

Software hands-on training kit: What software do I need for my SDR?

Instructions for Hands-on Examples

!The numbering of these chapters corresponds to the ones in the .ppt presentation. E.g. "1. IIO Oscilloscope" has the same number here as in the .ppt presentation.

! **Most of the commands** that are required to be run in the terminal are also within [terminal_commands.txt](#) file from [Desktop](#).

Note: "#" -> comment; "\$" -> **command**, don't type "\$" in terminal

1. IIO Oscilloscope

1.1. Connect Rx1 and Tx1 using the SMA cable from the kit by making a loopback between them as depicted in the picture below:



1.2. Make sure the back panel "STAT" blue LED is blinking. This shows that the boot stage is successful (see picture below). **If the LEDs are red**, it means that the board is in shut down mode (to start it, press the pushbutton once and wait for a blinking pattern).

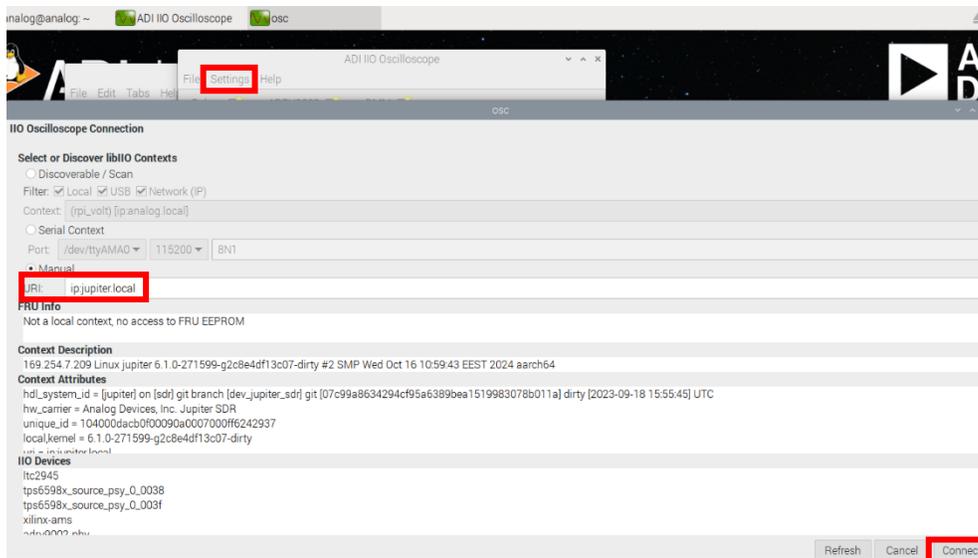


1.3. Open IIO Oscilloscope: **open the terminal** from the top bar icon, **type "osc"** in it (as in the picture below) and **press enter**. Wait a few seconds for the IIO Oscilloscope app to open.

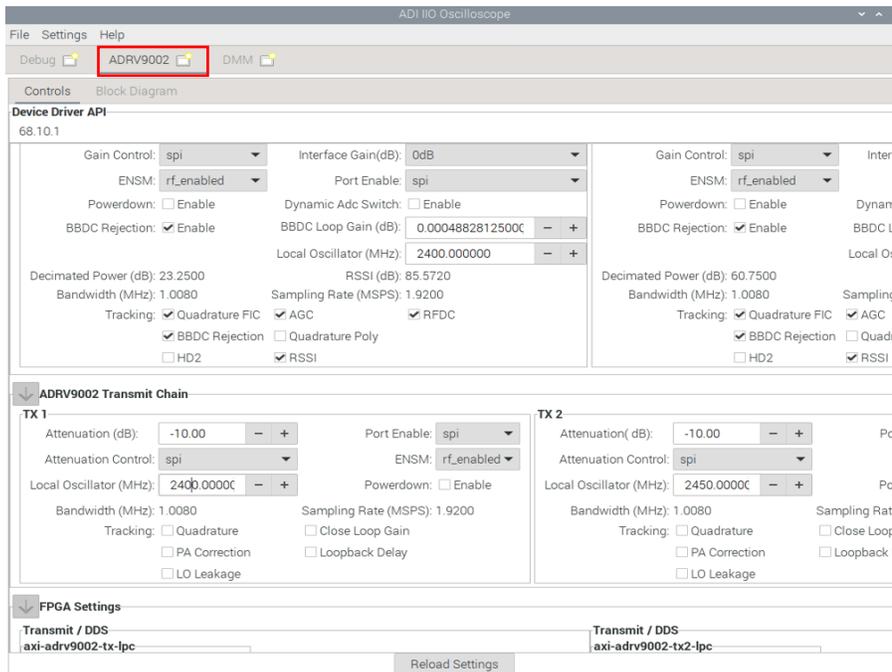


1.4. To connect with the board, make sure first the Ethernet cable is connected between Jupiter SDR and the Raspberry Pi Keyboard. To connect Jupiter to the IIO Oscilloscope app, go to:

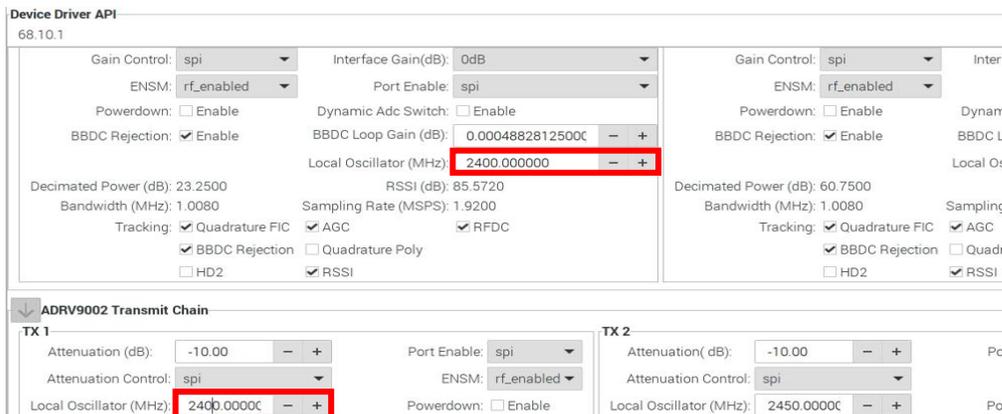
"Settings" -> **"Connect"** and type **"ip:jupiter.local"** in the **"Manual"** section, press **"Refresh"** and then **"Connect"** (see picture below).



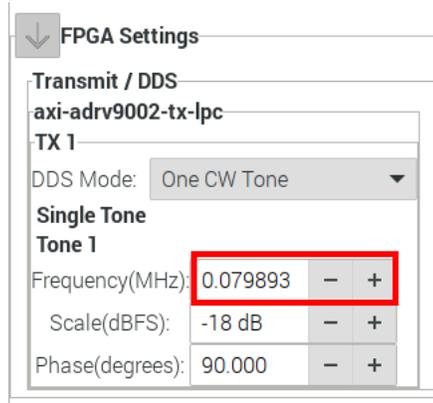
1.5. Use DDS On loopback: We will transmit a 80KHz tone on top of the Tx LO and receive it on the Rx path. First, select "ADRV9002" pane from the top bar. Scroll down until you see the settings shown in the below images in "ADRV9002" pane.



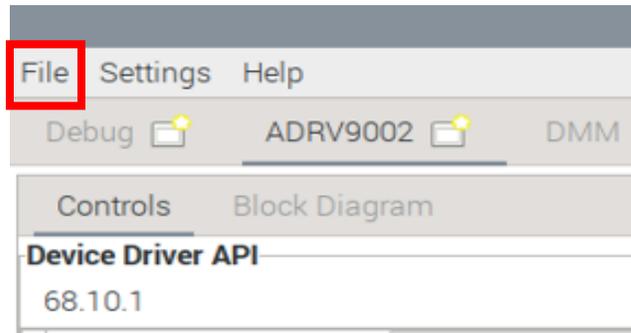
1.6. Make sure the Tx1 LO and Rx1 LO have the same frequency (see pictures below).



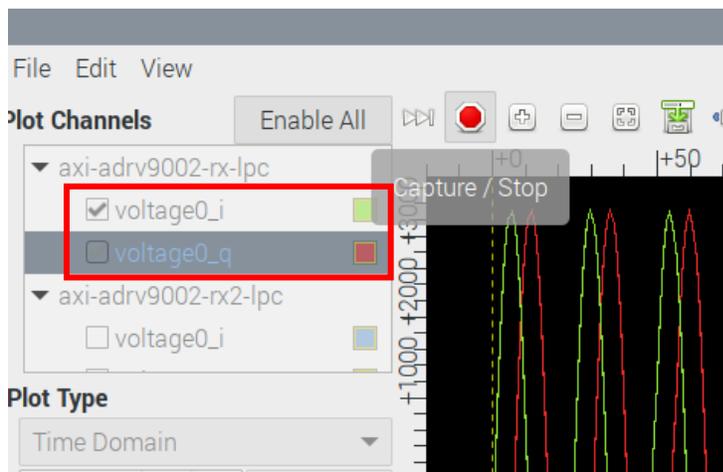
1.7. Scroll down until you see the DDS settings. If you change the DDS to another frequency, make sure it is within 1MHz BW ($f_{dds} = (-500\text{kHz}, 500\text{kHz})$).



