ADI FTC

Software hands-on training kit: Customizing ADI Kuiper 2 Linux Distribution

Maria-Larisa Radu, Alisa Roman

Embedded Software Engineer, Associate Engineer

analog.com

Booklet

Contents

Pre-requisites

Common Linux commands:

- ***cd***: used to change the current working directory in Linux and other Unix-like operating systems
- ***ls***: lists files and directories within the file system, and shows detailed information about them
- ***cat***: displays the contents of one or multiple text files
- ***echo***: prints out its arguments as standard output; use > or >> with the *echo* command to print the output to a file instead of displaying it in the terminal
- ***mv***: moves a file from one location to another, rename a file with or without moving it

Workshop Preparation

Make sure the Raspberry Pi 5 in front of you is booted and is connected to the monitor, mouse, keyboard and WiFi.

The booklet and the .ppt presentation can also be found in the Kuiper image on Raspberry Pi 5 on the desktop named "booklet.pdf" and "powerpoint.pdf".

Glossary

Commands are color-coded according to the board they run on.

```
    This command is executed on the Raspberry Pi 5 Workstation.
```

```
    This command is executed on the CoraZ7s target board.
```

```
    This command is executed on the Jupiter SDR target board.
```

```
    This command is executed on the Raspberry Pi 4 target board.
```

Poweroff: before disconnecting a board from the power cable run on that image in the terminal the following command:

```
sudo poweroff
```

INFO: Each board includes optional exercises you can complete if you finish early. After you finish the exercises, before the board swap, make sure the poweroff the carrier boards (CoraZ7s, Jupiter SRD, Raspberry pi 4)

Build a Custom Kuiper 2 Image

In this exercise you will learn how to build a Kuiper 2 image that runs a custom script.

Steps:

1. Open a terminal, clone the adi-kuiper-gen repository, move inside it and start Visual Studio Code:

```
git clone https://github.com/analogdevicesinc/adi-kuiper-gen
cd adi-kuiper-gen
code .
```

2. Modify the *config* file in the project as follows:

Line in config	Old string	Modified string
Line 8	TARGET_ARCHITECTURE=armhf	TARGET_ARCHITECTURE=arm64
Line 23	CONFIG_DESKTOP=n	CONFIG_DESKTOP=y
Line 30	CONFIG_LIBIIO=n	CONFIG_LIBIIO=y
Line 226	EXTRA_SCRIPT=	EXTRA_SCRIPT=stages/07.extra-tweaks/01.extra-scripts/examples/extra-script-example.sh

Make sure you saved the file.

3. Create a new file at the location *stages/07.extra-tweaks/01.extra-scripts/examples/* named *custom-file.txt* and write to it “Hello FTC25” (or anything you want). You can to that by right-clicking on *examples* folder and pressing *Add new file*.
4. Open the file *extra-scripts-example.sh* from location *stages/07.extra-tweaks/01.extra-scripts/examples/* and add the following line at the end of the file:

```
cp stages/07.extra-tweaks/01.extra-scripts/examples/custom-file.txt /home/analog
```

Make sure to save the files. You can close now Visual Studio Code.

5. In the terminal you have opened start the build by running:

```
sudo bash build-docker.sh
```

Password: *analog*

Congratulations! You started your first Kuiper 2 custom build!

Do not close the terminal until the build is finished. For any other command you will need to open a new terminal.

CoraZ7S Exercises

For this workshop you will get: a coraZ7S board, an ethernet cable, a USB cable, and an ADXL355 pmod.

Prepare SD card

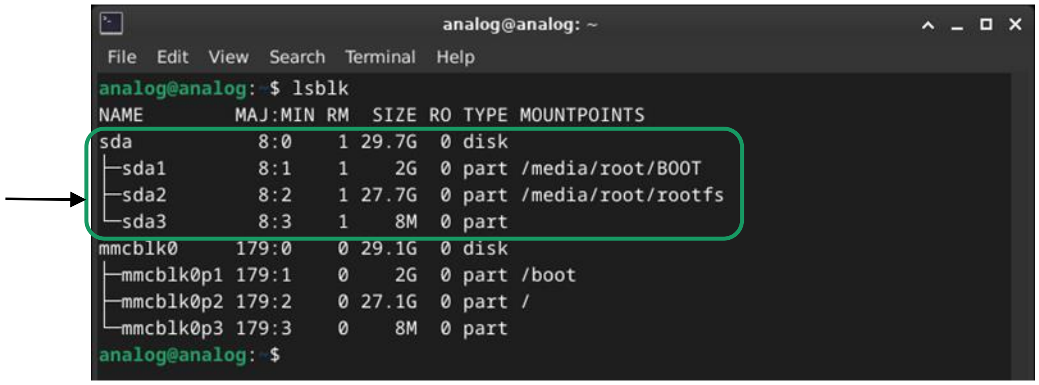
The first thing you must do is prepare the SD Card with CoraZ7s boot files.

Steps

1. Make sure the adapter is connected through USB to the workstation (RPi5).
2. Insert the SD Card you received with the CoraZ7s board into the SD slot on the adapter.
3. Run the following command to see the storage devices connected to the workstation (typically /dev/sda):

```
lsblk
```

Identify the SD card (verify its size and mountpoint).



```
analog@analog:~$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda           8:0    1 29.7G  0 disk
├─sda1        8:1    1    2G  0 part /media/root/BOOT
├─sda2        8:2    1 27.7G  0 part /media/root/rootfs
└─sda3        8:3    1    8M  0 part
mmcblk0     179:0    0 29.1G  0 disk
├─mmcblk0p1 179:1    0    2G  0 part /boot
├─mmcblk0p2 179:2    0 27.1G  0 part /
└─mmcblk0p3 179:3    0    8M  0 part
analog@analog:~$
```

4. Run the following command to list the available project boot files on the SD Card.

```
sudo configure-setup.sh -b /media/root/BOOT --help
```

Make sure you see the adxl355/coraz7s pair among the eval_board/carrier pairs. If missing, it’s possible you somehow got a wrong SD card.

