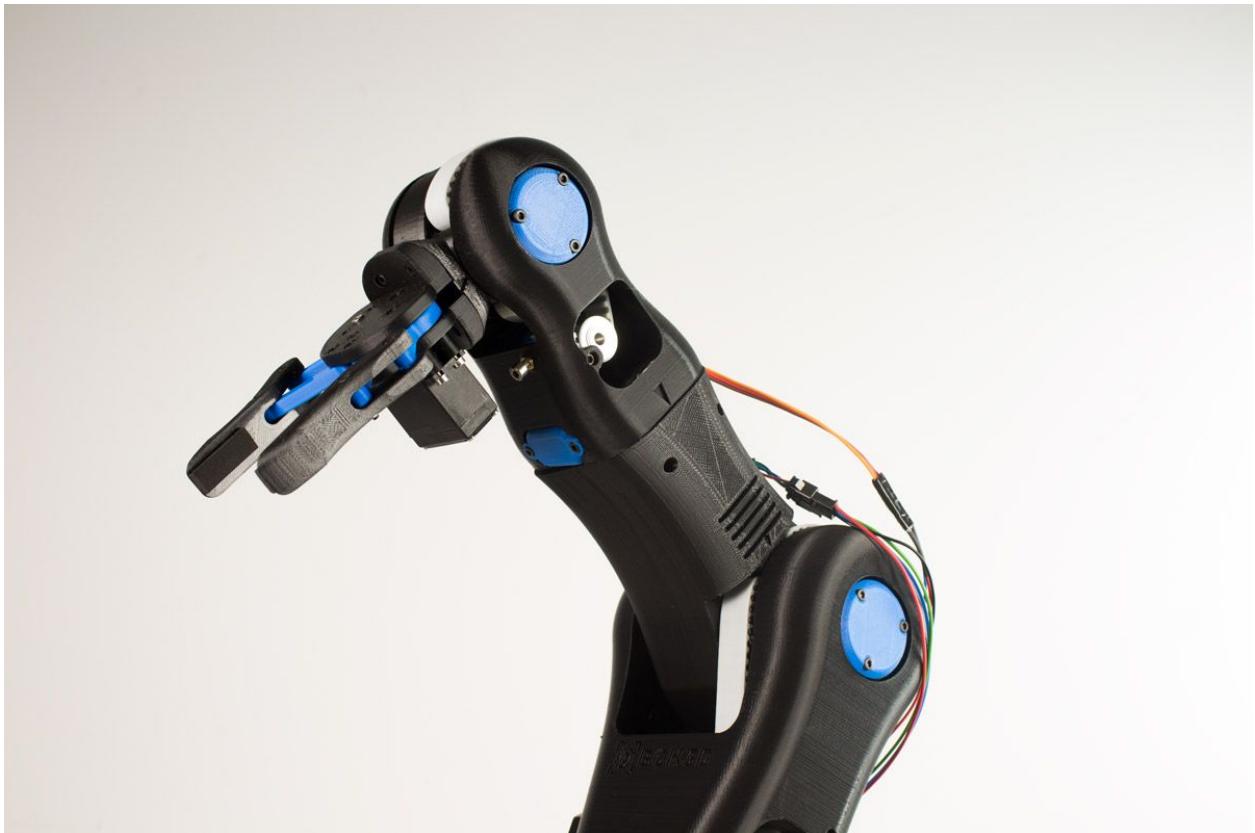


Agustín Blum, Matías Zusmanovsky y Nahuel Castro

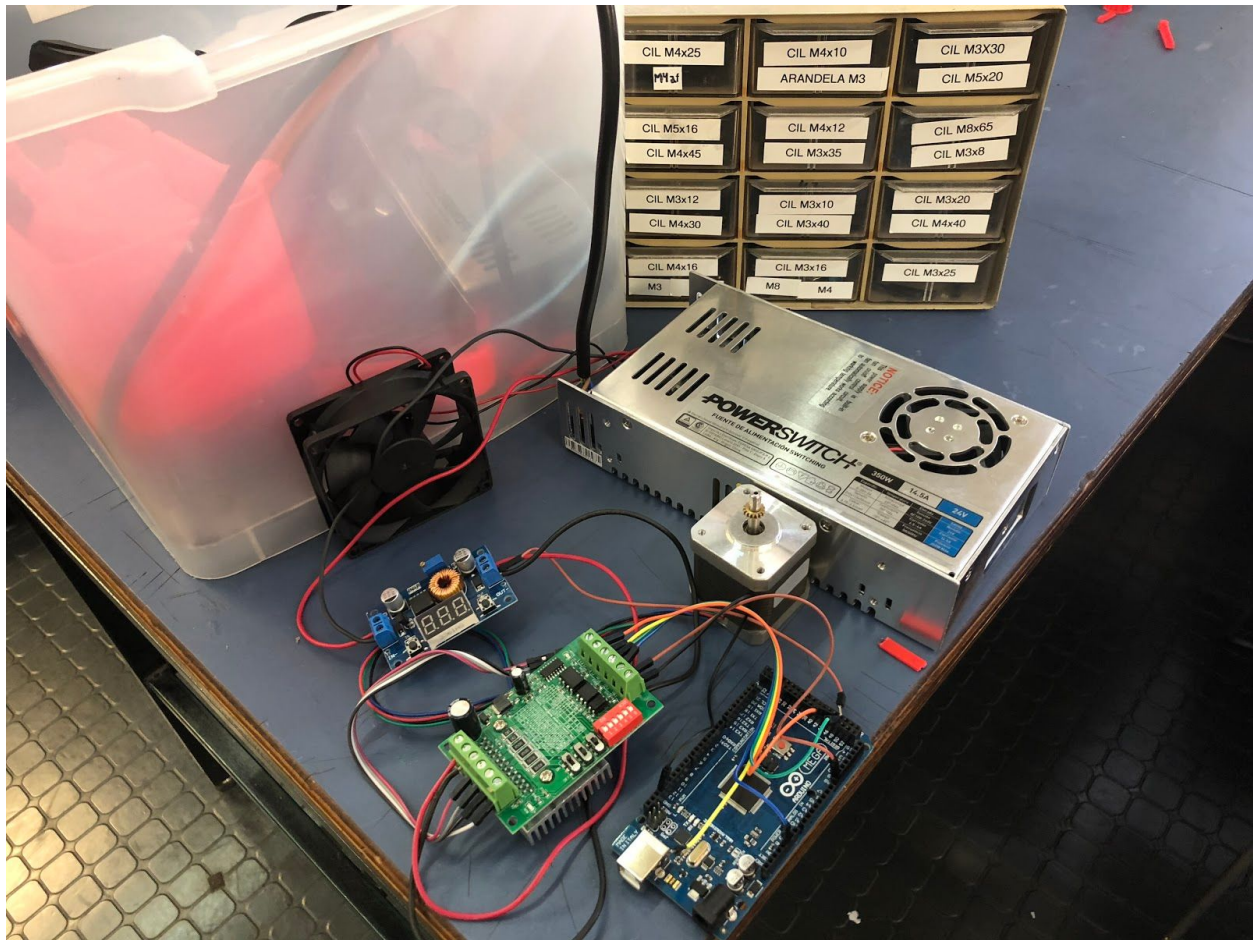
BRAZO ROBÓTICO (BCN3D Moveo)

INFORME



Introducción

Hemos comenzado bajando la carpeta con sus archivos y manual de la página oficial. Imprimimos el manual y preparamos los archivos con los g-code para comenzar a imprimir las piezas en 3D. Tuvimos que preparar cada una individualmente seleccionando el relleno y por ejemplo para algunas piezas tuvimos que hacerlas en 2 impresiones distintas ya que no entraban en la cama. También pedimos todo lo necesario para armar el brazo, esto incluye tornillos, arandelas y tuercas (las cuales hemos ordenado y etiquetado), muchos rulemanes, fuentes switching, correas, motores paso a paso, drivers para los motores, entre otras pequeñas cosas.



A medida de que fueron llegando las cosas que tuvimos que pedirle a nuestro profesor comenzamos con el ensamblaje del brazo. Sin embargo algunas piezas no llegaron entonces no pudimos armar algunas partes del mismo así que mientras tanto fuimos investigando sobre cómo conectar y hacer funcionar el brazo. Luego de averiguar, cuando llegaron las fuentes, drivers, arduino y teníamos un motor de prueba, comenzamos con las pruebas. Primero tuvimos un inconveniente ya que no sabíamos cómo conectar los pines del Arduino al driver del motor pero finalmente pudimos hacerlo y usamos el programa Pronterface para la PC para probar. Buscamos en internet cómo conectar el driver al motor y buscamos el datasheet de cada cosa para regular la corriente y demás cosas para que funcionara todo a la perfección. También usamos un reductor de tensión para bajar de 24V que entrega la fuente a 12V que utiliza el motor. Luego de conectar todo, cargamos el programa de Marlin al Arduino y abrimos el Pronterface. El programa se conectó al Arduino

exitosamente y logramos poder conectar el motor desde la interfaz del programa en la computadora.

Siempre que tuvimos algún inconveniente intentamos resolverlo por nuestra cuenta buscando en internet, y si no consultándolo con nuestro tutor. También hay otro grupo que está realizando el mismo proyecto así que estuvimos trabajando en equipo y nos ayudamos con cualquier problema que surja.

Errores o problemas

Puesto que la mayor parte de la información de armado era explicada en la página oficial de BCN Technologies, no hubo mucho lugar a errores. No obstante, hubo algunos errores al momento de improvisar algunas alternativas durante el ensamblado de algunas piezas.

