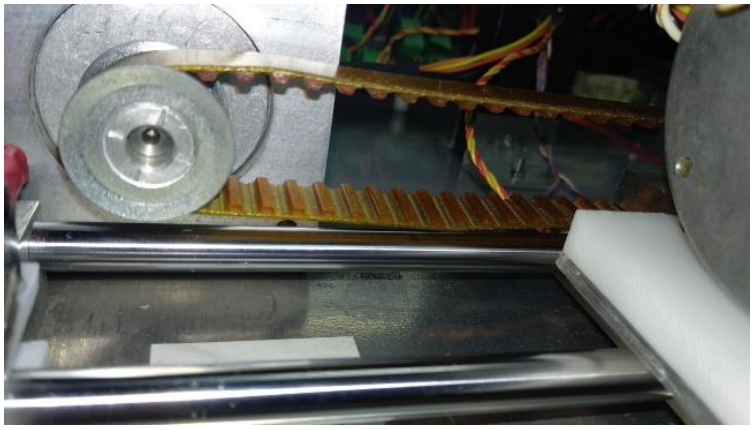


Ilustración 4.1.4 Transmisión del sistema: combinación de correa y rodamiento.

El sistema de desplazamiento consiste en la combinación de las guías con las correas dentadas. Se realiza a través de unas deslizaderas de acero inoxidable en forma rectangular, que presentan dos agujeros transversales que permiten introducirlas dentro de las guías, y una serie de agujeros para el anclaje de las mismas a la correa y la colocación de otros elementos del sistema.

Ilustración 4.1.5 Deslizadera de acero del sistema.

4.1.3. Elementos diseñados.

El resto de elementos han sido diseñados a partir del programa informático SolidWorks e impresos con una impresora 3D. Para la creación de las piezas se han utilizado una especie de filamentos termoplásticos, concretamente de tipo HIPS, que es un poliestireno de alto impacto y aunque no sea biodegradable si es muy duro y rígido. Presenta una densidad de 1.04 g/cm^3, sin embargo es endeble frente a la radiación UV.

Con la impresora se han diseñado soportes para los finales de carrera, elementos de sujeción de los ejes del sistema, la pieza de sujeción de la herramienta taladro y la pieza básica de soporte para el eje Y.

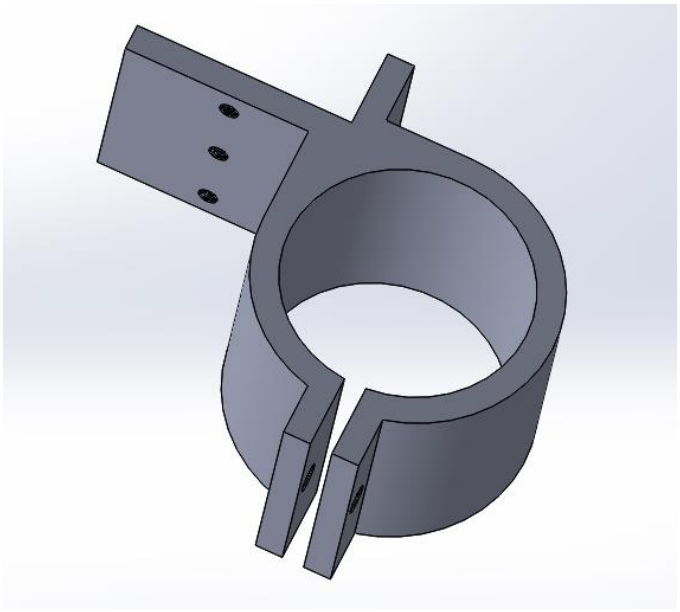


Ilustración 4.1.6 Ejemplo de pieza diseñada con el programa SolidWorks.

En el ANEXO D se encuentran los detalles técnicos de todas y cada una de las piezas diseñadas

4.2. Electrónica.

En este capítulo se van a exponer todos los dispositivos electrónicos involucrados en el funcionamiento del sistema. Se detallarán sus especificaciones técnicas y su función.

4.2.1. Fuente de alimentación.

La fuente de electricidad del sistema es un **generador de corriente continua**, el cual alimenta a una gran cantidad de componentes electrónicos. Se ha utilizado un generador "Manson EP-603", que presenta las siguientes características:

- Corriente de salida: 0…2.5 A
- Tensiones de salida: 12V CC → 0.5A ; 5V CC → 0.5A ; 0…30 V CC
- Fuente alimentación: 230V CA 50/60Hz

- Protecciones: anti cortocircuito y sobrecargas
- Clase de alimentación: multicanal
- Resolución de salida: 1V/100mA
- Dimensiones y peso: 145x150x200 mm 2.8 kg

Ilustración 4.2.1 Fuente de alimentación Manson EP-603

Sin embargo, por la necesidad de obtener diferentes tensiones de salida y con el fin de no utilizar otra fuente de alimentación, se han utilizado dos **transformadores de corriente continua** DC-DC-QS 003, cuyas características son las siguientes:

- Tensión de entrada: 4.5…30 V
- Tensión de salida: 1.5…30 V ajustable
- Corriente de salida: 0…12 A
- Eficiencia de conversión: 95%
- Protecciones: contra cortocircuitos y por calentamiento
- Dimensiones: 60x51x22 mm

Ilustración 4.2.2 Transformador DC-DC-QS 003.

Por tanto el generador alimentará a ambos transformadores con su salida variable de corriente continua y además aportará dos salidas fijas de corriente continua de 12 y 5 V.

4.2.2. Microcontrolador.

Para el control del sistema se va a utilizar un **microcontrolador** de ST Microelectronics, perteneciente a la familia de STM32 Nucleo board, exactamente el "NUCLEO-L053R8", que presenta las siguientes características:

- Procesador ARM 32-bit Cortex-M0.
- Frecuencia de 75M Hz (125 DMIPS).
- Memoria flash de 512k bytes y memoria SRAM de 128k bytes.
- Módulos de extensión STM Morpho y Arduino UNO R3.
- Alimentación: ST-LINK (3.3V), USB Vbus (5V) o fuentes externas (7…12V).
- Dos pulsadores: USER y RESET.
- Tres indicaciones LED para comunicación USB (LD1), usuario (LD2) y alimentación (LD3).
- Depurador/programador ST-LINK/V2-1 con conexión SWD.
- Compatible con entornos de desarrollo integrado (IDE) como IAR, Keil, GCC.
- El USB soporta tres interfaces: puerto COM virtual, almacenamiento masivo y puerto de depuración.
- Biblioteca HAL completa de software STM32.

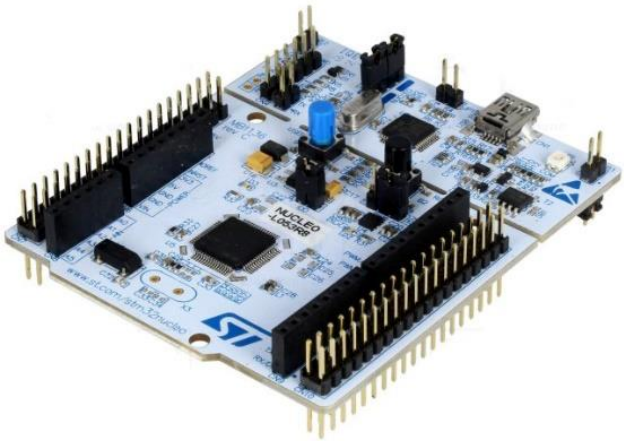


Ilustración 4.2.3 Microcontrolador X-NUCLEO-L053R8.

El microcontrolador será el cerebro del sistema, es decir, todas las acciones del sistema serán ordenadas y supervisadas por este dispositivo.

4.2.3. Drivers.

Los **drivers** a utilizar para los motores PAP son de ST Microelectronics, concretamente la placa "X-NUCLEO-IHM01A1", montada con el dispositivo L6474 y diseñada especialmente para su control mediante los microcontroladores de la familia "NUCLEO". Presenta las siguientes especificaciones técnicas:

- Rango de voltaje 8…45 V.
- Corriente de fase máxima 3 A r.m.s.
- Leds de alimentación y fallo.
- Avanzado control de corriente.
- Protección de la etapa de potencia.
- Resolución máxima de 1/16 micro pasos.
- Compatibilidad con STM Nucleo board y Arduino UNO R3.
- Soluciones multi-motor.
- Comunicación SPI hasta 5Mbit/s.

Ilustración 4.2.4 Driver controlador X-NUCELO-IHM01A1.

El sistema presenta dos motores PAP, uno por cada eje, y por consecuencia se necesitarán dos drivers. En función de la cantidad de motores a controlador los drivers presentan diferentes configuraciones. Para el control de dos motores PAP, la configuración es la siguiente:

- **MOTOR 1**

  - Posición jumpers: JP1 off, JP5 (PWR) en UV5, JP6 (IDD) on
  - Resistencias montadas → R1, R4, R7 y R10
  - Resistencias desmontadas → R2, R3, R5, R6, R8, R9, R11 y R12

Ilustración 4.2.5 Configuración de las resistencias para el control del primer motor de un total de dos.

- **MOTOR 2**

  - Posición jumpers: JP1 off, JP5 (PWR) en UV5, JP6 (IDD) on
  - Resistencias montadas → R2, R5, R8 y R12
  - Resistencias desmontadas → R1, R3, R4, R6, R7, R9, R10 y R11

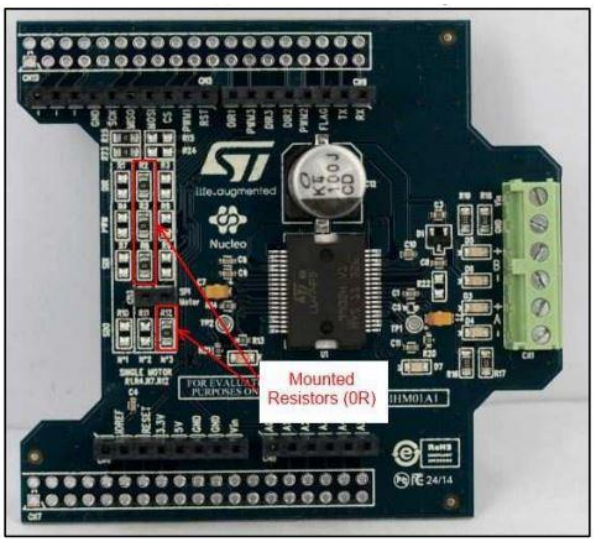


Ilustración 4.2.6 Configuración de las resistencias para el control del segundo motor de un total de dos.

4.2.4. Motores PAP.

Para el movimiento de los ejes del sistema se ha optado por la utilización de dos **motores paso a paso híbridos** de la marca RS PRO, concretamente el modelo 440-442 que presenta las siguientes características:

- Voltaje nominal 5V
- Corriente nominal 1 A.
- Resistencia interna 5Ω
- Inductancia 9m H
- Par de retención 500m Nm
- Angulo de paso 1.8º
- Precisión de ángulo de paso 5%

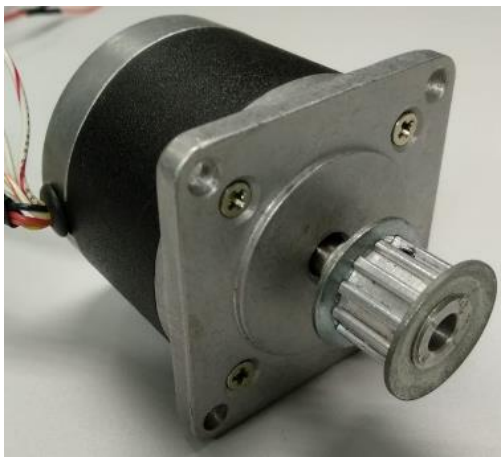


Ilustración 4.2.7 Motor PAP híbrido RS Pro 440-442.

Se trata de motores con 8 hilos, es decir, según el tipo de conexionado, las etapas de potencia del motor pueden ser unipolares o bipolares. En los unipolares la excitación de los devanados realiza de uno en uno mientras que en los bipolares se realiza por pares con inversión de la corriente.

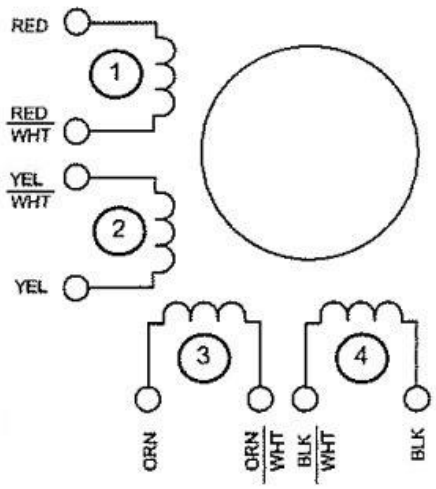
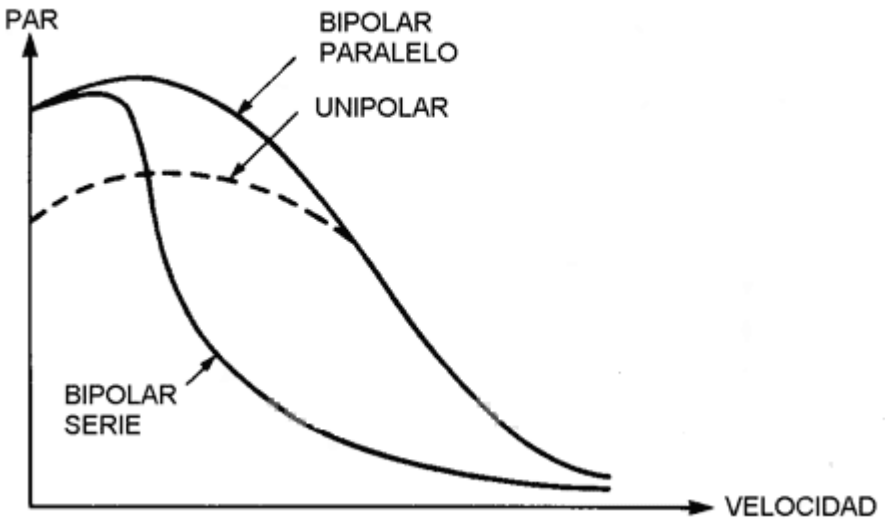


Ilustración 4.2.8 Distribución de los hilos de conexionado del motor PAP RS Pro 440-442.

La elección del conexionado depende de las necesidades de par y velocidad que tenga el sistema. La configuración unipolar obtiene un mayor rendimiento a alta velocidad mientras que la bipolar lo obtiene a baja velocidad. Como se puede observar en el gráfico 4.2.1, se puede obtener un mayor par con la configuración bipolar, pero este decaerá en función de la velocidad, mientras que con la configuración unipolar este decaimiento es menos pronunciado.



Gráfica 4.2.1 Relación par-velocidad de los distintos motores PAP.

En el caso que concierne se ha optado por una etapa de potencia bipolar, dado que se prioriza el tener un par elevado a una velocidad elevada. Ahora se debe decidir si la conexión de los devanados se realiza en serie o en paralelo. El conexionado en paralelo permite una gama de velocidades más amplia que en serie, pero un mayor consumo, puesto que la corriente por fase se duplica.

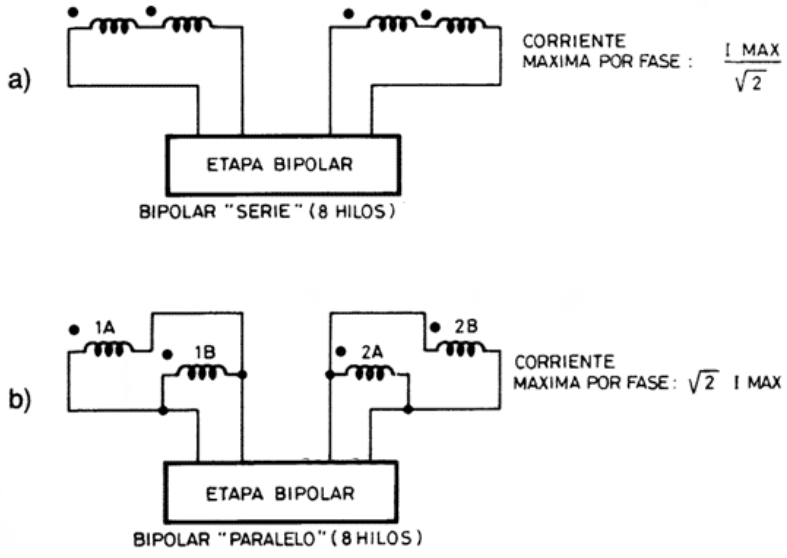
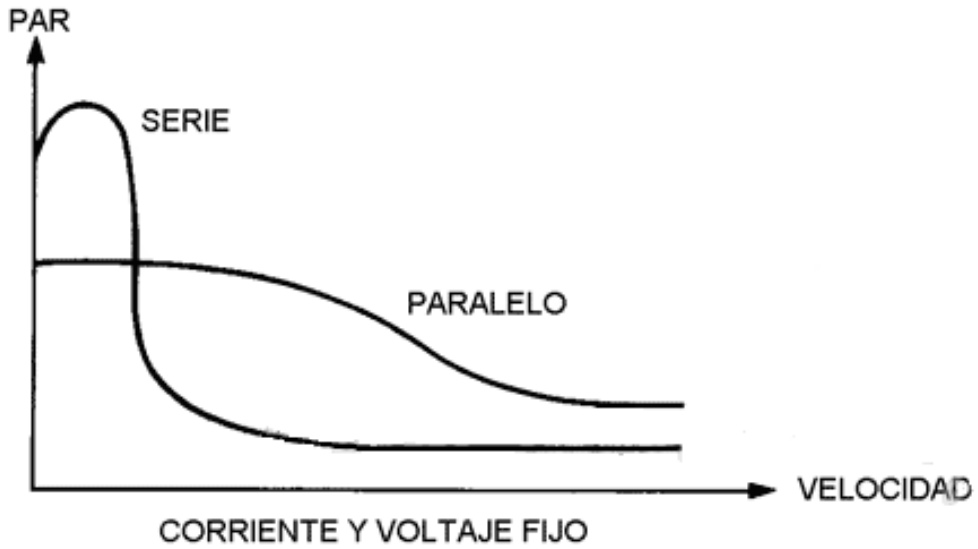


Ilustración 4.2.9 Posibles conexiones de los devanados de un motor PAP bipolar.

Hasta el momento sólo se ha considerado el máximo par teórico disponible del motor, asumiendo que las únicas limitaciones son las dimensiones del motor y la temperatura de trabajo, siempre a la máxima corriente disponible para cada caso. Pero cuando se realiza en condiciones de corriente y voltaje fijos, el par de la conexión en serie casi duplica al paralelo.



Gráfica 4.2.2 Relación par-velocidad según la conexión del motor PAP.

Por tanto se ha optado por una conexión **bipolar en serie**, priorizando un par elevado a una velocidad alta, con un consumo menor, que garantizará que se cumplan todos los requisitos demandados por el sistema.

La distribución de los hilos será de la siguiente forma:

- **Amarillo** → devanado A +
- **Blanco/amarillo + blanco/rojo** → Unidos entre si y libres (conexión serie)
- **Rojo** → devanado A -
- **Naranja** → devanado B +
- **Blanco/naranja + blanco/negro** → Unidos entre si y libres (conexión serie)
- **Negro** → devanado B -

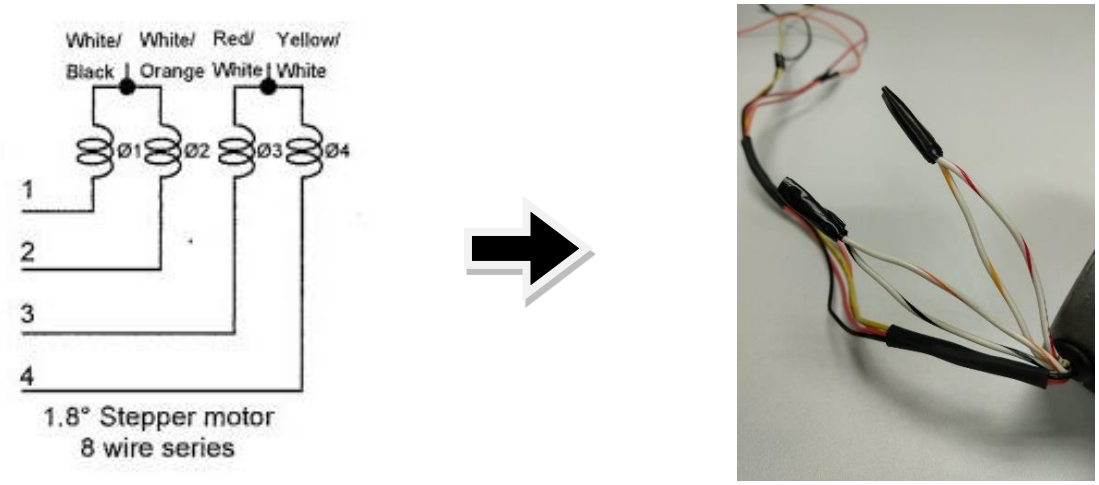


Ilustración 4.2.10 Conexión de cableado en serie del motor PAP.

Como se ha podido comprobar el voltaje nominal para el motor es de 5 V mientras que el voltaje mínimo para el control mediante el driver "X-NUCLEO-IHM01A1" es de 8 V. Para un buen funcionamiento se ha optado por utilizar un voltaje fijo superior a 8 V, incorporando un par de resistencias de 1Ω en dos de los cuatro devanados del motor, con el fin de reducir el amperaje que podría ser dañino para el mismo.

Ilustración 4.2.11 Resistencias para la reducción del amperaje del motor PAP.

4.2.5. Taladro.

El sistema también cuenta con otro motor, pero en este caso será un **motor de corriente continua de imanes permanentes** que realizará la función del taladro. El motor que se ha utilizado es el Spindle 775 y presenta las siguientes características:

- Voltaje nominal 12…36 V
- Corriente de carga a 12V de 1.7 A (pico de inicio de 2.2 A)
- Velocidad alrededor de 15000 rpm (en función del voltaje de entrada)

- Peso 356 g.
- Dimensiones: 98x42 mm

Ilustración 4.2.12 Motor de continua Spindle 775.

Unido a su eje ira conectado una pieza de sujeción de brocas, específicamente la ER11:

Ilustración 4.2.13 Campana de sujeción de brocas ER11.

Para la protección del motor frente a inversiones de polaridad en la corriente se ha instalado un diodo semiconductor DIOTEC “BY 251”, que presenta las siguientes características:

- Tensión de retorno máxima 200 V
- Corriente conductiva máxima 20 A

Ilustración 4.2.14 Conexión de diodo en anti-paralelo al motor de corriente continua.

Poniendo el **diodo en anti-paralelo** al motor, se deriva la intensidad a través del diodo y del propio motor. De esta forma se reduce notablemente la chispa en el contacto o incluso llega a desaparecer.

4.2.6. Relé.

Para el control del taladro es necesario la aplicación de un **relé**, que suministre la corriente a partir de las órdenes del microcontrolador. El módulo relé a utilizar es de la marca Sunfounder, específicamente el “SRD-05VDC-SL-C”, que presenta las siguientes características:

- Voltaje de trabajo 5 V.
- Control de soporte de señales CC y CA (carga de 220 CA).
- Luz de encendido e indicador de señal.
- Conexión con cable anti-reversa de 3 pines.
- Dimensiones pcb: 2x4.3 cm

Ilustración 4.2.15 Modulo relé Sunfounder SRD-05VDC-SL-C.

En un **filtro R-C**, el condensador se encarga de absorber la intensidad de la sobretensión que se genera al abrir el contacto. Así se trata de proteger los contactos del relé.

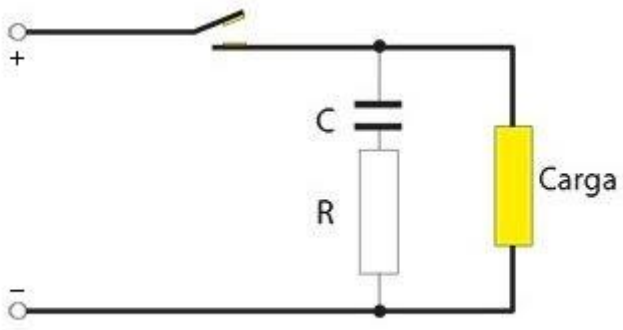


Ilustración 4.2.16 Filtro R-C.

4.2.7. Finales de carrera y circuito anti rebote.

Los **finales de carrera** a utilizar han sido designados debido a su tamaño, y por tanto serán distintos en función del eje sobre el que vayan colocados. Se pueden encontrar finales de carrera de la marca OMRON, los "V-156-1C25", para el eje X y de la marca CHERRY, los "DC1C-A1RC", para el eje Y. Presentan las siguientes especificaciones técnicas:

- **EJE X**

  - Valor de corriente 15 A
  - Voltaje máximo 250VAC
  - Palanca con rodillo de bisagra corta.
  - Fuerza máxima de operación 1.23 N

Ilustración 4.2.17 Final de carrera OMRON V-155-1C25.

- **EJE Y**

  - Valor de corriente 6 A
  - Voltaje máximo 250VAC

- Palanca con rodillo de bisagra corta.
- Fuerza máxima de operación 0.8 N

Ilustración 4.2.18 Final de carrera CHERRY DC1C-A1RC.

Debido a que este tipo de componentes suele producir imprecisiones debidas a mal contacto, ruido en la señal o retornos, se ha optado por la colocación de un **circuito anti rebote**, que filtrará la señal del final de carrera antes de ser enviada al microcontrolador. Se ha diseñado un circuito con el integrador SN74L compuesto por 4 puertas NAND que evitarán el rebote de la señal. El circuito es alimentado con 5V que provendrán del microcontrolador y tiene la capacidad de controlar dos finales de carrera, por lo que se necesitarán dos circuitos, uno por cada eje.

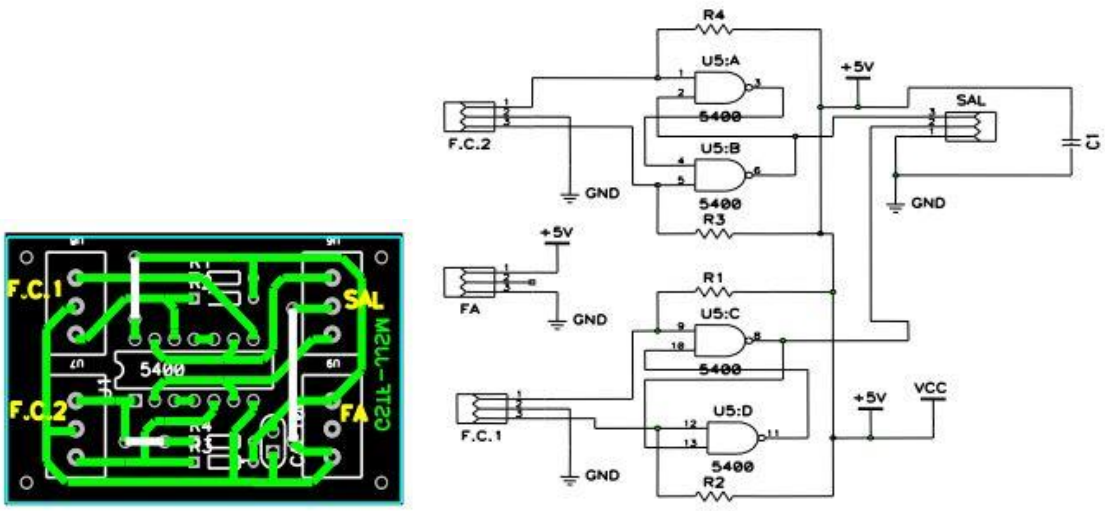


Ilustración 4.2.19 Circuito impreso y conexiones del circuito anti rebote.

4.2.8. Convertidor USB/serie.

Las comunicaciones serie UART permite enviar datos entre diferentes circuitos a velocidades relativamente altas y de forma sencilla. Para la comunicación entre el HOST, la CPU, con el microcontrolador va a ser necesario la utilización de un **convertidor USB/serie**, concretamente

el chip FTDi 232R. Para la comprobación del envío correcto de datos, se enviarán también al software de comunicación serie. Sus especificaciones técnicas son las siguientes:

- Conexión mini-USB.
- EEPROM de 1024 bits
- Protocolo USB completo manejado en el chip
- Tasas de transferencia de datos de 300 baudios a 3 Mbaudios
- Búfer de recepción de 128 bytes y búfer de transmisión de 256 bytes que utiliza tecnología de suavizado de búfer para permitir un alto rendimiento de datos.
- Leds de transmisión y recepción de señal

Ilustración 4.2.20 Convertidor serie FTDi232R

Para realizar una buena comunicación se deben configurar una serie de parámetros:

- **Baudrate**: consisten en el número de unidades de señal por segundo, es decir, la velocidad expresada en bits/s. Para una buena comunicación debe ser igual en ambos componentes conectados.
- **Paridad**: es un dígito binario que indica si el número de bits con un valor de 1 en un conjunto de bits es par o impar. Lo bits de paridad conforman el método de detección de errores más simple.
- **Data bits**: consiste en la longitud del paquete de datos de información que se envían cada vez. Comúnmente este tamaño es el de un byte.
- **Stop bits**: Son los bits de parada, que indican el final del paquete de información enviado.
- **Control de flujo**: puede ser por hardware o por software.

4.3. Software.

En este capítulo se va a exponer el software utilizado para el diseño y programación del sistema. Se explicará el funcionamiento de cada uno de los programas.

4.3.1. SolidWorks.

SolidWorks es un software de diseño asistido por computador (CAD) para modelado mecánico en 2D y 3D. El programa permite modelar piezas y conjuntos, para extraer posteriormente de ellos planos técnicos además de todo tipo de información para su producción.

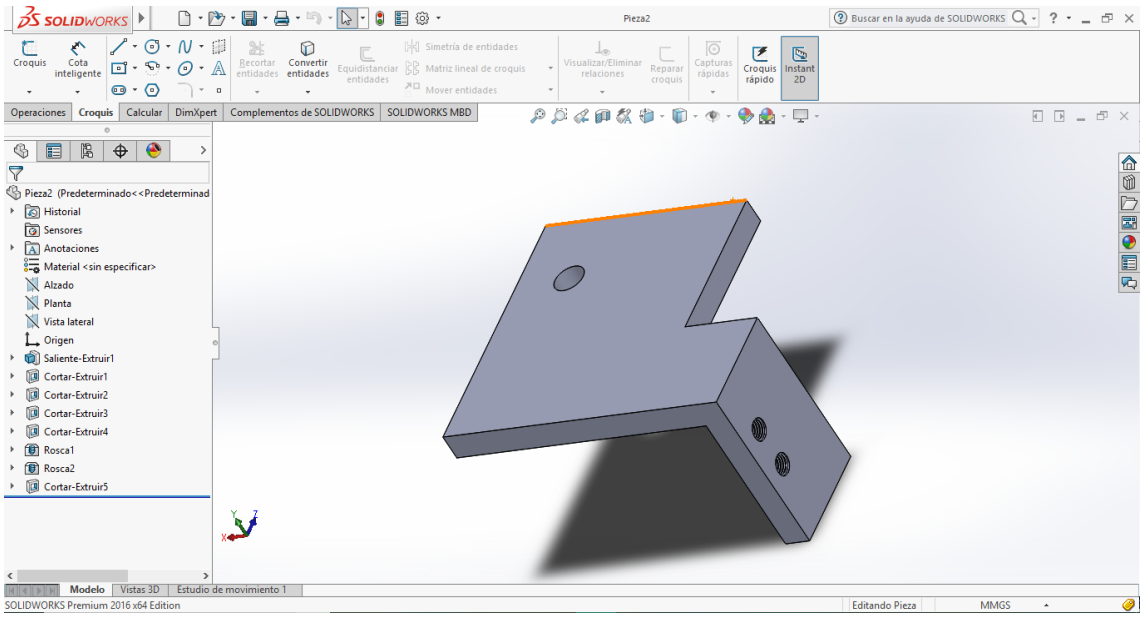


Ilustración 4.3.1 Interface del programa SolidWorks.

