Escuela Superior de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

Diseño y fabricación de robot tipo delta de bajo coste mediante impresión 3D

Septiembre 2022

Autor: Ricardo Melián Maury
Tutor: Santiago Torres Álvarez

Universidad de La Laguna

Agradecimientos

A todos los profesores que he tenido en la Universidad de La Laguna, en especial a Benjamín González Díaz, Silvia Alayón Miranda, Beatriz Rodríguez Mendoza, Leopoldo Acosta Sánchez y a mi tutor, Santiago Torres Álvarez, por haber fomentado el desarrollo de mi curiosidad y mi pasión por la ingeniería al impartir una docencia sublime y dar un trato personal incomparable aún teniendo en cuenta la situación por la que hemos pasado durante estos últimos años.

A mi familia, sobre todo a mis padres, por el apoyo incondicional que siempre me han dado. Por no soltarme nunca de la mano, empujarme a ser mejor persona cada día y creer siempre en mí más que yo mismo. Soy quien soy gracias a ustedes.

A los amigos cercanos que me ha brindado mi paso por la universidad: Miguel, Yaky, Carlos, Eladio, Luis O., Luis M., David, Gaga y Laura. Sin ellos las innumerables horas muertas, el estrés de las clases y los malos tragos no habrían sido aguantables,
sin embargo, han llegado a ser momentos entrañables.
Gracias por formar parte de esta etapa de mi vida.

Mil gracias a todos los mencionados, no ha sido fácil, pero ha valido la pena el esfuerzo.

Resumen

La industria se encuentra cada vez más automatizada. Las operaciones de montaje, posicionamiento y manipulación, entre otras, son trabajos sencillos y repetitivos que se pueden acelerar y realizar con más precisión utilizando máquinas especializadas reemplazando la labor humana.

La robótica ofrece soluciones que se ajustan a las necesidades de la industria. En particular, para las manipulaciones muy precisas, a velocidades muy altas y con una carga no muy pesada existe un robot que destaca entre el resto por su productividad, rendimiento y versatilidad, el robot Delta [1].

Este tipo de robots destaca por ofrecer una gran aceleración y velocidad, además de un extenso espacio de trabajo y precisión. Por ello, entre otras aplicaciones, es ampliamente utilizado en líneas de producción para el control de calidad del producto, utilizándose para descartar o clasificar de manera muy rápida los objetos que pasan por una cinta transportadora. Así como también son útiles en manipulaciones muy precisas como pueden ser el posicionamiento de componentes electrónicos en una placa de circuitos impresos a altas velocidades. Con su empleo se garantiza la eficiencia y productividad de un proceso mejorando la competitividad de la empresa, reduciendo el tamaño y coste de las líneas de producción [1].

En este proyecto, se realiza un estudio de su cinemática directa, cinemática inversa y espacio de trabajo en función de las dimensiones de cada uno de sus elementos. El estudio de la cinemática del robot se elaborará tanto matemáticamente como gráficamente para facilitar la comprensión del funcionamiento de este tipo de máquina, aportando pequeñas herramientas fácilmente utilizables que modelizan su comportamiento.

Adicionalmente, se procederá al diseño tanto mecánico como electrónico de un robot Delta de pequeñas dimensiones, bajo coste y fabricado mediante impresión 3D en su amplia mayoría para aplicaciones como la propia impresión 3D, el posicionamiento de componentes electrónicos en una PCB o cualquier otra operación que requiera un espacio de trabajo reducido.

Abstract

The industry is increasingly becoming automated. Assembly, positioning and handling operations, among others, are simple and repetitive tasks that can be accelerated and performed with more precision using specialized machines replacing human labour.

Robotics offers solutions according to the industry needs. In particular, for very precise manipulations, at very high speeds and with a not very heavy load, the Delta Robot stands out from the rest for its productivity, performance and versatility [1].

This type of robot stands out for offering great acceleration and speed, as well as a large workspace and precision. For this reason, among other applications, it is widely used in production lines for product quality control, being used to very quickly discard or classify objects passing through a conveyor belt. Moreover it is quite useful in very precise manipulation tasks such as the positioning of electronic components on a printed circuit board at high speeds. With its use, the efficiency and productivity of a process is guaranteed, improving the competitiveness of the company, reducing the size and cost of production lines [1].

In this project, a study of its direct kinematics, inverse kinematics and workspace is carried out based on the dimensions of its elements. The study of the kinematics of the robot will be elaborated both mathematically and graphically, in order to facilitate the understanding of the operation of this type of machine, providing small easily usable tools that model its behaviour.

Additionally, both the mechanical and electronic design of a small, low-cost Delta robot will be carried out, mostly manufactured by 3D printing for applications such as the proper 3D printing, the positioning of electronic components on a PCB or any other operation that requires a small work space.

Índice general

Índice de figuras

Capítulo 1: Introducción

En la actualidad, se utilizan muchos tipos de máquinas y robots para automatizar procesos y elevar la productividad y eficiencia de líneas de producción a la vez que se reducen costes y plantilla. Entre los robots industriales utilizados para este fin se encuentra el robot Delta. Sin embargo, fuera de su campo de aplicación, no son muy conocidos ni siquiera por personas con formación en el ámbito de la industria y los procesos de fabricación, ya que son otro tipo de robots como el robot antropomórfico, robot SCARA o robot cartesiano entre otros, los que toman más protagonismo.

En este proyecto, la principal motivación ha sido la de dar a conocer fuera de su campo de aplicación de manera tanto analítica como gráfica el funcionamiento del robot Delta. Para ello, se realiza un estudio de su cinemática directa, cinemática inversa y espacio de trabajo, aportando recursos que muestran dicho funcionamiento de manera clara y fácilmente manipulables para su comprensión e incluso simulación.

Por otra parte, se procederá al diseño tanto mecánico como electrónico de un robot de este tipo de pequeñas dimensiones, bajo coste y fabricado mediante impresión 3D en su amplia mayoría, demostrando tanto la validez del estudio previamente mencionado y el hecho de que es posible disponer de uno de ellos de manera asequible.

1.1. Objetivos

Este proyecto tiene como objetivo principal dar a conocer y promover la utilización y estudio del robot Delta, aportando las bases necesarias para comprender su funcionamiento y proporcionando recursos tanto para el diseño como para la enseñanza de las características del robot. Para cumplir este objetivo se ha optado por utilizar herramientas ampliamente disponibles y que se pueden utilizar con un coste muy reducido.

Con el objetivo de aportar estas bases e información se han realizado las siguientes tareas:

- Pequeño estudio del estado del arte con propósito de divulgación de las funciones que tienen actualmente los robots Delta, sus características y las prestaciones que son capaces de proporcionar.

- Realización de un modelo en Geogebra que demuestra la cinemática directa y permite manipular las variables de actuación, observar el movimiento que provoca esta acción en el robot y comprender visualmente las matemáticas detrás del robot.

- Realización de un modelo en Geogebra que demuestra la cinemática inversa y permite manipular el punto del efector final del robot, observar cómo se deben posicionar las variables de actuación para alcanzar dicho punto y comprender visualmente las matemáticas detrás de esta operación.

- Desarrollo de una pequeña herramienta en python que, introduciendo unas dimensiones determinadas por el usuario para cada elemento del robot, genera el espacio de trabajo que será posible alcanzar y lo representa gráficamente además de poder exportar e importar dicho espacio de trabajo en formato Excel. Además, permite resolver la cinemática inversa del manipulador, ya que permite introducir una posición para el efector final del robot y obtener las variables articulares necesarias para llegar a ella, además de permitir la representación gráfica del robot en esa posición.

- Diseño de un robot Delta de pequeño tamaño utilizando el programa de diseño asistido por computadora “Autodesk Fusion 360” teniendo siempre en cuenta la premisa de que sea imprimible en 3D fácilmente y sin necesidad de soportes.

- Fabricación del robot Delta diseñado y la adición de la electrónica necesaria para su funcionamiento como motores, placa controladora, fuente de alimentación y sensores final de carrera y de distancia.

1.2. Metodología

Para llevar a cabo las tareas mencionadas anteriormente, se han utilizado las siguientes herramientas disponibles gratuitamente para el uso personal:

- **Geogebra:** Una calculadora gráfica con capacidad 3D en la que se modelará tanto la cinemática directa como la cinemática inversa.
- **Autodesk Fusion 360:** Un programa de diseño asistido por ordenador en el que se diseñarán los distintos elementos que constituyen el robot delta que se va a fabricar.
- **Ultimaker Cura:** Un software para impresión 3D en el que se modifican los parámetros de impresión y se transforma a “GCODE” el objeto a imprimir.
- **Firmware Marlin:** Firmware que es el estándar en la industria de la impresión 3D que interpreta el “GCODE” proporcionado y realiza la operación de la impresora.
- **Lenguaje Python:** Lenguaje de programación interpretado que se utilizó para generar el espacio de trabajo del robot Delta.
- **Librerías de python:**
  - **Librería Matplotlib:** Librería que facilita la representación en 3D de series de datos, en nuestro caso, de posiciones alcanzables por el robot y la representación gráfica del propio robot en el espacio.
  - **Librería numpy:** Librería para el cálculo numérico que utilizamos para las operaciones matemáticas necesarias para la resolución de la cinemática inversa.
  - **Librería pandas:** Librería de análisis y manipulación de datos de código abierto, la utilizamos para importar datos desde formato Excel.
  - **Librería xlsxwriter:** Librería cuya función es crear y exportar hojas de cálculo en formato Excel.

Por otra parte, la secuencia de tareas que se ha realizado es la siguiente:

1. Estudio de la cinemática inversa.
2. Creación del modelo en Geogebra para la cinemática inversa.
3. Estudio de la cinemática directa.
4. Creación del modelo en Geogebra para la cinemática directa.
5. Desarrollo de la herramienta de python para el espacio de trabajo del robot.
6. Elección de las dimensiones del prototipo.
7. Diseño de los elementos del robot utilizando Autodesk Fusion 360.
8. Impresión 3D de las piezas que conforman el robot.
9. Ensamblaje del robot.
10. Dotación de electrónica para el funcionamiento del robot.

En este proyecto no se contempla el desarrollo del software para el funcionamiento del robot ya que este es ampliamente dependiente de la aplicación a la que se destinará el robot, pero sí se dota al robot de la electrónica necesaria para su funcionamiento.

1.3. Justificación

La justificación de este trabajo consiste en los siguientes aspectos:

- Divulgación: Se da a entender fácilmente en qué consiste y cómo funciona el Robot de tipo Delta fuera del ámbito en el que se suelen utilizar y de una manera atractiva.

- Capacidad de mejora:
  - Social: Al dar a conocer de una manera más comprensible este tipo de robots se puede generar interés en el estudio de los robot Delta en personas con diferentes puntos de vista que posiblemente de otra manera no se hubieran interesado.
  - Este proyecto: Al emplear herramientas de uso personal gratuito, accesibles para todo el mundo, es posible dar una continuidad a este trabajo, tanto creando mejoras a lo existente, como desarrollando un software para su funcionamiento o alguna característica inexplorada.

- Replicabilidad: Con la información contenida en este proyecto es posible replicar el prototipo y cualquier persona que quiera disponer de él puede hacerlo fácilmente siguiendo el trabajo realizado.

- Valor práctico: Será un robot interesante desde el punto de vista académico, para la realización de prácticas de asignaturas relacionadas con la robótica. Además, podrá usarse como método de impresión 3D o bien para la manipulación de objetos, complementando la acción de los manipuladores antropomórficos presentes habitualmente en los laboratorios. Por otra parte, cualquier persona que disponga de un robot Delta, o esté pensando en modificar o fabricar uno, puede utilizar las herramientas y la información presentadas en este trabajo para dar solución a la cinemática inversa de su robot o investigar cualquier otro parámetro que sea de su interés.

1.4. Contenidos

La estructura de la presente memoria se resume en los siguientes capítulos:

- Capítulo 2: Antecedentes y estado del arte
- Capítulo 3: Cinemática inversa
- Capítulo 4: Cinemática Directa
- Capítulo 5: Modelos cinemáticos en Geogebra
- Capítulo 6: Herramienta de python y determinación de dimensiones del prototipo
- Capítulo 7: Diseño de los elementos del robot
- Capítulo 8: Impresión 3D de los elementos del robot
- Capítulo 9: Estructura mecánica final del robot y dotación de la electrónica

Capítulo 2: Antecedentes y estado del arte

2.1. ¿Por qué automatizar los procesos industriales?

Una de las razones principales por la que la industria tiende cada vez más hacia una automatización de los procesos presentes en una cadena de producción es la eficiencia. Las máquinas y robots que se emplean en dichos procesos son de gran valor para una empresa, ya que son capaces de encargarse de tareas repetitivas, de manera más rápida que un operario humano, sin el descanso necesario para un humano y con unos resultados en cuanto a precisión y repetibilidad inigualable por un humano [4].

Estas máquinas ofrecen una alta inversión inicial en el momento de su adquisición. Sin embargo, el coste de su operación consiste casi exclusivamente en gastos de mantenimiento programables. Por el contrario, si quisiéramos realizar el mismo trabajo con labor humana sería necesaria una plantilla con un coste mayor que el que supone esta automatización, tendríamos resultados de peor calidad y una menor optimización del uso de los materiales al desperdiciar más materia prima si la automatización del proceso está bien implementada. Debido a esto, son una alternativa más rentable y recuperan el coste de su inversión rápidamente.

En resumen, los beneficios de la automatización industrial son los siguientes [4][5][6]:

- Reducción de costes
- Reducción del tiempo en la ejecución de tareas
- Productividad aumentada
- Eliminación de cuellos de botella
- Mejor calidad global
- Mayor consistencia del producto
- Mayor seguridad para los empleados
- Disponibilidad aumentada
- Mayor Confiabilidad

Sin embargo, existe un dilema social en la automatización de procesos industriales. Gracias al empleo de máquinas y robots se logra aumentar la eficiencia de la cadena de producción pero se destruyen empleos y, por tanto, la calidad de vida de los que habrían sido trabajadores en una planta de producción, generando una mayor desigualdad social entre la población cualificada y los que no han podido acceder a estudios superiores [7][8].

Por otro lado, no es un futuro oscuro. La automatización industrial crea nuevos puestos de trabajo de mayor cualificación y mejor remunerados, deja libre a la población menos formada para que cursen estudios superiores y posibilita un aumento de su calidad de vida al realizar trabajos menos forzados y costosos.

2.2. Introducción al robot Delta

El robot delta es un robot de tres grados de libertad. Está formado por dos bases que se mantienen siempre paralelas unidas por tres brazos simétricamente distribuídos basados en el uso de paralelogramos dispuestos a 0º, 120º y 240º cada uno.

La base superior permanece inmóvil, sin embargo, la base inferior es móvil y es donde se encuentra el efector final. Cada brazo está formado por un bíceps que puede rotar en su plano vertical en torno a su punto de anclaje en la base superior y, en su otro extremo, está unido al antebrazo que conecta el bíceps con la base del efector final y que puede rotar en todas las direcciones tanto en la unión con el bíceps como en la unión con la base del efector final. Este robot posee 3 variables de actuación, los ángulos θ_1, θ_2 y θ_3, que son el ángulo de los bíceps con respecto al eje horizontal [1][2][3].

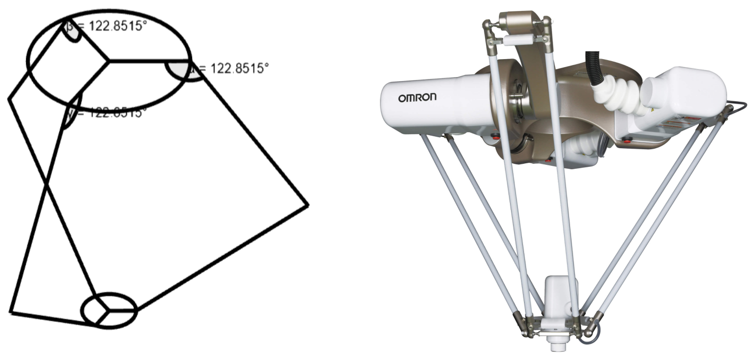


Figura 1: Robot Delta (modelo y real). Bibliografía de imágenes [1]

Como podemos observar en la Fig. 1, el robot tipo delta consta de 4 parámetros en su construcción, el radio de la base superior, el radio de la base del efector final, el largo de los bíceps y el largo de los antebrazos.

2.3. El papel de los robots delta en la industria

El profesor Reymond Clavel de la École Polytechnique Fedérale de Lausanne en Suiza fue el primero en construir a mediados de la década de los 80 un robot de tipo Delta. Desde entonces, han habido muchos avances en sus prestaciones y se han incorporado a la producción industrial para convertir las cadenas de producción en procesos más rentables y eficientes [1].

Los robots Delta son ampliamente utilizados para operaciones de manipulación de tipo “Pick & place” de objetos ligeros entre 3 - 12 kilogramos, encajando con precisión componentes electrónicos en una placa de circuitos impresos o clasificando con gran velocidad productos en cintas transportadoras. Sus aplicaciones son muy variadas gracias a su versatilidad y rendimiento incluso llegándose a utilizar para operaciones quirúrgicas [9][10].

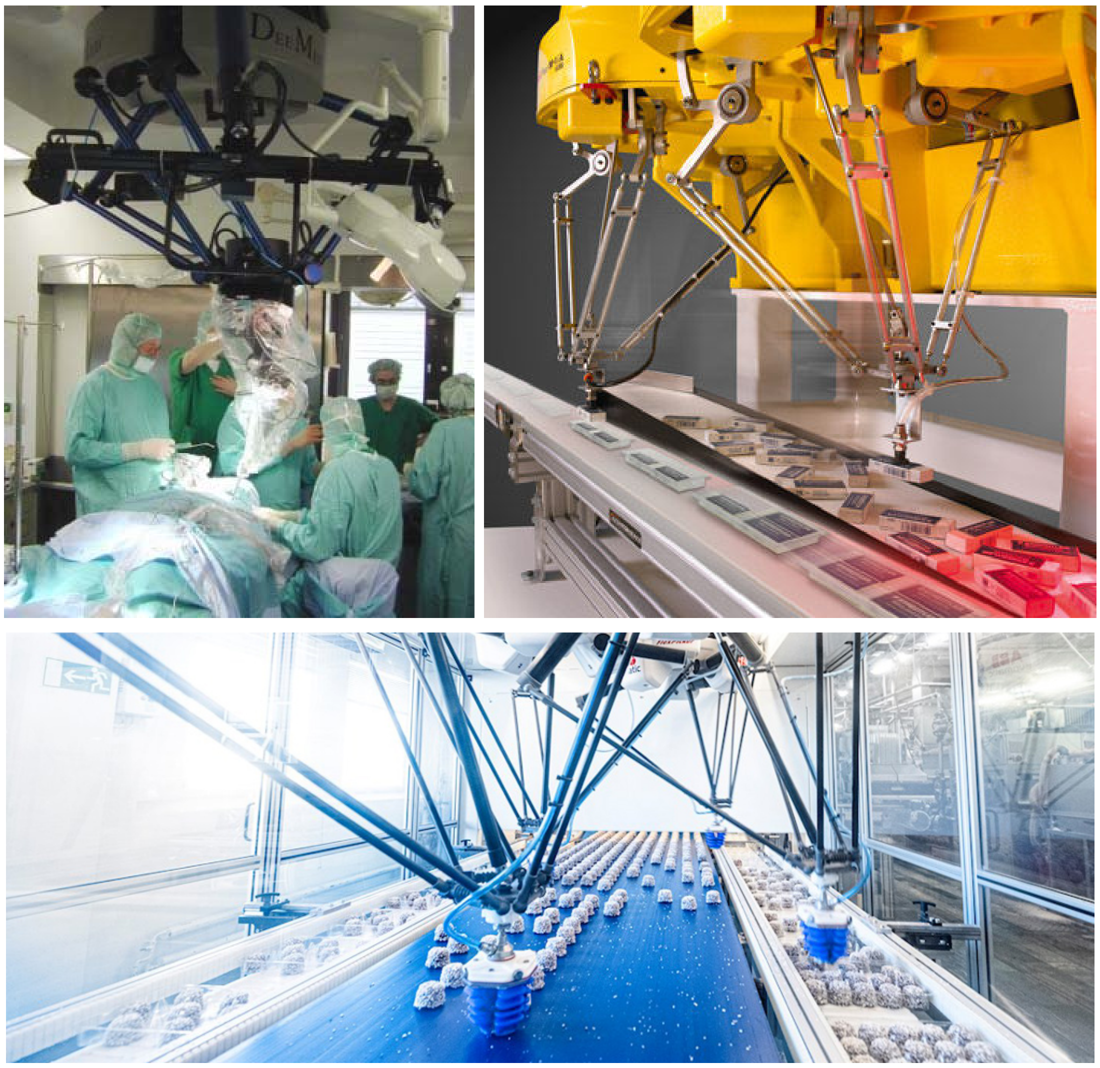


Figura 2: Aplicaciones del robot Delta. Bibliografía de imágenes [2][3][4]

<u>2.4. Opciones comerciales y sus características</u>

En función de las necesidades que existen en una cadena de producción, tales como capacidad de carga, velocidad, precisión y espacio de trabajo, diferentes fabricantes han desarrollado robots Delta para su comercialización.

Existe un gran número de opciones al alcance de la industria a la hora de adquirir un robot de tipo Delta. El precio de los mismos varía según las prestaciones que sean necesarias y, pese a que existen modelos desde los 10.000€, la gama media de los robot Delta comienza a partir de los 25.000€. Sin embargo, estos robots consiguen un retorno y posterior beneficio de la inversión efectuada rápidamente [1][3].

Las opciones comerciales de fabricantes de robots para automatización más destacadas de robots de tipo delta son las siguientes:

KUKA KR DELTA

Robot de alta velocidad para operaciones higiénicas de manejo y recogida de piezas pequeñas en sectores industriales como la industria alimentaria, farmacéutica y electrónica entre otros. Posee las siguientes características [11]:

- Alcance: 1.200 mm
- Carga útil: 3 kg
- Repetibilidad: +/- 0.1 mm
- Huella: 350 mm
- Peso: 95 kg

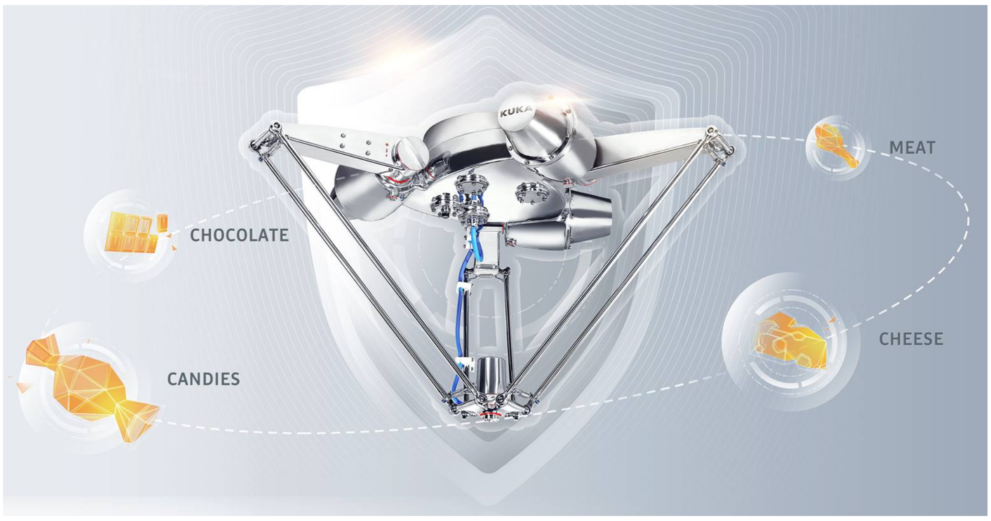


Figura 3: Robot KUKA KR DELTA. Bibliografía de imágenes [5]

Omron X-Delta 3+1

Robot de alta velocidad con amplias opciones de espacio de trabajo, carga útil y con opciones de detección anticolisión y clasificación IP65, IP67, IP69K. Posee las siguientes características [12]:

- Grados de libertad: 3 + 1 (eje de rotación opcional)
- Hasta 200 ciclos por minuto
- Modelos con un rango de trabajo de 500 a 1.600 mm
- Rango de carga útil: 1 a 8 kg
- Rango de clase IP: IP65, IP67, IP69K
- Opción de detección anticolisión

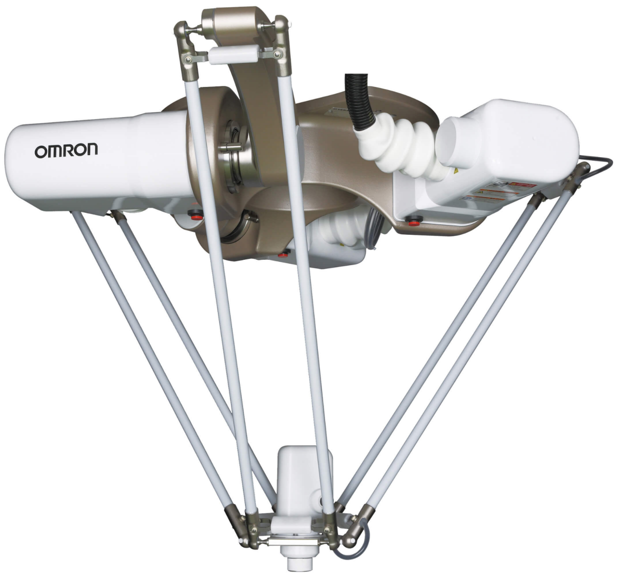


Figura 4: Robot Omron X-Delta 3+1. Bibliografía de imágenes [1]

ABB IRB 360 FlexPicker

Robot de alta velocidad líder en tecnología durante más de 20 años según la marca, capaz de realizar aplicaciones de preparación de pedidos y optimizado para aplicaciones de packaging. Posee las siguientes características [13]:

- cargas útiles de 1 kg, 3 kg, 6 kg y 8 kg
- alcances de 1130 mm y 1600 mm
- Software de visión integrado
- robot delta industrial más rápido del mundo

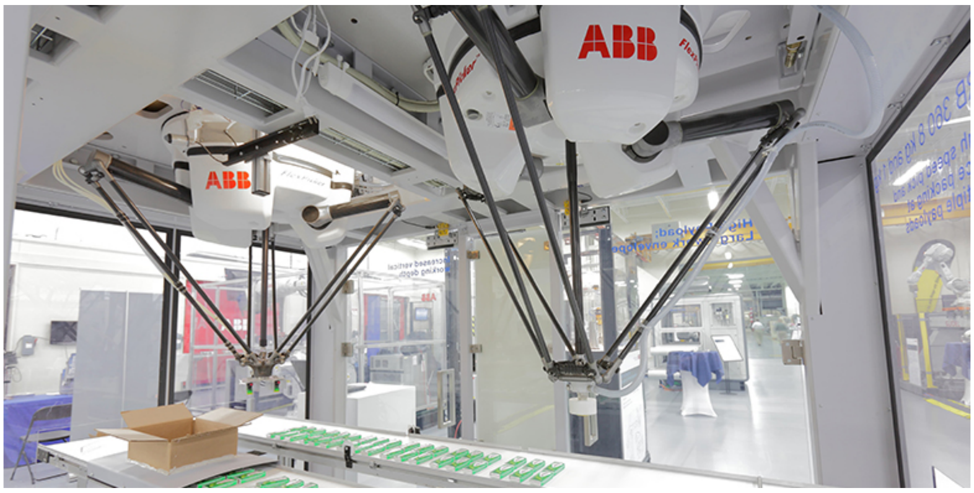


Figura 5: Robot ABB IRB 360 FlexPicker. Bibliografía de imágenes [4]

Como hemos podido comprobar con estos tres ejemplos de robots delta comerciales, son robots de movimiento rápido, capacidad de carga menor a 10 kilogramos y con un espacio de trabajo de entre medio metro y un poco más de metro y medio.

Capítulo 3: Cinemática inversa

3.1. Introducción

Para resolver la cinemática inversa del robot de tipo delta será necesario seguir los siguientes pasos:

1. Hallar los puntos de anclaje de los bíceps en la base superior con respecto al origen de coordenadas, así como los puntos de anclaje de los antebrazos con la base del efector final.
2. Construir las circunferencias con centro en los puntos de anclaje de los bíceps en la base superior que describen el movimiento de los bíceps en cada uno de sus planos de rotación.
3. Construir las esferas centradas en los puntos de anclaje de los antebrazos con la base del efector final que describen el movimiento de los antebrazos.
4. Intersectar las esferas que describen el movimiento de los antebrazos con los planos que contienen las circunferencias que describen el movimiento de los bíceps, dando lugar a circunferencias de intersección.
5. Hallar los puntos de intersección entre las circunferencias de intersección entre plano y esfera y las circunferencias que describen el movimiento de los bíceps, estos puntos de intersección son la posición de las uniones entre bíceps y antebrazos denominadas codos.
6. A partir de la intersección entre estas dos circunferencias conocer el valor de las variables articulares.

Estos pasos se detallarán en los siguientes apartados de este capítulo.

3.2. Puntos de anclaje de los bíceps en la base superior

Para obtener los puntos de anclaje de los bíceps en la base superior situamos el origen en el centro de la base superior y uno de los brazos en el eje x, quedando el resto a 120º y 240º respectivamente.

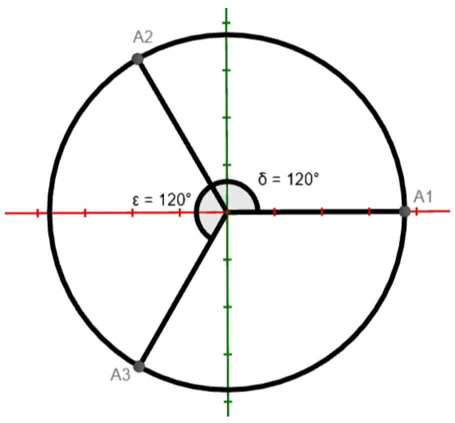


Figura 6: Base superior del robot

En la Fig. 6 se observa la base superior del robot. Se puede obtener la posición de los puntos A_1, A_2 y A_3 respecto al origen O con las siguientes fórmulas siendo Φ_i el ángulo con el eje x de cada uno de los brazos:

$$A_i = \left(O_x + radioBase \cdot cos(\phi_i),\ O_y + radioBase \cdot sen(\phi_i),\ O_z\right)$$

Simplificando para el brazo 1 debido a que el ángulo Φ_1 es nulo:

$$A_1 = \left(O_x + radioBase,\ O_y,\ O_z\right)$$

De esta manera hemos obtenido los puntos de anclaje de los bíceps en la base superior en donde centraremos las circunferencias que describen los bíceps.

3.3. Puntos de anclaje de los antebrazos en la base del efector final

Para obtener los puntos de anclaje de los antebrazos en la base del efector final procederemos como en el apartado anterior. Sin embargo, utilizaremos el punto final como centro de la base del efector final y uno de los brazos en el eje x, quedando el resto a 120º y 240º respectivamente.

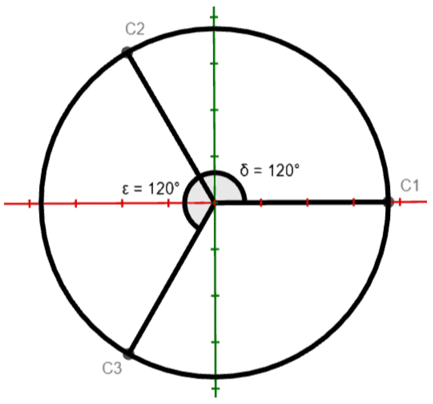


Figura 7: Base del efector final

En la Fig. 7 se observa la base del efector final. Se puede obtener la posición de los puntos C_1, C_2 y C_3 respecto al punto final P con las siguientes fórmulas siendo Φ_i el ángulo con el eje x de cada uno de los brazos:

$$C_i = \left(P_x + radioEfector \cdot cos(\phi_i),\ P_y + radioEfector \cdot sen(\phi_i),\ P_z\right)$$

Simplificando para el brazo 1 debido a que el ángulo Φ_1 es nulo:

$$C_1 = \left(P_x + radioEfector,\ P_y,\ P_z\right)$$

De esta manera hemos obtenido los puntos de anclaje de los antebrazos en la base del efector final en donde centraremos las esferas que describen los antebrazos.

3.4. Planos que contienen las circunferencias que describen los bíceps

Será necesario encontrar los planos que contienen las circunferencias que describen los bíceps para encontrar la circunferencia de intersección entre las esferas que describen el movimiento de los antebrazos y el plano de su bíceps correspondiente para luego encontrar el punto de unión del brazo.

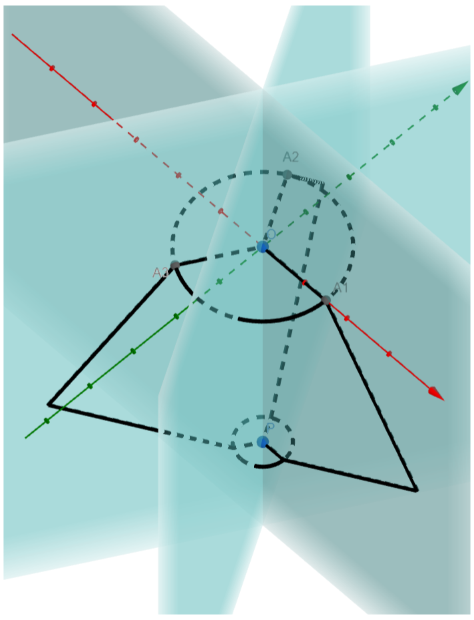


Figura 8: Planos que contienen las circunferencias que describen los bíceps

Como se puede observar en la Fig. 8, debido a que los 3 planos que deseamos construir son verticales y pasan por el centro de la base superior comparten uno de los vectores directores y el punto del origen. Por tanto:

$$\bar{u}_1 = \bar{u}_2 = \bar{u}_3 = (0,\ 0,\ 1)$$

$$\bar{v}_i = \left(cos(\phi_i),\ sen(\phi_i),\ 0\right)$$

$$O = \left(O_x,\ O_y,\ O_z\right)$$

La ecuación general de los planos queda:

$$p_i : -\ sen(\phi_i)x + cos(\phi_i)y + \left(sen(\phi_i)O_x - cos(\phi_i)O_y\right) = 0$$

Simplificando para el brazo 1 debido a que el ángulo Φ_1 es nulo:

$$p_1 : y - O_y = 0$$

3.5. Intersección de las esferas con los planos

Nos interesa conocer el centro de la circunferencia de intersección entre el plano y la esfera, así como su radio, para poder obtener el punto de intersección entre esta nueva circunferencia y la que describe el movimiento del bíceps del brazo. Se puede observar la intersección entre esfera y plano en la Fig. 9.

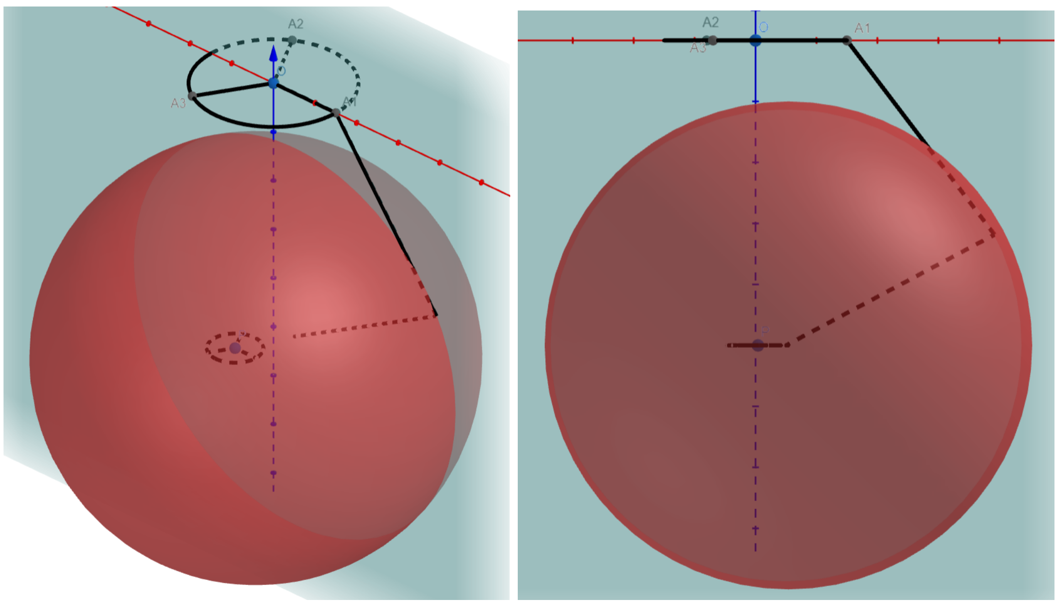


Figura 9: Intersección esfera y plano del brazo 1

Al tratar de intersectar un plano con una esfera se obtiene, en función de la distancia "d" entre el centro de la esfera y el plano, lo siguiente:

- Si d < radio de la esfera, circunferencia de intersección
- Si d = radio de la esfera, punto de intersección
- Si d > radio de la esfera, no hay intersección

La fórmula general de la distancia entre un plano y un punto es:

$$d = \frac{\left|Ax_0+By_0+Cz_0+D\right|}{\sqrt{A^2+B^2+C^2}}$$

Aplicándola para nuestro caso nos queda la siguiente fórmula general:

$$d_i = \left|sen(\phi_i)\left(O_x - C_{ix}\right) + cos(\phi_i)\left(C_{ix} - O_y\right)\right|$$

Simplificando para el brazo 1 debido a que el ángulo Φ_1 es nulo:

$$d_1 = \left|C_{1y} - O_y\right|$$

Para conocer el radio de la nueva circunferencia construiremos el siguiente triángulo cuya hipotenusa es el radio de la esfera, es decir, el largo del antebrazo, el cateto que une el centro de la esfera con el plano es la distancia que acabamos de calcular y el otro cateto será el radio de nuestra circunferencia de intersección.

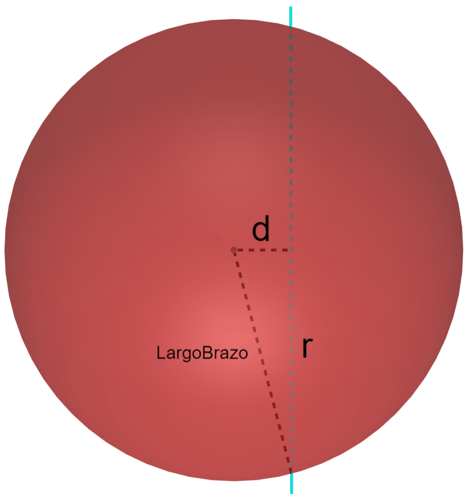


Figura 10: Triángulo para conocer el radio de la circunferencia

Por tanto, tal y como observamos en la Fig. 10, podemos conocer el radio de la circunferencia de intersección aplicando el teorema de pitágoras.

$$r_i = \sqrt{largoBrazo^2 - d_i^2}$$

Por último, para hallar el centro de la nueva circunferencia, crearemos una recta perpendicular al plano que pase por el centro de la esfera y la intersección con el plano será el centro que buscamos.

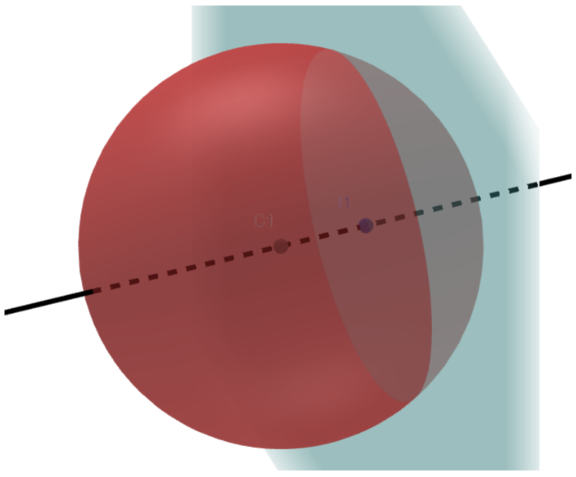


Figura 11: Recta normal al plano que pasa por el centro de la esfera

Por tanto, utilizando como vector director de la recta el vector normal del plano y haciendo que pase por el centro de la esfera correspondiente, nos queda la siguiente recta paramétrica:

$$R_i = \begin{cases} x = C_{ix} - sen(\phi_i)t \\ y = C_{iy} + cos(\phi_i)t \\ z = C_{iz} \end{cases}$$

Simplificando para el brazo 1 debido a que el ángulo Φ_1 es nulo:

$$R_1 = \begin{cases} x = C_{1x} \\ y = C_{1y} + t \\ z = C_{1z} \end{cases}$$

Sustituímos estas rectas en la ecuación de los planos para obtener el parámetro "t":

$$t_i = sen(\phi_i)\left(C_{ix} - O_x\right) + cos(\phi_i)\left(O_y - C_{ix}\right)$$

Simplificando para el brazo 1 debido a que el ángulo Φ_1 es nulo:

$$t_1 = O_y - C_{1y}$$

Por tanto, el punto de intersección es:

$$I_i = \begin{cases} x = C_{ix} - sen(\phi_i)\left(sen(\phi_i)\left(C_{ix} - O_x\right) + cos(\phi_i)\left(O_y - C_{ix}\right)\right) \\ y = C_{iy} + cos(\phi_i)\left(sen(\phi_i)\left(C_{ix} - O_x\right) + cos(\phi_i)\left(O_y - C_{ix}\right)\right) \\ z = C_{iz} \end{cases}$$

Simplificando para el brazo 1 debido a que el ángulo Φ_1 es nulo:

$$I_1 = \left(C_{1x}, O_y, C_{1z}\right)$$

3.6. Obtención de las variables de actuación

Una vez obtenido el centro de la circunferencia de intersección, su radio, el centro de la circunferencia que describe el movimiento del bíceps y su radio, el largo del bíceps, podemos encontrar el ángulo con la horizontal que ha de tener el bíceps para que el efector final se sitúe en el punto deseado resolviendo los siguientes triángulos.

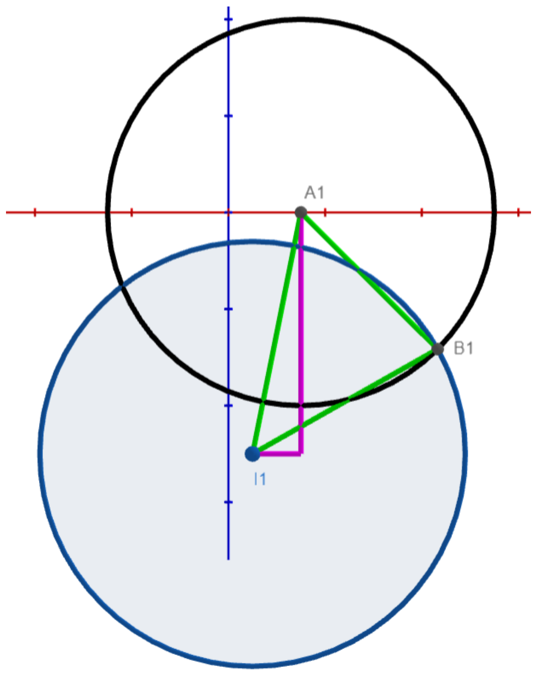


Figura 12: Intersección de circunferencias

Sin embargo, dependiendo de la posición del centro de la circunferencia de intersección con respecto al centro de la circunferencia del bíceps tendremos dos casos diferentes.

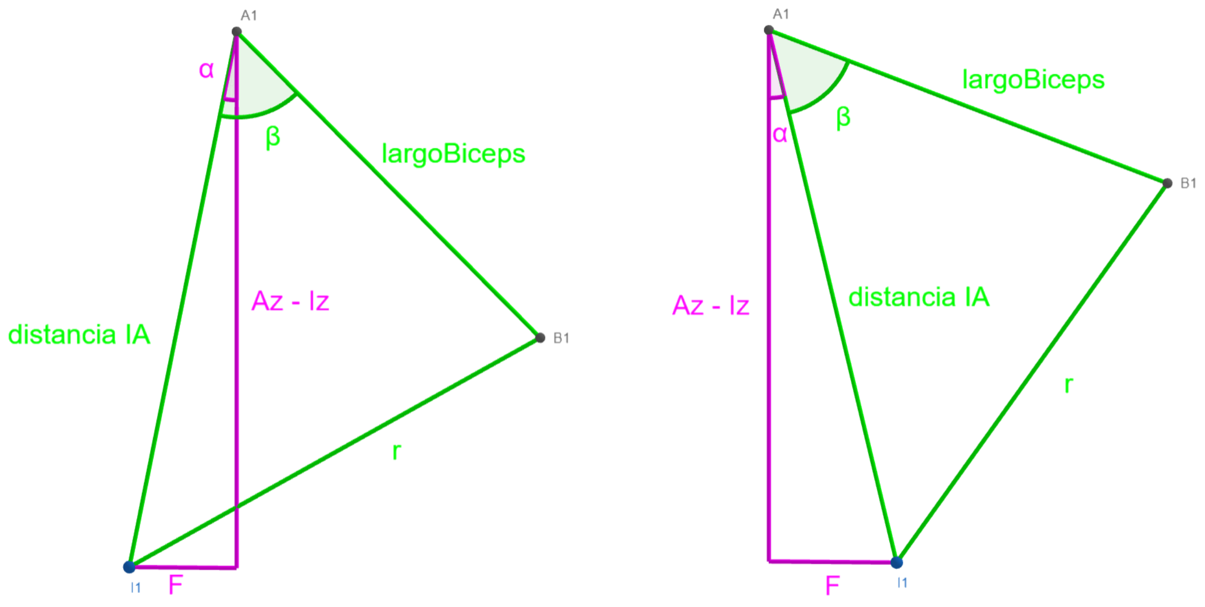


Figura 13: Casos posibles de los triángulos

Por tanto, tal y como se puede observar en la Fig. 13, para encontrar el ángulo con la horizontal del bíceps deberemos hacer lo siguiente:

- Si la distancia horizontal al centro de la base superior del robot del centro de la circunferencia de intersección es mayor a la del centro de la circunferencia del bíceps:

$$\Theta_i = \left(\beta_i + \alpha_i\right) - 90º$$

- Si la distancia horizontal al centro de la base superior del robot del centro de la circunferencia de intersección es menor a la del centro de la circunferencia del bíceps:

$$\Theta_i = \left(\beta_i - \alpha_i\right) - 90º$$

Los ángulos α y β siempre se podrán obtener de la misma manera sin importar esta condición. Obtendremos α resolviendo con pitágoras el triángulo rectángulo resultante.

$$\alpha_i = arccos\left(\frac{A_{iz} - I_{iz}}{d_{IAi}}\right)$$

Por otro lado, obtendremos β utilizando el teorema del coseno sobre el triángulo irregular generado.

$$\beta_i = arccos\left(-\frac{r_i^2 - d_{IAi}^2 - largoBiceps^2}{2 \cdot largoBiceps \cdot d_{IAi}}\right)$$

3.7. Condiciones de posicionamiento

La cinemática inversa planteada en este documento no tiene en cuenta configuraciones de los ángulos del robot que darían lugar a un posible atascamiento o que son físicamente imposibles de llegar sin fuerzas externas.

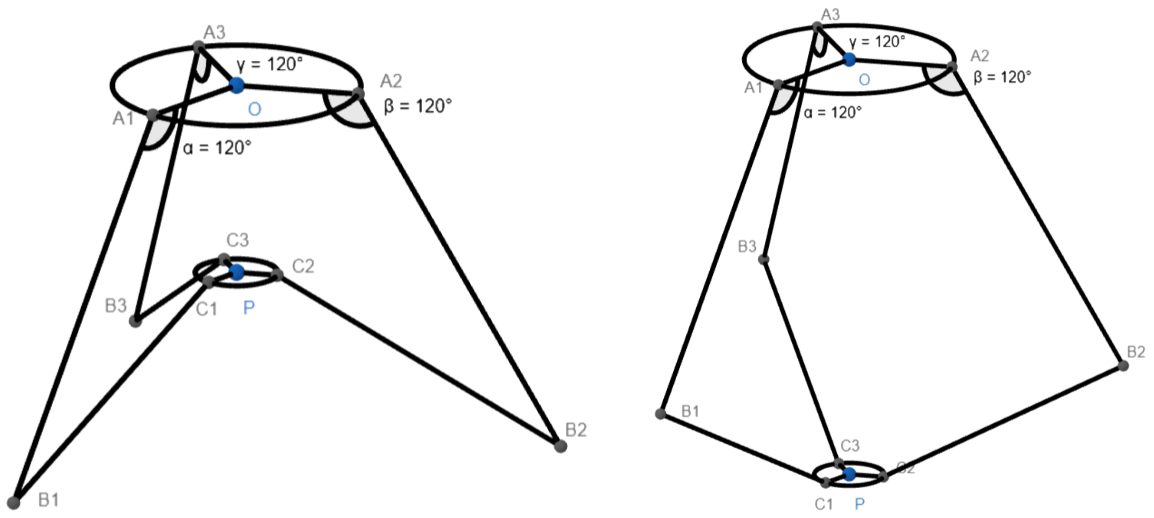


Figura 14: Dos posiciones diferentes para los mismos ángulos

Como se puede observar en la Fig. 14, para los mismos ángulos existe una configuración en la que la base del efector final se encuentra por encima de los codos de los brazos. Debemos descartar cualquier configuración que genere este caso. La base del efector final siempre deberá permanecer por debajo de los codos de los brazos.

Capítulo 4: Cinemática Directa

4.1. Introducción

Para resolver la cinemática directa del robot de tipo delta será necesario seguir los siguientes pasos:

1. Hallar los puntos de anclaje de los bíceps en la base superior con respecto al origen de coordenadas.
2. Construir las circunferencias centradas en los puntos de anclaje de los bíceps en la base superior que describen el movimiento de los bíceps y averiguar la posición de los extremos de los bíceps sustituyendo el valor de las variables articulares introducidas como dato.
3. Construir las rectas que pasan por los extremos de los bíceps y por el origen en su misma altura.
4. Encontrar los centros de las esferas que describen el movimiento de los antebrazos. Esto se efectuará encontrando los puntos pertenecientes a las rectas anteriores que se encuentren a una distancia de los extremos de los bíceps igual al radio del efector final.
5. Construir las esferas de radio igual al largo de los antebrazos, centradas en los puntos recién descritos para, posteriormente, intersectarlas entre sí para obtener la posición del centro de la base del efector final.
6. Seguir el mismo procedimiento que para los puntos de anclaje de los bíceps en la base superior para encontrar los puntos de anclaje de los antebrazos en la base del efector final.

Cada uno de estos pasos se explica en los apartados siguientes.

4.2. Puntos de anclaje de los bíceps en la base superior

Para obtener los puntos de anclaje de los bíceps en la base superior situamos el origen en el centro de la base superior y uno de los brazos en el eje x, quedando el resto a 120º y 240º respectivamente.

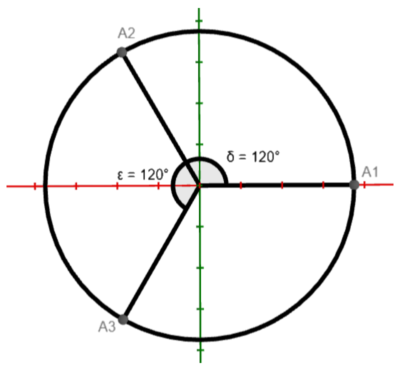


Figura 15: Base superior del robot

En la Fig. 15 se observa la base superior del robot, se puede obtener la posición de los puntos A_1, A_2 y A_3 respecto al origen O con las siguientes fórmulas siendo Φ_i el ángulo con el eje x de cada uno de los brazos:

$$A_i = \left(O_x + radioBase \cdot cos(\phi_i),\ O_y + radioBase \cdot sen(\phi_i),\ O_z\right)$$

Simplificando para el brazo 1 debido a que el ángulo Φ_1 es nulo:

$$A_1 = \left(O_x + radioBase,\ O_y,\ O_z\right)$$

De esta manera hemos obtenido los puntos de anclaje de los bíceps en la base superior en donde centraremos las circunferencias que describen los bíceps.

4.3. Extremos de los bíceps

Para hallar los extremos de los bíceps será necesario construir las circunferencias que describen su movimiento. Para ello, será necesario disponer de sus centros, los puntos de anclaje en la base superior del robot, los vectores que la forman, siendo el vector de la componente "z" común a las tres circunferencias y el radio, que no es más que el largo de los bíceps.

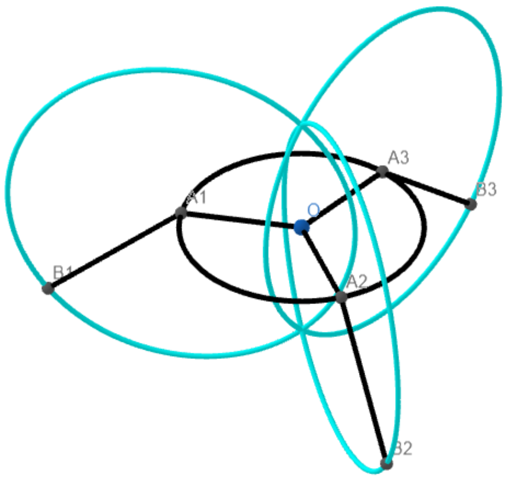


Figura 16: Circunferencias de los bíceps

Por tanto, conociendo los puntos de anclaje de los bíceps en la base superior del robot y los siguientes vectores:

$$\overline{u}_1 = \overline{u}_2 = \overline{u}_3 = (0,\ 0,\ 1)$$

$$\overline{v}_i = \left(cos(\phi_i),\ sen(\phi_i),\ 0\right)$$

Podemos construir las circunferencias que describen el movimiento de los bíceps:

$$C_i(t) = \begin{cases} x = A_{ix} + largoBíceps \cdot cos(\phi_i) \cdot cos(t) \\ y = A_{iy} + largoBíceps \cdot sen(\phi_i) \cdot cos(t) \\ z = A_{iz} + largoBíceps \cdot sen(t) \end{cases}$$

Si sustituímos el parámetro *t* por Θ_i obtendremos el punto final del bíceps:

$$B_i = \left(A_{ix} + largoBíceps \cdot cos(\phi_i) \cdot cos(\Theta_i),\ A_{iy} + largoBíceps \cdot sen(\phi_i) \cdot cos(\Theta_i),\ A_{iz} + largoBíceps \cdot sen(\Theta_i)\right)$$

Simplificando para el brazo 1 debido a que el ángulo Φ_1 es nulo:

$$B_1 = \left(A_{1x} + largoBíceps \cdot cos(\Theta_1),\ A_{1y},\ A_{1z} + largoBíceps \cdot sen(\Theta_1)\right)$$

4.4. Centros de las esferas

Para hallar los centros de las esferas que describen el posible movimiento de los antebrazos es necesario construir las rectas que pasan por los extremos de los bíceps y por el origen en su misma altura. Los centros estarán situados a una distancia de los extremos de los bíceps igual al radio del efector final.

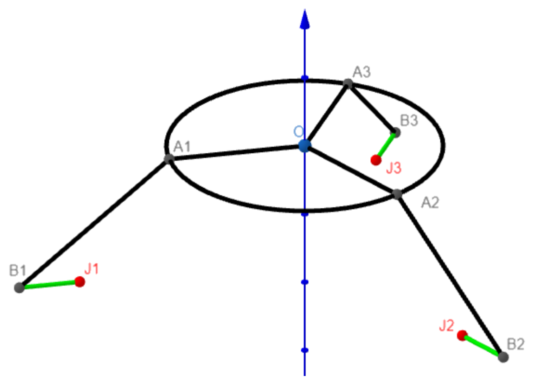


Figura 17: Centros de las esferas de los antebrazos

Por tanto, utilizaremos como vector director de las rectas el siguiente vector:

$$\bar{v}_i = \left(-\ cos(\phi_i),\ -\ sen(\phi_i),\ 0\right)$$

Haciendo que la recta pase por cada extremo de los bíceps nos queda la siguiente ecuación paramétrica de la recta:

$$r_i(t) = \begin{cases} x = B_{ix} - cos(\phi_i) \cdot t \\ y = B_{iy} - sen(\phi_i) \cdot t \\ z = B_{iz} \end{cases}$$

Como nos queremos alejar del extremo de los bíceps una distancia igual al radio de la base del efector final sustituimos t por esa distancia para hallar el centro de las esferas:

$$J_i = \left(B_{ix} - cos(\phi_i) \cdot radioEfector,\ B_{iy} - sen(\phi_i) \cdot radioEfector,\ B_{iz}\right)$$

Simplificando para el brazo 1 debido a que el ángulo Φ_1 es nulo:

$$J_1 = \left(B_{1x} - radioEfector,\ B_{1y},\ B_{1z}\right)$$

4.5. Esferas de los antebrazos

Una vez hemos hallado los centros de las esferas y, debido a que sabemos que su radio será el largo de los antebrazos, podemos desarrollar su fórmula matemática.

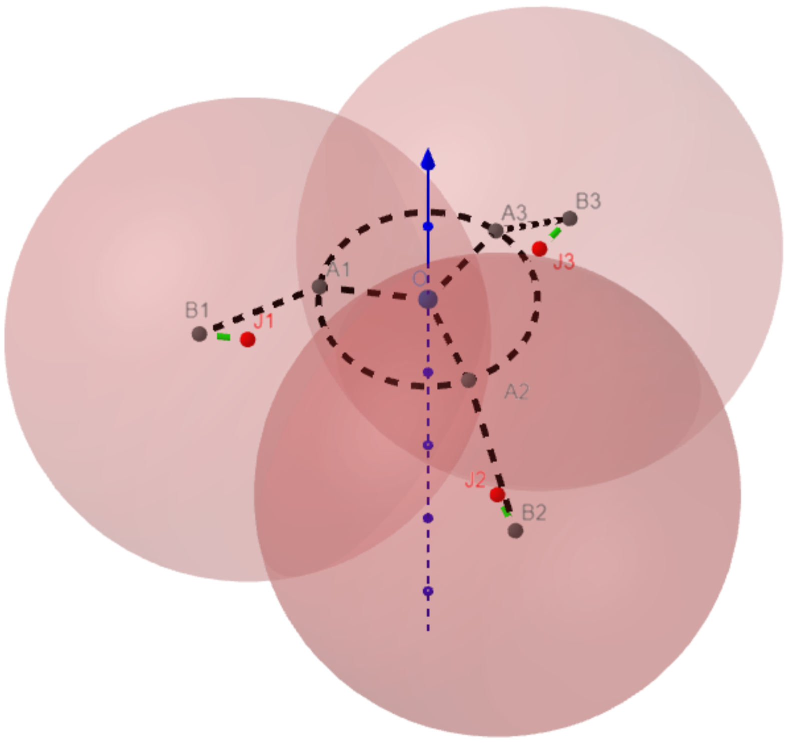


Figura 18: Esferas de los antebrazos

La ecuación general de estas esferas es:

$$\left(x - J_{ix}\right)^2 + \left(y - J_{iy}\right)^2 + \left(z - J_{iz}\right)^2 = largoBrazo^2$$

Que desarrollada queda:

$$x^2 + y^2 + z^2 - 2xJ_{ix} - 2yJ_{iy} - 2zJ_{iz} + J_{ix}^2 + J_{iy}^2 + J_{iz}^2 - largoBrazo^2 = 0$$

4.6. Intersección de las esferas de los antebrazos

Si escogemos una de las ecuaciones de las esferas y se las restamos a las otras dos nos quedan dos planos, que conjuntamente definen una recta:

$$2\left(J_{1x} - J_{2x}\right)x + 2\left(J_{1y} - J_{2y}\right)y + 2\left(J_{1z} - J_{2z}\right)z + \left(J_{2x}^2 - J_{1x}^2 + J_{2y}^2 - J_{1y}^2 + J_{2z}^2 - J_{1z}^2\right) = 0$$

$$2\left(J_{1x} - J_{3x}\right)x + 2\left(J_{1y} - J_{3y}\right)y + 2\left(J_{1z} - J_{3z}\right)z + \left(J_{3x}^2 - J_{1x}^2 + J_{3y}^2 - J_{1y}^2 + J_{3z}^2 - J_{1z}^2\right) = 0$$

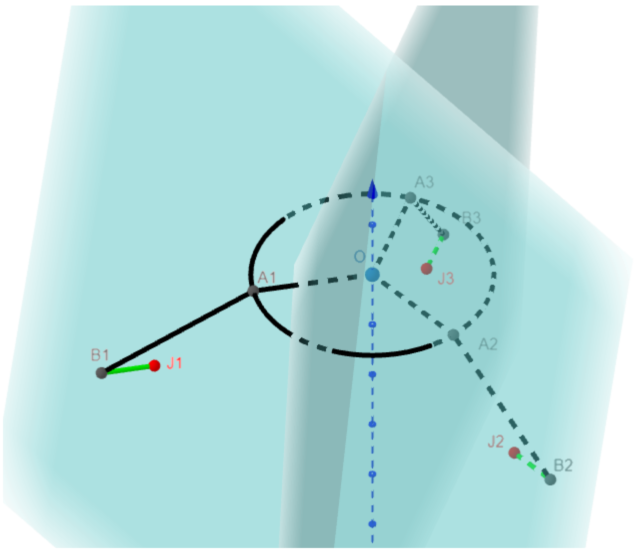


Figura 19: Planos de intersección

Para encontrar la recta de intersección debemos sacar un punto de la intersección de los planos y encontrar el vector director de la recta realizando el producto vectorial entre los vectores normales a los planos. Para hacer más fácil la comprensión utilizaremos las siguientes sustituciones para los planos:

$$A_1x + B_1y + C_1z + D_1 = 0$$

$$A_2x + B_2y + C_2z + D_2 = 0$$

Por tanto, comenzamos por encontrar un punto arbitrario perteneciente a la recta de intersección:

$$A_1B_2x + B_1B_2y + C_1B_2z + D_1B_2 = 0$$

$$A_2B_1x + B_2B_1y + C_2B_1z + D_2B_1 = 0$$

Nos queda:

$$\left(A_1B_2 - A_2B_1\right)x + \left(C_1B_2 - C_2B_1\right)z + \left(D_1B_2 - D_2B_1\right) = 0$$

tomando como valor arbitrario para *z = 1*

$$x = \frac{-\left(C_1B_2 - C_2B_1\right) - \left(D_1B_2 - D_2B_1\right)}{\left(A_1B_2 - A_2B_1\right)}$$

Sustituímos estas variables en la ecuación de cualquiera de los planos:

$$A_1\left(\frac{-\left(C_1B_2 - C_2B_1\right) - \left(D_1B_2 - D_2B_1\right)}{\left(A_1B_2 - A_2B_1\right)}\right) + B_1y + C_1 + D_1 = 0$$

Despejamos la componente *y*:

$$y = \left(-A_1\left(\frac{-\left(C_1B_2 - C_2B_1\right) - \left(D_1B_2 - D_2B_1\right)}{\left(A_1B_2 - A_2B_1\right)}\right) - C_1 - D_1\right) \div B_1$$

De esta manera obtenemos el punto:

$$I = \left(\frac{-\left(C_1B_2 - C_2B_1\right) - \left(D_1B_2 - D_2B_1\right)}{\left(A_1B_2 - A_2B_1\right)}, \left(-A_1\left(\frac{-\left(C_1B_2 - C_2B_1\right) - \left(D_1B_2 - D_2B_1\right)}{\left(A_1B_2 - A_2B_1\right)}\right) - C_1 - D_1\right) \div B_1, 1\right)$$

Para obtener el vector director de la recta realizamos el producto vectorial entre los vectores normales a los planos quedando el vector director de la recta como sigue:

$$\overline{v} = \left(B_1C_2 - B_2C_1, A_2C_1 - A_1C_2, A_1B_2 - A_2B_1\right)$$

Representamos la recta de forma paramétrica:

$$r(t) = \left[\begin{array}{l} x = \frac{-(C_1B_2-C_2B_1)-(D_1B_2-D_2B_1)}{(A_1B_2-A_2B_1)} + (B_1C_2 - B_2C_1)t \\ y = \left(- A_1\left(\frac{-(C_1B_2-C_2B_1)-(D_1B_2-D_2B_1)}{(A_1B_2-A_2B_1)}\right) - C_1 - D_1\right) \div B_1 + (A_2C_1 - A_1C_2)t \\ z = 1 + (A_1B_2 - A_2B_1)t \end{array} \right.$$

El centro de la base del efector final se encuentra contenido en esta recta. Para ello será necesario sustituir sus componentes en una de las ecuaciones de las esferas para hallar el parámetro t que define este punto.

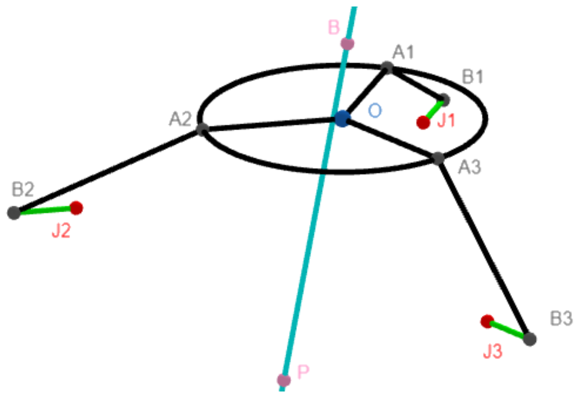


Figura 20: Recta de intersección

Por tanto, para hallar el parámetro t haremos el siguiente cambio de variable:

$$r(t) = \left[\begin{array}{l} x = E_x + F_x t \\ y = E_y + F_y t \\ z = 1 + F_z t \end{array} \right.$$

Sustituimos las componentes de esta recta en la ecuación de una de las esferas:

$$(E_x + F_x t)^2 + (E_y + F_y t)^2 + (1 + F_z t)^2 - 2(E_x + F_x t)J_{1x} - 2(E_y + F_y t)J_{1y} - 2(1 + F_z t)J_{1z} + J_{1x}^2 + J_{1y}^2 + J_{1z}^2 - largoBrazo^2 = 0$$

despejando el parámetro t nos queda:

$$t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

siendo:

$$a = \left(F_x^2 + F_y^2 + F_z^2\right)$$

$$b = \left(2\left[F_x\left(E_x - J_{1x}\right) + F_y\left(E_y - J_{1y}\right) + F_z\left(1 - J_{1z}\right)\right]\right)$$

$$c = \left(E_x^2 + E_y^2 + 1 + J_{1x}^2 + J_{1y}^2 + J_{1z}^2 - largoBrazo^2 - 2\left(E_x J_{1x} + E_y J_{1y} + J_{1z}\right)\right)$$

Sustituyendo el parámetro t en la ecuación de la recta obtenemos el centro de la base del efector final. Como podemos observar, se pueden obtener un número de soluciones variable según:

- No hay intersección, esferas no se cortan → sin solución
- Existe intersección, esferas se cortan en dos puntos → 2 soluciones
- Existe intersección, esferas tangentes → 1 solución

Deberemos escoger la solución que sitúa la base del efector final más abajo, ya que la otra solución daría lugar a tener la base del efector final por encima de los codos de los bíceps, una configuración que debemos descartar.

4.7. Puntos de anclaje de los antebrazos en la base del efector final

Para obtener los puntos de anclaje de los antebrazos en la base del efector final procederemos como con la base superior del robot, sin embargo, utilizaremos el punto final como centro de la base del efector final y uno de los brazos en el eje x, quedando el resto a 120º y 240º respectivamente.

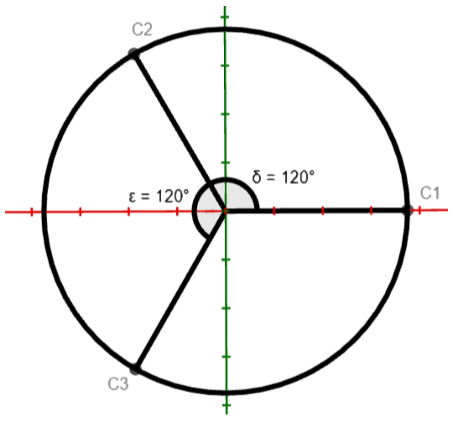


Figura 21: Base del efector final

En la Fig. 7 se observa la base del efector final, se puede obtener la posición de los puntos C_1, C_2 y C_3 respecto al punto final P con las siguientes fórmulas siendo Φ_i el ángulo con el eje x de cada uno de los brazos:

$$C_i = \left(P_x + radioEfector \cdot cos(\phi_i), P_y + radioEfector \cdot sen(\phi_i), P_z\right)$$

Simplificando para el brazo 1 debido a que el ángulo Φ_1 es nulo:

$$C_1 = \left(P_x + radioEfector, P_y, P_z\right)$$

De esta manera hemos obtenido los puntos de anclaje de los antebrazos en la base del efector final y, por tanto, toda la configuración de los puntos del robot en la cinemática directa.