A la hora de colocar las tapas en las piezas que utilizaban virutas, se intentó implementar un método alternativo para reemplazar los insertos a calor (de los cuales no disponíamos). Para ello, utilizamos silicona caliente e introducimos los tornillos en la misma para generar un molde. Esto concluye en que el calor de la silicona y su aplicador derritieron el PLA, ampliando el diámetro de los orificios, y deformándolos. Luego decidimos utilizar los insertos especificados en el manual.

Cómo armar el brazo

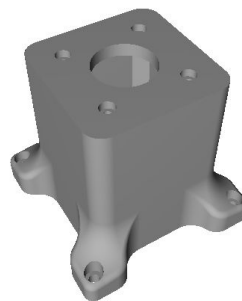
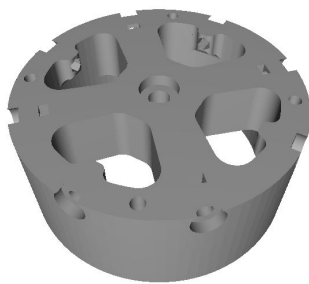
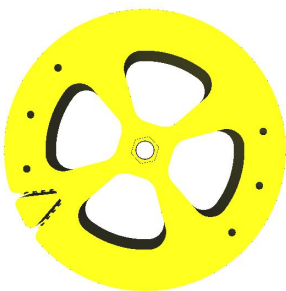
Para empezar, hay que imprimir todas las piezas en 3D con los archivos .STL ya incluidos en la carpeta descargada para el brazo. Hay que convertirlos en G-code y así imprimir pieza por pieza.



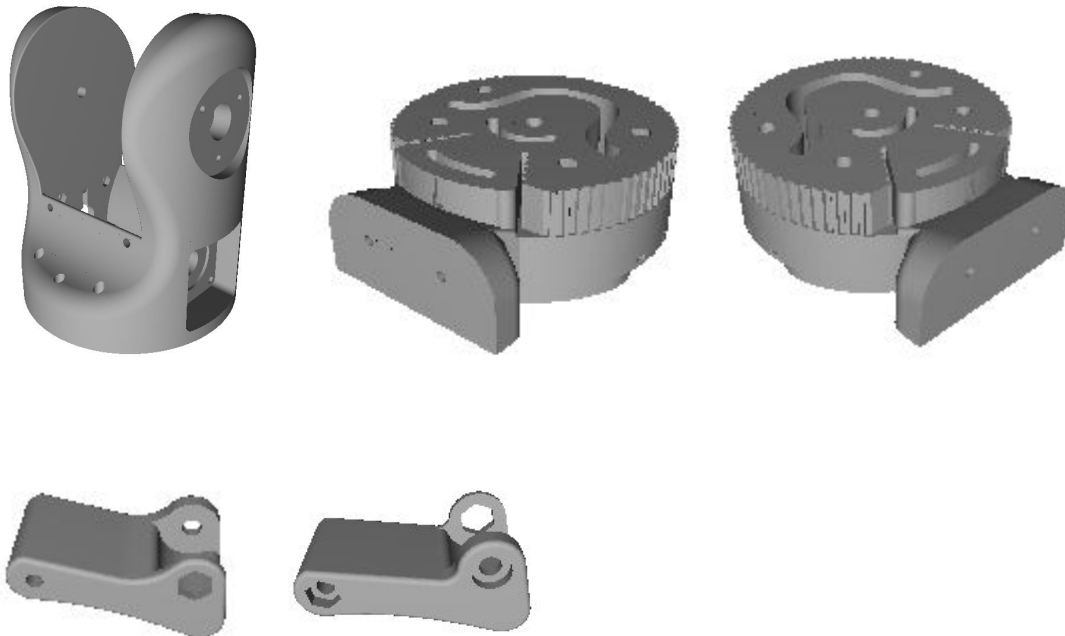
Luego de tener todas las piezas impresas como en la foto anterior, debemos comprar todos los materiales incluidos en la lista de materiales y comenzar armarlo. El instructivo se puede encontrar dentro de la carpeta que incluye todo lo necesario para poder armarlo.

Piezas impresas en 3D

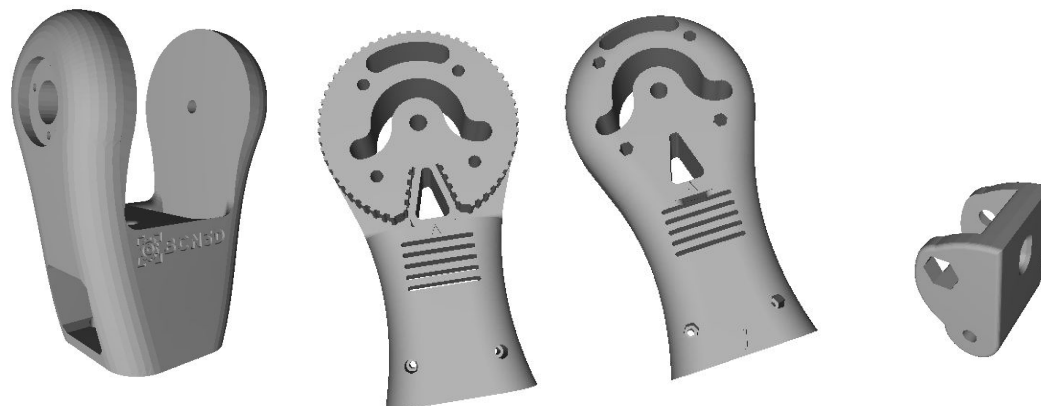
ARTICULACIÓN 1:



