### Link:

### <https://kb.ettus.com/Building_and_Installing_the_USRP_Open-Source_Toolchain_(UHD_and_GNU_Radio)_on_Linux#Building_and_installing_UHD_from_source_code>

### UHD Version From GIT

### linux; GNU C++ version 7.4.0; Boost\_106501; UHD\_3.14.1.HEAD-0-g0347a6d8

Building and installing UHD from source code

UHD is open-source, and is hosted on GitHub. You can browse the code online at the link below, which points to version 3.14.0.0, which is the the latest release at the time of this writing.

* [UHD repository on GitHub](https://github.com/EttusResearch/uhd/tree/v3.14.0.0)

There are several good reasons to build GNU Radio from source code, especially for doing development and prototyping. It it enables an easy way to customize the location of the installation, and to install multiple UHD versions in parallel, and switch between them. It also provides much more flexibility in upgrading and downgrading versions, and allows the user to modify the code and create customized versions, which could possibly include a patch or other bug-fix.

To build UHD from source code, clone the GitHub repository, check out a branch or tagged release of the repository, and build and install. Please follow the steps below. Make sure that no USRP device is connected to the system at this point.

First, make a folder to hold the repository.

cd $HOME

mkdir workarea

cd workarea

Next, clone the repository and change into the cloned directory.

git clone https://github.com/EttusResearch/uhd

cd uhd

Next, checkout the desired UHD version. You can get a full listing of tagged releases by running the command:

git tag -l

*Example truncated output of git tag -l:*

$ git tag -l

...

release\_003\_009\_004

release\_003\_009\_005

release\_003\_010\_000\_000

**Note**: As of UHD Version 3.10.0.0, the versioning scheme has changed to be a quadruplet format. Each element and version will follow the format of: **Major.API.ABI.Patch**. Additional details on this versioning change can be found [here](https://files.ettus.com/manual/page_semver.html).

After identifying the version and corresponding release tag you need, check it out:

# Example: For UHD 3.9.5:

git checkout release\_003\_009\_005

# Example: For UHD 3.14.0.0

git checkout v3.14.0.0

Next, create a build folder within the repository, invoke CMake and build UHD.

cd host

mkdir build

cd build

cmake ../

make

Next, you can optionally run some basic tests to verify that the build process completed properly.

make test

Next, install UHD, using the default install prefix, which will install UHD under the /usr/local/lib folder. You need to run this as root due to the permissions on that folder.

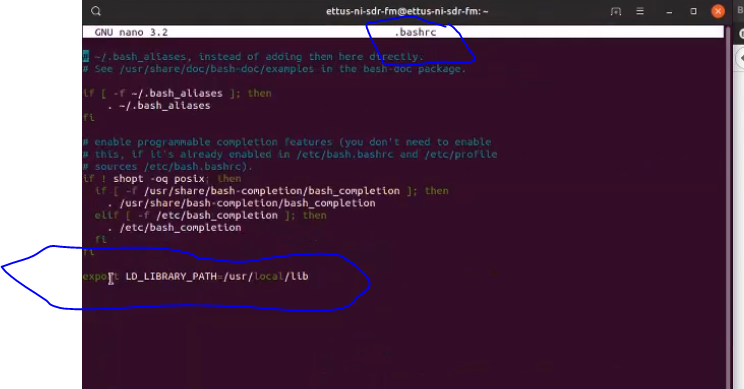
sudo make install

Next, update the system's shared library cache.

sudo ldconfig

Finally, make sure that the LD\_LIBRARY\_PATH environment variable is defined and includes the folder under which UHD was installed. Most commonly, you can add the line below to the end of your $HOME/.bashrc file:

export LD\_LIBRARY\_PATH=/usr/local/lib



If the LD\_LIBRARY\_PATH environment variable is already defined with other folders in your $HOME/.bashrc file, then add the line below to the end of your $HOME/.bashrc file to preserve the current settings.

export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:/usr/local/lib

For this change to take effect, you will need to close the current terminal window, and open a new terminal.

At this point, UHD should be installed and ready to use. You can quickly test this, with no USRP device attached, by running uhd\_find\_devices. You should see something similar to the following.

linux; GNU C++ version 4.8.4; Boost\_105400; UHD\_003.010.000.HEAD-0-g6e1ac3fc

No UHD Devices Found

## Ubuntu 18.04 Terminal Shots

### cmake ../ 🡪 klieberschnitzel@ubuntu18-04-3:~/workarea-uhd/uhd/host/build$ cmake ../

--

-- Configuring the python interpreter...

-- Python interpreter: /usr/bin/python2

-- Override with: -DPYTHON\_EXECUTABLE=<path-to-python>

-- Python runtime interpreter: /usr/bin/python2

-- Override with: -DRUNTIME\_PYTHON\_EXECUTABLE=<path-to-python>

-- Working off of feature or development branch. Updating version number.

