O-RAN Implementation

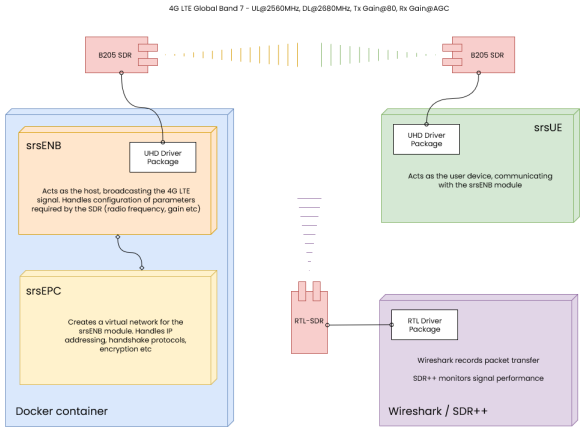
Document Summary

The O-RAN R&D project enables the implementation of a 4G LTE or 5G indoor cellular testbed using open-source software (srsRAN) and Software Defined Radios (SDRs). The aim is to replicate the functionalities of a mobile base station, including the Radio Access Network (RAN) and mobile core network, using virtualization technologies and commercial off-the-shelf (COTS) hardware to offer an alternative to traditional proprietary solutions. This document is a technical guide for implementation.

Document Contents

---

srsRAN environment overview:



System and hardware requirements

There are two main parts of a mobile base-station – the radio access network (RAN) and the mobile core network. These parts are typically bundled and sold as proprietary hardware and software. However, the advance in virtualization technologies and commercial off-the-shelf (COTS) hardware has made it possible to implement such a base station using open-source software and Software Defined Radios (SDR).

These installation and operation steps specifically apply to the equipment used in our testbed. If the SDR module you are using is different, the driver steps may not be applicable. Please refer to the manufacturer's driver installation guide for your SDR.

Ettus USRP B205mini-i: 1x1, 70MHz-6GHz SDR/Cognitive Radio

We are using the Teltonika Mobile Indoor 3dBi Tilt/Swivel Antenna with SMA Connector for 3G, 4G and LTE. The external connections and respective power information for the B205-mini is as follows:

Component ID	Description	Details
USB3	USB Connector	USB 3.0
J1 Antenna	TRX (Transmission and Reception)	TX power +20 dBm max RX power -15 dBm max
J2 Antenna	RX2 (Reception)	RX power -15 dBm max
J3 Antenna	External 10 MHz/PPS Reference (Clock)	+15 dBm max

Laptop

OS
Ubuntu 22.04.4 LTS

CPU
Intel Core i5-8365 1.60GHz 4cores/8threads

RAM
16GB DDR4

Networking
WWAN Adapter (with SIM)
WiFi

Software Elements

Required dependencies and build tools:

- cmake
- gcc (v9.4.0 or later) **OR** Clang (v10.0.0 or later)
- libfftw
- libsctp
- yaml-cpp
- mbedTLS
- googletest

srsRAN Package:

- srsEPC
- srsENB
- srsUE

UHD Package:

- UHD RF drivers
- GNU Radio

Minimum hardware configuration for 4G/5G NSA network implementation:

MODE	USRPS	CPU CORES	THREADS	RAM	NIC	RF CHANNELS	MTU
4G	B-2xx	2	4	4GB	USB 3.0	1x1	~1500 Bytes
4G	X3xx, N3xx	4	8	8GB	1Gbps	1x1	~1500 Bytes
5G	X3xxs, N3xxs	8	16	16GB	10Gbps	2x2	~9000 Bytes
5G (Simulated)	None	2	4	4GB	None	None	None

Native Installation

Install UHD drivers for the SDR

Step 1: Update and Install Required Packages

This step ensures the system has all necessary libraries and tools for building UHD.

```
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-opengl python3-pyqt5 python3-requests python3-scipy python3-setuptools python3-six
python3-sphinx python3-yaml python3-zmq python3-ruamel.yaml swig wget
```

Step 2: Create a Working Directory

This step creates the necessary working directories and prepares the system for UHD installation:

```
mkdir -p /root/workarea && cd /root/workarea
```

Step 3: Clone the UHD Repository

Download the latest UHD repository from GitHub:

```
git clone https://github.com/EttusResearch/uhd.git && \
cd uhd && git checkout v4.7.0.0
```