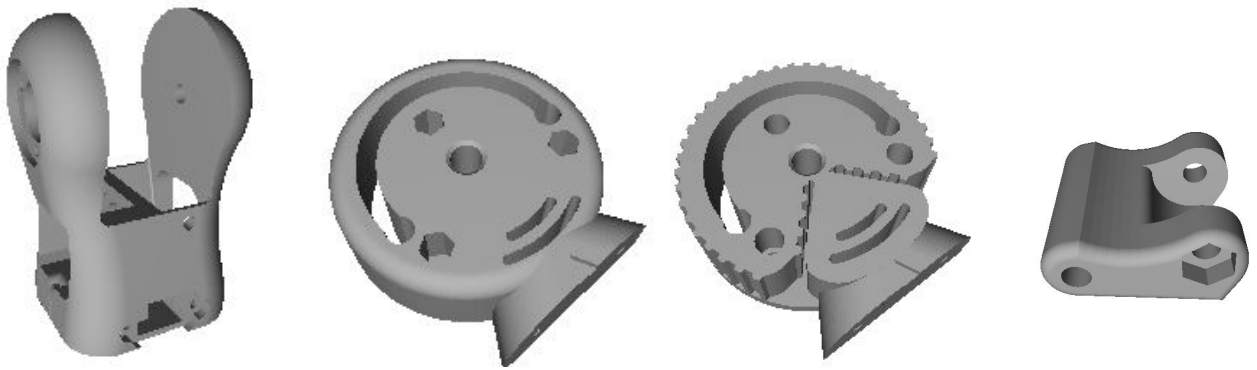
ARTICULACIÓN 2:



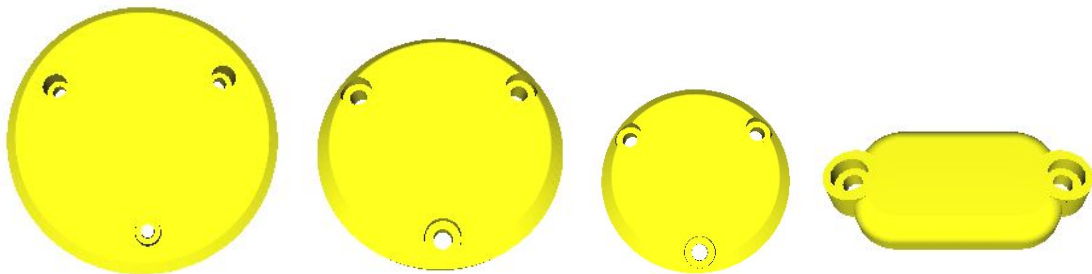
ARTICULACIÓN 3:



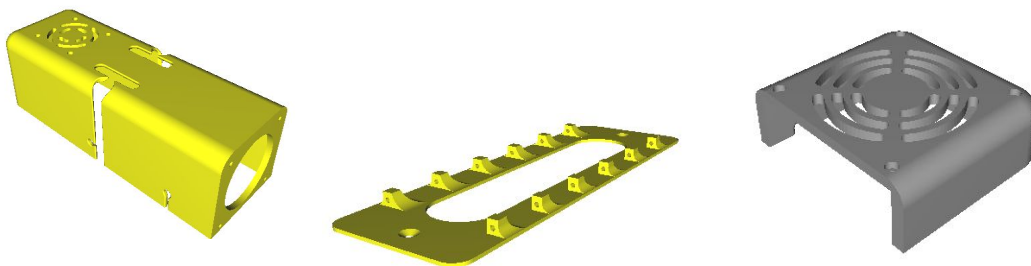
ARTICULACIÓN 4:



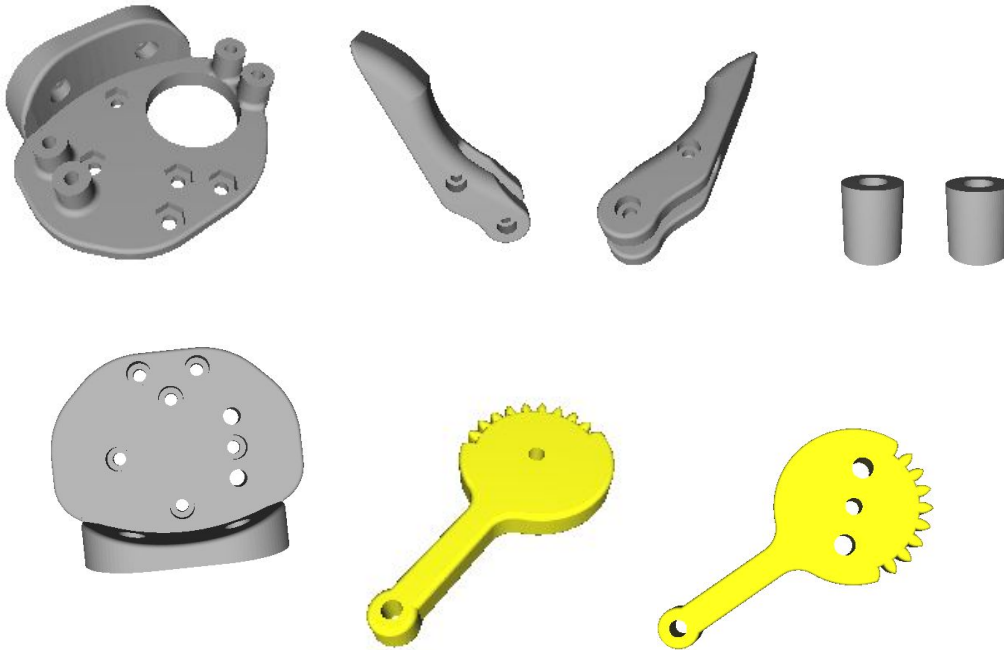
TAPAS:



CAJA ELECTRÓNICA:



HERRAMIENTA (PINZA):



POLEAS:



Hardware

El BCN3D MOVEO posee motores paso a paso, los cuales son más precisos que los motores comunes a la hora de manipularlos y emplearlos en tareas que requieren de mayor minuciosidad; drivers para los motores; un cooler, utilizado para ventilar los drivers; un microcontrolador Arduino Mega 2560; y las piezas de la estructura, las cuales se imprimen en el material PLA mediante el uso de impresoras 3D.

Asimismo también se requiere de otros componentes para el ensamblado del producto: rodamientos, tuercas, tornillos, virutas, correas y poleas, las cuales también fueron impresas en 3D.

Algunas de las piezas que debieron ser impresas para el armado del brazo, eran demasiado largas. por ello, debimos dividir las en dos partes, las cuales luego pegamos con cloroformo.

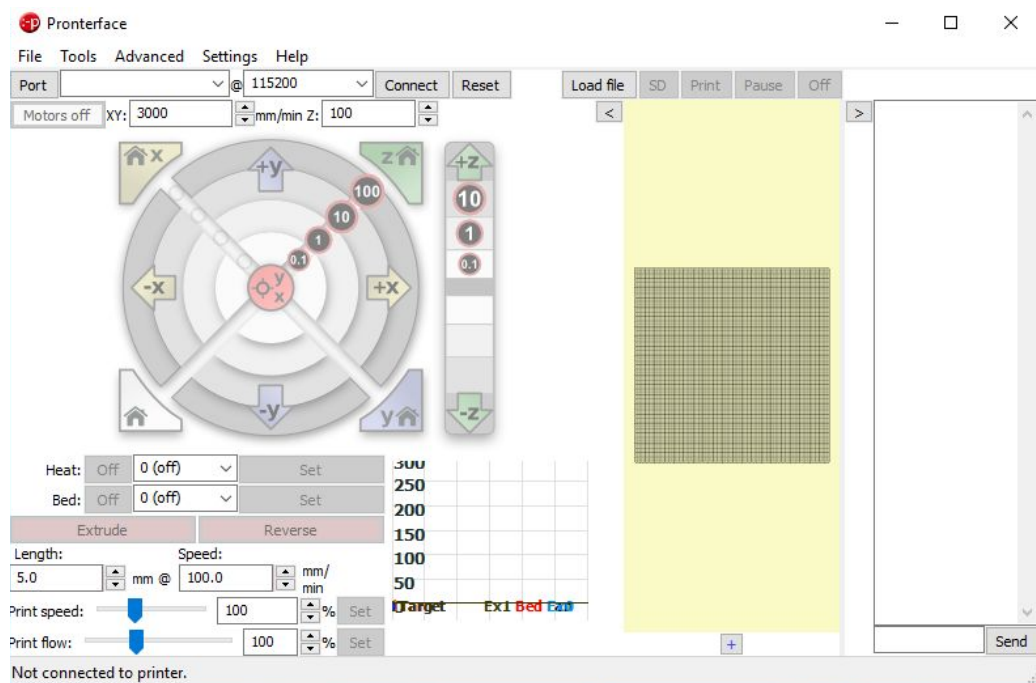
Software

Para que el brazo robótico esté operativo, es necesario programar el Arduino Mega con el código necesario para su control. El código utilizado para controlar dicho equipo es en realidad el firmware Marlin que se usa generalmente para controlar impresoras 3D. El firmware Marlin fue adaptado por BCN Technologies para su uso en este proyecto.

Con el objetivo de controlar el brazo del robot para que se mueva a donde se desee, se utiliza el G-Code. La combinación del firmware Marlin y Arduino es una plataforma para interpretar y ejecutar el G-Code.

