PackRF Deployed Design

We will now be moving onto the PackRF hardware and the modem design which has been deployed to it. The SD cards have been prepared and configured with the Modem design built from MATLAB and Simulink***. To prepare to operate these devices make sure that an Ethernet cable is plugged into one device, the JTAG connected is connected to the second PackRF, and both have power***. The JTAG connector for reference is shown in Figure 1, and connects to the ribbon cable from the PackRF box.

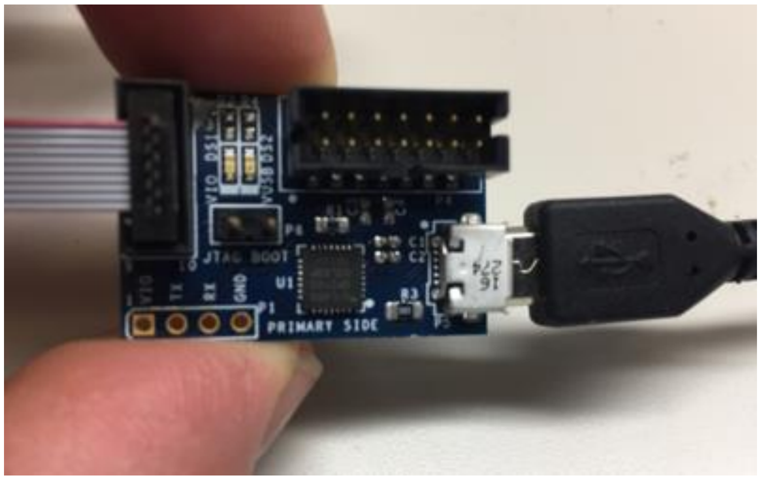
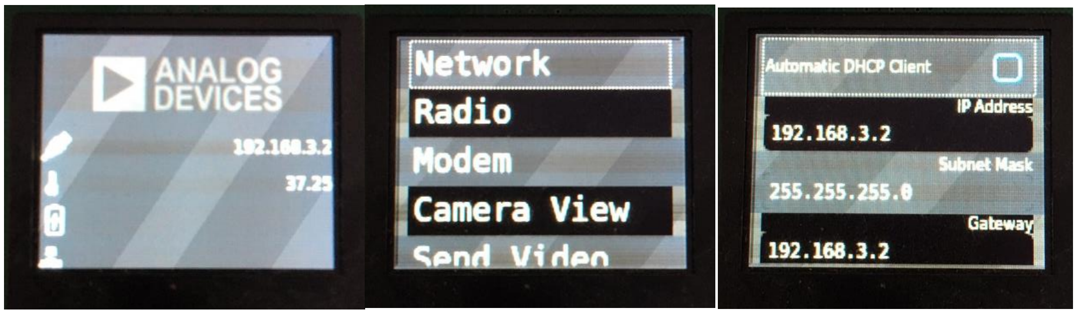


Figure 1

Once both devices are connected and have power, turn them on using their silver power button. It will take ~10 seconds for the RFSOMs to boot and a menu will appear on the small LCD screens. By default, the modem in each PackRF is not started or configured. Since each modem is half of a link they must be configured in a pair. We will go through this process of manually accessing the radios and enabling the modems. Then we will perform this configuration through the GUI on the PackRF itself.

Setting Static IP Before Connecting

Before we cannot to one of the PackRF devices over Ethernet, we need to configure the device to use a static IP address. ***On the device itself, by using the scroll wheel, navigate to Network, and deselect "Automatic DHCP Client". Then choose an IP address of 192.168.3.2.***



Automatic DHCP Client ☐

IP Address
192.168.3.2

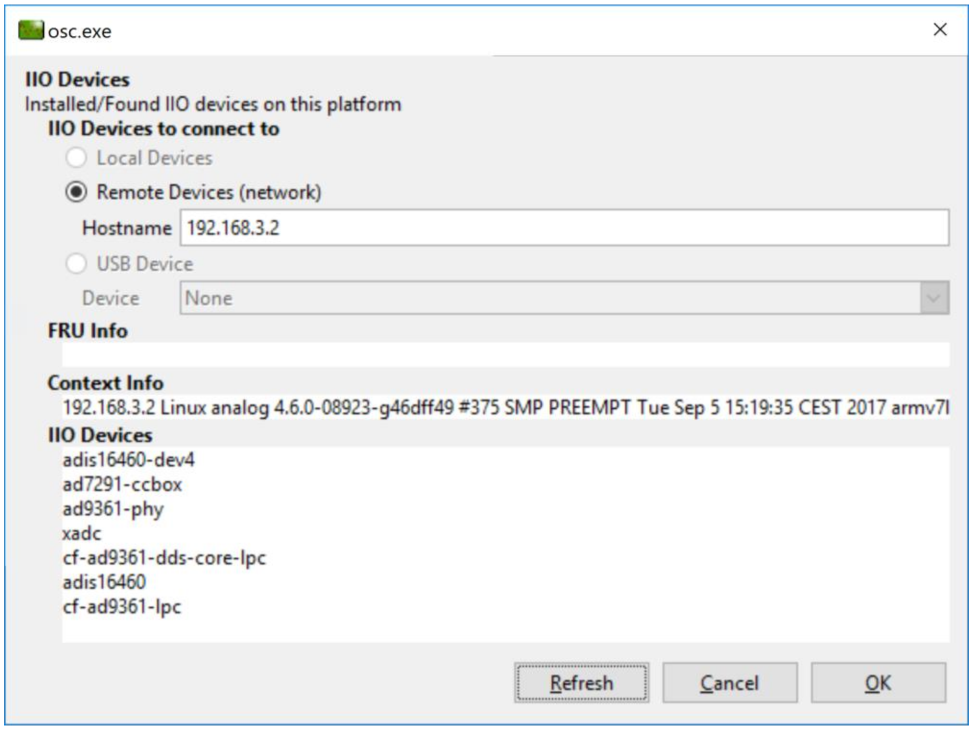
Subnet Mask
255.255.255.0

Gateway
192.168.3.2

Now we can talk to the first PackRF directly over Ethernet.

Just another IIO Device

Since the PackRF and software infrastructure was built using the ADI software/hardware infrastructure, we are still able to access the device as we did Pluto with IIO-Scope. ***Launch the IIO-Scope application from Lab 1 and connect to the PackRF attached through Ethernet.*** You must use the "Remote Devices (network)" context now since the device is connected over Ethernet, as shown in Figure 2.

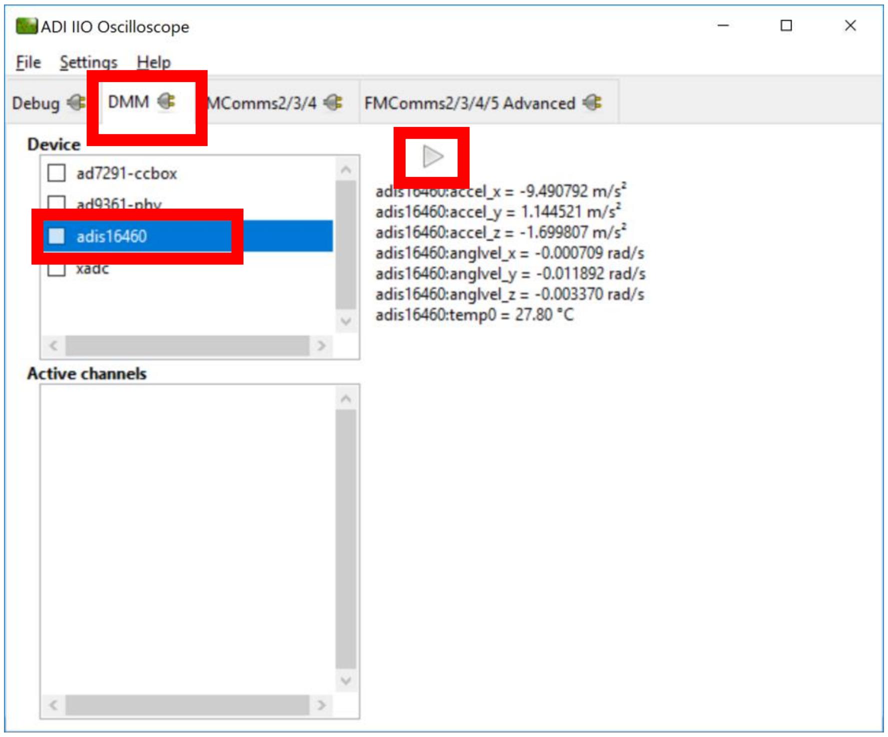

osc.exe

IIO Devices
Installed/Found IIO devices on this platform
IIO Devices to connect to
Local Devices
[✓ Remote Devices (network)]
Hostname 192.168.3.2
USB Device
Device None
FRU Info

Context Info
192.168.3.2 Linux analog 4.6.0-08923-g46dff49 #375 SMP PREEMPT Tue Sep 5 15:19:35 CEST 2017 armv7l
IIO Devices
adis16460-dev4
ad7291-ccbox
ad9361-phy
xadc
cf-ad9361-dds-core-lpc
adis16460
cf-ad9361-lpc

Figure 2

After connecting, click on the IIO-Scope main panel (not the plot) and select the DMM tab as shown in Figure 3. You will observe four devices, most of which are related to the AD936X chip as in Pluto. However, there is an additional entry ADIS16460 which is for the gyroscope inside the PackRF. Since this is an ADI gyroscope and we maintain IIO drivers for the device it also appears in IIO-Scope automatically. ***Select the "adis16460" from the Device menu and click on the Play/Pause button ▷ to start collecting data from the gyroscope.***



```
adis16460:accel_x = -9.490792 m/s²
adis16460:accel_y = 1.144521 m/s²
adis16460:accel_z = -1.699807 m/s²
adis16460:anglvel_x = -0.000709 rad/s
adis16460:anglvel_y = -0.011892 rad/s
adis16460:anglvel_z = -0.003370 rad/s
adis16460:temp0 = 27.80 °C
```

Figure 3

Once running pick up the PackRF and rotate the box. You should observe out measurement change with your movement. The gyro inside the PackRF is military grade and very robust to shock. You can also observe this data in the Capture window as previously in Lab 1. ***Select the channels of interest as shown in Figure 4 and press the play button to view live data from the sensor.***

