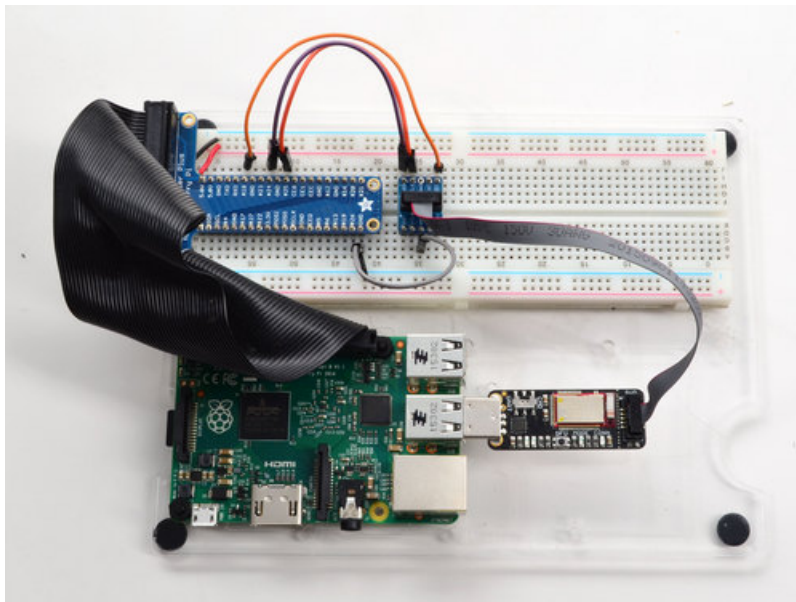


Programming Microcontrollers using OpenOCD on a Raspberry Pi

Created by lady ada

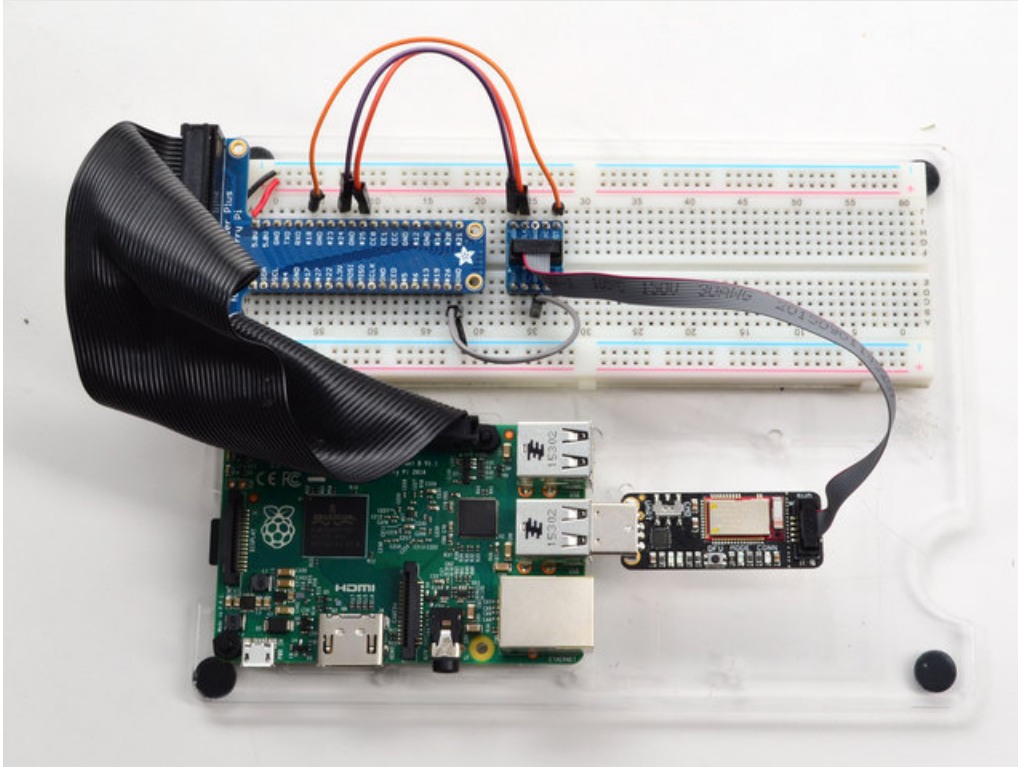


Last updated on 2018-08-22 03:53:00 PM UTC

Guide Contents

Guide Contents	2
Overview	3
OpenOCD	3
Compiling OpenOCD	4
Compiling OpenOCD	4
Wiring and Test	10
Connecting to Target	10
Wire up the target to SWD	10
Create OpenOCD config	11
More Options	16

Overview



Yay you have finally moved on from 8-bit chips and are ready to try out some 32-bit hotness! Those ARM Cortex chips look fun, some have built in bluetooth, or 2.4ghz radios, or usb...all you have to do is learn how to program them.

OpenOCD

On your way to learning how to use your favorite new ARM Cortex you may have heard of [OpenOCD \(https://adafruit.it/fMy\)](https://adafruit.it/fMy). OpenOCD is the software that we will use to do the actual programming of chips. Unlike the AVR ISP programming protocol, every ARM chip is significantly different to program, with platform-unique commands, flash locations, fuse bits, settings, etc. Teasing out those details is a struggle and if you change chips you have to start all over *even if both chips are, say, Cortex-M3 based!*

Each chip fab tends to supply its own programming software - Atmel has Atmel Studio, Nordic has NRFGGo, ST has ST Link - but often times that software is Windows only.

OpenOCD is great because its cross platform, open source, and has support for a vast number of chips & programmers.

You can use OpenOCD with dongle-programmers such as J-Link and ST-Link or even an FTDI chip. But, if you have a spare Raspberry Pi (and who doesn't these days?) you can use it as a native OpenOCD programmer with just a few wires.

It's also really *fast* to program chips natively, and if you have to program a mess of chips, it can make things speedy - an extra 30 seconds adds up when you're doing 1000!

Compiling OpenOCD

Compiling OpenOCD takes about 15 minutes but is worth the effort to get the latest code. You'll need to have command line access and a Pi on the Internet so you can download packages and software.

Thanks to <https://petervanhoyweghen.wordpress.com/2015/10/11/burning-zero-bootloader-with-beaglebone-as-swd-programmer/> (<https://adafru.it/mbC>) for the great tutorial, we're just adapting it for Pi usage!

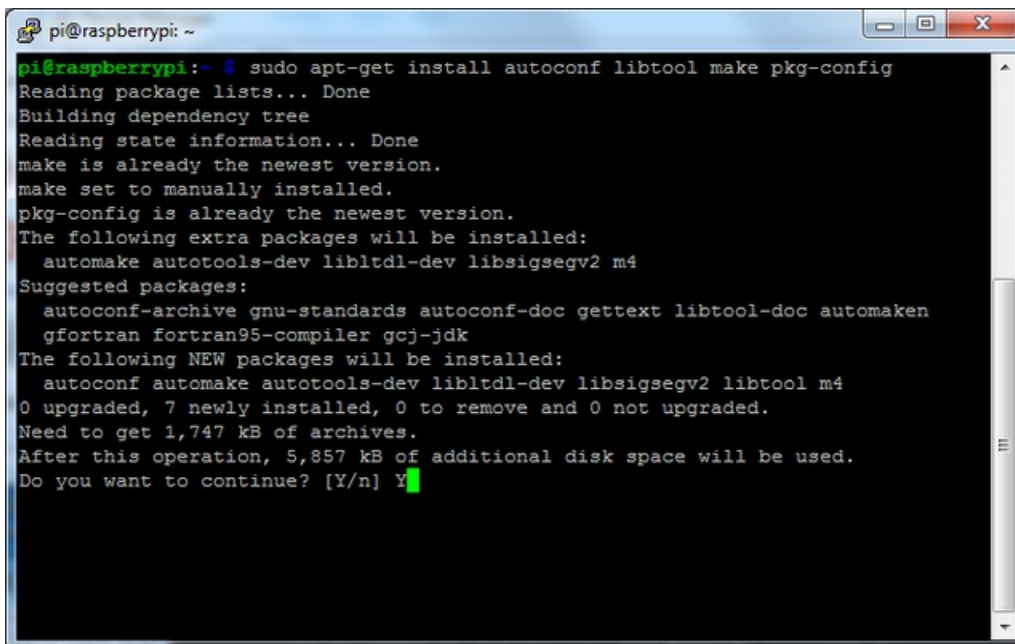
Compiling OpenOCD

Start by doing a fresh `sudo apt-get update` this will make sure you have the latest packages and repository set up.

