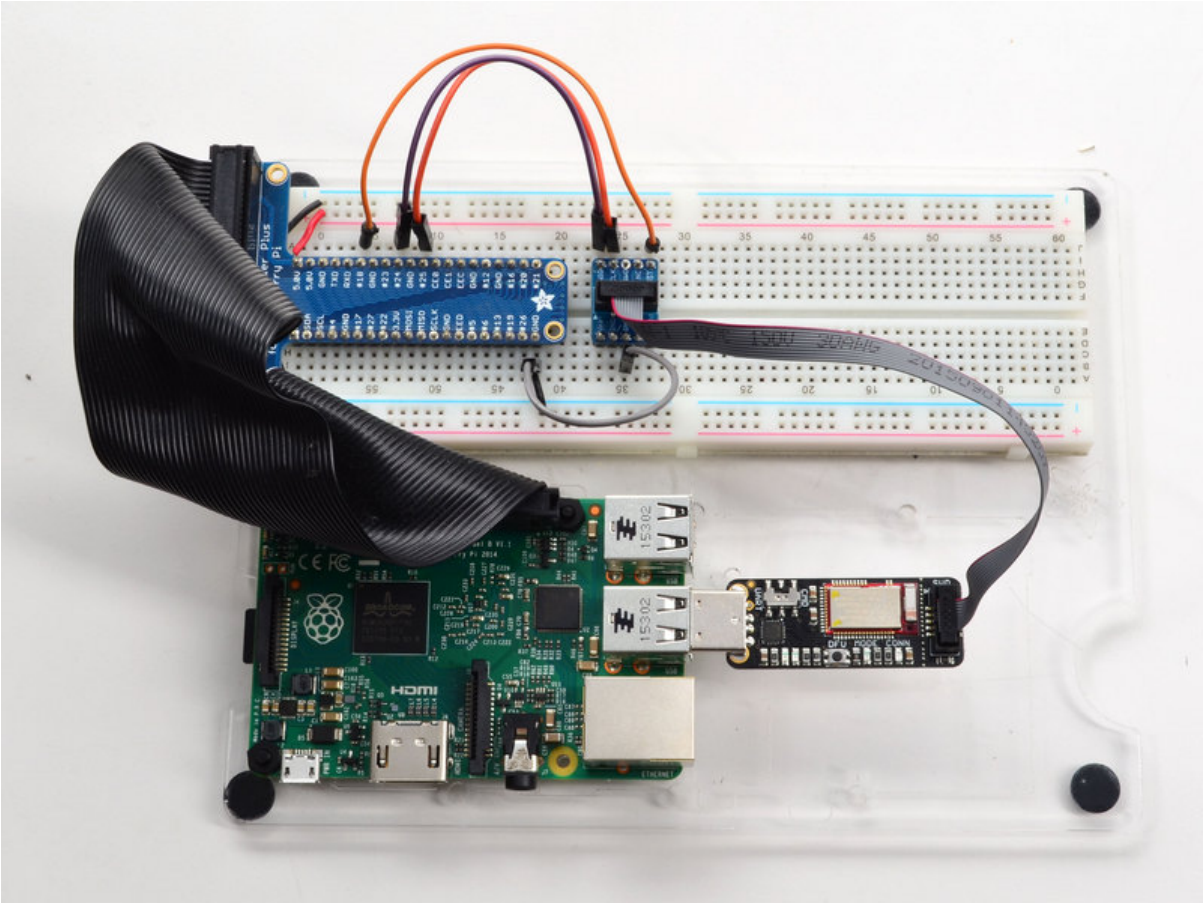
Programming Microcontrollers using OpenOCD on a Raspberry Pi

