

Things Read to setup your Barduino + CNC shield

→ Grbl_Esp32. ino + all libraries from this repo https://gitlab.com/fablabbcn-projects/cnc-machines/six-pack-cnc/-/tree/master/firmware-grblesp32/firmware-3AxisBrushedMotor/Grbl_Esp32

→ Documentation for the motor driver: <https://www.pololu.com/product/2132>
<https://www.pololu.com/product/2133>

NEMA 11x 3 + NEMA 17 x3

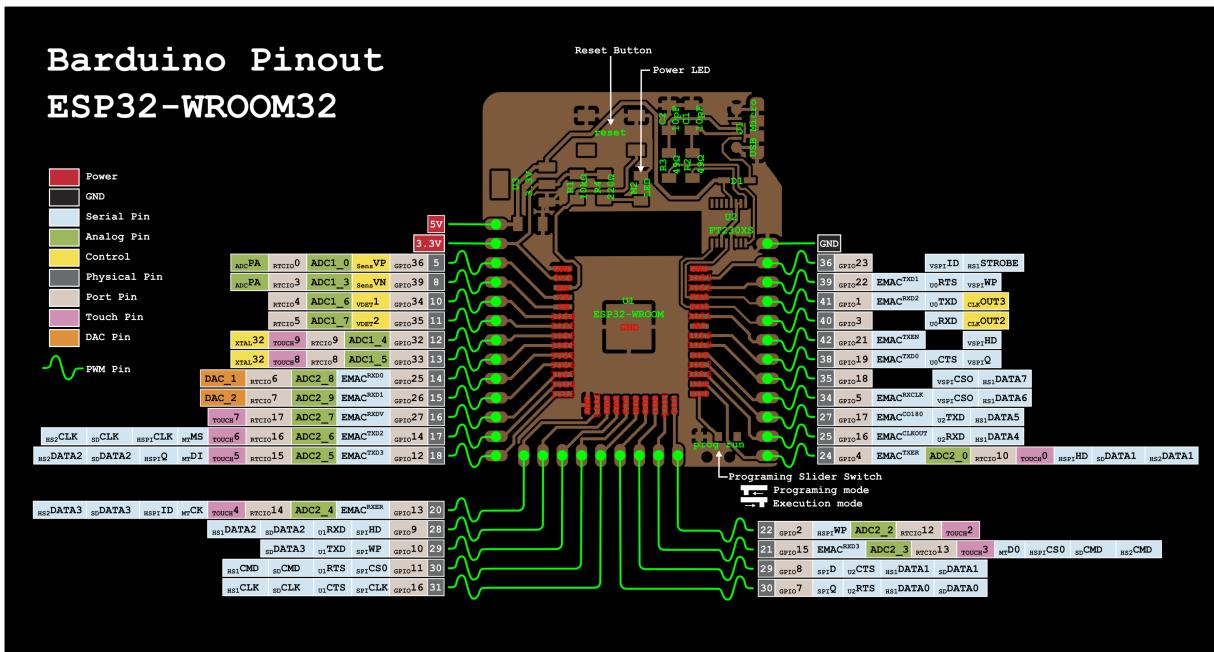
NEMA14 Size: 35 mm square × 36 mm, not including the shaft (NEMA 14)

Barduino Info

<https://gitlab.com/fablabbcn-projects/electronics/barduino>

Barduino 2.2 → <https://gitlab.com/fablabbcn-projects/electronics/barduino/-/tree/master/barduino-microusb-2.2>

Interface FTDI-USB* Fabbable Yes CNC Shield Compatible Yes



Libraries

https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

https://dl.espressif.com/dl/package_esp32_index.json

Shield - original repo

https://github.com/bdring/Grbl_Esp32/wiki/Development-Roadmap

ESP32- Web UI

<https://github.com/luc-github/ESP3D-WEBUI>

Things to do to Use the CNC shield + Barduino

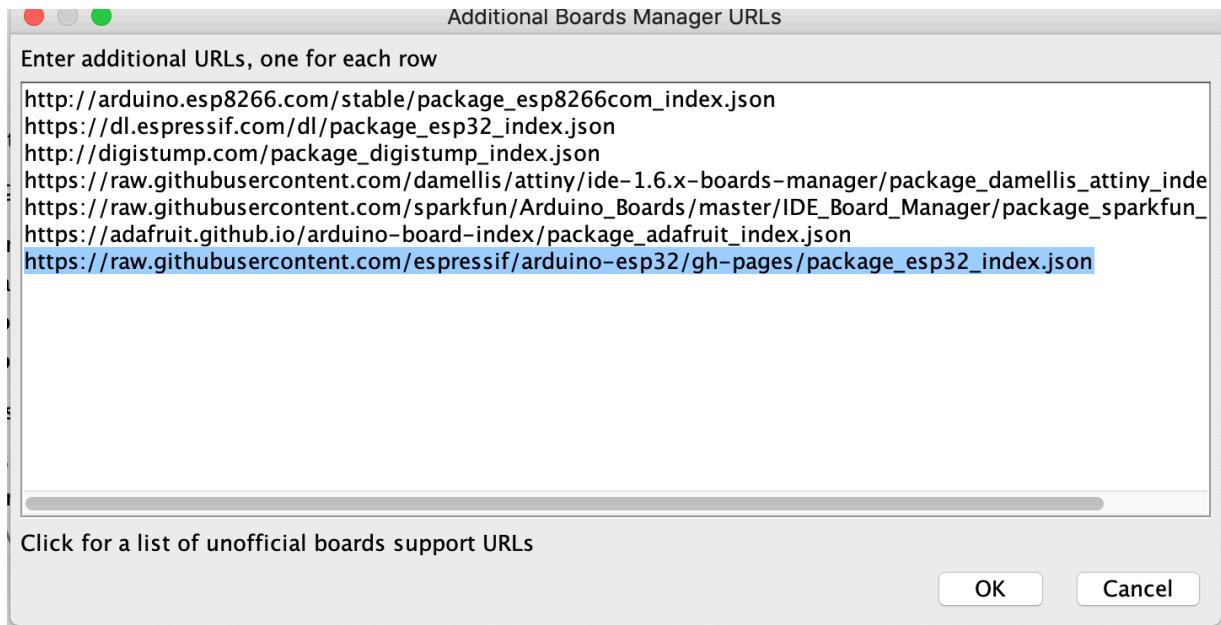
1- Download or Clone this repo —>https://gitlab.com/fablabbcn-projects/cnc-machines/six-pack-cnc/-/tree/master/firmware-grblesp32/firmware-3AxisBrushedMotor/Grbl_Esp32

2- Copy this two libraries from the libraries folder on the Firmware libraries folder to your Arduino Library folder. Mine is on Documents/Arduino/Libraries.