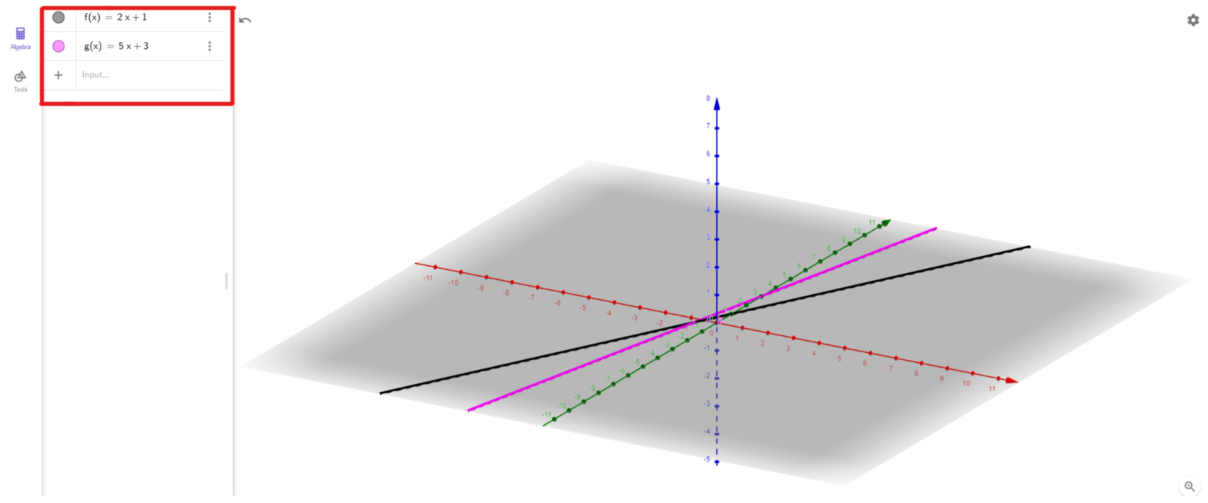
Capítulo 5: Modelos cinemáticos en Geogebra

A lo largo de la explicación tanto de la cinemática inversa como directa, se han ido utilizando imágenes para facilitar la comprensión de lo expuesto matemáticamente. Estas imágenes han sido extraídas de los modelos realizados en Geogebra, donde se pueden observar todos los detalles de la cinemática inversa y directa, modificar los parámetros de construcción del robot y manipular tanto el punto final para el modelo de cinemática inversa, como las variables articulares en el modelo de la cinemática directa.

5.1. ¿Qué es Geogebra?

GeoGebra es un software matemático dinámico para todos los niveles educativos que reúne geometría, álgebra, hojas de cálculo, gráficos, estadísticas y cálculo en un solo motor. Sus características principales son [14]:

- Aplicaciones de Geometría, Álgebra y Álgebra Informática.
- Interfaz fácil de usar, pero con muchas funciones potentes.
- Herramientas para crear recursos de aprendizaje interactivos.
- Está disponible en muchos idiomas.
- Software de código abierto.


$f(x) = 2x + 1$

$g(x) = 5x + 3$

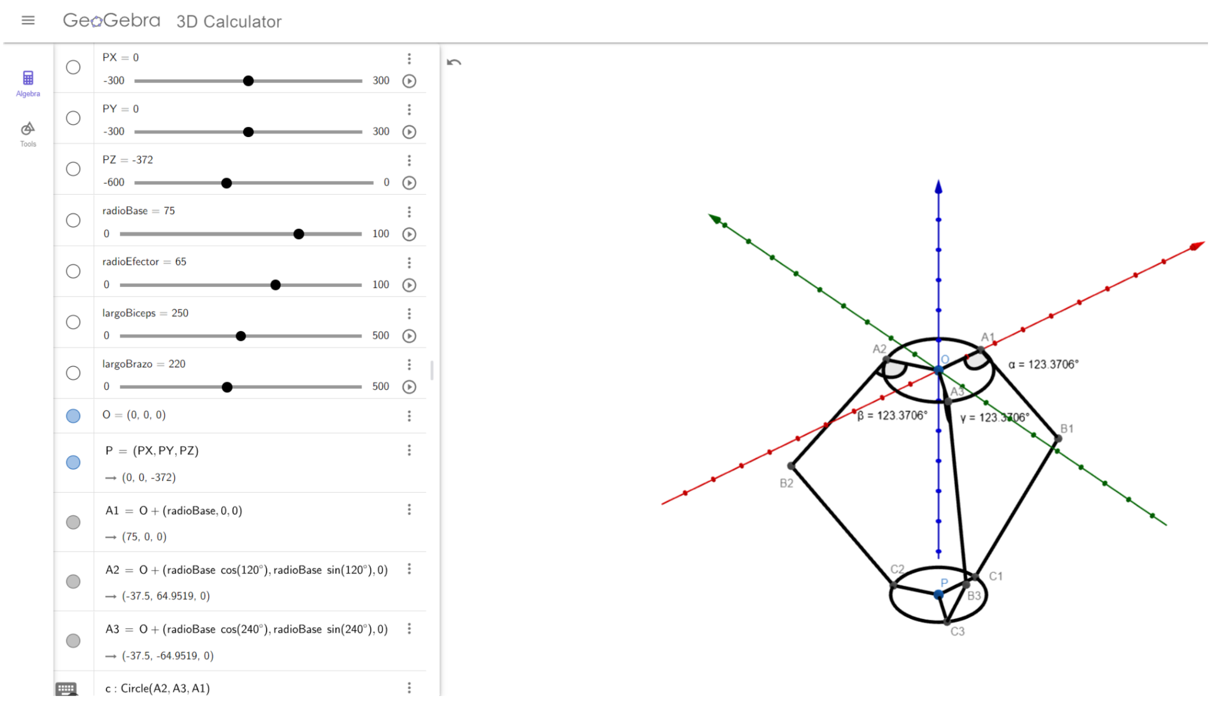
Figura 22: Visión general de Geogebra

En la Fig. 22 podemos observar la vista general de Geogebra. En el menú vertical de la izquierda se pueden introducir ecuaciones para observar gráficamente su representación en la vista 3D del centro de la pantalla.

Sin embargo, la potencia de esta calculadora gráfica no radica ahí. Existe la posibilidad de añadir y modificar parámetros para observar la evolución de las fórmulas matemáticas en función de dicho parámetro, característica que ha sido utilizada para los modelos cinemáticos. Por otro lado, tiene muchas opciones de personalización, como puede ser el color de las funciones y puntos o la escala y perspectiva de la vista 3D. Además, tiene soporte para scripting en su propio lenguaje y en javascript.

5.2. Modelo cinemático inverso

Se ha realizado el modelo de la cinemática inversa del robot Delta en Geogebra siguiendo el proceso matemático descrito en el capítulo 3.



PX = 0
-300 300

PY = 0
-300 300

PZ = -372
-600 0

radioBase = 75
0 100

radioEfector = 65
0 100

largoBiceps = 250
0 500

largoBrazo = 220
0 500

O = (0, 0, 0)

P = (PX, PY, PZ)
→ (0, 0, -372)

A1 = O + (radioBase, 0, 0)
→ (75, 0, 0)

A2 = O + (radioBase cos(120°), radioBase sin(120°), 0)
→ (-37.5, 64.9519, 0)

A3 = O + (radioBase cos(240°), radioBase sin(240°), 0)
→ (-37.5, -64.9519, 0)

c : Circle(A2, A3, A1)

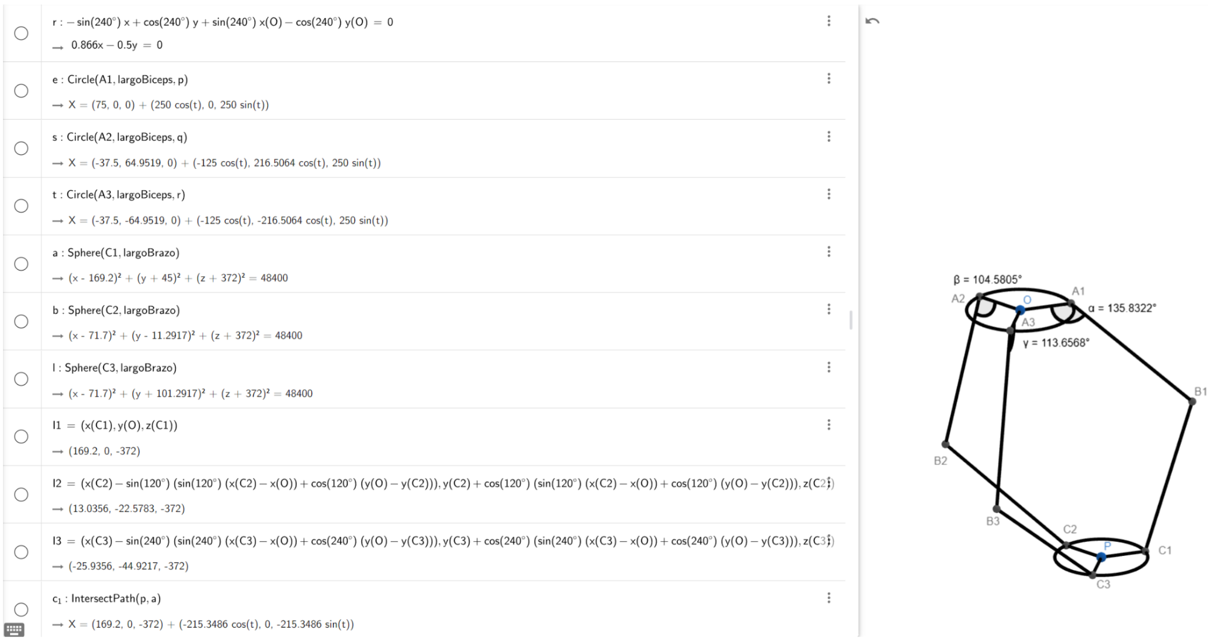
Figura 23: Visión general del modelo cinemático inverso

En la Fig. 23 podemos observar la visión general del modelo cinemático inverso. En el menú de ecuaciones de la izquierda, encabezando las 53 entradas matemáticas presentes en el modelo, podemos encontrar los distintos parámetros presentados en forma de controles deslizantes que pueden ser modificados por el usuario.

Es posible escoger una configuración para los elementos del robot Delta y una posición determinada por medio de los controles deslizantes para obtener, tanto en la vista 3D de la calculadora gráfica como en la entrada correspondiente del menú lateral, los ángulos necesarios para alcanzar el punto final deseado.

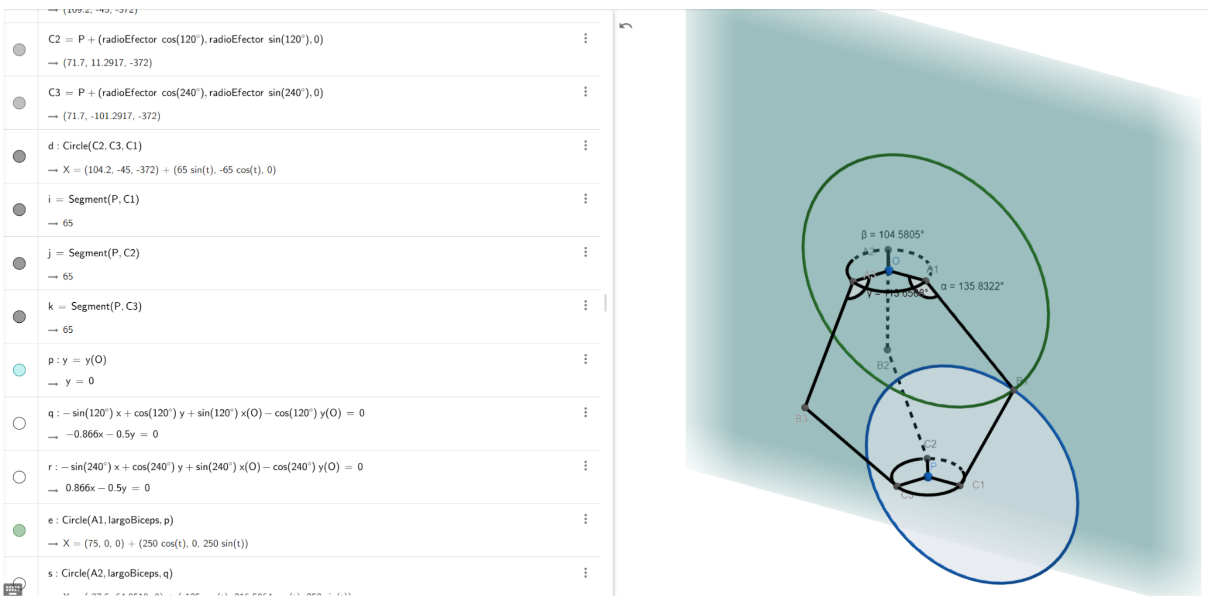
Adicionalmente, es posible mostrar y ocultar cada uno de los elementos presentes en el modelo para obtener una visión más clara de lo que estemos analizando pulsando en el círculo situado a la izquierda de cada entrada matemática en el menú lateral.

A continuación se muestran múltiples vistas de diferentes configuraciones del robot con distintos elementos mostrados para dar una idea general de la conveniencia que supone para la enseñanza este modelo de la cinemática inversa del robot Delta.



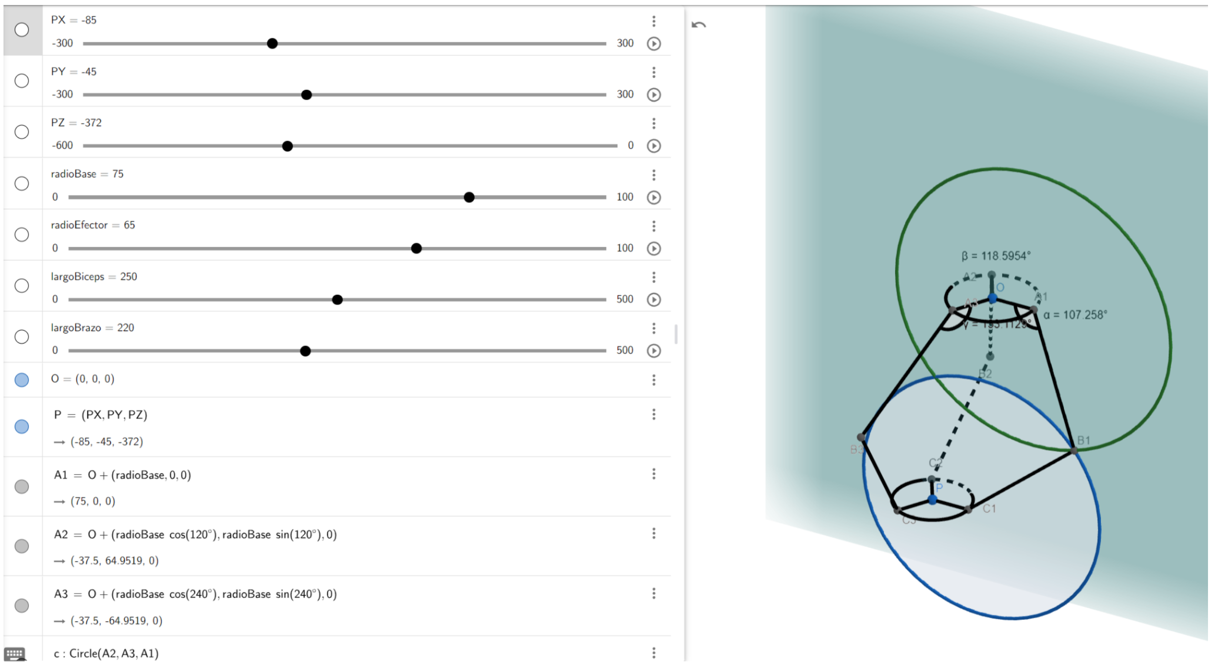
```
r : − sin(240°) x + cos(240°) y + sin(240°) x(O) − cos(240°) y(O) = 0
→ 0.866x − 0.5y = 0

e : Circle(A1, largoBiceps, p)
→ X = (75, 0, 0) + (250 cos(t), 0, 250 sin(t))

s : Circle(A2, largoBiceps, q)
→ X = (-37.5, 64.9519, 0) + (-125 cos(t), 216.5064 cos(t), 250 sin(t))

t : Circle(A3, largoBiceps, r)
→ X = (-37.5, -64.9519, 0) + (-125 cos(t), -216.5064 cos(t), 250 sin(t))

a : Sphere(C1, largoBrazo)
→ (x - 169.2)² + (y + 45)² + (z + 372)² = 48400

b : Sphere(C2, largoBrazo)
→ (x - 71.7)² + (y - 11.2917)² + (z + 372)² = 48400

l : Sphere(C3, largoBrazo)
→ (x - 71.7)² + (y + 101.2917)² + (z + 372)² = 48400

I1 = (x(C1), y(O), z(C1))
→ (169.2, 0, -372)

I2 = (x(C2) − sin(120°) (sin(120°) (x(C2) − x(O)) + cos(120°) (y(O) − y(C2))), y(C2) + cos(120°) (sin(120°) (x(C2) − x(O)) + cos(120°) (y(O) − y(C2))), z(C2
→ (13.0356, -22.5783, -372)

I3 = (x(C3) − sin(240°) (sin(240°) (x(C3) − x(O)) + cos(240°) (y(O) − y(C3))), y(C3) + cos(240°) (sin(240°) (x(C3) − x(O)) + cos(240°) (y(O) − y(C3))), z(C3
→ (-25.9356, -44.9217, -372)

c₁ : IntersectPath(p, a)
→ X = (169.2, 0, -372) + (-215.3486 cos(t), 0, -215.3486 sin(t))
```

Figura 24: Modelo con todos los elementos ocultos excepto configuración final del robot



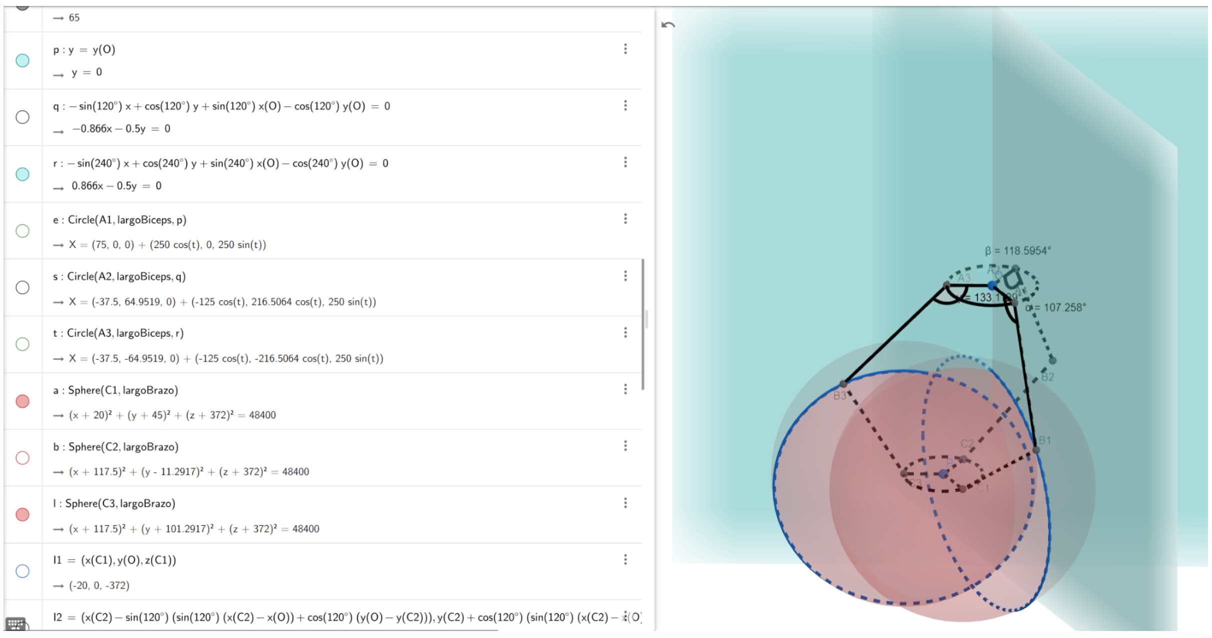
```
→ (169.2, -45, -372)

C2 = P + (radioEfector cos(120°), radioEfector sin(120°), 0)
→ (71.7, 11.2917, -372)

C3 = P + (radioEfector cos(240°), radioEfector sin(240°), 0)
→ (71.7, -101.2917, -372)

d : Circle(C2, C3, C1)
→ X = (104.2, -45, -372) + (65 sin(t), -65 cos(t), 0)

i = Segment(P, C1)
→ 65

j = Segment(P, C2)
→ 65

k = Segment(P, C3)
→ 65

p : y = y(O)
→ y = 0

q : − sin(120°) x + cos(120°) y + sin(120°) x(O) − cos(120°) y(O) = 0
→ −0.866x − 0.5y = 0

r : − sin(240°) x + cos(240°) y + sin(240°) x(O) − cos(240°) y(O) = 0
→ 0.866x − 0.5y = 0

e : Circle(A1, largoBiceps, p)
→ X = (75, 0, 0) + (250 cos(t), 0, 250 sin(t))

s : Circle(A2, largoBiceps, q)
```

Figura 25: Intersección entre las circunferencias que dan lugar a la posición del codo



```
PX = -85
-300 ———●——— 300
PY = -45
-300 ———●——— 300
PZ = -372
-600 ———●——— 0
radioBase = 75
0 ———●——— 100
radioEfector = 65
0 ———●——— 100
largoBiceps = 250
0 ———●——— 500
largoBrazo = 220
0 ———●——— 500
O = (0, 0, 0)
P = (PX, PY, PZ)
→ (-85, -45, -372)
A1 = O + (radioBase, 0, 0)
→ (75, 0, 0)
A2 = O + (radioBase cos(120°), radioBase sin(120°), 0)
→ (-37.5, 64.9519, 0)
A3 = O + (radioBase cos(240°), radioBase sin(240°), 0)
→ (-37.5, -64.9519, 0)
c : Circle(A2, A3, A1)
```

Figura 26: Misma vista pero variando los controles deslizantes para la posición



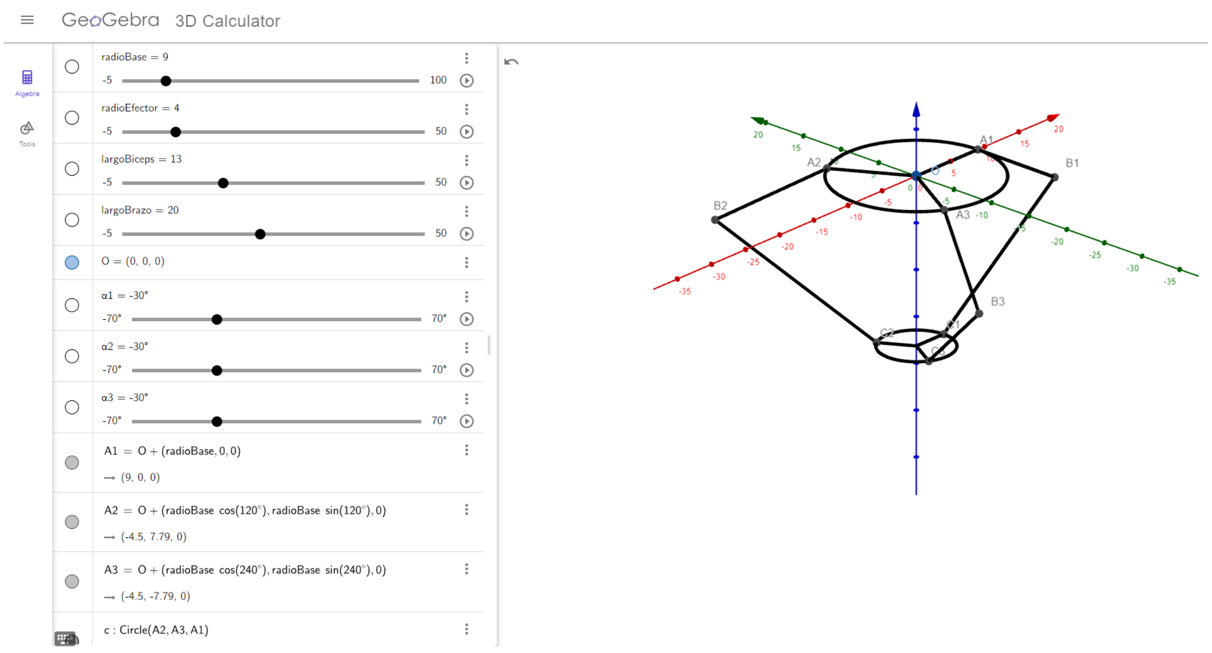
```
→ 65
p : y = y(O)
→ y = 0
q : − sin(120°) x + cos(120°) y + sin(120°) x(O) − cos(120°) y(O) = 0
→ −0.866x − 0.5y = 0
r : − sin(240°) x + cos(240°) y + sin(240°) x(O) − cos(240°) y(O) = 0
→ 0.866x − 0.5y = 0
e : Circle(A1, largoBiceps, p)
→ X = (75, 0, 0) + (250 cos(t), 0, 250 sin(t))
s : Circle(A2, largoBiceps, q)
→ X = (-37.5, 64.9519, 0) + (-125 cos(t), 216.5064 cos(t), 250 sin(t))
t : Circle(A3, largoBiceps, r)
→ X = (-37.5, -64.9519, 0) + (-125 cos(t), -216.5064 cos(t), 250 sin(t))
a : Sphere(C1, largoBrazo)
→ (x + 20)² + (y + 45)² + (z + 372)² = 48400
b : Sphere(C2, largoBrazo)
→ (x + 117.5)² + (y - 11.2917)² + (z + 372)² = 48400
l : Sphere(C3, largoBrazo)
→ (x + 117.5)² + (y + 101.2917)² + (z + 372)² = 48400
l1 = (x(C1), y(O), z(C1))
→ (-20, 0, -372)
l2 = (x(C2) − sin(120°) (sin(120°) (x(C2) − x(O)) + cos(120°) (y(O) − y(C2))), y(C2) + cos(120°) (sin(120°) (x(C2) − x(O)
```

Figura 27: Circunferencias de intersección entre esferas y planos

Como podemos observar analizando las Fig. 23-27, es posible obtener una visión detallada de cualquiera de los pasos geométricos necesarios para la resolución de la cinemática inversa. La posibilidad de modificar los parámetros de construcción del robot o la posición del punto final a la vez que se visualizan estas acciones permite un mejor entendimiento del funcionamiento del robot.

5.3. Modelo cinemático directo

Se ha realizado el modelo de la cinemática inversa del robot Delta en Geogebra siguiendo el proceso matemático descrito en el capítulo 3.



radioBase = 9
-5 100

radioEfector = 4
-5 50

largoBiceps = 13
-5 50

largoBrazo = 20
-5 50

O = (0, 0, 0)

α1 = -30°
-70° 70°

α2 = -30°
-70° 70°

α3 = -30°
-70° 70°

A1 = O + (radioBase, 0, 0)
→ (9, 0, 0)

A2 = O + (radioBase cos(120°), radioBase sin(120°), 0)
→ (-4.5, 7.79, 0)

A3 = O + (radioBase cos(240°), radioBase sin(240°), 0)
→ (-4.5, -7.79, 0)

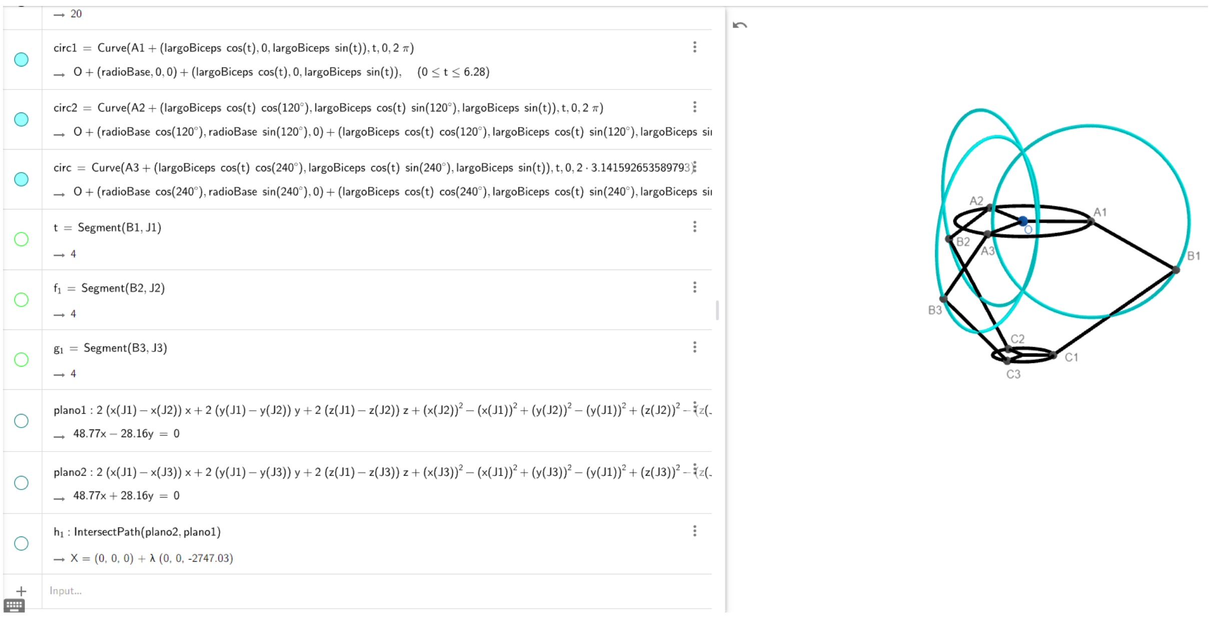
c : Circle(A2, A3, A1)

Figura 28: Visión general del modelo cinemático directo

En la Fig. 28 podemos observar la visión general del modelo cinemático directo. Al igual que en el modelo para la cinemática inversa, en el menú de ecuaciones de la izquierda, encabezando las 49 entradas matemáticas presentes en el modelo, podemos encontrar los distintos parámetros presentados en forma de controles deslizantes que pueden ser modificados por el usuario.

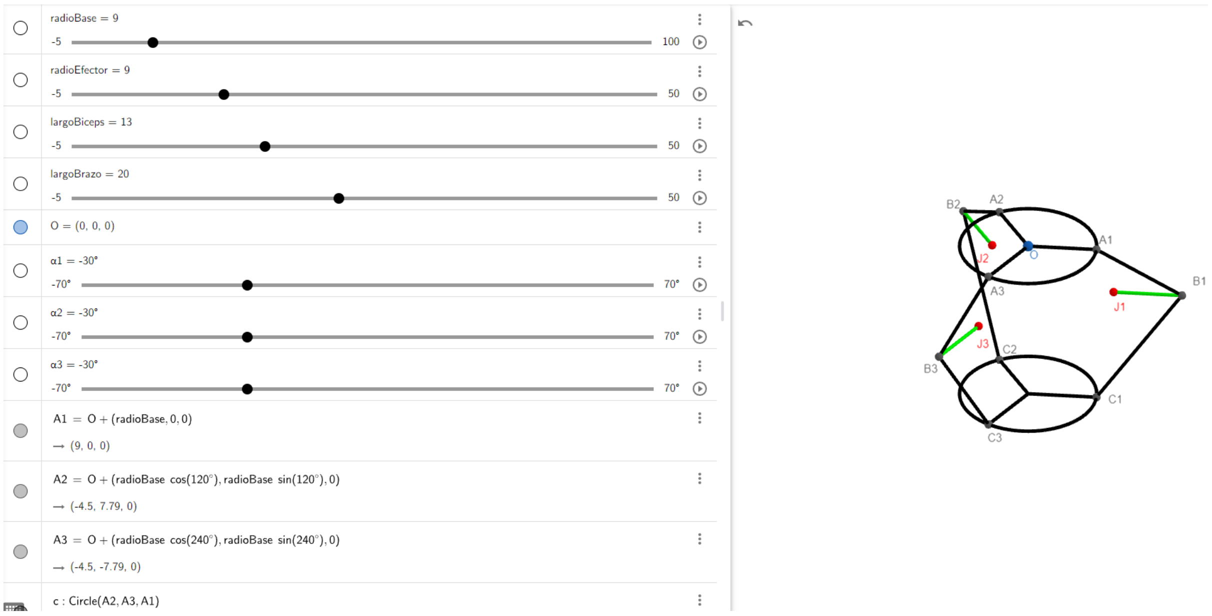
Es posible escoger una configuración para los elementos del robot Delta y unos ángulos con la horizontal de cada uno de los tres brazos del robot, introducidos por medio de los controles deslizantes para obtener, tanto en la vista 3D de la calculadora gráfica como en la entrada correspondiente del menú lateral, la posición final de la base del efector final del robot, así como la configuración de cada uno de sus elementos.

A continuación se muestran múltiples vistas de diferentes configuraciones del robot con distintos elementos mostrados para dar una idea general de la conveniencia que supone para la enseñanza este modelo de la cinemática directa del robot Delta.



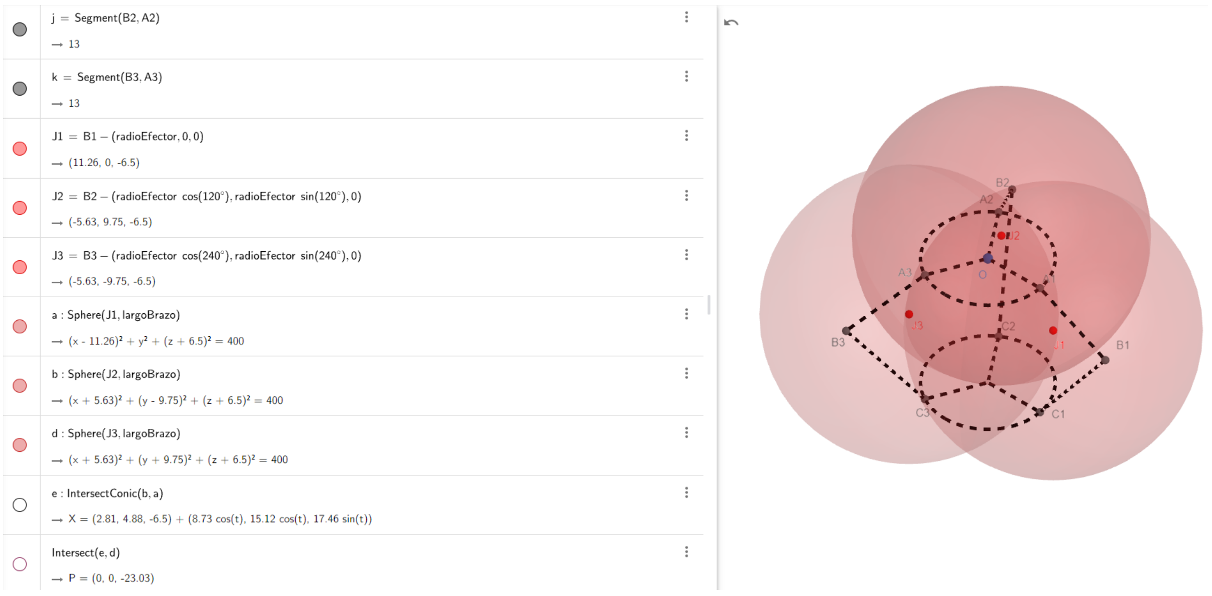
```
→ 20
circ1 = Curve(A1 + (largoBiceps cos(t), 0, largoBiceps sin(t)), t, 0, 2 π)
→ O + (radioBase, 0, 0) + (largoBiceps cos(t), 0, largoBiceps sin(t)),    (0 ≤ t ≤ 6.28)
circ2 = Curve(A2 + (largoBiceps cos(t) cos(120°), largoBiceps cos(t) sin(120°), largoBiceps sin(t)), t, 0, 2 π)
→ O + (radioBase cos(120°), radioBase sin(120°), 0) + (largoBiceps cos(t) cos(120°), largoBiceps cos(t) sin(120°), largoBiceps si
circ = Curve(A3 + (largoBiceps cos(t) cos(240°), largoBiceps cos(t) sin(240°), largoBiceps sin(t)), t, 0, 2 · 3.141592653589793)
→ O + (radioBase cos(240°), radioBase sin(240°), 0) + (largoBiceps cos(t) cos(240°), largoBiceps cos(t) sin(240°), largoBiceps si
t = Segment(B1, J1)
→ 4
f₁ = Segment(B2, J2)
→ 4
g₁ = Segment(B3, J3)
→ 4
plano1 : 2 (x(J1) − x(J2)) x + 2 (y(J1) − y(J2)) y + 2 (z(J1) − z(J2)) z + (x(J2))² − (x(J1))² + (y(J2))² − (y(J1))² + (z(J2))² − (z(
→ 48.77x − 28.16y = 0
plano2 : 2 (x(J1) − x(J3)) x + 2 (y(J1) − y(J3)) y + 2 (z(J1) − z(J3)) z + (x(J3))² − (x(J1))² + (y(J3))² − (y(J1))² + (z(J3))² − (z(
→ 48.77x + 28.16y = 0
h₁ : IntersectPath(plano2, plano1)
→ X = (0, 0, 0) + λ (0, 0, -2747.03)
Input...
```

Figura 29: Circunferencias que describen el movimiento de los bíceps



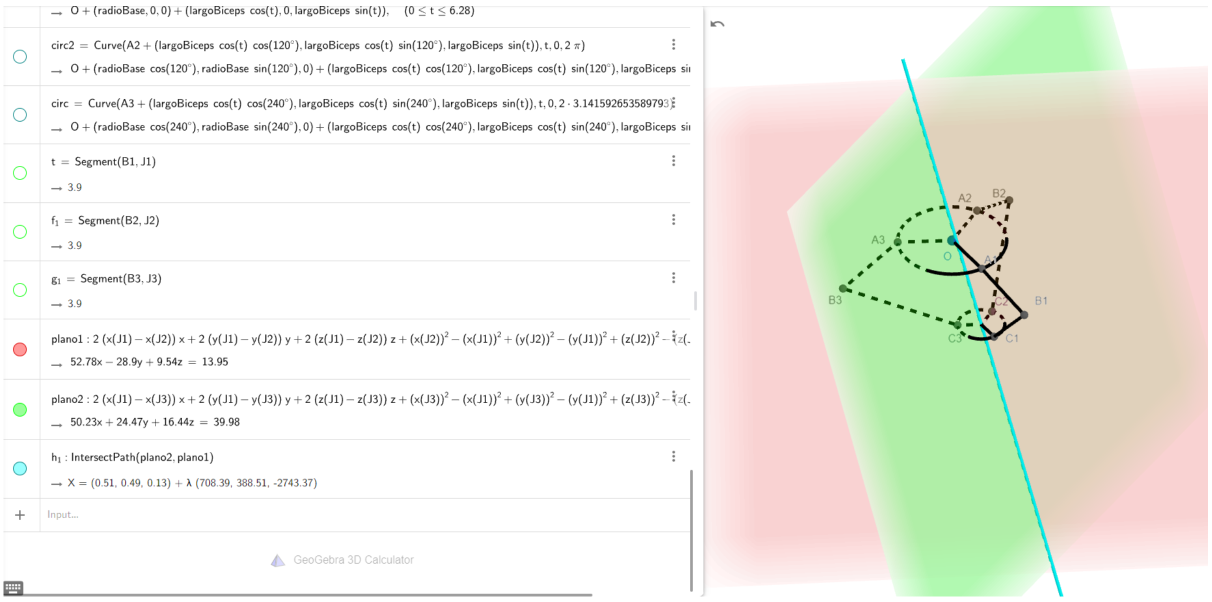
```
radioBase = 9
-5                100
radioEfector = 9
-5                50
largoBiceps = 13
-5                50
largoBrazo = 20
-5                50
O = (0, 0, 0)
α1 = -30°
-70°              70°
α2 = -30°
-70°              70°
α3 = -30°
-70°              70°
A1 = O + (radioBase, 0, 0)
→ (9, 0, 0)
A2 = O + (radioBase cos(120°), radioBase sin(120°), 0)
→ (-4.5, 7.79, 0)
A3 = O + (radioBase cos(240°), radioBase sin(240°), 0)
→ (-4.5, -7.79, 0)
c : Circle(A2, A3, A1)
```

Figura 30: Centros de las esferas cuando el radio del efector final es muy grande



```
j = Segment(B2, A2)
→ 13
k = Segment(B3, A3)
→ 13
J1 = B1 − (radioEfector, 0, 0)
→ (11.26, 0, -6.5)
J2 = B2 − (radioEfector cos(120°), radioEfector sin(120°), 0)
→ (-5.63, 9.75, -6.5)
J3 = B3 − (radioEfector cos(240°), radioEfector sin(240°), 0)
→ (-5.63, -9.75, -6.5)
a : Sphere(J1, largoBrazo)
→ (x - 11.26)² + y² + (z + 6.5)² = 400
b : Sphere(J2, largoBrazo)
→ (x + 5.63)² + (y - 9.75)² + (z + 6.5)² = 400
d : Sphere(J3, largoBrazo)
→ (x + 5.63)² + (y + 9.75)² + (z + 6.5)² = 400
e : IntersectConic(b, a)
→ X = (2.81, 4.88, -6.5) + (8.73 cos(t), 15.12 cos(t), 17.46 sin(t))
Intersect(e, d)
→ P = (0, 0, -23.03)
```

Figura 31: Esferas cuya intersección dan la posición del centro de la base de efector final



```
→ O + (radioBase, 0, 0) + (largoBiceps cos(t), 0, largoBiceps sin(t)),   (0 ≤ t ≤ 6.28)
circ2 = Curve(A2 + (largoBiceps cos(t) cos(120°), largoBiceps cos(t) sin(120°), largoBiceps sin(t)), t, 0, 2 π)
→ O + (radioBase cos(120°), radioBase sin(120°), 0) + (largoBiceps cos(t) cos(120°), largoBiceps cos(t) sin(120°), largoBiceps si
circ = Curve(A3 + (largoBiceps cos(t) cos(240°), largoBiceps cos(t) sin(240°), largoBiceps sin(t)), t, 0, 2 · 3.141592653589793
→ O + (radioBase cos(240°), radioBase sin(240°), 0) + (largoBiceps cos(t) cos(240°), largoBiceps cos(t) sin(240°), largoBiceps si
t = Segment(B1, J1)
→ 3.9
f₁ = Segment(B2, J2)
→ 3.9
g₁ = Segment(B3, J3)
→ 3.9
plano1 : 2 (x(J1) − x(J2)) x + 2 (y(J1) − y(J2)) y + 2 (z(J1) − z(J2)) z + (x(J2))² − (x(J1))² + (y(J2))² − (y(J1))² + (z(J2))² − (z(
→ 52.78x − 28.9y + 9.54z = 13.95
plano2 : 2 (x(J1) − x(J3)) x + 2 (y(J1) − y(J3)) y + 2 (z(J1) − z(J3)) z + (x(J3))² − (x(J1))² + (y(J3))² − (y(J1))² + (z(J3))² − (z(
→ 50.23x + 24.47y + 16.44z = 39.98
h₁ : IntersectPath(plano2, plano1)
→ X = (0.51, 0.49, 0.13) + λ (708.39, 388.51, -2743.37)
Input...
GeoGebra 3D Calculator
```

Figura 32: Recta que contiene el punto final resultante de la intersección de los planos

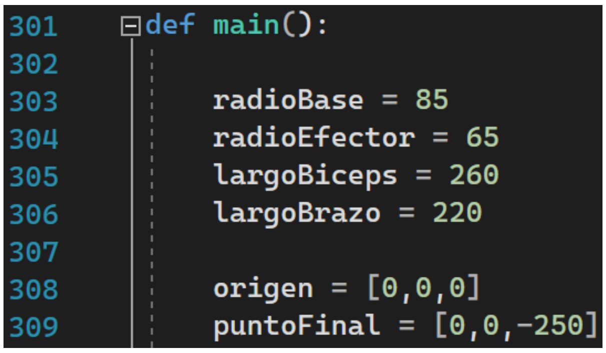
Como podemos observar analizando las Fig. 28-32, es posible obtener una visión detallada de cualquiera de los pasos geométricos necesarios para la resolución de la cinemática directa. La posibilidad de modificar los parámetros de construcción del robot o la posición del punto final a la vez que se visualizan estas acciones permite un mejor entendimiento del funcionamiento del robot.

Capítulo 6: Herramienta de python y determinación de dimensiones del prototipo

Una de las características más importantes que puede tener un robot es el espacio de trabajo que posee. Este espacio de trabajo viene determinado por las dimensiones de cada uno de los elementos que componen el robot. La herramienta desarrollada en python es capaz de, introduciendo unos valores para cada elemento del robot Delta, devolver el espacio de trabajo que es capaz de alcanzar el robot.

6.1. Visión general

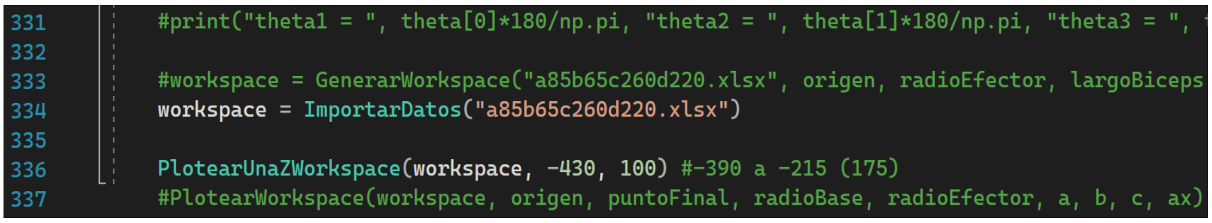
La utilización de esta herramienta es muy sencilla. Es necesario modificar los parámetros que componen el robot, elegir si se desea generar el espacio de trabajo o importarlo y elegir si representar el espacio de trabajo en tres dimensiones o escoger solo una altura "z" y obtener su representación en dos dimensiones. Adicionalmente, se puede introducir un punto final para obtener el valor de las variables articulares.



```
def main():

    radioBase = 85
    radioEfector = 65
    largoBiceps = 260
    largoBrazo = 220

    origen = [0,0,0]
    puntoFinal = [0,0,-250]
```

Figura 33: Lugar en el que modificar los parámetros del robot

Los parámetros del robot se pueden modificar en el principio del main. Además, también es posible modificar en este lugar el origen del robot, que corresponde al centro de la base superior del mismo. Por último se puede escoger un punto final para el que resolver la cinemática inversa y obtener las variables de actuación.



```
    #print("theta1 = ", theta[0]*180/np.pi, "theta2 = ", theta[1]*180/np.pi, "theta3 = ",

    #workspace = GenerarWorkspace("a85b65c260d220.xlsx", origen, radioEfector, largoBiceps
    workspace = ImportarDatos("a85b65c260d220.xlsx")

    PlotearUnaZWorkspace(workspace, -430, 100) #-390 a -215 (175)
    #PlotearWorkspace(workspace, origen, puntoFinal, radioBase, radioEfector, a, b, c, ax)
```

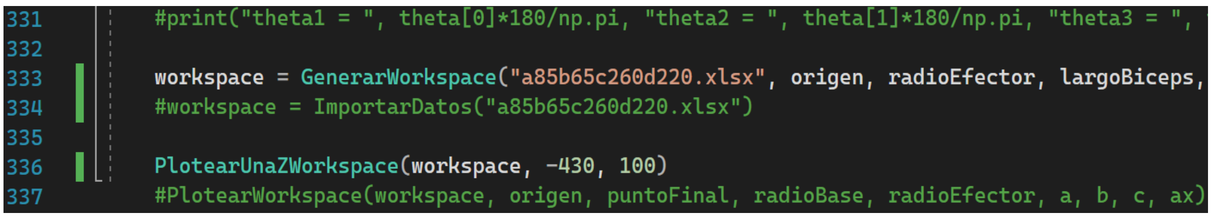
Figura 34: Lugar en el que se elige la funcionalidad del programa

En la Fig. 34 observamos el lugar en el que se elige la funcionalidad del programa. Se puede escoger si recibir la información de los ángulos para los que se resuelve la cinemática inversa comentando o descomentando la línea 331.

Si se desea generar el espacio de trabajo para las dimensiones introducidas se debe descomentar la línea 333 y comentar la 334. Por el contrario, si se quiere importar los datos desde un archivo con formato Excel se debe hacer lo opuesto, es decir, comentar la línea 333 y descomentar la 334.

En la Fig. 34 encontramos una configuración del programa que importará los datos desde un archivo Excel. Para elegir el nombre del archivo a importar será necesario cambiar el nombre entre comillas de la línea 334.

De la misma manera se puede cambiar el nombre del archivo que se generará al crear un nuevo espacio de trabajo modificando el nombre que aparece entre comillas en la línea 333. Se expone un ejemplo de configuración del programa que genera un nuevo espacio de trabajo en vez de importarlo en la Fig. 35.



```
331         #print("theta1 = ", theta[0]*180/np.pi, "theta2 = ", theta[1]*180/np.pi, "theta3 = ",
332
333         workspace = GenerarWorkspace("a85b65c260d220.xlsx", origen, radioEfector, largoBiceps,
334         #workspace = ImportarDatos("a85b65c260d220.xlsx")
335
336         PlotearUnaZWorkspace(workspace, -430, 100)
337         #PlotearWorkspace(workspace, origen, puntoFinal, radioBase, radioEfector, a, b, c, ax)
```

Figura 35: Generar un nuevo espacio de trabajo

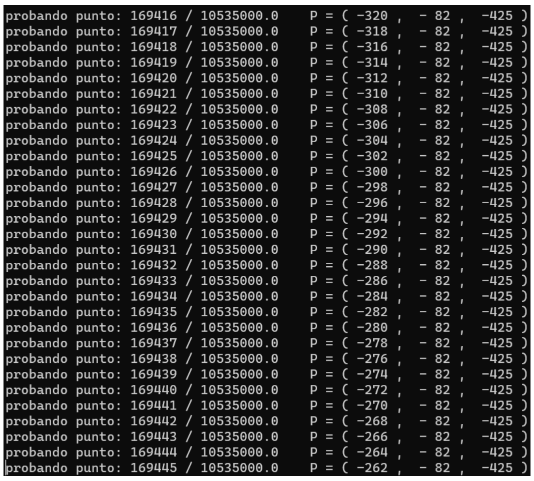
Por último, para elegir si representar en tres dimensiones el espacio de trabajo se puede comentar o descomentar la línea 337. Para escoger si observar una altura “z” en concreto y su correspondiente proyección en dos dimensiones se debe comentar o descomentar la línea 336. En los parámetros de esta función se puede introducir la altura “z” que se quiere observar modificando el segundo parámetro.

Por otra parte, se puede modificar el tercer parámetro que define el radio de la circunferencia a pintar en el centro del espacio de trabajo correspondiente a esa altura “z”. Esta circunferencia sirve para identificar las dimensiones máximas de un cilindro dentro del espacio de trabajo, útil para aplicaciones como impresión 3D.

El código empleado para desarrollar esta herramienta se encuentra en el Anexo I al final de este documento.

6.2. Resultados proporcionados por la herramienta

Una vez elegidas las dimensiones del robot el programa comenzará a probar puntos en el espacio para determinar si existe una solución posible para la cinemática inversa que dé lugar a esa posición final.



```
probando punto: 169416 / 10535000.0    P = ( -320 ,  - 82 ,  -425 )
probando punto: 169417 / 10535000.0    P = ( -318 ,  - 82 ,  -425 )
probando punto: 169418 / 10535000.0    P = ( -316 ,  - 82 ,  -425 )
probando punto: 169419 / 10535000.0    P = ( -314 ,  - 82 ,  -425 )
probando punto: 169420 / 10535000.0    P = ( -312 ,  - 82 ,  -425 )
probando punto: 169421 / 10535000.0    P = ( -310 ,  - 82 ,  -425 )
probando punto: 169422 / 10535000.0    P = ( -308 ,  - 82 ,  -425 )
probando punto: 169423 / 10535000.0    P = ( -306 ,  - 82 ,  -425 )
probando punto: 169424 / 10535000.0    P = ( -304 ,  - 82 ,  -425 )
probando punto: 169425 / 10535000.0    P = ( -302 ,  - 82 ,  -425 )
probando punto: 169426 / 10535000.0    P = ( -300 ,  - 82 ,  -425 )
probando punto: 169427 / 10535000.0    P = ( -298 ,  - 82 ,  -425 )
probando punto: 169428 / 10535000.0    P = ( -296 ,  - 82 ,  -425 )
probando punto: 169429 / 10535000.0    P = ( -294 ,  - 82 ,  -425 )
probando punto: 169430 / 10535000.0    P = ( -292 ,  - 82 ,  -425 )
probando punto: 169431 / 10535000.0    P = ( -290 ,  - 82 ,  -425 )
probando punto: 169432 / 10535000.0    P = ( -288 ,  - 82 ,  -425 )
probando punto: 169433 / 10535000.0    P = ( -286 ,  - 82 ,  -425 )
probando punto: 169434 / 10535000.0    P = ( -284 ,  - 82 ,  -425 )
probando punto: 169435 / 10535000.0    P = ( -282 ,  - 82 ,  -425 )
probando punto: 169436 / 10535000.0    P = ( -280 ,  - 82 ,  -425 )
probando punto: 169437 / 10535000.0    P = ( -278 ,  - 82 ,  -425 )
probando punto: 169438 / 10535000.0    P = ( -276 ,  - 82 ,  -425 )
probando punto: 169439 / 10535000.0    P = ( -274 ,  - 82 ,  -425 )
probando punto: 169440 / 10535000.0    P = ( -272 ,  - 82 ,  -425 )
probando punto: 169441 / 10535000.0    P = ( -270 ,  - 82 ,  -425 )
probando punto: 169442 / 10535000.0    P = ( -268 ,  - 82 ,  -425 )
probando punto: 169443 / 10535000.0    P = ( -266 ,  - 82 ,  -425 )
probando punto: 169444 / 10535000.0    P = ( -264 ,  - 82 ,  -425 )
probando punto: 169445 / 10535000.0    P = ( -262 ,  - 82 ,  -425 )
```

Figura 36: Programa generando el espacio de trabajo del robot

Una vez terminado el proceso, se guardará el espacio de trabajo en formato Excel con el nombre introducido. Si hemos elegido observar el espacio representado en tres dimensiones se nos abrirá una ventana como la que se muestra en la Fig. 37.

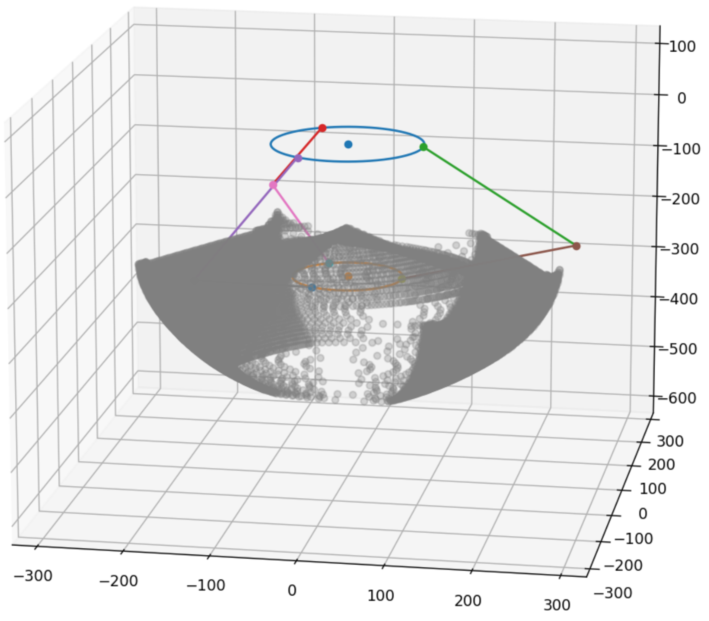


Figura 37: Representación en tres dimensiones del espacio de trabajo

Por otro lado, si hemos escogido observar una altura “z” en concreto y su correspondiente proyección en dos dimensiones obtendremos una ventana con la información que se muestra en la Fig. 38.

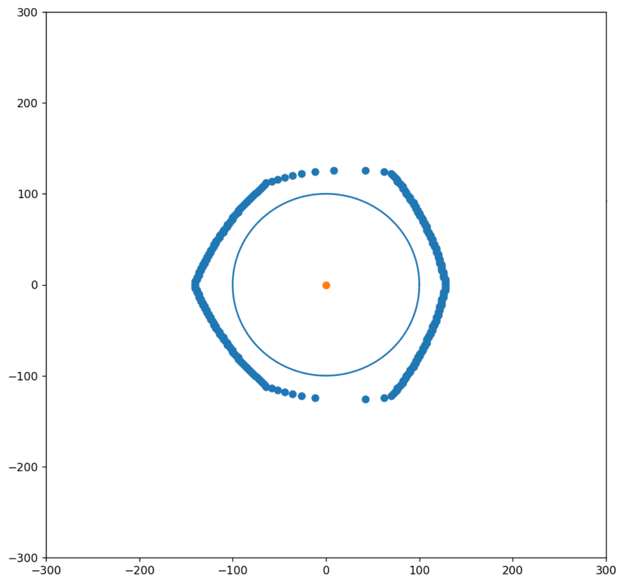


Figura 38: Representación en dos dimensiones de una altura del espacio de trabajo

En la Fig. 38 podemos observar los límites del espacio de trabajo en la altura “z” elegida y una circunferencia con el radio que hayamos introducido. De esta representación se extrae que, si quisiéramos obtener un cilindro de espacio de trabajo para impresión 3D, en esta altura sería posible.

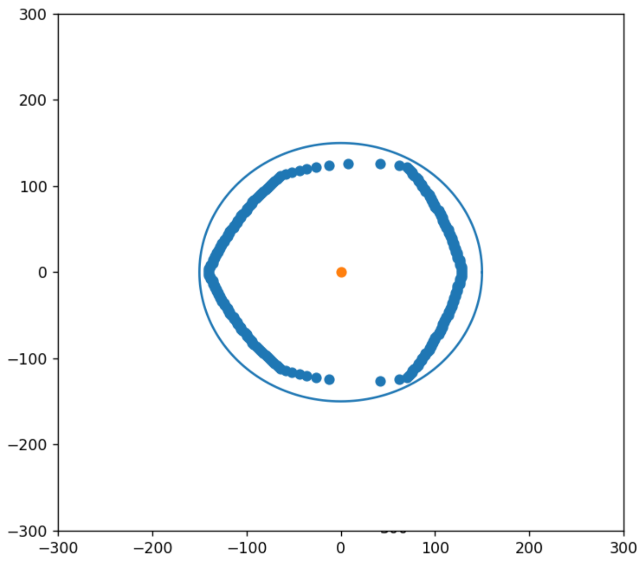
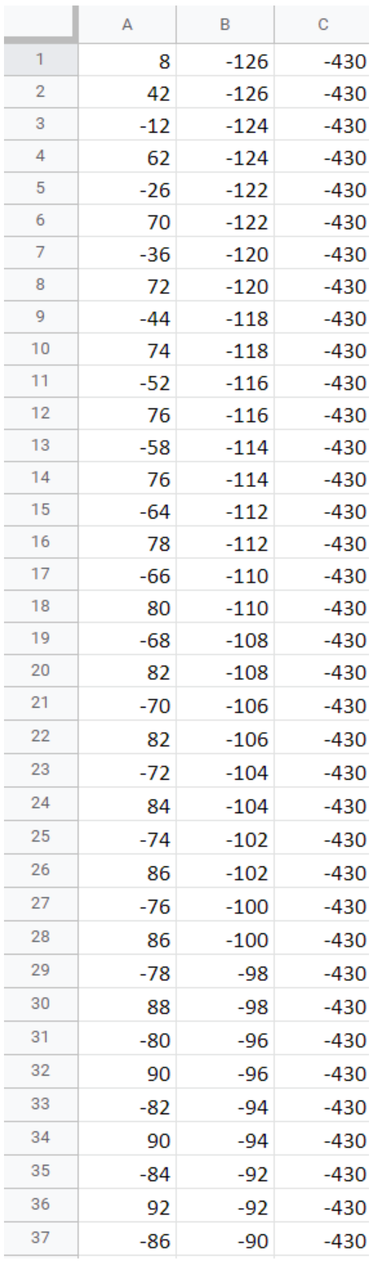


Figura 39: Representación en dos dimensiones de una altura del espacio de trabajo

Por el contrario, en la Fig.39 observamos un espacio de trabajo en el que no cabría un cilindro con el radio introducido para la altura “z” elegida, por lo que deberemos modificar las dimensiones del robot o elegir un cilindro más pequeño que esté dentro de los límites del espacio de trabajo.

Por último, al generar el espacio de trabajo se exportan automáticamente a formato Excel los puntos límites del espacio de trabajo. Cada fila corresponde a un punto y cada columna a una componente del punto, siendo la columna A la componente “x”, la columna B la componente “y” y la columna C la componente “z”.



	A	B	C
1	8	-126	-430
2	42	-126	-430
3	-12	-124	-430
4	62	-124	-430
5	-26	-122	-430
6	70	-122	-430
7	-36	-120	-430
8	72	-120	-430
9	-44	-118	-430
10	74	-118	-430
11	-52	-116	-430
12	76	-116	-430
13	-58	-114	-430
14	76	-114	-430
15	-64	-112	-430
16	78	-112	-430
17	-66	-110	-430
18	80	-110	-430
19	-68	-108	-430
20	82	-108	-430
21	-70	-106	-430
22	82	-106	-430
23	-72	-104	-430
24	84	-104	-430
25	-74	-102	-430
26	86	-102	-430
27	-76	-100	-430
28	86	-100	-430
29	-78	-98	-430
30	88	-98	-430
31	-80	-96	-430
32	90	-96	-430
33	-82	-94	-430
34	90	-94	-430
35	-84	-92	-430
36	92	-92	-430
37	-86	-90	-430

Figura 40: Datos exportados a formato Excel

6.3. Determinación de las dimensiones del prototipo

Para determinar las dimensiones del prototipo es necesario conocer las restricciones que van a estar presentes como el radio de la base del robot o el radio de la base del efector final del mismo. En nuestro caso, deseamos introducir una polea de gran diámetro en los puntos de anclaje de los bíceps en la base superior del robot para ayudar con la precisión de los movimientos del mismo y mejorar la capacidad de carga. Por tanto, teniendo en cuenta que queremos utilizar una polea de diámetro muy parecido a 150 mm, el radio de la base superior lo fijamos en 90 mm. Esto se debe a que las poleas deben caber en el centro de la base sin chocarse entre sí con cierta holgura.

Por otro lado, se ha diseñado la base del efector final para ser perfectamente compatible con el cabezal de impresión 3D de la marca “Creality 3D” y, debido a esto, el radio del efector final deberá ser 65 mm.

Por último, para que el robot sea útil en trabajos de impresión 3D o cualquier otro tipo de manipulaciones hemos fijado como objetivo un espacio de trabajo en el que se pueda introducir un cilindro de radio 100 mm y altura superior a 150 mm.

Teniendo en cuenta todas estas restricciones para el diseño, hemos utilizado la herramienta para encontrar una configuración de brazos que nos permita obtener el espacio de trabajo deseado. El resultado ha sido un bíceps de 250 mm y un antebrazo de 220 mm. Podemos observar el espacio de trabajo para esta configuración en la Fig. 41.

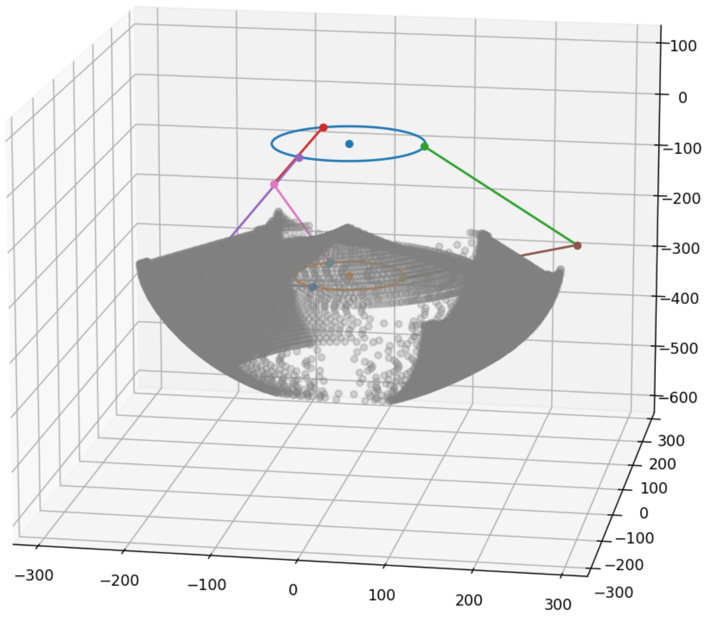


Figura 41: Espacio de trabajo del prototipo representado en tres dimensiones

Por otra parte, para confirmar que es posible introducir un cilindro de las dimensiones deseadas en el espacio de trabajo podemos analizar la altura más baja y la altura más alta en la que es posible introducir una circunferencia del radio deseado.

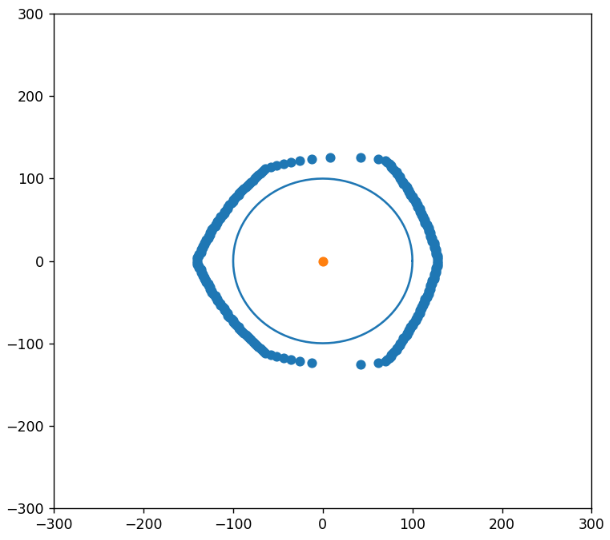


Figura 42: Espacio de trabajo del prototipo en la altura -430 mm

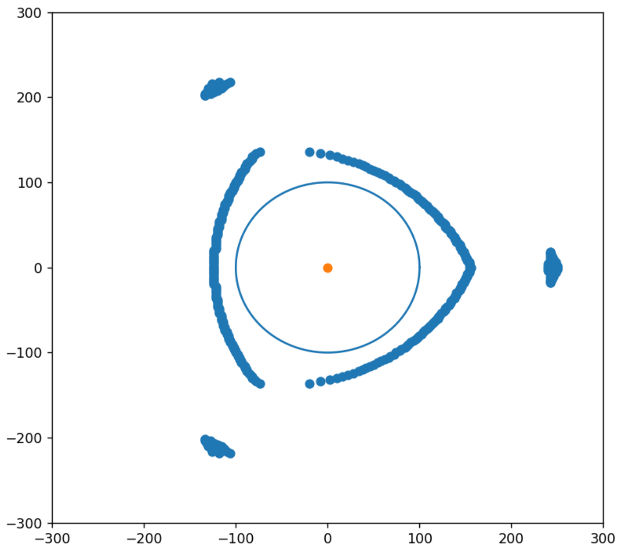


Figura 43: Espacio de trabajo del prototipo en la altura -240 mm

Observando las Fig 42 y 43 determinamos que, para un robot con las dimensiones dadas, es posible introducir un cilindro de radio 100 mm y altura 190 mm en el espacio de trabajo del robot. Además, hemos decidido elegir una configuración con un poco de margen de error ya que el paso desde el modelo ideal a un robot construído podría resultar en imperfecciones que reduzcan el espacio de trabajo.

Capítulo 7: Diseño de los elementos del robot

Todos los elementos que conforman el prototipo del robot Delta han sido diseñados con la premisa de que sean fácilmente imprimibles en 3D, sin necesidad de utilizar soportes y con el menor número de inconvenientes posible. Este diseño se ha llevado a cabo en el software Autodesk Fusion 360.

El robot irá atornillado a una estructura de madera. Los elementos que forman la estructura del prototipo y que se han diseñado e impreso en 3D son:

- 3x Anclaje de los motores
- 3x Anclaje de los bíceps
- 1x Base del efector final
- 3x Antebrazo
- 3x Parte derecha del bíceps
- 3x Parte izquierda del bíceps
- 3x Polea perteneciente al bíceps
- 3x Guarda de polea perteneciente al bíceps
- 6x Separador para bíceps
- 12x Separador para rótula
- 3x Soporte para el fin de carrera
    - Parte unida al soporte
    - Parte unida al fin de carrera
- 1x Soporte de la fuente de alimentación
- 4x Soporte para la pantalla LCD
- 4x Soporte para placa controladora

Aparte de estos elementos, se han empleado:

- Tornillos y tuercas varias
- 2x Plancha cuadrada de madera de 50x50x16 cm
- 3x Listón de madera de 5,5x3,5x62 cm
- 12x Junta de rótula M3

El diseño de cada uno de estos elementos se explica en los apartados siguientes.