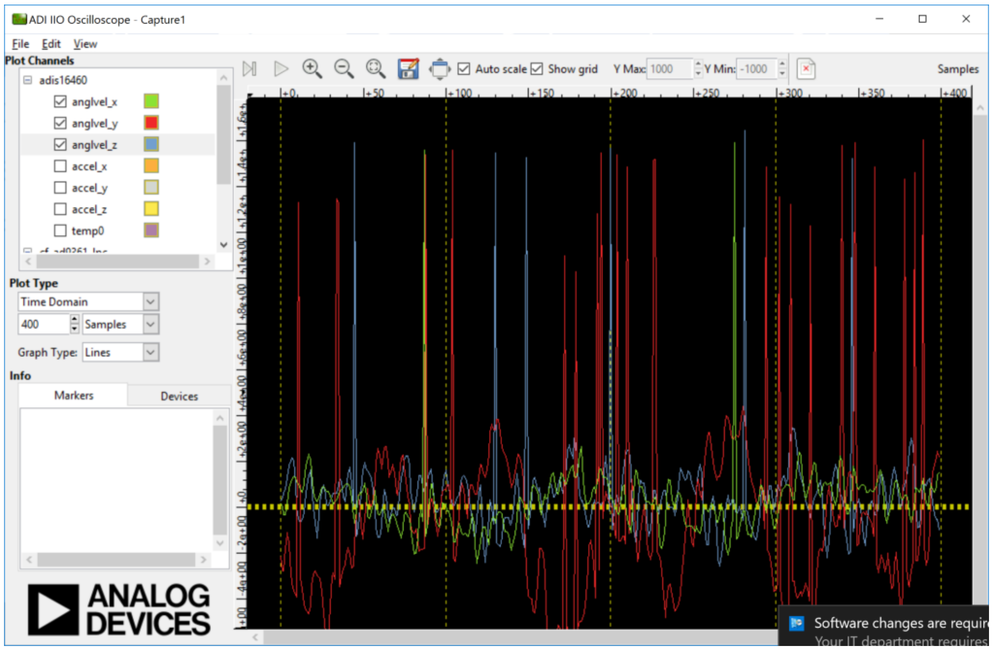


Figure 4

This demonstrates the usefulness and wide spread usages of IIO software for all different devices.

Next, we will look at streaming data from the AD9361 as previously with Pluto. For the FPGA design for this implementation a second set of DMAs were added so IIO-Scope could be used to monitor input data to the transceiver while the Modem design passed data back through the first DMA. Also, using an added mux IQ data can be passed directly to the AD9361 IP Core. The data paths are outlined in Figure 5. This allows a simple debugging option where signal injection can easily be performed through the libiio interface.

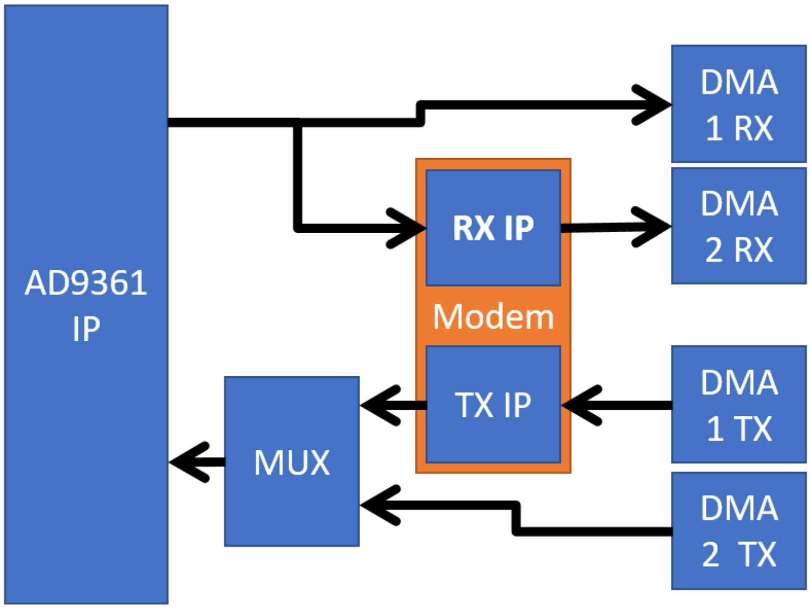
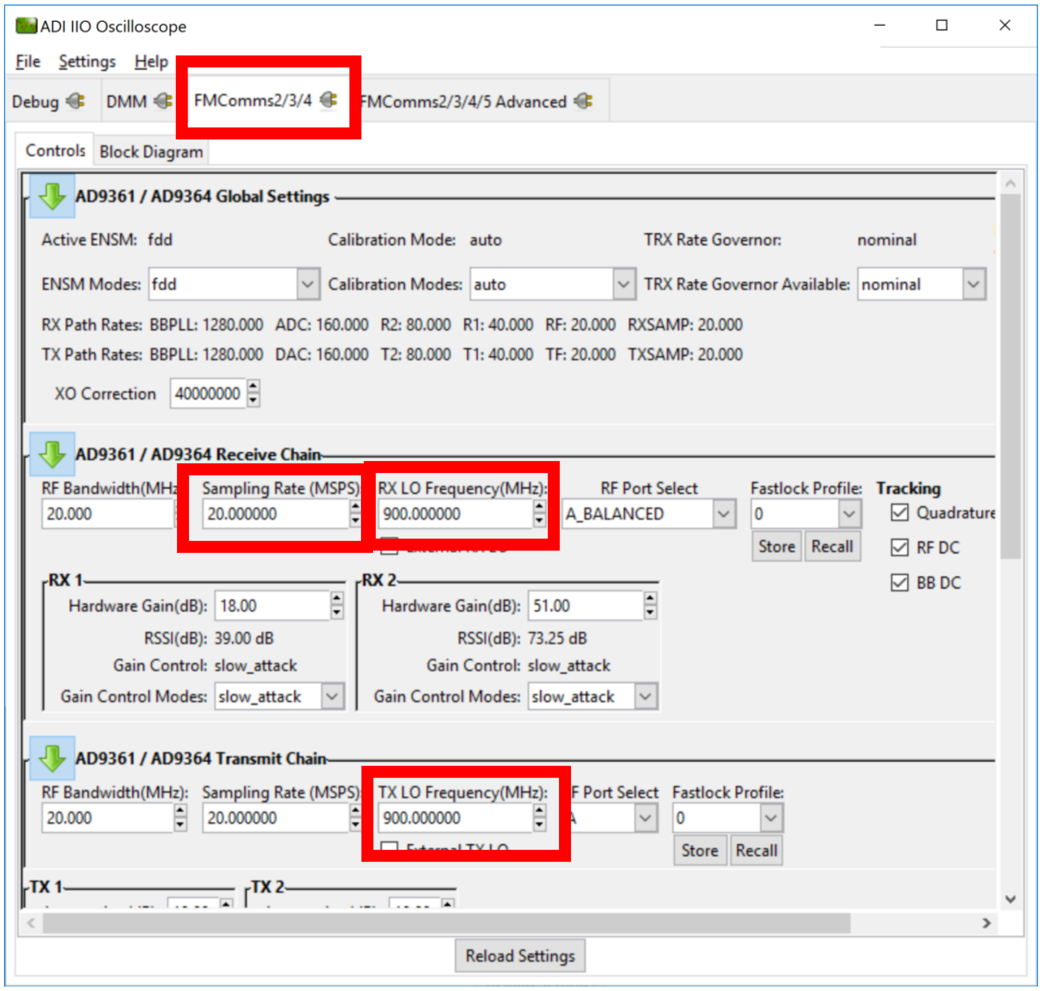


Figure 5

To add these additional DMAs required some work in Vivado, but this provided an easy mechanism for debug outside of Vivado eventually.

We will first test this by moving to the FMComms2/3/4 tab. ***Move to the FMComms2/3/4 tab on the top bar, as marked by Figure 6. Also, make the transmit and receive LO identical at 900 MHz so we can see what we are transmitting. Finally, make sure the sample rates are set to 20MHz.*** By default, the mux in the FPGA should be configured to use the directly connected DMA interface.



AD9361 / AD9364 Global Settings

Active ENSM: fdd   Calibration Mode: auto   TRX Rate Governor: nominal

ENSM Modes: fdd   Calibration Modes: auto   TRX Rate Governor Available: nominal

RX Path Rates: BBPLL: 1280.000 ADC: 160.000 R2: 80.000 R1: 40.000 RF: 20.000 RXSAMP: 20.000

TX Path Rates: BBPLL: 1280.000 DAC: 160.000 T2: 80.000 T1: 40.000 TF: 20.000 TXSAMP: 20.000

XO Correction 40000000

AD9361 / AD9364 Receive Chain

RF Bandwidth(MHz): 20.000   Sampling Rate (MSPS): 20.000000   RX LO Frequency(MHz): 900.000000   RF Port Select: A_BALANCED   Fastlock Profile: 0   Tracking: ☑ Quadrature ☑ RF DC ☑ BB DC

RX 1: Hardware Gain(dB): 18.00   RSSI(dB): 39.00 dB   Gain Control: slow_attack   Gain Control Modes: slow_attack

RX 2: Hardware Gain(dB): 51.00   RSSI(dB): 73.25 dB   Gain Control: slow_attack   Gain Control Modes: slow_attack

AD9361 / AD9364 Transmit Chain

RF Bandwidth(MHz): 20.000   Sampling Rate (MSPS): 20.000000   TX LO Frequency(MHz): 900.000000   ☐ External TX LO   RF Port Select: A   Fastlock Profile: 0

TX 1   TX 2

Figure 6

Now, scroll down to the "FPGA Settings" and enabled the "One CW Tone" setting from the "DDS Mode". Next, as outlined in Figure 7, in the "Single Tone" settings update the frequency and scale. This will start transmitting a waveform in a loopback like configuration.