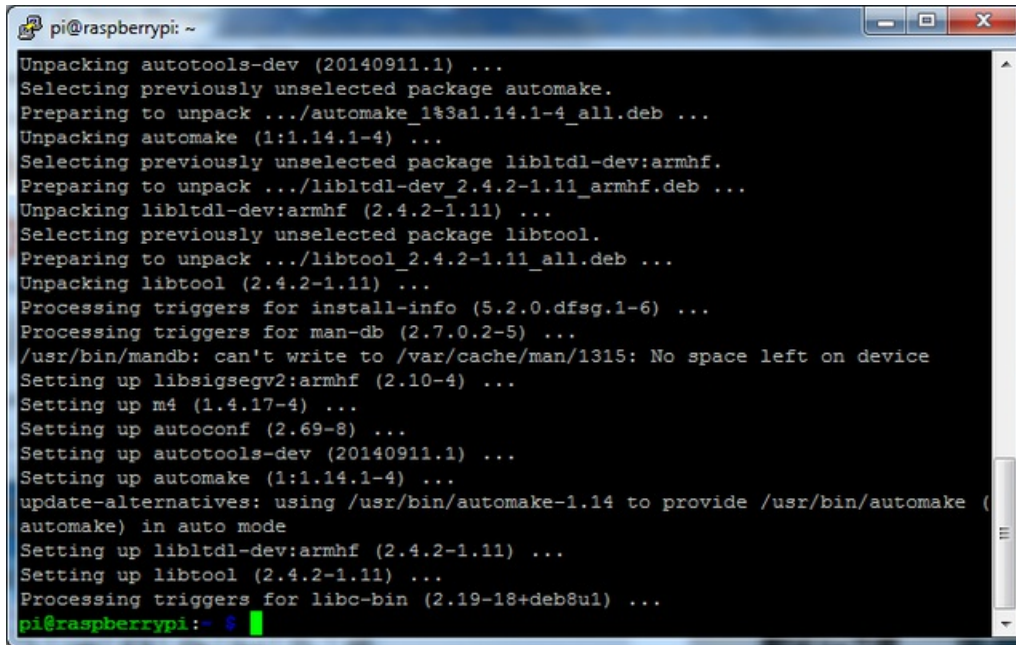
Next, run

```
sudo apt-get install git autoconf libtool make pkg-config libusb-1.0-0 libusb-1.0-0-dev
```

to install all the tools you'll need to compile OpenOCD. OpenOCD changes a lot and is under constant development so we do suggest compiling your own!



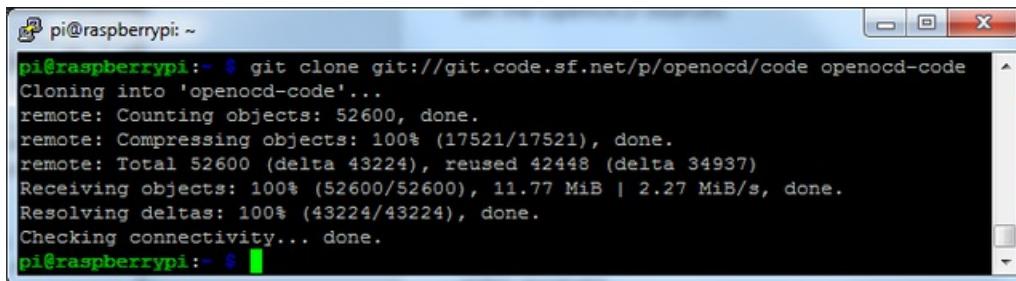
```
pi@raspberrypi: ~  
pi@raspberrypi:~$ sudo apt-get install autoconf libtool make pkg-config  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
make is already the newest version.  
make set to manually installed.  
pkg-config is already the newest version.  
The following extra packages will be installed:  
  automake autotools-dev libltdl-dev libsigsegv2 m4  
Suggested packages:  
  autoconf-archive gnu-standards autoconf-doc gettext libtool-doc automaken  
  gfortran fortran95-compiler gcj-jdk  
The following NEW packages will be installed:  
  autoconf automake autotools-dev libltdl-dev libsigsegv2 libtool m4  
0 upgraded, 7 newly installed, 0 to remove and 0 not upgraded.  
Need to get 1,747 kB of archives.  
After this operation, 5,857 kB of additional disk space will be used.  
Do you want to continue? [Y/n] Y
```

A terminal window titled 'pi@raspberrypi: ~' showing the output of an apt-get install command. The output lists the unpacking and setting up of several packages: autotools-dev, automake, libltdl-dev:armhf, and libtool. It also shows the processing of triggers for install-info, man-db, and libc-bin. A warning message indicates that /usr/bin/mandb cannot write to /var/cache/man/1315 due to no space left on the device. The prompt 'pi@raspberrypi:~\$' is visible at the bottom.

```
pi@raspberrypi: ~
Unpacking autotools-dev (20140911.1) ...
Selecting previously unselected package automake.
Preparing to unpack .../automake_1%3a1.14.1-4_all.deb ...
Unpacking automake (1:1.14.1-4) ...
Selecting previously unselected package libltdl-dev:armhf.
Preparing to unpack .../libltdl-dev_2.4.2-1.11_armhf.deb ...
Unpacking libltdl-dev:armhf (2.4.2-1.11) ...
Selecting previously unselected package libtool.
Preparing to unpack .../libtool_2.4.2-1.11_all.deb ...
Unpacking libtool (2.4.2-1.11) ...
Processing triggers for install-info (5.2.0.dfsg.1-6) ...
Processing triggers for man-db (2.7.0.2-5) ...
/usr/bin/mandb: can't write to /var/cache/man/1315: No space left on device
Setting up libsigsegv2:armhf (2.10-4) ...
Setting up m4 (1.4.17-4) ...
Setting up autoconf (2.69-8) ...
Setting up autotools-dev (20140911.1) ...
Setting up automake (1:1.14.1-4) ...
update-alternatives: using /usr/bin/automake-1.14 to provide /usr/bin/automake (
automake) in auto mode
Setting up libltdl-dev:armhf (2.4.2-1.11) ...
Setting up libtool (2.4.2-1.11) ...
Processing triggers for libc-bin (2.19-18+deb8u1) ...
pi@raspberrypi:~$
```

Download the latest source code for OpenOCD with

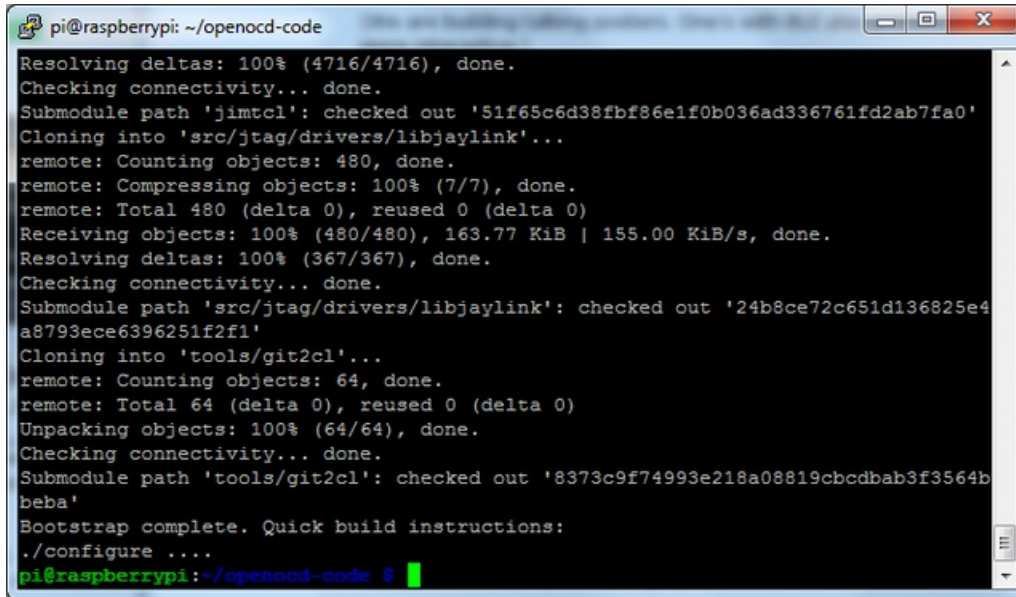
`git clone http://openocd.zylin.com/openocd`

