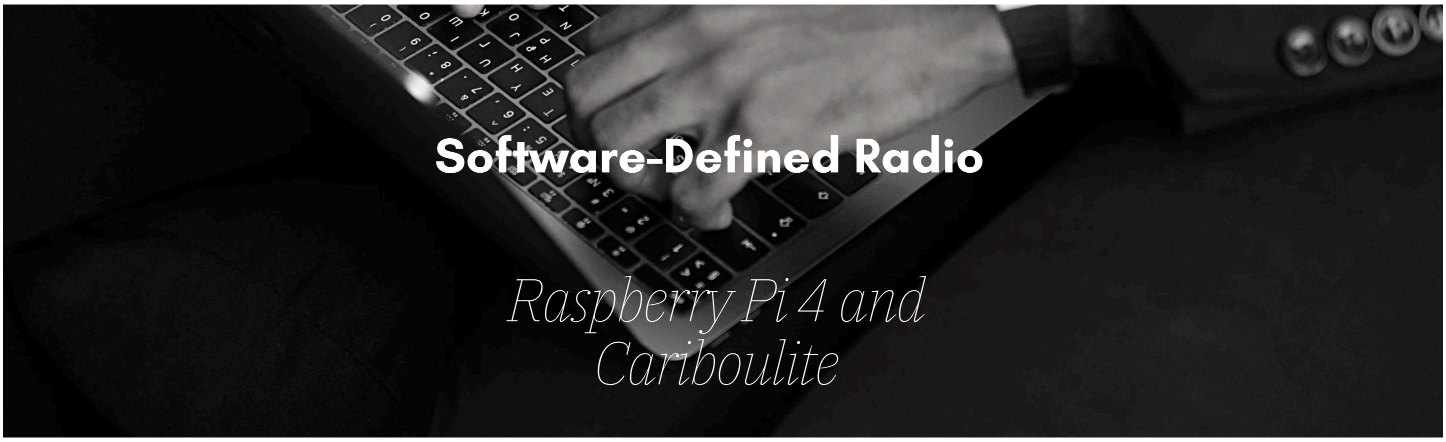
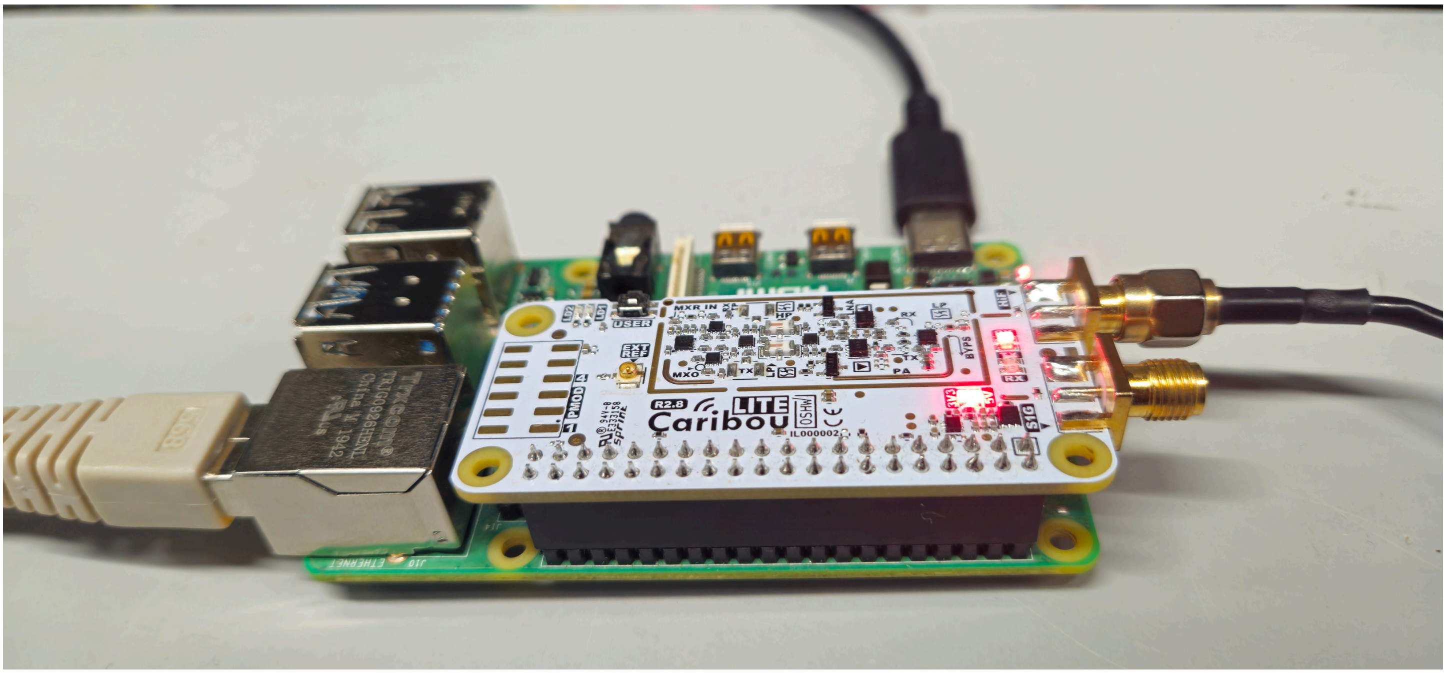


Software-Defined Radio

Raspberry Pi 4 and Cariboulite

Project:

- First goal was to get the CaribouLite SDR device operational with Raspberry Pi 4.
- Second goal was to confirm that CaribouLite is capable of transmitting and receiving a radio frequency signal.



TOOLS AND SOFTWARE UTILIZED

The following hardware components were used during the project:

- CaribouLite - Software-defined radio (SDR) device
- Raspberry Pi 4 - Used as the host system to control the SDR and run necessary software
- Antenna - Connected to the CaribouLite
- Micro SD-card
- Spectrum Analyzer - for the signal testing purposes

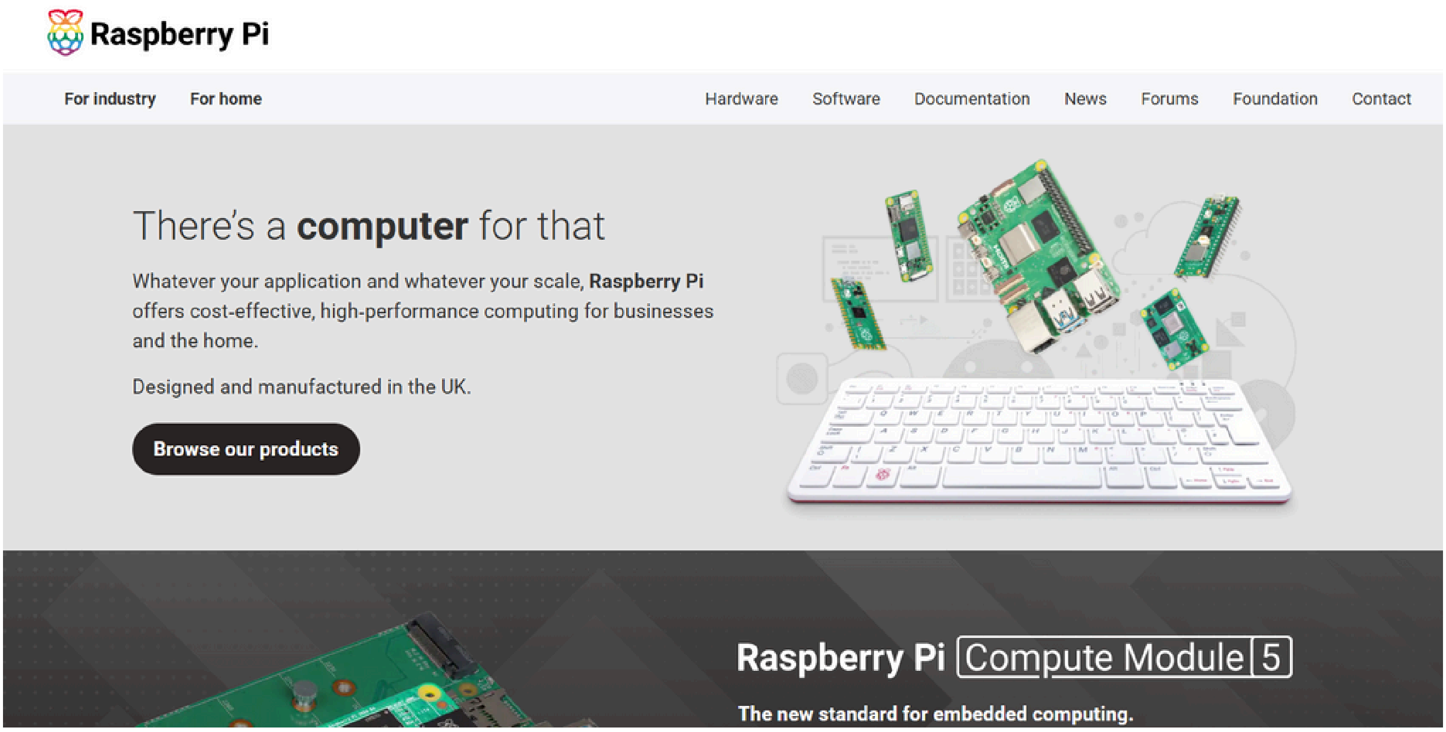
For the software part, the following tools and libraries were used:

- Raspberry Pi Bookworm OS – Debian 12-version
- CaribouLite's own drivers - Spesific drivers required for enabling communication
- SoapySDR – A hardware abstraction library for interfacing with the CaribouLite SDR
- SDR++ – A graphical SDR application used to monitor and analyze signals
- SSH – remote access to the Raspberry Pi
- SDRAngel - A versatile SDR application designed for both signal reception and transmission, featuring a modular interface and support for a wide range of hardware and digital modes

PROJECT STEPS

Raspberry Pi 4 installation

- The first step of the project was to select a suitable operating system. I chose a Debian-based OS, version 12 (Bookworm), primarily because it is lightweight and well-supported on the Raspberry Pi.
    - I used Raspberry Pi Imager to flash the OS image to the microSD card.
    - I created SSH keys to enable secure remote access to the Raspberry Pi.
- For the basic configurations, i followed Raspberry Pi's own documentation available on their website!



There's a **computer** for that

Whatever your application and whatever your scale, **Raspberry Pi** offers cost-effective, high-performance computing for businesses and the home.

Designed and manufactured in the UK.

The new standard for embedded computing.

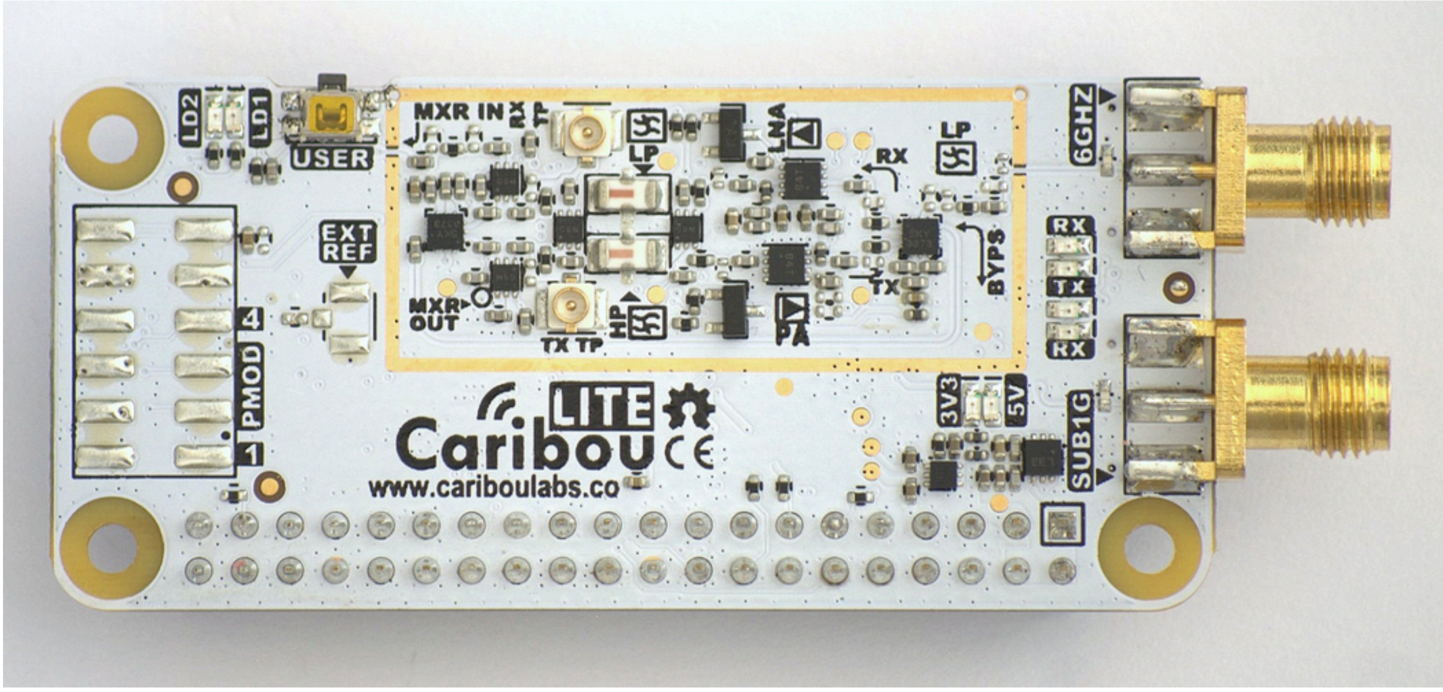
The official Raspberry Pi website is like a 'holy bible' for us tech enthusiasts, with everything related to Raspberry Pi products

PROJECT STEPS

CaribouLite

CaribouLite has an official Github repository available here.

To ensure proper functionality, I followed the step-by-step installation guide. This included cloning the repository, installing required dependencies and setting up the drivers on the Raspberry Pi.



Picture taken from CaribouLite’s Github repository.