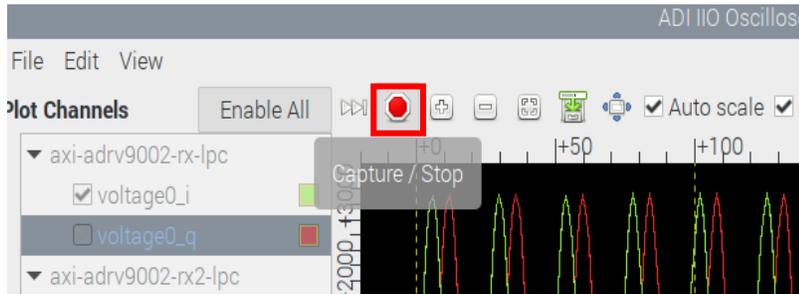
1.8. Open and start the Plot: Go to "File" -> "New Plot".



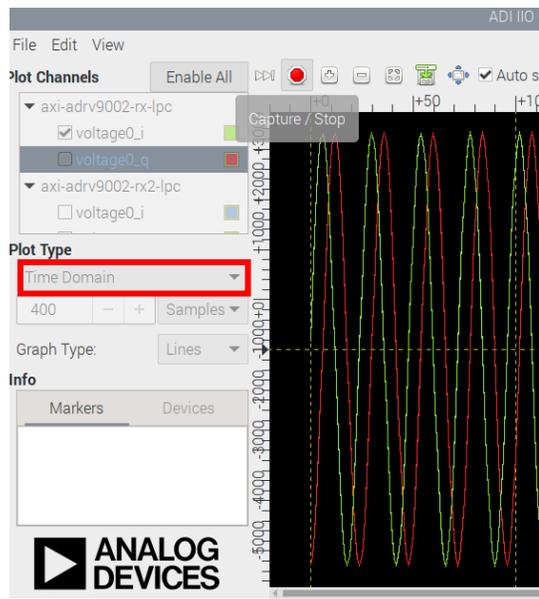
1.9. On the channels window, check both "voltage0_i" and "voltage0_q" from "axi-adrv9002-rx-lpc".



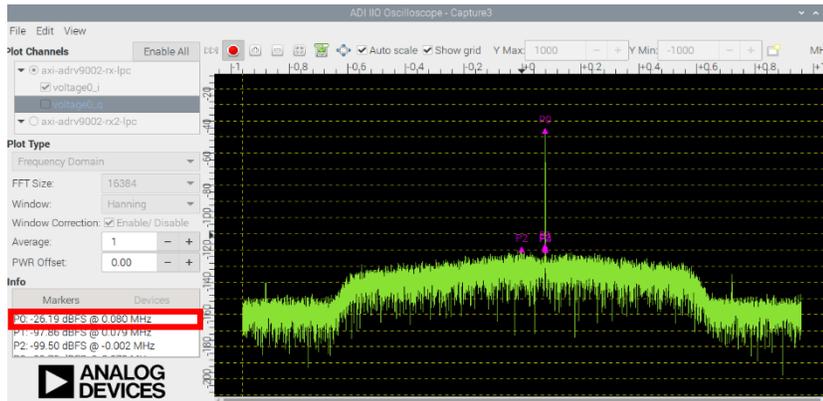
1.10. To start capturing data, press the arrow button from the top bar (above plots). A red dot should appear in its place as in the below picture.



1.11. If you want to switch between Time and Frequency domain, you can change the Plot Type. Stop the plot before you want to switch to Frequency Domain (see below picture).

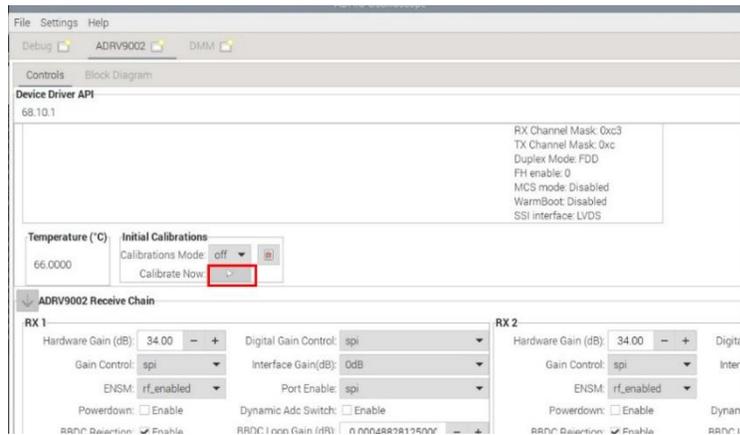


1.12. To see peak markers on the Frequency domain, right click on the plot and select "Show Markers".



1.13. Tweak Tx Attenuation and observe the change in the plot: Decrease or increase the attenuation for Tx1 and observe the change on the FFT Plot. The range for TX is -41dB to 0dB (see picture below).

1.14. (for future use) If you see an image of the complex sinusoid or if spurs appear in the domain when you switch the LO to a higher frequency, press the calibration button from IIO Oscilloscope (shown below).



1.15. Close IIO Oscilloscope and Reboot Jupiter:

Method 1:

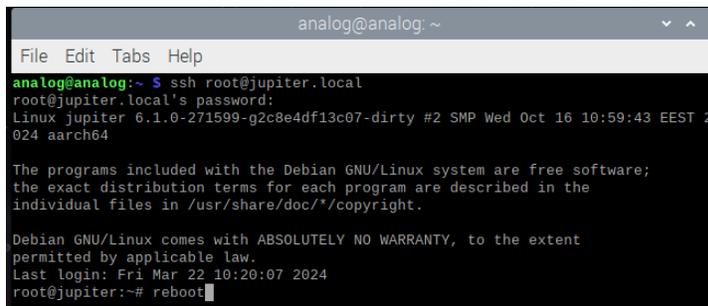
Open the terminal and run the following commands:

```
$ ssh root@jupiter.local
```

press Enter and enter "analog" as password

```
$ reboot
```

press Enter



Method 2:

Press once the push-button on the back of the Jupiter

Wait for the LEDs to turn red

Press once more the push-button to boot again

Wait for the STAT LED to blink again (blue color)

2. Transmit and receive a complex sinusoid with GNU Radio

2.1. Connect Rx1 and Tx1 using the SMA cable from the kit by making a loopback (as shown below).



2.2. To open GNU Radio companion app, run the following commands:

Open the Terminal

```
$ cd /home/analog/Desktop
```

```
$ sudo ./start-grc.sh
```

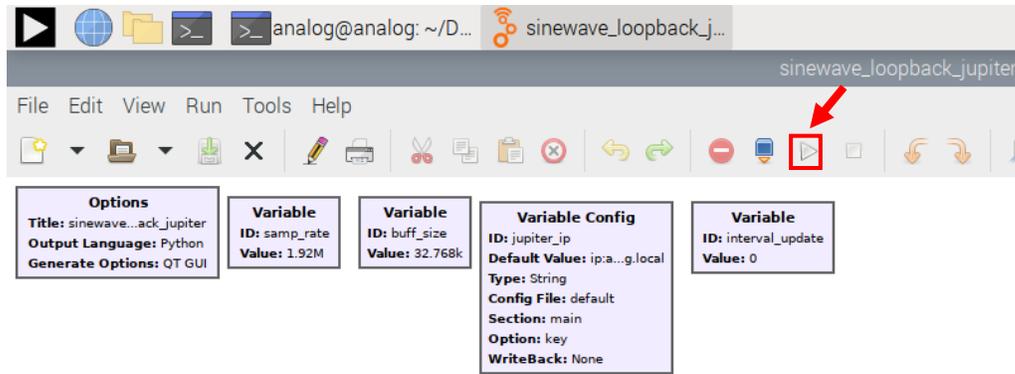
enter "analog" as password and press enter.

```
analog@analog: ~/Desktop
File Edit Tabs Help
analog@analog:~ $ cd Desktop/
analog@analog:~/Desktop $ sudo ./start-grc.sh
```