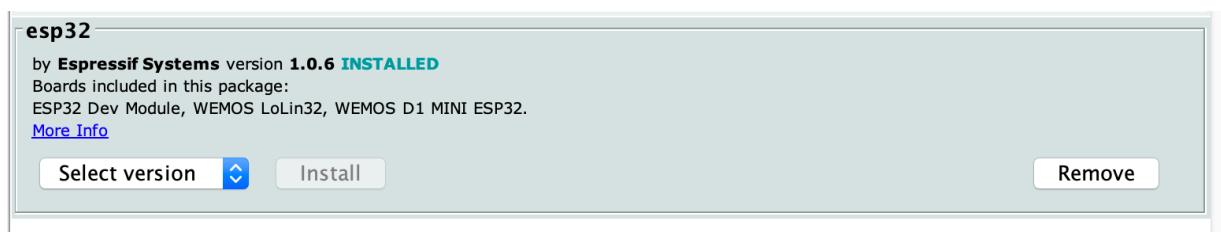
- ESP32SSDP
- arduinoWebSockets

3- Install Board Manager for ESP32 —> Arduino —> Preferences

https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json



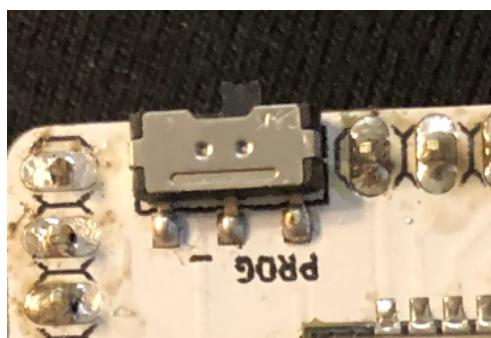
4 - Install ESP32 Espressif on Board Manager



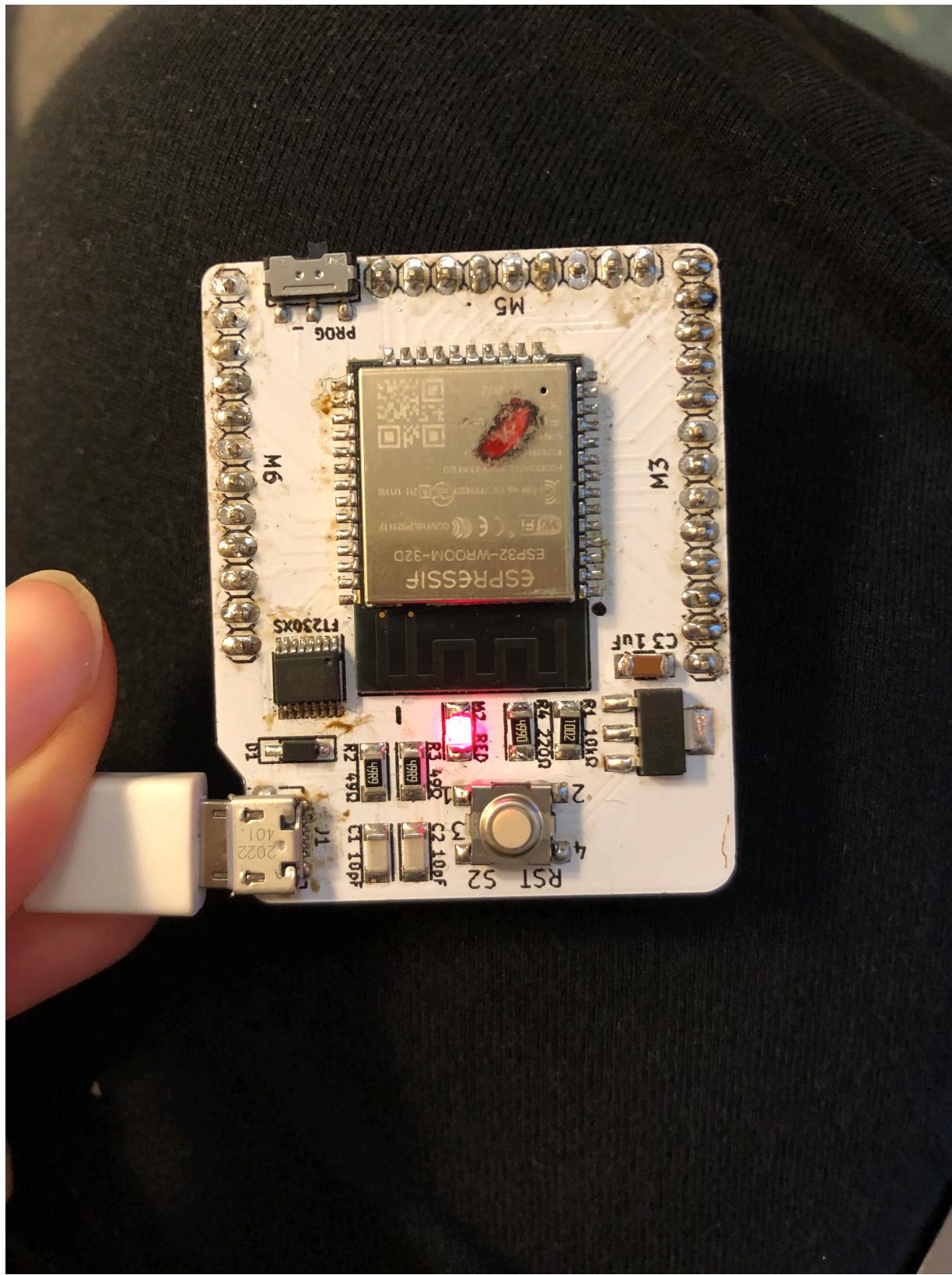
5- Reading installations instructions from here:

<https://fabacademy.org/2020/labs/barcelona/students/david-prieto/projects/AssemblingSPML/#assembling-a-six-pack-milling-machine>

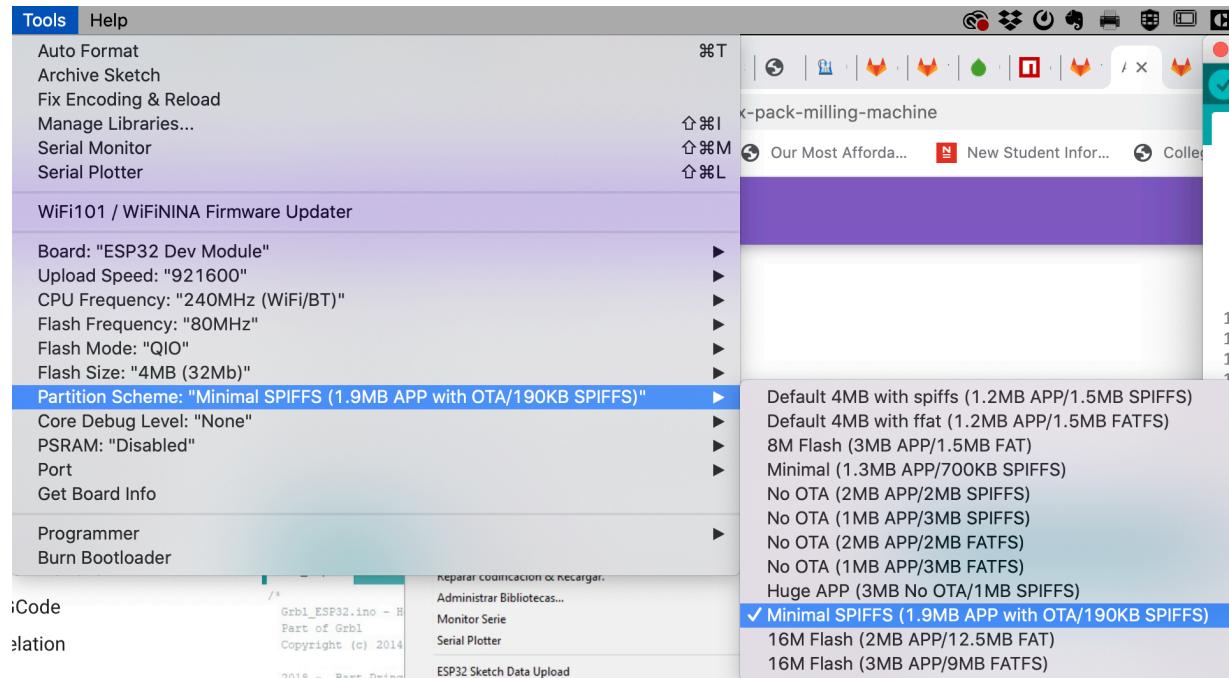
Before 6 - Make sure that before you connect the board the switch is on programming mode:



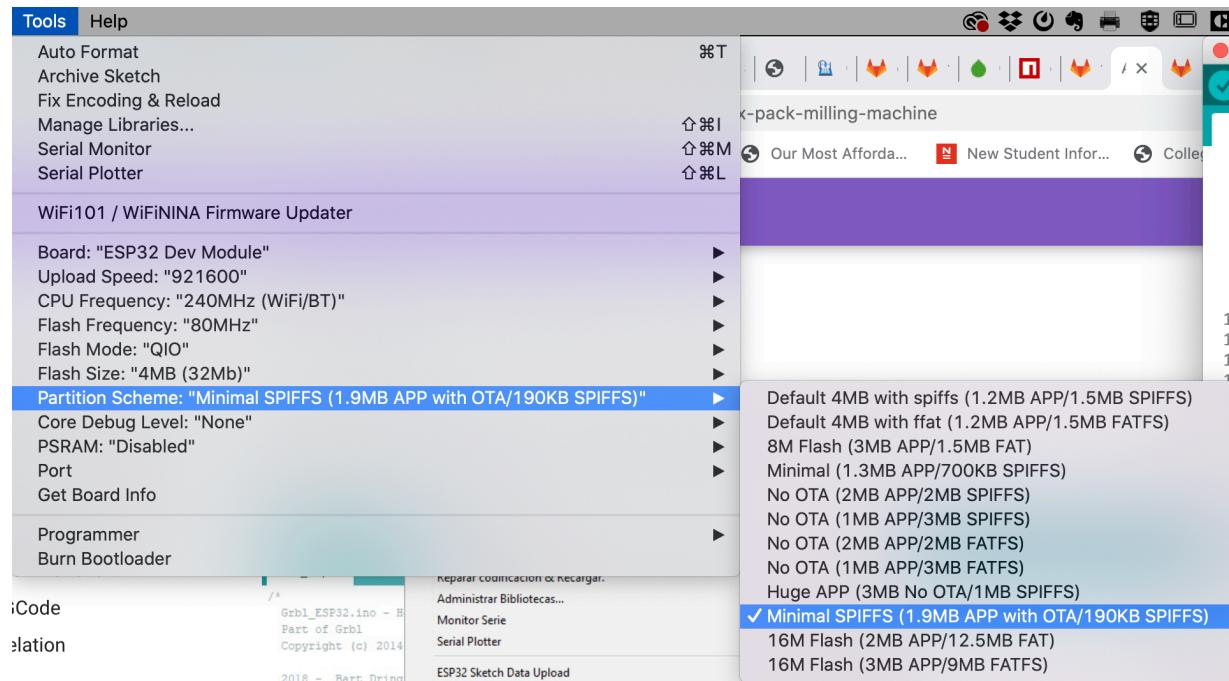
6- Original Grbl documentation says to remove the barduino board from the shield to program the firmware to prevent any possible damage to the shield. Connect the board:



7- select board. ESP32 - dev Module



8- change Partition scheme to Minimal SPIFFS



9- Compile Firmware Arduino File - it will take a while depending on the machine.

The screenshot shows the Arduino IDE interface with the title bar "Grbl_Esp32 | Arduino 1.8.13". The code editor contains the "Grbl_Esp32.ino" file. The code includes comments about the license (GNU General Public License), the hardware (ESP32), and the software (Grbl). It defines system global variables like `int32_t sys_position[N_AXIS]` and `volatile uint8_t sys_probe_state`. The `void setup()` function initializes WiFi with `WiFi.persistent(false)`, `WiFi.disconnect(true)`, `WiFi.enableSTA(false)`, and `WiFi.enableAP(false)`.

```
Grbl_Esp32 | Arduino 1.8.13

Grbl_Esp32.ino - Header for system level commands and real-time processes
Part of Grbl
Copyright (c) 2014-2016 Sungeun K. Jeon for Gnea Research LLC

2018 - Bart Dring This file was modified for use on the ESP32
CPU. Do not use this with Grbl for atMega328P

Grbl is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.
Grbl is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.
You should have received a copy of the GNU General Public License
along with Grbl. If not, see <http://www.gnu.org/licenses/>.

/*
#include "grbl.h"
#include "WiFi.h"

// Declare system global variable structure
system_t sys;
int32_t sys_position[N_AXIS]; // Real-time machine (aka home) position vector in steps.
int32_t sys_probe_position[N_AXIS]; // Last probe position in machine coordinates and steps.
volatile uint8_t sys_probe_state; // Probing state value. Used to coordinate the probing cycle with stepper ISR.
volatile uint8_t sys_rt_exec_state; // Global realtime executor bitflag variable for state management. See EXEC bitmasks.
volatile uint8_t sys_rt_exec_alarm; // Global realtime executor bitflag variable for setting various alarms.
volatile uint8_t sys_rt_exec_motion_override; // Global realtime executor bitflag variable for motion-based overrides.
volatile uint8_t sys_rt_exec_accessory_override; // Global realtime executor bitflag variable for spindle/coolant overrides.
#ifndef DEBUG
    volatile uint8_t sys_rt_exec_debug;
#endif

void setup() {
    WiFi.persistent(false);
    WiFi.disconnect(true);
    WiFi.enableSTA(false);
    WiFi.enableAP(false);
}

Compiling sketch...
Linking everything together...
/Users/carla/Library/Arduino15/packages/esp32/tools/xtensa-esp32-elf-gcc/1.22.0-97-gc752ad5-5.2.0/bin/xtensa-esp32-elf-gcc -nostdlib -L/Users/
40
ESP32 Dev Module on /dev/cu.usbserial-D30993JK
```