A partir de este programa se han diseñado las piezas que forman parte de la estructura del sistema y que posteriormente han sido creadas por una impresora 3D. Para su impresión se debe crear el fichero en formato .STL, que es el que reconoce la impresora.

4.3.2. STM32-CubeMX.

CubeMX es un programa que contiene un conjunto de herramientas que facilitan y agilizan el desarrollo de una aplicación en una plataforma de STM32, con el fin de simplificar y acelerar el trabajo de los desarrolladores.

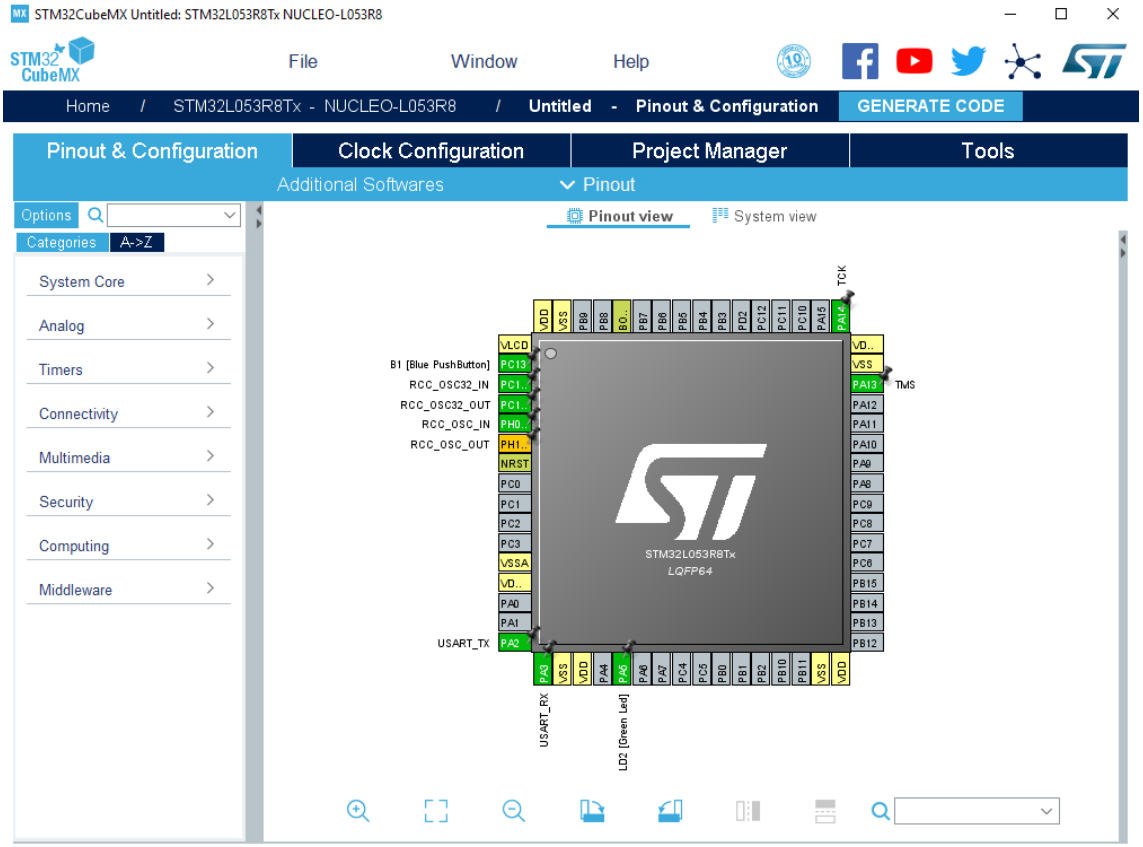


Ilustración 4.3.2 Interface del programa STM32-CubeMX.

Con la ayuda de este programa se ha creado una base de forma rápida, sobre la cual se realiza la programación de la aplicación.

4.3.3. arm Keil - µVision IDE.

µVision IDE es un programa capaz de combinar la gestión de proyectos, el entorno de ejecución, las instalaciones de compilación, la edición del código fuente y la depuración de programas en un único entorno. En dicho entorno se puede probar, verificar y optimizar el código para una aplicación.

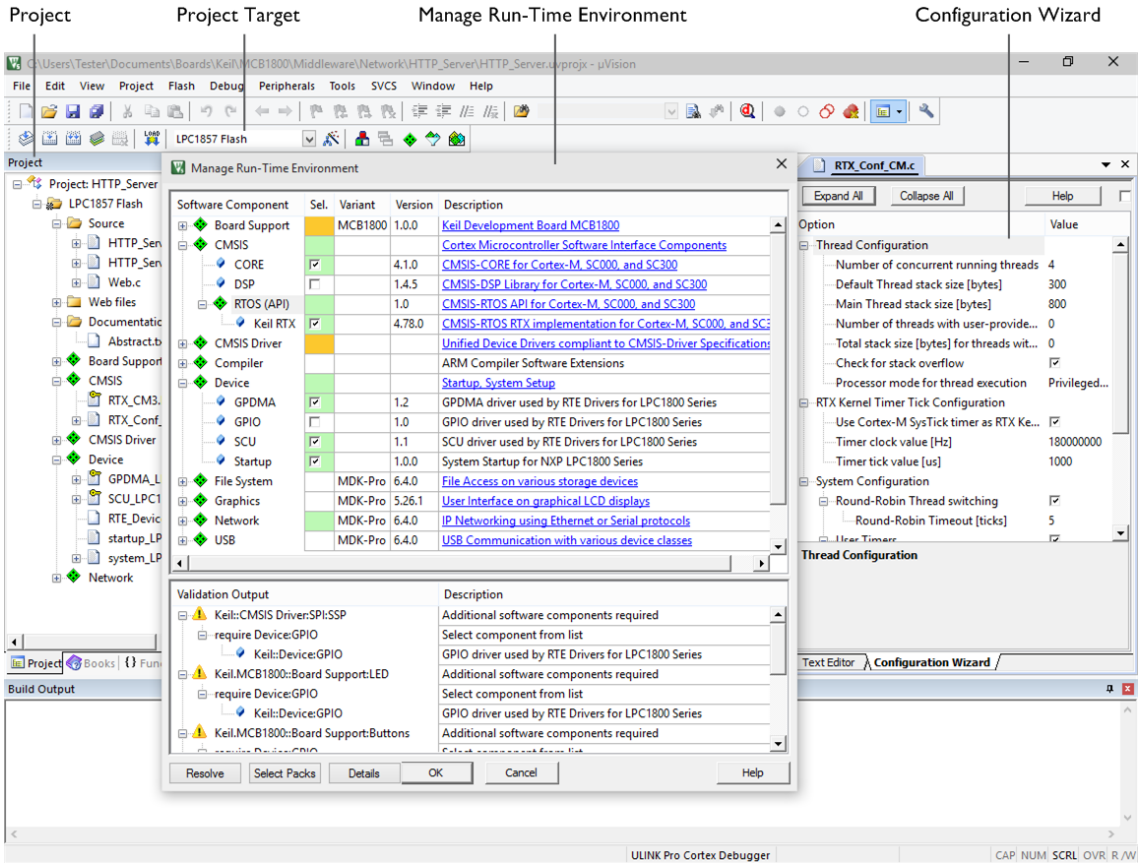


Ilustración 4.3.3 Interface del programa µVision IDE.

A través de dicha aplicación se va a desarrollar el intérprete que analice los comandos “G-code” y ordene el movimiento a los motores del sistema.

4.3.4. Txapu-CNC.

Txapu-CNC es un programa diseñado para una fresadora, capaz de realizar mecanizados sobre una superficie (X,Y) a distintas alturas (Z). El programa se puede dividir en dos partes, una cuya función es la de edición y transmisión de “G-codes”, y otra cuya función es de recepción e interpretación de dichos “G-codes”.

Para el caso que acontece solo va a ser necesaria la parte de edición y transmisión (TxapuCNC_TX), dado que la recepción e interpretación (TxapuCNC_RX) está programada para Arduino. En el trabajo se ha programado dicha parte con el software anteriormente mencionado (µVision IDE).

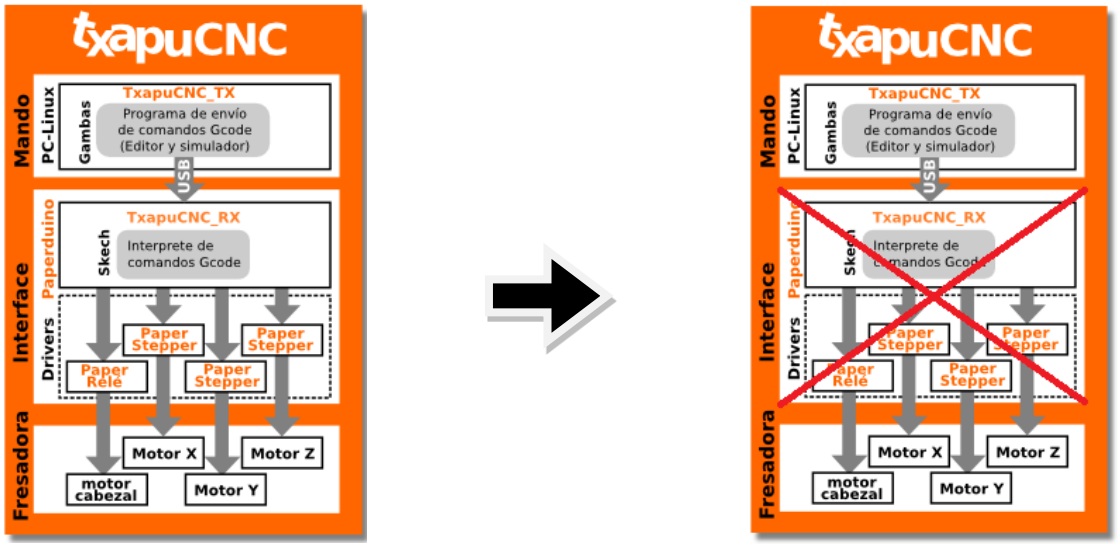


Ilustración 4.3.4 Orden de mando del programa TxapuCNC original frente al utilizado por el sistema.

TxapuCNC_TX es un programa realizado en gambas (Linux) que consta de un editor de programas “G-code”, un simulador para ver visualmente el funcionamiento de los programas y un subprograma de envío de dichos comandos “G-code” a la máquina de control numérico.

- **Editor de programas**:
  Es un editor de textos con resaltado para comandos “G-code”, que envía por el puerto serie una a una las líneas del editor, esperando previamente a recibir una señal de “OK” del microcontrolador.
- **Simulador**:
  Es un visualizador de programas “G-code” (2D) que muestra el recorrido en el plano XY. Dispone de zoom, desplazamiento, simulación rápida y con retardo…
- **Control**:
  Es una consola de mando desde donde puedes manejar directamente la máquina de control numérico.

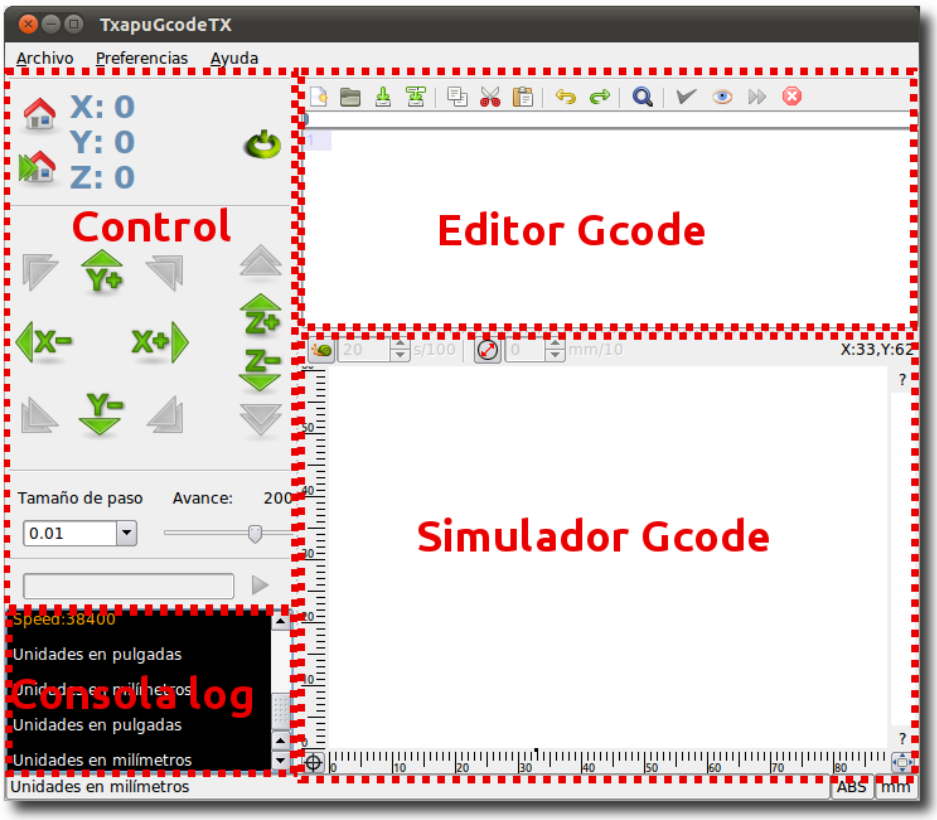


Ilustración 4.3.5 Interface del programa TxapuCNC.

A continuación se muestra una tabla con todos los comandos reconocidos por el programa "Txapuzas CNC":

Comando	Ejemplo	Descripción	TX	RX
G0	G0 X10	Movimiento lineal Rápido	Si	Si
G1,G01	G1 X10 Y15 Z0 [F100]	Movimiento lineal Controlado (Avance: 100)	Si	Si
G2,G02	G02 X60 Y30 I30 J-10 F02	Movimiento curvo (sentido horario) Controlado	Si	Si
G3,G03	G03 X60 Y30 I10 J20	Movimiento curvo (antihorario) Controlado	Si	Si
G4,G04	G4 P200	Pausa con retardo (Retardo: 200ms)	Si	Si
G20	G20	Definir Unidades en Pulgadas	Si	Si
G21	G21	Definir Unidades en milímetros	Si	Si
G28	G28	Ir a Origen	Si	Si
G30	G30 X10 Y20 Z30	Ir a Origen a través de un punto	Si	Si
G90	G90	Definir Coordenadas absolutas	Si	Si
G91	G91	Definir Coordenadas relativas	Si	Si
G92	G92	Definir punto actual como origen	Si	Si
M0	M0	Paro (Pausa programada)	Si	No
M3,M03	M3	Marcha del cabezal	Si	Si
M5,M05	M5	Paro del cabezal	Si	Si

Tabla 4.3.1 Comandos G reconocidos por el programa TxapuCNC.

5. DESCRIPCIÓN DEL SOFTWARE: DESARROLLO Y PROGRAMACIÓN

5.1. Arquitectura del programa.

Con el fin de hacer portables y fáciles de implementar a los sistemas basados en microcontroladores ARM, estos vienen acompañados de un software genérico. La solución de firmware de STM32Cube está construida alrededor de tres niveles totalmente independientes que pueden interactuar entre ellos de forma muy fácil, como se puede ver en el diagrama siguiente:

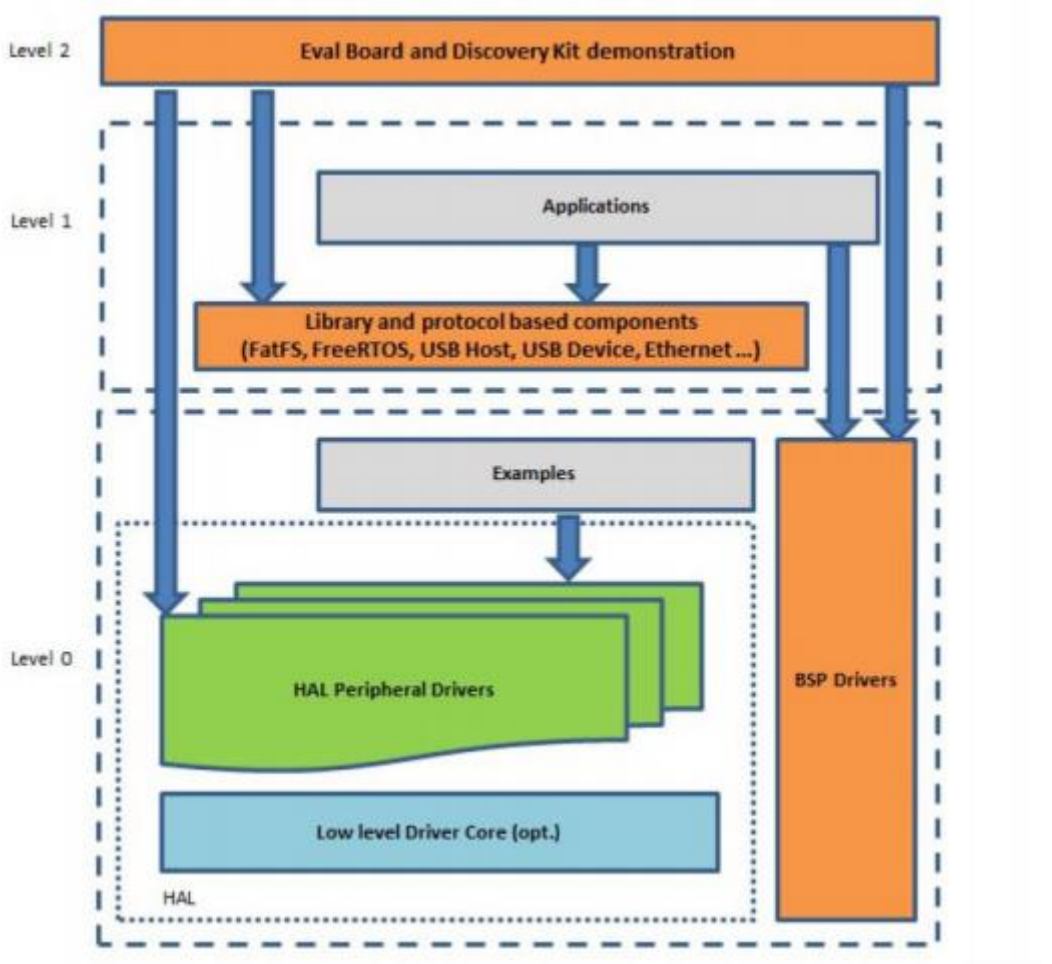


Diagrama 5.1.1 Arquitectura por niveles del software de STM32Cube.

- **Nivel 0**: este nivel se encuentra divido en tres sub-capas, que son:

    1. **Board Support Package (BSP)**: esta capa ofrece un conjunto de APIs (relación entre dos aplicaciones para el intercambio de datos) relativas a los componentes de hardware encontrados en las placas (puertos E/S, conexión de audio, pantalla táctil, controlador SRAM, controlador LCD, etc…).

    2. **Hardware Abstraction Layer (HAL)**: esta capa proporciona los controladores de bajo nivel y los métodos de interfaz de hardware para interactuar con las capas superiores (aplicaciones, librerías…)

    3. **Ejemplos de uso de periféricos básicos**: esta capa aloja los ejemplos construidos alrededor de los periféricos STM32 utilizando solo los recursos HAL y BSP.

- **Nivel 1**: este nivel se encuentra dividido en dos sub-capas, que son:

    1. **Middleware components**: conjunto de bibliotecas para dispositivos USB, STemWin, FreeRTOS, FatFS, LwIP y PolarSSL.

    2. **Ejemplos basados en los componentes del middleware**: Cada componente de middleware viene con uno o más ejemplos (o aplicaciones) que muestran cómo usarlo.

- **Nivel 2**: Este nivel es una capa única con una demostración global, en tiempo real y gráfica basada en la capa de servicio de middleware, la capa de abstracción de bajo nivel y las aplicaciones de uso de periféricos básicos para funciones basadas en placa.

Para la placa del driver también disponemos de una serie de funciones totalmente compatibles con el software que acabamos de comentar. El software del sistema es una expansión para STM32Cube y, como tal, cumple totalmente con la arquitectura de STM32Cube y lo expande para permitir el desarrollo de aplicaciones que utilizan controladores de motores paso a paso basados en el componente L6474.

Combinando el software de los drivers con el software implementado con el microcontrolador (interprete de comandos) se obtiene un **GRBL propio**.

Las capas utilizadas por el software de la aplicación para acceder y utilizar la placa de expansión del controlador del motor son:

- **STM32Cube HAL layer**: proporciona un conjunto simple genérico de múltiples instancias de API (interfaces de programación de aplicaciones) para interactuar con las capas superiores (aplicaciones, bibliotecas y pilas).

- **Based support package (BSP) layer**: proporciona soporte para todos los periféricos en la placa Nucleo STM32, aparte de la MCU.

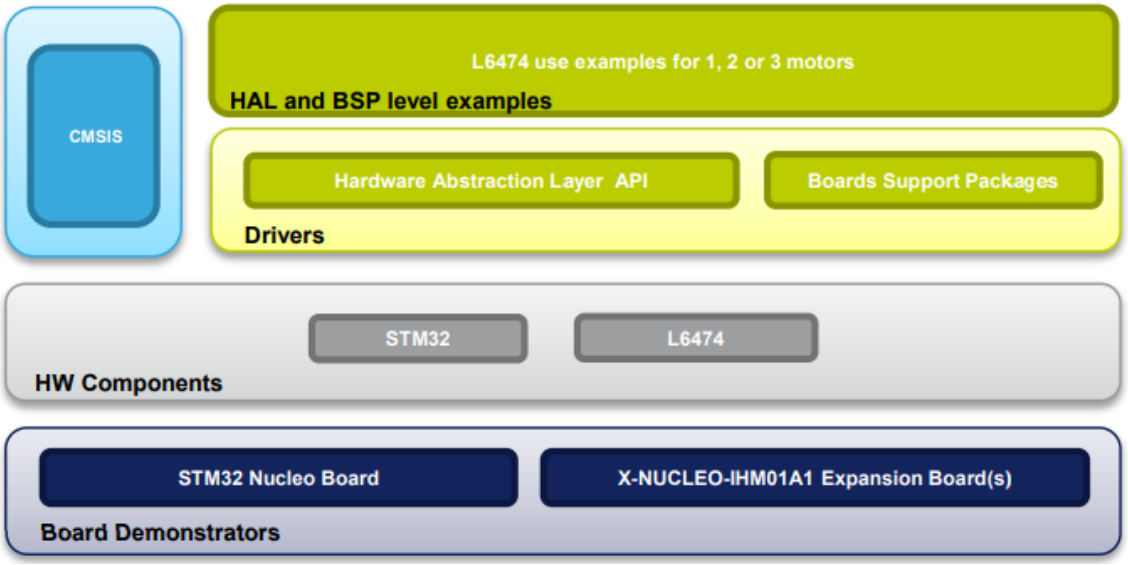


Diagrama 5.1.2 Arquitectura de un microcontrolador STM32 de la familia Nucleo junto a los drivers controladores IHM01A1.

5.2. Configuración STM32Cube.

Para la utilización de las funciones del microcontrolador se deben de configurar los pines a utilizar previamente. A continuación se describe la configuración que se ha optado para el sistema.

5.2.1. Pines E/S.

Para el control del sistema es necesario el uso de entradas y salidas digitales, que informen del estado del sistema y a su vez produzcan una respuesta en caso necesario.

El sistema presenta cuatro entrada digitales, que se corresponden con los estados de los finales de carrera, es decir, dos por eje. Indican cuando se ha alcanzado el límite del desplazamiento sobre el eje. Los pines del microcontrolador usados para ello son:

PIN	FUNCIÓN
PC 11	GPIO Exti interrupt → Entrada final de carrera 1 EJE Y
PC 12	GPIO Exti interrupt → Entrada final de carrera 2 EJE Y
PC 2	GPIO Exti interrupt → Entrada final de carrera 1 EJE X
PC 3	GPIO Exti interrupt → Entrada final de carrera 2 EJE X

Tabla 5.2.1 Entradas digitales del microcontrolador.

En cuanto a las salidas, se han utilizado para diferentes funciones.

Dos salidas digitales serán las encargadas de controlar el sentido de giro de cada uno de los motores. Los pines utilizados son:

PIN	FUNCIÓN
PA 8	Dirección motor 1
PB 5	Dirección motor 2

Tabla 5.2.2 Salidas digitales del microcontrolador.

Una salida digital se corresponderá con la señal necesaria para indicar al relé su actuación sobre el taladro. El pin utilizado es:

PIN	FUNCIÓN
PC 8	GPIO Output → Salida relé

Tabla 5.2.3 Salidas digitales del microcontrolador.

Finalmente dos salidas digitales se utilizaran para el “flag interrupt”, que es un bit encargado de indicar al sistema la interrupción, y para iniciar un reseteo del chip L6474 del driver. Los pines utilizados son:

PIN	FUNCIÓN
PA 10	Flag interrupt
PA 9	L6474 reset

Tabla 5.2.4 Salidas digitales del microcontrolador.

5.2.2. Temporizadores y generación de PWM.

Como se ha visto en el apartado 3.5.1, para el movimiento de los motores PAP es necesario enviar una serie de pulsos que se transformaran en movimientos angulares. El conjunto de pulsos

recibe el nombre de **señal PWM** (pulse width modulation), que indicará el número de pasos a realizar por el motor.

Estas señales es preciso que estén enlazadas y controladas por un “timer”, dado que así se puede controlar tanto la frecuencia en el envío de pulsos como la velocidad y posición de los motores. Por tanto cada señal PWM vendrá controlada por su propio “timer”.

En el sistema se han de controlar dos motores, es por ello que se deben usar dos “timers” para la creación de ambas señales PWM. Los pines utilizados por el microcontrolador son:

PIN	FUNCIÓN
PC 7	PWM 1 → Timer 22 Channel 2
PB 3	PWM 2 → Timer 2 Channel 2

Tabla 5.2.5 Timers utilizados por el microcontrolador.

5.2.3. Comunicación UART y SPI.

La comunicación con el PC se realiza a través del puerto serie, es decir, empleando la **comunicación USART** (Universal Asynchronus Receiver-Transmitter) entre el microcontrolador y el convertidor FTDi 232R. Los datos que contienen los comandos G generados para el sistema circulan mediante esta conexión.

Este método de comunicación necesita del uso de dos señales:

- **Transmisión** (TX): salida de datos.
- **Recepción** (RX): entrada de datos.

La transmisión (TX) de datos del microcontrolador debe conectarse con la recepción (RX) de datos del convertidor y viceversa. Los pines utilizados por el microcontrolador son:

PIN	FUNCIÓN
PA 2	Comunicación USART 2 TX
PA 3	Comunicación USART 2 RX
PC 4	Comunicación USART 1 TX
PC 5	Comunicación USART 1 RX

Tabla 5.2.6 Pines utilizados para la comunicación USART del microcontrolador.

En cuanto a la comunicación SPI, es utilizada para la comunicación del microcontrolador con los drivers “X-NUCLEO IHM01A1” de los motores PAP. Esta comunicación es realizada mediante el bus SPI (Serial Peripheral Interface).

Para ello se emplea una señal de reloj, el envío de datos, la recepción de datos y una conexión correspondiente a la selección de chip, que conecta o desconecta la operación del dispositivo con el que se desea comunicar.

Con cada pulso del reloj el maestro envía una cadena de bits, cuyo destino será el esclavo que previamente haya seleccionado el chip de selección.

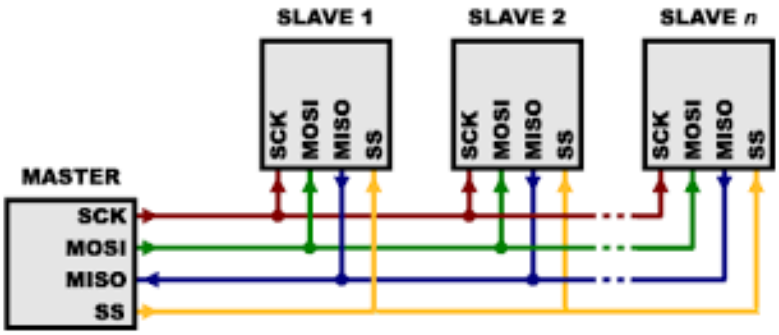


Ilustración 5.2.1 Comunicación SPI entre el microcontrolador y el resto de dispositivos.

Las funciones de estas 4 conexiones son:

- **SCK (clock)**: señal de reloj del bus que marca la sincronización. Con cada pulso se indica el envío o lectura de un bit.
- **MOSI (master output slave input)**: salida de datos del maestro y entrada de datos al esclavo.
- **MISO (master input slave output)**: entrada de datos al maestro y salida de datos del esclavo.
- **SS/CS (chip select)**: selecciona un esclavo para el envío de datos.

Este tipo de conexión permite al maestro comunicarse a través del mismo bus con varios esclavos y decidir sobre cual actuar. Es ideal para el sistema, ya que el microcontrolador actuará de maestro y los drivers de los motores PAP serán esclavos. Los pines usados por el microcontrolador son:

PIN	FUNCIÓN
PB 6	CS: chip select
PA 7	Comunicación SPI 1 Salida de datos maestro, entrada de datos esclavo (MOSI)
PA 6	Comunicación SPI 1 Entrada de datos maestro, salida de datos esclavo (MISO)
PA 5	Señal de reloj del bus SPI (SCK → serial clock)

Tabla 5.2.7 Pines utilizados por el microcontrolador para la comunicación SPI.

5.2.4. Resumen.

En la siguiente tabla se resumen todos los pines utilizados por el microcontrolador para el funcionamiento del sistema:

PIN	FUNCIÓN
PC 11	GPIO Exti interrupt → Entrada final de carrera 1 EJE Y
PC 12	GPIO Exti interrupt → Entrada final de carrera 2 EJE Y
PC 2	GPIO Exti interrupt → Entrada final de carrera 1 EJE X
PC 3	GPIO Exti interrupt → Entrada final de carrera 2 EJE X
PA 8	Dirección motor 1
PB 5	Dirección motor 2
PC 8	GPIO Output → Salida relé
PA 10	Flag interrupt
PA 9	L6474 reset
PC 7	PWM 1 → Timer 22 Channel 2
PB 3	PWM 2 → Timer 2 Channel 2
PA 2	Comunicación USART 2 TX
PA 3	Comunicación USART 2 RX
PC 4	Comunicación USART 1 TX
PC 5	Comunicación USART 1 RX
PB 6	CS: chip select
PA 7	Comunicación SPI 1 Salida de datos maestro, entrada de datos esclavo (MOSI)
PA 6	Comunicación SPI 1 Entrada de datos maestro, salida de datos esclavo (MISO)
PA 5	Señal de reloj del bus SPI (SCK → serial clock)

Tabla 5.2.8 Resumen de todos los pines utilizados por el microcontrolador.

5.3. Programación en Keil.

En este capítulo se va a exponer detalladamente toda la programación realizada para la aplicación del sistema.

5.3.1. Organización proyecto.

El proyecto está organizado por carpetas las cuales pertenecen tanto al driver como al microcontrolador. Las carpetas que contienen el código necesario para el funcionamiento de los drivers no serán editadas, pero si configuradas para el uso de los motores seleccionados.

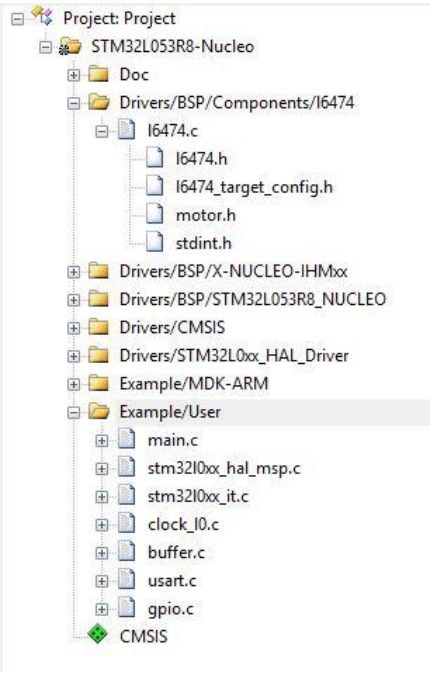


Ilustración 5.3.1 Organización de carpetas y ficheros del proyecto.