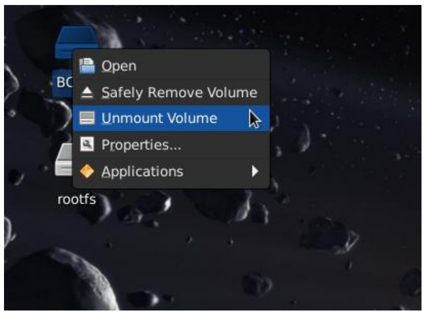
Now let’s run the script to prepare the boot files for our CoraZ7s carrier.

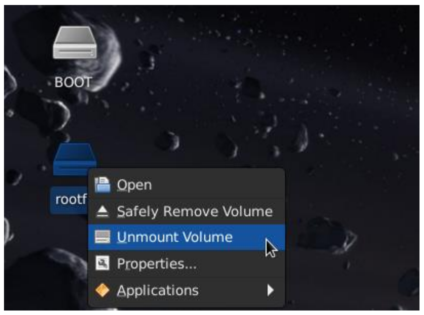
```
sudo configure-setup.sh -b /media/root/BOOT adxl355 coraz7s
```

If successful you should see the following

```
'/media/analog/BOOT/zynq-common/uImage' -> '/media/analog/BOOT/uImage'

'/media/analog/BOOT/ zynq-coraz7s-adxl355/BOOT.BIN' -> '/media/analog/BOOT/BOOT.BIN'

'/media/analog/BOOT/ zynq-coraz7s-adxl355/devicetree.dtb' ->
'/media/analog/BOOT/devicetree.dtb'

'/media/analog/BOOT/ zynq-coraz7s-adxl355//uEnv.txt' -> '/media/analog/BOOT/uEnv.txt'

Successfully prepared boot partition for running project adxl355 on coraz7s.
```

5. SAFELY unmount the SD card partitions (BOOT and rootfs) by right clicking each one and selecting unmount. The password is *analog*.

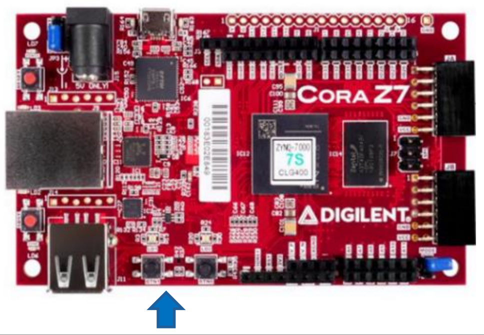
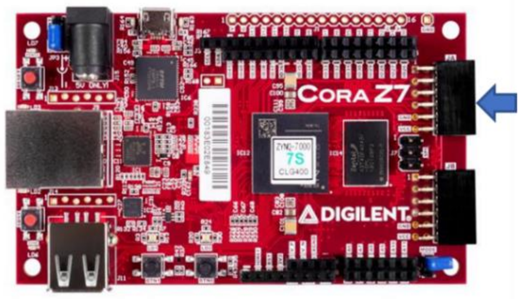
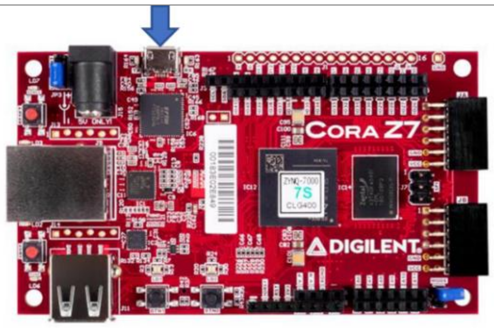
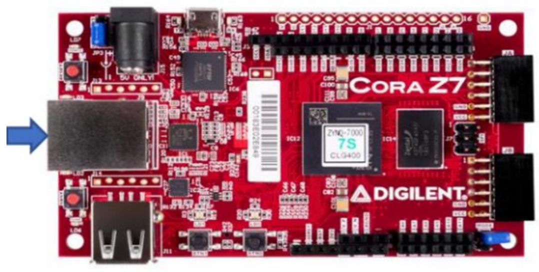




6. Extract the SD Card from the adapter.

Now the SD Card is prepared for CoraZ7s!

Hardware Setup

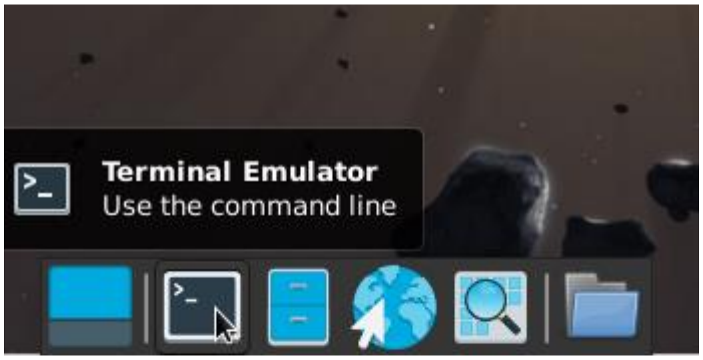
Insert the SD card prepared in the SD port of CoraZ7s.	
Connect the ADXL355 to the JA port of the CoraZ7s.	
Connect the UART port of the CoraZ7s to a USB port of your Raspberry Pi 5 using the USB cable received.	
Connect the Ethernet port of the CoraZ7s to the Ethernet port of your Raspberry Pi 5.	