Para testear los motores paso a paso moverse en distintos ejes del Moveo, utilizamos un programa llamado Pronterface. Es un programa predeterminado implementado en el testeo de impresoras 3D, puede ser descargado de Internet y modificado como el programador lo desee.

Con él, se pueden mover los ejes individualmente hacia las direcciones y controlar los extrusores (en casos como los de las impresoras 3D). Establecimos una comunicación entre Pronterface y el Arduino para poder hacer mover a los motores a distintos ejes.



En el Manual de Usuario se explica como usarlo.

Ejes

Los ejes del BCN3D MOVEO son los siguientes, siendo W y V dos ejes que le permiten mayor movilidad y alcance a zonas hacia las cuales el brazo no dispondría de acceso sin dichos ejes extra. Posteriormente, en el programa, serán denominados Y2 y Z2

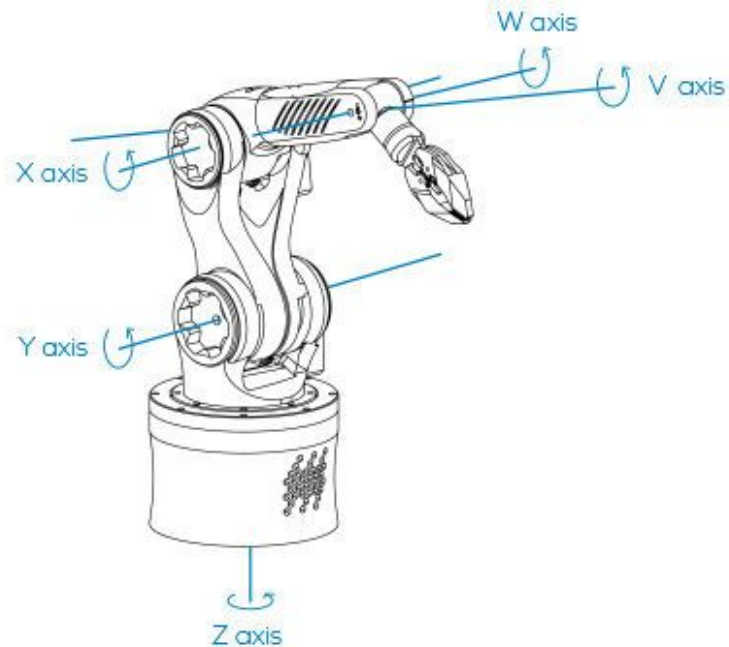


Diagrama de conexiones

Usamos una fuente switching de 24V. Esta se conecta a 220v. Luego usamos reguladores de tensión para poder llegar a los 12V y así poder alimentar a los motores sin ningún problema.



Los Drivers son quienes controlan los motores (nosotros usamos el Stepper Driver TB6560), el cual se conecta en los pines correspondientes a cada motor en el arduino, teniendo cada driver los siguientes

pinos:

EN + (Positive Enable)

EN - (Negative Enable)

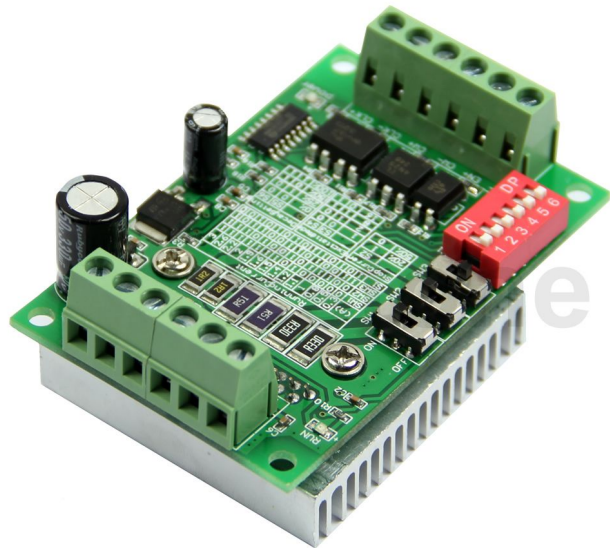
CLK + (Positive Step)

