

# How to revive a broken Arduino using a Mac

These are instructions for doing this on a mac -- there are a few extra steps which might be a lot less hassle to do on windows but I haven't tried. If you're a student reading this, chances are you don't have an Atmel-ICE programmer to hand: if you're having Arduino trouble chat to one of us first to make sure this is the right solution to your issue.

## Get the right configuration of avrdude

Mac permissions mean that using the AVR ISP you get some weird hassle and it won't work properly in the IDE. You can get around this by rebuilding avrdude to make sure it includes a library called HIDAPI. I followed the steps in [this comment](#), which explains everything really well. I reproduced the steps here for clarity as a few things had changed / there were a couple of external instructions to follow.

round 1: HIDAPI setup

1. `brew install autoconf automake`
2. `git clone git://github.com/libusb/hidapi.git`
3. `cd hidapi` then `./bootstrap` (this should run fine)
4. run to get the configuration for your OS `./configure`
5. compile: `make`
6. install: `sudo make install`

round 2: avrdude rebuild (leave the hidapi folder to do this)

1. download latest version of avrdude: `git clone git@github.com:avrdudes/avrdude.git`
2. edit `avrdude/src/configure.ac` (NB: I'm not actually *sure* if you have to do this anymore you could try it without. But I did do this and it didn't cause any issues, so). You want to change this:

```
AH_TEMPLATE([HAVE_LIBHIDAPI],  
            [Define if HID support is enabled via libhidapi])  
AC_SEARCH_LIBS([hid_init], [hidapi hidapi-libusb hidapi-hidraw], [have_libhidapi=yes])
```

to this:

```
AH_TEMPLATE([HAVE_LIBHIDAPI],  
            [Define if HID support is enabled via libhidapi])
```

```
for _hidapi_lib in hidapi hidapi-hidraw hidapi-libusb; do
    PKG_CHECK_MODULES([hidapi],[$_hidapi_lib],[have_libhidapi=yes],[true])
done
AC_SEARCH_LIBS([hid_init], [hidapi hidapi-libusb hidapi-hidraw], [have_libhidapi=yes])
```

3. run `./build.sh`, and check that when it prints the Configuration Summary it managed to successfully find `libhidapi` (should look like this):

```
-- Configuration summary:
-- -----
-- DON'T HAVE libelf
-- DO HAVE   libusb
-- DO HAVE   libusb_1_0
-- DO HAVE   libhidapi
...
-- -----
```

4. final bit: run `sudo cmake --build build_darwin --target install` to install. Check the install locations it prints -- you care about where the main and the conf files are for later in the process:

```
-- Install configuration: "RelWithDebInfo"
-- Installing: /usr/local/bin/avrdude <- important
-- Installing: /usr/local/lib/libavrdude.a
-- Installing: /usr/local/include/libavrdude.h
-- Installing: /usr/local/etc/avrdude.conf <- important
-- Installing: /usr/local/share/man/man1/avrdude.1
```

Once you're at this step you're in a good place to use avrdude directly from the command line.