Created by lady ada

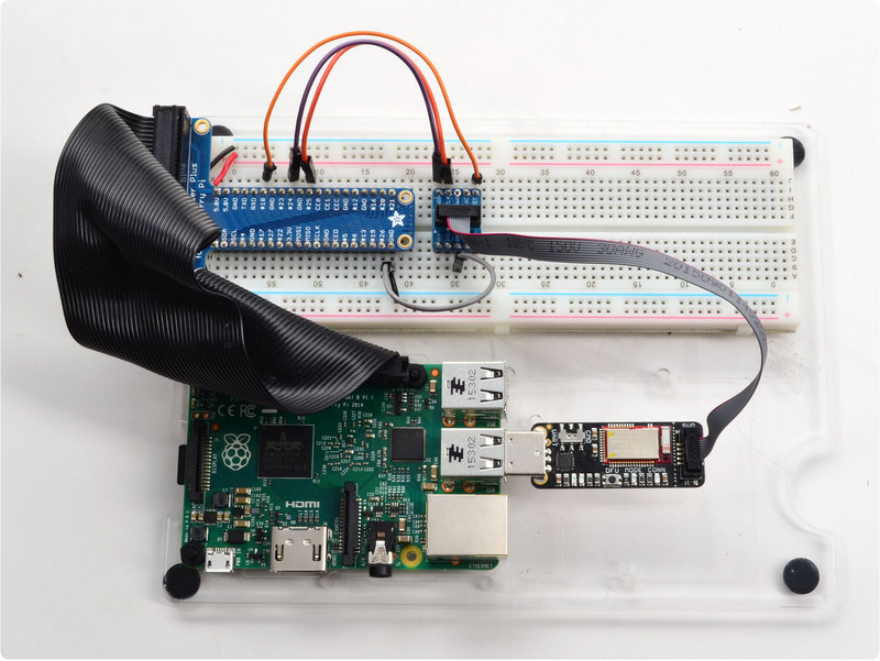


https://learn.adafruit.com/programming-microcontrollers-using-openocd-on-raspberry-pi

Last updated on 2024-03-08 02:24:30 PM EST

Table of Contents

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://adafru.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 NRFGo, 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!