7.1. Anclaje de los motores

Los motores deben estar anclados a la plancha de madera de manera que puedan conectarse por medio de correas a la polea perteneciente a los bíceps. Además, estas correas deben poder tensarse para el correcto funcionamiento de la cadena cinemática.

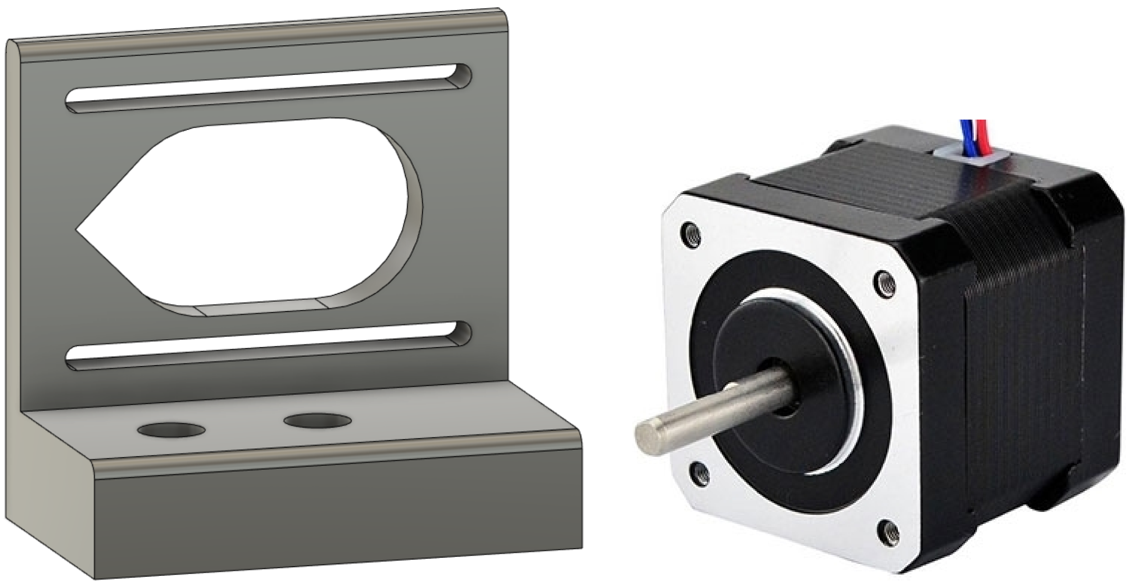


Figura 44: Anclaje de los motores. Bibliografía de imágenes [6]

Se ha diseñado el anclaje de los motores para ser anclado a la plancha de madera por medio de dos tornillos de 3,5 mm de diámetro a través de la base, quedando los tornillos embutidos en el anclaje sin sobresalir. Esto se debe a que el motor irá posicionado encima, apoyado en la base.

Por otro lado, se ha ideado un sistema en el que es posible tensar la correa que va desde el motor hasta la polea perteneciente al bíceps. Este sistema consiste en posicionar el motor en uno de los extremos laterales del soporte con la correa puesta y tirar de él horizontalmente hasta que quede tensa. Esto es posible gracias a los agujeros que recorren todo el soporte en vez de ser cuatro circunferencias para los tornillos, dando lugar al posible deslizamiento del motor en el soporte.

El pico que se puede observar a la izquierda en la Fig. 44 se debe a que la pieza se va a imprimir en 3D apoyada sobre el lateral derecho, de manera que los agujeros quedan posicionados de manera vertical. Esto es así porque no sería posible imprimir la pieza en horizontal al no tener soporte donde imprimir estos agujeros. Por tanto, es necesario que el ángulo con la vertical del agujero grande sea inferior a 45º para poder imprimirse correctamente. Si no se hubiera introducido el pico en el diseño, este requisito no se habría cumplido y no habría sido posible la impresión de la pieza, ya que el cabezal de impresión intentaría imprimir en el aire sin nada que pueda soportar esa parte de la pieza.

7.2. Anclaje de los bíceps

Los bíceps deben ir anclados a la plancha de madera por medio de unos soportes que dejen espacio suficiente para la polea y para el movimiento de los bíceps. Además, estos soportes deberán estar situados en la plancha de madera de manera que los centros de sus puntos de anclaje para el bíceps estén a 90 mm del centro del robot y dispuestos a 120º entre sí.

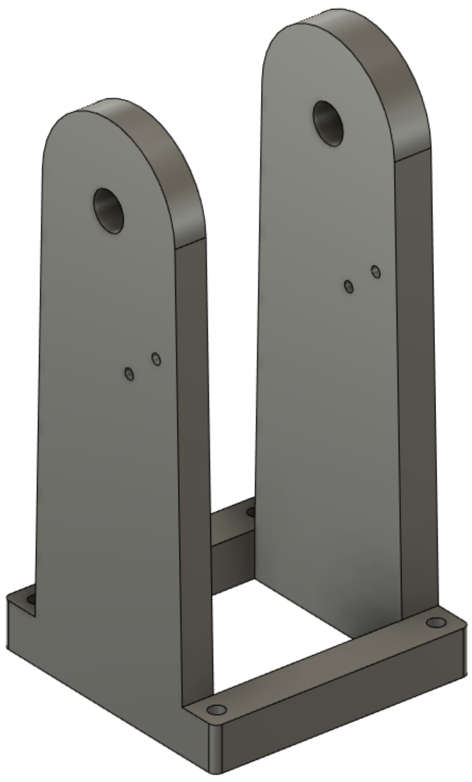


Figura 45: Anclaje de los bíceps

El eje de los bíceps se trata de un tornillo de 10 mm de diámetro, este eje atraviesa el anclaje y queda inmóvil gracias a una tuerca al otro lado del soporte. La altura a la que se encuentra este eje de la chapa de madera es 122.5 mm para que quepa con holgura la polea del bíceps.

Por otro lado, el anclaje de los bíceps se une a la plancha de madera por medio de cuatro tornillos. Adicionalmente, se han dispuesto dos agujeros en los laterales del soporte para poder situar el soporte de los fines de carrera.

7.3. Base del efector final

La base del efector final ha sido diseñada para ser compatible con el cabezal de impresión de la marca Creality presente en la mayoría de sus modelos económicos. El radio del efector final es de 65 mm y tiene 6 puntos de anclaje para los antebrazos dispuestos a 120º entre sí.

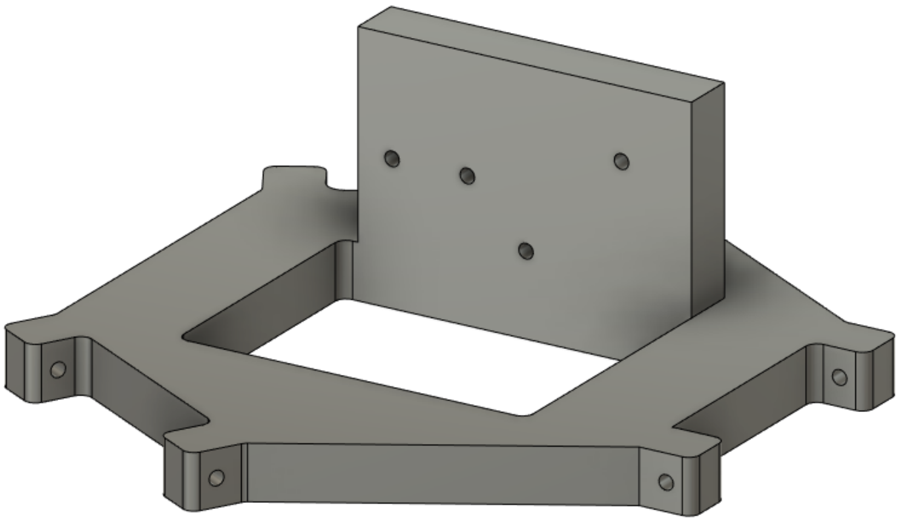


Figura 46: Base del efector final

Los agujeros que se pueden observar en la Fig. 46 corresponden con los necesarios para anclar el cabezal de impresión anteriormente mencionado. Sin embargo, al tratarse de un elemento que alcanza temperaturas muy altas para fundir el material que se va a imprimir (hasta más de 240ºC para el plástico ABS), sería necesaria una plancha de metal que impida que se funda el plástico de la base.

Los agujeros para el anclaje de los antebrazos han sido dimensionados un poco más pequeños que el tornillo que los va a unir. El tornillo será de 3 mm de diámetro y los agujeros son de 2,8 mm de diámetro para que el tornillo se abra paso y enrosque en el plástico.

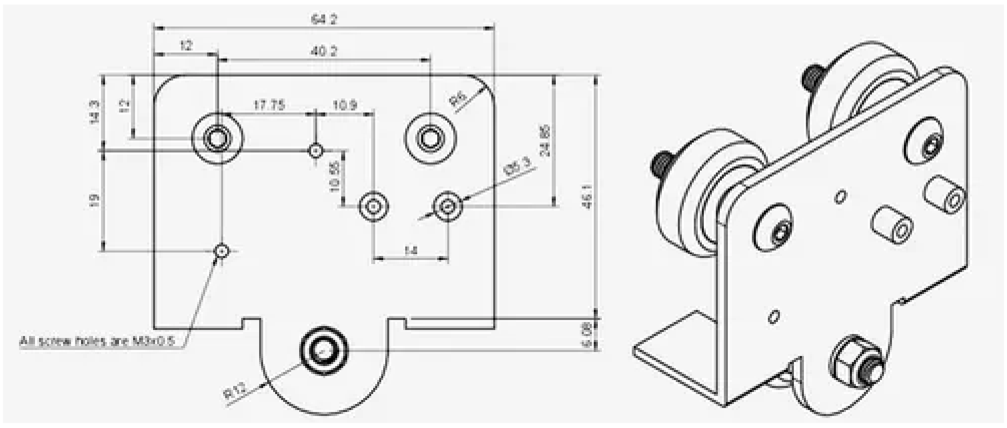


Figura 47: *Base del cabezal de impresión de Creality. Bibliografía de imágenes [7]*

7.4. Antebrazo

Los antebrazos deben ser elementos rígidos de largo 220 mm. no deben flexar y deben ser ligeros para poder alcanzar aceleraciones y velocidades altas.

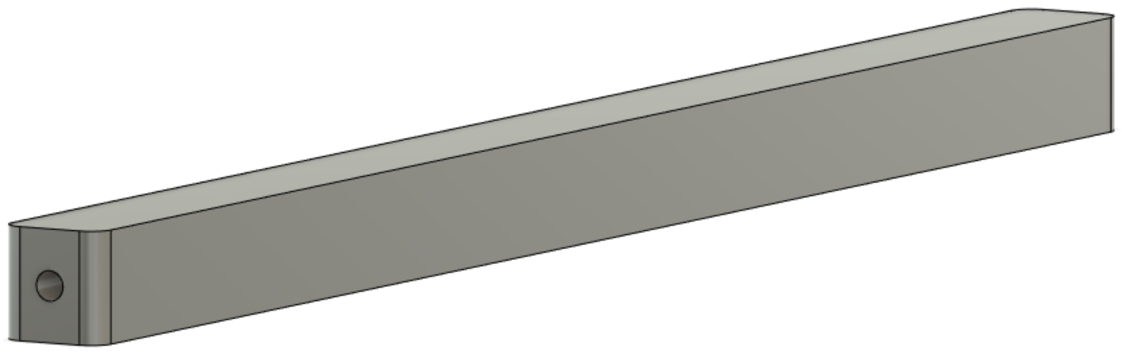


Figura 48: Antebrazo

Se ha decidido diseñar de forma cuadrada para poder imprimirse en horizontal en una impresora 3D. Si se hubiera optado por una forma cilíndrica habría sido necesario imprimir esta pieza de forma vertical al no poseer los soportes necesarios para una buena impresión. Además, las esquinas han sido redondeadas para evitar que se levanten al enfriarse el plástico en la cama de impresión.

Por otro lado, los agujeros para el anclaje de las rótulas han sido dimensionados un poco más pequeños que el tornillo que los va a unir. El tornillo será de 3 mm de diámetro y los agujeros son de 2,8 mm de diámetro para que el tornillo se abra paso y enrosque en el plástico.

7.5.1. Bíceps

El bíceps está compuesto por la parte derecha del bíceps, la polea, la guarda de polea y la parte izquierda del bíceps. Estas piezas quedan unidas entre sí por dos tornillos que las atraviesan a todas y son aseguradas con tuercas al otro extremo.

7.5.1. Parte derecha del bíceps

El bíceps tiene de largo entre el centro del eje de anclaje del bíceps en el soporte y el centro de los anclajes para los antebrazos exactamente 220 mm.

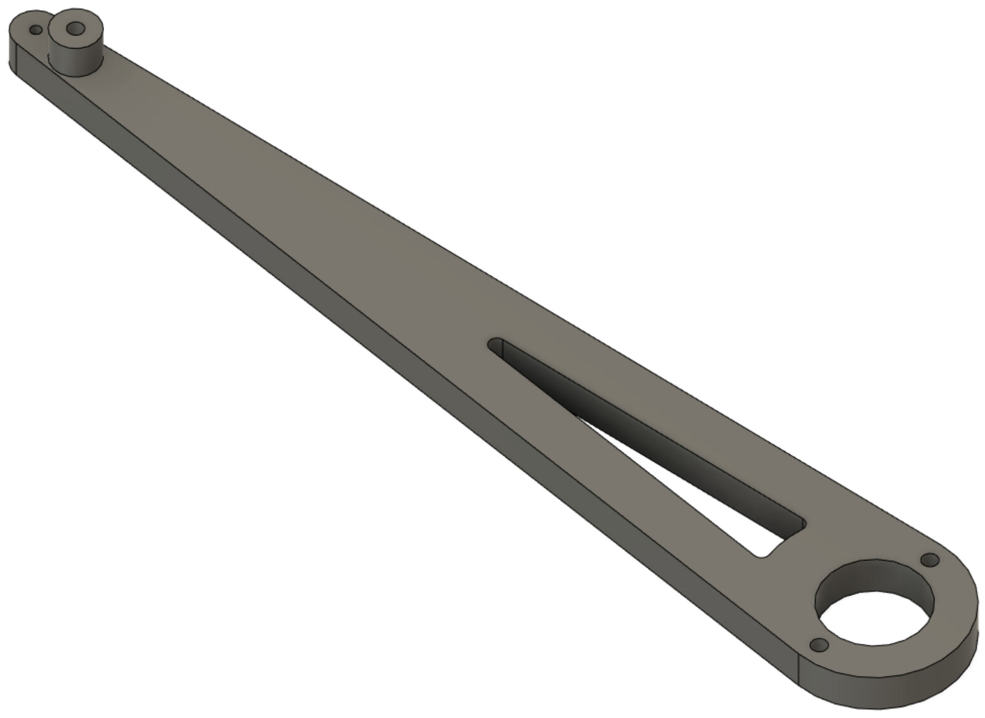


Figura 49: Parte derecha del bíceps

Tal y como se puede observar en la Fig. 49, se ha optado por quitar un poco de material en el centro del bíceps ya que añade peso y no provoca que la pieza sea más robusta, al no ser la parte más frágil del bíceps.

Por otra parte, se ha incluido un pequeño separador en el extremo de la parte derecha del bíceps, cuyo largo es exactamente el grosor de la polea sumado a la guarda de polea. Esto es para que, al ensamblar el bíceps, la pieza quede totalmente rectilínea.

Los agujeros de los tornillos presentes en la base del bíceps son de 4 mm de diámetro, para que los atraviese un tornillo de la misma medida. En el extremo del bíceps encontramos un agujero de 2.8 mm de diámetro cuyo tornillo será de 3 mm de diámetro para que el tornillo se abra paso y enrosque en el plástico. Por último, el agujero presente en el separador del bíceps es de 4 mm de diámetro para que lo atraviese un tornillo de la misma medida.

7.5.2. Parte izquierda del bíceps

Al igual que la parte derecha del bíceps, este elemento tiene de largo entre el centro del eje de anclaje del bíceps en el soporte y el centro de los anclajes para los antebrazos exactamente 220 mm.

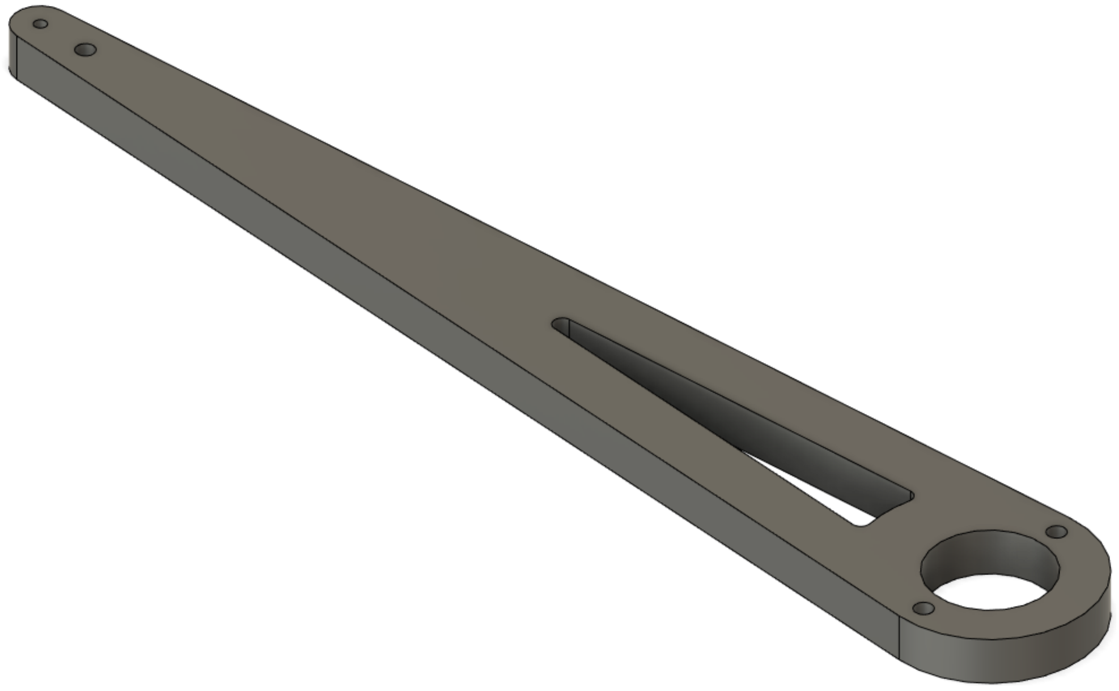


Figura 50: Parte izquierda del bíceps

Tal y como se puede observar en la Fig. 50, se ha procedido de la misma manera que con la parte derecha del bíceps y se ha optado por quitar un poco de material en el centro del bíceps ya que añade peso y no provoca que la pieza sea más robusta, al no ser la parte más frágil del bíceps.

Por el contrario, en la parte izquierda del bíceps no está presente el separador de los bíceps ya que solo es necesario que se encuentre en la parte derecha. Por último, todas las medidas de los agujeros presentes en esta pieza son de las mismas dimensiones que los presentes en la parte derecha del bíceps.

7.5.3. Polea perteneciente al bíceps

La polea perteneciente al bíceps consta de 235 dientes, dando lugar a una polea de diámetro 149.6 mm. Esto se debe a que está diseñada para engranar con una correa de tipo GT2.

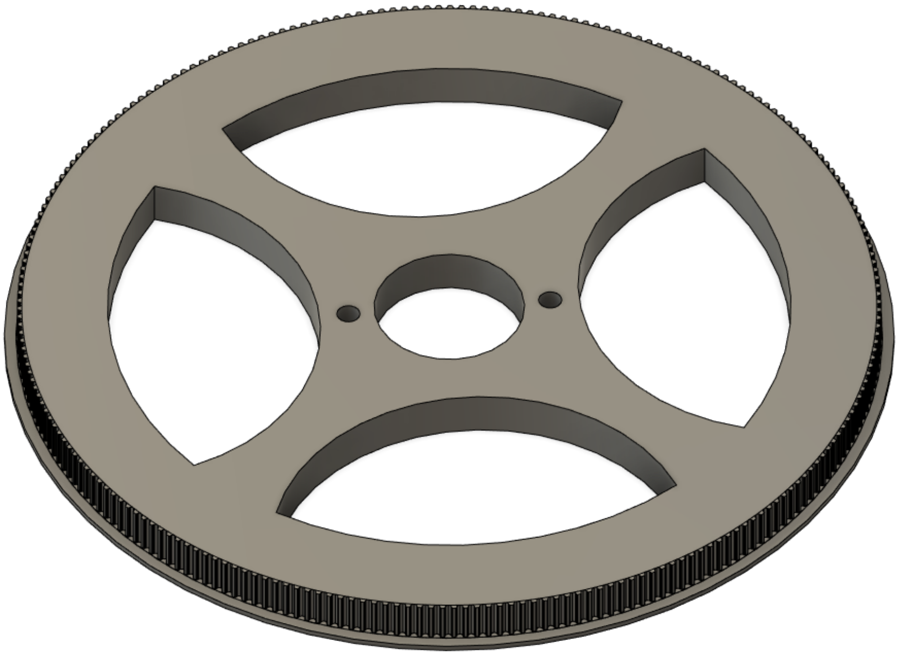


Figura 51: Polea perteneciente al bíceps

Al igual que como se procedió con el bíceps, se ha eliminado material que no era necesario, dando lugar a una pieza mucho más ligera y menos desperdicio de plástico. La polea acoplada al eje del motor es de 16 dientes, por lo que tenemos una relación de transmisión de aproximadamente 14,7:1. Para modelar los dientes de esta polea ha sido necesario tener en cuenta la curvatura de la correa a la hora de engranar con la polea.

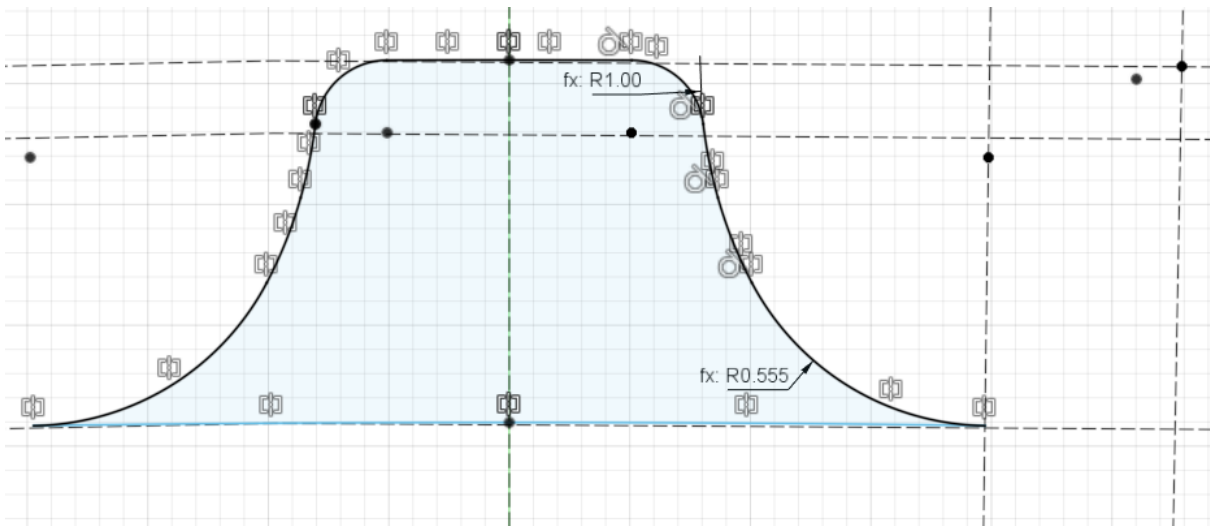


Figura 52: Forma final del diente de la polea

<u>7.5.4. Guarda de polea perteneciente al bíceps</u>

Debido a las limitaciones presentes en la impresión 3D, es necesario separar la guarda superior de la polea ya que es imposible imprimirla sin soportes sobre los que depositar el plástico.

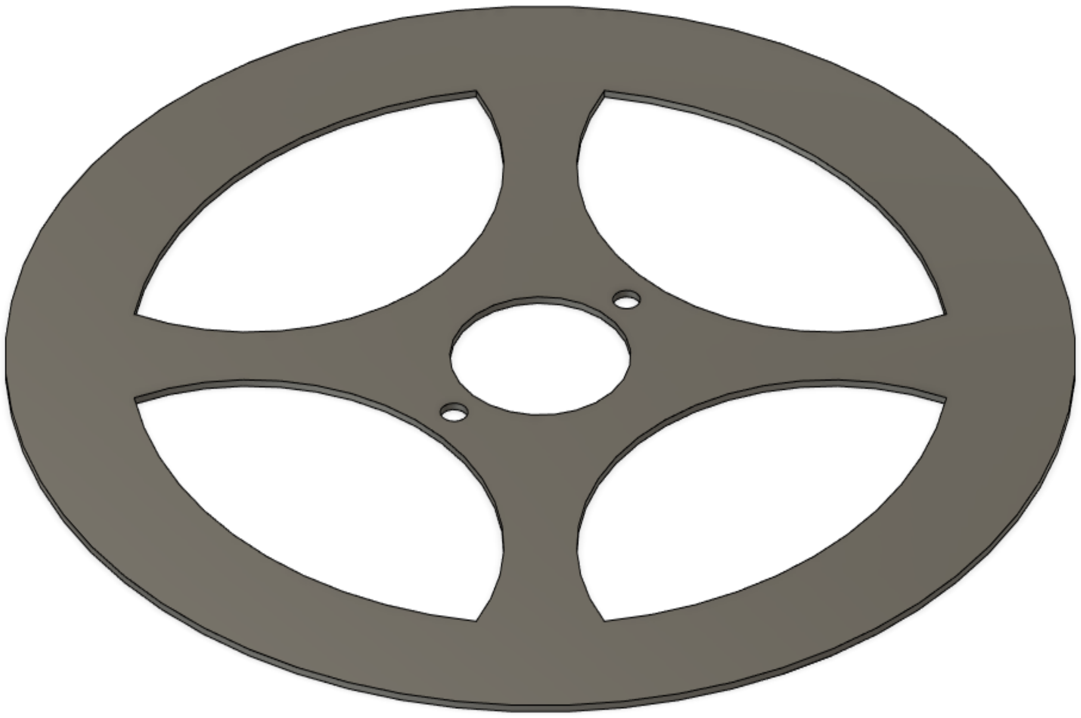


Figura 53: Guarda de polea perteneciente al bíceps

El grosor de la guarda de polea perteneciente al bíceps es idéntico al presente en la polea, esto es, 1 mm. Su objetivo es que la correa no se salga de la polea durante el movimiento de la misma. Esta guarda se atornilla junto a la polea, además, se utiliza pegamento fuerte para impedir que se doble y no cumpla su función al tener solo 1 mm de grosor.

7.6. Separador para bíceps

Los antebrazos deben permanecer paralelos entre sí. La diferencia de distancia entre los anclajes de los antebrazos en la base del efector final y los anclajes en los bíceps iría en contra de esta premisa. Por tanto, es necesario añadir en los extremos de los bíceps unos pequeños separadores para igualar exactamente la distancia entre los anclajes.

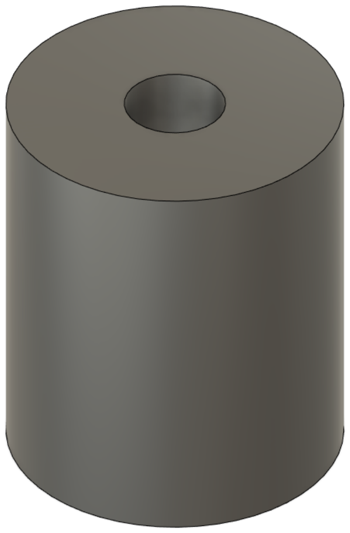


Figura 54: Separador para bíceps

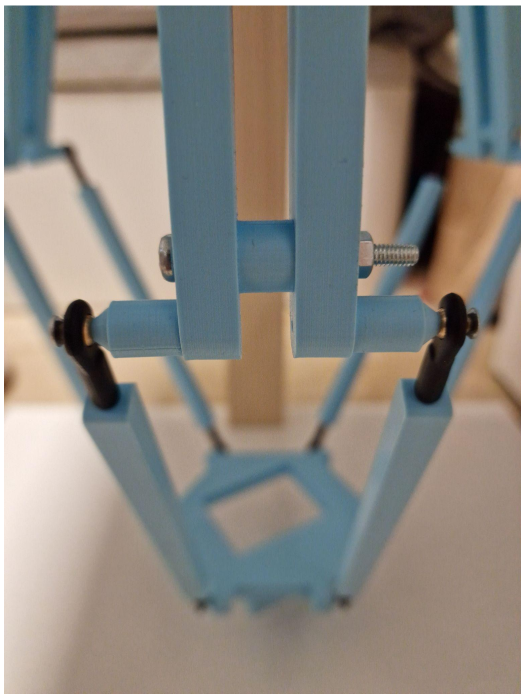


Figura 55: Separador para bíceps montado en el bíceps

7.7. Separador para rótula

Para unir los extremos de los antebrazos tanto con la base del efector final como con los bíceps utilizamos unas juntas denominadas rótulas. Estas uniones permiten la rotación en todas las direcciones, sin embargo, esta rotación se puede ver obstruida por otros elementos de la construcción del robot.

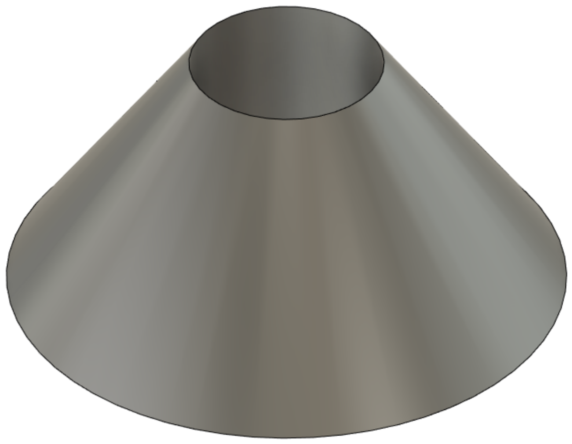


Figura 56: Separador para rótula

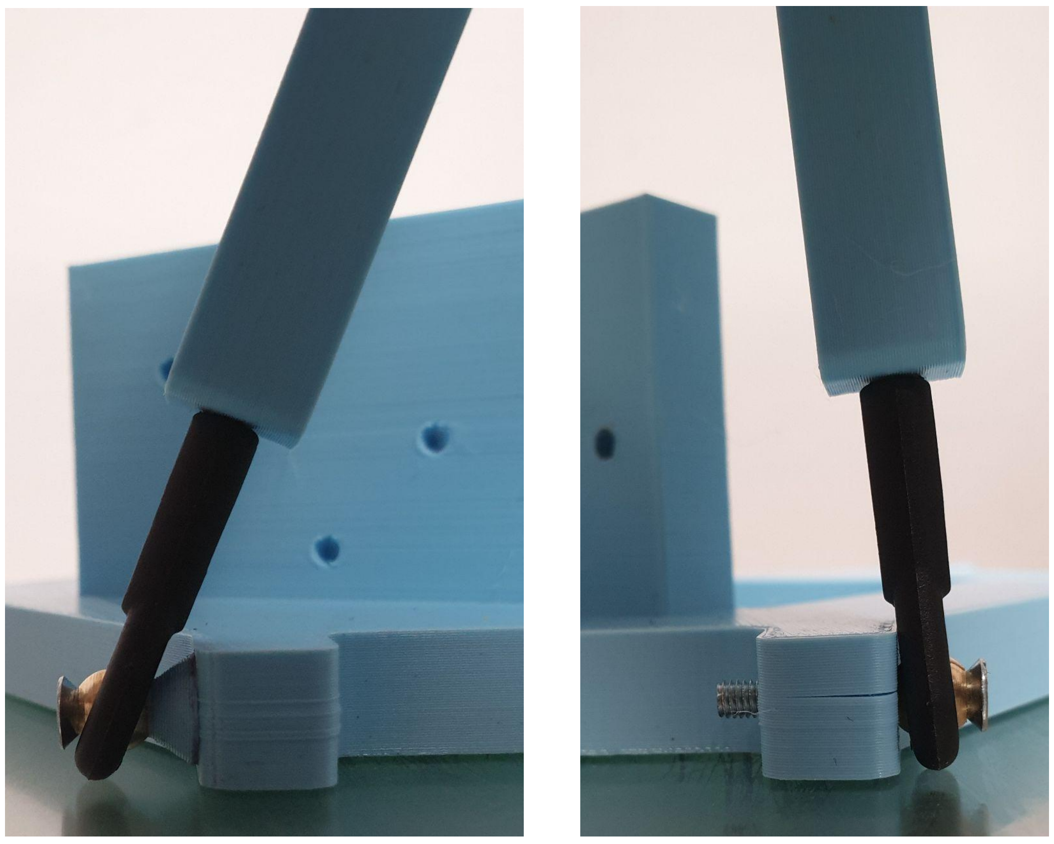


Figura 57: Diferencia en ángulo de rotación con el separador de rótula

7.8. Soporte para el sensor fin de carrera

Es necesario tener una manera de obtener una referencia de la posición en la que se encuentran los bíceps para poder actuar en consecuencia y disponer los brazos en el ángulo adecuado para alcanzar un punto final.

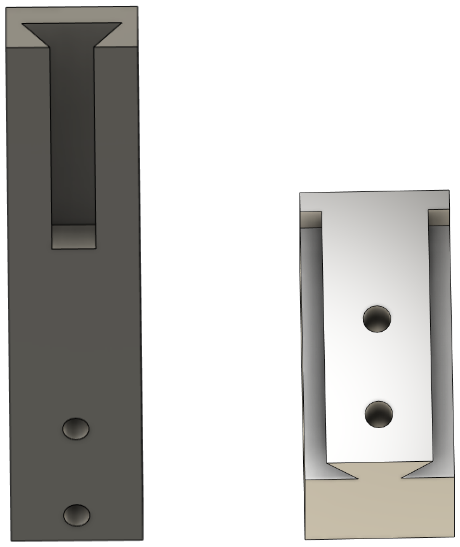


Figura 58: Soporte para el fin de carrera

Este soporte va atornillado en los anclajes de los bíceps con la plancha de madera para que siempre se encuentre en una posición determinada y, al hacer contacto el bíceps con el interruptor, obtengamos la referencia de la posición del mismo. Además, se ha optado por diseñar un mecanismo de deslizamiento, en el que la pieza en la que se acopla el fin de carrera es fácilmente extraíble del soporte por si es necesario quitarlo del recorrido del bíceps para el movimiento del mismo.

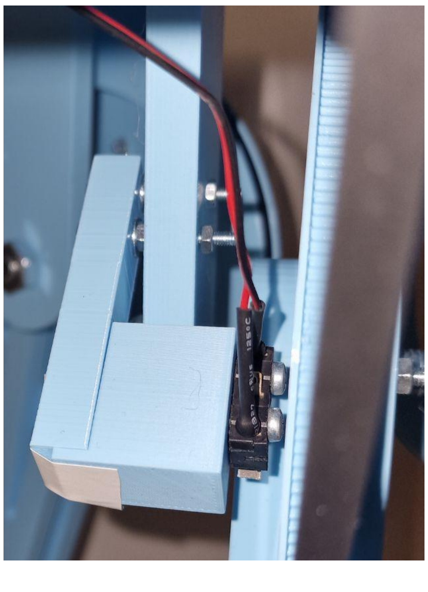


Figura 59: Soporte para el fin de carrera ensamblado

7.9. Soporte de la fuente de alimentación

La fuente de alimentación es un componente pesado, con riesgo eléctrico y debe ser debidamente asegurado en la plancha de madera, sin dar la opción a que se mueva ni que el soporte pueda fallar.

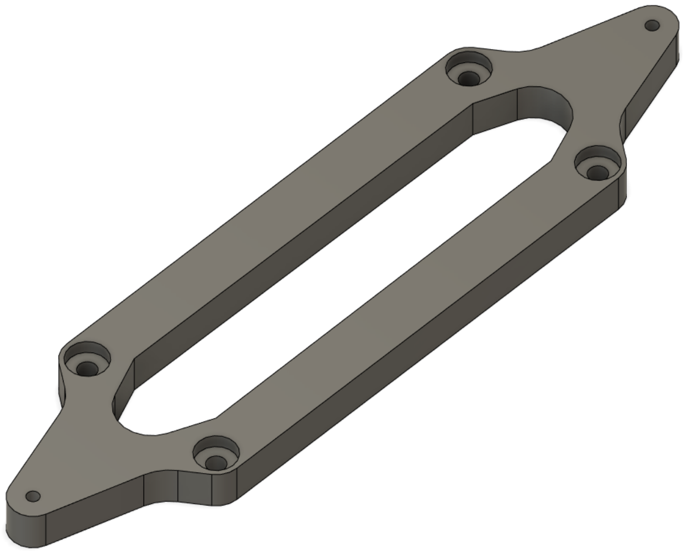


Figura 60: Soporte de la fuente de alimentación

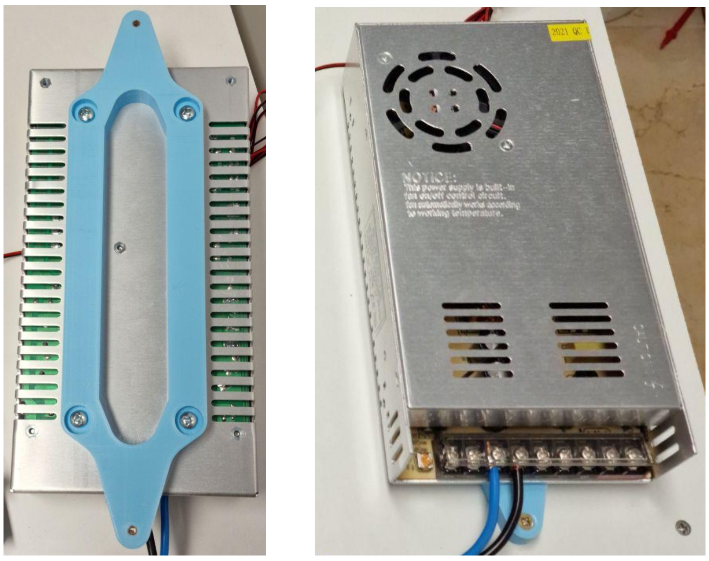


Figura 61: Soporte de la fuente de alimentación ensamblado

Como se puede observar en las Fig. 60 y 61, los tornillos quedan dentro del soporte, pudiendo así atornillar el soporte a la plancha de madera para aportar la estabilidad necesaria a la fuente de alimentación.

7.10. Soporte para la pantalla LCD

La pantalla LCD tiene conectores por la parte trasera y debe ser separada con bastante holgura de la plancha de madera, por tanto, son necesarios unos soportes que la dejen a una altura adecuada y que sea fácilmente operable desde la parte superior.

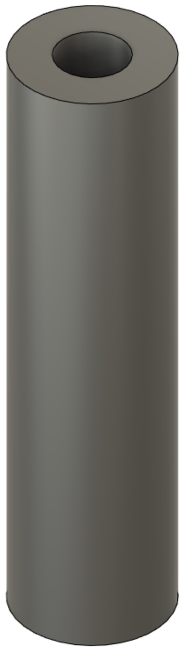


Figura 62: Soporte para la pantalla LCD

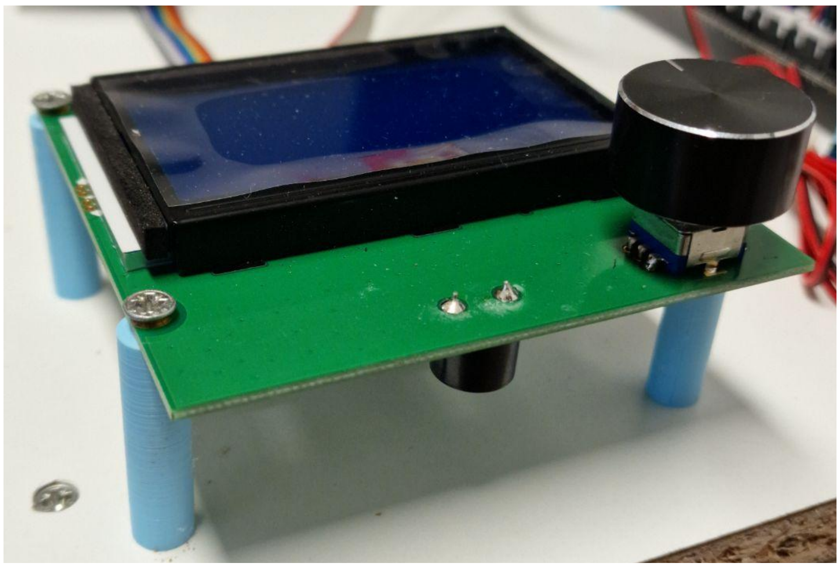


Figura 63: Soporte para la pantalla LCD ensamblado

7.11. Soporte para placa controladora

La placa controladora posee en su parte posterior muchas de las soldaduras pertenecientes al circuito electrónico, con elementos que sobresalen y que no deben estar en contacto directo con la plancha de madera, tanto para no dañar los componentes y las conexiones como para no crear, de manera no intencionada, cortocircuitos.

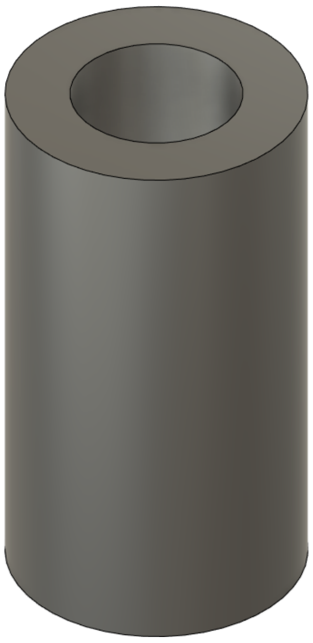


Figura 64: Soporte para la placa controladora

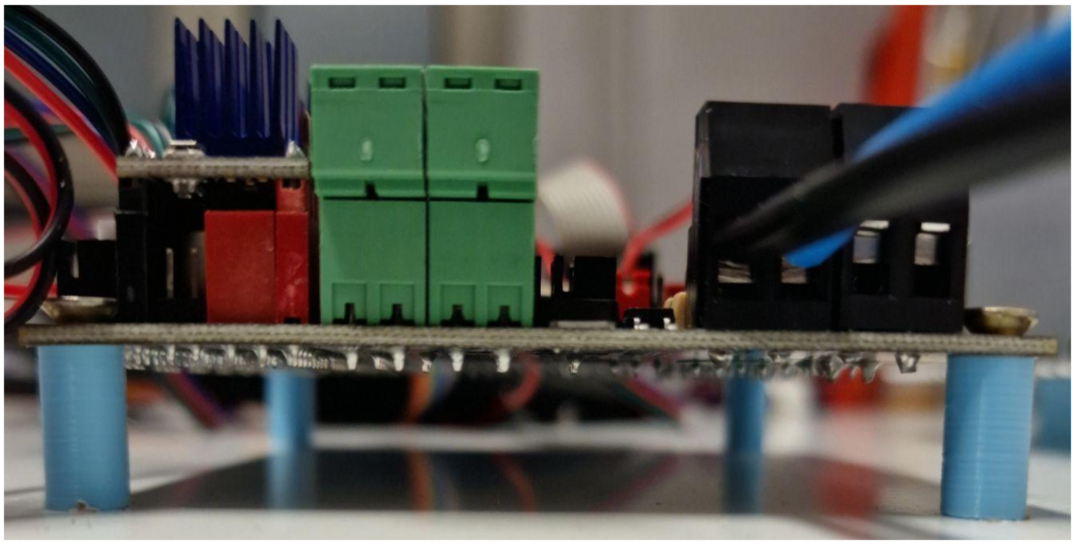


Figura 65: Soporte para la placa controladora ensamblado

Capítulo 8: Impresión 3D de los elementos del robot

8.1. La impresora

Tal y como mencionamos en el capítulo anterior, todos los elementos del robot han sido diseñados para ser fácilmente impresos en 3D. La impresora que hemos utilizado para imprimir todas las piezas se trata de una impresora Creality CR-10S.

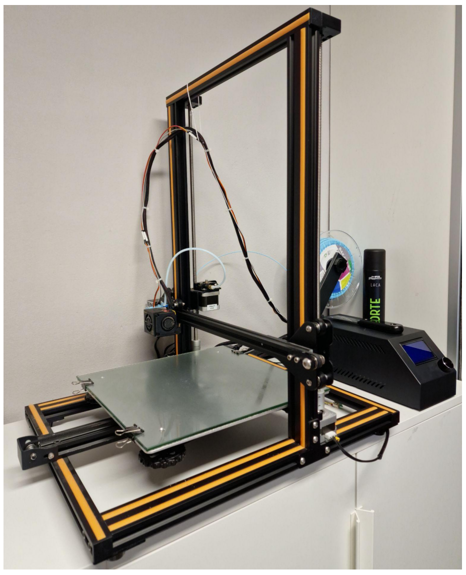


Figura 66: Impresora Creality CR-10S utilizada para imprimir las piezas

Las características principales de esta impresora son:

- Volumen de impresión de 300 x 300 x 400 mm.
- Boquilla de impresión de 0,4 mm de diámetro.
- Diámetro del filamento de 1,75 mm de diámetro.
- 270 W de potencia.

Esta impresora ha sufrido diversas modificaciones para obtener una calidad de impresión superior, como una placa controladora más potente, unos drivers para los motores paso a paso de mejor calidad, logrando que su operación sea silenciosa y un sensor de nivelación de cama automático con repetibilidad de alrededor de 0.005 mm.

Por otra parte, el firmware presente en la placa controladora es Marlin 2.1.1. Este firmware ha sido configurado y compilado manualmente para incluir las características de nivelado automático, compatibilidad de los drivers de los motores paso a paso y características generales de la impresora.

Además, su estructura rígida logra que no se introduzcan imperfecciones en las piezas imprimidas a gran velocidad por el efecto de las inercias de los componentes de la impresora 3D.

8.2. Material de impresión

El material elegido para las piezas que conforman el robot Delta es el PLA (ácido poliláctico). Este material es un poliéster biodegradable y bioactivo, se deriva de materias primas naturales y renovables, como el maíz, y pertenece a los poliésteres como un polímero sintético. El almidón (glucosa) se extrae de las plantas y se convierte en dextrosa mediante la adición de enzimas. Esto es fermentado por microorganismos en ácido láctico, que a su vez se convierte en polilactida. La polimerización se produce con cadenas moleculares, similares en sus propiedades a los polímeros a base de petróleo [15][16].

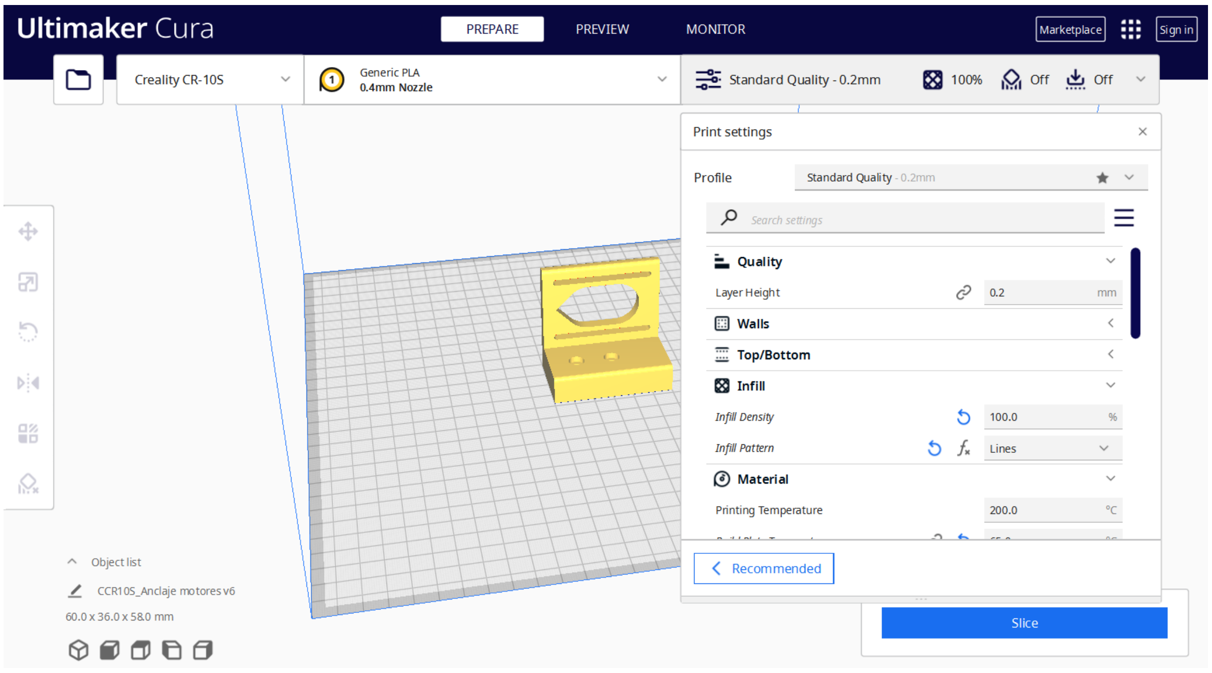
Figura 67: PLA para impresión 3D. Bibliografía de imágenes [8]

Las principales características de este material son [17]:

- Facilidad de impresión muy alta
- No desprende olor al imprimir
- Temperatura de extrusión 200 a 240 ºC
- Densidad 1.210-1.430 g/cm^3
- Resistencia a tracción 3309 MPa
- Resistencia al impacto muy baja
- Temperatura de deformación 55ºC
- Resistencia UVA y humedad muy baja
- Buena reciclabilidad

8.3. Software de impresión

Tanto la configuración de impresión como la traducción a formato GCODE de los archivos STL que contienen las piezas del robot Delta se han realizado en el software Cura.



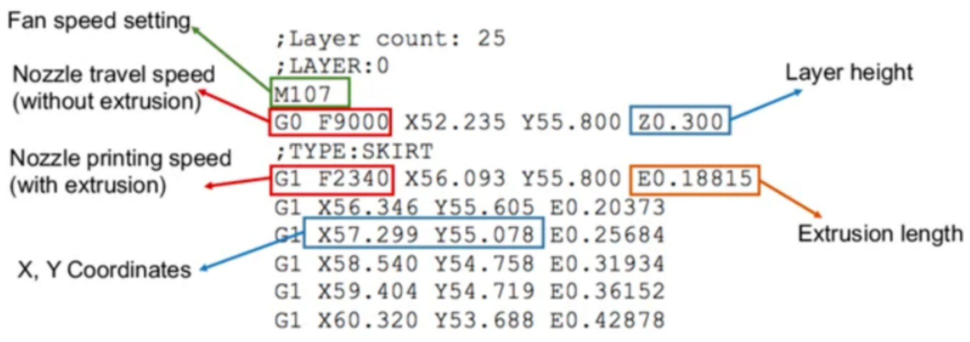
Creality CR-10S
Generic PLA
0.4mm Nozzle
Standard Quality - 0.2mm 100% Off Off

Print settings
Profile: Standard Quality - 0.2mm
Quality
Layer Height 0.2 mm
Walls
Top/Bottom
Infill
Infill Density 100.0 %
Infill Pattern Lines
Material
Printing Temperature 200.0 °C

CCR10S_Anclaje motores v6
60.0 x 36.0 x 58.0 mm

Figura 68: Visión general del software Cura

En este programa es posible convertir a comandos GCODE la pieza que queremos imprimir con la configuración deseada de temperatura, velocidad, etc. El G-Code consta de comandos G y M, cada uno de ellos con un movimiento o acción asignado. La combinación de estos comandos permitirá a la impresora 3D entender qué patrón seguir con el fin de crear la pieza final [18].



```
;Layer count: 25
;LAYER:0
M107
G0 F9000 X52.235 Y55.800 Z0.300
;TYPE:SKIRT
G1 F2340 X56.093 Y55.800 E0.18815
G1 X56.346 Y55.605 E0.20373
G1 X57.299 Y55.078 E0.25684
G1 X58.540 Y54.758 E0.31934
G1 X59.404 Y54.719 E0.36152
G1 X60.320 Y53.688 E0.42878
```

Figura 69: GCODE. Bibliografía de imágenes [9]

8.4. Configuración de impresión

Para la correcta impresión de las piezas es necesario hallar la temperatura de impresión, velocidad de impresión y muchas otras características que mejor influyan en el resultado final de la impresión. Como norma general para todas las piezas, tras investigar para dar con la mejor solución, se han empleado las siguientes configuraciones:

- **Altura de capa:** 0,2 mm.
- **Ancho de muro:** 0,8 mm.
- **Relleno:**
  - **Tipo de relleno:** líneas o cúbico
  - **Porcentaje de relleno:** 20 - 35%
- **Temperatura de impresión:** 200ºC.
- **Temperatura de la cama caliente:** 65ºC.
- **Velocidad de impresión:**
  - **Velocidad de impresión de muros:** 25 mm/s.
  - **Velocidad de impresión de relleno:** 50 mm/s.
  - **Velocidad de impresión de la primera capa:** 10 mm/s.
- **Retracción activada:**
  - **Distancia de retracción:** 6 mm.
  - **Velocidad de retracción:** 45 mm/s.
  - **Salto en "z" mientras retraído:** activado.
- **Velocidad de ventilador:**
  - **Velocidad de ventilador para las primeras capas:** 0%
  - **Velocidad de ventilador para el resto de capas:** 100%
- **Ayuda para la adhesión a la cama caliente:**
  - **Tipo de ayuda:** Falda.
  - **Número de líneas para la falda:** 1.
  - **Distancia de la falda a la pieza:** 5 mm.

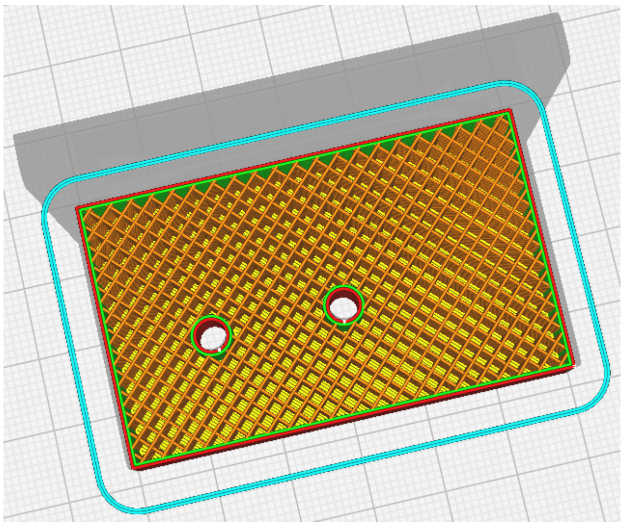


Figura 70: Ejemplo de configuración de Cura

Una vez aportada la configuración utilizada para la impresión de las piezas, procedemos a aportar una breve descripción de cada una de las configuraciones para facilitar su comprensión:

- **Altura de capa:** Es la altura de cada capa en milímetros. Valores grandes ocasionan unas impresiones más rápidas en una resolución más pequeña, por el contrario, valores pequeños son capaces de producir muchos detalles en buena resolución pero en un tiempo mucho mayor.
- **Ancho de muro:** 0,8 mm. Es el ancho de los muros en la dirección horizontal.
- **Relleno:** El interior de la pieza, por dentro de los muros.
    - **Tipo de relleno:** Patrón que se sigue para rellenar la pieza por dentro de los muros.
    - **Porcentaje de relleno:** Porcentaje del volumen interior de la pieza que se debe ocupar con plástico, el resto será aire. Valores más altos aportarán mayor resistencia a la pieza, a costa de utilizar más cantidad de material y un peso mayor del producto final.
- **Temperatura de impresión:** Temperatura a la que se funde el plástico que pasa por la boquilla de impresión.
- **Temperatura de la cama caliente:** Temperatura a la que está la plataforma sobre la que se imprime. Se utiliza para mejorar la adhesión del plástico a la plataforma.
- **Velocidad de impresión:** Velocidad a la que se imprime la pieza.
    - **Velocidad de impresión de muros:** Velocidad a la que se imprimen los muros exteriores de la pieza.
    - **Velocidad de impresión de relleno:** Velocidad a la que se imprime el patrón de relleno del interior de la pieza.
    - **Velocidad de impresión de la primera capa:** Velocidad a la que se imprime la primera capa de la pieza. Ayuda con la adhesión a la cama de la pieza.
- **Retracción activada:** A la hora de realizar movimientos en los que no se va a depositar plástico retraer el filamento por la boquilla de impresión.
    - **Distancia de retracción:** Cuanta distancia se retrae el filamento.
    - **Velocidad de retracción:** Velocidad a la que se retrae el filamento.
    - **Salto en “z” mientras retraído:** Aumentar la posición en vertical para no interferir con partes de la pieza ya impresas.
- **Velocidad de ventilador:** Velocidad del ventilador que apunta a la pieza impresa.
    - **Velocidad de ventilador para las primeras capas:** Es mejor dejarlo desactivado para mejorar la adhesión a la cama caliente de la pieza.
    - **Velocidad de ventilador para el resto de capas:** Es importante que el plástico se enfríe suficientemente rápido para poder imprimir la siguiente capa encima.
- **Ayuda para la adhesión a la cama caliente:**
    - **Tipo de ayuda:** Falda. Lo primero en imprimirse es una serie de líneas alrededor de la pieza que se va a imprimir para purgar el filamento que puede haber goteado y mejorar la adhesión de la pieza a la plataforma.
    - **Número de líneas para la falda:** Número de líneas a imprimir alrededor de la pieza.
    - **Distancia de la falda a la pieza:** Distancia que se deja entre la pieza y las líneas de la falda.

Por último, a modo de ejemplo, proporcionamos figuras en las que se observa la polea perteneciente a los bíceps durante el proceso de impresión.

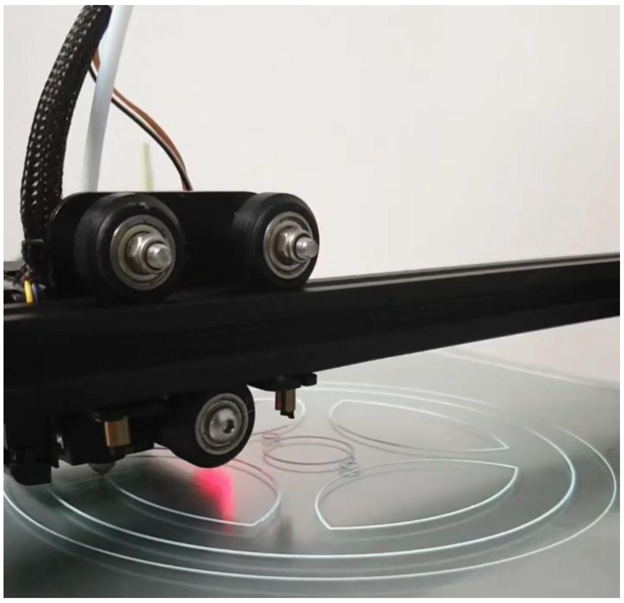


Figura 71: Primera capa de impresión de la polea

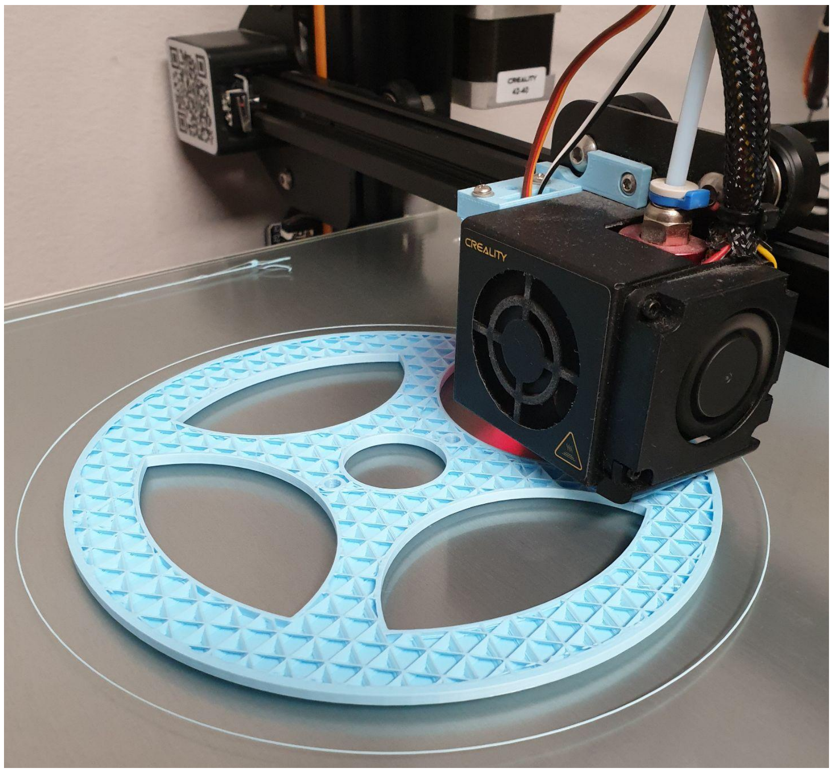


Figura 72: Impresión de la polea en un estado más avanzado

Capítulo 9: Estructura mecánica final del robot y dotación de la electrónica

9.1. Estructura mecánica final

Tras haber llevado a cabo la investigación de la cinemática del robot, el diseño de los componentes que lo forman y la impresión 3D de los mismos, hemos ensamblado por completo el prototipo y el resultado se puede observar a continuación.

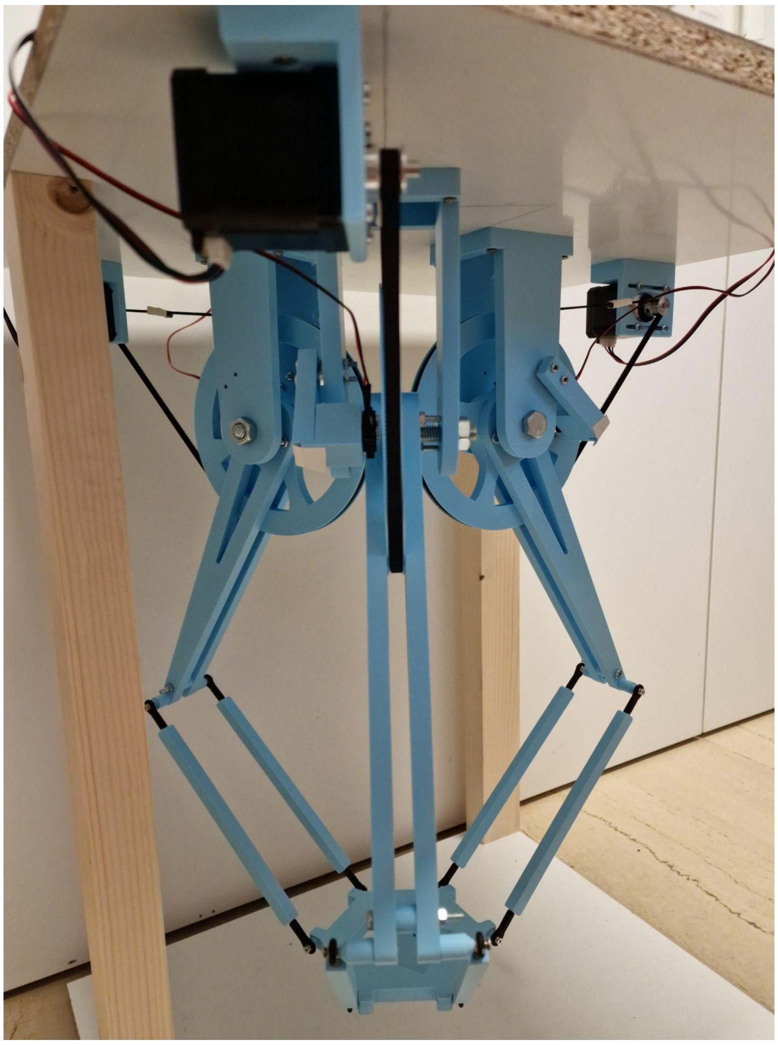


Figura 73: Estructura mecánica final vista desde uno de los brazos

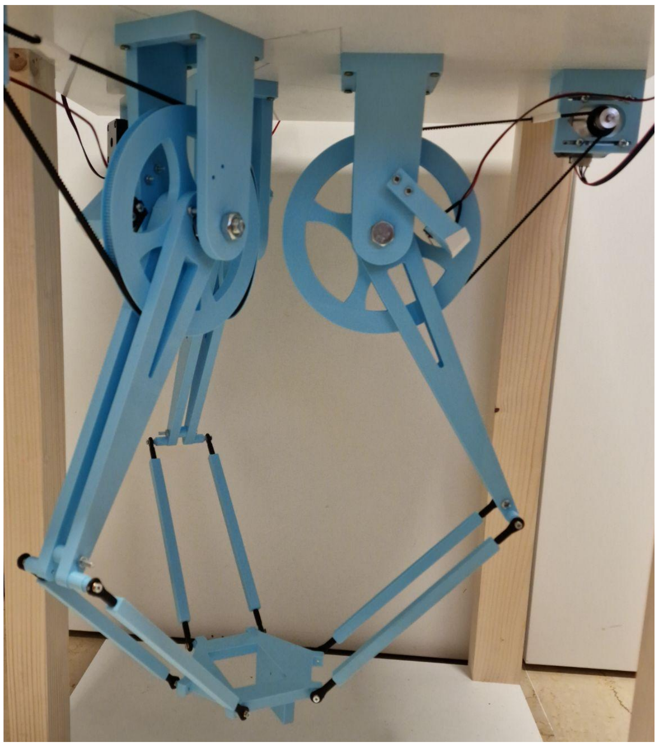


Figura 74: Estructura mecánica final vista de cerca

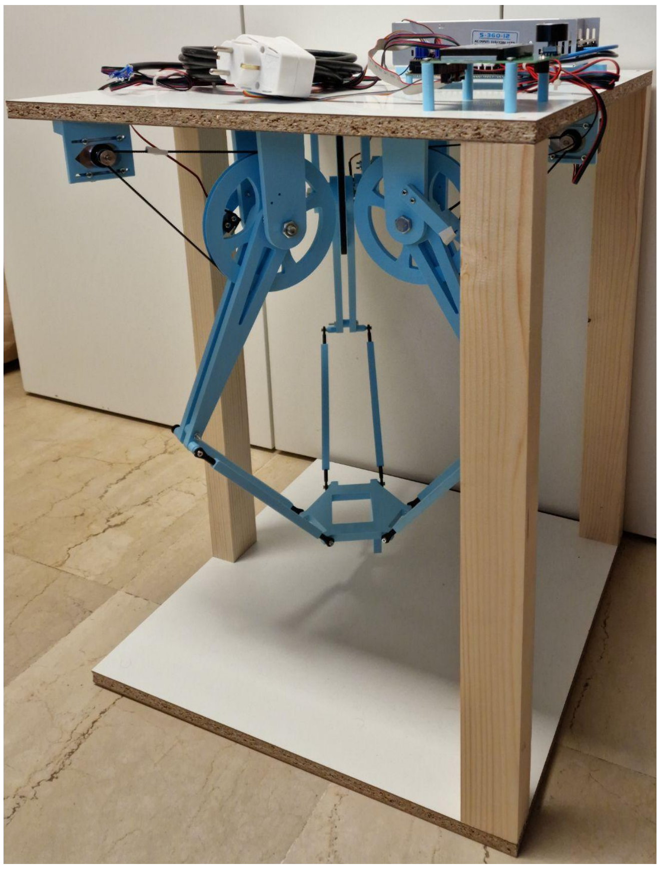


Figura 75: Estructura mecánica final vista de lejos

9.2. Correa de acople entre motor y brazos

La correa que acopla el eje del motor con el bíceps correspondiente a cada uno de los brazos es del tipo GT2. Es una correa dentada que es ampliamente utilizada en el campo de la impresión 3D. En nuestro caso, es necesario comprar la correa abierta, cortarla a la longitud apropiada y unir los extremos para cerrarla.

