GETTING STARTED GUIDE

NI RF Data Recording API

This document provides basic information about how to setup the NI RF Data Recording API.

Contents

System Requirements 2
Software 2
Hardware 2
Building and Installing UHD Python API 4
Pre-requirements 4
Install Python 4
Un-install pre-installed UHD 4
Install other Required Packages 5
UHD Python API Installation 5
UHD Installation Test 7
Connect the USRP 7
USRP Probe 8
Load FPGA Images onto the Device 8
FPGA Image Flavors 8
Downloading the UHD FPGA Images 8
Load the Images onto the On-board Flash 9
Increase Performance of Interface 9
UHD Python API Test 11
Benchmarking USRP Speed in Python 11
Install RF Data Recording API Support Packages 12
SigMF Installation 12
npTDMS to Read TDMS Files 12
Colour Change of Linux Terminal 12
Setup Networking 13
Setup the Host Interface 13
Multiple Devices per Host 13
Change the USRP's IP address 13
Network Setup Example 14
Setup Hardware 15
Configuring the USRPs system: Cabled RF 15

Appendix 15
Troubleshooting 15
Known Issues and Limitations 15

System Requirements

Software

Ubuntu 20.04 is used in our test environment. However, you can build UHD on different operating systems. Look to: Building and Installing the USRP Open-Source Toolchain (UHD and GNU Radio) on Linux - Ettus Knowledge Base

Hardware

To use the NI RF Data Recording API, you need at least one NI RF USRP device (RF Mode is Tx or Rx only). The API has been tested on NI USRP X310 only. The devices should be connected to single or different host computers based on the operation mode and the investigated application. Figure 1 shows the setup of three TX stations and one RX Station. Table 1 presents the required hardware for this configuration.

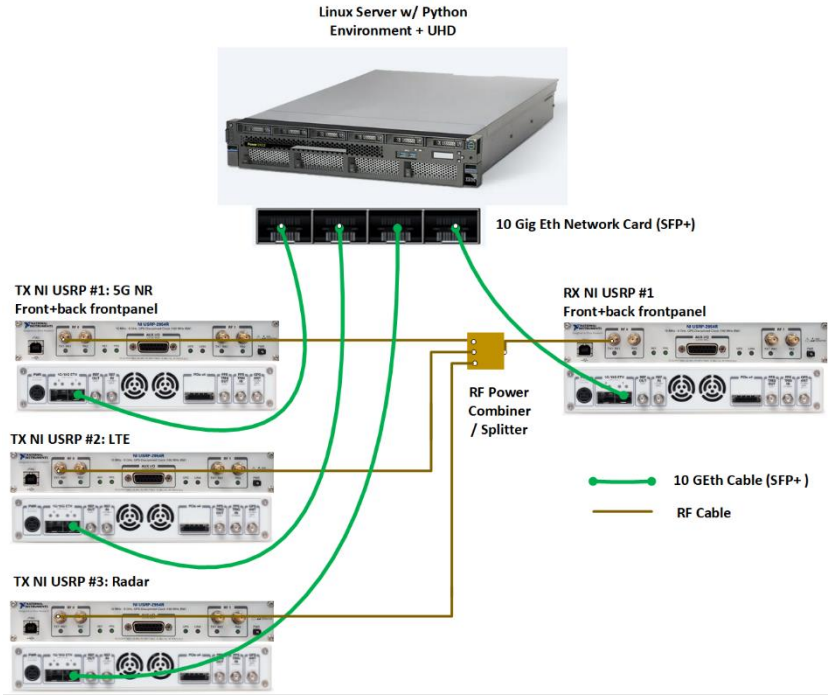

Figure 1 Hardware Configuration

Table 1 Required Hardware Accessories (cabled setup)

Item	Number
Host PC (Linux Server)	1
X310 USRP Device	4
SMA Cable	4
30 dB Attenuator	1
10 Gig Eth Cable (SFP+)*	4

Note: The USRP X310 can be connect to the host machine using different options. Look to Network Connectivity Guide.