A terminal window titled 'pi@raspberrypi: ~' showing the output of a git clone command. The output displays the progress of cloning the repository, including counting objects, compressing objects, and resolving deltas. The prompt 'pi@raspberrypi:~\$' is visible at the bottom.

```
pi@raspberrypi:~$ git clone git://git.code.sf.net/p/openocd/code openocd-code
Cloning into 'openocd-code'...
remote: Counting objects: 52600, done.
remote: Compressing objects: 100% (17521/17521), done.
remote: Total 52600 (delta 43224), reused 42448 (delta 34937)
Receiving objects: 100% (52600/52600), 11.77 MiB | 2.27 MiB/s, done.
Resolving deltas: 100% (43224/43224), done.
Checking connectivity... done.
pi@raspberrypi:~$
```

Change into the code directory and run the bootstrapper with:

`cd openocd-code`
`./bootstrap`

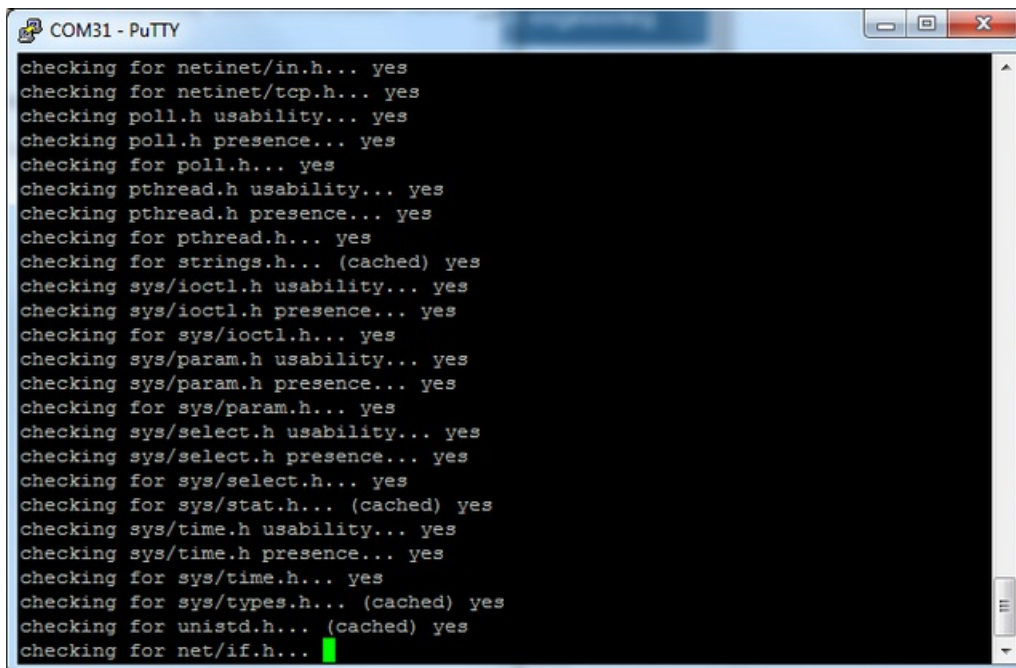
A terminal window titled 'pi@raspberrypi: ~/openocd-code' showing the output of the 'bootstrap' script. The script clones the 'jimtcl' and 'libjaylink' submodules, then clones the 'git2cl' tool. It reports progress for each step, including object counting, compression, and unpacking. The process concludes with 'Bootstrap complete. Quick build instructions: ./configure' and the prompt 'pi@raspberrypi:~/openocd-code \$' is visible.

```
pi@raspberrypi: ~/openocd-code
Resolving deltas: 100% (4716/4716), done.
Checking connectivity... done.
Submodule path 'jimtcl': checked out '51f65c6d38fbf86e1f0b036ad336761fd2ab7fa0'
Cloning into 'src/jtag/drivers/libjaylink'...
remote: Counting objects: 480, done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 480 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (480/480), 163.77 KiB | 155.00 KiB/s, done.
Resolving deltas: 100% (367/367), done.
Checking connectivity... done.
Submodule path 'src/jtag/drivers/libjaylink': checked out '24b8ce72c651d136825e4a8793e6396251f2f1'
Cloning into 'tools/git2cl'...
remote: Counting objects: 64, done.
remote: Total 64 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (64/64), done.
Checking connectivity... done.
Submodule path 'tools/git2cl': checked out '8373c9f74993e218a08819cbcdab3f3564b'
beba'
Bootstrap complete. Quick build instructions:
./configure ....
pi@raspberrypi:~/openocd-code $
```

Next, we will compile OpenOCD with the Raspberry Pi native GPIO twiddling support - this will work on various Raspberry Pi's despite being called 'bcm2835gpio'

`./configure --enable-sysfsgpio --enable-bcm2835gpio`

If you're following this guide on a non-Pi embedded linux board, you can skip the `--enable-bcm2835gpio` part and try to just use `sysfsgpio`. `Sysfsgpio` is much slower than native GPIO twiddling but it may not matter too much in your application.

