
sdr-setup-notes Documentation

HB9FXQ, Frank Werner-Krippendorf

Sep 02, 2019

Contents:

1	About this notes	1
1.1	Goals	1
1.2	Drawbacks	1
1.3	What else to explore?	2
1.4	Devices used in this Tutorial	2
2	Setup VM	3
2.1	Way A - Use my demo VM	3
2.2	Way B - Setup your own VM ← “Choose this way :-)”	3
2.3	Guest Additions	4
2.4	Way C - Native Linux Setup option	10
3	Setup Requirements	11
4	Hack RF one	13
4.1	Clone and build HackHF Repo:	13
4.2	Run	13
5	Native rtl_sdr Setup	15
5.1	Clone and build	15
5.2	Run	17
5.3	RTL_433 tool setup	17
6	GNU Radio Setup	21
6.1	The Pybombs way	21
6.2	Add GRC environment to user profile	22
6.3	Installing OOT Modules from source	22
6.4	Run GRC	23
6.5	Run GQRX	23
6.6	Use RTL-SDR with GQRX:	23
6.7	Building experimental stuff outside the prefix directory	23
7	Build from source	31
7.1	Run with an RTL-SDR stick	31
8	URH	33
8.1	Installation	33

8.2	Basic use	33
9	Baudline	35
9.1	Use case	35
9.2	Run	35
9.3	Basic use	35
10	Inspectrum, GRC Burst Tags / Great M. Ossmann stuff (Not running in VM)	39
10.1	Use case	39
10.2	libliquid-dev	39
10.3	Clone and build	39
10.4	Basic use	40
11	For LimeSDR owners	41
11.1	Clone and install	41
11.2	Run	41
12	# WIP: Native SDRplay source blocks	45
13	BETA gr-sdrplay	47

CHAPTER 1

About this notes



Fig. 1: Creative Commons Attribution-NonCommercial 4.0 International

Please consider a small donation to tip me a coffee: <https://paypal.me/hb9fxq>

Post to the USKA academy workshop (<https://www.uska.ch/2018/02/06/uska-academy-2018/>) around GNU Radio, a few peers reached out to me with questions about how to get started, without the LiveUSB system used in the workshop in Lucerne.

This tutorial might help - might give some directions how to setup a SDR-Linux environment. Please feel free to edit or pullrequest my Markdown notes on my [GitHub repository](<https://github.com/krippendorf/sdr-setup-notes>).

All steps are tested with best effort only for Linux mint 18.3 - no Support!

1.1 Goals

- Option 1) Create a VM to explore some SDR stuff under Linux (start here: [Setup VM](#))
- Option 2) Install SDR stuff on a bare metal PC (start here: [Setup requirements](#))
- Get some tools around RTL-SDR Dongles
- Get a up-to-date GNU Radio installation - from Source, not Distro packages
- Target Hardware UHD, RTL based sticks, PlutoSDR, HackRF One

1.2 Drawbacks

- GNU Radio setup can be a pain. It'll cost a bit time to install manually. Installation from source has some good advantages. This tutorial is about installation from source and not using a distribution's package manager.

- Performance in a VM is not as good as a native Linux install on bare metal. Cool graphic stuff might not work and do not expect super-high data rates when routing USB devices from the HOST to the GUEST.
- The setup takes some time, but gives you some good practice with linux setup procedures
- *I can not give any support if anything goes wrong*

1.3 What else to explore?

A curated list of some cool, SDR related tools to discover

- <https://www.rtl-sdr.com/big-list-rtl-sdr-supported-software/>

1.4 Devices used in this Tutorial

- <https://greatscottgadgets.com/hackrf/>
- <https://www.rtl-sdr.com/product/rtl-sdr-blog-v3-r820t2-rtl2832u-1ppm-tcxo-sma-software-defined-radio-dongle-only/>
- <http://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/adalm-pluto.html>
- <https://www.ettus.com/product/details/USRP-B200mini-i>

2.1 Way A - Use my demo VM

I've uploaded my result from this tutorial to Google Drive <https://drive.google.com/open?id=1Spth19iKjler56iSXPmO325akPrYKGfa>

You can simply import it with Virtualbox 5. it is not perfect - see it as a way to save some time. **Anyway, I'd recommend to go Way B) below to get some hands on with Linux.**

All SDR related stuff is in ~/wrk User / password in this sample VM is: sdr

2.1.1 Import to Virtualbox

First download and install Virtualbox and from <https://www.virtualbox.org/wiki/Downloads>

Important: Also install Oracle VM VirtualBox Extension Pack, since we want to use USB devices from within the guest machine

This works on Windows, Linux and OS X

Double click the downloaded ova file and follow the wizard:

2.2 Way B - Setup your own VM ← “Choose this way :-)”

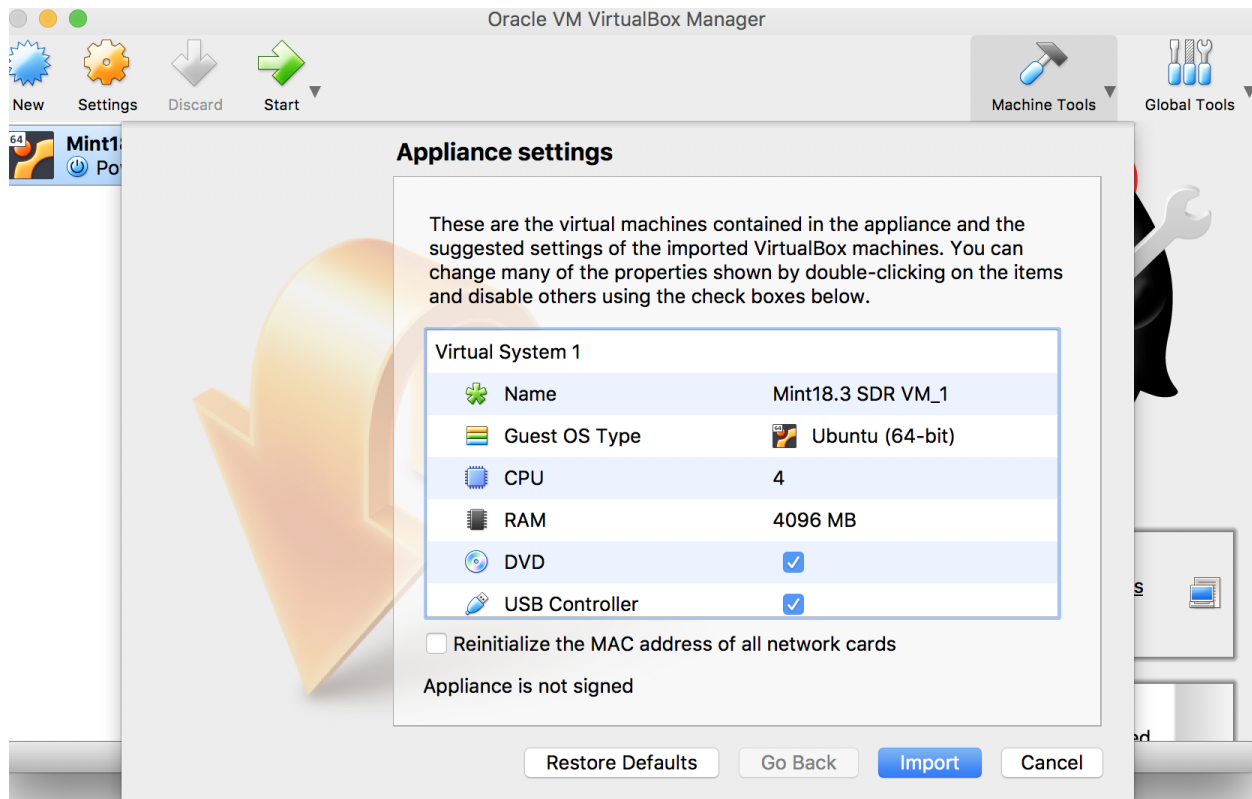
If you want to use a VM with virtualbox start here, otherwise, skip to step “**Native Linux Setup Option**”

Create a new VM in Virtualbox and tweak some settings:

Select Type “Linux” with Version “Ubuntu (64-bit)” If 64-bit is not in the list: Enable virtualization options in your system BIOS: <https://www.howtogeek.com/213795/how-to-enable-intel-vt-x-in-your-computers-bios-or-uefi-firmware/>

Create a virtual hard drive. around 50GB should be a good choice.

Adjust the memory to max within the green range of the slider:



Adjust the number of virtual processors to max within the green range of the slider and be sure to enable PAE/NX.

Be sure to enable VT-x/AMD-V Hardware virtualization!

Start the VM and provide the Linux Mint 64 Bit ISO, when asked. The ISO image is available from <https://www.linuxmint.com/edition.php?id=246> In this document user and password are set to 'sdr'. I've selected the option to install 3rd party applications during the setup.

2.3 Guest Additions

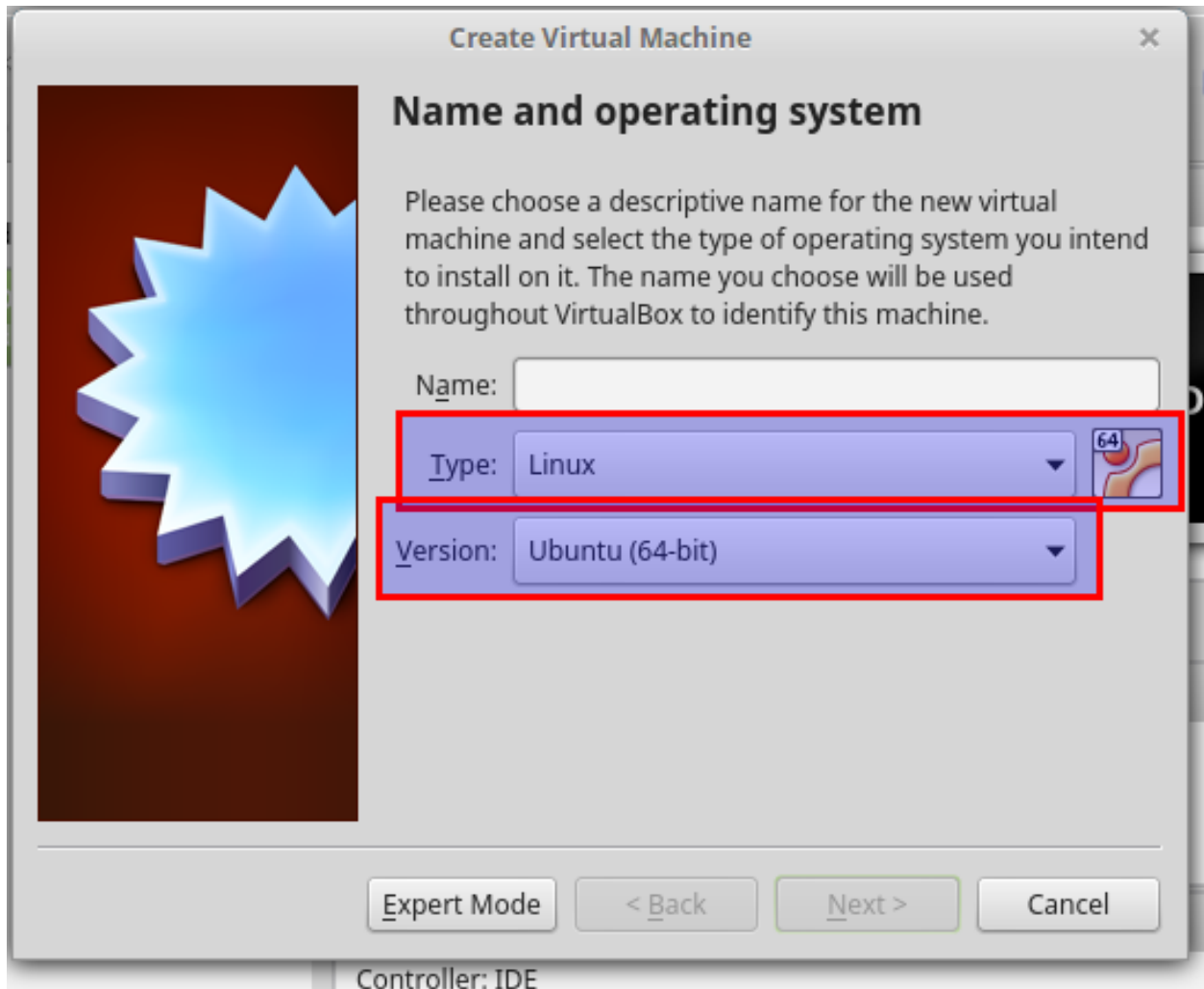
Complete the setup and after reboot install "Guest Additions":