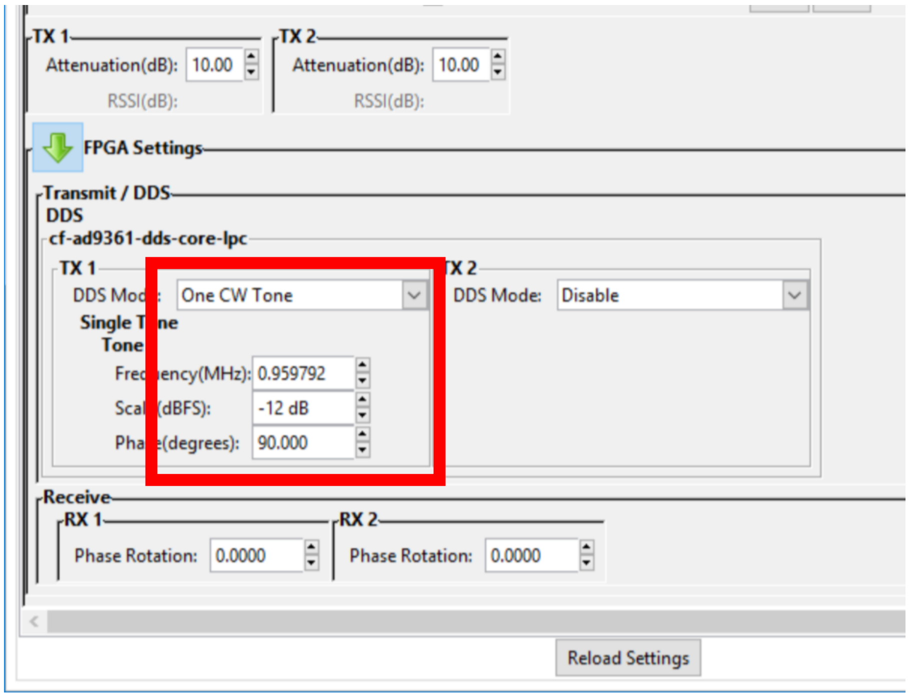
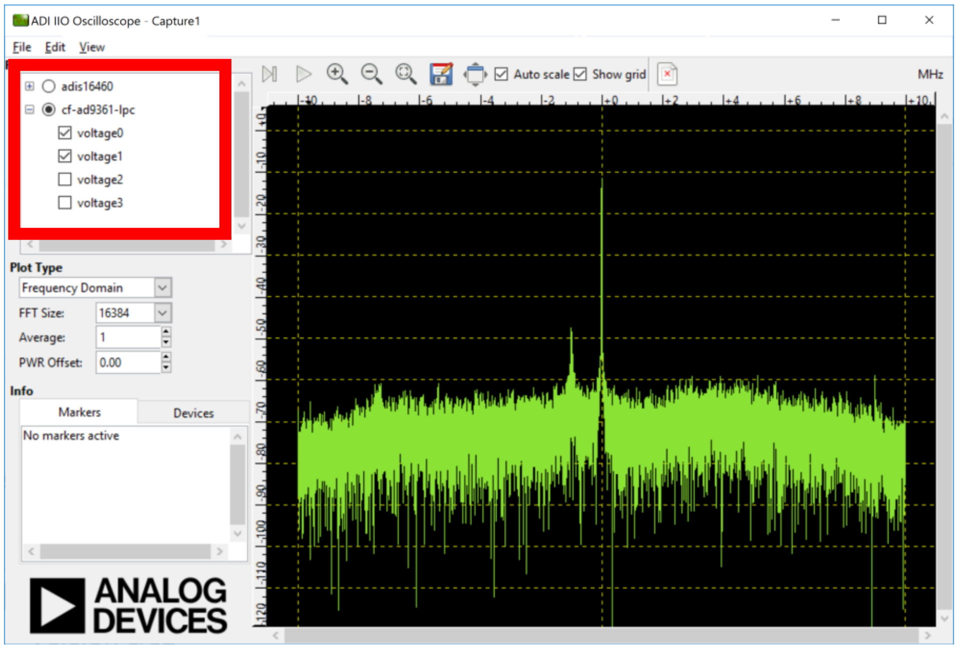

TX 1
Attenuation(dB): 10.00
RSSI(dB):
TX 2
Attenuation(dB): 10.00
RSSI(dB):
FPGA Settings
Transmit / DDS
DDS
cf-ad9361-dds-core-lpc
TX 1
DDS Mode: One CW Tone
Single Tone
Tone
Frequency(MHz): 0.959792
Scale(dBFS): -12 dB
Phase(degrees): 90.000
TX 2
DDS Mode: Disable
Receive
RX 1
Phase Rotation: 0.0000
RX 2
Phase Rotation: 0.0000
Reload Settings

Figure 7

Next, we can change our attention back to the "Capture" window of IIO-Scope. Like Pluto we can use IIO-Scope to debug our design visually. ***In the Capture window, make sure "cf-ad9361-lpc" is selected in the "Plot Channels" menu and voltage0 and voltage1 are checked as shown in Figure 8.*** For reference, since the adis16460 is an IIO device with streaming interfaces it also automatically appears in this list and we can visualize the available outputs.



Plot Type: Frequency Domain
FFT Size: 16384
Average: 1
PWR Offset: 0.00
No markers active

Figure 8

In the "Plot Type" select "Frequency Domain" from the dropdown menu and then click the play button at the top of the plot. With "One CW Tone" selected your spectrum should appear as in Figure 9, with a boxlike waveform.

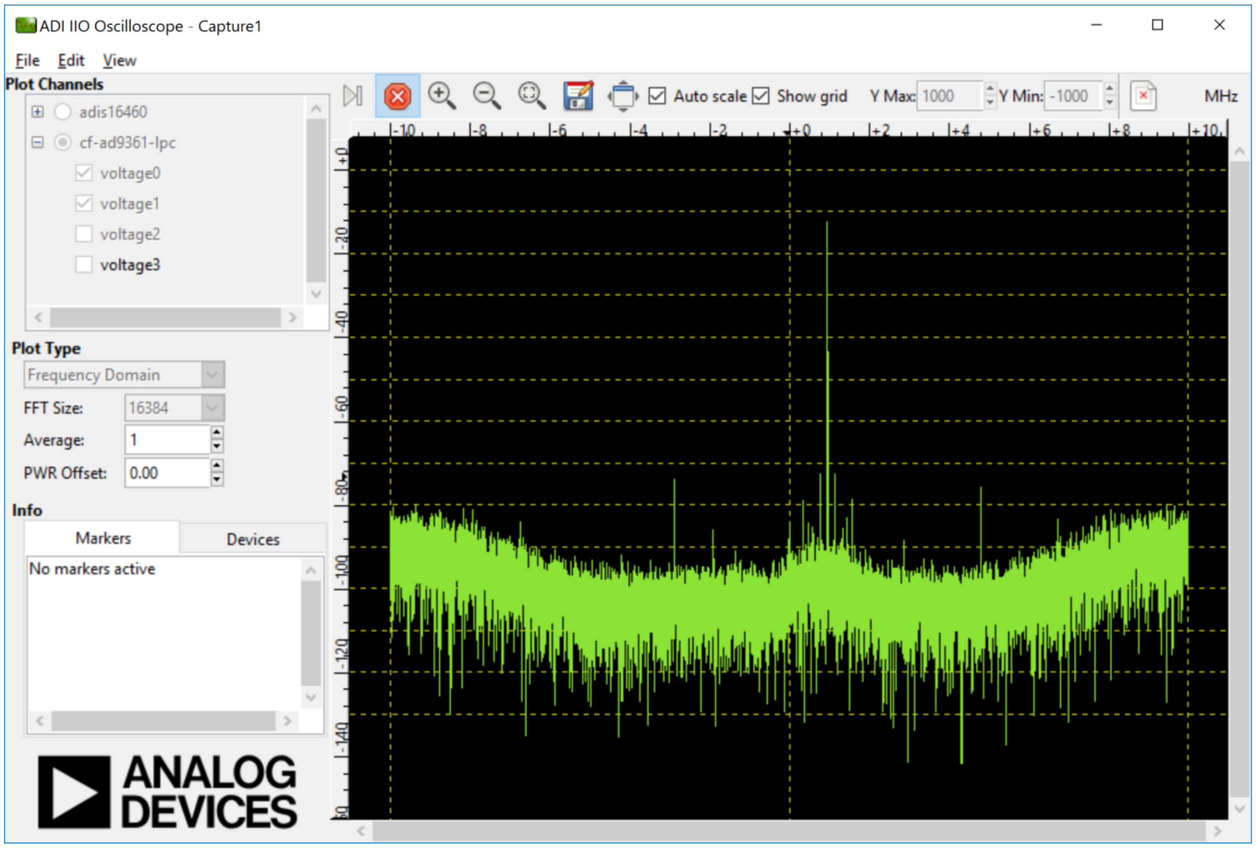


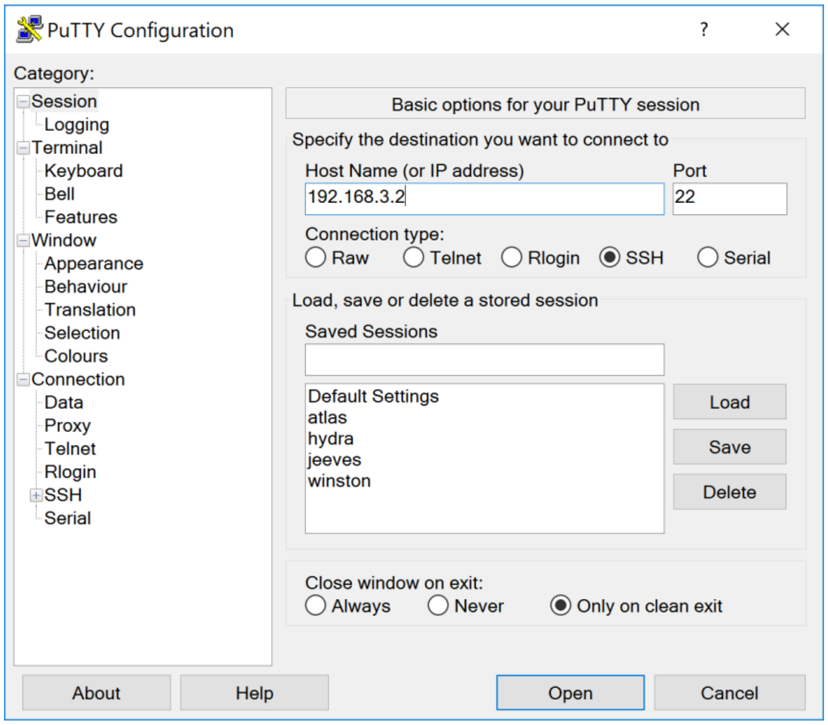
Figure 9

In the default configuration the PackRF with the deployed modem design here works as a default reference design with IQ streaming. This will change in the next section when we start configuring the Modem IP. ***Keep this Capture window open and running for now.***

Accessing the PackRF Remotely and Configuring the Modem

At this point in the design we are completely untethered from MATLAB and Simulink. Therefore, we must utilize a console to interact with the modem and transceiver. Unlike the MATLAB stripped down SD card image, the one used in this design uses a much more standardize Linux image which contains many common tools found on desktop Linux OSs. Even a GUI is run and displayed on the screen by the RF-SOM in this configuration.