By now, your CoraZ7s board should be up and running. It may take a few seconds for it to boot.

Exercise 1 – CoraZ7s Environment Exploration

Steps

1. On your Raspberry Pi 5, search for *Terminal* and open it:



2. Write the following command to discover the available IIO devices:

```
iio_info -s
```

Look for the IP address correspondent to *ftc25-cora* machine. You will need it for the next steps. To find it easier you can search by the hostname of the carrier:

```
iio_info -s | grep ftc25-cora
```

The output should look like this:

```
0: 169.254.35.68 (xadc) [ip:ftc25-cora.local]
```

3. Write the following command to connect via ssh to the CoraZ7s (replace the IP address in the command with the address of your board):

```
ssh -X analog@169.254.35.68
```

The password is *analog*.

At this point you should see this in your terminal:

```
analog@ftc25-pi5:~$ ssh root@169.254.35.68
root@169.254.35.68's password:
Linux analog 6.1.70-35308-ge2e62cc28c80 #1525 SMP Fri Aug 15 07:04:44 EEST 2025
armhf
```

```
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 Oct 24 19:11:04 2025 from 169.254.35.68
root@ftc25-cora:~#
```

You are now connected to the CoraZ7s. Every command you run from this point forward will be executed on the CoraZ7s, not your workstation.

Exercise 2 – Build a kernel object for ADXL355 driver

This exercise will demonstrate how to create a kernel object for an existing driver and load it in such a way that the system recognizes it as part of the kernel.
By running the command *iio_info* in the terminal you will see in the output that there is no *adxl355* device found.

Steps:

1. Move to the ftc25-cora folder:

```
cd ftc25-cora
```

2. Create a Makefile:

```
sudo touch Makefile
```

3. Open the Makefile with a file editor of your choice (nano, Vim, Code):

```
sudo nano Makefile
```

4. Write to the Makefile the following lines and then save and exit the file with CRTL+S, then CTRL+X:

```
obj-m := adxl355_spi.o

all:
	make -C /usr/src/linux-headers-6.1.0/ M=$(shell pwd) modules

clean:
	make -C /usr/src/linux-headers-6.1.0/ M=$(shell pwd) modules
```

Make sure to keep the indentation.

5. Run *make* to build the kernel object:

```
sudo make
```

6. Load the new kernel object:

```
sudo insmod adxl355_spi.ko
```

7. Check again with “*iio_info*” to see readings from the accelerometer.

```
iio_info
```

Exercise 3 – Play a Snake game with the accelerometer

Using the accelerometer module, you will have the chance to play a little oldie but goldie SNAKE game.

Steps:

1. Run the game controls in the background:

```
python3 game.py &
```

2. Start the Snake game:

```
/usr/games/nsnake
```

Move the accelerometer board around and observe the output. Be careful not to disconnect the wires.
Have fun!

Extra - Turn on an LED and make it blink with a heartbeat rhythm (Optional)

Steps

1. Switch to *root* user for this exercise:

```
su
```

Password: *analog*

2. Move to the LEDs directory using the command below:

```
cd /sys/class/leds
```

3. Here you can see 6 directories corresponding to the 3 colors of each RGB LED on the board:

```
ls
```

INFO: Each one of them contains the following attributes:

brightness	stores the status of the LED possible values: *0, 1*
trigger	changes the status of the LED
max_brightness	indicates the max brightness value that can be written to *brightness* attribute
device	a symbolic link pointing to the parent device of the LED

They may contain other folders and files, depending on the settings.

The important features of the *trigger* function are:

none	the default state where no specific trigger is set for the LED
timer	this trigger causes the LED to blink at a specified rate set in *delay_on* and *delay_off* attributes
oneshot	turns the LED on for a short period and then turns it off; the period is set in *oneshot*
heartbeat	causes the LED to blink in a pattern that mimics a heartbeat
cpu	links the LED activity to CPU activity

The *delay_on* and *delay_off* attributes are active when the timer features is enabled, and it will appear in the LED folder

4. Turn on the RED color of the first LED by setting the corresponding *brightness* to 1:

```
echo 1 >> led0_red/brightness
```

5. Make the red color of the first led blink in heartbeat mode:

```
echo heartbeat >> led0_red/trigger
```

This exercise will demonstrate how to execute a series of commands within a script, rather than running each command individually in the terminal. You will also learn how any Linux script can be turned into a bash command:

6. Now run a script that executes a series of commands that change the LEDs colors:

```
leds.sh
```

After you finish the exercises on CoraZ7s, please make sure to run "sudo poweroff" in the terminal connected to the target board.

Jupiter SDR Exercises

You received a Jupiter board with an SD Card, a power cable, one loopback cable and 2 antennas.

Prepare SD card

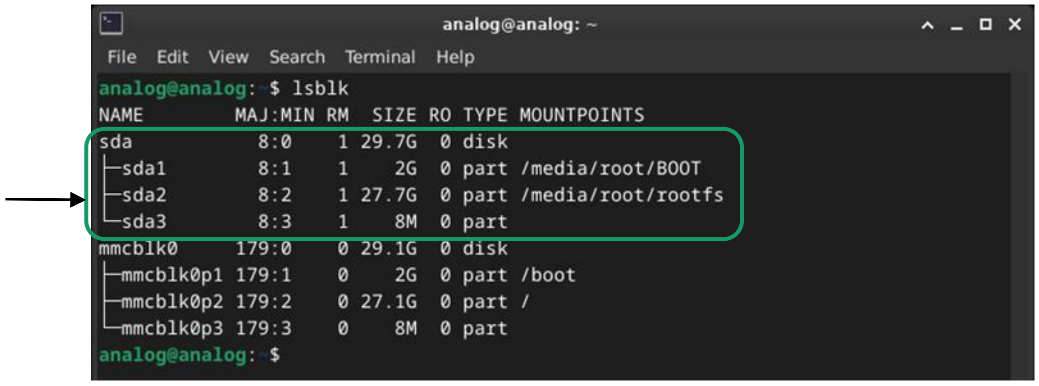
The first thing you must do is prepare the SD Card with Jupiter SDR boot files.

