Machine Week:
Arduino CNC Shield Kit with GRBL

Installation and Setup

Useful Links

Amazon page for Longruner Arduino CNC shield kit:
https://www.amazon.com/Longruner-Arduino-Professional-Mechanical-LKB02/dp/B072N4FMRN

Setup instructions for this kit (read this!):
https://www.makerstore.com.au/download/publications/CNC-Shield-Guide-v1.0.pdf

“Pre-Flight Checklist” for operating boards like this:
https://blog.protoneer.co.nz/arduino-cnc-shield-v3-00-assembly-guide/

GRBL command quick reference:
https://www.sainsmart.com/blogs/news/grbl-v1-1-quick-reference
G-code reference:
https://reprap.org/wiki/G-code

Rather exhaustive Youtube video that covers a lot of the basics:
https://www.youtube.com/watch?v=K0XfRPi_h2M
https://www.youtube.com/watch?v=J4bMAIEFCYU

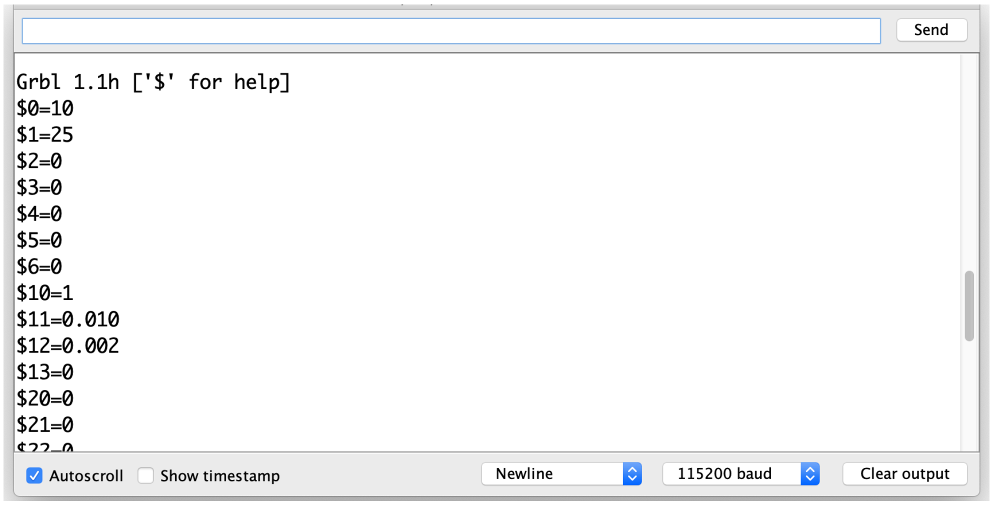
Universal Gcode Sender: a useful way to send Gcode commands with a graphical interface instead of cryptic text codes:
https://winder.github.io/ugs_website/

Overall Setup Plan

- The stepper motors need to be sent a series of current pulses, each pulse moves them a fraction of a rotation.
- The CNC shield contains the high-power electronics to send these pulses.
- The Arduino tells the CNC shield when to send pulses.
- The Arduino has a program on it called “GRBL”, that receives commands from the serial port and sends pulses to move the motors however you want.
- You can type these commands directly from the Arduino IDE’s serial monitor, or send them from a CNC control program on a PC, or send them from another Arduino or other microcontroller.

Software setup

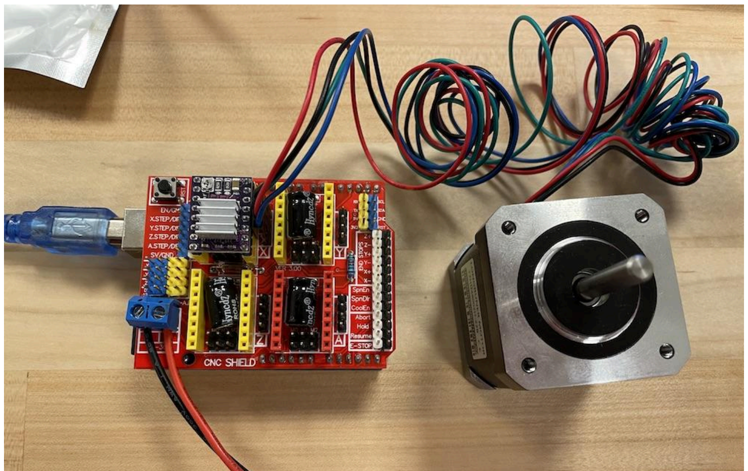
1. Connect CNC shield to Arduino, plug in Arduino to computer
2. Download GRBL controller for Arduino: https://github.com/gnea/grbl and unzip if it's a zip file
3. There are several nested folders in this download, frustratingly all called "grbl". You want the one that has a bunch of c files and an "examples" folder inside it. Copy this folder into your Documents / Arduino / libraries folder.
4. Open Arduino
5. Open Examples / grbl / grblUpload . If you don't see this example, you messed up step 3.
6. Click Upload button
7. Open Serial Monitor in Arduino, set 115200 baud, "Newline" line ending. You should see "`Grbl 1.1h [`$` for help]`'. If not check baud rate and serial port etc.
8. Type "?" and hit return. The GRBL controller should respond with something its status, something like "`<Idle|MPos:0.000,0.000,0.000|FS:0,0>`"
9. Type "`$$`" and hit return. The GRBL controller will respond with its current settings. This also makes sure you got the "Newline" setting right.



```
Grbl 1.1h ['$' for help]
$0=10
$1=25
$2=0
$3=0
$4=0
$5=0
$6=0
$10=1
$11=0.010
$12=0.002
$13=0
$20=0
$21=0
$22=0
```