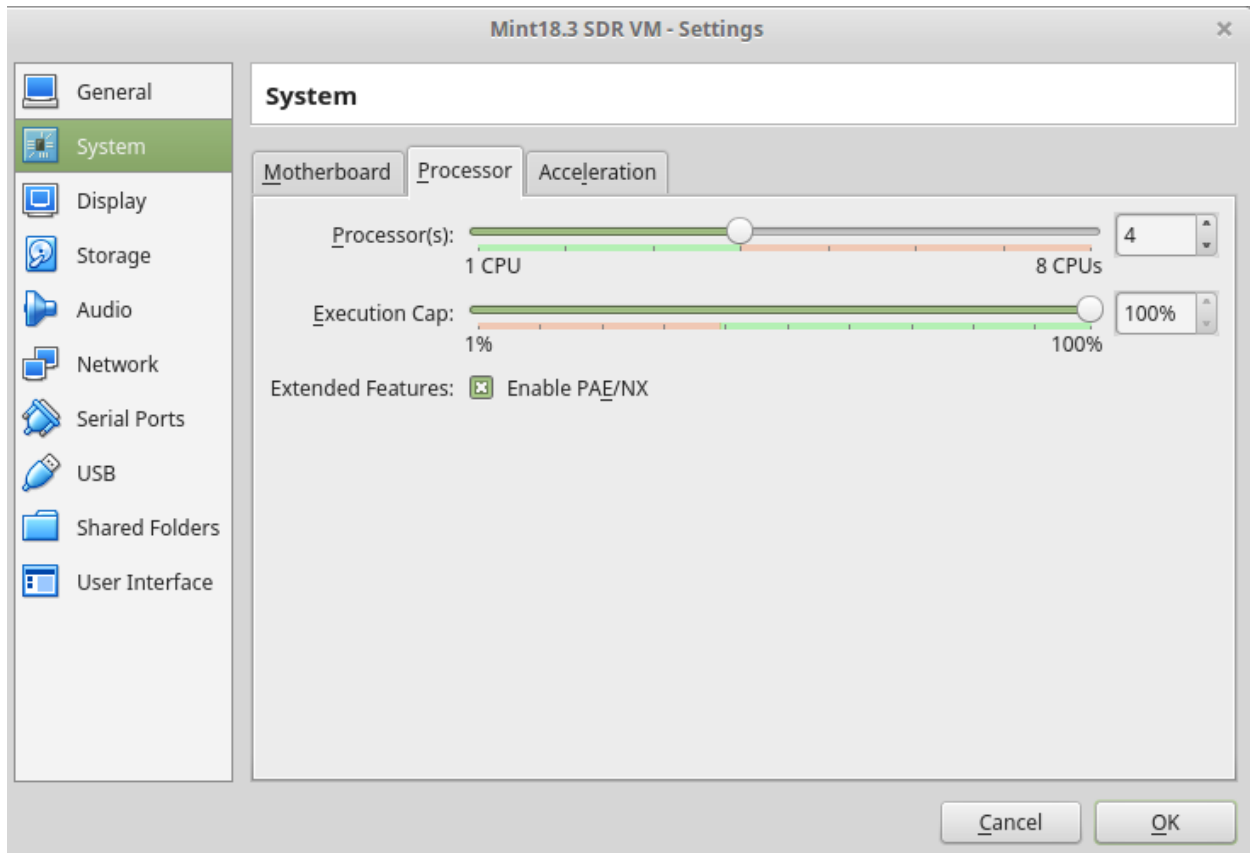
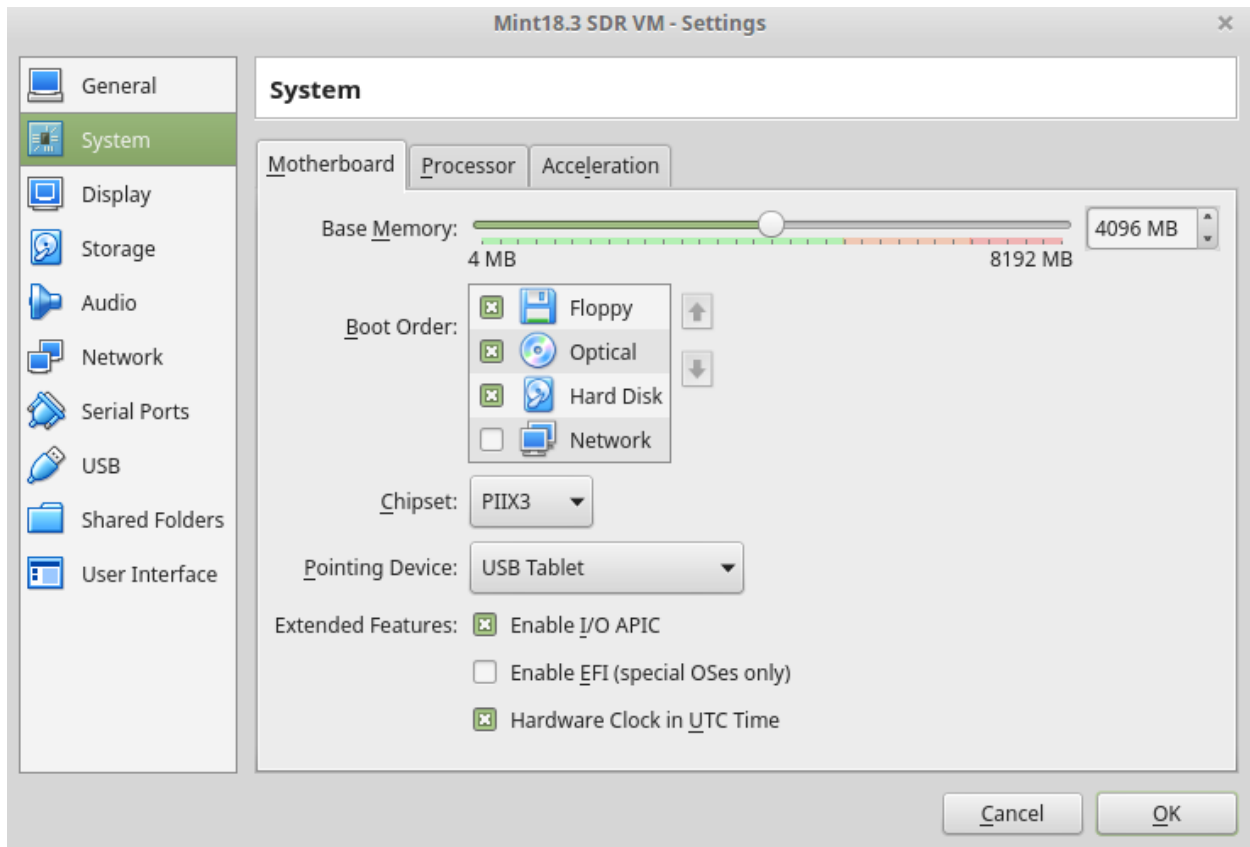
In VB Machine Window:

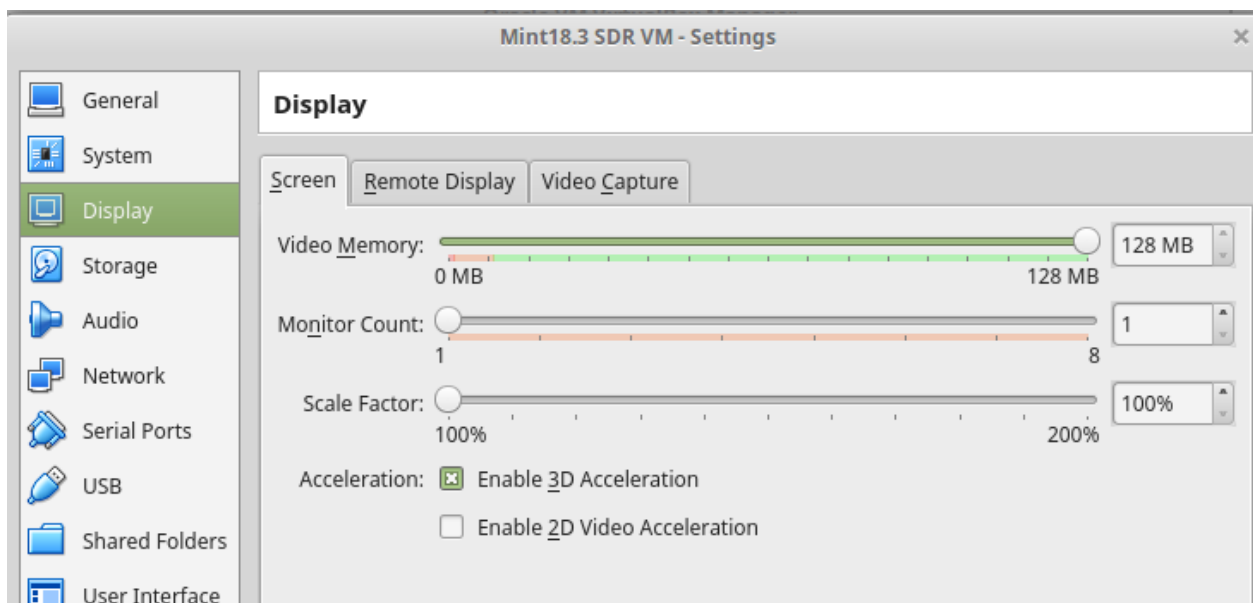
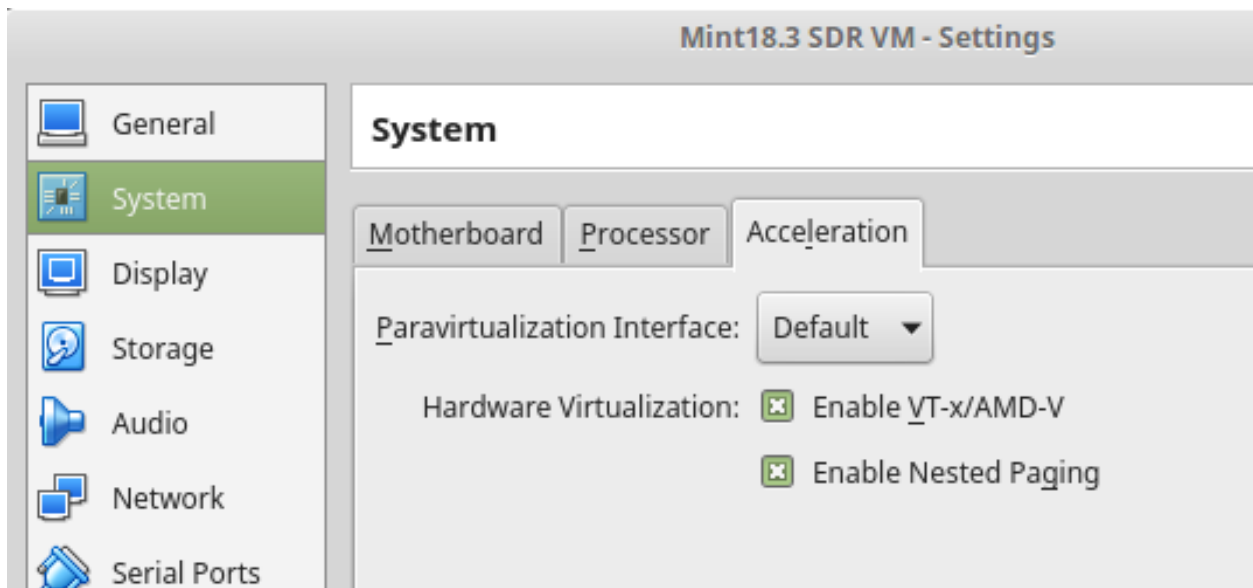
Menu **"Devices"->"Insert Guest Additions CD Image"** and follow the instructions.

