UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño y programacion de un sistema CNC de dos eje para usos docentes.

Trabajo Fin de Máster

Máster Universitario en Ingeniería Mecatrónica

AUTOR/A: Rams Roda, Aaron

Tutor/a: Serrano Martín, Juan José

CURSO ACADÉMICO: 2021/2022

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DEL DISEÑO

MÁSTER UNIVERSITARIO EN INGENIERÍA MECATRÓNICA

TRABAJO FIN DE MÁSTER

"Diseño y programación de un sistema CNC de dos ejes para uso docente"

Autor:

Aarón Rams Roda

Tutor:

D. Juan José Serrano Martín

Junio 2022

A mi tutor Juan Jose Serrano Martín por la ayuda brindada.

RESUMEN

El objetivo de este TFM es realizar el diseño y la programación de una máquina de control numérico (CNC) de dos ejes, con funcionalidad de taladradora. La finalidad del proyecto es la del desarrollo de una máquina CNC para uso docente. Los alumnos de mecatrónica saben cómo funciona una máquina de CNC, pero no han visto como funciona el programa interno de estas máquinas. El objetivo es hacer una versión reducida que conlleva todas las funcionalidades de una CNC real, desde la llegada de las líneas de los GCODES, la detección de los tipos de órdenes G y M, la decodificación y el análisis de las líneas para generar los movimientos de los motores de los ejes y de la herramienta. Este código maneja los diferentes tipos de dispositivos y elementos (mecánicos, electrónicos y eléctricos) que se estudian en muchas asignaturas del máster. Para el desarrollo del mismo hay que preparar programas de prueba de las diferentes partes que pueden servir de aprendizaje durante la asignatura de SS.EE. El código puede ser similar al código GRBL usado por muchas máquinas CNC pequeñas e impresoras 3D, pero de un tamaño más pequeño para poder manejarlo durante un curso de un semestre.

Palabras clave: sistemas embebidos, máquinas CNC, programación en C. taladro automático

ABSTRACT

The goal of this TFM is to carry out the design and programming of a two-axis Computer Numeric Control (CNC), with drilling functionality. The purpose of the project is to develop a CNC machine for educational use. Mechatronics students know how a CNC machine works, but they have not seen how the internal program of these machines works. The objective is to make a reduced version that includes all the functionalities of a real CNC, from the arrival of the GCODES lines, the detection of the types of orders G and M, the decoding and parsing of the lines to generate the movements of the motors of the axes and of the tool. This code handles the different types of devices and elements (mechanical, electronic and electrical) that are studied in many subjects of the master's degree. For the development of the same it is necessary to prepare test programs of the different parts that can serve as learning during the subject of SS.EE. The code can be similar to the GRBL code used by many small CNC machines and 3D printers, but smaller in size to handle over a one-semester course.

Keywords: embedded systems, CNC machines, C programming, automatic drill

ÍNDICE

JUSTIFICACIÓN DEL PROYECTO

El desarrollo de este proyecto 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 de D. Juan José Serrano Martín, para el diseño de una máquina CNC para uso docente. El proyecto engloba todos los conocimientos adquiridos 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 el funcionamiento de las máquinas CNC y sobre los componentes necesarios para su diseño y programación.

Para ello, sobre una placa de circuito se monta una etapa de potencia y otra de control, que junto con la programación, permitirá el funcionamiento de la máquina taladradora.

MEMORIA

Diseño y programación de un sistema CNC de dos ejes para uso docente.

1. INTRODUCCIÓN

1.1. MOTIVACIÓN

La motivación para realizar el TFM reside en englobar todos los conocimientos adquiridos a lo largo del máster y llevarlos a la práctica, mediante la realización de esta máquina CNC.

Además, el uso de GCODES para la programación de sistemas CNC, hace que sea un proyecto muy interesante.

La parte docente es fundamental en este proyecto. Su uso va destinado al estudio de las máquinas CNC para futuros alumnos. Por tanto, el diseño de la máquina debe ser simple para que se pueda aprender de forma sencilla.

También, cabe destacar, la importancia que tiene desarrollar un proyecto como este para el futuro de tu vida laboral. Se trata de un proyecto, en el que se diseña y se programa una máquina que se encuentra en todo tipo de industrias. Por tanto, este proyecto contribuye a obtener experiencia de cara al futuro.

1.2. OBJETIVO

El objetivo de este proyecto es el desarrollo de una máquina de control numérico (CNC), de dos ejes, para uso docente. Se trata de la implementación de una taladradora programable que se desplaza a lo largo de un eje X y Z. De este modo, la máquina realizará automáticamente un patrón de ciclos de perforación predeterminado.

Este proyecto es desarrollado, completamente, para uso docente. Se trata de fomentar la motivación y el interés de los alumnos. Cuanto más motivado esté un alumno mayor implicación tendrá en su estudio y mayor dedicación pondrá en aquello que esté realizando. Como consecuencia, tendrá mayor facilidad para alcanzar sus metas académicas y permitirá progresar en sus habilidades. A causa de lo que se ha comentado anteriormente, se puede observar la gran utilidad que tiene un proyecto como este, para el uso docente de los alumnos.

Por tanto, la máquina se diseña y se programa de forma simple para facilitar su comprensión, utilizando componentes estudiados a lo largo del máster.

1.3. ANTECEDENTES

Las máquinas CNC tienen sus orígenes en los años 40 y 50, en Estados Unidos. Fue el Ingeniero John T. Parsons quién usó máquinas existentes en aquella época con unas modificaciones para que se le pudieran pasar los números mediante tarjetas perforadas. Este fue el inicio del control numérico.

Sin embargo, ante la necesidad de uso de herramientas más complejas y de obtener procesos automatizados, cada vez más rápidos y precisos, las máquinas CNC han ido evolucionando. Las primeras máquinas estaban controladas por un ordenador central con electrónica de válvulas, relés y cableados. Actualmente, estas máquinas están controladas por un microprocesador, cuyo reducido tamaño y baja potencia, han hecho posible el diseño de máquinas como la impresora 3D.

A medida que la tecnología evoluciona, en un futuro, puede haber elementos aún más increíbles para agregar a su historia.

2. MARCO TEÓRICO

2.1. Evolución a lo largo de la historia

El control numérico marcó el inicio de la Segunda Revolución Industrial. El Control Numérico por Computadora, lo que se conoce hoy por CNC, nació debido a los requisitos de precisión y complejidad de las piezas de las que requería la industria aeroespacial.

El primer desarrollo en el área del Control Numérico por Computadora (CNC) lo realizó John T. Parsons en la década de 1940. El concepto de Control Numérico implicaba el uso de datos en un sistema de referencia para definir las superficies de contorno de las hélices de un helicóptero.

Los primeros equipos de Control Numérico con electrónica de válvulas, relés y cableado, tenían un volumen mayor que las propias máquinas-herramienta. La programación usada en aquel momento era de forma manual en lenguaje máquina muy complejo y muy lenta de programar. Estos equipos consumían una gran cantidad de energía y generaban demasiado 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. Sin embargo, seguían teniendo un gran tamaño (ver Ilustración 1).

Ilustración 1. Una de las primeras máquinas CNC desarrolladas.

De la misma forma que sucede con otros equipos tecnológicos, el sistema CNC se fue perfeccionando y mejorando con el paso del tiempo hasta llegar a los equipos de la actualidad.

2.2. Estado actual

Actualmente el sistema de Control Numérico por Computadora es altamente eficiente y agilizan la producción en las industrias.

Cada maquinaria cuenta con una gran variedad de herramientas, ofreciendo opciones de operación en cada pieza a tratar. Y, generalmente, cuentan con cambio automático capaz de optimizar el tiempo de manipulación de la máquina por parte de los trabajadores. Esto permite elaborar mejores piezas de forma más rápida.

La automatización de las máquinas CNC también ofrece mayor seguridad, debido a que el operador evita su intervención y como resultado, se reduce los posibles accidentes de trabajo.

El equipo industrial es fundamental para la cadena de producción de cualquier empresa, ya que las necesidades de la industria se basan en la rapidez, la eficiencia y la calidad. Por ello, el uso de estas máquinas-herramienta resulta indispensable (ver Ilustración 2).

Ilustración 2. Maquina CNC industrial de 3 ejes.

Además, las máquinas CNC actuales tienen una pantalla con la que se puede controlar todos los movimientos de la máquina (ver Ilustración 3).

Ilustración 3. Pantalla de una máquina CNC.

2.3. Tipos de máquinas CNC

Las máquinas CNC se pueden clasificar según las funciones que realiza y según el tipo de control en el que estén basadas.

2.3.1. Control

Las máquinas CNC tienen 3 tipos de control diferente:

- Punto a punto: se mecaniza en el material, exclusivamente, los puntos iniciales y finales, pero no la trayectoria entre un punto y otro. No controlan ni el trazado ni la velocidad.

- Paraxial: se puede gobernar de forma precisa tanto la posición como la trayectoria. Se debe tener en cuenta que la trayectoria es paralela a los ejes de la máquina

- Interpolar o continuo: ofrece la posibilidad de realizar mecanizados a lo largo de trayectorias de cualquier tipo. Son las máquinas más polivalentes en cuanto a mecanizado.

2.3.2. Finalidad

Algunas de las máquinas CNC más usadas en la industria por sus diferentes tipos de finalidades son:

- Fresadora: máquina-herramienta que se utiliza para crear piezas con formas personalizadas mediante un proceso de mecanizado, usando una pieza

giratoria llamada “fresa” (ver Ilustración 4).

Ilustración 4. Ejemplo de una fresadora actual.

- Cortadora FOAM: máquina-herramienta capaz de realizar cortes a través de un alambre caliente que se mueve entre dos ejes paralelos X y Y (ver Ilustración 5). El material utilizado es el FOAM (espuma visco elástica).

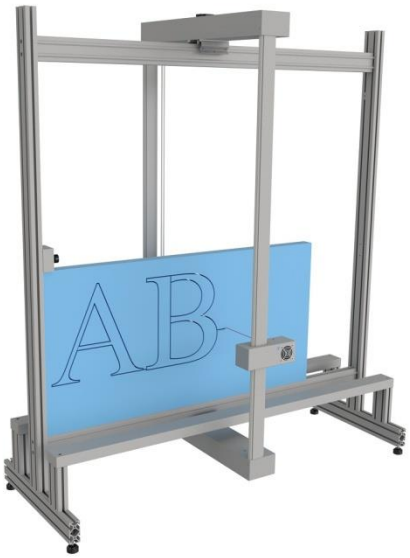


Ilustración 5. Ejemplo de una cortadora de FOAM actual.

- Rectificadora: máquina-herramienta que realiza el proceso de mecanizado por abrasión. Realiza el proceso de arranque de material mediante una rueda robusta, lo que permite obtener, un acabado de la pieza con dimensiones precisas y con menos rugosidades (ver Ilustración 6).

Ilustración 6. Ejemplo de una rectificadora actual.

- Impresora 3D: máquina capaz de imprimir figuras con volumen a partir de un diseño realizado por ordenador (ver Ilustración 7).

Ilustración 7. Ejemplo de una impresora 3D.

3. IMPLEMENTACIÓN DE UNA MÁQUINA CNC

Una máquina CNC es una máquina de Control Numérico por Computadora que sirve para controlar de forma automática el proceso de un trabajo. El funcionamiento de este tipo de máquina se basa en GCODES (códigos G y M), es decir, una serie de instrucciones secuenciales que hace posible que se active la máquina.

Las máquinas CNC están formadas por una unidad de control y una serie de motores que permiten activar la máquina-herramienta.

Para comprender el funcionamiento de una máquina CNC y los elementos que la componen, se va a explicar el procedimiento llevado a cabo para realizar una taladradora CNC.

3.1. Funcionamiento

En el caso del presente proyecto se va a realizar la implementación de una máquina CNC con funcionalidad de taladradora, la cual tras recibir varias líneas completas de GCODES, lleva a cabo un ciclo de taladrado.

El ciclo de taladrado consiste en la programación de tres distancias donde la máquina realizará los agujeros. Una vez terminado el último, la máquina vuelve a la posición de "Homing" del principio.

Todas las instrucciones son enviadas desde un ordenador hasta el microcontrolador a través de la USART.

3.2. Mecánica y estructura

En cuanto al diseño del sistema, cabe destacar, que consiste en dos ejes perpendiculares entre sí que permiten el desplazamiento longitudinal y transversal de un taladro.

Por tanto, la estructura principal se encuentra compuesta por barras metálicas, de acero inoxidable, que proporcionan soporte y robustez.

A continuación, se detallan los materiales usados para la estructura del sistema.

3.2.1. Base

El sistema se encuentra anclado en una base de metacrilato transparente que aporta soporte y robustez al sistema (ver Ilustración 7). Además, como el peso del metacrilato es reducido, se puede transportar el sistema fácilmente. La base tiene una serie de almohadillas en la parte inferior para adherirse a la superficie sobre la que se coloque.

Ilustración 8. Base de metacrilato.

3.2.2. Estructura

La estructura principal está compuesta por finas planchas, de acero inoxidable, que proporcionan firmeza al sistema (ver Ilustración 9).

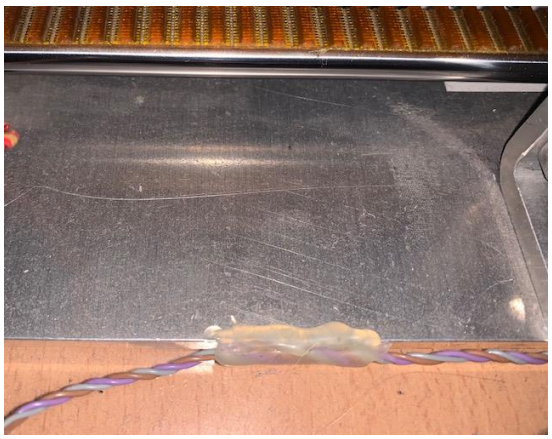


Ilustración 9. Plancha de acero usada en la estructura.

3.2.3. Varilla lisa

Para llevar a cabo el desplazamiento de las distintas partes de la máquina, se dispone de varillas lisas calibradas, de acero inoxidable, de 10 mm de diámetro (ver Ilustración 10).

Ilustración 10. Varilla lisa de acero inoxidable.

Cada eje cuenta con dos varillas, que van sujetas a los extremos de la estructura, para evitar vibraciones y flexiones.

3.2.4. Correa y rueda dentada

Para la transmisión del movimiento a lo largo del eje X y Z se emplea el conjunto de correa y rueda dentada (ver Ilustración 11). De esta forma el motor transmite el movimiento a una rueda dentada, y a través de la correa llega el movimiento a la otra rueda dentada, cerrando así el sistema de movimiento.

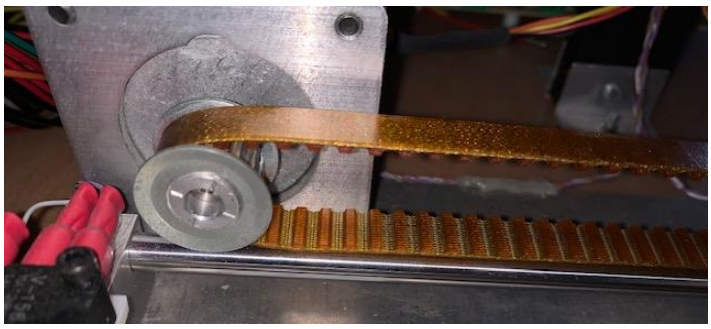


Ilustración 11. Rueda dentada y correa.

3.2.5. Rodamiento lineal y deslizadera

El sistema de desplazamiento consiste en la combinación de las guías con las correas dentadas. Para ello, a través de un rodamiento lineal se desplazan unas deslizaderas de acero inoxidable en forma rectangular (ver Ilustración 12). Tienen dos agujeros transversales que permiten introducirlas dentro de las guías, y una serie de agujeros para el anclaje de las mismas a la correa.

Ilustración 12. Deslizadera de acero.

3.3. Componentes

En este apartado, se procede a explicar y detallar los componentes utilizados para el desarrollo de la taladradora CNC.

3.3.1. Fuente de alimentación BM PS01

En cuanto a la fuente de alimentación del sistema, se utiliza el modelo BM PS01 (ver Ilustración 13).

Ilustración 13. Fuente de alimentación BM PS01.

Las características de esta fuente de alimentación son:

- Corriente de salida: 0…16 A
- Tensión de salida: 12V CC
- Fuente de alimentación: 230V CA 50/60 Hz

3.3.2. Transformador DC-DC-QS 003

Para la alimentación de los componentes electrónicos se dispone de una fuente de alimentación fija con una tensión de salida de 12V. Sin embargo, como se comentará más adelante en el trabajo, tanto el driver como los motores utilizados necesitan una tensión inferior. Por tanto, es necesario el uso de un transformador, concretamente se usa el transformador DC-DC-QS 003 (ver Ilustración 14).

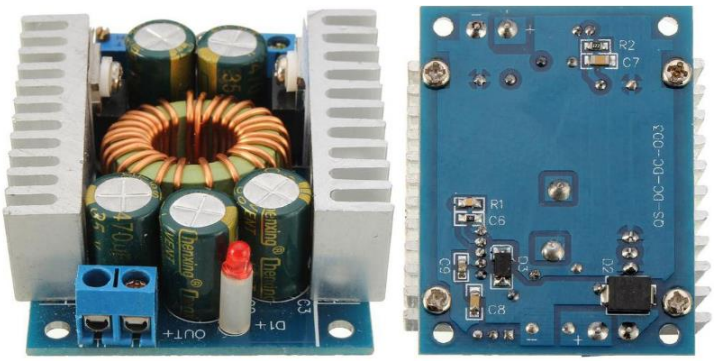


Ilustración 14. Transformador DC-DC-QS 003.

Las características del transformador de corriente continua son las siguientes:

- Tensión de entrada: 4.5…30V
- Tensión de salida: 1.5…30V
- Corriente de salida: 0…12 A
- Eficiencia de conversión: 95%

En la taladradora es necesario el uso de dos transformadores. Uno alimenta a los al driver utilizado para el control de los motores y el otro alimenta al propio taladro. Esto se debe a que es necesario el uso de diferentes tensiones.

Por tanto, la fuente de alimentación fija alimenta a ambos transformadores. El transformador utilizado para el driver tiene una tensión de salida de 8.2V, mientras que el otro tiene una tensión de salida de 10V.

3.3.3. Microcontrolador

Para el control del sistema se va a utilizar un microcontrolador de ST Microelectronics, exactamente el STM32F446 (ver Ilustración 15).

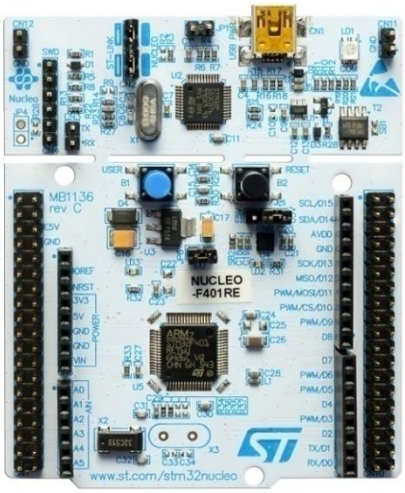


Ilustración 15. Microcontrolador STM32F446.

Se trata de un circuito integrado programable capaz de ejecutar un programa previamente diseñado, verificado y compilado que se encuentra almacenado en su memoria. Están compuestos por varios bloques funcionales que cumplen tareas específicas.

- CPU: es la unidad central de proceso que interpreta las instrucciones de un programa informático mediante la realización de operaciones básicas aritméticas y lógicas.

- Memoria: en los microcontroladores la memoria de instrucciones y datos está integrada en el propio chip. Una parte debe ser no volátil, tipo ROM, y se destina a contener el programa de instrucciones que gobierna la aplicación. Otra parte de memoria es tipo RAM, volátil y se destina a guardar las variables y datos.

Además, el microcontrolador cuenta con diferentes posibilidades de comunicación como SPI y Puerto serie. Emplea un consumo reducido y son capaces de llevar a cabo una gran cantidad de tareas de manera rápida y eficaz.

3.3.4. Driver

Para el control de los motores es necesario el uso de un driver. Esto se debe a que la salida del microcontrolador alcanza un valor aproximado de 5V y una corriente muy reducida. Por tanto, ni la tensión ni la corriente son suficientes para el funcionamiento del motor.

Con el uso del driver, se controlan los motores desde el microcontrolador, pero se pueden alimentar desde una fuente de alimentación externa. De esta forma, tanto la tensión como la corriente son suficientes para el correcto funcionamiento de los motores, incluso de forma simultánea.

El driver utilizado es la CNC Shield, montada con el controlador DRV8825 (ver Ilustración 16).