A terminal window titled 'COM31 - PuTTY' showing the output of the 'configure' script. The script is checking for the presence and usability of various system headers and libraries. All checks are successful, with some marked as '(cached)'. The output ends with 'checking for net/if.h... yes' and a green cursor.

```
COM31 - PuTTY
checking for netinet/in.h... yes
checking for netinet/tcp.h... yes
checking poll.h usability... yes
checking poll.h presence... yes
checking for poll.h... yes
checking pthread.h usability... yes
checking pthread.h presence... yes
checking for pthread.h... yes
checking for strings.h... (cached) yes
checking sys/ioctl.h usability... yes
checking sys/ioctl.h presence... yes
checking for sys/ioctl.h... yes
checking sys/param.h usability... yes
checking sys/param.h presence... yes
checking for sys/param.h... yes
checking sys/select.h usability... yes
checking sys/select.h presence... yes
checking for sys/select.h... yes
checking for sys/stat.h... (cached) yes
checking sys/time.h usability... yes
checking sys/time.h presence... yes
checking for sys/time.h... yes
checking for sys/types.h... (cached) yes
checking for unistd.h... (cached) yes
checking for net/if.h... yes
```

```
COM31 - PuTTY
config.status: creating libjaylink/version.h
config.status: creating libjaylink.pc
config.status: creating config.h
config.status: executing depfiles commands
config.status: executing libtool commands

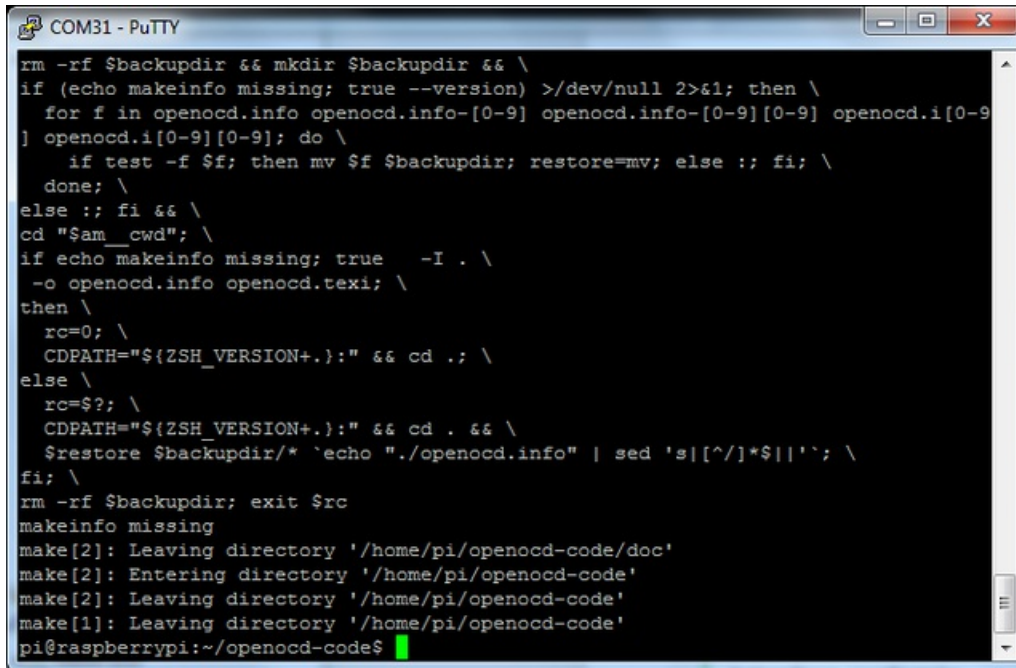
OpenOCD configuration summary
-----
MPSSE mode of FTDI based devices      yes (auto)
Segger J-Link JTAG Programmer         yes (auto)
ST-Link JTAG Programmer               yes (auto)
TI ICDI JTAG Programmer               yes (auto)
Keil ULINK JTAG Programmer            yes (auto)
Altera USB-Blaster II Compatible      yes (auto)
Versaloon-Link JTAG Programmer        yes (auto)
OSBDM (JTAG only) Programmer          yes (auto)
eStick/opensoc JTAG Programmer        yes (auto)
Andes JTAG Programmer                yes (auto)
USBProg JTAG Programmer               no
Raisonance RLink JTAG Programmer      no
Olimex ARM-JTAG-EW Programmer         no
CMSIS-DAP Compliant Debugger          no

pi@raspberrypi:~/openocd-code$
```

Note that when done, it won't mention GPIO support in the configuration summary, that's OK!

Run make

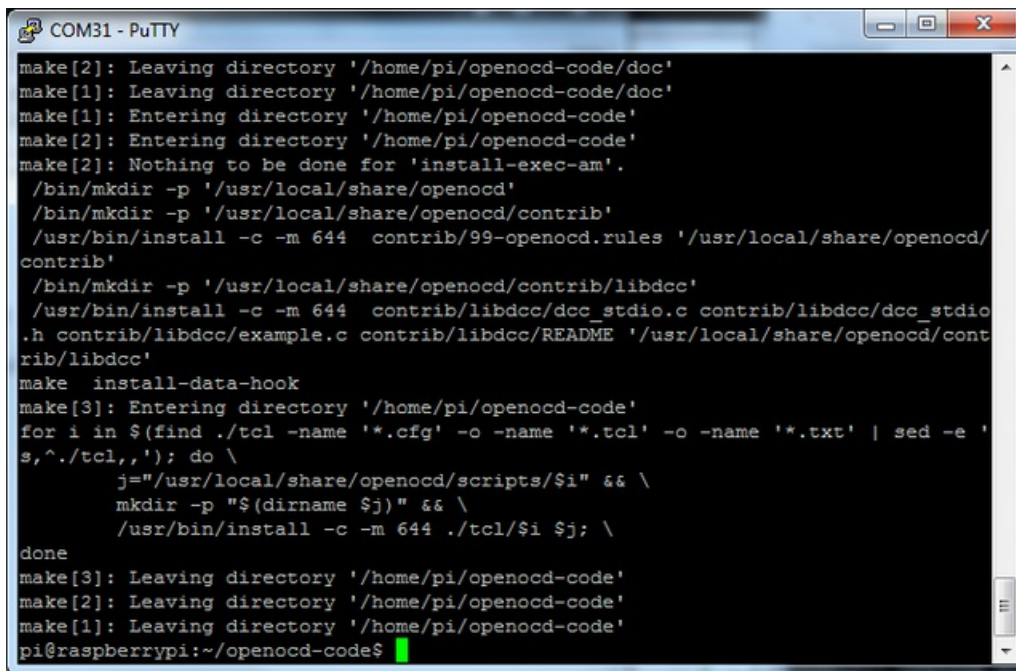
```
COM31 - PuTTY
-I../src -I../src/helper -DPKGDATA_DIR="/usr/local/share/openocd/" -DBINDIR="/usr/local/bin" -I../jimtcl -I../jimtcl -isystem /usr/include/libusb-1.0 -g -O2 -Wall -Wstrict-prototypes -Wformat-security -Wshadow -Wextra -Wno-unused-parameter -Wbad-function-cast -Wcast-align -Wredundant-decls -Werror -MT libocdtagdrivers_la-ulink.lo -MD -MP -MF .deps/libocdtagdrivers_la-ulink.Tpo -c ulink.c -o libocdtagdrivers_la-ulink.o
mv -f .deps/libocdtagdrivers_la-ulink.Tpo .deps/libocdtagdrivers_la-ulink.Plo
/bin/bash ../libtool --tag=CC --mode=compile gcc -std=gnu99 -DHAVE_CONFIG_H -I. -I../src -I../src/helper -DPKGDATA_DIR="/usr/local/share/openocd/" -DBINDIR="/usr/local/bin" -I../jimtcl -I../jimtcl -isystem /usr/include/libusb-1.0 -g -O2 -Wall -Wstrict-prototypes -Wformat-security -Wshadow -Wextra -Wno-unused-parameter -Wbad-function-cast -Wcast-align -Wredundant-decls -Werror -MT versaloon/usbtogpio/libocdtagdrivers_la-usbtogpio.lo -MD -MP -MF versaloon/usbtogpio/.deps/libocdtagdrivers_la-usbtogpio.Tpo -c versaloon/usbtogpio/libocdtagdrivers_la-usbtogpio.c `test -f 'versaloon/usbtogpio/usbtogpio.c' || echo './versaloon/usbtogpio/usbtogpio.c'`
libtool: compile: gcc -std=gnu99 -DHAVE_CONFIG_H -I. -I../src -I../src/helper -DPKGDATA_DIR="/usr/local/share/openocd/" -DBINDIR="/usr/local/bin" -I../jimtcl -I../jimtcl -isystem /usr/include/libusb-1.0 -g -O2 -Wall -Wstrict-prototypes -Wformat-security -Wshadow -Wextra -Wno-unused-parameter -Wbad-function-cast -Wcast-align -Wredundant-decls -Werror -MT versaloon/usbtogpio/libocdtagdrivers_la-usbtogpio.lo -MD -MP -MF versaloon/usbtogpio/.deps/libocdtagdrivers_la-usbtogpio.Tpo -c versaloon/usbtogpio/usbtogpio.c -o versaloon/usbtogpio/libocdtagdrivers_la-usbtogpio.o
```



```
rm -rf $backupdir && mkdir $backupdir && \
if (echo makeinfo missing; true --version) >/dev/null 2>&1; then \
  for f in openocd.info openocd.info-[0-9] openocd.info-[0-9][0-9] openocd.i[0-9]
] openocd.i[0-9][0-9]; do \
    if test -f $f; then mv $f $backupdir; restore=mv; else ;; fi; \
  done; \
else ;; fi && \
cd "$sam_cwd"; \
if echo makeinfo missing; true -I . \
-o openocd.info openocd.texi; \
then \
  rc=0; \
  CDPATH="${ZSH_VERSION+.:}" && cd .; \
else \
  rc=$?; \
  CDPATH="${ZSH_VERSION+.:}" && cd . && \
  $restore $backupdir/* `echo "/openocd.info" | sed 's|[^/]*$||'`; \
fi; \
rm -rf $backupdir; exit $rc
makeinfo missing
make[2]: Leaving directory '/home/pi/openocd-code/doc'
make[2]: Entering directory '/home/pi/openocd-code'
make[2]: Leaving directory '/home/pi/openocd-code'
make[1]: Leaving directory '/home/pi/openocd-code'
pi@raspberrypi:~/openocd-code$
```