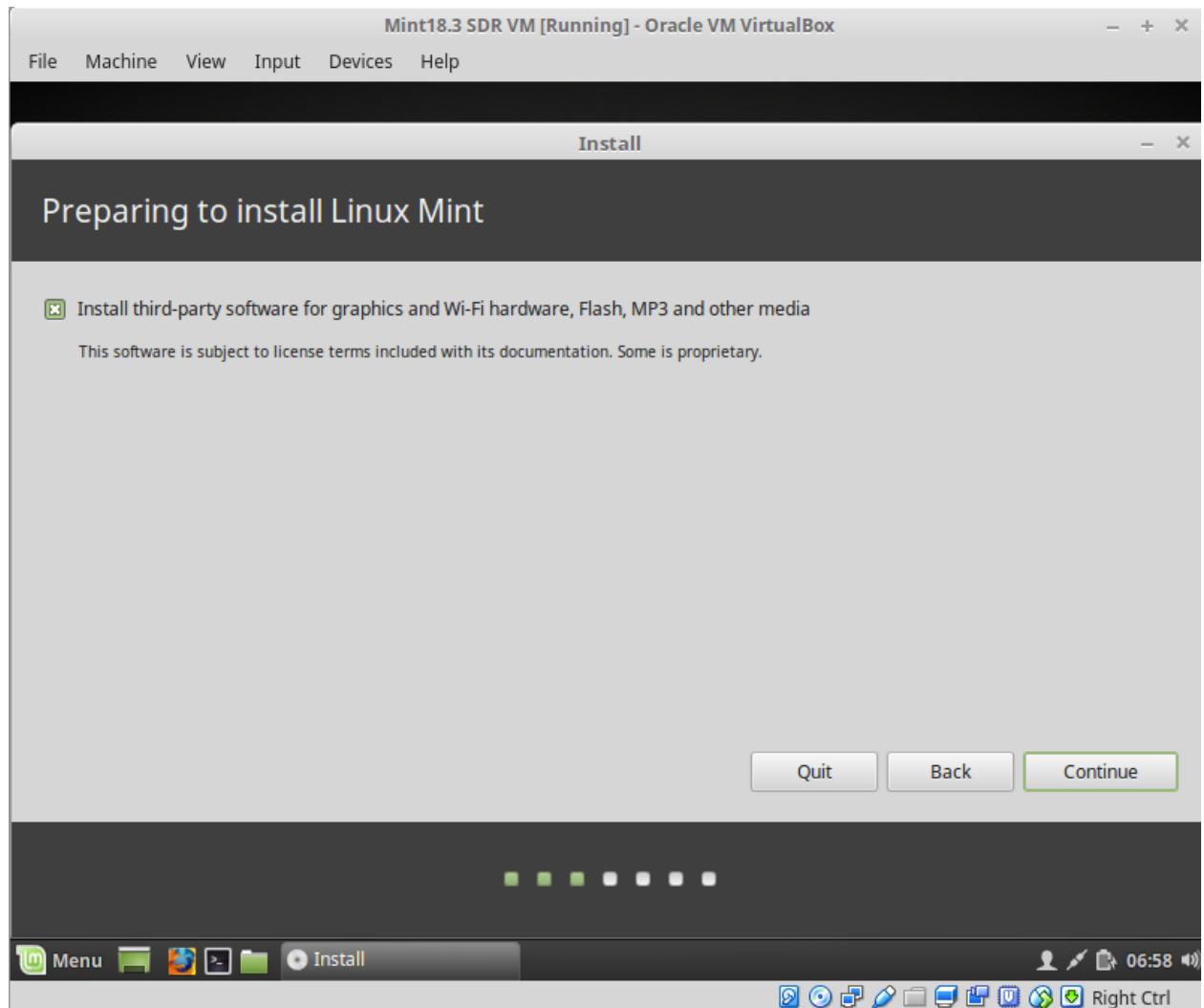
Open a Terminal window to add your user to the group vboxsf and dialout

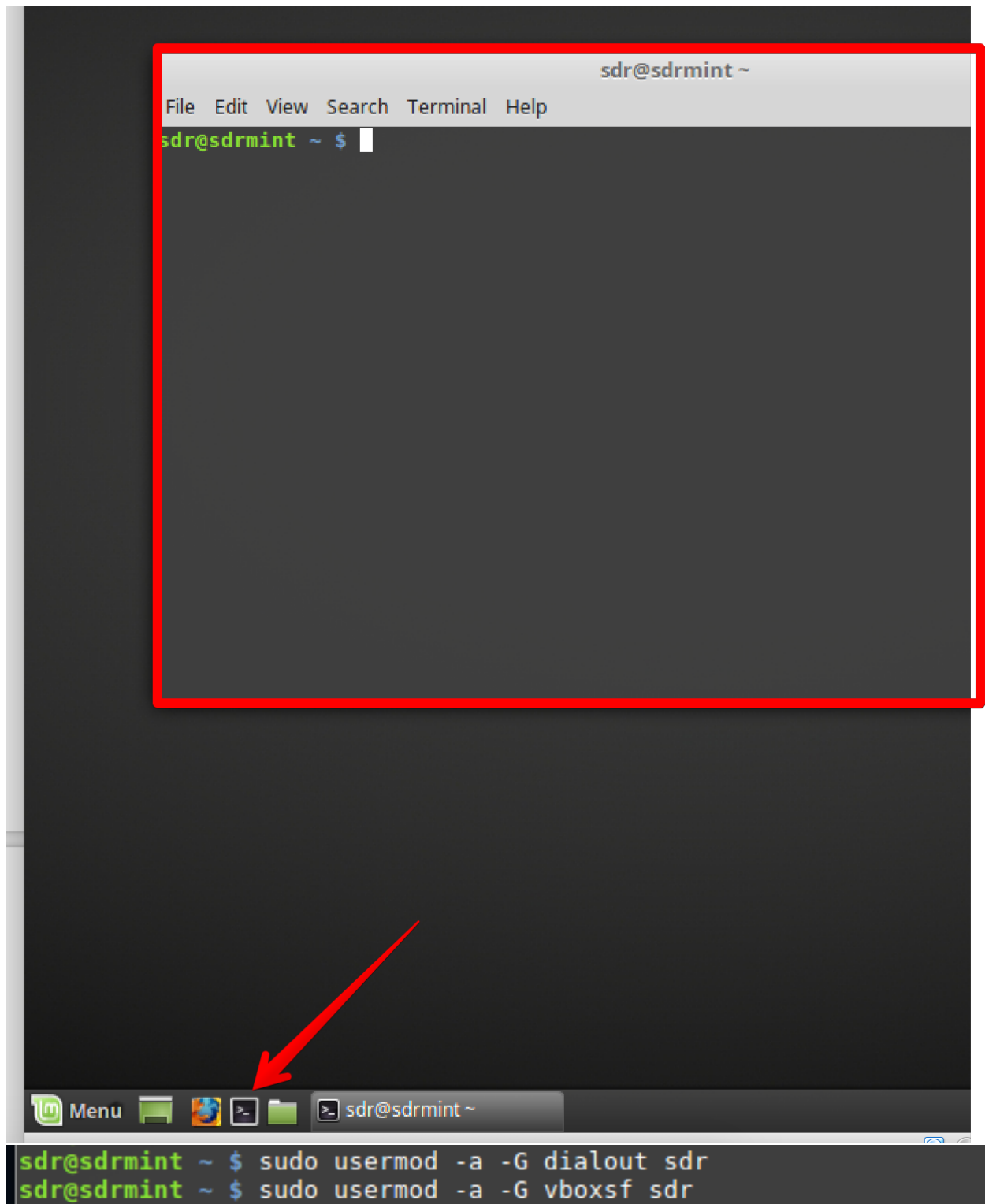
To open terminal you can use the small icon in the quick launch bar:











```
sudo usermod -a -G vboxsf sdr
sudo usermod -a -G dialout sdr
```

Then do a system update:

```
sudo apt-get update && sudo apt-get upgrade -y
```

```
sdr@sdrmint ~ $ sudo apt-get update && sudo apt-get upgrade -y
Hit:1 http://archive.canonical.com/ubuntu xenial InRelease
Hit:2 http://archive.ubuntu.com/ubuntu xenial InRelease
Hit:3 http://security.ubuntu.com/ubuntu xenial-security InRelease
Hit:4 http://archive.ubuntu.com/ubuntu xenial-updates InRelease
Hit:5 http://archive.ubuntu.com/ubuntu xenial-backports InRelease
Ign:6 http://packages.linuxmint.com sylvia InRelease
Hit:7 http://packages.linuxmint.com sylvia Release
Reading package lists... Done
Reading package lists... Done
Building dependency tree
```

Reboot the VM!

2.4 Way C - Native Linux Setup option

All following steps are valid for a native Linux Setup! Finding a good Linux compatible hardware can be hard. . . . An unsorted, random list of directions:

- <https://www.tuxedocomputers.com/>
- <https://www.techradar.com/news/best-linux-laptops-of-2018>
- <https://www.slant.co/topics/1184/~laptops-for-linux>
- <https://hblok.net/blog/posts/2017/03/21/linux-compatible-notebooks-and-laptops/>

From my experience most important is to have working graphics, touchpad and sound. Personally I swear on not to bleeding edge Lenovo aka. “IBM” or Dell hardware. . . this is not the topic of this tutorial, but keep in mind to focus on hardware specs to be well supported by Linux when looking for a new computer.

CHAPTER 3

Setup Requirements

All following steps work on either a native machine or within a VM.

Open a terminal and run:

```
sudo apt install -y build-essential cmake libusb-1.0-0-dev pkg-config libfftw3-dev  
→ htop curl wget git zsh python-pip virtualenv libtool autoconf pkg-config libxml2-  
→ dev vim ncdu libfftw3-dev
```

```
sudo pip install --upgrade pip
```

Still in the terminal create your work directory:

```
mkdir ~/wrk && cd ~/wrk
```

Note: ~ is a shortcut for /home/"username" under Linux.

4.1 Clone and build HackHF Repo:

```
git clone https://github.com/mossmann/hackrf.git
cd hackrf/host
mkdir build && cd build && cmake .. && make
sudo make install && sudo ldconfig
```

For details see <https://github.com/mossmann/hackrf>

4.2 Run

Connect the HackRF (Virtualbox Menu Devices->USB->GreatScot...) and test with: `hackrf_info`

Check if the device is responding:

```
sdr@sdrmint ~/wrk/hackrf/host/build $ hackrf_info
hackrf_info version: git-5e9cad6
libhackrf version: git-5e9cad6 (0.5)
Found HackRF
Index: 0
Serial number: 00000000000000000308066e623a9264b
Board ID Number: 2 (HackRF One)
Firmware Version: 2018.01.1 (API:1.02)
Part ID Number: 0xa000cb3c 0x00464f5e
sdr@sdrmint ~/wrk/hackrf/host/build $
```

Native rtl_sdr Setup

5.1 Clone and build

In your “wrk” Directory clone and build rtl_sdr

```
cd ~/wrk
git clone git://git.osmocom.org/rtl-sdr.git
cd rtl-sdr/
mkdir build
cd build
cmake ../ -DINSTALL_UDEV_RULES=ON
make
sudo make install
sudo ldconfig
sudo cp ../rtl-sdr.rules /etc/udev/rules.d/
```

Create further udev Blacklist file:

```
sudo nano /etc/modprobe.d/blacklist-rtl.conf
```

Append: blacklist dvb_usb_rtl28xxu to the file

To exit nano and save changes press CTRL-X, then type y, then enter to save and exit.