- Host PC Linux server: Recommended—with 10Gig Eth card (SFP+).
- SMA cable: Female/female cable that is included with the USRP RIO device.
- USRP RIO device: USRP-X310, USRP-2940/2942/2943/2944/2950/2952/2953/2954 Software Defined Radio Reconfigurable Devices with 40 MHz, 120 MHz, or 160 MHz bandwidth.
- Attenuator with 30 dB attenuation and male/female SMA connectors that are included with the USRP RIO device.

Ensure your host has enough free disk space and RAM.

Caution: Before using your hardware, read all product documentation to ensure compliance with safety, EMC, and environmental regulations.

Caution: To ensure the specified EMC performance, operate the RF devices only with shielded cables and accessories.

Caution: To ensure the specified EMC performance, the length of all I/O cables except for those connected to the GPS antenna input of the USRP device must be no longer than 3 m (10 ft.).

Caution: The USRP RIO RF devices are not approved or licensed for transmission over the air using an antenna. As a result, operating this product with an antenna may violate local laws. Ensure that you are in compliance with all local laws before operating this product with an antenna.

Building and Installing UHD Python API

The following instructions are based on Ubuntu 20.04 OS. To support different OS, look to the links below.

- Building UHD: Building and Installing the USRP Open-Source Toolchain (UHD and GNU Radio) on Linux - Ettus Knowledge Base
- UHD Python API: USRP Hardware Driver and USRP Manual: Python API (ettus.com)

Pre-requirements

To use the RF Data Recording API, the UHD version should be at least 4.2. To be able to build UHD version 4.xx based on the current UHD source code, the CMake requires at least Python 3.5. Ubuntu 20.04 uses Python version 3.8. For more information about UHD build dependences look to: https://files.ettus.com/manual/page_build_guide.html.

If the user would like to use a newer Python version, look to the next section. The getting started guide has been made based on Python 3.9. Otherwise, skip install Python section.

Install Python

- To install Python, assume 3.9.10 (used in our test environment), run the following commands on the terminal:
- Update the packages list and install the prerequisites:

```
sudo apt update
sudo apt install software-properties-common
```

- Add the deadsnakes PPA to your system's sources list:

```
sudo add-apt-repository ppa:deadsnakes/ppa
```

- When prompted, press `[Enter]` to continue.

```
sudo apt install python3.9
```

- Verify that the installation was successful by typing:

```
python3.9 --version
```

- Make Python 3.9 your default:

```
sudo update-alternatives --install /usr/bin/python python /usr/bin/python3.9 1
sudo update-alternatives --set python /usr/bin/python3.9
alias python='/usr/bin/python3.9'
```

- Export the path to Python. For example, go to: /home/user/.bashrc, then add the following line:

```
export PYTHONPATH=/usr/local/lib/python3.9/site-packages/
```

Un-install pre-installed UHD

- If UHD is already installed, check the UHD version by issuing UHD commands on terminal, for example:

```
Uhd_find_devices
```

- Uninstall UHD if it is already installed on the system and its version is less than 4.0. For example, if the old UHD is UHD_3.15.0:

```
sudo apt-get remove libuhd3.15.0
```

- If you installed UHD manually, apt cannot know about this. To uninstall it, go to the UHD build directory and run:

```
 sudo make uninstall
```

Install other Required Packages

- Pip, pandas, sympy, scipy, and NumPy:

```
sudo apt install python3-pip
sudo python3.9 -m pip install pandas sympy numpy scipy
```

UHD Python API Installation

- Currently, the Python API is part of Master branch (EttusResearch/uhd: The USRP™ Hardware Driver Repository (github.com)), but not part of the release. You need to build UHD binary to get the UHD Python API.
- To install the Python API when building UHD from source, make sure you have the CMake variable ENABLE_PYTHON_API set to ON (e.g., by running CMake -DENABLE_PYTHON_API=ON). UHD requires Python header files in order to compile the Python API. Installing the Python API is described here: https://files.ettus.com/manual/page_python.html
- For building and installing UHD, follow the instructions given in the following link: Building and Installing the USRP Open-Source Toolchain (UHD and GNU Radio) on Linux - Ettus Knowledge Base. The commands summary is listed below.
- To install any missing package, assume NumPy is missing, use the following command where Python version should be specified

```
sudo python3.9 -m pip install numpy
```

Commands Summary:

- On Ubuntu 20.04 system, run:

```
   sudo apt-get -y install autoconf automake build-essential ccache cmake cpufrequtils
doxygen ethtool fort77 g++ gir1.2-gtk-3.0 git gobject-introspection gpsd gpsd-clients
inetutils-tools libasound2-dev libboost-all-dev libcomedi-dev libcppunit-dev libfftw3-bin
libfftw3-dev libfftw3-doc libfontconfig1-dev libgmp-dev libgps-dev libgsl-dev
liblog4cpp5-dev libncurses5 libncurses5-dev libpulse-dev libqt5opengl5-dev libqwt-qt5-dev
libsdl1.2-dev libtool libudev-dev libusb-1.0-0 libusb-1.0-0-dev libusb-dev libxi-dev
libxrender-dev libzmq3-dev libzmq5 ncurses-bin python3-cheetah python3-click python3-
click-plugins python3-click-threading python3-dev python3-docutils python3-gi python3-gi-
cairo python3-gps python3-lxml python3-mako python3-numpy python3-numpy-dbg python3-
opengl python3-pyqt5 python3-requests python3-scipy python3-setuptools python3-six
python3-sphinx python3-yaml python3-zmq python3-ruamel.yaml swig wget
```

- To build UHD from source code, clone the GitHub repository, check out a branch or tagged release of the repository, 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
```

- Next, create a build folder within the repository.

```
cd host
mkdir build
cd build
```

- Next, invoke CMake.

```
cmake ..
```

- If CMake can't find the Python headers or library, specify the `PYTHON_INCLUDE_DIR` and / or `PYTHON_LIBRARY` CMake variables manually.

```
cmake -DPYTHON_INCLUDE_DIR=/usr/include/python3.9 -DPYTHON_LIBRARY=/usr/lib/x86_64-linux-gnu/libpython3.9.so ..
```

Note: If the CMake is not successful, look to: [Buipythonlding and Installing the USRP Open-Source Toolchain (UHD and GNU Radio) on Linux - Ettus Knowledge Base]

- The output from CMake should be something like the figure below:

```
- Utilizing the python install directory: /usr/local/lib/python3.8/site-packages
-
- ######################################################
- # UHD enabled components
- ######################################################
-   * LibUHD
-   * LibUHD - C API
-   * LibUHD - Python API
-   * Examples
-   * Utils
-   * Tests
-   * USB
-   * B100
-   * B200
-   * USRP1
-   * USRP2
-   * X300
-   * MPMD
-   * SIM
-   * N300
-   * N320
-   * E320
-   * E300
-   * X400
-   * OctoClock
-   * Manual
-   * API/Doxygen
-   * Man Pages
-
- ######################################################
- # UHD disabled components
- ######################################################
-   * DPDK
```

Figure 2 Output of CMake Command: UHD Python API is enabled

- Once the CMake command succeeds without errors, build UHD.

```
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.

UHD Installation Test

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.

```
[INFO] [UHD] linux; GNU C++ version 9.4.0; Boost_107100; UHD_4.2.0.git-413-g159c21b7#
```

Connect the USRP

The installation of UHD should now be complete. At this point, connect the USRP to the host computer.

- Look to **Setup Networking Section** in this document for more details about how to setup the network.
- Look to X300/X310 - Ettus Knowledge Base for more information about choosing a host interface for X300/X310 USRPs.
- Look to USRP Hardware Driver and USRP Manual: USRP X3x0 Series (ettus.com) for more information about setup networking.

If the interface is Ethernet, then open a terminal window, and try to ping the USRP with "ping 192.168.10.2". The USRP should respond to the ping requests. Also try running "uhd_find_devices", the output should be similar to the following:

```
agaber@dre-elbe-s05:~/workarea$ uhd_find_devices
[INFO] [UHD] linux; GNU C++ version 9.3.0; Boost_107100; UHD_4.2.0.git-290-g2c7ce2db
--------------------------------------------------
-- UHD Device 0
--------------------------------------------------
Device Address:
    serial: 30F1628
    addr: 192.168.40.2
    fpga: HG
    name:
    product: X310
    type: x300
```

USRP Probe

To check if the UHD image of the connected USRP is *compatible* with the installed UHD, run the following command:

```
uhd_usrp_probe --args addr=192.168.10.2
```

If the output of `uhd_usrp_probe` does not show any warning, you do not need to update Image. However, if there are errors regarding the FPGA version compatibility number, you will have to update the FPGA image before you can start using your USRP.