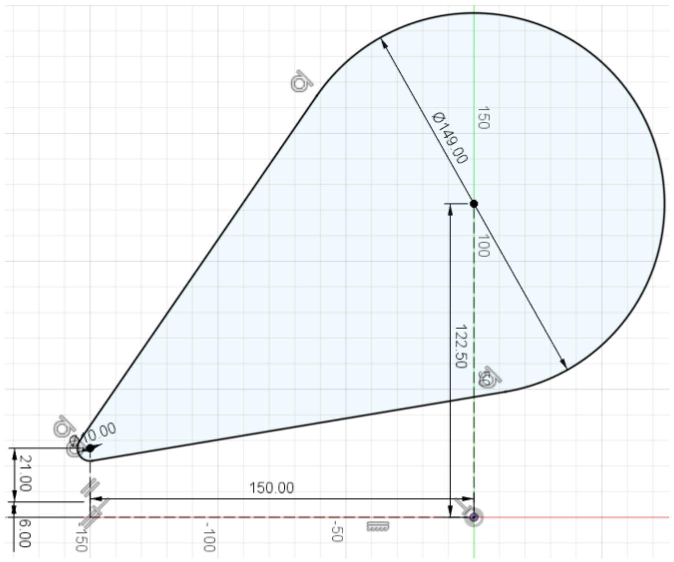


Figura 76: Obtención del largo de la correa

Para obtener el largo necesario de la correa situamos dos circunferencias con radios iguales a los de las poleas que va a unir separadas entre sí de la manera que estarán una vez ensambladas en la estructura final. El largo necesario de la correa es 633 mm. Para poder tener una longitud de unión de 15 mm el largo necesario asciende a 663 mm.

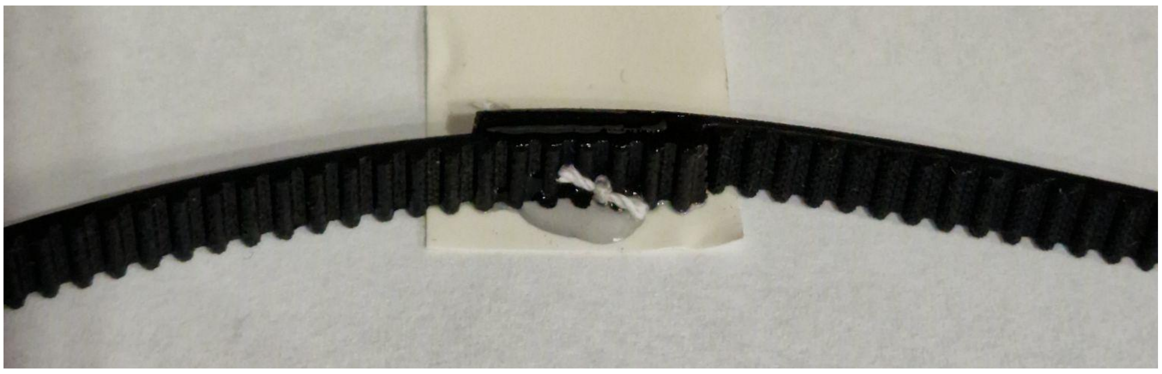


Figura 77: Cierre de la correa

Para cerrar la correa primero se da una puntada de hilo que une ambas partes y las inmoviliza. Posteriormente, se introduce pegamento fuerte en la unión y, cuando se seca por completo, se envuelve en cinta aislante. Esta parte de la correa no es apta para su uso en poleas, sin embargo, se puede introducir en nuestra estructura de manera que siempre quede en el espacio entre ambas poleas.

9.3. Juntas de rótula

Como ya mencionamos anteriormente, para unir los extremos de los antebrazos tanto con la base del efector final como con los bíceps utilizamos unas juntas denominadas rótulas. Estas uniones permiten la rotación en todas las direcciones.

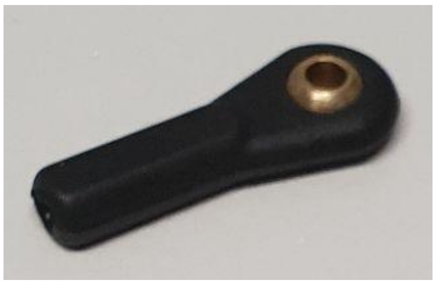


Figura 78: Junta de rótula

Sin embargo, esta junta viene con un agujero de 3 mm en su base para enroscar un tornillo de la misma medida y nuestros antebrazos no poseen ningún tornillo. Por tanto, es necesario cortar una varilla roscada de 3 mm para unir ambos elementos.

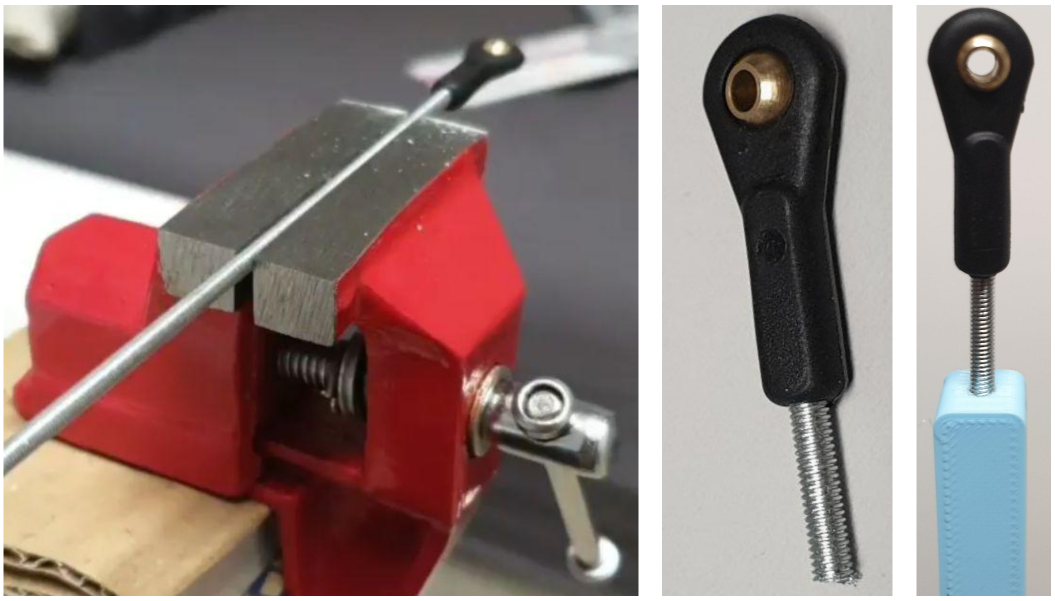


Figura 79: Proceso de unión de la rótula con el antebrazo

De esta manera quedan unidas las juntas de rótula con los antebrazos. En total, es necesario realizar esta operación doce veces, una para la unión de cada extremo de los antebrazos.

9.4. Dotación de la electrónica

9.4.1. Placa controladora

La placa controladora elegida es la Bigtreetech SKR 2.0. Se trata de una placa controladora de 32 bits con las siguientes características destacadas [19]:

- Microprocesador de arquitectura ARM.
- Protecciones en los conectores de los termistores.
- Protecciones térmicas.
- Protección en drivers si estos son colocados en una posición incorrecta.
- Puerto dedicado a wifi.
- 3 ventiladores controlables.
- Adaptador para tarjeta micro SD además de incluir un conector USB.
- Salida para 5 drivers de motores paso a paso.
- Entradas para 3 finales de carrera.
- Entradas para sensores de nivelación automática.

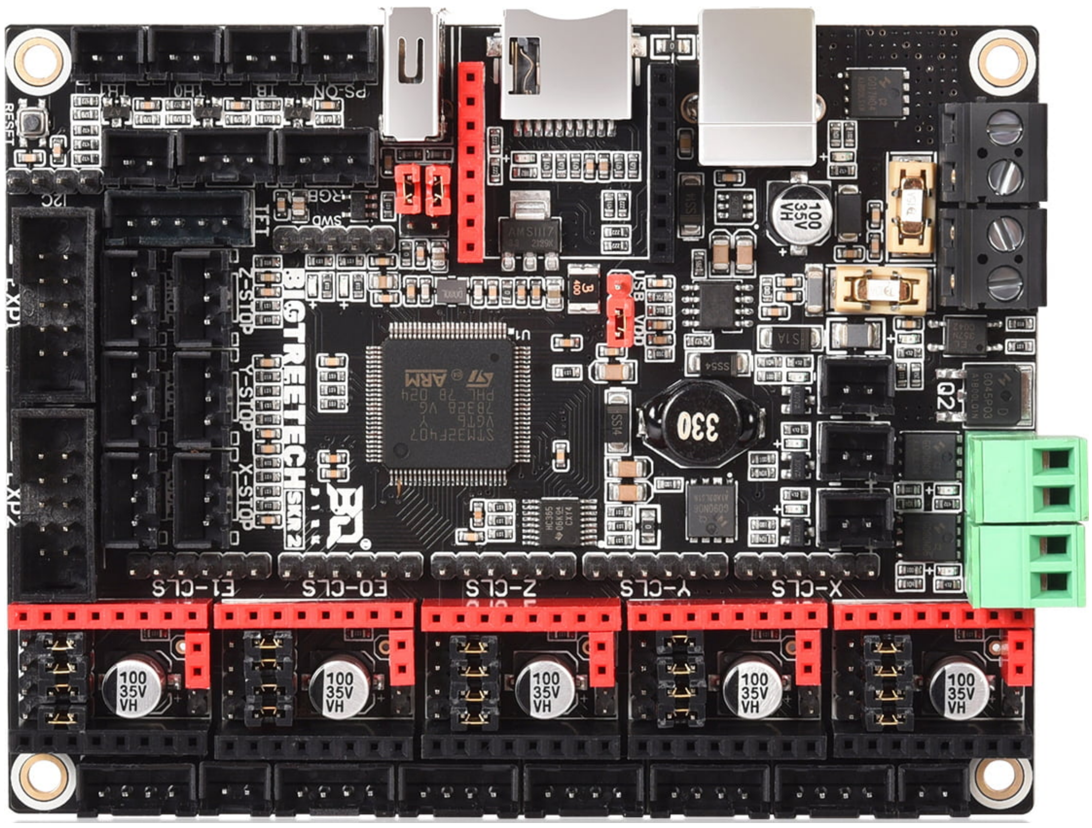


Figura 80: Placa controladora Bigtreetech SKR 2.0.

Esta placa controladora ha sido diseñada expresamente para impresión 3D y es ampliamente utilizada en este campo. Es una opción económica pero potente y con todas las características que necesita este trabajo.

9.4.2. Drivers de motores paso a paso

Los drivers de motores paso a paso elegidos son los TMC 2208. Se trata de un chip de accionamiento de motor paso a paso de dos fases ultra silencioso, que permite un control sinusoidal perfecto incluso en sistemas con frecuencias de pulso limitadas y son ampliamente utilizados en la impresión 3D. Compatible con la electrónica de impresoras 3D existentes, eliminando los costosos costos del rediseño [20].

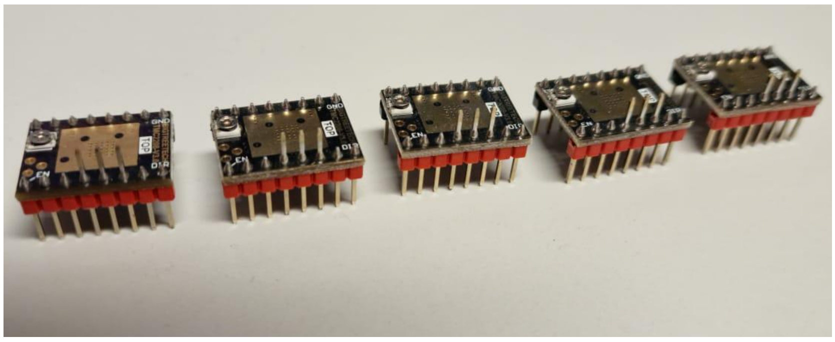


Figura 81: Drivers TMC 2208

Entre sus características destacan:

- Microsteps nativos: hasta 1/256.
- Voltaje lógico: 3-5V.
- Voltaje del motor: 5.5-36V.
- Corriente de la fase del motor max: 1.2 A RMS, 2.0 A pico.
- Regulador interno de voltaje.

Estos drivers irán conectados en la placa controladora, sirviendo de interfaz entre la misma y los motores.

9.4.3. Interruptores final de carrera

Es necesario tener una manera de obtener una referencia de la posición en la que se encuentran los bíceps para poder actuar en consecuencia y disponer los brazos en el ángulo adecuado para alcanzar un punto final. Para ello utilizaremos 3 interruptores final de carrera.

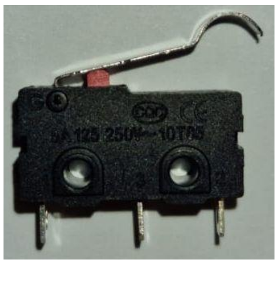


Figura 82: Interruptor final de carrera

9.4.4. Motores

Hemos elegido motores tipo NEMA 17. Este tipo de motores es ampliamente utilizado en la impresión 3D, ya que ofrecen mucha precisión y un par considerable, manteniendo un coste reducido.

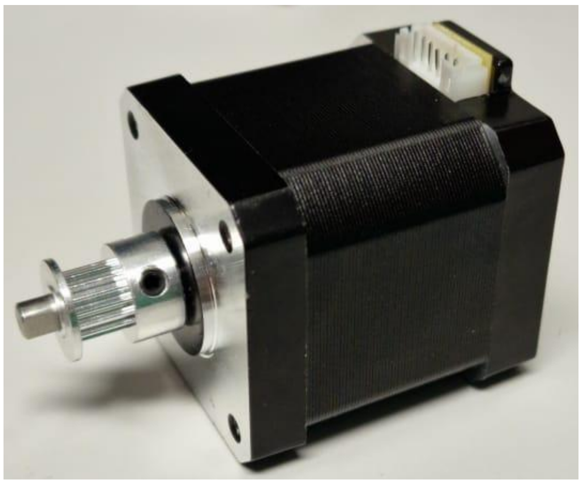


Figura 83: Motor NEMA 17

Este motor tiene una precisión de 1,8º por paso, 23 Ncm de par y un peso de 270 gramos. En su eje acoplamos una polea de 16 dientes para utilizar la correa que conecta el motor con el bíceps.

9.4.5. Pantalla LCD

Hemos optado por utilizar la pantalla LCD de la marca Creality, presente en muchas de sus impresoras 3D. Se trata de una pantalla de tipo 12864 con rueda selectora incorporada para el control del robot Delta.

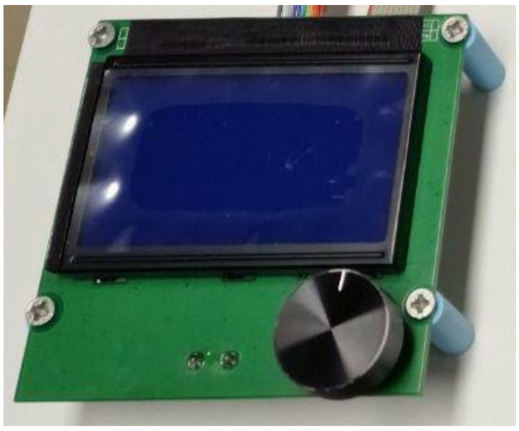
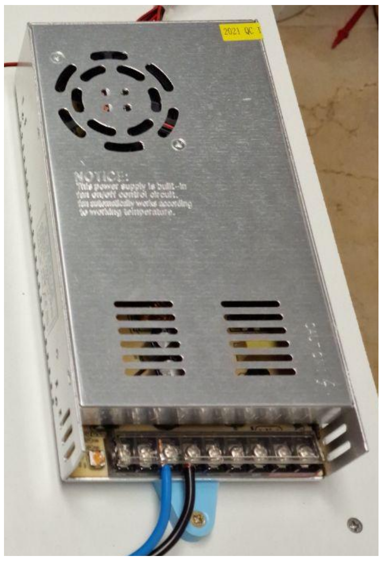


Figura 84: Pantalla LCD de Creality

9.4.6. Fuente de alimentación

Debido a que la placa controladora trabaja con una tensión continua de 12/24 V es necesario incluir una fuente de alimentación que transforme la tensión alterna presente en la red de 230V a una tensión continua admisible por la placa. Para ello hemos elegido una fuente de alimentación de 360W de potencia, que posee una salida de tensión continua de 12 V y 30 A.



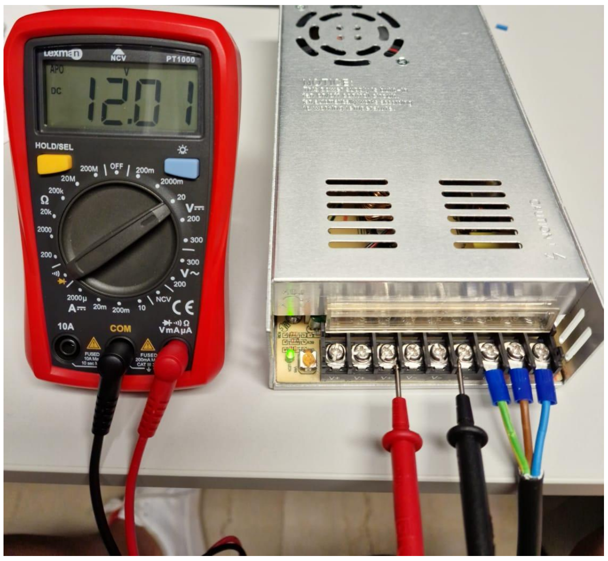


Figura 85: Fuente de alimentación y su regulación

Al conectarla a la red es necesario ajustar con un destornillador el voltaje de salida girando levemente el potenciómetro de ajuste para obtener 12 voltios a la salida. Además, ha sido necesario confeccionar un cable de alimentación que se enchufe a la red y tenga como extremo tres terminales para conectar a la fuente.

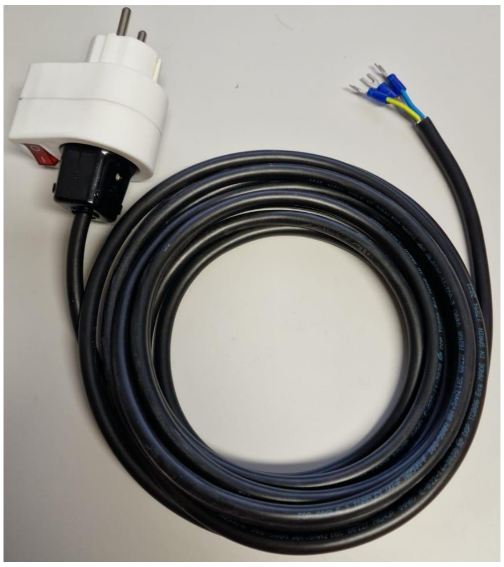
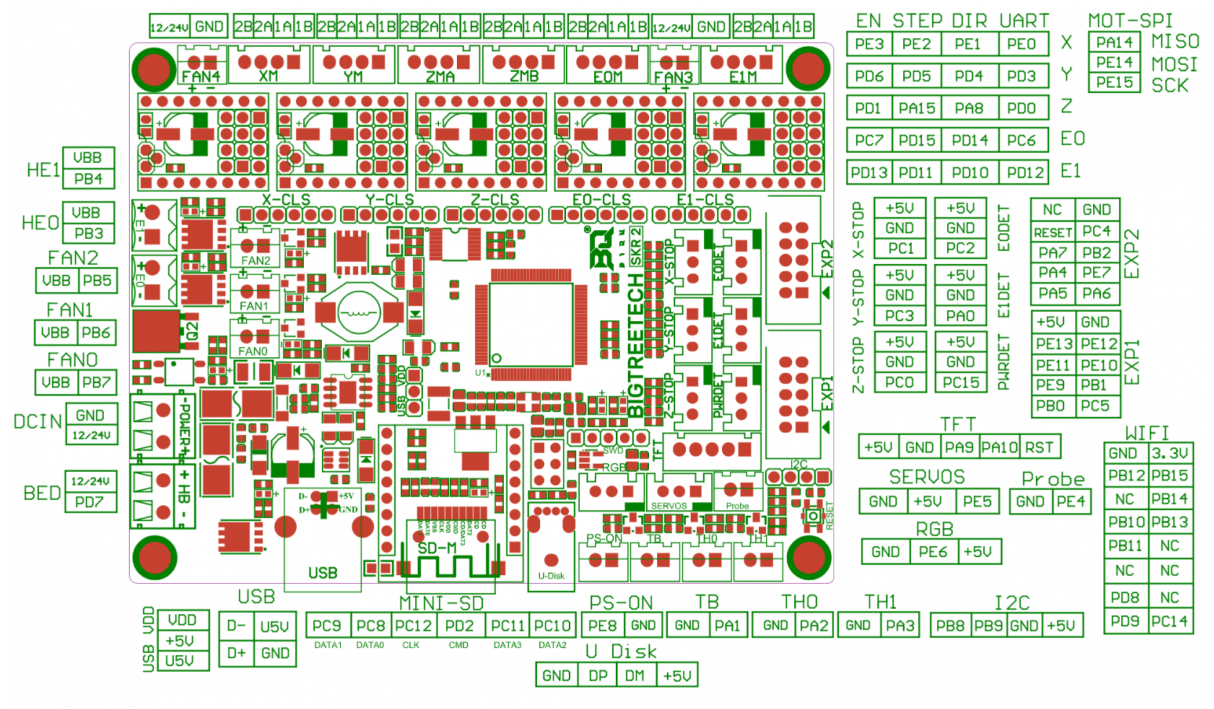


Figura 86: Cable para fuente de alimentación

9.4.7. Conexión de los componentes electrónicos

Una vez definidos los componentes que vamos a utilizar es necesario conectarlos de manera precisa. Se puede observar la configuración de pines de la placa controladora en la Fig. 87.



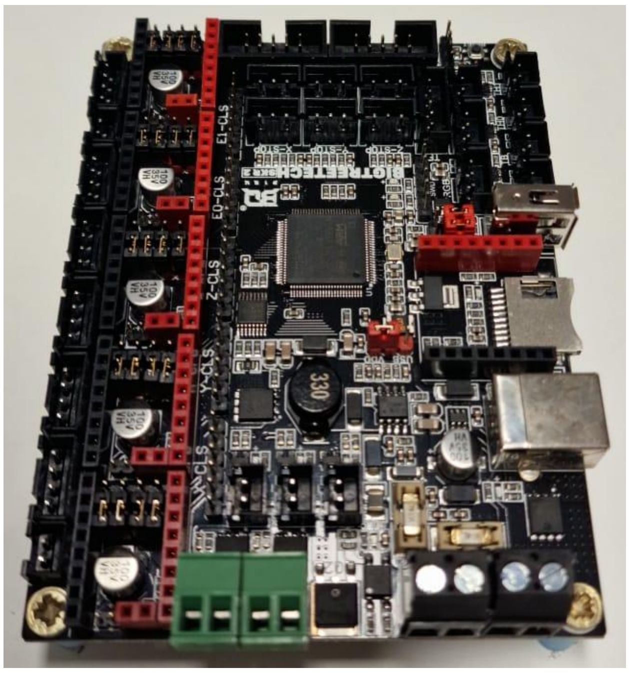


Figura 87: Estado inicial de la placa controladora y pinout

Por tanto, procederemos de la siguiente manera:

1. Introducimos los drivers TMC 2208 en sus pines correspondientes.

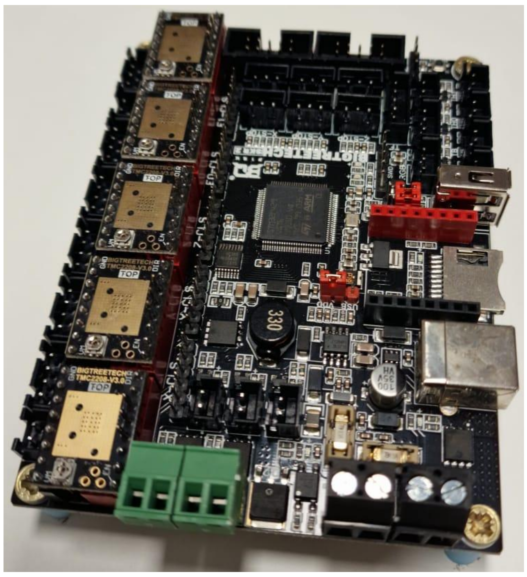


Figura 88: Paso 1 de la conexión

2. Pegamos los radiadores encima de los drivers TMC 2208 y conectamos cada uno de los motores a su salida.

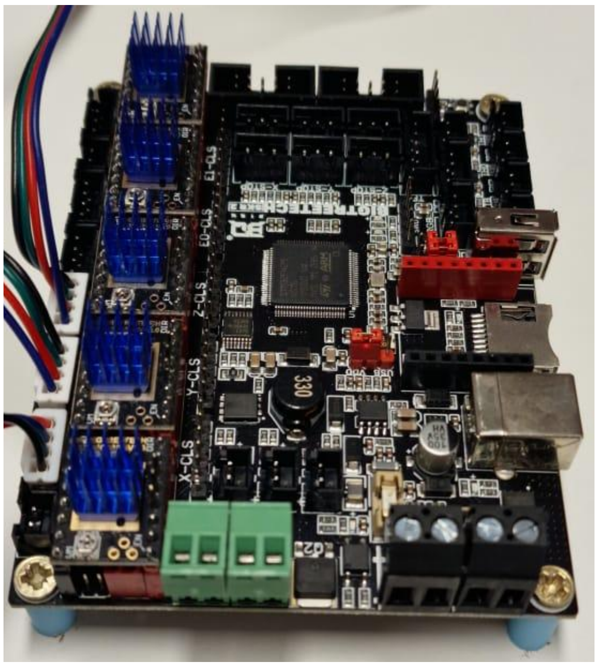


Figura 89: Paso 2 de la conexión

3. Conectamos los 3 finales de carrera cada uno en la posición de su correspondiente brazo.

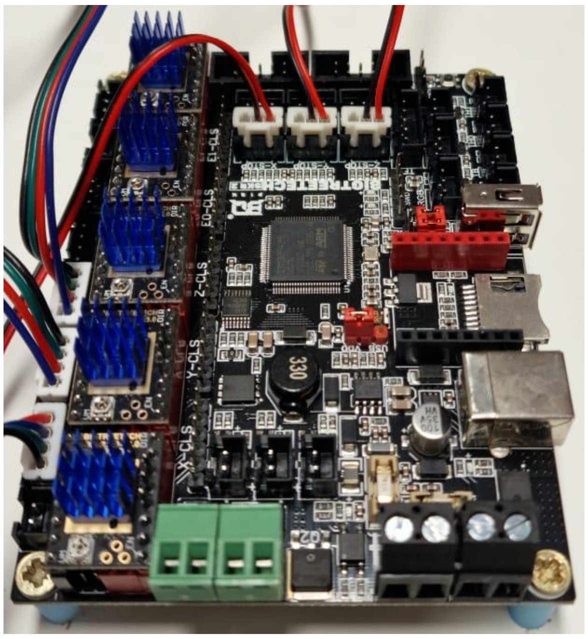


Figura 90: Paso 3 de la conexión

4. Conectamos la pantalla LCD a la placa. Hacemos esto conectado el conector EXP1 y el conector EXP2 en sus sitios correspondientes.

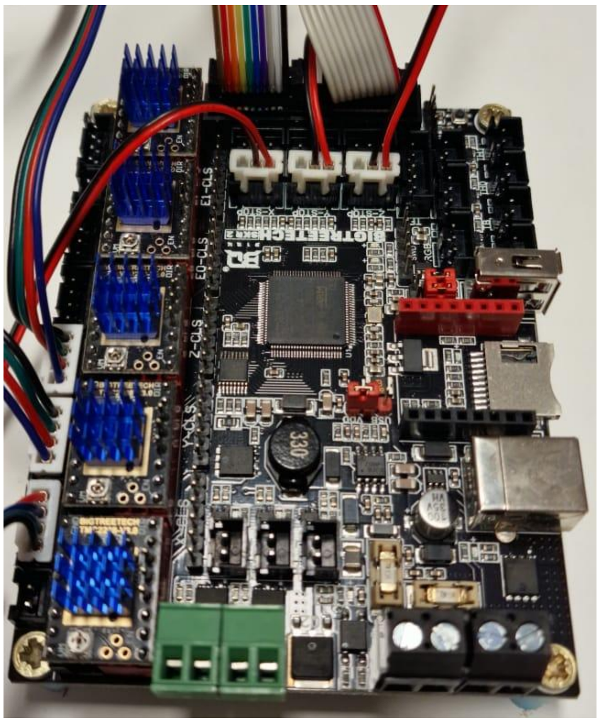


Figura 91: Paso 4 de la conexión

5. Por último, conectamos la placa controladora con la fuente de alimentación, el cable negro es 0 V y el cable azul es 12 V.

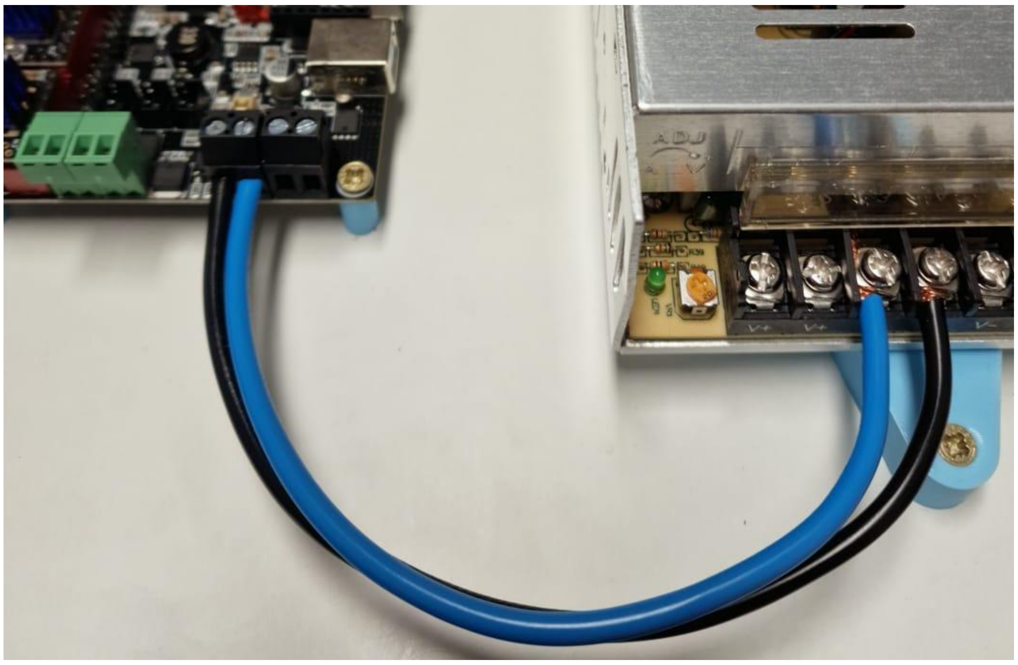


Figura 92: Paso 5 de la conexión

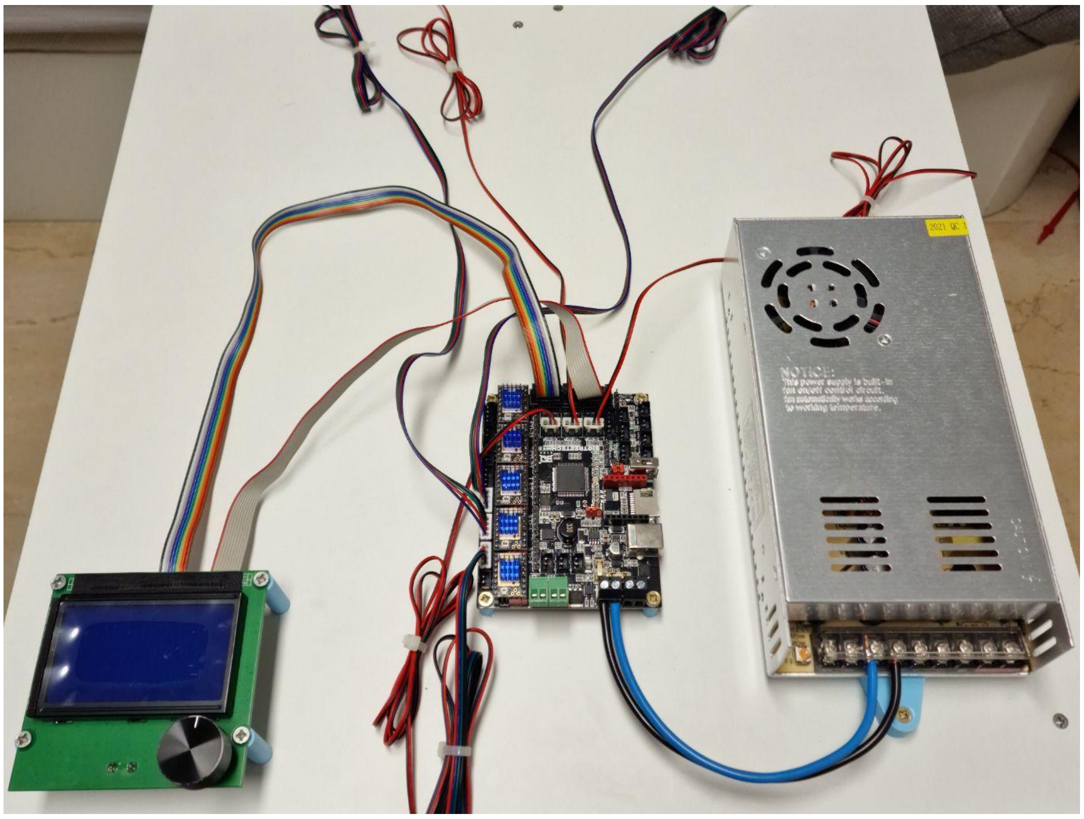


Figura 93: Toda la electrónica conectada

Presupuesto

Id	Tipo	Modelo	Descripción	Ud.	Coste unitario	Coste
1	Electrónica	BIGTREETECH SKR 2.0 con TMC 2208	Placa controladora de 32 bits. Drivers motores paso a paso TMC 2208.	1	60,35€	60,35€
2	Electrónica	Final de carrera	Final de carrera mecánico de 3 pines.	3	1,86€	5,58€
3	Electrónica	Pantalla Creality 12864	Pantalla LCD tipo 12864.	1	18,60€	18,60€
4	Electrónica	Motor Nema 17	Motor paso a paso tipo Nema 17 47 mm.	3	15,79€	15,79€
5	Electrónica	Fuente de alimentación 12V 30 A	Fuente de alimentación de potencia 360 W, 12V 30 A.	1	30,37€	30,37€
6	Electrónica	Cables	Cables varios	1	10,00€	10,00€
7	Estructura	Polea GT2 16 Dientes	Polea para correa tipo GT2 con 16 dientes.	3	1,86€	5,58€
8	Estructura	Plancha de madera	Plancha de madera 50x50x16 cm.	2	6,00€	12,00€
9	Estructura	Listón de madera	Listón de madera 5,5x3,5x62 cm.	3	3,00€	9,00€
10	Estructura	Junta de rótula	Junta de rótula M3.	12	0,90€	10,80€
11	Estructura	Tornillería	Tornillos y tuercas varias.	1	12,00€	12,00€
12	Impresión 3D	Sakata PLA 850	PLA 1.75mm 1KG.	1	18,50€	18,50€
13	Ingeniería	Trabajo Ingeniería	300 h de trabajo de Ingeniero Técnico Industrial	300	25€	7500€
		Subtotal: 7708,57€	**I.G.I.C. (7%):** 539,60€			**Total:** 8248,17€

Conclusiones

El robot Delta es una interesante morfología para el desarrollo de aplicaciones que requieran alta precisión y velocidades de movimientos en espacios reducidos o medios. El objetivo principal de este trabajo ha sido la construcción a bajo coste de un prototipo de robot Delta, para dar a conocer sus prestaciones y familiarizarse con su uso. Asimismo se aportan los modelos cinemáticos del robot y se diseña e implementa la electrónica necesaria para su uso posterior.

Por lo tanto, las principales conclusiones de este trabajo son las siguientes. En primer lugar, se ha conseguido el diseño de un prototipo de bajo costo, realizado utilizando herramientas de bajo coste, programas de diseño asistido por ordenador gratuitos e impresión 3D para la estructura.

En segundo lugar, se han desarrollado los modelos cinemáticos directo e inverso de la estructura, siendo posteriormente confirmados de manera experimental tanto por la herramienta de python como con los modelos realizados en Geogebra.

Además, estos modelos matemáticos se muestran mediante sus expresiones matemáticas completas y, adicionalmente, representados gráficamente utilizando una interfaz sencilla y versátil como es el Geogebra que permite la comprensión detallada de cada paso seguido para resolver la cinemática.

En tercer lugar, se ha diseñado e implementado la estructura mecánica del robot haciendo uso de impresión 3D, siendo el resultado satisfactorio. Adicionalmente, se adjuntan figuras y esquemas de las diferentes piezas que hacen falta para su correcto funcionamiento.

En cuarto lugar, se ha dotado a la estructura mecánica de los actuadores y sensores necesarios para su posterior utilización, así como de la electrónica que, introduciendo un software para la aplicación que se le quiera dar, permite su completo control y manipulación.

Este documento incluye un presupuesto que confirma la realización a bajo coste del prototipo final, y su posibilidad de réplica para su utilización en el ámbito académico, personal o profesional.

Con todo ello, el presente proyecto ha alcanzado los objetivos propuestos inicialmente, exponiendo la información de manera clara y fácilmente comprensible. Además, la redacción y defensa del mismo permite al autor la obtención del Grado en Ingeniería Electrónica Industrial y Automática, y un primer acercamiento a la realización de un proyecto en el ámbito propio de la Ingeniería y la redacción del documento asociado al mismo.

Conclusions

The Delta robot has an interesting morphology for the development of applications that require high precision and movement speeds in small or medium spaces. The main objective of this project has been the low-cost construction of a Delta robot prototype, to publicize its features and become familiar with its use. Likewise, the kinematic models of the robot are provided and the electronics necessary for its subsequent use are designed and implemented.

Therefore, the main conclusions of this work are the following. In the first place, the design of a low-cost prototype has been achieved, made using low-cost tools, free computer-aided design programs and 3D printing for the structure.

Secondly, the direct and inverse kinematic models of the structure have been developed, being subsequently confirmed experimentally both by the python tool and with the models made in Geogebra.

In addition, these mathematical models are shown by means of their complete mathematical expressions and, additionally, represented graphically using a simple and versatile interface such as Geogebra, which allows a detailed understanding of each step followed to solve the kinematics.

Thirdly, the mechanical structure of the robot has been designed and implemented using 3D printing, the result being satisfactory. Additionally, figures and diagrams of the different pieces that are needed for its correct operation are attached.

Fourth, the mechanical structure has been equipped with the necessary actuators and sensors for its subsequent use, as well as electronics that, by introducing software for the desired application, allow complete control and manipulation.

This document includes a budget that confirms the low-cost realization of the final prototype, and its possibility of replicating it for use in the academic, personal or professional field.

With all this, this project has achieved the initially proposed objectives, exposing the information in a clear and easily understandable way. In addition, the writing and defense of this project allows the author to obtain the Degree in Industrial Electronic and Automatic Engineering, and a first approach to the realization of a project in the field of Engineering and the writing of documents associated with it.

Bibliografía

Bibliografía de información

1. https://revistaderobots.com/robots-y-robotica/robot-delta-aplicaciones-y-precios/
2. https://es.wikipedia.org/wiki/Robot_Delta
3. https://urany.net/blog/robots-delta-qu%C3%A9-c%C3%B3mo-cu%C3%A1ndo-y-por-qu%C3%A9#razones-para-incluirlos-en-tus-l%C3%ADneas-de-producci%C3%B3n
4. https://www.edsrobotics.com/blog/automatizacion-procesos-industriales/
5. https://www.helpsystems.com/es/recursos/guias/automatizacion-de-procesos-5-principales-beneficios-en-empresas
6. https://itpeers.com/es/2019/02/01/os-principais-beneficios-da-automatizacao-de-processos-nas-empresas/
7. https://www.enriquedans.com/2018/11/los-dilemas-de-la-automatizacion.html
8. https://elpais.com/retina/2019/05/24/tendencias/1558680372_855666.html
9. https://www.innovacion-tecnologia.com/robotica/que-son-los-robots-delta-aplicaciones-y-precios/
10. https://www.researchgate.net/figure/Figura-27-El-robot-Delta-en-aplicaciones-quirurgicas_fig7_39425275
11. https://www.kuka.com/es-es/productos-servicios/sistemas-de-robot/robot-industrial/kr-delta-robot-hm
12. https://industrial.omron.es/es/products/x-delta-3+1
13. https://new.abb.com/products/robotics/es/robots-industriales/irb-360
14. https://www.geogebra.org/about?lang=es
15. https://www.3dnatives.com/es/ecologico-realmente-filamento-pla-230720192/#!
16. https://descubrearduino.com/que-es-pla/
17. https://abax3dtech.com/2020/12/15/pla-y-petg-caracteristicas-diferenciasy-aplicaciones/
18. https://www.3dnatives.com/es/g-code-proceso-impresion-3d-230920212/
19. https://3dwork.io/guia-completa-skr2/#Breve_introduccion
20. https://www.gams3d.com/2019/06/11/descripcion-y-ajuste-drivers-tmc2208/

Bibliografía de imágenes

1. https://industrial.omron.es/es/products/x-delta-3+1
2. https://www.directindustry.com/prod/fanuc-europe-corporation/product-32007-471024.html
3. https://www.researchgate.net/figure/Figura-27-El-robot-Delta-en-aplicaciones-quirurgicas_fig7_39425275
4. https://new.abb.com/products/robotics/es/robots-industriales/irb-360
5. https://www.kuka.com/es-es/productos-servicios/sistemas-de-robot/robot-industrial/kr-delta-robot-hm
6. https://www.ato.com/nema-17-stepper-motor-6v-1a-1-8-degree-2-phase-4-wires
7. https://supsys.mn/carro-ender-3-k.html
8. https://descubrearduino.com/que-es-pla/
9. https://www.3dnatives.com/es/g-code-proceso-impresion-3d-230920212/

Anexo 1: Código de la herramienta de python

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
import pandas as pd
import time, xlsxwriter

def CinematicaInversa(origen, puntoFinal, radioEfector, largoBiceps, largoBrazo, a,
                       sen120, cos120, sen240, cos240, radioBase):

    #solo para que no existan problemas en el programa
    b = [[0,0,0], [0,0,0], [0,0,0]]

    #Puntos de anclaje de los brazos en el efector final
    c1 = [puntoFinal[0] + radioEfector, puntoFinal[1], puntoFinal[2]]
    c2 = [puntoFinal[0] + radioEfector*cos120, puntoFinal[1] + radioEfector*sen120, puntoFinal[2]] c3 =
    [puntoFinal[0] + radioEfector*cos240, puntoFinal[1] + radioEfector*sen240, puntoFinal[2]]

    c = [c1, c2, c3]

    #t que resuelve la ecuacion parametrica de la recta perpendicular al plano de las circunferencias de los biceps
    #que pasa por el centro de las esferas para encontrar el centro de la circunferencia de intersección entre plano
    #y esfera
    t1 = origen[1] - c1[1]
    t2 = sen120*(c2[0] - origen[0]) + cos120*(origen[1] - c2[1])
    t3 = sen240*(c3[0] - origen[0]) + cos240*(origen[1] - c3[1])

    t = [t1, t2, t3]

    #distancia de esfera centrada en los puntos de anclaje de los brazos en el efector final
    #a plano de las circunferencias de los los biceps que pasa por los puntos de anclaje de los biceps en la base
    d = []

    for k in range(3):

        dActual = np.abs(t[k])

        #Si la distancia al plano es mayor que el radio de la esfera
        if(dActual > largoBrazo):

            return [np.nan, np.nan, np.nan], c, b

        else:
            d.append(dActual)

    #centro de la circunferencia de intersección entre plano y esfera
    i1 = [c1[0], origen[1], c1[2]]
    i2 = [c2[0] - sen120*t2, c2[1] + cos120*t2, c2[2]]
    i3 = [c3[0] - sen240*t3, c3[1] + cos240*t3, c3[2]]

    i = [i1, i2, i3]

    #Cambio a 2D

    theta = []

    for j in range(3):

        #Radio de la circunferencia de intersección entre plano y esfera
        r = np.sqrt(np.square(largoBrazo) - np.square(d[j]))

        #distancia entre los centros de las circunferencias
        dia = np.sqrt(np.square(i[j][0] - a[j][0]) + np.square(i[j][1] - a[j][1]) + np.square(i[j][2] - a[j][2])) if(dia

        > (r + largoBiceps)):

            return [np.nan, np.nan, np.nan], c, b

        #angulo entre la vertical y la recta que une los centros de las circunferencias
        alpha = np.arccos((a[j][2] - i[j][2]) / dia)

        #dependiendo de si alpha esta dentro de beta o fuera
        dIOrigen = np.sqrt(np.square(i[j][0] - origen[0]) + np.square(i[j][1] - origen[1]))
        if((dIOrigen < np.sqrt(np.square(i[j][0] - a[j][0]) + np.square(i[j][1] - a[j][1]))) or (dIOrigen < radioBase)): alpha
            = -alpha

        #angulo entre la recta que une los centros de las circunferencias y el biceps
        beta = np.arccos(-(np.square(r) - np.square(largoBiceps) - np.square(dia)) / (2*largoBiceps*dia))
        #angulo entre la horizontal y el biceps (variable articular)
        theta.append(beta + alpha - 90*np.pi/180)

    b, posible = ComprobarResolucion(a, largoBiceps, largoBrazo, theta, sen120, cos120, sen240, cos240, c)
    if(posible == False):

        return [np.nan, np.nan, np.nan], c, b

    for j in range(3):

        if(b[j][2] < puntoFinal[2]):

            return [np.nan, np.nan, np.nan], c, b

        if(theta[j] > 90):

            return [np.nan, np.nan, np.nan], c, b

    return theta, c, b

def ComprobarResolucion(a, largoBiceps, largoBrazo, theta, sen120, cos120, sen240, cos240, c):

    b1 = [a[0][0] + largoBiceps*np.cos(theta[0]), a[0][1], a[0][2] + largoBiceps*np.sin(theta[0])] b2 =
    [a[1][0] + largoBiceps*cos120*np.cos(theta[1]), a[1][1] + largoBiceps*sen120*np.cos(theta[1]), a[1][2] +
    largoBiceps*np.sin(theta[1])]
    b3 = [a[2][0] + largoBiceps*cos240*np.cos(theta[2]), a[2][1] + largoBiceps*sen240*np.cos(theta[2]),
          a[2][2] + largoBiceps*np.sin(theta[2])]

    b = [b1, b2, b3]
    epsilon = 10e-7

    posible = True

    for i in range(3):

            distanciaBiceps = np.sqrt(np.square(b[i][0] - a[i][0]) + np.square(b[i][1] - a[i][1]) + np.square(b[i][2] - a[i][2]))

        if (np.abs(distanciaBiceps - largoBiceps) > epsilon):
            print("distancia no coincide, brazo ", i, " distancia = ", distanciaBiceps, " largo Biceps = ", largoBiceps)
            posible = False
            time.sleep(8)


            distanciaBrazo = np.sqrt(np.square(b[i][0] - c[i][0]) + np.square(b[i][1] - c[i][1]) + np.square(b[i][2] - c[i][2]))

        if (np.abs(distanciaBrazo - largoBrazo) > epsilon):
            print("distancia no coincide, brazo ", i, " distancia = ", distanciaBrazo, " largo Brazo = ", largoBrazo)
            posible = False
            time.sleep(8)

    return b, posible

def GenerarWorkspace(nombre, origen, radioEfector, largoBiceps, largoBrazo, a, sen120, cos120, sen240, cos240, radioBase):

    print("Comenzamos a generar Workspace")

    xInferior = -350
    xSuperior = 350
    yInferior = -350
    ySuperior = 350
    zInferior = -430
    zSuperior = 0

    pasoX = 2
    pasoY = 2
    pasoZ = 5

    totalPuntos = ((xSuperior - xInferior)/pasoX) * ((ySuperior - yInferior)/pasoY) * ((zSuperior - zInferior)/pasoZ)
    puntoActual = 1

    workspace = [[],[],[]]

    ultimoPuntoValido = False
    puntoAnterior = [0,0,0]

    for i in range(zInferior, zSuperior, pasoZ):
        for j in range(yInferior, ySuperior, pasoY):
            for k in range(xInferior, xSuperior, pasoX):

                print("probando punto:", puntoActual, "/", totalPuntos, " P = (", '{:=4d}'.format(k),
                     ", ", '{:=4d}'.format(j), ", ", '{:=4d}'.format(i), ")")

                theta, c, b = CinematicaInversa(origen, [k,j,i], radioEfector, largoBiceps, largoBrazo,
                                               a, sen120, cos120, sen240, cos240, radioBase)

                    if((np.isnan(theta[0]) != True) and (np.isnan(theta[1]) != True) and (np.isnan(theta[2]) != True)):
                    if(ultimoPuntoValido == False):
                        workspace[0].append(k)
                        workspace[1].append(j)
                        workspace[2].append(i)

                    ultimoPuntoValido = True

                else:

                    if(ultimoPuntoValido == True):
                        workspace[0].append(puntoAnterior[0])
                        workspace[1].append(puntoAnterior[1])
                        workspace[2].append(puntoAnterior[2])

                    ultimoPuntoValido = False

                puntoActual += 1
                puntoAnterior = [k,j,i]

    ExportarDatos(workspace, nombre)
    return workspace

def PlotearWorkspace(workspace, origen, puntoFinal, radioBase, radioEfector, a, b, c, ax):

    print("Ploteamos Workspace")

    ax.scatter(workspace[0], workspace[1], workspace[2], color='grey', alpha=0.3)

    RepresentarRobot(a, b, c, origen, puntoFinal, radioBase, radioEfector, ax)

    plt.show()

    print("ploteado.")

def RepresentarRobot(a, b, c, origen, puntoFinal, radioBase, radioEfector, ax):

    ax.scatter(origen[0],origen[1],origen[2])
    ax.scatter(puntoFinal[0],puntoFinal[1],puntoFinal[2])

    alpha = np.linspace(0, 2 * np.pi, 201)
    baseX = origen[0] + radioBase*np.cos(alpha)
    baseY = origen[1] + radioBase*np.sin(alpha)
    ax.plot(baseX, baseY, origen[2])

    efectorX = puntoFinal[0] + radioEfector*np.cos(alpha)
    efectorY = puntoFinal[1] + radioEfector*np.sin(alpha)
    ax.plot(efectorX, efectorY, puntoFinal[2])

    ax.scatter(a[0][0],a[0][1],a[0][2])
    ax.scatter(a[1][0],a[1][1],a[1][2])
    ax.scatter(a[2][0],a[2][1],a[2][2])

    ax.scatter(b[0][0],b[0][1],b[0][2])
    ax.scatter(b[1][0],b[1][1],b[1][2])
    ax.scatter(b[2][0],b[2][1],b[2][2])

    ax.scatter(c[0][0],c[0][1],c[0][2])
    ax.scatter(c[1][0],c[1][1],c[1][2])
    ax.scatter(c[2][0],c[2][1],c[2][2])

    ax.plot([a[0][0],  b[0][0]],  [a[0][1],  b[0][1]],  [a[0][2],  b[0][2]])
    ax.plot([a[1][0],  b[1][0]],  [a[1][1],  b[1][1]],  [a[1][2],  b[1][2]])
    ax.plot([a[2][0], b[2][0]], [a[2][1], b[2][1]], [a[2][2], b[2][2]])

    ax.plot([c[0][0],  b[0][0]],  [c[0][1],  b[0][1]],  [c[0][2],  b[0][2]])
    ax.plot([c[1][0],  b[1][0]],  [c[1][1],  b[1][1]],  [c[1][2],  b[1][2]])
    ax.plot([c[2][0], b[2][0]], [c[2][1], b[2][1]], [c[2][2], b[2][2]])

def PlotearUnaZWorkspace(workspace, i, radio):

    ax = plt.axes()
    ax.set_xlim(-300, 300)
    ax.set_ylim(-300, 300)

    alpha = np.linspace(0, 2 * np.pi, 201)
    baseX = radio*np.cos(alpha)
    baseY = radio*np.sin(alpha)
    ax.plot(baseX, baseY)

    encontradoEsteZ = False
    indiceInicial = 0
    indiceFinal = 0
    nuevaLista = [[],[]]
    for j in range(len(workspace[2])):

        if((workspace[2][j] == i) and (encontradoEsteZ == False)):

            indiceInicial = j
            encontradoEsteZ = True

        elif ((workspace[2][j] != i) and (encontradoEsteZ == True)):

            indiceFinal = j - 1
            break

    for j in range(indiceInicial, indiceFinal + 1, 1):

        nuevaLista[0].append(workspace[0][j])
        nuevaLista[1].append(workspace[1][j])

    ax.scatter(nuevaLista[0], nuevaLista[1])
    ax.scatter(0,0)
    plt.show()


def ExportarDatos(datos, nombre):

    print("Exportando datos")

    workbook = xlsxwriter.Workbook(nombre)
    worksheet = workbook.add_worksheet()
    row = 0

    for col, data in enumerate(datos):
        worksheet.write_column(row, col, data)

    workbook.close()

    print("Datos exportados")

def ImportarDatos(nombre):

    print("Importando datos")

    df = pd.read_excel (nombre, sheet_name='Sheet1')
    workspaceS = df.values.tolist()
    workspace = [[],[],[]]

    for i in range(len(workspaceS)):

        workspace[0].append(workspaceS[i][0])
        workspace[1].append(workspaceS[i][1])
        workspace[2].append(workspaceS[i][2])

    print("Datos importados")

    return workspace

def main():

    radioBase = 90
    radioEfector = 65
    largoBiceps = 250
    largoBrazo = 220

    origen = [0,0,0]
    puntoFinal = [0,0,-250]

    sen120 = np.sin(120*np.pi/180)
    cos120 = np.cos(120*np.pi/180)
    sen240 = np.sin(240*np.pi/180)
    cos240 = np.cos(240*np.pi/180)

    ax = plt.axes(projection="3d")
    ax.set_xlim3d(-300, 300)
    ax.set_ylim3d(-300, 300)
    ax.set_zlim3d(-600, 100)

    #Puntos de anclaje de los biceps en la base, solo una vez ya que no cambian
    a1 = [origen[0] + radioBase, origen[1], origen[2]]
    a2 = [origen[0] + radioBase*cos120, origen[1] + radioBase*sen120, origen[2]] a3
    = [origen[0] + radioBase*cos240, origen[1] + radioBase*sen240, origen[2]] a =
    [a1, a2, a3]

    print("Resolviendo cinematica ejemplo")
     theta, c, b = CinematicaInversa(origen, puntoFinal, radioEfector, largoBiceps, largoBrazo, a,
                              sen120, cos120, sen240, cos240, radioBase)

    #print("theta1 = ", theta[0]*180/np.pi, "theta2 = ", theta[1]*180/np.pi, "theta3 = ", theta[2]*180/np.pi)

     workspace = GenerarWorkspace("a90b65c250d220.xlsx", origen, radioEfector, largoBiceps, largoBrazo, a,
                                  sen120, cos120, sen240, cos240, radioBase)
    #workspace = ImportarDatos("a90b65c250d220.xlsx")

    PlotearUnaZWorkspace(workspace, -240, 100)
    #PlotearWorkspace(workspace, origen, puntoFinal, radioBase, radioEfector, a, b, c, ax)