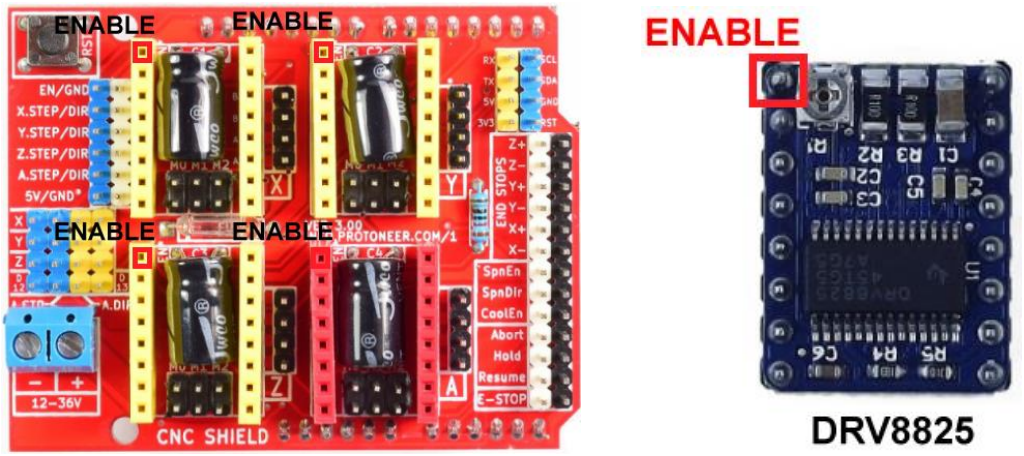


Ilustración 16. CNC Shield y controlador DRV8825.

En cuanto a la CNC Shield, es necesario alimentarla por dos sitios diferentes. La primera alimentación se realiza mediante el transformador a 8.2V. La segunda alimentación se lleva a cabo mediante un pin de salida de 5V del microcontrolador.

3.3.5. Motor paso a paso

En cuanto a los motores, cabe destacar que el más indicado para el desarrollo del proyecto son los motores paso a paso. La razón de su uso es debido a que no se precisa del uso de encoders para determinar la posición y por tanto, es posible trabajar en bucle abierto y conocer tanto la posición como la velocidad del mismo de manera sencilla.

Se procede a explicar los diferentes tipos de motor paso a paso que existen, y tras analizarlos, se escoge el más útil para este tipo de máquina CNC.

Un motor paso a paso es un dispositivo electromecánico que convierte una serie de pulsos eléctricos en desplazamientos angulares. Estos motores son ideales para la construcción de mecanismos que requieran mucha precisión en el movimiento. Su principal característica es la capacidad de moverse un paso por cada pulso aplicado, a diferencia del resto de motores de corriente continua que giran libremente. Además, estos motores pueden ser controlados, lo que permite variar la velocidad de giro cambiando el tiempo transcurrido entre pasos.

Los pasos del motor pueden variar desde 90º hasta pequeños movimientos de 1.8º. Además, estos motores poseen la habilidad de quedar enclavados en una posición si una o más de sus bobinas está energizada o bien totalmente libres de corriente.

Existen diferentes tipos de motores paso a paso:

- Motor paso a paso de reductancia variable: en este tipo de motor, el rotor consta de varios dientes hechos de hierro dulce. Cuando las bobinas del estator son alimentadas por una corriente continua, el diente del rotor es atraído por el campo magnético (ver Ilustración 17). El estator del motor tiene las bobinas conectadas a un terminal común. El conductor común se conecta habitualmente al borne positivo y las bobinas son alimentadas siguiendo una secuencia consecutiva.

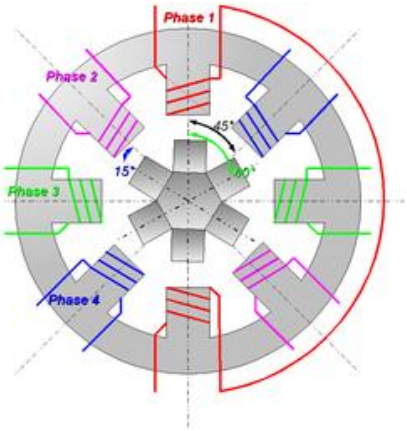


Ilustración 17. Motor paso a paso de reductancia variable.

- Motor paso a paso de imanes permanentes: este tipo de motores están constituidos por un rotor (sin dientes) sobre el que van aplicados distintos imanes permanentes, y por un cierto número de bobinas excitadoras bobinadas en su estator (ver Ilustración 18). Cuando las bobinas del estator generan un campo magnético, el rotor tiende a alinearse. Se basa en la fuerza de atracción de dos imanes, uno permanente y otro temporal.

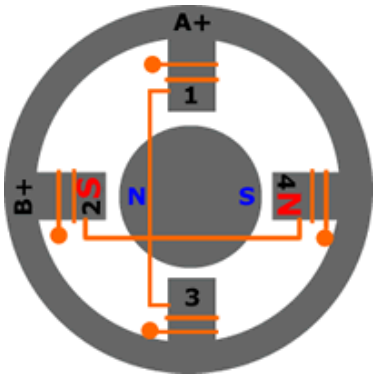


Ilustración 18. Motor paso a paso de imanes permanentes.

- Motor paso a paso híbrido: el funcionamiento de este motor se basa en la combinación del motor de reductancia variable y del motor de imanes permanentes. Consiste en un estator dentado y un rotor constituido por anillos de acero dulce dentado (ver Ilustración 19). Estos anillos están 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).

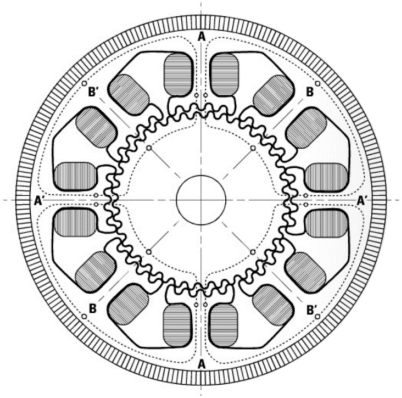


Ilustración 19. Motor paso a paso híbrido.

Por tanto, una vez explicado cada tipo de motor y según sus características, se escoge el motor paso a paso híbrido. El motor empleado, concretamente, es el RS PRO 440-442 (ver Ilustración 20). Presenta las siguientes características:

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

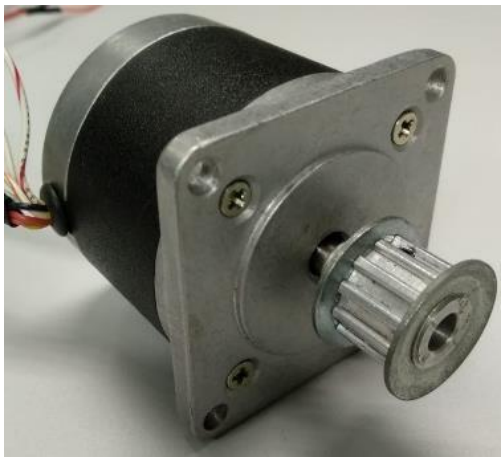


Ilustración 20. Motor RS PRO 440-442 empleado en la taladradora.

Se emplean dos motores, uno para mover el eje X y otro para el eje Z. 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 (ver Ilustración 21). En los unipolares la excitación de los devanados se realiza de uno en uno mientras que en los bipolares se realiza por pares con inversión de corriente.

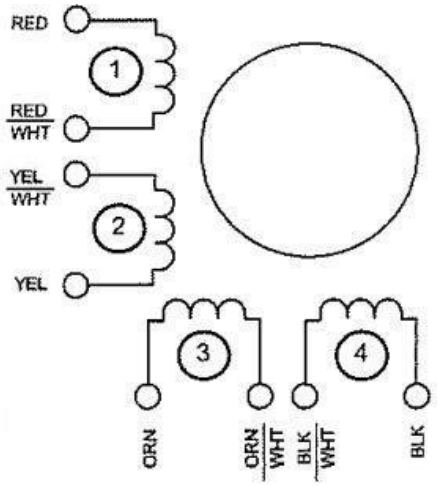


Ilustración 21. Distribución del conexionado del motor RS PRO 440-442.

La elección del conexionado de 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 (ver Ilustración 22).

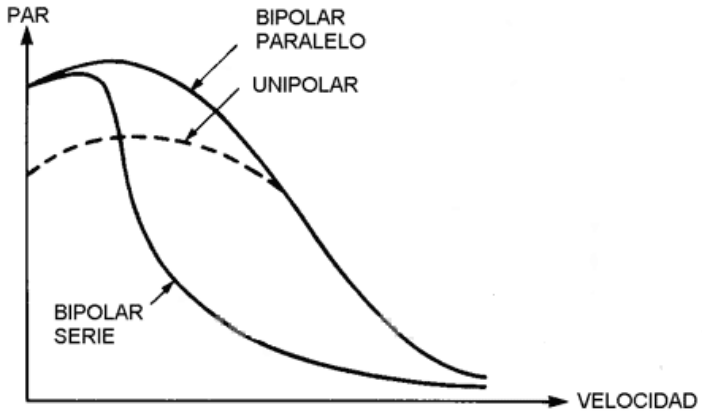


Ilustración 22. Gráfica par-velocidad de las diferentes configuraciones del motor paso a paso.

En la gráfica anterior se observa que se obtiene un mayor par con la configuración bipolar. Sin embargo, a medida que aumenta la velocidad, el par disminuye notablemente. En cambio, con la configuración unipolar, el par que se consigue es menor, pero no disminuye tanto cuando aumenta la velocidad.

Para la realización de este proyecto, se ha optado por una configuración bipolar, dado que se prioriza el tener un par elevado a una velocidad elevada.

Ahora, se decide si la conexión de los devanados se realiza en serie o en paralelo. El conexionado en paralelo tiene un par bajo y a medida que aumenta la velocidad se reduce de forma insignificante. En cambio, el conexionado en serie tiene un par elevado a velocidades bajas y disminuye significativamente al aumentar la velocidad (ver Ilustración 23).

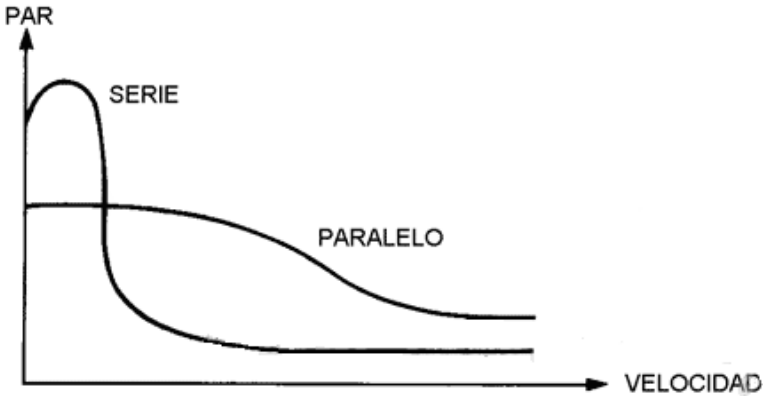


Ilustración 23. Gráfica par-velocidad de las conexiones en serie y en paralelo de un motor paso a paso.

Por tanto, como se prioriza un par elevado, se ha optado por una conexión bipolar en serie. Una vez se sabe la configuración del motor, se puede saber la distribución de los hilos (ver Ilustración 24). Queda de la siguiente forma:

- Amarillo: devanado A+
- Blanco/amarillo + blanco/rojo: unidos entre sí y sin conectar
- Rojo: devanado A-
- Naranja: devanado B+
- Blanco/naranja + blanco/negro: unidos entre sí y sin conectar
- Negro: devanado B-

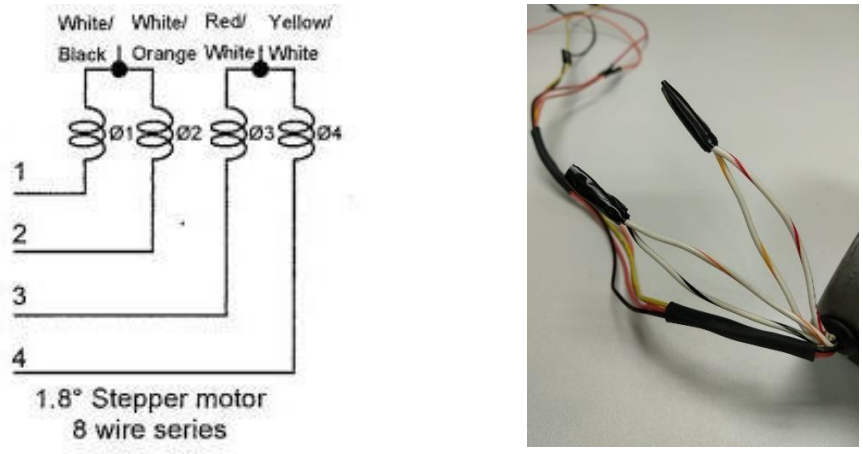


Ilustración 24. Conexionado del motor paso a paso en serie.

Como se ha comentado anteriormente, el voltaje nominal de los motores es de 5V mientras que el mínimo para el driver utilizado en el control de los motores es de 8V. Por tanto, para que el sistema funcione correctamente, se decide utilizar un voltaje de 8.2V e incorporar un par de resistencias en dos de los cuatro devanados del motor (ver Ilustración 25). De esta forma, se reduce la intensidad del motor ya que puede ser dañina para el mismo.

Ilustración 25. Resistencia para reducir la intensidad del motor.

3.3.6. Taladro

El sistema cuenta con otro motor para realizar los ciclos de taladrado. En este caso, el motor utilizado es de corriente continua de imanes permanentes. Concretamente, el motor es el Spindle 775 (ver Ilustración 26). Presenta las siguientes características:

- Tensión nominal: 10…36V
- Intensidad de carga a 12V: (2.2 A de arranque) 1.7 A
- Velocidad de funcionamiento a 24V: 10000 rpm

Ilustración 26. Motor Spindle 775.

Unido a su eje esta acoplada una pieza de sujeción de brocas, específicamente la ER11 (ver Ilustración 27).

Ilustración 27. Pieza de sujeción de brocas ER11.

Además, para proteger al motor frente a inversiones de polaridad en la corriente se ha instalado un semiconductor DIOTEC "BY 251" (ver Ilustración 28). Presenta las siguientes características:

- Tensión de retorno máxima: 200V
- Corriente directa máxima: 20 A

Ilustración 28. Diodo de protección DIOTEC "BY 251".

Al colocar el diodo en anti-paralelo con el motor, se deriva la intensidad a través del diodo y del propio motor. De esta forma se reduce de notablemente la chispa producida en el contacto.

3.3.7. Relé

Para el control del motor del taladro, se hace uso de un relé. Este permite el paso de la corriente eléctrica a partir del microcontrolador. Se utiliza para controlar un circuito de salida de mayor potencia que el de entrada. El relé utilizado, específicamente, es el "SRD-05VDC-SL-C" (ver Ilustración 29). Presenta las siguientes características:

- Voltaje de operación de la bobina: 5V
- Intensidad de la bobina: 0.75 A
- Voltaje máximo de carga: 30V DC
- Intensidad máxima de carga: 10 A
- Tiempo de acción: 10 ms / 5 ms

Ilustración 29. Relé SRD-05VDC-SL-C.

Además, en el módulo del relé, se coloca un filtro R-C para proteger los contactos del relé y así, aumentar su tiempo de vida (ver Ilustración 30). El filtro se encarga de absorber la intensidad de la sobretensión que se genera al abrir el contacto.

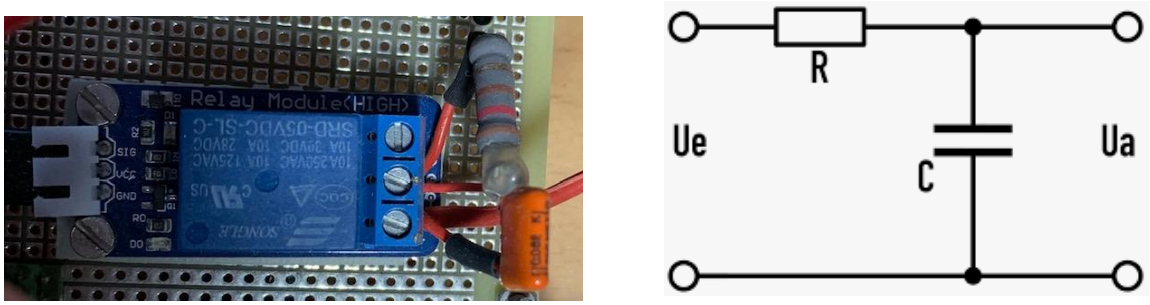


Ilustración 30. Filtro R-C colocado en el modulo del relé.

En la ilustración anterior se puede observar el circuito del filtro R-C. En la salida del filtro se conecta el motor Spindle del taladro.

3.3.8. Finales de carrera

Los finales de carrera son dispositivos mecánicos situados al final del recorrido con el objetivo de enviar señales que puedan modificar el estado de un circuito. En este caso, los finales de carrera se sitúan en el final del recorrido de los ejes X y Z. Además, suponen el punto de partida del sistema (Homing).

En cada eje se sitúan distintos finales de carrera debido a su tamaño. En el eje X se encuentran finales de carrera de la marca OMRON, concretamente los "V-156-1C25" y en el eje Z, se emplean los de la marca CHERRY, específicamente los "DC1C-A1RC" (ver Ilustración 31 y 32). Presentan las siguientes características:

EJE X

- Tensión máxima: 250V AC
- Intensidad: 15 A
- Fuerza máxima: 1.23 N

Ilustración 31. Final de carrera OMRON "V-156-1C25".

EJE Y

- Tensión máxima: 250 V AC
- Intensidad: 6 A
- Fuerza máxima: 0.8 N

Ilustración 32. Final de carrera CHERRY "DC1C-A1RC".

Los finales de carrera suelen producir imprecisiones y oscilaciones debidas al mal contacto o al ruido eléctrico. Por ello, se ha optado por la colocación de un circuito anti rebote para cada eje, que mantiene un nivel lógico estable en todo momento (ver Ilustración 33).

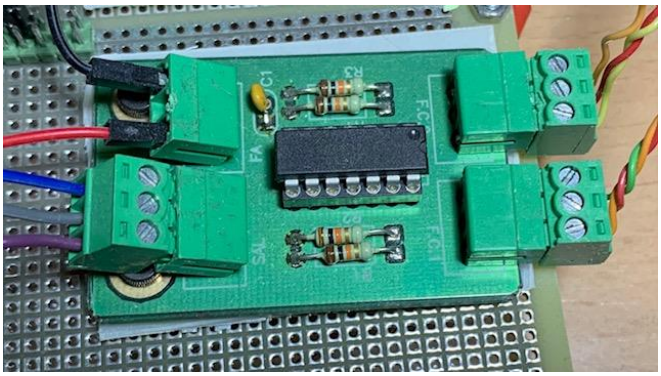


Ilustración 33. Circuito anti rebote para los finales de carrera.

El circuito se ha diseñado con un integrador SN74L compuesto por cuatro puertas NAND que evitan el rebote de la señal (ver Ilustración 34). El circuito se alimenta con un pin de 5V del microcontrolador. Cada circuito anti rebote tiene la capacidad de controlar 2 finales de carrera, por tanto, se usa un circuito para cada eje de la taladradora.

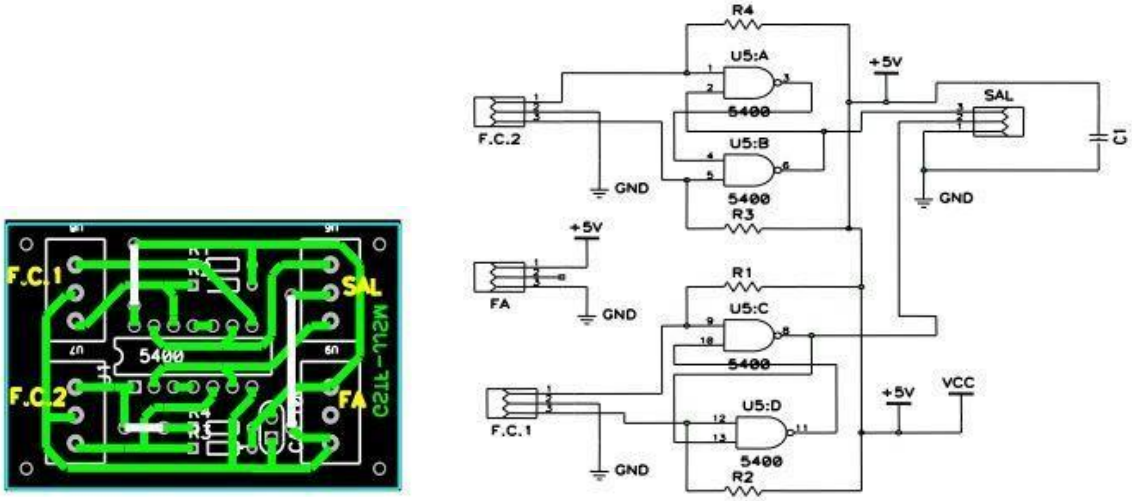


Ilustración 34. Circuito impreso del anti rebote.