At this point the RF-SOM has been configurated to utilize a static IP address, which is either 192.168.3.2 or another user defined value. ***Open PuTTY and create a session as in Figure 10, to connect to the device connected through Ethernet.*** For convenience we will refer to PackRF with address 192.168.3.2 as Radio 1, and PackRF connected with the JTAG cable as Radio 2.



Specify the destination you want to connect to

Host Name (or IP address): 192.168.3.2

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

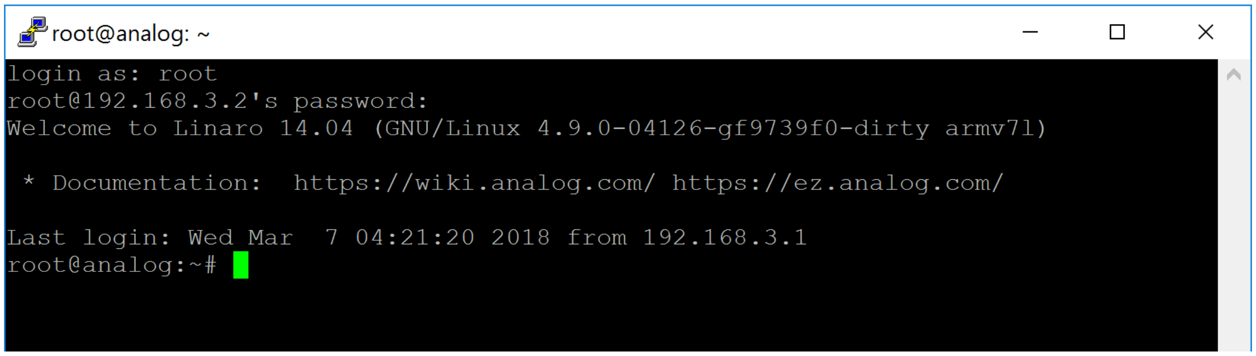
Saved Sessions

Default Settings
atlas
hydra
jeeves
winston

Close window on exit: Always Never [✓ Only on clean exit]

Figure 10

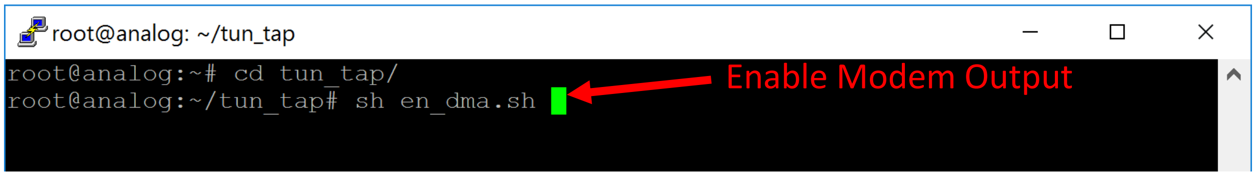
A console should appear with a login prompt. ***The default username and password for the RF-SOM is root : analog, login with as this user.*** Once logged into the RF-SOM you should observe outputs similar to Figure 11.



```
login as: root
root@192.168.3.2's password:
Welcome to Linaro 14.04 (GNU/Linux 4.9.0-04126-gf9739f0-dirty armv7l)

 * Documentation:  https://wiki.analog.com/ https://ez.analog.com/

Last login: Wed Mar  7 04:21:20 2018 from 192.168.3.1
root@analog:~#
```

Figure 11

Since we are logged into the PackRF now we can examine the transmitted waveform of the Modem itself. We will do so by toggling the transmitter mux as outlined in Figure 5. Toggle the mux by moving into the “tun_tap” folder and run the script “sh en_dma.sh” to select the Modem output to the AD9361 IP core. This command is also provided in Figure 12.



```
root@analog:~# cd tun_tap/
root@analog:~/tun_tap# sh en_dma.sh
```

Figure 12

After "sh en_dma.sh" is run, look back at the "Capture" window from IIO-Scope. You will notice it will change to something like Figure 13. ***Run the command "en_dds.sh" and you will observe the waveform move back to the CW tone as before.***

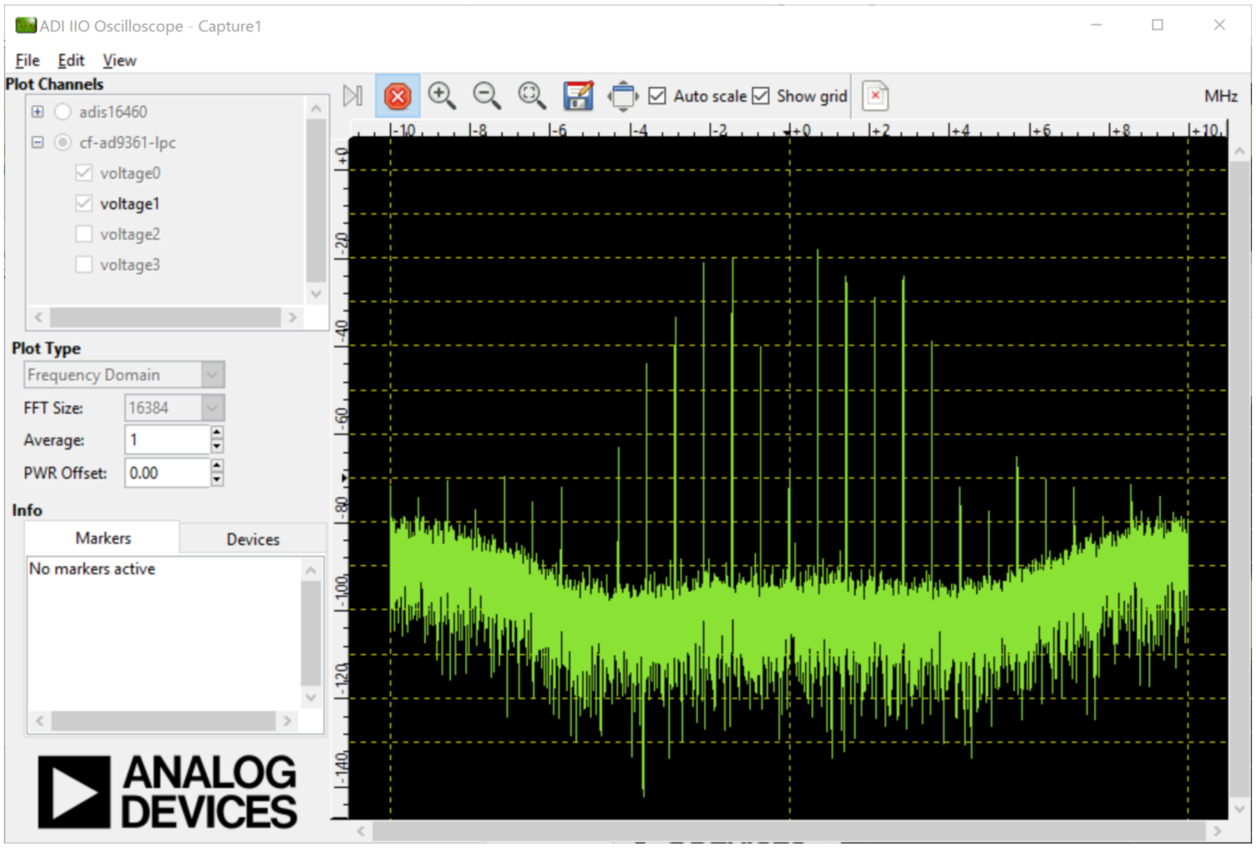
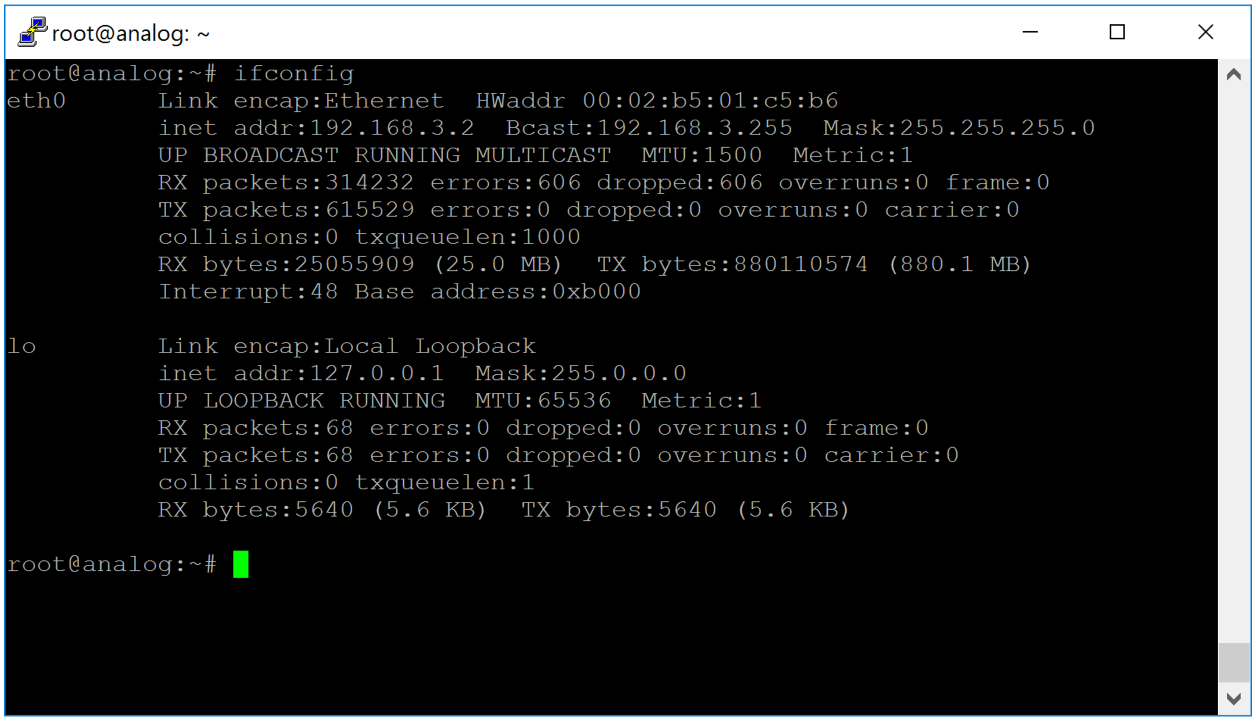