main()
```

Anexo II: Datasheets de componentes

Product description

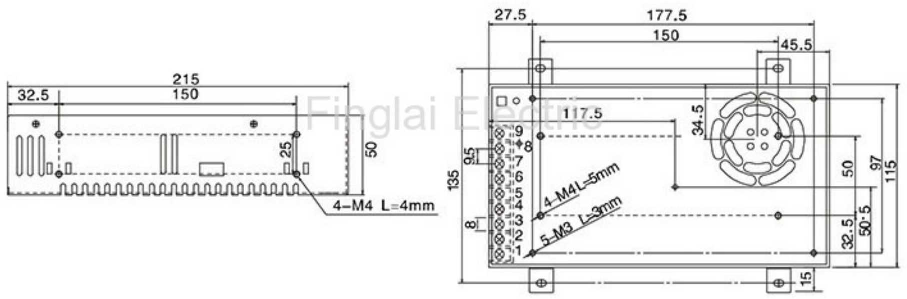
Model number and parameters

Model number	S-360-5	S-360-7.5	S-360-12	S-360-13.5	S-360-15	S-360-24	S-360-27	S-360-48
Online store: This site	US $22.16 / piece Order	US $22.16 / piece Order	Visiting...	US $22.16 / piece Order	US $22.16 / piece Order	US $13.4 / piece Order	US $22.16 / piece Order	US $22.16 / piece Order
DC output voltage	5V	7.5V	12V	13.5V	15V	24V	27V	48V
Output voltage error (Note: 2)	±2%	±2%	±1%	±1%	±1%	±1%	±1%	±1%
Rated output current	60A	40A	30A	26.7A	24A	15A	13.3A	7.5A
Output current rage	0-50A	0-40A	0-30A	0-26.7A	0-24A	0-15A	0-13.3A	0-7.5A
Wave and noise (Note: 3)	75mVp-p	75mVp-p	75mVp-p	75mVp-p	75mVp-p	75mVp-p	75mVp-p	75mVp-p
Inlets tability (Note: 4)	±0.5%	±0.5%	±0.5%	±0.5%	±0.5%	±0.5%	±0.5%	±0.5%
Load stability (Note: 5)	±1%	±1%	±0.5%	±0.5%	±0.5%	±0.5%	±0.5%	±0.5%
DC output power	250W	300W	360W	360W	360W	360W	360W	360W
Effciency	78%	78%	83%	83%	83%	84%	85%	86%
Adjustable range for DC voltage	4.5-5.6V	6-9V	10-13.2V	12-15V	13.5-18V	20-26.4V	26-32V	41-56V

General

Model number	S-360 series
AC input voltage range	85-132VAC/176-264VAC, selected by switch, 47-63Hz; 248-370VDC
Input current	6.5A/115V, 4A/230V
AC inrush current	Cold-start current: 25A/115V, 50A/230V
Leakage current	3.5mA/240VAC max
Overload proctection	105%-150% Type:Foldback current limiting Rest:auto-recovery
Over-voltage protection	5.75-6.75V, 9.4-10.9V, 13.8-16.2V, 15.5-18.2V, 18-21V, 27.6-32.4V, 33.7-39.2V, 57.6-67.2V
Temperature coefficient	±0.03%/℃ (0-50℃)
Setup rise hold up time	200ms, 100ms, 20ms
Vibration	10-500HZ,2G 10min/1cycle, Periodfor 60min, Each axes
Dielectric strength	1.5KVac, 50/60 Hz for 1 min between input and output 1.5KVac, 50/60 Hz for 1 min between input and enclosure 0.5KVac, 50/60 Hz for 1 min between output and enclosure
Isolation resistance	100MΩ min. at 500 VDC
Working temperature humidity	-10℃-60℃ (Refer to output characteristic figure), 20%-90%RH
Store temperature humidity	-20℃-85℃, 10%-95%RH
External dimension	215x115x50 mm
Weight	1.15kg
Safety standards	UL1950
EMC standards	FCC part 15J condition level A
Note	1. The testing condition for the above paramenter is: 230VAC input, rated load, 25℃, Humidity 70%RH. 2. Error, include the setting error, line stability and load stability. (Note:5) 3. Wave test: adopting A12 double wire for 20MHZ,and 0.1uF-47uF capacitor short-circuit for interrupting. 4. Inlet voltage stability test: when is over load,the lowest voltage of enter is representative to the highest voltage 5. Load stability test: The load is from 0% to 100%.

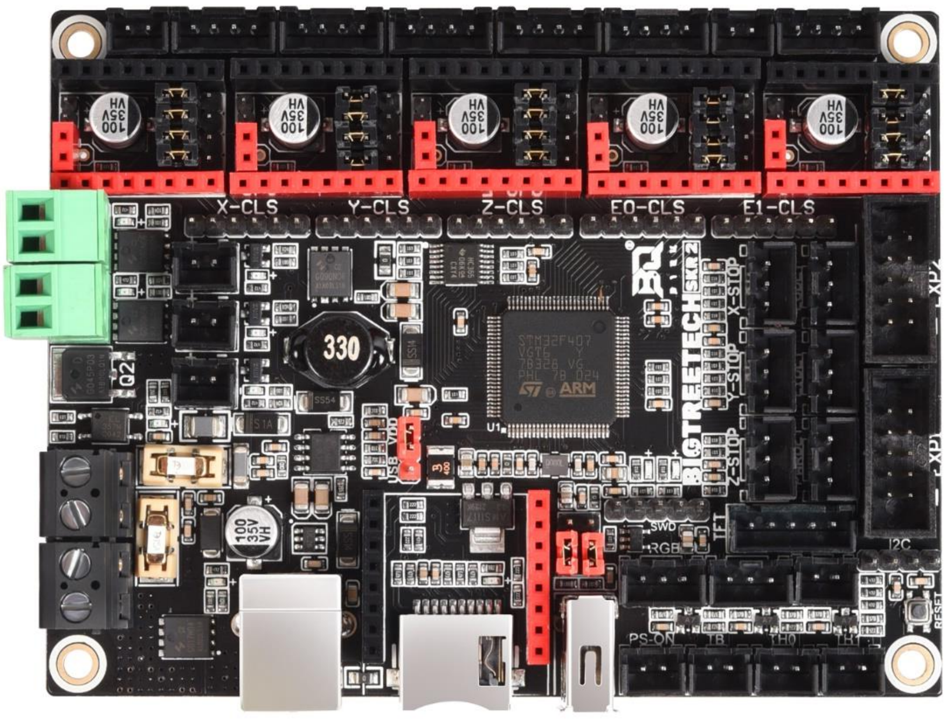
S-360 series general switching power supply mounting dimensions



Shenzhen BIGTREE technology co., LTD.
BIG TREE TECH

BIGTREETECH SKR 2

User Guide



I. Introduction to SKR Motherboard

The BIGTREETECH SKR 2 motherboard is a 32-bit 3D printer motherboard designed and manufactured by the 3D printing team of Shenzhen BIQU Technology Co., Ltd. It is a successor to the SKR1.4 and SKR1.4 Turbo series of motherboards and adds many features that we are sure the community will love.

1. Main board features:

1) Uses a 32-bit ARM Cortex-M4 series STM32F407VGT6 main control chip with a core frequency of *168MHz* that packs enough performance to run even the most demanding UI while ensuring stutter free printing.
2) Integrates a AOZ1284PI power regulator which supports 12-24V power input and a maximum output current of 4A – enough to power external components such as LEDs and even a raspberry pi;

3) Support all versions of the company's serial screen, SPI screen and LCD screen;

4) Upgrade the configuration firmware via SD card. This burning method is simple, convenient and efficient;

5) TMC Motor drivers can be used in SPI or UART mode by simply adjusting the onboard jumpers beneath each motor driver. Additionally you can connect or disconnect the TMC DIAG pin using an onboard jumper allowing you to use hard endstops or sensorless homing without the need to cut any pins.

6) Supports functions such as power loss print resume, filament runout detection, shutdown after printing, BLtouch and other ABL sensors, RGB lights, etc. Note that external modules will be required for many of these functions;

7) Use high-performance MOSFET to reduce heat generation;

8) The use of replaceable fuses makes the replacement process more convenient;

9) Includes a protection circuit on thermistor inputs which protects against short circuits between the heater cartridge and the bed heating element. This is a common mistake made by many users when replacing a nozzle or working on the hotend so we expect this feature to be a welcome one;

10) Includes an additional heater protection circuit which protects against runaway heating on the bed and hotend heaters. By default, the heaters are off so if the MCU or MOSFETs are damaged, the heaters will shut down instead of enter into thermal runaway;

11) Introduces a new “anti-reversal” stepper driver protection which is achieved through a combination of new hardware and Marlin firmware. This protects against motherboard or driver damage due to incorrect driver insertion;

12) ESP8266 WIFI module (ESP12S or ESP-07) interface for customers to use RRF firmware;

13) Onboard push-pull TF card slot (SDIO working mode) and U disk interface;

14) A filter circuit is added to the power input terminal to reduce ripple and noise interference;
15) The fuses now use a smaller package which leaves more space on the PCB for more features and makes it easier to swap them out;
16) The number of PWM controllable fan interfaces has been increased from one to three.

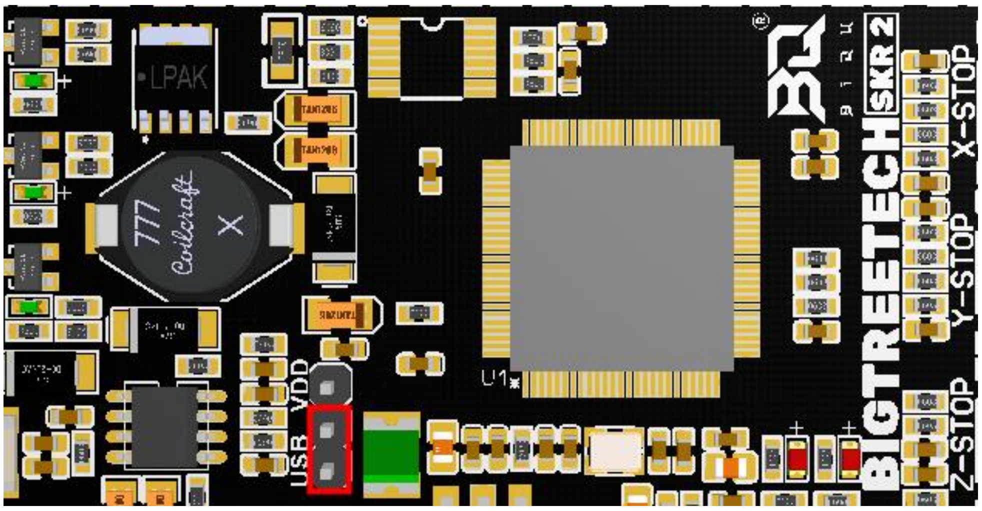
2. Main board parameters:

Appearance size: 110*85mm For details, please refer to: BIGTREETECH SKR 2-SIZE.pdf
Installation size: 102*76mm
Microprocessor: ARM Cortex-M4 CPU STM32F407VGT6
Input voltage: DC12V-DC24V
Logic voltage: DC 3.3V
Firmware support: Marlin, Reprap Firmware
WIFI interface: ESP-12S, ESP-07S
Fan interface: three CNC fans, two normally closed fans (non-controllable)
Expansion interface: I2C, Servos, Probe, PS-ON, PWR-DET, Fil-DET, RGB, etc.
Motor driver: support TMC5160, TMC2209, TMC2225, TMC2226, TMC2208, TMC2130, ST820, LV8729, DRV8825, A4988, etc., and can be externally connected to a motor drive
Drive working mode support: SPI, UART, STEP/DIR
Motor drive interface: X, Y, Z (double Z axis), E0, E1, five channels (each channel has a closed loop drive interface)
Temperature sensor interface: TH0, TH1, TB, 3 channels 100K NTC (thermal resistance)
Display: serial touch screen, SPI touch screen, LCD display
PC communication interface: square USB, easy to plug and unplug, communication baud rate 115200
Support file format: G-code
Support machine structure: XYZ, delta, kossel, Ultimaker, corexy
Recommended software: Cura, Simplify3D, pronterface, Repetier-host, Makerware

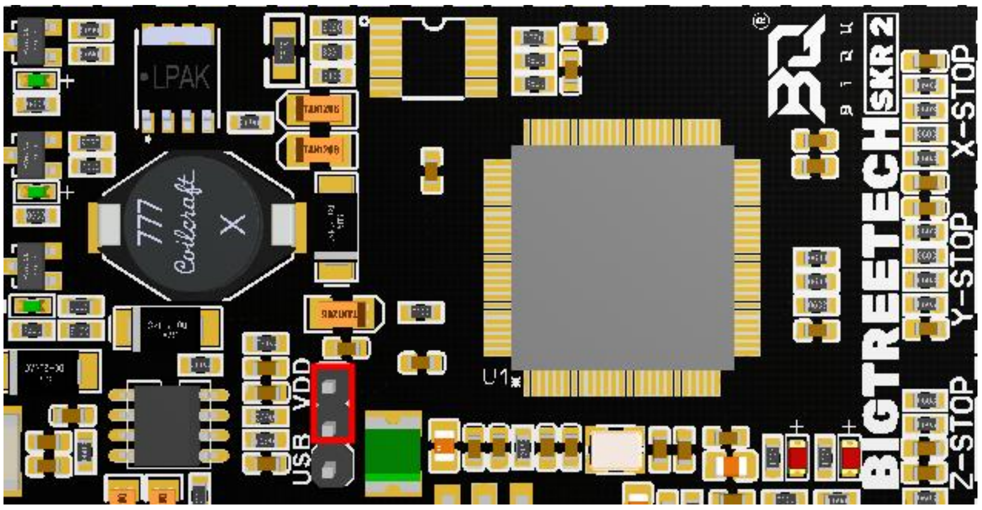
II. Power on the motherboard

After the SKR motherboard is powered on, the D6 LED will light up, indicating that the power supply is at nominal levels. The 5V SEL jumper, in the middle of the board, allows you to select whether you would like to use the 5V USB power input or the onboard 5V regulated supply. Configure the jumper as shown below:

1) When using USB to power the motherboard:

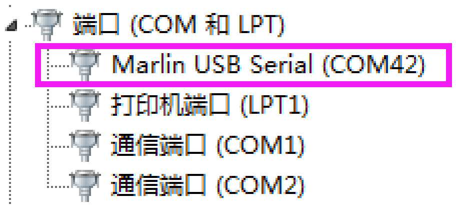


2) When using 12V-24V power supply:



III. Communication between motherboard and computer

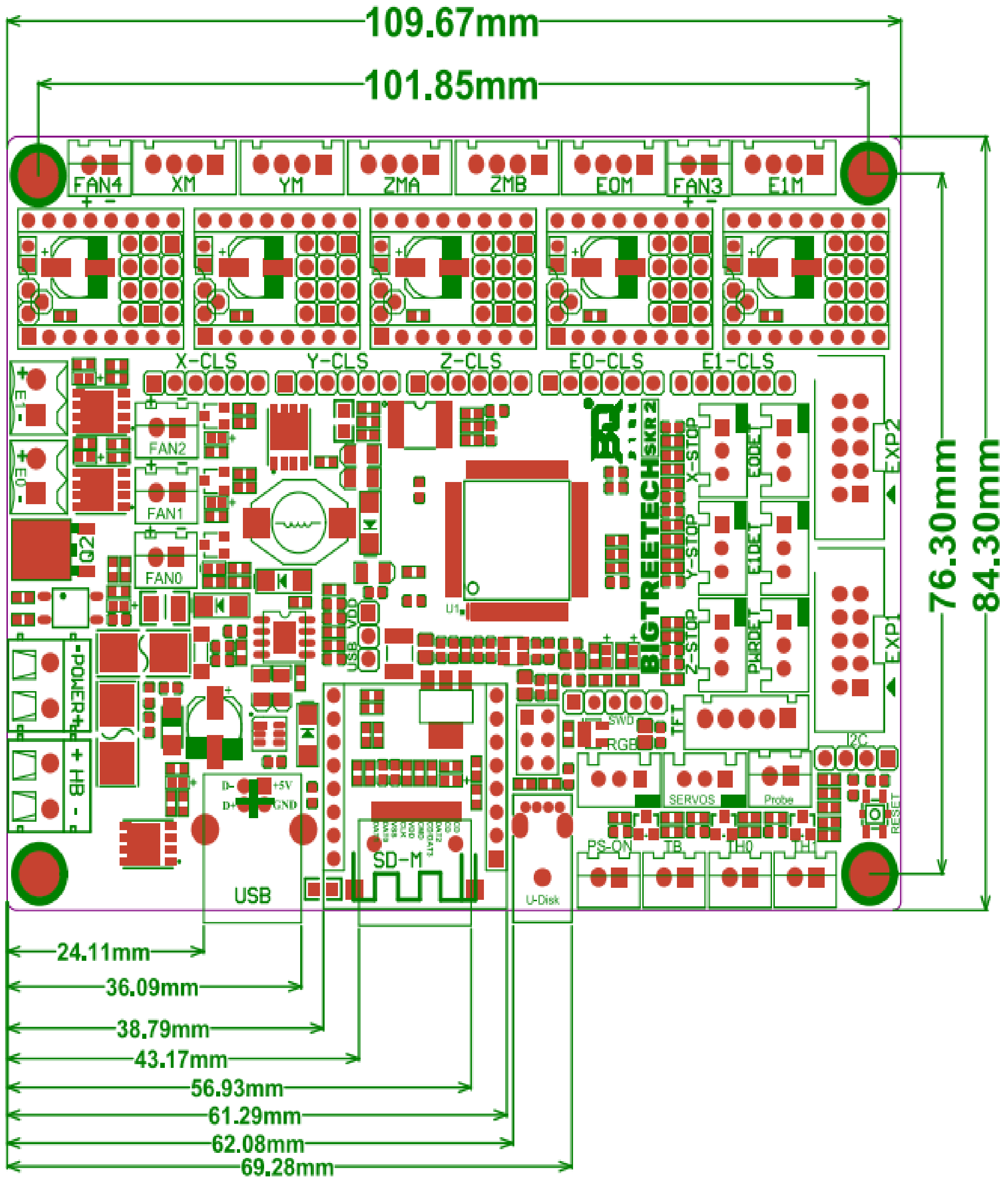
When using Marlin2.0 firmware the motherboard will enumerate as a virtual serial port in both macOS and windows. An example of the enumerated device within the windows device manager is shown below.



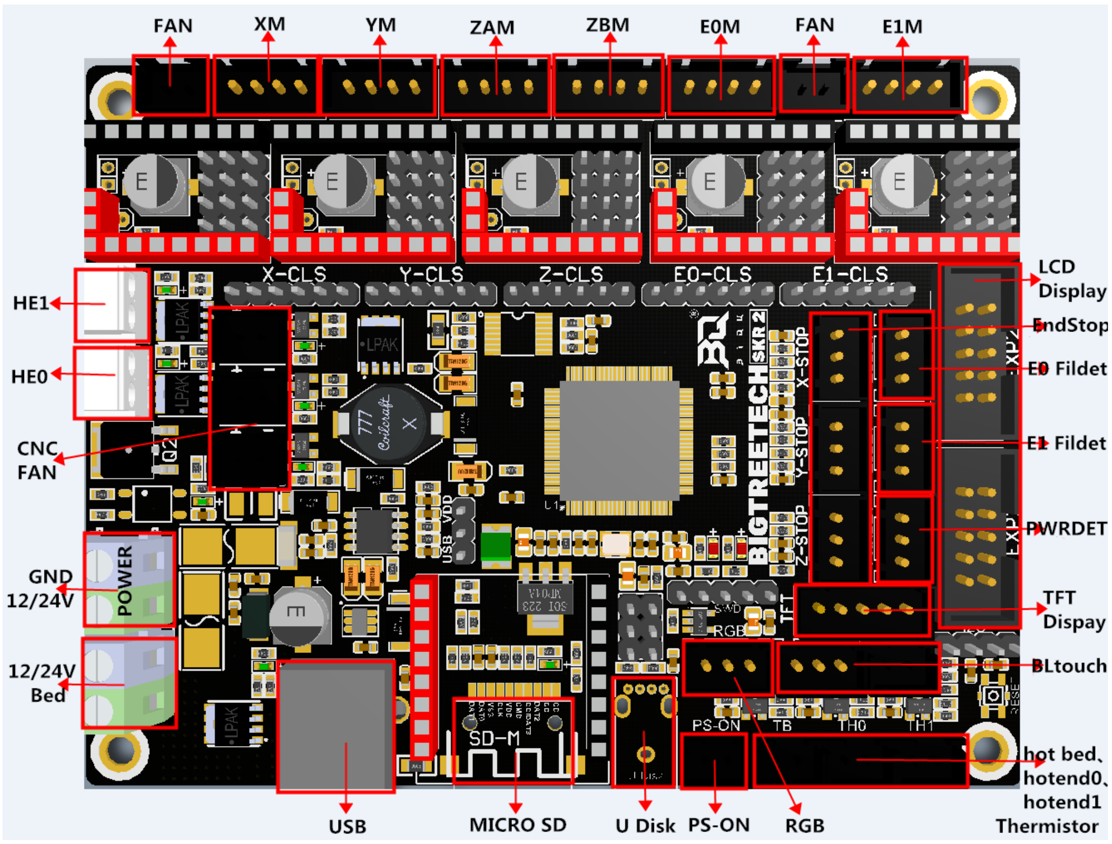
```
端口 (COM 和 LPT)
    Marlin USB Serial (COM42)
    打印机端口 (LPT1)
    通信端口 (COM1)
    通信端口 (COM2)
```

IV. Motherboard interface description

1. Motherboard size:

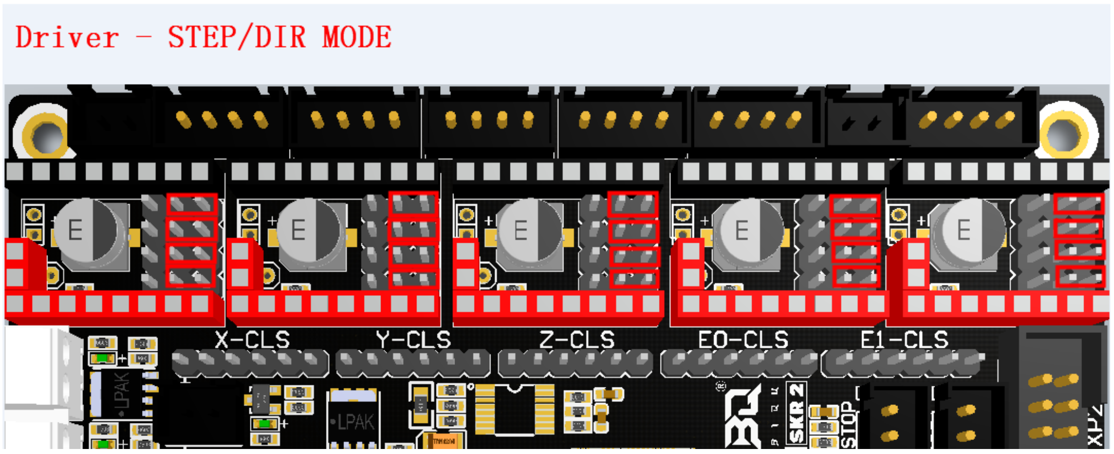


2. Motherboard wiring diagram:



3. Selection method of drive working mode:

① Normal STEP/DIR mode: (such as: A4988, DRV8825, LV8729, ST820, etc.) according to the drive subdivision table to select the shorting cap to short-circuit MS0-MS2.



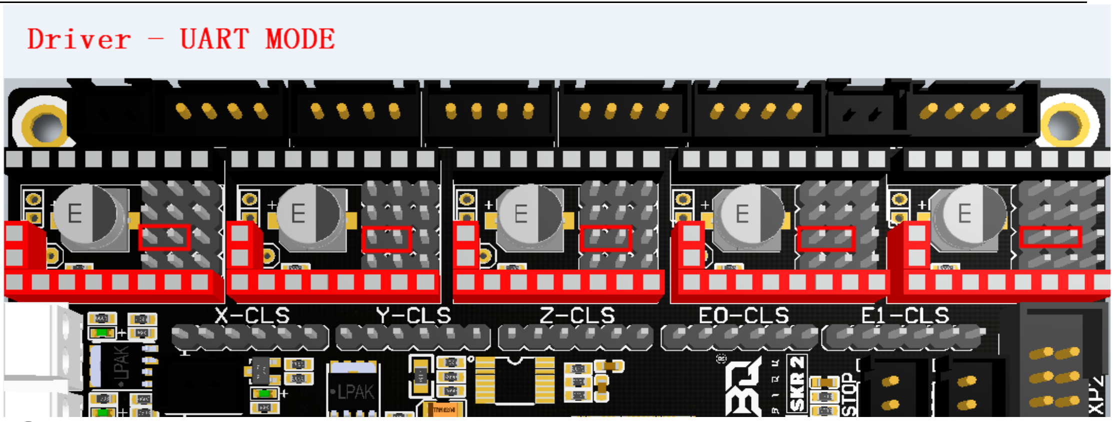
Note: If you use A4988 or DRV8825 driver, you must short-circuit RST and SLP with a jumper cap to work normally.

驱动芯片	MS1	MS2	MS3	细分	Excitation Mode
A4988 最大 16 细分 35V 2A	L	L	L	Full Step	2 Phase
	H	L	L	1/2	1-2 Phase
	L	H	L	1/4	W1-2 Phase
	H	H	L	1/8	2W1-2 Phase
	H	H	H	1/16	4W1-2 Phase
驱动电流计算公式 Rs=0.1Ω	Imax = Vref / (8 * Rs)				

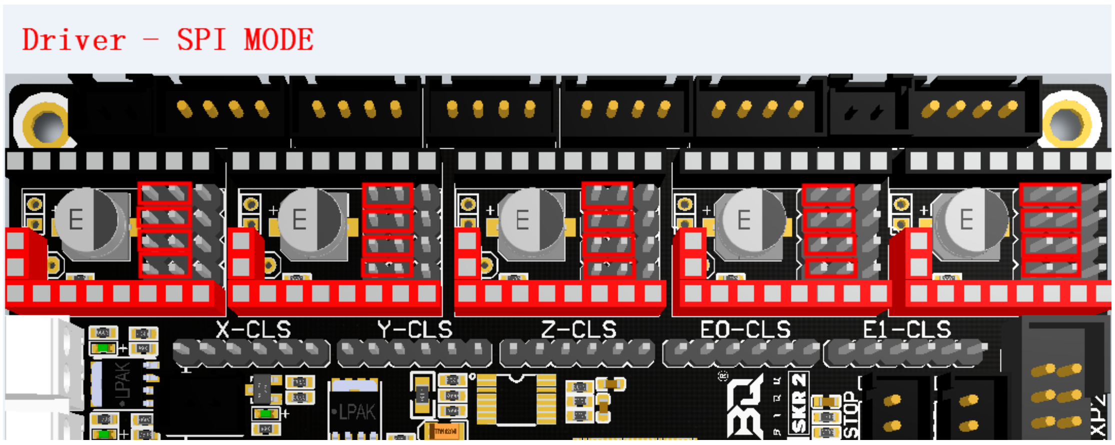
驱动芯片	MD3	MD2	MD1	细分	Excitation Mode
LV8729 最大 128 细分 36V 1.8A	L	L	L	Full Step	2 Phase
	L	L	H	1/2	1-2 Phase
	L	H	L	1/4	W1-2 Phase
	L	H	H	1/8	2W1-2 Phase
	H	L	L	1/16	4W1-2 Phase
	H	L	H	1/32	8W1-2 Phase
	H	H	L	1/64	16W1-2 Phase
	H	H	H	1/128	32W1-2 Phase
驱动电流计算公式 Rs=0.22Ω	$I_{OUT} = (VREF / 5) / RF1$				

驱动芯片	MODE2	MODE1	MODE0	细分	Excitation Mode
DRV8825 最大 32 细分 8.2V-45V 2.5A at 24V T=25℃	L	L	L	Full Step	2 Phase
	L	L	H	1/2	1-2 Phase
	L	H	L	1/4	W1-2 Phase
	L	H	H	1/8	
	H	L	L	1/16	
	H	L	H	1/32	
	H	H	L	1/32	
	H	H	H	1/32	
驱动电流计算公式 Rs=0.1Ω	$I_{CHOP} = \frac{V_{REFX}}{5 \bullet R_{ISENSE}}$				

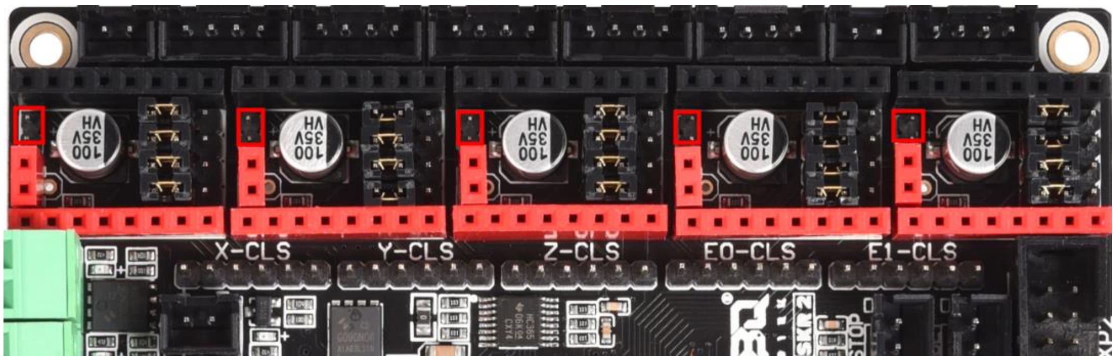
② UART mode driven by TMC: (such as: TMC2208, TMC2209, TMC2225, etc.) Each axis uses a shorting cap to short the position of the red box in the figure, and the subdivision and drive current are set by firmware.



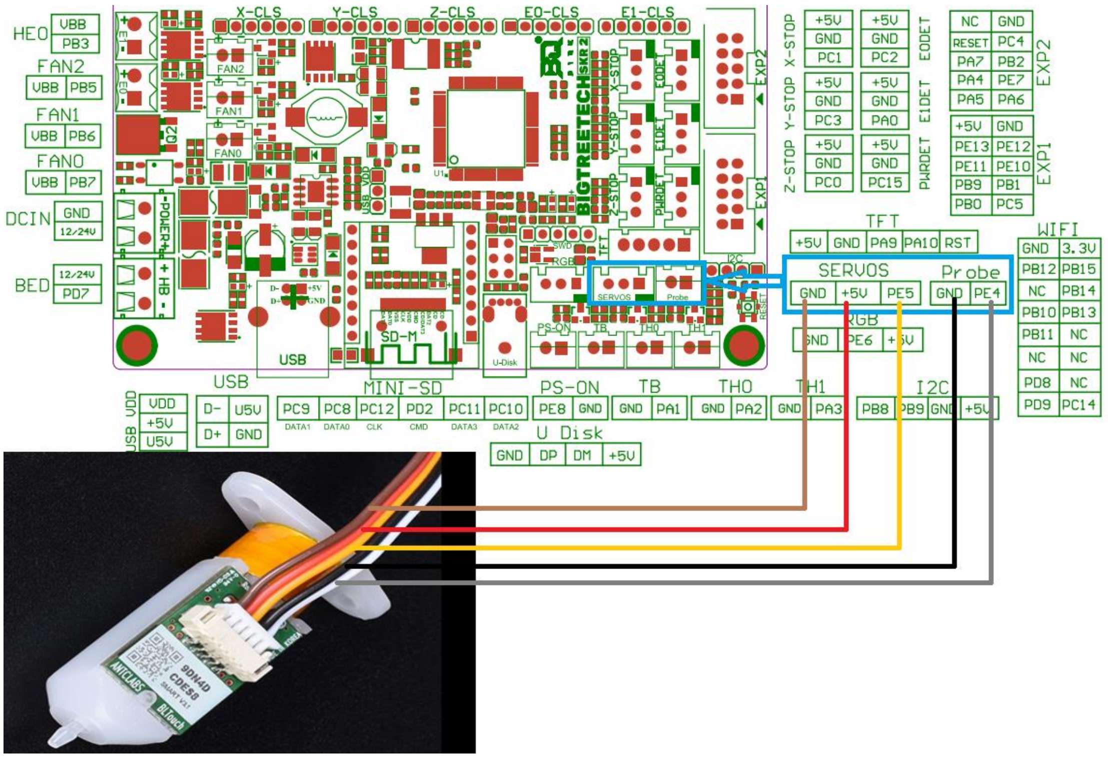
③ SPI mode of TMC drive: (such as: TMC2130, TMC5160, TMC5161, etc.) Use four shorting caps for each axis to short the position of the red box in the figure, and the subdivision and drive current can be set by firmware.



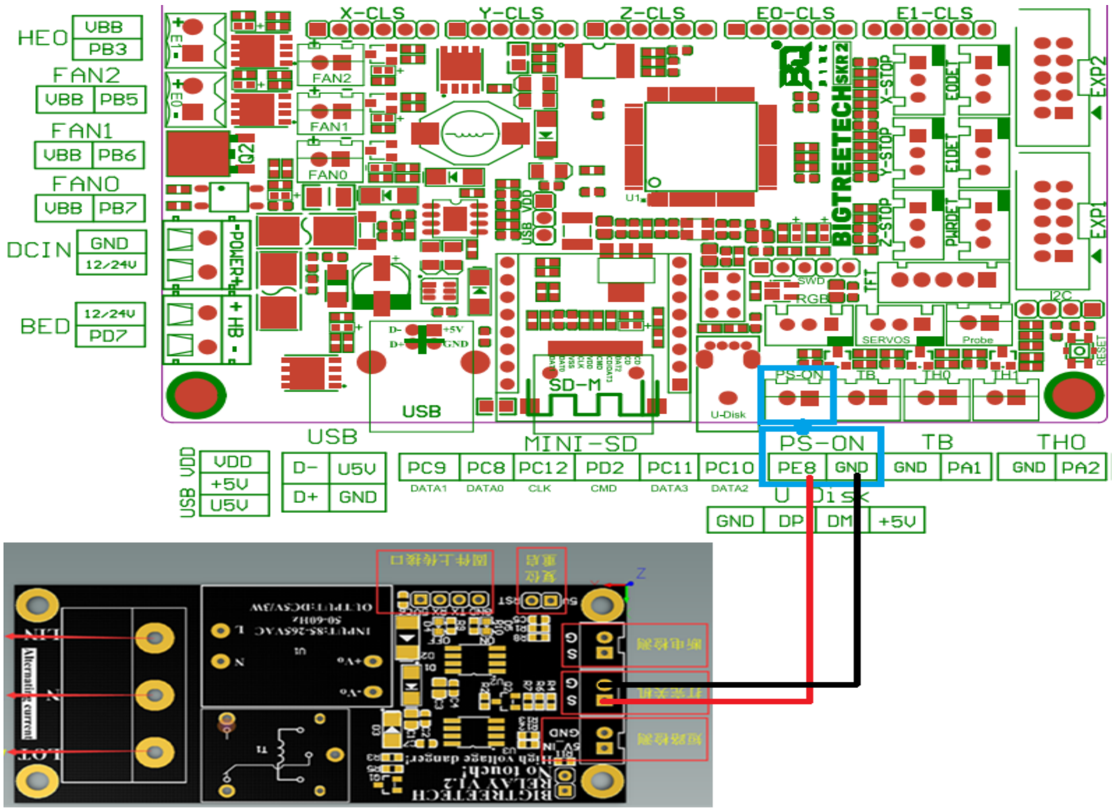
④DIAG Pin of TMC drive:
As shown in the picture, insert the short-circuit cap when using the sensorlesshoming function, and do not insert it when not using it. There is no need to cut the driving diag pin.



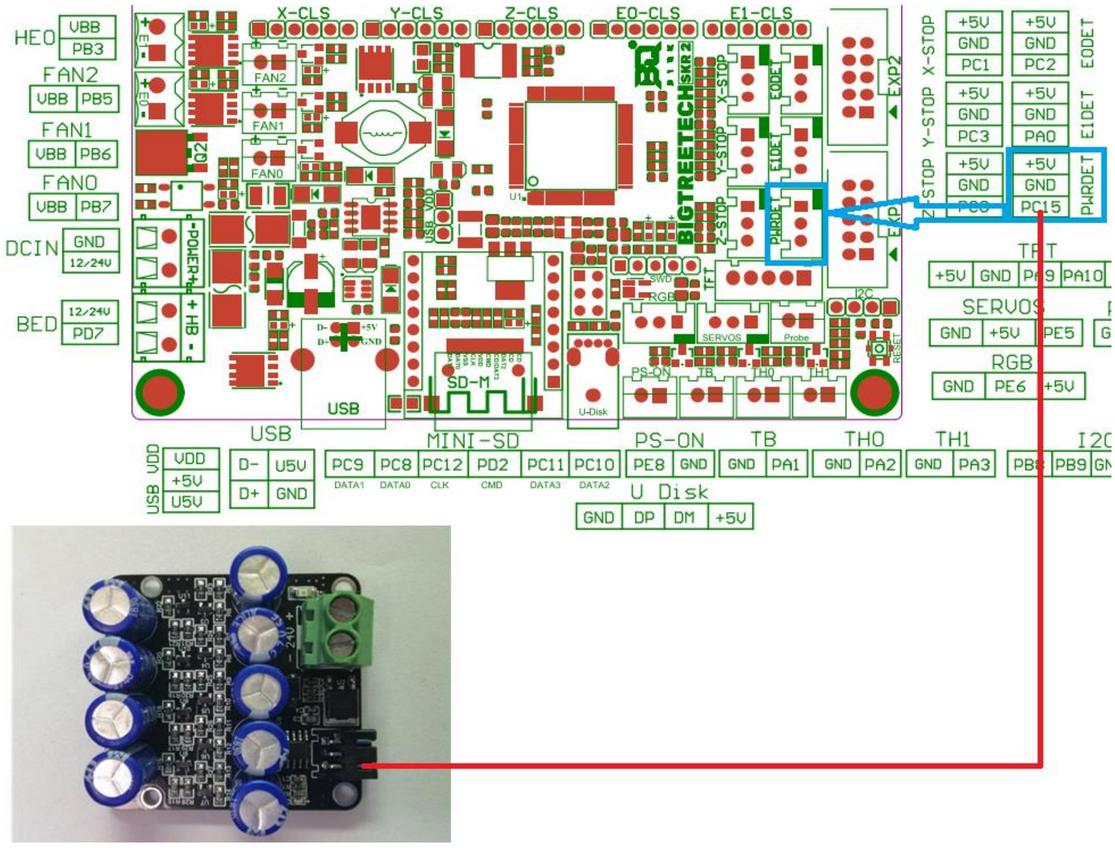
4. The connection between BIGTREETECH SKR 2 and BLtouch:



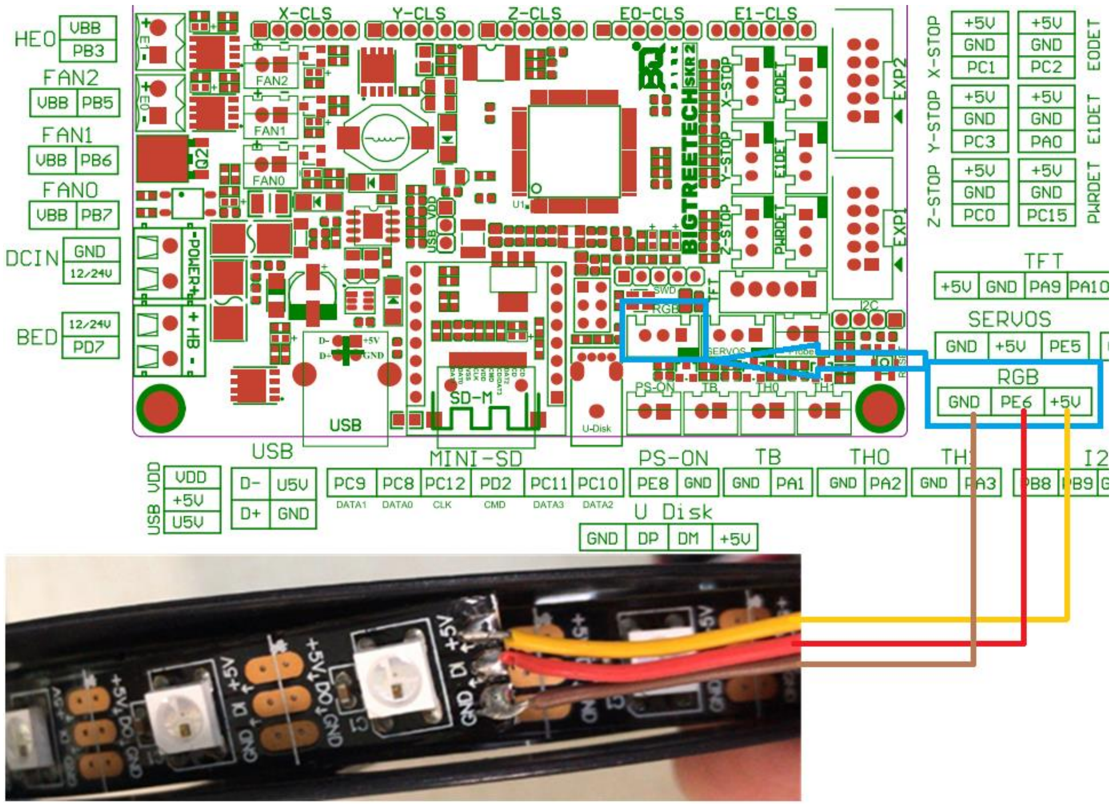
5. Connection between BIGTREETECH SKR 2 and shutdown (Relay V1.2):



6. The connection between BIGTREETECH SKR 2 and power failure (UPS 24V V1.0):



7. Connection between BIGTREETECH SKR 2 and RGB-LED drive:

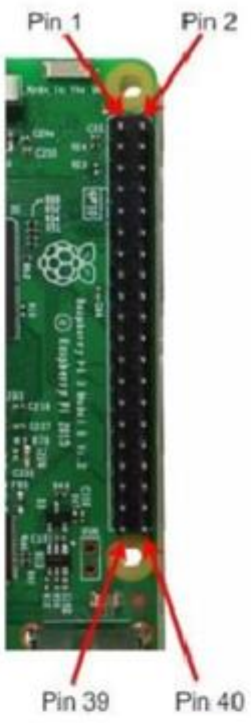


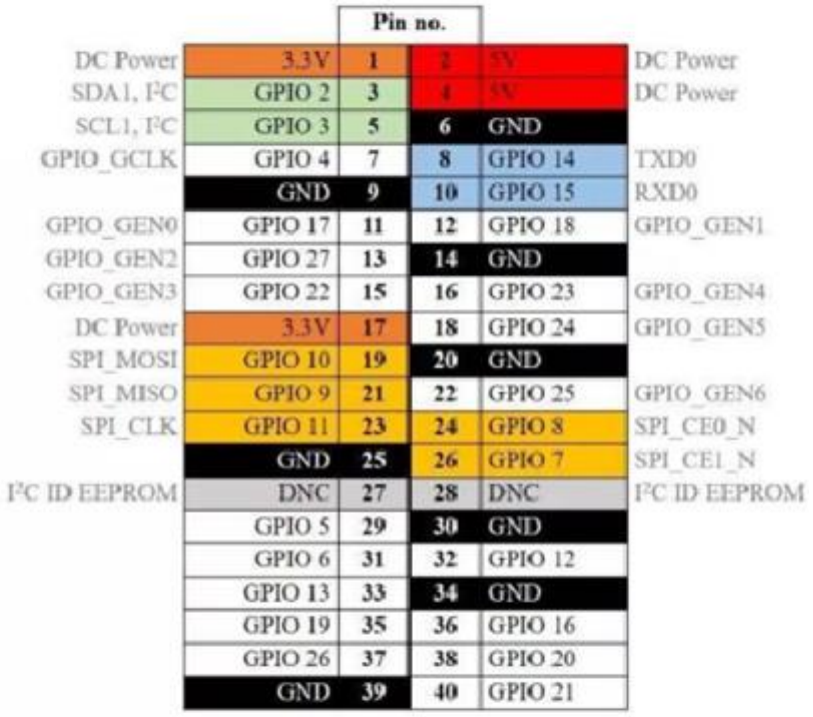
8.Connecting the BIGTREETECH SKR 2 to a Raspberry Pi using TTL UART:

No need to connect V+

Rasberry Pi 3 and SKR V1.4 both with 3.3V logic.

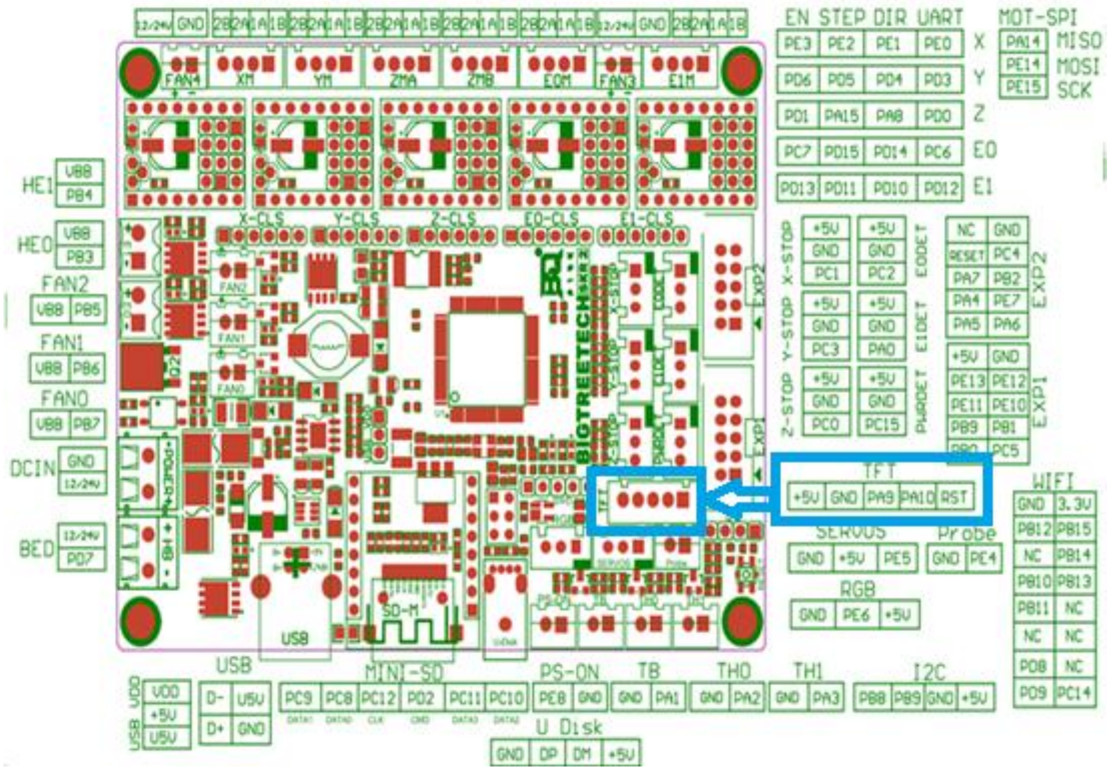
PI 3B+ @ GPIO connector TXD0 - RXD0 – GND (8-0-10)



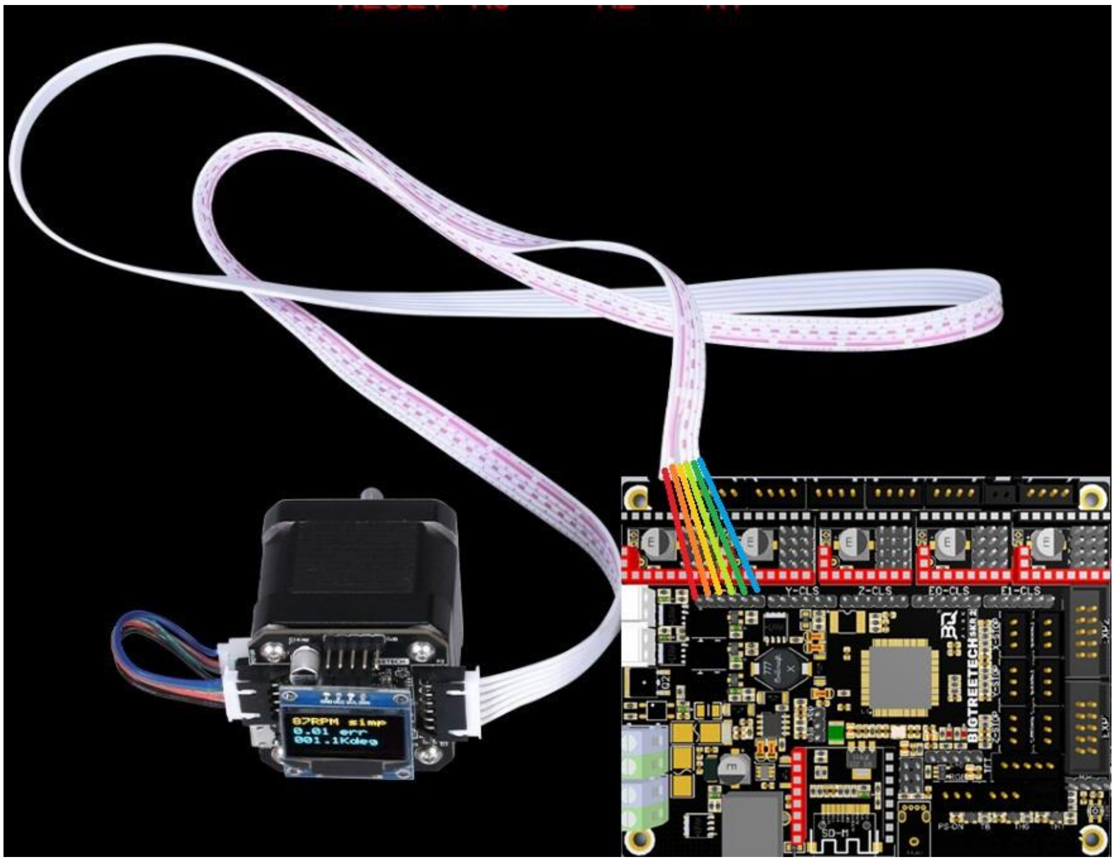


		Pin no.			
DC Power	3.3V	1	2	5V	DC Power
SDA1, I²C	GPIO 2	3	4	5V	DC Power
SCL1, I²C	GPIO 3	5	6	GND	
GPIO_GCLK	GPIO 4	7	8	GPIO 14	TXD0
	GND	9	10	GPIO 15	RXD0
GPIO_GEN0	GPIO 17	11	12	GPIO 18	GPIO_GEN1
GPIO_GEN2	GPIO 27	13	14	GND	
GPIO_GEN3	GPIO 22	15	16	GPIO 23	GPIO_GEN4
DC Power	3.3V	17	18	GPIO 24	GPIO_GEN5
SPI_MOSI	GPIO 10	19	20	GND	
SPI_MISO	GPIO 9	21	22	GPIO 25	GPIO_GEN6
SPI_CLK	GPIO 11	23	24	GPIO 8	SPI_CE0_N
	GND	25	26	GPIO 7	SPI_CE1_N
I²C ID EEPROM	DNC	27	28	DNC	I²C ID EEPROM
	GPIO 5	29	30	GND	
	GPIO 6	31	32	GPIO 12	
	GPIO 13	33	34	GND	
	GPIO 19	35	36	GPIO 16	
	GPIO 26	37	38	GPIO 20	
	GND	39	40	GPIO 21	

BIGTREETECH SKR 2 @ TFT connector Tx-Rx-GND

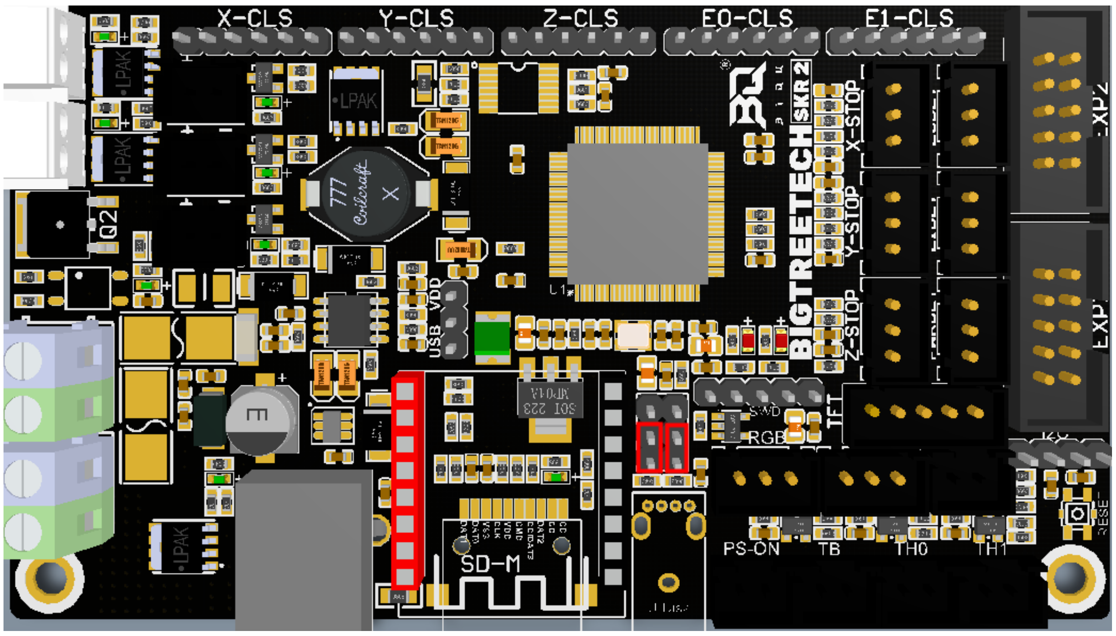


9. Connection between BIGTREETECH SKR 2 and closed-loop drive:

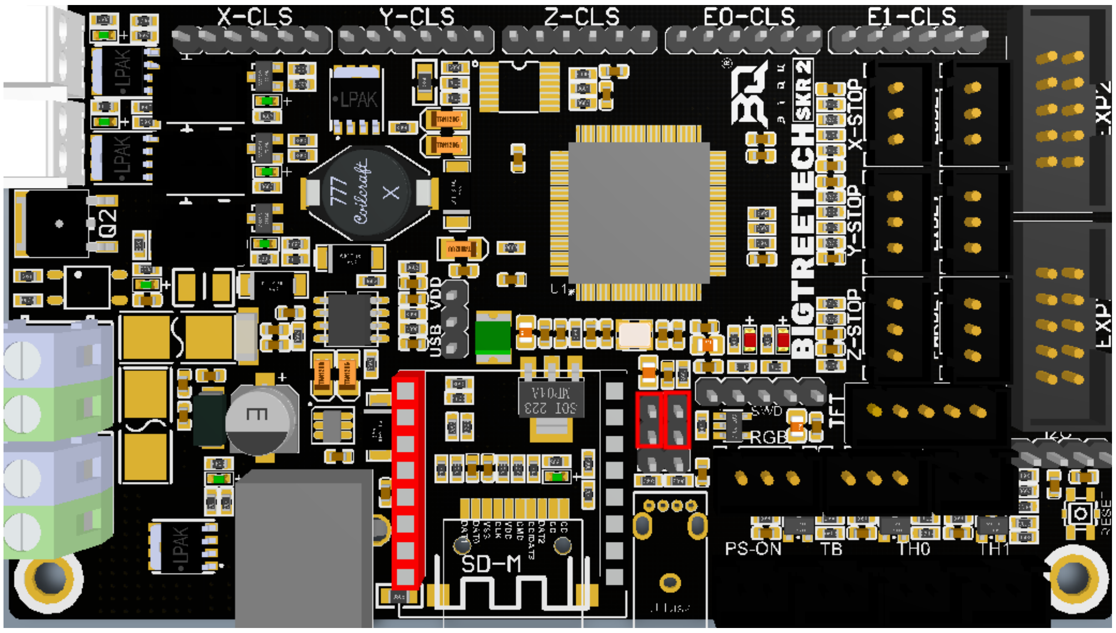


10. U disk function and RRF WIFI function selection:

①U disk function: When the U disk function is selected, the WIFI module only works in UART mode. At this time, RRF firmware cannot be used, only Marlin firmware can be used;



②RRF WIFI function: If selected (as per the jumper configuration below), the U disk function will not be available as WIFI will work through SPI mode, and RRF firmware can be used.



V. Motherboard firmware description

The motherboard comes pre-installed with firmware that is used for testing at the factory. The firmware is configured for use with an i3 style printer however you will likely find it useful to configure and install your own firmware.

Due to the abundance of printer types and firmware distributions in the market, BIGTREETECH cannot offer customized firmware for each application. You may, however, be able to find firmware for you application by trying one of the following:

Ask customer service or technical personnel to obtain;

Compile your own using a distribution of your choice.

Download from our original website:https://github.com/bigtreetech

If you decide to use Marlin then you will need to download the source code, use Visual Studio Code to configure the code to your application, find the firmware.bin file within the pio-build folder, copy it to the SD card and then reset the motherboard. There is an abundance of information on the internet which elaborates on these steps should you need more guidance.

If you require more information about the pin numbers when compiling your firmware, please refer to the BIGTREETECH SKR 2-PIN.pdf document.

VI. Matters needing attention:

1. The jumper that allows selection between USB power or on-board power must be inserted so as to select one of the two options otherwise no power will be supplied to the logic section of the motherboard and the power LED will not light up.

2. The power of the hot bed connected to the main board must be less than or equal to 10A. If you want to use a high-power hot bed, it is recommended to choose a hot bed with 24V power supply and use 24V to power the main board;

3. The jumpers that allow selection between SPI WiFi (for RRF) or U disk functionality should be inserted to select at least one option.;

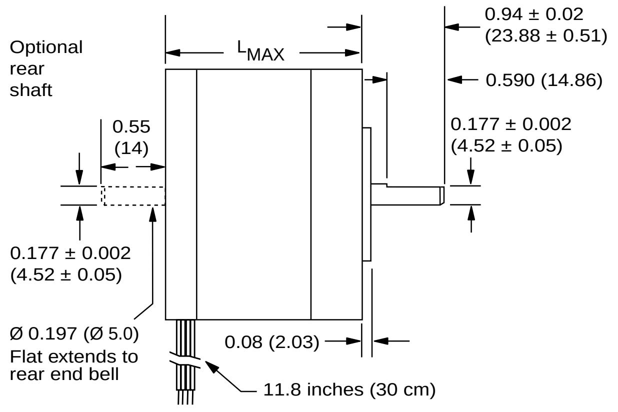
4.The driver anti-reversal insertion function is a newly developed function from BIGTREETECH. It is currently only supported in Marlin firmware and therefore not available when using RRF firmware. Therefore, when you are not using Marlin firmware, please carefully check whether the driver is inserted correctly to avoid causing damage to the driver and/or the motherboard. Customers can also choose to install the Marlin firmware first and then install the RRF firmware after they are confident that the hardware installation is correct.

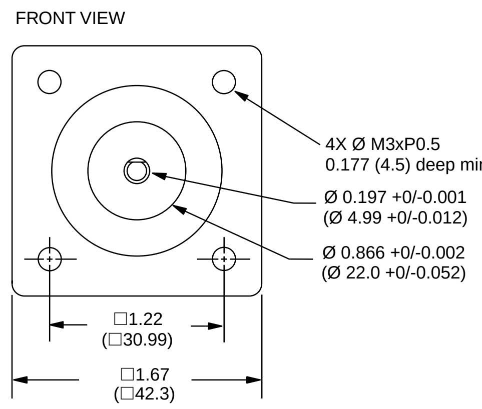
5. This motherboard uses a push-pull type SD slot which does not eject when pushing on the card after insertion. The installation stroke is far shorter than a push-push type SD slot and therefore customers should be gentle when inserting or removing a card. Returns for damage caused by mis-handling the SD card will not be entertained.
d

Quick Reference

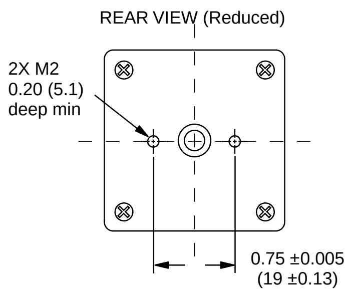
NEMA size 17 1.8° 2-phase stepper motor

Mechanical Specifications
Dimensions in inches (mm)





FRONT VIEW
4X Ø M3xP0.5 0.177 (4.5) deep min
Ø 0.197 +0/-0.001 (Ø 4.99 +0/-0.012)
Ø 0.866 +0/-0.002 (Ø 22.0 +0/-0.052)
□1.22 (□30.99)
□1.67 (□42.3)



Notes and Warnings

Installation, configuration and maintenance must be carried out by qualified technicians only. You must have detailed information to be able to carry out this work.

- Unexpected dangers may be encountered when working with this product!
- Incorrect use may destroy this product and connected components!

For more information, go to www.imshome.com

Specifications

1.5 Amp motors		Single length	Double length	Triple length
Part number		**M-1713-1.5 •** *(1)*	**M-1715-1.5•** *(1)*	**M-1719-1.5•** *(1)*
Holding torque	oz-in	32	60	75
	N-cm	23	42	53
Detent torque	oz-in	1.7	2.1	3.5
	N-cm	1.2	1.5	2.5
Rotor inertia	oz-in-sec^2	0.000538	0.0008037	0.0011562
	kg-cm^2	0.038	0.057	0.082
Weight	oz	7.4	8.1	12.7
	grams	210	230	360
Phase current	amps	1.5	1.5	1.5
Phase resistance	ohms	1.3	2.1	2.0
Phase inductance	mH	2.1	5.0	3.85

(1) Indicate S for single-shaft or D for double-shaft. Example M-1713-1.5S

Motor stack length inches (mm)	Single	Double	Triple
LMAX	1.34 (34.0)	1.57 (40)	1.89 (48)

Wiring and Connections

Signals and wire colors	
Phase A	Red
Phase /A	Blue
Phase B	Green
Phase /B	Black

Part Numbers

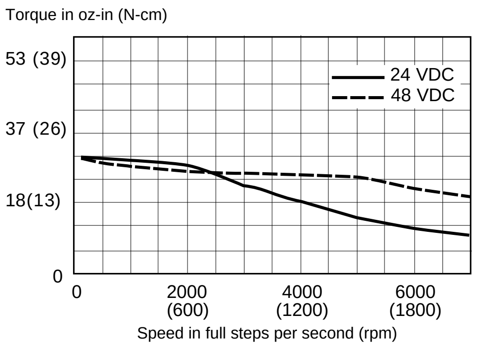
Example:	**M - 1 7 1 3 - 1 . 5 S**
Stepper motor frame size **M-17** = NEMA 17 (1.7"/42 mm)	**M - 1 7** 1 3 - 1 . 5 S
Motor length **13-** = single stack **15-** = double stack **19-** = triple stack	M - 1 7 **1 3** 1 . 5 S
Phase current **1.5** = 1.5 Amps	M - 1 7 1 3 - **1 . 5** S
Shaft **S** = single, front shaft only **D** = double, front and rear shafts	M - 1 7 1 3 - 1 . 5 **S**
Optional optical encoder *(1)* **ES** = Single-end **ED** = Differential **Line count** 100, 200, 250, 400, 500 or 1000 *(2)*	M - 1 7 1 3 - 1 . 5 **E S 1 0 0**

(1) An encoder replaces the shaft designator in the part number.
(2) All encoders have an index mark, except the 1000 line count version.

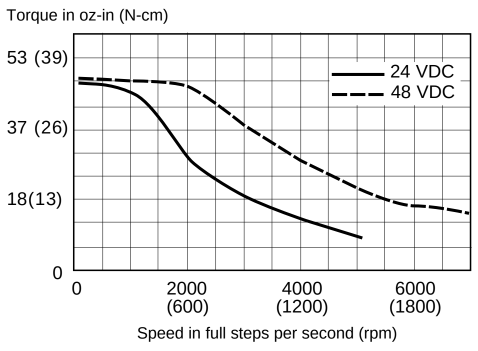
Torque-speed performance

Measured at 1.5 Amps RMS

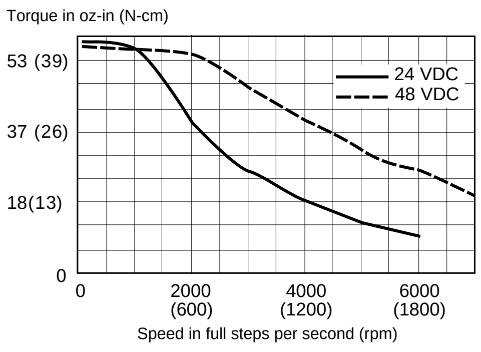
M-1713-1.5



M-1715-1.5

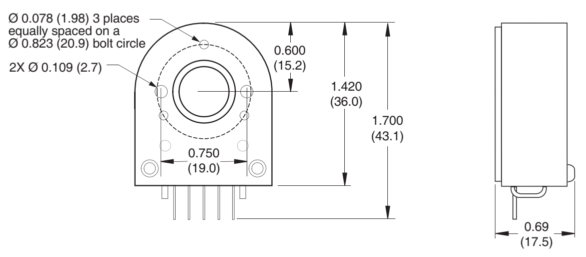


M-1719-1.5



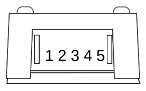
Optical Encoder Option

Dimensions in inches (mm)



Connectivity

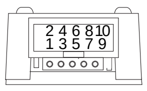
single-end encoder



	wire	*function*
1	Brown	Ground
2	Violet	Index
3	Blue	Channel A
4	Orange	+5 VDC input
5	Yellow	Channel B

optional interface cable available: ES-CABLE-2

differential encoder

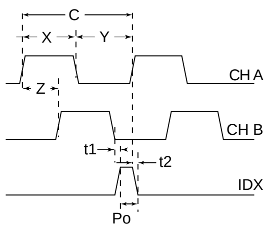


pin	*function*	*pin*	*function*
1	no connect	6	Channel A+
2	+5 VDC input	7	Channel B –
3	Ground	8	Channel B+
4	no connect	9	Index –
5	Channel A –	10	Index +

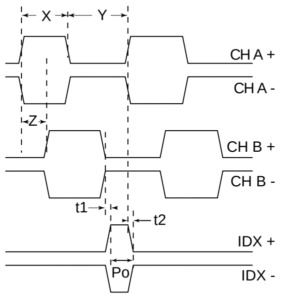
interface cable included

Timing

single-end encoder



differential encoder

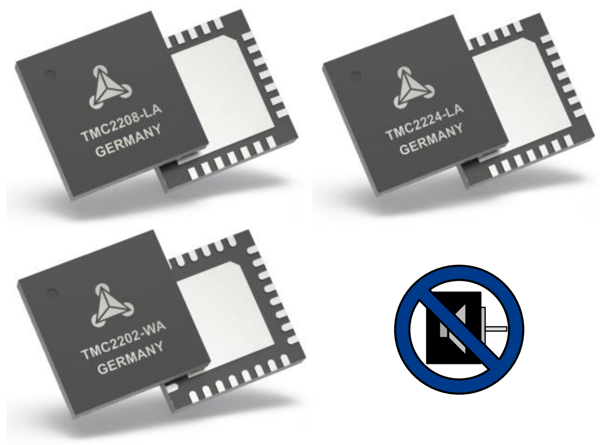


Parameter	Symbol	Min	Typ	Max	Units
Cycle error			3	5.5	ºe
Symmetry		130	180	230	ºe
Quadrature		40	90	140	ºe
Index pulse width	Po	60	90	120	ºe
Index rise (after Ch A or B rise)	t1	-300	100	250	ns
Index fall (after Ch A or B fall)	t2	70	150	1000	ns

C	One cycle: 360 electrical degrees (ºe).
X/Y	Symmetry: the measure of the relationship between X and Y, nominally 180ºe.
Z	Quadrature: the phase lead or lag between channels A and B, nominally 90ºe.
Po	Index pulse width, nominally 90 ºe.
	NOTE: Rotation is as viewed from the cover side of the encoder.

TMC2208/2 & TMC2224 family Datasheet

TMC2202, TMC2208, TMC2224 Step/Dir Drivers for Two-Phase Bipolar Stepper Motors up to 2A peak - StealthChop™ for Quiet Movement - UART Interface Option.



Applications

Compatible Design Upgrade
3D Printers
Printers, POS
Office and home automation
Textile, Sewing Machines
CCTV, Security
ATM, Cash recycler
HVAC

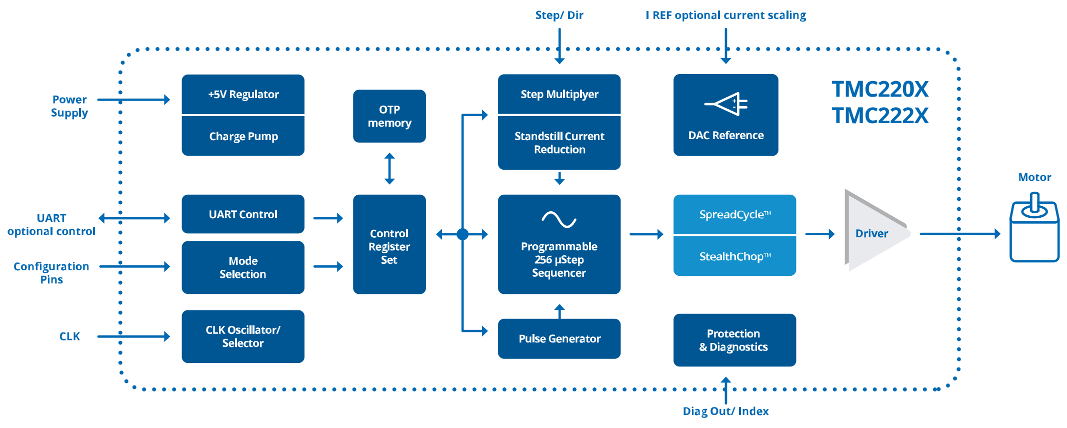
Features and Benefits

2-phase stepper motors up to 2A coil current (peak)
STEP/DIR Interface with 2, 4, 8, 16 or 32 microstep pin setting
Smooth Running 256 microsteps by **MicroPlyer™** interpolation
StealthChop2™ silent motor operation
SpreadCycle™ highly dynamic motor control chopper
Low RDSon LS 280mΩ & HS 290mΩ (typ. at 25°C)
Voltage Range 4.75… 36V DC
Automatic Standby current reduction (option)
Internal Sense Resistor option (no sense resistors required)
Passive Braking and Freewheeling
Single Wire UART & OTP for advanced configuration options
Integrated Pulse Generator for standalone motion
Full Protection & Diagnostics
Choice of QFN and wettable QFN packages for best fit

Description

The TMC2202, TMC2208 and TMC2224 are ultra-silent motor driver ICs for two-phase stepper motors. Their pinning is compatible to a number of legacy drivers. TRINAMICs sophisticated StealthChop2 chopper ensures noiseless operation, maximum efficiency and best motor torque. Its fast current regulation and optional combination with SpreadCycle allow for highly dynamic motion. Integrated power-MOSFETs handle motor current up to 1.4A RMS. Protection and diagnostic features support robust and reliable operation. A simple to use UART interface opens up more tuning and control options. Application specific tuning can be stored to OTP memory. Industries' most advanced STEP/DIR stepper motor driver family upgrades designs to noiseless and most precise operation for cost-effective and highly competitive solutions.

Block Diagram

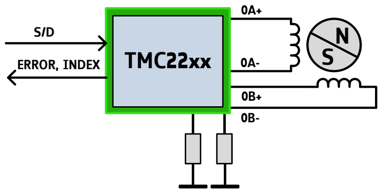


TRINAMIC Motion Control GmbH & Co. KG
Hamburg, Germany

APPLICATION EXAMPLES: SIMPLE SOLUTIONS – HIGHLY EFFECTIVE

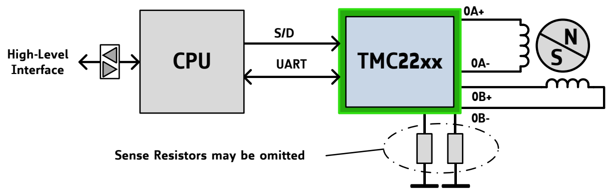
The TMC22xx family scores with power density, integrated power MOSFETs, smooth and quiet operation, and a congenial simplicity. The TMC22xx covers a wide spectrum of applications from battery systems to embedded applications with up to 2A motor current per coil. TRINAMICs unique chopper modes SpreadCycle and StealthChop2 optimize drive performance. StealthChop reduces motor noise to the point of silence at low velocities. Standby current reduction keeps costs for power dissipation and cooling down. Extensive support enables rapid design cycles and fast time-to-market with competitive products.

STANDALONE REPLACEMENT FOR LEGACY STEPPER DRIVER



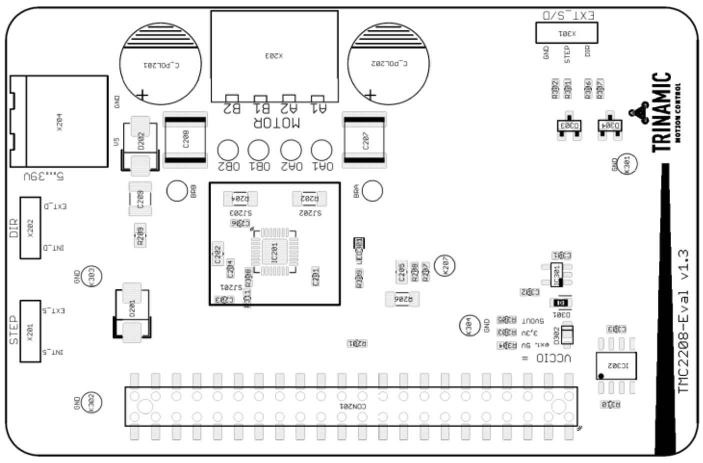
In this example, configuration is hard wired via pins. Software based motion control generates STEP and DIR (direction) signals, INDEX and ERROR signals report back status information.

UART INTERFACE FOR FULL DIAGNOSTICS AND CONTROL



A CPU operates the driver via step and direction signals. It accesses diagnostic information and configures the TMC22xx via the UART interface. The CPU manages motion control and the TMC22xx drives the motor and smoothens and optimizes drive performance.

TMC2208-EVAL EVALUATION BOARD



The TMC22xx-EVAL is part of TRINAMICs universal evaluation board system which provides a convenient handling of the hardware as well as a user-friendly software tool for evaluation. The TMC22xx evaluation board system consists of three parts: LANDUNGSBRÜCKE (base board), ESELSBRÜCKE (connector board with test points), and TMC22xx-EVAL.

ORDER CODES

Order code	PN	Description	Size [mm^2]
TMC2208-LA	00-0150	StealthChop driver; QFN28 (RoHS)	5 x 5
TMC2224-LA	00-0154	StealthChop driver; QFN28 (RoHS)	5 x 5
TMC2202-WA	00-0159	StealthChop driver; wettable edge QFN32 (RoHS)	5 x 5
TMC2208-EVAL	40-0182	Evaluation board for TMC2208 stepper motor driver	85 x 55
TMC2224-EVAL	40-0183	Evaluation board for TMC2224 stepper motor driver	85 x 55
ESELSBRÜCKE	40-0098	Connector board fitting to V1.3 and future 22xx-EVAL	61 x 38
LANDUNGSBRÜCKE	40-0167	Baseboard for TMC2208-EVAL and further evaluation boards	85 x 55

Table of Contents

1 Principles of Operation

The TMC22xx family of stepper drivers is intended as a drop-in upgrade for existing low cost stepper driver applications. Its silent drive technology StealthChop enables non-bugging motion control for home and office applications. A highly efficient power stage enables high current from a tiny package.

The TMC22xx requires just a few control pins on its tiny package. They allow selection of the most important setting: the desired microstep resolution. A choice of 2, 4, 8, 16 or 32 microsteps adapts the driver to the capabilities of the motion controller. Some package options also allow chopper mode selection by pin.

Even at low microstepping rate, the TMC22xx offers a number of unique enhancements over comparable products: TRINAMICs sophisticated StealthChop2 chopper plus the microstep enhancement MicroPlyer ensure noiseless operation, maximum efficiency and best motor torque. Its fast current regulation and optional combination with SpreadCycle allow for highly dynamic motion. Protection and diagnostic features support robust and reliable operation. A simple-to-use 8 bit UART interface opens up more tuning and control options. Application specific tuning can be stored to on-chip OTP memory. Industries' most advanced step & direction stepper motor driver family upgrades designs to noiseless and most precise operation for cost-effective and highly competitive solutions.

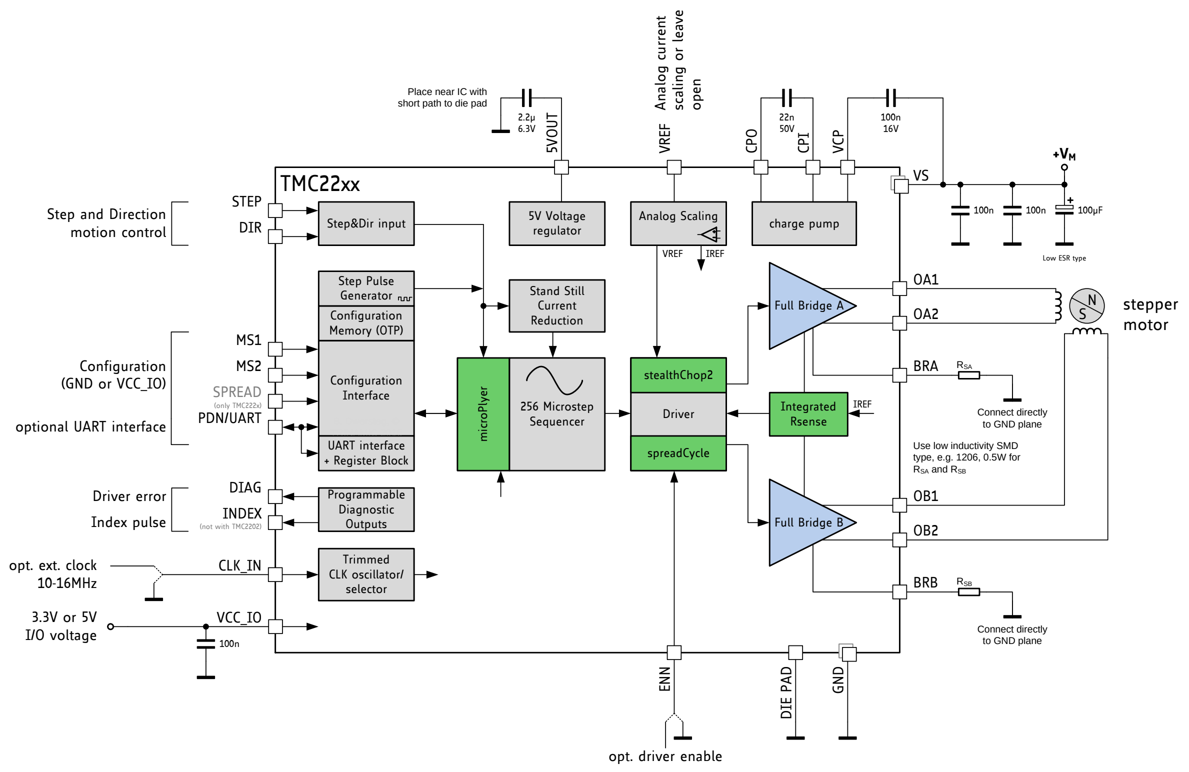


Figure 1.1 TMC22xx basic application block diagram

Three Modes of Operation:

Option 1: Standalone STEP/DIR Driver (Legacy Mode)

A CPU (µC) generates step & direction signals synchronized to additional motors and other components within the system. The TMC22xx operates the motor as commanded by the configuration pins and STEP/DIR signals. Motor run current either is fixed, or set by the CPU using the analog input VREF. The pin PDN_UART selects automatic standstill current reduction. Feedback from the driver to the CPU is granted by the INDEX and DIAG output signals. Enable or disable the motor using the ENN pin.

OPTION 2: Standalone STEP/DIR Driver with OTP pre-configuration

Additional options enabled by pre-programming OTP memory (label UART & OTP):

+ Tuning of the chopper to the application for application tailored performance
+ Cost reduction by switching the driver to internal sense resistor mode
+ Adapting the automatic power down level and timing for best application efficiency

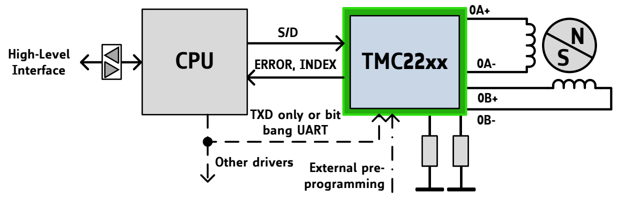


Figure 1.2 Stand-alone driver with pre-configuration

To enable the additional options, either one-time program the driver's OTP memory, or store configuration in the CPU and transfer it to the on-chip registers following each power-up. Operation uses the same signals as Option 1. Programming does not need to be done within the application - it can be executed during testing of the PCB! Alternatively, use bit-banging by CPU firmware to configure the driver. Multiple drivers can be programmed at the same time using a single TXD line.

OPTION 3: STEP/DIR Driver with Full Diagnostics and Control

Similar to Option 2, but pin PDN_UART is connected to the CPU UART interface.

Additional options (label UART):

+ Detailed diagnostics and thermal management
+ Passive braking and freewheeling for flexible, lowest power stop modes
+ More options for microstep resolution setting (fullstep to 256 microstep)
+ Software controlled motor current setting and more chopper options

This mode allows replacing all control lines like ENN, DIAG, INDEX, MS1, MS2, and analog current setting VREF by a single interface line. This way, only three signals are required for full control: STEP, DIR and PDN_UART. Even motion without external STEP pulses is provided by an internal programmable step pulse generator: Just set the desired motor velocity. However, no ramping is provided by the TMC22xx. Access to multiple driver ICs is possible using an analog multiplexer IC.

1.1 Key Concepts

The TMC22xx implements advanced features which are exclusive to TRINAMIC products. These features contribute toward greater precision, greater energy efficiency, higher reliability, smoother motion, and cooler operation in many stepper motor applications.

StealthChop2™	No-noise, high-precision chopper algorithm for inaudible motion and inaudible standstill of the motor. Allows faster motor acceleration and deceleration than StealthChop™ and extends StealthChop to low stand still motor currents.
SpreadCycle™	High-precision cycle-by-cycle current control algorithm for highest dynamic movements.
MicroPlyer™	Microstep interpolator for obtaining full 256 microstep smoothness with lower resolution step inputs starting from fullstep

In addition to these performance enhancements, TRINAMIC motor drivers offer safeguards to detect and protect against shorted outputs, output open-circuit, overtemperature, and undervoltage conditions for enhancing safety and recovery from equipment malfunctions.

