Compiling Grbl

Joacim Breiler edited this page on May 29, 2023 · 59 revisions

This wiki is intended to provide various instructions on how to compile grbl. Once compiled, you should have a brand new .hex file to flash to your Arduino. Please feel free to contribute more up-to-date or alternative methods.

Via the Arduino IDE (All Platforms): Recommended for all users.

Thanks to the great people working on the Arduino IDE, it has everything you need to compile grbl included in their software package. This method compiles the Grbl source code and automatically uploads it to an Arduino. You can't directly flash a pre-compiled .hex file through the IDE interface. See our Flashing Grbl to an Arduino wiki page for how to do this if you only have a .hex file.

NOTE: Before starting, delete prior Grbl library installations. Otherwise, you'll have compiling issues! On a Mac, Arduino libraries are located in `~/Documents/Arduino/libraries/`. *On Windows, it's in* `My Documents\Arduino\libraries`. *On Linux (Ubuntu), it's in* `/usr/share/arduino/libraries` If you cannot find the grbl library in order to delete and reinstall, check the sketchbook location in the IDE via `file -> preferences` and view `/sketchbook/location/path` at the top.

If you are using Arduino IDE 1.x

1. Download the Grbl source code.
   - Click on the `<>Code` Tab
   - Click the Clone or download button on the Grbl home page.
   - Click the `Download ZIP`
   - Unzip the download and you'll have a folder called `grbl-XXX`, where `XXX` is the release version.
2. Launch the Arduino IDE
   - Make sure you are using the most recent version of the Arduino IDE!
3. Load the `grbl folder` into the Arduino IDE as a Library.
   IMPORTANT: Select the `grbl` folder *inside* the `grbl-XXX` folder, which only contains the source files and an example directory. If you accidentally select the `.zip` file or the wrong folder, you will need to navigate to your Arduino library, delete the grbl library, and re-do Step 3.
   - Click the `Sketch` drop-down menu, navigate to `Include Library` and select `Add .ZIP Library`. The `Add .ZIP Library` command supports both a .ZIP file or a folder. In our case, there is no `.ZIP` file.
   - You can confirm that the library has been added. Click the `Sketch` drop-down menu again, navigate to `Include Library`, then scroll to the bottom of the list where you should see `grbl`.
4. Open the `GrblUpload` Arduino example.
   - Click the `File` drop-down menu, navigate to `Examples->Grbl`, and select `GrblUpload`.
   - Do not alter this example in any way! Grbl does not use any Arduino code. Altering this example may cause the Arduino IDE to reference Arduino code and compiling will fail.
5. Compile and upload Grbl to your Arduino.
   - Connect your Arduino Uno to your computer.
   - Make sure your board is set to the Arduino Uno in the `Tool->Board` menu and the serial port is selected correctly in `Tool->Serial Port`. (There are some controller boards on ebay that have the Arduino Pro bootloader on it, if you get error messages like "avrdude: stk500_getsync() attempt n of 10: not in sync: resp=0x20" then choose another board, try Arudino Pro/Pro Mini)
   - Click the `Upload`, and Grbl should compile and flash to your Arduino! (Flashing with a programmer also works by using the `Upload Using Programmer` menu command.)

Clone this wiki locally

https://github.com/gnea/grbl.wil

NOTE: If your environment is clean and Arduino IDE compiler still throws "warning: [...] redefined" messages, you may need to *uncheck* File -> Preferences -> "Aggressively cache compiled core".

No fuss! No muss!

NOTE: If you are having upload issues, try re-burning the Arduino bootloader. If you have a spare Arduino, it's easy!

Last updated: 2018-03-04

If you are using Arduino IDE 2.x

1. Click on the <> Code tab.
2. Click the Clone or download button on the Grbl home page.
3. Select "Download ZIP" from the menu.
4. Wait for the download to finish.
5. Unzip the downloaded `grbl-master.zip` file.
6. Copy the `grbl` subfolder of the unzipped folder to the `libraries` subfolder of your Arduino sketchbook folder.
   ⓘ If you don't know the location of the sketchbook folder, you can find it by opening the Arduino IDE preferences (File > Preferences) and looking at the path shown in the "Sketchbook location" field there.
   The correct installation structure must look like this:

```
<Sketchbook location>/
├── libraries/
│   └── grbl/
│       ├── examples/
│       │   └── ...
│       ├── grbl.h
│       └── ...
└── ...
```