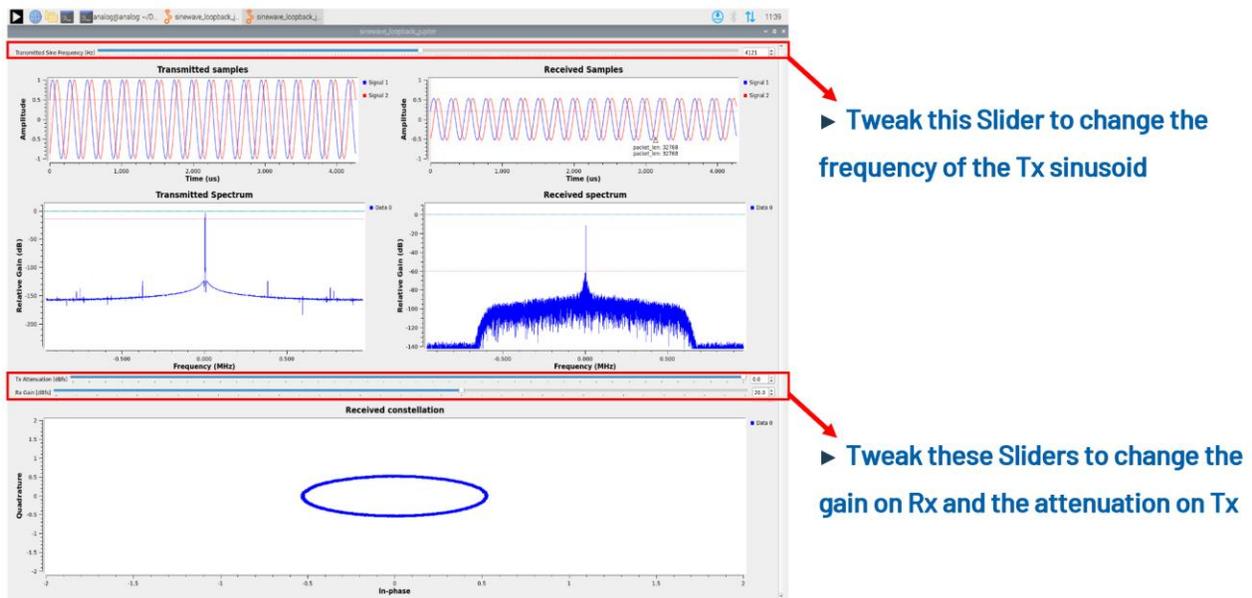
2.3. Within GNU Radio companion app, open the following ".grc" flowgraph from "File" -> "Open":

```
/home/analog/Desktop/ftc_2024/1_sinewave_loopback_gnuradio/sinewave_loopback_jupiter.grc
```

2.4. To run the flowgraph press the arrow from the top bar inside the app as depicted in the following picture:



2.5. After you run the flowgraph, you can **tweak the sliders** to modify the frequency of the sinewave transmitted, the gain on RX path and the attenuation on TX path.



Note: To stop the running plot: click on the mouse wheel -> stop

To start again the plot: click on the mouse wheel -> start

To zoom on a plot: encircle with left click the area you want to zoom

To hide a trace on a plot: click on the label corresponding to that trace from the top right corner of the plot.

2.6. **Close** the running flowgraph from the right corner of the window.

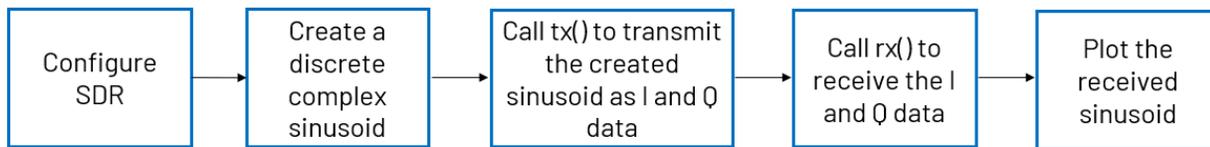
3. Transmit and receive a complex sinusoid with Python

3.1. For this example, use two SMA cables to make loopback between TX1 -> RX1 and TX2 -> RX2 as depicted below:

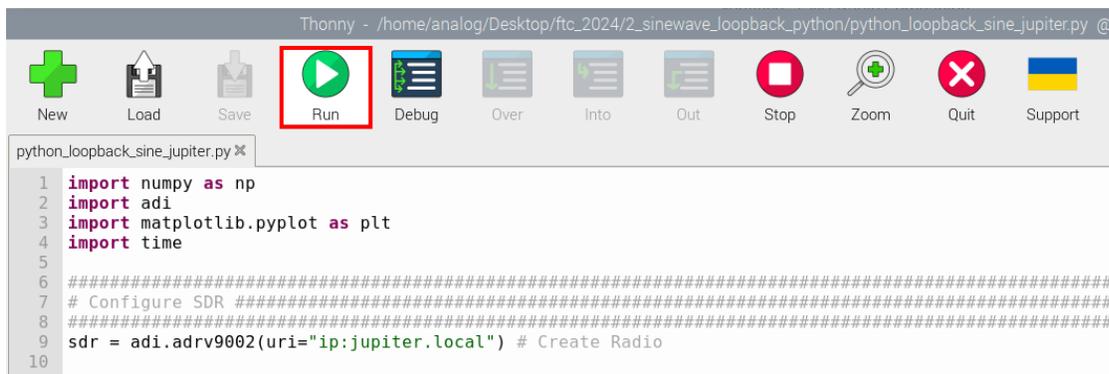


3.2. On your desktop, open "ftc_2024" folder and then open "2_sinewave_loopback_python" folder. In this folder, right click on "python_loopback_sine_jupiter.py" and click on "Thonny". Thonny is an IDE that can be used to edit and run python code.

3.3. Observe in the python code the following sections:



3.4. To run the "python_loopback_sine_jupiter.py" python script, click on the run button.



3.5. Close the generated plots after you analyzed them.

3.6. In the Terminal, run the following commands in the terminal to **change the current profile with a 61.44MHz Sample Rate profile**:

first connect to Jupiter with ssh:

```
$ ssh root@jupiter.local
```

enter "analog" as password

copy the following command from **terminal_commands.txt** from Desktop and paste it using

"ctrl+shift+v" in the terminal

```
$ cat /home/analog/workspace/jupiter_profiles/jupiter_61_44MHz_profile.json > /sys/bus/iio/devices/iio\:device1/profile_config
```

exit ssh connection

```
$ exit
```

in "/home/analog/workspace/jupiter_profiles" are more ".json" profiles that you can try with

different sampling rates

3.7. Run again "python_loopback_sine_jupiter.py" with **Thonny IDE**. You should see the same plots and same frequencies as with the previous profile, but with more samples and a wider spectrum. You can also see the changed frequency.

Note: It may take longer to run these plots with the new profile because more points are needed to be plotted. Please be patient 😊

3.8. **Close** the generated plots after you analyzed them.

3.9. **Reboot** Jupiter to reverse to the default profile required for the following examples.

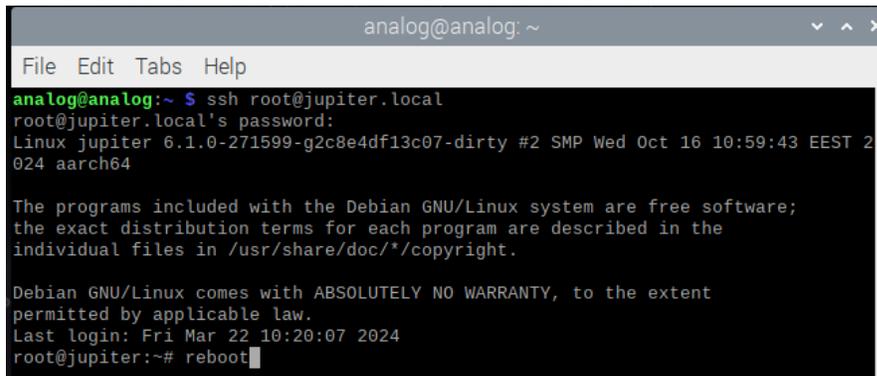
Method 1:

open terminal and run the following commands

```
$ ssh root@jupiter.local
```

enter "analog" as password

\$ reboot

A terminal window titled 'analog@analog: ~' with a menu bar containing 'File Edit Tabs Help'. The terminal output shows a successful SSH connection to 'root@jupiter.local'. The user 'analog' enters the command '\$ ssh root@jupiter.local'. The terminal displays the system's boot information, including the kernel version 'Linux jupiter 6.1.0-271599-g2c8e4df13c07-dirty #2 SMP Wed Oct 16 10:59:43 EEST 2024 aarch64'. It also shows the Debian GNU/Linux warranty disclaimer and the last login time 'Fri Mar 22 10:20:07 2024'. Finally, the user 'root' enters the command '# reboot' at the 'root@jupiter:~#' prompt.

```
analog@analog:~ $ ssh root@jupiter.local
root@jupiter.local's password:
Linux jupiter 6.1.0-271599-g2c8e4df13c07-dirty #2 SMP Wed Oct 16 10:59:43 EEST 2024 aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Mar 22 10:20:07 2024
root@jupiter:~# reboot
```

Method 2:

Press once the push-button on the back of the Jupiter

Wait for the LEDs to turn red

Press once more the push-button to boot again

Wait for the blue "STAT" LED to blink to indicate a successful boot

4. How to explore and tweak device attributes

4.1. To see all devices, run the following command in the terminal (it is not required to connect with ssh to Jupiter and run this command on the target):

```
$ iio_attr -u ip:jupiter.local -d
```

```
analog@analog:~$ iio_attr -u ip:jupiter.local -d
IIO context has 10 devices:
hwmon0, ltc2945: found 0 device attributes
hwmon1, tps6598x_source_psy_0_0038: found 0 device attributes
hwmon2, tps6598x_source_psy_0_003f: found 0 device attributes
iio:device0, xilinx-ams: found 1 device attributes
iio:device1, adrv9002-phy: found 17 device attributes
iio:device2, axi-adrv9002-rx-lpc: found 3 device attributes
iio:device3, axi-adrv9002-rx2-lpc: found 1 device attributes
iio:device4, axi-adrv9002-tx-lpc: found 3 device attributes
iio:device5, axi-adrv9002-tx2-lpc: found 3 device attributes
iio_sysfs trigger: found 2 device attributes
```