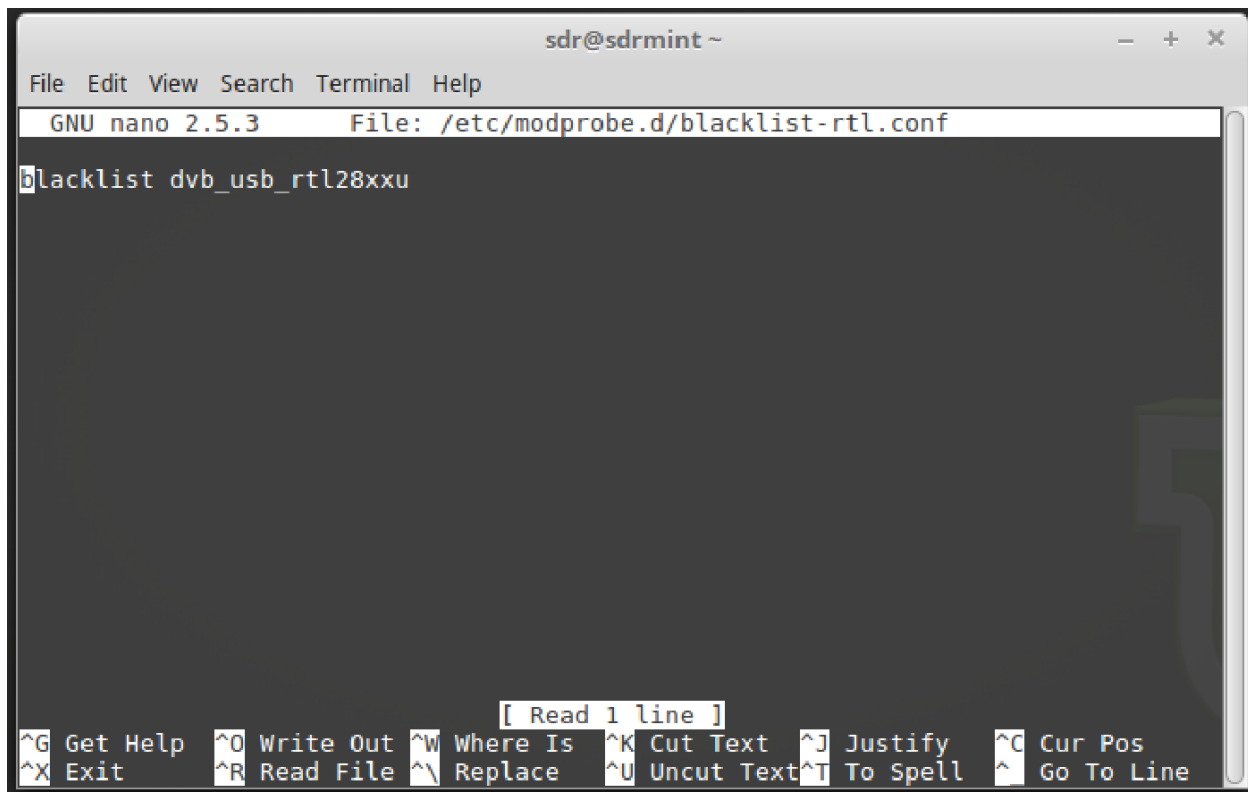
```
sudo udevadm control --reload-rules && udevadm trigger
```

Reboot the VM / Restart PC

Connect to your RTL Stick (**Virtual Box Menu Devices->USB-> Realtec RTL**) The name may depend on the manufacturer of your Stick

Open a terminal and enter

```
sdr@sdrmint ~/wrk $ git clone git://git.osmocom.org/rtl-sdr.git
Cloning into 'rtl-sdr'...
remote: Counting objects: 1740, done.
remote: Compressing objects: 100% (532/532), done.
remote: Total 1740 (delta 1252), reused 1621 (delta 1190)
Receiving objects: 100% (1740/1740), 382.68 KiB | 0 bytes/s, done.
Resolving deltas: 100% (1252/1252), done.
Checking connectivity... done.
sdr@sdrmint ~/wrk $ cd rtl-sdr/
sdr@sdrmint ~/wrk/rtl-sdr $ mkdir build
sdr@sdrmint ~/wrk/rtl-sdr $ cd build
sdr@sdrmint ~/wrk/rtl-sdr/build $ cmake ../ -DINSTALL_UDEV_RULES=ON
-- The C compiler identification is GNU 5.4.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Build type not specified: defaulting to release.
-- Extracting version information from git describe...
-- Found PkgConfig: /usr/bin/pkg-config (found version "0.29.1")
-- Checking for module 'libusb-1.0'
-- Found libusb-1.0, version 1.0.20
```



5.2 Run

rtl_test

check the output if it finds the stick:

```
pi@raspberrypi:~ $ rtl_test
Found 1 device(s):
 0: Realtek, RTL2838UHIDIR, SN: 00000001

Using device 0: Generic RTL2832U OEM
Found Rafael Micro R820T tuner
Supported gain values (29): 0.0 0.9 1.4 2.7 3.7 7.7 8.7 12.5 14.4 15.7 16.6 19.7
20.7 22.9 25.4 28.0 29.7 32.8 33.8 36.4 37.2 38.6 40.2 42.1 43.4 43.9 44.5 48.0
49.6
[R82XX] PLL not locked!
Sampling at 2048000 S/s.

Info: This tool will continuously read from the device, and report if
samples get lost. If you observe no further output, everything is fine.

Reading samples in async mode...
Allocating 15 zero-copy buffers
lost at least 68 bytes
█
```

press **ctrl+c** to cancel rtl_test.

For details see <https://osmocom.org/projects/rtl-sdr/wiki>

5.3 RTL_433 tool setup

in your “wrk” Directory clone and build rtl_433:

```
cd ~/wrk
git clone https://github.com/merbanan/rtl_433.git
cd rtl_433/ && mkdir build && cd build && cmake ../ && make
sudo make install
```

Start rtl_433 (RTL-SDR)

rtl_433

Even if you do not own your own temperature sensor you’ll get a good amount of measurements from your urban or non urban area.

More on https://github.com/merbanan/rtl_433

Special hint for all MQTT fanboys:

(mosquitto_pub can be installed by: `sudo apt install -y mosquitto-clients`)

`rtl_433 -F json -U | mosquitto_pub -t home/rtl_433 -l`

If mosquitto runs on another server append:

`-p port -u username, -P password -t topic -R 433MHz device number`

```
[ 99%] Building C object tests/characteristics/data-test.01/data-test.o
[100%] Linking C executable data-test
[100%] Built target data-test
sdr@sdrmint ~/wrk/rtl_433/build $ sudo make install
[ 97%] Built target rtl_433
[ 98%] Built target data
[100%] Built target data-test
Install the project...
-- Install configuration: "Release"
-- Installing: /usr/local/include/rtl_433.h
-- Installing: /usr/local/include/rtl_433_devices.h
-- Installing: /usr/local/bin/rtl_433
-- Set runtime path of "/usr/local/bin/rtl_433" to ""
sdr@sdrmint ~/wrk/rtl_433/build $ rtl_433
Registering protocol [1] "Rubicson Temperature Sensor"
Registering protocol [2] "Prologue Temperature Sensor"
Registering protocol [3] "Waveman Switch Transmitter"
Registering protocol [4] "LaCrosse TX Temperature / Humidity Sensor"
Registering protocol [5] "Acurite 609TXC Temperature and Humidity Se
Registering protocol [6] "Oregon Scientific Weather Sensor"
Registering protocol [7] "KlikAanKlikUit Wireless Switch"
Registering protocol [8] "AlectoV1 Weather Sensor (Alecto WS3500 WS4
Registering protocol [9] "Cardin S466-TX2"
```

```
Tuned to 433920000 Hz.
2018-06-24 09:54:13 : Nexus Temperature/Humidity
    House Code:      8
    Battery:         LOW
    Channel:         1
    Temperature:     21.10 C
    Humidity:        50 %
2018-06-24 09:55:10 : Nexus Temperature/Humidity
    House Code:      8
    Battery:         LOW
    Channel:         1
    Temperature:     21.10 C
    Humidity:        50 %
^C Signal caught, exiting!
sdr@sdrmint ~/wrk/rtl_433/build $
```

```
rtl_433 -F json -U -R 32 | mosquitto_pub -h 192.168.x.xxx -p 1883 -u admin -P admin -  
↪t home/rtl_433 -l
```

Will pipe the output to network as JSON formatted MQTT messages.

6.1 The Pybombs way

```
sudo pip install setuptools
sudo pip install git+git://github.com/gnuradio/pybombs.git
```

In your workdirectory ~/wrk generate a folder for pybombs to use as a prefix. This will install all sources and binaries into that prefix to make sure it'll be separated from the system library and bin paths. Resulting in a clean environment:

```
cd ~/wrk
mkdir grc_wrk
pybombs recipes add gr-recipes git+https://github.com/gnuradio/gr-recipes.git
pybombs prefix init -a default ~/wrk/grc_wrk/default/ -R gnuradio-default
```

Now it'll pull all required sources and build gnuradio companion. A few miles of output text will be generated on the screen and it'll take a good amount of time. So time to mess up the lab or work through your email inbox.

After setup completed do:

```
cd ~/wrk/grc_wrk/default
source ./setup_env.sh
```

This enables the GNU Radio environment installed with pybombs...

Let's add some more GR-* OOT Modules and GQRX

```
pybombs install gr-osmosdr
pybombs install gqrx
pybombs install gr-iio
sudo ldconfig
volk_profile
```

Volk profile will create a profile to compute FFT optimized for the system it runs on. It'll take quite a while, but will result in better graphics performance. The graphics performance in a VM is not good anyway. To use high-performance tools like gr-fosphor a native setup is required.

Only run the following 3 lines, if you use an USRP:

```
sudo cp ~/wrk/grc_wrk/default/lib/uhd/uhd-usrp.rules /etc/udev/rules.d/uhd-usrp.rules
sudo udevadm control --reload-rules
sudo udevadm trigger
```

Other OOT Modules I'd suggest to take a look at is:

- gr-paint <https://github.com/drmpeg/gr-paint>
- gr-gsm <https://github.com/ptrkrysik/gr-gsm> Ohhm, only use if you operate a cell network for development purposes!
- gr-ieee802-11 <https://github.com/bastibl/gr-ieee802-11> <https://github.com/bastibl/gr-ieee802-15-4>
- re-DECTed > <https://github.com/znuh/re-DECTed>
- rtty/psk31 stuff: > <https://github.com/bitglue/gr-radioteletype>
- Must-have: when not in a VM, using a native Linux PC with a good GPU: gr-fosphor <https://github.com/osmocom/gr-fosphor> (hint: when running on a recent PC hardware.... do a `sudo apt install libfreetype6-dev ocl-icd-opencl-dev python-opengl` before installing.)

Most of them can be installed using the default pybombs recipes...

6.2 Add GRC environment to user profile

In a terminal window run:

```
mkdir ~/bin
echo 'source ~/wrk/grc_wrk/default/setup_env.sh' >> ~/.profile
echo 'source ~/wrk/grc_wrk/default/setup_env.sh' >> ~/.bashrc
echo 'export PATH=~/wrk/bin:$PATH' >> ~/.profile
echo 'export PATH=~/wrk/bin:$PATH' >> ~/.bashrc
```

Reboot, or login/logout!

6.3 Installing OOT Modules from source

Since we've installed GNU Radio in it's own prefix directory custom OOT modules e.g. GR-LORA must be built towards that directory.

```
cd ~/wrk
```

```
git clone git://github.com/BastilleResearch/gr-lora.git
cd gr-lora
mkdir build && cd build
cmake ../
make && make install
sudo ldconfig
```

The cmake switch `-DCMAKE_INSTALL_PREFIX=~/wrk/grc_wrk/default` will ensure to target the right prefix. Keep this in mind for your further installations of OOT modules.