Steps

1. Make sure the adapter is connected through USB to the workstation (RPi5).
2. Insert the SD Card you received with the Jupiter SDR board into the SD slot on the adapter.
3. Run the following command to see the storage devices connected to the workstation (typically */dev/sda*):

```
lsblk
```

Identify the SD card (verify its size and mountpoint).



```
analog@analog:~$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda           8:0    1 29.7G  0 disk
├─sda1        8:1    1    2G  0 part /media/root/BOOT
├─sda2        8:2    1 27.7G  0 part /media/root/rootfs
└─sda3        8:3    1    8M  0 part
mmcblk0     179:0    0 29.1G  0 disk
├─mmcblk0p1 179:1    0    2G  0 part /boot
├─mmcblk0p2 179:2    0 27.1G  0 part /
└─mmcblk0p3 179:3    0    8M  0 part
analog@analog:~$
```

4. Run the following command to list the available project boot files on the SD Card.

```
sudo configure-setup.sh -b /media/root/BOOT --help
```

Make sure you see the jupiter_srd/jupiter_sdr pair among the eval_board/carrier pairs. If missing, it's possible you somehow got a wrong SD card.

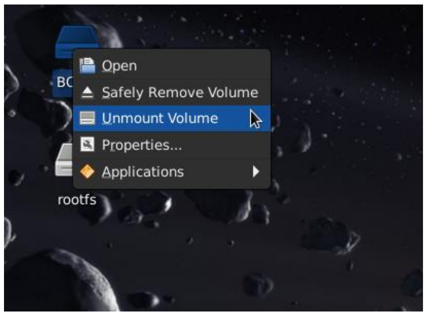
Now let's run the script to prepare the boot files for our Jupiter SDR carrier.

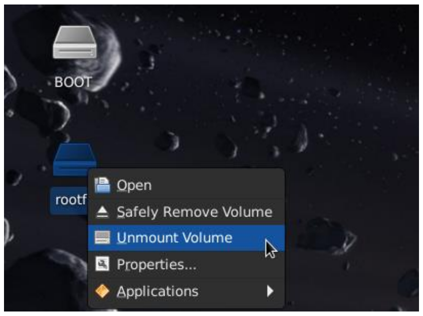
```
sudo configure-setup.sh -b /media/root/BOOT jupiter_sdr jupiter_sdr
```

If successful you should see the following

```
'/media/analog/BOOT/zynqmp-common/Image' -> '/media/analog/BOOT/Image'
'/media/analog/BOOT/zynqmp-jupiter-sdr/BOOT.BIN' -> '/media/analog/BOOT/BOOT.BIN'
'/media/analog/BOOT/zynqmp-jupiter-sdr/boot.scr' -> '/media/analog/BOOT/boot.scr'
'/media/analog/BOOT/zynqmp-jupiter-sdr/system.dtb' -> '/media/analog/BOOT/system.dtb'
'/media/analog/BOOT/zynqmp-common/uEnv.txt' -> '/media/analog/BOOT/uEnv.txt'
Successfully prepared boot partition for running project jupiter_sdr on jupiter_sdr.
```

5. SAFELY unmount the SD card partitions (BOOT and rootfs) by right clicking each one and selecting unmount. The password is *analog*.



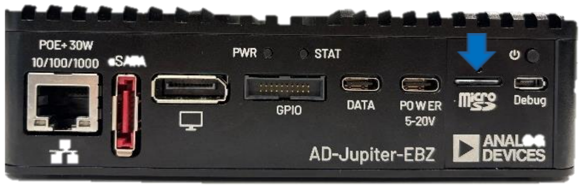


6. Extract the SD Card from the adapter.

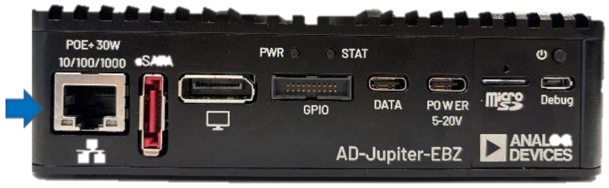
Now the SD Card is prepared for Jupiter SDR!

Hardware Setup

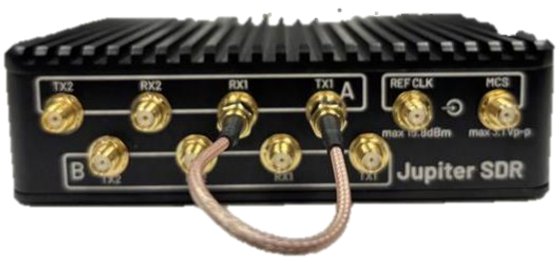
Insert the SD card prepared in the SD port of Jupiter SDR



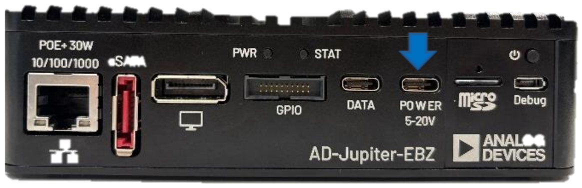
Connect the Ethernet port of the Jupiter SDR to the Ethernet port of your Raspberry Pi 5.