Assuming compilation completes successfully as above, you can install with

`sudo make install`



```
make[2]: Leaving directory '/home/pi/openocd-code/doc'
make[1]: Leaving directory '/home/pi/openocd-code/doc'
make[1]: Entering directory '/home/pi/openocd-code'
make[2]: Entering directory '/home/pi/openocd-code'
make[2]: Nothing to be done for 'install-exec-am'.
/bin/mkdir -p '/usr/local/share/openocd'
/bin/mkdir -p '/usr/local/share/openocd/contrib'
/usr/bin/install -c -m 644 contrib/99-openocd.rules '/usr/local/share/openocd/
contrib'
/bin/mkdir -p '/usr/local/share/openocd/contrib/libdccc'
/usr/bin/install -c -m 644 contrib/libdccc/dcc_stdio.c contrib/libdccc/dcc_stdio
.h contrib/libdccc/example.c contrib/libdccc/README '/usr/local/share/openocd/cont
rib/libdccc'
make install-data-hook
make[3]: Entering directory '/home/pi/openocd-code'
for i in $(find ./tcl -name '*.cfg' -o -name '*.tcl' -o -name '*.txt' | sed -e '
s,^./tcl,,'); do \
  j="/usr/local/share/openocd/scripts/$i" && \
  mkdir -p "$(dirname $j)" && \
  /usr/bin/install -c -m 644 ./tcl/$i $j; \
done
make[3]: Leaving directory '/home/pi/openocd-code'
make[2]: Leaving directory '/home/pi/openocd-code'
make[1]: Leaving directory '/home/pi/openocd-code'
pi@raspberrypi:~/openocd-code$
```

That's pretty much it!

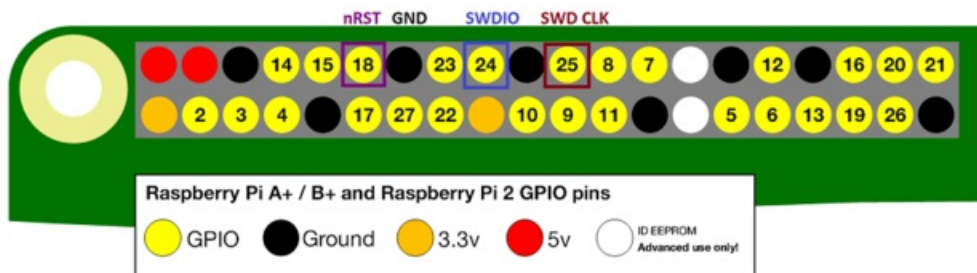
You can see the list of interfaces available in `/usr/local/share/openocd/scripts/interface`

There's a *lot* of options, in particular check out `raspberrypi2-native.cfg` and `raspberrypi-native.cfg` if you are interested in using OpenOCD with a non-Pi, look at `sysfsgpio-raspberrypi.cfg` which can help you port to a different linux computer


```

pi@raspberrypi: /usr/local/share/openocd/scripts/interface
pi@raspberrypi:~$ cd /usr/local/share/openocd/scripts/interface
pi@raspberrypi:/usr/local/share/openocd/scripts/interface$ ls
altera-usb-blaster2.cfg      hitex_str9-comstick.cfg      openrd.cfg
altera-usb-blaster.cfg      icebear.cfg                  osbdm.cfg
arm-jtag-ew.cfg             jlink.cfg                    parport.cfg
at91rm9200.cfg              jtagkey2.cfg                 parport_dlc5.cfg
axm0432.cfg                 jtagkey2p.cfg                raspberrypi2-native.cfg
busblaster.cfg              jtagkey.cfg                  raspberrypi-native.cfg
buspirate.cfg               jtagkey-tiny.cfg             redbee-econotag.cfg
calao-usb-a9260-c01.cfg     jtag-lock-pick_tiny_2.cfg    redbee-usb.cfg
calao-usb-a9260-c02.cfg     jtag_vpi.cfg                 rlink.cfg
calao-usb-a9260.cfg         kt-link.cfg                  sheevaplug.cfg
chameleon.cfg               lisa-l.cfg                   signalizer.cfg
cmsis-dap.cfg               luminary.cfg                  signalizer-h2.cfg
cortino.cfg                 luminary-icdi.cfg            signalizer-h4.cfg
digilent-hs1.cfg            luminary-lm3s811.cfg         signalizer-lite.cfg
dlp-usb1232h.cfg            minimodule.cfg               stlink-v1.cfg
dummy.cfg                   nds32-aice.cfg               stlink-v2-1.cfg
estick.cfg                  neodb.cfg                    stlink-v2.cfg
flashlink.cfg               ngxtech.cfg                  stm32-stick.cfg
flossjtag.cfg               olimex-arm-usb-ocd.cfg        sysfsgpio-raspberrypi.cfg
flossjtag-noeeprom.cfg      olimex-arm-usb-ocd-h.cfg      ti-icdi.cfg
flyswatter2.cfg             olimex-arm-usb-tiny-h.cfg     turtelizer2.cfg
flyswatter.cfg              olimex-jtag-tiny.cfg          ulink.cfg
ftdi                        oocdlink.cfg                 usb-jtag.cfg
hilscher_nxhx10_etm.cfg     opendous.cfg                 usbprog.cfg
hilscher_nxhx500_etm.cfg    opendous_ftdi.cfg            vpaclink.cfg
hilscher_nxhx500_re.cfg     openjtag.cfg                  vsllink.cfg
hilscher_nxhx50_etm.cfg     openocd-usb.cfg              xds100v2.cfg
hilscher_nxhx50_re.cfg      openocd-usb-hs.cfg
pi@raspberrypi:/usr/local/share/openocd/scripts/interface$

```



Wiring and Test

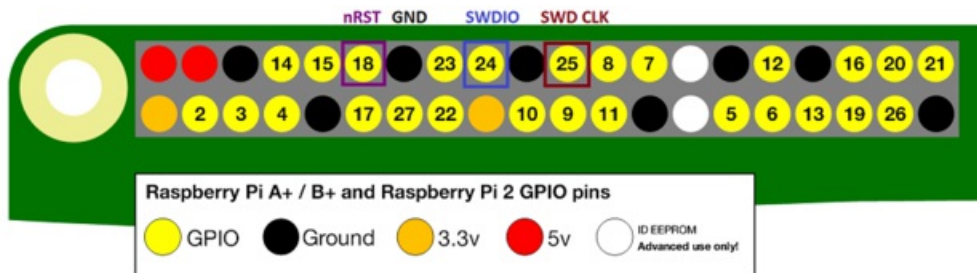
Connecting to Target

OK you've done the compiling, now you are ready to connect!