-- Using UHD Images Directory: /usr/local/share/uhd/images

-- Build type not specified: defaulting to release.

--

-- Configuring Boost C++ Libraries...

-- Looking for optional Boost components...

-- Boost version: 1.65.1

-- Found the following Boost libraries:

-- python

-- Looking for required Boost components...

-- Boost version: 1.65.1

-- Found the following Boost libraries:

-- chrono

-- date\_time

-- filesystem

-- program\_options

-- regex

-- system

-- unit\_test\_framework

-- serialization

-- thread

-- atomic

-- Boost include directories: /usr/include

-- Boost library directories: /usr/lib/x86\_64-linux-gnu

-- Boost libraries: /usr/lib/x86\_64-linux-gnu/libboost\_chrono.so;/usr/lib/x86\_64-linux-gnu/libboost\_date\_time.so;/usr/lib/x86\_64-linux-gnu/libboost\_filesystem.so;/usr/lib/x86\_64-linux-gnu/libboost\_program\_options.so;/usr/lib/x86\_64-linux-gnu/libboost\_regex.so;/usr/lib/x86\_64-linux-gnu/libboost\_system.so;/usr/lib/x86\_64-linux-gnu/libboost\_unit\_test\_framework.so;/usr/lib/x86\_64-linux-gnu/libboost\_serialization.so;/usr/lib/x86\_64-linux-gnu/libboost\_thread.so;/usr/lib/x86\_64-linux-gnu/libboost\_atomic.so;/usr/lib/x86\_64-linux-gnu/libpthread.so

--

-- Python checking for Python version 2.7 or greater

-- Python checking for Python version 2.7 or greater - found

--

-- Python checking for Mako templates 0.4.2 or greater

-- Python checking for Mako templates 0.4.2 or greater - found

--

-- Python checking for requests 2.0 or greater

-- Python checking for requests 2.0 or greater - found

--

-- Python checking for numpy 1.7 or greater

-- Python checking for numpy 1.7 or greater - found

--

-- Configuring LibUHD support...

-- Dependency Boost\_FOUND = 1

-- Dependency HAVE\_PYTHON\_PLAT\_MIN\_VERSION = TRUE

-- Dependency HAVE\_PYTHON\_MODULE\_MAKO = TRUE

-- Enabling LibUHD support.

-- Override with -DENABLE\_LIBUHD=ON/OFF

--

-- Configuring LibUHD - C API support...

-- Dependency ENABLE\_LIBUHD = ON

-- Enabling LibUHD - C API support.

-- Override with -DENABLE\_C\_API=ON/OFF

--

-- Configuring LibUHD - Python API support...

-- Dependency ENABLE\_LIBUHD = ON

-- Dependency BOOST\_PYTHON\_FOUND = ON

-- Dependency HAVE\_PYTHON\_MODULE\_NUMPY = TRUE

-- Dependency PythonLibs\_FOUND = TRUE

-- Enabling LibUHD - Python API support.

-- Override with -DENABLE\_PYTHON\_API=ON/OFF

--

-- Configuring Examples support...

-- Dependency ENABLE\_LIBUHD = ON

-- Enabling Examples support.

-- Override with -DENABLE\_EXAMPLES=ON/OFF

--

-- Configuring Utils support...

-- Dependency ENABLE\_LIBUHD = ON

-- Enabling Utils support.

-- Override with -DENABLE\_UTILS=ON/OFF

--

-- Configuring Tests support...

-- Dependency ENABLE\_LIBUHD = ON

-- Enabling Tests support.

-- Override with -DENABLE\_TESTS=ON/OFF

--

-- Could NOT find LIBERIO (missing: LIBERIO\_LIBRARY LIBERIO\_INCLUDE\_DIR)

-- Could NOT find dpdk (missing: DPDK\_INCLUDE\_DIR)

--

-- Configuring LIBERIO support...

-- Dependency ENABLE\_LIBUHD = ON

-- Dependency LIBERIO\_FOUND = FALSE

-- Disabling LIBERIO support.

-- Override with -DENABLE\_LIBERIO=ON/OFF

--

-- Configuring USB support...

-- Dependency ENABLE\_LIBUHD = ON

-- Dependency LIBUSB\_FOUND = TRUE

-- Enabling USB support.

-- Override with -DENABLE\_USB=ON/OFF

--

-- Configuring GPSD support...

-- Dependency ENABLE\_LIBUHD = ON

-- Dependency ENABLE\_GPSD =

-- Dependency LIBGPS\_FOUND = TRUE

-- Disabling GPSD support.

-- Override with -DENABLE\_GPSD=ON/OFF

--

-- Configuring B100 support...

-- Dependency ENABLE\_LIBUHD = ON

-- Dependency ENABLE\_USB = ON

-- Enabling B100 support.