Figure 13

Next, we can begin to start the TUN/TAP daemon which manages data passed between the kernel and modem design running on the FPGA. However, before we do this we will examine the available networking interfaces. ***Run the command "ifconfig" in the console, which should print out something similar to Figure 14, with the IP associated with your given board assigned to eth0.***

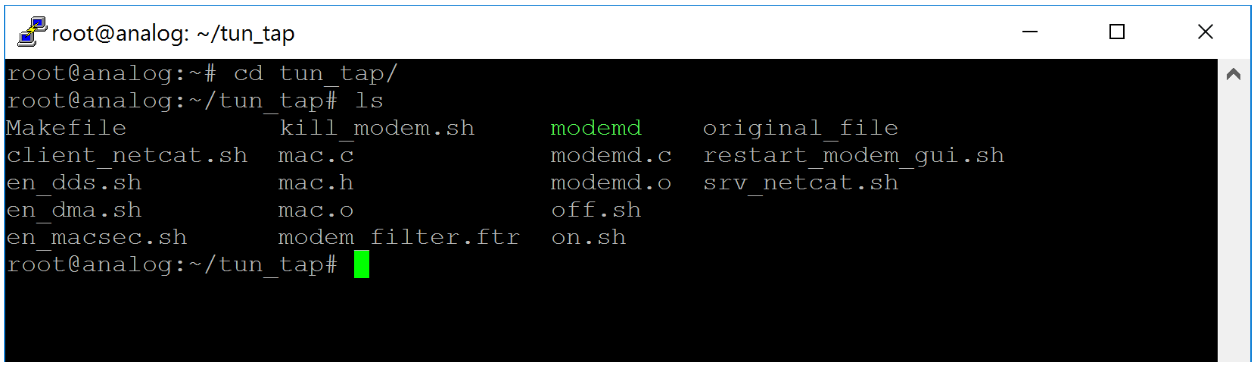


```
root@analog:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:02:b5:01:c5:b6
          inet addr:192.168.3.2  Bcast:192.168.3.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:314232 errors:606 dropped:606 overruns:0 frame:0
          TX packets:615529 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:25055909 (25.0 MB)  TX bytes:880110574 (880.1 MB)
          Interrupt:48 Base address:0xb000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:68 errors:0 dropped:0 overruns:0 frame:0
          TX packets:68 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:5640 (5.6 KB)  TX bytes:5640 (5.6 KB)

root@analog:~#
```

Figure 14

You will notice by this output that there are only two available interfaces: lo (loopback), and eth0. Therefore, our TUN/TAP interface has not been initialized yet.

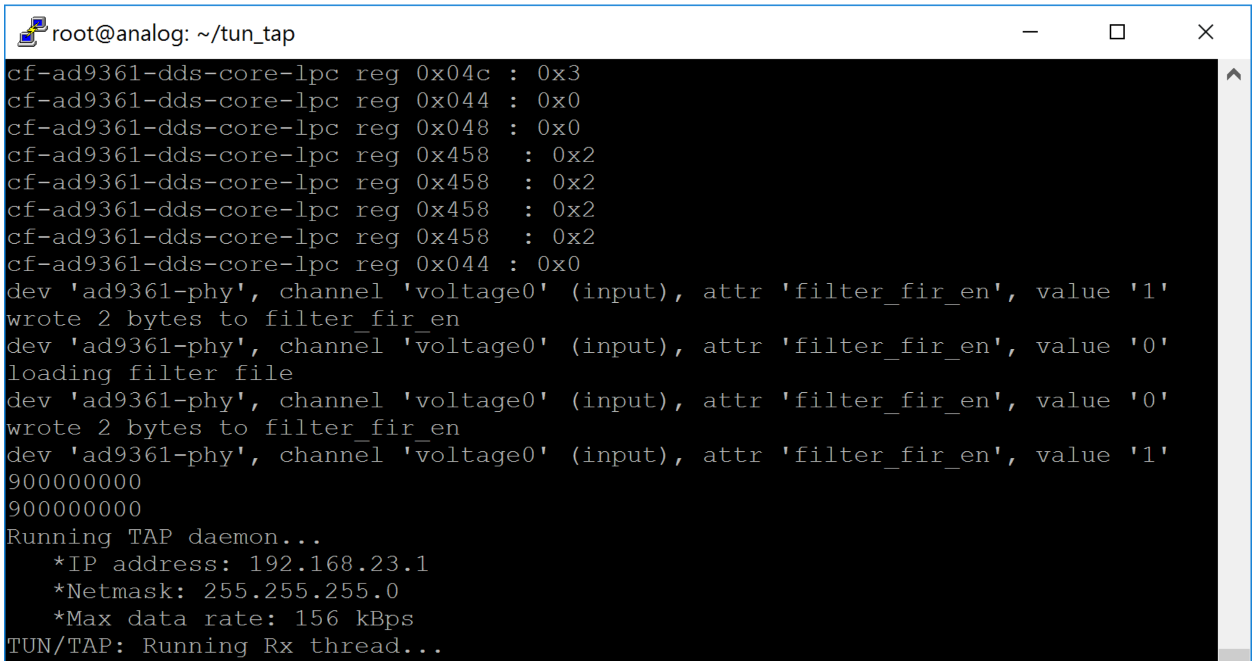
To initialize the TUN/TAP interface enter the "tun_tap" folder, which contains all the source used to implement our TUN/TAP design, and run the script "restart_modem.sh" without any arguments. The commands are provided by example in Figure 15.



```
root@analog:~# cd tun_tap/
root@analog:~/tun_tap# ls
Makefile          kill_modem.sh     modemd    original_file
client_netcat.sh  mac.c             modemd.c  restart_modem_gui.sh
en_dds.sh         mac.h             modemd.o  srv_netcat.sh
en_dma.sh         mac.o             off.sh
en_macsec.sh      modem_filter.ftr  on.sh
root@analog:~/tun_tap#
```

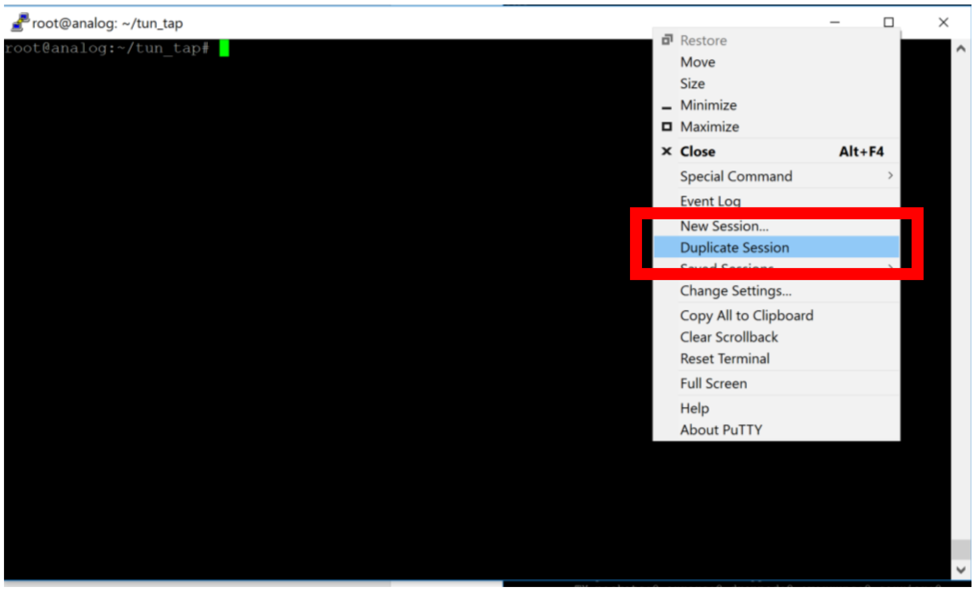
Figure 15

By default, this will configure the modem, provide some status information, then bring up the TUN/TAP interface. The values displayed include some debug register values, and we also print out some transceiver configuration information such as: center frequency, sample rate, and RF bandwidth. If you wish to investigate you can look at the bash script "restart_modem_gui.sh", which outlines our usage of the IIO devices through file descriptors to talk with the modem and transceiver.



```
root@analog: ~/tun_tap
cf-ad9361-dds-core-lpc reg 0x04c : 0x3
cf-ad9361-dds-core-lpc reg 0x044 : 0x0
cf-ad9361-dds-core-lpc reg 0x048 : 0x0
cf-ad9361-dds-core-lpc reg 0x458  : 0x2
cf-ad9361-dds-core-lpc reg 0x458  : 0x2
cf-ad9361-dds-core-lpc reg 0x458  : 0x2
cf-ad9361-dds-core-lpc reg 0x458  : 0x2
cf-ad9361-dds-core-lpc reg 0x044 : 0x0
dev 'ad9361-phy', channel 'voltage0' (input), attr 'filter_fir_en', value '1'
wrote 2 bytes to filter_fir_en
dev 'ad9361-phy', channel 'voltage0' (input), attr 'filter_fir_en', value '0'
loading filter file
dev 'ad9361-phy', channel 'voltage0' (input), attr 'filter_fir_en', value '0'
wrote 2 bytes to filter_fir_en
dev 'ad9361-phy', channel 'voltage0' (input), attr 'filter_fir_en', value '1'
900000000
900000000
Running TAP daemon...
   *IP address: 192.168.23.1
   *Netmask: 255.255.255.0
   *Max data rate: 156 kBps
TUN/TAP: Running Rx thread...
```