Load FPGA Images onto the Device

For more information, look to: **USRP Hardware Driver and USRP Manual: USRP X3x0 Series (ettus.com)**

FPGA Image Flavors

The USRP-X Series devices contains two SFP+ ports for the two Ethernet channels. Because the SFP+ ports support both 1 Gigabit (SFP) and 10 Gigabit (SFP+) transceivers, several FPGA images are shipped with UHD to determine the behavior of the above interfaces.

Table 2 UHD Images and related SFP+ interface

FPGA Image Flavor	SFP+ Port 0 Interface	SFP+ Port 1 Interface
HG (Default)	1 Gigabit Ethernet	10 Gigabit Ethernet
XG	10 Gigabit Ethernet	10 Gigabit Ethernet
HA	1 Gigabit Ethernet	Aurora
XA	10 Gigabit Ethernet	Aurora

Note: The Aurora images need to be built manually from the FPGA source code.

FPGA images are shipped in 2 formats:

- **LVBITX**: LabVIEW FPGA configuration bitstream format (for use over PCI Express and Ethernet)
- **BIT**: Xilinx configuration bitstream format (for use over Ethernet and JTAG)

Downloading the UHD FPGA Images

To get the latest images, simply use the uhd_images_downloader script (see Firmware and FPGA Images). On Unix systems, use this command:

```
$ [sudo] uhd_images_downloader
```

Note: It is recommended to download the HG image, you will be able to use both the 1 Gig Eth and the 10 Gig Eth connection.

Load the Images onto the On-board Flash

To change the FPGA image stored in the on-board flash, the USRP-X Series device can be reprogrammed over the network or PCI Express. Once you have programmed an image into the flash, that image will be automatically loaded on the FPGA during the device boot-up sequence.

Note: Different hardware revisions require different FPGA images. Determine the revision number from the sticker on the rear of the device. If you are manually specifying an FPGA path, the utility will not try to detect your device information, and you will need to use this number to select which image to burn.

Note: The image loader utility will default to using the appropriate BIT file if no custom FPGA image path is specified, but it is compatible with BIN, BIT, and LVBITX images.

Use the `uhd_image_loader` utility to load the image from the default path and update the FPGA image. On the command line, run:

```
uhd_image_loader --args="type=x300,addr=<IP address>"
```

- To specify the image type, run the following command:

```
uhd_image_loader --args="type=x300,addr=192.168.40.2,fpga=HG"
```

- To load the FPGA image from a given path, run the following command:

```
uhd_image_loader --args="type=x300,addr=192.168.40.2, --fpga-
path=”your_path\usrp_x310_fpga_HG.bit”
```

- Wait a few minutes until it is successful, then restart the USRP.
- To check if the UHD image of the connected USRP is compatible with the installed UHD, run the following command:

```
uhd_usrp_probe --args addr=192.168.10.2
```

It should show the USRPs specifications without any error.

Increase Performance of Interface

To know the network interfaces, run `ifconfig` command. The output looks like the following:

```
enp3s0f0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 9000
        inet 192.168.40.1  netmask 255.255.255.0  broadcast 192.168.40.255
        inet6 fe80::b1e:9f17:2b67:6030  prefixlen 64  scopeid 0x20<link>
        ether 90:e2:ba:d1:c8:28  txqueuelen 1000  (Ethernet)
        RX packets 2228470  bytes 175930400 (175.9 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2115340  bytes 419244814 (419.2 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp3s0f1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 9000
        inet 192.168.60.1  netmask 255.255.255.0  broadcast 192.168.60.255
        inet6 fe80::ada4:5fc3:9320:4628  prefixlen 64  scopeid 0x20<link>
        ether 90:e2:ba:d1:c8:29  txqueuelen 1000  (Ethernet)
        RX packets 1627865  bytes 122994502 (122.9 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1521434  bytes 315288179 (315.2 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp7s0f0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 9000
        inet 192.168.80.1  netmask 255.255.255.0  broadcast 192.168.80.255
        inet6 fe80::5c37:52b0:6069:26de  prefixlen 64  scopeid 0x20<link>
        ether 3c:fd:fe:9f:d5:c0  txqueuelen 1000  (Ethernet)
        RX packets 2240387  bytes 5245791862 (5.2 GB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1523059  bytes 125786867 (125.7 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 3 Output of ifconfig command

- Set the MTU size of all Eth ports connected to USRPs to 9000 to get maximum sampling rate. Assume, we need to configure the network interface of "enp3s0f0".:

```
sudo ip link set enp3s0f0 mtu 9000
```

- Resize buffer to be more efficient. To check the default and maximum amount for the receive socket memory, run the following commands:

```
$ cat /proc/sys/net/core/rmem_default
$ cat /proc/sys/net/core/rmem_max
```

- To check the default and maximum amount for the send socket memory, run the following commands:

```
$ cat /proc/sys/net/core/wmem_default
$ cat /proc/sys/net/core/wmem_max
```

- To set them, use the following Commands:

```
sudo sysctl -w net.core.wmem_max=24912805
sudo sysctl -w net.core.rmem_max=24912805
```

- To check if the value has been changed:

```
$ sudo sysctl net.core.rmem_max
$ sudo sysctl net.core.wmem_max
```

- To make the change permanent, add the following lines to the /etc/sysctl.conf file, which is used during the boot process:

```
net.core.rmem_default=24912805
net.core.wmem_default=24912805
net.core.rmem_max=24912805
```

```
net.core.wmem_max=24912805
```

For more information, look to:

USRP Hardware Driver and USRP Manual: Transport Notes (ettus.com)

UHD Python API Test

Once it's built and installed, you'll be able to import the UHD Python module. The Python has an object called MultiUSRP which is an equivalent of the C++ multi_usrp API. The methods on both classes are the same and take the same arguments.

- On the terminal, run the following commands:

```
python3.9
import uhd
usrp = uhd.usrp.MultiUSRP("addr=192.168.40.2")
samples = usrp.recv_num_samps(10000, 100e6, 1e6, [0], 50)
print(samples[0:10])
```

- If you able to import UHD and open the MultiUSRP session, then it is working fine.

Note: If you are not able to import UHD and open the MultiUSRP session, then the reason could be the following:

- The UHD Python API is not enabled during UHD build
- The UHD Python API created using another python version
- Or you need to export the path to python. For example, go to: /home/user/.bashrc, then add the following line:

```
export PYTHONPATH=/usr/local/lib/python3.9/site-packages/
```

Benchmarking USRP Speed in Python

The maximum sampling rate of UHD Python API is 100 MS/s. You can run this command on terminal:

```
python3.9 /usr/local/lib/uhd/examples/python/benchmark_rate.py --
args="type=x300,addr=192.168.40.2,num_recv_frames=1000" --rx_rate 100e6 --channels 0 --
rx_channels 0
```

The output on terminal looks like the following. The number of dropped samples is zero.

```
[INFO] [UHD] linux; GNU C++ version 9.3.0; Boost_107100; UHD_4.2.0.git-290-g2c7ce2db
[INFO] [X300] X300 initialization sequence...
[INFO] [X300] Maximum frame size: 8000 bytes.
[INFO] [GPS] Found an internal GPSDO: LC_XO, Firmware Rev 0.929a
[INFO] [X300] Radio 1x clock: 200 MHz
[WARNING] [0/Radio#0] Non-critical minor compat number mismatch for `0/Radio#0':
Expecting 0.1, got 0.0.
[WARNING] [0/Radio#1] Non-critical minor compat number mismatch for `0/Radio#1':
Expecting 0.1, got 0.0.[14:40:54.59]
 [INFO] (MainThread) Using Device: Single USRP:  Device: X-Series Device  Mboard 0: X310
RX Channel: 0    RX DSP: 0    RX Dboard: A    RX Subdev: UBX RX  RX Channel: 1    RX DSP:
```

```
1    RX Dboard: B    RX Subdev: UBX RX  TX Channel: 0    TX DSP: 0    TX Dboard: A    TX
Subdev: UBX TX  TX Channel: 1    TX DSP: 1    TX Dboard: B    TX Subdev: UBX TX

[14:40:54.59] [INFO] (MainThread) Selected [0] RX channels and no TX channels

[14:40:54.59] [INFO] (MainThread) Setting device timestamp to 0...