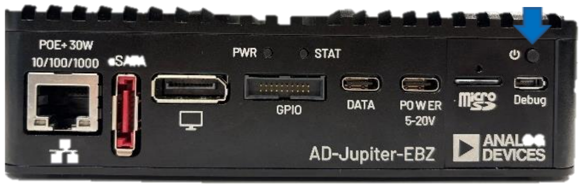
Connect the loopback cable to the Jupiter SDR to A TX1 and A RX1 like in the picture.



Connect the power cable you received to the power port of the Jupiter SDR



Make sure to press the power button.

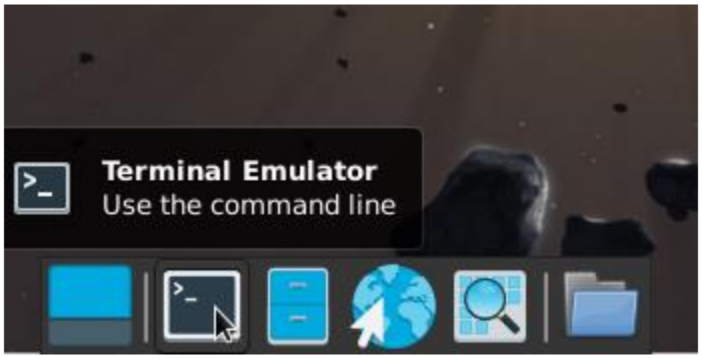


By now, your Jupiter SDR board should be up and running. It may take a few seconds for it to boot.

Exercise 1 – Jupiter SDR Environment Exploration

Steps

1. On your Raspberry Pi 5, search for *Terminal* and open it:



2. Write the following command to discover the available IIO devices:

```
iio_info -s
```

Look for the IP address correspondent to *ftc25-jupiter* machine. You will need it for the next steps. To find it easier you can search by the hostname of the carrier:

```
iio_info -s | grep ftc25-jupiter
```

The output should look like this:

```
0: 169.254.35.68
(ltc2945,tps6598x_source_psy_0_0038,tps6598x_source_psy_0_003f,xilinx-ams,adrv9002-
phy,axi-adrv9002-rx-lpc,axi-adrv9002-rx2-lpc,axi-adrv9002-tx-lpc,axi-adrv9002-tx2-
lpc) [ip:ftc25-jupiter.local]
```

3. Connect to the Jupiter SDR board as the **root** user using one of the following commands. Make sure to replace the IP address with the actual address of the board.

```
ssh root@169.254.35.68
```

The password is *analog*.

At this point you should see something like this in your terminal:

```
analog@ftc25-pi5:~$ ssh root@169.254.35.68
root@169.254.35.68's password:
Linux analog 6.1.70-35308-ge2e62cc28c80 #1525 SMP Fri Aug 15 07:04:44 EEST 2025
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 Oct 24 19:11:04 2025 from 169.254.35.68
root@ftc25-jupiter:~#
```

You are now connected to Jupiter SDR. Every command you run from this point forward will be executed on the Jupiter SDR, not your workstation.

Exercise 2 – Jupiter Configuration

Steps

1. You should be connected as the **root** user to Jupiter by the end of the previous exercise.
2. Write the Jupiter profile to configure the driver used to control the board to prepare for the following exercises

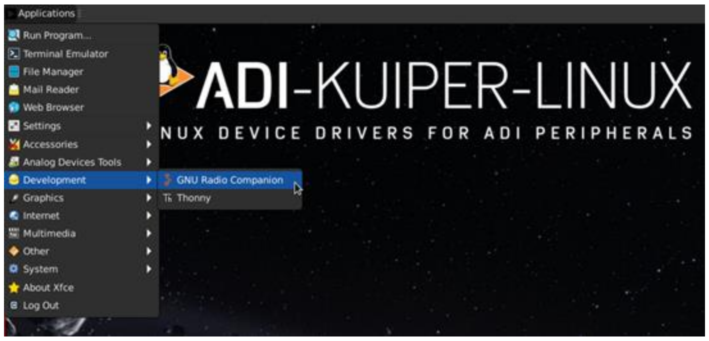
```
cat /home/analog/ftc25-
jupiter/jupiter_1_92MHz_profile.json > /sys/bus/iio/devices/iio:device1/profile_con
fig
```

Now you have the hardware and driver parts ready for the GNU Radio exercises!