Como se observa en la imagen anterior, existen tres carpetas principales, que son:

- **Doc**: donde se almacena documentación sobre el proyecto.
- **Drivers**: donde se almacena todo el código de control de los drivers "X-NUCLEO-IHM01A1"
- **Example**: donde se almacena el código del microcontrolador que será desarrollado por el usuario.

Dentro de la carpeta Example/User se encuentra el programa principal (main.c) y el resto de ficheros implementados por el usuario de los cuales se sirve el programa para el funcionamiento del sistema (buffer.c, usart.c, gpio.c).

5.3.2. Configuración periféricos.

La configuración de los periféricos se encuentra en el fichero gpio.c para las E/S de los GPIO y en el fichero uart.c para las conexiones del puerto serie con el microcontrolador.

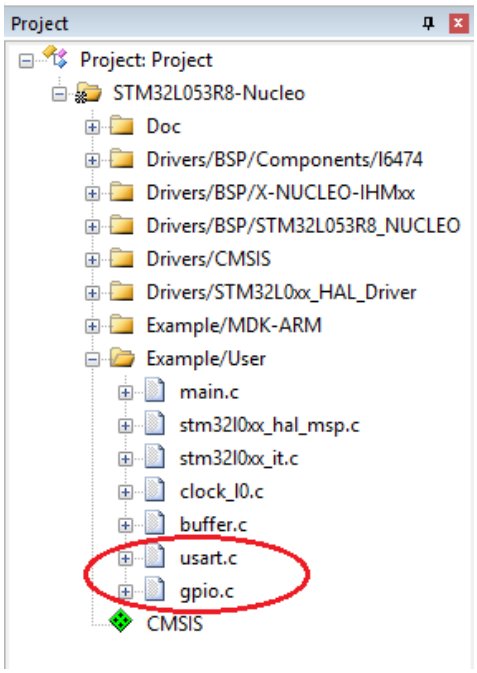


Ilustración 5.3.2 Localización de los ficheros para la configuración de periféricos.

La configuración de los pines del GPIO es:

- **GPIO PC8** → modo salida
- **GPIO PC11 y PC12** → modo interrupción en caída, es decir, activa la interrupción cuando la señal enviada realiza una bajada de la tensión.
- **GPIO PC2 y PC3** → modo interrupción en caída.

La configuración de los puertos serie es:

1. **Hlpuart1(comunicación con software serie)**
    - Baudrate: 38400 bits/s
    - Paridad: no
    - Data bits: 8 bits (1 byte)
    - Stop bits: 1 bit
    - Control de flujo: no
2. **Huart2 (comunicación con software Txapu CNC)**
    - Baudrate: 19200 bits/s

- Paridad: no
- Data bits: 8 bits (1 byte)
- Stop bits: 1 bit
- Control de flujo: no

5.3.3. Configuración motores.

La configuración de los parámetros de cada uno de los motores se realiza dentro del fichero "l6474_target_config.h", en el cual se ajusta la aceleración, velocidad y par de los motores, el tipo de señal PWM a utilizar, entre otras cosas.

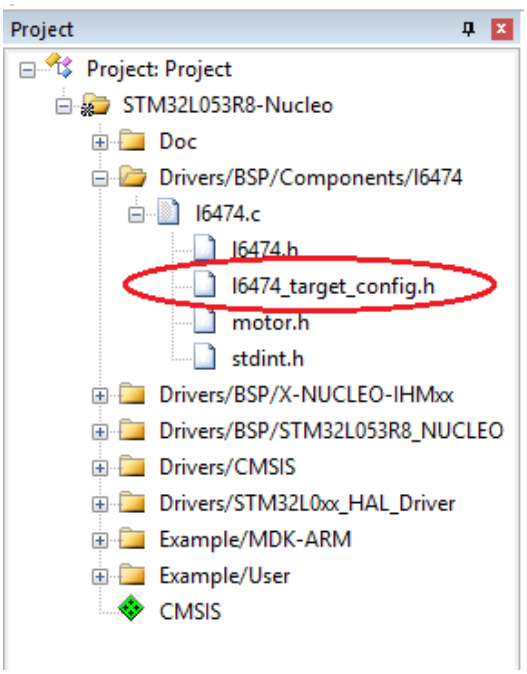


Ilustración 5.3.3 Localización del fichero para la configuración de los motores PAP.

Los drivers usados tienen la capacidad de controlar hasta tres motores, pero en el caso del sistema solo se van a utilizar dos, es por ello que se configurará para que el número máximo de dispositivos a usar sea dos. En cuanto a los motores, se han configurado con los siguientes parámetros:

- **Velocidad máxima** (motor1 y motor2): 400 pasos/s
- **Velocidad mínima** (motor1 y motor2): 300 pasos/s
- **Regulación de par DAC** (motor1 y motor2): 500m A
- **Selección de pasos** (motor1 y motor2): 1/16 → "microstepping"

El resto de parámetros se han dejado en sus valores predeterminados.

5.3.4. Funciones driver.

Como se ha comentado con anterioridad, los drivers del sistema presentan su propio código de funcionamiento. En ese código se encuentran una serie de funciones básicas para el control y movimiento de los motores. Estas funciones se encuentran implementadas en el fichero "x-nucleo-ihmxx.c".

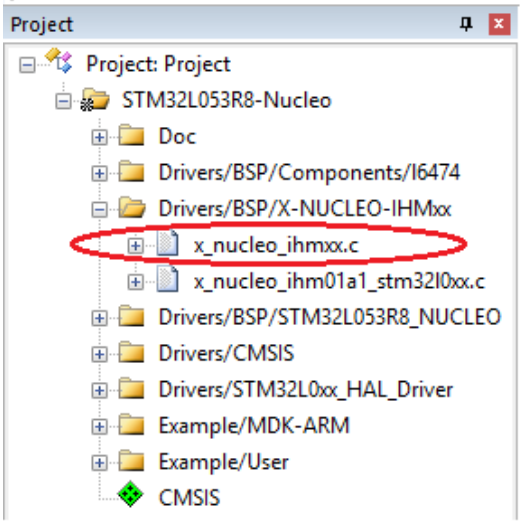


Ilustración 5.3.4 Localización del fichero con las funciones de control de motores PAP de los drivers.

A continuación se van a explicar las principales funciones utilizadas en el proyecto:

- **BSP_MotorControl_GetPosition** (uint8_t deviceId): devuelve el valor de la posición en la que se encuentra el motor.
- **BSP_MotorControl_SetHome** (uint8_t deviceId, int32_t homePosition): establece la posición indicada como posición de inicio (home).
- **BSP_MotorControl_GoHome** (uint8_t deviceId): mueve al motor a la posición de inicio (home).
- **BSP_MotorControl_GoTo** (uint8_t deviceId, int32_t targetPosition): solicita al motor que se desplace a la posición especificada.
- **BSP_MotorControl_Move** (uint8_t deviceId, motorDir_t direction, uint32_t stepCount): mueve el motor el número de pasos especificado.
- **BSP_MotorControl_Run** (uint8_t deviceId, motorDir_t direction): mueve el motor desde la velocidad mínima hasta alcanzar la velocidad máxima usando la aceleración del motor. No tiene destino marcado, seguirá moviéndose mientras no se indique lo contrario.
- **BSP_MotorControl_WaitWhileActive** (uint8_t deviceId): espera mientras el motor este realizando un movimiento.

- **BSP_MotorControl_CmdDisable** (uint8_t deviceId): desactiva el controlador del motor.
- **BSP_MotorControl_HardStop && BSP_MotorControl_SoftStop** (uint8_t deviceId): realiza una parada del motor. En modo hard la parada será instantánea mientras que en modo soft la parada se realiza usando la deceleración del motor.
- **BSP_MotorControl_SetMaxSpeed && BSP_MotorControl_SetMinSpeed** (uint8_t deviceId): establece la velocidad máxima y mínima del motor con el valor indicado.

Variables

deviceId → número de identificación de dispositivo (0…1)

direction → dirección de movimiento (FORWARD/BACKWARD)

homePosition → posición de inicio

targetPosition → posición deseada

stepCount → número de pasos

5.3.5. Programa principal.

El programa principal se encuentra en el fichero “main.c” y se sirve del resto de ficheros para configurar y aplicar funciones. A continuación se realizará una explicación de las secciones del programa principal.

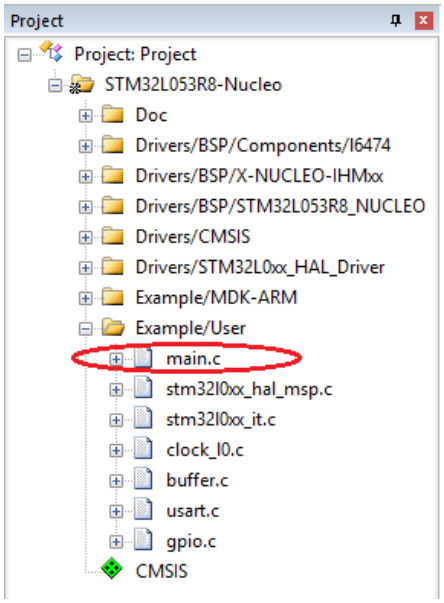


Ilustración 5.3.5 Localización del fichero con el programa principal.

El primer pasos será definir las variable a utilizar durante el proceso. Una vez definidas se activarán las librerías, se configurará el reloj del sistema, se configurará el uso de dos drivers y se inicializarán los periféricos (GPIO, USART).

Seguidamente se prepara el sistema para su uso, pero antes de empezar se avisa al usuario que espere mientras se prepara. Se activa la función de interrupción para la recepción de datos a través del puerto serie, se realiza un movimiento de "homing" de los motores y por protección se pone en modo reset el relé que controla al taladro. Una vez preparado el sistema se avisa al usuario de que puede empezar con el envío de comandos.

Finalmente, se entra en un bucle infinito, en el cual se analizan los comando recibidos por el puerto serie y se realizan una serie de acciones pertinentes, en función del tipo de comando recibido. Para indicar que el comando ha sido finalizado se envía un "OK" al software "Txapu CNC".

En el ANEXO A se encuentran las imágenes con toda la programación del sistema

5.3.6. Funciones implementadas.

A continuación se van a exponer las funciones implementadas en el programa principal, que sirven para analizar el comando recibido y proporcionar una respuesta.

La función "**Proceso_Letras**" se ha creado con el fin de identificar un carácter especificado por el usuario dentro de una cadena de caracteres y devolver los caracteres posteriores hasta encontrarse con un espacio en blanco, un salto de línea o un fin de cadena. La cadena a identificar será el comando recibido a través del puerto serie. En dicho comando aparecerán separados por espacios los distintos parámetros a tener en cuenta (desplazamiento X, desplazamiento Y, velocidad F, profundidad D, pausa P).

Con la función "**Proceso_Valores**" se obtendrán los valores pertenecientes a los parámetros de la cadena que contiene el comando recibido por el puerto serie y se almacenarán en sus pertinentes variables. Dicha función se hace servir de la función "Proceso_Letras" para obtener el valor de los parámetros, para seguidamente transformar dichos valores. Las transformaciones que realiza son las siguientes:

- Desplazamiento X/Y y profundidad D en milímetros/pulgadas a número de pasos.
- Definir el sentido de giro del motor y convertir los valores negativos para coordenadas absolutas.
- Pasar el tiempo de espera a milisegundos.
- Configurar las velocidades del motor.

La función "**Proceso_Comando**" sirve de intérprete del sistema, es decir, en ella se encuentran definidos todos los comandos a usar por el sistema. La función analiza la cadena recibida y

dependiendo del comando recibido realizará una serie de acciones que son las que definen al comando. Previamente ya han sido almacenados los valores de los parámetros del comando en sus pertinentes variables.

Ahora se procede a comentar las funciones del programa principal que son usadas para la comunicación con el pc vía USART a través del puerto serie.

La función "**Mensa_Debug_HLPUART1**" sirve para transmitir mensajes a través del puerto serie y que serán recibidos por el software de recepción. Se hace servir de la instancia hlpuart1.

La función "**USART2_PutsString**" realiza la misma función que la anterior pero a través de la instancia huart2. En este caso los mensajes serán enviados al software "Txapu CNC".

Además la función "**HAL_UART_RxCpltCallback**" se usa para recibir datos de otros dispositivos y verificar esos datos. Se hace servir de la instancia huart2 para recibir el comando, enviado por el software "Txapu CNC", a través del puerto serie, almacenarlo en el Rx_buffer y volver a activar la interrupción en modo recepción para futuros mensajes.

Otra de las funciones implementada, pero que no tiene nada que ver con las mencionadas anteriormente, es la función "**Movimiento_Homing"**, que realiza el movimiento de los motores a una posición inicial y posteriormente guarda dicha posición como punto de partida, inicio o "home", para que sea posible su vuelta siempre que sea necesario.

En el ANEXO A se encuentran las imágenes con toda la programación del sistema

5.3.7. Interrupción final de carrera.

El uso de los finales de carrera es para delimitar el límite de desplazamiento sobre un eje, es por ello, que siempre deben estar preparados en caso de que se activen. Por tanto, se ha creado una interrupción la cual en función del final de carrera accionado, detendrá el motor del eje que limita y posteriormente realizará un movimiento de desactivación del mismo o de "homing". Se puede encontrar implementada dicha interrupción en el fichero "stm32l0xx_hal_msp.c".

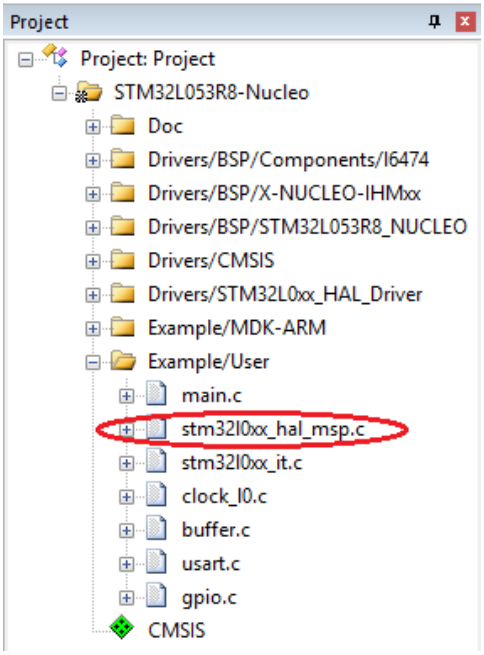
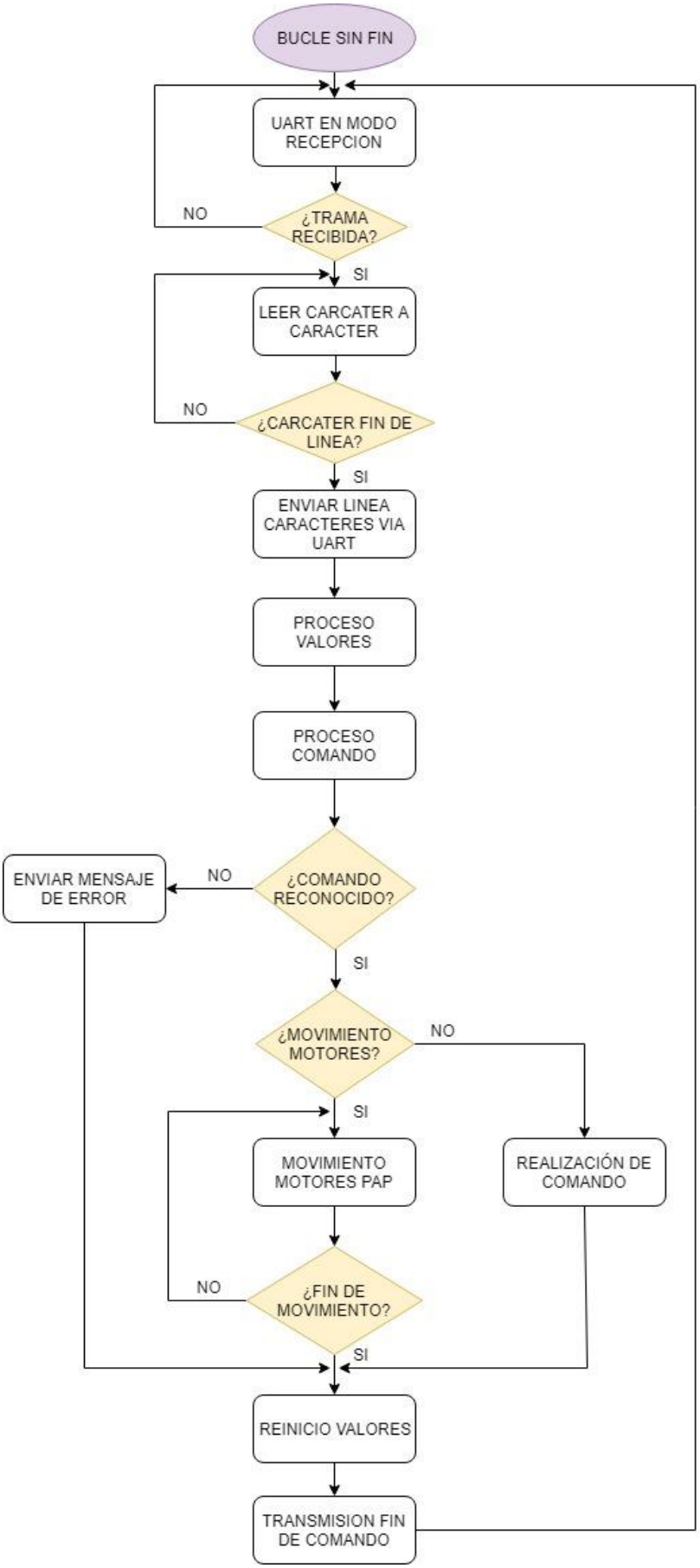


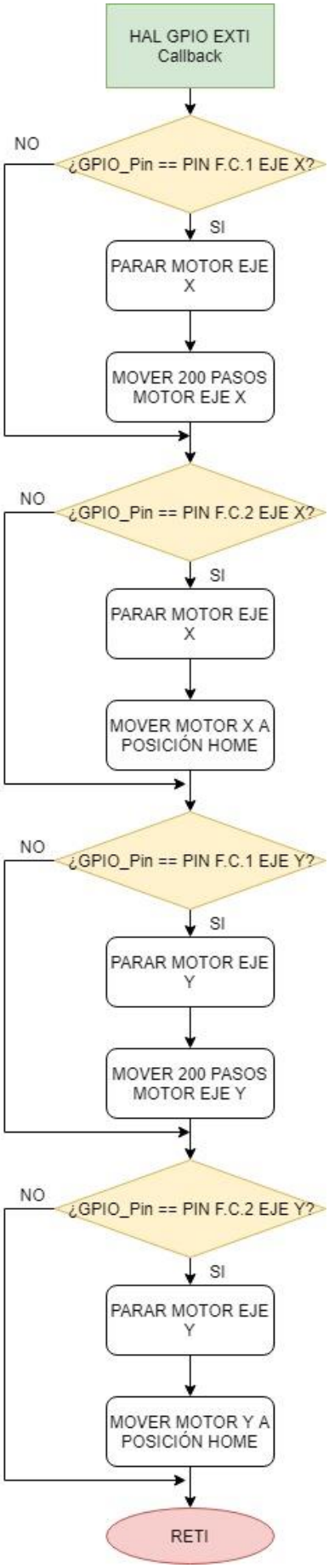
Ilustración 5.3.6 Localización del fichero con la programación para las interrupciones.

5.4. Flujogramas.

Un flujograma o diagrama de flujo es la representación gráfica de un proceso o aplicación. Sirve para facilitar la comprensión del programa y hacerlo más intuitivo. A continuación se presentan una serie de flujogramas que permiten la comprensión de la aplicación programada.


PROJECT
DECLARAR VARIABLES
INICIALIZAR LIBRERÍA HAL
CONFIGURAR TIMERS
CONFIGURAR DRIVERS IHM01A1
INICIALIZAR PERIFÉRICOS
ENVIAR MENSAJE DE ESPERA AL USUARIO
MOVIMIENTO HOMING MOTORES PAP
NO
¿FIN MOVIMIENTO MOTORES?
SI
ENVIAR MENSAJE DE SISTEMA PREPARADO AL USUARIO
BUCLE SIN FIN


BUCLE SIN FIN
UART EN MODO RECEPCION
NO
¿TRAMA RECIBIDA?
SI
LEER CARCATER A CARACTER
NO
¿CARCATER FIN DE LINEA?
SI
ENVIAR LINEA CARACTERES VIA UART
PROCESO VALORES
PROCESO COMANDO
ENVIAR MENSAJE DE ERROR
NO
¿COMANDO RECONOCIDO?
SI
¿MOVIMIENTO MOTORES?
NO
SI
MOVIMIENTO MOTORES PAP
REALIZACIÓN DE COMANDO
NO
¿FIN DE MOVIMIENTO?
SI
REINICIO VALORES
TRANSMISION FIN DE COMANDO


HAL GPIO EXTI Callback
NO
¿GPIO_Pin == PIN F.C.1 EJE X?
SI
PARAR MOTOR EJE X
MOVER 200 PASOS MOTOR EJE X
NO
¿GPIO_Pin == PIN F.C.2 EJE X?
SI
PARAR MOTOR EJE X
MOVER MOTOR X A POSICIÓN HOME
NO
¿GPIO_Pin == PIN F.C.1 EJE Y?
SI
PARAR MOTOR EJE Y
MOVER 200 PASOS MOTOR EJE Y
NO
¿GPIO_Pin == PIN F.C.2 EJE Y?
SI
PARAR MOTOR EJE Y
MOVER MOTOR Y A POSICIÓN HOME
RETI

6. G-CODES EMPLEADOS

En este capítulo se van a exponer los comandos en **lenguaje "G-code"** empleados por el sistema. Lógicamente, por razones de diseño, no todos los comandos reconocidos por el software "Txapuzas CNC" son aplicables al sistema. Algunos de estos comandos están creados para la realización de desplazamientos o movimientos circulares, los cuales no son posibles de realizar dado que el taladro actúa sobre el plano horizontal. Por tanto el sistema solo realizará desplazamientos lineales.

Por tanto los comandos permitidos por el sistema y que son reconocidos por el software "Txapuzas CNC" son:

Comando	Ejemplo	Descripción	TX	RX
G0	G0 X10	Movimiento lineal Rápido	Si	Si
G1,G01	G1 X10 Y15 Z0 [F100]	Movimiento lineal Controlado (Avance: 100)	Si	Si
~~G2,G02~~	~~G02 X00 Y00 I00 J-10 F02~~	~~Movimiento curvo (sentido horario) Controlado~~	~~Si~~	~~Si~~
~~G3,G03~~	~~G03 X00 Y00 I10 J00~~	~~Movimiento curvo (antihorario) Controlado~~	~~Si~~	~~Si~~
G4,G04	G4 P200	Pausa con retardo (Retardo: 200ms)	Si	Si
G20	G20	Definir Unidades en Pulgadas	Si	Si
G21	G21	Definir Unidades en milímetros	Si	Si
G28	G28	Ir a Origen	Si	Si
G30	G30 X10 Y20 Z30	Ir a Origen a través de un punto	Si	Si
G90	G90	Definir Coordenadas absolutas	Si	Si
G91	G91	Definir Coordenadas relativas	Si	Si
G92	G92	Definir punto actual como origen	Si	Si
M0	M0	Paro (Pausa programada)	Si	No
M3,M03	M3	Marcha del cabezal	Si	Si
M5,M05	M5	Paro del cabezal	Si	Si

Tabla 5.4.1 Comandos G empleados por el sistema.

Además, se han habilitado algunos de los comandos pertenecientes al manual "The Nist RS274NGC Interpreter" que permiten la realización de movimientos para herramientas de taladro.

Estos comandos son:

- **G81 (Drilling cycle without peck)**: ciclo de taladro sin picoteo → realiza una perforación a con una profundidad indicada.
- **G82 (Spot drilling cycle)**: ciclo de taladro con parada → realiza una parada al llegar a la profundidad indicada.

- **G83 (High-speed drilling peck)**: ciclo de taladro con picoteo → realiza una serie de perforaciones a distintas profundidades.

7. CONVERSIÓN DE MOVIMIENTO

7.1. Tipo de movimiento.

El tipo de movimiento del elemento de entrada del mecanismo es diferente del tipo de movimiento que tenga el elemento de salida, es decir, el tipo de movimiento se transforma en otro distinto, esto es conocido con el nombre de mecanismo de transformación.
Los mecanismos de transformación pueden ser, a su vez, agrupados en dos grandes grupos:

- **Mecanismos de transformación circular-lineal**: En este caso, el elemento de entrada tiene movimiento circular, mientras que el elemento de salida tiene movimiento lineal.

- **Mecanismos de transformación circular-alternativo**: En este caso, el elemento de entrada tiene movimiento circular, mientras que el elemento de salida tiene movimiento alternativo.

En el caso que concierne se trata de un mecanismo de **transformación circular-lineal**. Este mecanismo convierte el movimiento circular de un piñón en uno lineal continuo por parte de la correa dentada. El piñón al girar y estar engranado a la correa, empuja a ésta, provocando su desplazamiento lineal.

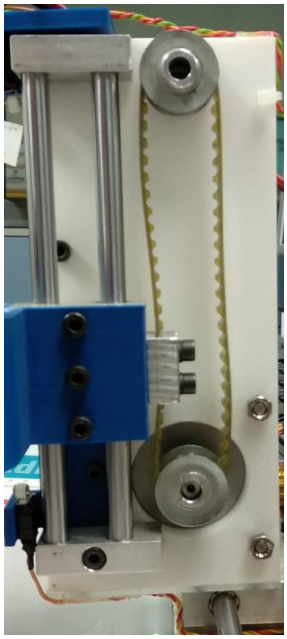


Tabla 7.1.1 Mecanismo de desplazamiento del EJE Y del sistema.

7.2. Calculo de pasos.

En capítulos anteriores se ha explicado el funcionamiento de los motores PAP y se han expuesto sus características. Ahora bien, en el trabajo se desea que los motores recorran las distancias que se les indica y para ello va a ser necesario calcular la distancia que recorren con cada paso, que será útil para convertir la distancia que se desea avanzar en el número de pasos que debe realizar el motor.

Los motores RS PRO "440-442" presentan un ángulo de paso de 1.8º. Se va a proceder al cálculo del número de pasos por vuelta:

$$1\ vuelta \rightarrow 360°$$
$$1\ paso \rightarrow 1.8°$$
$$\frac{360°}{1.8°} = 200\ pasos/rev$$

Sin embargo, para aumentar la precisión de los motores se va a usar "microstepping", con una reducción de 1/16 pasos:

$$200\ pasos/rev * 16 = 3200\ pasos/rev$$

Ahora se procede al cálculo de la distancia lineal que recorre la correa cuando el piñón da una vuelta completa. Dicha distancia coincide con la longitud total de la circunferencia del piñón. Se procede a su cálculo:

$$longitud = \pi * diametro = \pi * 20 = 62.831\ mm$$

Por tanto, una vez calculada la distancia que recorre la correa cuando el piñón realiza una vuelta completa y el número de pasos que realiza el motor por vuelta, se puede calcular la distancia recorrida por paso:

$$\frac{62.831\ mm}{3200\ pasos} = 0.0196\ mm/paso$$

8. CONEXIONADO DEL SISTEMA

En este capítulo se van a exponer todas las conexiones presentes en el sistema, que comunican y alimentan todos los dispositivos electrónicos.

Como se ha mencionado con anterioridad todos los dispositivos electrónicos se encuentran situados sobre una placa pcb, bajo la cual se han realizado las pertinentes conexiones a los pines del microcontrolador que van a ser utilizados. Al realizar de esta forma las conexiones se facilita la extracción de los dispositivos.

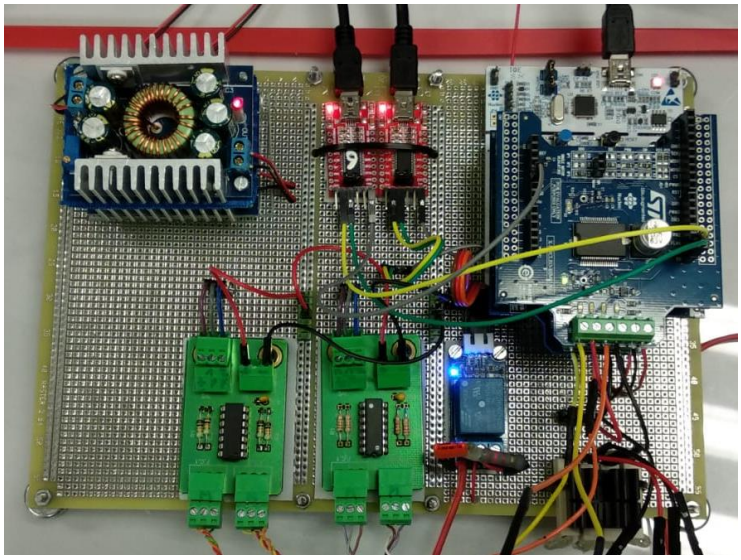


Ilustración 7.2.1 Distribución de los dispositivos electrónicos sobre la placa PCB.

La conexión entre la CPU y el convertidor USB/serie FTDI 232R se realiza a partir de un cable USB/mini-USB. Se van a utilizar dos convertidores, uno para la transmisión y recepción de datos mediante el software “Txapuzas CNC” y otro para la transmisión de datos con el software de comunicación serie. También será necesario un tercer cable que sirve tanto para la alimentación (5 V) del microcontrolador como para el envío del programa compilado.

Ilustración 7.2.2 Ejemplo de cable conexión USB/mini-USB.

Para la conexión del convertidor USB/serie con el microcontrolador se utilizan cables macho-hembra para una conexión directa o hembra-hembra para conectar con los pines de la placa pcb. Los pines del convertidor a conectar con los del microcontrolador son:

Convertidor FTDI 232R (1)	Microcontrolador
GND	GND
TX	PA 3 (USART2-RX)
RX	PA 2 (USART2-TX)

Tabla 7.2.1 Conexiones convertidor serie 1 con microcontrolador.

Convertidor FTDI 232R (2)	Microcontrolador
GND	GND
TX	PA 10 (USART1-RX)

Tabla 7.2.2 Conexiones convertidor serie 2 con microcontrolador

En cuanto a los finales de carrera, presentan tres salidas, una común, una normalmente cerrada (NC) y otra normalmente abierta (NA). Su conexión con el circuito anti rebote se realiza con unos cables creados especialmente para su conexión.

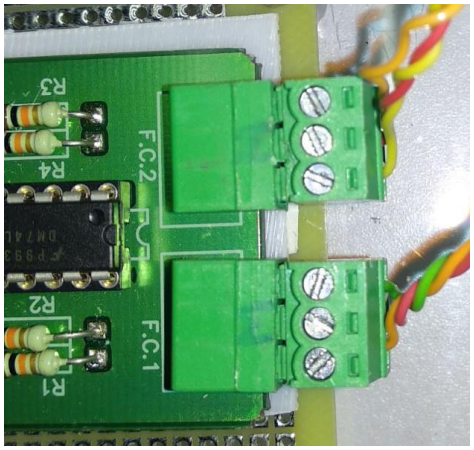


Ilustración 7.2.3 Conexión de los finales de carrera con el circuito anti rebote.