Step 4: Build and Install UHD

Now, move into the host directory, build the UHD host libraries, and run the tests:

```
cd host && mkdir build && cd build && \
cmake .. && make && make test
```

Step 5: Install UHD Libraries

Finally, install the compiled UHD libraries and update the system library path:

```
sudo make install && sudo ldconfig
```

Step 6: Configure Environment Variable

Add the library path to the bash configuration file to ensure it's available for future sessions:

```
echo 'export LD_LIBRARY_PATH=/usr/local/lib' >> ~/.bashrc
```

Step 7: Download UHD Firmware Images

This step ensures the necessary UHD firmware is downloaded. Run:

```
sudo uhd_images_downloader
```

Stage 2: Test and Verify the Operation of the USRP

The included USB 3.0 cable provides power and data connectivity for the USRP Bus Series. The host-side of the cable must be plugged into either a USB 2.0 or 3.0 port. Note that a USB 2.0 link provides less bandwidth, and USB 3.0 is recommended.

Once the software tools are installed on the host computer, verify the correct operation of the USRP by running the utility programs on the host computer.

Devices attached to your system can be discovered using the `uhd_find_devices` program. This program scans your system for supported devices and prints out an enumerated list of discovered devices and their addresses. The list of discovered devices can be narrowed down by specifying device address args.

```
uhd_find_devices
```

Device address arguments can be supplied to narrow the scope of the search:

```
uhd_find_devices --args="type=usrp1"

-- OR --

uhd_find_devices --args="serial=12345678"
```

Stage 3: Install srs-RAN software

Step 1: Install Dependencies for srsRAN

Installs essential tools and libraries for compiling and running srsRAN.

```
apt-get update && apt-get install -y
build-essential cmake libfftw3-dev libmbedtls-dev
libboost-program-options-dev libconfig++-dev libsctp-dev
```

Step 2: Add Software Radio Systems Repository

Adds the SRS PPA for accessing the required packages.

```
sudo add-apt-repository ppa:softwareradiosystems/srsran && apt-get update
```

Step 3: Install Additional Libraries for srsGUI

Installs libraries required for the srsGUI interface.

```
apt-get install -y
libboost-system-dev libboost-test-dev libboost-thread-dev
libqwt-qt5-dev qtbase5-dev
```

Step 4: Clone and Build srsGUI

Clones the **srsGUI** repository and builds the graphical user interface.

```
git clone https://github.com/srsLTE/srsGUI.git &&
cd srsGUI && mkdir build && cd build && cmake ../ && make
```

Step 5: Clone and Build srsRAN

Clones the **srsRAN 4G** repository, builds the source, and installs it:

```
cd /root && git clone https://github.com/srsRAN/srsRAN_4G.git &&
cd srsRAN_4G && mkdir build && cd build && cmake ../ && make && sudo make install
```

Step 6: Install srsRAN Configuration Files

Installs configuration files for both user and service mode.

```
sudo ./srsran_install_configs.sh user && sudo ./srsran_install_configs.sh service
```

In the event that the above install command fails, a srsUE example configuration file can be found here

---

Docker Installation

How to install and prepare Docker for srsRAN use

The srsRAN ecosystem relies upon access to a hardware radio unit (for example the B205-mini). Due to USB passthrough complications, we recommend running Docker on a native Linux install (ideally Ubuntu 22.04). If you wish to use a different virtual environment, please refer to the steps below.

Install Docker On Windows

You need to be running Windows Subsystem for Linux (WSL). This enables nested operating systems, in our case Ubuntu processes will run within Windows. This allows us to pass a USB device from Windows to WSL, which is necessary because Docker Desktop (which can run natively in Windows) does not fully support USB passthrough. Essentially we are using WSL to launch the docker container.

Step 1: Download Docker Desktop

- Visit https://docs.docker.com/desktop/install/windows-install/
- Download the latest Docker Desktop Installer executable
- Run the Installer
- Double-click `Docker Desktop Installer.exe`
- Ensure "Use WSL 2" option is selected during configuration
- Follow the installation wizard

Step 2: Post-Installation

Restart your computer after installation completes. After restart, Docker Desktop will start automatically. Accept the Docker Subscription Service Agreement when prompted.