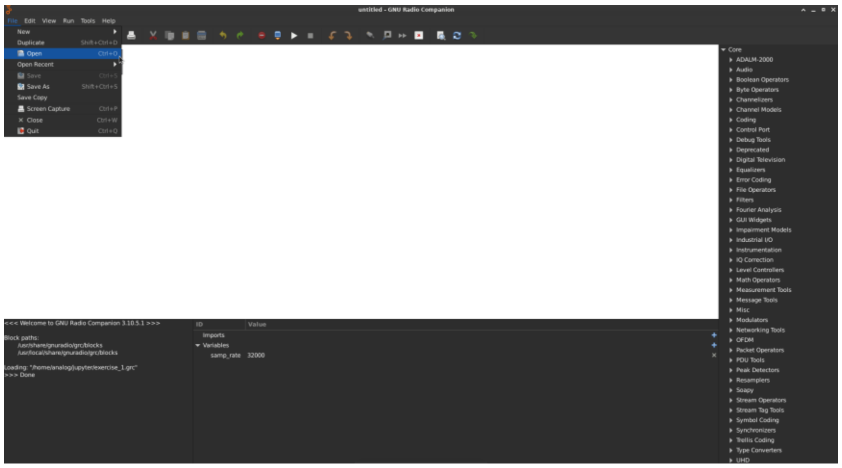
Exercise 3 – Loopback Complex Sinewave

Steps

1. Open GNU Radio Companion, the application used for the following exercises:

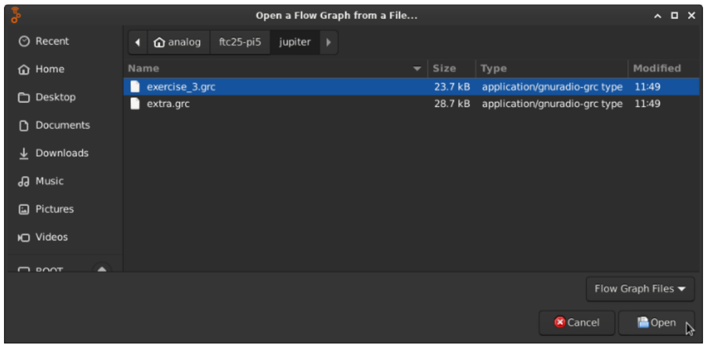


2. Open the exercise_3.grc file from */home/analog/ftc25-pi5/jupiter* folder



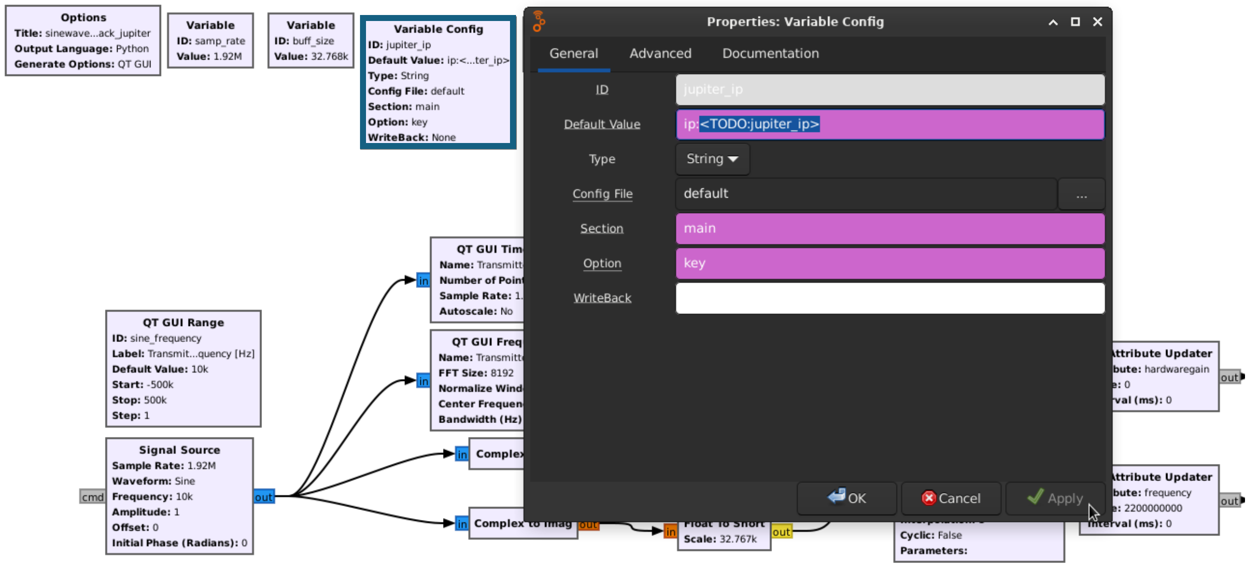
```
<<< Welcome to GNU Radio Companion 3.10.5.1 >>>

Block paths:
	/usr/share/gnuradio/grc/blocks
	/usr/local/share/gnuradio/grc/blocks

Loading: "/home/analog/jupyter/exercise_1.grc"
>>> Done
```

ID	Value
Imports	
Variables	
samp_rate	32000



Name	Size	Type	Modified
exercise_3.grc	23.7 kB	application/gnuradio-grc type	11:49
extra.grc	28.7 kB	application/gnuradio-grc type	11:49

3. Double-click on the Variable Config block and fill the placeholder IP variable with the actual IP address of the Jupiter board. If you forgot it, you could simply use the $ iio_info -s command again.



Options
Title: sinewave...ack_jupiter
Output Language: Python
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 1.92M

Variable
ID: buff_size
Value: 32.768k

Variable Config
ID: jupiter_ip
Default Value: ip:<...ter_ip>
Type: String
Config File: default
Section: main
Option: key
WriteBack: None

QT GUI Range
ID: sine_frequency
Label: Transmit...quency [Hz]
Default Value: 10k
Start: -500k
Stop: 500k
Step: 1

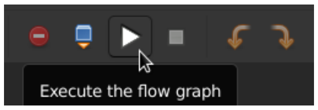
Signal Source
Sample Rate: 1.92M
Waveform: Sine
Frequency: 10k
Amplitude: 1
Offset: 0
Initial Phase (Radians): 0

Float To Short
Scale: 32.767k

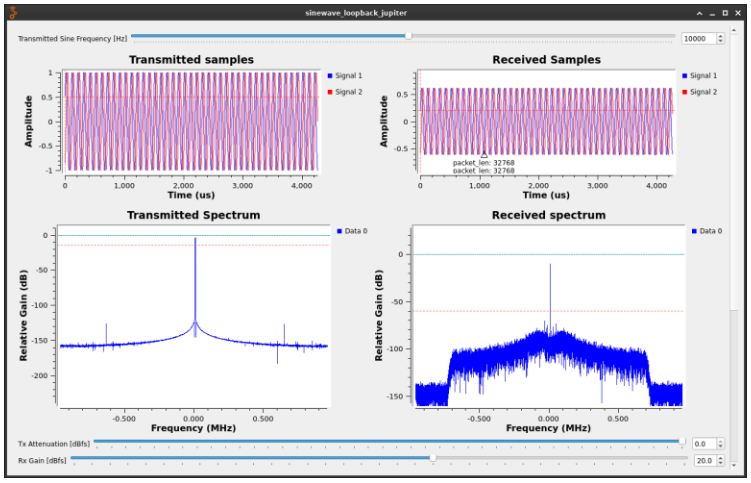
Properties: Variable Config

ID	jupiter_ip
Default Value	ip:<TODO:jupiter_ip>
Type	String
Config File	default
Section	main
Option	key
WriteBack	