3.3.9. Convertidor USB/serie

Las comunicaciones serie USART permiten enviar datos entre diferentes circuitos a velocidades altas y de forma sencilla. En este proyecto se pretende realizar una comunicación entre el HOST y la CPU para el envío de las ordenes GCODES, y entre la CPU y el microcontrolador para la recepción de las ordenes. Cuando el microcontrolador recibe los GCODES, realiza el ciclo de taladrado.

Para ello, se usa un convertidor USB/serie, concretamente el chip FTDi 232R (ver Ilustración 35). Sus características son:

- Conexión mini-USB
- EEPROM de 2014 bits
- Búfer de recepción de 128 bytes y búfer de transmisión de 256 bytes
- Leds de transmisión y recepción de señal.

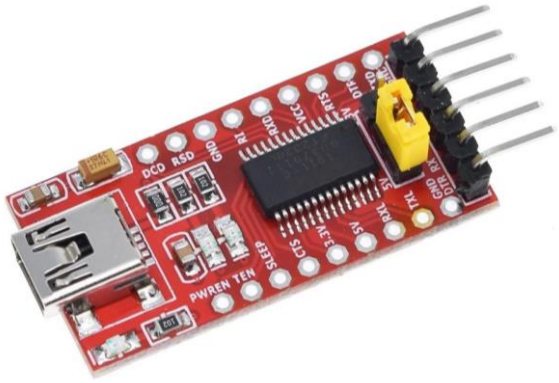


Ilustración 35. Convertidor USB/serie FTDi 232R.

4. PROGRAMACIÓN DEL MICROCONTROLADOR

Para realizar la programación del microcontrolador se pretende simplificar el proceso en la medida de lo posible.

En cuanto al software, se ha utilizado el "STM32CubeIDE", creado por STMicroelectronics. Es un software muy intuitivo en el que se pueden configurar fácilmente los pines de entradas y salidas (ver Ilustración 36).

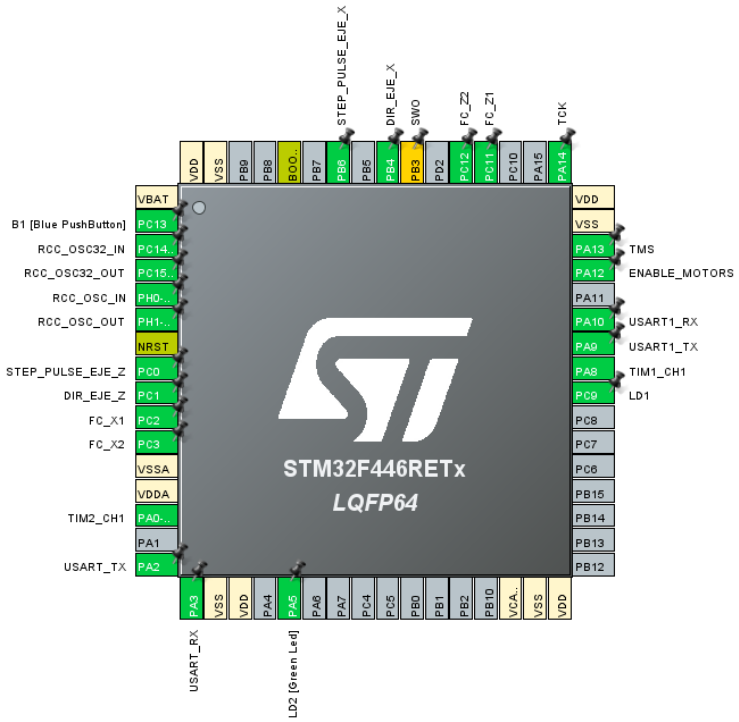
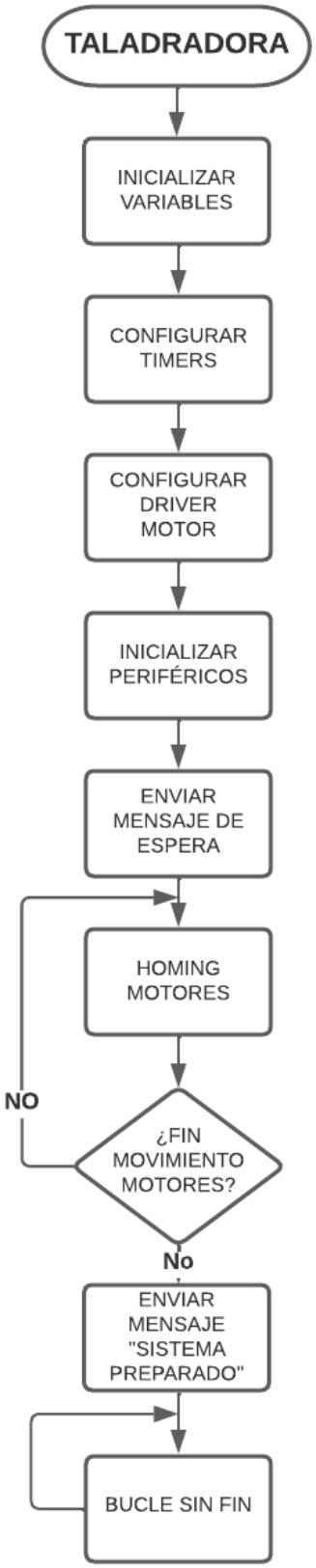
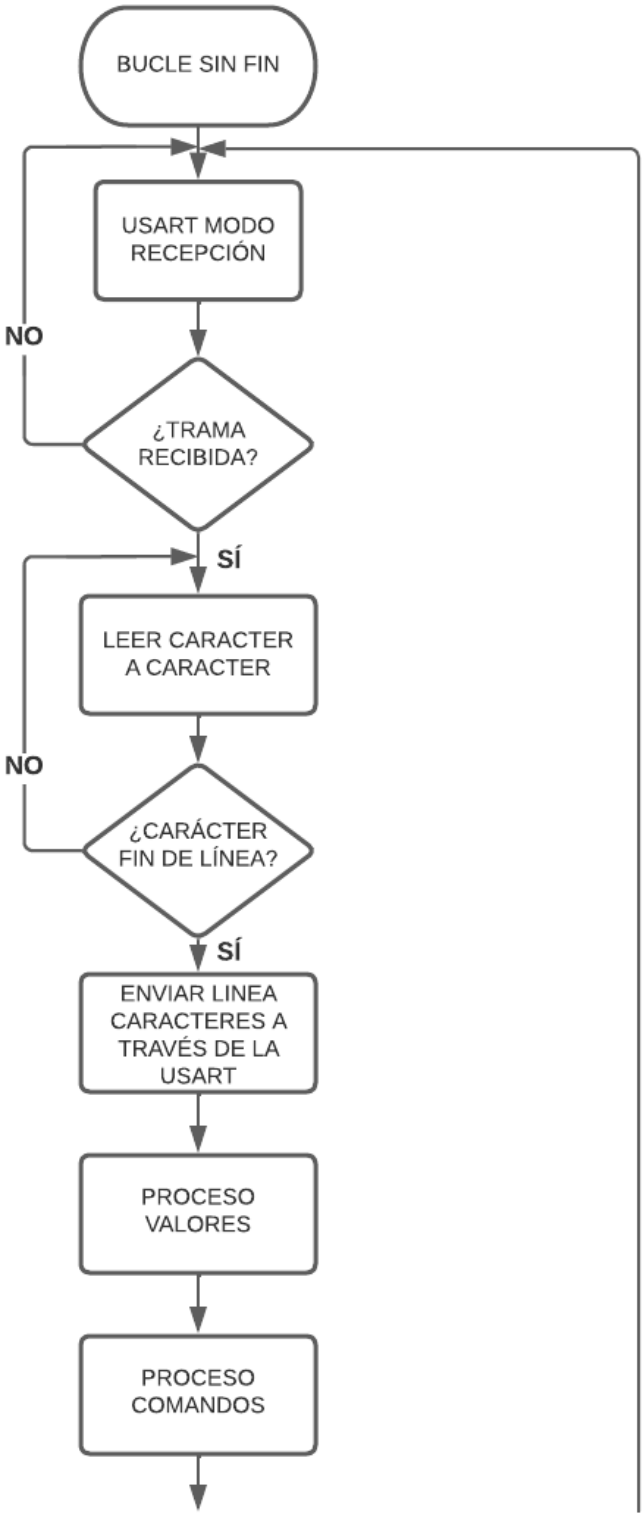


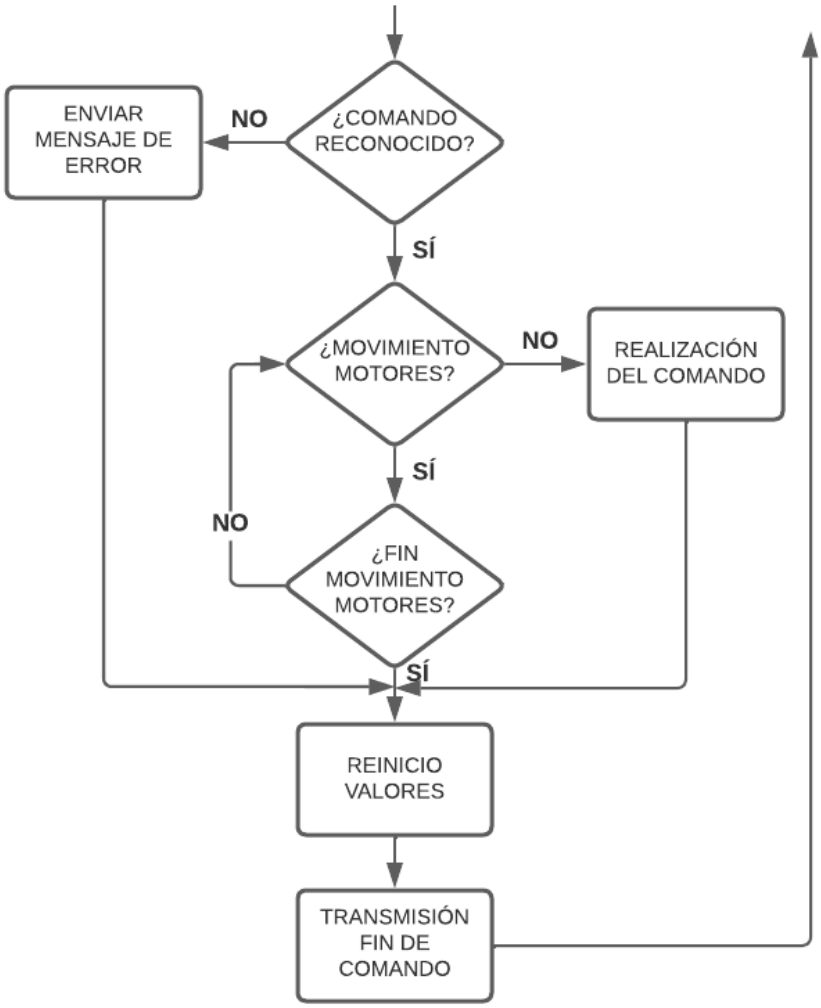
Ilustración 36. Configuración de entradas y salidas en STM32CubeIDE.

Antes de configurar las entradas y salidas del microcontrolador, es necesario realizar un flujograma para tener claro el funcionamiento de la taladradora. Mediante el flujograma se representa gráficamente los pasos y las decisiones tomadas en cada momento.

4.1. Flujograma




BUCLE SIN FIN
USART MODO RECEPCIÓN
NO
¿TRAMA RECIBIDA?
SÍ
LEER CARACTER A CARACTER
NO
¿CARÁCTER FIN DE LÍNEA?
SÍ
ENVIAR LINEA CARACTERES A TRAVÉS DE LA USART
PROCESO VALORES
PROCESO COMANDOS


¿COMANDO RECONOCIDO?
NO
ENVIAR MENSAJE DE ERROR
SÍ
¿MOVIMIENTO MOTORES?
NO
REALIZACIÓN DEL COMANDO
SÍ
NO
¿FIN MOVIMIENTO MOTORES?
SÍ
REINICIO VALORES
TRANSMISIÓN FIN DE COMANDO

4.1. Configuración E/S

Para llevar a cabo el control del sistema, es necesario el uso de entradas y salidas digitales. A continuación se representan todos los pines utilizados del microcontrolador.

El sistema presenta cuatro entradas digitales, que corresponden a los finales de carrera colocados en los extremos de los ejes. Se utilizan dos entradas digitales por cada eje. Los pines del microcontrolador utilizados son:

PIN	FUNCIÓN
PC2	Final de carrera 1 "Eje X"
PC3	Final de carrera 2 "Eje X"
PC11	Final de carrera 1 "Eje Z"
PC12	Final de carrera 2 "Eje Z"

Tabla 1. Entradas digitales usadas para los finales de carrera.

En cuanto a las salidas digitales, cabe destacar, el uso de cinco pines para el control de los motores. Se utilizan dos salidas, una para cada motor, para gestionar la dirección de giro y otras dos salidas, una para cada motor, para los pulsos. La última salida, se emplea para habilitar y deshabilitar todos los motores. Los pines del microcontrolador utilizados son:

PIN	FUNCIÓN
PB4	Dirección motor eje X
PB6	Pulso eje X
PC0	Dirección motor eje Z
PC1	Pulso eje Z
PC8	Relé
PA12	Habilitación motores

Tabla 2. Salidas digitales usadas para los motores.

4.2. Generación de señales PWM

Para el movimiento de los motores paso a paso es necesario enviar una serie de pulsos que se transforman en movimientos angulares. Estas señales están dirigidas por un "Timer", ya que de esta forma se puede controlar la frecuencia, velocidad y posición de los motores. Como en el sistema se utilizan dos motores, se configuran dos señales PWM, una para cada motor. Los pines del microcontrolador utilizados son:

PIN	FUNCIÓN
PA8	PWM 1 → Timer 1 Channel 1
PA0	PWM 2 → Timer 2 Channel 1

Tabla 3. Timers usados por el microcontrolador.

4.3. Comunicación USART

Como se ha visto en el apartado 3.3.8, la comunicación del sistema se realiza entre el HOST y la CPU para el envío de las ordenes GCODES, y entre la CPU y el microcontrolador para la recepción de las ordenes. Todo ello, se realiza por medio del puerto serie, es decir, empleando la comunicación USART (Universal Asynchronous Receiver-Transmitter). Este método de comunicación precisa del uso de dos señales:

- Transmisión (Tx): salida de datos
- Recepción (Rx): entrada de datos

Los pines empleados son:

PIN	FUNCIÓN
PA8	PWM 1 → Timer 1 Channel 1
PA0	PWM 2 → Timer 2 Channel 1

Tabla 4. Pines utilizados para la comunicación USART.

4.4. Comandos GCODE necesarios

Los GCODES son un conjunto instrucciones G y M que le permiten al sistema entender que comandos tienen que seguir para realizar un ciclo de trabajo. Cada comando G y M tiene una acción diferente asignada.

Lógicamente, por razones de diseño, no todos los comandos son aplicables al sistema. Alguno de estos comandos están creados para la realización de desplazamientos circulares, los cuales no son posibles de realizar dado que el taladro actúa sobre el plano horizontal. Por tanto, el sistema solo realiza desplazamientos lineales.

Por consiguiente, los comando implementados, y que por tanto, son los que comprende la máquina se explican, a continuación, de forma detallada.

4.4.1. Comandos G

- G0: Movimiento rápido. Este ciclo realiza un movimiento rápido según el valor especificado en cada uno de los ejes.

- G21: Milímetro como unidad de trabajo seleccionada.

- G28: “Homming”. Vuelve a la posición de inicio

- G81: Ciclo de taladrado. Este ciclo consiste en un proceso de taladrado simple. Primero, se realiza un movimiento rápido en el plano XY desde la posición de “homming” hasta la deseada para realizar el taladrado. Después, se realiza el movimiento del eje Z desde la posición de “homming” hasta la posición especificada para el taladrado. Se realiza el taladro y se vuelve a velocidad rápida a la posición de Z inicial. Por último, se vuelve a la velocidad rápida a la posición de X inicial.

- G90: Distancia absoluta. Todas las posiciones están respecto al 0 (homming).

4.4.2. Comandos M

- M3: Se activa el taladro

- M5: Se desactiva el taladro

4.5. Movimiento de los motores

En primer lugar, al iniciarse el sistema, ambos motores se desplazan a la posición de origen (homming). Esta posición es alcanzada cuando la estructura llega al final del recorrido y toca el final de carrera de cada eje. A partir de ese momento, ya se conoce en todo momento la posición de cada uno de los ejes y su velocidad.

4.5.1. Transformación de movimiento circular a lineal

En cuanto al movimiento de los motores, cabe destacar, que el tipo de movimiento de entrada es diferente al de salida. Por tanto, se produce una transformación de movimiento. Los mecanismos de transformación se clasifican en dos grupos:

- Transformación de movimiento circular-lineal: el elemento de entrada tiene movimiento circular, mientras que el elemento de salida tiene movimiento lineal.

- Transformación de movimiento circular-alternativo: el elemento de entrada tiene movimiento circular, mientras que el elemento de salida tiene movimiento alternativo.

En este proyecto el mecanismo utilizado es de transformación de movimiento circular-lineal. Este mecanismo convierte el movimiento de un piñón, situado en el eje del motor, en uno lineal continuo por parte de la correa dentada (ver Ilustración 37).

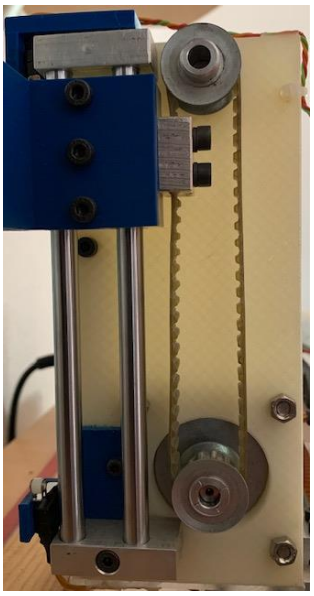


Ilustración 37. Mecanismo de desplazamiento del eje Z.

4.5.1. Cálculo de pasos del motor

En apartados anteriores, se explica el funcionamiento de los motores paso a paso y se exponen sus características. En el sistema se necesita conocer en todo momento la distancia que recorren los motores. Para ello, se calcula la distancia que recorren con cada paso. Con esto, se convierte la distancia que se desea avanzar en el número de pasos que debe efectuar el motor.

El motor RS PRO "440-442" presenta un ángulo de paso de 1.8º. Por tanto, se puede calcular el número de paso que da el motor por cada vuelta.

$$1\ vuelta \rightarrow 360º$$

$$1\ paso \rightarrow 1.8º$$

$$nº\ pasos = \frac{360º}{1.8º} = 200\ pasos/rev$$

Una vez se conoce el número de pasos que da el motor en una vuelta, se procede a calcular la distancia lineal que recorre la correa cuando el piñón da una vuelta completa. La distancia coincide con la longitud total de la circunferencia del piñón.

$$longitud\ recorrida = \pi \cdot diámetro\ piñón = \pi \cdot 20\ mm = 62.83\ mm$$

Por tanto, una vez calculada la distancia recorrida por la correa cuando el piño realiza una vuelta y calculado el número de pasos que realiza el motor por vuelta, se calcula la distancia recorrida por paso.

$$distancia\ por\ paso = \frac{62.83\ mm}{200\ pasos} = 0.31\ mm/paso$$

Se puede concluir que el motor avanza 0.31 mm por cada paso que da. Con este dato, se puede conocer exactamente la posición del motor en cada uno de los ejes.

5. PROGRAMACIÓN EN STM32CUBEIDE

En este capítulo se expone toda la programación realizada para que el sistema funcione correctamente.

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

5.1. Configuración motores

Para configurar los motores se han realizado diversas funciones dentro de un fichero denominado "STEPPER.c". A continuación, se va a mostrar y explicar las funciones utilizadas:

- STEEPER_Move_Blocking_X(uint8_t au8_STEPPER_Instance, uint32_t au32_Steps, uint8_t au8_DIR)

- STEEPER_Move_Blocking_Z(uint8_t au8_STEPPER_Instance, uint32_t au32_Steps, uint8_t au8_DIR)

- STEEPER_MoveSlow_Blocking_X(uint8_t au8_STEPPER_Instance, uint32_t au32_Steps, uint8_t au8_DIR)

- STEEPER_MoveSlow_Blocking_Z(uint8_t au8_STEPPER_Instance, uint32_t au32_Steps, uint8_t au8_DIR)

Se hace uso de cuatro funciones para el movimiento de los motores. Las dos primeras funciones sirven para mover el eje X y el eje Z a velocidad media y las dos últimas sirven para mover los ejes a velocidad lenta.

Con estas funciones se puede controlar tanto la velocidad como la posición del motor. Para ello, consta de tres parámetros:

- au8_STEPPER_Instance: Este parámetro indica el motor que se desea mover, es decir, el motor del eje X o del eje Z.
- au32_Steps: Este parámetro, indica el número de pasos que ejecuta el motor. En el apartado anterior, se ha calculado la distancia que recorre por cada paso. De esta forma, se consigue mover el motor a la distancia propuesta.
- au8_DIR: Este parámetro indica el sentido de giro del motor, es decir, horario o antihorario.