In this case, I'll be connecting to an Atmel ATSAMD21G18 Cortex-M0 over SWD and uploading the Arduino bootloader to it. You can, of course, connect to *any* processor that OpenOCD supports but this is the one I've got handy

Wire up the target to SWD

Of course connections must be made! Note that we are using the "BCM" pin numbering convention (<https://adafru.it/jEa>)

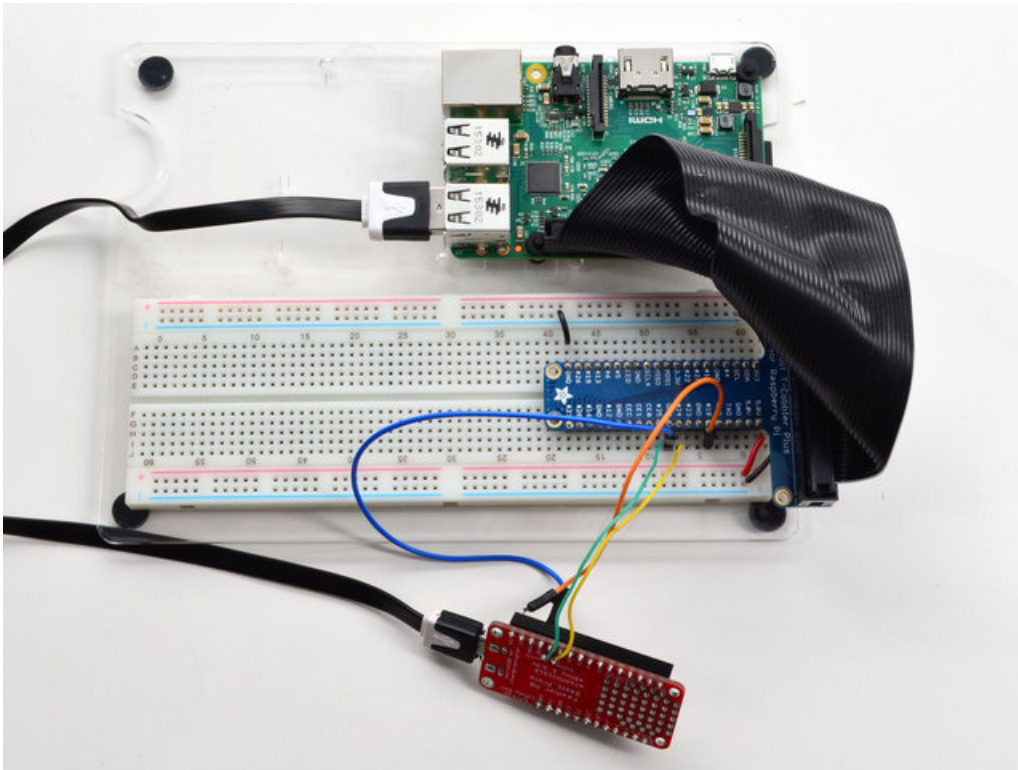


Connect:

- Target GND to Pi GND
- Target SWDIO to Raspberry Pi #24
- Target SWCLK to Raspberry Pi #25
- Target Reset to Raspberry Pi #18 (may not be required)
- If powering the chip directly from the Pi, connect 3.3V to 3.3V (I'm just powering the chip over USB)

Of course, this assumes that your chip is running at 3.3V logic. For 1.8 or 5V logic, level shifting may be required.

You can later change the pins used in the `interfaces` configuration file but for now, I suggest just going with the default



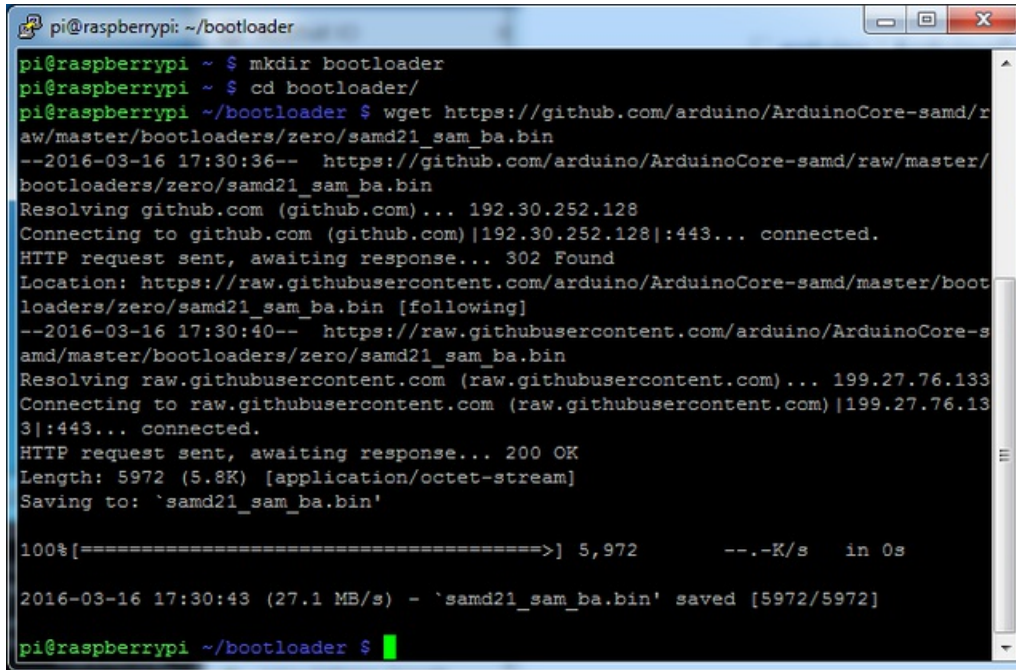
Create OpenOCD config

The easiest way to connect is creating a new directory in your home dir

```
cd ~  
mkdir bootloader  
cd bootloader
```

and then putting the file you want to program there, in this case I'm going to just grab the latest Arduino Zero bootloader (of course, substitute your own binary or hex!)

```
wget https://github.com/arduino/ArduinoCore-samd/raw/master/bootloaders/zero/samd21_sam_ba.bin
```

A terminal window titled 'pi@raspberrypi: ~/bootloader' showing the following commands and output:

```
pi@raspberrypi ~ $ mkdir bootloader
pi@raspberrypi ~ $ cd bootloader/
pi@raspberrypi ~/bootloader $ wget https://github.com/arduino/ArduinoCore-samd/raw/master/bootloaders/zero/samd21_sam_ba.bin
--2016-03-16 17:30:36-- https://github.com/arduino/ArduinoCore-samd/raw/master/bootloaders/zero/samd21_sam_ba.bin
Resolving github.com (github.com)... 192.30.252.128
Connecting to github.com (github.com)|192.30.252.128|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://raw.githubusercontent.com/arduino/ArduinoCore-samd/master/bootloaders/zero/samd21_sam_ba.bin [following]
--2016-03-16 17:30:40-- https://raw.githubusercontent.com/arduino/ArduinoCore-samd/master/bootloaders/zero/samd21_sam_ba.bin
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 199.27.76.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|199.27.76.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5972 (5.8K) [application/octet-stream]
Saving to: `samd21_sam_ba.bin'

100%[=====>] 5,972      --.-K/s   in 0s

2016-03-16 17:30:43 (27.1 MB/s) - `samd21_sam_ba.bin' saved [5972/5972]

pi@raspberrypi ~/bootloader $
```

