For GNURadio v3.8.5.0 installing on Patrick's Ubuntu 20.04 image

- ***Install dependencies (FOR ETTUS DEMOS ONLY) - run Neel's script as far as we can go (SD card space allowing), then install the following below***
  - *http://66.228.35.220/Scripts/*
  - *sudo ./Script_Install_Packages_Ubuntu_20.04.pl*
  - *sudo apt-get install libcodec2-dev*
  - *sudo apt-get install libcodec2-0.9*
  - *sudo apt-get install libavcodec-extra*
  - *sudo apt install python3-pip*
  - *sudo apt install libqt5svg5-dev*
  -
  - ***Replace examples folder in gr-digital folder with old one Patrick had generated***
  -
  -

ACHTUNG!!! For regular Non-Ettus demos start from Volk onward

- ***Remember to do 'sudo apt-get update' prior to executing the following commands***
- **and install new UHD v4.1.0.5**
  a) **sudo apt-get remove libuhd3.15.0 uhd-host**

- **Build and install Volk**

  Since Volk is no longer considered as a submodule of GNU Radio (GNU Radio commit #80c04479da962d048d41165081b026aafdaa0316),

  you **MUST FIRST** install Volk, and then install GNU Radio.

  The basic idea is the same, but instead of building Volk along with GNU Radio, you need to clone and build it separately. For this example, we will start in the home directory. You can, of course, use any directory you wish and the results will be the same.

  - `cd`
  - `git clone --recursive https://github.com/gnuradio/volk.git`
  - `cd volk`
  - `mkdir build`
  - `cd build`

  Note: In the following command, you can add `-DCMAKE_INSTALL_PREFIX=XXX` to install Volk into the PREFIX `XXX`; if not specified, then the PREFIX is `/usr/local`. See other CMake options in Common cmake flags.

  - `cmake -DCMAKE_BUILD_TYPE=Release -DPYTHON_EXECUTABLE=/usr/bin/python3 ../`
  - `make`
  - `make test`
  - `sudo make install`

If you're running Linux, then always remember to do the following command after installing any library:

- `sudo ldconfig`

- **Setup GNURadio**

For GNU Radio 3.8.5.0

Installing Dependencies

Refer to this page for your specific Linux distro to find how to install dependencies. For example, on Ubuntu 20.04 use this command.

See dependency list at top of document for all specific dependencies to install

Installing UHD v4.1.0.5

The UHD source is stored in a git repository. To download it, follow these instructions:

- `git clone https://github.com/EttusResearch/uhd.git`
- `cd uhd`
- `git checkout v4.1.0.5`
- `cd host`
- `sudo apt install python3-pip`
- `pip install setuptools  (May not be needed)`
- `mkdir build`
- `cd build`
- `cmake ../`
- `make`
- `make test  (This step is optional)`
- `sudo make install`

Installing GNU Radio

For this example, we will start in the home directory; you can, of course, use any directory you wish and the results will be the same.

- `cd`
- `git clone https://github.com/gnuradio/gnuradio.git`
- `cd gnuradio`

Note: In the following command, we will be checking out GNURadio v3.8.5.0. Volk needs to be built beforehand so the submodule update command can find the installed libvolk for cmake.

- `git checkout v3.8.5.0`
- `git submodule update --init --recursive`
- `mkdir build`
- `cd build`

Note: In the following command, you can add `-DCMAKE_INSTALL_PREFIX=XXX` to install GNU Radio into the PREFIX `XXX`; if not specified, then the PREFIX is `/usr/local`. See other CMake options in Common cmake flags.

- `cmake -DCMAKE_BUILD_TYPE=Release -DPYTHON_EXECUTABLE=/usr/bin/python3 ../`
- `make -j3`
  (e.g. if you want to use 3 CPU cores during the build. To use 8 do -j8, to use 1 leave out the -j flag.)
- `sudo make install`

If you're running Linux, then always remember to do the following command after installing any library:

- `sudo ldconfig`

Go to Finding the Python library to set your PYTHONPATH and LD_LIBRARY_PATH.

After setting these environment variables, you need to do `sudo ldconfig` again for the Linux dynamic library loader to find the just-installed GNU Radio libraries.

If you have installed in a custom path with `-DCMAKE_INSTALL_PREFIX=XXX`, you will need to add that path to $PATH in order to find gnuradio-companion.

NOTE: There may be a dependency for Qt50wt6 and an error message may appear in cmake where the qt gui controls aren't installed. Add the following in order to ensure qt-gui builds properly.