PROJECT STEPS

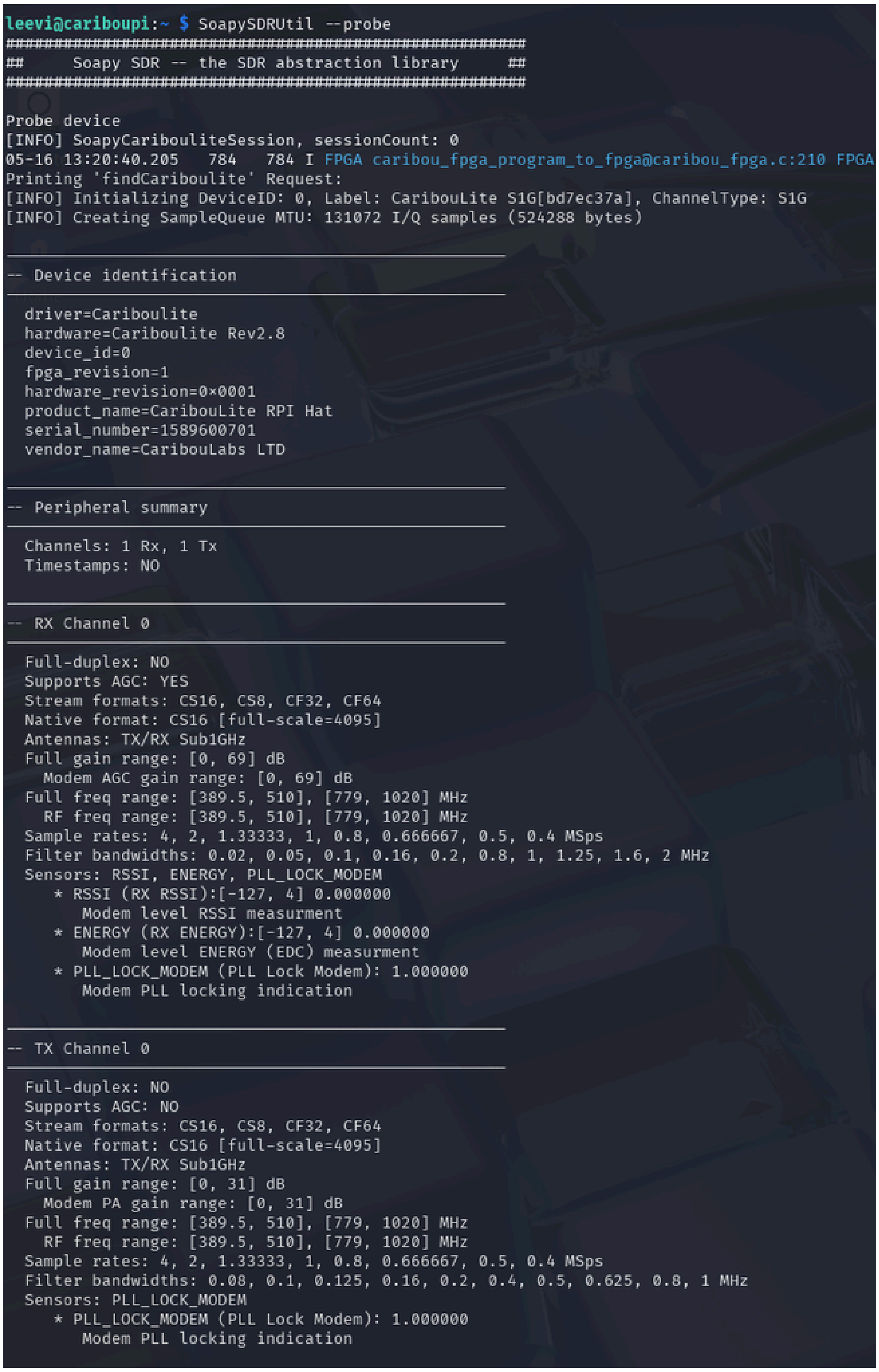
SoapySDR

To interface with the CaribouLite through SDR applications, I installed and configured SoapySDR, a modular and device-independent SDR support library.

I followed basic installation steps:

- Installed required package: soapysdr-tools libsoapysdr libsoapysdr-dev
- I cloned the official Github repository
- Created a build directory and ran CMake to configure the build environment
- Lastly ran make install to compile and install

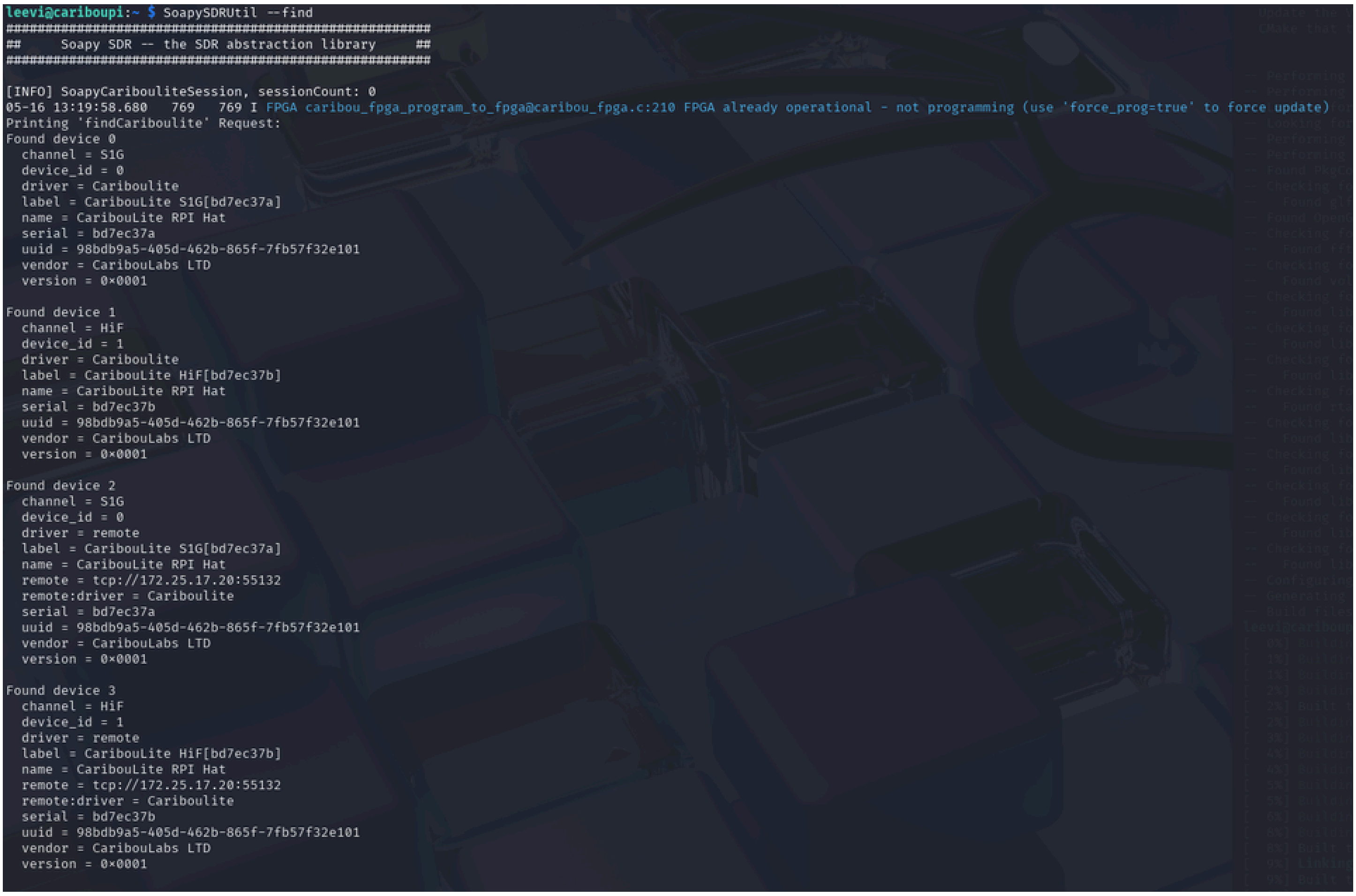
Testing the SoapySDR



```
leevi@cariboupi:~ $ SoapySDRUtil --probe
######################################################
##     Soapy SDR -- the SDR abstraction library     ##
######################################################

Probe device
[INFO] SoapyCaribouliteSession, sessionCount: 0
05-16 13:20:40.205   784   784 I FPGA caribou_fpga_program_to_fpga@caribou_fpga.c:210 FPGA
Printing 'findCaribouLite' Request:
[INFO] Initializing DeviceID: 0, Label: CaribouLite S1G[bd7ec37a], ChannelType: S1G
[INFO] Creating SampleQueue MTU: 131072 I/Q samples (524288 bytes)

----------------------------------------------------
-- Device identification
----------------------------------------------------
  driver=Cariboulite
  hardware=Cariboulite Rev2.8
  device_id=0
  fpga_revision=1
  hardware_revision=0x0001
  product_name=CaribouLite RPI Hat
  serial_number=1589600701
  vendor_name=CaribouLabs LTD

----------------------------------------------------
-- Peripheral summary
----------------------------------------------------
  Channels: 1 Rx, 1 Tx
  Timestamps: NO

----------------------------------------------------
-- RX Channel 0
----------------------------------------------------
  Full-duplex: NO
  Supports AGC: YES
  Stream formats: CS16, CS8, CF32, CF64
  Native format: CS16 [full-scale=4095]
  Antennas: TX/RX Sub1GHz
  Full gain range: [0, 69] dB
    Modem AGC gain range: [0, 69] dB
  Full freq range: [389.5, 510], [779, 1020] MHz
    RF freq range: [389.5, 510], [779, 1020] MHz
  Sample rates: 4, 2, 1.33333, 1, 0.8, 0.666667, 0.5, 0.4 MSps
  Filter bandwidths: 0.02, 0.05, 0.1, 0.16, 0.2, 0.8, 1, 1.25, 1.6, 2 MHz
  Sensors: RSSI, ENERGY, PLL_LOCK_MODEM
     * RSSI (RX RSSI):[-127, 4] 0.000000
        Modem level RSSI measurment
     * ENERGY (RX ENERGY):[-127, 4] 0.000000
        Modem level ENERGY (EDC) measurment
     * PLL_LOCK_MODEM (PLL Lock Modem): 1.000000
        Modem PLL locking indication

----------------------------------------------------
-- TX Channel 0
----------------------------------------------------
  Full-duplex: NO
  Supports AGC: NO
  Stream formats: CS16, CS8, CF32, CF64
  Native format: CS16 [full-scale=4095]
  Antennas: TX/RX Sub1GHz
  Full gain range: [0, 31] dB
    Modem PA gain range: [0, 31] dB
  Full freq range: [389.5, 510], [779, 1020] MHz
    RF freq range: [389.5, 510], [779, 1020] MHz
  Sample rates: 4, 2, 1.33333, 1, 0.8, 0.666667, 0.5, 0.4 MSps
  Filter bandwidths: 0.08, 0.1, 0.125, 0.16, 0.2, 0.4, 0.5, 0.625, 0.8, 1 MHz
  Sensors: PLL_LOCK_MODEM
     * PLL_LOCK_MODEM (PLL Lock Modem): 1.000000
        Modem PLL locking indication
```