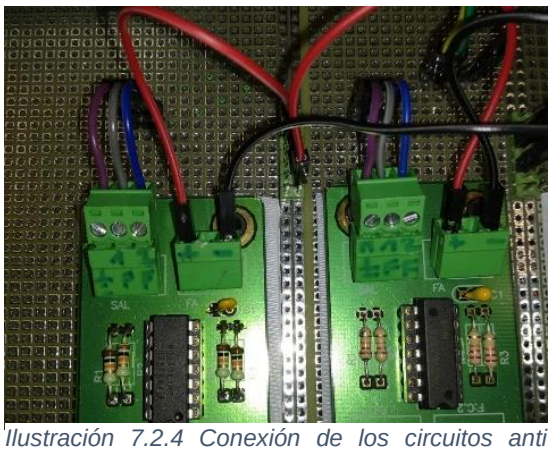


Ilustración 7.2.4 Conexión de los circuitos anti rebote con el microcontrolador y su alimentación.

Se comunica con el microcontrolador a través de las conexiones de la placa pcb. Su alimentación también proviene del microcontrolador, ya que su consumo es mínimo.

Circuito anti rebote (1)	Microcontrolador
GND	GND
5 V	E5V
F.C.1 EJE X	PC 2
F.C.2 EJE X	PC 3
Común	GND

Tabla 7.2.3 Conexiones circuito anti rebote 1 con microcontrolador.

Circuito anti rebote (2)	Microcontrolador
GND	GND
5 V	E5V
F.C.1 EJE Y	PC 11
F.C.2 EJE Y	PC 12
Común	GND

Tabla 7.2.4 Conexiones circuito anti rebote 2 con microcontrolador

La conexión del relé con el microcontrolador también se realiza a través de la placa PCB. En este caso presenta tres pines de salida que son:

Relé	Microcontrolador
GND	GND
5 V	5V
Señal	PC 8

Tabla 7.2.5 Conexiones del relé con el microcontrolador.

Por otro lado, para la conexión al taladro se ha diseñado un tipo de conexión específica con cable de cobre de un grosor superior al resto que asegure una buena resistencia al voltaje empleado para el funcionamiento del taladro. El cable con polaridad positiva del taladro irá conectado directamente al relé mientras que el de polaridad negativa irá a la fuente de alimentación. Desde el relé se conectara a la fuente de alimentación para cerrar el lazo, dado que realiza la función de interruptor.

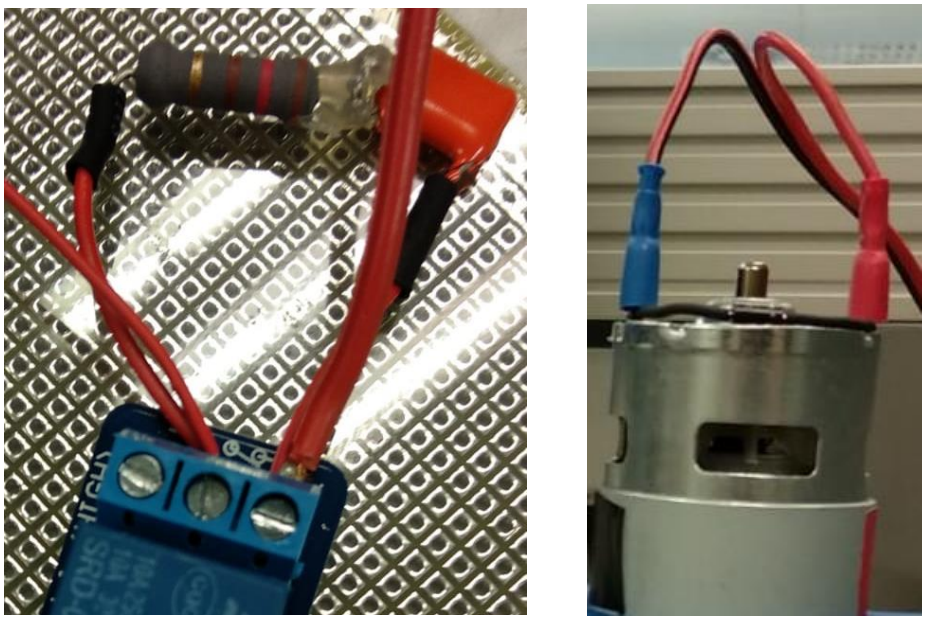


Ilustración 7.2.5 Conexión del taladro con el relé y filtro R-C.

En cuanto a los driver, se acoplan de forma directa con el microcontrolador. A su vez también presenta 4 salidas pertenecientes a los devanados del motor a controlar y dos entradas que sirven para la alimentación de los mismos.

Dos de los devanados del motor se conectaran de forma directa mientras que los otros dos deben de pasar a través de las resistencias preparadas para la reducción de la corriente que llega al motor.

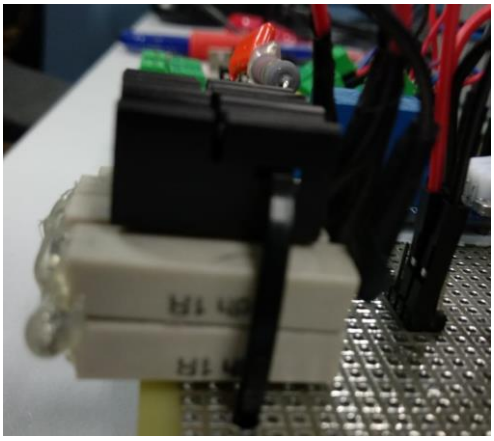
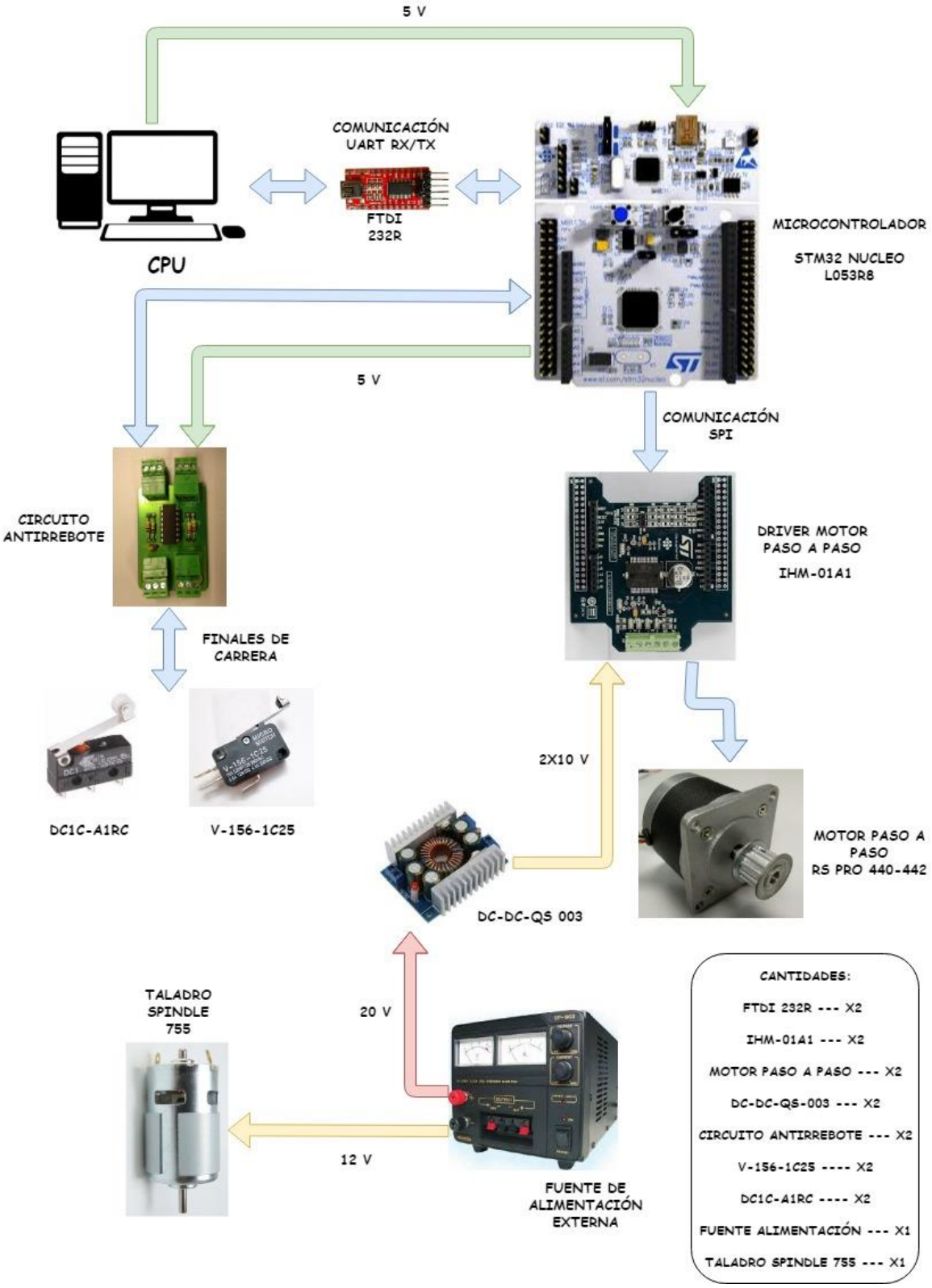


Ilustración 7.2.6 Resistencias para la reducción del amperaje de los motores PAP.

La alimentación de los motores se realiza a través de los transformadores, que irán conectados de forma directa con las entradas del driver y a la salida de la fuente de alimentación. La fuente proporciona 20 V en continua los cuales son convertidos por los transformadores en alrededor de 8 V en continua para cada motor.

A continuación se puede observar un esquema resumen con todas las conexiones del sistema:



En el ANEXO C se encuentra un plano detallado de las conexiones de los dispositivos electrónicos del sistema

9. **PRESUPUESTO**

9.1. Introducción.

Uno de los objetivos del trabajo es valorar económicamente la viabilidad del trabajo, por ello es necesario realizar un presupuesto sobre el sistema.

En este capítulo se va a reflejar y plasmar en forma de factura una **estimación del coste** todos los elementos presentes a la hora de diseñar y fabricar el sistema CNC.
En el vendrán recogidos tanto los materiales mecánicos como electrónicos, mostrando el precio de cada uno de ellos. Además se tendrán en cuenta las licencias de software necesario para llevar a cabo el sistema.

También se tendrá en cuenta el precio de mano de obra para el desarrollo del proyecto por una sola persona con un nivel profesional equivalente al de un ingeniero junior.

Por supuesto, una vez contabilizado el gasto en material, gasto en licencias y gasto en mano de obra no será necesario la aplicación del IVA, dado que ya vendrá impuesto sobre los elementos utilizados y no se trata de un producto para el mercado, sino para su uso por el personal docente de la universidad.

9.2. Desarrollo del presupuesto.

El desarrollo del presupuesto se ha divido en tres partes, correspondientes a los materiales usados para la construcción, las licencias de los software necesarios y la mano de obra invertida.

9.2.1. Materiales.

Partes mecánicas:

Nº de producto	Elemento	Coste (€/unidad)	Cantidad (unidad)	Importe total (€)
1	Base metacrilato (50x50 mm)	11.52	1	11.52
2	Varillas lisas calibradas	5.06	4	20.24
3	Estructura de acero	50	-	50
4	Polea dentada	4.45	4	17.08

5	Correa dentada EJE X	7.02	1	7.02
6	Correa dentada EJE Y	3.48	1	3.48
IMPORTE TOTAL				109.34

Tabla 9.2.1 Listado de materiales mecánicos y su presupuesto.

Partes electrónicas:

Nº de producto	Elemento	Coste (€/unidad)	Cantidad (unidad)	Importe total (€)
1	Microcontrolador X-NUCLEO-L053R8	11.52	1	11.52
2	Driver controlador IHM01A1	11.08	2	22.16
3	Convertidor serie FTDi232R	5.99	2	11.98
4	Motor PAP RS PRO 442-440	35.83	2	71.66
5	Motor Spindle 755	10.54	1	10.54
6	Transformador DC-DC-QS-003	12.25	2	24.50
7	Circuito anti rebote	5.80	2	11.60
8	Fuente alimentación Manson EP-603	124.74	1	124.74
9	F.C. V-156-1C25	0.93	2	1.86
10	F.C. DC1C-A1RC	3.62	2	7.24
11	Relé Sunfounder SRD-05VDC-SL-C	5.31	1	5.31
12	Resistencia motor	0.55	4	2.20
13	Resistencia relé	0.16	1	0.16
14	Condensador relé	0.50	1	0.50
15	Diodo DIOTEC BY 251	0.59	1	0.59
16	Placa PCB	2.29	1	2.29
17	Cable USB/serie	1.16	3	3.48
18	PC	339.99	1	339.99
IMPORTE TOTAL				652.32

Tabla 9.2.2 Listado de materiales electrónicos y su presupuesto.

IMPORTE TOTAL DE MATERIALES (€)	**761.66**

9.2.2. Licencias.

Para el empleo de algunos softwares es necesario abona un pago de licencia de uso. Los programas usados que lo requieren son:

Elemento	Tipo	Importe total (€)
Solidworks 2016x64	Licencia de estudiante	120.00
Microsoft Office 2013 Home & Student	Licencia de estudiante	139.99
IMPORTE TOTAL		259.99

Tabla 9.2.3 Listado de software utilizado y coste de las licencias.

9.2.3. Mano de obra.

La mano de obra del trabajo ha sido realizada por un ingeniero junior, y puede dividirse en tres funciones, una de ingeniería, otra de supervisión y otra técnica.

Tipo	Cantidad (horas)	Coste unitario (€/hora)	Importe total (€)
Ingenieril	275	12	3300
Supervisión	25	12	300
Técnica	50	8	400
IMPORTE TOTAL			4000

Tabla 9.2.4 Presupuesto de la mano de obra.

9.3. Presupuesto final.

Tipo	Importe total (€)
Materiales	761.66
Licencias	259.99
Mano de obra	4000
TOTAL	**5021.65**

Tabla 9.3.1 Presupuesto final del coste del sistema.

Este presupuesto es una estimación de lo que supondría la creación del sistema, sin embargo, el precio real ha sido mucho menor, ya que la gran mayoría de materiales son reutilizados, las licencias son gratuitas por formar parte de la universidad y la mano de obra es inexistente ya que es un trabajo realizado por un alumno con el fin de obtener su título de Máster en Ingeniería Mecatrónica.

10. MEJORAS DEL SISTEMA

Una vez finalizado el sistema y verificado su funcionamiento, se comprueba que cumple con todos los requisitos y objetivos marcados al principio del trabajo. Sin embargo, que cumpla con los objetivos no supone que el sistema no pueda ser mejorado, al contrario existen muchas posibilidades de mejora que podrían llevarse a cabo.

Antes de plantear estas posibles mejoras se debe tener en cuenta que el sistema diseñado está construido para su uso por el personal docente con el fin de ayudar u optimizar el aprendizaje sobre maquinas CNC a los futuros alumnos del máster y no con el fin de ser un prototipo de última generación de máquinas CNC.

Algunas de las posibles mejoras son:

- **Movimientos circulares**: como se ha podido observar una de las principales carencias del sistema es la imposibilidad del uso de movimientos circulares, y esto es debido a que se ha diseñado el sistema para actuar sobre el plano X-Y (horizontal). Una posible solución sería la de diseñar una pieza que sujetase la herramienta para actuar sobre el plano X-Z (vertical), aunque en este caso quedaría descartado el uso de un taladro como herramienta y dejaría paso a otras posibles herramientas, como puede ser un láser para marcado.
- **Multiherramienta**: como se ha mencionado en el punto anterior, una posible mejora seria la implementación del uso de otro tipo de herramientas, como podrían ser herramientas laser o de corte.
- **3 ejes**: al tratar de simplificar las funciones de un sistema CNC se optó por el uso de solo 2 ejes, sin embargo sería posible el uso de un tercero, ya que los drivers permiten el control de hasta tres motores PAP. Esta mejora supondría un cambio total en el diseño de la estructura del sistema y la incorporación de más elementos electrónicos.
- **Precisión**: Los motores PAP son una clase de motores bastante preciso, además con el uso de una señal de micro pasos (microstepping) esta precisión aumenta, sin embargo siempre cabe la posibilidad de que el motor realice un fallo y se coma algún paso. Una posible mejora seria el uso de encoders que contabilizasen la cantidad de pasos a realizar para evitar este tipo de fallos.

11. CONCLUSIÓN

El principal objetivo de este trabajo era, como su propio título indica, el diseño e implementación de un sistema CNC para uso docente. Y una vez terminado se puede decir que cumple con su finalidad.

La idea era llevar a cabo un **sistema simplificado** de maquina CNC que contenga una diversidad de elementos, tanto mecánicos como electrónicos, que lo conviertan en un sistema de ingeniería mecatrónica.

Se trata de un trabajo que engloba todos los conocimientos adquiridos por el autor a lo largo de su periodo estudiantil en el Máster de Ingeniería Mecatrónica. Durante su realización se han puesto a prueba **conocimientos informáticos**, a la hora de programar el microcontrolador junto con el resto de dispositivos, **conocimiento mecánicos**, a la hora de construir la estructura o el diseño de piezas necesarias, y **conocimientos en electrónica**, para la conexión y uso de todos los dispositivos electrónicos presentes en el sistema.

Además, se ha logrado obtener un **firmware propio**, basado en GRBL, combinando el código de control de los motores utilizado por los drivers junto al intérprete de comandos “G-code” programado. Gracias a esta simplificación se facilitará la comprensión del GRBL, que es la base de toda máquina CNC, independientemente de su función.

Se espera que este trabajo sea una buena **aportación para las futuras generaciones** de alumnos del máster y que les ayude a comprender e indagar, de forma más profunda, en los aspectos del uso de máquinas CNC, una tecnología que desde sus comienzos más primarios hace décadas, sigue a día de hoy evolucionando de forma continua.

Una vez finalizado se puede decir que se han superado todos los retos que suponía el desarrollo del sistema de forma satisfactoria.

12. BIBLIOGRAFÍA

Libros:

1. **Título**: C: A REFERENCE MANUAL
   Autores: Samuel P. Harbison y Guy L. Steele Jr.
   Editorial: Prentice Hall

2. **Título**: CNC Machining Handbook
   Autor: Alan Overby
   Editorial: McGraw Hill

3. **Título**: MANUALES INGENIERÍAS Y ARQUITECTURA: APRENDIENDO C
   Autores: José María Rodríguez Corral y José Galindo Gómez
   Editorial: UCA

4. **Título**: STEPPING MOTORS AND THEIR MICROPROCESSOR CONTROL
   Autor: Takashi Kenjo
   Editorial: Oxford Science Publications

Páginas web:

http://www.alciro.org/alciro/Plotter-Router-Fresadora-CNC_1/notores-paso-a-paso-8-hilos_433.htm
http://robots-argentina.com.ar/MotorPP_basico.htm
https://www.youtube.com/watch?v=QhPIPNL8qZs
https://www.staticboards.es/blog/dominar-motor-paso-a-paso-con-grbl/
https://www.staticboards.es/blog/motores-paso-paso/
http://www.retrowiki.es/viewtopic.php?t=200030598
https://www.waveshare.com/wiki/STM32CubeMX_Tutorial_Series:_EXTI

13. ANEXOS

13.1. ANEXO A: Programación.

-Programa principal *(main.c)*

```
81  int main(void)
82  {
83      //Se crean las siguientes variables que serán utilizadas por el programa principal.
84
85      static char car;           //variable para un carácter
86      static char linea[1024];   //variable para una cadena de caracteres
87
88      uint8_t i, estado;         //variables del contador y del estado del buffer
89
90      float Valor_X, Valor_Y, Valor_F, Valor_P, Valor_D;   //variables para los valores enviados al sistema
91
92      int pulgadas, absoluto, direccion_x, direccion_y;    //variables para definir funciones del sistema
93
94    /* STM32xx HAL library initialization */
95    HAL_Init();
96
97    /* Configure the system clock */
98    SystemClock_Config();
99
100     /* Set the L6474 library to use 2 device */
101   BSP_MotorControl_SetNbDevices(BSP_MOTOR_CONTROL_BOARD_ID_L6474, 2);
102
103     /* Initialisation of first device */
104   BSP_MotorControl_Init(BSP_MOTOR_CONTROL_BOARD_ID_L6474, NULL);
105   /* Initialisation of second device */
106   BSP_MotorControl_Init(BSP_MOTOR_CONTROL_BOARD_ID_L6474, NULL);
107
108     /* Attach the function MyFlagInterruptHandler (defined below) to the flag interrupt */
109   BSP_MotorControl_AttachFlagInterrupt(MyFlagInterruptHandler);
110
```

```
111   /* Attach the function Error_Handler (defined below) to the error Handler*/
112   BSP_MotorControl_AttachErrorHandler(Error_Handler);
113
114     /* Initialize all configured peripherals */
115     MX_USART2_UART_Init();
116     MX_GPIO_Init();
117     MX_LPUART1_UART_Init();
118
119     /* USER CODE BEGIN 2 */
120     //Se avisa al usuario de que espere hasta que el sistema se posicione en home
121     Mensa_Debug_HLPUART1("Espere mientras se prepara el sistema... \n");
122
123     //Habilitamos la interrupción para la recepción de datos a través de la comuniación UART
124     HAL_UART_Receive_IT(&huart2, (uint8_t*) Rx_data, 1);
125
126     //Se realizan los movimientos necesarios para que el sistema se posicione en home
127   Movimiento_Homing();
128
129     //Por protección el relé que controla el taladro se pone a reset
130     HAL_GPIO_WritePin(GPIOC, GPIO_PIN_8, GPIO_PIN_RESET);
131
132     //Una vez en posición home, se avisa al usuario que el sistema esta preparado
133     Mensa_Debug_HLPUART1("Sistema preparado: \n");
134
135   /* USER CODE END 2 */
136
```

```
137   /* Infinite loop */
138   while(1)
139   {
140       i=0;
141       car = USART2_Get_char();//Se guarda un carácter en la variable car
142
143       //Se crea un bucle que terminará cuando la variable car sea un salto de linea
144       while(car != '\n'){
145       estado = BufferIsEmpty(U2Rx);//Se comprueba el estado del buffer de recepción y se almacena en la variable estado
146
147           //En caso de que la variable estado sea distinto de SUCCESS --> distinto de 0
148           //se almacenan los caracteres en orden de uno en uno en la variable linea[i]
149               if (estado != SUCCESS){
150                       linea[i] = car;
151                       car = USART2_Get_char();
152                       i++;
153               }
154       }
155
156       //Se almacena el ultimo caracter de salto de linea recibido y se añade un fin de linea.
157       linea[i] = car;
158       linea[i+1] = NULL;
159
160       //La variable linea es enviada por el puerto serie para ser visualizada por el usuario
161       Mensa_Debug_HLPUART1(linea);
162
163       HAL_Delay(250);
164
165       //Con la función 'Proceso_Valores' se obtienen los valores numéricos que aparecen en la linea de caracteres
166       // recibida y se almacenan en sus respectivas variables.
167       Proceso_Valores(linea, &Valor_X, &Valor_Y, &Valor_F, &Valor_P, &Valor_D, pulgadas, &direccion_x, &direccion_y, absoluto);
```

```
168
169             HAL_Delay(250);
170
171             // Con la funcón 'Proceso_Comando' se analiza la linea de caracteres recibida para obtener el tipo
172             // de comando (g-code) y realizar así el movimiento/funcion específico de dicho comando.
173             Proceso_Comando(linea, Valor_X, Valor_Y, Valor_F, Valor_P, Valor_D, direccion_x, direccion_y, &pulgadas, &absoluto);
174
175             HAL_Delay(250);
176
177             //Con la funcion 'Reinicio_Valores' reseteamos los valores numéricos que aparecen en la linea de caracteres
178             // recibida.
179             Reinicio_Valores(&Valor_X, &Valor_Y, &Valor_F, &Valor_P);
180
181             HAL_Delay(500);
182
183             //Una vez realizado el comando se envía el OK, que indica a "Txapu" que se ha finalizado y puede proceder a otro envío
184             USART2_PutString ("OK");
185     }
186
187
188 }
```

-Funciones *(main.c):*

- **Proceso_Letras**

```
199  void Proceso_Letras (char linea_comando[1024], char letra, float *valor)
200  {
201
202      static char buff[50];  //Se crea un buffer que almacenará los valores
203
204      uint8_t e, k, inicio, final, numero, len;
205
206
207  len = strlen(linea_comando);  //esta función calcula la cantidad de caracteres que tiene la cadena linea_comando
208
209      //Se crea un bucle que recorre la linea de caracteres recibida hasta encontrar el caracter
210      //deseado y devolver el valor numérico tras este
211      for(e=0;e <= len ;e++)
212          {
213      if(linea_comando[e]== letra){
214                  e++;
215                  inicio=e;
216                  k=0;
217
218              while((linea_comando[e]!=' ')&&(linea_comando[e]!='\r')&&(linea_comando[e]!='\n')&&(linea_comando[e]!=NULL)){
219                  final=e;
220                  e++;
221              }
222              for(numero=inicio;numero<final+1;numero++){
223                  buff[k]=linea_comando[numero];
224                  k++;
225              }
226              *valor=atoi(buff);
227              memset(buff, 0, k);//se vacia el buffer
228          }
229      }
230  }
```

- **Proceso_Valores**

```
void Proceso_Valores (char linea_comando[1024], float *valor_x, float *valor_y, float *valor_f, float *valor_p, float *valor_d,
                                                int pulg, int *dir_x, int *dir_y, int abs)
{

//Se utiliza la funcion Proceso_Letras para almacenar los caracteres numericos de la linea comando referentes a cada letra
Proceso_Letras(linea_comando, 'X', valor_x);
Proceso_Letras(linea_comando, 'Y', valor_y);
Proceso_Letras(linea_comando, 'F', valor_f);
Proceso_Letras(linea_comando, 'P', valor_p);
Proceso_Letras(linea_comando, 'D', valor_d);

        //Ahora se procede al calculo de los pasos que debe girar el motor a partir de las distancias
        //a recorrer en milimetros o en pulgadas

        if(pulg==1){
            *valor_x = *valor_x/(0.00077);
            *valor_y = *valor_y/(0.00077);
            *valor_d = *valor_d/(0.00077);
        }
        else{
            *valor_x = *valor_x/(0.0196);
            *valor_y = *valor_y/(0.0196);
            *valor_d = *valor_d/(0.0196);
        }

        //Se procede a la obtención de la dirección en la que debe girar el motor

        if(*valor_x <= 0 && abs==0){
            *dir_x=FORWARD;
            *valor_x = *valor_x*(-1);
        }
        else{
```

```
268             *dir_x=BACKWARD;
269         }
270         if(*valor_y <= 0 && abs==0){
271           *dir_y=FORWARD;
272             *valor_y= *valor_y*(-1);
273         }
274         else{
275             *dir_y=BACKWARD;
276         }
277
278         //En coordenadas absolutas cambiamos el signo de los valores debido a las características del motor
279         if(abs==1){
280             *valor_x = *valor_x*(-1);
281             *valor_y= *valor_y*(-1);
282         }
283
284     *valor_p = *valor_p * 1000; //El valor de tiempo de espera se pasa a milisegundos
285
286     //Se configura la velocidad al valor indicado
287     //Para mejorar la rampa de aceleracion a la velocidad minima se le resta un valor de 100
288     BSP_MotorControl_SetMaxSpeed(1,*valor_f);
289     BSP_MotorControl_SetMinSpeed(1,*valor_f-100);
290     BSP_MotorControl_SetMaxSpeed(0,*valor_f);
291     BSP_MotorControl_SetMinSpeed(0,*valor_f-100);
292
293 }
```