10 - Upload the code

The screenshot shows the Arduino IDE interface with the title bar "Grbl_Esp32 | Arduino 1.8.13". The main window displays the code for the "Grbl_ESP32.ino" sketch. The code includes comments about the license (GNU General Public License), copyright (2014-2016 Sungeun K. Jeon for Gnea Research LLC), and modifications (2018 by Bart Dring). It defines system global variables like `sys` (a `system_t` struct) and various bitflags for system management. The `setup()` function is shown with code to initialize WiFi. Below the code editor, a status bar indicates "Compiling sketch..." and "Linking everything together...". A terminal window at the bottom shows the command being run: "/Users/carla/Library/Arduino15/packages/esp32/tools/xtensa-esp32-elf-gcc/1.22.0-97-gc752ad5-5.2.0/bin/xtensa-esp32-elf-gcc -nostdlib -L/Users/".

```
Grbl_Esp32 | Arduino 1.8.13

Grbl_Esp32.ino - Header for system level commands and real-time processes
Part of Grbl
Copyright (c) 2014-2016 Sungeun K. Jeon for Gnea Research LLC
2018 - Bart Dring This file was modified for use on the ESP32
CPU. Do not use this with Grbl for atMega328P

Grbl is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.
Grbl is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.
You should have received a copy of the GNU General Public License
along with Grbl. If not, see <http://www.gnu.org/licenses/>.

/*
#include "grbl.h"
#include "WiFi.h"

// Declare system global variable structure
system_t sys;
int32_t sys_position[N_AXIS]; // Real-time machine (aka home) position vector in steps.
int32_t sys_probe_position[N_AXIS]; // Last probe position in machine coordinates and steps.
volatile uint8_t sys_probe_state; // Probing state value. Used to coordinate the probing cycle with stepper ISR.
volatile uint8_t sys_rt_exec_state; // Global realtime executor bitflag variable for state management. See EXEC bitmasks.
volatile uint8_t sys_rt_exec_alarm; // Global realtime executor bitflag variable for setting various alarms.
volatile uint8_t sys_rt_exec_motion_override; // Global realtime executor bitflag variable for motion-based overrides.
volatile uint8_t sys_rt_exec_accessory_override; // Global realtime executor bitflag variable for spindle/coolant overrides.
#endif DEBUG
volatile uint8_t sys_rt_exec_debug;
#endif

void setup() {
  WiFi.persistent(false);
  WiFi.disconnect(true);
  WiFi.enableSTA(false);
}

Compiling sketch...
Linking everything together...
/Users/carla/Library/Arduino15/packages/esp32/tools/xtensa-esp32-elf-gcc/1.22.0-97-gc752ad5-5.2.0/bin/xtensa-esp32-elf-gcc -nostdlib -L/Users/
ESP32 Dev Module on /dev/cu.usbserial-D30993JK
40
```

DONE!!!!

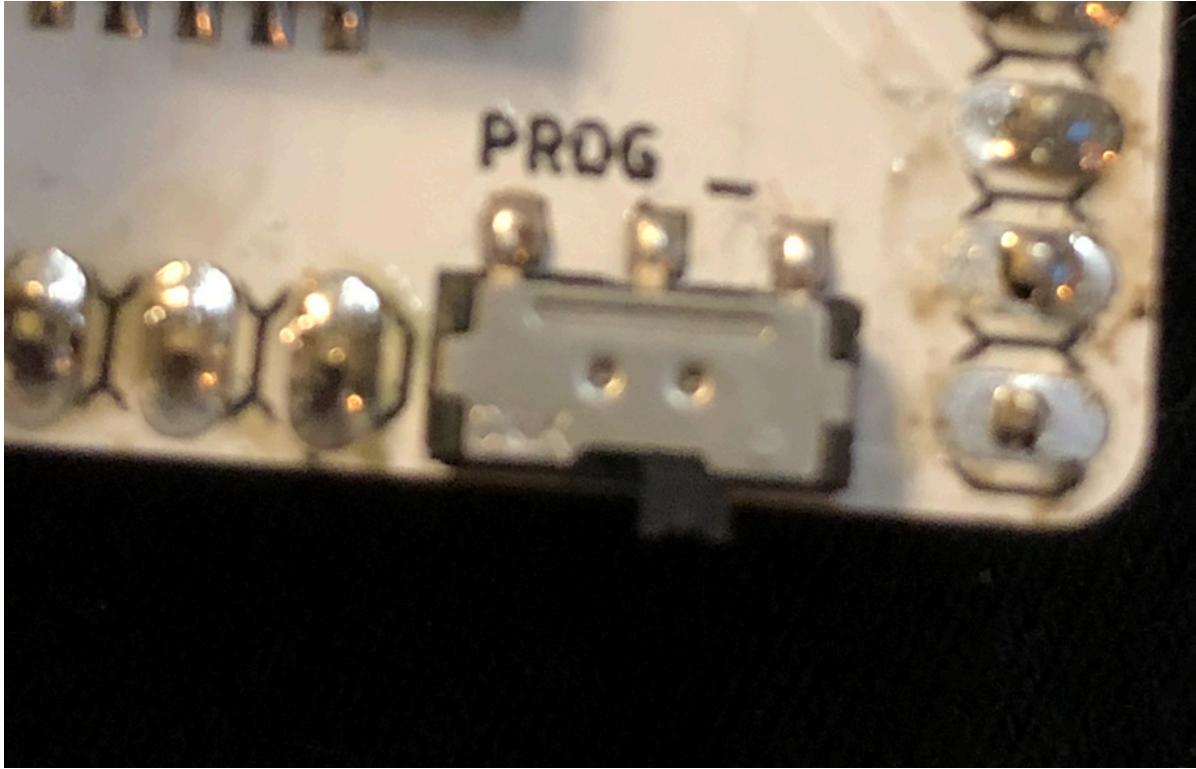
```
Done uploading.  
Writing at 0x0000c000 ... (53 %)  
Writing at 0x0000a000 ... (55 %)  
Writing at 0x00004000 ... (56 %)  
Writing at 0x0000a8000 ... (58 %)  
Writing at 0x0000ac000 ... (59 %)  
Writing at 0x00000000 ... (61 %)  
Writing at 0x00004000 ... (62 %)  
Writing at 0x00008000 ... (64 %)  
Writing at 0x0000b000 ... (65 %)  
Writing at 0x0000c0000 ... (67 %)  
Writing at 0x00004000 ... (68 %)  
Writing at 0x00008000 ... (70 %)  
Writing at 0x0000cc000 ... (71 %)  
Writing at 0x0000d000 ... (73 %)  
Writing at 0x0000d4000 ... (74 %)  
Writing at 0x0000d8000 ... (76 %)  
Writing at 0x0000dc000 ... (77 %)  
Writing at 0x0000e0000 ... (79 %)  
Writing at 0x0000e4000 ... (80 %)  
Writing at 0x0000e8000 ... (82 %)  
Writing at 0x0000ec000 ... (83 %)  
Writing at 0x0000f0000 ... (85 %)  
Writing at 0x0000f4000 ... (86 %)  
Writing at 0x0000f8000 ... (88 %)  
Writing at 0x0000fc000 ... (89 %)  
Writing at 0x00100000 ... (91 %)  
Writing at 0x00104000 ... (92 %)  
Writing at 0x00108000 ... (94 %)  
Writing at 0x0010c000 ... (95 %)  
Writing at 0x00110000 ... (97 %)  
Writing at 0x00114000 ... (98 %)  
Writing at 0x00118000 ... (100 %)  
Wrote 1798624 bytes (1093259 compressed) at 0x00010000 in 15.7 seconds (effective 915.5 kbit/s)...  
Hash of data verified.  
Compressed 3072 bytes to 129...  
Writing at 0x00008000 ... (100 %)  
Wrote 3072 bytes (129 compressed) at 0x00008000 in 0.0 seconds (effective 1489.7 kbit/s)...  
Hash of data verified.  
  
Leaving...  
Hard resetting via RTS pin...
```