Most Important Devices used in the examples presented in this workshop

```
'adrv9002-phy' -> contains most of the configuration attributes (hardwaregain, LO freq. etc.)
'axi-adrv9002-rx-lpc' -> ADC on the RX1 path
'axi-adrv9002-rx2-lpc' -> ADC on the RX2 path
'axi-adrv9002-tx-lpc' -> DAC on the TX1 path
'axi-adrv9002-tx2-lpc' -> DAC on the TX2 path
```

4.2. Run the 'iio_info' command on the target to see all devices, all channels, and all attributes, but first connect to the board with ssh:

```
$ ssh root@jupiter.local
```

```
# insert 'analog' as password
```

```
$ iio_info
```

```
# use the following command to clear the terminal:
```

```
$ clear
```

```
# to see channels and attributes for a specific device, e.g. for adrv9002-phy, run the following command:
```

```
$ iio_info | grep -A 1000 'adrv9002-phy'
```

```
# the above command displays 1000 lines after the line where the string 'adrv9002-phy' is found in the iio_info contents
```

4.3. Change hardware gain attribute of RX1 to 5dB using libiio and the terminal.

Since "adrv9002-phy" contains most of the configuration attributes, first you need to look at this device channels and attributes to find the exact channel name and channel attribute name you want to find. To do that, run the following command:

```
$ iio_info | grep -A 200 adrv9002-phy
```

```
voltage0: (input)
32 channel-specific attributes found:
  attr 0: agc_tracking_en value: 1
  attr 1: bbdc_loop_gain_raw value: 1048576
  attr 2: bbdc_rejection_en value: 1
  attr 3: bbdc_rejection_tracking_en value: 1
  attr 4: decimated_power value: 16.500 dB
  attr 5: digital_gain_control_mode value: spi
  attr 6: digital_gain_control_mode_available value: automatic spi
  attr 7: dynamic_adc_switch_en value: 0
  attr 8: en value: 1
  attr 9: ensm_mode value: rf_enabled
  attr 10: ensm_mode_available value: calibrated primed rf_enabled
  attr 11: gain_control_mode value: spi
  attr 12: gain_control_mode_available value: spi pin automatic
  attr 13: hardwaregain value: 34.000000 dB
  attr 14: hd_tracking_en value: 0
  attr 15: interface_gain value: 0dB
  attr 16: interface_gain_available value: 0dB
  attr 17: nco_frequency ERROR: Unknown error 524
  attr 18: orx_bbdc_rejection_en ERROR: No such device (19)
  attr 19: orx_bbdc_rejection_tracking_en ERROR: No such device (19)
```

if you scroll up, you should find that the channel name is 'voltage0' and the attribute name is 'hardwaregain'.

to change the hardwaregain to 5 dB for RX1, run the following command in your terminal:

Note: you can copy and paste with `ctrl+c`, `ctrl + shift + v` the following command from

`terminal_commands.txt` from `Desktop`:

```
$ iio_attr -c -i adrv9002-phy voltage0 hardwaregain 5
```

after running this command, you can open `iio-oscilloscope` to see that the hardwaregain on RX1 is

now 5

run the following command in the terminal to exit the ssh connection:

```
$ exit
```

4.4. After playing with changing attributes from terminal, **reboot Jupiter:**

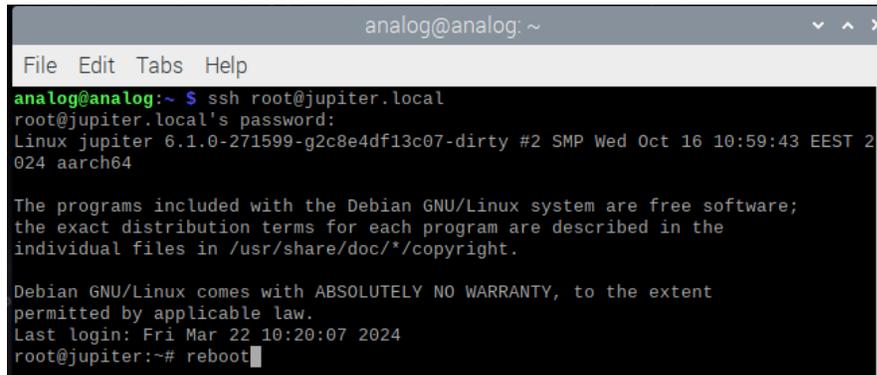
Method 1:

open terminal

\$ ssh root@jupiter.local

press Enter and enter "analog" as password

\$ reboot

A screenshot of a terminal window titled 'analog@analog: ~'. The window has a menu bar with 'File', 'Edit', 'Tabs', and 'Help'. The terminal content shows a user at 'analog@analog:~' typing '\$ ssh root@jupiter.local'. The prompt changes to 'root@jupiter.local's password:' and the user enters 'analog'. The terminal then displays system information: 'Linux jupiter 6.1.0-271599-g2c8e4df13c07-dirty #2 SMP Wed Oct 16 10:59:43 EEST 2024 aarch64'. It shows the Debian GNU/Linux license text: 'The programs included with the Debian GNU/Linux system are free software; the exact distribution terms for each program are described in the individual files in /usr/share/doc/*/copyright. Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law. Last login: Fri Mar 22 10:20:07 2024'. Finally, the prompt is 'root@jupiter:~# reboot' with a cursor at the end.

Method 2:

Press once the push-button on the back of the Jupiter

Wait for the LEDs to turn red

Press once more the push-button to boot again

Wait for the blue "STAT" LED to blink to indicate a successful boot

5. Doppler Radar with GNU Radio

5.1. For this example, connect one antenna to RX1 and one to TX1 on Jupiter female SMA ports as depicted in the picture below



5.2. Open GNU Radio (if you closed it previously) by running the following commands in the terminal:

```
$ cd /home/analog/Desktop
```

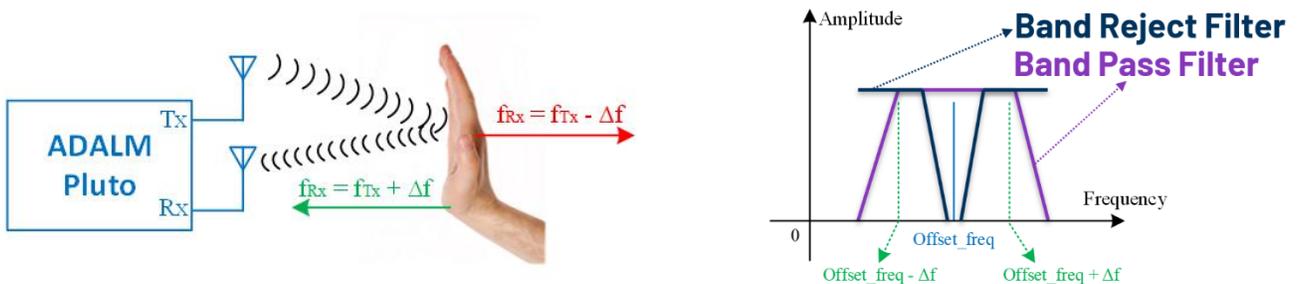
```
$ sudo ./start-grc.sh
```

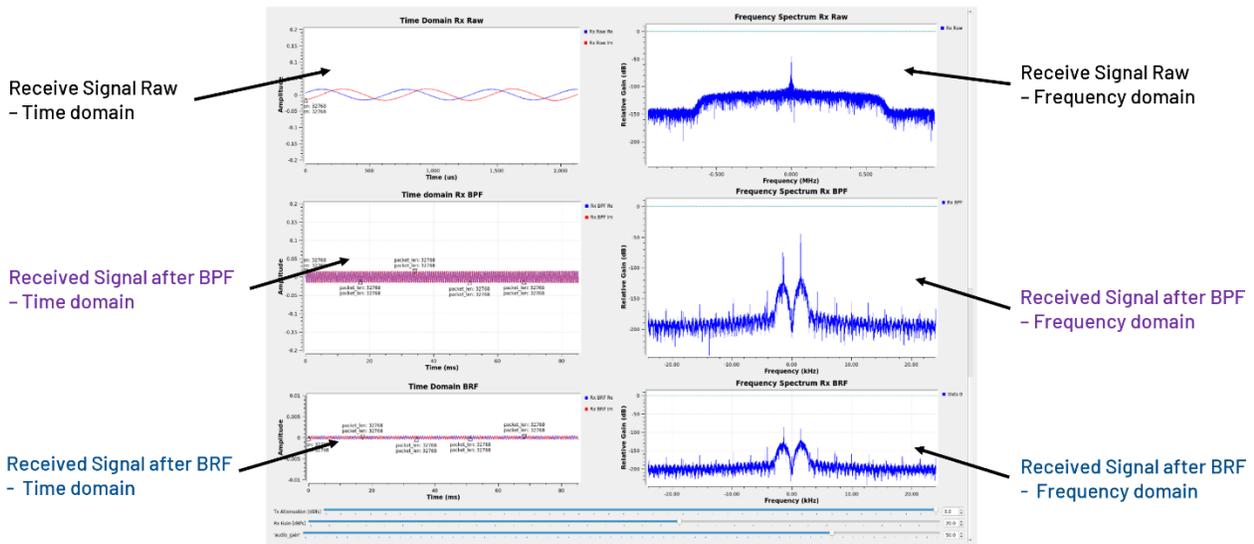
```
analog@analog: ~/Desktop
File Edit Tabs Help
analog@analog:~ $ cd Desktop/
analog@analog:~/Desktop $ sudo ./start-grc.sh
```