-- Override with -DENABLE\_B100=ON/OFF

--

-- Configuring B200 support...

-- Dependency ENABLE\_LIBUHD = ON

-- Dependency ENABLE\_USB = ON

-- Enabling B200 support.

-- Override with -DENABLE\_B200=ON/OFF

--

-- Configuring E300 support...

-- Dependency ENABLE\_LIBUHD = ON

-- Disabling E300 support.

-- Override with -DENABLE\_E300=ON/OFF

--

-- Configuring USRP1 support...

-- Dependency ENABLE\_LIBUHD = ON

-- Dependency ENABLE\_USB = ON

-- Enabling USRP1 support.

-- Override with -DENABLE\_USRP1=ON/OFF

--

-- Configuring USRP2 support...

-- Dependency ENABLE\_LIBUHD = ON

-- Enabling USRP2 support.

-- Override with -DENABLE\_USRP2=ON/OFF

--

-- Configuring X300 support...

-- Dependency ENABLE\_LIBUHD = ON

-- Enabling X300 support.

-- Override with -DENABLE\_X300=ON/OFF

--

-- Configuring N230 support...

-- Dependency ENABLE\_LIBUHD = ON

-- Enabling N230 support.

-- Override with -DENABLE\_N230=ON/OFF

--

-- Configuring MPMD support...

-- Dependency ENABLE\_LIBUHD = ON

-- Enabling MPMD support.

-- Override with -DENABLE\_MPMD=ON/OFF

--

-- Configuring N300 support...

-- Dependency ENABLE\_LIBUHD = ON

-- Dependency ENABLE\_MPMD = ON

-- Enabling N300 support.

-- Override with -DENABLE\_N300=ON/OFF

--

-- Configuring N320 support...

-- Dependency ENABLE\_LIBUHD = ON

-- Dependency ENABLE\_MPMD = ON

-- Enabling N320 support.

-- Override with -DENABLE\_N320=ON/OFF

--

-- Configuring E320 support...

-- Dependency ENABLE\_LIBUHD = ON

-- Dependency ENABLE\_MPMD = ON

-- Enabling E320 support.

-- Override with -DENABLE\_E320=ON/OFF

--

-- Configuring OctoClock support...

-- Dependency ENABLE\_LIBUHD = ON

-- Enabling OctoClock support.

-- Override with -DENABLE\_OCTOCLOCK=ON/OFF

--

-- Configuring DPDK support...

-- Dependency ENABLE\_MPMD = ON

-- Dependency DPDK\_FOUND = FALSE

-- Disabling DPDK support.

-- Override with -DENABLE\_DPDK=ON/OFF

--

--

--

-- Configuring priority scheduling...

-- Priority scheduling supported through pthread\_setschedparam.

-- Setting thread names is supported through pthread\_setname\_np.

--

-- Configuring high resolution timing...

-- High resolution timing supported through clock\_gettime.

--

-- Configuring module loading...

-- Module loading supported through dlopen.

--

-- Processing NI-RIO FPGA LVBITX Bitstreams...

-- Using x300.lvbitx\_base for codegen

-- Using x310.lvbitx\_base for codegen

--

-- USB support enabled via libusb.

--

-- Configuring interface address discovery...

-- Interface address discovery supported through getifaddrs.

--

-- Loading build info.

--

-- Adding B2XX device test target

-- Adding X3x0 device test target

-- Adding N3XX device test target

-- Adding E32x device test target

--

--

-- Configuring Manual support...

-- Dependency DOXYGEN\_FOUND = YES

-- Enabling Manual support.

-- Override with -DENABLE\_MANUAL=ON/OFF

--

-- Configuring API/Doxygen support...

-- Dependency DOXYGEN\_FOUND = YES

-- Enabling API/Doxygen support.

-- Override with -DENABLE\_DOXYGEN=ON/OFF

--

-- Configuring Man Pages support...

-- Dependency GZIP\_FOUND = TRUE

-- Dependency NOT\_WIN32 = TRUE

-- Enabling Man Pages support.

-- Override with -DENABLE\_MAN\_PAGES=ON/OFF

--

-- Python checking for virtualenv

-- Python checking for virtualenv - "assert hasattr(sys, 'real\_prefix')" failed

-- Utilizing the python install directory: /usr/local/lib/python2.7/dist-packages

--

-- ######################################################

-- # UHD enabled components

-- ######################################################

-- \* LibUHD

-- \* LibUHD - C API

-- \* LibUHD - Python API

-- \* Examples

-- \* Utils

-- \* Tests

-- \* USB

-- \* B100

-- \* B200

-- \* USRP1

-- \* USRP2

-- \* X300

-- \* N230

-- \* MPMD

-- \* N300

-- \* N320

-- \* E320

-- \* OctoClock

-- \* Manual

-- \* API/Doxygen

-- \* Man Pages

--

-- ######################################################

-- # UHD disabled components

-- ######################################################

-- \* LIBERIO

-- \* GPSD

-- \* E300

-- \* DPDK

--

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- \* You are building a development branch of UHD.