40

ESP32 Dev Module on /dev/cu.usbserial-D30993JK

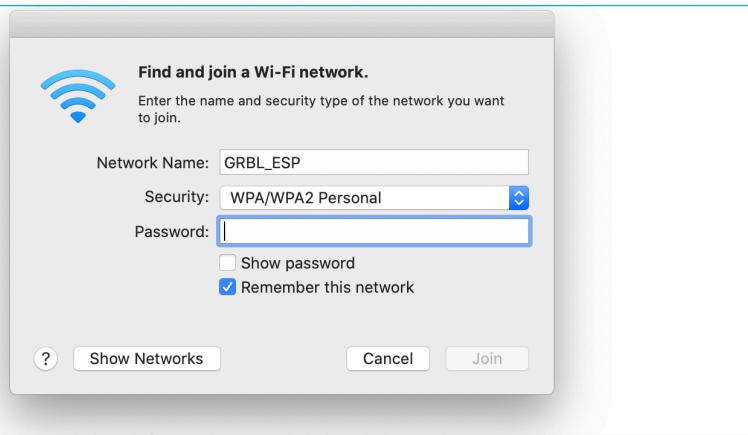
11 - Disconnect the board.

12- Change the switch to Execution Mode



13 - Check for the ESP32 wifi connection. Select Wificonfig.h file to see the Wifi connection details or check screenshot here:

```
51 #define ESP_SAVE_ONLY 0
52 #define ESP_APPLY_NOW 1
53
54 //defaults values
55 #define DEFAULT_HOSTNAME "grblesp"
56 #ifdef CONNECT_TO_SSID
57     #define DEFAULT_STA_SSID CONNECT_TO_SSID
58     #define DEFAULT_STA_PWD SSID_PASSWORD
59 #else //!CONNECT_TO_SSID
60     #define DEFAULT_STA_SSID "GRBL_ESP"
61     #define DEFAULT_STA_PWD "12345678"
62 #endif //CONNECT_TO_SSID
63 #define DEFAULT_STA_IP "0.0.0.0"
64 #define DEFAULT_STA_GW "0.0.0.0"
65 #define DEFAULT_STA_MK "0.0.0.0"
66 #define DEFAULT_AP_SSID "GRBL_ESP"
67 #define DEFAULT_AP_PWD "12345678"
68 #define DEFAULT_AP_IP "192.168.0.1"
69 #define DEFAULT_AP_MK "255.255.255.0"
70 #define DEFAULT_AP_CHANNEL 1
71 #define DEFAULT_WEBSERVER_PORT 80
72 #define DEFAULT_HTTP_STATE 1
```



If you try to do it on programming mode, it won't work:



14 — This is the page that pops-up on MacOS (also probably on Windows, too). It will show you the main website of the board. As it's not a fully functional browser, close it.

Using your browser of preference, visit 192.168.0.1. Make sure you are still connected to the wireless network GRBL_ESP.

Join "GRBL_ESP"

Firmware Interface Help v1.1f (20200412)

File index.html.gz is missing, please upload it

Flash Filesystem

Choose Files no files selected Upload

Refresh /

Type	Name	Size	Time
------	------	------	------

Status: Ok | Total space: 169.38 KB | Used space: 0 B | Occupation: 0%

< > captive.apple.com Cancel

15- We need to upload the index.html.gz file from the data folder that was on the firmware files. Thanks to Tue and David from 2020 Barcelona Fabacademy who had an awesome documentation: <http://academany.fabcloud.io/fabacademy/2020/>

labs/barcelona/barcelona-machines/light-machine/

Firmware Interface Help

v1.1f (20200412)

File index.html.gz is missing, please upload it

Flash Filesystem

Choose Files No file chosen Upload

Refresh /

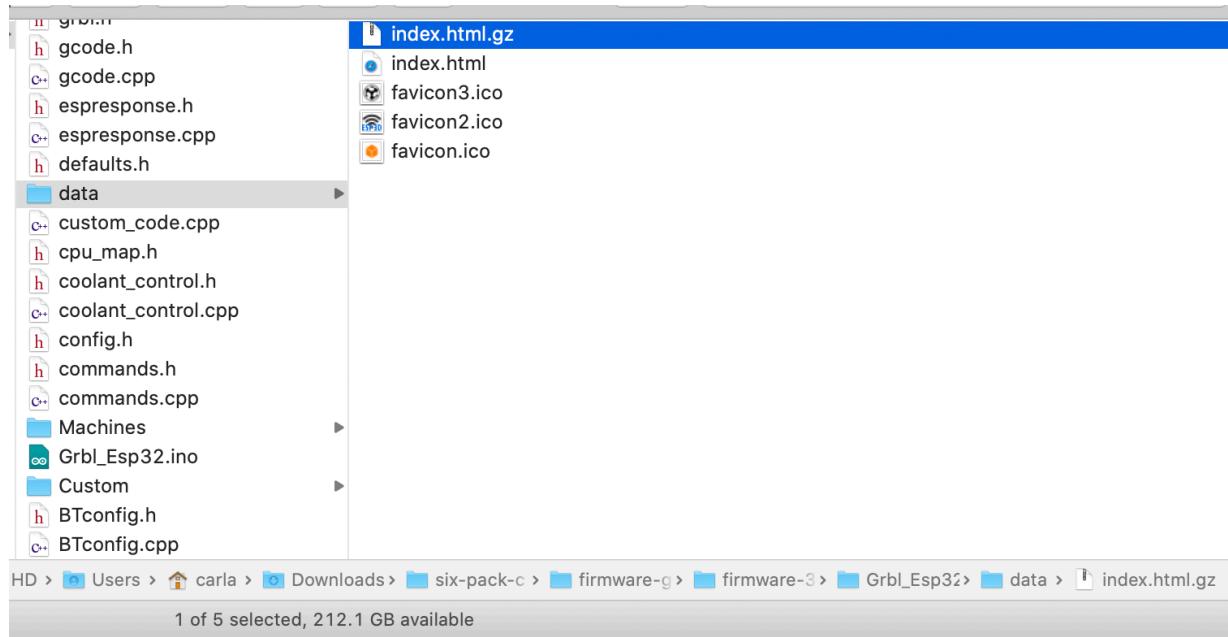
Type Name Size Time

Status: Ok | Total space: 169.38 KB | Used space: 0 B | Occupation: 0%

Firmware Update

Choose File No file chosen Update

After finding the file → Go to 192.168.0.1



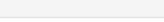
File index.html.gz is missing, please upload it

Flash Filesystem

[Choose Files](#) index.html.gz Uploading... 

[Refresh](#)  /

Type Name Size Time

Status: Ok | Total space: 169.38 KB | Used space: 0 B | Occupation:  0%

Firmware Update

[Choose File](#) No file chosen [Update](#)

It will say Uploaded 100% and a new button will appear "Go to ESP3D interface". Click on it and it will take you to the motor control interface.