7. Launch Arduino IDE.
   ❗ Make sure you are using the most recent version of Arduino IDE!
8. Select File > Examples > grbl > grblUpload from the Arduino IDE menus.
   ⚠ Do not alter this example in any way! Grbl does not use any Arduino code. Altering this example may cause compilation to fail.
9. Connect your Arduino Uno to your computer.
10. Select Tools > Board > Arduino AVR Boards > Arduino Uno from the Arduino IDE menus.
11. Select the port of your Uno from the Tools > Port menu.
12. Select Sketch > Upload from the Arduino IDE menus.

Advanced usage

Most users are just fine with Grbl's default build, but you can customize Grbl by editing the `config.h` file.

⚠ Make sure you are editing the file at `<Sketchbook location>/libraries/grbl/config.h`, not the file in the folder you copied the `grbl` folder from.

`config.h` enables or disables all of Grbl's additional compile-time options. There are comments in the file that explain what they all do. Once edited and saved, just follow the steps above to flash your custom Grbl build!

The following methods are for reference only.

Build as a Normal Arduino IDE project:

Per https://github.com/gnea/grbl/pull/716 and the discussion at https://github.com/gnea/grbl/issues/715 you can create an empty file `grbl.ino` in the grbl directory and compile as normal.

1. Download and unpack grbl from https://github.com/gnea/grbl (e.g. `git clone git@github.com:gnea/grbl.git`)
2. Create an empty `grbl.ino` file. (e.g. `touch path_to_grbl/grbl/grbl.ino`)
3. Open the grbl.ino file in the Arduino IDE and compile as normal.

An advantage of this method is you can edit distinct copies of the files without risking a common library.

For Mac OS X:

Last updated: 2012-01-29 by chamnit. (Tested on OS X 10.7, 10.6, 10.4 and the Arduino IDE r22,v1.0)

This method of compiling Grbl uses the Mac OSX terminal and command line to access the Arduino IDE's compilers without having to use the Arduino IDE. This produces the same firmware as the Arduino IDE method above.

First, you'll need to make sure you have the most up-to-date Arduino IDE version installed on your Mac. The trickiest part is setting up the environment path for the compilers included in the Arduino software. To do this, you'll need to first locate where they are. Depending on where you place your Arduino.app software, this will usually be located in */Applications/Arduino.app* for most people. The complete path is then: `/Applications/Arduino.app/Contents/Java/hardware/tools/avr/bin/`

To add the compiler path: Open the Terminal.app in /Applications/Utilities.

Then type: `nano ~/.bashrc` to edit your shell config file.

Now add this line at the end of the file: `export PATH=$PATH:/Applications/Arduino.app/Contents/Java/hardware/tools/avr/bin/` or whatever your path happens to be.

Press *Crtl-X* to exit and select *Yes* to save the file. Now you have added the compiler path. You will need to close the current working window and re-open a new one for the path to be loaded correctly.

NOTE: If you are having problems, you may need to add this same PATH to your .bash_profile file. The process is exactly the same, just switch out the names.

To compile: Once your paths are setup, all you will need to do is go to your grbl directory and type `make`. (To clear all of the old compilation files from a previous build, type `make clean` first.) This should call *avr-gcc*, begin compiling grbl, and create a brand new firmware file called *grbl.hex* that may then be flashed to your Arduino.

For Windows:

Last updated: 2018-08-11 by chamnit. (Tested on Windows 10 and the Arduino IDE 1.8.5)

You can use the Arduino platform as well since it comes with most of the stuff you need to compile Grbl through a command prompt interface.

1. First download the Arduino IDE Windows installer from arduino.cc. Do not use the Windows App store version, as it hides these files.