-- \* These branches are designed to provide early access

-- \* to UHD and USRP features, but should be considered

-- \* unstable and/or experimental!

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Building version: 3.14.1.HEAD-0-g0347a6d8

-- Using install prefix: /usr/local

-- Configuring done

-- Generating done

-- Build files have been written to: /home/klieberschnitzel/workarea-uhd/uhd/host/build

klieberschnitzel@ubuntu18-04-3:~/workarea-uhd/uhd/host/build$

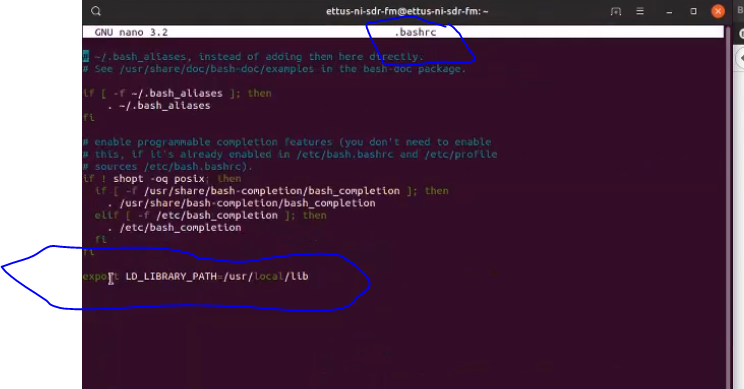
### make 🡪 klieberschnitzel@ubuntu18-04-3:~/workarea-uhd/uhd/host/build$ make

make test

sudo make install

sudo ldconfig

nano .bashrc



### Close and reopen terminal window

sudo uhd\_images\_downloader

uhd\_find\_devices

uhd\_usrp\_probe

cd workarea-uhd/host/utils

### If uhd\_find\_devices does not work. Need to copy rules to /etc/udev (for USRP B series).

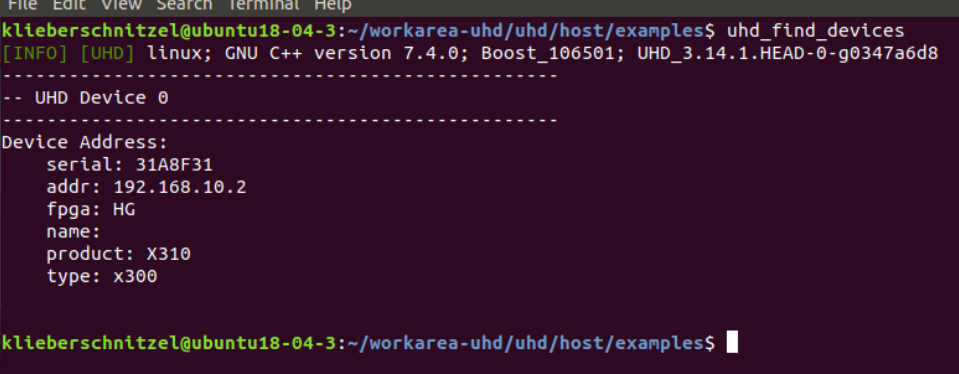
### NOTE: I didn’t need for x310

sudo cp uhd-usrp.rules /etc/udev/rules.d

sudo udevadm control --reload-rules

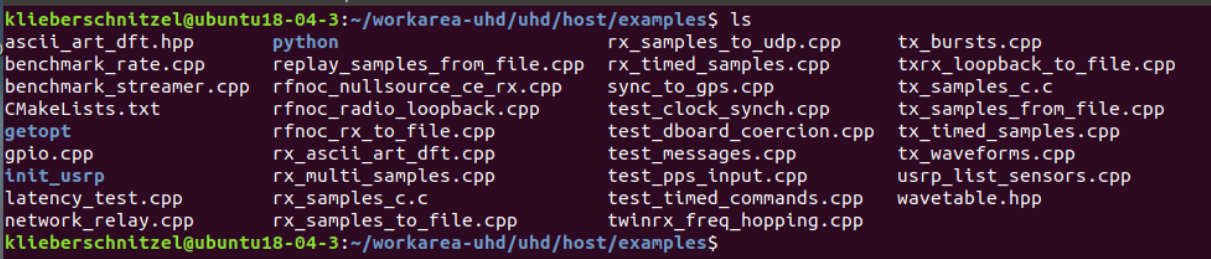
sudo udevadm trigger

uhd\_find\_devices



workarea-uhd/uhd/host/examples/ls

This lists the example code that can be run with the USRP for testing.



workarea-uhd/uhd/host/build/ls