Step 3: Install WSL 2 if required:

Open PowerShell as administrator and run:

```
wsl --install
```

Step 4: Verify Docker installation:

Open PowerShell as administrator and run:

```
docker run hello-world
```

Step 5: Optional Configuration

To configure Docker to start automatically with Windows:

- Open Docker Desktop
- Go to Settings
- Enable "Start Docker Desktop when you log in"

Troubleshooting:

If you encounter any issues with user permissions, run Powershell and add your user account to the docker-users group:

```
net localgroup docker-users <username> /add
```

Configure Docker for WSL USB passthrough

Step 1: Setup

Visit https://github.com/dorssel/usbipd-win/releases and install the `usbipd-win` tool, which allows you to share USB devices with WSL 2.

Step 2: Identify your USB device

Open PowerShell as administrator and run:

```
usbipd list
```

This will show you a list of USB devices with their bus IDs:

```
PS C:\Windows\system32> usbipd list
Connected:
BUSID  VID:PID    DEVICE                                                        STATE
1-2    2708:0009  USB Input Device, Audient iD4                                 Not shared
3-6    1050:0407  USB Input Device, Microsoft Usbccid Smartcard Reader          Not shared
3-11   0bda:0852  Realtek Bluetooth Adapter                                     Not shared
3-12   048d:5702  USB Input Device                                              Not shared
3-14   0c70:f00d  QUADRO, USB Input Device                                       Not shared
4-1    5241:00ac  USB Input Device                                              Not shared
6-4    046d:c547  LIGHTSPEED Receiver, USB Input Device                         Not shared
7-1    00de:0081  USB Input Device                                              Not shared
7-4    2500:0022  WestBridge                                                    Shared
```

Step 3: Attach the USB device to WSL 2

In the same PowerShell window, run:

```
usbipd attach --wsl --auto-attach --busid <busid>
```

Replace `<busid>` with the ID of your USB device from Step 2. Usually you would need to reattach the USB device each time you restart your computer or disconnect/reconnect the device. Due to the way the UHD drivers handle the USB interface, we found this created issues. Therefore we amend the command to include the `--auto-attach` option.

In your WSL 2 Linux distribution, the device should now be visible. You can check with:

```
ls /dev/bus/usb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 002: ID 2500:0022 Ettus Research LLC USRP B205-mini
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

You can see here that the Ettus B205-mini has been successfully passed from Windows to Ubuntu (WSL). It is on `Bus 001`, and is `Device 002`. You may have to specify the distribution in order for `usbipd` to work. In our case it looked like this:

```
PS C:\Windows\system32> wsl --list
Windows Subsystem for Linux Distributions:
docker-desktop-data (Default)
docker-desktop
Ubuntu-22.04
```

```
PS C:\Windows\system32> usbipd attach --wsl Ubuntu-22.04 --auto-attach --busid 7-4
usbipd: info: Selecting a specific distribution is no longer required. Please file an issue if you believe
that the default selection mechanism is not working for you.
usbipd: info: Using WSL distribution 'Ubuntu-22.04' to attach; the device will be available in all WSL 2
distributions.
usbipd: info: Using IP address 172.17.160.1 to reach the host.
usbipd: info: Starting endless attach loop; press Ctrl+C to quit.
WSL Attached
```

Step 4: Pass the device from Ubuntu to Docker

Now, when running your Docker container, you can use the `--device` flag:

```
sudo docker run -it --privileged --device=/dev/bus/usb/001/002 srsdocker
```

Replace `<bus>` and `<device>` with your appropriate numbers from Step 2. Also note the added `--privileged` modifier in order to avoid permissions errors.

Install and Configure Docker On Mac

Visit https://docs.docker.com/desktop/install/mac-install/ and download the latest Docker Desktop Installer `.dmg` file.

Option 1: Interactive Install Method

- Double-click the Docker.dmg file
- Drag the Docker icon to your Applications folder
- Double-click Docker.app in the Applications folder to launch
- Accept the Docker Subscription Service Agreement

Option 2: Install from command line

Open a terminal and run the following:

```
sudo hdiutil attach Docker.dmg
sudo /Volumes/Docker/Docker.app/Contents/MacOS/install
sudo hdiutil detach /Volumes/Docker
```

Install and Configure Docker On Linux

Visit https://docs.docker.com/desktop/install/linux/ubuntu/ for information on how Linux handles Docker containers. Before you install Docker Engine for the first time on a new host machine, you need to set up the Docker repository. Afterward, you can install and update Docker from the repository.

Step 1: Set up the Docker repository

Open a terminal and run the following:

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

Step 2: Install the Docker packages

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

Step 3: Verify install