[⚙ Firmware](#) [🌐 Interface](#) [ℹ Help](#) v1.1f (20200412)

[Go to ESP3D interface](#)

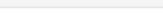
Flash Filesystem

[Choose Files](#) No file chosen [Upload](#)

[Refresh](#)  /

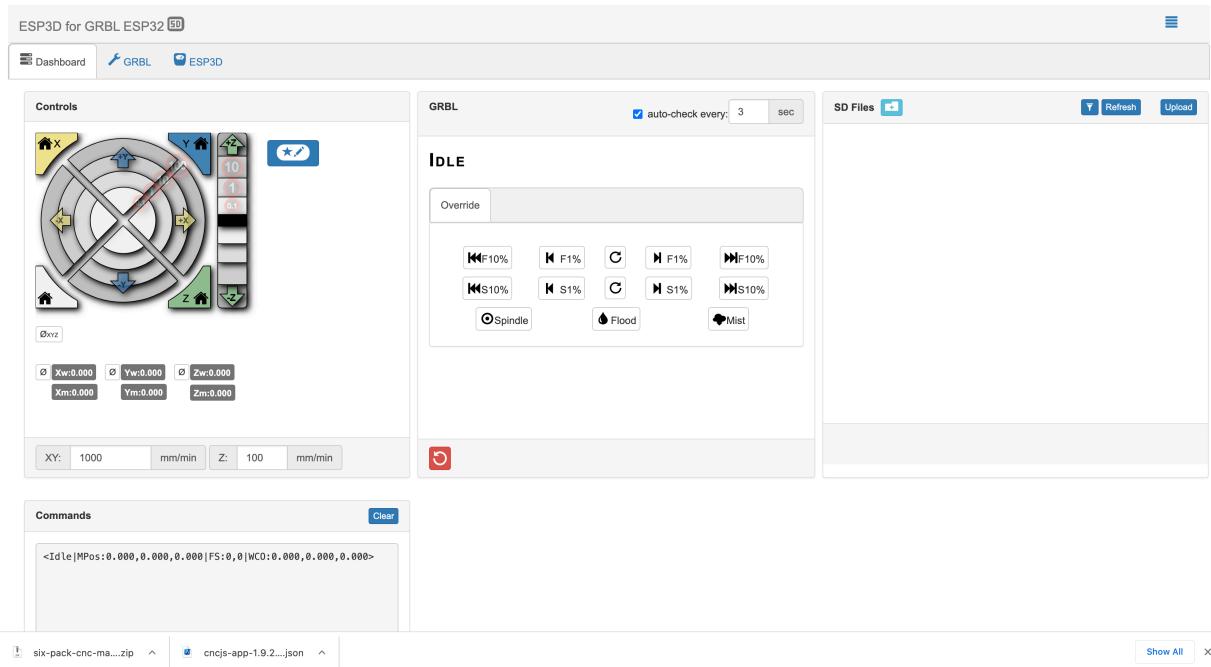
Type Name Size Time

	index.html.gz	125.56 KB	
-------------------------------------------------------------------------------------	---------------	-----------	-------------------------------------------------------------------------------------

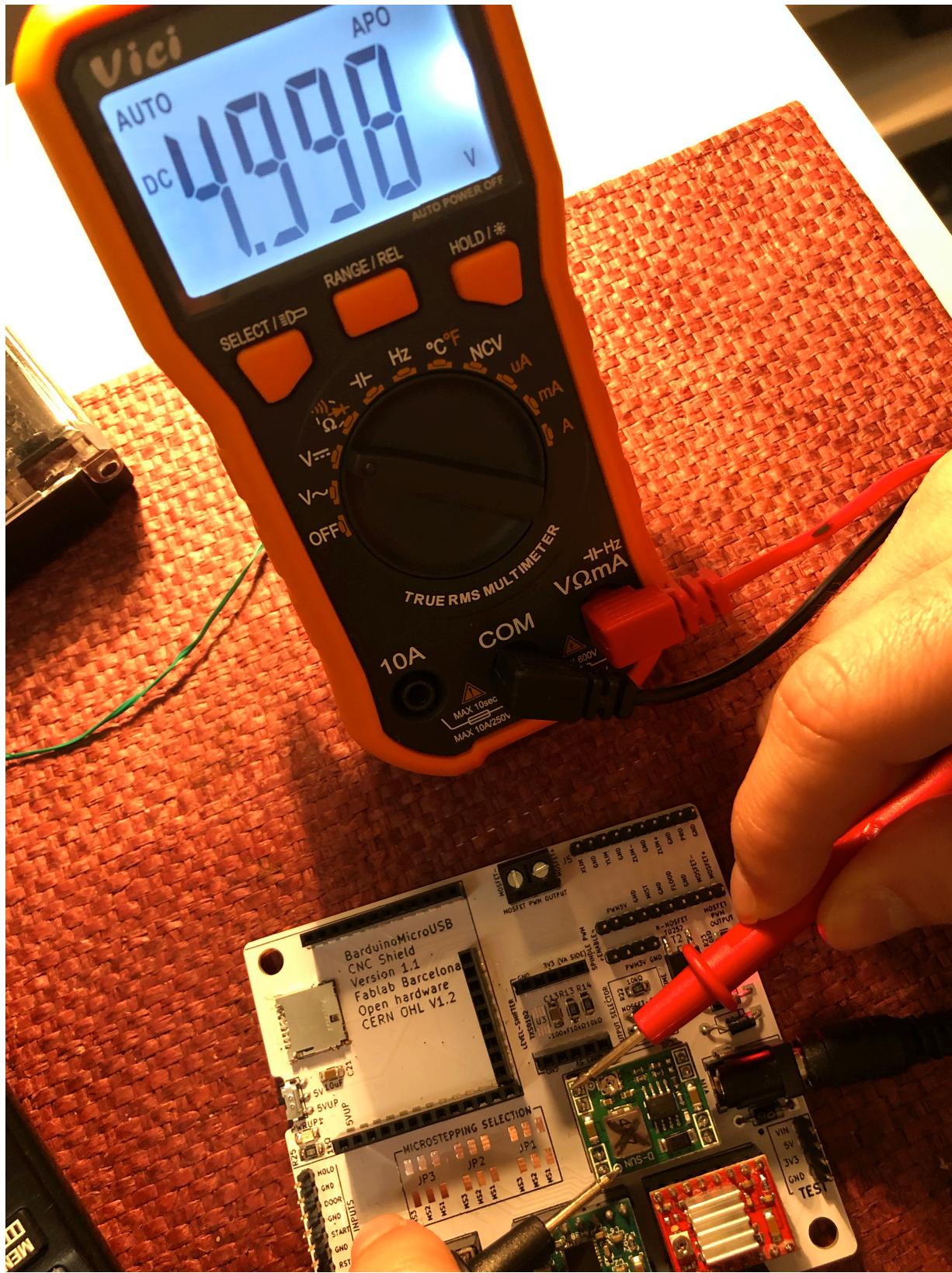
Status: Ok | Total space: 169.38 KB | Used space: 126.97 KB | Occupation:  74%