- **Proceso_Comando**

```
299  void Proceso_Comando(char linea_comando[1024],float valor_x, float valor_y, float valor_f, float valor_p,
300                                                  float valor_d, int dir_x, int dir_y, int *pulg, int *abs)
301  {
302      static char* buff;
303      static char* buff1;
304
305      uint8_t Num_Comando, len, e;
306      int32_t pos_x, pos_y;
307
308
309      len = strlen(linea_comando);  //'strlen' calcula la cantidad de caracteres que tiene la cadena linea_comando
310
311      buff=strtok(linea_comando," ");  //'strtok' trocea la cadena linea_comando en función del carácter
312                                       // que se ha especificado
313
314      for(e=0;e < len ;e++)
315          {
316
317          if(linea_comando[e]=='G'){
318
319              buff1=strtok(buff,"G");
320              Num_Comando=atoi(buff1);   //'atoi' transforma de char a int la variable designada
321
322          //Dependiendo del valor del comando se realizará una serie de funciones
323              switch(Num_Comando){
324                  case 0:
325                  if(*abs==1){
326                  BSP_MotorControl_GoTo(1,valor_y);
327                  BSP_MotorControl_GoTo(0,valor_x);
328                  BSP_MotorControl_WaitWhileActive(1);
329                  BSP_MotorControl_WaitWhileActive(0);
330                  BSP_MotorControl_CmdDisable(1);
```

```
330                 BSP_MotorControl_CmdDisable(1);
331                 BSP_MotorControl_CmdDisable(0);
332                 }
333                 else{
334               BSP_MotorControl_Move(1,dir_y,valor_y);
335                 BSP_MotorControl_Move(0,dir_x,valor_x);
336                 BSP_MotorControl_WaitWhileActive(1);
337                 BSP_MotorControl_WaitWhileActive(0);
338                 BSP_MotorControl_CmdDisable(1);
339                 BSP_MotorControl_CmdDisable(0);
340                 }
341                 break;
342
343                 case 1:
344                 if(*abs==1){
345                 BSP_MotorControl_GoTo(1,valor_y);
346                 BSP_MotorControl_WaitWhileActive(1);
347                 BSP_MotorControl_SoftStop(1);
348                 BSP_MotorControl_WaitWhileActive(1);
349                 BSP_MotorControl_CmdDisable(1);
350
351                 BSP_MotorControl_GoTo(0,valor_x);
352                 BSP_MotorControl_WaitWhileActive(0);
353                 BSP_MotorControl_SoftStop(0);
354                 BSP_MotorControl_WaitWhileActive(0);
355                 BSP_MotorControl_CmdDisable(0);
356                 }
357                 else{
358                 BSP_MotorControl_Move(1, dir_y, valor_y);
359                 BSP_MotorControl_WaitWhileActive(1);
360                 BSP_MotorControl_SoftStop(1);
361                 BSP_MotorControl_WaitWhileActive(1);
```

```
361                 BSP_MotorControl_WaitWhileActive(1);
362                 BSP_MotorControl_CmdDisable(1);
363
364                 BSP_MotorControl_Move(0, dir_x, valor_x);
365                 BSP_MotorControl_WaitWhileActive(0);
366                 BSP_MotorControl_SoftStop(0);
367                 BSP_MotorControl_WaitWhileActive(0);
368                 BSP_MotorControl_CmdDisable(0);
369                 }
370                 break;
371
372                 case 4:
373                 BSP_MotorControl_SoftStop(0);
374                 BSP_MotorControl_WaitWhileActive(0);
375                 BSP_MotorControl_SoftStop(1);
376                 BSP_MotorControl_WaitWhileActive(1);
377                 HAL_Delay(valor_p);
378                 break;
379
380                 case 20:
381                 *pulg=1;
382                 break;
383
384                 case 21:
385                 *pulg=0;
386                 break;
387
388                 case 28:
389                 BSP_MotorControl_GoHome(1);
390                 BSP_MotorControl_WaitWhileActive(1);
391                 BSP_MotorControl_CmdDisable(1);
392                 BSP_MotorControl_GoHome(0);
```

```
392             BSP_MotorControl_GoHome(0);
393             BSP_MotorControl_WaitWhileActive(0);
394             BSP_MotorControl_CmdDisable(0);
395             break;
396
397             case 30:
398             if(*abs==1){
399             BSP_MotorControl_GoTo(1,valor_y);
400             BSP_MotorControl_WaitWhileActive(1);
401             BSP_MotorControl_GoTo(0,valor_x);
402             BSP_MotorControl_WaitWhileActive(0);
403             }
404             else{
405             BSP_MotorControl_Move(1, dir_y, valor_y);
406             BSP_MotorControl_WaitWhileActive(1);
407             BSP_MotorControl_Move(0, dir_x, valor_x);
408             BSP_MotorControl_WaitWhileActive(0);
409             }
410             BSP_MotorControl_GoHome(1);
411             BSP_MotorControl_WaitWhileActive(1);
412             BSP_MotorControl_CmdDisable(1);
413             BSP_MotorControl_GoHome(0);
414             BSP_MotorControl_WaitWhileActive(0);
415             BSP_MotorControl_CmdDisable(0);
416             break;
417
418             case 81:
419             //DRILLING CYCLE WITHOUT PECK
420             HAL_GPIO_WritePin(GPIOC, GPIO_PIN_8, GPIO_PIN_SET);
421             HAL_Delay(1000);
422             BSP_MotorControl_SetMaxSpeed(1,500);
423             BSP_MotorControl_SetMinSpeed(1,400);
```

```
423             BSP_MotorControl_SetMinSpeed(1,400);
424             HAL_Delay(1000);
425             BSP_MotorControl_Move(1, BACKWARD, valor_d);
426             BSP_MotorControl_WaitWhileActive(1);
427             HAL_Delay(1000);
428             BSP_MotorControl_SetMaxSpeed(1,400);
429             BSP_MotorControl_SetMinSpeed(1,300);
430             BSP_MotorControl_Move(1, FORWARD, valor_d);
431             BSP_MotorControl_WaitWhileActive(1);
432             HAL_Delay(1000);
433             HAL_GPIO_WritePin(GPIOC, GPIO_PIN_8, GPIO_PIN_RESET);
434             BSP_MotorControl_CmdDisable(1);
435             break;
436
437             case 82:
438             //SPOT DRILLING CYCLE
439             HAL_GPIO_WritePin(GPIOC, GPIO_PIN_8, GPIO_PIN_SET);
440             HAL_Delay(1000);
441
442             BSP_MotorControl_Move(1, BACKWARD, valor_d);
443             BSP_MotorControl_WaitWhileActive(1);
444             HAL_Delay(valor_p);
445
446             BSP_MotorControl_Move(1, FORWARD, valor_d);
447             BSP_MotorControl_WaitWhileActive(1);
448             HAL_Delay(1000);
449
450             HAL_GPIO_WritePin(GPIOC, GPIO_PIN_8, GPIO_PIN_RESET);
451             BSP_MotorControl_CmdDisable(1);
452             break;
453
454             case 83:
```

```
455             //HIGH-SPEED DRILLING PECK (DEEPER HOLES)
456             HAL_GPIO_WritePin(GPIOC, GPIO_PIN_8, GPIO_PIN_SET);
457
458             BSP_MotorControl_SetMaxSpeed(1,500);
459             BSP_MotorControl_SetMinSpeed(1,400);
460             HAL_Delay(1000);
461
462             BSP_MotorControl_Move(1, BACKWARD, (valor_d/3));
463             BSP_MotorControl_WaitWhileActive(1);
464             HAL_Delay(500);
465             BSP_MotorControl_Move(1, FORWARD, (valor_d/3));
466             BSP_MotorControl_WaitWhileActive(1);
467             HAL_Delay(500);
468
469             BSP_MotorControl_Move(1, BACKWARD, (valor_d-(valor_d/3)));
470             BSP_MotorControl_WaitWhileActive(1);
471             HAL_Delay(500);
472             BSP_MotorControl_Move(1, FORWARD, (valor_d-(valor_d/3)));
473             BSP_MotorControl_WaitWhileActive(1);
474             HAL_Delay(500);
475
476             BSP_MotorControl_Move(1, BACKWARD, valor_d);
477             BSP_MotorControl_WaitWhileActive(1);
478             HAL_Delay(500);
479             BSP_MotorControl_SetMaxSpeed(1,400);
480             BSP_MotorControl_SetMinSpeed(1,300);
481             HAL_Delay(500);
482             BSP_MotorControl_Move(1, FORWARD, valor_d);
483             BSP_MotorControl_WaitWhileActive(1);
484             HAL_Delay(1000);
485
```

```
486                 HAL_GPIO_WritePin(GPIOC, GPIO_PIN_8, GPIO_PIN_RESET);
487                 BSP_MotorControl_CmdDisable(1);
488                 break;
489
490                 case 90:
491                 *abs=1;
492                 break;
493
494                 case 91:
495                 *abs=0;
496                 break;
497
498                 case 92:
499                     pos_x = BSP_MotorControl_GetPosition(1);
500                     BSP_MotorControl_SetHome(0, pos_x);
501                     pos_y = BSP_MotorControl_GetPosition(1);
502                     BSP_MotorControl_SetHome(1, pos_y);
503                 break;
504
505                 default:
506                 Mensa_Debug_HLPUART1("ERROR (COMANDO NO DISPONIBLE)\n");
507                 break;
508             }
509
510         }
511
512         else if(linea_comando[e]=='M'){
513
514             buff1=strtok(buff,"M");
515             Num_Comando=atoi(buff1);
516
```

```
517             switch(Num_Comando){
518             case 0:
519             BSP_MotorControl_SoftStop(1);
520             BSP_MotorControl_WaitWhileActive(1);
521             BSP_MotorControl_SoftStop(0);
522             BSP_MotorControl_WaitWhileActive(0);
523             BSP_MotorControl_CmdDisable(0);
524             BSP_MotorControl_CmdDisable(1);
525             break;
526
527             case 3:
528             HAL_GPIO_WritePin(GPIOC, GPIO_PIN_8, GPIO_PIN_SET);
529             break;
530
531             case 5:
532             HAL_GPIO_WritePin(GPIOC, GPIO_PIN_8, GPIO_PIN_RESET);
533             break;
534
535             default:
536             Mensa_Debug_HLPUART1("ERROR (COMANDO NO DISPONIBLE)\n");
537             break;
538             }
539         }
540 }
541
542 }
```

- **Reinicio_Valores**

```
//Con la funcion 'Reinicio_Valores' reseteamos los valores numericos que aparecen en la linea de caracteres
// recibida.

void Reinicio_Valores(float *valor_x, float *valor_y, float *valor_f, float *valor_p){

    *valor_x=0;
    *valor_y=0;
    *valor_p=0;
    *valor_f=0;

}


//Con la función 'Movimiento_Homing' se realiza un homing (vuelta a casa) de los motores al punto de partida.

void Movimiento_Homing(){

    uint32_t pos_x, pos_y;

    BSP_MotorControl_Run(1,FORWARD);
    BSP_MotorControl_WaitWhileActive(1);
    pos_y=BSP_MotorControl_GetPosition(1);
    BSP_MotorControl_SetHome(1, pos_y);
    BSP_MotorControl_CmdDisable(1);

    BSP_MotorControl_Run(0,FORWARD);
    BSP_MotorControl_WaitWhileActive(0);
    pos_x=BSP_MotorControl_GetPosition(0);
    BSP_MotorControl_SetHome(0, pos_x);
    BSP_MotorControl_CmdDisable(0);
}
```

- **Funciones de comunicación puerto serie**

```
593  void Mensa_Debug_HLPUART1( char buff_mem[1024])
594  {
595      if(HAL_UART_Transmit(&hlpuart1, (uint8_t *)buff_mem, strlen((char *)buff_mem),500)!= HAL_OK)
596      {
597          Error_Handler2();
598      }
599      HAL_Delay(2);
600      return;
601  }
602
603  void USART2_PutString (char Tx_data[96])
604  {
605      if(HAL_UART_Transmit(&huart2, (uint8_t*) Tx_data, (uint16_t) strlen(Tx_data), 200)!= HAL_OK)
606    {
607      Error_Handler3();
608    }
609  }
610
611  void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
612  {
613      if (huart->Instance == USART2)  //current UART
614          {
615          BufferPut(&U2Rx, Rx_data[0]);//   add data to Rx_buffer
616          HAL_UART_Receive_IT(&huart2, (uint8_t*) Rx_data, 1);    //activate UART receive for next char received
617          }
618  }
```

-<u>Configuración periféricos</u> *(uart.c y gpio.c)*

- **Comunicación USART**

```
53  /* LPUART1 init function */
54  void MX_LPUART1_UART_Init(void)
55  {
56    hlpuart1.Instance = LPUART1;
57    hlpuart1.Init.BaudRate = 38400;
58    hlpuart1.Init.WordLength = UART_WORDLENGTH_8B;
59    hlpuart1.Init.StopBits = UART_STOPBITS_1;
60    hlpuart1.Init.Parity = UART_PARITY_NONE;
61    hlpuart1.Init.Mode = UART_MODE_TX_RX;
62    hlpuart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
63    hlpuart1.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
64    hlpuart1.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
65    if (HAL_UART_Init(&hlpuart1) != HAL_OK)
66    {
67      Error_Handler3();
68    }
69  }
70  /* USART2 init function */
71  void MX_USART2_UART_Init(void)
72  {
73    huart2.Instance = USART2;
74    huart2.Init.BaudRate = 19200;
75    huart2.Init.WordLength = UART_WORDLENGTH_8B;
76    huart2.Init.StopBits = UART_STOPBITS_1;
77    huart2.Init.Parity = UART_PARITY_NONE;
78    huart2.Init.Mode = UART_MODE_TX_RX;
79    huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
80    huart2.Init.OverSampling = UART_OVERSAMPLING_16;
81    huart2.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
82    huart2.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
83    if (HAL_UART_Init(&huart2) != HAL_OK)
84    {
85      Error_Handler3();
86    }
```

- **Configuración pines E/S**

```
60  void MX_GPIO_Init(void)
61  {
62      GPIO_InitTypeDef GPIO_InitStruct = {0};
63      /* GPIO Ports Clock Enable */
64      __HAL_RCC_GPIOC_CLK_ENABLE();
65      __HAL_RCC_GPIOB_CLK_ENABLE();
66
67      /*Configure GPIO pin : PC8 */
68      GPIO_InitStruct.Pin = GPIO_PIN_8;
69      GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
70      GPIO_InitStruct.Pull = GPIO_NOPULL;
71      GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
72      HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
73
74      /*Configure GPIO pins : PC11 PC12 */
75      GPIO_InitStruct.Pin = GPIO_PIN_11 | GPIO_PIN_12;
76      GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
77      GPIO_InitStruct.Pull = GPIO_NOPULL;
78      HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
79
80      /*Configure GPIO pin : PC2 PC3 */
81      GPIO_InitStruct.Pin = GPIO_PIN_2 | GPIO_PIN_3;
82      GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
83      GPIO_InitStruct.Pull = GPIO_NOPULL;
84      HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
85
86      /* EXTI interrupt init*/
87      HAL_NVIC_SetPriority(EXTI2_3_IRQn, 0, 0);
88      HAL_NVIC_EnableIRQ(EXTI2_3_IRQn);
89
90      HAL_NVIC_SetPriority(EXTI4_15_IRQn, 0, 0);
91      HAL_NVIC_EnableIRQ(EXTI4_15_IRQn);
92  }
```

-<u>Interrupción final de carrera</u> *(stm32lxx_hal_msp.c)*

```
271  void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
272  {
273    if (GPIO_Pin == BSP_MOTOR_CONTROL_BOARD_FLAG_PIN){
274      BSP_MotorControl_FlagInterruptHandler();
275    }
276      if (GPIO_PIN_11 == GPIO_Pin){
277          BSP_MotorControl_HardStop(1);
278          BSP_MotorControl_WaitWhileActive(1);
279          BSP_MotorControl_Move(1,BACKWARD,200);
280          __HAL_GPIO_EXTI_CLEAR_IT(GPIO_Pin);
281          __HAL_GPIO_EXTI_CLEAR_FLAG(GPIO_Pin);
282      }
283      if (GPIO_PIN_12 == GPIO_Pin){
284          BSP_MotorControl_HardStop(1);
285          BSP_MotorControl_WaitWhileActive(1);
286          BSP_MotorControl_GoHome(1);
287          __HAL_GPIO_EXTI_CLEAR_IT(GPIO_Pin);
288          __HAL_GPIO_EXTI_CLEAR_FLAG(GPIO_Pin);
289      }
290      if (GPIO_PIN_2 == GPIO_Pin){
291          BSP_MotorControl_HardStop(0);
292          BSP_MotorControl_WaitWhileActive(0);
293          BSP_MotorControl_Move(0,BACKWARD,200);
294          __HAL_GPIO_EXTI_CLEAR_IT(GPIO_Pin);
295          __HAL_GPIO_EXTI_CLEAR_FLAG(GPIO_Pin);
296      }
297      if (GPIO_PIN_3 == GPIO_Pin){
298          BSP_MotorControl_HardStop(0);
299          BSP_MotorControl_WaitWhileActive(0);
300          BSP_MotorControl_GoHome(0);
301          __HAL_GPIO_EXTI_CLEAR_IT(GPIO_Pin);
302          __HAL_GPIO_EXTI_CLEAR_FLAG(GPIO_Pin);
303      }
304  }
```

-Configuración motores *(l6474.c → l6474_target_config.h)*

```
63  /// The maximum number of devices in the daisy chain
64  #define MAX_NUMBER_OF_DEVICES                   (2)
65
66  /************************ Speed Profile  *******************************/
67
68  /// Acceleration rate in step/s2 for device 0 (must be greater than 0)
69  #define L6474_CONF_PARAM_ACC_DEVICE_0         (160)
70  /// Acceleration rate in step/s2 for device 1 (must be greater than 0)
71  #define L6474_CONF_PARAM_ACC_DEVICE_1         (160)
72
73
74  /// Deceleration rate in step/s2 for device 0 (must be greater than 0)
75  #define L6474_CONF_PARAM_DEC_DEVICE_0         (160)
76  /// Deceleration rate in step/s2 for device 1 (must be greater than 0)
77  #define L6474_CONF_PARAM_DEC_DEVICE_1         (160)
78
79
80  /// Maximum speed in step/s for device 0 (30 step/s < Maximum speed <= 10 000 step/s )
81  #define L6474_CONF_PARAM_MAX_SPEED_DEVICE_0   (400)
82  /// Maximum speed in step/s for device 1 (30 step/s < Maximum speed <= 10 000 step/s )
83  #define L6474_CONF_PARAM_MAX_SPEED_DEVICE_1   (400)
84
85
86  /// Minimum speed in step/s for device 0 (30 step/s <= Minimum speed < 10 000 step/s)
87  #define L6474_CONF_PARAM_MIN_SPEED_DEVICE_0   (300)
88  /// Minimum speed in step/s for device 1 (30 step/s <= Minimum speed < 10 000 step/s)
89  #define L6474_CONF_PARAM_MIN_SPEED_DEVICE_1   (300)
90
```

```
93  /************************ Phase Current Control ******************************/
94
95  // Current value that is assigned to the torque regulation DAC
96  /// TVAL register value for device 0 (range 31.25mA to 4000mA)
97  #define L6474_CONF_PARAM_TVAL_DEVICE_0  (500)
98  /// TVAL register value for device 1 (range 31.25mA to 4000mA)
99  #define L6474_CONF_PARAM_TVAL_DEVICE_1  (500)
100
101
102 /// Fall time value (T_FAST field of T_FAST register) for device 0 (range 2us to 32us)
103 #define L6474_CONF_PARAM_FAST_STEP_DEVICE_0  (L6474_FAST_STEP_12us)
104 /// Fall time value (T_FAST field of T_FAST register) for device 1 (range 2us to 32us)
105 #define L6474_CONF_PARAM_FAST_STEP_DEVICE_1  (L6474_FAST_STEP_12us)
106
107
108 /// Maximum fast decay time (T_OFF field of T_FAST register) for device 0 (range 2us to 32us)
109 #define L6474_CONF_PARAM_TOFF_FAST_DEVICE_0  (L6474_TOFF_FAST_8us)
110 /// Maximum fast decay time (T_OFF field of T_FAST register) for device 1 (range 2us to 32us)
111 #define L6474_CONF_PARAM_TOFF_FAST_DEVICE_1  (L6474_TOFF_FAST_8us)
112
113
114 /// Minimum ON time (TON_MIN register) for device 0 (range 0.5us to 64us)
115 #define L6474_CONF_PARAM_TON_MIN_DEVICE_0 (3)
116 /// Minimum ON time (TON_MIN register) for device 1 (range 0.5us to 64us)
117 #define L6474_CONF_PARAM_TON_MIN_DEVICE_1 (3)
118
119
120 /// Minimum OFF time (TOFF_MIN register) for device 0 (range 0.5us to 64us)
121 #define L6474_CONF_PARAM_TOFF_MIN_DEVICE_0 (21)
122 /// Minimum OFF time (TOFF_MIN register) for device 1 (range 0.5us to 64us)
123 #define L6474_CONF_PARAM_TOFF_MIN_DEVICE_1 (21)
```

13.2. ANEXO B: Documentación técnica.

life.augmented

X-NUCLEO-IHM01A1

Stepper motor driver expansion board based on L6474 for STM32 Nucleo

Data brief

Features

- Voltage range from 8 V to 45 V
- Phase current up to 3 $A_{r.m.s.}$
- Power OK and fault LEDs
- Advanced current control
- Fully protected power stage
- Up to 1/16 microstepping resolution
- Compatible with Arduino UNO R3 connector
- Compatible with STM32 Nucleo boards
- Suitable for multi-motor solutions
- RoHS compliant

Description

The X-NUCLEO-IHM01A1 is a stepper motor driver expansion board based on the L6474.

It provides an affordable and easy-to-use solution for driving a stepper motor in your STM32 Nucleo project.

The advanced current control of the L6474 and a complete set of protection features offer high levels of both performance and robustness.

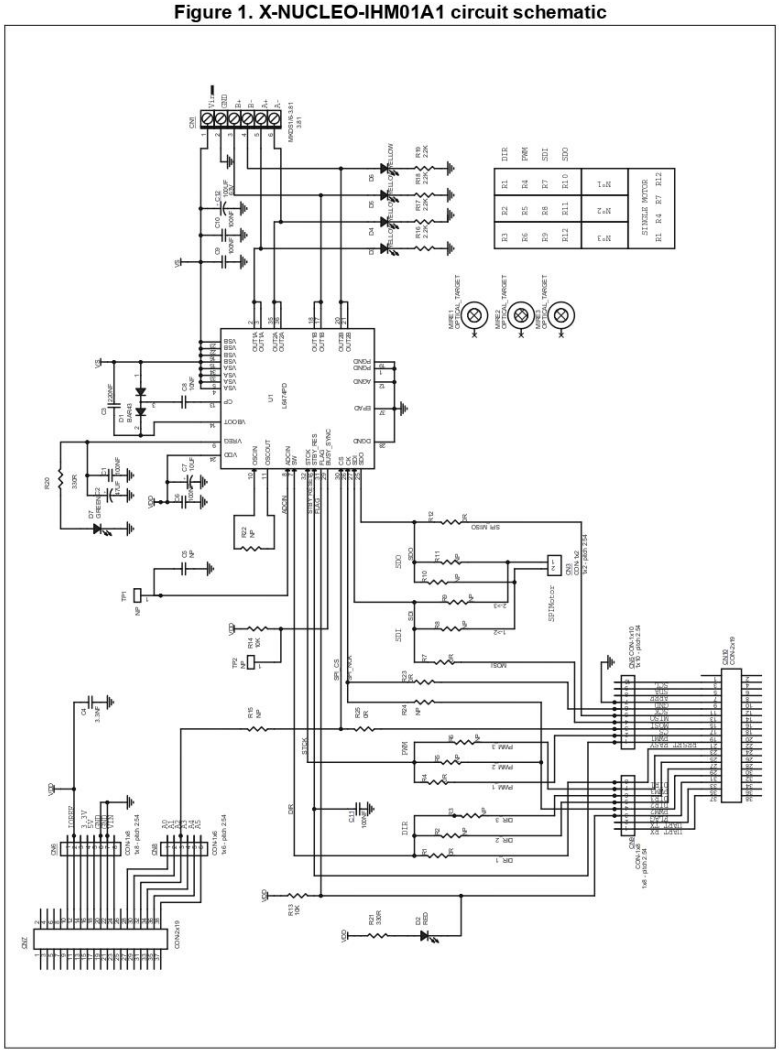
The X-NUCLEO-IHM01A1 is compatible with the Arduino UNO R3 connector, and supports the addition of other boards which can be stacked to drive up to three stepper motors with a single STM32 Nucleo board.

March 2015 DocID026559 Rev 3 1/4

For further information contact your local STMicroelectronics sales office. www.st.com

1 Schematic diagram

Figure 1. X-NUCLEO-IHM01A1 circuit schematic



RS PRO

ENGLISH

Datasheet

Size	Rear shaft	No. of wires	**RS** stock no.
17	No	6	440-420
	Yes		440-436
	No		191-8299
	No		191-8306
23	No	8	440-442
	Yes	8	440-458
	No	8	191-8328
	No	8	191-8334
	No	8	191-8340
	No	8	191-8356
	No	8	191-8362
	No	8	191-8378
	No	8	191-8384
34	Yes	8	440-464
	No	8	440-470

1.8° step angle

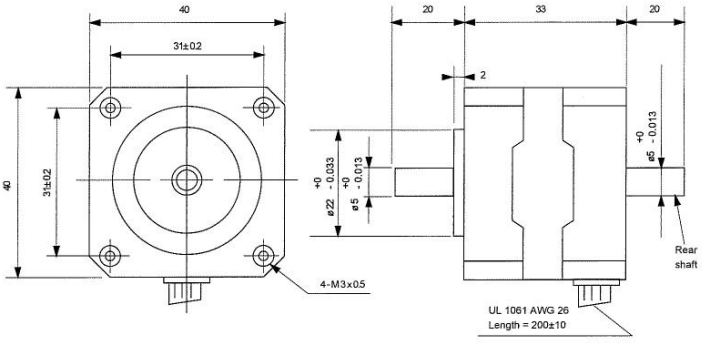
These 4 phase hybrid stepper motors are capable of delivering much higher working torques and stepping rates than permanent magnet (7.5° and 15°) types. Whilst at the same time maintaining a high detent torque even when not energised. This feature is particularly important for positional integrity. Many of the motors are directly compatible with the RS stepper motor drive boards (RS stock nos. 332-098, 342-051 and 440-240).

Size 34 motors and a number of size 23 motors are supplied in 8-lead configuration which allows the maximum flexibility when connecting to the drive boards.

Rear extension shafts are provided on three of the motors to enable connection of other drive requirements and feedback devices.

Size 17

RS stock no. 440-436 with rear shaft shown

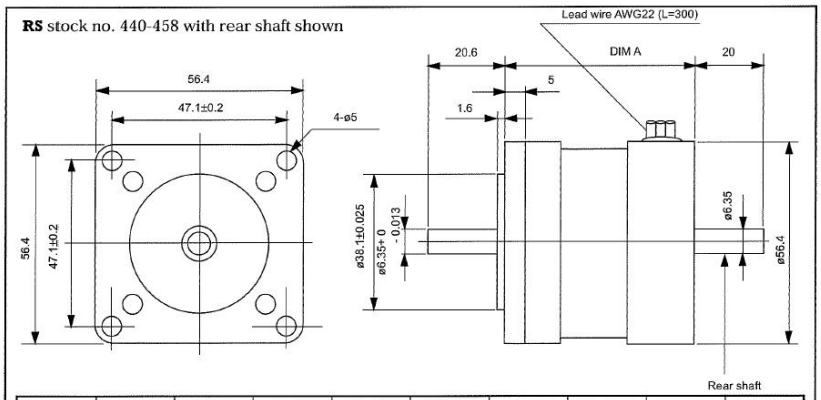


RS, Professionally Approved Products, gives you professional quality parts across all products categories. Our range has been testified by engineers as giving comparable quality to that of the leading brands without paying a premium price.

RS PRO

ENGLISH

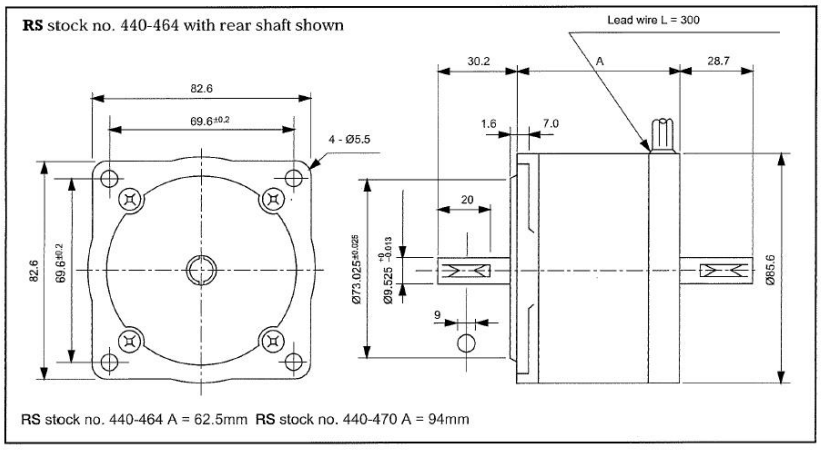
RS stock no. 440-458 with rear shaft shown



RS stock no.	440-442	440-458	191-8328	191-8334	191-8340	191-8356	191-8362	191-8378	191-8384
DIM A	50.5	50.5	38.5	38.5	50.5	50.5	76.0	101.5	101.5

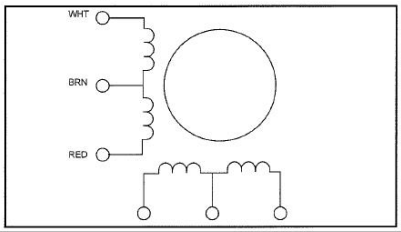
Size 34

RS stock no. 440-464 with rear shaft shown



RS stock no. 440-464 A = 62.5mm **RS** stock no. 440-470 A = 94mm

6 Wire configuration

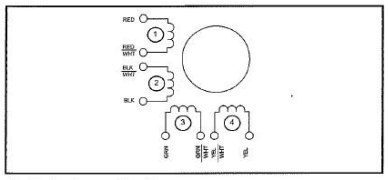


Exciting sequence and direction of rotation when facing mounting flange end.

Step	White	Blue	Red	Yellow	Brown	CW
1	On	On			+dcV	↓
2		On	On			
3			On	On		
4	On			On		

RS, Professionally Approved Products, gives you professional quality parts across all products categories. Our range has been testified by engineers as giving comparable quality to that of the leading brands without paying a premium price.

ENGLISH



Exciting sequence and direction of rotation when facing mounting flange end.

Step	Red	Green	Black	Yellow	Com	CW
1	On	On			+dcV	↓
2		On	On			
3			On	On		
4	On			On		

Technical specification

RS stock no.	440-420	440-436	440-442	440-458	440-464	440-470
Rated voltage (V)	5	12	5	12	3	2.5
Rated current (I)	0.5	0.16	1	0.6	2	4.5
Resistance (Ω)	10	75	5	20	1.5	0.56
Inductance (mH)	6	36	9	32	4.5	2.8
Detent torque (mHm)	5	4	30	30	40	100
Holding torque (mNm)	70	70	500	500	1200	2200
Step angle accuracy (%)	5	5	5	5	5	5
Step angle	1.8	1.8	1.8	1.8	1.8	1.8
Insulation class	B	B	B	B	B	B

RS stock no.	191-8299	191-8306	191-8328	191-8334	191-8340	191-8356	191-8362	191-8378	191-8384
Rated voltage (V)	12	15	5	12	12	12	5.4	3.4	6
Rated current (I)	0.4	0.4	1	0.4	0.48	0.6	1.4	2.85	1.8
Resistance (Ω)	30	45	5	40	25	20	3.8	1.2	3.5
Inductance (mH)	14	22	5.7	40	33	32	6.8	1.5	7.3
Detent torque (mHm)	3.5	3.5	14.8	14.8	29.6	29.6	56.5	77.6	77.6
Holding torque (mNm)	100	100	260	260	494	494	882	1200	1200
Step angle accuracy (%)	5	5	5	5	5	5	5	5	5
Step angle	1.8	1.8	1.8	1.8	1.8	1.8	1.8	1.8	1.8
Insulation class	B	B	B	B	B	B	B	B	B

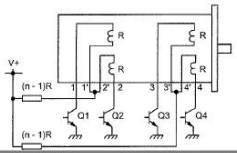
Resonance

Certain operating frequencies cause resonance and the motor loses track of the drive input. Audible vibration may accompany resonance conditions. These frequencies should be avoided if possible. Driving the motor on the half step mode (see motor drive methods) greatly reduces the effect of resonance. Alternatively extra load inertia and external damping may be added to shift resonance regions away from the operating frequency.

Motor drive methods

The normal way of driving a 4-phase stepper motor is shown in Figure 1.

Figure 1 **Unipolar drive**

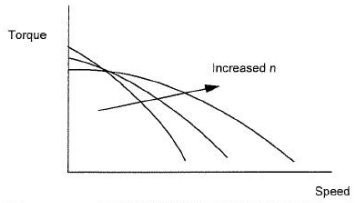


This is commonly known as the 'Unipolar L/nr drive'. Here the current in each winding, when energised, flows in one direction only 'n', value is ≥1 (but not necessarily an integer) and nR is the sum of the external resistance plus the winding resistance (R). By selecting a higher value for n (ie. larger external resistance) and using a higher dc supply to maintain the rated voltage and current for each winding, improved torque speed characteristics can be obtained. Thus a 6V, 6Ω motor (1A per phase) can be driven from a 6Vdc supply without any series resistor, in the L/R mode. Alternatively it can be driven from a 24Vdc supply using 18Ω series resistance in the L/4R mode with much improved performance.

RS PRO

ENGLISH

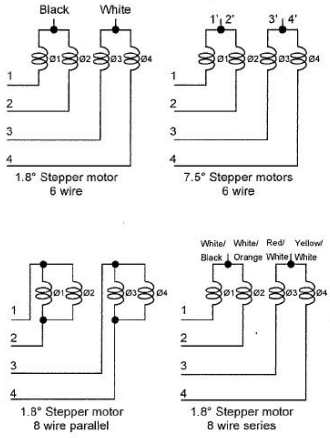
Figure 2 Effect on motor performance of higher supply voltages and larger series limiting resistance



Connection to **RS** bipolar stepper motor board

When the windings of the RS stepper motors are assigned (Ø1-Ø4) as shown in Figure 3, they can be connected to the board according to Figure 1.

Figure 3 Typical motor winding connections



When using 8 lead motors with coils in parallel the motor current should be set no greater than:

$$I \text{ per phase} \times \sqrt{2}$$

When using 6 lead or 8 lead motors with coils in series the motor current should be set no greater than:

$$I \text{ per phase} \times \frac{1}{\sqrt{2}}$$

Motors with 4 leads have a bipolar rating and can be used according to manufacturer's specification.

To step a motor in a particular direction a specific switching sequence for the drive transistors Q1-Q4 needs to be followed. If this sequence is in Table 1 (known as the unipolar full step mode) it results in the rotor advancing through one complete step at a time.

Table 1 Full step mode

Step No.	Q1	Q2	Q3	Q4
	ON	OFF	OFF	ON
1	ON	OFF	ON	OFF
2	OFF	ON	ON	OFF
3	OFF	ON	OFF	ON
4	ON	OFF	OFF	ON
5	ON	OFF	ON	OFF

Start position (arbitrary) → 1
Above sequence repeating → 5
Anti-clockwise ↑
Clockwise ↓

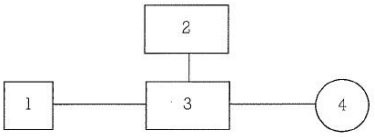
Table 2 Half step mode

Step No.	Q1	Q2	Q3	Q4
	ON	OFF	ON	OFF
1	ON	OFF	OFF	OFF
2	ON	OFF	OFF	ON
3	OFF	OFF	OFF	ON
4	OFF	ON	OFF	ON
5	OFF	ON	OFF	OFF
6	OFF	ON	ON	OFF
7	OFF	OFF	ON	OFF
8	ON	OFF	ON	OFF
9				

Start position → 1
Above sequence repeating → 9
Anti-clockwise ↑
Clockwise ↓

Typical stepper motor control system

The operation of a stepper motor requires the presence of the following elements:

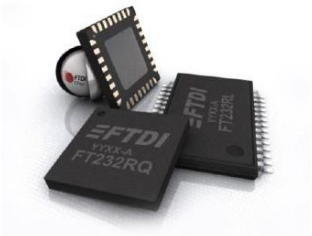


1. **A control unit**. Usually a microprocessor based unit which gives step and direction signals to the drive card. RS stepper motor control board (RS stock no. 440-098) is ideally suited for this function.
2. **Power supply**. Giving the required voltage and current for the drive card using a linear power supply.
3. **Drive card**. This converts the signals from the control unit in to the required stepper motor sequence. RS stock nos. 332-098, 342-051 and 440-240 are designed for the function.
4. **Stepper motor**.