To test and verify that the CaribouLite device was correctly recognized by SoapySDR, first i used command “SoapySDRUtil --probe”. This command probes all connected SDR devices and displays detailed information about each one.

As shown in the screenshot, the command successfully detected the CaribouLite device and displayed its specifications. It also identified the two available channels on the device: RX (receive) and TX (transmit).



```
leevi@cariboupi:~ $ SoapySDRUtil --find
######################################################
##     Soapy SDR -- the SDR abstraction library     ##
######################################################

[INFO] SoapyCaribouliteSession, sessionCount: 0
05-16 13:19:58.680   769   769 I FPGA caribou_fpga_program_to_fpga@caribou_fpga.c:210 FPGA already operational - not programming (use 'force_prog=true' to force update)
Printing 'findCariboulite' Request:
Found device 0
  channel = S1G
  device_id = 0
  driver = Cariboulite
  label = CaribouLite S1G[bd7ec37a]
  name = CaribouLite RPI Hat
  serial = bd7ec37a
  uuid = 98bdb9a5-405d-462b-865f-7fb57f32e101
  vendor = CaribouLabs LTD
  version = 0x0001

Found device 1
  channel = HiF
  device_id = 1
  driver = Cariboulite
  label = CaribouLite HiF[bd7ec37b]
  name = CaribouLite RPI Hat
  serial = bd7ec37b
  uuid = 98bdb9a5-405d-462b-865f-7fb57f32e101
  vendor = CaribouLabs LTD
  version = 0x0001

Found device 2
  channel = S1G
  device_id = 0
  driver = remote
  label = CaribouLite S1G[bd7ec37a]
  name = CaribouLite RPI Hat
  remote = tcp://172.25.17.20:55132
  remote:driver = Cariboulite
  serial = bd7ec37a
  uuid = 98bdb9a5-405d-462b-865f-7fb57f32e101
  vendor = CaribouLabs LTD
  version = 0x0001

Found device 3
  channel = HiF
  device_id = 1
  driver = remote
  label = CaribouLite HiF[bd7ec37b]
  name = CaribouLite RPI Hat
  remote = tcp://172.25.17.20:55132
  remote:driver = Cariboulite
  serial = bd7ec37b
  uuid = 98bdb9a5-405d-462b-865f-7fb57f32e101
  vendor = CaribouLabs LTD
  version = 0x0001
```

After device information, i ran the “SoapySDRUtil --find”-command to perform a quick scan for connected SDR devices. This command lists all devices all available SDR devices on the system. It also found the two available channels on the device, HiF and S1G, and they are also available for remote TCP connection. Later identified that the remote connection was in wrong port.

PROJECT STEPS

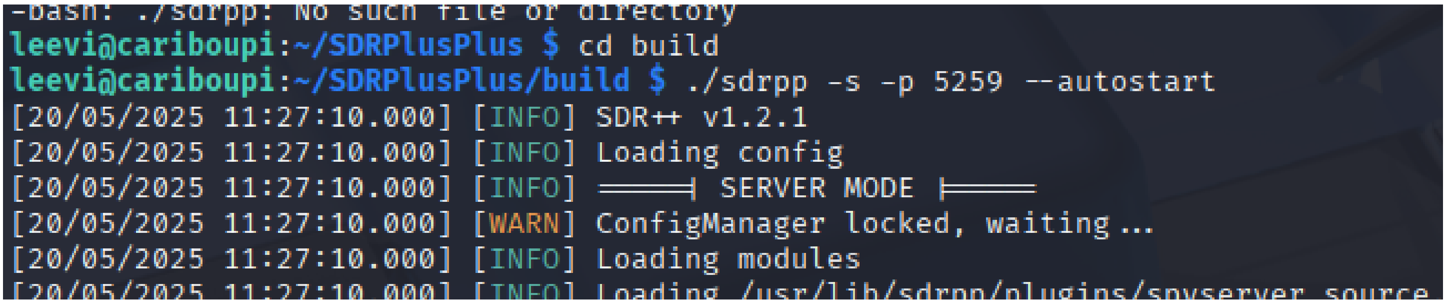
SDR++

SDR++ is an open-source software-defined radio (SDR) application that offers broad hardware support, including full compatibility with the SoapySDR module.

It is also cross-platform, meaning you can download and run the appropriate version for your operating system (Windows, Linux, macOS, or BSD). The software features a modular design, which allows users to install custom plugins and extend its functionality according to their needs.

Setup

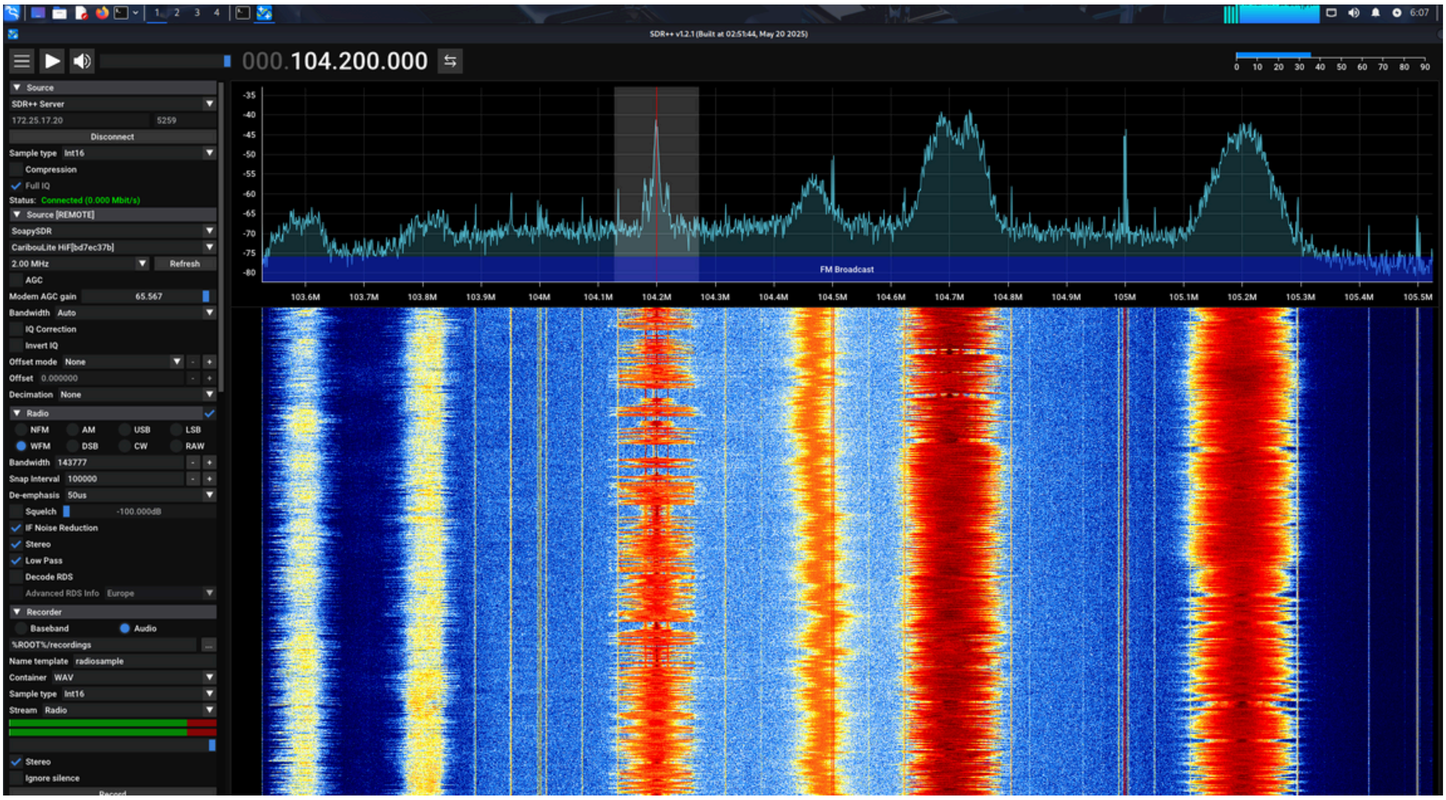
My goal was to set up a lightweight SDR++ server that could run on the Raspberry Pi and be accessed remotely over SSH. This setup would allow me to monitor and control the SDR from another device without needing a graphical user interface on the Pi itself. So I did that first on to Pi.



```
-bash: ./sdrpp: No such file or directory
leevi@cariboupi:~/SDRPlusPlus $ cd build
leevi@cariboupi:~/SDRPlusPlus/build $ ./sdrpp -s -p 5259 --autostart
[20/05/2025 11:27:10.000] [INFO] SDR++ v1.2.1
[20/05/2025 11:27:10.000] [INFO] Loading config
[20/05/2025 11:27:10.000] [INFO] ═════╡ SERVER MODE ╞═════
[20/05/2025 11:27:10.000] [WARN] ConfigManager locked, waiting...
[20/05/2025 11:27:10.000] [INFO] Loading modules
[20/05/2025 11:27:10.000] [INFO] Loading /usr/lib/sdrpp/plugins/spyserver_source
```