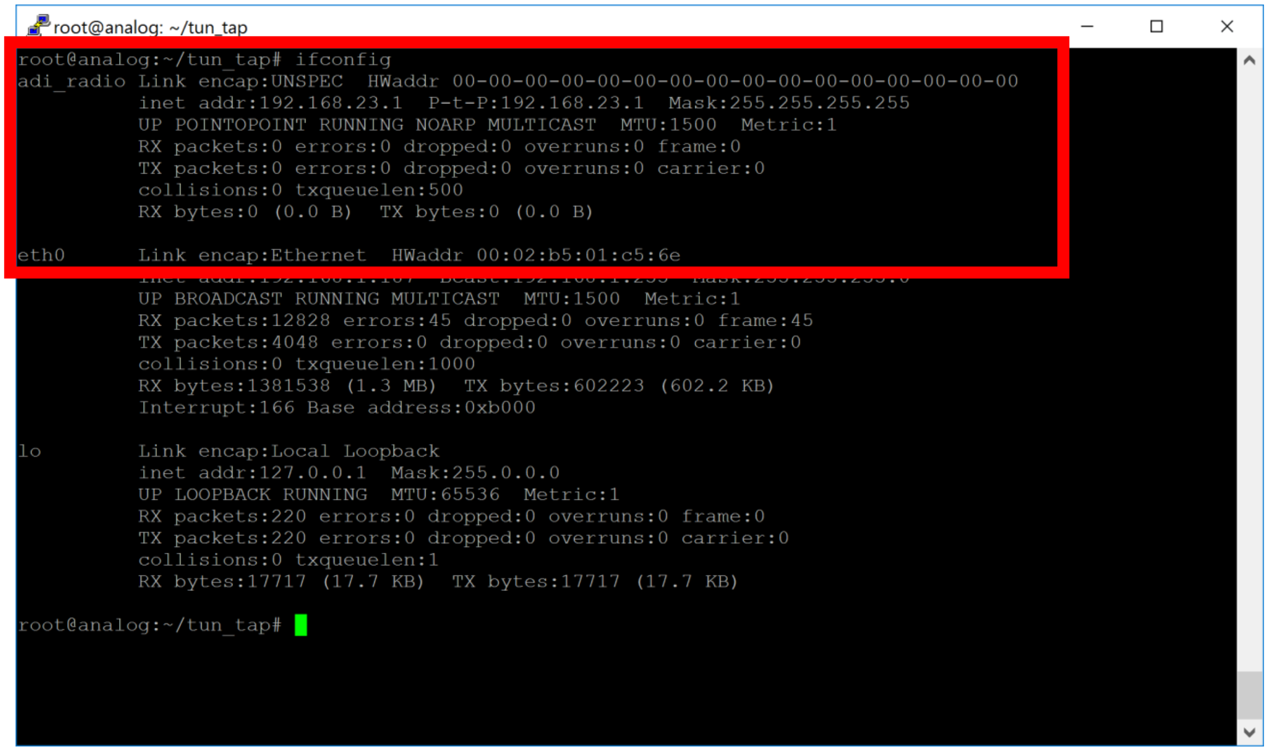
Since status information is provided during runtime we do not want to launch this process in the background. However, we do want to further investigate the current PackRF device. ***Next open a second copy of PuTTY, as we did previously, and login to Radio 1 with IP 192.168.3.1.*** An easy way to do this is to right click on the top of the currently open window and select "Duplicate Session" as in Figure 16.



```
root@analog: ~/tun_tap
root@analog:~/tun_tap#
```

Figure 16

Once logged in again, type "ifconfig" again and you should observe a new interface similar to Figure 17. The new networking interface "adi_radio" will contain the IP address of the modem assigned previously. Since the modem is exposed through TUN/TAP to the OS and other applications it is just another interface it can use, and will automatically route traffic with subnet 192.168.23.X through it.



```
root@analog: ~/tun_tap
root@analog:~/tun_tap# ifconfig
adi_radio Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:192.168.23.1  P-t-P:192.168.23.1  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth0      Link encap:Ethernet  HWaddr 00:02:b5:01:c5:6e
          [illegible]
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:12828 errors:45 dropped:0 overruns:0 frame:45
          TX packets:4048 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1381538 (1.3 MB)  TX bytes:602223 (602.2 KB)
          Interrupt:166 Base address:0xb000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:220 errors:0 dropped:0 overruns:0 frame:0
          TX packets:220 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:17717 (17.7 KB)  TX bytes:17717 (17.7 KB)

root@analog:~/tun_tap#
```

Figure 17

Before we can start passing data through this interface we need to set up the second PackRF device. Since the second PackRF is connected through a JTAG interface we need to determine the connect COM port. ***To do so open Device Manager from the start menu as in Figure 18***.

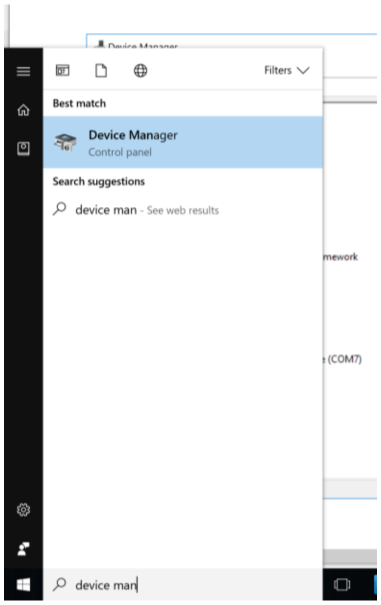


Figure 18

After Device Manager is launched scroll down to "Ports (COM & LPT)". Expanding this category will show the UART device and the COM port, which is COM7 in this case of Figure 19.

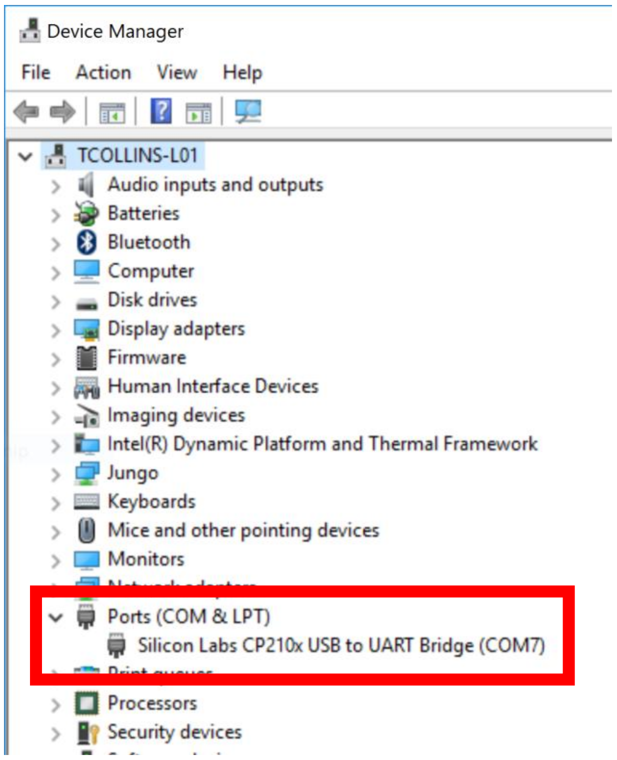
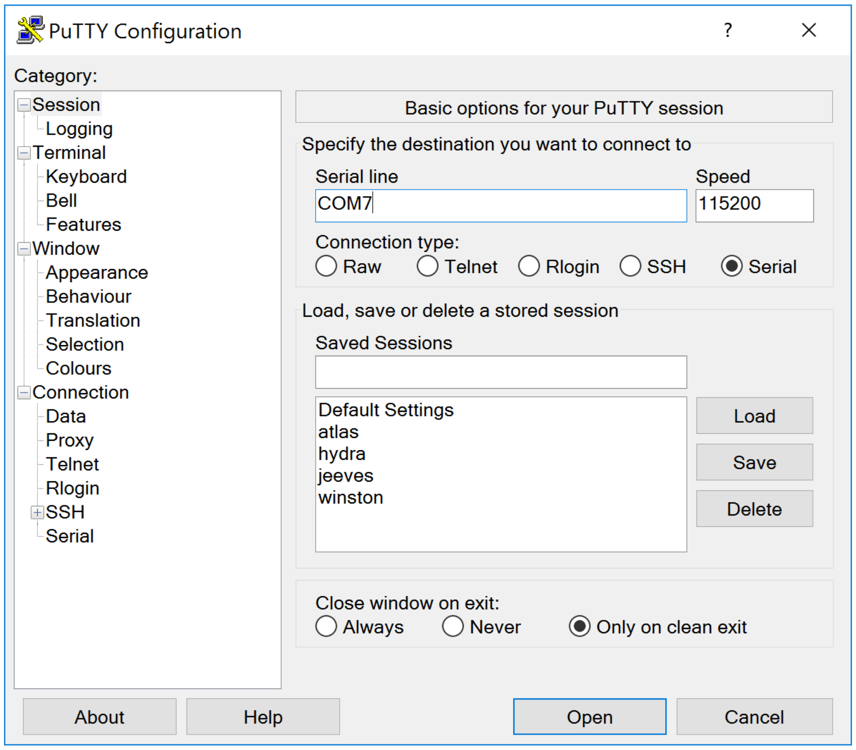


Figure 19

Finally, start by launching a third session of PuTTY. Select "Serial" as a connection type and update the speed to 115200 and the Serial line to be the found COM port. Your configuration should look like Figure 20. This PackRF has the same username and password as previously (root:analog).