RS, Professionally Approved Products, gives you professional quality parts across all products categories. Our range has been testified by engineers as giving comparable quality to that of the leading brands without paying a premium price.

Future Technology Devices International Ltd.

FT232R USB UART IC Datasheet



The FT232R is a USB to serial UART interface with the following advanced features:

- Single chip USB to asynchronous serial data transfer interface.
- Entire USB protocol handled on the chip. No USB specific firmware programming required.
- Fully integrated 1024 bit EEPROM storing device descriptors and CBUS I/O configuration.
- Fully integrated USB termination resistors.
- Fully integrated clock generation with no external crystal required plus optional clock output selection enabling a glue-less interface to external MCU or FPGA.
- Data transfer rates from 300 baud to 3 Mbaud (RS422, RS485, RS232) at TTL levels.
- 128 byte receive buffer and 256 byte transmit buffer utilising buffer smoothing technology to allow for high data throughput.
- FTDI's royalty-free Virtual Com Port (VCP) and Direct (D2XX) drivers eliminate the requirement for USB driver development in most cases.
- Unique USB FTDIChip-ID™ feature.
- Configurable CBUS I/O pins.
- Transmit and receive LED drive signals.
- UART interface support for 7 or 8 data bits, 1 or 2 stop bits and odd / even / mark / space / no parity
- FIFO receives and transmits buffers for high data throughput.
- Synchronous and asynchronous bit bang interface options with RD# and WR# strobes.
- Device supplied pre-programmed with unique USB serial number.
- Supports bus powered, self-powered and high-power bus powered USB configurations.
- Integrated +3.3V level converter for USB I/O.
- Integrated level converter on UART and CBUS for interfacing to between +1.8V and +5V logic.
- True 5V/3.3V/2.8V/1.8V CMOS drive output and TTL input.
- Configurable I/O pin output drive strength.
- Integrated power-on-reset circuit.
- Fully integrated AVCC supply filtering - no external filtering required.
- UART signal inversion option.
- +3.3V (using external oscillator) to +5.25V (internal oscillator) Single Supply Operation.
- Low operating and USB suspend current.
- Low USB bandwidth consumption.
- UHCI/OHCI/EHCI host controller compatible.
- USB 2.0 Full Speed compatible.
- -40°C to 85°C extended operating temperature range.
- Available in compact Pb-free 28 Pin SSOP and QFN-32 packages (both RoHS compliant).

3 Device Pin Out and Signal Description

3.1 28-LD SSOP Package

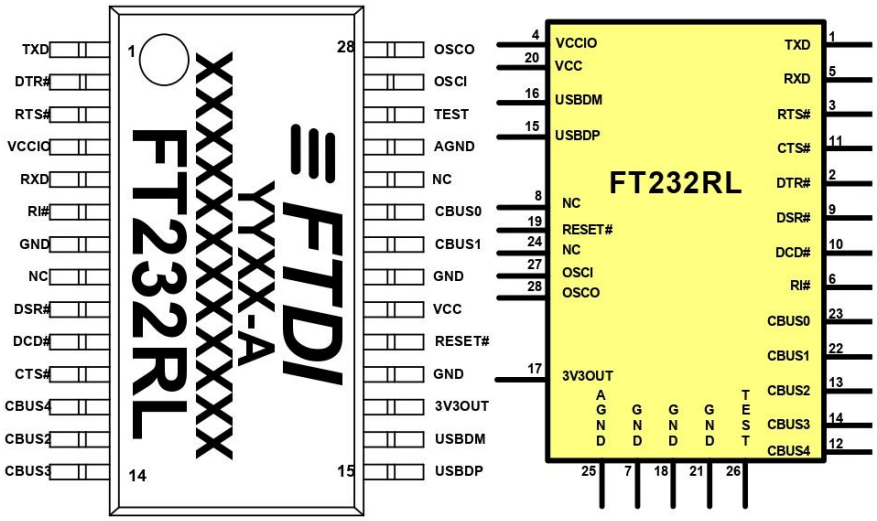


Figure 3.1 SSOP Package Pin Out and Schematic Symbol

3.2 SSOP Package Pin Out Description

Note: The convention used throughout this document for active low signals is the signal name followed by#

Pin No.	Name	Type	Description
15	USBDP	I/O	USB Data Signal Plus, incorporating internal series resistor and 1.5kΩ pull up resistor to 3.3V.
16	USBDM	I/O	USB Data Signal Minus, incorporating internal series resistor.

Table 3.1 USB Interface Group

Pin No.	Name	Type	Description
4	VCCIO	PWR	+1.8V to +5.25V supply to the UART Interface and CBUS group pins (1...3, 5, 6, 9...14, 22, 23). In USB bus powered designs connect this pin to 3V3OUT pin to drive out at +3.3V levels, or connect to VCC to drive out at 5V CMOS level. This pin can also be supplied with an external +1.8V to +2.8V supply in order to drive outputs at lower levels. It should be noted that in this case this supply should originate from the same source as the supply to VCC. This means that in bus powered designs a regulator which is supplied by the +5V on the USB bus should be used.
7, 18, 21	GND	PWR	Device ground supply pins
17	3V3OUT	Output	+3.3V output from integrated LDO regulator. This pin should be decoupled to ground using a 100nF capacitor. The main use of this pin is to provide the internal +3.3V supply to the USB transceiver cell and the

Pin No.	Name	Type	Description
			internal 1.5kΩ pull up resistor on USBDP. Up to 50mA can be drawn from this pin to power external logic if required. This pin can also be used to supply the VCCIO pin.
20	VCC	PWR	+3.3V to +5.25V supply to the device core. (see Note 1)
25	AGND	PWR	Device analogue ground supply for internal clock multiplier

Table 3.2 Power and Ground Group

Pin No.	Name	Type	Description
8, 24	NC	NC	No internal connection
19	RESET#	Input	Active low reset pin. This can be used by an external device to reset the FT232R. If not required can be left unconnected, or pulled up to VCC.
26	TEST	Input	Puts the device into IC test mode. Must be tied to GND for normal operation, otherwise the device will appear to fail.
27	OSCI	Input	Input 12MHz Oscillator Cell. Optional – Can be left unconnected for normal operation. (see Note 2)
28	OSCO	Output	Output from 12MHZ Oscillator Cell. Optional – Can be left unconnected for normal operation if internal Oscillator is used. (see Note 2)

Table 3.3 Miscellaneous Signal Group

Pin No.	Name	Type	Description
1	TXD	Output	Transmit Asynchronous Data Output.
2	DTR#	Output	Data Terminal Ready Control Output / Handshake Signal.
3	RTS#	Output	Request to Send Control Output / Handshake Signal.
5	RXD	Input	Receiving Asynchronous Data Input.
6	RI#	Input	Ring Indicator Control Input. When remote wake up is enabled in the internal EEPROM taking RI# low (20ms active low pulse) can be used to resume the PC USB host controller from suspend.
9	DSR#	Input	Data Set Ready Control Input / Handshake Signal.
10	DCD#	Input	Data Carrier Detect Control Input.
11	CTS#	Input	Clear To Send Control Input / Handshake Signal.
12	CBUS4	I/O	Configurable CBUS output only Pin. Function of this pin is configured in the device internal EEPROM. Factory default configuration is SLEEP#. See CBUS Signal Options, Table 3.9.
13	CBUS2	I/O	Configurable CBUS I/O Pin. Function of this pin is configured in the device internal EEPROM. Factory default configuration is TXDEN. See CBUS Signal Options, Table 3.9.
14	CBUS3	I/O	Configurable CBUS I/O Pin. Function of this pin is configured in the device internal EEPROM. Factory default configuration is PWREN#. See CBUS Signal Options, Table 3.9. PWREN# should be used with a 10kΩ resistor pull up.
22	CBUS1	I/O	Configurable CBUS I/O Pin. Function of this pin is configured in the device internal EEPROM. Factory default configuration is RXLED#. See CBUS Signal Options, Table 3.9.
23	CBUS0	I/O	Configurable CBUS I/O Pin. Function of this pin is configured in the device internal EEPROM. Factory default configuration is TXLED#. See CBUS Signal Options, Table 3.9.

Table 3.4 UART Interface and CUSB Group (see note 3)

Notes:

1. The minimum operating voltage VCC must be +4.0V (could use VBUS=+5V) when using the internal clock generator. Operation at +3.3V is possible using an external crystal oscillator.
2. For details on how to use an external crystal, ceramic resonator, or oscillator with the FT232R, please refer Section 7.6
3. When used in Input Mode, the input pins are pulled to VCCIO via internal 200kΩ resistors. These pins can be programmed to gently pull low during USB suspend (PWREN# = "1") by setting an option in the internal EEPROM.

7.4 USB to MCU UART Interface

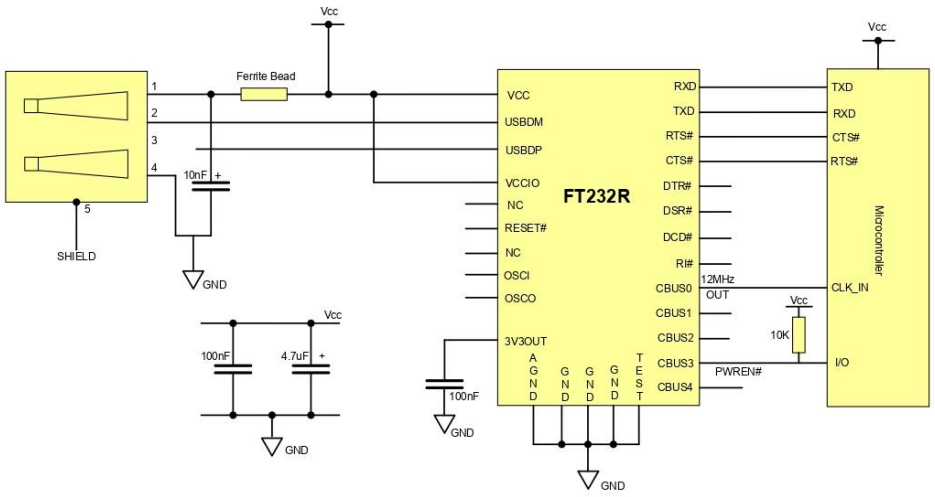


Figure 7.4 USB to MCU UART Interface

An example of using the FT232R as a USB to Microcontroller (MCU) UART interface is shown in Figure 7.4. In this application the FT232R uses TXD and RXD for transmission and reception of data, and RTS# / CTS# signals for hardware handshaking. Also in this example CBUS0 has been configured as a 12MHz output to clock the MCU.

Optionally, RI# could be connected to another I/O pin on the MCU and used to wake up the USB host controller from suspend mode. If the MCU is handling power management functions, then a CBUS pin can be configured as PWREN# and would also be connected to an I/O pin of the MCU.

8 Internal EEPROM Configuration

Following a power-on reset or a USB reset the FT232R will scan its internal EEPROM and read the USB configuration descriptors stored there. The default factory programmed values of the internal EEPROM are shown in Table 8.1.

Parameter	Value	Notes
USB Vendor ID (VID)	0403h	FTDI default VID (hex)
USB Product UD (PID)	6001h	FTDI default PID (hex)
Serial Number Enabled?	Yes	
Serial Number	See Note	A unique serial number is generated and programmed into the EEPROM during device final test.
Pull down I/O Pins in USB Suspend	Disabled	Enabling this option will make the device pull down on the UART interface lines when in USB suspend mode (PWREN# is high).
Manufacturer Name	FTDI	
Product Description	FT232R USB UART	
Max Bus Power Current	90mA	
Power Source	Bus Powered	
Device Type	FT232R	
USB Version	0200	Returns USB 2.0 device description to the host. Note: The device is a USB 2.0 Full Speed device (12Mb/s) as opposed to a USB 2.0 High Speed device (480Mb/s).
Remote Wake Up	Enabled	Taking RI# low will wake up the USB host controller from suspend in approximately 20 ms.
High Current I/Os	Disabled	Enables the high drive level on the UART and CBUS I/O pins.
Load VCP Driver	Enabled	Makes the device load the VCP driver interface for the device.
CBUS0	TXLED#	Default configuration of CBUS0 – Transmit LED drive.
CBUS1	RXLED#	Default configuration of CBUS1 – Receive LED drive.
CBUS2	TXDEN	Default configuration of CBUS2 – Transmit data enable for RS485
CBUS3	PWREN#	Default configuration of CBUS3 – Power enable. Low after USB enumeration, high during USB suspend mode.
CBUS4	SLEEP#	Default configuration of CBUS4 – Low during USB suspend mode.
Invert TXD	Disabled	Signal on this pin becomes TXD# if enable.
Invert RXD	Disabled	Signal on this pin becomes RXD# if enable.
Invert RTS#	Disabled	Signal on this pin becomes RTS if enable.
Invert CTS#	Disabled	Signal on this pin becomes CTS if enable.
Invert DTR#	Disabled	Signal on this pin becomes DTR if enable.
Invert DSR#	Disabled	Signal on this pin becomes DSR if enable.
Invert DCD#	Disabled	Signal on this pin becomes DCD if enable.
Invert RI#	Disabled	Signal on this pin becomes RI if enable.

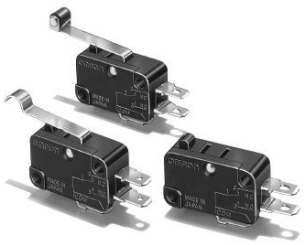
Table 8.1 Default Internal EEPROM Configuration

The internal EEPROM in the FT232R can be programmed over USB using the FTDI utility program FT_PROG. FT_PROG can be downloaded from FTDI Utilities on the FTDI website (www.ftdichip.com). Version 2.8a or later is required for the FT232R chip. Users who do not have their own USB Vendor ID but who would like to use a unique Product ID in their design can apply to FTDI for a free block of unique PIDs. Contact FTDI support for this service.

Miniature Basic Switch that Offers High Reliability and Security

- Wide variation of best-selling microswitches with switching currents of 10 to 21 A.
- Can be used for interrupting current when doors are opened or closed.
- Available in two types of cases: thermoplastic resin and thermosetting resin.
- Indium contact models available for DC load

RoHS Compliant



Model Number Legend

V-[1][2]-[3][4][5][6]

1. Ratings
21: 21 A at 250 VAC
16: 16 A at 250 VAC
15: 15 A at 250 VAC
10: 10 A at 250 VAC

2. Actuator
None : Pin plunger
1 : Short hinge lever
2 : Hinge lever
3 : Long hinge lever
4 : Simulated roller lever
5 : Short hinge roller lever
6 : Hinge roller lever

3. Contact Form
1: SPDT
2: SPST-NC
3: SPST-NO

4. Terminals
A : Solder terminals
C2 : Quick-connect terminals (#187)
C : Quick-connect terminals (#250)

5. Maximum Operating Force
6: 3.92 N {400 gf}
5: 1.96 N {200 gf}
4: 0.98 N {100 gf}
Note: These values are for the pin plunger models.

6. Heat Resistance
None : Standard (80°C)
-T : Heat-resistive (150°C)

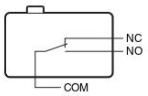
Actuator	Terminals	Contact form	Ratings: Maximum operating force (OF)	15A	10A	Heat-resistive: 15A	Heat-resistive: 10A
Hinge roller lever 	Solder terminals (A)	SPDT	2.45N	V-156-1A6	---	V-156-1A6-T	---
		SPST-NC		V-156-2A6	---	---	---
		SPST-NO		V-156-3A6	---	---	---
		SPDT	1.23N	V-156-1A5	V-106-1A5	V-156-1A5-T	V-106-1A5-T
		SPST-NC		V-156-2A5	V-106-2A5	---	---
		SPST-NO		V-156-3A5	V-106-3A5	---	---
		SPDT	0.59N	---	V-106-1A4	---	V-106-1A4-T
		SPST-NC		---	V-106-2A4	---	---
		SPST-NO		---	V-106-3A4	---	---
	Quick-connect terminals (#187) (C2)	SPDT	2.45N	V-156-1C26	---	V-156-1C26-T	---
		SPST-NC		V-156-2C26	---	---	---
		SPST-NO		V-156-3C26	---	---	---
		SPDT	1.23N	V-156-1C25	V-106-1C25	V-156-1C25-T	V-106-1C25-T
		SPST-NC		V-156-2C25	V-106-2C25	---	---
		SPST-NO		V-156-3C25	V-106-3C25	---	---
		SPDT	0.59N	---	V-106-1C24	---	V-106-1C24-T
		SPST-NC		---	V-106-2C24	---	---
		SPST-NO		---	V-106-3C24	---	---
	Quick-connect terminals (#250) (C)	SPDT	2.45N	V-156-1C6	---	V-156-1C6-T	---
		SPST-NC		V-156-2C6	---	---	---
		SPST-NO		V-156-3C6	---	---	---
		SPDT	1.23N	V-156-1C5	V-106-1C5	V-156-1C5-T	V-106-1C5-T
		SPST-NC		V-156-2C5	V-106-2C5	---	---
		SPST-NO		V-156-3C5	V-106-3C5	---	---
		SPDT	0.59N	---	V-106-1C4	---	V-106-1C4-T
		SPST-NC		---	V-106-2C4	---	---
		SPST-NO		---	V-106-3C4	---	---

For DC load (V-21(IN) models)

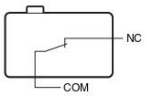
Actuator	Terminals	Contact form	Ratings: Maximum operating force (OF)	30VDC 12A
Pin plunger	Quick-connect terminals (#250) (C)	SPDT	3.92N	V-21-1C6(IN)

Contact form

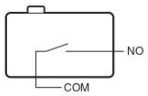
SPDT



SPST-NC

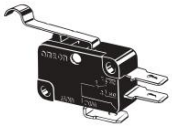


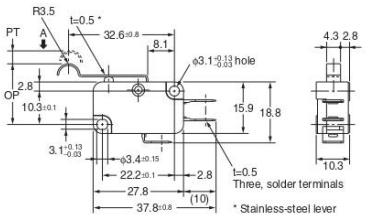
SPST-NO



Refer to **"Micro Switch Common Accessories"** for Separators (sold separately), Actuators (sold separately) and Terminal Connectors (sold separately).

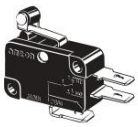
●Simulated roller lever
V-154-1□6
V-154-1□5
V-104-1□5
V-104-1□4

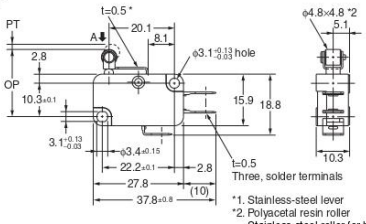




Operating characteristics	Model	V-154-1□6	V-154-1□5 V-104-1□5	V-104-1□4
OF max.		2.45N	1.23N	0.59N
RF min.		0.25N	0.14N	0.06N
PT max.		4.0mm		
OT min.		1.6mm		
MD max.		1.5mm		
OP		18.7±1.2mm		

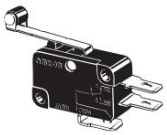
●Short hinge roller lever
V-155-1□6
V-155-1□5
V-105-1□5
V-105-1□4

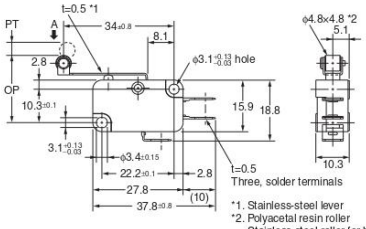




Operating characteristics	Model	V-155-1□6	V-155-1□5 V-105-1□5	V-105-1□4
OF max.		4.71N	2.35N	1.18N
RF min.		0.49N	0.49N	0.15N
PT max.		1.6mm		
OT min.		0.8mm		
MD max.		0.6mm		
OP		20.7±0.6mm		

●Hinge roller lever
V-156-1□6
V-156-1□5
V-106-1□5
V-106-1□4





Operating characteristics	Model	V-156-1□6	V-156-1□5 V-106-1□5	V-106-1□4
OF max.		2.45N	1.23N	0.59N
RF min.		0.25N	0.14N	0.06N
PT max.		4.0mm		
OT min.		1.6mm		
MD max.		1.5mm		
OP		20.7±1.2mm		

Note 1. Unless otherwise specified, a tolerance of ±0.4 mm applies to all dimensions.
Note 2. The operating characteristics are for operation in the A direction (⬇).

Precautions

★Please read "Common Precautions" for correct use.

Precautions for Safe Use

●Soldering

- Connecting to Solder Terminals
  Complete the soldering at the iron tip temperature of 250 to 350°C (60W) within 5 seconds, and do not apply any external force for 1 minute after soldering.
  Be sure to apply only the minimum required amount of flux.It may result in contact failure once the flux penetrates into the internal part of the Switch.
- Connecting to Quick-connect Terminals #187
  Insert the receptacle of quick-connect terminal #187 straight toward the terminal.
  Applying excessive external force horizontally or vertically may cause deformation of terminals and may damage the housings.
- Connecting to Quick-connect Terminals #250
  Insert the receptacle of quick-connect terminal #250 straight toward the terminal.
  Applying excessive external force horizontally or vertically may cause deformation of terminals and may damage the housings.

Precautions for Correct Use

●Mounting

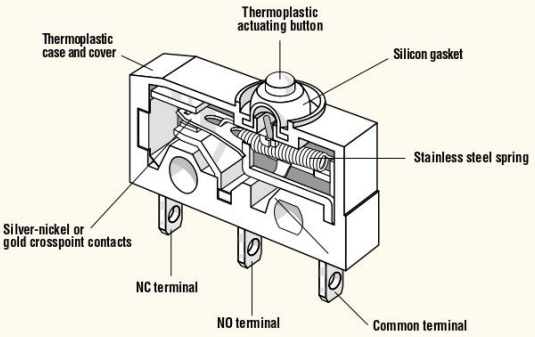
Use M3 mounting screw with plane washers or spring washers to securely mount the Switch.Tighten the screws to a torque of 0.39 to 0.59N·m {4 to 6 kgf·cm}.

SUBMINIATURE — SEALED

DC Series

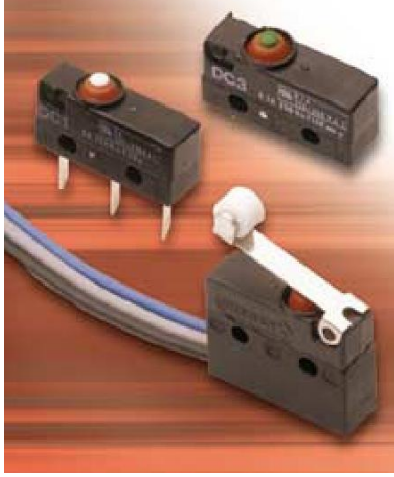
Features

- Enclosed switch complying with IP 6K7
- Silicon-free version available
- Models available for operating temperatures up to 120°C
- Rated for currents up to 10 amp at 250VAC
- Range of auxiliary actuators available (can also be retrofitted)
- Various contact materials available, depending on application
- Mechanical life: min. 1 x10^6 operations
- Wide variety of connection options



Electrical Ratings

Switch Series	EN61058 Rating	UL1054 Rating	Electrical Life at Rated Load According to EN (Min. Operations)	According to UL (Min. Operations)
DC1	6A, 250V~	5A, 125/250VAC	10,000	6,000
DC2	10(1.5)A, 250V~	10.1A, 125/250VAC; 1/4HP, 125VAC	10,000	6,000
DC3	0.1A, 250V~	0.1A, 125/250VAC	50,000	6,000
DC4	3A, 250V~	3A, 125/250VAC	50,000	6,000



Specifications

Electrical

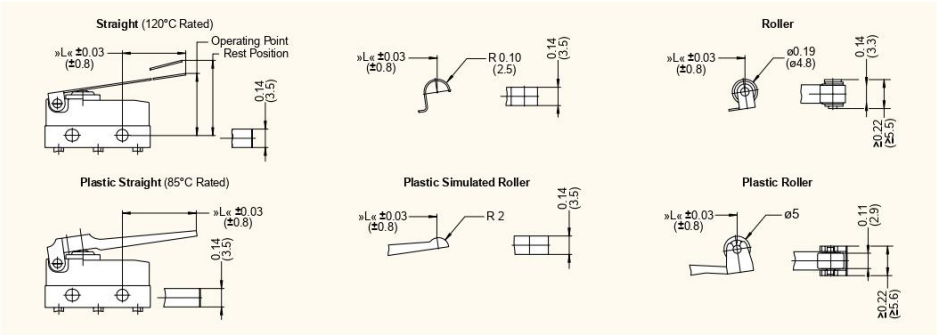
Temperature Rating:	-40° to +85°C /+120°C *(without wire leads)* -40° to +85C / +105°C *(with wire leads)*
Flammability Rating:	UL94V-O *(PET, PBT)* UL94HB *(POM)*

Materials

Case:	PET / PBT
Actuator:	POM *(T85)*, PBT *(T120)*
Auxiliary Actuator:	Stainless Steel or Plastic
Terminals:	Silver-Plated Copper-Zinc
Contacts:	Silver Alloy *(DC3 — Gold Crosspoint)*

PBT = Polybutyleneterephthalate • PET = Polyethyleneterephthalate • POM = Polyacetal

Auxiliary Actuators inches (mm)

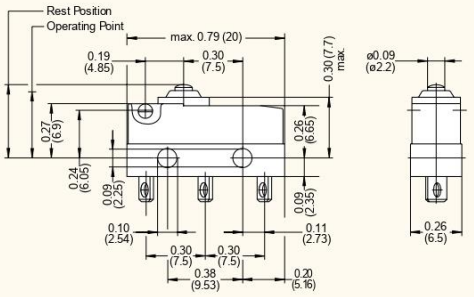


Actuator Specifications

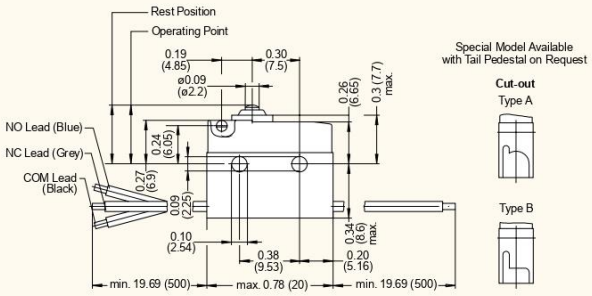
Actuator Code	Switch Type	Maximum Operating Force (cN) DC1, DC3, DC4	Maximum Operating Force (cN) DC2	Maximum Pre-Travel inches (mm) DC1, DC3, DC4	Maximum Pre-Travel inches (mm) DC2	Operating Point inches (mm) DC1, DC3, DC4	Operating Point inches (mm) DC2	Minimum Over-Travel inches (mm)	Max. Movement Differential inches (mm) DC1, DC3, DC4	Max. Movement Differential inches (mm) DC2	Max. Rest Position inches (mm)	Actuation Length inches (mm)
AA		200	340	0.04 (1.0)	0.04 (1.0)	0.33±0.01 (8.4±0.3)	0.33±0.01 (8.4±0.3)	0.02 (0.6)	0.004 (0.10)	0.004 (0.10)	0.37 (9.3)	— —
LB		80	150	0.18 (4.5)	0.20 (5.0)	0.42±0.05 (10.7±1.3)	0.42±0.06 (10.7±1.6)	0.06 (1.5)	0.02 (0.5)	0.03 (0.7)	0.55 (14.0)	0.189 (4.8)
LC		70	120	0.20 (5.0)	0.22 (5.5)	0.44±0.06 (11.1±1.5)	0.44±0.07 (11.1±1.8)	0.06 (1.5)	0.02 (0.6)	0.04 (1.0)	0.59 (15.0)	0.276 (7.0)
LD		*forces and travel available upon request*										1.654 (42.0)
SB		90	160	0.18 (4.5)	0.20 (5.0)	0.63±0.05 (16.0±1.3)	0.63±0.06 (16.0±1.6)	0.06 (1.5)	0.02 (0.5)	0.03 (0.7)	0.75 (19.0)	0.098 (2.5)
SC		80	130	0.20 (5.0)	0.22 (5.5)	0.65±0.06 (16.4±1.5)	0.65±0.07 (16.4±1.8)	0.06 (1.5)	0.02 (0.6)	0.04 (1.0)	0.79 (20.0)	0.185 (4.7)
SD		*forces and travel available upon request*										1.563 (39.7)
RB		90	160	0.18 (4.5)	0.20 (5.0)	0.62±0.05 (15.8±1.3)	0.62±0.06 (15.8±1.6)	0.06 (1.5)	0.02 (0.5)	0.03 (0.7)	0.75 (19.0)	0.098 (2.5)
RC		80	130	0.20 (5.0)	0.22 (5.5)	0.64±0.06 (16.2±1.5)	0.64±0.07 (16.2±1.8)	0.06 (1.5)	0.02 (0.6)	0.04 (1.0)	0.79 (20.0)	0.185 (4.7)
RD		*forces and travel available upon request*										1.563 (39.7)
WB		68	115	0.18 (4.5)	0.18 (4.5)	0.44±0.06 (11.1±1.5)	0.44±0.06 (11.1±1.5)	0.08 (2.0)	0.02 (0.6)	0.02 (0.6)	0.59 (15.0)	0.276 (7.0)
WC		50	85	0.24 (6.0)	0.24 (6.0)	0.48±0.07 (12.2±1.8)	0.48±0.07 (12.2±1.8)	0.12 (3.0)	0.03 (0.8)	0.03 (0.8)	0.67 (17.0)	0.551 (14.0)
VB		73	125	0.18 (4.5)	0.18 (4.5)	0.47±0.06 (11.9±1.4)	0.47±0.06 (11.9±1.4)	0.08 (2.0)	0.02 (0.6)	0.02 (0.6)	0.59 (15.0)	0.220 (5.6)
ZB		73	125	.018 (4.5)	0.18 (4.5)	0.63±0.06 (16.0±1.4)	0.63±0.06 (16.0±1.4)	0.06 (1.5)	0.02 (0.6)	0.02 (0.6)	0.75 (19.0)	0.205 (5.2)

CHERRY

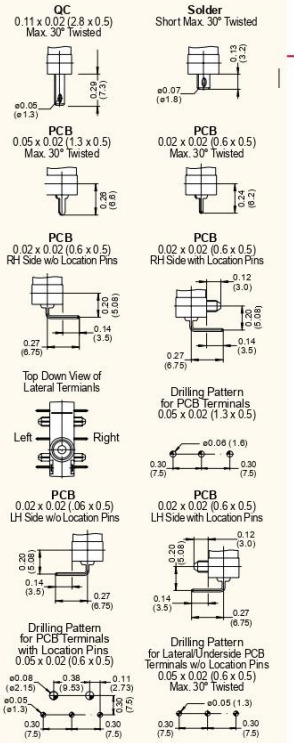
Dimensions inches (mm)



Model with Connecting Leads



Terminal Options inches (mm)



Ordering Information

DC 3 G B 3 L C

Series/Prefix

Code	UL Rating	EN 61058 Rating
1	5A, 125/250VAC	6A, 250V~
2	10.1A, 125/250VAC 1/4HP, 125VAC	10(1.5)A, 250V~
3	0.1A, 125/250VAC	0.1A, 250V~
4[4]	3A, 250VAC	3A, 250V~

Contact Configuration			
+85°C Operating Temp		+105°/120°C Operating Temp	
Code		Code	
E	SPST NO	A	SPST NO
F	SPST NC	B	SPST NC
G	SPDT	C	SPDT

Code	Terminal Type inches (mm)	Notes
A	Short Solder	1
B	0.02[1] (0.5[2]) Leads *(No UL)*	5, 9
C	0.03[1] (0.75[2]) Leads *(No UL)*	5
H	0.05 x 0.02 (1.3 x 0.5) PCB	1
K	0.02 x 0.02 (0.6 x 0.5) PCB	1, 7
L	0.11 x 0.02 (2.8 x 0.5) QC	1
N	Without Leads *(Type B)*	6
P	Without Leads *(Type A)*	6
Q	Without Leads *(No Cut-Out)*	8
S	Welding	1
T	Leads, 20 AWG	3, 11
U	Leads, 22AWG	9, 11
Only Available in North America		
D	Leads 18 AWG *(UL Only)*	5, 8, 10
E	Leads 16 AWG *(UL Only)*	5
M	Leads 20 AWG *(UL Only)*	3, 5

Code	Actuator Type	Actuation Length inches (mm) A	B	C	D
AA	Button only	—	—	—	—
L	Lever	—	0.189 (4.8)	0.276 (7.0)	1.654 (42.0)
R	Roller	—	0.098 (2.5)	0.185 (4.7)	1.563 (39.7)
S	Simulated Roller	—	0.098 (2.5)	0.185 (4.7)	1.563 (39.7)
V[2]	Plastic Sim. Roller	—	0.220 (5.6)	—	—
W[2]	Plastic Straight	—	0.276 (7.0)	0.551 (14.0)	—
Z[2]	Plastic with Roller	—	0.205 (5.2)	—	—

Code	Terminal Configuration
1	30° Twisted
3	Leads on Actuator Side
4	Leads on Opposite Side from Actuator
5	Leads Attached to Underside
6	PCB Terminals Right-hand Side
7	PCB Terminals Left-hand Side
8	PCB Terminals Right-hand Side with Location Pins
9	PCB Terminals Left-hand Side with Location Pins

Notes:
Wire leads 500mm min. (approximately 20" long)
1. Only available with terminal configuration 1.
2. +85°C rated only.
3. Only available with DC1, DC3 and DC4.
4. Rating code 4 only as lead-version — leads 0.5mm².
5. Wire versions only rated to +105°C.
6. Only available with terminal configuration 3 or 4.
7. Only available with terminal configuration 1, 6, 7, 8 or 9.
8. Only available with terminal configuration 5.
9. Only available with DC3 and DC4.
10. Not available with DC2.
11. Not available in North America.