1.2 Control Interfaces

The TMC22xx supports both, discrete control lines for basic mode selection and a UART based single wire interface with CRC checking. The UART interface automatically becomes enabled when correct UART data is sent. When using UART, the pin selection may be disabled by control bits.

1.2.1 UART Interface

The single wire interface allows unidirectional operation (for parameter setting only), or bi-directional operation for full control and diagnostics. It can be driven by any standard microcontroller UART or even by bit banging in software. Baud rates from 9600 Baud to 500k Baud or even higher (when using an external clock) may be used. No baud rate configuration is required, as the TMC22xx automatically adapts to the masters' baud rate. The frame format is identical to the intelligent TRINAMIC controller & driver ICs TMC5130 and TMC5072. A CRC checksum allows data transmission over longer distance. For fixed initialization sequences, store the data including CRC into the µC, thus consuming only a few 100 bytes of code for a full initialization. CRC may be ignored during read access, if not desired. This makes CRC use an optional feature! The IC has a fixed address. Multiple drivers can be programmed in parallel by tying together all interface pins, in case no read access is required. An optional addressing can be provided by analog multiplexers, like 74HC4066.

From a software point of view the TMC22xx is a peripheral with a number of control and status registers. Most of them can either be written only or are read only. Some of the registers allow both, read and write access. In case read-modify-write access is desired for a write only register, a shadow register can be realized in master software.

1.3 Moving and Controlling the Motor

1.3.1 STEP/DIR Interface

The motor is controlled by a step and direction input. Active edges on the STEP input can be rising edges or both rising and falling edges as controlled by a special mode bit (DEDGE). Using both edges cuts the toggle rate of the STEP signal in half, which is useful for communication over slow interfaces such as optically isolated interfaces. The state sampled from the DIR input upon an active STEP edge determines whether to step forward or back. Each step can be a fullstep or a microstep, in which there are 2, 4, 8, 16, 32, 64, 128, or 256 microsteps per fullstep. A step impulse with a low state on DIR increases the microstep counter and a high state decreases the counter by an amount controlled by the microstep resolution. An internal table translates the counter value into the sine and cosine values which control the motor current for microstepping.

1.3.2 Internal Step Pulse Generator

Some applications do not require a precisely co-ordinate motion – the motor just is required to move for a certain time and at a certain velocity. The TMC22xx comes with an internal pulse generator for these applications: Just provide the velocity via UART interface to move the motor. The velocity sign automatically controls the direction of the motion. However, the pulse generator does not integrate a ramping function. Motion at higher velocities will require ramping up and ramping down the velocity value via software.

STEP/DIR mode and internal pulse generator mode can be mixed in an application!

1.4 StealthChop2 & SpreadCycle Driver

StealthChop is a voltage chopper based principle. It especially guarantees that the motor is absolutely quiet in standstill and in slow motion, except for noise generated by ball bearings. Unlike other voltage mode choppers, StealthChop2 does not require any configuration. It automatically learns the best settings during the first motion after power up and further optimizes the settings in subsequent motions. An initial homing sequence is sufficient for learning. Optionally, initial learning parameters can be stored to OTP. StealthChop2 allows high motor dynamics, by reacting at once to a change of motor velocity.

For highest velocity applications, SpreadCycle is an option to StealthChop2. It can be enabled via input pin (TMC222x) or via UART and OTP. StealthChop2 and SpreadCycle may even be used in a combined configuration for the best of both worlds: StealthChop2 for no-noise stand still, silent and smooth performance, SpreadCycle at higher velocity for high dynamics and highest peak velocity at low vibration.

SpreadCycle is an advanced cycle-by-cycle chopper mode. It offers smooth operation and good resonance dampening over a wide range of speed and load. The SpreadCycle chopper scheme automatically integrates and tunes fast decay cycles to guarantee smooth zero crossing performance.

Benefits of using StealthChop2:
- Significantly improved microstepping with low cost motors
- Motor runs smooth and quiet
- Absolutely no standby noise
- Reduced mechanical resonance yields improved torque

1.5 Precise clock generator and CLK input

The TMC22xx provides a factory trimmed internal clock generator for precise chopper frequency and performance. However, an optional external clock input is available for cases, where quartz precision is desired, or where a lower or higher frequency is required. For safety, the clock input features timeout detection, and switches back to internal clock upon fail of the external source.

1.6 Automatic Standstill Power Down

An automatic current reduction drastically reduces application power dissipation and cooling requirements. Per default, the stand still current reduction is enabled by pulling PDN_UART input to GND. It reduces standstill power dissipation to less than 33% by going to slightly more than half of the run current.
Modify stand still current, delay time and decay via UART, or pre-programmed via internal OTP. Automatic freewheeling and passive motor braking are provided as an option for stand still. Passive braking reduces motor standstill power consumption to zero, while still providing effective dampening and braking!

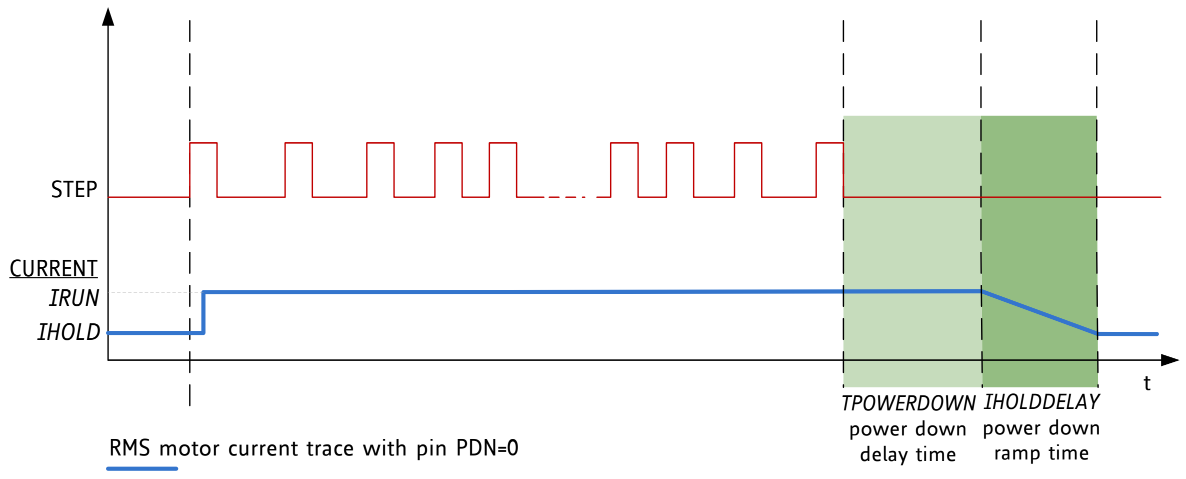


Figure 1.3 Automatic Motor Current Power Down

1.7 Index Output

The index output gives one pulse per electrical rotation, i.e. one pulse per each four fullsteps. It shows the internal sequencer microstep 0 position (*MSTEP* near 0). This is the power on position. In combination with a mechanical home switch, a more precise homing is enabled.

2 Pin Assignments

The TMC22xx family comes in a number of package variants in order to fit different footprints. Please check for availability.

2.1 Package Outline TMC2208

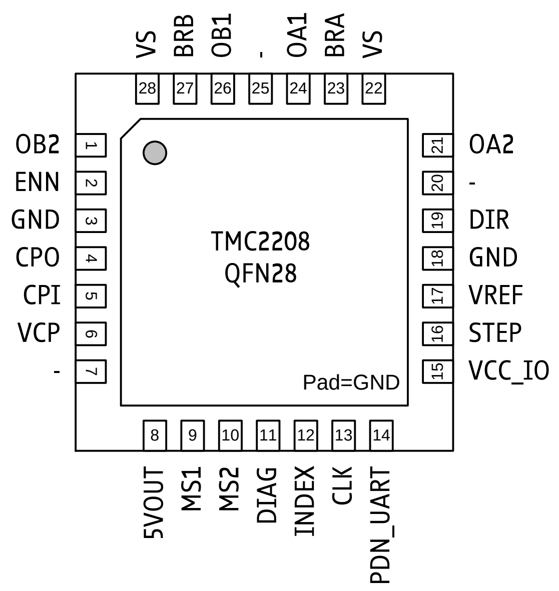


Figure 2.1 TMC2208 Pinning Top View – type: QFN28, 5x5mm², 0.5mm pitch

2.2 Signal Descriptions TMC2208

Pin	Number	Type	Function
OB2	1		Motor coil B output 2
ENN	2	DI	Enable not input. The power stage becomes switched off (all motor outputs floating) when this pin becomes driven to a high level.
GND	3, 18		GND. Connect to GND plane near pin.
CPO	4		Charge pump capacitor output.
CPI	5		Charge pump capacitor input. Tie to CPO using 22nF 50V capacitor.
VCP	6		Charge pump voltage. Tie to VS using 100nF capacitor.
N.C.	7, 20, 25		Unused pin, leave open or connect to GND for compatibility to future versions.
5VOUT	8		Output of internal 5V regulator. Attach 2.2µF to 4.7µF ceramic capacitor to GND near to pin for best performance. Provide the shortest possible loop to the GND pad.
MS1	9	DI (pd)	Microstep resolution configuration (internal pull-down resistors) MS2, MS1: 00: 1/8, 01: 1/2, 10: 1/4 11: 1/16
MS2	10	DI (pd)	
DIAG	11	DO	Diagnostic output. Hi level upon driver error. Reset by ENN=high.
INDEX	12	DO	Configurable index output. Provides index pulse.
CLK	13	DI	CLK input. Tie to GND using short wire for internal clock or supply external clock.
PDN_UART	14	DIO	Power down not control input (low = automatic standstill current reduction). Optional UART Input/Output. Power down function can be disabled in UART mode.
VCC_IO	15		3.3V to 5V IO supply voltage for all digital pins.

Pin	Number	Type	Function
STEP	16	DI	STEP input
VREF	17	AI	Analog reference voltage for current scaling or reference current for use of internal sense resistors (optional mode)
DIR	19	DI (pd)	DIR input (internal pull-down resistor)
VS	22, 28		Motor supply voltage. Provide filtering capacity near pin with shortest possible loop to GND pad.
OA2	21		Motor coil A output 2
BRA	23		Sense resistor connection for coil A. Place sense resistor to GND near pin. Tie to GND when using internal sense resistor.
OA1	24		Motor coil A output 1
OB1	26		Motor coil B output 1
BRB	27		Sense resistor connection for coil B. Place sense resistor to GND near pin. Tie to GND when using internal sense resistor.
Exposed die pad	-		Connect the exposed die pad to a GND plane. Provide as many as possible vias for heat transfer to GND plane. Serves as GND pin for power drivers and analogue circuitry.

2.3 Package Outline TMC2202

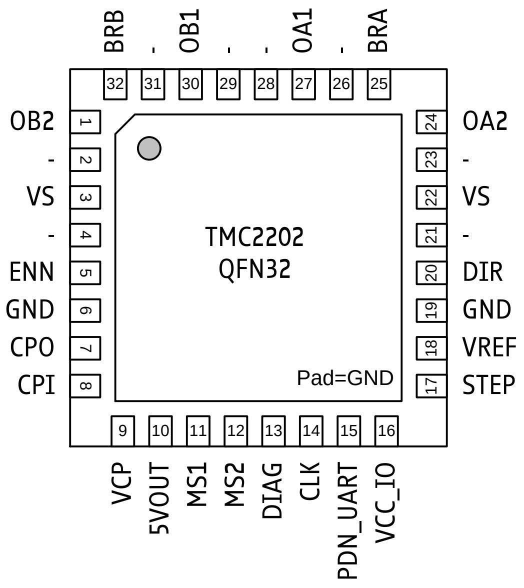


Figure 2.2 TMC2202 Pinning Top View – type: QFN32, 5x5mm², 0.5mm pitch

2.4 Signal Descriptions TMC2202

Pin	Number	Type	Function
OB2	1		Motor coil B output 2
N.C.	2, 4, 21, 23, 26, 28, 29, 31		Unused pin, leave open to provide for higher creeping voltage distances.
VS	3, 22		Motor supply voltage. Provide filtering capacity near pin with shortest possible loop to GND pad.
ENN	5	DI	Enable not input. The power stage becomes switched off (all motor outputs floating) when this pin becomes driven to a high level.

Pin	Number	Type	Function
GND	6, 19		GND. Connect to GND plane near pin.
CPO	7		Charge pump capacitor output.
CPI	8		Charge pump capacitor input. Tie to CPO using 22nF 50V capacitor.
VCP	9		Charge pump voltage. Tie to VS using 100nF capacitor.
5VOUT	10		Output of internal 5V regulator. Attach 2.2µF to 4.7µF ceramic capacitor to GND near to pin for best performance. Provide the shortest possible loop to the GND pad.
MS1	11	DI (pd)	Microstep resolution configuration (internal pull-down resistors) MS2, MS1: 00: 1/8, 01: 1/2, 10: 1/4 11: 1/16
MS2	12	DI (pd)	
DIAG	13	DO	Diagnostic output. Hi level upon driver error. Reset by ENN=high.
CLK	14	DI	CLK input. Tie to GND using short wire for internal clock or supply external clock.
PDN_UART	15	DIO	Power down not control input (low = automatic standstill current reduction). Optional UART Input/Output. Power down function can be disabled in UART mode.
VCC_IO	16		3.3V to 5V IO supply voltage for all digital pins.
STEP	17	DI	STEP input
VREF	18	AI	Analog reference voltage for current scaling or reference current for use of internal sense resistors (optional mode)
DIR	20	DI (pd)	DIR input (internal pull-down resistor)
OA2	24		Motor coil A output 2
BRA	25		Sense resistor connection for coil A. Place sense resistor to GND near pin. Tie to GND when using internal sense resistor.
OA1	27		Motor coil A output 1
OB1	30		Motor coil B output 1
BRB	32		Sense resistor connection for coil B. Place sense resistor to GND near pin. Tie to GND when using internal sense resistor.
Exposed die pad	-		Connect the exposed die pad to a GND plane. Provide as many as possible vias for heat transfer to GND plane. Serves as GND pin for power drivers and analogue circuitry.

2.5 Package Outline TMC2224

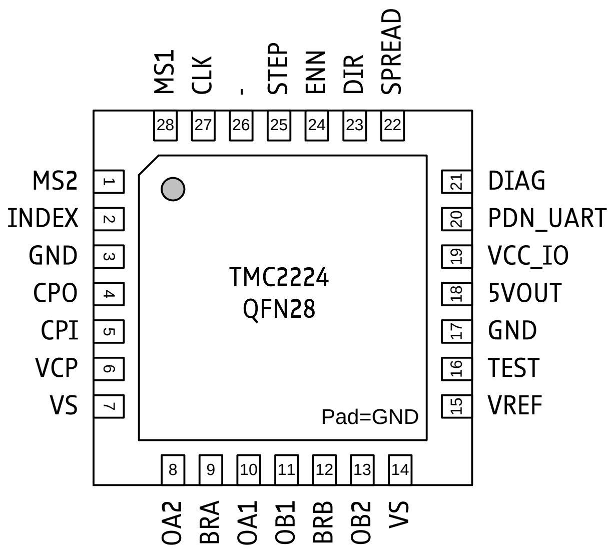


Figure 2.3 TMC2224 Pinning Top View – type: QFN28, 5x5mm², 0.5mm pitch

2.6 Signal Descriptions TMC2224

Pin	Number	Type	Function
MS1	28	DI (pd)	Microstep resolution configuration (internal pull-down resistors)
MS2	1	DI (pd)	MS2, MS1: 00: 1/4, 01: 1/8, 10: 1/16, 11: 1/32
INDEX	2	DO	Configurable index output. Provides index pulse.
GND	3, 17		GND. Connect to GND plane near pin.
CPO	4		Charge pump capacitor output.
CPI	5		Charge pump capacitor input. Tie to CPO using 22nF 50V capacitor.
VCP	6		Charge pump voltage. Tie to VS using 100nF capacitor.
VS	7, 14		Motor supply voltage. Provide filtering capacity near pin with shortest possible loop to GND pad.
OA2	8		Motor coil A output 2
BRA	9		Sense resistor connection for coil A. Place sense resistor to GND near pin. Tie to GND when using internal sense resistor.
OA1	10		Motor coil A output 1
OB1	11		Motor coil B output 1
BRB	12		Sense resistor connection for coil B. Place sense resistor to GND near pin. Tie to GND when using internal sense resistor.
OB2	13		Motor coil B output 2
VREF	15	AI	Analog reference voltage for current scaling or reference current for use of internal sense resistors (optional mode)
TEST	16		Connect to GND. May alternatively be left open or connected to VREF.
5VOUT	18		Output of internal 5V regulator. Attach 2.2µF to 4.7µF ceramic capacitor to GND near to pin for best performance. Provide the shortest possible loop to the GND pad.
VCC_IO	19		3.3V to 5V IO supply voltage for all digital pins.
PDN_UART	20	DIO (pd)	Power down not control input (low = automatic standstill current reduction). (internal pull-down resistor) Optional UART Input/Output. Power down function can be disabled in UART mode.
DIAG	21	DO	Diagnostic output. Hi level upon driver error. Reset by ENN=high.
SPREAD	22	DI (pd)	Chopper mode selection: Low=StealthChop, High=SpreadCycle
DIR	23	DI (pd)	DIR input (internal pull-down resistor)
ENN	24	DI	Enable not input. The power stage becomes switched off (all motor outputs floating) when this pin becomes driven to a high level.
STEP	25	DI (pd)	STEP input (internal pull-down resistor)
N.C.	26		Unused pin, leave open or connect to GND for compatibility to future versions.
CLK	27	DI	CLK input. Tie to GND using short wire for internal clock or supply external clock.
Exposed die pad	-		Connect the exposed die pad to a GND plane. Provide as many as possible vias for heat transfer to GND plane. Serves as GND pin for power drivers and analogue circuitry.

3 Sample Circuits

The sample circuits show the connection of external components in different operation and supply modes. The connection of the bus interface and further digital signals is left out for clarity.

3.1 Standard Application Circuit

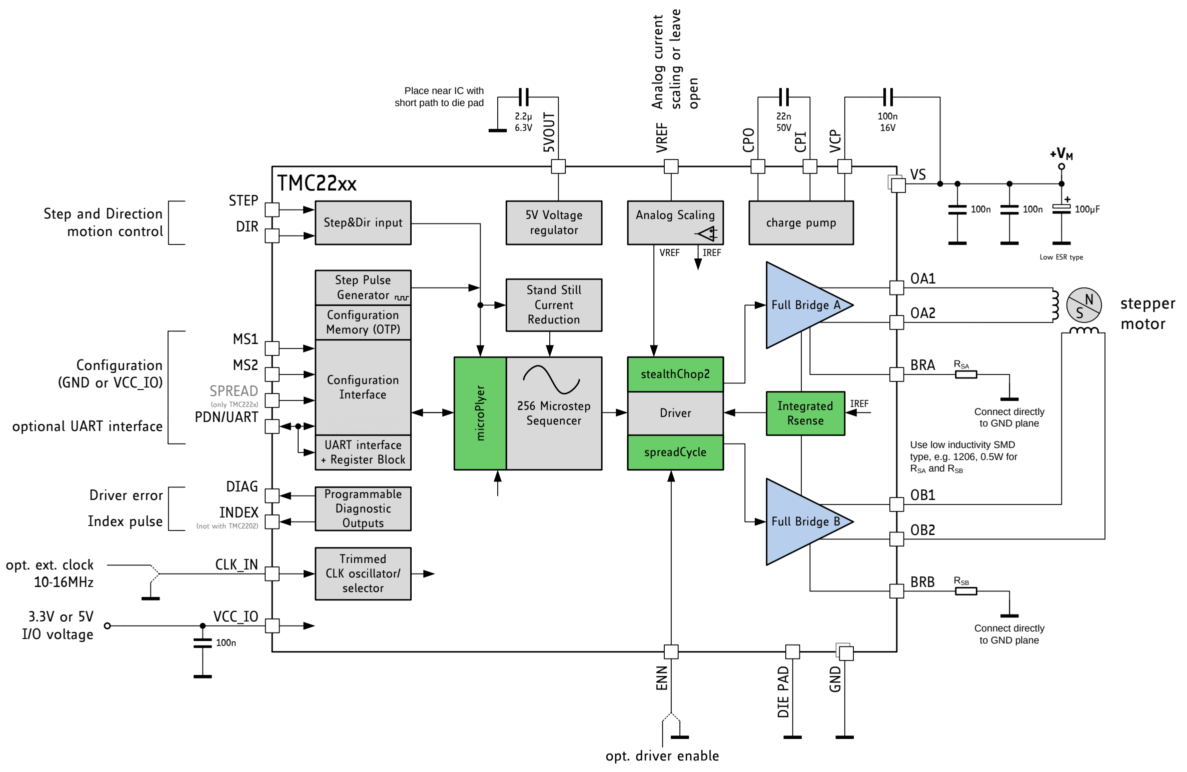


Figure 3.1 Standard application circuit

The standard application circuit uses a minimum set of additional components. Two sense resistors set the motor coil current. See chapter 8 to choose the right sense resistors. Use low ESR capacitors for filtering the power supply. The capacitors need to cope with the current ripple cause by chopper operation. A minimum capacity of 100µF near the driver is recommended for best performance. Current ripple in the supply capacitors also depends on the power supply internal resistance and cable length. VCC_IO can be supplied from 5VOUT, or from an external source, e.g. a 3.3V regulator.

Basic layout hints
Place sense resistors and all filter capacitors as close as possible to the related IC pins. Use a solid common GND for all GND connections, also for sense resistor GND. Connect 5VOUT filtering capacitor directly to 5VOUT and the die pad. See layout hints for more details. Low ESR electrolytic capacitors are recommended for VS filtering.

3.2 Internal RDSon Sensing

For cost critical or space limited applications, sense resistors can be omitted. For internal current sensing, a reference current set by a tiny external resistor programs the output current. For calculation of the reference resistor, refer chapter 9.1.

Attention
Be sure to switch the IC to RDSon mode, before enabling drivers: Set *otp_internalRsense* = 1.

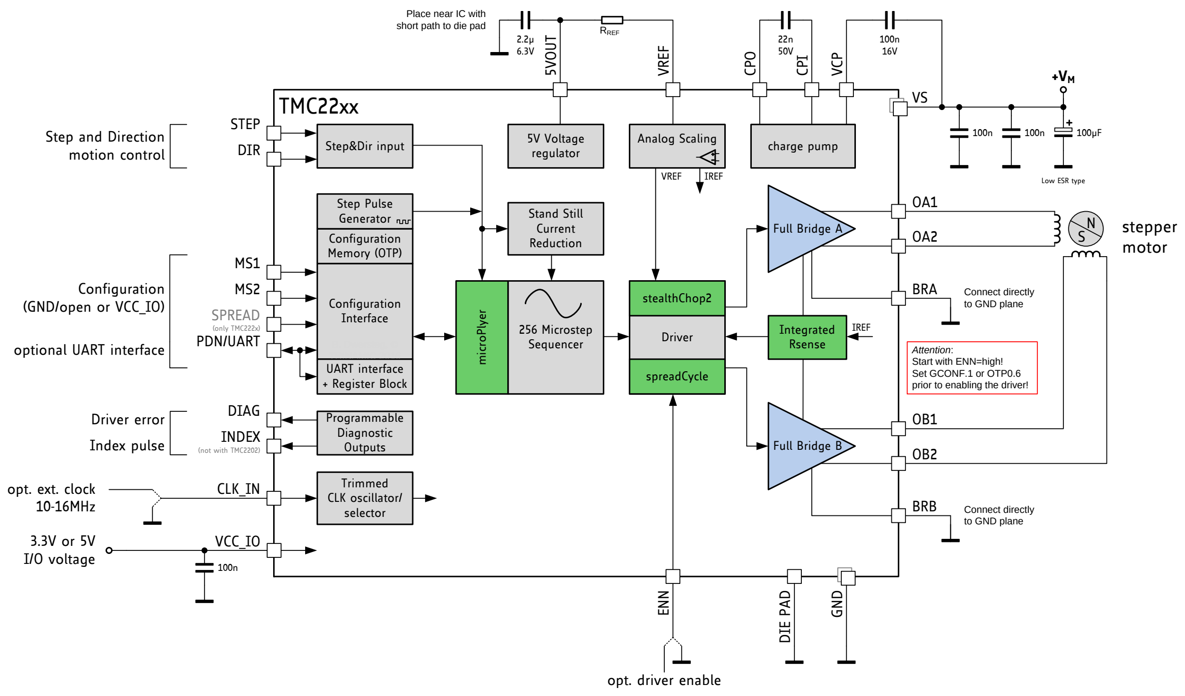


Figure 3.2 Application circuit using RDSon based sensing

3.3 5V Only Supply

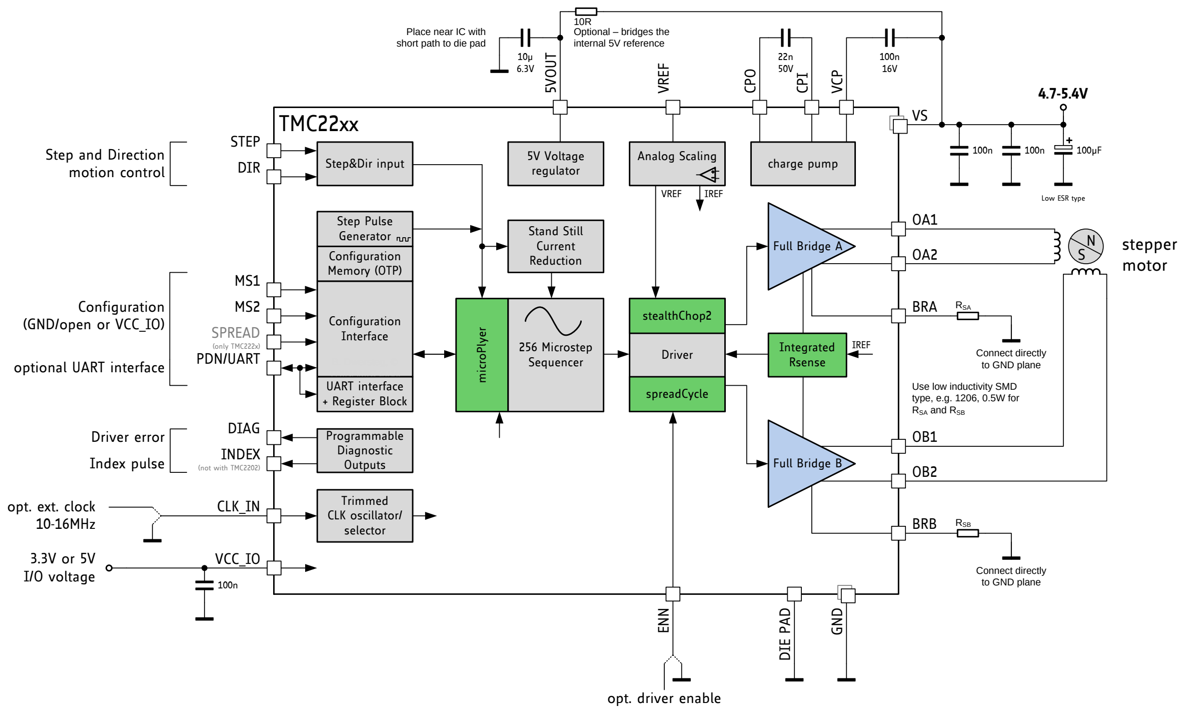


Figure 3.3 5V only operation

While the standard application circuit is limited to roughly 5.2 V lower supply voltage, a 5 V only application lets the IC run from a 5 V +/-5% supply. In this application, linear regulator drop must be minimized. Therefore, the internal 5V regulator is filtered with a higher capacitance. An optional resistor bridges the internal 5V regulator by connecting 5VOUT to the external power supply. This RC filter keeps chopper ripple away from 5VOUT. With this resistor, the external supply is the reference for the absolute motor current and must not exceed 5.5V.

3.4 Configuration Pins

The TMC22xx family members provide three or four configuration pins depending on the package option. These pins allow quick configuration for standalone operation. Several additional options can be set by OTP programming. In UART mode, the configuration pins can be disabled in order to set a different configuration via registers.

PDN_UART: Configuration of Standstill Power Down

PDN_UART	Current Setting
GND	Enable automatic power down in standstill periods
VCC_IO	Disable
UART interface	When using the UART interface, the configuration pin should be disabled via *GCONF.pdn_disable* = 1. Program *IHOLD* as desired for standstill periods.

Options for TMC220x devices, only:

MS1/MS2: Configuration of Microstep Resolution for Step Input (TMC220x)

MS2	MS1	Microstep Setting
GND	GND	8 microsteps
GND	VCC_IO	2 microsteps (half step)
VCC_IO	GND	4 microsteps (quarter step)
VCC_IO	VCC_IO	16 microsteps

Options for TMC222x devices, only:

SPREAD (only with TMC222x): Selection of Chopper Mode

SPREAD	Chopper Setting
GND or Pin open / not available	StealthChop is selected. Automatic switching to SpreadCycle in dependence of the step frequency can be programmed via OTP.
VCC_IO	SpreadCycle operation.

MS1/MS2: Configuration of Microstep Resolution for Step Input (TMC222x)

MS2	MS1	Microstep Setting
GND	GND	4 microsteps (quarter step)
GND	VCC_IO	8 microsteps
VCC_IO	GND	16 microsteps
VCC_IO	VCC_IO	32 microsteps

3.5 High Motor Current

When operating at a high motor current, the driver power dissipation due to MOSFET switch on-resistance significantly heats up the driver. This power dissipation will significantly heat up the PCB cooling infrastructure, if operated at an increased duty cycle. This in turn leads to a further increase of driver temperature. An increase of temperature by about 100°C increases MOSFET resistance by roughly 50%. This is a typical behavior of MOSFET switches. Therefore, under high duty cycle, high load conditions, thermal characteristics have to be carefully taken into account, especially when increased environment temperatures are to be supported. Refer the thermal characteristics and the layout hints for more information. As a thumb rule, thermal properties of the PCB design become critical for the tiny QFN 5mm x 5mm package at or above 1A RMS motor current for increased periods of time. Keep in mind that resistive power dissipation raises with the square of the motor current. On the other hand, this means that a small reduction of motor current significantly saves heat dissipation and energy.

Pay special attention to good thermal properties of your PCB layout, when going for 1A RMS current or more.

An effect which might be perceived at medium motor velocities and motor sine wave peak currents above roughly 1.4A peak is a slight sine distortion of the current wave when using SpreadCycle. It results from an increasing negative impact of parasitic internal diode conduction, which in turn negatively influences the duration of the fast decay cycle of the SpreadCycle chopper. This is, because the current measurement does not see the full coil current during this phase of the sine wave, because an increasing part of the current flows directly from the power MOSFETs' drain to GND and does not flow through the sense resistor. This effect with most motors does not negatively influence the smoothness of operation, as it does not impact the critical current zero transition. The effect does not occur with StealthChop.

3.6 Driver Protection and EME Circuitry

Some applications have to cope with ESD events caused by motor operation or external influence. Despite ESD circuitry within the driver chips, ESD events occurring during operation can cause a reset or even a destruction of the motor driver, depending on their energy. Especially plastic housings and belt drive systems tend to cause ESD events of several kV. It is best practice to avoid ESD events by attaching all conductive parts, especially the motors themselves to PCB ground, or to apply electrically conductive plastic parts. In addition, the driver can be protected up to a certain degree against ESD events or live plugging / pulling the motor, which also causes high voltages and high currents into the motor connector terminals. A simple scheme uses capacitors at the driver outputs to reduce the dV/dt caused by ESD events. Larger capacitors will bring more benefit concerning ESD suppression, but cause additional current flow in each chopper cycle, and thus increase driver power dissipation, especially at high supply voltages. The values shown are example values – they may be varied between 100pF and 1nF. The capacitors also dampen high frequency noise injected from digital parts of the application PCB circuitry and thus reduce electromagnetic emission. A more elaborate scheme uses LC filters to de-couple the driver outputs from the motor connector. Varistors in between of the coil terminals eliminate coil overvoltage caused by live plugging. Optionally protect all outputs by a varistor to GND against ESD voltage.

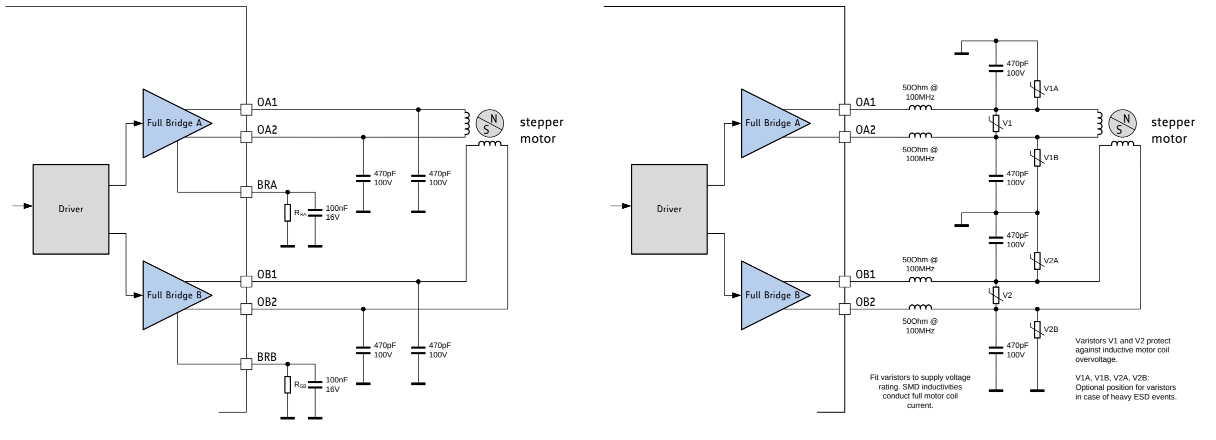

Driver
Full Bridge A
Full Bridge B
OA1
OA2
BRA
OB1
OB2
BRB
R_{SA}
R_{SB}
100nF 16V
470pF 100V
stepper motor
N
S
50Ohm @ 100MHz
V1A
V1
V1B
V2A
V2
V2B
Fit varistors to supply voltage rating. SMD inductivities conduct full motor coil current.
Varistors V1 and V2 protect against inductive motor coil overvoltage.
V1A, V1B, V2A, V2B: Optional position for varistors in case of heavy ESD events.

Figure 3.4 Simple ESD enhancement and more elaborate motor output protection

4 UART Single Wire Interface

The UART single wire interface allows control of the TMC22xx with any microcontroller UART. It shares transmit and receive line like an RS485 based interface. Data transmission is secured using a cyclic redundancy check, so that increased interface distances (e.g. over cables between two PCBs) can be bridged without danger of wrong or missed commands even in the event of electro-magnetic disturbance. The automatic baud rate detection makes this interface easy to use.

4.1 Datagram Structure

4.1.1 Write Access

UART WRITE ACCESS DATAGRAM STRUCTURE																			
each byte is LSB…MSB, highest byte transmitted first																			
0 … 63																			
sync + reserved								**8 bit slave address**			**RW + 7 bit register addr.**			**32 bit data**			**CRC**		
0…7								8…15			16…23			24…55			56…63		
1	0	1	0	Reserved (don't cares but included in CRC)				*SLAVEADDR*=0			register address		1	data bytes 3, 2, 1, 0 (high to low byte)			CRC		
0	1	2	3	4	5	6	7	8	…	15	16	…	23	24	…	55	56	…	63

A sync nibble precedes each transmission to and from the TMC22xx and is embedded into the first transmitted byte, followed by an addressing byte (0 for TMC22xx). Each transmission allows a synchronization of the internal baud rate divider to the master clock. The actual baud rate is adapted and variations of the internal clock frequency are compensated. Thus, the baud rate can be freely chosen within the valid range. Each transmitted byte starts with a start bit (logic 0, low level on SWIOP) and ends with a stop bit (logic 1, high level on SWIOP). The bit time is calculated by measuring the time from the beginning of start bit (1 to 0 transition) to the end of the sync frame (1 to 0 transition from bit 2 to bit 3). All data is transmitted bytewise. The 32 bit data words are transmitted with the highest byte first.

A minimum baud rate of 9000 baud is permissible, assuming 20 MHz clock (worst case for low baud rate). Maximum baud rate is $f_{CLK}/16$ due to the required stability of the baud clock.

The slave address *SLAVEADDR* is always 0 for the TMC22xx.

The communication becomes reset if a pause time of longer than 63 bit times between the start bits of two successive bytes occurs. This timing is based on the last correctly received datagram. In this case, the transmission needs to be restarted after a failure recovery time of minimum 12 bit times of bus idle time. This scheme allows the master to reset communication in case of transmission errors. Any pulse on an idle data line below 16 clock cycles will be treated as a glitch and leads to a timeout of 12 bit times, for which the data line must be idle. Other errors like wrong CRC are also treated the same way. This allows a safe re-synchronization of the transmission after any error conditions. Remark, that due to this mechanism an abrupt reduction of the baud rate to less than 15 percent of the previous value is not possible.

Each accepted write datagram becomes acknowledged by the receiver by incrementing an internal cyclic datagram counter (8 bit). Reading out the datagram counter allows the master to check the success of an initialization sequence or single write accesses. Read accesses do not modify the counter.

The UART line must be logic high during idle state. Therefore, the power down function cannot be assigned by the pin PDN_UART in between of transmissions. In an application using the UART interface, set the desired power down function by register access and set *pdn_disable* in GCONF to disable the pin function.

4.1.2 Read Access

UART READ ACCESS REQUEST DATAGRAM STRUCTURE																
each byte is LSB…MSB, highest byte transmitted first																
sync + reserved								**8 bit slave address**			**RW + 7 bit register address**			**CRC**		
0…7								8…15			16…23			24…31		
1	0	1	0	Reserved (don't cares but included in CRC)				*SLAVEADDR*=0			register address		0	CRC		
0	1	2	3	4	5	6	7	8	…	15	16	…	23	24	…	31

The read access request datagram structure is identical to the write access datagram structure, but uses a lower number of user bits. Its function is the addressing of the slave and the transmission of the desired register address for the read access. The TMC22xx responds with the same baud rate as the master uses for the read request.

In order to ensure a clean bus transition from the master to the slave, the TMC22xx does not immediately send the reply to a read access, but it uses a programmable delay time after which the first reply byte becomes sent following a read request. This delay time can be set in multiples of eight bit times using *SENDDELAY* time setting (default=8 bit times) according to the needs of the master.

UART READ ACCESS REPLY DATAGRAM STRUCTURE																			
each byte is LSB…MSB, highest byte transmitted first																			
0 …… 63																			
sync + reserved								**8 bit master address**			**RW + 7 bit register addr.**			**32 bit data**			**CRC**		
0…7								8…15			16…23			24…55			56…63		
1	0	1	0	reserved (0)				0xFF			register address		0	data bytes 3, 2, 1, 0 (high to low byte)			CRC		
0	1	2	3	4	5	6	7	8	…	15	16	…	23	24	…	55	56	…	63

The read response is sent to the master using address code %11111111. The transmitter becomes switched inactive four bit times after the last bit is sent.

Address %11111111 is reserved for read access replies going to the master.

4.2 CRC Calculation

An 8 bit CRC polynomial is used for checking both read and write access. It allows detection of up to eight single bit errors. The CRC8-ATM polynomial with an initial value of zero is applied LSB to MSB, including the sync- and addressing byte. The sync nibble is assumed to always be correct. The TMC22xx responds only to correctly transmitted datagrams containing its own slave address. It increases its datagram counter for each correctly received write access datagram.

$$CRC = x^8 + x^2 + x^1 + x^0$$

Serial calculation example