Para controlar la velocidad del motor se recurre a la propia función. Esta creada mediante dos retardos que al aumentarlos o disminuirlos cambia la velocidad. Cuando se aumenta el retardo la velocidad disminuye y cuando el retardo disminuye la velocidad aumenta.

Sin embargo, este sistema se configura a una velocidad media para evitar posibles

vibraciones mecánicas

5.2. Programa principal

El programa principal se encuentra en el fichero "main.c" y se sirve del fichero "STEPPER.c" para su correcto funcionamiento. A continuación se explica las secciones del programa principal.

El primer paso es definir las variables a utilizar en el proceso. Una vez definidas, se configura el reloj del sistema y se inician los periféricos (GPIO, USART).

Ahora se explican las funciones utilizadas en el programa principal, para entender de forma sencilla el funcionamiento del sistema.

La primera función es la de "homming". Con esta función, se pretende que el sistema empiece siempre en la misma posición. Para ello, se configura el sentido de giro y se hace girar el motor, tanto del eje X como del Z, hasta que toque el final de carrera. Una vez se pulsa el final de carrera, el sistema se detiene.

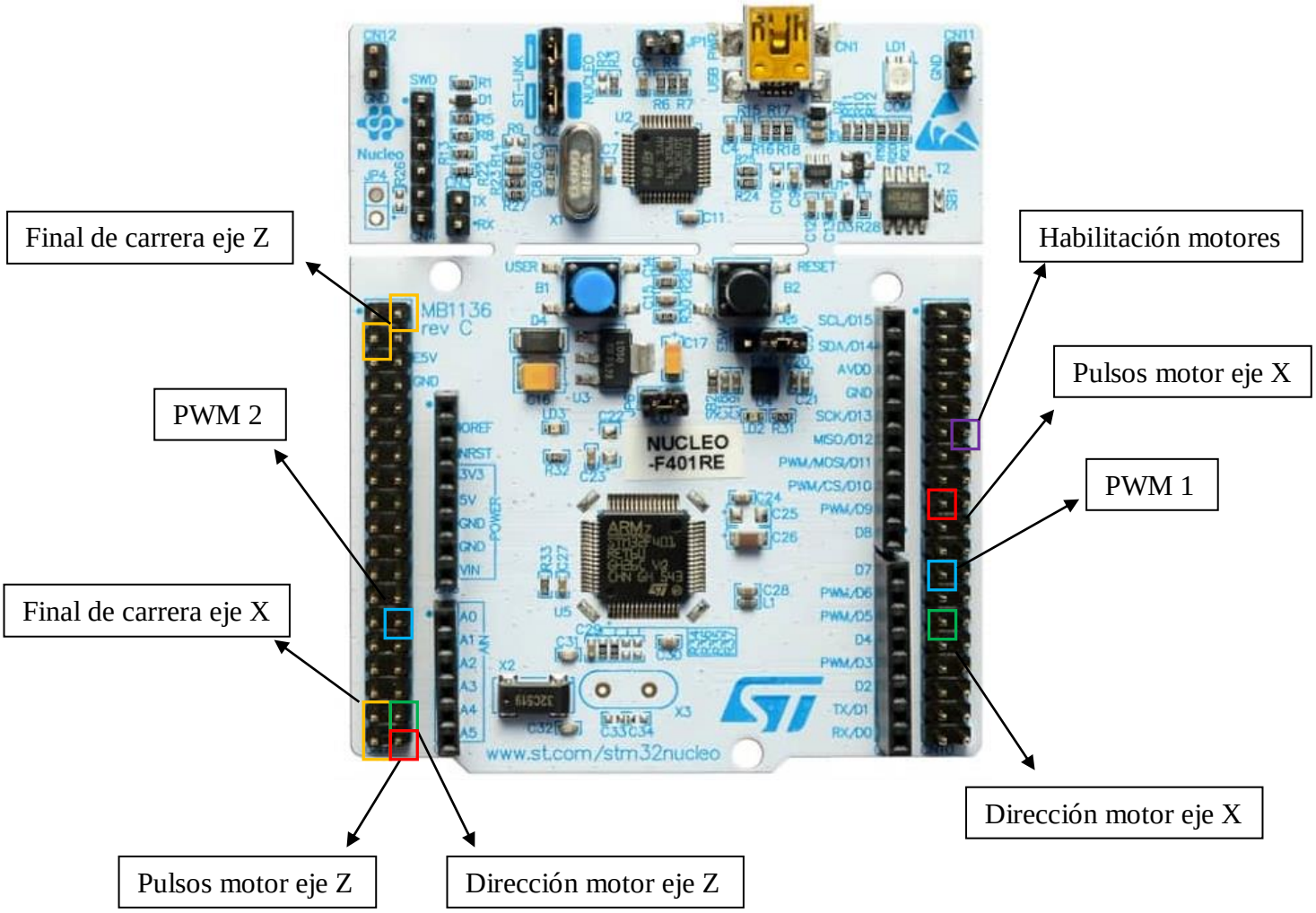
La segunda función es la interrupción de la USART. Cuando se envía un GCODE, la USART almacena el GCODE carácter a carácter. En ese momento, la variable "Fin_Linea" se pone a 0 y en el programa principal se ejecutará la orden almacenada.

Una vez se ha explicado las funciones del programa principal, se dispone a explicar el código restante.

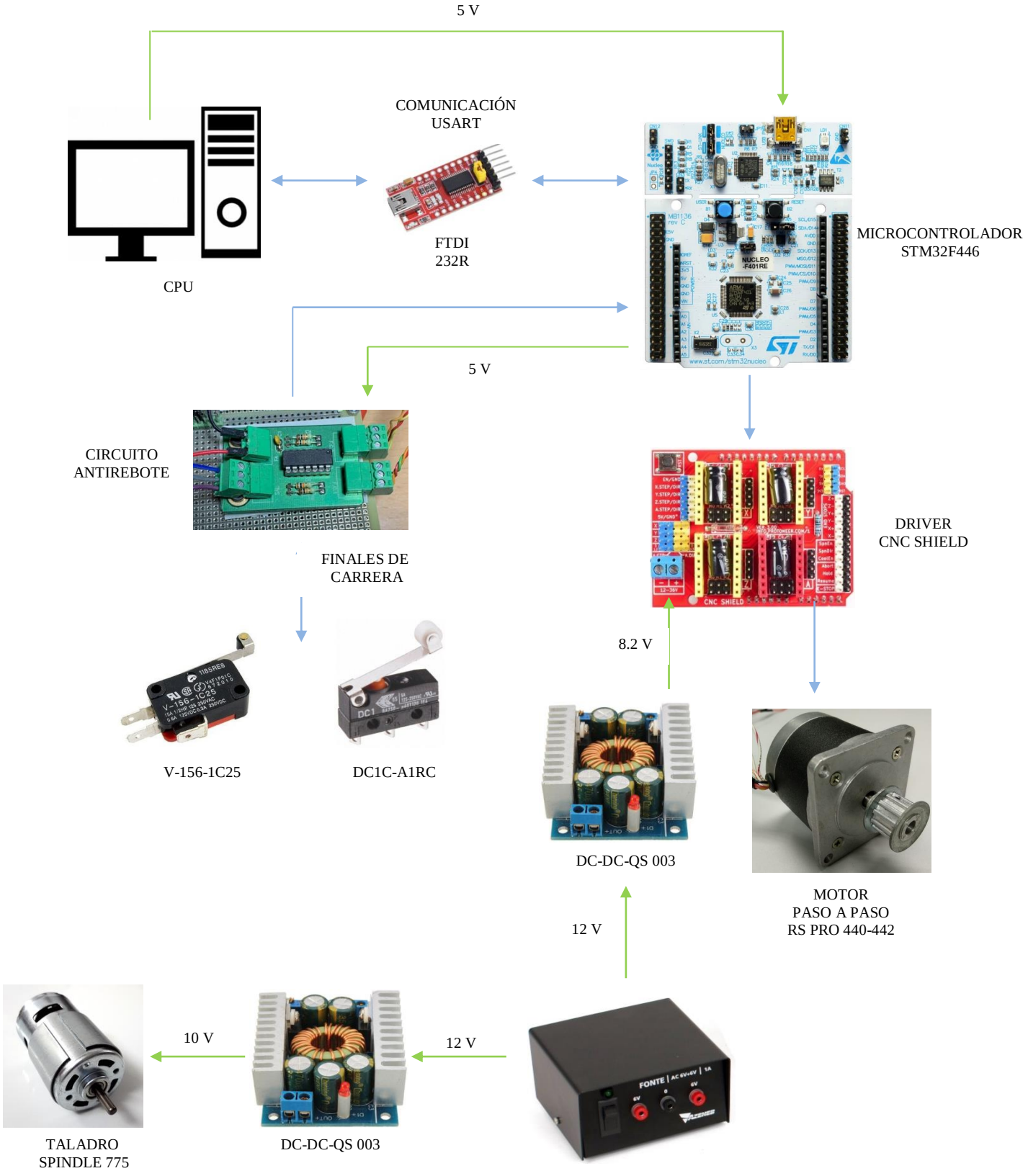
Antes del inicio del sistema, se realiza un movimiento de "homming" de los motores. Además, se envía por la USART un mensaje para saber que se ha inicializado el sistema.

Una vez termina el "homing", entra en un bucle infinito en el que el sistema espera hasta que le llega una línea de GCODE por la USART. Esto es así debido a que la USART está configurada por interrupciones. Cuando llega una línea de GCODE, la variable "Fin_Linea" se pone a 0 y entonces entra en la estructura "switch". Dentro del "switch" hay dos casos, uno para las instrucciones "G" y otro para las instrucciones "M". Dependiendo del GCODE recibido, el sistema ejecuta una instrucción u otra. Una vez ejecutada la instrucción, la variable "Fin_Linea" se pone a 1 y por tanto el sistema se queda esperando en un bucle infinito hasta que reciba otra instrucción GCODE.

6. DISEÑO Y CONEXIONADO DEL SISTEMA



A continuación se observa un esquema con todas las conexiones del sistema.



*En el ANEXO C se encuentra un plano detallado de las conexiones de los dispositivos electrónicos.

7. CONFIGURACIÓN PINES STM32CUBEIDE

Una vez se conoce en que pines se conectan los dispositivos electrónicos en el microcontrolador, se muestra como queda la configuración de los pines en el software "STM32CubeIDE" (ver Ilustración 38).

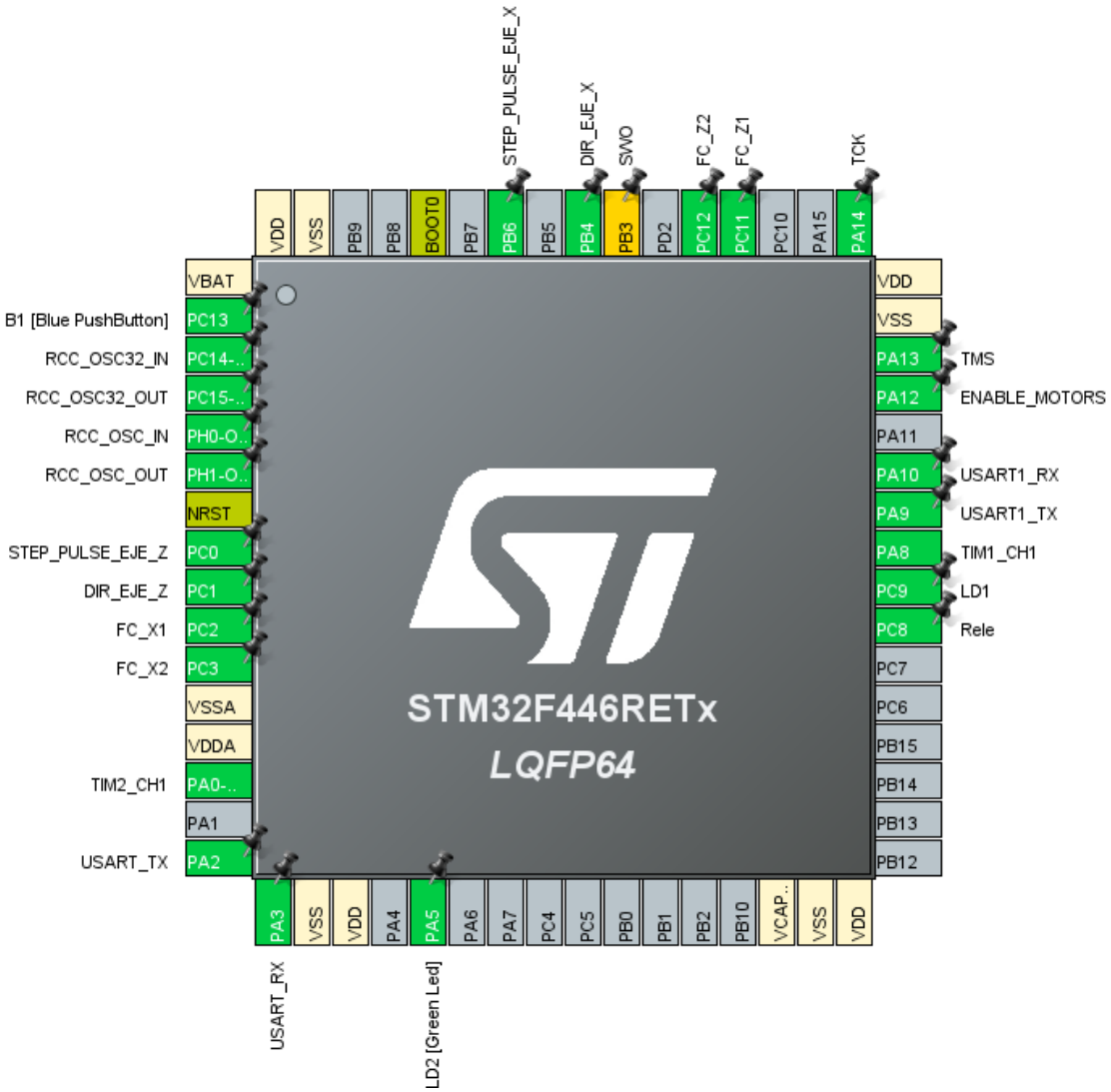


Ilustración 38. Configuración pines STM32CubeIDE.

8. POSIBLES MEJORAS DEL SISTEMA

Una vez finalizado el sistema y verificado su correcto funcionamiento, se comprueba que cumple todos los objetivos marcados al principio del proyecto. Sin embargo, como cualquier sistema, es posible mejorar tanto la parte mecánica como la parte electrónica. Incluso se pueden llevar a cabo modificaciones en el software para mejorar ciertos aspectos que pueden observarse durante el funcionamiento prolongado del sistema.

Antes de plantear cualquier tipo de mejora, se debe tener en cuenta que el sistema está diseñado y construido para uso docente con la finalidad de ayudar en el aprendizaje sobre las máquinas CNC. Por tanto, no debe ser un sistema demasiado sofisticado.

En cuanto a las mejoras, se pueden dividir según sean mecánicas o electrónicas.

En la parte mecánica, una posible mejora es la multiherramienta. Se podría implementar el uso de otro tipo de herramientas, como herramientas láser o de corte. De esta forma, la máquina sería más versátil y podría utilizarse para realizar diferentes tipos de trabajo.

Además, podría reducirse posibles rozamientos lubricando toda la estructura de forma adecuada. Con esto, se mejoraría notablemente el movimiento de los ejes del sistema.

En la parte electrónica se podrían realizar diversas mejoras. Una de ellas, es la implementación de un tercer eje. Al tratar de simplificar el sistema se optó por el uso de dos ejes, aunque sería posible es uso de un tercero ya que el driver permite el control de hasta cuatro motores.

Además, otra mejora, sería realimentar el sistema para llevar a cabo el control de los motores en bucle cerrado mediante el empleo de encoders, para limitar la posible pérdida de pasos. También, se pueden sustituir los motores paso a paso actuales por servomotores, lo que dotaría al sistema de mayor precisión.

En cuanto a la programación, es posible llevar a cabo ciertas mejoras de una manera gradual, es decir, aunque funcione correctamente, conforme se va usando es posible corregir ciertos detalles que surgen debido al uso prolongado.

9. CONCLUSIÓN

En principal objetivo del proyecto es el diseño y programación de un sistema CNC para uso docente. Una vez terminado, se puede observar que cumple con su finalidad.

La idea era desarrollar un sistema CNC simplificado que tuviera una gran diversidad de elementos, tanto mecánicos como electrónicos.

Se trata de un proyecto que engloba todos los conocimientos adquiridos durante el Máster de Ingeniería Mecatrónica. Los conocimientos mecánicos han sido necesarios para llevar a cabo la construcción de la estructura. Los conocimientos electrónicos han sido realmente útiles para el conexionado y el uso de los elementos electrónicos. También, cabe destacar, los conocimientos informáticos que han sido imprescindibles para programar el microcontrolador. Todo ello, ha permitido el desarrollo de un proyecto como este.

Además, se ha obtenido un firmware propio, baso en GRBL, combinando el código de control de los motores junto a los comandos GCODE. Esto facilita la comprensión del GRBL que es la base de cualquier maquina CNC.

Con este proyecto, se espera ayudar y motivar a las futuras generaciones de alumnos, a comprender el uso de las máquinas CNC ya que es una tecnología realmente importante a día de hoy.

LISTADO DE TABLAS E IMÁGENES

Listado de tablas

Listado de figuras

REFERENCIAS

Ingeniería Mecafenix. 2017. *Motores paso a paso ¿Qué es y cómo funcionan?* Recuperado (11/09/2022) de: https://www.ingmecafenix.com/electricidad-industrial/motor-paso-a-paso/

Anónimo. 2018. *Mecanismos de transformación del movimiento.* Recuperado (11/09/2022) de: https://aprendemostecnologia.org/maquinas-y-mecanismos/mecanismos-de-transformacion-del-movimiento/

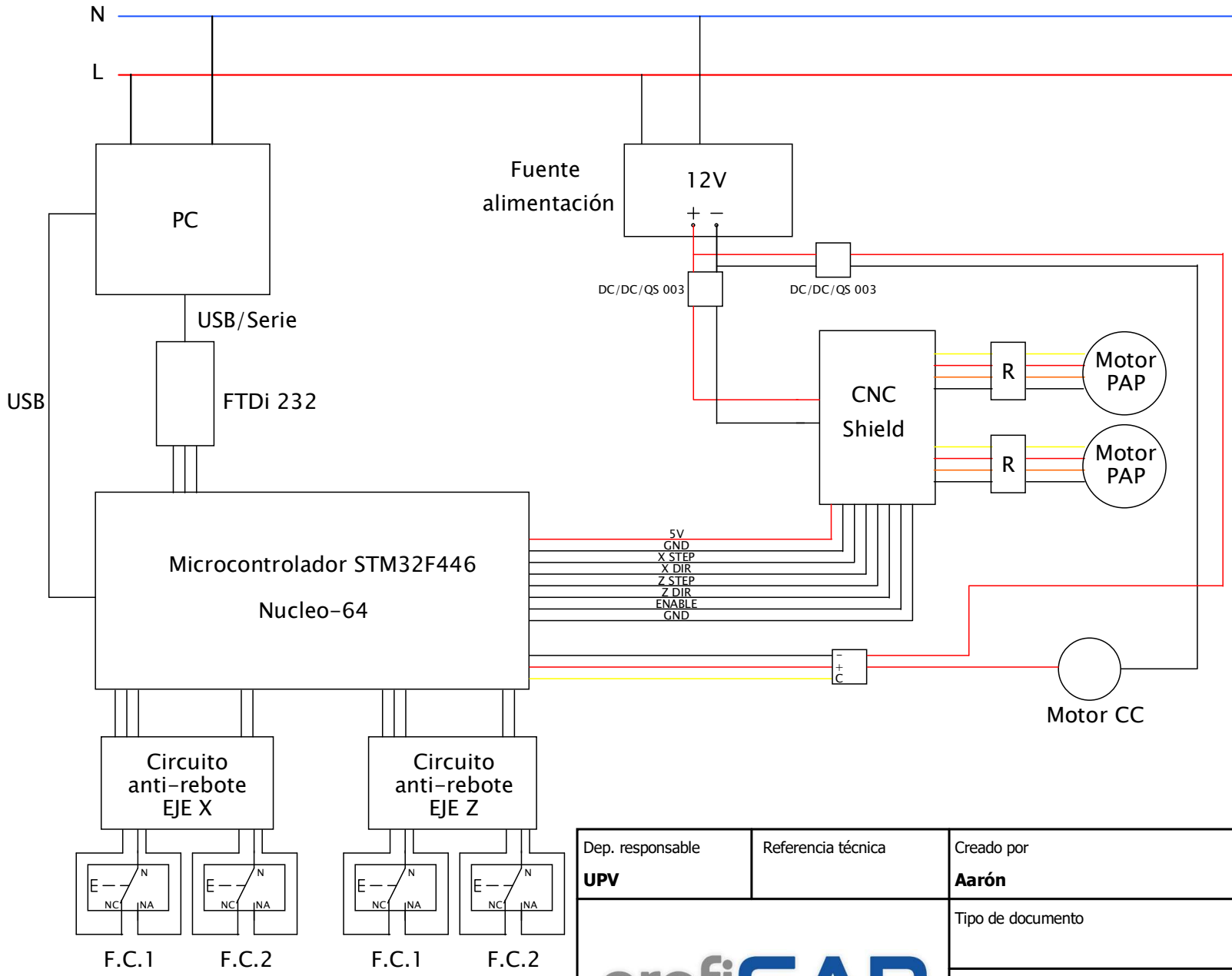
Bertus Kruger. 2015. *Arduino CNC Shield.* Recuperado (11/09/2022) de: https://blog.protoneer.co.nz/arduino-cnc-shield/#BoardLayout

Naylamp Mechatronics. 2021. *DRV8825 Driver motor paso a paso*. Recuperado (11/09/2022) de: https://naylampmechatronics.com/driver-pap-stepper/126-driver-pap-drv8825.html

ST. *Nuclo-F446RE.* Recuperado (11/09/2022) de: https://www.st.com/en/evaluation-tools/nucleo-f446re.html

PLANOS

Análisis estadístico para la detección y reducción de daños en la carrocería del modelo Ford Kuga en la factoría Ford en Almussafes (Valencia, España).