5.3. In GNU Radio companion app, open “doppler_radar_jupiter.grc” from File -> Open:

```
/home/analog/Desktop/ftc_2024/3_doppler_radar_gnudio/
```

5.4. Run the flowgraph, plug the headphones and move your hand in front of the antennas. Due to doppler effect, you should hear the movement of your hand into your headphones.





Note: To stop the running plot: click on the mouse wheel -> stop

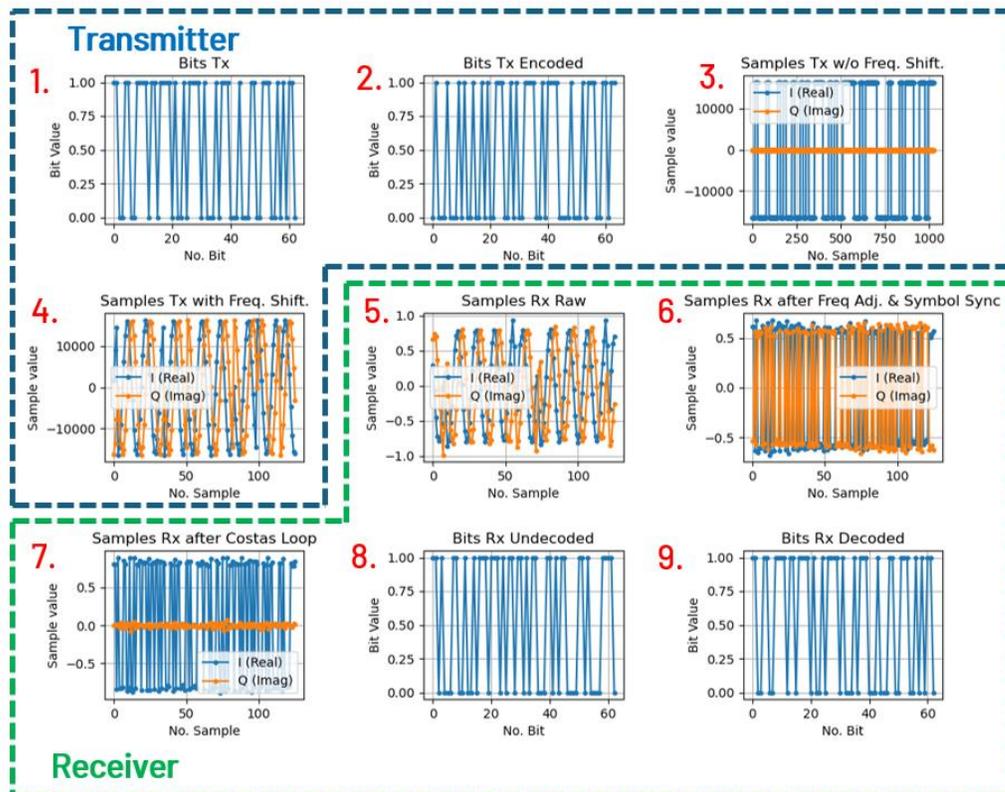
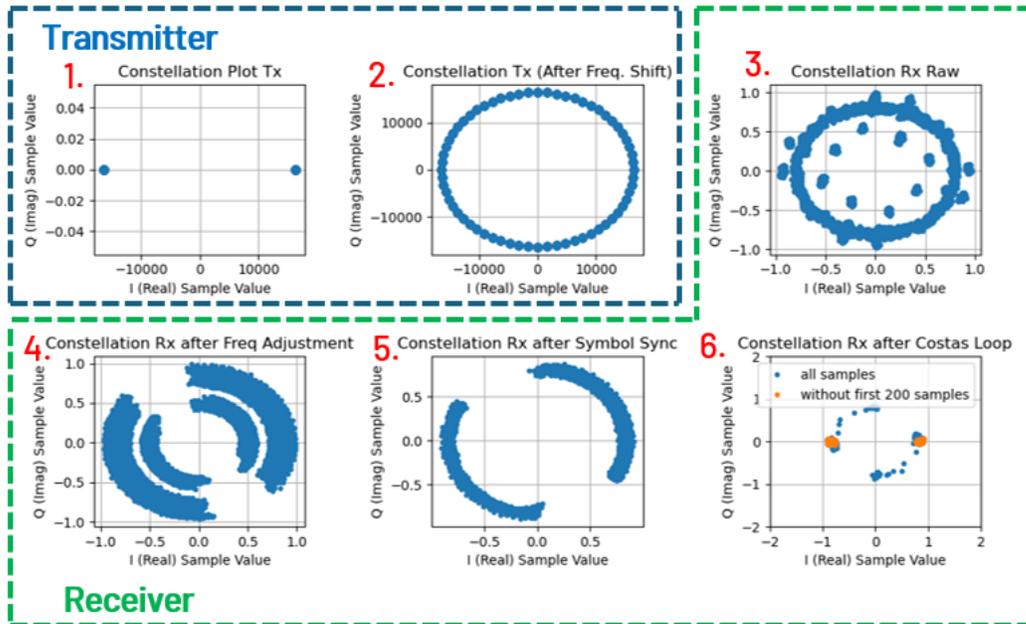
To start again the plot: click on the mouse wheel -> start

To zoom on a plot: encircle with left click the area you want to zoom

To hide a trace on a plot: click on the label corresponding to that trace from the top right corner of the plot.

5.5. **Close** the running flowgraphs after you finished analyzing them.

6.4. Observe how the data looks after each section of code, which corresponds to a block from the diagram below. Observe that on the time plots, "1." (Bits TX) and "9." (Bits RX Decoded) are the same.



6.5. Close the generated plots after you finished analyzing them.

7. Phase Shift Keying – QPSK in GNU Radio

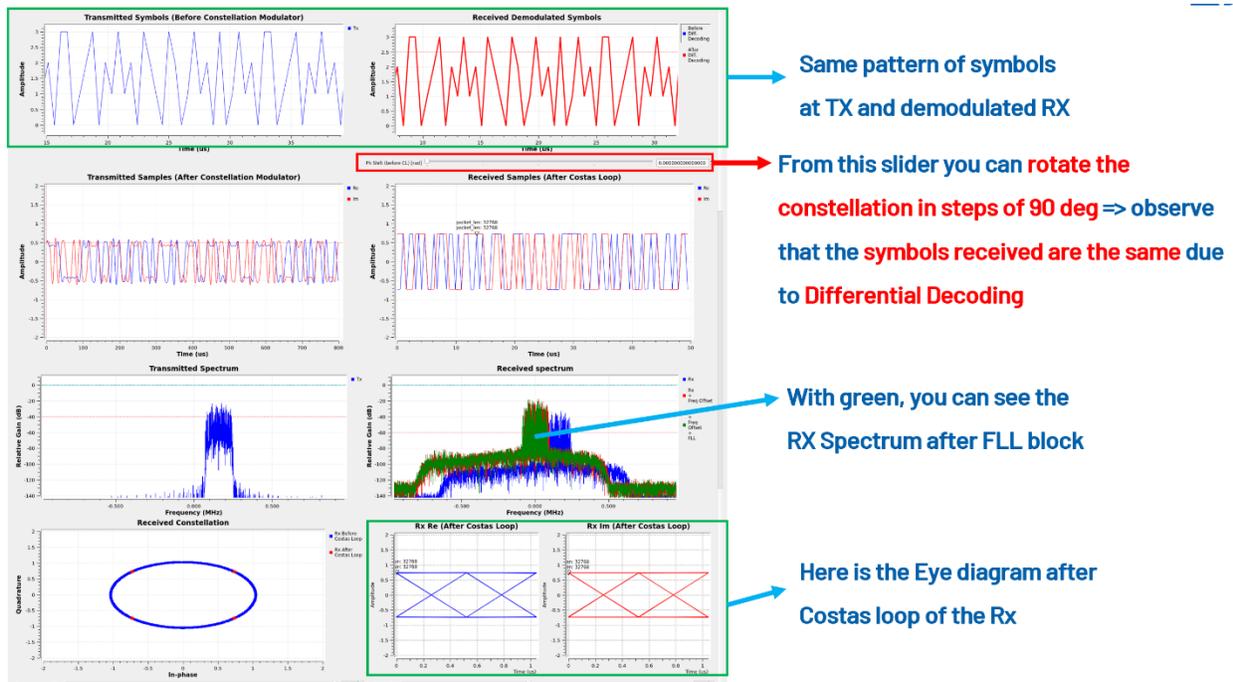
7.1. For this example, connect Rx1 and Tx1 using the SMA cable from the kit, making a loopback between them (as shown below).