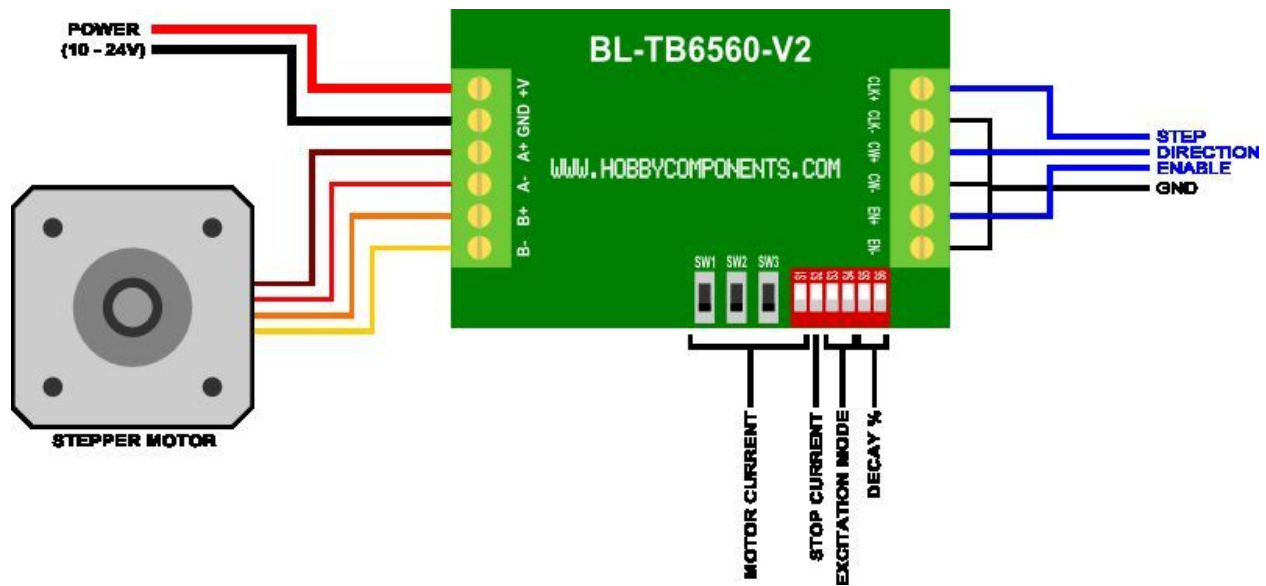
CLK - (Negative Step)

CW + (Positive Direction)

CW - (Negative Direction)

Todos los negativos irán conectados a GND, mientras que los positivos dispondrán de sus propios pines en el microcontrolador Arduino Mega 2560. (conexiones Driver-Arduino) pág.13 - 16.





Arduino Mega 2560

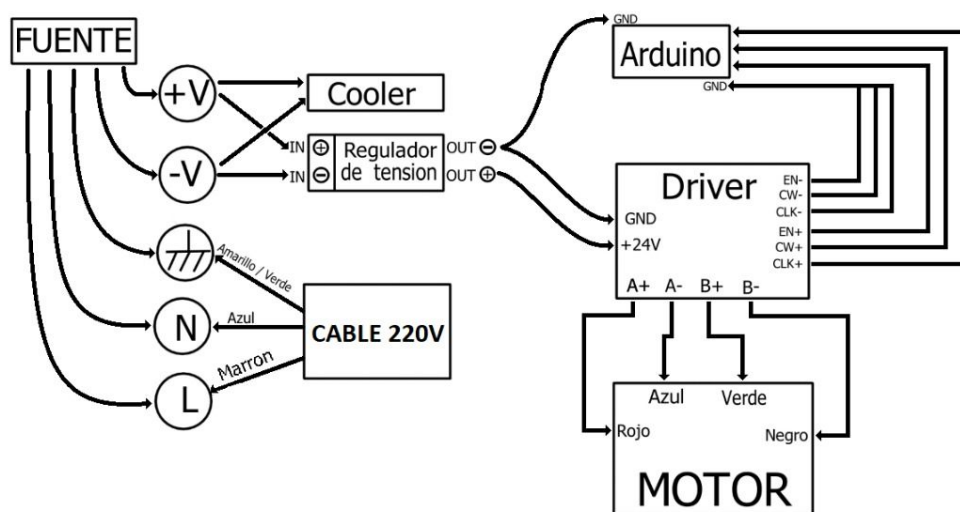


Del cable de 220V, el cable a tierra se conecta con la "masa" de la fuente de 24V 320W, el cable de fase a "L" y el cable neutro al "N".

La tensión +V es conectada con el "Cooler" y a la vez con la entrada del "Regulador de Tensión" (IN +).

Y la tensión -V también es conectada con el "Cooler" y con otra entrada del "Regulador" (IN -).

La salida + del Regulador (OUT +) es conectada con +24V y la otra salida - del Regulador (OUT -) con la masa del stepper driver tb6560 y la masa del Arduino Mega.



Conexiones Stepper Driver TB6560 - Arduino Mega 2560

EJE X

Driver	Arduino Mega 2560
EN-	GND
EN+	D38
CW-	GND
CW+	A1
CLK-	GND
CLK+	A0

EJE Y

Stepper Driver TB6560	Arduino Mega 2560
EN-	GND
EN+	A2
CW-	GND
CW+	A7
CLK-	GND
CLK+	A6

EJE Z

Stepper Driver TB6560	Arduino Mega 2560
EN-	GND
EN+	A8
CW-	GND
CW+	D48
CLK-	GND
CLK+	D46

EJE Y2

Stepper Driver TB6560	Arduino Mega 2560
EN-	GND
EN+	D30
CW-	GND
CW+	D34
CLK-	GND
CLK+	D36