Hardware Setup

1. Unplug everything. **EVERYTHING**.
2. Carefully read all the directions here before you start. Pay attention to safety warnings! https://www.makerstore.com.au/download/publications/CNC-Shield-Guide-v1.0.pdf
3. Plug the Pololu stepper drivers (little purple chips) into the top of the board in the X,Y,Z slots. Make sure the "EN" enable pin matches the "EN" marking on the board. <u>If you put them in backwards you'll ruin them!</u>



4. Stick the heat sinks (aluminum fins) onto the top of the chips. The heat sinks are not optional!
5. Plug a stepper motor into each of the X,Y,Z headers next to the stepper drivers. Don't run a driver without a motor plugged in!
6. Don't be stupid like me and plug the steppers into the X,Y,Z sockets next to the USB port, wasting two hours of your life with an oscilloscope to figure out why it isn't working, when you just plugged it in the wrong place.
7. Plug the Arduino into your computer. Or the computer of someone you don't like very much.
8. Attach the motor power: I have provided a 24V, 2A power supply and a barrel jack connector. The board and motors can handle more, but this should be plenty. Slide the wires into the side of the screw terminal block and screw them down tight. Make sure red = positive, black = negative, <u>if you get this backwards you'll ruin the whole board</u>.

Hardware Testing

1. Open Arduino program and start Serial Monitor as before.
2. Type "?" and hit return to make sure the GRBL controller is responding, as before.
3. Type "`F50`" and hit return: this sets the "feed rate" to 50 units per minute. GRBL should reply "`ok`". (What's a "unit"? See below)
4. Type "`G01 X5`" ("Advance the X-motor at the feed rate to position 5"). The X-motor should turn for a bit then stop.

You can test the other motors too, with "`G01 Y5`" etc.

About G-code units

When the stepper turns, it moves something in the real world. A 3d printer's print head might move a few millimeters, for instance. Your G-code commands refer to this physical distance: it might be in units of millimeters, or inches, or revolutions or degrees of angle of a wheel, it's up to you. The GRBL controller has a setting to determine how many steps the motor must turn to move one "unit" in the real world.

`$100`:	X-steps needed to move 1 unit
`$101`:	Y-steps needed to move 1 unit
`$102`:	Z-steps needed to move 1 unit

So for example, our stepper motor has 200 steps per revolution. So if we type

```
$100=200
```

Then when we give the commands "`G92`" (Reset zero position) "`G01 X1`" (Move to x=1), the X-motor will turn exactly 1 revolution. If we had set `$100=100`, then the same command would rotate the X-motor half a revolution.

About microstepping

Each full step advances the stepper motors by 1/200 of a revolution. This is kind of a big distance, so the motion can be jerky, causing lots of vibration. To get smoother motion with finer control, the stepper drivers can do "microstepping", advancing by a fraction of a full step every time. To set this up, you must insert "jumpers" onto the headers underneath the drivers on the CNC shield. See page 8-9 of the CNC shield guide for details.

You'll have to change the "steps per unit" setting to account for this. For example, if you enable ½-step microstepping for the X-axis, then you'll need to make the `$100` setting twice as big.

Some useful G-code commands and examples

Commands:

`F50`	Set feed rate to 50 units/minutex
`G1 X5`	Move X axis to position 5 at feed rate
`G1 Y3`	Move Y axis to position 3 at feed rate
`G0 Z6`	RAPID move Z axis to position 6, ignoring feed rate. How rapid? See $110 below
`G92`	Set the current position to be X=0, Y=0, Z=0.
`!`	Immediately stop all movement.
`~`	Resume after stop
`?`	Show current position and status
`$$`	Show current settings

Settings:

`$100=200`	Set X-axis to 200 steps per unit. ($101 and $102, same for Y and Z axes)
`$110=100`	Set maximum X movement rate to 100 units/minute ($111 and $112, for Y and Z)

Appendix: controlling a servo with GRBL

This section describes how to control a servo motor with GRBL. This is not needed for a standard stepper-motor setup, but servos are useful for things like plotters, where you don't need precise control for raising and lowering the pen.

Follow all the steps above, with the following changes:

In "Software Setup", the file you should download is https://github.com/cprezzi/grbl-servo

Connect the servo to the CNC shield as follows:

- Ground (brown/black wire) to the GND pin on the top right of the board.
- Power (red wire) to the +5V pin on the top right of the board.
- Signal (orange or white wire) to the "End Stop Z+" pin on the top right of the board.

In this installation, the servo is controlled by sending “Spindle” commands. These usually start and stop the cutting motor on a CNC machine, but here they move the servo.

Test: Send “S255M3”: servo should move to the right. Send “M5”: servo should move left.

Servo commands:

- M3: “Spindle on”: servo goes right (distance is set by S-commmand)
- M5: “Spindle off”: servo goes full left deflection.
- S: “Spindle speed”: controls max travel of servo. Send S255 to get full travel, S128 is halfway etc.