6.4 Run GRC

type command: `gnuradio-companion`

GNU Radio Companion (ready for UHD, RTL-SDR, ADALM Pluto, RedPitaya, Aircspy)

Note: We've installed all GNU Radio stuff in the prefix path. Do not try to mix stuff with packages installed from the distribution's Package manager... Do not simply install via "sudo apt install gnuradio" ... etc... since this will mix up things. Be careful when installing third party PPAs to not mess up your setup.

6.5 Run GQRX

You can now run the following tools *from a terminal!*

gqrX (UHD, RTL-SDR, RedPitaya, Aircspy)

command: `gqrX`

6.6 Use RTL-SDR with GQRX:

Start GQRX by typing:

"gqrX" in a terminal

Select your stick and set an appropriate input sample rate.

Try to find a good gain value for the signal of interest by finding the best distance between the noise floor and the signal peak

Experiment a bit with the FFT settings to get the best out the visualization

6.7 Building experimental stuff outside the prefix directory

When compiling libs from source, like special forks of GR-OSMOSDR etc... remember to adjust the target paths like shown with gr-lora above! If you don't want to set your well working GNU Radio prefix directory at the risk of getting messed up, you can compile and install experimental stuff towards a total different target location.

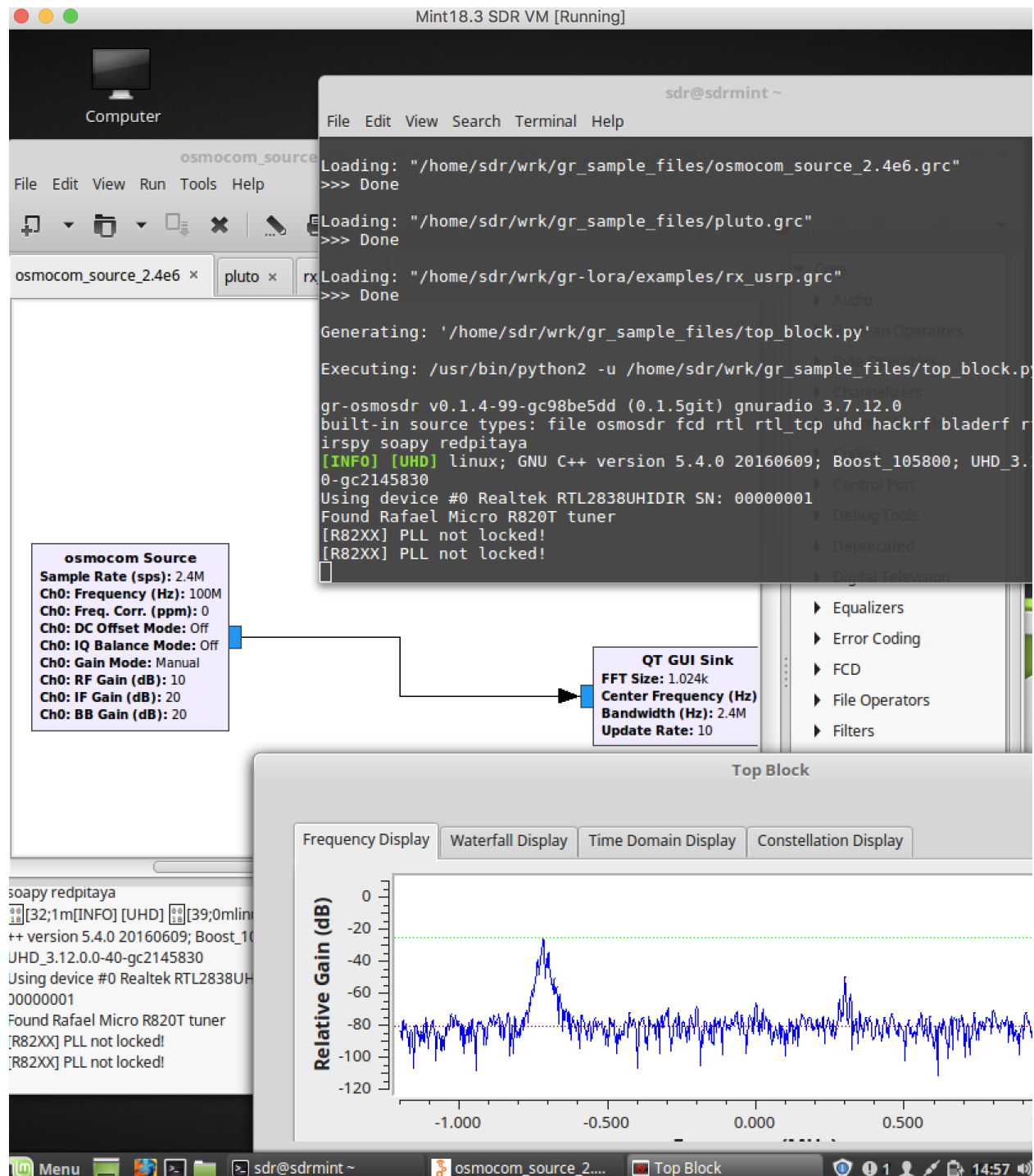
Let's say you got a brand new SDRPlay and need the non-standard gr-osmosdr source, from a 3rd party source repo to go for a test drive.

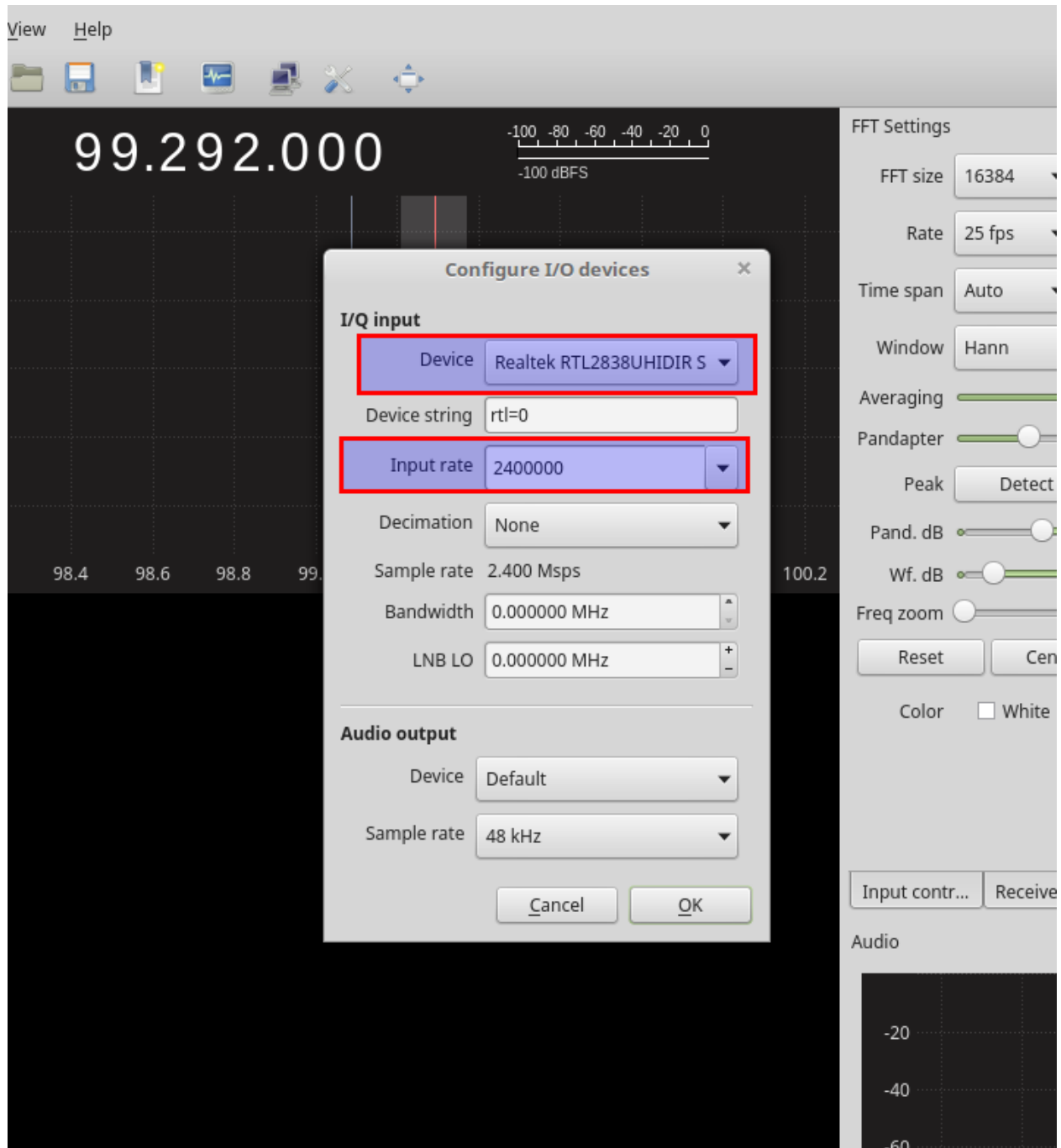
For SDRplay, first install the proprietary driver system wide. Download the API/HW Driver.

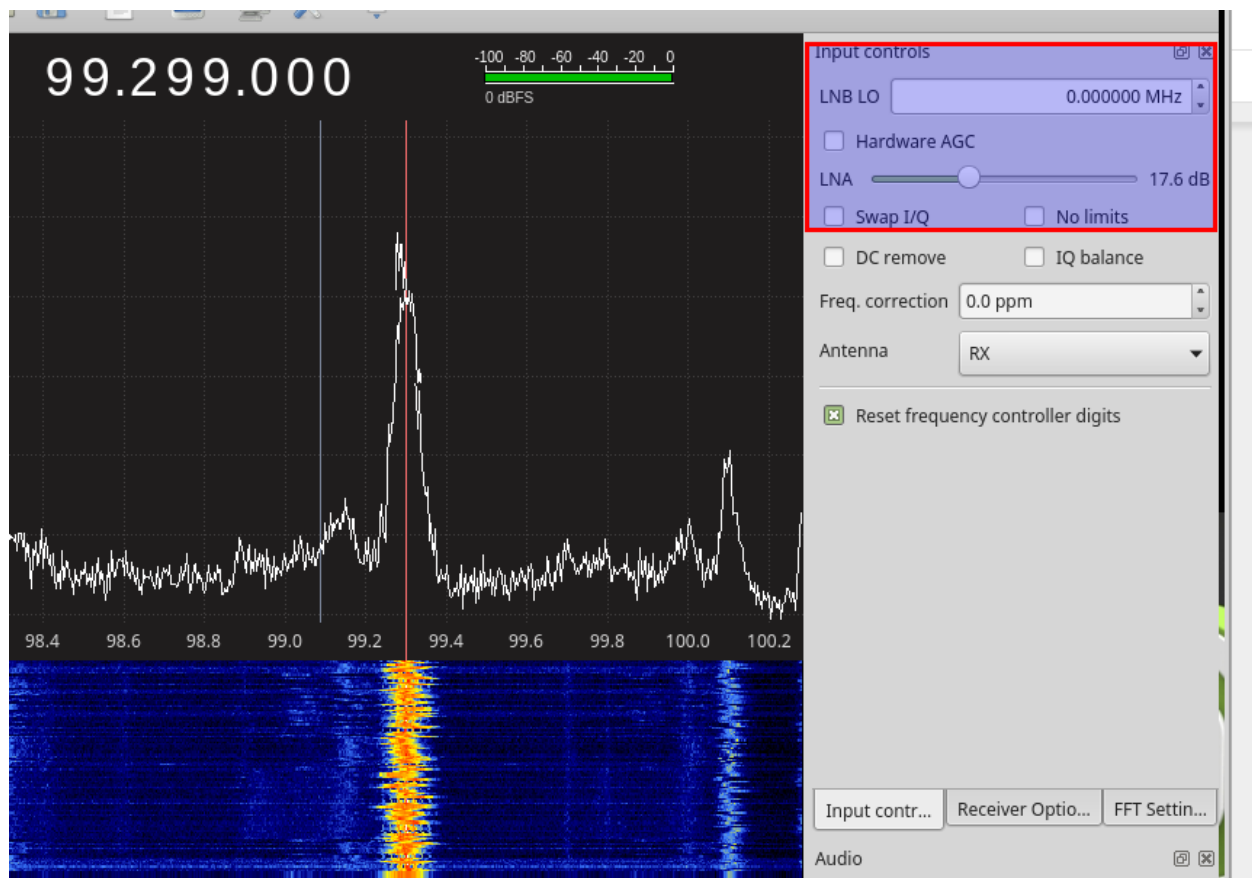
When Downloaded run:

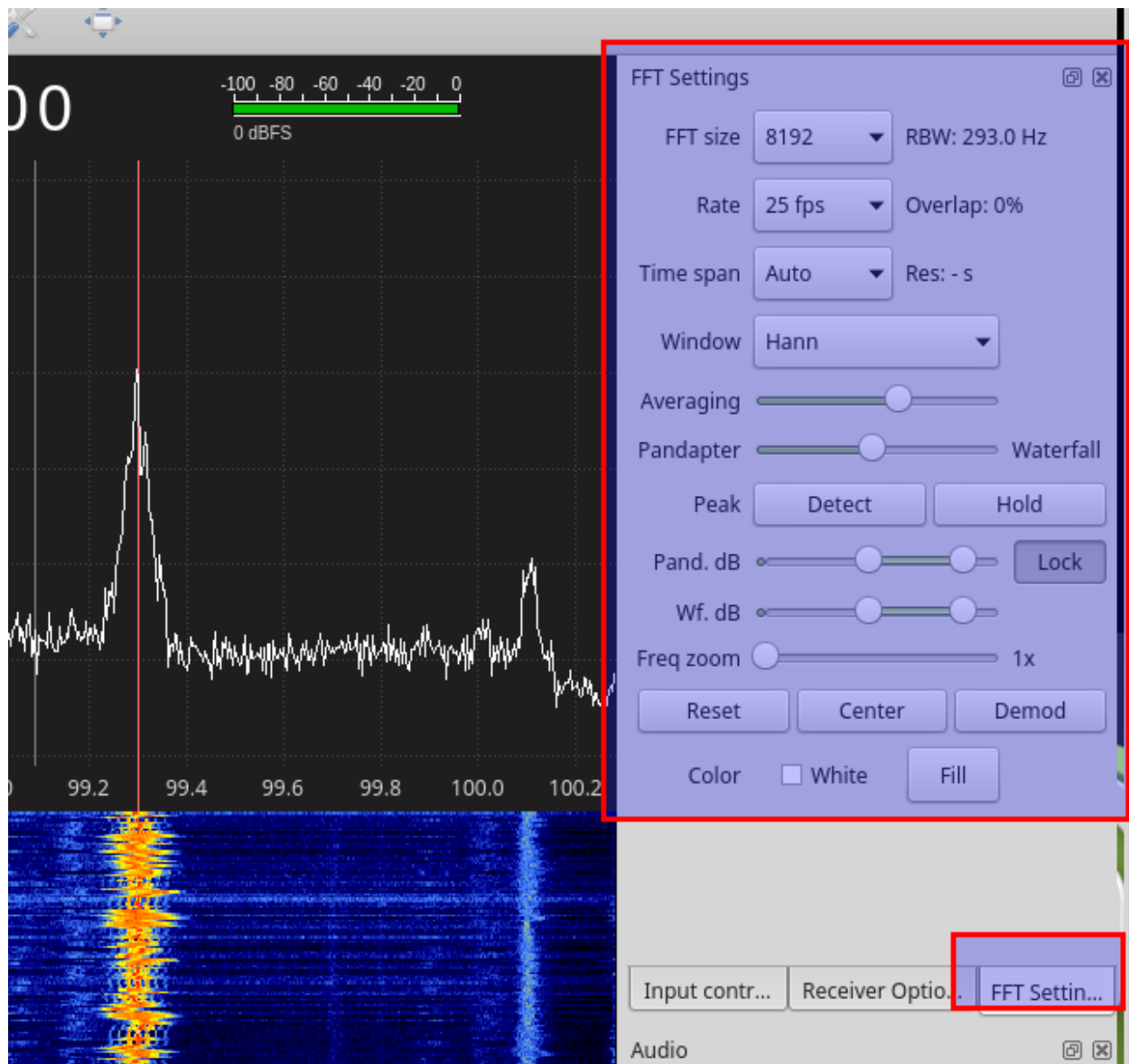
```
chmod +x ~/Downloads/SDRplay_RSP_API-Linux-2.13.1.run
sudo ~/Downloads/SDRplay_RSP_API-Linux-2.13.1.run
sudo ldconfig
```

Now build checkout the special sdrplay2 branch of the gr-osmosdr fork from sdrplay. We want gr-osmosdr build results installed within our home directory in a separate folder, /home/sdr/libs/gr-osmosdr-sdrplay












Downloads


SOFTWARE

 Windows

 Linux x86

 Mac

 Raspberry Pi

 Android

 ARM64

API/HW DRIVER –
V2.13 (20TH JUNE 2018)



(All RSPs – single tuner mode
only)

ADS-B (DUMP1090)

DC


```
cd ~/wrk
git clone https://github.com/sdrplay/gr-osmosdr
cd gr-osmosdr && git checkout sdrplay2 && mkdir build && cd build
mkdir -p ~/wrk/libs/gr-osmosdr-sdrplay
cmake -DCMAKE_INSTALL_PREFIX=~/wrk/libs/gr-osmosdr-sdrplay -DENABLE_NONFREE=yes -
↪DENABLE_BLADERF=OFF ..
make && make install
```

To make GNU Radio, GQRX etc make sure to pick up the shared libraries from your experimental directory, set the `LD_LIBRARY_PATH` & `PYTHONPATH` to the target directory.

```
LD_LIBRARY_PATH=~/wrk/libs/gr-osmosdr-sdrplay/lib:$LD_LIBRARY_PATH
PYTHONPATH=~/wrk/libs/gr-osmosdr-sdrplay/lib/python-2.7/dist-packages:$PYTHONPATH
```

Within that shell you can now start `gnuradio-companion` or `gqrx` and find the SDRplay as OSMOCOM-Source.

-> When sure, you want to target your prefix directory, the `cmake` switch would be `-DCMAKE_INSTALL_PREFIX=~/wrk/grc_wrk/default`

If using an original RSP1, a few lines need to be added to `/etc/modprobe.d/blacklist.conf` by

```
sudo nano /etc/modprobe.d/blacklist.conf
```

Enter at the end of the file:

```
blacklist sdr_msi3101
blacklist msi001
blacklist msi2500
```


CHAPTER 7

Build from source

```
cd ~/wrk
git clone https://github.com/antirez/dump1090.git
cd dump1090
make
```

7.1 Run with an RTL-SDR stick

To run go to the dump1090 directory we just cd-ED into, connect and bind the USB RTL-SDR stick in virtualbox and type:

```
./dump1090 --net --interactive
```

Open Firefox and navigate to <http://127.0.0.1:8080>

You're now able to track the planes above your -> Works best outdoors :-)

The screenshot shows a Mozilla Firefox browser window with the address bar set to `127.0.0.1:8080/`. The main content area displays a map of Europe, specifically focusing on Switzerland and the surrounding regions. A yellow star is placed on the map, indicating a specific location. To the right of the map, a sidebar titled "Dump1090" provides flight information:

- 3 planes on screen.
- ICAO: 738065
- ELY346**
- Altitude: 28525 feet
- Speed: 457 knots
- Coordinates: 47.022788, 7.731558

Below the browser window, a terminal window titled "sdr@sdrmint ~/wrk/dump1090" displays a table of flight data. The table has columns for Hex, Flight, Altitude, Speed, Lat, Lon, Track, Messages, and Seen. The data is as follows:

Hex	Flight	Altitude	Speed	Lat	Lon	Track	Messages	Seen
4b2927		6775	0	0.000	0.000	0	19	4 sec
4ca9d4		37000	0	0.000	0.000	0	3	7 sec
491316		0	0	0.000	0.000	0	3	19 sec
4b197e	SAZ241	31925	457	46.683	6.937	210	83	32 sec
4b17ad		8800	0	0.000	0.000	0	32	0 sec
4b4361		6175	0	0.000	0.000	0	61	0 sec
4b0e47		0	0	0.000	0.000	0	1	51 sec
4ca645		37000	0	0.000	0.000	0	6	50 sec
738065	ELY346	28525	457	47.023	7.732	51	101	1 sec
3c49ce		37000	491	0.000	0.000	211	37	5 sec
3c49d5	EWG5XC	35975	428	46.503	7.765	29	188	2 sec

8.1 Installation

More details on <https://github.com/jopohl/urh#linux>

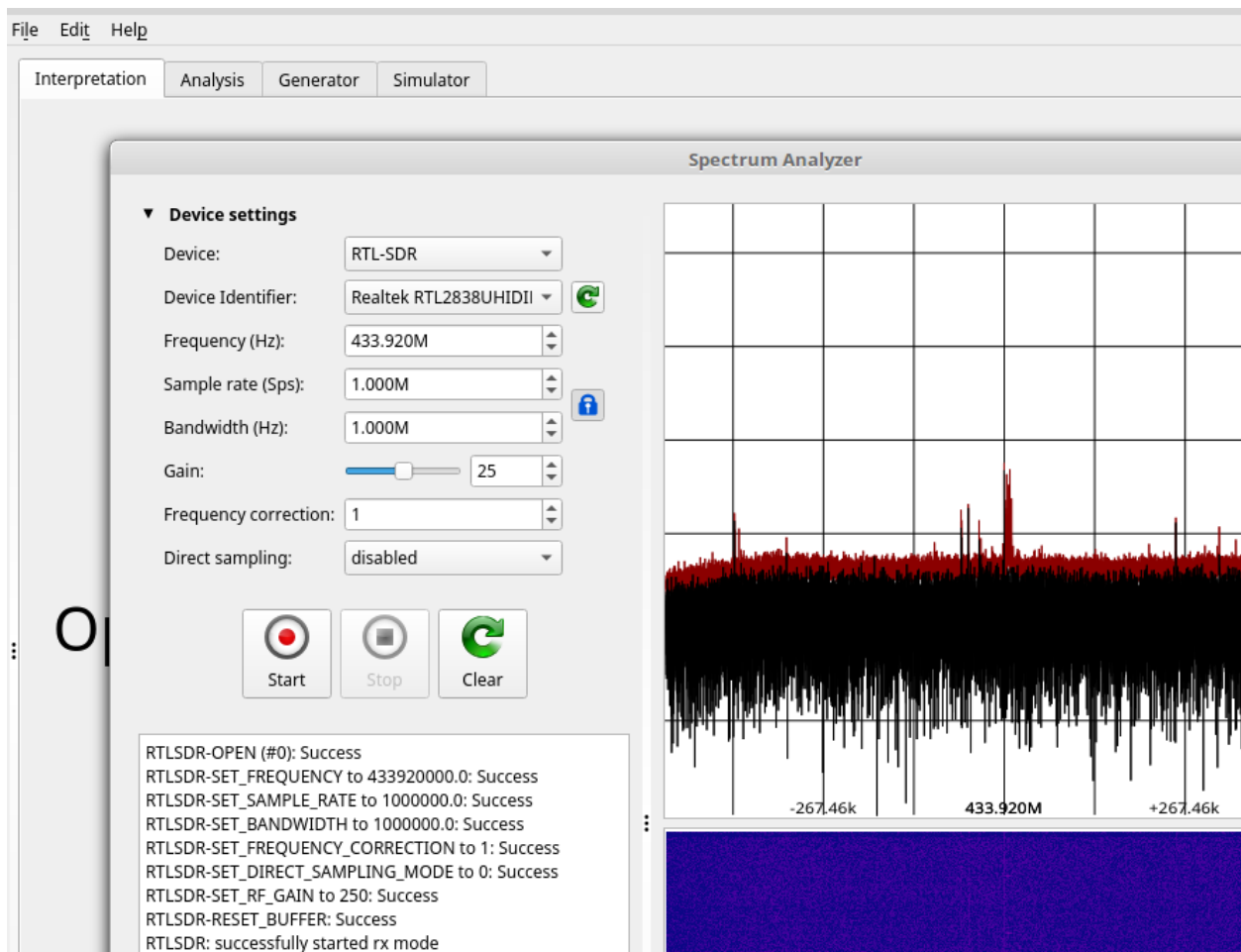
```
sudo apt-get install python3-pip python3-dev
sudo pip3 install --upgrade pip
sudo pip3 install setuptools
sudo pip3 install urh
```