[14:40:54.60] [INFO] (Thread-1  ) Testing receive rate 100.000 Msps on 1 channels

[14:41:04.61] [DEBUG] (MainThread) Sending signal to stop!

[14:41:04.61] [DEBUG] (MainThread) RX Statistics Dictionary: {'num_rx_samps': 1000886216,
'num_rx_dropped': 0, 'num_rx_overruns': 0, 'num_rx_seqerr': 0, 'num_rx_timeouts': 0,
'num_rx_late': 0}

[14:41:04.61] [DEBUG] (MainThread) TX Statistics Dictionary: {}[

14:41:04.61] [DEBUG] (MainThread) TX Async Statistics Dictionary: {}[14:41:04.61] [INFO]
(MainThread) Benchmark rate summary:    Num received samples:     1000886216    Num
dropped samples:      0    Num overruns detected:    0    Num transmitted samples:  0
Num sequence errors (Tx): 0    Num sequence errors (Rx): 0    Num underruns detected:   0
Num late commands:        0    Num timeouts (Tx):        0    Num timeouts (Rx):        0
```

Install RF Data Recording API Support Packages

SigMF Installation

The Signal Metadata Format (SigMF) specifies a way to describe sets of recorded digital signal samples with metadata written in JSON. See SigMF/sigmf-spec.md at sigmf-v1.x · gnuradio/SigMF (github.com).

- Install sigmf for a selected python

```
sudo python3.9 -m pip install sigmf
```

- To run the included QA tests:

```
sudo python3.9 -m pip install pytest
```

npTDMS to Read TDMS Files

To create wireless standard complaint waveforms, the NI RFmx Waveform Creator, i.e. the NI RFmx 5G NR waveform creator. To read TDMS files, the npTDMS is used. The npTDMS is a cross-platform Python package for reading and writing TDMS files as produced by LabVIEW, and is built on top of the numpy package. For more information, see Welcome to npTDMS's documentation — npTDMS 1.3.1 documentation. To install npTDSM, run the following command:

```
sudo python3.9 -m pip install npTDMS
```

Note: There are optional features available that require additional dependencies. These are hdf for hdf export, pandas for pandas DataFrame export, and thermocouple_scaling for using thermocouple scalings. You can specify these extra features when installing npTDMS to also install the dependencies they require:

```
sudo python3.9 -m pip install npTDMS[hdf,pandas,thermocouple_scaling]
```

Colour Change of Linux Terminal

To make the output coloured on Python terminal, download those packages:

```
sudo python3.9 -m pip install colored
sudo python3.9 -m pip install termcolor
```

Setup Networking

The USRP-X Series only supports Gigabit and Ten Gigabit Ethernet and will not work with a 10/100 Mbps interface.

Please note that 10 Gigabit Ethernet defines the protocol, not necessary the medium. For example, you may use 10GigE over optical with optical SFP+ transceiver modules.

Note: For more info, see: USRP Hardware Driver and USRP Manual: USRP X3x0 Series (ettus.com)

Note: For more Info about supported host interfaces, see: X300/X310 - Ettus Knowledge Base

Setup the Host Interface

The USRP-X Series communicates at the IP/UDP layer over the Gigabit and Ten Gigabit Ethernet. The default IP address for the USRP X300/X310 device depends on the Ethernet Port and interface used. You must configure the host Ethernet interface with a static IP address on the same subnet as the connected device to enable communication, as shown in the following table:

Ethernet Interface	USRP Ethernet Port	Default USRP IP Address	Host Static IP Address	Host Static Subnet Mask	Address EEPROM key
Gigabit	Port 0 (HG Image)	192.168.10.2	192.168.10.1	255.255.255.0	`ip-addr0`
Ten Gigabit	Port 0 (XG Image)	192.168.30.2	192.168.30.1	255.255.255.0	`ip-addr2`
Ten Gigabit	Port 1 (HG/XG Image)	192.168.40.2	192.168.40.1	255.255.255.0	`ip-addr3`

As you can see, the X300/X310 actually stores different IP addresses, which all address the device differently: Each combination of Ethernet port and interface type (i.e., Gigabit or Ten Gigabit) has its own IP address. As an example, when addressing the device through 1 Gigabit Ethernet on its first port (Port 0), the relevant IP address is the one stored in the EEPROM with key `ip-addr0`, or 192.168.10.2 by default.

See **Configuring the host's IP address** on details how to change your machine's IP address and MTU size to work well with the X300.

Multiple Devices per Host

For maximum throughput, one Ethernet interface per USRP is recommended, although multiple devices may be connected via an Ethernet switch. In any case, each Ethernet interface should have its own subnet, and the corresponding USRP device should be assigned an address in that subnet.

Change the USRP's IP address

You may need to change the USRP's IP address for several reasons:

- to satisfy your particular network configuration
- to use multiple USRP-X Series devices on the same host computer

- to set a known IP address into USRP (in case you forgot)

To change the USRP's IP address, you must know the current address of the USRP, and the network must be setup properly as described above. You must also know which IP address of the X300 you want to change, as identified by their address EEPROM key (e.g. `ip-addr3`, see the table above). Run the following commands:

```
cd <install-path>/lib/uhd/utils
./usrp_burn_mb_eeprom --args="type=x300,addr=192.168.40.2" --values="ip-addr3=192.168.60.2"
```

Power cycle the USRP to get the new IP.

Network Setup Example

Figure 1 shows four X310 USRPs. Assume:

- The UHD image is HG
- Four 10 Gig Eth cables (SFP+) are available

Then, do the following:

- Connect the SFP+ Port 1 of each USRP to the Linux server. Figure 4 shows the rear panel.

Figure 4 X310 Rear Panel

To change the IP address of USRPs, do that one by one. Other USRPs should be off or leave the related Ethernet interfaces not configured or set the related Ethernet interfaces down. Since the default IP of USRPs is the same.

Configuration for USRP-X Series device 0:

The default ip-addr3 is 192.168.40.2. No need to change. So, the configuration will be:

- Ethernet interface IPv4 address: `192.168.40.1`
- Ethernet interface subnet mask: `255.255.255.0`
- USRP-X Series device IPv4 address: `192.168.40.2`

Configuration for USRP-X Series device 1:

First, switch off device 0 or set its Ethernet interface down. Then, use the following default config to be able to communicate with USRP:

- Ethernet interface IPv4 address: `192.168.40.1`
- Ethernet interface subnet mask: `255.255.255.0`
- USRP-X Series device IPv4 address: `192.168.40.2`

To change the IP address, i.e. to 192.168.50.2, run the following command:

```
./usrp_burn_mb_eeprom --args="type=x300,addr=192.168.40.2" --values="ip-addr3=192.168.50.2"
```

Then, Power cycle the USRP to get the new IP.

Change the IP on host to 192.168.50.1. To activate the new IP on the Host, set the interface to be Up and then Down. The configuration will be then:

- Ethernet interface IPv4 address: `192.168.50.1`
- Ethernet interface subnet mask: `255.255.255.0`
- USRP-X Series device IPv4 address: `192.168.50.2`

Repeat the above steps for the other USRPs.

Setup Hardware

The following descriptions assume that there are four USRPs. They are referred to as TX USRP1, TX USRP2, TX USRP3, and RX USRP as it is shown in Figure 1.

Configuring the USRPs system: Cabled RF

1. Ensure the USRP devices are properly connected to the host systems
2. Complete the following steps to create RF connections as shown in Figure 1.
    1. Connect RF0/TX1 ports on TX USRP1, TX USRP2 and TX USRP3 to RF cables
    2. Connect the other ends of the three RF cables to RF Power splitter.
    3. Connect the common port of power splitter to 30 dB attenuator
    4. Connect the other end of the attenuator to RF cable
    5. Connect the RF cable to RF0/TX1 port on RX USRP
3. Power on the USRP devices.

The RF cables and power splitter should support the operating frequency.

Appendix

Troubleshooting

Known Issues and Limitations

- Tested on X310 USRP only

[1] If you are transmitting over the air, make sure to consider the instructions given in the "RF Multi Station Mode: Over-the-Air Transmission" section. The USRP devices and NI-5791 are not approved or licensed for transmission over the air using an antenna. As a result, operating those products with an antenna may violate local laws.