Firmware Update

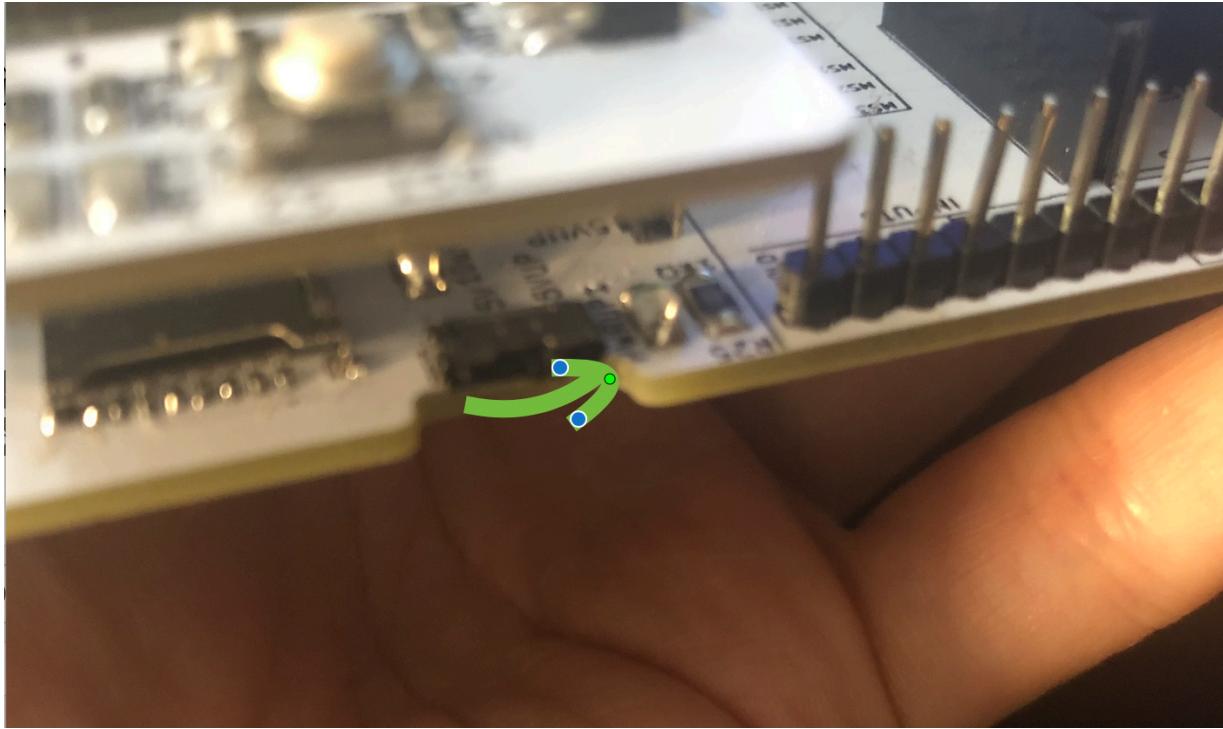
[Choose File](#) No file chosen [Update](#)



16 — Check the Voltage of the Voltage regulator.



17 - Now you can connect the Barduino to the driver board. Move the switch under the board to 5VUP before mounting the Barduino to the header pins.



18 - WARNING - Time to setup the current limit for each motor driver —> Watch this video and come back

<https://www.youtube.com/watch?v=89BHS9hfSUK>

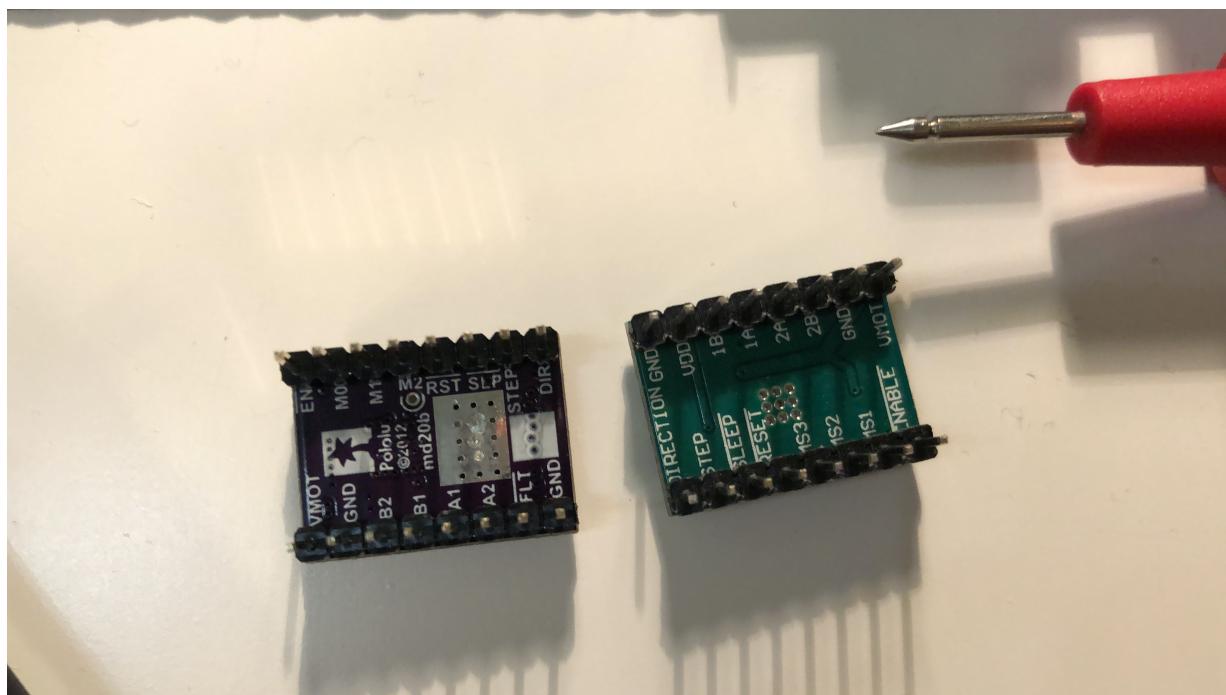
As you'll see you need to check the specs for your motor, meaning NEMA 11, 14, & 17 might have different specs. If we setup each motor driver according to its motor they won't be interchangeable (unless it's the same model) or the motor could get damaged

16A Purple one - Polulu driver DRV8825

The other two drivers look like knockouts of the other polulu models. I couldn't find the specs

DRV8825	DRV8834	DRV8880	MP6500	TB67S279FTG	TB67S249FTG	STSPIN820	STSPIN220
8.2 V 45 V	2.5 V 10.8 V	6.5 V 45 V	4.5 V 35 V	10 V 47 V	10 V 47 V	7 V 45 V	1.8 V 10 V
1 A 2.2 A	1.5 A 2 A	1 A 1.6 A	1.5 A 2.5 A	1.1 A 2 A	1.6 A 4.5 A	0.9 A 1.5 A	1.1 A 1.3 A
1/32 4	1/32 4	1/16 4	1/8 4	1/32 4	1/32 4	1/256 4	1/256 4
high max voltage, high current	low-voltage operation, high current	AutoTune, digital current reduction, high max voltage	high current	digital current control, high current	Auto Gain Control, ADMD, high max voltage	Auto Gain Control, ADMD, high max voltage, high current	128 and 256 microsteps, high max voltage 64, 128, and 256 microsteps, low-voltage operation
49	\$8.95	\$5.95	\$6.95	\$5.95	\$5.95	\$7.75	\$9.95
Additional cooling required (additional cooling required)							