7.2. In GNU Radio companion app, open from "File" -> "Open":

`/home/analog/Desktop/ftc_2024/5_qpsk_loopback_gnuradio/qpsk_loopback_jupiter.grc`

7.3. Run the flowgraph and observe that the pattern of symbols received is the same as the one transmitted



7.4. Close the running plots after you finished analyzing them.

8. Phase shift Keying – Receive a message in GNU Radio

8.1. For this example, connect only one antenna to the RX1 of Jupiter as shown below. **Wait for the presenters to start the transmission.**



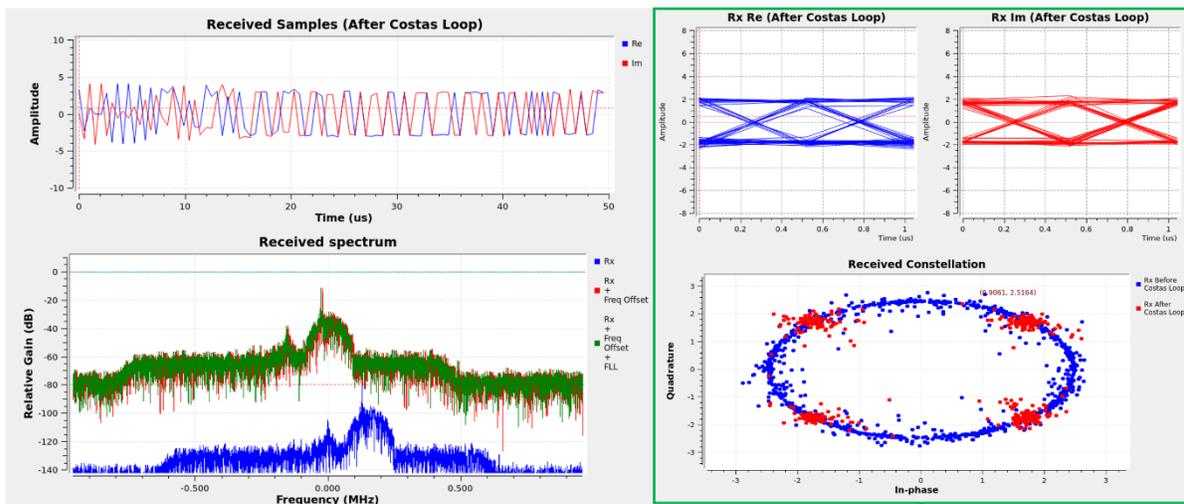
8.2. In GNU Radio companion app, open from "File" -> "Open" "console_message_receiver_jupiter.grc" from:

```
"/home/analog/Desktop/ftc_2024/6_console_message_point_to_point_gnuradio/receiver_jupiter"
```

8.3. Run the flowgraph and observe the received message in the 'msg_rx' section:

```
'msg_rx': 'ANALOG DEVICES INC FTC2024'
```

Observe that in this example, because the data is sent in bursts, the constellation is noisier because the digital feedback loops are not locked at the beginning of each burst received.



8.4. **Close** the running plots after you finished analyzing them.

9. Phase shift Keying – Receive a message and store it in a file

9.1. For this example, connect **only one antenna to the RX1** of Jupiter as shown below. **Wait for the presenters to start the transmission.**

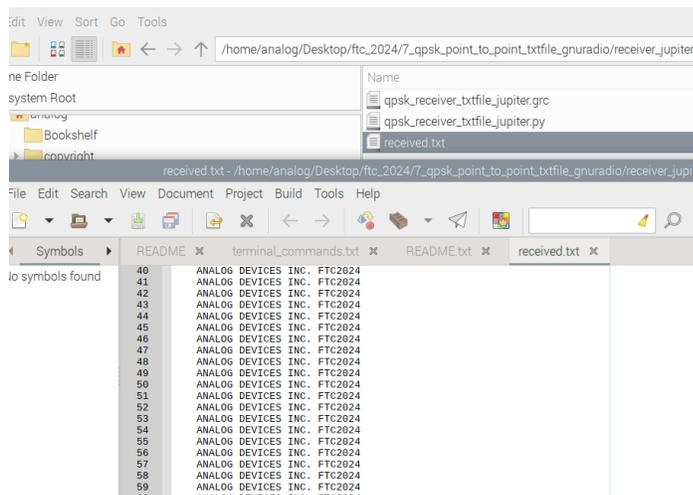


9.2. In GNU Radio companion app, open from **“File” -> “Open”** **“qpsk_receiver_txtfile_jupiter.grc”**:
`/home/analog/Desktop/ftc_2024/7_qpsk_point_to_point_txtfile_gnuradio/receiver_jupiter`

9.3. Run the example **for a few seconds** and **then stop it**. After you stopped the flowgraph, open **“receive.txt”** file from:

`“/home/analog/Desktop/ftc_2024/7_qpsk_point_to_point_txtfile_gnuradio/receiver_jupiter”`

You should see inside **“receive.txt”** multiple instances of the received message. You can delete that file and run again the flowgraph.



9.4. **Close** the **running flowgraph** and **“receive.txt”** file.

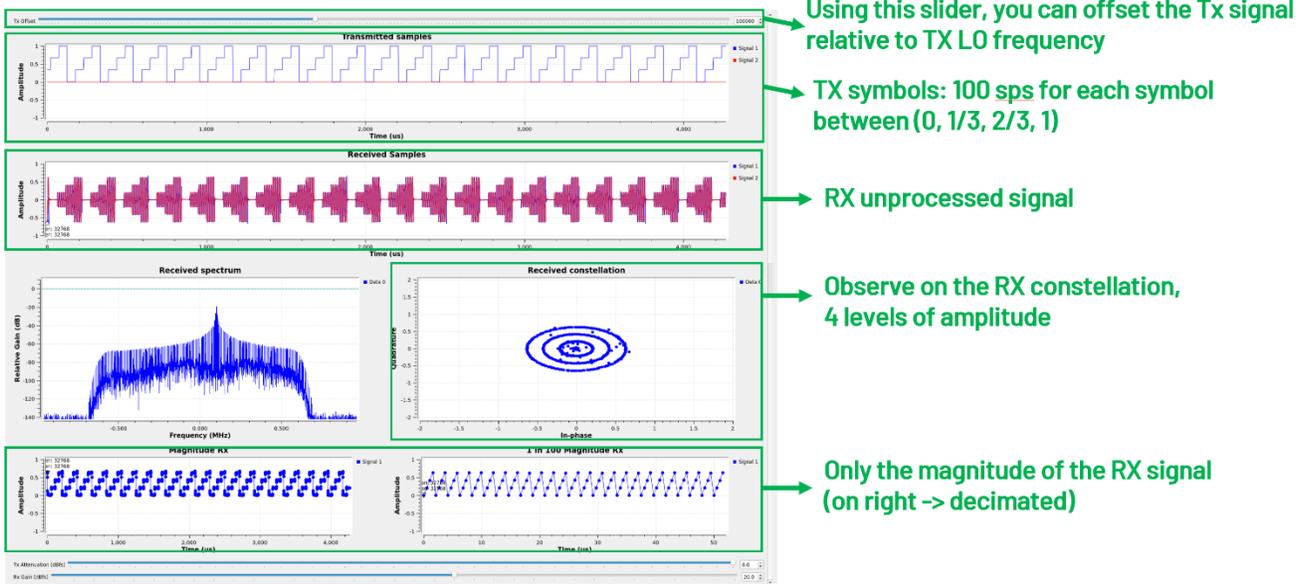
10. Amplitude Shift Keying – GNU Radio Example

10.1. For this example, connect Rx1 and Tx1 using the SMA cable from the kit, making a loopback between them (as shown below).