```
CRC = (CRC << 1) OR (CRC.7 XOR CRC.1 XOR CRC.0 XOR [new incoming bit])
```

C-Code Example for CRC calculation

```
void swuart_calcCRC(UCHAR* datagram, UCHAR datagramLength)
{
  int i,j;
  UCHAR* crc = datagram + (datagramLength-1); // CRC located in last byte of message
  UCHAR currentByte;

  *crc = 0;
  for (i=0; i<(datagramLength-1); i++) {      // Execute for all bytes of a message
    currentByte = datagram[i];                // Retrieve a byte to be sent from Array
    for (j=0; j<8; j++) {
      if ((*crc >> 7) ^ (currentByte&0x01))   // update CRC based result of XOR operation
      {
        *crc = (*crc << 1) ^ 0x07;
      }
      else
      {
        *crc = (*crc << 1);
      }
      currentByte = currentByte >> 1;
    } // for CRC bit
  } // for message byte
}
```

4.3 UART Signals

The UART interface on the TMC22xx uses a single bi-direction pin:

UART Interface Signal	
PDN_UART	Non-inverted data input and output. I/O with Schmitt Trigger and VCC_IO level.

The IC checks PDN_UART for correctly received datagrams with its own address continuously. It adapts to the baud rate based on the sync nibble, as described before. In case of a read access, it switches on its output drivers and sends its response using the same baud rate. The output becomes switched off four bit times after transfer of the last stop bit.

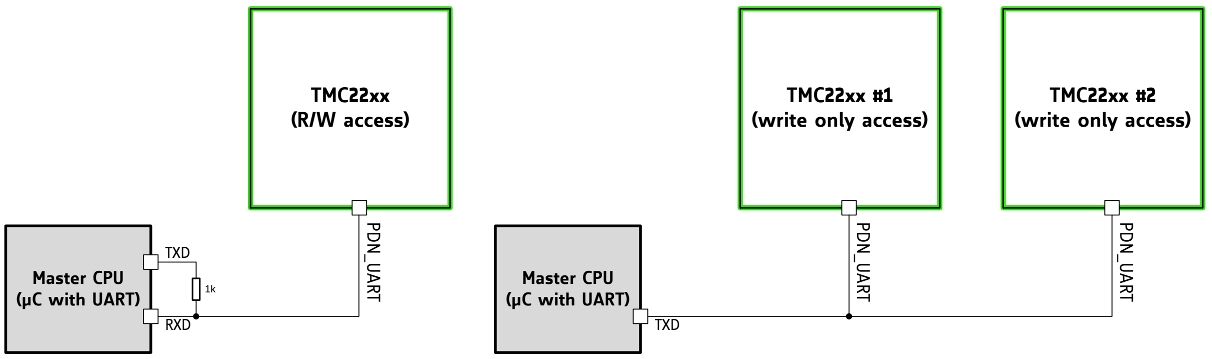


Figure 4.1 Attaching the TMC22xx to a microcontroller UART

4.4 Addressing Multiple Slaves

Write Only Access

If read access is not used, and all slaves are to be programmed with the same initialization values, no addressing is required. All slaves can be programmed in parallel like a single device (Figure 4.1.).

Addressing Multiple Slaves

As the TMC22xx uses a fixed UART address, in principle only one IC can be accessed per UART interface channel. Adding analog switches allows separated access to individual ICs. This scheme is similar to an SPI bus with individual slave select lines (Figure 4.2).

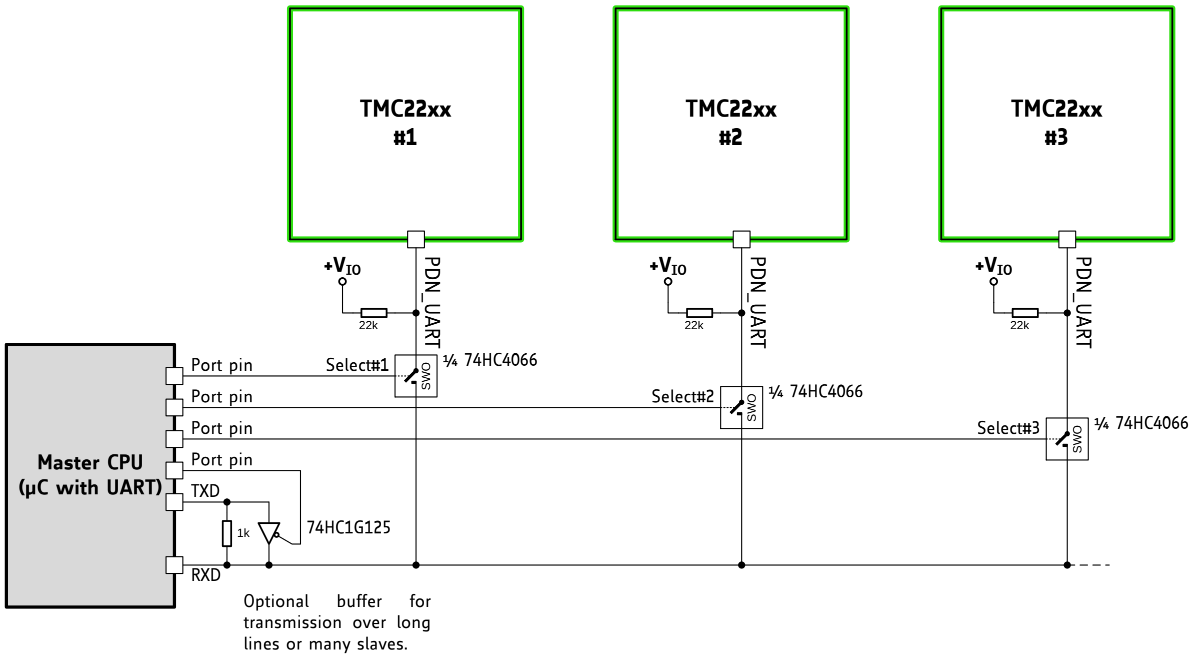


Figure 4.2 Addressing multiple TMC22xx via single wire interface using analog switches

Proceed as follows to control multiple slaves:

- Set the UART to 8 bits, no parity. Select a baud rate safely within the valid range. At 250kBaud, a write access transmission requires 320µs (=8 Bytes * (8+2) bits * 4µs).
- Before starting an access, activate the select pin going to the analog switch by setting it high. All other slaves select lines shall be off, unless a broadcast is desired.
- When using the optional buffer, set TMC22xx transmission send delay to an appropriate value allowing the µC to switch off the buffer before receiving reply data.
- To start a transmission, activate the TXD line buffer by setting the control pin low.
- When sending a read access request, switch off the buffer after transmission of the last stop bit is finished.
- Take into account, that all transmitted data also is received by the RXD input.

5 Register Map

This chapter gives an overview of the complete register set. Some of the registers bundling a number of single bits are detailed in extra tables. The functional practical application of the settings is detailed in dedicated chapters.

Note
- *Reset default*: All registers become reset to 0 upon power up, unless otherwise noted.
- Add 0x80 to the address **Addr** for write accesses!

NOTATION OF HEXADECIMAL AND BINARY NUMBERS	
0x	precedes a hexadecimal number, e.g. 0x04
%	precedes a multi-bit binary number, e.g. %100

NOTATION OF R/W FIELD	
R	Read only
W	Write only
R/W	Read- and writable register
R+C	Clear upon read

OVERVIEW REGISTER MAPPING

REGISTER	**DESCRIPTION**
General Configuration Registers	These registers contain - global configuration - global status flags - OTP read access and programming - interface configuration
Velocity Dependent Driver Feature Control Register Set	This register set offers registers for - driver current control, stand still reduction - setting thresholds for different chopper modes - internal pulse generator control
Chopper Register Set	This register set offers registers for - optimization of StealthChop2 and SpreadCycle and read out of internal values - passive braking and freewheeling options - driver diagnostics - driver enable / disable

5.1 General Registers

GENERAL CONFIGURATION REGISTERS (0X00…0X0F)					
R/W	**Addr**	**n**	***Register***	**Description / *bit names***	
				Bit	***GCONF* – Global configuration flags**
RW	0x00	10	*GCONF*	0	*I_scale_analog* (*Reset default=1*) 0: Use internal reference derived from 5VOUT 1: Use voltage supplied to VREF as current reference
				1	*internal_Rsense* (*Reset default: OTP*) 0: Operation with external sense resistors 1: Internal sense resistors. Use current supplied into VREF as reference for internal sense resistor. VREF pin internally is driven to GND in this mode.
				2	*en_SpreadCycle* (*Reset default: OTP*) 0: StealthChop PWM mode enabled (depending on velocity thresholds). Initially switch from off to on state while in stand still, only. 1: SpreadCycle mode enabled A high level on the pin SPREAD (TMC222x, only) inverts this flag to switch between both chopper modes.
				3	*shaft* 1: Inverse motor direction
				4	*index_otpw* 0: INDEX shows the first microstep position of sequencer 1: INDEX pin outputs overtemperature prewarning flag (*otpw*) instead
				5	*index_step* 0: INDEX output as selected by *index_otpw* 1: INDEX output shows step pulses from internal pulse generator (toggle upon each step)
				6	*pdn_disable* 0: PDN_UART controls standstill current reduction 1: PDN_UART input function disabled. Set this bit, when using the UART interface!
				7	*mstep_reg_select* 0: Microstep resolution selected by pins MS1, MS2 1: Microstep resolution selected by MSTEP register
				8	*multistep_filt* (*Reset default=1*) 0: No filtering of STEP pulses 1: Software pulse generator optimization enabled when fullstep frequency > 750Hz (roughly). TSTEP shows filtered step time values when active.
				9	*test_mode* 0: Normal operation 1: Enable analog test output on pin ENN (pull-down resistor off), ENN treated as enabled. *IHOLD*[1..0] selects the function of DCO: 0…2: T120, DAC, VDDH *Attention: Not for user, set to 0 for normal operation!*

GENERAL CONFIGURATION REGISTERS (0X00…0X0F)					
R/W	**Addr**	**n**	***Register***	**Description / bit names**	
R+ WC	0x01	3	*GSTAT*	**Bit**	***GSTAT* – Global status flags** (Re-Write with '1' bit to clear respective flags)
				0	*reset* 1: Indicates that the IC has been reset since the last read access to *GSTAT*. All registers have been cleared to reset values.
				1	*drv_err* 1: Indicates, that the driver has been shut down due to overtemperature or short circuit detection since the last read access. Read DRV_STATUS for details. The flag can only be cleared when all error conditions are cleared.
				2	*uv_cp* 1: Indicates an undervoltage on the charge pump. The driver is disabled in this case. This flag is not latched and thus does not need to be cleared.
R	0x02	8	*IFCNT*		Interface transmission counter. This register becomes incremented with each successful UART interface write access. Read out to check the serial transmission for lost data. Read accesses do not change the content. The counter wraps around from 255 to 0.
W	0x03	4	*SLAVECONF*	**Bit**	***SLAVECONF***
				11..8	SENDDELAY for read access (time until reply is sent): 0, 1: 8 bit times 2, 3: 3*8 bit times 4, 5: 5*8 bit times 6, 7: 7*8 bit times 8, 9: 9*8 bit times 10, 11: 11*8 bit times 12, 13: 13*8 bit times 14, 15: 15*8 bit times
W	0x04	16	*OTP_PROG*	**Bit**	***OTP_PROGRAM* – OTP programming** Write access programs OTP memory (one bit at a time), Read access refreshes read data from OTP after a write
				2..0	*OTPBIT* Selection of OTP bit to be programmed to the selected byte location (n=0..7: programs bit n to a logic 1)
				5..4	*OTPBYTE* Selection of OTP programming location (0, 1 or 2)
				15..8	*OTPMAGIC* Set to *0xbd* to enable programming. A programming time of minimum 10ms per bit is recommended (check by reading *OTP_READ*).
R	0x05	24	*OTP_READ*	**Bit**	***OTP_READ*** (Access to OTP memory result and update) *See separate table!*
				7..0	*OTP0* byte 0 read data
				15..8	*OTP1* byte 1 read data
				23..16	*OTP2* byte 2 read data
R	0x06	10 + 8	*IOIN*	**Bit**	***INPUT*** (Reads the state of all input pins available)
				0	ENN (TMC220x)
				1	PDN_UART (TMC222x)
				2	MS1 (TMC220x), SPREAD (TMC222x)
				3	MS2 (TMC220x), DIR (TMC222x)

GENERAL CONFIGURATION REGISTERS (0X00…0X0F)					
R/W	**Addr**	**n**	***Register***	**Description / *bit names***	
				4	DIAG (TMC220x), ENN (TMC222x)
				5	STEP (TMC222x)
				6	PDN_UART (TMC220x), MS1 (TMC222x)
				7	STEP (TMC220x), MS2 (TMC222x)
				8	SEL_A: Driver type 1: TMC220x 0: TMC222x
				9	DIR (TMC220x)
				31.. 24	*VERSION*: 0x20=first version of the IC Identical numbers mean full digital compatibility.
RW	0x07	5+2	*FACTORY_ CONF*	4..0	*FCLKTRIM* (*Reset default: OTP*) 0…31: Lowest to highest clock frequency. Check at charge pump output. The frequency span is not guaranteed, but it is tested, that tuning to 12MHz internal clock is possible. The devices come preset to 12MHz clock frequency by OTP programming.
				9..8	*OTTRIM* (Default: OTP) %00: OT=143°C, OTPW=120°C %01: OT=150°C, OTPW=120°C %10: OT=150°C, OTPW=143°C %11: OT=157°C, OTPW=143°C

5.1.1 *OTP_READ* – OTP configuration memory

The OTP memory holds power up defaults for certain registers. All OTP memory bits are cleared to 0 by default. Programming only can set bits, clearing bits is not possible. Factory tuning of the clock frequency affects *otp0.0* to *otp0.4*. The state of these bits therefore may differ between individual ICs.

0x05: *OTP_READ* – OTP MEMORY MAP

Bit	Name	Function	Comment	
23	*otp2.7*	*otp_en_SpreadCycle*	This flag determines if the driver defaults to SpreadCycle or to StealthChop.	
			0	Default: StealthChop (*GCONF.en_SpreadCycle*=0) OTP 1.0 to 1.7 and 2.0 used for StealthChop SpreadCycle settings: *HEND*=0; *HSTART*=5; *TOFF*=3
			1	Default: SpreadCycle (*GCONF.en_SpreadCycle*=1) OTP 1.0 to 1.7 and 2.0 used for SpreadCycle StealthChop settings: *PWM_GRAD*=0; *TPWM_THRS*=0; *PWM_OFS*=36; *pwm_autograd*=1
22	*otp2.6*	*OTP_IHOLD*	Reset default for standstill current *IHOLD* (used only if current reduction enabled, e.g. pin PDN_UART low). %00: *IHOLD*= 16 (53% of *IRUN*) %01: *IHOLD*= 2 (9% of *IRUN*) %10: *IHOLD*= 8 (28% of *IRUN*) %11: *IHOLD*= 24 (78% of *IRUN*) (Reset default for run current *IRUN*=31)	
21	*otp2.5*			
20	*otp2.4*	*OTP_IHOLDDELAY*	Reset default for *IHOLDDELAY* %00: *IHOLDDELAY*= 1 %01: *IHOLDDELAY*= 2 %10: *IHOLDDELAY*= 4 %11: *IHOLDDELAY*= 8	
19	*otp2.3*			
18	*otp2.2*	*otp_PWM_FREQ*	Reset default for *PWM_FREQ*: 0: *PWM_FREQ*=%01=2/683 1: *PWM_FREQ*=%10=2/512	
17	*otp2.1*	*otp_PWM_REG*	Reset default for *PWM_REG*: 0: *PWM_REG*=%1000: max. 4 increments / cycle 1: *PWM_REG*=%0010: max. 1 increment / cycle	
16	*otp2.0*	*otp_PWM_OFS*	Depending on *otp_en_SpreadCycle*	
			0	0: *PWM_OFS*=36 1: *PWM_OFS*=00 (no feed forward scaling); *pwm_autograd*=0
		OTP_CHOPCONF8	1	Reset default for *CHOPCONF*.8 (*hend1*)
15	*otp1.7*	*OTP_TPWMTHRS*	Depending on *otp_en_SpreadCycle*	
14	*otp1.6*		0	Reset default for *TPWM_THRS* as defined by (0..7): 0: *TPWM_THRS*= 0 1: *TPWM_THRS*= 200 2: *TPWM_THRS*= 300 3: *TPWM_THRS*= 400 4: *TPWM_THRS*= 500 5: *TPWM_THRS*= 800 6: *TPWM_THRS*= 1200 7: *TPWM_THRS*= 4000
13	*otp1.5*			
		OTP_CHOPCONF7…5	1	Reset default for *CHOPCONF*.5 to *CHOPCONF*.7 (*hstrt1*, *hstrt2* and *hend0*)
12	*otp1.4*	*otp_pwm_autograd*	Depending on *otp_en_SpreadCycle*	
			0	0: *pwm_autograd*=1 1: *pwm_autograd*=0

0x05: *OTP_READ* – OTP MEMORY MAP				
Bit	**Name**	**Function**	**Comment**	
		OTP_CHOPCONF4	1	Reset default for CHOPCONF.4 (*hstrt0*); (*pwm_autograd*=1)
11	*otp1.3*	*OTP_PWM_GRAD*	Depending on *otp_en_SpreadCycle*	
10	*otp1.2*		0	Reset default for *PWM_GRAD* as defined by (0..15): 0: *PWM_GRAD*= 14 1: *PWM_GRAD*= 16 2: *PWM_GRAD*= 18 3: *PWM_GRAD*= 21 4: *PWM_GRAD*= 24 5: *PWM_GRAD*= 27 6: *PWM_GRAD*= 31 7: *PWM_GRAD*= 35 8: *PWM_GRAD*= 40 9: *PWM_GRAD*= 46 10: *PWM_GRAD*= 52 11: *PWM_GRAD*= 59 12: *PWM_GRAD*= 67 13: *PWM_GRAD*= 77 14: *PWM_GRAD*= 88 15: *PWM_GRAD*= 100
9	*otp1.1*			
8	*otp1.0*			
		OTP_CHOPCONF3...0	1	Reset default for *CHOPCONF*.0 to *CHOPCONF*.3 (*TOFF*)
7	*otp0.7*	*otp_TBL*	Reset default for *TBL*: 0: *TBL*=%10 1: *TBL*=%01	
6	*otp0.6*	*otp_internalRsense*	Reset default for *GCONF.internal_Rsense* 0: External sense resistors 1: Internal sense resistors	
5	*otp0.5*	*otp_OTTRIM*	Reset default for *OTTRIM*: 0: *OTTRIM*= %00 (143°C) 1: *OTTRIM*= %01 (150°C) (internal power stage temperature about 10°C above the sensor temperature limit)	
4	*otp0.4*	*OTP_FCLKTRIM*	Reset default for *FCLKTRIM* 0: lowest frequency setting 31: highest frequency setting *Attention: This value is pre-programmed by factory clock trimming to the default clock frequency of 12MHz and differs between individual ICs! It should not be altered.*	
3	*otp0.3*			
2	*otp0.2*			
1	*otp0.1*			
0	*otp0.0*			

5.2 Velocity Dependent Control

VELOCITY DEPENDENT DRIVER FEATURE CONTROL REGISTER SET (0X10…0X1F)

R/W	Addr	n	*Register*	Bit	Description / *bit names*
W	0x10	5 + 5 + 4	*IHOLD_IRUN*	**Bit**	***IHOLD_IRUN* – Driver current control**
				4..0	*IHOLD* (*Reset default: OTP*) Standstill current (0=1/32 … 31=32/32) In combination with StealthChop mode, setting *IHOLD*=0 allows to choose freewheeling or coil short circuit (passive braking) for motor stand still.
				12..8	*IRUN* (*Reset default=31*) Motor run current (0=1/32 … 31=32/32) *Hint:* Choose sense resistors in a way, that normal *IRUN* is 16 to 31 for best microstep performance.
				19..16	*IHOLDDELAY* (*Reset default: OTP*) Controls the number of clock cycles for motor power down after standstill is detected (*stst*=1) and *TPOWERDOWN* has expired. The smooth transition avoids a motor jerk upon power down. 0: instant power down 1..15: Delay per current reduction step in multiple of 2^18 clocks
W	0x11	8	*TPOWER DOWN*		*TPOWERDOWN* (*Reset default=20*) Sets the delay time from stand still (*stst*) detection to motor current power down. Time range is about 0 to 5.6 seconds. *0…((2^8)-1)* * 2^18 t_{CLK} *Attention: A minimum setting of 2 is required to allow automatic tuning of StealthChop PWM_OFFS_AUTO.*
R	0x12	20	*TSTEP*		Actual measured time between two 1/256 microsteps derived from the step input frequency in units of 1/fCLK. Measured value is (2^20)-1 in case of overflow or stand still. The *TSTEP* related threshold uses a hysteresis of 1/16 of the compare value to compensate for jitter in the clock or the step frequency: (*Txxx**15/16)-1 is the lower compare value for each *TSTEP* based comparison. This means, that the lower switching velocity equals the calculated setting, but the upper switching velocity is higher as defined by the hysteresis setting.
W	0x13	20	*TPWMTHRS*		Sets the upper velocity for StealthChop voltage PWM mode. *TSTEP* ≥ *TPWMTHRS* - StealthChop PWM mode is enabled, if configured When the velocity exceeds the limit set by *TPWMTHRS*, the driver switches to SpreadCycle. 0: Disabled
W	0x22	24	*VACTUAL*		*VACTUAL* allows moving the motor by UART control. It gives the motor velocity in +-(2^23)-1 [µsteps / t] 0: Normal operation. Driver reacts to STEP input. /=0: Motor moves with the velocity given by *VACTUAL.* Step pulses can be monitored via INDEX output. The motor direction is controlled by the sign of *VACTUAL*.

5.3 Sequencer Registers

The sequencer registers have a pure informative character and are read-only. They help for special cases like storing the last motor position before power off in battery powered applications.

MICROSTEPPING CONTROL REGISTER SET (0X60…0X6B)					
R/W	**Addr**	**n**	***Register***	**Description / *bit names***	**Range [Unit]**
R	0x6A	10	*MSCNT*	Microstep counter. Indicates actual position in the microstep table for *CUR_A*. *CUR_B* uses an offset of 256 into the table. Reading out *MSCNT* allows determination of the motor position within the electrical wave.	0…1023
R	0x6B	9 + 9	*MSCURACT*	bit 8… 0: *CUR_A* (signed): Actual microstep current for motor phase A as read from the internal sine wave table (not scaled by current setting) bit 24… 16: *CUR_B* (signed): Actual microstep current for motor phase B as read from the internal sine wave table (not scaled by current setting)	+/-0...255

5.4 Chopper Control Registers

DRIVER REGISTER SET (0X6C…0X7F)						
R/W	**Addr**	**n**	***Register***	**Description / *bit names***		**Range [Unit]**
RW	0x6C	32	***CHOPCONF***	Chopper and driver configuration *See separate table!*		*Reset default= 0x10000053*
R	0x6F	32	***DRV_ STATUS***	Driver status flags and current level read back *See separate table!*		
RW	0x70	22	***PWMCONF***	StealthChop PWM chopper configuration *See separate table!*		*Reset default= 0xC10D0024*
R	0x71	9+8	*PWM_SCALE*	Results of StealthChop amplitude regulator. These values can be used to monitor automatic PWM amplitude scaling (255=max. voltage).		
				bit 7… 0	*PWM_SCALE_SUM*: Actual PWM duty cycle. This value is used for scaling the values *CUR_A* and *CUR_B* read from the sine wave table.	0…255
				bit 24… 16	*PWM_SCALE_AUTO*: 9 Bit signed offset added to the calculated PWM duty cycle. This is the result of the automatic amplitude regulation based on current measurement.	signed -255…+255
R	0x72	8+8	*PWM_AUTO*	These automatically generated values can be read out in order to determine a default / power up setting for *PWM_GRAD* and *PWM_OFS*.		
				bit 7… 0	*PWM_OFS_AUTO*: Automatically determined offset value	0…255
				bit 23… 16	*PWM_GRAD_AUTO*: Automatically determined gradient value	0…255

5.4.1 *CHOPCONF* – Chopper Configuration

0X6C: *CHOPCONF* – CHOPPER CONFIGURATION			
Bit	**Name**	**Function**	**Comment**
31	*diss2vs*	Low side short protection disable	0: Short protection low side is on 1: Short protection low side is disabled
30	*diss2g*	short to GND protection disable	0: Short to GND protection is on 1: Short to GND protection is disabled
29	*dedge*	enable double edge step pulses	1: Enable step impulse at each step edge to reduce step frequency requirement. This mode is not compatible with the step filtering function (*multistep_filt*)
28	*intpol*	interpolation to 256 microsteps	1: The actual microstep resolution (*MRES*) becomes extrapolated to 256 microsteps for smoothest motor operation. (Default: 1)
27	*mres3*	*MRES* micro step resolution	%0000: Native 256 microstep setting.
26	*mres2*		
25	*mres1*		%0001 … %1000: 128, 64, 32, 16, 8, 4, 2, FULLSTEP Reduced microstep resolution. The resolution gives the number of microstep entries per sine quarter wave. When choosing a lower microstep resolution, the driver automatically uses microstep positions which result in a symmetrical wave. Number of microsteps per step pulse = 2^*MRES* (Selection by pins unless disabled by *GCONF. mstep_reg_select*)
24	*mres0*		
23	-	reserved	set to 0
22			
21			
20			
19			
18			
17	*vsense*	sense resistor voltage based current scaling	0: Low sensitivity, high sense resistor voltage 1: High sensitivity, low sense resistor voltage
16	*tbl1*	*TBL* blank time select	%00 … %11: Set comparator blank time to 16, 24, 32 or 40 clocks *Hint*: %00 or %01 is recommended for most applications (Default: OTP)
15	*tbl0*		
14	-	reserved	set to 0
13			
12			
11			
10	*hend3*	*HEND* hysteresis low value *OFFSET* sine wave offset	%0000 … %1111: Hysteresis is -3, -2, -1, 0, 1, …, 12 (1/512 of this setting adds to current setting) This is the hysteresis value which becomes used for the hysteresis chopper. (Default: OTP, resp. 5 in StealthChop mode)
9	*hend2*		
8	*hend1*		
7	*hend0*		
6	*hstrt2*	*HSTRT* hysteresis start value added to *HEND*	%000 … %111: Add 1, 2, …, 8 to hysteresis low value *HEND* (1/512 of this setting adds to current setting) *Attention: Effective HEND+HSTRT ≤ 16.* *Hint:* Hysteresis decrement is done each 16 clocks
5	*hstrt1*		
4	*hstrt0*		

0x6C: *CHOPCONF* – Chopper Configuration			
Bit	**Name**	**Function**	**Comment**
			(Default: OTP, resp. 0 in StealthChop mode)
3	*toff3*	*TOFF* off time and driver enable	Off time setting controls duration of slow decay phase N_{CLK}= 24 + 32**TOFF* %0000: Driver disable, all bridges off %0001: 1 – use only with *TBL* ≥ 2 %0010 … %1111: 2 … 15 (Default: OTP, resp. 3 in StealthChop mode)
2	*toff2*		
1	*toff1*		
0	*toff0*		

5.4.2 *PWMCONF* – Voltage PWM Mode StealthChop

0x70: *PWMCONF* – VOLTAGE MODE PWM STEALTHCHOP				
Bit	**Name**	**Function**	**Comment**	
31 30 29 28	*PWM_LIM*	PWM automatic scale amplitude limit when switching on	Limit for *PWM_SCALE_AUTO* when switching back from SpreadCycle to StealthChop. This value defines the upper limit for bits 7 to 4 of the automatic current control when switching back. It can be set to reduce the current jerk during mode change back to StealthChop. It does not limit *PWM_GRAD* or *PWM_GRAD_AUTO* offset. (Default = 12)	
27 26 25 24	*PWM_REG*	Regulation loop gradient	User defined maximum PWM amplitude change per half wave when using *pwm_autoscale*=1. (1…15): 1: 0.5 increments (slowest regulation) 2: 1 increment (default with *OTP*2.1=1) 3: 1.5 increments 4: 2 increments … 8: 4 increments (default with *OTP*2.1=0) … 15: 7.5 increments (fastest regulation)	
23	-	reserved	set to 0	
22	-	reserved	set to 0	
21 20	*freewheel1* *freewheel0*	Allows different standstill modes	Stand still option when motor current setting is zero (*I_HOLD*=0). %00: Normal operation %01: Freewheeling %10: Coil shorted using LS drivers %11: Coil shorted using HS drivers	
19	*pwm_ autograd*	PWM automatic gradient adaptation	0	Fixed value for *PWM_GRAD* (*PWM_GRAD_AUTO* = *PWM_GRAD*)
			1	Automatic tuning (only with *pwm_autoscale*=1) *PWM_GRAD_AUTO* is initialized with *PWM_GRAD* and becomes optimized automatically during motion. Preconditions 1. *PWM_OFS_AUTO* has been automatically initialized. This requires standstill at IRUN for >130ms in order to a) detect standstill b) wait > 128 chopper cycles at *IRUN* and c) regulate *PWM_OFS_AUTO* so that -1 < *PWM_SCALE_AUTO* < 1 2. Motor running and 1.5 * *PWM_OFS_AUTO* < *PWM_SCALE_SUM* < 4* *PWM_OFS_AUTO* and *PWM_SCALE_SUM* < 255. Time required for tuning *PWM_GRAD_AUTO* About 8 fullsteps per change of +/-1.
18	*pwm_ autoscale*	PWM automatic amplitude scaling	0	User defined feed forward PWM amplitude. The current settings *IRUN* and *IHOLD* have no influence! The resulting PWM amplitude (limited to 0…255) is: *PWM_OFS* * ((*CS_ACTUAL*+1) / 32) + *PWM_GRAD* * 256 / *TSTEP*
			1	Enable automatic current control *(Reset default)*
17	*pwm_freq1*	PWM frequency	%00: f_{PWM}=2/1024 f_{CLK}	

0x70: *PWMCONF* – Voltage mode PWM StealthChop			
Bit	**Name**	**Function**	**Comment**
16	*pwm_freq0*	selection	%01: f_{PWM}=2/683 f_{CLK} %10: f_{PWM}=2/512 f_{CLK} %11: f_{PWM}=2/410 f_{CLK}
15 14 13 12 11 10 9 8	*PWM_ GRAD*	User defined amplitude gradient	Velocity dependent gradient for PWM amplitude: *PWM_GRAD* * 256 / *TSTEP* This value is added to *PWM_AMPL* to compensate for the velocity-dependent motor back-EMF. Use *PWM_GRAD* as initial value for automatic scaling to speed up the automatic tuning process. To do this, set *PWM_GRAD* to the determined, application specific value, with *pwm_autoscale*=0. Only afterwards, set *pwm_autoscale*=1. Enable StealthChop when finished. Alternatively program the determined value to OTP. It automatically will be loaded upon power up, even when StealthChop becomes enabled right away. *Hint:* After initial tuning, the required initial value can be read out from *PWM_GRAD_AUTO.*
7 6 5 4 3 2 1 0	*PWM_ OFS*	User defined amplitude (offset)	User defined PWM amplitude offset (0-255) related to full motor current (*CS_ACTUAL*=31) in stand still. (*Reset default=36*) Use *PWM_OFS* as initial value for automatic scaling to speed up the automatic tuning process. To do this, set *PWM_OFS* to the determined, application specific value, with *pwm_autoscale*=0. Only afterwards, set *pwm_autoscale*=1. Enable StealthChop when finished. *PWM_OFS* = 0 will disable scaling down motor current below a motor specific lower measurement threshold. This setting should only be used under certain conditions, i.e. when the power supply voltage can vary up and down by a factor of two or more. It prevents the motor going out of regulation, but it also prevents power down below the regulation limit. *PWM_OFS* > 0 allows automatic scaling to low PWM duty cycles even below the lower regulation threshold. This allows low (standstill) current settings based on the actual (hold) current scale (register *IHOLD_IRUN*).

5.4.3 DRV_STATUS – Driver Status Flags

0X6F: DRV_STATUS – DRIVER STATUS FLAGS AND CURRENT LEVEL READ BACK			
Bit	**Name**	**Function**	**Comment**
31	*stst*	standstill indicator	This flag indicates motor stand still in each operation mode. This occurs 2^20 clocks after the last step pulse.
30	*stealth*	StealthChop indicator	1: Driver operates in StealthChop mode 0: Driver operates in SpreadCycle mode
29	-	reserved	Ignore these bits.
28			
27			
26			
25			
24			
23	-	reserved	Ignore these bits.
22			
21			
20	*CS_ ACTUAL*	actual motor current / smart energy current	Actual current control scaling, for monitoring the function of the automatic current scaling.
19			
18			
17			
16			
15	-	reserved	Ignore these bits.
14			
13			
12			
11	*t157*	157°C comparator	1: Temperature threshold is exceeded
10	*t150*	150°C comparator	1: Temperature threshold is exceeded
9	*t143*	143°C comparator	1: Temperature threshold is exceeded
8	*t120*	120°C comparator	1: Temperature threshold is exceeded
7	*olb*	open load indicator phase B	1: Open load detected on phase A or B. *Hint:* This is just an informative flag. The driver takes no action upon it. False detection may occur in fast motion and standstill. Check during slow motion, only.
6	*ola*	open load indicator phase A	
5	*s2vsb*	low side short indicator phase B	1: Short on low-side MOSFET detected on phase A or B. The driver becomes disabled. The flags stay active, until the driver is disabled by software (TOFF=0) or by the ENN input. Flags are separate for both chopper modes.
4	*s2vsa*	low side short indicator phase A	
3	*s2gb*	short to ground indicator phase B	1: Short to GND detected on phase A or B. The driver becomes disabled. The flags stay active, until the driver is disabled by software (TOFF=0) or by the ENN input. Flags are separate for both chopper modes.
2	*s2ga*	short to ground indicator phase A	
1	*ot*	overtemperature flag	1: The selected overtemperature limit has been reached. Drivers become disabled until *otpw* is also cleared due to cooling down of the IC. The overtemperature flag is common for both bridges.
0	*otpw*	overtemperature pre-warning flag	1: The selected overtemperature pre-warning threshold is exceeded. The overtemperature pre-warning flag is common for both bridges.

6 StealthChop™

StealthChop is an extremely quiet mode of operation for stepper motors. It is based on a voltage mode PWM. In case of standstill and at low velocities, the motor is absolutely noiseless. Thus, StealthChop operated stepper motor applications are very suitable for indoor or home use. The motor operates absolutely free of vibration at low velocities. With StealthChop, the motor current is applied by driving a certain effective voltage into the coil, using a voltage mode PWM. With the enhanced StealthChop2, the driver automatically adapts to the application for best performance. No more configurations are required. Optional configuration allows for tuning the setting in special cases, or for storing initial values for the automatic adaptation algorithm. For high velocity drives consider SpreadCycle in combination with StealthChop.

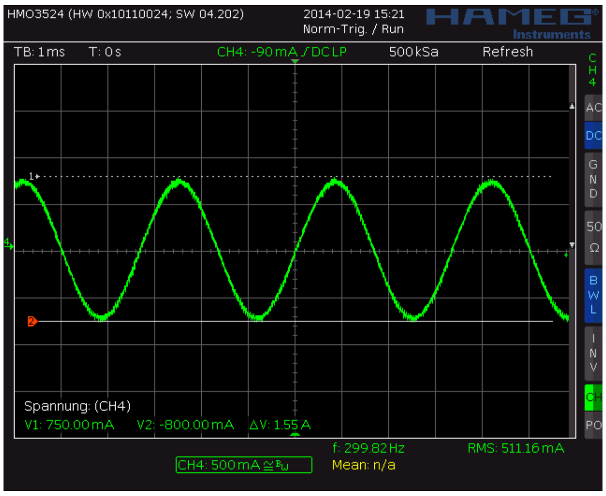


Figure 6.1 Motor coil sine wave current with StealthChop (measured with current probe)

6.1 Automatic Tuning

StealthChop2 integrates an automatic tuning procedure (AT), which adapts the most important operating parameters to the motor automatically. This way, StealthChop2 allows high motor dynamics and supports powering down the motor to very low currents. Just two steps have to be respected by the motion controller for best results: Start with the motor in standstill, but powered with nominal run current (AT#1). Move the motor at a medium velocity, e.g. as part of a homing procedure (AT#2). Figure 6.2 shows the tuning procedure.
Border conditions in for AT#1 and AT#2 are shown in the following table:

AUTOMATIC TUNING TIMING AND BORDER CONDITIONS			
Step	**Parameter**	**Conditions**	**Duration**
AT#1	*PWM_ OFS_AUTO*	- Motor in standstill and actual current scale (*CS*) is identical to run current (*IRUN*). - If standstill reduction is enabled (pin PDN_UART=0), an initial step pulse switches the drive back to run current. - Pins VS and VREF at operating level.	≤ 2^20+2*2^18 t_{CLK}, ≤ 130ms (with internal clock)
AT#2	*PWM_ GRAD_AUTO*	- Motor must move at a velocity, where a significant amount of back EMF is generated and where the full run current can be reached. Conditions: - 1.5 * *PWM_OFS_AUTO* < *PWM_SCALE_SUM* < 4 * *PWM_OFS_AUTO* - *PWM_SCALE_SUM* < 255. *Hint: A typical range is 60-300 RPM. Determine best conditions with the evaluation board and monitor PWM_SCALE_AUTO going down to zero during tuning.*	8 fullsteps are required for a change of +/-1. For a typical motor with *PWM_GRAD_AUTO* optimum at 64 or less, up to 400 fullsteps are required when starting from OTP default 14.

Determine best conditions for automatic tuning with the evaluation board.
Monitor *PWM_SCALE_AUTO* going down to zero during the constant velocity phase in AT#2 tuning. This indicates a successful tuning.

Attention:
Operating in StealthChop without proper tuning can lead to high motor currents during a deceleration ramp, especially with low resistive motors and fast deceleration settings. Follow the automatic tuning process and check optimum tuning conditions using the evaluation board. It is recommended to use an initial value for settings *PWM_OFS* and *PWM_GRAD* determined per motor type. Avoid hard stops from high velocities, even after tuning StealthChop settings. A deceleration ramp is required in each case.

When powering up in StealthChop, make sure that the supply voltage ramps to the final voltage. Powering up to a lower voltage (e.g. 5V) will lead to wrong tuning, in case the motor becomes enabled at the lower voltage. Alternatively disable the motor during power-up.

Known Limitations:
Successful completion of AT#1 tuning phase is not safely detected in all cases. It might require multiple motor start / stop events to safely detect completion.
Successful determination is mandatory for AT#2: Tuning of *PWM_GRAD* will not start when AT#1 has not completed.

Solution a):
Complete automatic tuning phase AT#1 process, by using a slow-motion sequence which leads to standstill detection in between of each two steps. Use a velocity of 8 (6 Hz) or lower and execute minimum 10 steps during AT#1 phase.

Solution b):
Complete automatic tuning phase AT#1 process, by a doing sequence of four or more initial motions (using reduced acceleration, to match the fact that AT#2 has not yet completed) after power up. Make sure, that the driver is at standstill more than 130ms in between of each two motions.

Solution c):
Store application specific *PWM_GRAD* to OTP memory.
Or, store initial parameters for *PWM_GRAD_AUTO* for the application and initialize by UART interface.
Therefore, use the motor and operating conditions determined for the application and do a complete automatic tuning sequence (refer to *a)*). Note the resulting *PWM_GRAD_AUTO* value and use it for initialization of *PWM_GRAD*, or program the value to OTP memory. With this, tuning of AT#2 phase is not mandatory in the application and can be skipped. Automatic tuning will optimize settings during further operation. Combine with *a)* if desired.

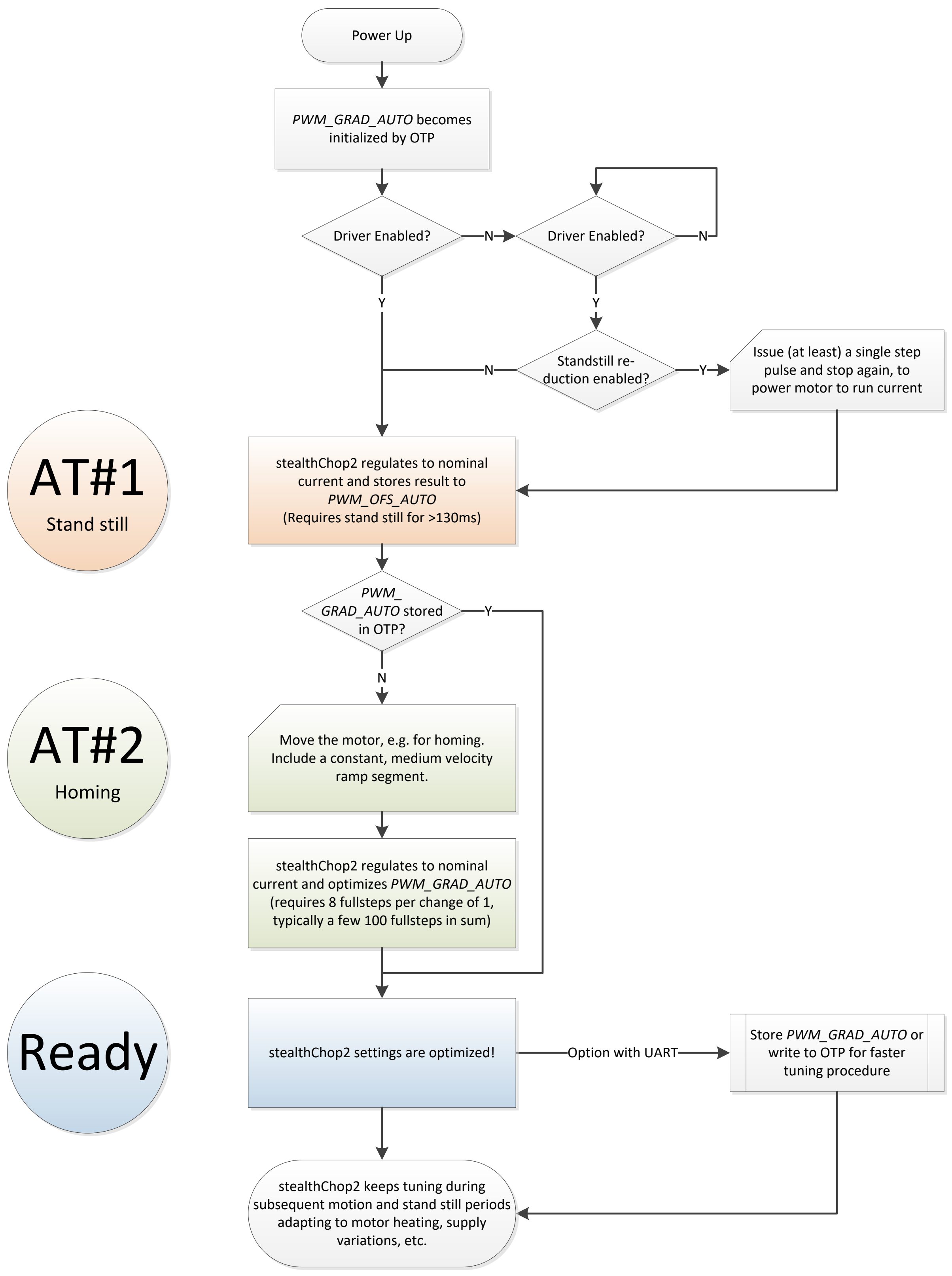

Power Up
PWM_GRAD_AUTO becomes initialized by OTP
Driver Enabled?
N
Driver Enabled?
N
Y
Y
Standstill re-duction enabled?
N
Y
Issue (at least) a single step pulse and stop again, to power motor to run current
AT#1
Stand still
stealthChop2 regulates to nominal current and stores result to PWM_OFS_AUTO (Requires stand still for >130ms)
PWM_GRAD_AUTO stored in OTP?
Y
N
AT#2
Homing
Move the motor, e.g. for homing. Include a constant, medium velocity ramp segment.
stealthChop2 regulates to nominal current and optimizes PWM_GRAD_AUTO (requires 8 fullsteps per change of 1, typically a few 100 fullsteps in sum)
Ready
stealthChop2 settings are optimized!
Option with UART
Store PWM_GRAD_AUTO or write to OTP for faster tuning procedure
stealthChop2 keeps tuning during subsequent motion and stand still periods adapting to motor heating, supply variations, etc.

Figure 6.2 StealthChop2 automatic tuning procedure

Attention
Modifying VREF or the supply voltage VS invalidates the result of the automatic tuning process. Motor current regulation cannot compensate significant changes until next AT#1 phase. Automatic tuning adapts to changed conditions whenever AT#1 and AT#2 conditions are fulfilled in the later operation.

6.2 StealthChop Options

In order to match the motor current to a certain level, the effective PWM voltage becomes scaled depending on the actual motor velocity. Several additional factors influence the required voltage level to drive the motor at the target current: The motor resistance, its back EMF (i.e. directly proportional to its velocity) as well as the actual level of the supply voltage. Two modes of PWM regulation are provided: The automatic tuning mode (AT) using current feedback (*pwm_autoscale* = 1, *pwm_autograd* = 1) and a feed forward velocity-controlled mode (*pwm_autoscale* = 0). The feed forward velocity-controlled mode will not react to a change of the supply voltage or to events like a motor stall, but it provides very stable amplitude. It does not use nor require any means of current measurement. This is perfect when motor type and supply voltage are well known. Therefore, we recommend the automatic mode, unless current regulation is not satisfying in the given operating conditions.

It is recommended to operate in automatic tuning mode.

Non-automatic mode (*pwm_autoscale=0*) should be taken into account only with well-known motor and operating conditions. In this case, programming via the UART interface is required. The operating parameters *PWM_GRAD* and *PWM_OFS* can be determined in automatic tuning mode initially.

Hint: In non-automatic mode the power supply current directly reflects mechanical load on the motor.

The StealthChop PWM frequency can be chosen in four steps in order to adapt the frequency divider to the frequency of the clock source. A setting in the range of 20-50kHz is good for most applications. It balances low current ripple and good higher velocity performance vs. dynamic power dissipation.

CHOICE OF PWM FREQUENCY FOR STEALTHCHOP

Clock frequency f_{CLK}	PWM_FREQ=%00 f_{PWM}=2/1024 f_{CLK}	PWM_FREQ=%01 f_{PWM}=2/683 f_{CLK} (default)	PWM_FREQ=%10 f_{PWM}=2/512 f_{CLK} (OTP option)	PWM_FREQ=%11 f_{PWM}=2/410 f_{CLK}
18MHz	35.2kHz	52.7kHz	70.3kHz	87.8kHz
16MHz	31.3kHz	46.9kHz	62.5kHz	78.0kHz
12MHz (internal)	23.4kHz	35.1kHz	46.9kHz	58.5kHz
10MHz	19.5kHz	29.3kHz	39.1kHz	48.8kHz
8MHz	15.6kHz	23.4kHz	31.2kHz	39.0kHz

Table 6.1 Choice of PWM frequency – green / light green: recommended

6.3 StealthChop Current Regulator

In StealthChop voltage PWM mode, the autoscaling function (*pwm_autoscale* = 1, *pwm_auto_grad* = 1) regulates the motor current to the desired current setting. Automatic scaling is used as part of the automatic tuning process (AT), and for subsequent tracking of changes within the motor parameters. The driver measures the motor current during the chopper on time and uses a proportional regulator to regulate *PWM_SCALE_AUTO* in order match the motor current to the target current. *PWM_REG* is the proportionality coefficient for this regulator. Basically, the proportionality coefficient should be as small as possible in order to get a stable and soft regulation behavior, but it must be large enough to allow the driver to quickly react to changes caused by variation of the motor target current (e.g. change of VREF). During initial tuning step AT#2, *PWM_REG* also compensates for the change of motor velocity. Therefore, a high acceleration during AT#2 will require a higher setting of *PWM_REG*. With careful selection of homing velocity and acceleration, a minimum setting of the regulation gradient often is sufficient (*PWM_REG*=1). *PWM_REG* setting should be optimized for the fastest required acceleration and deceleration ramp (compare Figure 6.3 and Figure 6.4). The quality of the setting *PWM_REG* in phase AT#2 and the finished automatic tuning procedure (or non-automatic settings for *PWM_OFS* and *PWM_GRAD*) can be examined when monitoring motor current during an acceleration phase Figure 6.5.

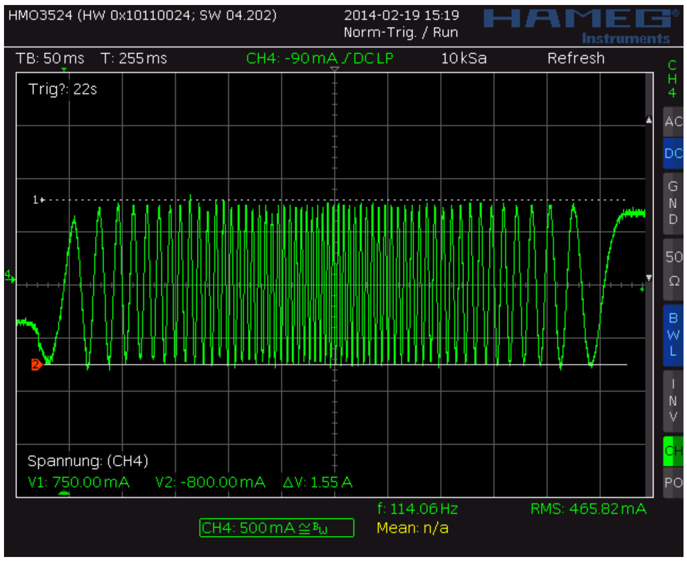


Figure 6.3 Scope shot: good setting for PWM_REG

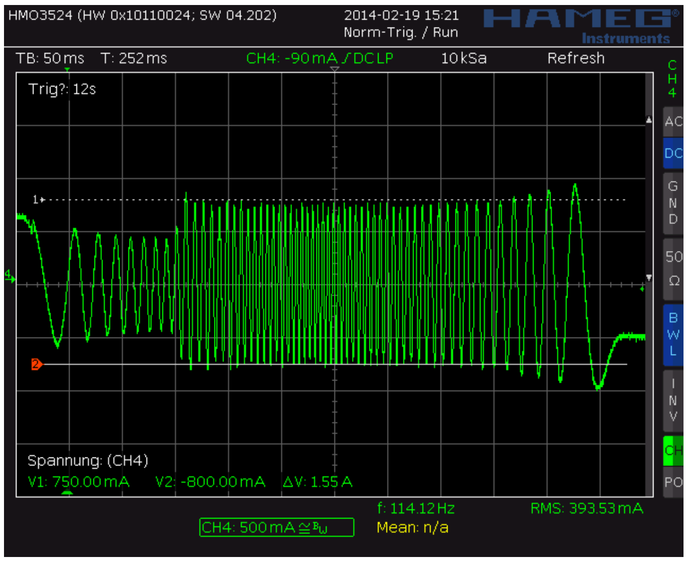


Figure 6.4 Scope shot: too small setting for PWM_REG during AT#2

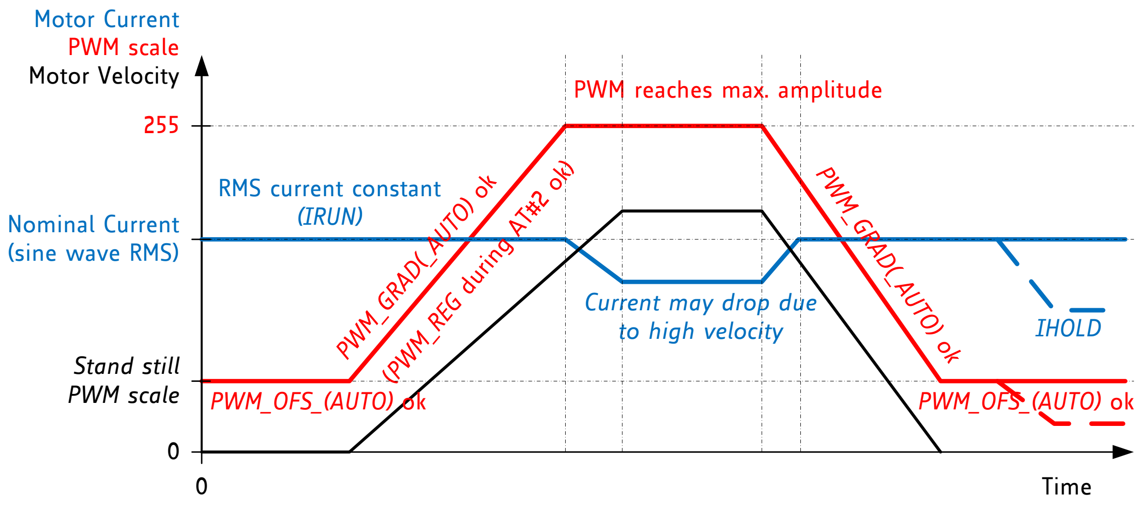


Figure 6.5 Successfully determined PWM_GRAD(_AUTO) and PWM_OFS(_AUTO)

Quick Start
For a quick start, see the Quick Configuration Guide in chapter 14.

6.3.1 Lower Current Limit

The StealthChop current regulator imposes a lower limit for motor current regulation. As the coil current can be measured in the shunt resistor during chopper on phase only, a minimum chopper duty cycle allowing coil current regulation is given by the blank time as set by *TBL* and by the chopper frequency setting. Therefore, the motor specific minimum coil current in StealthChop autoscaling mode rises with the supply voltage and with the chopper frequency. A lower blanking time allows a lower current limit. It is important for the correct determination of *PWM_OFS_AUTO*, that in AT#1 the run current set by the sense resistor, VREF and *IRUN* is well within the regulation range. Lower currents (e.g. for standstill power down) are automatically realized based on *PWM_OFS_AUTO* and *PWM_GRAD_AUTO* respectively based on *PWM_OFS* and *PWM_GRAD* with non-automatic current scaling. The freewheeling option allows going to zero motor current.

Lower motor coil current limit for StealthChop2 automatic tuning:

$$I_{Lower\ Limit} = t_{BLANK} * f_{PWM} * \frac{V_M}{R_{COIL}}$$

With V_M the motor supply voltage and R_{COIL} the motor coil resistance.
$I_{Lower\ Limit}$ can be treated as a thumb value for the minimum nominal *IRUN* motor current setting.

EXAMPLE:

A motor has a coil resistance of 5Ω, the supply voltage is 24V. With *TBL*=%01 and *PWM_FREQ*=%00, t_{BLANK} is 24 clock cycles, f_{PWM} is 2/(1024 clock cycles):

$$I_{Lower\ Limit} = 24\ t_{CLK} * \frac{2}{1024\ t_{CLK}} * \frac{24V}{5\Omega} = \frac{24}{512} * \frac{24V}{5\Omega} = 225mA$$

This means, the motor target current for automatic tuning must be 225mA or more, taking into account all relevant settings. This lower current limit also applies for modification of the motor current via the analog input VREF.

Attention
For automatic tuning, a lower coil current limit applies. The motor current in automatic tuning phase AT#1 must exceed this lower limit. $I_{LOWER\ LIMIT}$ can be calculated or measured using a current probe. Setting the motor run-current or hold-current below the lower current limit during operation by modifying *IRUN* and *IHOLD* is possible after successful automatic tuning.

The lower current limit also limits the capability of the driver to respond to changes of VREF.

6.4 Velocity Based Scaling

Velocity based scaling scales the StealthChop amplitude based on the time between each two steps, i.e. based on *TSTEP*, measured in clock cycles. This concept basically does not require a current measurement, because no regulation loop is necessary. A pure velocity-based scaling is available via UART programming, only, when setting *pwm_autoscale* = 0. The basic idea is to have a linear approximation of the voltage required to drive the target current into the motor. The stepper motor has a certain coil resistance and thus needs a certain voltage amplitude to yield a target current based on the basic formula I=U/R. With R being the coil resistance, U the supply voltage scaled by the PWM value, the current I results. The initial value for PWM_AMPL can be calculated:

$$PWM\_AMPL = \frac{374 * R_{COIL} * I_{COIL}}{V_M}$$

With V_M the motor supply voltage and I_{COIL} the target RMS current

The effective PWM voltage U_{PWM} (1/SQRT(2) x peak value) results considering the 8 bit resolution and 248 sine wave peak for the actual PWM amplitude shown as *PWM_SCALE*:

$$U_{PWM} = V_M * \frac{PWM\_SCALE}{256} * \frac{248}{256} * \frac{1}{\sqrt{2}} = V_M * \frac{PWM\_SCALE}{374}$$

With rising motor velocity, the motor generates an increasing back EMF voltage. The back EMF voltage is proportional to the motor velocity. It reduces the PWM voltage effective at the coil resistance and thus current decreases. The TMC22xx provides a second velocity dependent factor (*PWM_GRAD*) to compensate for this. The overall effective PWM amplitude (*PWM_SCALE_SUM*) in this mode automatically is calculated in dependence of the microstep frequency as:

$$PWM\_SCALE\_SUM = PWM\_OFS + PWM\_GRAD * 256 * \frac{f_{STEP}}{f_{CLK}}$$

With f_{STEP} being the microstep frequency for 256 microstep resolution equivalent
and f_{CLK} the clock frequency supplied to the driver or the actual internal frequency

As a first approximation, the back EMF subtracts from the supply voltage and thus the effective current amplitude decreases. This way, a first approximation for *PWM_GRAD* setting can be calculated:

$$PWM\_GRAD = C_{BEMF}\left[\frac{V}{\frac{rad}{s}}\right] * 2\pi * \frac{f_{CLK} * 1.46}{V_M * MSPR}$$

C_{BEMF} is the back EMF constant of the motor in Volts per radian/second.
MSPR is the number of microsteps per rotation, e.g. 51200 = 256µsteps multiplied by 200 fullsteps for a 1.8° motor.

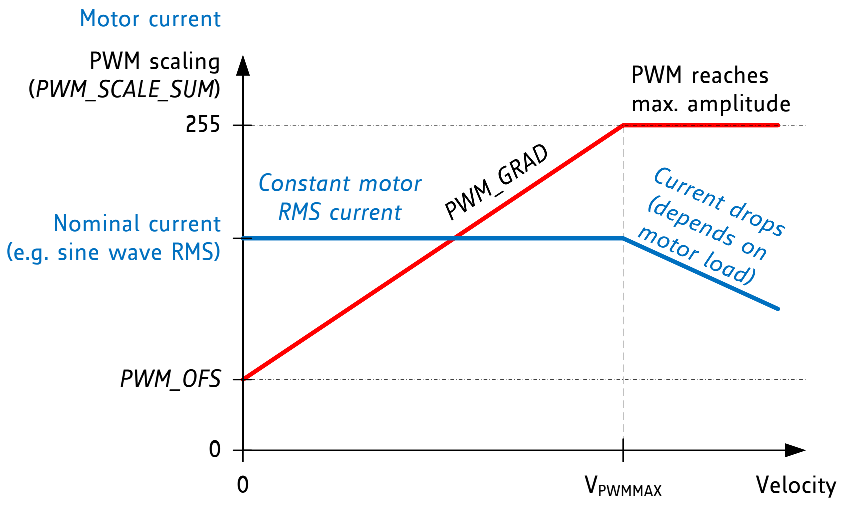


Figure 6.6 Velocity based PWM scaling (pwm_autoscale=0)

Hint
The values for *PWM_OFS* and *PWM_GRAD* can easily be optimized by tracing the motor current with a current probe on the oscilloscope. Alternatively, automatic tuning determines these values and they can be read out from *PWM_OFS_AUTO* and *PWM_GRAD_AUTO*.

Understanding the back EMF constant of a motor

The back EMF constant is the voltage a motor generates when turned with a certain velocity. Often motor datasheets do not specify this value, as it can be deducted from motor torque and coil current rating. Within SI units, the numeric value of the back EMF constant C_{BEMF} has the same numeric value as the numeric value of the torque constant. For example, a motor with a torque constant of 1 Nm/A would have a C_{BEMF} of 1V/rad/s. Turning such a motor with 1 rps (1 rps = 1 revolution per second = 6.28 rad/s) generates a back EMF voltage of 6.28V. Thus, the back EMF constant can be calculated as:

$$C_{BEMF}\left[\frac{V}{rad/s}\right] = \frac{HoldingTorque[Nm]}{2 * I_{COILNOM}[A]}$$

$I_{COILNOM}$ is the motor's rated phase current for the specified holding torque
HoldingTorque is the motor specific holding torque, i.e. the torque reached at $I_{COILNOM}$ on both coils. The torque unit is [Nm] where 1Nm = 100Ncm = 1000mNm.
The voltage is valid as RMS voltage per coil, thus the nominal current is multiplied by 2 in this formula, since the nominal current assumes a full step position, with two coils operating.

6.5 Combining StealthChop and SpreadCycle

For applications requiring high velocity motion, SpreadCycle may bring more stable operation in the upper velocity range. To combine no-noise operation with highest dynamic performance, the TMC22xx allows combining StealthChop and SpreadCycle based on a velocity threshold (Figure 6.7). A velocity threshold (*TPWMTHRS*) can be preprogrammed to OTP to support this mode even in standalone operation. With this, StealthChop is only active at low velocities.

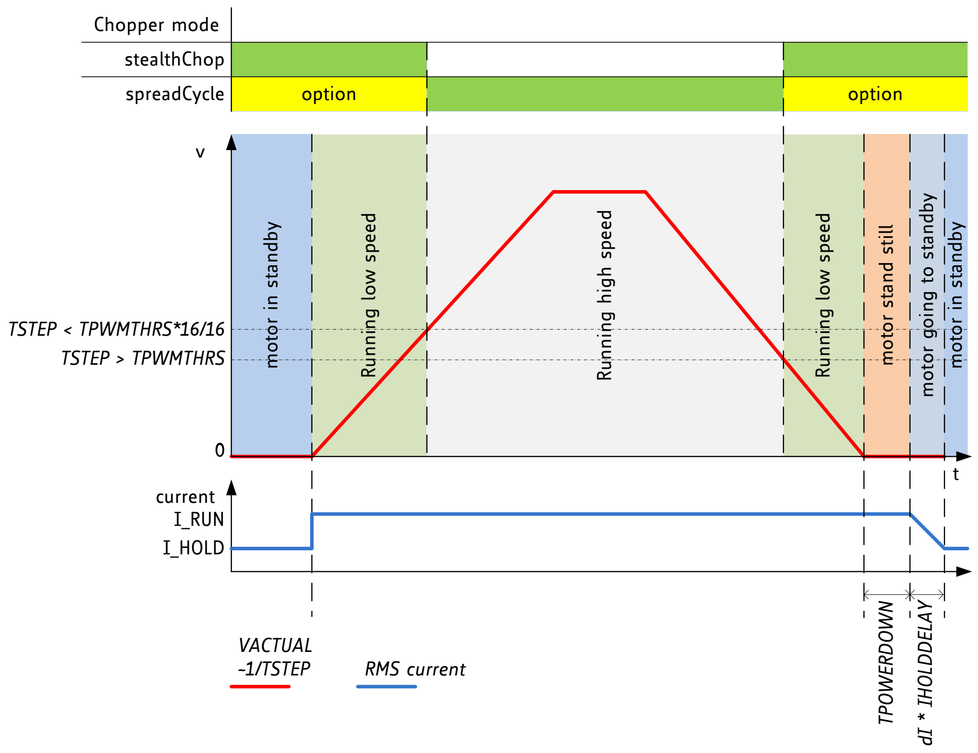


Figure 6.7 TPWMTHRS for optional switching to SpreadCycle

As a first step, both chopper principles should be parameterized and optimized individually (SpreadCycle settings may be programmed to OTP memory). In a next step, a transfer velocity has to be fixed. For example, StealthChop operation is used for precise low speed positioning, while SpreadCycle shall be used for highly dynamic motion. *TPWMTHRS* determines the transition velocity. Read out *TSTEP* when moving at the desired velocity and program the resulting value to *TPWMTHRS*. Use a low transfer velocity to avoid a jerk at the switching point.

A jerk occurs when switching at higher velocities, because the back-EMF of the motor (which rises with the velocity) causes a phase shift of up to 90° between motor voltage and motor current. So when switching at higher velocities between voltage PWM and current PWM mode, this jerk will occur with increased intensity. A high jerk may even produce a temporary overcurrent condition (depending on the motor coil resistance). At low velocities (e.g. 1 to a few 10 RPM), it can be completely neglected for most motors. Therefore, consider the switching jerk when choosing *TPWMTHRS*. Set *TPWMTHRS* zero if you want to work with StealthChop only.

When enabling the StealthChop mode the first time using automatic current regulation, the motor must be at stand still in order to allow a proper current regulation. When the drive switches to StealthChop at a higher velocity, StealthChop logic stores the last current regulation setting until the motor returns to a lower velocity again. This way, the regulation has a known starting point when returning to a lower velocity, where StealthChop becomes re-enabled. Therefore, neither the velocity threshold nor the supply voltage must be considerably changed during the phase while the chopper is switched to a different mode, because otherwise the motor might lose steps or the instantaneous current might be too high or too low.

A motor stall or a sudden change in the motor velocity may lead to the driver detecting a short circuit or to a state of automatic current regulation, from which it cannot recover. Clear the error flags and restart the motor from zero velocity to recover from this situation.

Hint
Start the motor from standstill when switching on StealthChop the first time and keep it stopped for at least 128 chopper periods to allow StealthChop to do initial standstill current control.

6.6 Flags in StealthChop

UART

As StealthChop uses voltage mode driving, status flags based on current measurement respond slower, respectively the driver reacts delayed to sudden changes of back EMF, like on a motor stall.

Attention
A motor stall, or abrupt stop of the motion during operation in StealthChop can trigger an overcurrent condition. Depending on the previous motor velocity, and on the coil resistance of the motor, it significantly increases motor current for a time of several 10ms. With low velocities, where the back EMF is just a fraction of the supply voltage, there is no danger of triggering the short detection.

6.6.1 Open Load Flags

In StealthChop mode, status information is different from the cycle-by-cycle regulated SpreadCycle mode. OLA and OLB show if the current regulation sees that the nominal current can be reached on both coils.

- A flickering OLA or OLB can result from asymmetries in the sense resistors or in the motor coils.
- An interrupted motor coil leads to a continuously active open load flag for the coil.
- One or both flags are active, if the current regulation did not succeed in scaling up to the full target current within the last few fullsteps (because no motor is attached or a high velocity exceeds the PWM limit).

If desired, do an on-demand open load test using the SpreadCycle chopper, as it delivers the safest result. With StealthChop, *PWM_SCALE_SUM* can be checked to detect the correct coil resistance.

6.6.2 PWM_SCALE_SUM Informs about the Motor State

Information about the motor state is available with automatic scaling by reading out *PWM_SCALE_SUM*. As this parameter reflects the actual voltage required to drive the target current into the motor, it depends on several factors: motor load, coil resistance, supply voltage, and current setting. Therefore, an evaluation of the *PWM_SCALE_SUM* value allows checking the motor operation point. When reaching the limit (255), the current regulator cannot sustain the full motor current, e.g. due to a drop in supply voltage.

6.7 Freewheeling and Passive Braking

StealthChop provides different options for motor standstill. These options can be enabled by setting the standstill current *IHOLD* to zero and choosing the desired option using the *FREEWHEEL* setting. The desired option becomes enabled after a time period specified by *TPOWERDOWN* and *IHOLD_DELAY*. Current regulation becomes frozen once the motor target current is at zero current in order to ensure a quick startup. With the freewheeling options, both freewheeling and passive braking can be realized. Passive braking is an effective eddy current motor braking, which consumes a minimum of energy, because no active current is driven into the coils. However, passive braking will allow slow turning of the motor when a continuous torque is applied.

Hint
Operate the motor within your application when exploring StealthChop. Motor performance often is better with a mechanical load, because it prevents the motor from stalling due mechanical oscillations which can occur without load.

PARAMETERS RELATED TO STEALTHCHOP			
Parameter	**Description**	**Setting**	**Comment**
en_spread_ cycle	General disable for use of StealthChop (register *GCONF*). The input SPREAD is XORed to this flag.	1	Do not use StealthChop
		0	StealthChop enabled
TPWMTHRS	Specifies the upper velocity for operation in StealthChop. Entry the *TSTEP* reading (time between two microsteps) when operating at the desired threshold velocity.	0 … 1048575	StealthChop is disabled if *TSTEP* falls *TPWMTHRS*
PWM_LIM	Limiting value for limiting the current jerk when switching from SpreadCycle to StealthChop. Reduce the value to yield a lower current jerk.	0 … 15	Upper four bits of 8 bit amplitude limit (Default=12)
pwm_ autoscale	Enable automatic current scaling using current measurement or use forward controlled velocity based mode.	0	Forward controlled mode
		1	Automatic scaling with current regulator
pwm_ autograd	Enable automatic tuning of *PWM_GRAD_AUTO*	0	disable, use *PWM_GRAD* from register instead
		1	enable
PWM_FREQ	PWM frequency selection. Use the lowest setting giving good results. The frequency measured at each of the chopper outputs is half of the effective chopper frequency f_{PWM}.	0	f_{PWM}=2/1024 f_{CLK}
		1	f_{PWM}=2/683 f_{CLK}
		2	f_{PWM}=2/512 f_{CLK}
		3	f_{PWM}=2/410 f_{CLK}
PWM_REG	User defined PWM amplitude (gradient) for velocity based scaling or regulation loop gradient when *pwm_autoscale*=1.	1 … 15	Results in 0.5 to 7.5 steps for *PWM_SCALE_AUTO* regulator per fullstep
PWM_OFS	User defined PWM amplitude (offset) for velocity based scaling and initialization value for automatic tuning of *PWM_OFFS_AUTO*.	0 … 255	*PWM_OFS*=0 disables linear current scaling based on current setting
PWM_GRAD	User defined PWM amplitude (gradient) for velocity based scaling and initialization value for automatic tuning of *PWM_GRAD_AUTO*.	0 … 255	Reset value can be pre-programmed by OTP
FREEWHEEL	Stand still option when motor current setting is zero (*I_HOLD*=0). Only available with StealthChop enabled. The freewheeling option makes the motor easy movable, while both coil short options realize a passive brake.	0	Normal operation
		1	Freewheeling
		2	Coil short via LS drivers
		3	Coil short cia HS drivers
PWM_SCALE _AUTO	Read back of the actual StealthChop voltage PWM scaling correction as determined by the current regulator. Should regulate to a value close to 0 during tuning procedure.	-255 … 255	(read only) Scaling value becomes frozen when operating in SpreadCycle
PWM_GRAD _AUTO PWM_OFS _AUTO	Allow monitoring of the automatic tuning and determination of initial values for *PWM_OFS* and *PWM_GRAD*.	0 … 255	(read only)
TOFF	General enable for the motor driver, the actual value does not influence StealthChop	0	Driver off
		1 … 15	Driver enabled
TBL	Comparator *blank time*. This time needs to safely cover the switching event and the duration of the ringing on the sense resistor. Choose a setting of 1 or 2 for typical applications. For higher capacitive loads, 3 may be required. Lower settings allow StealthChop to regulate down to lower coil current values.	0	16 t_{CLK}
		1	24 t_{CLK}
		2	32 t_{CLK}
		3	40 t_{CLK}

7 SpreadCycle Chopper

While StealthChop is a voltage mode PWM controlled chopper, SpreadCycle is a cycle-by-cycle current control. Therefore, it can react extremely fast to changes in motor velocity or motor load. SpreadCycle will give better performance in medium to high velocity range for motors and applications which tend to resonance. The currents through both motor coils are controlled using choppers. The choppers work independently of each other. In Figure 7.1 the different chopper phases are shown.

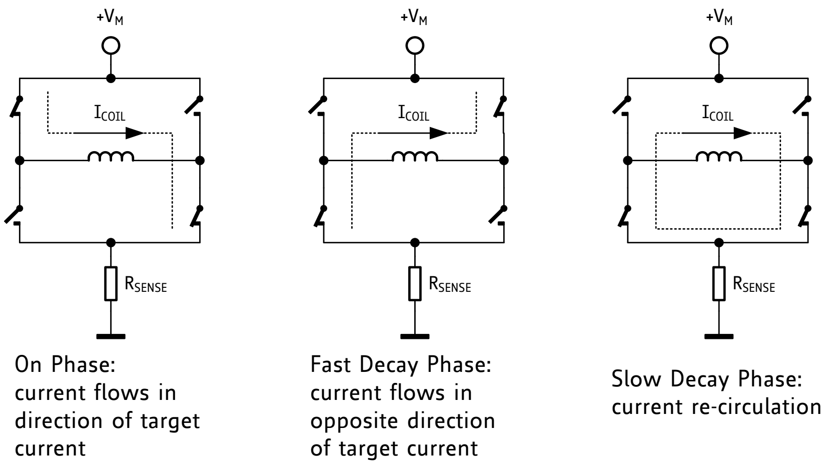

+V_M
I_COIL
R_SENSE
On Phase: current flows in direction of target current
Fast Decay Phase: current flows in opposite direction of target current
Slow Decay Phase: current re-circulation

Figure 7.1 Chopper phases

Although the current could be regulated using only on phases and fast decay phases, insertion of the slow decay phase is important to reduce electrical losses and current ripple in the motor. The duration of the slow decay phase is specified in a control parameter and sets an upper limit on the chopper frequency. The current comparator can measure coil current during phases when the current flows through the sense resistor, but not during the slow decay phase, so the slow decay phase is terminated by a timer. The on phase is terminated by the comparator when the current through the coil reaches the target current. The fast decay phase may be terminated by either the comparator or another timer.

When the coil current is switched, spikes at the sense resistors occur due to charging and discharging parasitic capacitances. During this time, typically one or two microseconds, the current cannot be measured. Blanking is the time when the input to the comparator is masked to block these spikes.

The SpreadCycle chopper mode cycles through four phases: on, slow decay, fast decay, and a second slow decay.

The chopper frequency is an important parameter for a chopped motor driver. A too low frequency might generate audible noise. A higher frequency reduces current ripple in the motor, but with a too high frequency magnetic losses may rise. Also power dissipation in the driver rises with increasing frequency due to the increased influence of switching slopes causing dynamic dissipation. Therefore, a compromise needs to be found. Most motors are optimally working in a frequency range of 16 kHz to 30 kHz. The chopper frequency is influenced by a number of parameter settings as well as by the motor inductivity and supply voltage.

Hint
A chopper frequency in the range of 16 kHz to 30 kHz gives a good result for most motors when using SpreadCycle. A higher frequency leads to increased switching losses.

7.1 SpreadCycle Settings

UART OTP

The SpreadCycle (patented) chopper algorithm is a precise and simple to use chopper mode which automatically determines the optimum length for the fast-decay phase. The SpreadCycle will provide superior microstepping quality even with default settings. Several parameters are available to optimize the chopper to the application.

Each chopper cycle is comprised of an on phase, a slow decay phase, a fast decay phase and a second slow decay phase (see Figure 7.3). The two slow decay phases and the two blank times per chopper cycle put an upper limit to the chopper frequency. The slow decay phases typically make up for about 30%-70% of the chopper cycle in standstill and are important for low motor and driver power dissipation.

Calculation of a starting value for the slow decay time *TOFF*:

EXAMPLE:

Target Chopper frequency: 25kHz.
Assumption: Two slow decay cycles make up for 50% of overall chopper cycle time

$$t_{OFF} = \frac{1}{25kHz} * \frac{50}{100} * \frac{1}{2} = 10\mu s$$

For the *TOFF* setting this means:

$$TOFF = (t_{OFF} * f_{CLK} - 24)/32$$