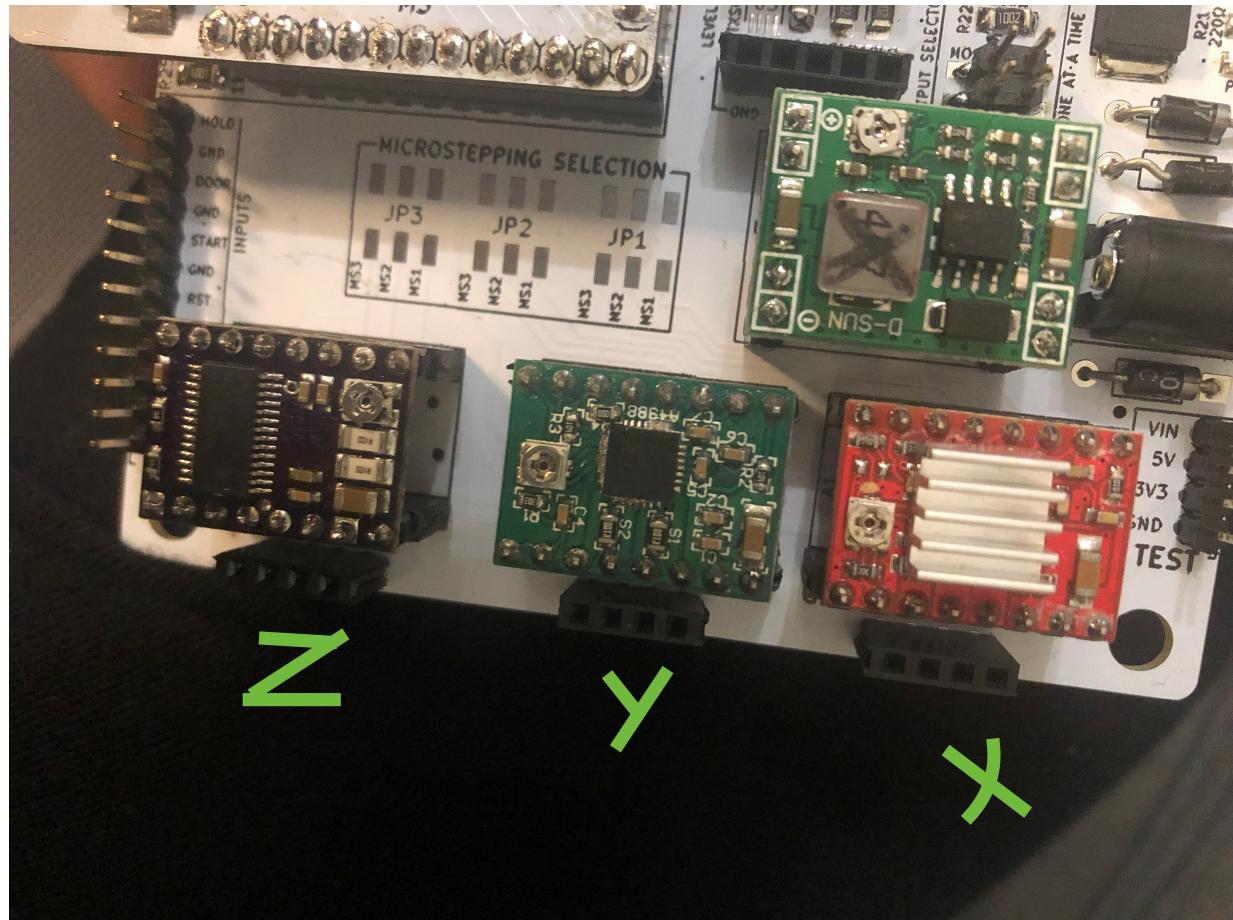
Not all have the same pinout orientation. → Check for GND to match the shield .



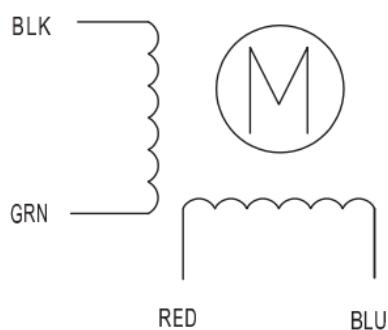
For the NEMA 14- SY35ST36 → The Voltage measured from the screw or VREF to GND needs to be around 500mV.

<https://www.pololu.com/file/0J689/SY35ST36-1004A.pdf>

19- Time to connect the motor- Make sure that the power supply is disconnected.



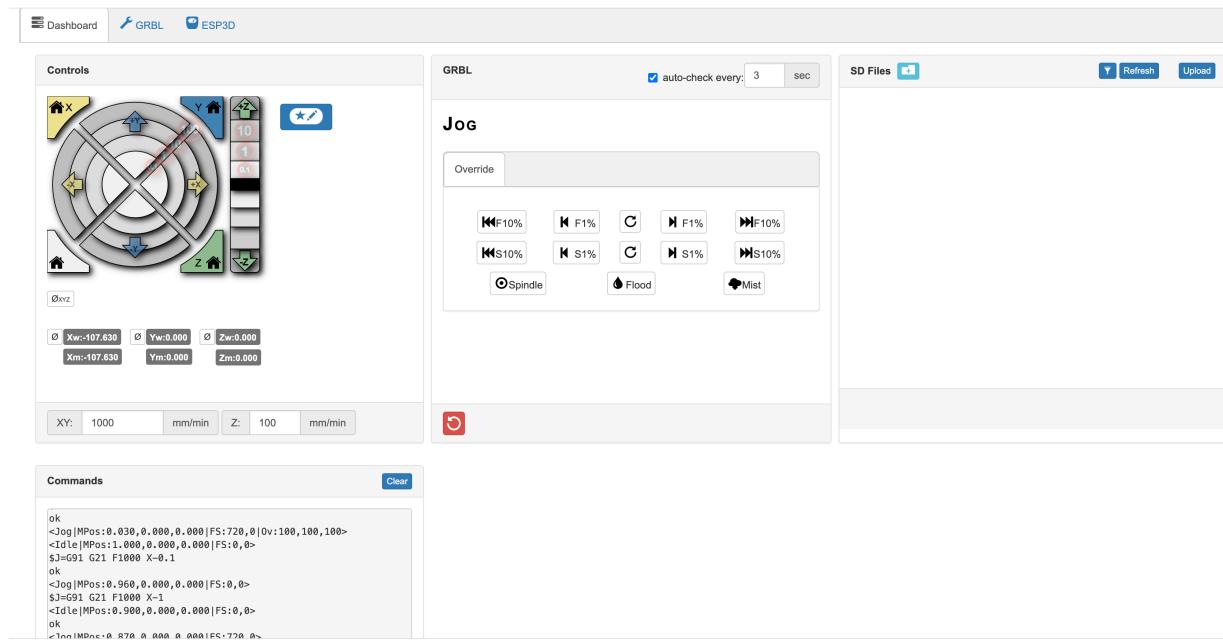
● Wiring Diagram:



● Dimensions:

I connected the wires according the wiring diagram on the motor's datasheet —> Black, green, red and blue.

20- Time to connect to the GRBL-ESP <http://192.168.0.1/>



When sending the commands the motors moves. Not super consistently, as it seems that wiring is a bit flickery. Wiring should be checked, everything else seems to be working.