10.2. In GNU Radio companion app, open from "File" -> "Open", "ASK_loopback_jupiter.grc" from:
"/home/analog/Desktop/ftc_2024/8_ask_loopback_gnuradio"

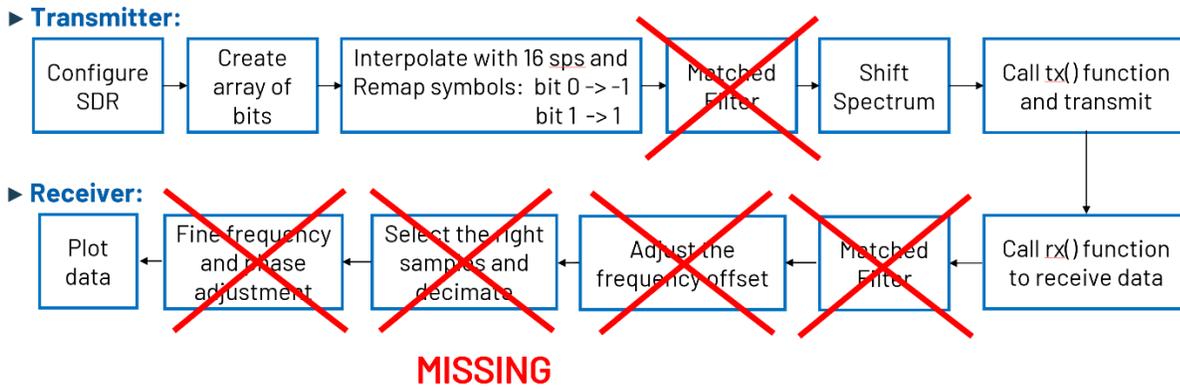
10.3. Run the flowgraph and observe the four different amplitudes of the received signal corresponding to the four symbols transmitted:



10.4. **Close** the running plots after you finished analyzing them.

11. BPSK without additional processing

In the following examples you will see why so many DSP blocks are required for receiving digital signals modulated with phase shift keying. All the DSP blocks required for synchronizing the received signal were removed and one at a time will be added back in the following examples.



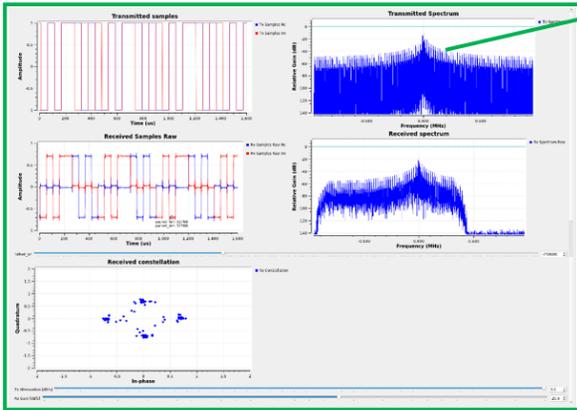
11.1. For this example, connect Rx1 and Tx1 using the SMA cable from the kit, making a loopback between them (as shown below).



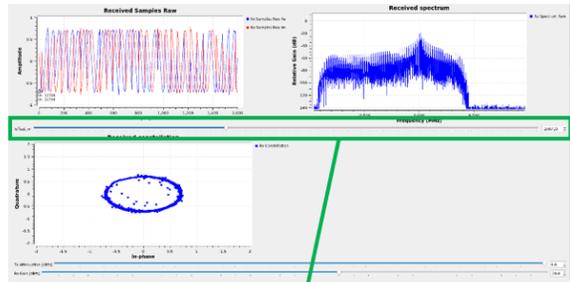
11.2. In GNU Radio companion app, open from File -> Open "[QPSK_raw_loopback_jupiter.grc](#)" flowgraph from:

`"/home/analog/Desktop/ftc_2024/10_qpsk_raw_loopback_gnuradio"`

11.3. Run the flowgraph and observe the following:



The spectrum is inefficiently used (spectrum of square wave)



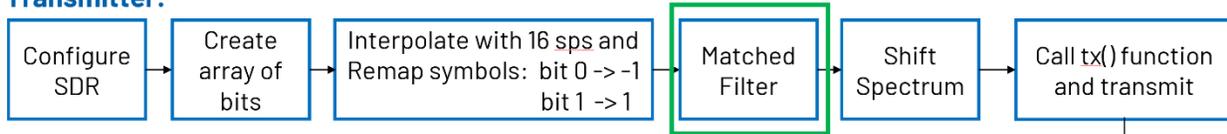
Try tweaking the frequency offset between TX and RX and see how the data looks -> this happens when we transmit and receive between two different devices.

No frequency offset because the example works on loopback and the LO is the same for RX and TX but has a phase offset between the two paths.

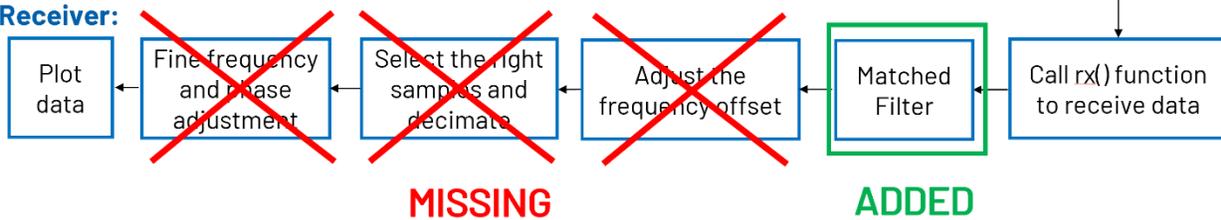
11.4. Close the running plots after you finished analyzing them.

12. QPSK – Constellation Modulator in GNU Radio

► Transmitter:



► Receiver:



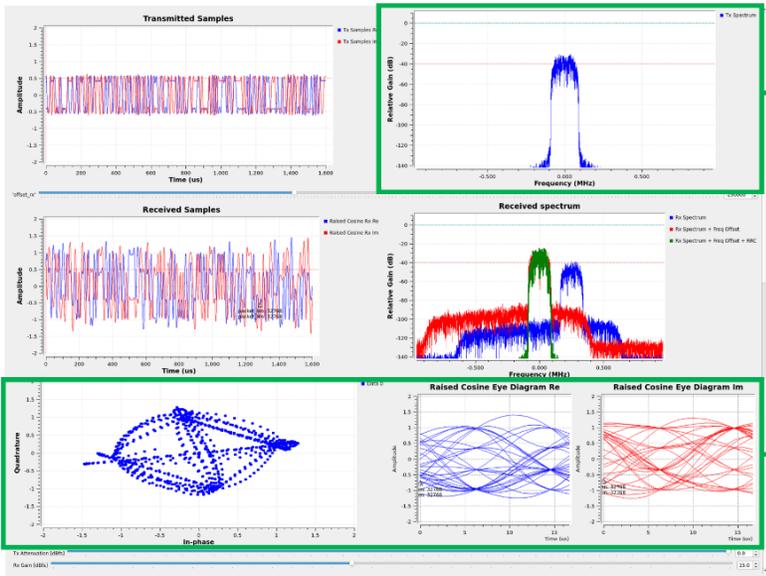
12.1. For this example, connect Rx1 and Tx1 using the SMA cable from the kit, making a loopback between them (as shown below).



12.2. In GNU Radio companion app, open from File -> Open
"constellation_modulator_loopback_jupiter.grc" flowgraph from:

"/home/analog/Desktop/ftc_2024/12_constellation_modulator_loopback_gnuradio"

12.3. Run the flowgraph and observe that the spectrum is now optimized by adding matched filters at TX and RX.



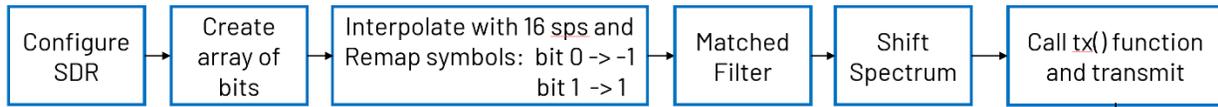
The BW is optimized due to matched Filters (one from Const. Modulator block at TX and one at the RX).

The RX signal is not decimated By selecting the right samples

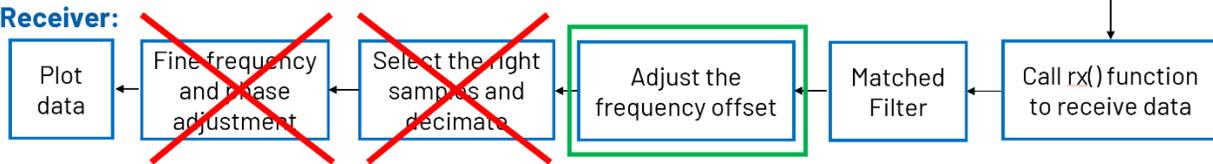
12.4. Close the running plots after you finished analyzing them.