4. Click the play button. It may take a few seconds for the visualization to open



5. You should see the following

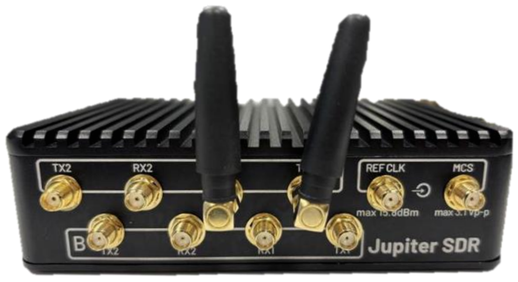


Congrats! You have a loopback transmission of a complex sinewave using your Jupiter board.

Bonus: You can try setting different frequencies using the slider at the top.

Extra – Doppler Effect (Optional)

Connect the antennas to the Jupiter SDR to A TX1 and A RX1 like in the picture.



Steps

1. Open the *extra.grc* file from *ftc25-pi5/jupiter* folder on desktop and fill in the board IP address. This is very similar to what was done in the previous exercise.
2. Press play and have fun with the Doppler effect visualization

After you finish the exercises on Jupiter SDR, please make sure to run "sudo poweroff" in the terminal connected to the target board.

Raspberry Pi 4 Exercises

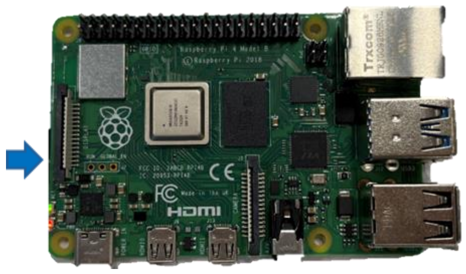
Prepare Setup

You received a Raspberry Pi 4 board, an SD Card and a power cable.

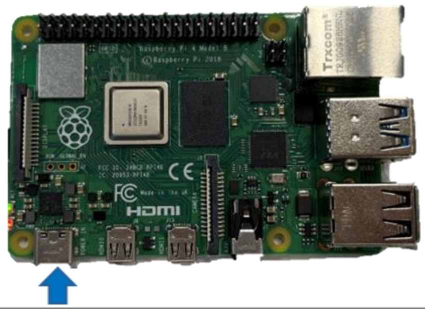
For this exercise, there is no need to configure the boot files for the board, since Kuiper runs on Raspberry Pi boards by default.

Hardware Setup

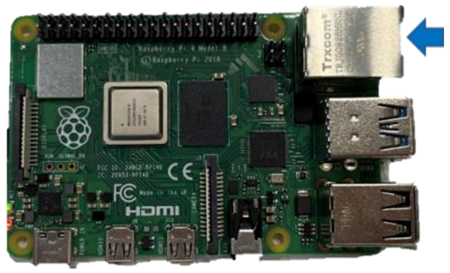
Make sure the Raspberry Pi 4 has an SD card in the SD port.



Connect the power cable you received to the power port of the Raspberry Pi 4



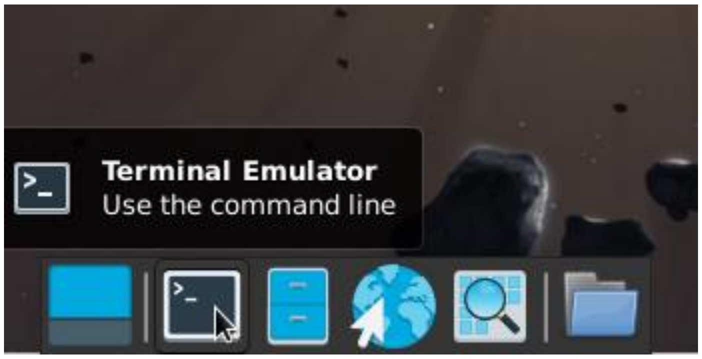
Connect the ethernet cable you received to the ethernet port of the Raspberry Pi 4



Exercise 1 – Raspberry Pi 4 Environment Exploration

Steps

1. On your Raspberry Pi 5, search for *Terminal* and open it:



2. Write the following command to discover the available IIO devices:

```
iio_info -s
```

Look for the IP address correspondent to *ftc25-pi4* machine. You will need it for the next steps. To find it more easily you can search by the hostname of the carrier:

```
iio_info -s | grep ftc25-pi4
```

The output should look like this:

```
0: 169.254.35.68 (cpu_termal,rpi_volt) [ip:ftc25-pi4.local]
```

Make sure you remember the IP address corresponding to the Raspberry Pi 4 board.

Exercise 2 – Monitor Workstation Temperature

Steps

1. Start the temperature monitoring python script

```
python3 ftc25-pi5/pi4/monitor_temperature.py
```

You should see now a thermometer that monitors the internal temperature of your RPi5 Workstation. Take your time to understand what is being displayed.

2. Open a new terminal instance like you did in the first step of Exercise 1.
3. In this new terminal use the following script to stress the CPU of the RPi5 and raise its temperature

```
python3 ftc25-pi5/pi4/stress.py
```

This script will stop by itself when a critical (>80 degrees Celsius) temperature is reached.

4. Go back to the initial terminal where the monitor_temperature.py script and observe the effect.
5. Press CTRL+C to stop the monitoring application.