Here is the command that shows the server is starting on port 5259. This was a crucial step to ensure the server is on correct port and correctly configured.

Next I set up an SDR++ client on a Linux-based virtual machine to connect to the server running on the Raspberry Pi.



SDR++ v1.2.1 (Built at 02:51:44, May 20 2025)

000.104.200.000

▼ Source

SDR++ Server ▼

172.25.17.20 5259

Disconnect

Sample type Int16 ▼

□ Compression

☑ Full IQ

Status: Connected (0.000 Mbit/s)

▼ Source [REMOTE]

SoapySDR ▼

CaribouLite HiF[bd7ec37b] ▼

2.00 MHz ▼ Refresh

□ AGC

Modem AGC gain 65.567

Bandwidth Auto ▼

□ IQ Correction

□ Invert IQ

Offset mode None ▼ - +

Offset 0.000000 - +

Decimation None ▼

▼ Radio ✓

○ NFM ○ AM ○ USB ○ LSB

● WFM ○ DSB ○ CW ○ RAW

Bandwidth 143777 - +

Snap Interval 100000 - +

De-emphasis 50us ▼

□ Squelch -100.000dB

☑ IF Noise Reduction

☑ Stereo

☑ Low Pass

□ Decode RDS

□ Advanced RDS Info Europe ▼

▼ Recorder

○ Baseband ● Audio

%ROOT%/recordings ...