Specify the destination you want to connect to

Serial line: COM7

Speed: 115200

Connection type: Raw Telnet Rlogin SSH [✓ Serial]

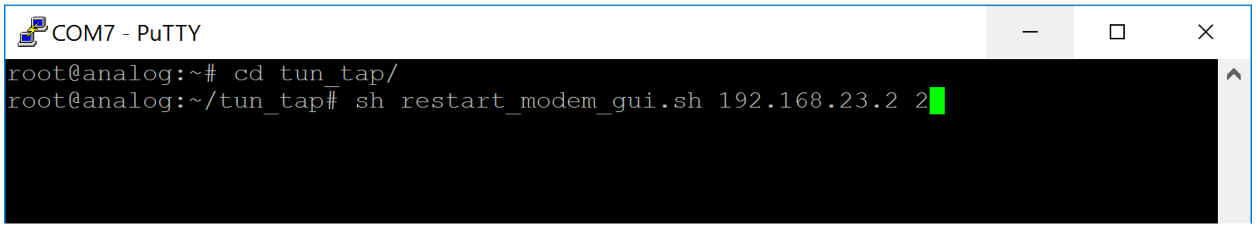
Saved Sessions:

Default Settings
atlas
hydra
jeeves
winston

Close window on exit: Always Never [✓ Only on clean exit]

Figure 20

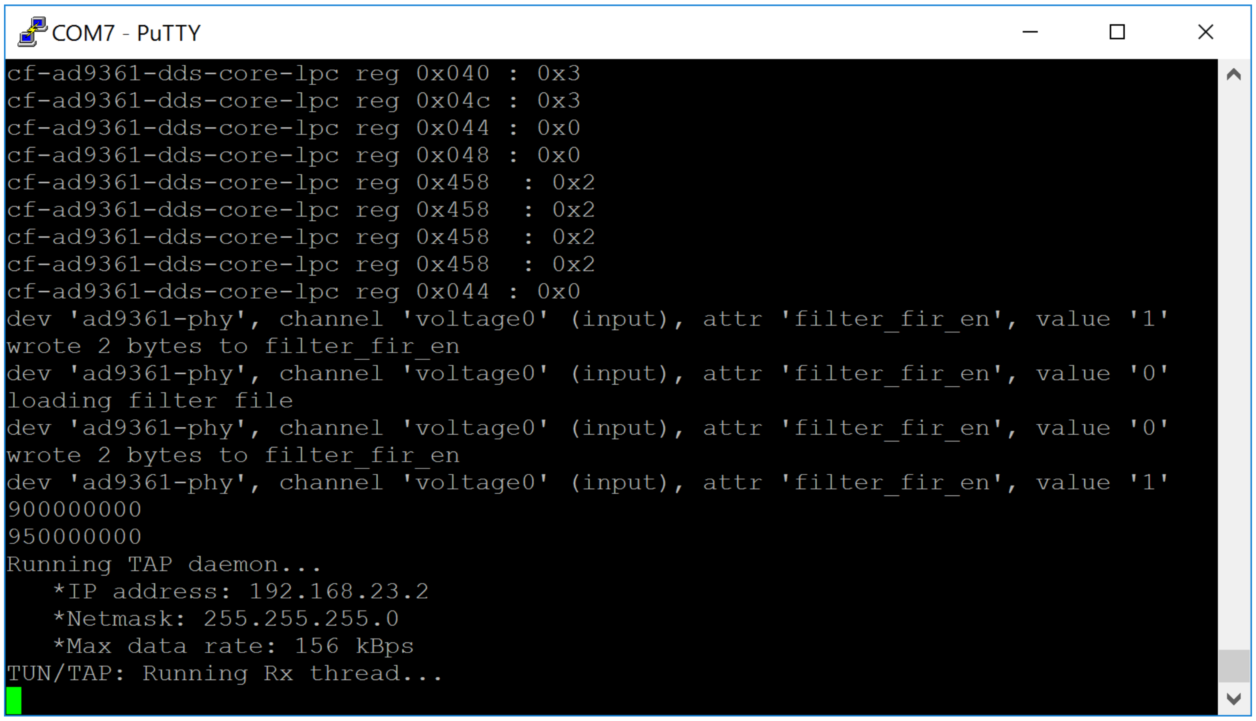
Again, navigate to the "tun_tap" folder. To configure the second PackRF on a different IP and channel mapping run the "restart_modem.sh" script with two arguments as in Figure 21.



```
root@analog:~# cd tun_tap/
root@analog:~/tun_tap# sh restart_modem_gui.sh 192.168.23.2 2
```

Figure 21

Like Radio 1, this will configure the modem, provide some status information, then bring up the TUN/TAP interface. The values displayed include some debug register values, and we also print out some transceiver configuration information such as: center frequency, sample rate, and RF bandwidth. This should look similar to Figure 22.



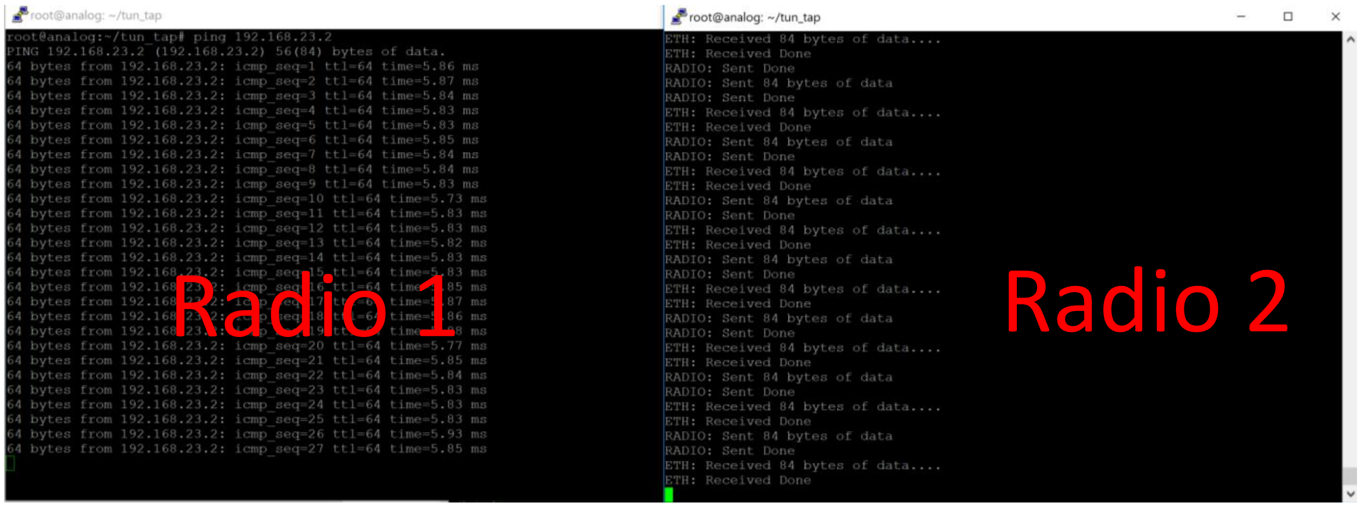
```
cf-ad9361-dds-core-lpc reg 0x040 : 0x3
cf-ad9361-dds-core-lpc reg 0x04c : 0x3
cf-ad9361-dds-core-lpc reg 0x044 : 0x0
cf-ad9361-dds-core-lpc reg 0x048 : 0x0
cf-ad9361-dds-core-lpc reg 0x458  : 0x2
cf-ad9361-dds-core-lpc reg 0x458  : 0x2
cf-ad9361-dds-core-lpc reg 0x458  : 0x2
cf-ad9361-dds-core-lpc reg 0x458  : 0x2
cf-ad9361-dds-core-lpc reg 0x044 : 0x0
dev 'ad9361-phy', channel 'voltage0' (input), attr 'filter_fir_en', value '1'
wrote 2 bytes to filter_fir_en
dev 'ad9361-phy', channel 'voltage0' (input), attr 'filter_fir_en', value '0'
loading filter file
dev 'ad9361-phy', channel 'voltage0' (input), attr 'filter_fir_en', value '0'
wrote 2 bytes to filter_fir_en
dev 'ad9361-phy', channel 'voltage0' (input), attr 'filter_fir_en', value '1'
900000000
950000000
Running TAP daemon...
   *IP address: 192.168.23.2
   *Netmask: 255.255.255.0
   *Max data rate: 156 kBps
TUN/TAP: Running Rx thread...
```

Figure 22

At this point we have both modem configured on each PackRF, which should allow us to communicate across the link.