Exercise 3 – Monitor Remotely RPi4 Temperature

Steps

1. Using one of the active terminals, open the monitor_temperature.py script in VS Code

```
code ftc25-pi5/pi4/monitor_temperature.py
```

2. Search for the following section

```
            #==================================================================
            # EXERCISE: Change this ONE line for remote monitoring!
            #==================================================================
            # For LOCAL monitoring (RPi5 monitors itself):
            #     ctx = iio.Context()
            #
            # For REMOTE monitoring (RPi5 monitors RPi4 at 169.254.35.68):
            #     ctx = iio.Context("ip:169.254.35.68")
            #==================================================================

            # Change the line below
            ctx = iio.Context()

            #==================================================================
```

Right now, the highlighted line of code uses the default local context, meaning the temperature sensor it monitors is the one on your Raspberry Pi 5 Workstation.

3. Replace `ctx = iio.Context()` with the following, making sure you use the actual IP address of your Raspberry Pi 4 target board, the one you discovered in Exercise 1. If you forgot it you can simply use the $ iio_info -s command once again.

```
ctx = iio.Context("ip:169.254.35.68")
```

4. Make sure you save the file and then close the VS Code editor.
5. In one of the active terminals start the monitoring script once again

```
python3 ftc25-pi5/pi4/monitor_temperature.py
```

Now the digital thermometer is monitoring remotely the temperature sensor on Raspberry Pi 4.

6. Press CTRL+C to stop the monitoring application.

Extra – Optional Challenge

Steps

1. Using one of the active terminals, start the monitor_temperature.py script

```
python3 ftc25-pi5/pi4/monitor_temperature.py
```

Let it run.

2. Using another terminal, copy the stress.py script to RPi4 using scp. Make sure you replace the IP address below with the actual IP address of the target board.

```
scp ftc25-pi5/pi4/stress.py analog@169.254.35.68:/home/analog
```

3. Connect to the target board using ssh. Don't forget to replace the IP address.

```
ssh analog@169.254.35.68
```

4. Start the stress script on the target board

```
python3 stress.py
```

5. Observe the effects on the digital thermometer running in the first terminal.
6. Press CTRL+C to stop the monitoring application.

After you finish the exercises on Raspberry Pi 4, please make sure to run "sudo poweroff" in a terminal connected to the target board.

Raspberry Pi 5 Networking Exercise

For this exercise you will communicate with other participants via a lightweight publish/subscribe protocol called MQTT.

Steps:

1. Open a terminal and run the following command to listen to a port where all participants will write messages:

```
mosquitto_sub -h test.mosquitto.org -t "ftc25-kuiper"
```

2. Open a new terminal and write the following command to send messages to the port everyone listens to:

```
mosquitto_pub -h test.mosquitto.org -t "ftc25-kuiper" -m "Hello from Pi $HOSTNAME"
```

Boot your Custom Kuiper 2 Image on RPi5

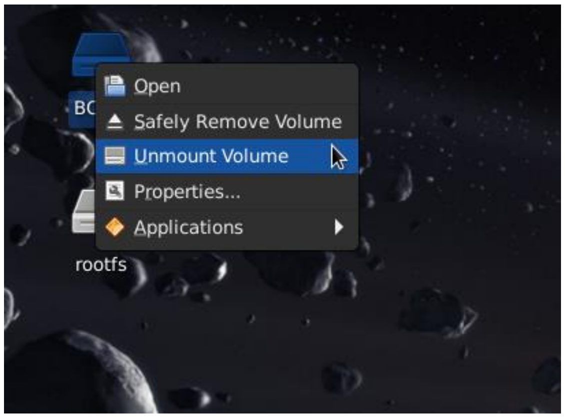
In this exercise you will write on an SD card the Kuiper 2 image you just built and will boot it on the Raspberry Pi 5 in front of you.

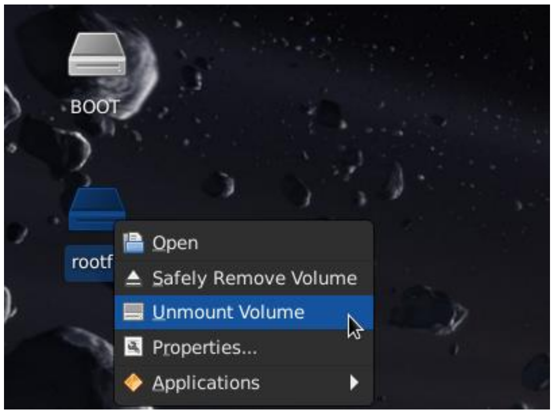
Steps:

1. Power-off the carrier board in front of you (CoraZ7s, Raspberry Pi 4 or Jupiter SDR), take the SD card from it and connect it to the Raspberry Pi 5.
2. Open a terminal in adi-kuiper-gen/kuiper-volume folder, unarchive and copy the Kuiper image on the SD card. Use lsblk command to check the partition. As you saw earlier, the SD card from the adapter is mounted at SDA.

```
lsblk
sudo unzip image_2025-11-10-ADI-Kuiper-Linux-arm64.zip
sudo dd if=image_2025-11-10-ADI-Kuiper-Linux-
arm64.img of=/dev/sda bs=1M status=progress conv=fsync
```

3. After the image is copied successfully, SAFELY unmount the SD card partitions (BOOT and rootfs) by right clicking each one and selecting unmount. The password is *analog*.





4. Extract the recently written SD card from the adapter.
5. Power off the Raspberry Pi 5 and disconnect it from the power cable, take out the SD card from it and replace it with the recently written card, and then connect back the Pi board to power.
6. Look after the file you copied with the extra script and check what it contains.

Congratulations! You booted your first Kuiper 2 custom image!