Specifications subject to change without notice.

life.augmented

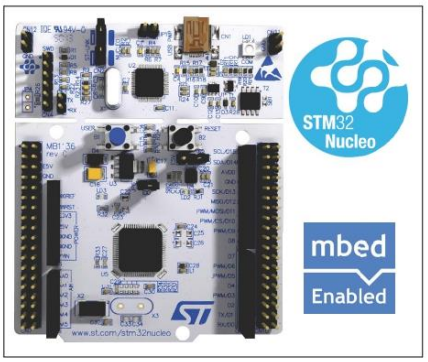
NUCLEO-XXXXRX

STM32™ Nucleo boards

Data brief

Features

- STM32™ microcontroller with LQFP64 package
- Two types of extension resources
  - Arduino Uno Revision 3 connectivity
  - STMicroelectronics Morpho extension pin headers for full access to all STM32 I/Os
- mbed-enabled (*mbed.org*)
- On-board ST-LINK/V2-1 debugger/programmer with SWD connector
  - selection-mode switch to use the kit as a standalone ST-LINK/V2-1
- Flexible board power supply
  - USB VBUS or external source (3.3 V, 5 V, 7 - 12 V)
  - Power management access point
- Three LEDs
  - USB communication (LD1), user LED (LD2), power LED (LD3)
- Two push buttons: USER and RESET
- USB re-enumeration capability: three different interfaces supported on USB
  - Virtual Com port
  - Mass storage
  - Debug port
- Supported by wide choice of Integrated Development Environments (IDEs) including IAR™, Keil®, GCC-based IDEs



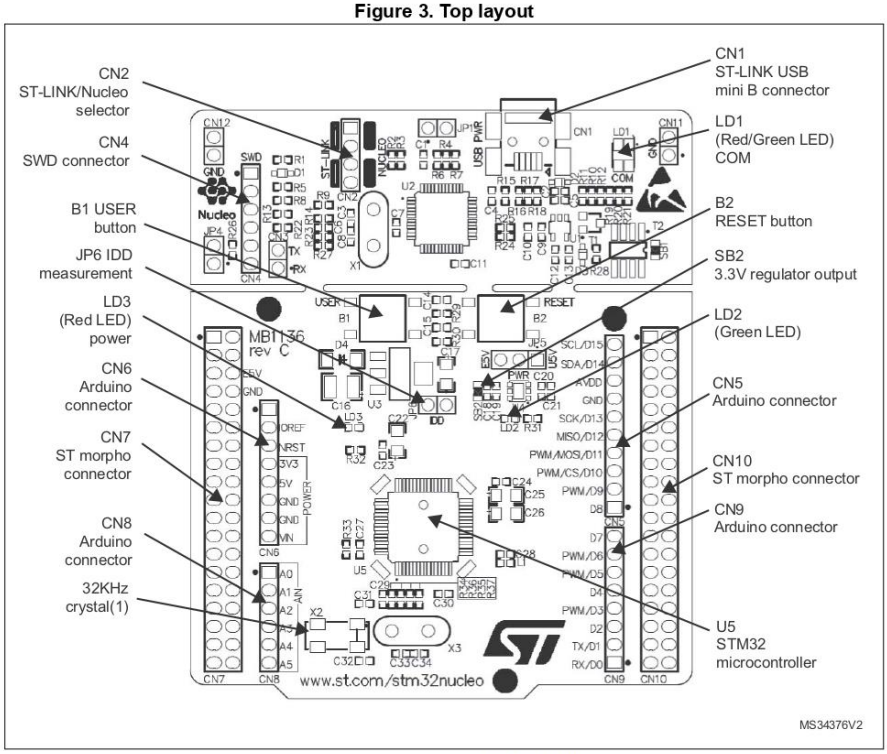
Description

The STM32 Nucleo board provides an affordable and flexible way for users to try out new ideas and build prototypes with any STM32 microcontroller line, choosing from the various combinations of performance, power consumption and features. The Arduino™ connectivity support and ST Morpho headers make it easy to expand the functionality of the STM32 Nucleo open development platform with a wide choice of specialized shields.
The STM32 Nucleo board does not require any separate probe as it integrates the ST-LINK/V2-1 debugger/programmer.
The STM32 Nucleo board comes with the STM32 comprehensive software HAL library together with various packaged software examples, as well as direct access to mbed online resources.

Table 1. Device summary

Reference	Part number
NUCLEO-XXXXRX	NUCLEO-F030R8, NUCLEO-F072RB, NUCLEO-F091RC, NUCLEO-F103RB, NUCLEO-F302R8, NUCLEO-F303RE, NUCLEO-F334R8, NUCLEO-F401RE, NUCLEO-F411RE NUCLEO-L053R8, NUCLEO-L152RE

Figure 3. Top layout



1. Crystal may be present or not depending on board version, refer to *Section 6.7.2*.

Figure 4. Bottom layout

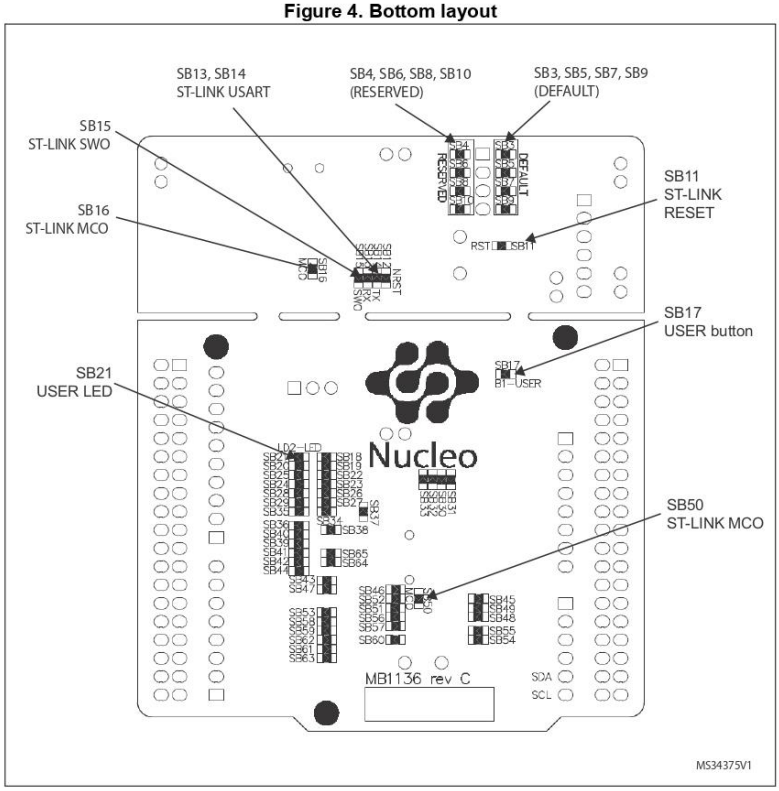


Figure 5. STM32 Nucleo board mechanical dimensions

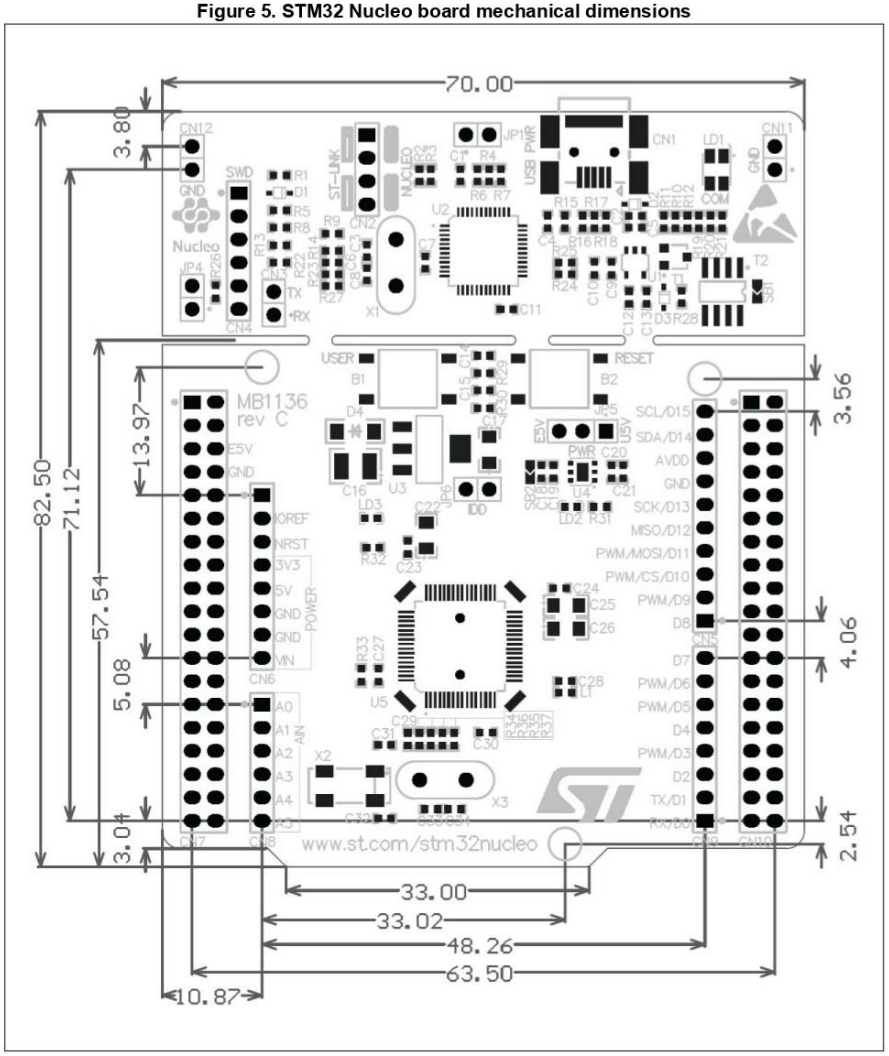


Figure 20. NUCLEO-L053R8

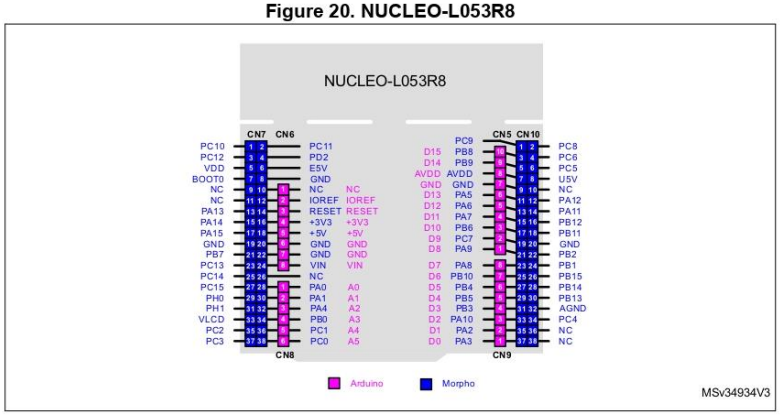
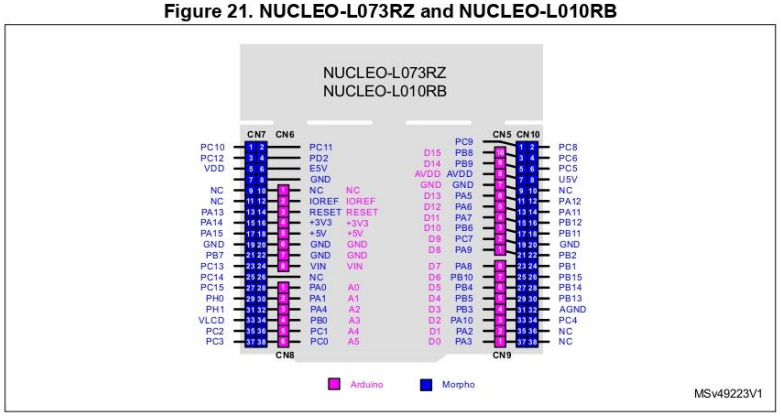


Figure 21. NUCLEO-L073RZ and NUCLEO-L010RB



Appendix A Electrical schematics

Figure 27 to *Figure 30* show the electrical schematics of the STM32 Nucleo-64 board.

Figure 27. Top and Power

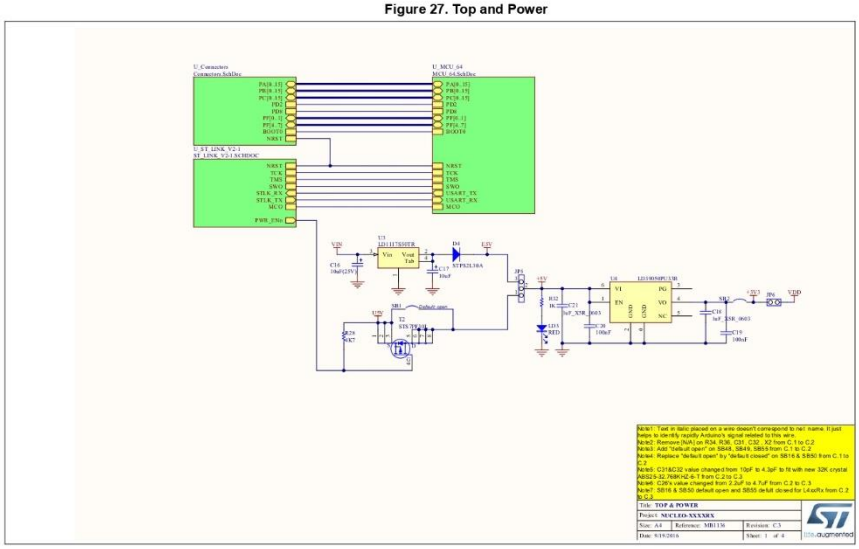


Figure 28. STM32 MCU

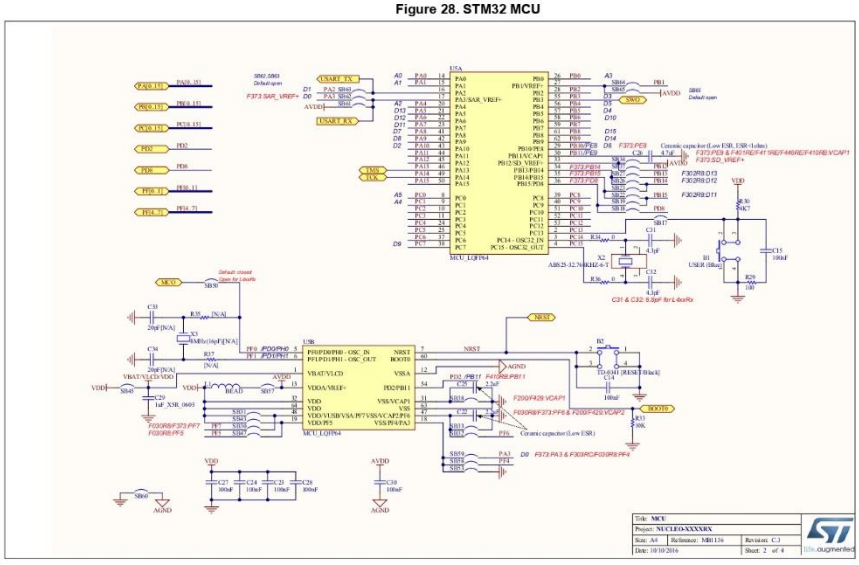


Figure 29. ST-LINK/V2-1

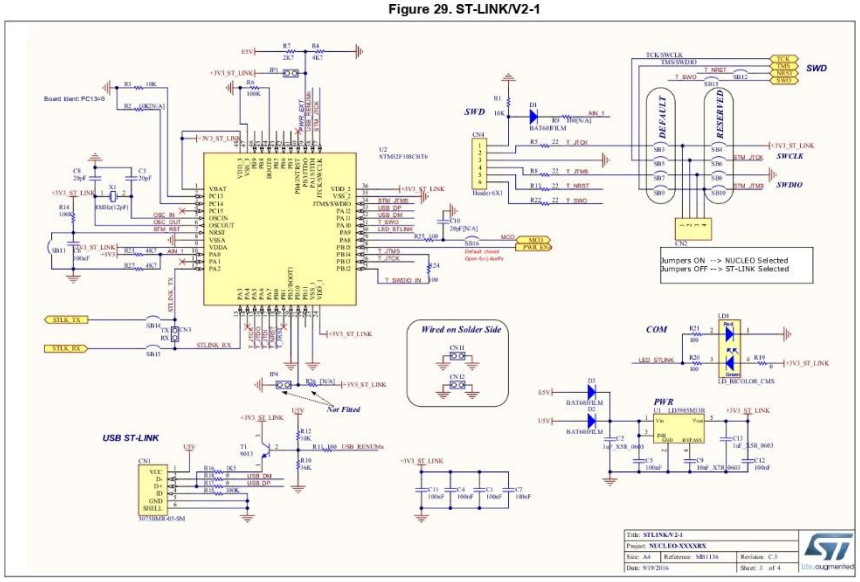
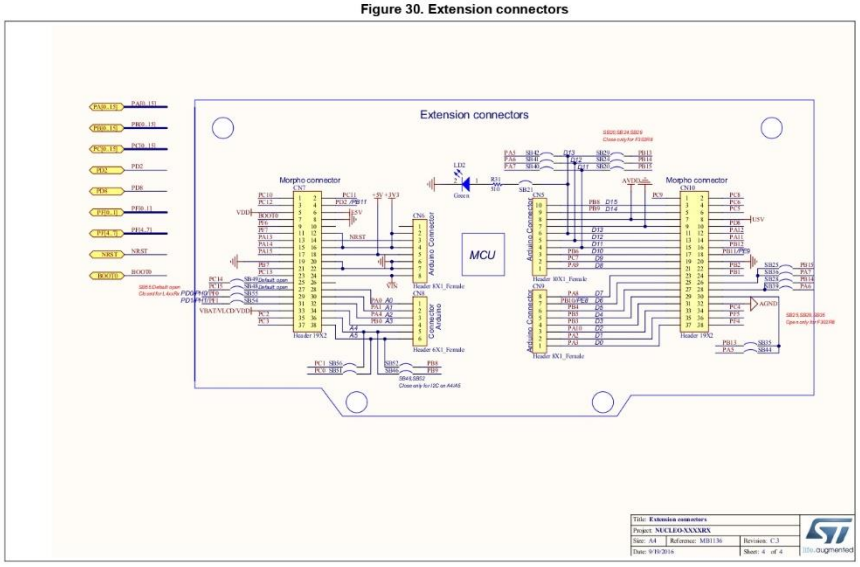


Figure 30. Extension connectors



SONGLE RELAY

松乐继电器 ® SONGLE RELAY	RELAY ISO9002	SRD

1. MAIN FEATURES

- Switching capacity available by 10A in spite of small size design for highdensity P.C. board mounting technique.
- UL,CUL,TUV recognized.
- Selection of plastic material for high temperature and better chemical solution performance.
- Sealed types available.
- Simple relay magnetic circuit to meet low cost of mass production.

2. APPLICATIONS

- Domestic appliance, office machine, audio, equipment, automobile, etc.
  (Remote control TV receiver, monitor display, audio equipment high rushing current use application.)

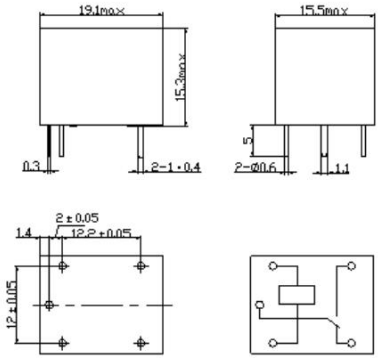
3. ORDERING INFORMATION

SRD	XX VDC	S	L	C
Model of relay	Nominal coil voltage	Structure	Coil sensitivity	Contact form
SRD	03、05、06、09、12、24、48VDC	S:Sealed type	L:0.36W	A:1 form A
				B:1 form B
		F:Flux free type	D:0.45W	C:1 form C

4. RATING

CCC	FILE NUMBER:CH0052885-2000	7A/240VDC
CCC	FILE NUMBER:CH0036746-99	10A/250VDC
UL /CUL	FILE NUMBER: E167996	10A/125VAC 28VDC
TUV	FILE NUMBER: R9933789	10A/240VAC 28VDC

5. DIMENSION(unit:mm) DRILLING(unit:mm) WIRING DIAGRAM



6. COIL DATA CHART (AT20°C)

Coil Sensitivity	Coil Voltage Code	Nominal Voltage (VDC)	Nominal Current (mA)	Coil Resistance (Ω) ±10%	Power Consumption (W)	Pull-In Voltage (VDC)	Drop-Out Voltage (VDC)	Max-Allowable Voltage (VDC)
SRD (High Sensitivity)	03	03	120	25	abt. 0.36W	75%Max.	10% Min.	120%
	05	05	71.4	70				
	06	06	60	100				
	09	09	40	225				
	12	12	30	400				
	24	24	15	1600				
	48	48	7.5	6400				
SRD (Standard)	03	03	150	20	abt. 0.45W	75% Max.	10% Min.	110%
	05	05	89.3	55				
	06	06	75	80				
	09	09	50	180				
	12	12	37.5	320				
	24	24	18.7	1280				
	48	48	10	4500	abt. 0.51W			

7. CONTACT RATING

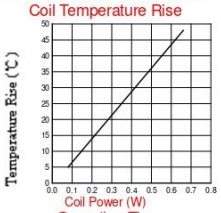
Item \ Type	SRD	
	FORM C	FORM A
Contact Capacity Resistive Load (cosΦ=1)	7A 28VDC 10A 125VAC 7A 240VAC	10A 28VDC 10A 240VAC
Inductive Load (cosΦ=0.4 L/R=7msec)	3A 120VAC 3A 28VDC	5A 120VAC 5A 28VDC
Max. Allowable Voltage	250VAC/110VDC	250VAC/110VDC
Max. Allowable Power Force	800VAC/240W	1200VA/300W
Contact Material	AgCdO	AgCdO

8. PERFORMANCE (at initial value)

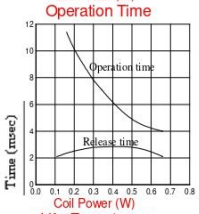
Item \ Type	SRD
Contact Resistance	100mΩ Max.
Operation Time	10msec Max.
Release Time	5msec Max.
Dielectric Strength Between coil & contact Between contacts	 1500VAC 50/60HZ (1 minute) 1000VAC 50/60HZ (1 minute)
Insulation Resistance	100 MΩ Min. (500VDC)
Max. ON/OFF Switching Mechanically Electrically	 300 operation/min 30 operation/min
Ambient Temperature	-25°C to +70°C
Operating Humidity	45 to 85% RH
Vibration Endurance Error Operation	 10 to 55Hz Double Amplitude 1.5mm 10 to 55Hz Double Amplitude 1.5mm
Shock Endurance Error Operation	 100G Min. 10G Min.
Life Expectancy Mechanically Electrically	 10^7 operations. Min. (no load) 10^5 operations. Min. (at rated coil voltage)
Weight	abt. 10grs.

9.REFERENCE DATA

Coil Temperature Rise

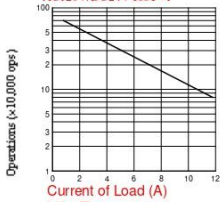


Operation Time

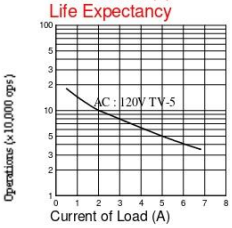


Life Expectancy

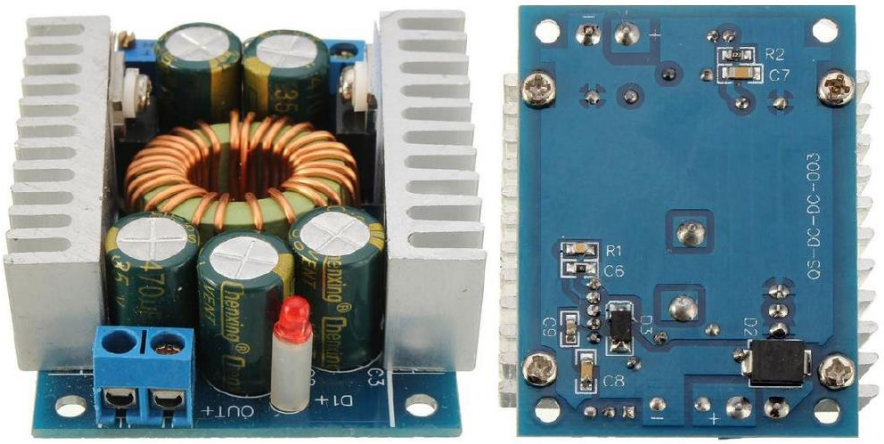
AC120V/DC24V cosΦ=1



Life Expectancy



QS-DC-DC-003
DC-DC step down DC-DC converter 1.25-30V, 12A, 100W



Specifications:

Module Type:	12A Step down
Input Voltage:	Input Voltage: 4.5-30V the input voltage must not exceed 30V !
Output voltage:	1.25-30V adjustable, CV.
Output current:	0-12A 100W with larger heatsink up to 200W
Operating temperature:	-40 to + 85 degrees
Operating frequency:	300 kHz
Conversion efficiency:	up to 95%
Module dimensions:	60 mm x 51 mm x 22 mm (L, W, H)

Features:

Adjustable Voltage CV.
Short circuit protection: current limitation 14A, Fuse.
Overheating protection: Yes (for temperature, automatic shutdown after disconnecting the output)
Input reverse polarity protection: No, (if required in the input line to the diode)
Module dimensions: 60 mm x 51 mm x 22 mm (Lxwxh)

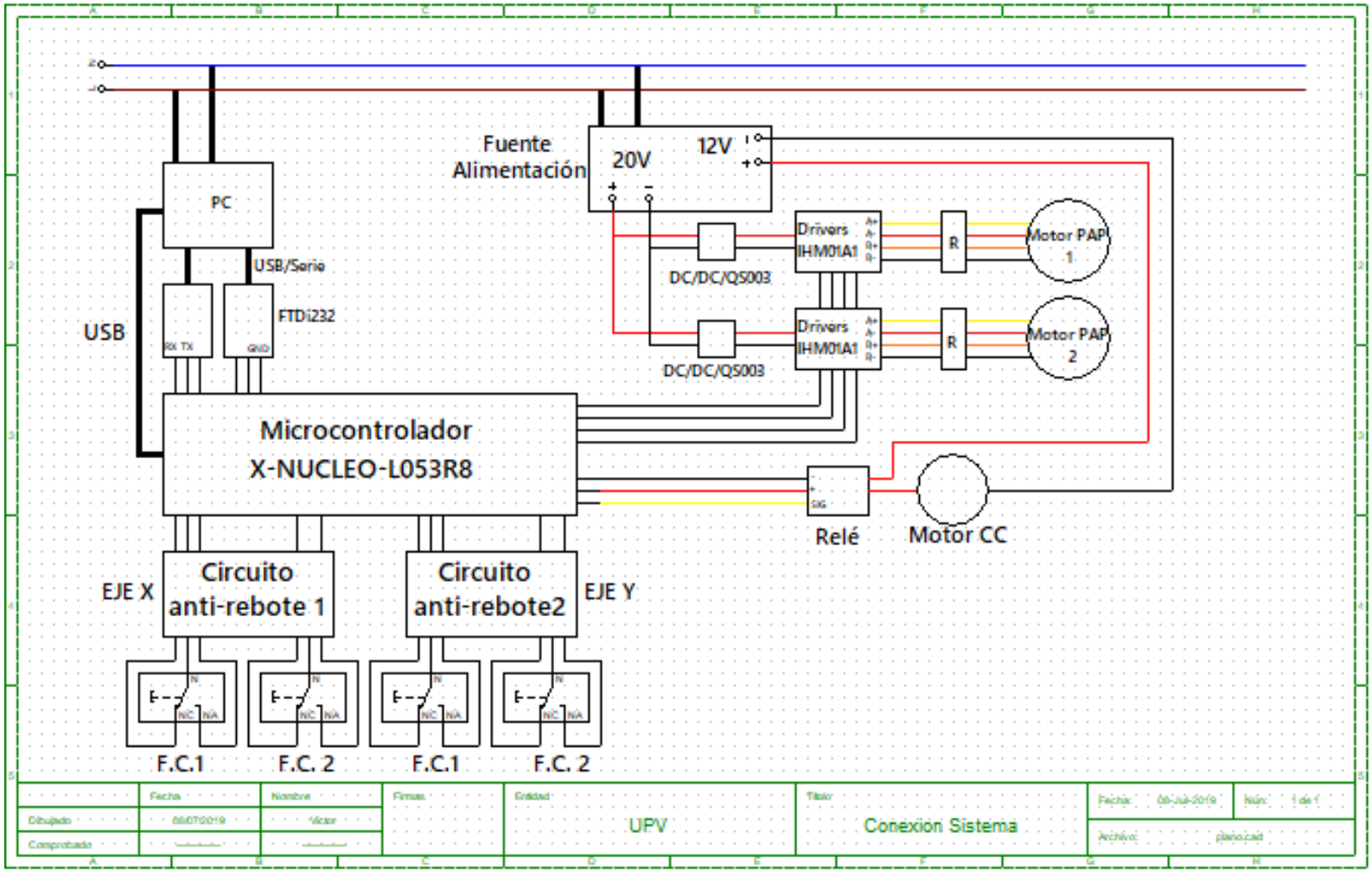
Applications:

1. DIY an output adjustable vehicle power supply, connect the 12V input, and output could be e.g.: 1.25-9V(adjustable freely). However, the output voltage is less than the input voltage.
2. Battery charger. Ni-MH, Lith.
3. Vehicle Power Supply Module Converter.
4. For your electronic equipment
5. Pre-supply system battery
6. Power Solar Charging
7. LED Driver

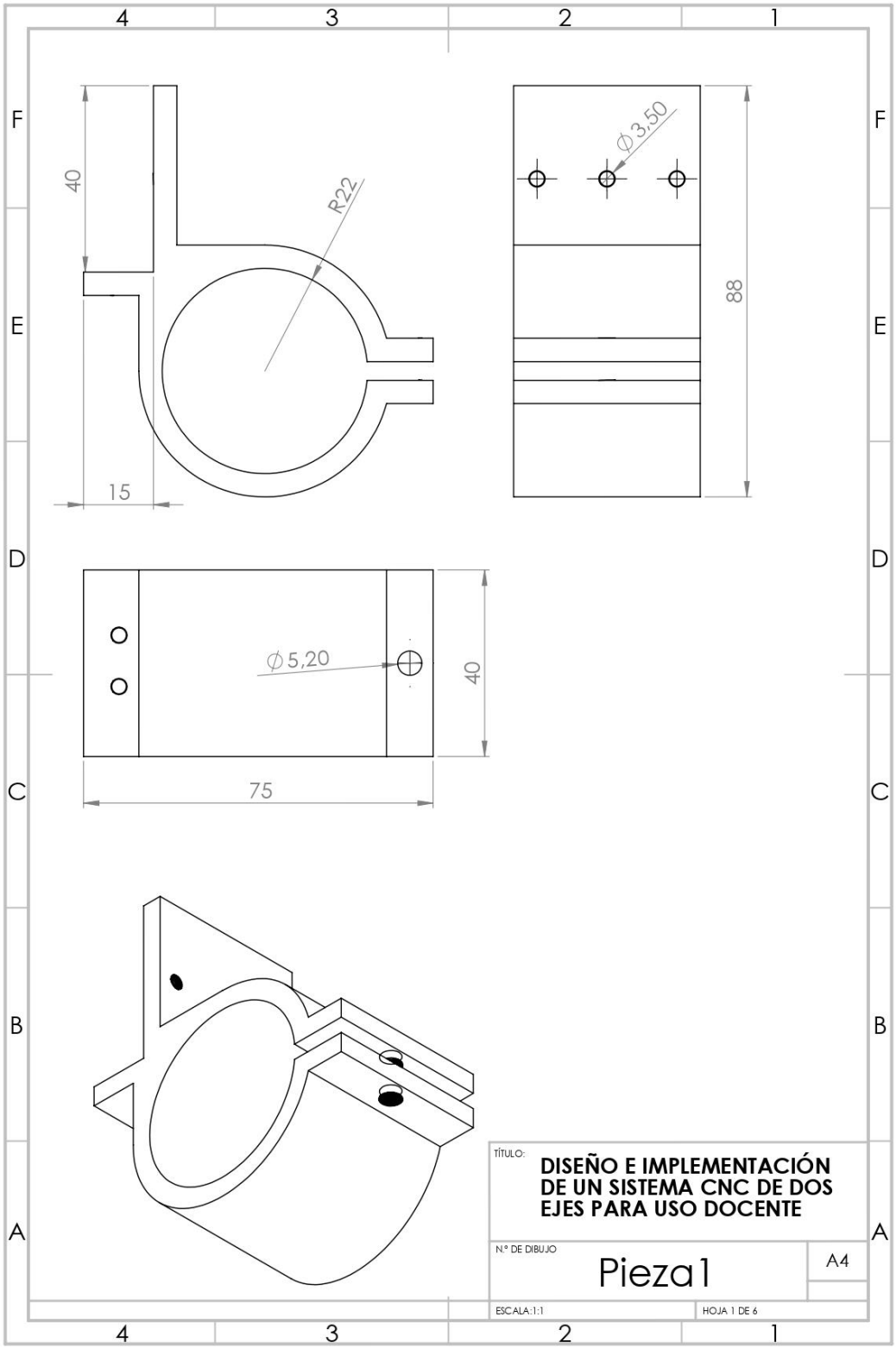
Module adjustment:

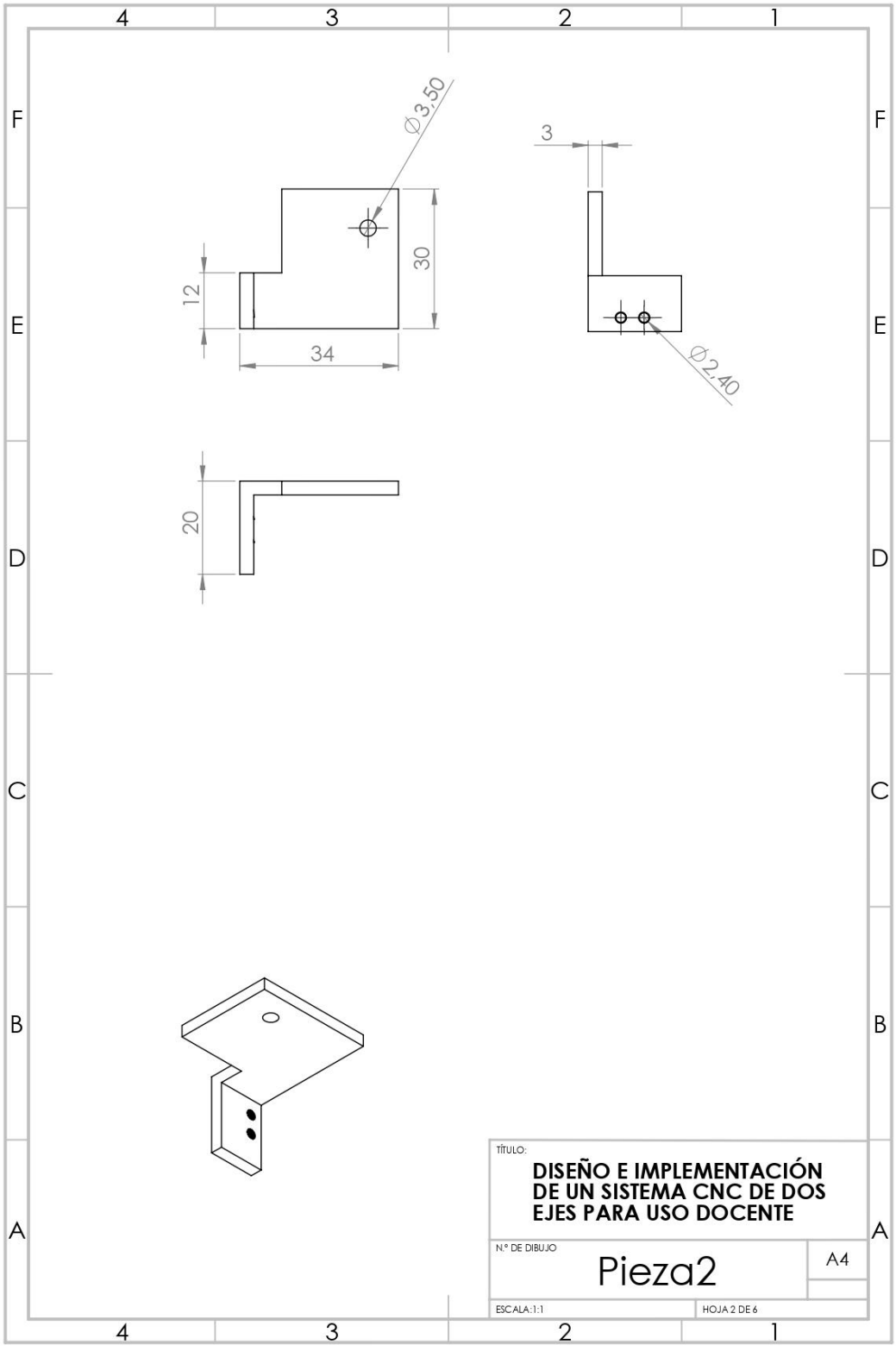
1. When the module output voltage can not be adjusted (the output voltage is equal to the input voltage) Please adjust the potentiometer counterclockwise (20 turns) or more till it works.
2. Turn trimmers CLOCKWISE to INCREASE, ANTICLOCKWISE to DECREASE.
3. When adjust the blue potentiometer ,please use a multimeter to monitor the voltage out.

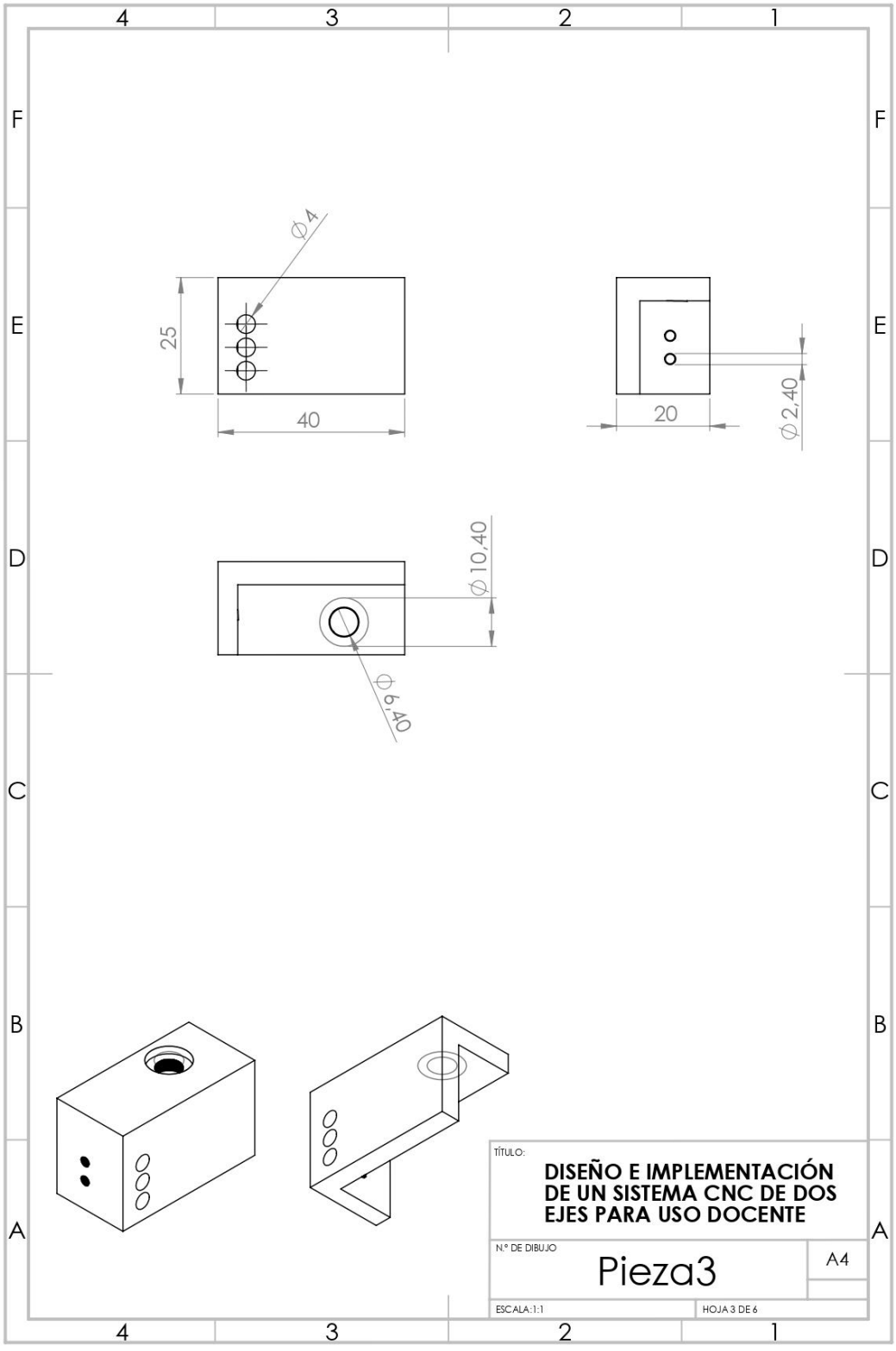
13.3. ANEXO C: Plano conexión sistema.

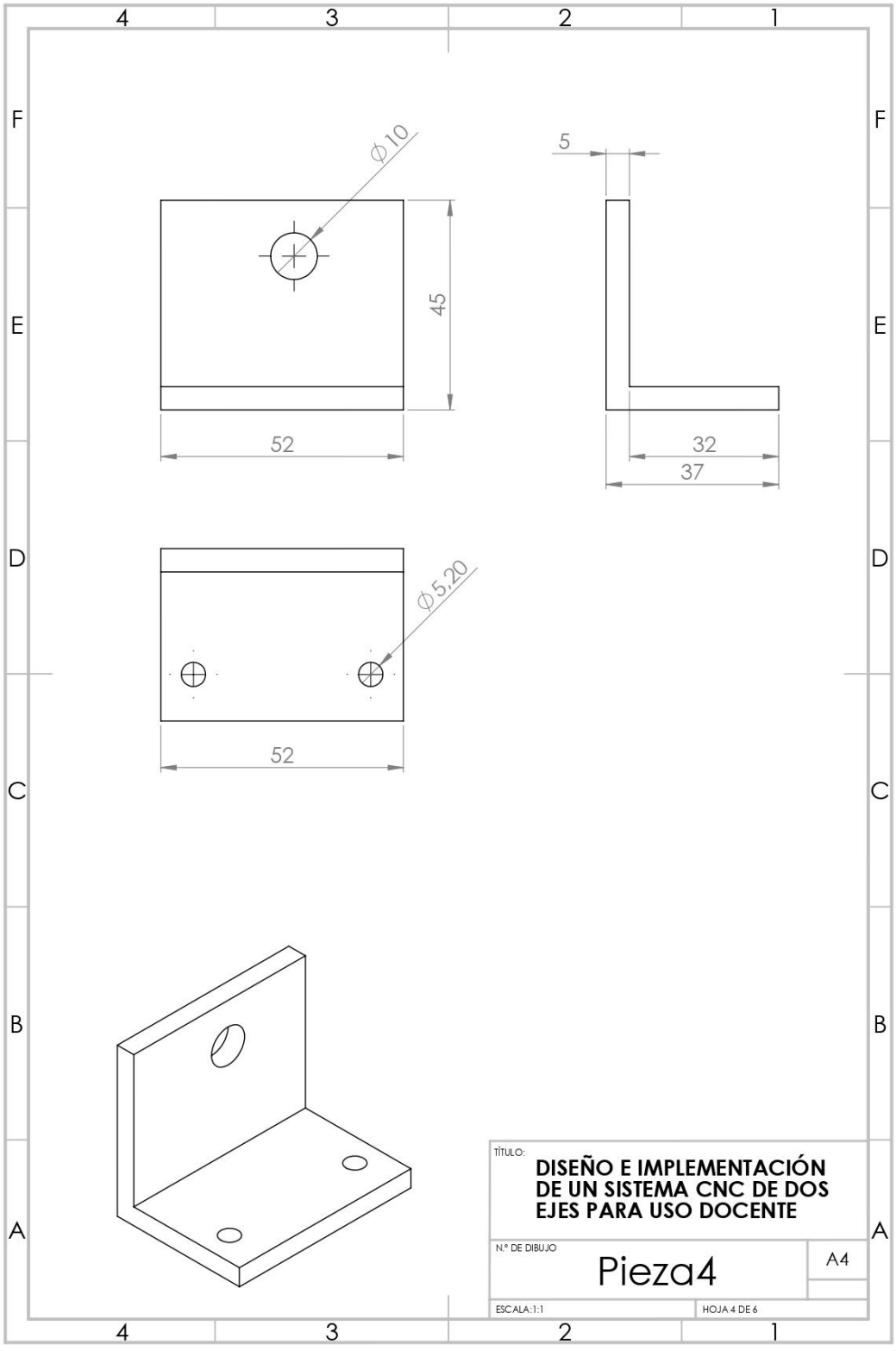


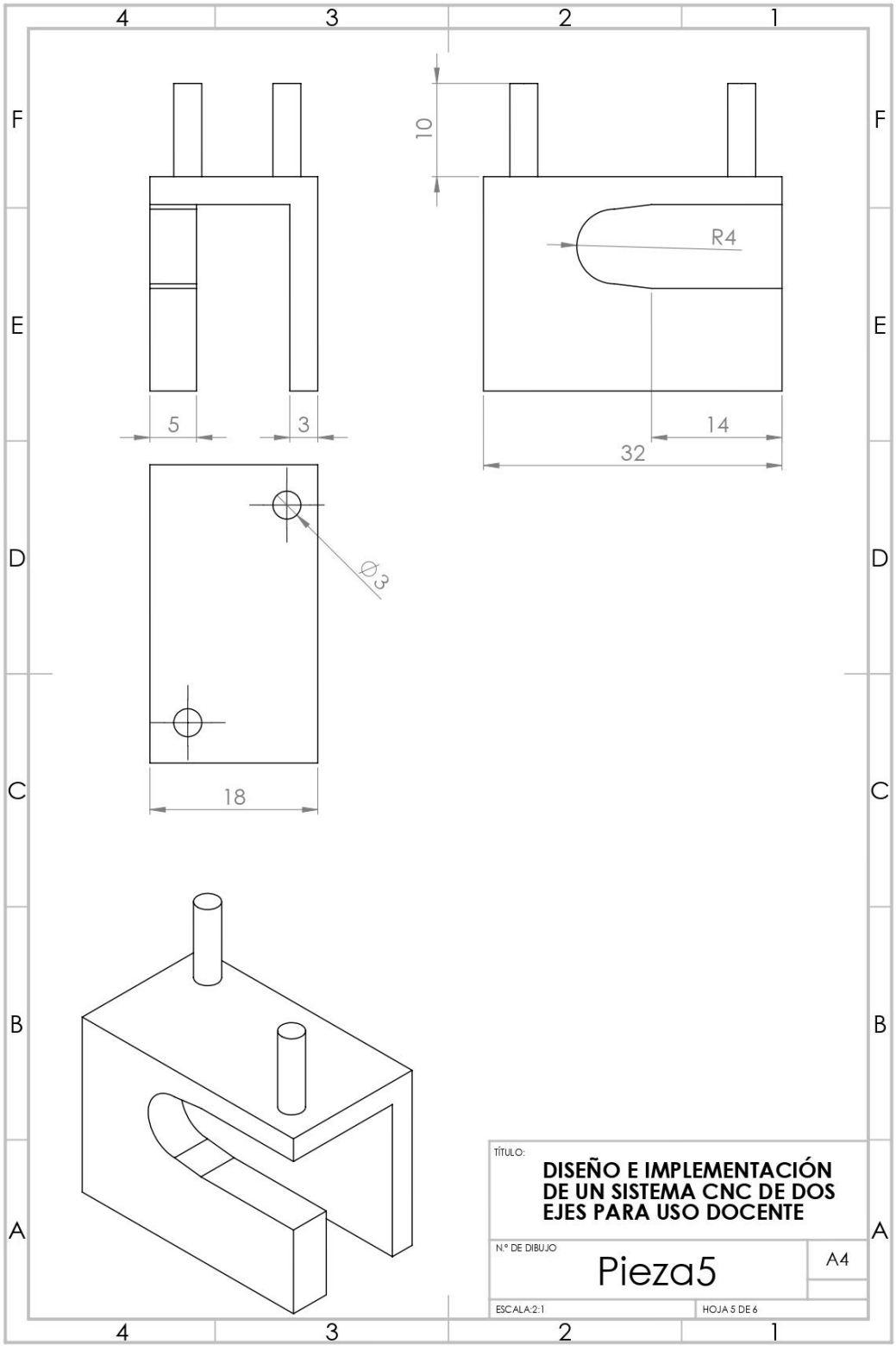
13.4. ANEXO D: Planos piezas diseñadas.

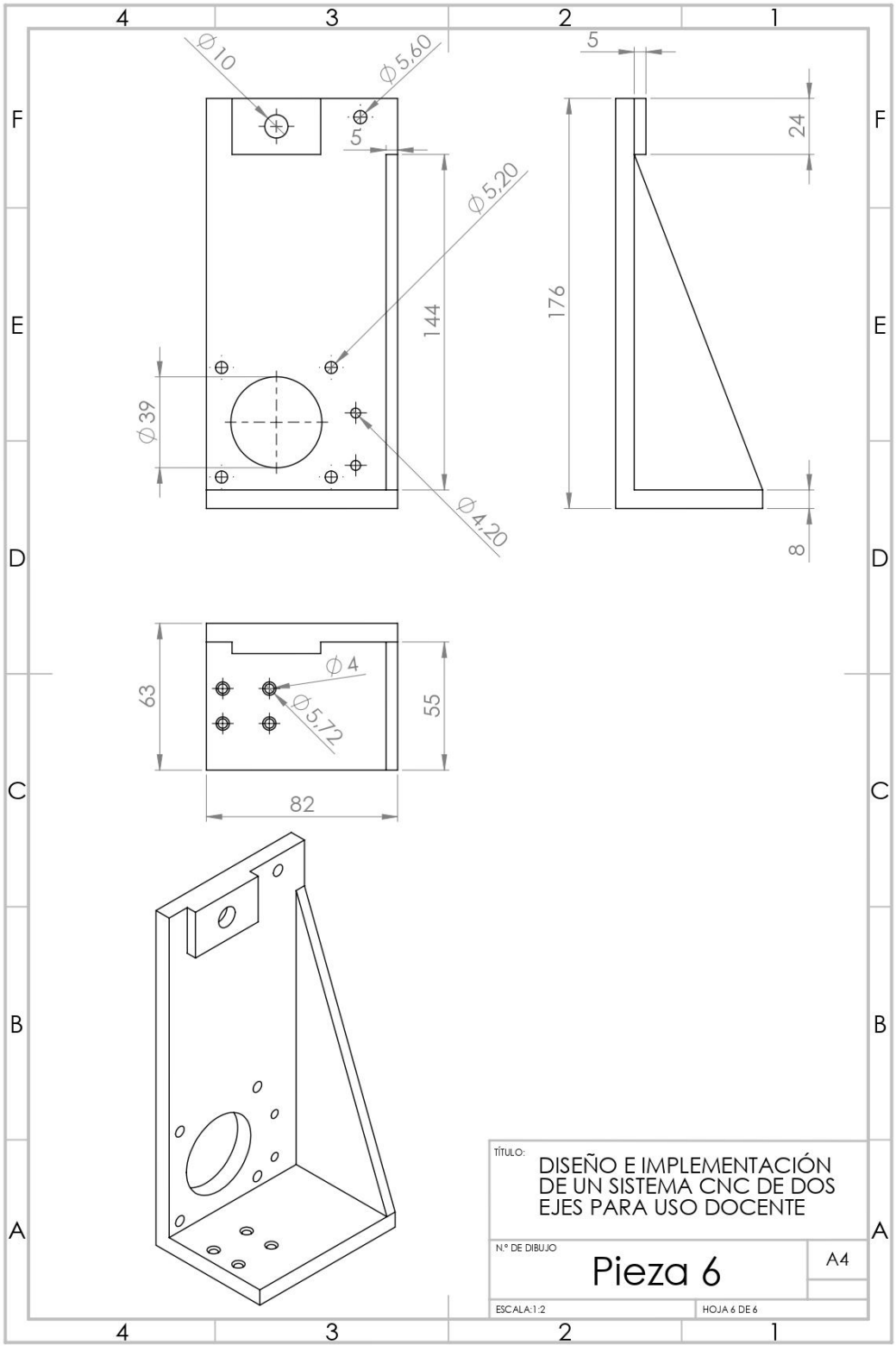



4
3
2
1
F
E
D
C
B
A
Ø3,50
3
30
12
34
Ø2,40
20
TÍTULO:
DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA CNC DE DOS EJES PARA USO DOCENTE
N.º DE DIBUJO
Pieza2
A4
ESCALA:1:1
HOJA 2 DE 6


4
3
2
1
F
E
D
C
B
A
Ø4
25
40
20
Ø2,40
Ø10,40
Ø6,40
TÍTULO:
DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA CNC DE DOS EJES PARA USO DOCENTE
N.º DE DIBUJO
Pieza3
A4
ESCALA:1:1
HOJA 3 DE 6


4
3
2
1
F
E
D
C
B
A
Ø10
45
52
5
32
37
Ø5.20
52
TÍTULO:
DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA CNC DE DOS EJES PARA USO DOCENTE
N.º DE DIBUJO
Pieza4
A4
ESCALA:1:1
HOJA 4 DE 6


4
3
2
1
F
E
D
C
B
A
10
R4
5
3
14
32
Ø3
18
TÍTULO:
DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA CNC DE DOS EJES PARA USO DOCENTE
N.º DE DIBUJO
Pieza5
A4
ESCALA:2:1
HOJA 5 DE 6


4
3
2
1
F
E
D
C
B
A
Ø10
Ø5,60
5
Ø5,20
144
Ø39
Ø4,20
5
24
176
8
Ø4
Ø5,72
63
55
82
TÍTULO:
DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA CNC DE DOS EJES PARA USO DOCENTE
N.º DE DIBUJO
Pieza 6
A4
ESCALA:1:2
HOJA 6 DE 6

13.5. ANEXO E: Imágenes del sistema.

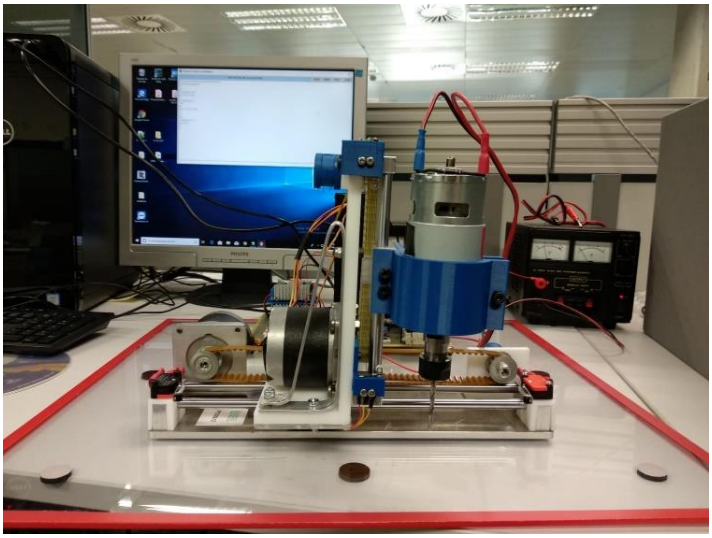


Imagen 13.5.1 Vista frontal del sistema CNC.

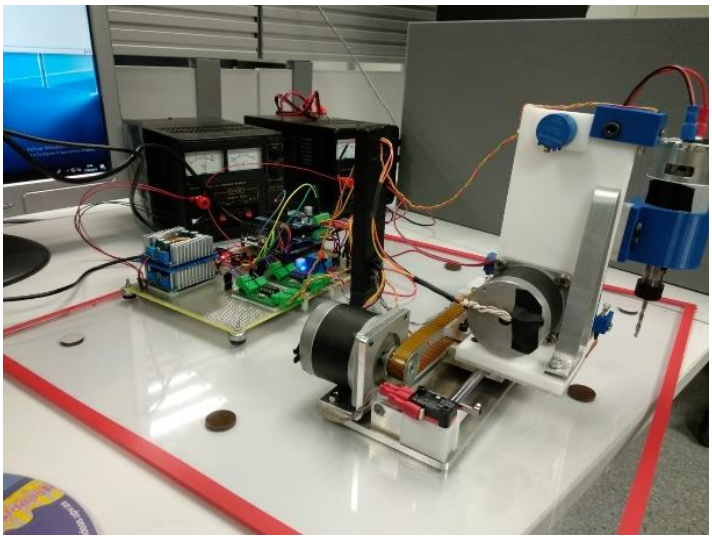


Imagen 13.5.2 Perspectiva lateral del sistema CNC.

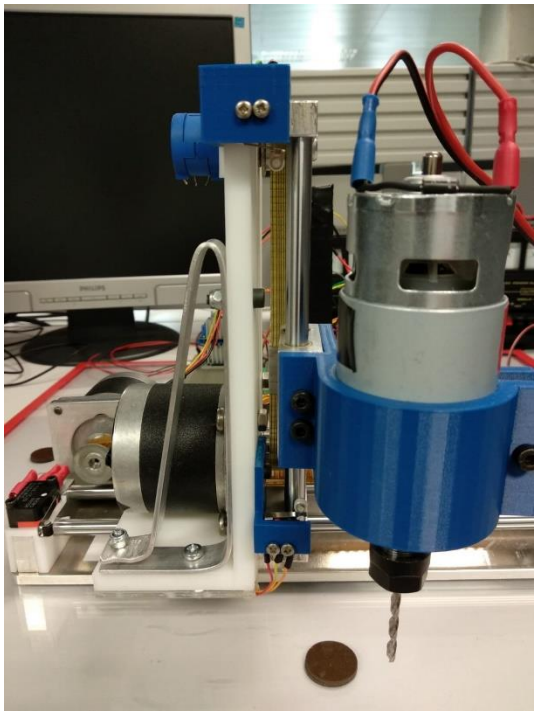


Imagen 13.5.3 Vista lateral del eje Y del sistema CNC.

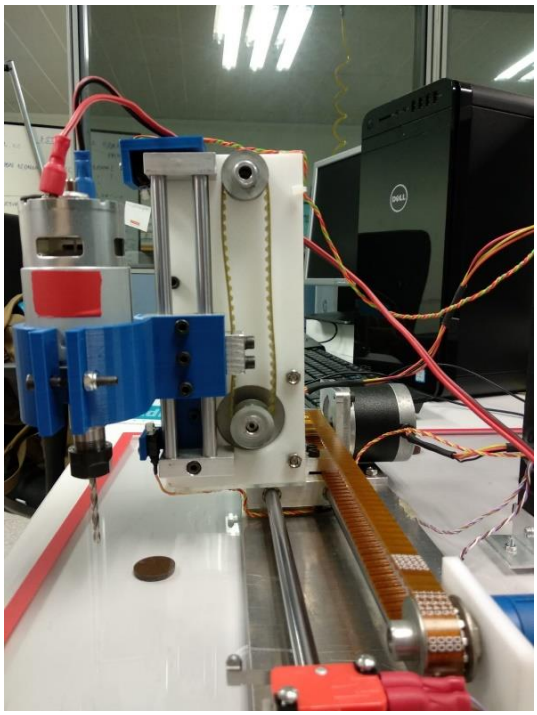


Imagen 13.5.4 Vista frontal del eje Y del sistema CNC.

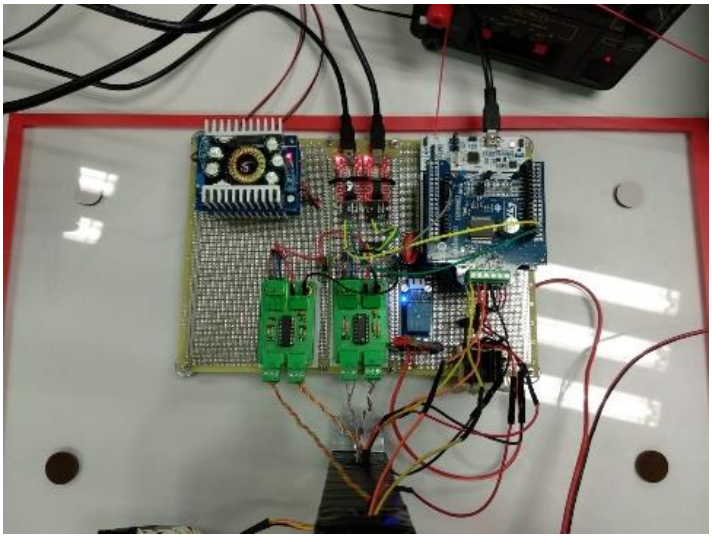


Imagen 13.5.5 Placa PCB con los componentes electrónicos del sistema CNC.