In the same directory, make a new file called **openocd.cfg**

nano openocd.cfg

and put the following into it:

```
source [find interface/raspberrypi2-native.cfg]
transport select swd

set CHIPNAME at91samd21g18
source [find target/at91samdXX.cfg]

# did not yet manage to make a working setup using srst
#reset_config srst_only
reset_config srst_nogate

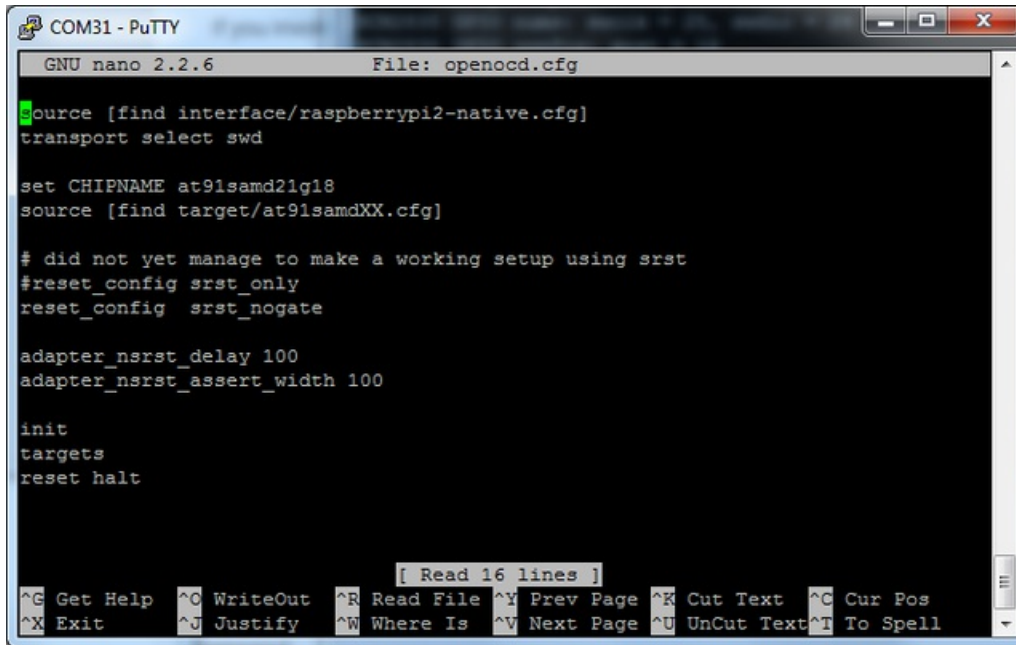
adapter_nsrst_delay 100
adapter_nsrst_assert_width 100

init
targets
reset halt
```

Change **raspberrypi2-native.cfg** to whatever config you are using, e.g. for a Pi Zero or 1 use **raspberrypi1-native.cfg** or **raspberrypi-native.cfg**

If you're using a Pi Zero/1 you may also need to add

```
bcm2835gpio_swd_nums 25 24
bcm2835gpio_trst_num 7
bcm2835gpio_srst_num 18
```

```
COM31 - PuTTY
GNU nano 2.2.6      File: openocd.cfg

source [find interface/raspberrypi2-native.cfg]
transport select swd

set CHIPNAME at91samd21g18
source [find target/at91samdXX.cfg]

# did not yet manage to make a working setup using srst
#reset_config srst_only
reset_config srst_nogate

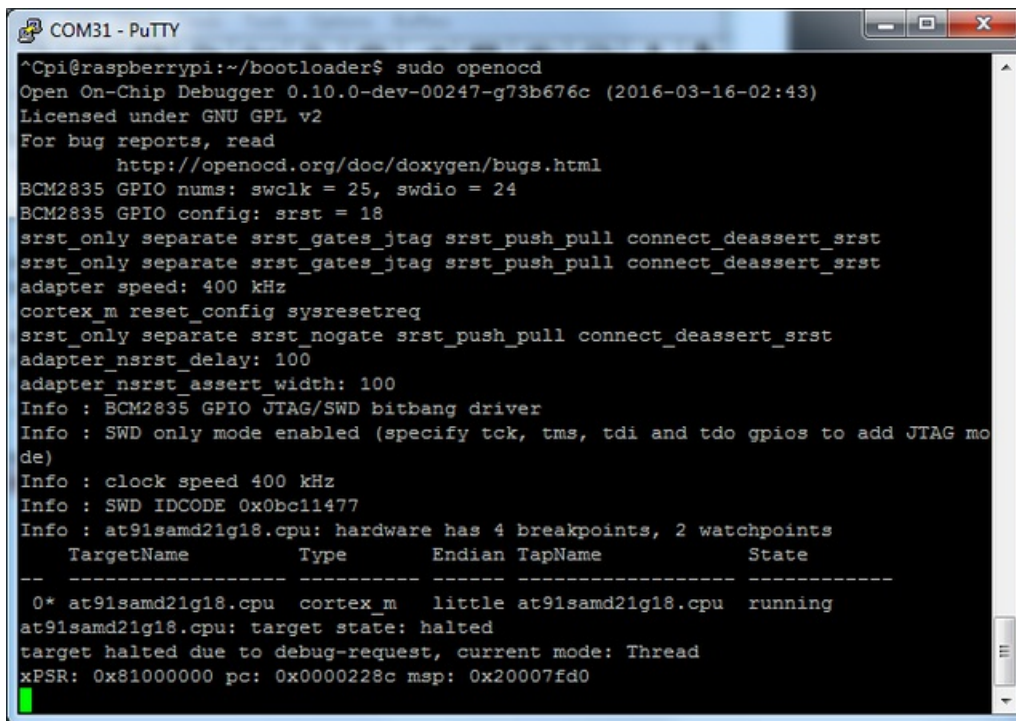
adapter_nsrst_delay 100
adapter_nsrst_assert_width 100

init
targets
reset halt

[ Read 16 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

You may need to also comment out `reset_config srst_nogate`, some people report that is required to make it work

Save the config file and then run `sudo openocd` (no other args, its all in the config!) in the directory. You *should* get the following indicating a good connection



```
COM31 - PuTTY

^Cpi@raspberrypi:~/bootloader$ sudo openocd
Open On-Chip Debugger 0.10.0-dev-00247-g73b676c (2016-03-16-02:43)
Licensed under GNU GPL v2
For bug reports, read
    http://openocd.org/doc/doxygen/bugs.html
BCM2835 GPIO nums: swclk = 25, swdio = 24
BCM2835 GPIO config: srst = 18
srst_only separate srst_gates_jtag srst_push_pull connect_deassert_srst
srst_only separate srst_gates_jtag srst_push_pull connect_deassert_srst
adapter speed: 400 kHz
cortex_m reset_config sysresetreq
srst_only separate srst_nogate srst_push_pull connect_deassert_srst
adapter_nsrst_delay: 100
adapter_nsrst_assert_width: 100
Info : BCM2835 GPIO JTAG/SWD bitbang driver
Info : SWD only mode enabled (specify tck, tms, tdi and tdo gpios to add JTAG mo
de)
Info : clock speed 400 kHz
Info : SWD IDCODE 0x0bc11477
Info : at91samd21g18.cpu: hardware has 4 breakpoints, 2 watchpoints
   TargetName      Type      Endian TapName      State
   -----
0* at91samd21g18.cpu cortex_m  little at91samd21g18.cpu running
at91samd21g18.cpu: target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x81000000 pc: 0x0000228c msp: 0x20007fd0
```

In particular make sure you get that **target state:halted** to you know it was able to connect!

Hit control-C to cancel out of openocd.

If you get **unknown** for state, or other errors, check your wiring! You may also need to powercycle or disconnect parts from the chip to get it into a good programming state. You may also need to change the programming frequency

```
pi@raspberrypi: ~/bootloader
^Cpi@raspberrypi ~/bootloader $ sudo openocd
Open On-Chip Debugger 0.10.0-dev-00247-g73b676c (2016-03-16-17:02)
Licensed under GNU GPL v2
For bug reports, read
    http://openocd.org/doc/doxygen/bugs.html
BCM2835 GPIO nums: swclk = 25, swdio = 24
BCM2835 GPIO config: srst = 18
srst_only separate srst_gates_jtag srst_push_pull connect_deassert_srst
srst_only separate srst_gates_jtag srst_push_pull connect_deassert_srst
adapter speed: 400 kHz
cortex_m reset_config sysresetreq
srst_only separate srst_nogate srst_push_pull connect_deassert_srst
adapter_nsrst_delay: 100
adapter_nsrst_assert_width: 100
Info : BCM2835 GPIO JTAG/SWD bitbang driver
Info : SWD only mode enabled (specify tck, tms, tdi and tdo gpios to add JTAG mo
de)
Info : clock speed 400 kHz
Info : SWD IDCODE 0x019e4838
Error: Could not initialize the debug port


| TargetName           | Type     | Endian | TapName           | State   |
|----------------------|----------|--------|-------------------|---------|
| 0* at91samd21g18.cpu | cortex_m | little | at91samd21g18.cpu | unknown |