Name template radiosample

Container WAV ▼

Sample type Int16 ▼

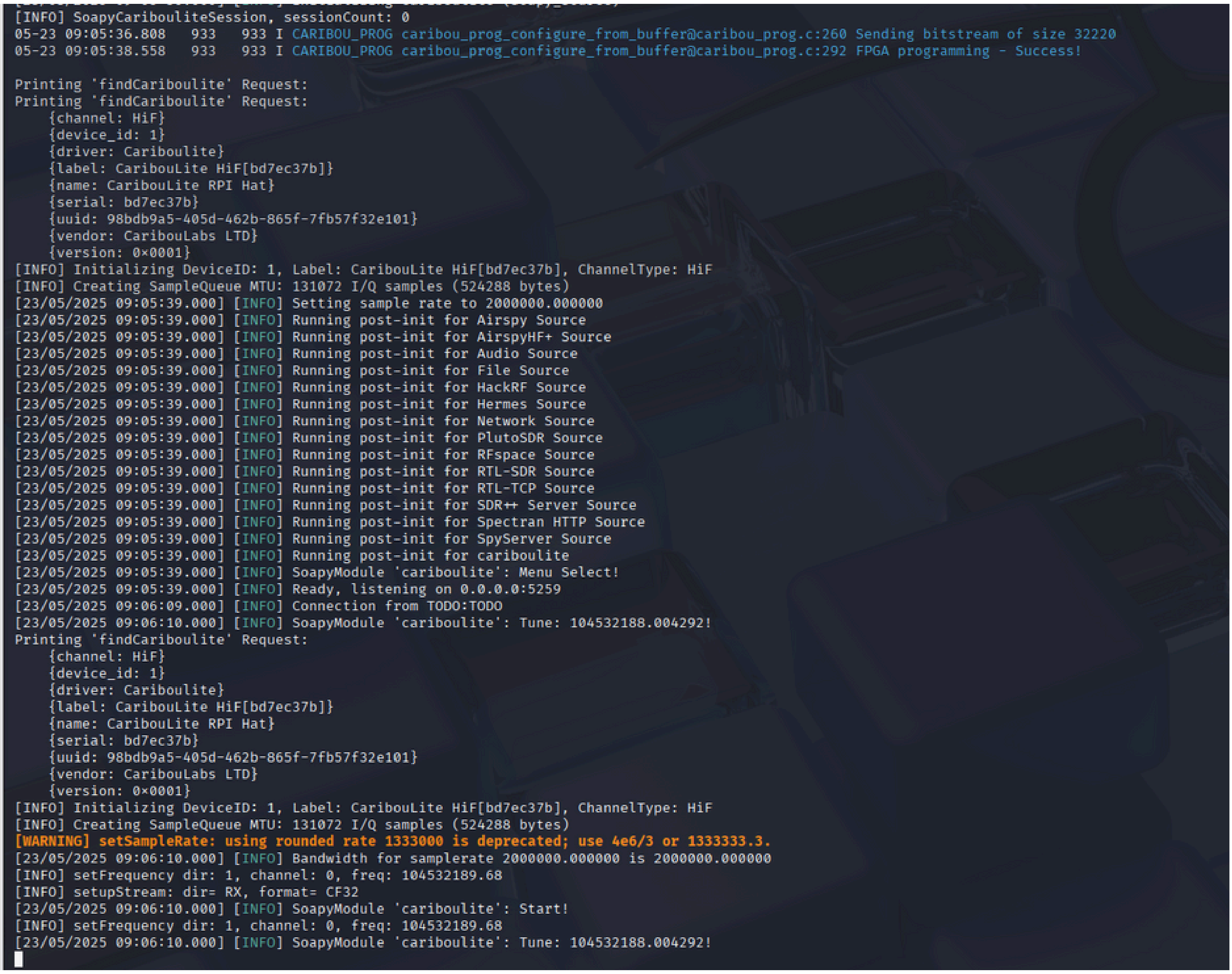
Stream Radio ▼

☑ Stereo

□ Ignore silence

Record

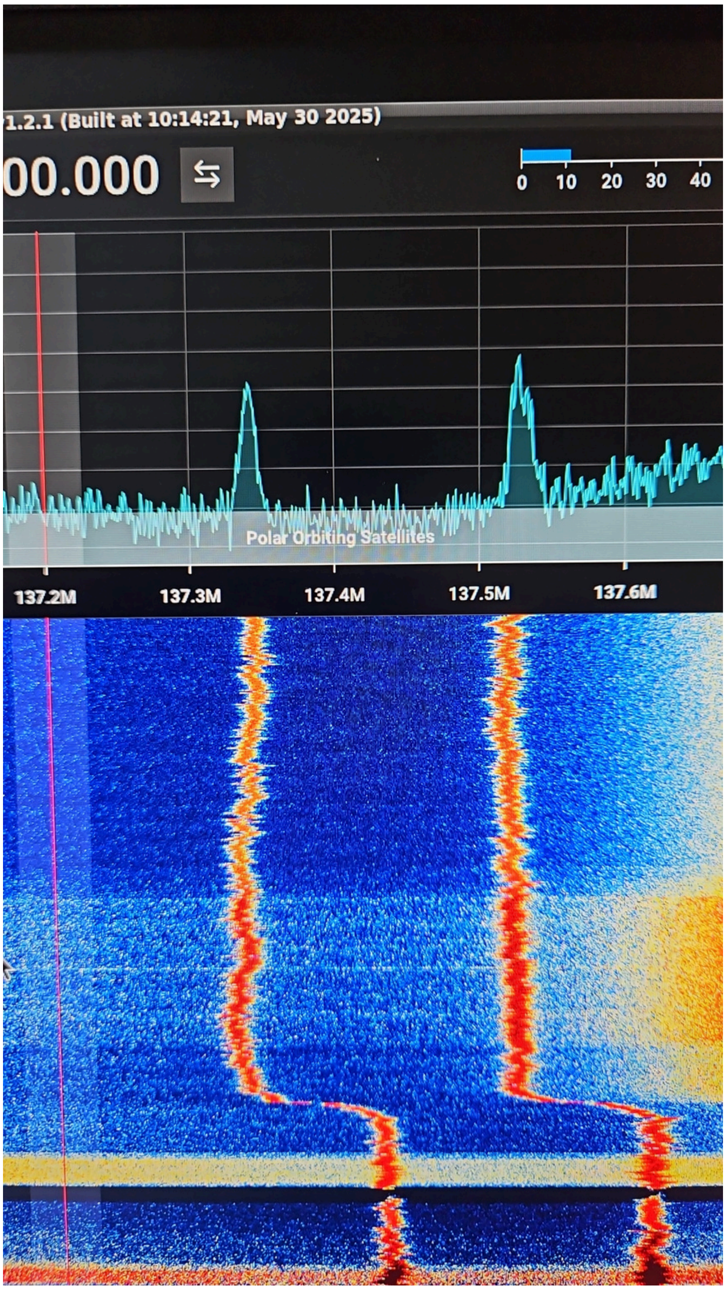
After succesfully connecting to the SDR++server, the software automatically selected the remote source, which in this case was the SoapySDR module and CaribouLite HiF (where 'HiF' stands for High Frequency). By adjusting the appropriate bandwidth and selecting WFM (Wideband-FM), I was able to listen to radio channels.



```
[INFO] SoapyCaribouliteSession, sessionCount: 0
05-23 09:05:36.808   933   933 I CARIBOU_PROG caribou_prog_configure_from_buffer@caribou_prog.c:260 Sending bitstream of size 32220
05-23 09:05:38.558   933   933 I CARIBOU_PROG caribou_prog_configure_from_buffer@caribou_prog.c:292 FPGA programming - Success!

Printing 'findCariboulite' Request:
Printing 'findCariboulite' Request:
    {channel: HiF}
    {device_id: 1}
    {driver: Cariboulite}
    {label: CaribouLite HiF[bd7ec37b]}
    {name: CaribouLite RPI Hat}
    {serial: bd7ec37b}
    {uuid: 98bdb9a5-405d-462b-865f-7fb57f32e101}
    {vendor: CaribouLabs LTD}
    {version: 0×0001}
[INFO] Initializing DeviceID: 1, Label: CaribouLite HiF[bd7ec37b], ChannelType: HiF
[INFO] Creating SampleQueue MTU: 131072 I/Q samples (524288 bytes)
[23/05/2025 09:05:39.000] [INFO] Setting sample rate to 2000000.000000
[23/05/2025 09:05:39.000] [INFO] Running post-init for Airspy Source
[23/05/2025 09:05:39.000] [INFO] Running post-init for AirspyHF+ Source
[23/05/2025 09:05:39.000] [INFO] Running post-init for Audio Source
[23/05/2025 09:05:39.000] [INFO] Running post-init for File Source
[23/05/2025 09:05:39.000] [INFO] Running post-init for HackRF Source
[23/05/2025 09:05:39.000] [INFO] Running post-init for Hermes Source
[23/05/2025 09:05:39.000] [INFO] Running post-init for Network Source
[23/05/2025 09:05:39.000] [INFO] Running post-init for PlutoSDR Source
[23/05/2025 09:05:39.000] [INFO] Running post-init for RFspace Source
[23/05/2025 09:05:39.000] [INFO] Running post-init for RTL-SDR Source
[23/05/2025 09:05:39.000] [INFO] Running post-init for RTL-TCP Source
[23/05/2025 09:05:39.000] [INFO] Running post-init for SDR++ Server Source
[23/05/2025 09:05:39.000] [INFO] Running post-init for Spectran HTTP Source
[23/05/2025 09:05:39.000] [INFO] Running post-init for SpyServer Source
[23/05/2025 09:05:39.000] [INFO] Running post-init for cariboulite
[23/05/2025 09:05:39.000] [INFO] SoapyModule 'cariboulite': Menu Select!
[23/05/2025 09:05:39.000] [INFO] Ready, listening on 0.0.0.0:5259
[23/05/2025 09:06:09.000] [INFO] Connection from TODO:TODO
[23/05/2025 09:06:10.000] [INFO] SoapyModule 'cariboulite': Tune: 104532188.004292!
Printing 'findCariboulite' Request:
    {channel: HiF}
    {device_id: 1}
    {driver: Cariboulite}
    {label: CaribouLite HiF[bd7ec37b]}
    {name: CaribouLite RPI Hat}
    {serial: bd7ec37b}
    {uuid: 98bdb9a5-405d-462b-865f-7fb57f32e101}
    {vendor: CaribouLabs LTD}
    {version: 0×0001}
[INFO] Initializing DeviceID: 1, Label: CaribouLite HiF[bd7ec37b], ChannelType: HiF
[INFO] Creating SampleQueue MTU: 131072 I/Q samples (524288 bytes)
[WARNING] setSampleRate: using rounded rate 1333000 is deprecated; use 4e6/3 or 1333333.3.
[23/05/2025 09:06:10.000] [INFO] Bandwidth for samplerate 2000000.000000 is 2000000.000000
[INFO] setFrequency dir: 1, channel: 0, freq: 104532189.68
[INFO] setupStream: dir= RX, format= CF32
[23/05/2025 09:06:10.000] [INFO] SoapyModule 'cariboulite': Start!
[INFO] setFrequency dir: 1, channel: 0, freq: 104532189.68
[23/05/2025 09:06:10.000] [INFO] SoapyModule 'cariboulite': Tune: 104532188.004292!
```

This screenshot above shows the server activity during bandwidth adjustment. This information is important if you encounter issues between the server and the client.

Want to see something cool?


1.2.1 (Built at 10:14:21, May 30 2025)
00.000
⇆
0
10
20
30
40
Polar Orbiting Satellites
137.2M
137.3M
137.4M
137.5M
137.6M

This screenshot is really interesting one. These two signal spikes originate from weather satellites—the one on the left is from the Russian Meteor-M N2, and the one on the right is from NOAA. Both of these satellites orbit at an altitude of approximately 830 km above the Earth's surface. Naturally, I had to try capturing those signals myself.