For a RaspberryPI:

```
sudo apt-get update
sudo apt-get install python3-numpy python3-psutil python3-zmq python3-pyqt5 g++
↪ libpython3-dev python3-pip
sudo pip3 install urh
```

8.2 Basic use

To run just type `urh` in a terminal



CHAPTER 9

Baudline

9.1 Use case

To analyze recorded raw IQ data I often use Baudline

To install, go to your wrk directory and download the tar.gz container containing the binaries:

```
cd ~/wrk
wget http://www.baudline.com/baudline_1.08_linux_x86_64.tar.gz
```

Unpack:

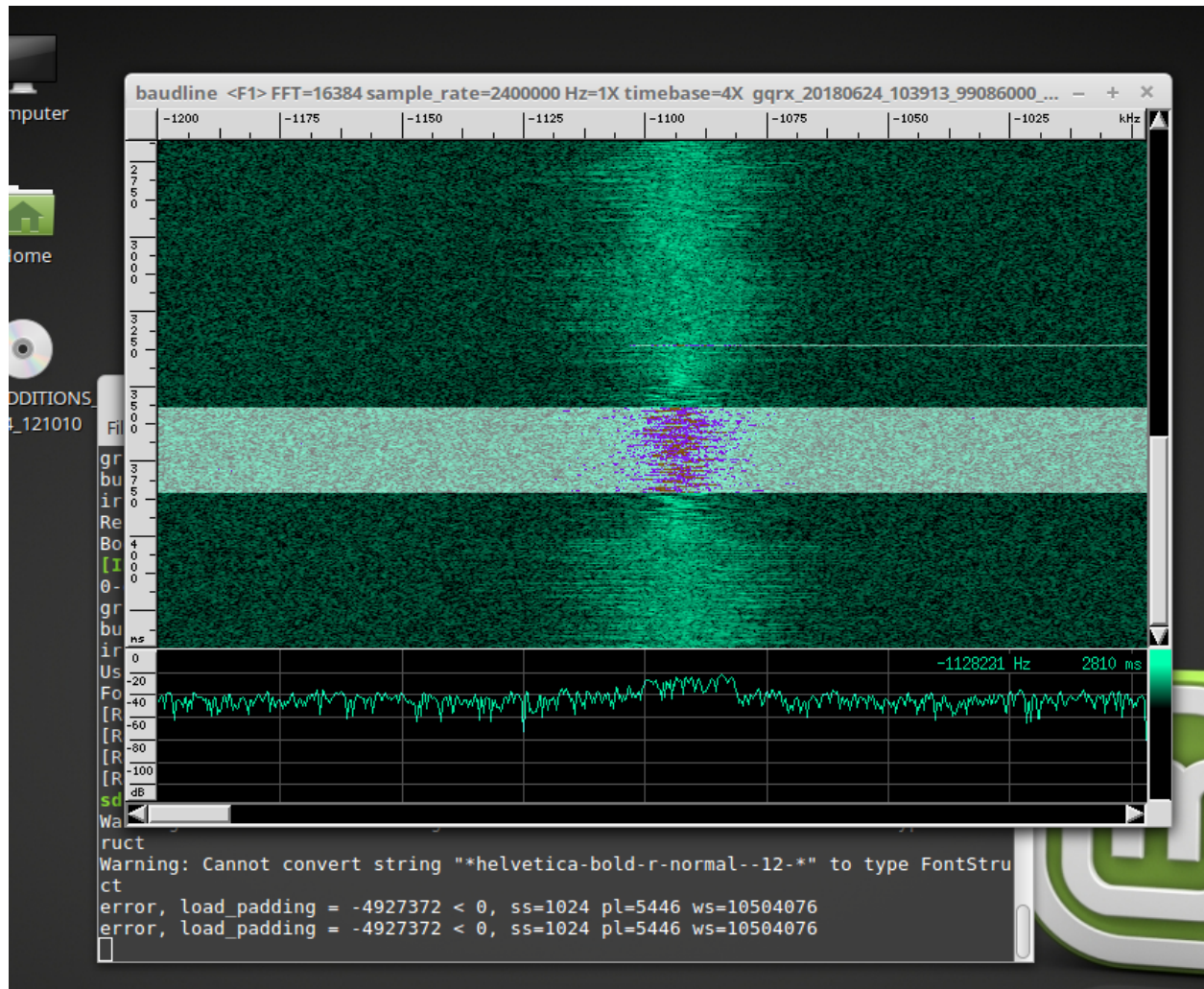
```
tar xvfz baudline_1.08_linux_x86_64.tar.gz
```

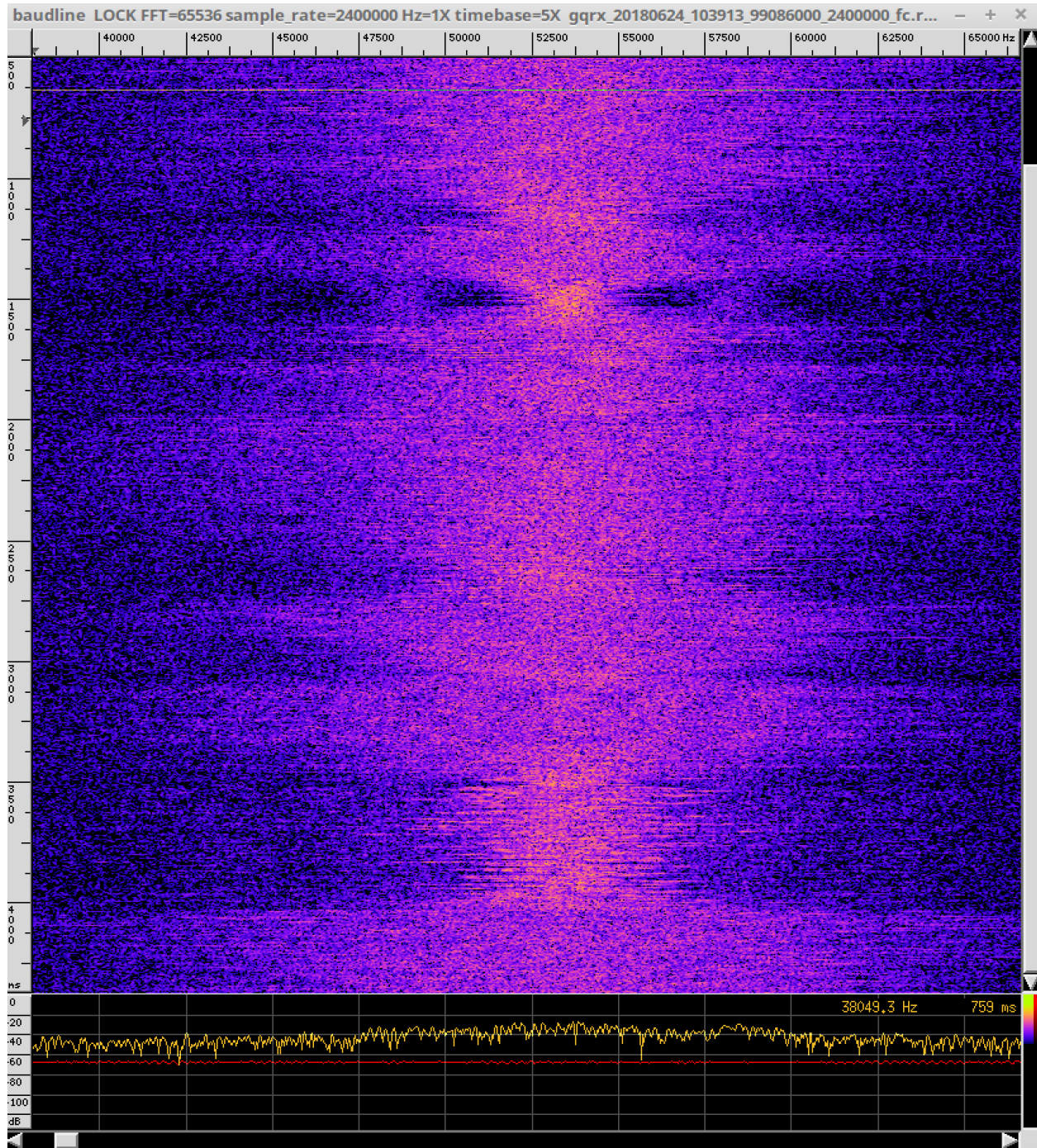
9.2 Run

```
./baudline_1.08_linux_x86_64/baudline
```

9.3 Basic use

The usage is a bit special - but worth the effort. Would be another document! The baudline website has a lot good informations: <http://www.baudline.com/>





CHAPTER 10

Inspectrum, GRC Burst Tags / Great M. Ossmann stuff (Not running in VM)

10.1 Use case

Ideas derived from the video of Michael Ossmann regarding “Whole Packet Clock Recovery” <https://www.youtube.com/watch?v=rQkBDMeODHc>

I’ve copied the GRC flow graph from the youtube video to GRC (No warranties that it is fully correct): <https://gist.github.com/krippendorf/149ffc7cca6ec33eb84daf2c70989829>

Inspectrum is a tool for analysing captured signals, primarily from software-defined radio receivers.

10.2 libliquid-dev

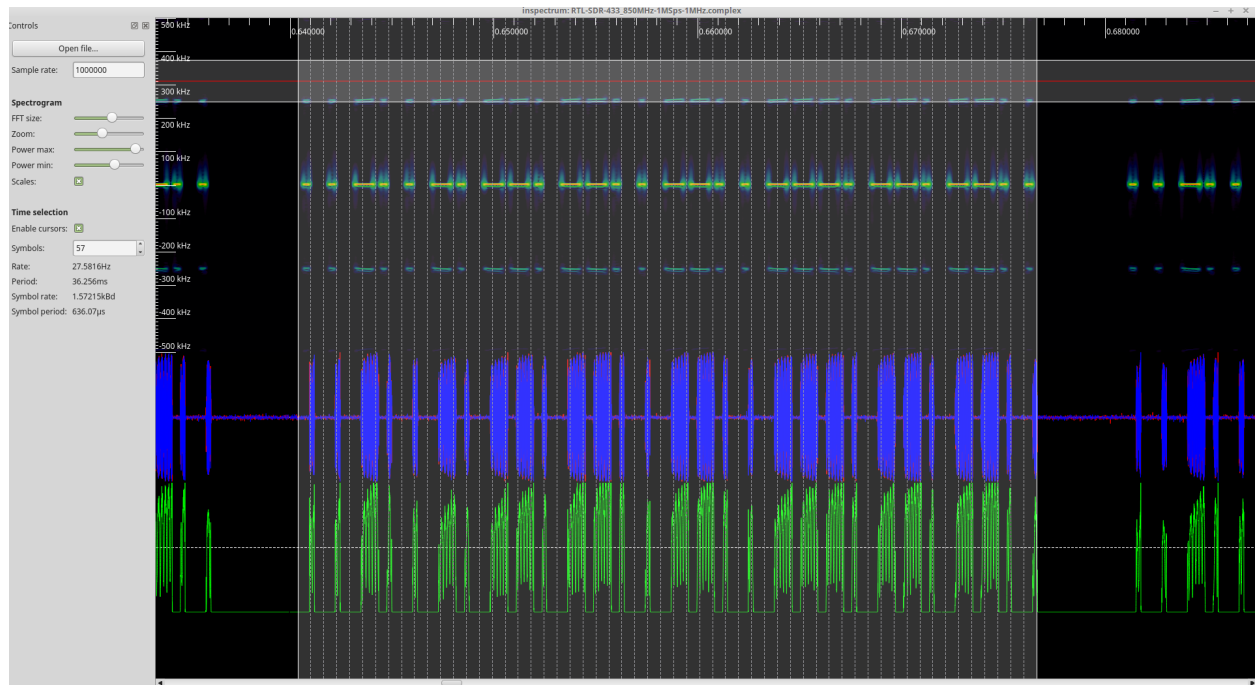
```
sudo apt-get update -y
echo "deb http://cz.archive.ubuntu.com/ubuntu artful main universe" | sudo tee -a /
→etc/apt/sources.list.d/temp.list
sudo apt-get update -y
sudo apt-get install libliquid-dev -y
sudo rm -f /etc/apt/sources.list.d/temp.list
sudo apt-get update -y
```