Error: Could not initialize the debug port
Error: Target not examined yet
in procedure 'reset' called at file "openocd.cfg", line 16
in procedure 'ocd_bouncer'
```

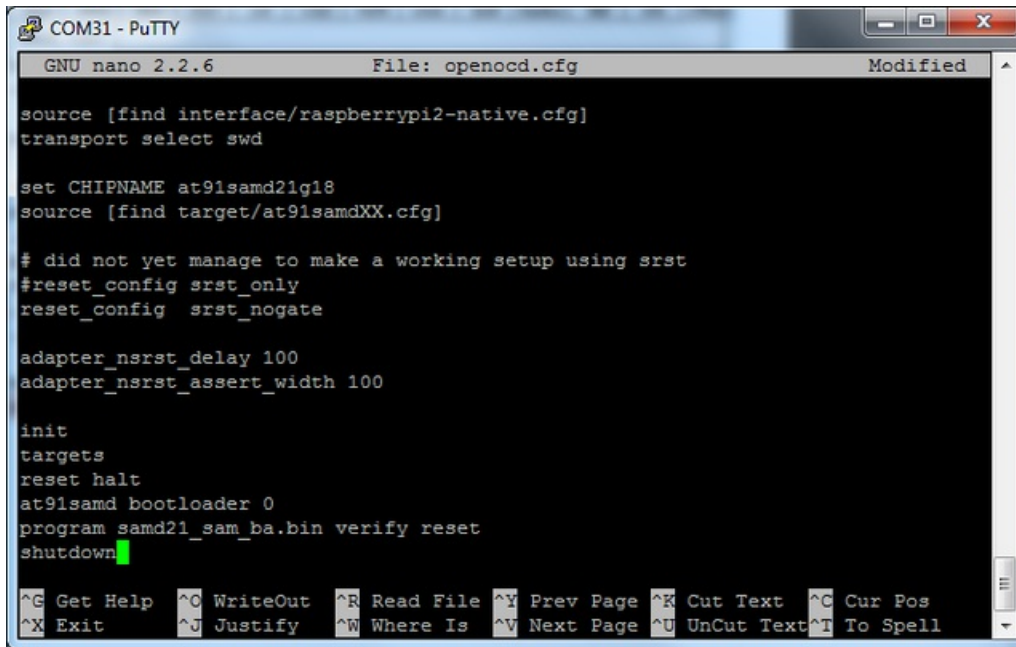
Hit control-C to cancel out of openocd (or you can **telnet 127.0.0.1 4444** if you want to send commands, won't be covered here).

Now you can change the **openocd.cfg** with nano to add commands for burning the binary file. At the bottom put in:

```
init
targets
reset halt
at91samd bootloader 0
program samd21_sam_ba verify
at91samd bootloader 8192
reset
shutdown
```

This will init, look for targets, reset and halt the chip, turn off bootloader protection, burn in the bootloader file and verify it, re-turn-on bootloader protection, reset and shutdown openocd

You can skip the bootloader protection parts if you are not burning in a bootloader, of course



```
COM31 - PuTTY
GNU nano 2.2.6      File: openocd.cfg      Modified

source [find interface/raspberrypi2-native.cfg]
transport select swd

set CHIPNAME at91samd21g18
source [find target/at91samdXX.cfg]

# did not yet manage to make a working setup using srst
#reset_config srst_only
reset_config srst_nogate

adapter_nsrst_delay 100
adapter_nsrst_assert_width 100

init
targets
reset halt
at91samd bootloader 0
program samd21_sam_ba.bin verify reset
shutdown
```

Of course, change the commands if you have a different file name, different chip, etc.

Save the file and run `sudo openocd` again:

```
pi@raspberrypi: ~/bootloader
pi@raspberrypi:~/bootloader $ sudo openocd
Open On-Chip Debugger 0.10.0-dev-00247-g73b676c (2016-03-16-02:43)
Licensed under GNU GPL v2
For bug reports, read
    http://openocd.org/doc/doxygen/bugs.html
BCM2835 GPIO nums: swclk = 25, swdio = 24
BCM2835 GPIO config: srst = 18
srst_only separate srst_gates_jtag srst_push_pull connect_deassert_srst
srst_only separate srst_gates_jtag srst_push_pull connect_deassert_srst
adapter speed: 400 kHz
cortex_m reset_config sysresetreq
srst_only separate srst_nogate srst_push_pull connect_deassert_srst
adapter_nsrst_delay: 100
adapter_nsrst_assert_width: 100
Info : BCM2835 GPIO JTAG/SWD bitbang driver
Info : SWD only mode enabled (specify tck, tms, tdi and tdo gpios to add JTAG mode)
Info : clock speed 400 kHz
Info : SWD IDCODE 0x0bc11477
Info : at91samd21g18.cpu: hardware has 4 breakpoints, 2 watchpoints
   TargetName           Type           Endian TapName           State
   -----
0* at91samd21g18.cpu cortex_m little at91samd21g18.cpu halted
at91samd21g18.cpu: target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x61000000 pc: 0x2000002e msp: 0x20007fd0
at91samd21g18.cpu: target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x61000000 pc: 0x2000002e msp: 0x20007fd0
** Programming Started **
auto erase enabled
Info : SAMD MCU: SAMD21G18A (256KB Flash, 32KB RAM)
wrote 16384 bytes from file samd21_sam_ba.bin in 0.708989s (22.567 KiB/s)
** Programming Finished **
** Verify Started **
verified 6480 bytes in 0.021981s (287.891 KiB/s)
** Verified OK **
** Resetting Target **
shutdown command invoked
pi@raspberrypi:~/bootloader $
```

Zoom! Programmed the bootloader in 0.02 seconds!

More Options

If you don't want to set up the configure file, you can actually do it all from the command line:

```
sudo openocd -f interface/raspberrypi2-native.cfg -c "transport select swd; set WORKAREASIZE 0;
adapter_nsrst_delay 100; adapter_nsrst_assert_width 100; source [find target/nrf51.cfg]" -c "init; reset; halt; nrf51
mass_erase; reset" -c "shutdown"
```

This will, for example, erase and reset a Nordic nRF51822 (which is a pretty finicky chip by the way, you may need to do hard resets to get it to talk to openocd)


```
pi@raspberrypi: ~  
pi@raspberrypi ~ $ sudo openocd -f interface/raspberrypi2-native.cfg -c "transport select swd; set WORKAREASIZE 0; adapter_nsrst_delay 100; adapter_nsrst_assert_width 100; source [find target/nrf51.cfg]" -c "init; halt; nrf51 mass_erase; reset" -c "shutdown"  
Open On-Chip Debugger 0.10.0-dev-00247-g73b676c (2016-03-16-17:02)  
Licensed under GNU GPL v2  
For bug reports, read  
    http://openocd.org/doc/doxygen/bugs.html  
BCM2835 GPIO nums: swclk = 25, swdio = 24  
BCM2835 GPIO config: srst = 18  
srst_only separate srst_gates_jtag srst_push_pull connect_deassert_srst  
adapter_nsrst_delay: 100  
adapter_nsrst_assert_width: 100  
cortex_m reset_config sysresetreq  
adapter speed: 1000 kHz  
Info : BCM2835 GPIO JTAG/SWD bitbang driver  
Info : SWD only mode enabled (specify tck, tms, tdi and tdo gpios to add JTAG mode)  
Info : clock speed 1001 kHz  
Info : SWD IDCODE 0x0bb11477  
Info : nrf51.cpu: hardware has 4 breakpoints, 2 watchpoints  
Info : nRF51822-QFAC(build code: A1) 256kB Flash  
Error: nrf51.cpu -- clearing lockup after double fault  
nrf51.cpu: target state: halted  
target halted due to debug-request, current mode: Handler HardFault  
xPSR: 0xc1000003 pc: 0xffffffff msp: 0xfffffd8  
Polling target nrf51.cpu failed, trying to reexamine  
Info : nrf51.cpu: hardware has 4 breakpoints, 2 watchpoints  
shutdown command invoked  
pi@raspberrypi ~ $
```