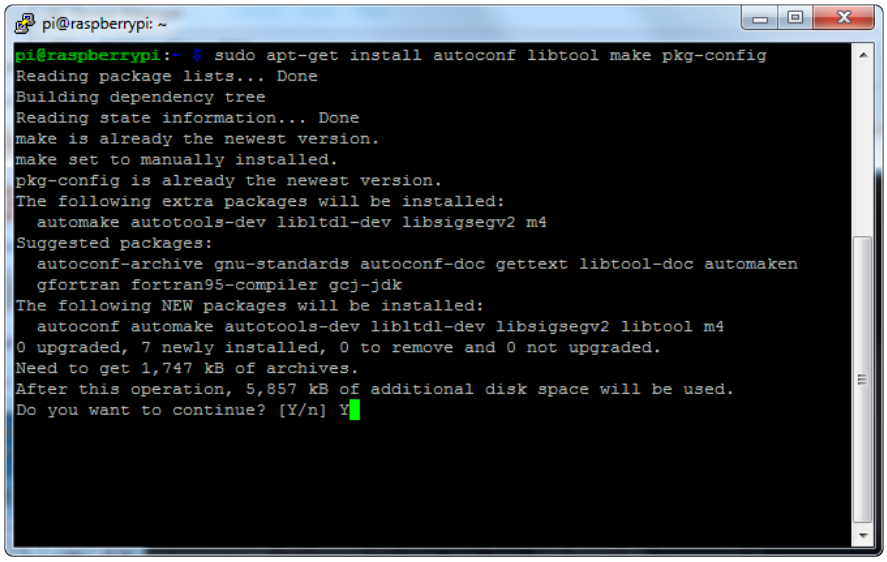
Compiling OpenOCD

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: ~
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

```
pi@raspberrypi: ~
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:

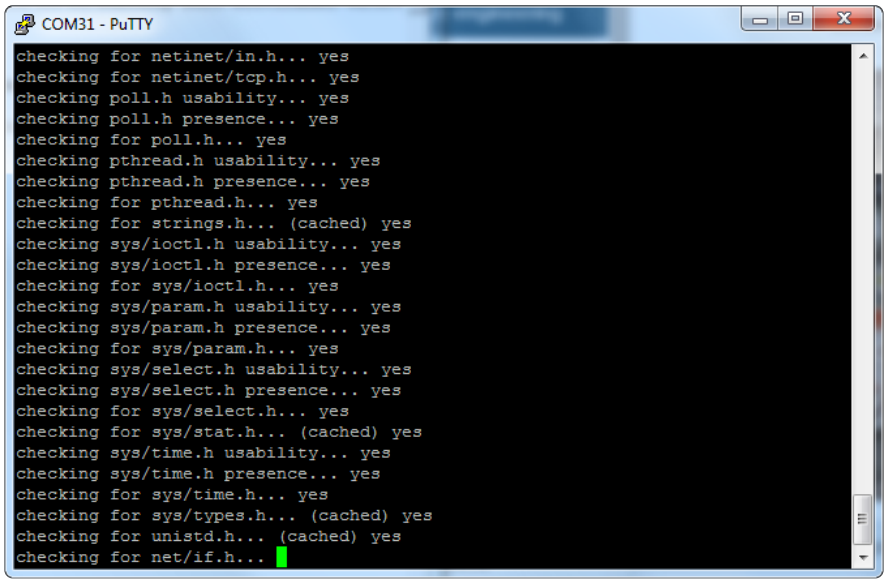
cd openocd-code
./bootstrap

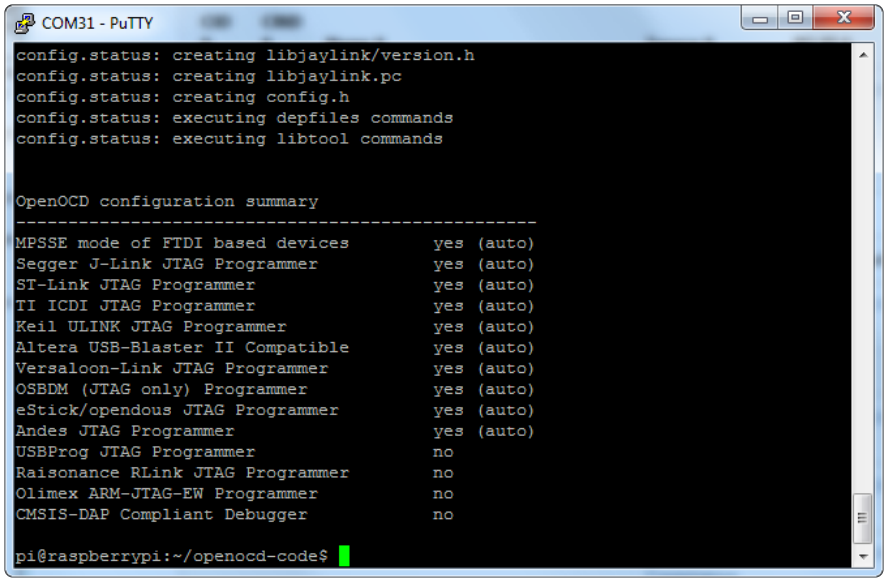
```
pi@raspberrypi: ~/openocd-code
Resolving deltas: 100% (4716/4716), done.
Checking connectivity... done.
Submodule path 'jimtcl': checked out '51f65c6d38fbf86e1f0b036ad336761fd2ab7fa0'
Cloning into 'src/jtag/drivers/libjaylink'...
remote: Counting objects: 480, done.
remote: Compressing objects: 100% (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 '24b8ce72c651d136825e4
a8793ece6396251f2f1'
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 '8373c9f74993e218a08819cbcdbab3f3564b
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