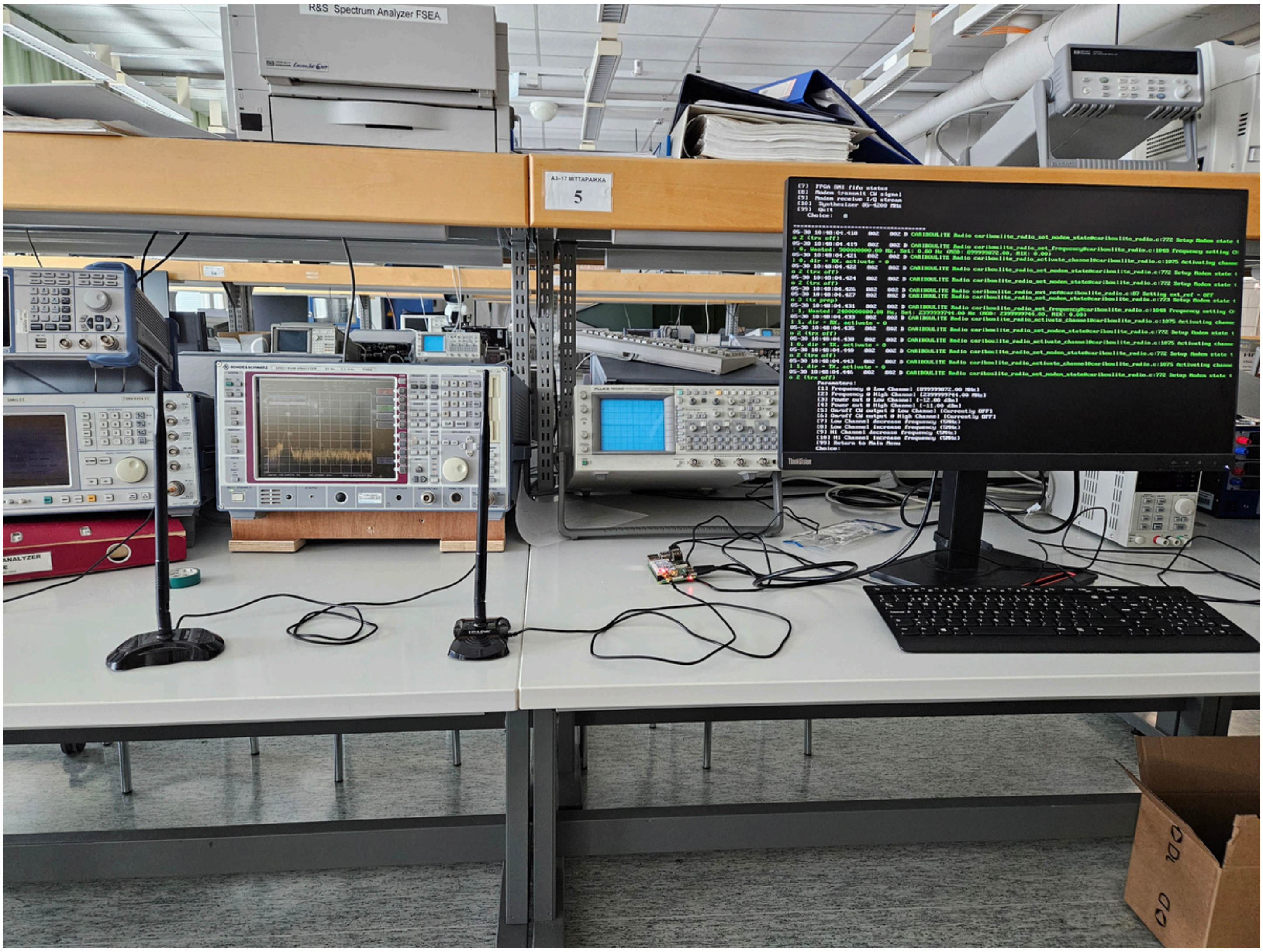
TROUBLESHOOTING

During the installation process i encountered a few issues(not in particular order):

- SDR++ software is highly delicate and sensitive to break. Especially the server+client part. I noticed that “cold-starting” the server, the software did not find SoapySDR-module or CaribouLite. Here is how i solved the problem in few steps:
    - First using the “SoapySDRUtil --find”-command on Raspberry Pi, if the command found the CaribouLite and its channels, you can move on to the next step
    - Second open up the SDR++ GUI on Pi and select the SoapySDR-module and CaribouLite. Test everything that the radio channels do show in the software as a waterfall. Close the software.
    - Now you should be ready to start the server on Pi. Use the command “ ***./sdrpp -s -p 5259 --autostart*** ” . This will force the software to start up in server mode on port 5259 and launch the server automatically. Now you should have correctly configured SDR++ server running on Pi.
    - Finally, you can now open the SDR++ GUI on another computer and select an SDR++ server from the menu on the left side.
- Building and configuring the SDR++ software takes time. Deleting and re-building the build-directory on the main directory helps a lot. Also if you encounter a module problems (like me) you should check the ***CMakeLists.txt-file***.

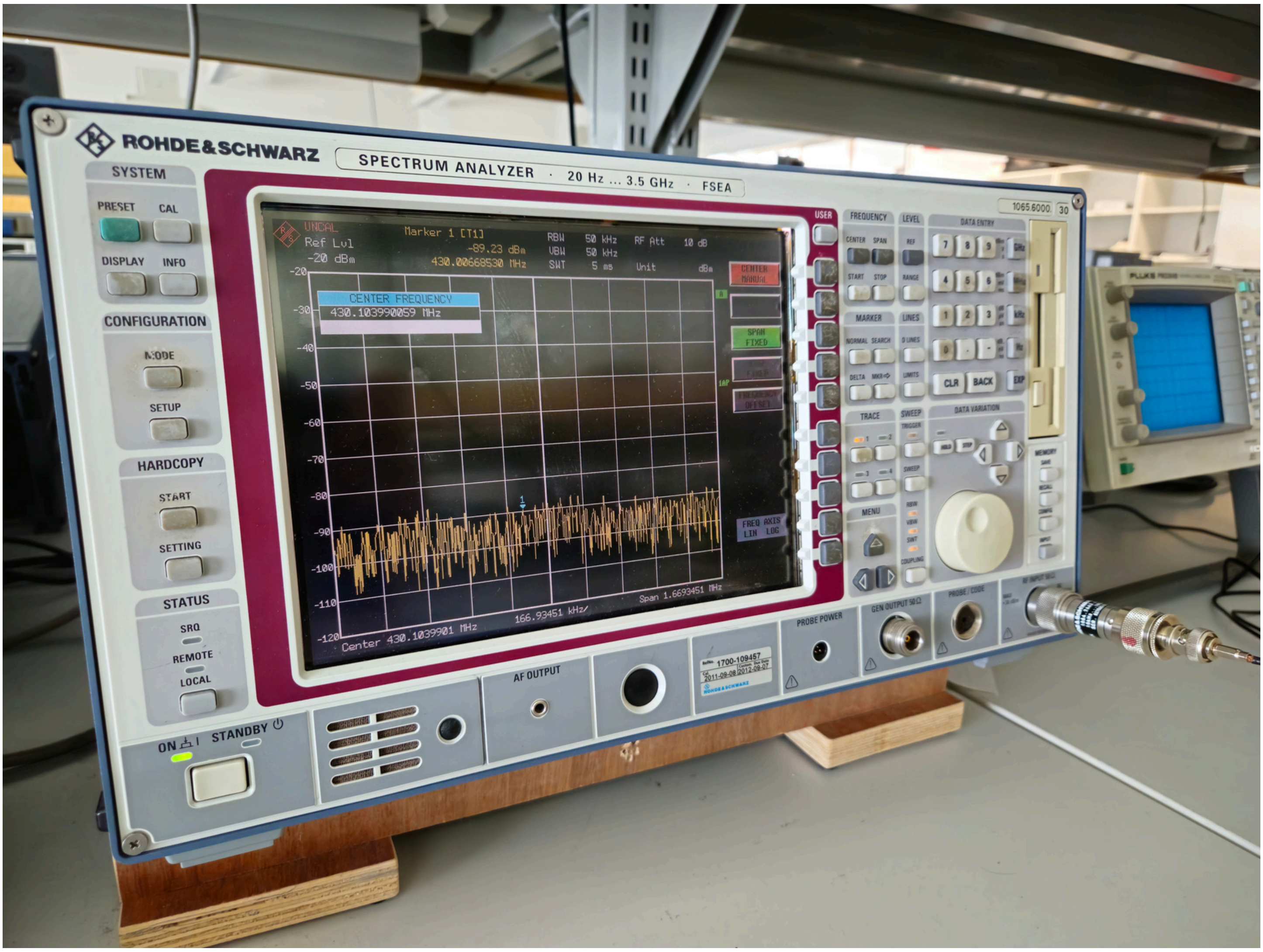
CARIBOULITE'S TX-CHANNEL

Next on the mission was to test the CaribouLite's TX-channel(transmission abilities).