...We only take libliquid-dev package from this repo. Make sure to not miss the `sudo rm -f` step

10.3 Clone and build

```
sudo apt-get install qt5-default libfftw3-dev cmake pkg-config -y
cd ~/wrk && git clone https://github.com/miek/inspectrum)
cd inspectrum
```

(continues on next page)



(continued from previous page)

```
mkdir build && cd build && cmake .. && make  
sudo make install
```

10.4 Basic use

The Python stuff mentioned in the video is here: <https://github.com/mossmann/clock-recovery>

Run with command: `inspectrum`

CHAPTER 11

For LimeSDR owners

This is not in the provided VM image - so install on your own!

11.1 Clone and install

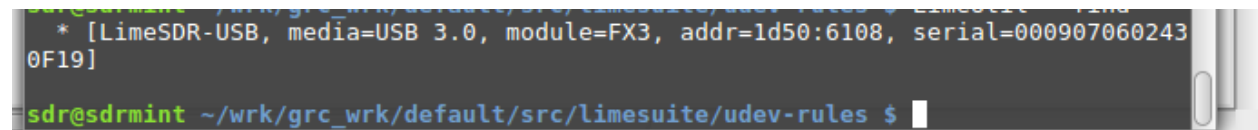
In the terminal type:

```
pybombs install gr-limesdr
cd ~/wrk/grc_wrk/default/src/limesuite/udev-rules
sudo chmod +x install.sh
sudo ./install.sh
```

11.2 Run

Connect your Lime and route it to the VM “Devices->USB etc etc.”

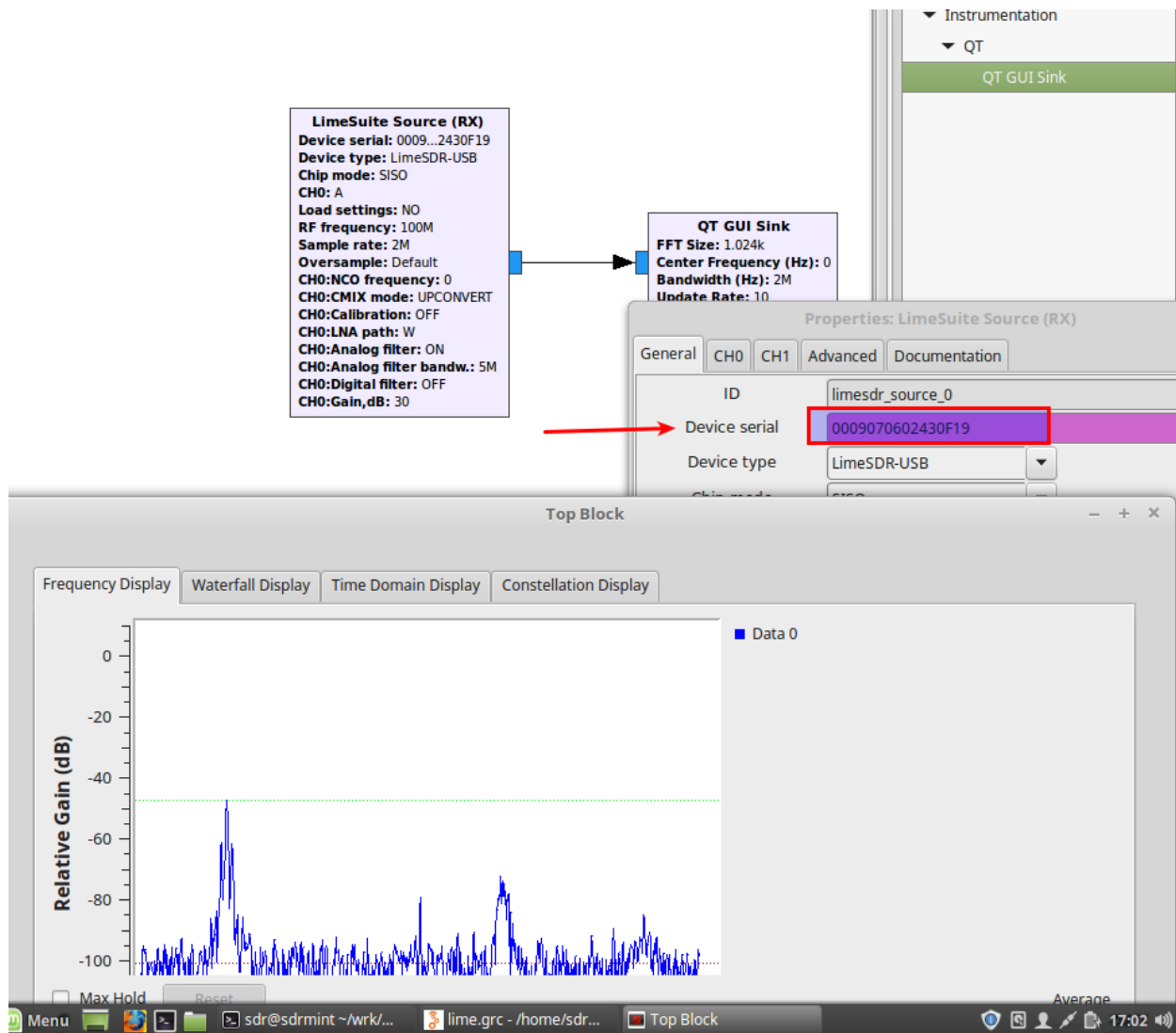
You can now use LimeUtil --find, remember to to a LimeUtil --update

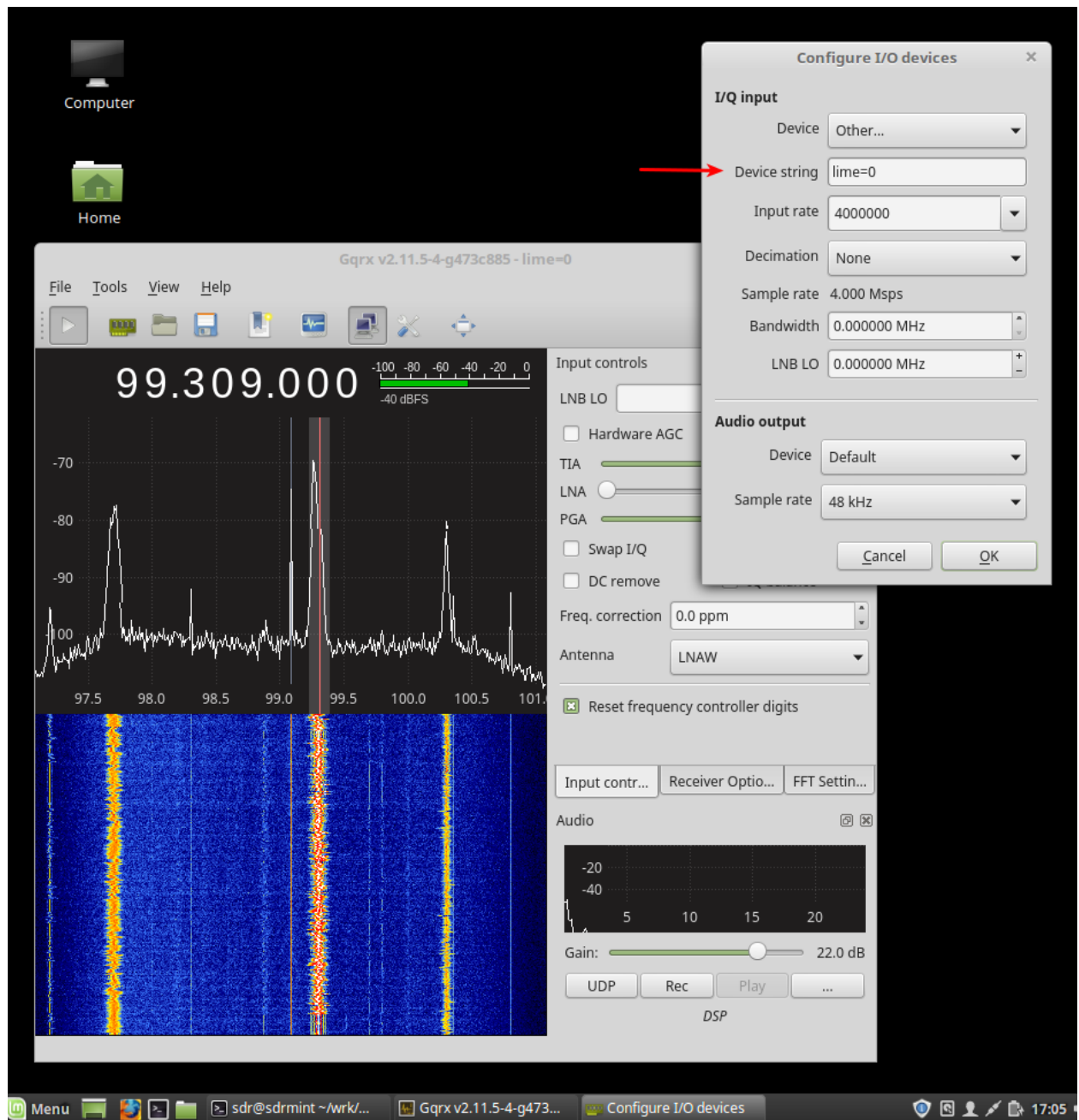


```
sdr@sdrmint ~/wrk/grc_wrk/default/src/limesuite/udev-rules $ LimeUtil --find
* [LimeSDR-USB, media=USB 3.0, module=FX3, addr=1d50:6108, serial=0009070602430F19]
sdr@sdrmint ~/wrk/grc_wrk/default/src/limesuite/udev-rules $
```

You can now use the GR-Lime source and sink blocks in GNU Radio companion:

To use the Lime in GQRX use the Device “**Other**” with Device String “**lime=0**”





CHAPTER 12

WIP: Native SDRplay source blocks

I'm working hard to make SDRPlay a part of this List - In my eyes with great performance for ham radio enthusiasts
- <https://twitter.com/HB9FXQ/status/1010926288641626112>

Now my code is feature complete and I've provided it to some friends for code review and testing. The manual will be updated soon.

CHAPTER 13

BETA gr-sdrplay

Make sure to download and install API/HW Driver – v2.13 (20th June 2018) from sdrplay.com

```
cd ~/wrk
source ~/wrk/grc_wrk/default/setup_env.sh
git clone https://gitlab.com/HB9FXQ/gr-sdrplay.git
cd gr-sdrplay && mkdir build && cd build && cmake .. && make && make install
sudo ldconfig
```

A sample file to open with GNU Radio is available under `~/wrk/gr-sdrplay/examples/development_gui.grc`