Dep. responsable **UPV**	Referencia técnica	Creado por **Aarón**	Aprobado por **Aarón**	Escala **1:1**
profiCAD		Tipo de documento	Estado del documento	Rev.
		Título, título suplementario **Conexionado taladradora**		
			Fecha de edición **07/09/2022**	Idioma / Hoja **1/1**

PRESUPUESTO

Análisis estadístico para la detección y reducción de daños en la carrocería del modelo Ford Kuga en la factoría Ford en Almussafes (Valencia, España).

10. ESTUDIO ECONÓMICO

Para realizar un proyecto es necesario valorar económicamente la viabilidad del trabajo, por ello es necesario realizar un presupuesto sobre el sistema.

Se procede a realizar una estimación del coste de todos los elementos presentes a la hora de diseñar y fabricar el sistema CNC. En esta estimación se reflejan todos los materiales, tanto mecánicos como electrónicos. Además, se tiene en cuenta las licencias del software utilizado y la mano de obra empleada.

Una vez desarrollado el estudio económico del proyecto no es necesario la aplicación del IVA, dado que viene impuesto sobre los elementos utilizados y no se trata de un producto para mercado, sino para uso docente.

10.1. Materiales

En el estudio económico de los materiales se realiza un desglose según si el material es mecánico o electrónico.

PRESUPUESTO ELEMENTOS MECÁNICOS			
Elemento	Coste (€/unidad)	Cantidad (unidad)	Total (€)
Base metacrilato (50x50 mm)	11,51	1	11,51
Varillas lisas calibradas	5,03	4	20,12
Estructura de acero	56	1	56
Polea dentada	8,1	4	32,4
Correa dentada eje X (30 cm)	9,49	1	9,49
Correa dentada eje Z (13 cm)	3,71	1	3,71
IMPORTE TOTAL			133,23

Tabla 5. Presupuesto de los elementos mecánicos.

PRESUPUESTO ELEMENTOS ELECTRÓNICOS			
Elemento	Coste (€/unidad)	Cantidad (unidad)	Total (€)
Microcontrolador STM32F446	18	1	18
Driver CNC Shield	4,45	1	4,45
Convertidor serie FTDi 232R	5,99	1	5,99
Motor PAP RS PRO 440-442	41,28	2	82,56
Motor Spindle 755	20,31	1	20,31
Transformador DC-DC-QS 003	12,17	1	12,17
Fuente de alimentación BM PS01	14,72	1	14,72
F.C. V-156-1C25	0,92	1	0,92
F.C. DC1C-A1RC	3,51	1	3,51
Relé SRD-05VDC-SL-C	4,5	1	4,5
Resistencia motor	0,67	4	2,68
Resistencia relé	0,12	1	0,12
Condensador relé	0,42	1	0,42
Diodo DIOTEC BY 251	0,52	1	0,52
Circuito anti rebote	5,8	2	11,6
Placa PCB	2,21	1	2,21
Cable USB/serie	2,35	1	2,35
PC	449,99	1	449,99
IMPORTE TOTAL			637,02

Tabla 6. Presupuesto de los elementos electrónicos.

Una vez se ha obtenido el presupuesto de los materiales según el tipo, se procede a calcular el importe total de los materiales en conjunto.

IMPORTE TOTAL MATERIALES (€)	770,25

Tabla 7. Presupuesto total de los materiales.

10.2. Licencias

Para el uso de algún software es necesario tener una licencia contratada. En este proyecto se han usado dos tipos diferentes de software, el "STM32CubeIde" y el "Microsoft Office 2022". Sin embargo, solo ha sido necesario obtener la licencia de "Microsoft Office 2022". Por tanto el presupuesto de la licencia sería el siguiente:

LICENCIA		
Elemento	Tipo	Total (€)
Microsoft Office 2022	Licencia de estudiante	135
IMPORTE TOTAL		135

Tabla 8. Presupuesto del software utilizado.

10.3. Mano de obra

En cuanto a la mano de obra, cabe destacar que ha sido realizada por un ingeniero. Sin embargo, el estudio económico se divide según las funciones realizadas. La primera función realizada es la técnica, ya que se necesita montar la estructura y cablearla. La otra función es la de ingeniería, ya que se programa el microcontrolador con todos los elementos electrónicos.

MANO DE OBRA			
Elemento	Cantidad (h)	Coste unitario (€/h)	Total (€)
Ingenieria	255	15	3825
Técnica	50	8	400
IMPORTE TOTAL			4225

Tabla 9. Presupuesto de la mano de obra.

10.4. Presupuesto final

Este presupuesto es una estimación de lo que costaría la creación del sistema. Sin embargo, el precio real ha sido menor ya que la gran mayoría de los materiales son reutilizados. Además, la licencia es gratuita por ser alumno de la universidad y la mano de obra es realizada por un alumno, con el fin de obtener el título del Máster de Ingeniería Mecatrónica.

PRESUPUESTO TOTAL	
Tipo	Total (€)
Materiales	770,25
Licencia	135
Mano de obra	4225
IMPORTE TOTAL	5130,25

Tabla 10. Presupuesto total del proyecto.

PLIEGO DE CONDICIONES

Diseño y programación de un sistema CNC de dos ejes para uso docente.

En el presente pliego de condiciones se declaran las extensiones del contrato que deben existir entre contratista y propiedad en la ejecución del trabajo de fin de máster.

11. CONDICIONES DE LOS MATERIALES

En el documento presente se desarrolla el contenido explicativo del proceso seguido para el diseño y la programación de una taladradora CNC mediante el uso de GCODES.

Toda la documentación generada por escrito en este documento, así como el plano de conexionado y los archivos de STM32 podrán ser accesibles para cualquiera que lo desee, tanto para su uso particular como para posteriores ampliaciones o variaciones sobre el trabajo. La regulación de este trabajo queda por tanto en disposición de la UPV para su distribución a través de cualquiera de sus plataformas.

El contenido generado en el proyecto a parte del presente documento son los archivos de código usados para la programación de la taladradora.

Las condiciones de trabajo de los diferentes softwares utilizados durante el proyecto, son los siguientes:

- STM32CubeIDE: Ha sido empleada la versión 1.8.0, disponible para su descarga libre.

- STM32CubeMX: Ha sido empleada la versión 6.6.1, disponible para su descarga libre.

12. OBLIGACIONES DE LAS PARTES

12.1. Obligaciones del contratante

El contratante deberá:

- Proporcionar acceso a las fuentes de información necesarias para que el contratista cumpla su labor.

- Proporcionar, tanto información como materiales necesarios para realizar las actividades bajo las medidas de seguridad impuestas en la empresa.

12.2. Obligaciones del contratista

El contratista empleará todas sus habilidades adquiridas para:

- Recopilar datos e información sobre la programación CNC.
- Elegir correctamente los componentes que se utilizan en el sistema.
- Introducir mejoras mecánicas y a nivel de programación a medida que se elabora la taladradora.

ANEXO A: PROGRAMACIÓN DEL SISTEMA

Programa principal (main.c)

```
int main(void)
{
  /* USER CODE BEGIN 1 */
    //uint8_t i;
    char Buffer_OK[] = "OK";
  uint8_t char_counter;
  char letter, car, eje, eje_mov;
  float value;
  uint32_t int_value;
  uint32_t valor;
  uint8_t buff_mov[64], buff_long[12];
  uint32_t valor_num;


  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART2_UART_Init();
  MX_USART1_UART_Init();
  MX_TIM1_Init();
  MX_TIM2_Init();
  MX_TIM14_Init();
  /* USER CODE BEGIN 2 */
  indice = 0;
  Fin_Linea = NO_SI;
  Print_Usart2("Sistema Inicializado \n");

  STEPPERS_Init();


  // Pone en marcha por Interrupciones la recepcion de los
  // caracteres de la linea de GCODES. Lee el primer caracter.
  HAL_UART_Receive_IT(&huart1, (uint8_t*) Rx_data, 1);

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */

  HAL_GPIO_WritePin(ENABLE_MOTORS_GPIO_Port, ENABLE_MOTORS_Pin, GPIO_PIN_RESET); //En RESET se habilitan los motores
  Homing();
  HAL_GPIO_WritePin(ENABLE_MOTORS_GPIO_Port, ENABLE_MOTORS_Pin, GPIO_PIN_SET); //En SET se deshabilitan los motores
```

```
154  while (1)
155  {
156
157      if (Fin_Linea == SI)
158      {
159
160            char_counter = 0;
161            int_value = 0;
162            valor = 0;
163            Print_Usart2(buffer_TLDR_Rx);
164            Print_Usart2("--- \n");
165
166            letter = buffer_TLDR_Rx[char_counter];
167            char_counter ++;
168            if (!read_number (buffer_TLDR_Rx, char_counter, &value)) { Print_Usart2("STATUS-0_BAD_NUMBER_FORMAT \n");
169            int_value = truncf(value);
170            sprintf(buff_long,"%u", int_value);
171            char_counter = char_counter + strlen(buff_long);
172            sprintf(buff_mov,"char_counter con b_long = %u \n", char_counter);
173            Print_Usart2(buff_mov);
174
175
176            switch(letter) {
177
178                // 'G' and 'M' Command Words: Parse commands and check for modal group violations.
180       case 'G':
181           // Determine 'G' command and its modal group
182           switch(int_value){
183               case 0: case 1:
184                   Print_Usart2("Entra case 0 o 1  \n");
185                   eje = buffer_TLDR_Rx[char_counter];
186                   while (eje == ' '){
187                       char_counter++;
188                       eje = buffer_TLDR_Rx[char_counter];
189                       Print_Usart2("Entra while  \n");
190                   }
191                   sprintf(buff_mov,"eje = %c \n", eje);
192                   Print_Usart2(buff_mov);
193                   if (eje == 'X'){
194                       Print_Usart2("Entra eje X  \n");
195                       char_counter++;
196                       if (!read_number(buffer_TLDR_Rx, char_counter, &value)) { Print_Usart2("STATUS-1_BAD_NUMBER_FORMAT \n"); }
197                       int_value = truncf(value);
198                       if (int_value > 200) int_value = 200; //Protección
199                       sprintf(buff_mov,"Mueve eje X %u mm \n", int_value);
200                       Print_Usart2(buff_mov);
201                       HAL_Delay(500);
202                       valor = (200 * int_value)/60;
203                       HAL_GPIO_WritePin(ENABLE_MOTORS_GPIO_Port, ENABLE_MOTORS_Pin, GPIO_PIN_RESET);
204                       STEPPER_Move_Blocking_X(STEPPER_EJE_X, valor, DIR_CCW);
205                   }
206                   if (eje == 'Z'){
207                       Print_Usart2("Entra eje Z  \n");
208                       char_counter++;
209                       if (!read_number(buffer_TLDR_Rx, char_counter, &value)) { Print_Usart2("STATUS-2_BAD_NUMBER_FORMAT \n"); }
210                       int_value = truncf(value);
211                       if (int_value > 40) int_value = 40; //Protección
212                       sprintf(buff_mov,"Mueve eje Z %u mm \n", int_value);
213                       Print_Usart2(buff_mov);
214                       HAL_Delay(500);
215                       valor = (200 * int_value)/60;
216                       HAL_GPIO_WritePin(ENABLE_MOTORS_GPIO_Port, ENABLE_MOTORS_Pin, GPIO_PIN_RESET);
217                       STEPPER_Move_Blocking_Z(STEPPER_EJE_Z, valor, DIR_CCW);
218                   }
219               break;
220
221                 case 21:
222                     Print_Usart2("Usa medidas en milimetros \n");
223                     HAL_Delay(500);
224                 break;
225
226                 case 28:
227                     Print_Usart2("Va al origen \n");
228                     HAL_GPIO_WritePin(ENABLE_MOTORS_GPIO_Port, ENABLE_MOTORS_Pin, GPIO_PIN_RESET);
229                     Homing();
230                     break;
231
```

```
232             case 81:
233                 HAL_GPIO_WritePin(ENABLE_MOTORS_GPIO_Port, ENABLE_MOTORS_Pin, GPIO_PIN_RESET);
234                 Print_Usart2("Ciclo de taladrado  \n");
235                 eje = buffer_TLDR_Rx[char_counter];
236                 while (eje == ' '){
237                     char_counter++;
238                     eje = buffer_TLDR_Rx[char_counter];
239                     Print_Usart2("Entra while  \n");
240                 }
241                 sprintf(buff_mov,"eje = %c \n", eje);
242                 Print_Usart2(buff_mov);
243                 if (eje == 'Z'){
244                     Print_Usart2("Entra eje Z  \n");
245                     char_counter++;
246                     if (!read_number(buffer_TLDR_Rx, char_counter, &value)) { Print_Usart2("STATUS-2_BAD_NUMBER_FORMAT \n"); }
247                     int_value = truncf(value);
248                     if (int_value > 20) int_value = 20; //Protección
249                     sprintf(buff_mov,"Mueve eje Z %u mm \n", int_value);
250                     Print_Usart2(buff_mov);
251                     HAL_Delay(500);
252                     valor = (200 * int_value)/60;
253                     HAL_GPIO_WritePin(ENABLE_MOTORS_GPIO_Port, ENABLE_MOTORS_Pin, GPIO_PIN_RESET);
254                     STEPPER_MoveSlow_Blocking_Z(STEPPER_EJE_Z, valor, DIR_CCW);
255                     HAL_Delay(3000);
256                     STEPPER_MoveSlow_Blocking_Z(STEPPER_EJE_Z, valor, DIR_CW);
257                 }
258             break;
259             default: Print_Usart2("STATUS_GCODE_UNSUPPORTED_COMMAND \n"); // [Unsupported G command]
260         }
261     break;
261       break;
262       case 'M':
263           // Determine 'M' command and its modal group
264           switch(int_value){
265               case 3:
266                   Print_Usart2(" ACTIVA el TALADRO \n");
267                   HAL_Delay(500);
268                   HAL_GPIO_WritePin(Rele_GPIO_Port, Rele_Pin, GPIO_PIN_SET);
269               break;
270
271               case 5:
272                   Print_Usart2(" DESACTIVA el TALADRO \n");
273                   HAL_Delay(500);
274                   HAL_GPIO_WritePin(ENABLE_MOTORS_GPIO_Port, ENABLE_MOTORS_Pin, GPIO_PIN_SET); //Se deshabilitan los motores
275                   HAL_GPIO_WritePin(Rele_GPIO_Port, Rele_Pin, GPIO_PIN_RESET);
276               break;
277               default: Print_Usart2("STATUS_MCODE_UNSUPPORTED_COMMAND \n"); // [Unsupported M command]
278           }
279   }
280               .
281               Fin_Linea = NO_SI;
282               HAL_Delay(20);
283               indice = 0;
284               buffer_TLDR_Rx[0] = '\0';
285               USART1_PutString (Buffer_OK);
286
287               HAL_UART_Receive_IT(&huart1, (uint8_t*) Rx_data, 1);
288       }
289
290       HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_SET);
291       HAL_Delay(100);
292       HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);
293       HAL_Delay(100);
294
295     /* USER CODE END WHILE */
296
297     /* USER CODE BEGIN 3 */
298
299   }
300   /* USER CODE END 3 */
}
```

Funciones (main.c)

```
/* USER CODE BEGIN 4 */

void Homing(void){

    HAL_GPIO_WritePin(DIR_EJE_Z_GPIO_Port, DIR_EJE_Z_Pin, GPIO_PIN_RESET);
    while (HAL_GPIO_ReadPin(FC_Z1_GPIO_Port, FC_Z1_Pin) == 1){
        HAL_GPIO_WritePin(STEP_PULSE_EJE_Z_GPIO_Port, STEP_PULSE_EJE_Z_Pin, GPIO_PIN_SET);
        HAL_Delay(8);
        HAL_GPIO_WritePin(STEP_PULSE_EJE_Z_GPIO_Port, STEP_PULSE_EJE_Z_Pin, GPIO_PIN_RESET);
        HAL_Delay(8);
    }

    HAL_GPIO_WritePin(DIR_EJE_X_GPIO_Port, DIR_EJE_X_Pin, GPIO_PIN_RESET);
    while (HAL_GPIO_ReadPin(FC_X1_GPIO_Port, FC_X1_Pin) == 1){
        HAL_GPIO_WritePin(STEP_PULSE_EJE_X_GPIO_Port, STEP_PULSE_EJE_X_Pin, GPIO_PIN_SET);
        HAL_Delay(8);
        HAL_GPIO_WritePin(STEP_PULSE_EJE_X_GPIO_Port, STEP_PULSE_EJE_X_Pin, GPIO_PIN_RESET);
        HAL_Delay(8);

    }
}
```

```
uint8_t read_number (char Buff_Rec[64], uint8_t car_counter, float *float_ptr)
{
    uint8_t indice_car, indice_val;
    uint8_t c;
    uint32_t valor;
    uint8_t Buff_Value[12], buff_debug[24];

    indice_val = 0;
    strcpy(Buff_Value, "");
    indice_car = car_counter;
    sprintf(buff_debug, "indice_car = %u \n", indice_car);
    Print_Usart2(buff_debug);
    c = Buff_Rec[indice_car];

    while (c >= '0' && c <= '9'){
        Buff_Value[indice_val] = c;
        sprintf(buff_debug, "ival = %u - icar = %u \n", indice_val, indice_car);
        Print_Usart2(buff_debug);

        indice_val++;
        indice_car++;
        c = Buff_Rec[indice_car];
    }
    if (strlen(Buff_Value) == 0) return(false);
    car_counter = indice_car;
    valor = atoi(Buff_Value);
    sprintf(buff_debug, "valor = %u \n", valor);
    Print_Usart2(buff_debug);
      float fval;
      fval = (float)valor;
      *float_ptr = fval;

    return(true);
}
```

```
409 void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
410 {
411     char car[2];
412 
413     if (huart->Instance == USART1)  //current UART
414     {
415         car[0] = Rx_data[0];
416         //USART2_Put_char(car);
417 
418         if (indice == 0)
419         {
420             if (car[0] == 'G' || car[0] == 'M'){
421                 buffer_TLDR_Rx[indice] = car[0];
422                 indice++;
423             }
424         }
425         else
426         {
427             if (car[0] != '\r'){
428                 buffer_TLDR_Rx[indice] = car[0];
429                 indice++;
430                 if (indice > 24){
431                     buffer_TLDR_Rx[indice] = '\0';
432                     Fin_Linea = SI;
433                     return;
434                 }
435             }
436             else{
437                 buffer_TLDR_Rx[indice] = '\0';
438                 Fin_Linea = SI;
439                 return;
440             }
441         }
442         HAL_UART_Receive_IT(&huart1, (uint8_t*) Rx_data, 1); //activate UART receive for next char received
443     }
444 }
```

```
446 void USART1_PutString (char Tx_data[94])
447 {
448     if(HAL_UART_Transmit_IT(&huart1, (uint8_t*) Tx_data, (uint16_t) strlen(Tx_data))!= HAL_OK)
449   {
450     Error_Handler3();
451   }
452 }
```

```
454 void USART1_Put_char(char ch1[1])
455 {
456     //HAL_UART_Transmit_IT(&huart1, (uint8_t*) ch1, 1);
457     if(HAL_UART_Transmit_IT(&huart1, (uint8_t*) ch1, 1)!= HAL_OK)
458   {
459     Error_Handler3();
460   }
461 
462 }
```

```
467 void Print_Usart2(uint8_t *texto)
468 {
469         if(HAL_UART_Transmit(&huart2, texto, (uint16_t) strlen(texto), 100)!= HAL_OK)
470         {
471             Error_Handler2();
472         }
473         return;
474 }
```

```
476 void USART2_Put_char(char ch2[1])
477 {
478     HAL_UART_Transmit(&huart2, (uint8_t*) ch2, 1, 1000);
479 }
```

```
481 void Error_Handler2(void)
482 {
483   while(1)
484   {
485     HAL_Delay (200);
486   }
487 }
488
489 void Error_Handler3(void)
490 {
491   while(1)
492   {
493     HAL_Delay (200);
494   }
495 }
```