Ping Across the Link

From Radio 1's console that is not captured by the modem daemon or process, we can start doing some simple tests to check the link. In Figure 23, we show two of our open terminals. The one on the left is Radio 1, which is pinging across the link to Radio 2. The terminal on the right, which is from Radio 2, we are monitoring the status messages from the modem daemon. Repeat this process by using the non-daemon open terminal for Radio 1. ***In this terminal (for Radio 1) run the command "ping 192.168.23.2"***. As we have seen 192.168.23.2 is the IP of the TUN/TAP interface for Radio 2.



```
root@analog:~/tun_tap# ping 192.168.23.2
PING 192.168.23.2 (192.168.23.2) 56(84) bytes of data.
64 bytes from 192.168.23.2: icmp_seq=1 ttl=64 time=5.86 ms
64 bytes from 192.168.23.2: icmp_seq=2 ttl=64 time=5.87 ms
64 bytes from 192.168.23.2: icmp_seq=3 ttl=64 time=5.84 ms
64 bytes from 192.168.23.2: icmp_seq=4 ttl=64 time=5.83 ms
64 bytes from 192.168.23.2: icmp_seq=5 ttl=64 time=5.83 ms
64 bytes from 192.168.23.2: icmp_seq=6 ttl=64 time=5.85 ms
64 bytes from 192.168.23.2: icmp_seq=7 ttl=64 time=5.84 ms
64 bytes from 192.168.23.2: icmp_seq=8 ttl=64 time=5.84 ms
64 bytes from 192.168.23.2: icmp_seq=9 ttl=64 time=5.83 ms
64 bytes from 192.168.23.2: icmp_seq=10 ttl=64 time=5.73 ms
64 bytes from 192.168.23.2: icmp_seq=11 ttl=64 time=5.83 ms
64 bytes from 192.168.23.2: icmp_seq=12 ttl=64 time=5.83 ms
64 bytes from 192.168.23.2: icmp_seq=13 ttl=64 time=5.82 ms
64 bytes from 192.168.23.2: icmp_seq=14 ttl=64 time=5.83 ms
64 bytes from 192.168.23.2: icmp_seq=15 ttl=64 time=5.83 ms
64 bytes from 192.168.23.2: icmp_seq=16 ttl=64 time=5.85 ms
64 bytes from 192.168.23.2: icmp_seq=17 ttl=64 time=5.87 ms
64 bytes from 192.168.23.2: icmp_seq=18 ttl=64 time=5.86 ms
64 bytes from 192.168.23.2: icmp_seq=19 ttl=64 time=5.[illegible]8 ms
64 bytes from 192.168.23.2: icmp_seq=20 ttl=64 time=5.77 ms
64 bytes from 192.168.23.2: icmp_seq=21 ttl=64 time=5.85 ms
64 bytes from 192.168.23.2: icmp_seq=22 ttl=64 time=5.84 ms
64 bytes from 192.168.23.2: icmp_seq=23 ttl=64 time=5.83 ms
64 bytes from 192.168.23.2: icmp_seq=24 ttl=64 time=5.83 ms
64 bytes from 192.168.23.2: icmp_seq=25 ttl=64 time=5.83 ms
64 bytes from 192.168.23.2: icmp_seq=26 ttl=64 time=5.93 ms
64 bytes from 192.168.23.2: icmp_seq=27 ttl=64 time=5.85 ms
```

```
ETH: Received 84 bytes of data....
ETH: Received Done
RADIO: Sent Done
RADIO: Sent 84 bytes of data
RADIO: Sent Done
ETH: Received 84 bytes of data....
ETH: Received Done
RADIO: Sent 84 bytes of data
RADIO: Sent Done
ETH: Received 84 bytes of data....
ETH: Received Done
RADIO: Sent 84 bytes of data
RADIO: Sent Done
ETH: Received 84 bytes of data....
ETH: Received Done
RADIO: Sent 84 bytes of data
RADIO: Sent Done
ETH: Received 84 bytes of data....
ETH: Received Done
RADIO: Sent 84 bytes of data
RADIO: Sent Done
ETH: Received 84 bytes of data....
ETH: Received Done
RADIO: Sent 84 bytes of data
RADIO: Sent Done
ETH: Received 84 bytes of data....
ETH: Received Done
RADIO: Sent 84 bytes of data
RADIO: Sent Done
ETH: Received 84 bytes of data....
ETH: Received Done
```

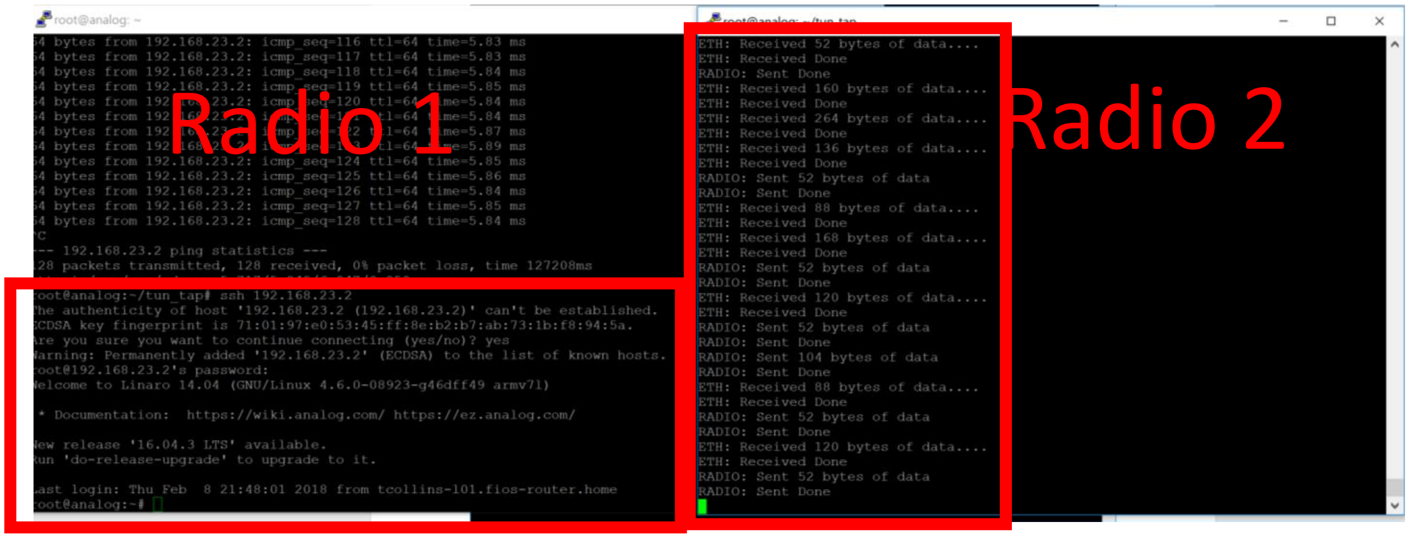
Figure 23

On Radio 2's daemon running terminal, we should observe messages like "RADIO: Sent 84 bytes of data". This means the TUN/TAP interface received a packet of 84 bytes from the modem on the FPGA. This is actually the size of a ping packet. We also should see "ETH: Received 84 bytes of data", which means that the Ethernet interface received a packet of 84 bytes. This is the ping acknowledgement which we will send back to Radio 1. Now it is possible to lose packets in the is process since: this is purely a UDP data transaction, and there is no retransmission intelligence built into the modem IP itself. This design usually operates at under 1% PER for pings (ping only measures in integers). ***Press Ctrl+C in the console of Radio 1 to stop pings before moving onto the next section.***

This demonstrates two useful features. First, we are able to interact seamlessly with applications running on the host OS, and second we are able to transmit data wireless with the AD9361 and our generated Modem IP.

SSH Through the Modem

Now that we know we can communicate through the TUN/TAP interface, we can try something more interesting. Staying in the console we will next SSH through the TUN/TAP interface from Radio 1 to Radio 2. ***Again, in the non-daemon console on Radio 1, type "ssh 192.168.23.2". You will be asked for a prompt and eventually a password like Figure 24. Accept the fingerprint (if asked) and type the password "analog" again to access the second PackRF.***



```
4 bytes from 192.168.23.2: icmp_seq=116 ttl=64 time=5.83 ms
4 bytes from 192.168.23.2: icmp_seq=117 ttl=64 time=5.83 ms
4 bytes from 192.168.23.2: icmp_seq=118 ttl=64 time=5.84 ms
4 bytes from 192.168.23.2: icmp_seq=119 ttl=64 time=5.85 ms
4 bytes from 192.168.23.2: icmp_seq=120 ttl=64 time=5.84 ms
4 bytes from 192.168.23.2: icmp_seq=121 ttl=64 time=5.84 ms
4 bytes from 192.168.23.2: icmp_seq=122 ttl=64 time=5.87 ms
4 bytes from 192.168.23.2: icmp_seq=123 ttl=64 time=5.89 ms
4 bytes from 192.168.23.2: icmp_seq=124 ttl=64 time=5.85 ms
4 bytes from 192.168.23.2: icmp_seq=125 ttl=64 time=5.86 ms
4 bytes from 192.168.23.2: icmp_seq=126 ttl=64 time=5.84 ms
4 bytes from 192.168.23.2: icmp_seq=127 ttl=64 time=5.85 ms
4 bytes from 192.168.23.2: icmp_seq=128 ttl=64 time=5.84 ms
C
-- 192.168.23.2 ping statistics ---
28 packets transmitted, 128 received, 0% packet loss, time 127208ms
oot@analog:~/tun_tap# ssh 192.168.23.2
he authenticity of host '192.168.23.2 (192.168.23.2)' can't be established.
CDSA key fingerprint is 71:01:97:e0:53:45:ff:8e:b2:b7:ab:73:1b:f8:94:5a.
re you sure you want to continue connecting (yes/no)? yes
arning: Permanently added '192.168.23.2' (ECDSA) to the list of known hosts.
oot@192.168.23.2's password:
elcome to Linaro 14.04 (GNU/Linux 4.6.0-08923-g46dff49 armv7l)

 * Documentation:  https://wiki.analog.com/ https://ez.analog.com/

ew release '16.04.3 LTS' available.
un 'do-release-upgrade' to upgrade to it.

ast login: Thu Feb  8 21:48:01 2018 from tcollins-101.fios-router.home
oot@analog:~#
```

```
ETH: Received 52 bytes of data....
ETH: Received Done
RADIO: Sent Done
ETH: Received 160 bytes of data....
ETH: Received Done
ETH: Received 264 bytes of data....
ETH: Received Done
ETH: Received 136 bytes of data....
ETH: Received Done
RADIO: Sent 52 bytes of data
RADIO: Sent Done
ETH: Received 88 bytes of data....
ETH: Received Done
ETH: Received 168 bytes of data....
ETH: Received Done
RADIO: Sent 52 bytes of data
RADIO: Sent Done
ETH: Received 120 bytes of data....
ETH: Received Done
RADIO: Sent 52 bytes of data
RADIO: Sent Done
RADIO: Sent 104 bytes of data
RADIO: Sent Done
ETH: Received 88 bytes of data....
ETH: Received Done
RADIO: Sent 52 bytes of data
RADIO: Sent Done
ETH: Received 120 bytes of data....
ETH: Received Done
RADIO: Sent 52 bytes of data
RADIO: Sent Done
```

Figure 24

From the still running log on Radio 2, we will begin to see packets of different sizes to appear. This is the SSH connect initializing and authenticating. ***Once connected type "ifconfig eth0".*** It should provide an interface with IP address of the second radio. This proves that we are in fact remotely connected to Radio. The connection here utilizes a TCP connection (not UDP), and if packets are lost retransmissions will occur by the application and OS itself in the transport layer.