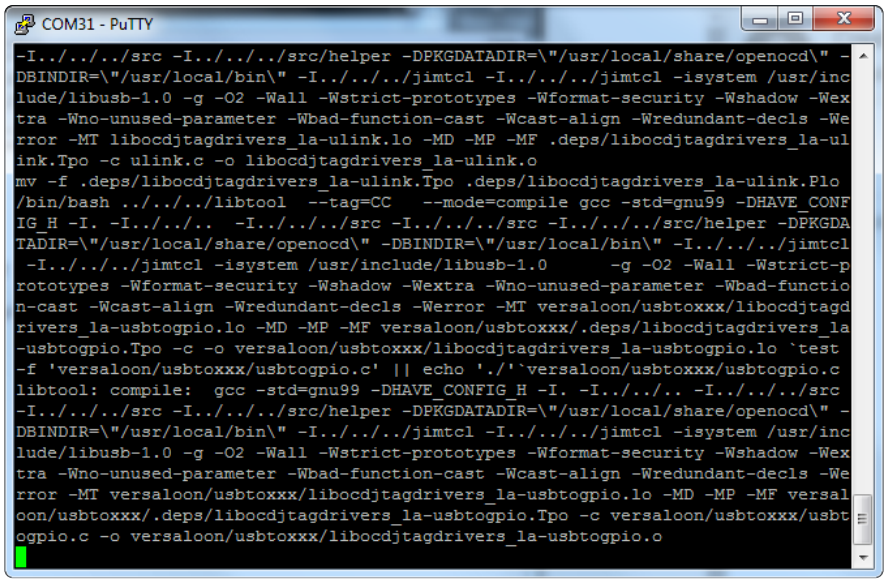
If you're following this guide on a non-Pi embedded linux board, you can skip the **--enable-bcm2835gpio** part and try to just use sysfsgpio. Sysfsgpio is much slower than native GPIO twiddling but it may not matter too much in your application.

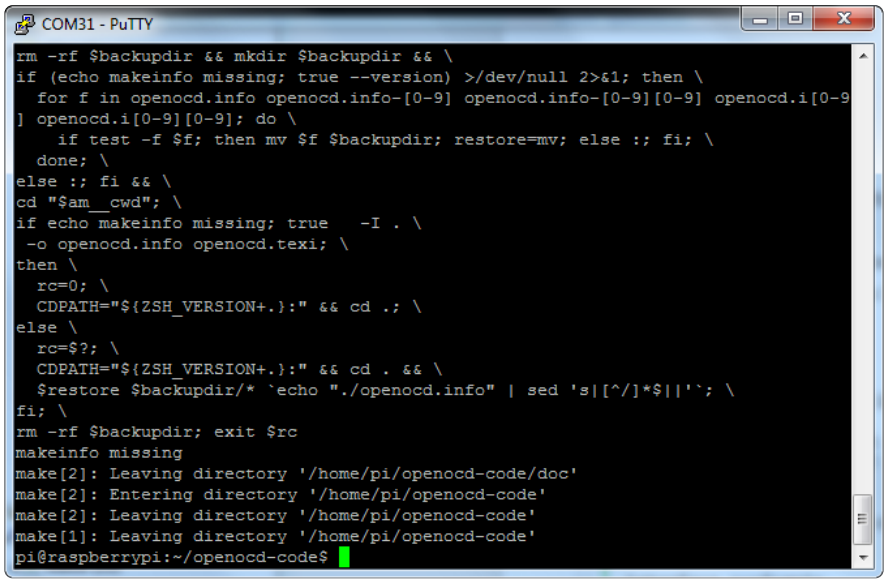




Note that when done, it wont mention GPIO support in the configuration summary, thats OK!