Configuración periféricos (USART.c y GPIO.c)

```c
void MX_USART1_UART_Init(void)
{

  /* USER CODE BEGIN USART1_Init 0 */

  /* USER CODE END USART1_Init 0 */

  /* USER CODE BEGIN USART1_Init 1 */

  /* USER CODE END USART1_Init 1 */
  huart1.Instance = USART1;
  huart1.Init.BaudRate = 9600;
  huart1.Init.WordLength = UART_WORDLENGTH_8B;
  huart1.Init.StopBits = UART_STOPBITS_1;
  huart1.Init.Parity = UART_PARITY_NONE;
  huart1.Init.Mode = UART_MODE_TX_RX;
  huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart1.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART1_Init 2 */

  /* USER CODE END USART1_Init 2 */

}

/* USART2 init function */

void MX_USART2_UART_Init(void)
{

  /* USER CODE BEGIN USART2_Init 0 */

  /* USER CODE END USART2_Init 0 */

  /* USER CODE BEGIN USART2_Init 1 */

  /* USER CODE END USART2_Init 1 */
  huart2.Instance = USART2;
  huart2.Init.BaudRate = 115200;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART2_Init 2 */

  /* USER CODE END USART2_Init 2 */

}
```

```
42 void MX_GPIO_Init(void)
43 {
44
45   GPIO_InitTypeDef GPIO_InitStruct = {0};
46
47   /* GPIO Ports Clock Enable */
48   __HAL_RCC_GPIOC_CLK_ENABLE();
49   __HAL_RCC_GPIOH_CLK_ENABLE();
50   __HAL_RCC_GPIOA_CLK_ENABLE();
51   __HAL_RCC_GPIOB_CLK_ENABLE();
52
53   /*Configure GPIO pin Output Level */
54   HAL_GPIO_WritePin(GPIOC, STEP_PULSE_EJE_Z_Pin|DIR_EJE_Z_Pin|Rele_Pin|LD1_Pin, GPIO_PIN_RESET);
55
56   /*Configure GPIO pin Output Level */
57   HAL_GPIO_WritePin(GPIOA, LD2_Pin|ENABLE_MOTORS_Pin, GPIO_PIN_RESET);
58
59   /*Configure GPIO pin Output Level */
60   HAL_GPIO_WritePin(GPIOB, DIR_EJE_X_Pin|STEP_PULSE_EJE_X_Pin, GPIO_PIN_RESET);
61
62   /*Configure GPIO pin : PtPin */
63   GPIO_InitStruct.Pin = B1_Pin;
64   GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
65   GPIO_InitStruct.Pull = GPIO_NOPULL;
66   HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);
67
68   /*Configure GPIO pins : PCPin PCPin PCPin PCPin */
69   GPIO_InitStruct.Pin = STEP_PULSE_EJE_Z_Pin|DIR_EJE_Z_Pin|Rele_Pin|LD1_Pin;
70   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
71   GPIO_InitStruct.Pull = GPIO_NOPULL;
72   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
73   HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
74
75   /*Configure GPIO pins : PCPin PCPin PCPin PCPin */
76   GPIO_InitStruct.Pin = FC_X1_Pin|FC_X2_Pin|FC_Z1_Pin|FC_Z2_Pin;
77   GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
78   GPIO_InitStruct.Pull = GPIO_NOPULL;
79   HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
81    /*Configure GPIO pins : PAPin PAPin */
82    GPIO_InitStruct.Pin = LD2_Pin|ENABLE_MOTORS_Pin;
83    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
84    GPIO_InitStruct.Pull = GPIO_NOPULL;
85    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
86    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
87
88    /*Configure GPIO pins : PBPin PBPin */
89    GPIO_InitStruct.Pin = DIR_EJE_X_Pin|STEP_PULSE_EJE_X_Pin;
90    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
91    GPIO_InitStruct.Pull = GPIO_NOPULL;
92    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
93    HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
94
95    /* EXTI interrupt init*/
96    HAL_NVIC_SetPriority(EXTI15_10_IRQn, 0, 0);
97    HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);
98
99 }
```

Configuración motores (STEPPER.c)

```
56  void STEPPER_Move_Blocking_X(uint8_t au8_STEPPER_Instance, uint32_t au32_Steps, uint8_t au8_DIR)
57  {
58      uint32_t i = 0;;
59
60      if(au8_DIR == DIR_CCW){
61          HAL_GPIO_WritePin(DIR_EJE_X_GPIO_Port, DIR_EJE_X_Pin, GPIO_PIN_SET);
62      }
63      else{
64          HAL_GPIO_WritePin(DIR_EJE_X_GPIO_Port, DIR_EJE_X_Pin, GPIO_PIN_RESET);
65      }
66
67          for(i=0; i<au32_Steps; i++)
68          {
69              if(HAL_GPIO_ReadPin(FC_X2_GPIO_Port, FC_X2_Pin) == 1){
70                  HAL_GPIO_WritePin(STEP_PULSE_EJE_X_GPIO_Port, STEP_PULSE_EJE_X_Pin, GPIO_PIN_SET);
71                  HAL_Delay(11);
72                  HAL_GPIO_WritePin(STEP_PULSE_EJE_X_GPIO_Port, STEP_PULSE_EJE_X_Pin, GPIO_PIN_RESET);
73                  HAL_Delay(11);
74              }
75          }
76  }
79  void STEPPER_MoveSlow_Blocking_X(uint8_t au8_STEPPER_Instance, uint32_t au32_Steps, uint8_t au8_DIR)
80  {
81      uint32_t i = 0;;
82
83      if(au8_DIR == DIR_CCW){
84          HAL_GPIO_WritePin(DIR_EJE_X_GPIO_Port, DIR_EJE_X_Pin, GPIO_PIN_SET);
85      }
86      else{
87          HAL_GPIO_WritePin(DIR_EJE_X_GPIO_Port, DIR_EJE_X_Pin, GPIO_PIN_RESET);
88      }
89
90          for(i=0; i<au32_Steps; i++)
91          {
92              if(HAL_GPIO_ReadPin(FC_X2_GPIO_Port, FC_X2_Pin) == 1){
93                  HAL_GPIO_WritePin(STEP_PULSE_EJE_X_GPIO_Port, STEP_PULSE_EJE_X_Pin, GPIO_PIN_SET);
94                  HAL_Delay(18);
95                  HAL_GPIO_WritePin(STEP_PULSE_EJE_X_GPIO_Port, STEP_PULSE_EJE_X_Pin, GPIO_PIN_RESET);
96                  HAL_Delay(18);
97              }
98          }
99  }
102 void STEPPER_Move_Blocking_Z(uint8_t au8_STEPPER_Instance, uint32_t au32_Steps, uint8_t au8_DIR)
103 {
104     uint32_t i = 0;
105
106     if(au8_DIR == DIR_CCW){
107         HAL_GPIO_WritePin(DIR_EJE_Z_GPIO_Port, DIR_EJE_Z_Pin, GPIO_PIN_SET);
108     }
109     else{
110         HAL_GPIO_WritePin(DIR_EJE_Z_GPIO_Port, DIR_EJE_Z_Pin, GPIO_PIN_RESET);
111     }
112
113         for(i=0; i<au32_Steps; i++)
114         {
115             if(HAL_GPIO_ReadPin(FC_Z2_GPIO_Port, FC_Z2_Pin) == 1){
116                 HAL_GPIO_WritePin(STEP_PULSE_EJE_Z_GPIO_Port, STEP_PULSE_EJE_Z_Pin, GPIO_PIN_SET);
117                 HAL_Delay(11);
118                 HAL_GPIO_WritePin(STEP_PULSE_EJE_Z_GPIO_Port, STEP_PULSE_EJE_Z_Pin, GPIO_PIN_RESET);
119                 HAL_Delay(11);
120             }
121         }
122 }
```

```
125 void STEPPER_MoveSlow_Blocking_Z(uint8_t au8_STEPPER_Instance, uint32_t au32_Steps, uint8_t au8_DIR)
126 {
127     uint32_t i = 0;
128 
129     if(au8_DIR == DIR_CCW){
130         HAL_GPIO_WritePin(DIR_EJE_Z_GPIO_Port, DIR_EJE_Z_Pin, GPIO_PIN_SET);
131     }
132     else{
133         HAL_GPIO_WritePin(DIR_EJE_Z_GPIO_Port, DIR_EJE_Z_Pin, GPIO_PIN_RESET);
134     }
135 
136         for(i=0; i<au32_Steps; i++)
137         {
138             if(HAL_GPIO_ReadPin(FC_Z2_GPIO_Port, FC_Z2_Pin) == 1){
139                 HAL_GPIO_WritePin(STEP_PULSE_EJE_Z_GPIO_Port, STEP_PULSE_EJE_Z_Pin, GPIO_PIN_SET);
140                 HAL_Delay(18);
141                 HAL_GPIO_WritePin(STEP_PULSE_EJE_Z_GPIO_Port, STEP_PULSE_EJE_Z_Pin, GPIO_PIN_RESET);
142                 HAL_Delay(18);
143             }
144         }
145 }
146 
```

ANEXO B: DOCUMENTACIÓN TÉCNICA

Fuente de alimentación BM PS01

Especificaciones Opiniones

B-Move BM-PS01. Potencia total: 500 W, Voltaje de entrada AC: 230 V, Corriente máxima de salida (+3.3V): 34 A. Alimentador de energía para tarjeta madre: 20+4 pin ATX, Longitud del cable de alimentación de la placa base: 40 cm. Utilizar con: PC, Factor de forma de fuente de alimentación (PSU): ATX. Color del producto: Negro, Tipo de enfriamiento: Activo, Diámetro de ventilador: 12 cm. Ancho: 140 mm, Profundidad: 150 mm, Altura: 86 mm

Control de energía	
Potencia total	**500 W**
Voltaje de entrada AC	**230 V**
Corriente máxima de salida (+3.3V)	**34 A**
Corriente máxima de salida (+12 V1)	**16 A**
Corriente máxima de salida (+12 V2)	**17 A**
Corriente máxima de salida (+5V)	**40 A**
Corriente máxima de salida (-12V)	**0,8 A**
Corriente máxima de salida (+5Vsb)	**2,5 A**

Puertos e Interfaces	
Alimentador de energía para tarjeta madre	**20+4 pin ATX**
Longitud del cable de alimentación de la placa base	**40 cm**
Número de conectores de energía SATA	**2**
Conectores de poder (4 pin) periferales (Molex)	**1**
Conector a unidad de disquetes	**Si**

Desempeño	
Utilizar con	**PC**
Factor de forma de fuente de alimentación (PSU)	**ATX**

Diseño	
Color del producto	**Negro**
Tipo de enfriamiento	**Activo**
Diámetro de ventilador	**12 cm**
Número de ventiladores	**1 Ventilador(es)**

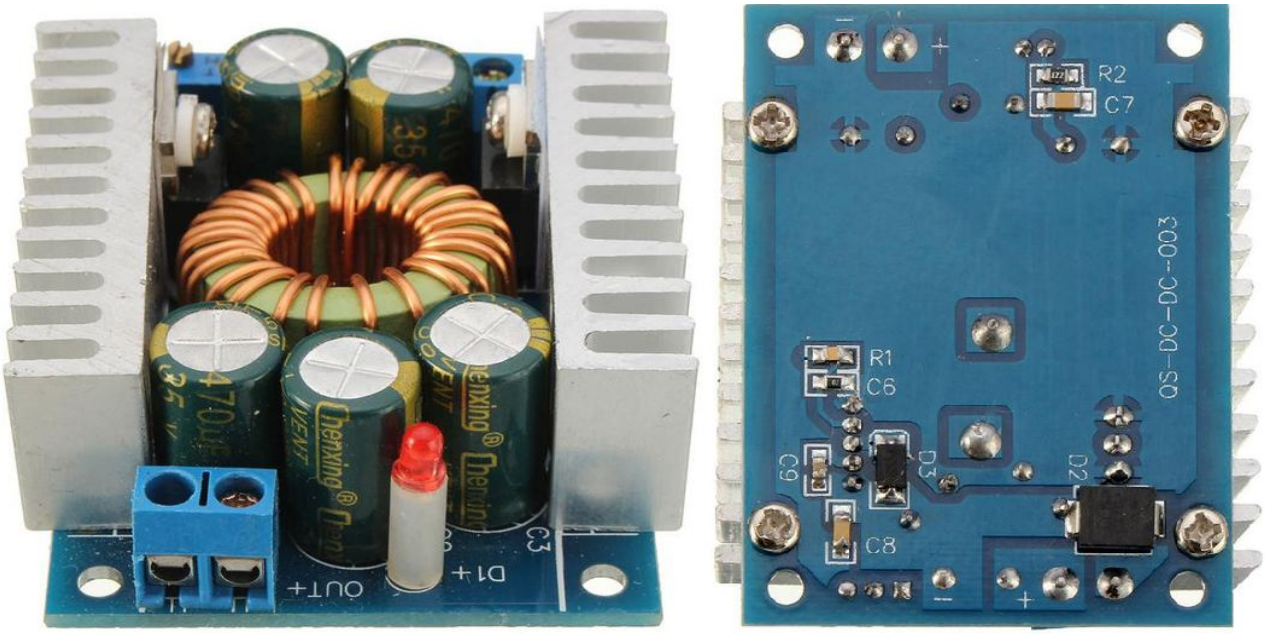
Peso y dimensiones	
Ancho	**140 mm**
Profundidad	**150 mm**
Altura	**86 mm**
Peso	**1,04 kg**

Las imágenes proporcionadas por el proveedor son únicamente una referencia de como es el producto, pudiendo haber diferencias de estas con las del producto final.

Comprar B-Move BM-PS01 unidad de fuente de alimentación 500 W 20+4 pin ATX ATX Negro

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

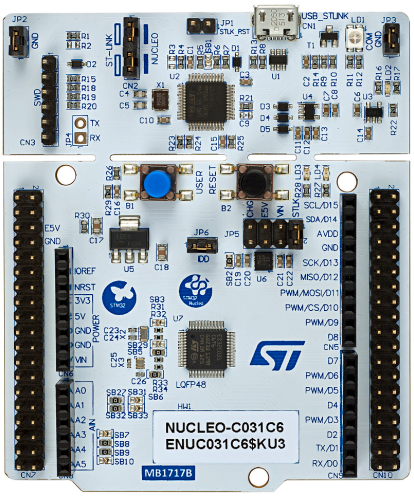
Module adjustment:

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.

NUCLEO-XXXXCX NUCLEO-XXXXRX
NUCLEO-XXXXRX-P

Data brief

STM32 Nucleo-64 boards



NUCLEO-C031C6 example. Boards with different references show different layouts. Picture is not contractual.

Product status link
NUCLEO-XXXXCX
NUCLEO-C031C6
NUCLEO-XXXXRX
NUCLEO-F030R8, NUCLEO-F070RB, NUCLEO-F072RB, NUCLEO-F091RC, NUCLEO-F103RB, NUCLEO-F302R8, NUCLEO-F303RE, NUCLEO-F334R8, NUCLEO-F401RE, NUCLEO-F410RB, NUCLEO-F411RE, NUCLEO-F446RE, NUCLEO-G070RB, NUCLEO-G071RB, NUCLEO-G0B1RE, NUCLEO-G431RB, NUCLEO-G474RE, NUCLEO-G491RE, NUCLEO-L010RB, NUCLEO-L053R8, NUCLEO-L073RZ, NUCLEO-L152RE, NUCLEO-L452RE, NUCLEO-L476RG
NUCLEO-XXXXRX-P
NUCLEO-L412RB-P, NUCLEO-L433RC-P, NUCLEO-L452RE-P

Features

- Common features
  - STM32 microcontroller in LQFP64 or LQFP48 package
  - 1 user LED shared with ARDUINO®
  - 1 user and 1 reset push-buttons
  - 32.768 kHz crystal oscillator
  - Board connectors:
    - ARDUINO® Uno V3 expansion connector
    - ST morpho extension pin headers for full access to all STM32 I/Os
  - Flexible power-supply options: ST-LINK USB V_{BUS} or external sources
  - On-board ST-LINK debugger/programmer with USB re-enumeration capability: mass storage, Virtual COM port, and debug port
  - Comprehensive free software libraries and examples available with the STM32Cube MCU Package
  - Support of a wide choice of Integrated Development Environments (IDEs) including IAR Embedded Workbench®, MDK-ARM, and STM32CubeIDE
- Board-specific features
  - External SMPS to generate V_{core} logic supply
  - 24 MHz or 48 MHz HSE
  - Board connectors:
    - External SMPS experimentation dedicated connector
    - Micro-B or Mini-B USB connector for the ST-LINK
    - MIPI® debug connector

Description

The STM32 Nucleo-64 board provides an affordable and flexible way for users to try out new concepts and build prototypes by choosing from the various combinations of performance and power consumption features, provided by the STM32 microcontroller. For the compatible boards, the external SMPS significantly reduces power consumption in Run mode.

The ARDUINO® Uno V3 connectivity support and the ST morpho headers allow the easy expansion of the functionality of the STM32 Nucleo open development platform with a wide choice of specialized shields.

The STM32 Nucleo-64 board does not require any separate probe as it integrates the ST-LINK debugger/programmer.

The STM32 Nucleo-64 board comes with the STM32 comprehensive free software libraries and examples available with the STM32Cube MCU Package.

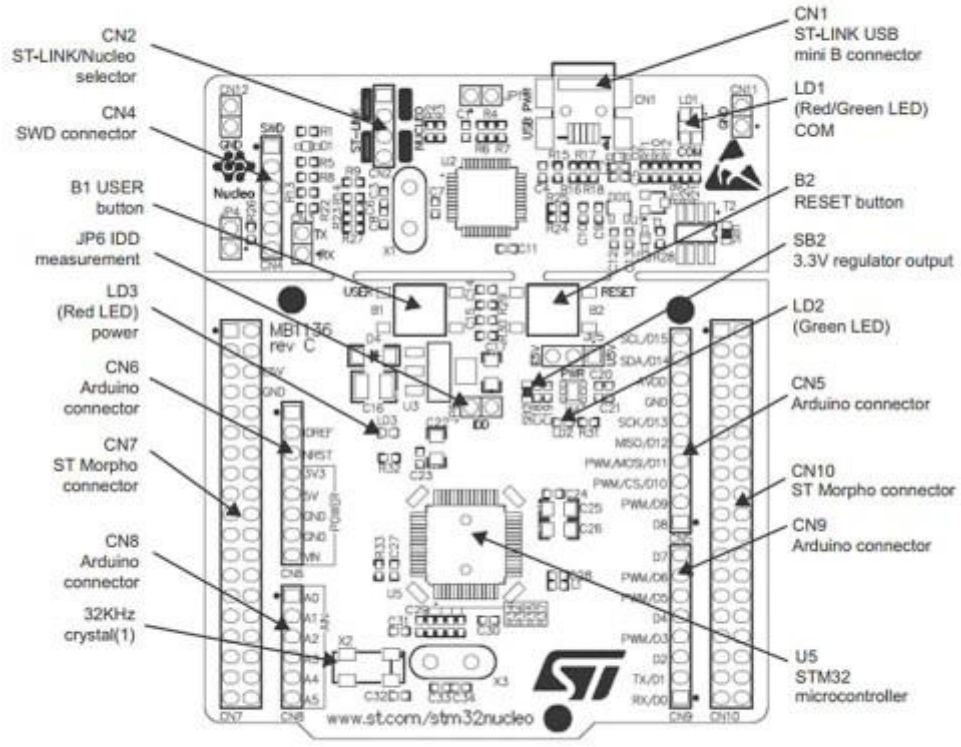
1 Ordering information

To order an STM32 Nucleo-64 board, refer to Table 1. For a detailed description of each board, refer to its user manual on the product web page. Additional information is available from the datasheet and reference manual of the target STM32.