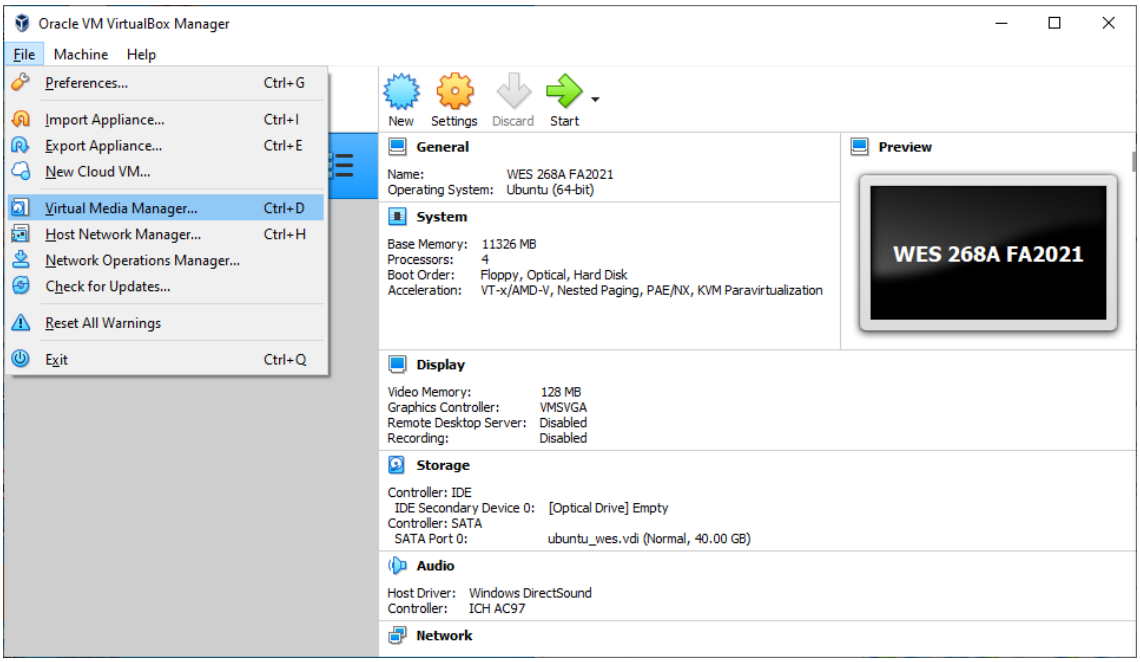
- **sudo apt install libqwt-qt5-dev python3-click-plugins libqt5opengl5-dev python3-zmq**

Additional Python dependencies:

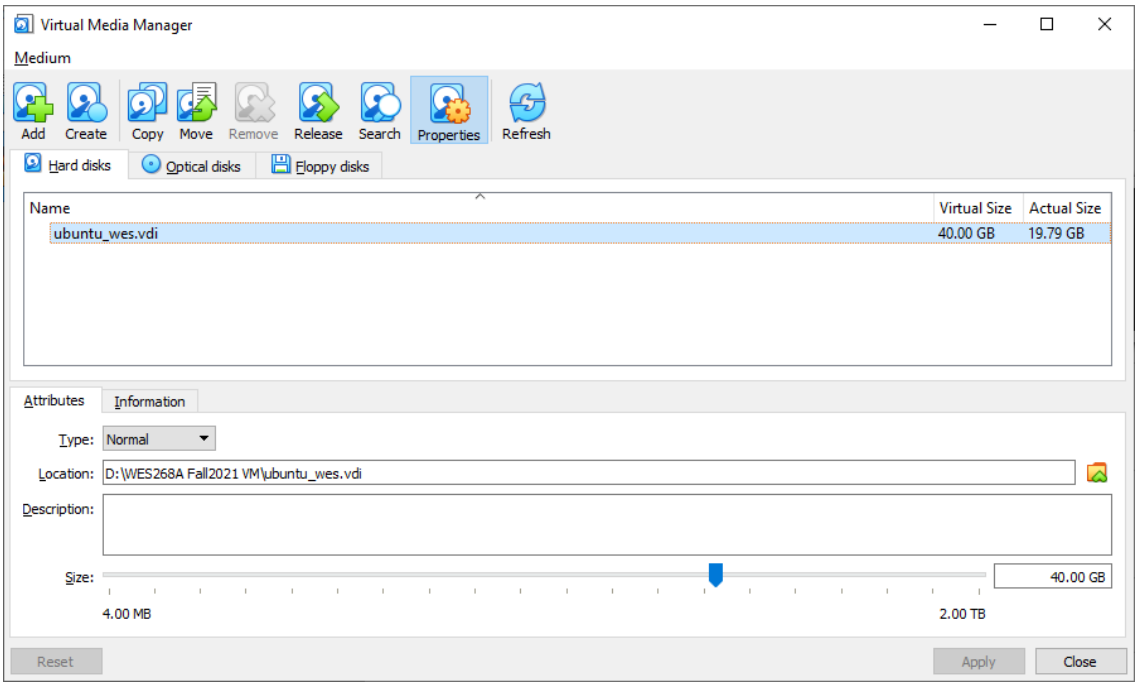
- **pip install ruamel.yaml**

- **Add gr-ettus to GNURadio**
  a) `$ git clone https://github.com/EttusResearch/gr-ettus.git`
  b) `$ cd gr-ettus`
  c) `$ git checkout a59219ad0c15e6c3f903a15dc26af42705bb4774`
  d) `$ mkdir build-host`
  e) `$ cd build-host`
  f) `$ cmake ..`
  g) `$ make -j4`
  h) `$ sudo make install`
  i) `$ sudo ldconfig`

- Increasing Virtualbox Disk Image Size
    a) Go to Virtualbox Virtual Media Manager. Ensure the VM is shutdown/off prior to going here.



    b) Resize virtual disk image to at least 40GB



    c) Boot into Ubuntu and run ‘gparted’ in terminal. Sudo password is **WES268**

Setting up Xilinx Vivado 2019.1 (needed to build USRP RFNoC FPGA Image)

- Dependencies
  - sudo apt install libncurses5 libtinfo5
- Download Xilinx Vivado HLx Design Suite as linked in the following KB article:
  - https://files.ettus.com/manual/md_usrp3_build_instructions.html
  - In case the above link does not work, the Xilinx download can be accessible here: https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/archive.html
- Setup license server
  - https://kastner.ucsd.edu/ryan/vivado-installation/
  - export XILINXD_LICENSE_FILE=2100@cselm2.ucsd.edu
- If Xilinx is installed in separate folder, need to do symbolic link in /opt for ettus scripts to work
  - cd /opt
  - sudo ln -s /media/wes/zdrive/tools/Xilinx .

  Optional Debugging
  - https://support.xilinx.com/s/question/0D52E00006iHlpOSAS/vivado-20172-simulation-xsim-433409-failed-to-compile-generated-c-file?language=en_US
  - cd /opt/Xilinx/Vivado/2019.1/bin
    - Rename gcc to gcc.orig
    - vi gcc
      - Add in the following script

```
#!/bin/bash

cmd=clang
logfile=/tmp/xlog

echo LD_LIBRARY_PATH=$LD_LIBRARY_PATH >>
$logfile
echo $0 $* >> $logfile
$0.orig $* 2>&1 | tee -a $logfile
```

      - Make script executable as if it's gcc:

```
chmod a+x gcc
```

      - Logs are stored in /tmp/xlog

Install OpenCV libraries in Linux environment

- **Follow https://linuxize.com/post/how-to-install-opencv-on-ubuntu-20-04/**
  - `sudo apt update`
  - `sudo apt install libopencv-dev python3-opencv`

Additional library dependencies (xtensor, etc.)

- **Xtl and Xtensor (provides numpy functionality for easier porting of MATLAB routines)**
  - **git clone https://github.com/xtensor-stack/xtl.git**
    - **cd xtl; mkdir build; cd build; cmake ../**
    - **make; sudo make install; sudo ldconfig**
  - **git clone https://github.com/xtensor-stack/xtensor.git**
    - **cd xtensor; mkdir build; cd build; cmake ../**
    - **make; sudo make install; sudo ldconfig**

Performance Improvements:

UHD: Thread priority scheduling

When UHD software spawns a new thread, it may try to boost the thread's scheduling priority. If setting the new priority fails, the UHD software prints a warning to the console, as shown below. This warning is harmless; it simply means that the thread will retain a normal or default scheduling priority.

```
UHD Warning:
    Unable to set the thread priority. Performance may be negatively affected.
    Please see the general application notes in the manual for instructions.
    EnvironmentError: OSError: error in pthread_setschedparam
```

Linux Notes:

Non-privileged users need special permission to change the scheduling priority. Add the following line to the file /etc/security/limits.conf:

```
@GROUP    - rtprio    99
```

Replace GROUP with a group in which your user is a member. You may need to log out and log back into the account for the settings to take effect. In most Linux distributions, a list of groups and group members can be found in the file /etc/group.