Run **make**





Assuming compilation completes successfully as above, you can install with

sudo make install

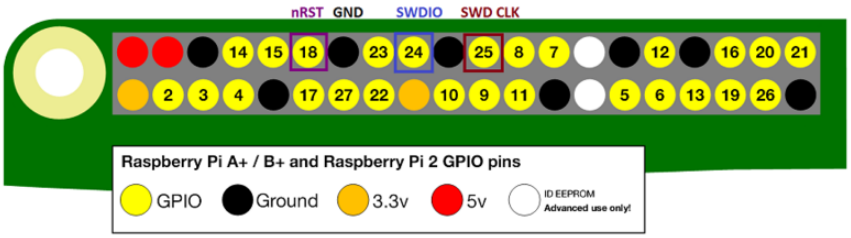
```
COM31 - PuTTY
make[2]: Leaving directory '/home/pi/openocd-code/doc'
make[1]: Leaving directory '/home/pi/openocd-code/doc'
make[1]: Entering directory '/home/pi/openocd-code'
make[2]: Entering directory '/home/pi/openocd-code'
make[2]: Nothing to be done for 'install-exec-am'.
 /bin/mkdir -p '/usr/local/share/openocd'
 /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/libdcc'
 /usr/bin/install -c -m 644  contrib/libdcc/dcc_stdio.c contrib/libdcc/dcc_stdio
.h contrib/libdcc/example.c contrib/libdcc/README '/usr/local/share/openocd/cont
rib/libdcc'
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                signalyzer.cfg
cmsis-dap.cfg             luminary.cfg              signalyzer-h2.cfg
cortino.cfg               luminary-icdi.cfg         signalyzer-h4.cfg
digilent-hs1.cfg          luminary-lm3s811.cfg      signalyzer-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