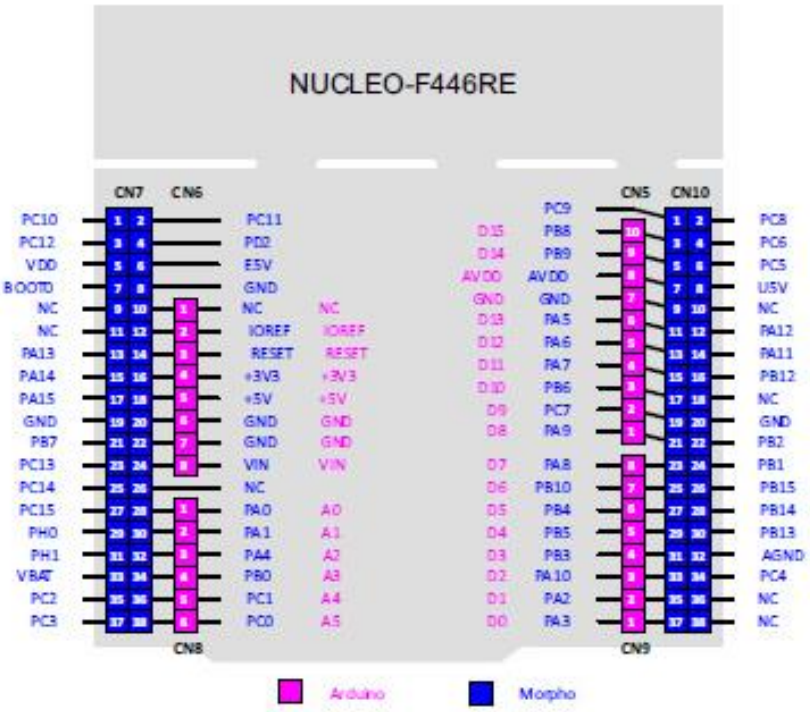
Table 1. List of available products

Order code	Board reference	User manual	Target STM32	Differentiating features
NUCLEO-C031C6	MB1717	UM2953	STM32C031C6T6	• ST-LINK/V2-1 on Micro-B USB connector • 48 MHz HSE • LQFP48
NUCLEO-F030R8	MB1136	UM1724	STM32F030R8T6	• ST-LINK/V2-1 on Mini-B USB connector • LQFP64
NUCLEO-F070RB			STM32F070RBT6	• ST-LINK/V2-1 on Mini-B USB connector • LQFP64
NUCLEO-F072RB			STM32F072RBT6	• ST-LINK/V2-1 on Mini-B USB connector • LQFP64
NUCLEO-F091RC			STM32F091RCT6U	• ST-LINK/V2-1 on Mini-B USB connector • LQFP64
NUCLEO-F103RB			STM32F103RBT6	• ST-LINK/V2-1 on Mini-B USB connector • LQFP64
NUCLEO-F302R8			STM32F302R8T6	• ST-LINK/V2-1 on Mini-B USB connector • LQFP64
NUCLEO-F303RE			STM32F303RET6	• ST-LINK/V2-1 on Mini-B USB connector • LQFP64
NUCLEO-F334R8			STM32F334R8T6	• ST-LINK/V2-1 on Mini-B USB connector • LQFP64
NUCLEO-F401RE			STM32F401RET6U	• ST-LINK/V2-1 on Mini-B USB connector • LQFP64
NUCLEO-F410RB			STM32F410RBT6U	• ST-LINK/V2-1 on Mini-B USB connector • LQFP64
NUCLEO-F411RE			STM32F411RET6U	• ST-LINK/V2-1 on Mini-B USB connector • LQFP64
NUCLEO-F446RE			STM32F446RET6U	• ST-LINK/V2-1 on Mini-B USB connector • LQFP64
NUCLEO-G070RB	MB1360	UM2324	STM32G070RBT6	• ST-LINK/V2-1 on Micro-B USB connector • LQFP64

Disposición y configuración del hardware



Disposición y configuración del hardware



CNC Shield

DESCRIPCIÓN

INFO

El Shield CNC GRBL v3.0 permite construir una máquina CNC de la manera más rápida y sencilla, solo necesitas agregar un Arduino Uno, unos cuantos Drivers pap A4988 o DRV8825 y una fuente de alimentación. Posee un diseño modular y Open Source. Compatible con GRBL que es un firmware OpenSource para Arduino que convierte código-G en comandos para motores Paso a Paso. Ideal para desarrollar proyectos como Router CNC, Cortadora Láser, Brazo robótico y hasta una Máquina Pick&Place.

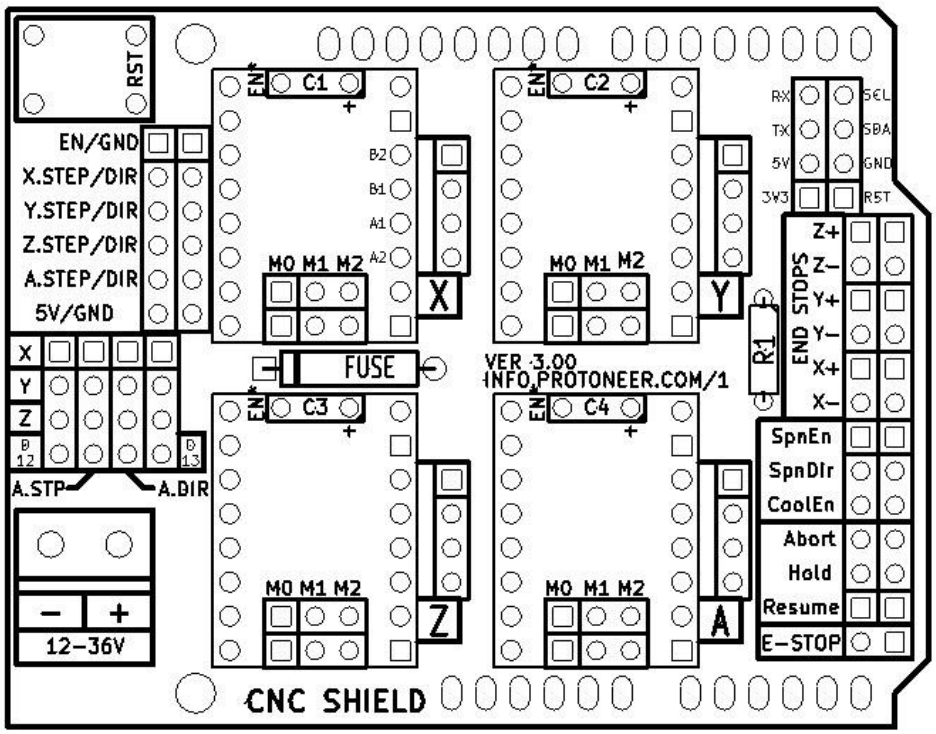
El shield CNC permite manejar 3 motores paso a paso (pap/stepper) de forma independiente (X, Y, Z) y 1 motor adicional (A) como duplicado de alguno de los anteriores. Es compatible con los drivers para motores paso a paso Pololu A4988 (Allegro) o los DRV8825 (Texas Inst.), el driver A4988 puede manejar motores paso a paso de hasta 2A por bobina y microstepping de 1/16, el driver DRV8825 es más versátil pues ofrece hasta 2.5A por bobina y microstepping de hasta 1/32. Podemos configurar de forma independiente la resolución de microstepping de cada driver con los 3 jumpers correspondientes.

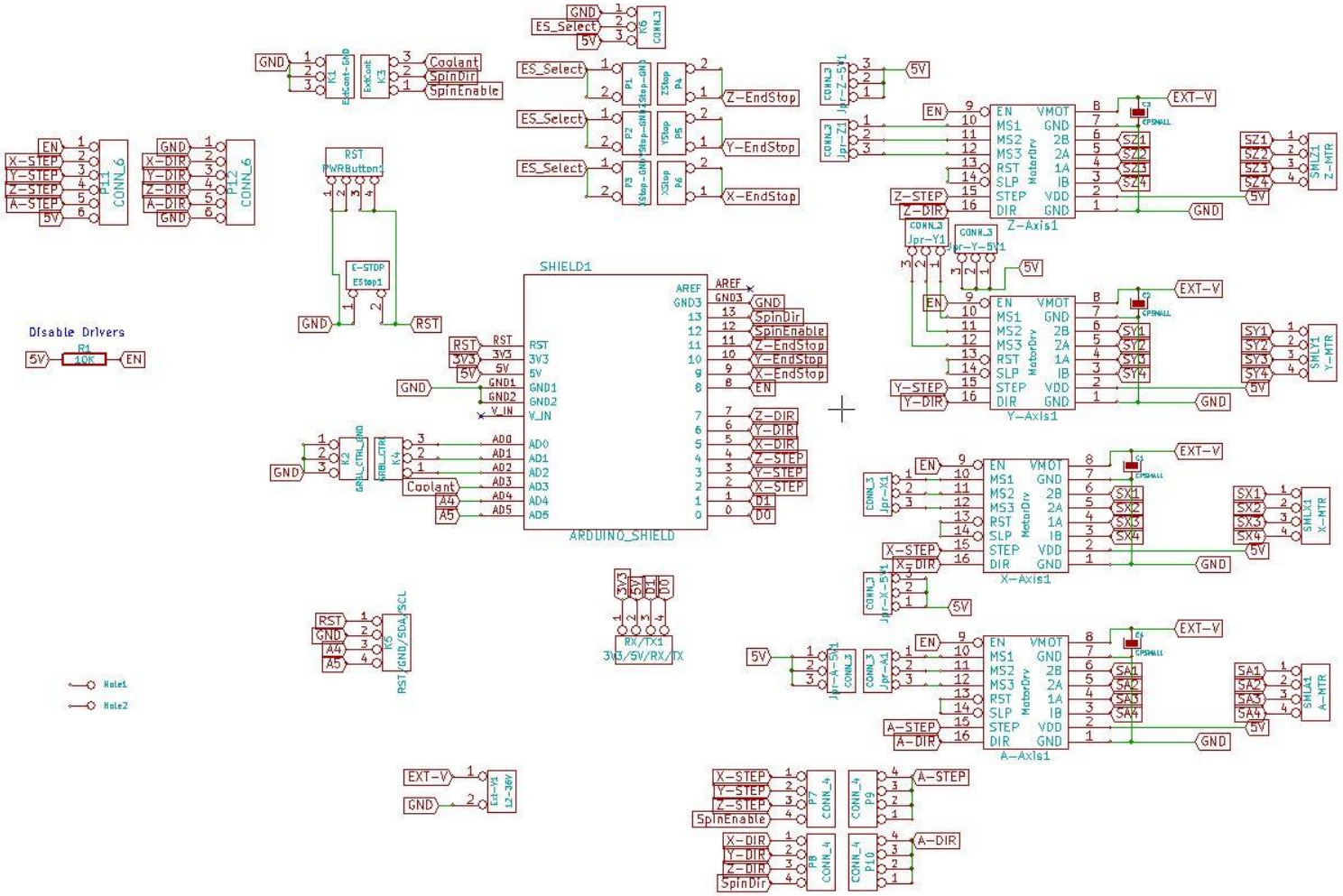
Para energizar el shield podemos utilizar una fuente de alimentación DC entre 12 a 36 voltios. La capacidad de corriente de la fuente debe ser de aprox. 2 amperios por cada motor, entonces si utilizamos 4 motores necesitaremos una fuente de 8 amperios. Recomendamos utilizar la fuente de alimentación 12V/8.5A o 24V/5A. El voltaje de potencia no está conectado al pin de alimentación "Vin" del Arduino Uno, por lo que tendremos que utilizar una fuente separada para el Arduino o realizar un puente entre el voltaje de potencia y el pin "Vin" del Arduino Uno (máx. 12V)

El shield CNC es compatible con los siguientes modelos de Arduino: Arduino Uno R3 y Arduino Leonardo. Si deseamos utilizar otro microcontrolador u Arduino (Mega, Nano, etc), su uso debe ser como un modulo cableado y no montado directamente sobre el Arduino como un shield , esto debido a la disposición de pines utilizado en su diseño.

ESPECIFICACIONES TÉCNICAS

- Voltaje de alimentación (potencia): 12V-36V DC
- Shield compatible con Arduino Uno R3 y Arduino Leonardo
- Compatible con firmware GRBL
- Soporta hasta 4 ejes independientes (X, Y, Z y duplicar uno de los anteriores o crear un eje a medida con los pines D12 y D13)
- Conexión para 2 finales de carrera por cada eje (6 en total)
- Habilitador y dirección de Spindle
- Habilitador de refrigerante (coolant)
- Diseñado para drivers Pololu A4988 o DRV8825
- Jumpers para configurar el micro-stepping de los drivers
- Los motores se pueden conectar usando header o Molex hembra de 4 pines
- Fusible en placa




SHIELD1
ARDUINO_SHIELD
Disable Drivers
R1
10K
PWRButton1
E-STOP
EStop1
Coolant
SpinDir
SpinEnable
Z-EndStop
Y-EndStop
X-EndStop
Z-Axis1
Y-Axis1
X-Axis1
A-Axis1
EXT-V
Hole1
Hole2

DRV8825

DESCRIPCIÓN

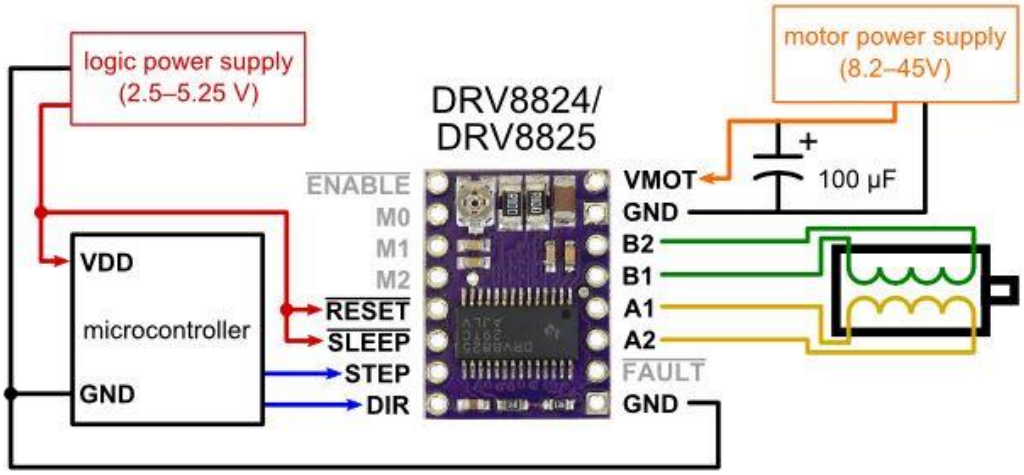
INFO

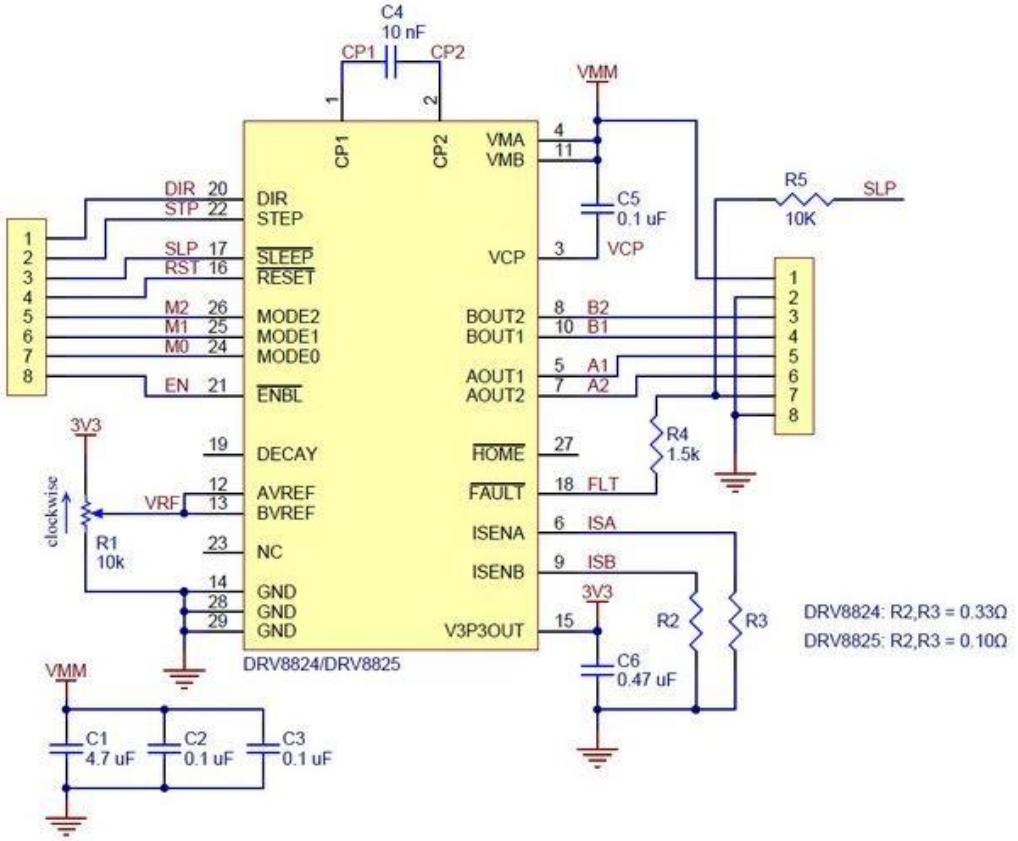
El driver DRV8825 permite controlar motores motores paso a paso bipolares de hasta 2.5A. Basado en el chip DRV88225 de Texas Instruments, es una mejora o evolución comparado al chip A4988. Ampliamente utilizado con placas de control de impresoras 3D y Máquinas CNC Open Source, como: RAMPS y CNC Shield. El Driver DRV8825 es pin-compatible con el Driver A4988, lo que significa que puede usarse como reemplazo directo de mejor rendimiento.

Permite regular la corriente máxima de salida por medio de un potenciómetro. Además posee seis resoluciones diferentes de microstepping (máx. 1/32). Trabaja con voltajes de alimentación entre 8.2V a 45V, puede suministrar 1.5A por bobina sin usar ventilación forzada o un disipador y soporta picos de corriente de hasta 2.5A. Para manejar el driver solo son necesarios 2 pines, uno para la dirección de giro (DIR) y otro para dar el paso (STEP). El pin Enable debe estar conectado a Tierra (GND) para que el motor funcione. El microstepping se configura con los pines MS1, MS2 y MS3 de acuerdo a la tabla del fabricante.

ESPECIFICACIONES TÉCNICAS

- Voltaje de alimentación-potencia(VMOT): 8.2V-45V DC (recomendado 12V/24VDC)
- Voltaje de control lógico: 3.3V-5V DC
- Corriente de salida: 1.5A por bobina (máx. 2.5A con ventilación)
- Interfaz de control de STEP y DIRECTION
- 6 resoluciones de pasos: full-step, half-step, 1/4, 1/8, 1/16 y 1/32
- Corriente máxima regulable por potenciómetro, para poder usar voltajes mas altos y lograr mejor resolución
- Regulador incluido
- Funciona con sistemas de 3.3 y 5V
- Protección de sobre temperatura, sobrecorriente y voltaje bajo
- Protección de corto a tierra y corto de carga
- PCB de 4 capas
- Pin-compatible con el Driver A4988


logic power supply
(2.5–5.25 V)
motor power supply
(8.2–45V)
DRV8824/
DRV8825
ENABLE
M0
M1
M2
RESET
SLEEP
STEP
DIR
VDD
microcontroller
GND
VMOT
GND
B2
B1
A1
A2
FAULT
GND
+
100 µF


C4
10 nF
CP1
CP2
VMM
DIR 20
STP 22
SLP 17
RST 16
M2 26
M1 25
M0 24
EN 21
DIR
STEP
SLEEP
RESET
MODE2
MODE1
MODE0
ENBL
VMA
VMB
VCP
BOUT2
BOUT1
AOUT1
AOUT2
HOME
FAULT
ISENA
ISENB
V3P3OUT
GND
NC
DECAY
AVREF
BVREF
C5
0.1 uF
R5
10K
SLP
B2
B1
A1
A2
R4
1.5k
FLT
ISA
ISB
3V3
R2
R3
C6
0.47 uF
DRV8824: R2,R3 = 0.33Ω
DRV8825: R2,R3 = 0.10Ω
clockwise
VRF
R1
10k
C1
4.7 uF
C2
0.1 uF
C3
0.1 uF
DRV8824/DRV8825

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

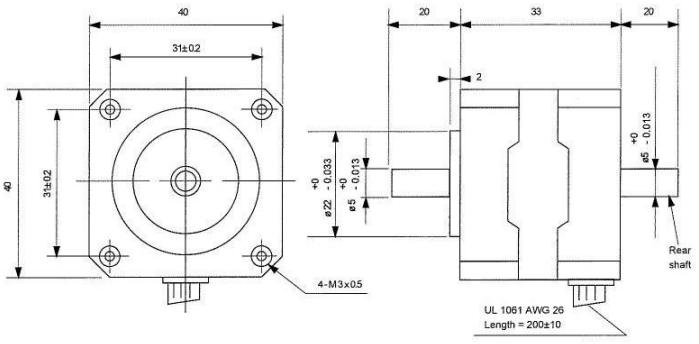
These 4 phase hybrid stepper motors are capable of delivering much higher working torques and stepping rates than permanent magnet (7.5° and 15°) types. Whilst at the same time maintaining a high detent torque even when not energised. This feature is particularly important for positional integrity. Many 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.