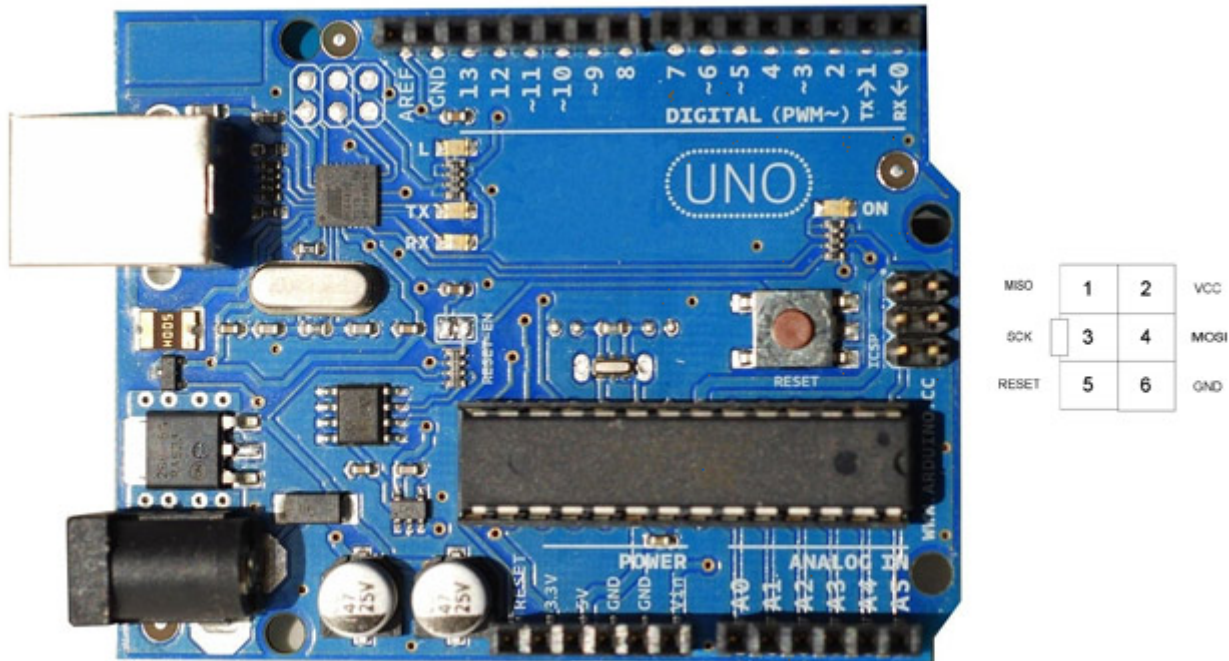
## Using avrdude to burn new bootloader to the arduino

afaik you don't actually need to do things to the fuses like the [Arduino bootloader instructions](#) suggest (and I have a horrible feeling that the 2 that ended up not working were the ones where I messed with the fuse settings). Instead, I just used the ATMEL-ICE tool to burn the hex file directly to the Arduino.

This is something of a shame as I did find this amazing [AVR fuse calculator website](#) and I wanted an excuse to show it to someone.

## Hardware setup

The Arduino needs to be externally powered (I used a USB wall supply to avoid confusing my computer) and the ISP header inserted the correct way round into the Arduino. The below image is for an Uno but it's still correct -- the little bump on the SCK pin faces toward the centre of the board.



Lastly, connect the ATMEL-ICE to the computer (I used USB-C to micro-USB)

## Finding and burning the bootloader

The Arduino IDE has a bunch of bootloaders saved locally, in the folder `/Applications/Arduino.app/Contents/Java/hardware/arduino/avr/bootloaders/`. You can also browse them online in the [Arduino Core AVR repository](#). For Leonardo, we want to use the Caterina-Leonardo bootloader.

The command that I ran was:

```
/usr/local/bin/avrdude -C /usr/local/etc/avrdude.conf -v -p atmega32u4 -c atmelice_isp -P usb -U  
flash:w:/Applications/Arduino.app/Contents/Java/hardware/arduino/avr/bootloaders/caterina/Caterina-Leonardo.hex:i
```

NB -- I'm pointing to the avrdude and conf files that are the specific ones that were installed in the previous step. By default your system installation of avrdude is probably not this one (but you can type `which avrdude` to check/reset). Note also that this configuration is for the Arduino Leonardo, using the Atmel ICE ISP. For an Uno, for example, you'd want `-p atmega328p` and to find what the current Uno bootloader is.

If you get the error:

```
avrdude stk500v2_command() error: command failed  
avrdude main() error: initialization failed, rc=-1
```

it's worth trying appending `-B 200kHz -F` to the avrdude like they suggest to remedy it, but if then you get the error:

```
avrdude: device signature = 0x000000  
avrdude main() error: Yikes! Invalid device signature.
```

I think ur arduino is borked beyond repair. RIP dude.

---

Revision #4

Created 20 December 2022 15:05:51 by agnes cameron

Updated 29 April 2024 22:24:58 by Tom Lynch