2. Open the Arduino IDE and navigate to the 'Board Manager' in the the 'Tools'/'Board: "XXX"' drop-down menu.
3. In the 'Board Manager', install the 'Arduino SAM Boards' (for the Arduino Due). This particular library contains the 'make' executable that Grbl needs to compile via Makefile.
4. Search in the Windows search bar for 'Environment Variables'. An option 'Edit the system environment variables' in the control panel should be the first option in the list. Select it.
5. A 'System Properties' windows should appear. Click the 'Environment Variables' button at the bottom of the window.
6. Highlight the 'Path' Variable under the 'User variables' and click the 'Edit' button.
7. Click the 'New' button and add these paths. Note that you may need to update the sam library version below.
    - `C:\Program Files (x86)\Arduino\hardware\tools\avr\avr\bin\`
    - `C:\Program Files (x86)\Arduino\hardware\tools\avr\bin\`
    - `%USERPROFILE%\AppData\Local\Arduino15\packages\arduino\hardware\sam\1.6.11\system\CMSIS\Examples\cmsis_example\gcc_atmel\`
    - `C:\Program Files (x86)\Arduino`
8. Click 'Ok' for windows and reboot the computer.
9. Once rebooted, open a windows command prompt (Search for 'command prompt'). Navigate to the Grbl source folder via 'cd' change directory commands. In the root of the Grbl source directory, where the Makefile is located, first type 'make clean' to wipe any old build files, then type 'make' and Grbl should compile.

An alternative is to use Atmel Studio, a customized version of Visual Studio.

Last update: 2014-07-18 by gerritv (tested on Windows 8.1, 64bit)

- Install Atmel Studio
- Install the Create From Makefile Extension (Tools/Extension Manager)
- run Tools/Create Project From Makefil
- select the Makefile from your grbl code directory
- Select Device, use ATmega328p for the Arduino Uno
- In Projects/Properties, uncheck Use External Makefile
- Add -DF_CPU=16000000 -mmcu=atmega328p to Project/Properties/Toolchain/AVR Gnu Compiler/Miscellaneous Other Flags

The last 2 steps need to be done for both Debug and Release configurations

Enjoy the benefits of Visual Studio for Atmel/AVR

For Linux:

Last updated: 2012-03-02 by speters. (Tested on ???)

Make sure you have the prerequisite libraries installed: *avr-gcc* and *arduino* (*sudo aptitude install arduino*)

At a terminal prompt, change directories to where the *grbl* source code located. Then type the following to compile and build the firmware:

```
make clean
make grbl.hex
```

For Arch Linux

Last updated: 2017-03-26 by brownjohnf. (Tested on 2017-03-06)

You may encounter an error about a missing `libtinfo.so.5`. I got grbl to compile and run using the following. WARNING: The following is a hack, and there are probably better solutions, but this was quick and has worked for me.

1. Install `libtinfo` from AUR with `yaourt -S libtinfo`
2. Check to see what you've got installed. In my case, it looked like the following, where you can see that `libtinfo.so.6` is installed, but not `5`.

```
$ ls -l /usr/lib | grep libtinfo
lrwxrwxrwx  1 root root       22 Mar 26 10:13 libtinfo.so -> /usr/lib/
libtinfo.so.6
lrwxrwxrwx  1 root root       27 Mar 26 10:13 libtinfo.so.6 -> /usr/lib/
libncursesw.so.6.0
```

3. You can get things working by symlinking `libncursesw.so.6.0` as `libtinfo.so.5`:

```
$ sudo ln -s /usr/lib/libncursesw.so.6.0  /usr/lib/libtinfo.so.5
```

For Ubuntu:

Last updated: 2021-05-08 by ooxi, based upon the work of EliteEng.

The following has been tested on Ubuntu 20.04 and an Arduino Uno. It will compile grbl from source code and flash it to your Arduino. It should in theory work with other flavours of Debian too.

On a brand new Ubuntu box, the install process goes like this:

1. install the avr build tools by running:

```
sudo apt-get install avrdude avr-libc gcc-avr make unzip
```

2. Compile the GRBL source code and create the firmware file:

```
cd /home ## or a location you want to download the source code to.
wget https://github.com/gnea/grbl/archive/master.zip
unzip master.zip
cd grbl-master
sudo make grbl.hex
```

3. To flash the firmware to your Arduino Uno, plug the Arduino in using the USB cable (Confirm that the device is located at /dev/ttyACM0 and run the following command:

```
sudo PROGRAMMER="-c arduino -P /dev/ttyACM0" make flash
```

That's it, the firmware should now be installed on your Arduino.

+ Add a custom footer