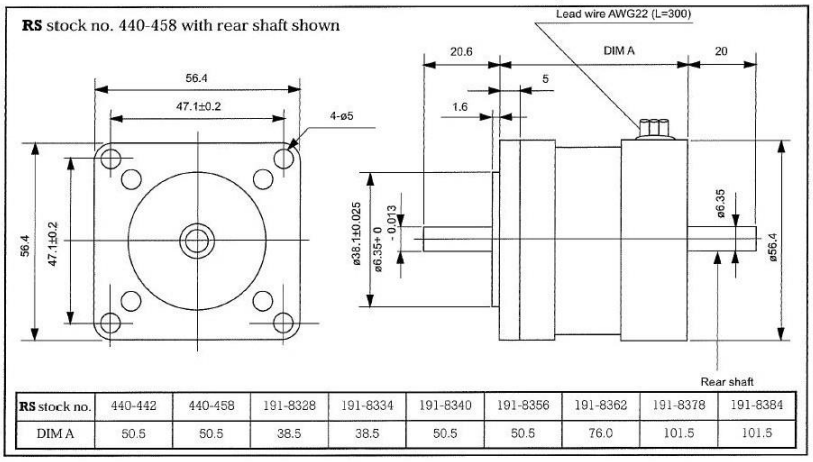
Rear extension shafts are provided on three of the motors to enable connection of other drive requirements and feedback devices.

Size 17

RS stock no. 440-436 with rear shaft shown

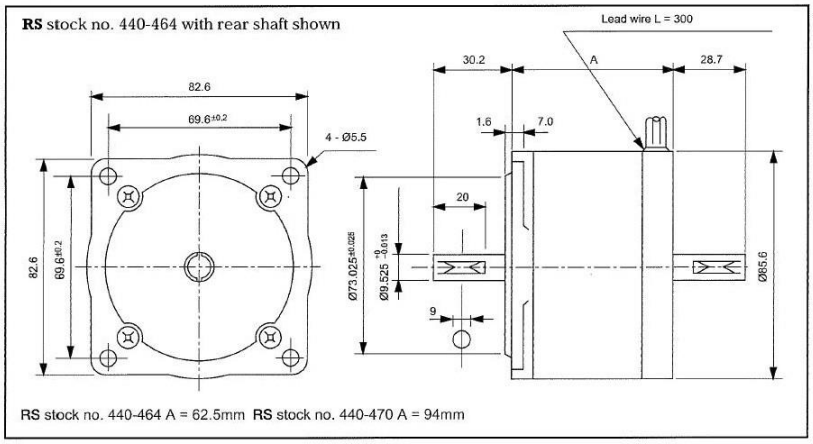


ENGLISH

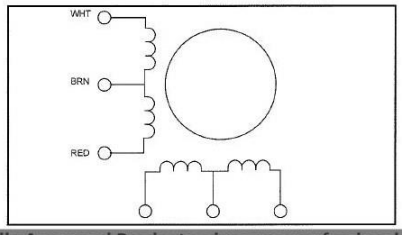


RS stock no.	440-442	440-458	191-8328	191-8334	191-8340	191-8356	191-8362	191-8378	191-8384
DIM A	50.5	50.5	38.5	38.5	50.5	50.5	76.0	101.5	101.5

Size 34

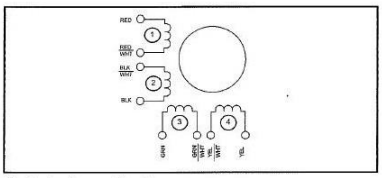


6 Wire configuration



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		

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

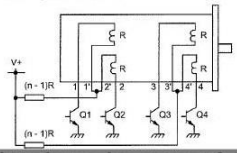
Resonance

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.

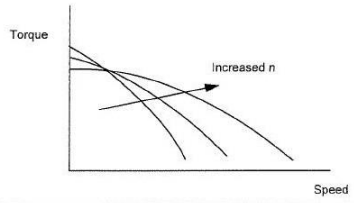
Figure 1 **Unipolar drive**



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.

ENGLISH

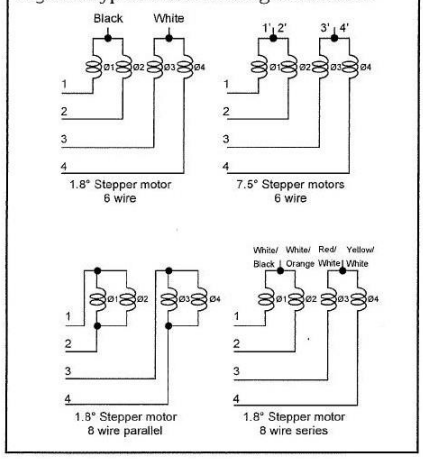
Figure 2 Effect on motor performance of higher supply voltages and larger series limiting resistance



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) → step 1
Above sequence repeating → step 5
Anti-clockwise ↑
Clockwise ↓

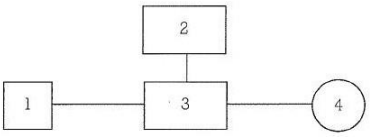
Table 2 Half step mode

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

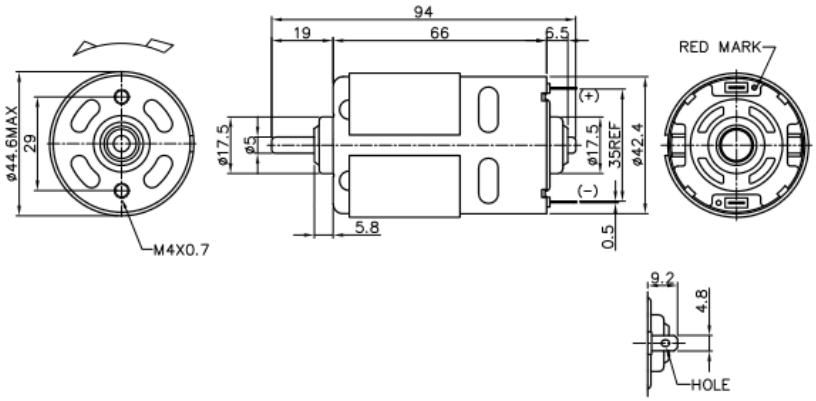
Typical stepper motor control system

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**.

775 series Ø45 mm 103-198 W



Model - □ - □
Cabron brush = C
Metal brush = M
V = Varistor
C = Capacitor

MOTOR DATA				
Part name	775-9008F-CC	775-9009F-C-CC	775-8013F-C-CC	775-5520F-CC
Diameter (mm)	45	45	45	45
Lenth (mm)	66	66	66	66
Nominal voltage (V)	7.2	12	18	24
Nominal speed (rpm)	12300	18000	18700	18400
Nominal torque (mNm)	80.0	102.6	100.8	94.3
Nominal current A	20.1	21.1	15.8	10.7
No load speed (rpm)	14600	21000	22000	21000
No load current A	3.20	2.80	3.00	1.70
Stall torque (mNm)	508.8	806.4	837.3	705.9
Starting current (A)	108.7	143.7	110.4	68.1
Output (W)	103	194	198	182
Efficiency (%)	72	77	70	71
Operating temperature deg. C	-10..+60	-10..+60	-10..+60	-10..+60

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.)

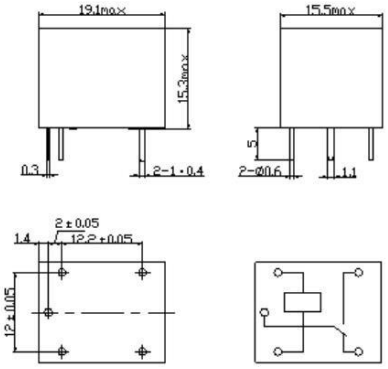
3. ORDERING INFORMATION

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

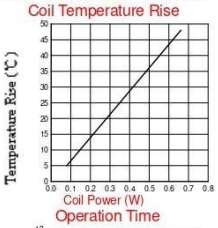
Item \ Type	SRD	
	FORM C	FORM A
Contact Capacity Resistive Load (cosΦ=1)	7A 28VDC 10A 125VAC 7A 240VAC	10A 28VDC 10A 240VAC
Inductive Load (cosΦ=0.4 L/R=7msec)	3A 120VAC 3A 28VDC	5A 120VAC 5A 28VDC
Max. Allowable Voltage	250VAC/110VDC	250VAC/110VDC
Max. Allowable Power Force	800VAC/240W	1200VA/300W
Contact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

8. PERFORMANCE (at initial value)

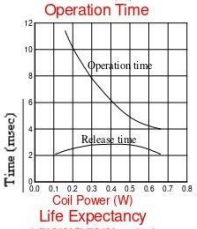
Item \ Type	SRD
Contact Resistance	100mΩ Max.
Operation Time	10msec Max.
Release Time	5msec Max.
Dielectric Strength Between coil & contact Between contacts	 1500VAC 50/60HZ (1 minute) 1000VAC 50/60HZ (1 minute)
Insulation Resistance	100 MΩ Min. (500VDC)
Max. ON/OFF Switching Mechanically Electrically	 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

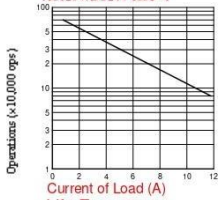


Operation Time

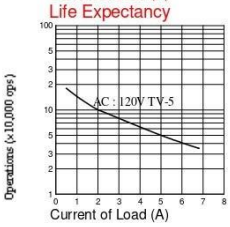


Life Expectancy

AC120V/DC24V cosΦ=1



Life Expectancy

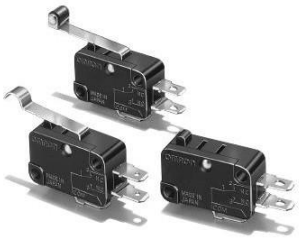


V
Miniature Basic Switch

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



V

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)

V

Miniature Basic Switch

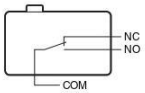
Actuator	Terminals	Contact form	Ratings: Maximum operating force (OF)	15A	10A	Heat-resistive 15A	Heat-resistive 10A
Hinge roller lever 	Solder terminals (A)	SPDT	2.45N	V-156-1A6	---	V-156-1A6-T	---
		SPST-NC		V-156-2A6	---	---	---
		SPST-NO		V-156-3A6	---	---	---
		SPDT	1.23N	V-156-1A5	V-106-1A5	V-156-1A5-T	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)

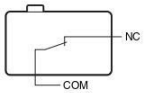
Actuator	Terminals	Contact form	Ratings: Maximum operating force (OF)	30VDC 12A
Pin plunger	Quick-connect terminals (#250) (C)	SPDT	3.92N	V-21-1C6(IN)

Contact form

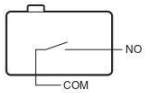
SPDT



SPST-NC



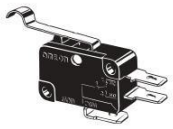
SPST-NO

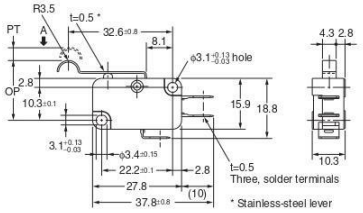


Refer to **"Micro Switch Common Accessories"** for Separators (sold separately), Actuators (sold separately) and Terminal Connectors (sold separately).

●Simulated roller lever

V-154-1□6
V-154-1□5
V-104-1□5
V-104-1□4



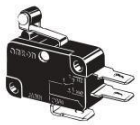


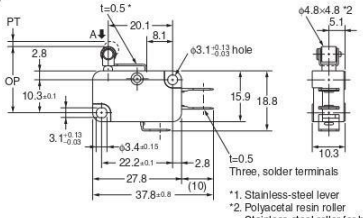
Operating characteristics	Model	V-154-1□6	V-154-1□5 V-104-1□5	V-104-1□4
OF max.		2.45N	1.23N	0.59N
RF min.		0.25N	0.14N	0.06N
PT max.		4.0mm		
OT min.		1.6mm		
MD max.		1.5mm		
OP		18.7±1.2mm		

V

●Short hinge roller lever

V-155-1□6
V-155-1□5
V-105-1□5
V-105-1□4

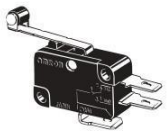


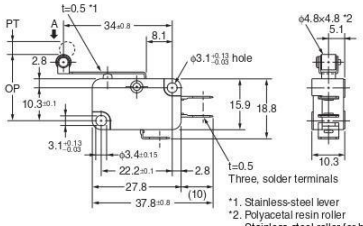


Operating characteristics	Model	V-155-1□6	V-155-1□5 V-105-1□5	V-105-1□4
OF max.		4.71N	2.35N	1.18N
RF min.		0.49N	0.49N	0.15N
PT max.		1.6mm		
OT min.		0.8mm		
MD max.		0.6mm		
OP		20.7±0.6mm		

●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

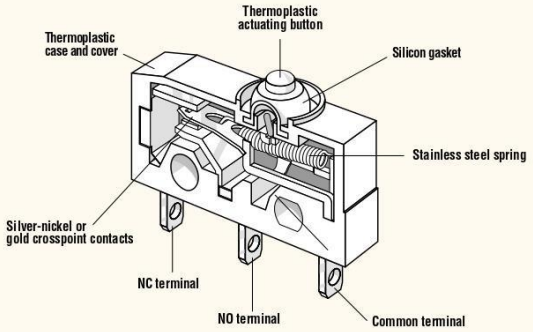
Use M3 mounting screw with plane washers or spring washers to securely mount the Switch.Tighten the screws to a torque of 0.39 to 0.59N·m {4 to 6 kgf·cm}.

SUBMINIATURE — SEALED

DC Series

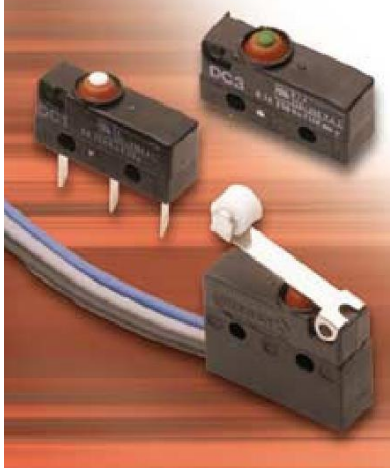
Features

- 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)	Electrical Life at Rated Load: 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

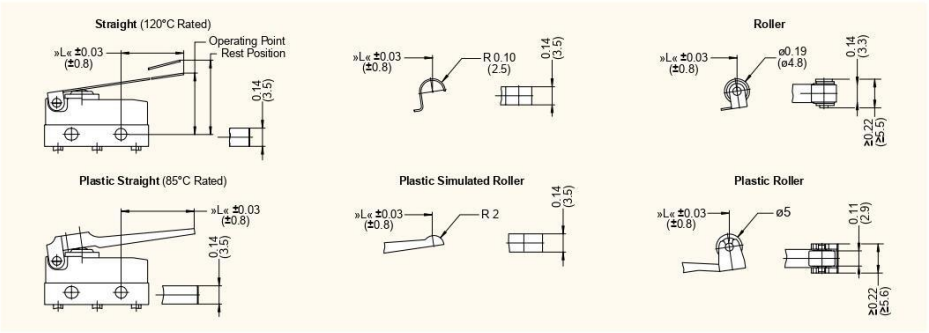
Temperature Rating:	-40° to +85°C /+120°C *(without wire leads)* -40° to +85C / +105°C *(with wire leads)*
Flammability Rating:	UL94V-O *(PET, PBT)* UL94HB *(POM)*

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

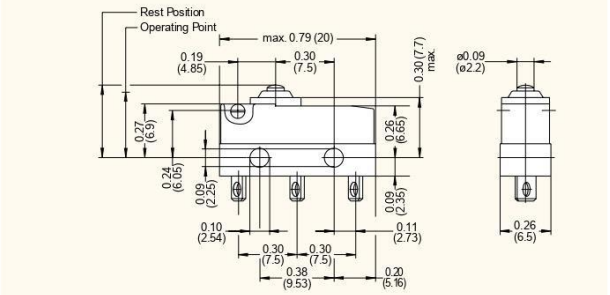
Auxiliary Actuators inches (mm)



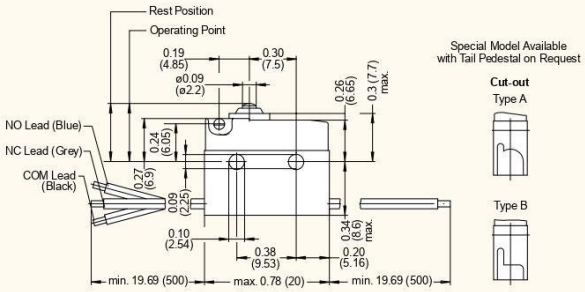
Actuator Specifications

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)

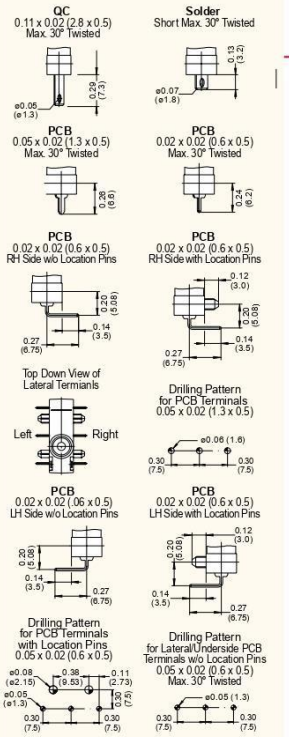
Dimensions inches (mm)



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[2] (0.5[2]) Leads *(No UL)*	5, 9
C	0.03[2] (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

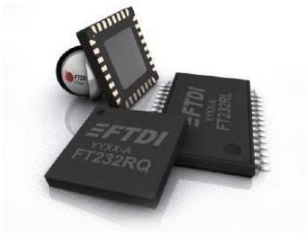
Code	Actuator Type	Actuation Length inches (mm) A	B	C	D
AA	Button only	—	—	—	—
L	Lever	—	0.189 (4.8)	0.276 (7.0)	1.654 (42.0)
R	Roller	—	0.098 (2.5)	0.185 (4.7)	1.563 (39.7)
S	Simulated Roller	—	0.098 (2.5)	0.185 (4.7)	1.563 (39.7)
V[2]	Plastic Sim. Roller	—	0.220 (5.6)	—	—
W[2]	Plastic Straight	—	0.276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

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

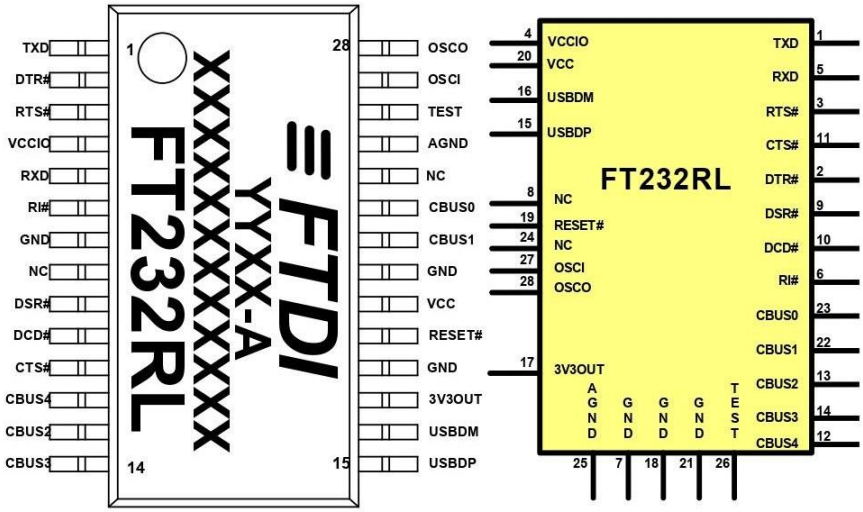


Figure 3.1 SSOP Package Pin Out and Schematic Symbol

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

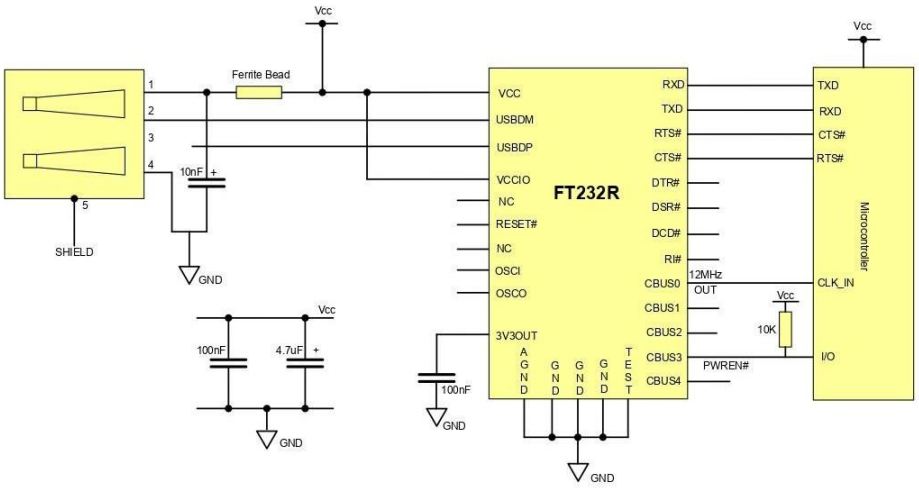


Figure 7.4 USB to MCU UART Interface

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.

ANEXO C: IMÁGENES DEL SISTEMA