With 12 MHz clock this gives a setting of TOFF=3.0, i.e. 3.
With 16 MHz clock this gives a setting of TOFF=4.25, i.e. 4 or 5.

The hysteresis start setting forces the driver to introduce a minimum amount of current ripple into the motor coils. The current ripple must be higher than the current ripple which is caused by resistive losses in the motor in order to give best microstepping results. This will allow the chopper to precisely regulate the current both for rising and for falling target current. The time required to introduce the current ripple into the motor coil also reduces the chopper frequency. Therefore, a higher hysteresis setting will lead to a lower chopper frequency. The motor inductance limits the ability of the chopper to follow a changing motor current. Further the duration of the on phase and the fast decay must be longer than the blanking time, because the current comparator is disabled during blanking.

It is easiest to find the best setting by starting from a low hysteresis setting (e.g. *HSTRT*=0, *HEND*=0) and increasing *HSTRT*, until the motor runs smoothly at low velocity settings. This can best be checked when measuring the motor current either with a current probe or by probing the sense resistor voltages (see Figure 7.2). Checking the sine wave shape near zero transition will show a small ledge between both half waves in case the hysteresis setting is too small. At medium velocities (i.e. 100 to 400 fullsteps per second), a too low hysteresis setting will lead to increased humming and vibration of the motor.

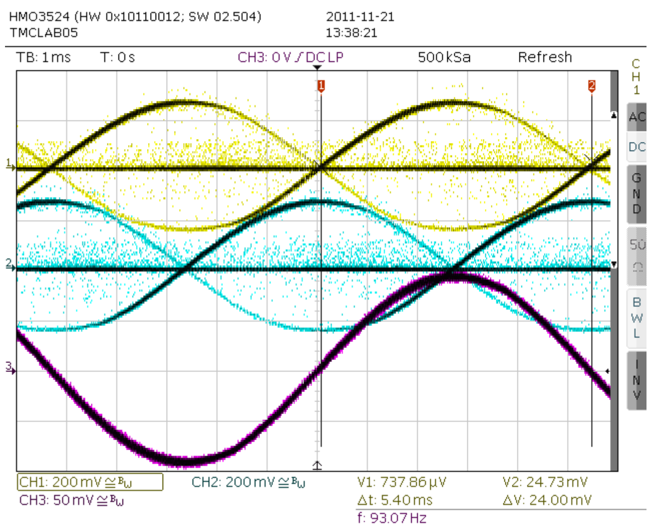


Figure 7.2 No ledges in current wave with sufficient hysteresis (magenta: current A, yellow & blue: sense resistor voltages A and B)

A too high hysteresis setting will lead to reduced chopper frequency and increased chopper noise but will not yield any benefit for the wave shape.

Quick Start
For a quick start, see the Quick Configuration Guide in chapter 14.
For detail procedure see Application Note AN001 - *Parameterization of SpreadCycle*

As experiments show, the setting is quite independent of the motor, because higher current motors typically also have a lower coil resistance. Therefore choosing a low to medium default value for the hysteresis (for example, effective hysteresis = 4) normally fits most applications. The setting can be optimized by experimenting with the motor: A too low setting will result in reduced microstep accuracy, while a too high setting will lead to more chopper noise and motor power dissipation. When measuring the sense resistor voltage in motor standstill at a medium coil current with an oscilloscope, a too low setting shows a fast decay phase not longer than the blanking time. When the fast decay time becomes slightly longer than the blanking time, the setting is optimum. You can reduce the off-time setting, if this is hard to reach.

The hysteresis principle could in some cases lead to the chopper frequency becoming too low, e.g. when the coil resistance is high when compared to the supply voltage. This is avoided by splitting the hysteresis setting into a start setting (*HSTRT+HEND*) and an end setting (*HEND*). An automatic hysteresis decrementer (HDEC) interpolates between both settings, by decrementing the hysteresis value stepwise each 16 system clocks. At the beginning of each chopper cycle, the hysteresis begins with a value which is the sum of the start and the end values (*HSTRT+HEND*), and decrements during the cycle, until either the chopper cycle ends or the hysteresis end value (*HEND*) is reached. This way, the chopper frequency is stabilized at high amplitudes and low supply voltage situations, if the frequency gets too low. This avoids the frequency reaching the audible range.

Hint
Highest motor velocities sometimes benefit from setting *TOFF* to 1 or 2 and a short *TBL* of 1 or 0.

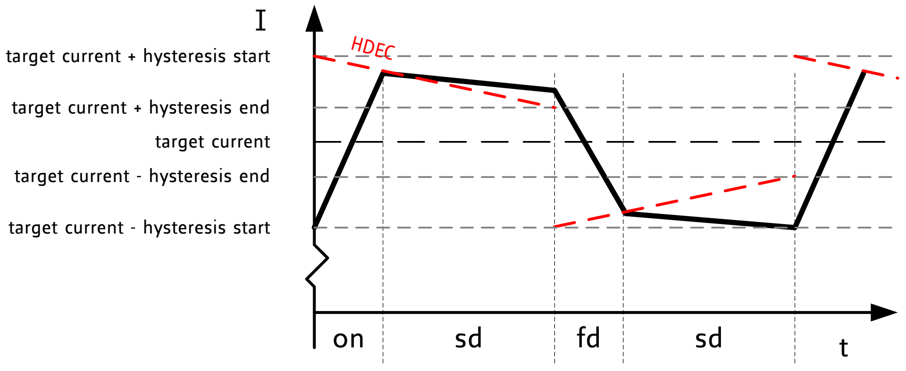

I
target current + hysteresis start
target current + hysteresis end
target current
target current - hysteresis end
target current - hysteresis start
HDEC
on
sd
fd
sd
t

Figure 7.3 SpreadCycle chopper scheme showing coil current during a chopper cycle

These parameters control SpreadCycle mode:

Parameter	Description	Setting	Comment
TOFF	Sets the slow decay time (*off time*). This setting also limits the maximum chopper frequency. For operation with StealthChop, this parameter is not used, but it is required to enable the motor. In case of operation with StealthChop only, any setting is OK. Setting this parameter to zero completely disables all driver transistors and the motor can free-wheel.	0	chopper off
		1…15	off time setting N_{CLK}= 24 + 32**TOFF* (1 will work with minimum blank time of 24 clocks)
TBL	Comparator *blank time*. This time needs to safely cover the switching event and the duration of the ringing on the sense resistor. For most applications, a setting of 1 or 2 is good. For highly capacitive loads, a setting of 2 or 3 will be required.	0	16 t_{CLK}
		1	24 t_{CLK}
		2	32 t_{CLK}
		3	40 t_{CLK}
HSTRT	*Hysteresis start* setting. This value is an offset from the hysteresis end value *HEND*.	0…7	*HSTRT*=1…8 This value adds to HEND.
HEND	*Hysteresis end* setting. Sets the hysteresis end value after a number of decrements. The sum *HSTRT*+*HEND* must be ≤16. At a current setting of max. 30 (amplitude reduced to 240), the sum is not limited.	0…2	-3…-1: negative HEND
		3	0: zero HEND
		4…15	1…12: positive HEND

Even at HSTRT=0 and HEND=0, the TMC22xx sets a minimum hysteresis via analog circuitry.

EXAMPLE:

A hysteresis of 4 has been chosen. You might decide to not use hysteresis decrement. In this case set:

HEND=6	(sets an effective end value of 6-3=3)
HSTRT=0	(sets minimum hysteresis, i.e. 1: 3+1=4)

In order to take advantage of the variable hysteresis, we can set most of the value to the HSTRT, i.e. 4, and the remaining 1 to hysteresis end. The resulting configuration register values are as follows:

HEND=0	(sets an effective end value of -3)
HSTRT=6	(sets an effective start value of hysteresis end +7: 7-3=4)

8 Selecting Sense Resistors

Set the desired maximum motor current by selecting an appropriate value for the sense resistor. The following table shows the RMS current values which can be reached using standard resistors and motor types fitting without additional motor current scaling.

CHOICE OF R_{SENSE} AND RESULTING MAX. MOTOR CURRENT

R_{SENSE} [Ω]	**RMS current [A]** VREF=2.5V (or open), *IRUN*=31, *vsense*=0 (standard)	**Fitting motor type** (examples)
1.00	0.22	
0.82	0.27	
0.75	0.29	300mA motor
0.68	0.32	400mA motor
0.50	0.43	
470m	0.46	500mA motor
390m	0.55	600mA motor
330m	0.64	700mA motor
270m	0.77	800mA motor
220m	0.92	1A motor
180m	1.09	1.2A motor
150m	1.28	
120m	1.53*)	
100m	1.77*)	1.5A motor

*) Value exceeds upper current rating, scaling down required, e.g. by reduced VREF.

Sense resistors should be carefully selected. The full motor current flows through the sense resistors. Due to chopper operation the sense resistors see pulsed current from the MOSFET bridges. Therefore, a low-inductance type such as film or composition resistors is required to prevent voltage spikes causing ringing on the sense voltage inputs leading to unstable measurement results. Also, a low-inductance, low-resistance PCB layout is essential. Any common GND path for the two sense resistors must be avoided, because this would lead to coupling between the two current sense signals. A massive ground plane is best. Please also refer to layout considerations in chapter 19.

The sense resistor needs to be able to conduct the peak motor coil current in motor standstill conditions, unless standby power is reduced. Under normal conditions, the sense resistor conducts less than the coil RMS current, because no current flows through the sense resistor during the slow decay phases. A 0.5W type is sufficient for most applications up to 1.2A RMS current.

Attention
Be sure to use a symmetrical sense resistor layout and short and straight sense resistor traces of identical length. Well matching sense resistors ensure best performance.
A compact layout with massive ground plane is best to avoid parasitic resistance effects.

9 Motor Current Control

The basic motor current is set by the resistance of the sense resistors. Several possibilities allow scaling down motor current, e.g. to adapt for different motors, or to reduce motor current in standstill or low load situations.

METHODS FOR SCALING MOTOR CURRENT

Method	Parameters	Range	Primary Use
Pin VREF voltage (chapter 9.1)	VREF input scales *IRUN* and *IHOLD*. Can be disabled by *GCONF.i_scale_analog*	2.5V: 100% … 0.5V: 20% >2.5V or open: 100% <0.5V: not recommended	- Fine tuning of motor current to fit the motor type - Manual tuning via poti - Delayed or soft power-up - Standstill current reduction (preferred only with SpreadCycle)
Pin ENN	Disable / enable driver stage	0: Motor enable 1: Motor disable	- Disable motor to allow freewheeling
Pin PDN_UART	Disable / enable standstill current reduction to *IHOLD*	0: Standstill current reduction enabled. 1: Disable	- Enable current reduction to reduce heat up in stand still
OTP memory	*OTP_IHOLD*, *OTP_IHOLDDELAY*	9% to 78% standby current. Reduction in about 300ms to 2.5s	- Program current reduction to fit application for highest efficiency and lowest heat up
OTP memory	*otp_internalRsense*	0: Use sense resistors 1: Internal resistors	- Save two sense resistors on BOM, set current by single inexpensive 0603 resistor.
UART interface	*IHOLD_IRUN* *TPOWERDOWN* *OTP*	*IRUN*, *IHOLD*: 1/32 to 32/32 of full scale current.	- Fine programming of run and hold (stand still) current - Change *IRUN* for situation specific motor current - Set OTP options
UART interface	*CHOPCONF.vsense* flag	0: Normal, most robust 1: Reduced voltage level	- Set *vsense* for half power dissipation in sense resistor to use smaller 0.25W resistors.

Select the sense resistor to deliver enough current for the motor at full current scale (VREF=2.5V). This is the default current scaling (*IRUN* = 31).

STANDALONE MODE RMS RUN CURRENT CALCULATION:

$$I_{RMS} = \frac{325mV}{R_{SENSE} + 30m\Omega} * \frac{1}{\sqrt{2}} * \frac{V_{VREF}}{2.5V}$$

IRUN and *IHOLD* allow for scaling of the actual current scale (*CS*) from 1/32 to 32/32 when using UART interface, or via automatic standstill current reduction:

RMS CURRENT CALCULATION WITH UART CONTROL OPTIONS OR HOLD CURRENT SETTING:

$$I_{RMS} = \frac{CS + 1}{32} * \frac{V_{FS}}{R_{SENSE} + 20m\Omega} * \frac{1}{\sqrt{2}}$$

CS is the current scale setting as set by the *IHOLD* and *IRUN*.
V_{FS} is the full scale voltage as determined by *vsense* control bit (please refer to electrical characteristics, V_{SRTL} and V_{SRTH}). Default is 325mV.

With analog scaling of V_{FS} (*I_scale_analog*=1, default), the resulting voltage V_{FS}' is calculated by:

$$V'_{FS} = V_{FS} * \frac{V_{VREF}}{2.5V}$$

with V_{VREF} the voltage on pin VREF in the range 0V to $V_{5VOUT}/2$

Hint
For best precision of current setting, measure and fine tune the current in the application.

Parameters for Motor Current Control			
Parameter	**Description**	**Setting**	**Comment**
IRUN	Current scale when motor is running. Scales coil current values as taken from the internal sine wave table. For high precision motor operation, work with a current scaling factor in the range 16 to 31, because scaling down the current values reduces the effective microstep resolution by making microsteps coarser.	0 … 31	scaling factor 1/32, 2/32, … 32/32 *IRUN* is full scale (setting 31) in standalone mode.
IHOLD	Identical to *IRUN*, but for motor in stand still.		
IHOLD DELAY	Allows smooth current reduction from run current to hold current. *IHOLDDELAY* controls the number of clock cycles for motor power down after *TPOWERDOWN* in increments of 2^18 clocks: 0=instant power down, 1..15: Current reduction delay per current step in multiple of 2^18 clocks. *Example:* When using *IRUN*=31 and *IHOLD*=16, 15 current steps are required for hold current reduction. A *IHOLDDELAY* setting of 4 thus results in a power down time of 4*15*2^18 clock cycles, i.e. roughly one second at 16MHz clock frequency.	0	instant *IHOLD*
		1 … 15	$1*2^{18}$ … $15*2^{18}$ clocks per current decrement
TPOWER DOWN	Sets the delay time from stand still (*stst*) detection to motor current power down. Time range is about 0 to 5.6 seconds.	0 … 255	*0…((2^8)-1)* * 2^18 t_{CLK} A minimum setting of 2 is required to allow automatic tuning of *PWM_OFFS_AUTO*
vsense	Allows control of the sense resistor *voltage range* for full scale current. The low voltage range allows a reduction of sense resistor power dissipation.	0	V_{FS} = 0.32 V
		1	V_{FS} = 0.18 V

9.1 Analog Current Scaling VREF

When a high flexibility of the output current scaling is desired, the analog input of the driver can be used for current control, rather than choosing a different set of sense resistors or scaling down the run current via the interface using *IRUN* or *IHOLD* parameters. This way, a simple voltage divider adapts a board to different motors.

VREF Scales the Motor Current

The TMC22xx provides an internal reference voltage for current control, directly derived from the 5VOUT supply output. Alternatively, an external reference voltage can be used. This reference voltage becomes scaled down for the chopper comparators. The chopper comparators compare the voltages on BRA and BRB to the scaled reference voltage for current regulation. When *I_scale_analog* in *GCONF* is enabled (default), the external voltage on VREF is amplified and filtered and becomes used as reference voltage. A voltage of 2.5V (or any voltage between 2.5V and 5V) gives the same current scaling as the internal reference voltage. A voltage between 0V and 2.5V linearly scales the current between 0 and the current scaling defined by the sense resistor setting. It is not advised to work with reference voltages below about 0.5V to 1V for full scale, because relative analog noise caused by digital circuitry and power supply ripple has an increased impact on the chopper precision at low VREF voltages. For best precision, choose the sense resistors in a way that the desired maximum current is reached with VREF in the range 2V to 2.4V. Be sure to optimize the chopper settings for the normal run current of the motor.

DRIVING VREF

The easiest way to provide a voltage to VREF is to use a voltage divider from a stable supply voltage or a microcontroller's DAC output. A PWM signal also allows current control. The PWM becomes transformed to an analog voltage using an additional R/C low-pass at the VREF pin. The PWM duty cycle controls the analog voltage. Choose the R and C values to form a low pass with a corner frequency of several milliseconds while using PWM frequencies well above 10 kHz. VREF additionally provides an internal low-pass filter with 3.5kHz bandwidth.

Hint
Using a low reference voltage (e.g. below 1V), for adaptation of a high current driver to a low current motor will lead to reduced analog performance. Adapt the sense resistors to fit the desired motor current for the best result.

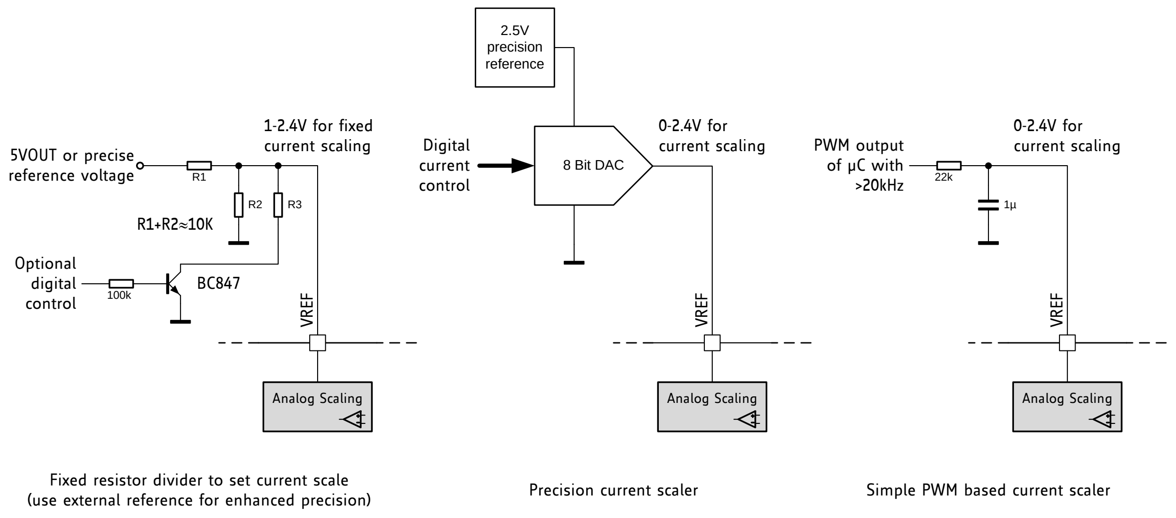

2.5V precision reference
5VOUT or precise reference voltage
R1
1-2.4V for fixed current scaling
R2
R3
R1+R2≈10K
Optional digital control
100k
BC847
VREF
Analog Scaling
Fixed resistor divider to set current scale (use external reference for enhanced precision)
Digital current control
8 Bit DAC
0-2.4V for current scaling
VREF
Analog Scaling
Precision current scaler
PWM output of µC with >20kHz
22k
1µ
0-2.4V for current scaling
VREF
Analog Scaling
Simple PWM based current scaler

Figure 9.1 Scaling the motor current using the analog input

10 Internal Sense Resistors

The TMC22xx provides the option to eliminate external sense resistors. In this mode the external sense resistors become omitted (shorted) and the internal on-resistance of the power MOSFETs is used for current measurement (see chapter 3.2). As MOSFETs are both, temperature dependent and subject to production stray, a tiny external resistor connected from +5VOUT to VREF provides a precise absolute current reference. This resistor converts the 5V voltage into a reference current. Be sure to directly attach BRA and BRB pins to GND in this mode near the IC package. The mode is enabled by setting *internal_Rsense* in *GCONF* (OTP option).

Comparing Internal Sense Resistors vs. Sense Resistors

Item	Internal Sense Resistors	External Sense Resistors
Ease of use	Need to set OTP parameter first	(+) Default
Cost	(+) Save cost for sense resistors	
Current precision	Slightly reduced	(+) Good
Current Range Recommended	200mA RMS to 1.2A RMS	50mA to 1.4A RMS
Recommended chopper	StealthChop, SpreadCycle shows slightly reduced performance at >1A	StealthChop or SpreadCycle

While the RDSon based measurements bring benefits concerning cost and size of the driver, it gives slightly less precise coil current regulation when compared to external sense resistors. The internal sense resistors have a certain temperature dependence, which is automatically compensated by the driver IC. However, for high current motors, a temperature gradient between the ICs internal sense resistors and the compensation circuit will lead to an initial current overshoot of some 10% during driver IC heat up. While this phenomenon shows for roughly a second, it might even be beneficial to enable increased torque during initial motor acceleration.

Principle of Operation

A reference current into the VREF pin is used as reference for the motor current. In order to realize a certain current, a single resistor (R_{REF}) can be connected between 5VOUT and VREF (pls. refer the table for the choice of the resistor). VREF input resistance is about 1kOhm. The resulting current into VREF is amplified 3000 times. Thus, a current of 0.5mA yields a motor current of 1.5A peak. For calculation of the reference resistor, the internal resistance of VREF needs to be considered additionally.

Choice of R_{REF} for Operation without Sense Resistors

R_{REF} [Ω]	Peak current [A] (*CS*=31, *vsense*=0)	RMS current [A] (*CS*=31, *vsense*=0)
6k8	1.92	1.35
7k5	1.76	1.24
8k2	1.63	1.15
9k1	1.49	1.05
10k	1.36	0.96
12k	1.15	0.81
15k	0.94	0.66
18k	0.79	0.55
22k	0.65	0.45
27k	0.60	0.42
33k	0.54	0.38

vsense=1 allows a lower peak current setting of about 55% of the value yielded with *vsense*=0 (as specified by V_{SRTH} / V_{SRTL}).

In RDSon measurement mode, connect the BRA and BRB pins to GND using the shortest possible path (i.e. lowest possible PCB resistance). RDSon based measurement gives best results when combined with StealthChop. When using SpreadCycle with RDSon based current measurement, slightly asymmetric current measurement for positive currents (on phase) and negative currents (fast decay phase) may result in chopper noise. This especially occurs at high die temperature and increased motor current.

Note
The absolute current levels achieved with RDSon based current sensing may depend on PCB layout exactly like with external sense resistors, because trace resistance on BR pins will add to the effective sense resistance. Therefore we recommend to measure and calibrate the current setting within the application.

Thumb rule
RDSon based current sensing works best for motors with up to 1A RMS current. The best results are yielded with StealthChop operation in combination with RDSon based current sensing.
For most precise current control and for best results with SpreadCycle, it is recommended to use external 1% sense resistors rather than RDSon based current control.

11 STEP/DIR Interface

The STEP and DIR inputs provide a simple, standard interface compatible with many existing motion controllers. The MicroPlyer step pulse interpolator brings the smooth motor operation of high-resolution microstepping to applications originally designed for coarser stepping.

11.1 Timing

Figure 11.1 shows the timing parameters for the STEP and DIR signals, and the table below gives their specifications. Only rising edges are active. STEP and DIR are sampled and synchronized to the system clock. An internal analog filter removes glitches on the signals, such as those caused by long PCB traces. If the signal source is far from the chip, and especially if the signals are carried on cables, the signals should be filtered or differentially transmitted.

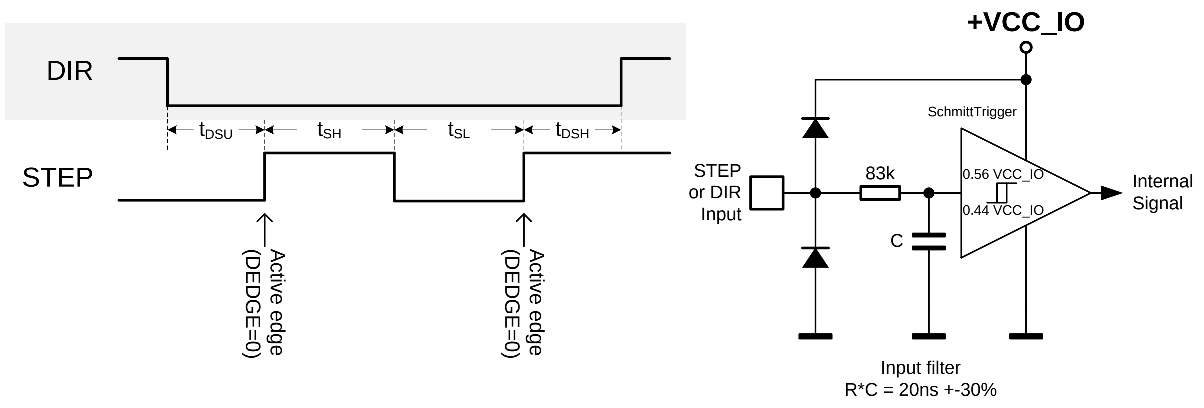


Figure 11.1 STEP and DIR timing, Input pin filter

STEP and DIR interface timing	AC-Characteristics clock period is t_{CLK}					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
step frequency (at maximum microstep resolution)	f_{STEP}				½ f_{CLK}	
fullstep frequency	f_{FS}				$f_{CLK}/512$	
STEP input minimum low time	t_{SL}		max(t_{FILTSD}, t_{CLK}+20)	100		ns
STEP input minimum high time	t_{SH}		max(t_{FILTSD}, t_{CLK}+20)	100		ns
DIR to STEP setup time	t_{DSU}		20			ns
DIR after STEP hold time	t_{DSH}		20			ns
STEP and DIR spike filtering time *)	t_{FILTSD}	rising and falling edge	13	20	30	ns
STEP and DIR sampling relative to rising CLK input	$t_{SDCLKHI}$	before rising edge of CLK input		t_{FILTSD}		ns

*) These values are valid with full input logic level swing, only. Asymmetric logic levels will increase filtering delay t_{FILTSD}, due to an internal input RC filter.

11.2 Changing Resolution

The TMC22xx includes an internal microstep table with 1024 sine wave entries to generate sinusoidal motor coil currents. These 1024 entries correspond to one electrical revolution or four fullsteps. The microstep resolution setting determines the step width taken within the table. Depending on the DIR input, the microstep counter is increased (DIR=0) or decreased (DIR=1) with each STEP pulse by the step width. The microstep resolution determines the increment respectively the decrement. At maximum resolution, the sequencer advances one step for each step pulse. At half resolution, it advances two steps. Increment is up to 256 steps for fullstepping. The sequencer has special provision to allow seamless switching between different microstep rates at any time. When switching to a lower microstep resolution, it calculates the nearest step within the target resolution and reads the current vector at that position. This behavior especially is important for low resolutions like fullstep and halfstep, because any failure in the step sequence would lead to asymmetrical run when comparing a motor running clockwise and counterclockwise.

EXAMPLES:

Fullstep:	Cycles through table positions: 128, 384, 640 and 896 (45°, 135°, 225° and 315° electrical position, both coils on at identical current). The coil current in each position corresponds to the RMS-Value (0.71 * amplitude). Step size is 256 (90° electrical)
Half step:	The first table position is 64 (22.5° electrical), Step size is 128 (45° steps)
Quarter step:	The first table position is 32 (90°/8=11.25° electrical), Step size is 64 (22.5° steps)

This way equidistant steps result and they are identical in both rotation directions. Some older drivers also use zero current (table entry 0, 0°) as well as full current (90°) within the step tables. This kind of stepping is avoided because it provides less torque and has a worse power dissipation in driver and motor.

Step position	table position	current coil A	current coil B
Half step 0	64	38.3%	92.4%
Full step 0	128	70.7%	70.7%
Half step 1	192	92.4%	38.3%
Half step 2	320	92.4%	-38.3%
Full step 1	384	70.7%	-70.7%
Half step 3	448	38.3%	-92.4%
Half step 4	576	-38.3%	-92.4%
Full step 2	640	-70.7%	-70.7%
Half step 5	704	-92.4%	-38.3%
Half step 6	832	-92.4%	38.3%
Full step 3	896	-70.7%	70.7%
Half step 7	960	-38.3%	92.4%

See chapter 3.4 for resolution settings available in stand-alone mode.

11.3 MicroPlyer Step Interpolator and Stand Still Detection

For each active edge on STEP, MicroPlyer produces microsteps at 256x resolution, as shown in Figure 11.2. It interpolates the time in between of two step impulses at the step input based on the last step interval. This way, from 2 microsteps (128 microstep to 256 microstep interpolation) up to 256 microsteps (full step input to 256 microsteps) are driven for a single step pulse.

The step rate for the interpolated 2 to 256 microsteps is determined by measuring the time interval of the previous step period and dividing it into up to 256 equal parts. The maximum time between two microsteps corresponds to 2^{20} (roughly one million system clock cycles), for an even distribution of 256 microsteps. At 12 MHz system clock frequency, this results in a minimum step input frequency of roughly 12 Hz for MicroPlyer operation. A lower step rate causes a standstill event to be detected. At that frequency, microsteps occur at a rate of (system clock frequency)/2^{16} ~ 256 Hz. When a stand still is detected, the driver automatically begins standby current reduction if selected by pin PDN.

Attention
MicroPlyer only works perfectly with a jitter-free STEP frequency.

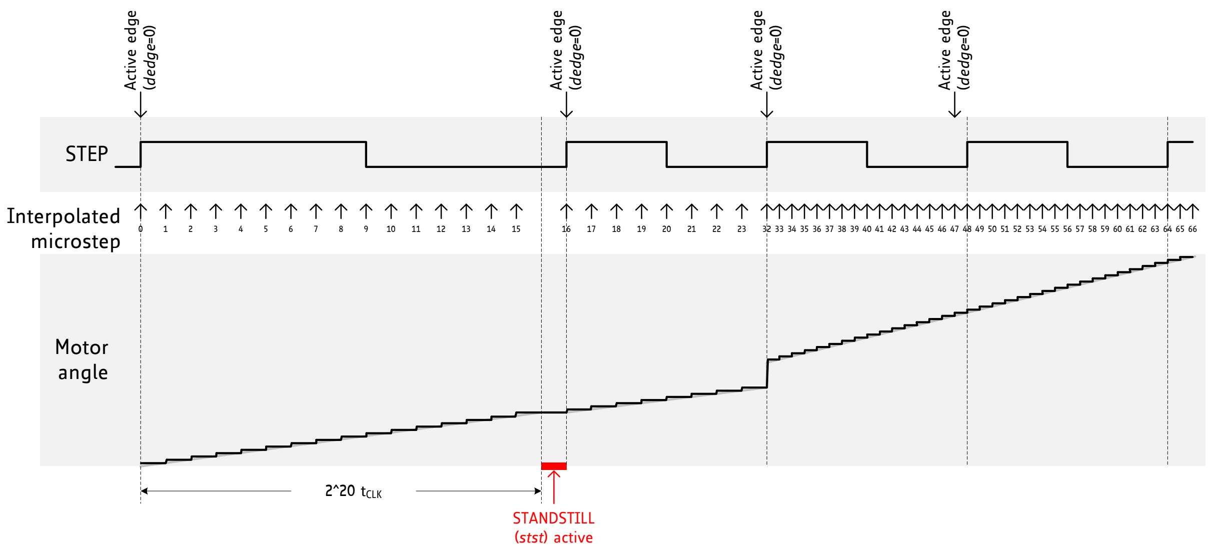


Figure 11.2 MicroPlyer microstep interpolation with rising STEP frequency (Example: 16 to 256)

In Figure 11.2, the first STEP cycle is long enough to set the *stst* bit standstill. Detection of standstill will enable the standby current reduction. This bit is cleared on the next STEP active edge. Then, the external STEP frequency increases. After one cycle at the higher rate MicroPlyer adapts the interpolated microstep rate to the higher frequency. During the last cycle at the slower rate, MicroPlyer did not generate all 16 microsteps, so there is a small jump in motor angle between the first and second cycles at the higher rate.

11.4 Index Output

An active INDEX output signals that the sine curve of motor coil A is at its positive zero transition. This correlates to the zero point of the microstep sequence. Usually, the cosine curve of coil B is at its maximum at the same time. Thus the index signal is active once within each electrical period, and corresponds to a defined position of the motor within a sequence of four fullsteps. The INDEX output this way allows the detection of a certain microstep pattern, and thus helps to detect a position with more precision than a stop switch can do.

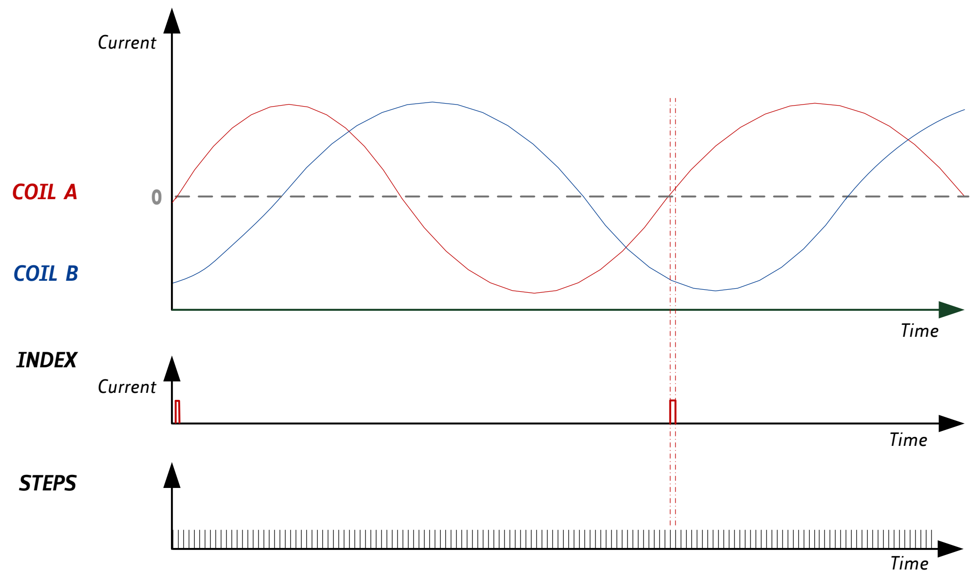


Figure 11.3 Index signal at positive zero transition of the coil A sine curve

Hint
The index output allows precise detection of the microstep position within one electrical wave, i.e. within a range of four fullsteps. With this, homing accuracy and reproducibility can be enhanced to microstep accuracy, even when using an inexpensive home switch.

12 Internal Step Pulse Generator

The TMC22xx family integrates a high resolution step pulse generator, allowing motor motion via the UART interface. However, no velocity ramping is provided. Ramping is not required, if the target motion velocity is smaller than the start & stop frequency of the motor. For higher velocities, ramp up the frequency in small steps to accelerate the motor, and ramp down again to decelerate the motor. Figure 12.1 shows an example motion profile ramping up the motion velocity in discrete steps. Choose the ramp velocity steps considerably smaller than the maximum start velocity of the motor, because motor torque drops at higher velocity, and motor load at higher velocity typically increases.

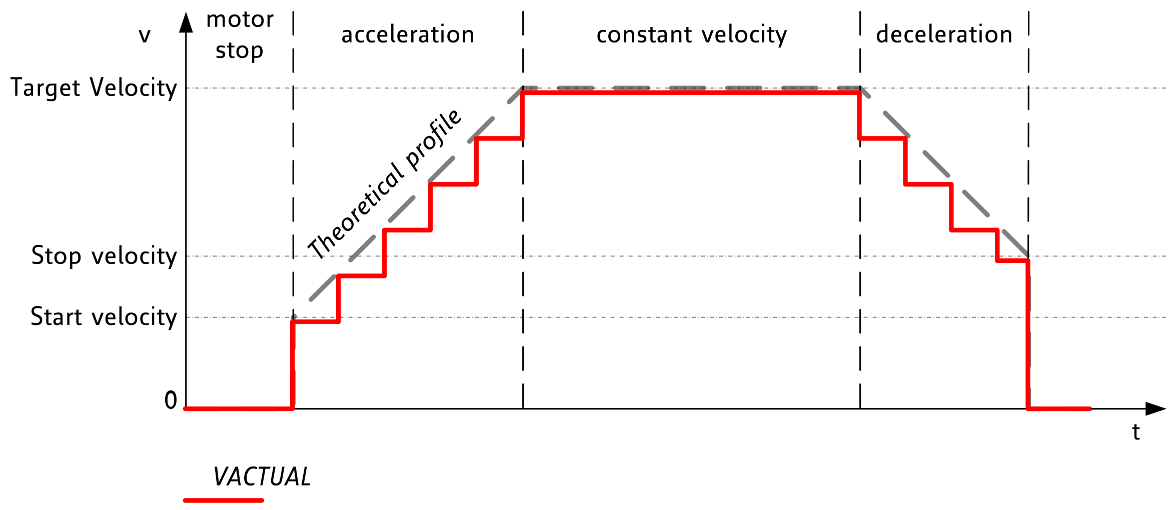


Figure 12.1 Software generated motion profile

PARAMETER VS. UNITS		
Parameter / Symbol	**Unit**	**calculation / description / comment**
f_{CLK}[Hz]	[Hz]	clock frequency of the TMC22xx in [Hz]

µstep velocity v[Hz]	µsteps / s	v[Hz] = *VACTUAL*[22xx] * (f_{CLK}[Hz]/2 / 2^23)
		With internal oscillator:
		v[Hz] = *VACTUAL*[22xx] * 0.715Hz

USC microstep count	counts	microstep resolution in number of microsteps (i.e. the number of microsteps between two fullsteps – normally 256)
rotations per second v[rps]	rotations / s	v[rps] = v[Hz] / USC / FSC FSC: motor fullsteps per rotation, e.g. 200
TSTEP, *TPWMTHRS*	-	*TSTEP* = f_{CLK} / f_{STEP} The time reference for velocity threshold is referred to the actual microstep frequency of the step input respectively velocity v[Hz].
VACTUAL	Two's complement signed internal velocity	*VACTUAL*[22xx] = (f_{CLK}[Hz]/2 / 2^23) / v[Hz] With internal oscillator: *VACTUAL*[22xx] = 0.715Hz / v[Hz]

Hint
To monitor internal step pulse execution, program the INDEX output to provide step pulses (*GCONF.index_step*). It toggles upon each step. Use a timer input on your CPU to count pulses. Alternatively, regularly poll *MSCNT* to grasp steps done in the previous polling interval. It wraps around from 1023 to 0.

13 Driver Diagnostic Flags

The TMC22xx drivers supply a complete set of diagnostic and protection capabilities, like short to GND protection, short to VS protection and undervoltage detection. A detection of an open load condition allows testing if a motor coil connection is interrupted. See the *DRV_STATUS* table for details.

13.1 Temperature Measurement

The driver integrates a four level temperature sensor (pre-warning and thermal shutdown) for diagnostics and for protection of the IC against excess heat. The thresholds can be adapted by UART or OTP programming. Heat is mainly generated by the motor driver stages, and, at increased voltage, by the internal voltage regulator. Most critical situations, where the driver MOSFETs could be overheated, are avoided by the short to GND protection. For many applications, the overtemperature pre-warning will indicate an abnormal operation situation and can be used to initiate user warning or power reduction measures like motor current reduction. The thermal shutdown is just an emergency measure and temperature rising to the shutdown level should be prevented by design.

TEMPERATURE THRESHOLDS

Overtemperature Setting	Comment
143°C (OTPW: 120°C)	Default setting. This setting is safest to switch off the driver stage before the IC can be destroyed by overheating. On a large PCB, the power MOSFETs reach roughly 150°C peak temperature when the temperature detector is triggered with this setting. This is a trip typical point for overtemperature shut down. The overtemperature pre-warning threshold of 120°C gives lots of headroom to react to high driver temperature, e.g. by reducing motor current.
150°C (OTPW: 120°C or 143°C)	Optional setting (OTP or UART). For small PCBs with high thermal resistance between PCB and environment, this setting provides some additional headroom. The small PCB shows less temperature difference between the MOSFETs and the sensor.
157°C (OTPW: 143°C)	Optional setting (UART). For applications, where a stop of the motor cannot be tolerated, this setting provides highest headroom, e.g. at high environment temperature ratings. Using the 143°C overtemperature pre-warning to reduce motor current ensures that the motor is not switched off by the thermal threshold.

Attention
Overtemperature protection cannot in all cases avoid thermal destruction of the IC. In case the rated motor current is exceed, e.g. by operating a motor in StealthChop with wrong parameters, or with automatic tuning parameters not fitting the operating conditions, excess heat generation can quickly heat up the driver before the overtemperature sensor can react. This is due to a delay in heat conduction over the IC die.

After triggering the overtemperature sensor (*ot* flag), the driver remains switched off until the system temperature falls below the pre-warning level (*otpw*) to avoid continuous heating to the shutdown level.

13.2 Short Protection

The TMC22xx power stages are protected against a short circuit condition by an additional measurement of the current flowing through each of the power stage MOSFETs. This is important, as most short circuit conditions result from a motor cable insulation defect, e.g. when touching the conducting parts connected to the system ground. The short detection is protected against spurious triggering, e.g. by ESD discharges, by retrying three times before switching off the motor.

Once a short condition is safely detected, the corresponding driver bridge (A or B) becomes switched off, and the *s2ga* or *s2gb* flag, respectively *s2vsa* or *s2vsb* becomes set. In order to restart the motor, disable and re-enable the driver. Note, that short protection cannot protect the system and the power stages for all possible short events, as a short event is rather undefined and a complex network of external components may be involved. Therefore, short circuits should basically be avoided.

13.3 Open Load Diagnostics

UART

Interrupted cables are a common cause for systems failing, e.g. when connectors are not firmly plugged. The TMC22xx detects open load conditions by checking, if it can reach the desired motor coil current. This way, also undervoltage conditions, high motor velocity settings or short and overtemperature conditions may cause triggering of the open load flag, and inform the user, that motor torque may suffer. In motor stand still, open load cannot always be measured, as the coils might eventually have zero current.

Open load detection is provided for system debugging.
In order to safely detect an interrupted coil connection, read out the open load flags at low or nominal motor velocity operation, only. If possible, use SpreadCycle for testing, as it provides the most accurate test. However, the *ola* and *olb* flags have just informative character and do not cause any action of the driver.

13.4 Diagnostic Output

The diagnostic output DIAG and the index output INDEX provide important status information. An active DIAG output shows that the driver cannot work normally. The INDEX output signals the microstep counter zero position, to allow referencing (homing) a drive to a certain current pattern. The function set of the INDEX output can be modified by UART. Figure 13.1 shows the available signals and control bits.

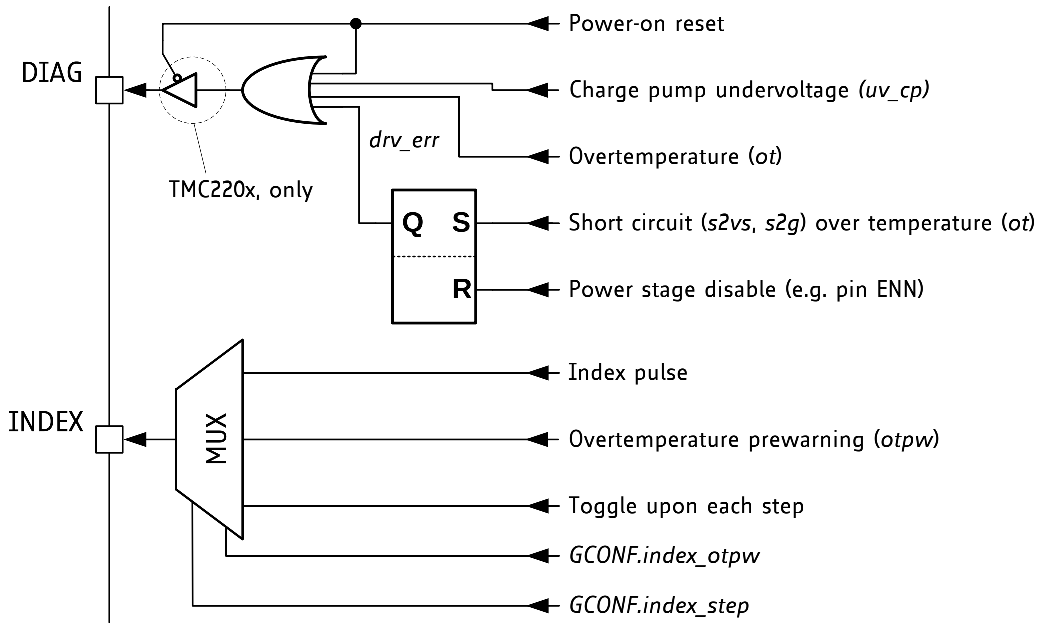

DIAG
TMC220x, only
drv_err
Q
S
R
Power-on reset
Charge pump undervoltage (uv_cp)
Overtemperature (ot)
Short circuit (s2vs, s2g) over temperature (ot)
Power stage disable (e.g. pin ENN)
INDEX
MUX
Index pulse
Overtemperature prewarning (otpw)
Toggle upon each step
GCONF.index_otpw
GCONF.index_step

Figure 13.1 DIAG and INDEX outputs

14 Quick Configuration Guide

This guide is meant as a practical tool to come to a first configuration. Do a minimum set of measurements and decisions for tuning the driver to determine UART settings or OTP parameters. The flow-charts concentrate on the basic function set to make a motor run smoothly. Once the motor runs, you may decide to explore additional features, e.g. freewheeling in more detail. A current probe on one motor coil is a good aid to find the best settings, but it is not a must.

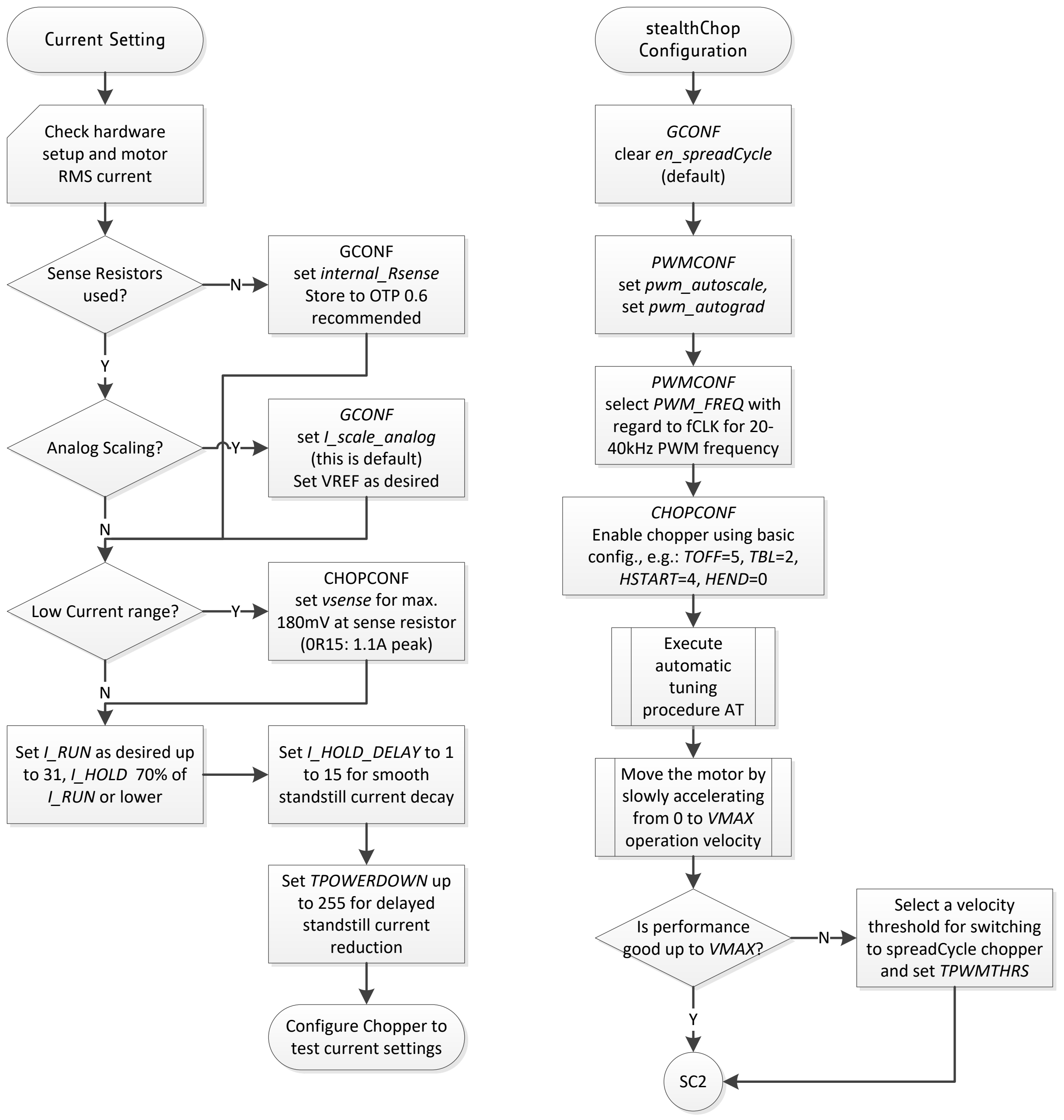

Current Setting
Check hardware setup and motor RMS current
Sense Resistors used?
N
GCONF set internal_Rsense Store to OTP 0.6 recommended
Y
Analog Scaling?
Y
GCONF set I_scale_analog (this is default) Set VREF as desired
N
Low Current range?
Y
CHOPCONF set vsense for max. 180mV at sense resistor (0R15: 1.1A peak)
N
Set I_RUN as desired up to 31, I_HOLD 70% of I_RUN or lower
Set I_HOLD_DELAY to 1 to 15 for smooth standstill current decay
Set TPOWERDOWN up to 255 for delayed standstill current reduction
Configure Chopper to test current settings
stealthChop Configuration
GCONF clear en_spreadCycle (default)
PWMCONF set pwm_autoscale, set pwm_autograd
PWMCONF select PWM_FREQ with regard to fCLK for 20-40kHz PWM frequency
CHOPCONF Enable chopper using basic config., e.g.: TOFF=5, TBL=2, HSTART=4, HEND=0
Execute automatic tuning procedure AT
Move the motor by slowly accelerating from 0 to VMAX operation velocity
Is performance good up to VMAX?
N
Select a velocity threshold for switching to spreadCycle chopper and set TPWMTHRS
Y
SC2

Figure 14.1 Current Setting and first steps with StealthChop

Hint
Use the evaluation board to explore settings and to generate the required configuration datagrams.

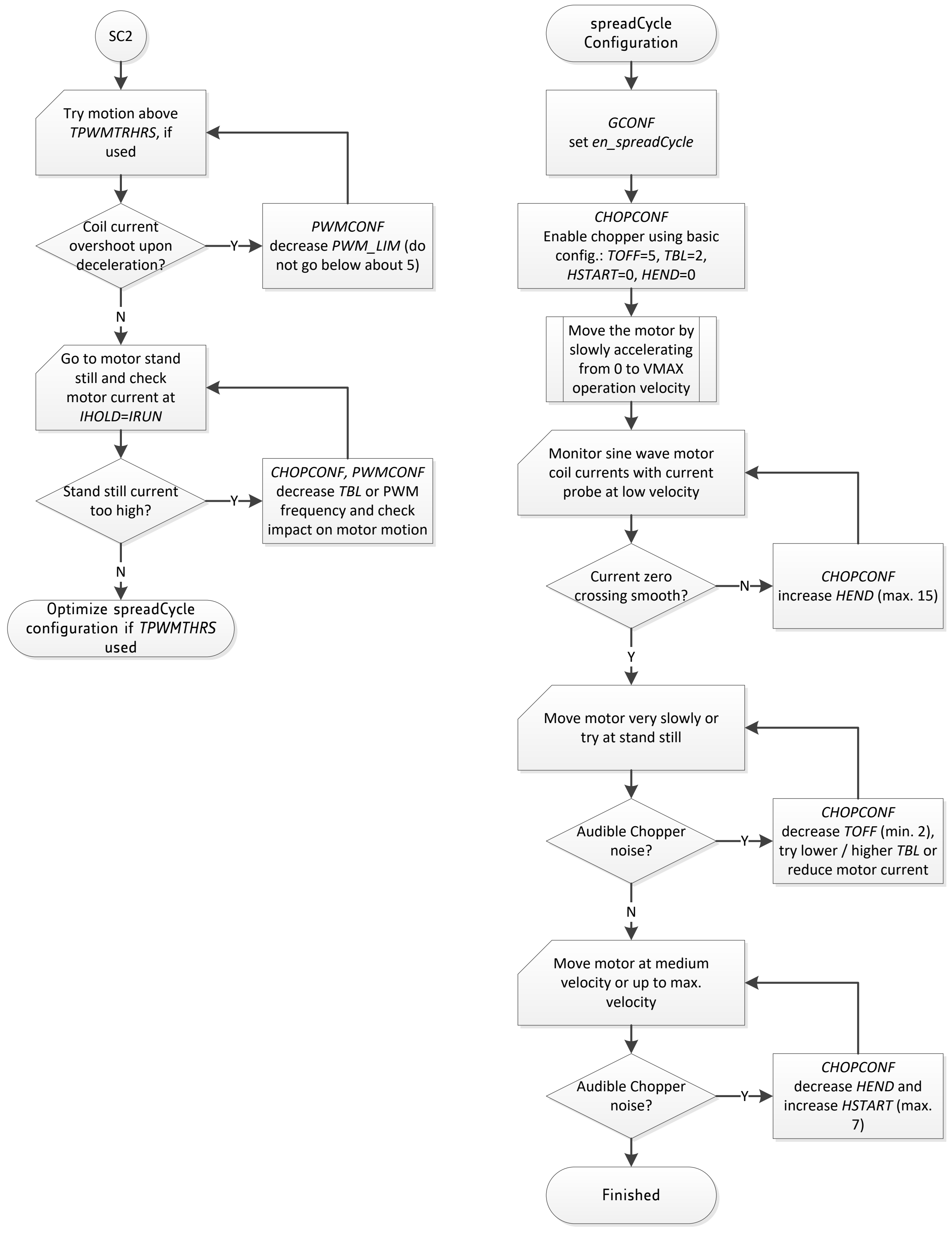

SC2
Try motion above *TPWMTRHRS*, if used
Coil current overshoot upon deceleration?
Y
PWMCONF decrease *PWM_LIM* (do not go below about 5)
N
Go to motor stand still and check motor current at *IHOLD*=*IRUN*
Stand still current too high?
Y
CHOPCONF, PWMCONF decrease *TBL* or PWM frequency and check impact on motor motion
N
Optimize spreadCycle configuration if *TPWMTHRS* used
spreadCycle Configuration
GCONF set *en_spreadCycle*
CHOPCONF Enable chopper using basic config.: *TOFF*=5, *TBL*=2, *HSTART*=0, *HEND*=0
Move the motor by slowly accelerating from 0 to VMAX operation velocity
Monitor sine wave motor coil currents with current probe at low velocity
Current zero crossing smooth?
N
CHOPCONF increase *HEND* (max. 15)
Y
Move motor very slowly or try at stand still
Audible Chopper noise?
Y
CHOPCONF decrease *TOFF* (min. 2), try lower / higher *TBL* or reduce motor current
N
Move motor at medium velocity or up to max. velocity
Audible Chopper noise?
Y
CHOPCONF decrease *HEND* and increase *HSTART* (max. 7)
Finished

Figure 14.2 Tuning StealthChop and SpreadCycle

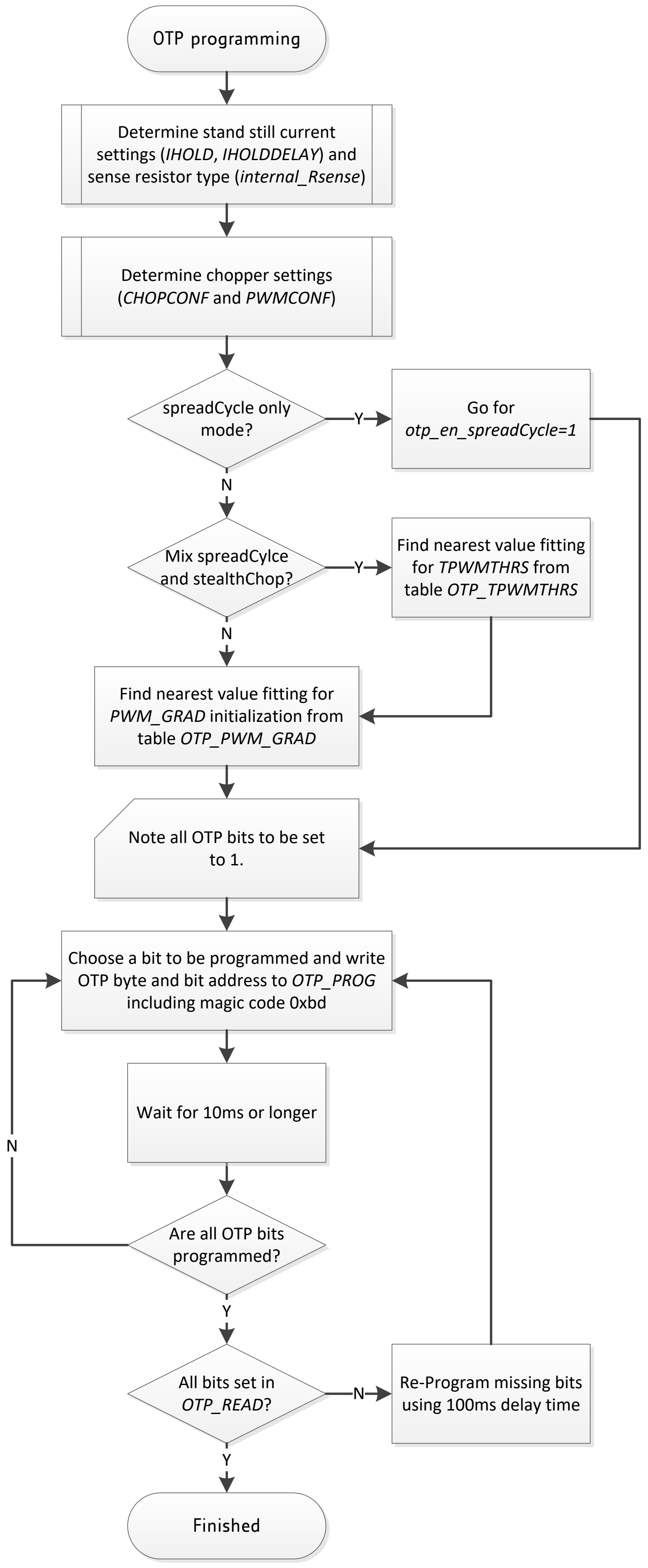

OTP programming
Determine stand still current settings (*IHOLD, IHOLDDELAY*) and sense resistor type (*internal_Rsense*)
Determine chopper settings (*CHOPCONF* and *PWMCONF*)
spreadCycle only mode?
Y
Go for *otp_en_spreadCycle=1*
N
Mix spreadCylce and stealthChop?
Y
Find nearest value fitting for *TPWMTHRS* from table *OTP_TPWMTHRS*
N
Find nearest value fitting for *PWM_GRAD* initialization from table *OTP_PWM_GRAD*
Note all OTP bits to be set to 1.
Choose a bit to be programmed and write OTP byte and bit address to *OTP_PROG* including magic code 0xbd
Wait for 10ms or longer
N
Are all OTP bits programmed?
Y
All bits set in *OTP_READ*?
N
Re-Program missing bits using 100ms delay time
Y
Finished

Figure 14.3 OTP programming

15 External Reset

The chip is loaded with default values during power on via its internal power-on reset. Some of the registers are initialized from the OTP at power up. In order to reset the chip to power on defaults, any of the supply voltages monitored by internal reset circuitry (VS, +5VOUT or VCC_IO) must be cycled. As +5VOUT is the output of the internal voltage regulator, it cannot be cycled via an external source except by cycling VS. It is easiest and safest to cycle VCC_IO in order to completely reset the chip. Also, current consumed from VCC_IO is low and therefore it has simple driving requirements. Due to the input protection diodes not allowing the digital inputs to rise above VCC_IO level, all inputs must be driven low during this reset operation. When this is not possible, an input protection resistor may be used to limit current flowing into the related inputs.

16 Clock Oscillator and Input

The clock is the timing reference for all functions: the chopper frequency, the blank time, the standstill power down timing, and the internal step pulse generator etc. The on-chip clock oscillator is calibrated in order to provide timing precise enough for most operation cases.

Using the Internal Clock

Directly tie the CLK input to GND near to the IC if the internal clock oscillator is to be used. The internal clock frequency is factory-trimmed to 12MHz by OTP programming.

Using an External Clock

When an external clock is available, a frequency of 10 MHz to 16 MHz is recommended for optimum performance. The duty cycle of the clock signal is uncritical, as long as minimum high or low input time for the pin is satisfied (refer to electrical characteristics). Up to 18 MHz can be used, when the clock duty cycle is 50%. Make sure, that the clock source supplies clean CMOS output logic levels and steep slopes when using a high clock frequency. The external clock input is enabled with the first positive polarity seen on the CLK input. Modifying the clock frequency is an easy way to adapt the StealthChop chopper frequency or to synchronize multiple drivers. Working with a very low clock frequency down to 4 MHz can help reducing power consumption and electromagnetic emissions, but it will sacrifice some performance.

Use an external clock source to synchronize multiple drivers, or to get precise motor operation with the internal pulse generator for motion. The external clock frequency selection also allows modifying the power down timing and the chopper frequency.

Hint
Switching off the external clock frequency would stop the chopper and could lead to an overcurrent condition. Therefore, TMC22xx has an internal timeout of 32 internal clocks. In case the external clock fails, it switches back to internal clock.

17 Absolute Maximum Ratings

The maximum ratings may not be exceeded under any circumstances. Operating the circuit at or near more than one maximum rating at a time for extended periods shall be avoided by application design.

Parameter	Symbol	Min	Max	Unit
Supply voltage operating with inductive load	V_{VS}	-0.5	39	V
Supply and bridge voltage max. *)	V_{VMAX}		40	V
I/O supply voltage	V_{VIO}	-0.5	5.5	V
digital supply voltage (when using external supply)	V_{5VOUT}	-0.5	5.5	V
Logic input voltage	V_I	-0.5	V_{VIO}+0.5	V
VREF input voltage (Do not exceed both, VCC_IO and 5VOUT by more than 10%, as this enables a test mode)	V_{VREF}	-0.5	6	V
Maximum current to / from digital pins and analog low voltage I/Os	I_{IO}		+/-10	mA
5V regulator output current (internal plus external load)	I_{5VOUT}		25	mA
5V regulator continuous power dissipation (V_{VM}-5V) * I_{5VOUT}	P_{5VOUT}		0.5	W
Power bridge repetitive output current	I_{Ox}		2.5	A
Junction temperature	T_J	-50	150	°C
Storage temperature	T_{STG}	-55	150	°C
ESD-Protection for interface pins (Human body model, HBM)	V_{ESDAP}		4	kV
ESD-Protection for handling (Human body model, HBM)	V_{ESD}		1	kV

*) Stray inductivity of GND and VS connections will lead to ringing of the supply voltage when driving an inductive load. This ringing results from the fast switching slopes of the driver outputs in combination with reverse recovery of the body diodes of the output driver MOSFETs. Even small trace inductivities as well as stray inductivity of sense resistors can easily generate a few volts of ringing leading to temporary voltage overshoot. This should be considered when working near the maximum voltage.

18 Electrical Characteristics

18.1 Operational Range

Parameter	Symbol	Min	Max	Unit
Junction temperature	T_J	-40	125	°C
Supply voltage (using internal +5V regulator)	V_{VS}	5.5	36	V
Supply voltage (using internal +5V regulator) for OTP programming	V_{VS}	6	36	V
Supply voltage (internal +5V regulator bridged: V_{5VOUT}=V_{VS})	V_{VS}	4.7	5.4	V
I/O supply voltage	V_{VIO}	3.00	5.25	V
VCC voltage when using optional external source (supplies digital logic and charge pump)	V_{VCC}	4.6	5.25	V
RMS motor coil current per coil (value for design guideline)	I_{RMS}		1.2	A
RMS motor coil current per coil with duty cycle limit, e.g. 1s on, 1s standby (value for design guideline)	I_{RMS}		1.4	A
Peak output current per motor coil output (sine wave peak) using external or internal current sensing	I_{Ox}		2.0	A

18.2 DC and Timing Characteristics

DC characteristics contain the spread of values guaranteed within the specified supply voltage range unless otherwise specified. Typical values represent the average value of all parts measured at +25°C. Temperature variation also causes stray to some values. A device with typical values will not leave Min/Max range within the full temperature range.

Power supply current	**DC-Characteristics** $V_{VS} = V_{VSA} = 24.0V$					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
Total supply current, driver disabled I_{VS}	I_S	f_{CLK}=12MHz		7	10	mA
Total supply current, operating, I_{VS}	I_S	f_{CLK}=12MHz, 35kHz chopper, no load		7.5		mA
Supply current, driver disabled, dependency on CLK frequency	I_{VS}	f_{CLK} variable, additional to I_{VS0}		0.3		mA/MHz
Internal current consumption from 5V supply on VCC pin	I_{VCC}	f_{CLK}=12MHz, 35kHz chopper		4.5		mA
IO supply current (typ. at 5V)	I_{VIO}	no load on outputs, inputs at V_{IO} or GND Excludes pullup / pull-down resistors		15	30	µA

Motor driver section	**DC- and Timing-Characteristics** $V_{VS} = 24.0V$					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
RDS_{ON} lowside MOSFET	R_{ONL}	measure at 100mA, 25°C, static state		0.28	0.38	Ω
RDS_{ON} highside MOSFET	R_{ONH}	measure at 100mA, 25°C, static state		0.29	0.39	Ω
slope, MOSFET turning on	t_{SLPON}	measured at 700mA load current (resistive load)	40	80	160	ns
slope, MOSFET turning off	t_{SLPOFF}	measured at 700mA load current (resistive load)	40	80	160	ns
Current sourcing, driver off	I_{OIDLE}	O_{XX} pulled to GND	120	180	250	µA

Charge pump	**DC-Characteristics**					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
Charge pump output voltage	V_{VCP}-V_{VS}	operating, typical f_{chop}<40kHz	4.0	V_{VCC} - 0.3	V_{VCC}	V
Charge pump voltage threshold for undervoltage detection	V_{VCP}-V_{VS}	using internal 5V regulator voltage	3.3	3.6	4.0	V
Charge pump frequency	f_{CP}			1/16 f_{CLKOSC}		

Linear regulator	**DC-Characteristics** $V_{VS} = V_{VSA} = 24.0V$					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
Output voltage	V_{5VOUT}	$I_{5VOUT} = 0mA$ $T_J = 25°C$	4.80	5.0	5.25	V
Output resistance	R_{5VOUT}	Static load		1		Ω
Deviation of output voltage over the full temperature range	$V_{5VOUT(DEV)}$	$I_{5VOUT} = 5mA$ T_J = full range		+/-30	+/-100	mV
Deviation of output voltage over the full supply voltage range	$V_{5VOUT(DEV)}$	$I_{5VOUT} = 5mA$ V_{VS} = variable		+/-15	+/-30	mV / 10V

Clock oscillator and input	**Timing-Characteristics**					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
Clock oscillator frequency (factory calibrated)	f_{CLKOSC}	t_J=-50°C		11.7		MHz
	f_{CLKOSC}	t_J= 25°C	11.5	12.0	12.5	MHz
	f_{CLKOSC}	t_J=150°C		12.1		MHz
External clock frequency (operating)	f_{CLK}		4	10-16	18	MHz
External clock high / low level time	t_{CLK}	CLK driven to 0.1 V_{VIO} / 0.9 V_{VIO}	16			ns
External clock first pulse to trigger switching to external CLK	t_{CLKH} / t_{CLKL}	CLK driven high	30	25		ns
External clock timeout detection in cycles of internal f_{CLKOSC}	$x_{timeout}$	External clock stuck at low or high	32		48	cycles f_{CLKOSC}

Detector levels	**DC-Characteristics**					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
V_{VS} undervoltage threshold for RESET	V_{UV_VS}	V_{VS} rising	3.5	4.2	4.6	V
V_{5VOUT} undervoltage threshold for RESET	V_{UV_5VOUT}	V_{5VOUT} rising		3.5		V
V_{VCC_IO} undervoltage threshold for RESET	V_{UV_VIO}	V_{VCC_IO} rising (delay typ. 10µs)	2.1	2.55	3.0	V
V_{VCC_IO} undervoltage detector hysteresis	$V_{UV\_VIOHYST}$			0.3		V
Short to GND detector threshold (V_{VS} - V_{Ox})	V_{OS2G}		2	2.5	3	V
Short to VS detector threshold (V_{Ox})	V_{OS2VS}		1.6	2	2.3	V
Short detector delay (high side / low side switch on to short detected)	t_{S2G}	High side output clamped to V_{SP}-3V	0.8	1.3	2	µs
Overtemperature prewarning 120°C	t_{OTPW}	Temperature rising	100	120	140	°C
Overtemperature shutdown or prewarning 143°C (appr. 153°C IC peak temp.)	t_{OT143}	Temperature rising	128	143	163	°C
Overtemperature shutdown 150°C (appr. 160°C IC peak temp.)	t_{OT150}	Temperature rising	135	150	170	°C
Overtemperature shutdown 157°C (appr. 167°C IC peak temp.)	t_{OT157}	Temperature rising	142	157	177	°C
Difference between temperature detector and power stage temperature, slow heat up	t_{OTDIFF}	Power stage causing high temperature, normal 4 Layer PCB		10		°C

Sense resistor voltage levels	**DC-Characteristics** **f_{CLK}=16MHz**					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
Sense input peak threshold voltage (low sensitivity)	V_{SRTL}	*vsense*=0 *csactual*=31 *CUR_A/B*=248 Hyst.=0; I_{BRxy}=0		325		mV
Sense input peak threshold voltage (high sensitivity)	V_{SRTH}	*vsense*=1 *csactual*=31 *CUR_A/B*=248 Hyst.=0; I_{BRxy}=0		180		mV
Sense input tolerance / motor current full scale tolerance -using internal reference	I_{COIL}	*I_scale_analog*=0, *vsense*=0	-5		+5	%
Sense input tolerance / motor current full scale tolerance -using external reference voltage	I_{COIL}	*I_scale_analog*=1, V_{AIN}=2V, *vsense*=0	-2		+2	%
Internal resistance from pin BRxy to internal sense comparator (additional to sense resistor)	R_{BRxy}			30		mΩ

Digital pins	**DC-Characteristics**					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
Input voltage low level	V_{INLO}		-0.3		0.3 V_{VIO}	V
Input voltage high level	V_{INHI}		0.7 V_{VIO}		V_{VIO}+0.3	V
Input Schmitt trigger hysteresis	V_{INHYST}			0.12 V_{VIO}		V
Output voltage low level	V_{OUTLO}	I_{OUTLO} = 2mA			0.2	V
Output voltage high level	V_{OUTHI}	I_{OUTHI} = -2mA	V_{VIO}-0.2			V
Input leakage current	I_{ILEAK}		-10		10	µA
Pullup / pull-down resistors	R_{PU}/R_{PD}		132	166	200	kΩ
Digital pin capacitance	C			3.5		pF

AIN/IREF input	**DC-Characteristics**					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
AIN_IREF input resistance to 2.5V (=5VOUT/2)	R_{AIN}	Measured to GND (*internalRsense*=0)	260	330	400	kΩ
AIN_IREF input voltage range for linear current scaling	V_{AIN}	Measured to GND (*IscaleAnalog*=1)	0	0.5-2.4	V_{5VOUT}/2	V
AIN_IREF open input voltage level	V_{AINO}	Open circuit voltage (*internalRsense*=0)		V_{5VOUT}/2		V
AIN_IREF input resistance to GND for reference current input	R_{IREF}	Measured to GND (*internalRsense*=1)	0.5	0.7	1.0	kΩ
AIN_IREF current amplification for reference current to coil current at maximum setting	$I_{REFAMPL}$	I_{IREF} = 0.25mA		3000		Times
Motor current full scale tolerance -using RDSon measurement (value for design guideline to calculate reproduction of certain motor current & torque)	I_{COIL}	*Internal_Rsense*=1, *vsense*=0, I_{IREF} = 0.25mA (after reaching thermal balance)	-10		+10	%

18.3 Thermal Characteristics

The following table shall give an idea on the thermal resistance of the package. The thermal resistance for a four-layer board will provide a good idea on a typical application. Actual thermal characteristics will depend on the PCB layout, PCB type and PCB size. The thermal resistance will benefit from thicker CU (inner) layers for spreading heat horizontally within the PCB. Also, air flow will reduce thermal resistance.

A thermal resistance of 30K/W for a typical board means, that the package is capable of continuously dissipating 3.3W at an ambient temperature of 25°C with the die temperature staying below 125°C. Note, that a thermally optimized layout is required.

Parameter	Symbol	Conditions	Typ	Unit
Typical power dissipation	P_D	StealthChop or SpreadCycle, 1A RMS in two phase motor, sinewave, 40 or 20kHz chopper, 24V, 110°C peak surface of package (motor QSH4218-035-10-027)	2.5	W
Typical power dissipation	P_D	StealthChop or SpreadCycle, 0.7A RMS in two phase motor, sinewave, 40 or 20kHz chopper, 24V, 68°C peak surface of package (motor QSH4218-035-10-027)	1.36	W
Thermal resistance junction to ambient on a multilayer board QFN28	R_{TMJA}	Dual signal and two internal power plane board (2s2p) as defined in JEDEC EIA JESD51-5 and JESD51-7 (FR4, 35µm CU, 70mm x 133mm, d=1.5mm)	30	K/W
Thermal resistance junction to case	R_{TJC}	Junction to heat slug of package	6	K/W

Table 18.1 Thermal characteristics TMC22xx

Note
A spread-sheet for calculating TMC22xx power dissipation is available on www.trinamic.com.

19 Layout Considerations

19.1 Exposed Die Pad

The TMC22xx uses its die attach pad to dissipate heat from the drivers and the linear regulator to the board. For best electrical and thermal performance, use a reasonable amount of solid, thermally conducting vias between the die attach pad and the ground plane. The printed circuit board should have a solid ground plane spreading heat into the board and providing for a stable GND reference.

19.2 Wiring GND

All signals of the TMC22xx are referenced to their respective GND. Directly connect all GND pins under the device to a common ground area (GND and die attach pad). The GND plane right below the die attach pad should be treated as a virtual star point. For thermal reasons, the PCB top layer shall be connected to a large PCB GND plane spreading heat within the PCB.

Attention
Especially the sense resistors are susceptible to GND differences and GND ripple voltage, as the microstep current steps make up for voltages down to 0.5 mV. No current other than the sense resistor current should flow on their connections to GND and to the TMC22xx. Optimally place them close to the IC, with one or more vias to the GND plane for each sense resistor. The two sense resistors for one coil should not share a common ground connection trace or vias, as also PCB traces have a certain resistance.

19.3 Supply Filtering

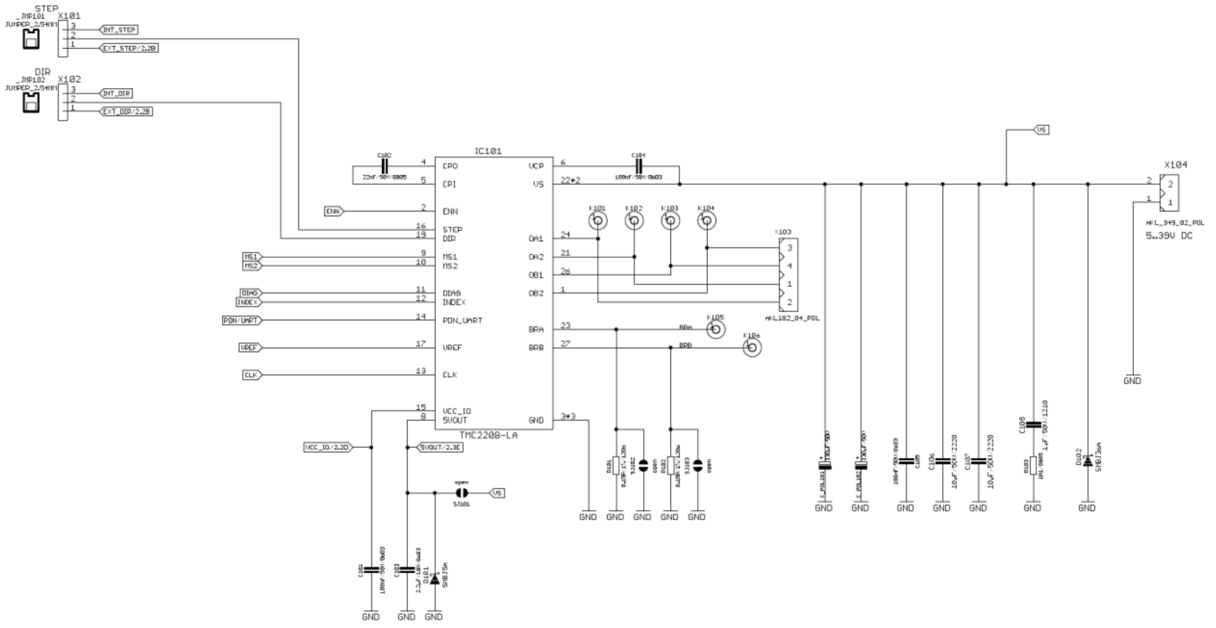
The 5VOUT output voltage ceramic filtering capacitor (2.2 to 4.7 µF recommended) should be placed as close as possible to the 5VOUT pin, with its GND return going directly to the die pad or the nearest GND pin. This ground connection shall not be shared with other loads or additional vias to the GND plane. Use as short and as thick connections as possible.

The motor supply pins VS should be decoupled with an electrolytic capacitor (47 µF or larger is recommended) and a ceramic capacitor, placed close to the device.

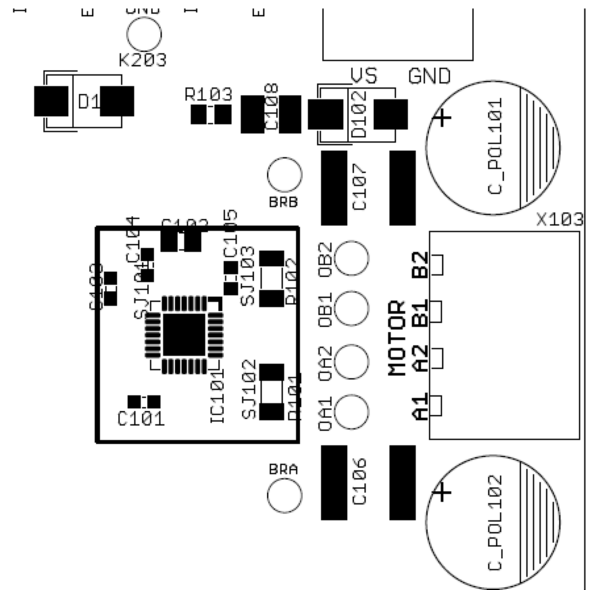
Take into account that the switching motor coil outputs have a high dV/dt. Thus capacitive stray into high resistive signals can occur, if the motor traces are near other traces over longer distances.

19.4 Layout Example TMC2208

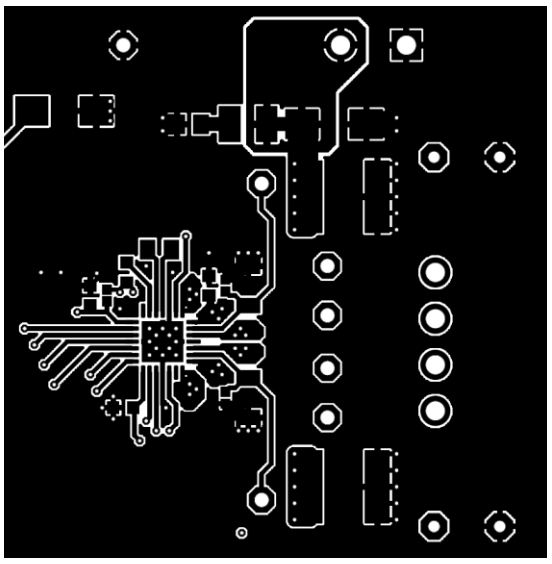
Schematic



Placement (Excerpt)



Top Layout (Excerpt, showing die pad vias)



The complete schematics and layout data for all TMC22xx evaluation boards are available on the TRINAMIC website.

20 Package Mechanical Data

20.1 Dimensional Drawings QFN28

Attention: Drawings not to scale.

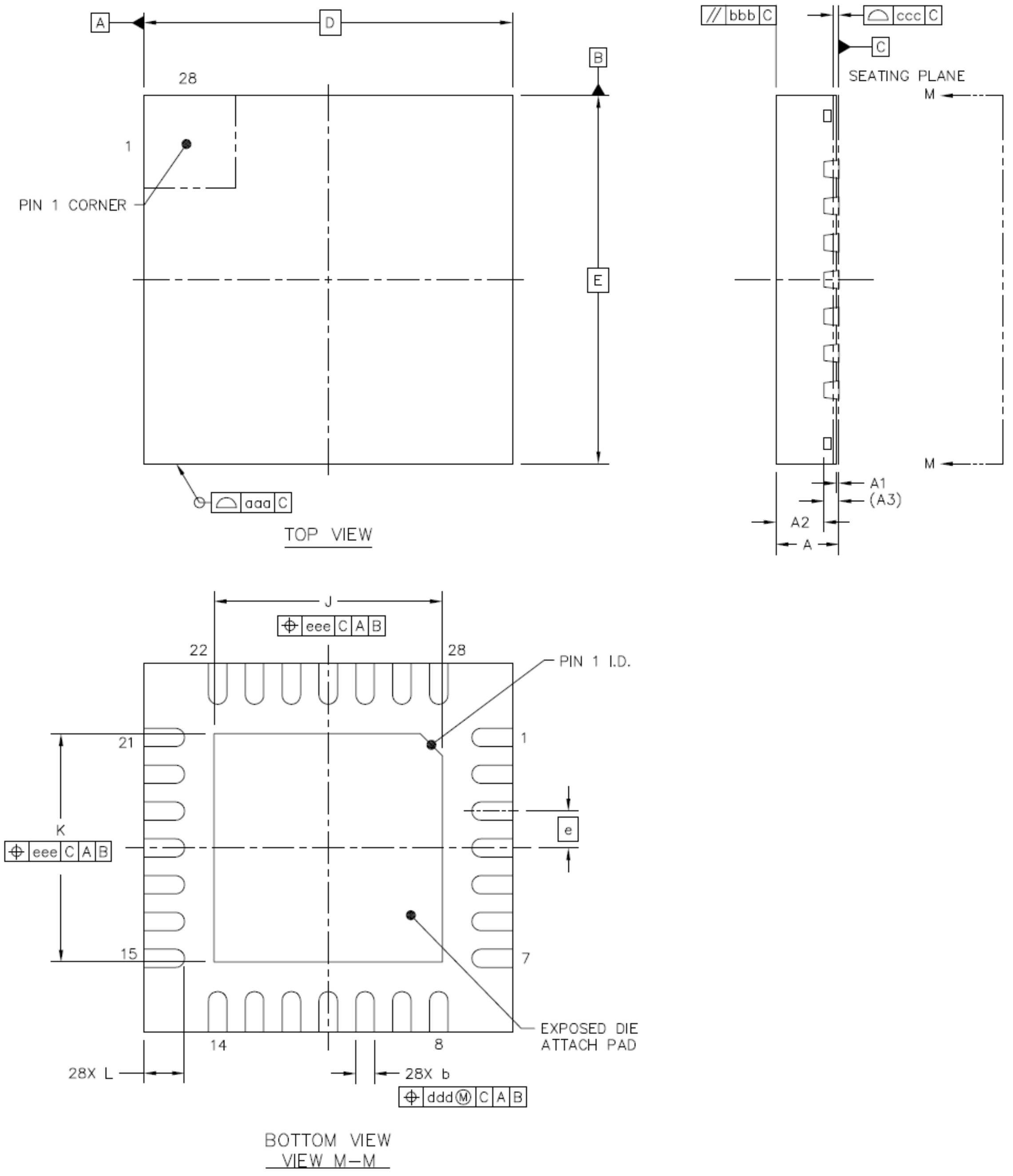


Figure 20.1 Dimensional drawings QFN28

Parameter [mm]	**Ref**	**Min**	**Nom**	**Max**
total thickness	A	0.8	0.85	0.9
stand off	A1	0	0.035	0.05
mold thickness	A2		0.65	
lead frame thickness	A3		0.203	
lead width	b	0.2	0.25	0.3
body size X	D		5.0	
body size Y	E		5.0	
lead pitch	e		0.5	
exposed die pad size X	J	3	3.1	3.2
exposed die pad size Y	K	3	3.1	3.2
lead length	L	0.5	0.55	0.6
package edge tolerance	aaa			0.1
mold flatness	bbb			0.1
coplanarity	ccc			0.08
lead offset	ddd			0.1
exposed pad offset	eee			0.1

20.2 Dimensional Drawings QFN32-WA

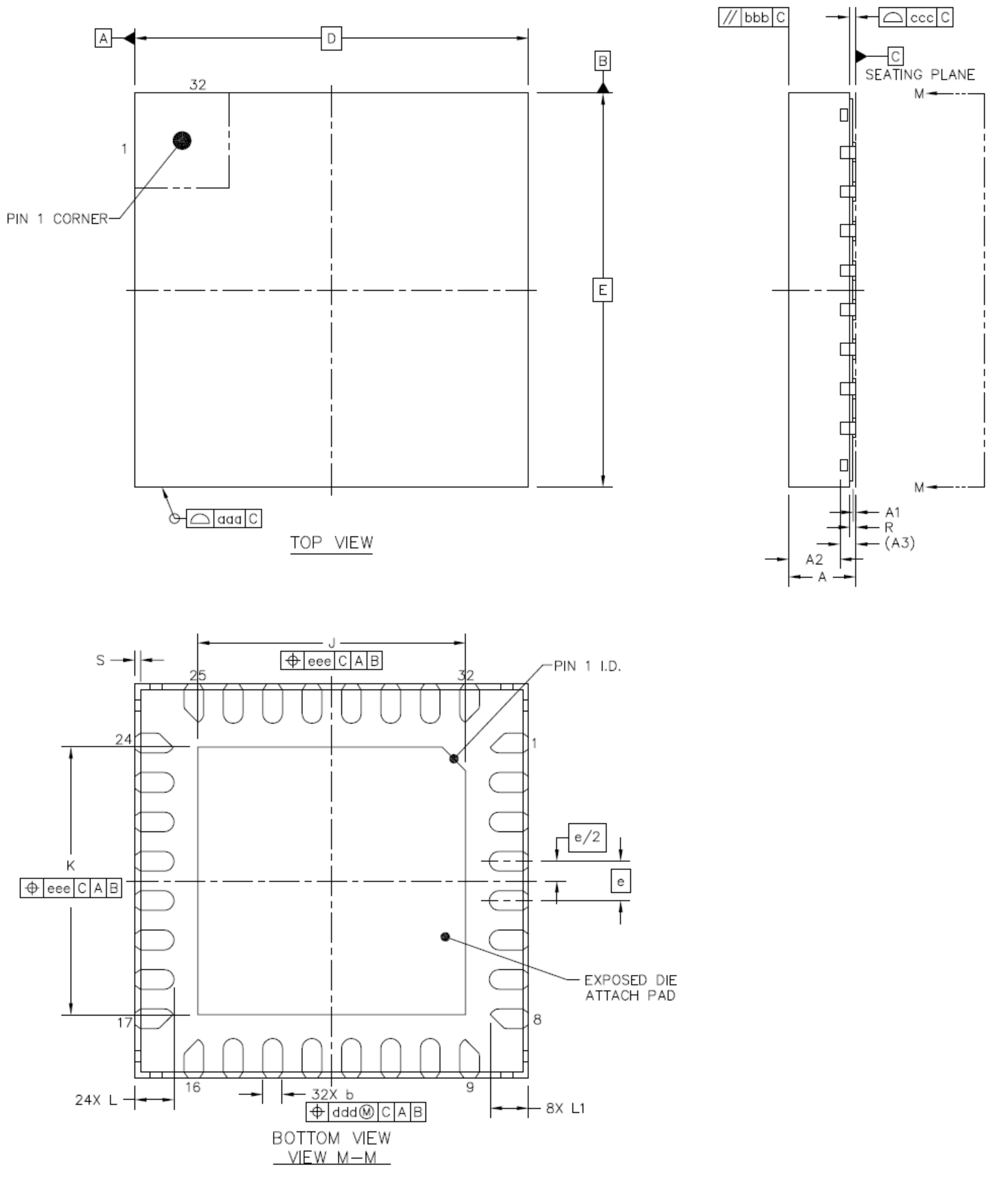


Figure 20.2 Dimensional drawings QFN32-WA

Parameter [mm]	Ref	Min	Nom	Max
total thickness	A	0.8	0.85	0.9
stand off	A1	0	0.035	0.05
mold thickness	A2		0.65	
lead frame thickness	A3		0.203	
lead width	b	0.2	0.25	0.3
body size X	D		5.0	
body size Y	E		5.0	
lead pitch	e		0.5	
exposed die pad size X	J	3.3	3.4	3.5
exposed die pad size Y	K	3.3	3.4	3.5
lead length	L	0.45	0.5	0.55
lead length	L1	0.4	0.5	0.55
package edge tolerance	aaa		0.1	
mold flatness	bbb		0.1	
coplanarity	ccc		0.08	
lead offset	ddd		0.1	
exposed pad offset	eee		0.1	
half-cut width	R	0.075		
half-cut depth	S			0.075

20.3 Package Codes

Type	Package	Temperature range	Code & marking
TMC2208-LA	QFN28 (RoHS)	-40°C … +125°C	TMC2208-LA
TMC2224-LA	QFN28 (RoHS)	-40°C … +125°C	TMC2224-LA
TMC2202-WA	QFN32 (RoHS)	-40°C … +125°C	TMC2202-WA
TMC… -T	-T suffix denotes tape on reel packed products		

21 Table of Figures

22 Revision History

Version	Date	Author BD= Bernhard Dwersteg	Description
V0.25	2016-Sep-02	BD	First complete datasheet
V0.28	2016-Sep-21	BD	Some detail wording, request for option packages, added CRC procedure
V1.00	2016-Nov-01	BD	Adapted min/max DC and Timing Characteristics (I_S, V_{CP_UV}, V_{UV_VS}, 25°C for clock calibration frequency) to fit test limits
V1.01	2016-Nov-16	BD	Corrected wording in DRV_STATUS
V1.02	2017-May-16	BD	Added QFN-32
V1.03	2017-May-31	BD	Hint on PWM_SCALE_AUTO during AT#2
V1.04	2018-May-17	BD	Minor corrections, Lead width missing in table
V1.05	2018-Jun-07	BD	Added errata / limitations for initial tuning of AT#1 / AT#2 phase Added precautions for power up with StealthChop
V1.06	2019-Feb-05	BD	Corrected timing requirements for CLK input (30ns for first pulse) / some minor fixes. Corrected calculation formula for sense resistor with 30mOhm internal resistance rather than 20mOhm.
V1.09	2019-Jul-17	BD	Minor fixes / added order codes, removed HTSSOP package

Table 22.1 Document Revisions

23 References

[TMC22xx-EVAL] TMC22xx Evaluation board: Manuals, software and PCB data available on www.trinamic.com

[AN001] Trinamic Application Note 001 - Parameterization of SpreadCycle™, www.trinamic.com

Calculation sheet TMC22xx_Calculations.xlsx www.trinamic.com