13. QPSK- Frequency Locked Loop in GNU Radio

► Transmitter:



► Receiver:



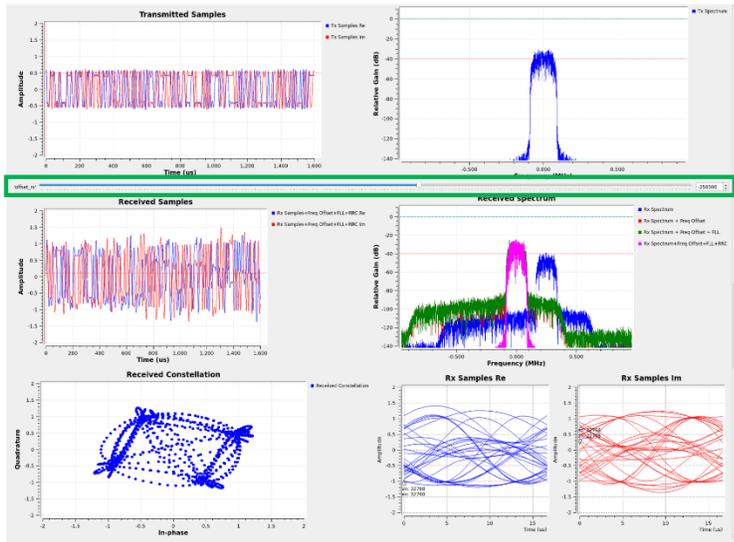
MISSING **ADDED**

13.1. Connect Rx1 and Tx1 using the SMA cable from the kit, making a loopback between them (as shown below).



13.2. In GNU Radio companion app, open from File -> Open, "[FLL_loopback_jupiter.grc](#)" from: `"/home/analog/Desktop/ftc_2024/13_fll_loopback_gnuradio"`

13.3. Run the flowgraph and observe how the spectrum after FLL block stays centered around DC at RX when you tweak the frequency offset slider:



Use this slider to tweak the frequency offset between TX and RX. Observe how the spectrum after FLL block stays centered around DC. A more precise frequency offset Correction still needs to be applied

13.4. Close the running plots after you finished analyzing them.

14. QPSK – Symbol Sync in GNU Radio

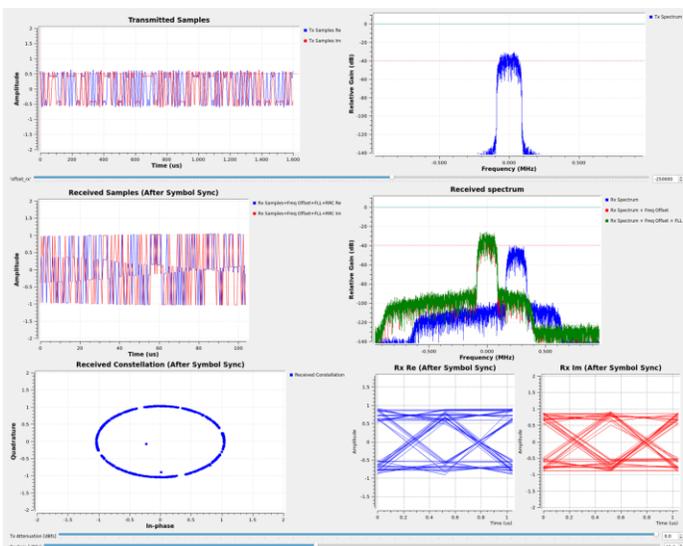
14.1. Connect Rx1 and Tx1 using the SMA cable from the kit, making a loopback between them (as shown below).



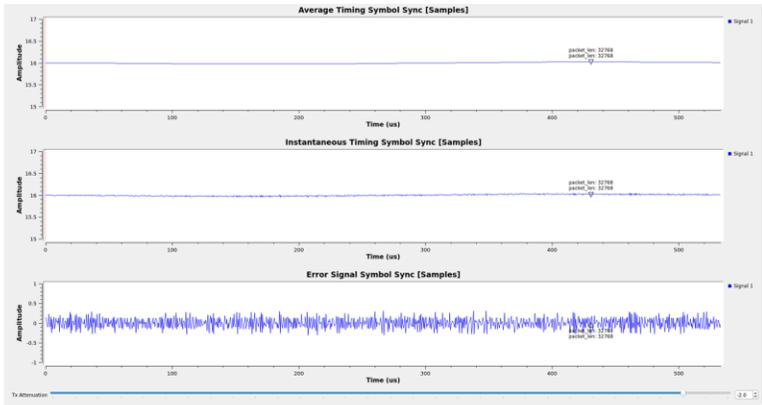
14.2. In GNU Radio companion app, open from File -> Open “[symbol_sync_loopback_jupiter.grc](#)” flowgraph from:

“/home/analog/Desktop/ftc_2024/14_symbol_sync_loopback_gnuradio”

14.3. Run the flowgraph and observe how all the symbols have the same amplitude in the constellation plot and observe that a remaining frequency and phase offset is still there because the FLL block cannot correct these with a very high precision. This block also applies a matched filter at RX.



Observe that now all the symbols have the same amplitude on the constellation plot but the imaginary and real parts of the data are still varying due to a remaining frequency and phase offset.

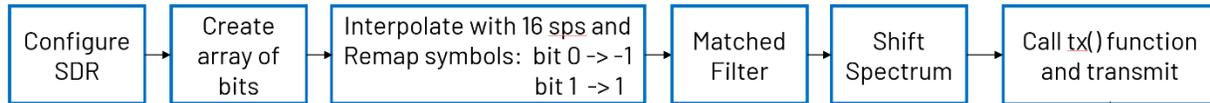


Here are plotted the main dynamic parameter of the symbol sync block. Observe that the timing stays around 16 = sps setting and the error stays around 0.

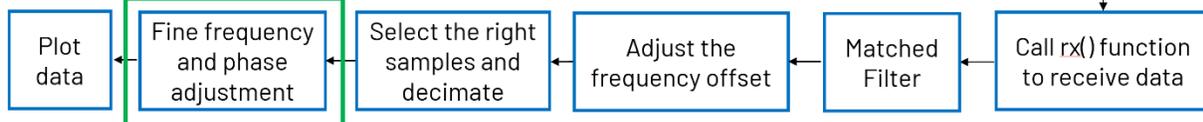
14.4. **Close** the running plots after you finished analyzing them.

15. QPSK – Costas Loop in GNU Radio

► Transmitter:



► Receiver:



ADDED

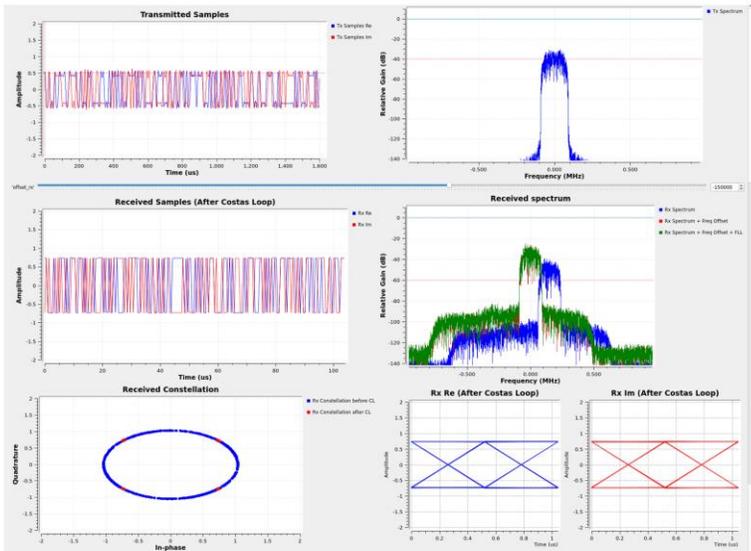
15.1. Connect Rx1 and Tx1 using the SMA cable from the kit, making a loopback between them (as shown below).



15.2. In GNU Radio companion app, open from File -> Open “[costas_loop_loopback_jupiter.grc](#)” flowgraph from:

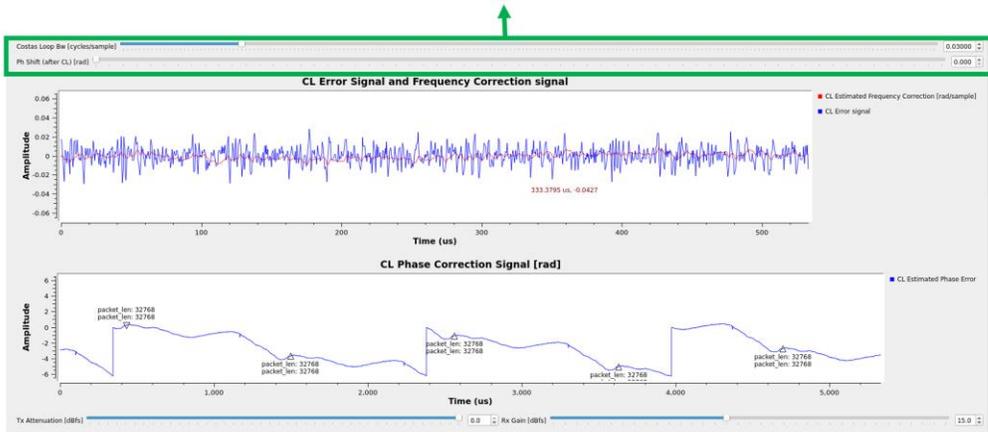
“/home/analog/Desktop/ftc_2024/15_costas_loop_loopback_gnuradio”

15.3. Run the flowgraph and observe how now the phase offset and frequency offset are removed completely.



Observe how the symbols are now separated as we wanted and the transitions in the Eye diagrams are less nosy.

Here you can tweak the bandwidth of the Costas Loop and see how it influences the error signal and the constellation plot. You can also add a phase shift in radians after Costas Loop.



Observe that the error signal is centered around 0 and the Phase Correction signal sits between 0 and 2π .

15.4. Close the running plots after you finished analyzing them.