EJE Z2

Stepper Driver TB6560	Arduino Mega 2560
EN-	GND
EN+	D30
CW-	GND
CW+	D34
CLK-	GND
CLK+	D36

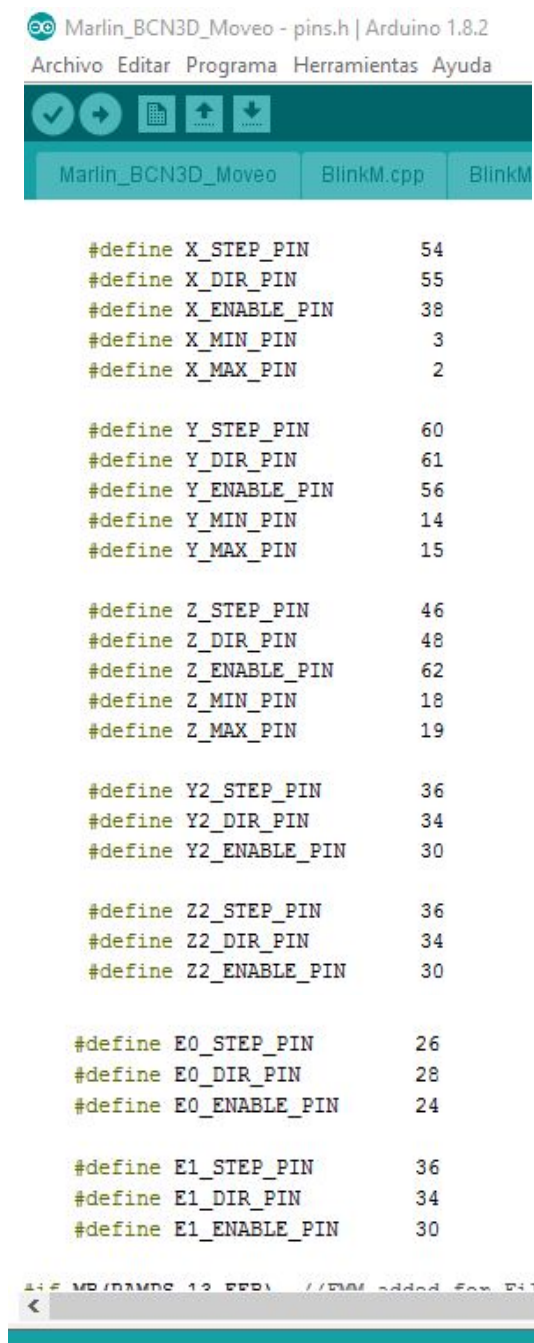
EJE E0

Stepper Driver TB6560	Arduino Mega 2560
EN-	GND
EN+	D24
CW-	GND
CW+	D28
CLK-	GND
CLK+	D26

EJE E1

Stepper Driver TB6560	Arduino Mega 2560
EN-	GND
EN+	D30
CW-	GND
CW+	D34
CLK-	GND
CLK+	D36

Las tablas con las conexiones finales entre el Stepper Driver TB6560 y el Arduino Mega 2560 las sacamos de esta manera:



```
#define X_STEP_PIN      54
#define X_DIR_PIN       55
#define X_ENABLE_PIN    38
#define X_MIN_PIN       3
#define X_MAX_PIN       2

#define Y_STEP_PIN      60
#define Y_DIR_PIN       61
#define Y_ENABLE_PIN    56
#define Y_MIN_PIN       14
#define Y_MAX_PIN       15

#define Z_STEP_PIN      46
#define Z_DIR_PIN       48
#define Z_ENABLE_PIN    62
#define Z_MIN_PIN       18
#define Z_MAX_PIN       19

#define Y2_STEP_PIN     36
#define Y2_DIR_PIN      34
#define Y2_ENABLE_PIN   30

#define Z2_STEP_PIN     36
#define Z2_DIR_PIN      34
#define Z2_ENABLE_PIN   30

#define E0_STEP_PIN     26
#define E0_DIR_PIN      28
#define E0_ENABLE_PIN   24

#define E1_STEP_PIN     36
#define E1_DIR_PIN      34
#define E1_ENABLE_PIN   30

// #if MB (RAMPS 1.2 4BB) // PWM added for E1
```

Los valores de los define del código de arduino representa a los pines del microcontrolador del Arduino Mega 2560. En el siguiente link, se encontrará un mapa que muestra cada pin del microcontrolador, con su respectivo nombre y ubicación.

<https://www.arduino.cc/en/Hacking/PinMapping2560>

Y en este link. se encontrará con una lista de los define que representa los pines del microcontrolador.

https://docs.google.com/document/d/1NW4JgM0IBQLc_weDuoWvY-b0EZL1Y6Yb7Ju72RJdMNo/edit

Cada pines del microcontrolador son relacionadas con los pines de Arduino Mega, como Digital pin, Analog pin, GND, VCC, etc.

Algunas imágenes durante en el proyecto