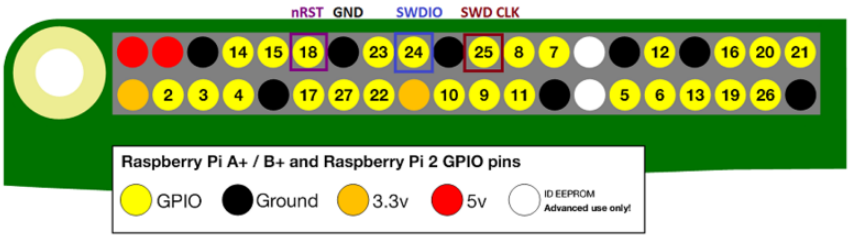
Connecting to Target

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)

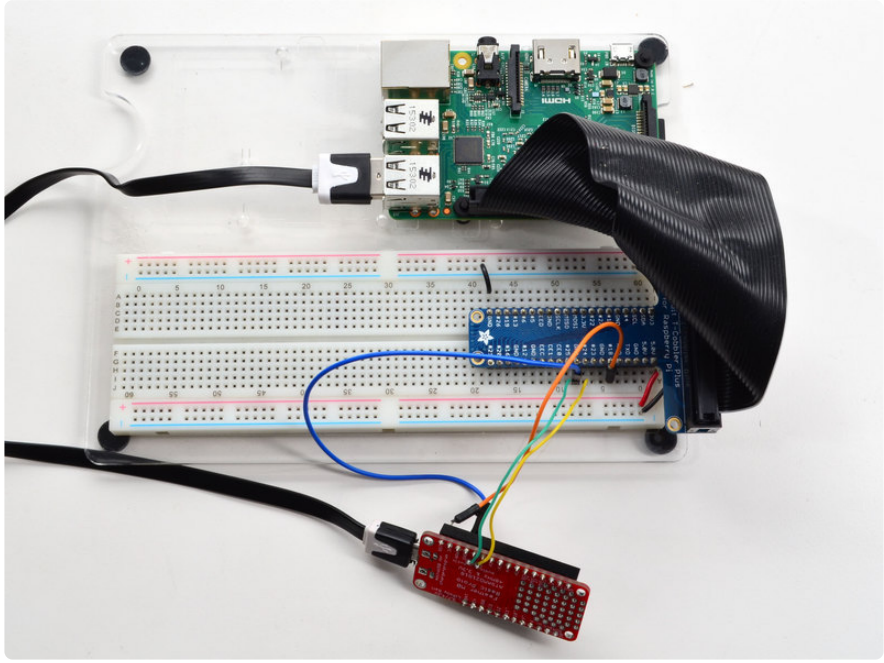


Connect:

- **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



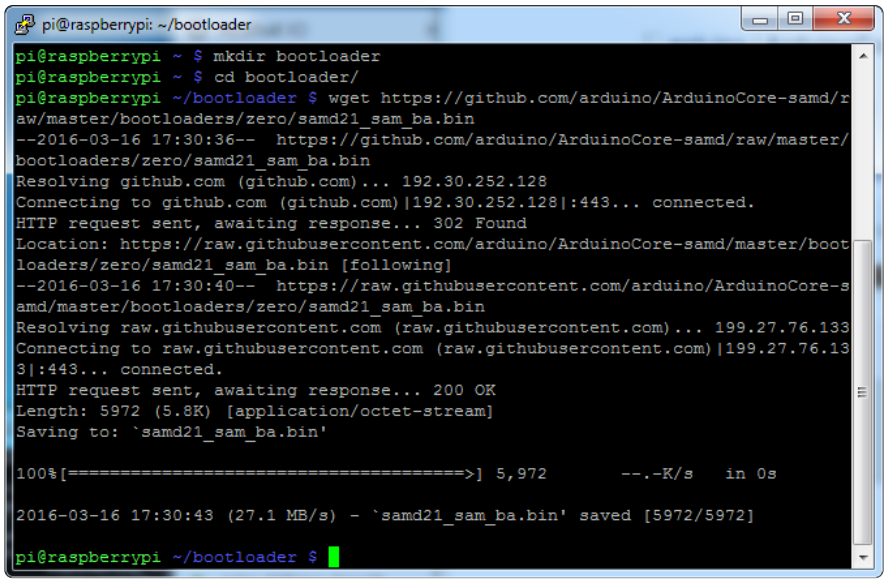
Create OpenOCD config

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



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

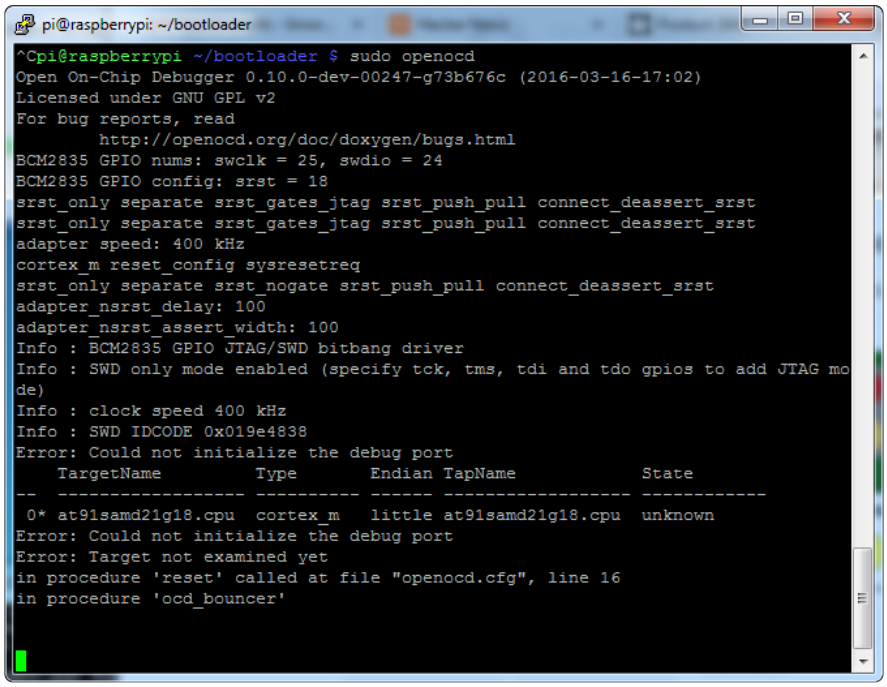
Save the config file and then run **sudo openocd** (no other args, its all in the config!) in the directory. You should get the following indicating a good connection

```
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



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

^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
```

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 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  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)