```
sudo docker run hello-world
```

Optional Step: Install Docker Desktop

Once you have the `apt` repository set up, you can install Docker Desktop if you wish. Download the latest DEB package and then run the following in terminal:

```
sudo apt-get update
sudo apt-get install ./docker-desktop-<ubuntu>.deb
```

At the end of the installation process, `apt` displays an error due to installing a downloaded package. You can ignore this error message. By default, Docker Desktop is installed at `/opt/docker-desktop`.

Step 4: Plug in the USRP

Once connected, navigate to your Linux terminal and the device should now be visible. You can check with:

```
ls /dev/bus/usb
```

Step 5: Pass the device from Ubuntu to Docker

Now, you can run the Docker container, and you use the `--device` flag:

```
docker run -it --rm --device=/dev/bus/usb/<bus>/<device> ubuntu ls -l /dev/bus/usb/<bus>/<device>
```

Replace `<bus>` and `<device>` with the appropriate numbers from Step 1.

Docker Operation

Please note that when issuing commands *within* the Docker container (as opposed to natively on a Linux test system), you can omit the `sudo` portion of the command. In order to correctly operate the USRP device, the Docker container must have admin privileges. Make sure to check the commands are applicable to your launch environment, and amend the `sudo` inclusion if not.

Stage 1: Run srsRAN software and verify network

Step 1: Start the EPC (Evolved Packet Core)

Within the running Docker container, open a terminal and run:

```
srsepc
```

Step 2: In a new terminal window, start the eNodeB

```
srsenb
```

Step 3: On the other test device, or in a third terminal, start the UE (User Equipment)

```
srsue
```

Step 4: Verify connectivity by checking the network interfaces

```
ifconfig
```

Look for interfaces named `srs_spgw_sgi` (on EPC) and `tun_srsue` (on UE).

Step 5: Test the connection:

```
ping 172.16.0.1
```

Stage 2: Configure srs-RAN settings to desired parameters

Step 1: Configure eNodeB settings in `/etc/srsran/enb.conf`

- Set the correct EARFCN for your frequency band
- Adjust TX gain based on your environment (move in 5dB increments for safety)
- Configure cell ID and other network parameters

Step 2: Configure UE settings in `/etc/srsran/ue.conf`:

- Set the correct EARFCN to match eNodeB
- Configure USIM parameters
- Adjust RF parameters for your SDR

Step 3: Configure EPC settings in `/etc/srsran/epc.conf`:

- Set up the MME parameters
- Configure the HSS database
- Set appropriate IP addressing for the network

Testing

Network Connectivity

Step 1: Start srsEPC

On the machine designated for the core network (Machine 1), start srsEPC by running in terminal:

```
srsepc
```

This will create a virtual network interface named srs_spgw_sgi with an IP address of 172.16.0.1.

Step 2: Start srsENB

On the same machine (Machine 1), in a separate terminal, start srsENB by running:

```
srsenb
```

Step 3: Start srsUE

On the second machine (Machine 2), run srsUE by executing:

```
srsue
```

This will create a virtual network interface named tun_srsue with an IP address in the 172.16.0.x range, such as 172.16.0.2. That IP address will be assigned to the UE device.

To test connectivity, run a ping command from the UE to the EPC's IP address:

```
ping 172.16.0.1
```

If you want to test from the EPC to the UE, first find out the UE's assigned IP address (e.g., 172.16.0.2) and then run:

```
ping <ue_ip_address>
```

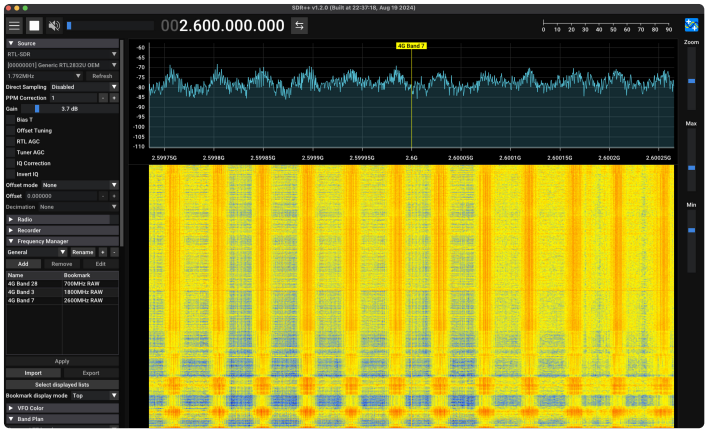
If you want to test constant connectivity, for example when monitoring range or line of sight, then you can run an endless ping from the UE device:

```
ping -t 172.16.0.1
```

Using RTL-SDR with SDR++ and Wireshark to test the connection:

You can test to see if there is a signal being output by the USRP with Wireshark and an appropriate radio receiver, such as a RTL-SDR. This will not enable actually decrypting the traffic, just observing that a signal exists. It is recommended only as an ad-hoc check.

Install appropriate software for interfacing with the receiver such as SDR++. Input the 4G LTE band 7 frequency parameters and check the waterfall graph for a signal.



File Transfer over srsRAN

Step 1: Install OpenSSH Server

Install an OpenSSH server to accept incoming SSH and `scp` connections:

```
sudo apt update
sudo apt install openssh-server
```

Step 2: Start the SSH Service

After installation, start and enable the SSH service to ensure it's running and set it to start on boot:

```
sudo systemctl start ssh
sudo systemctl enable ssh
```

Step 3: Check SSH Status

Verify that the SSH server is running:

```
sudo systemctl status ssh
```

You should see `active (running)` if the SSH service is working properly.

Step 4: Configure the Firewall

If you have a firewall enabled, allow SSH traffic through the firewall:

```
sudo ufw allow ssh
sudo ufw reload
```

Step 5: Determine the EPC/ENB Machine's IP Address

Check the IP address of the machine running `srsEPC` and `srsENB`:

```
ifconfig
```

Ensure this IP is accessible from the UE machine (e.g. 172.16.0.1).

Step 6: Create a User Account (Optional)

If you prefer not to use the root account, create a new user for file transfers from srsepc device:

```
sudo adduser filetransferuser
sudo passwd filetransferuser
```

Step 7: Perform the File Transfer via SCP

On the UE machine, use `scp` to transfer the file to the EPC/ENB machine:

```
scp /path/to/local/file transferuser@<EPC_IP>:/path/to/destination/
```

- Replace `<EPC_IP>` with the IP address of the EPC/ENB machine.
- Replace `/path/to/local/file` with the file you want to transfer.
- Replace `/path/to/destination/` with the directory on the EPC/ENB machine where you want the file to go.

```
scp testfile.txt 172.16.0.1:/home/katnap/transfer/received

-- or --

scp /home/katnap/transfer/sent/testfile.txt 172.16.0.1:/home/katnap/transfer/received
```

Step 8: Verify the File Transfer

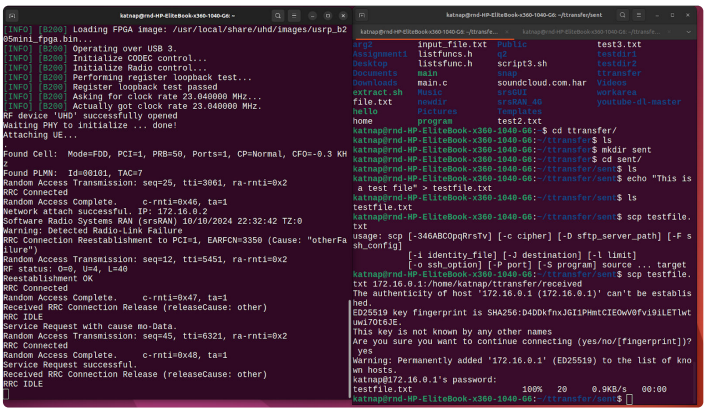
After the transfer is complete, log into the EPC/ENB machine and check the destination directory to ensure the file was received:

```
ls /path/to/destination/
```

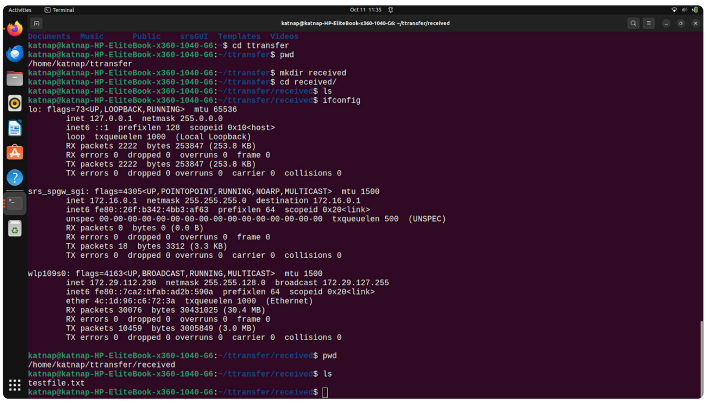
```
ls /home/katnap/transfer/received
```

Examples:

Client end should look like this:



Host end srsepc/srsenb should look like this:



Troubleshooting

Common issues and solutions

UHD Driver Installation Failures

```
sudo apt-get clean
sudo apt-get update
sudo apt-get install -f
```

Library Path Issues

```
sudo ldconfig
source ~/.bashrc
```

Permission Problems

```
sudo usermod -a -G usrp $USER
sudo chown $USER:$USER -R /usr/local/share/uhd
```

Missing Dependencies

```
sudo apt-get install -y libboost-all-dev
sudo apt-get install -y python3-mako
```

Missing UHD device

If the USRP device does not show when running the `uhd_find_devices` command, try these additional troubleshooting steps.

You can see here that the B205-mini was visible in WSL, and then running the command `uhd_find_devices` kicked it from the environment.

```
edward@EPC:/home$ sudo docker run -it --privileged --device=/dev/bus/usb/001/002 srsdocker
root@4413dbe0ad00:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srsGUI  srv
sys  tmp  usr  var
root@4413dbe0ad00:/# lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 002: ID 2500:0022 Ettus Research LLC USRP B205-mini
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
root@4413dbe0ad00:/# uhd_find_devices
[INFO] [UHD] linux; GNU C++ version 11.4.0; Boost_107400; UHD_4.7.0.HEAD-0-ga5ed1872
[INFO] [B200] Loading firmware image: /usr/local/share/uhd/images/usrp_b200_fw.hex...
No UHD Devices Found
root@4413dbe0ad00:/# lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
root@4413dbe0ad00:/# exit
```

When `uhd_find_devices` was executed, Powershell showed a disconnection:

```
WSL Attached
WSL Detached
WSL usbip: error: Attach Request for 7-4 failed - Device not found
```

Properties of devices attached to your system can be probed with the `uhd_usrp_probe` program. This program constructs an instance of the device and prints out its properties, such as detected daughterboards, frequency range, gain ranges, etc.

```
sudo uhd_usrp_probe --args=<device-specific-address-args>
```

Naming a USRP Device

For convenience purposes, you may assign a custom name to their USRP device once it is found. The USRP device can then be identified via name, rather than a difficult to remember serial or address. A name has the following properties:

- is composed of ASCII characters
- is 0-20 characters
- is not required to be unique

Run the following commands:

```
cd <install-path>/lib/uhd/utils
./usrp_burn_mb_eeprom --args=<optional device args> --values="name=lab1_xcvr"
```

Discovery via name

The keyword `name` can be used to narrow the scope of the search. Example with the find devices utility:

```
uhd_find_devices --args="name=lab1_xcvr"

-- OR --

uhd_find_devices --args="type=usrp1, name=lab1_xcvr"
```

Below is a table of the B205mini LED indicators and their meanings:

Component ID	Description	Details
PWR LED	Power Indicator	off = no power applied on = power applied (external or USB)
TRX LED	TX/RX Activity	off = no activity green = receiving red = transmitting orange = switching between transmitting and receiving
RX2 LED	RX2 Activity	off = no activity green = receiving
S0 LED	Reference Lock	off = no activity green = locked
S1 LED	Reference Present	off = reference level low or not present green = reference level high

---

References:

https://files.ettus.com/manual/page_install.html
https://files.ettus.com/manual/page_identification.html
https://files.ettus.com/manual/page_usrp_b200.html
https://kb.ettus.com/B200/B210/B200mini/B205mini_Getting_Started_Guides
https://docs.srsran.com/projects/4g/en/latest/usermanuals/source/1_setup.html
https://docs.srsran.com/projects/project/en/latest/user_manuals/source/installation.html
https://github.com/srsRAN/srsRAN_4G/blob/master/srsue/ue.conf.example