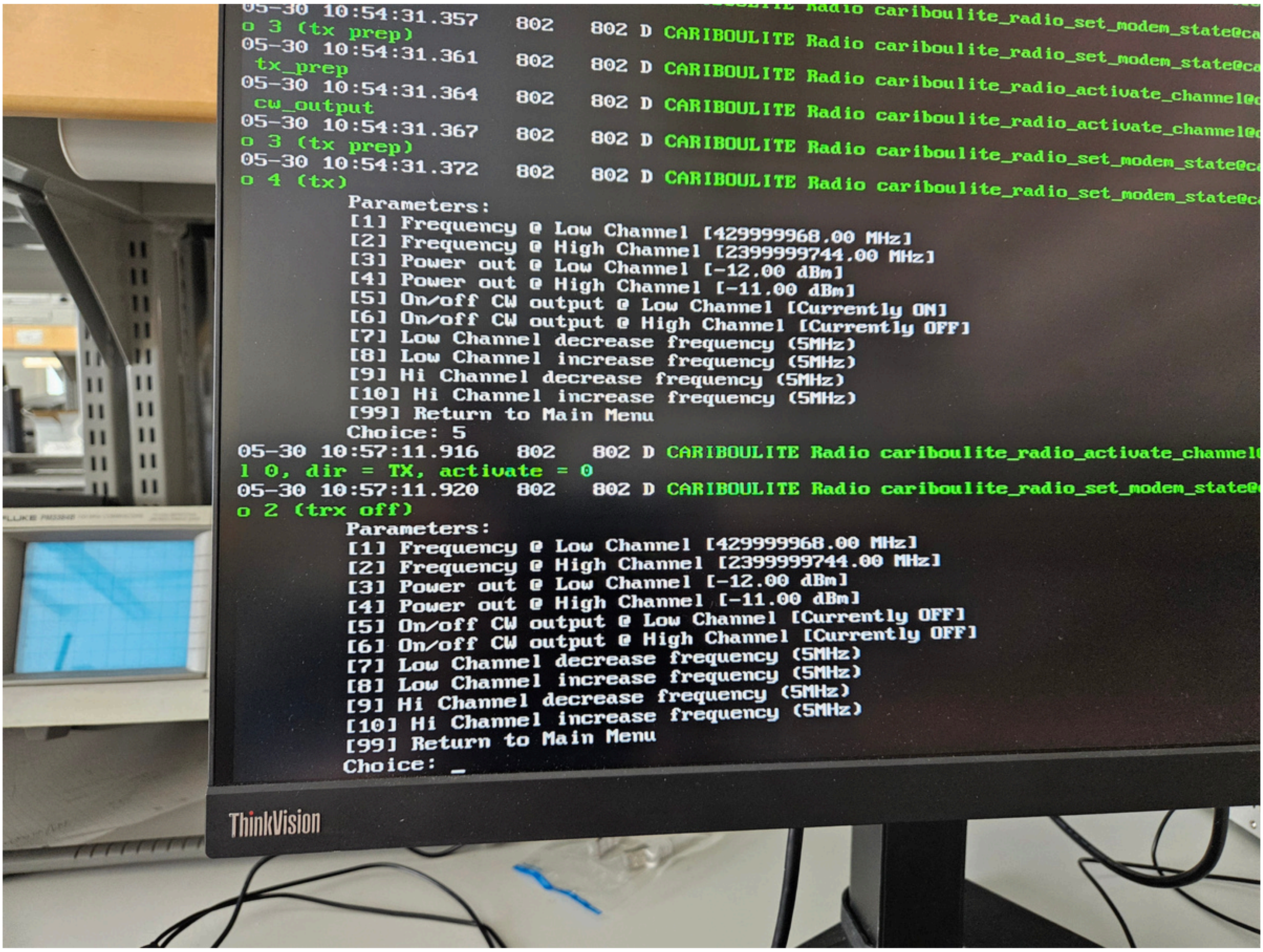


```
[7]  FPGA SMI fifo status
[8]  Modem transmit CW signal
[9]  Modem receive I/Q stream
[10]  Synthesizer 85-4200 MHz
[99]  Quit
   Choice:   8

==================================
05-30 10:48:04.418   802   802 D CARIBOULITE Radio cariboulite_radio_set_modem_state@cariboulite_radio.c:772 Setup Modem state t
o 2 (trx off)
05-30 10:48:04.419   802   802 D CARIBOULITE Radio cariboulite_radio_set_frequency@cariboulite_radio.c:1048 Frequency setting CH
: 0, Wanted: 900000000.00 Hz, Set: 0.00 Hz (MOD: 899999872.00, MIX: 0.00)
05-30 10:48:04.421   802   802 D CARIBOULITE Radio cariboulite_radio_activate_channel@cariboulite_radio.c:1075 Activating channe
l 0, dir = RX, activate = 0
05-30 10:48:04.422   802   802 D CARIBOULITE Radio cariboulite_radio_set_modem_state@cariboulite_radio.c:772 Setup Modem state t
o 2 (trx off)
05-30 10:48:04.424   802   802 D CARIBOULITE Radio cariboulite_radio_set_modem_state@cariboulite_radio.c:772 Setup Modem state t
o 2 (trx off)
05-30 10:48:04.426   802   802 D CARIBOULITE Radio cariboulite_radio_ext_ref@cariboulite_radio.c:87 Setting ext_ref = OFF
05-30 10:48:04.427   802   802 D CARIBOULITE Radio cariboulite_radio_set_modem_state@cariboulite_radio.c:773 Setup Modem state t
o 3 (tx prep)
05-30 10:48:04.431   802   802 D CARIBOULITE Radio cariboulite_radio_set_frequency@cariboulite_radio.c:1048 Frequency setting CH
: 1, Wanted: 2400000000.00 Hz, Set: 2399999744.00 Hz (MOD: 2399999744.00, MIX: 0.00)
05-30 10:48:04.433   802   802 D CARIBOULITE Radio cariboulite_radio_activate_channel@cariboulite_radio.c:1075 Activating channe
l 1, dir = RX, activate = 0
05-30 10:48:04.435   802   802 D CARIBOULITE Radio cariboulite_radio_set_modem_state@cariboulite_radio.c:772 Setup Modem state t
o 2 (trx off)
05-30 10:48:04.438   802   802 D CARIBOULITE Radio cariboulite_radio_activate_channel@cariboulite_radio.c:1075 Activating channe
l 0, dir = TX, activate = 0
05-30 10:48:04.440   802   802 D CARIBOULITE Radio cariboulite_radio_set_modem_state@cariboulite_radio.c:772 Setup Modem state t
o 2 (trx off)
05-30 10:48:04.443   802   802 D CARIBOULITE Radio cariboulite_radio_activate_channel@cariboulite_radio.c:1075 Activating channe
l 1, dir = TX, activate = 0
05-30 10:48:04.446   802   802 D CARIBOULITE Radio cariboulite_radio_set_modem_state@cariboulite_radio.c:772 Setup Modem state t
o 2 (trx off)
      Parameters:
      [1] Frequency @ Low Channel [899999872.00 MHz]
      [2] Frequency @ High Channel [2399999744.00 MHz]
      [3] Power out @ Low Channel [-12.00 dBm]
      [4] Power out @ High Channel [-11.00 dBm]
      [5] On/off CW output @ Low Channel [Currently OFF]
      [6] On/off CW output @ High Channel [Currently OFF]
      [7] Low Channel decrease frequency (5MHz)
      [8] Low Channel increase frequency (5MHz)
      [9] Hi Channel decrease frequency (5MHz)
      [10] Hi Channel increase frequency (5MHz)
      [99] Return to Main Menu
      Choice:
```

The test-setup above was quite simple. One spectrum analyzer and the CaribouLite-RaspberryPi combo with antennas.



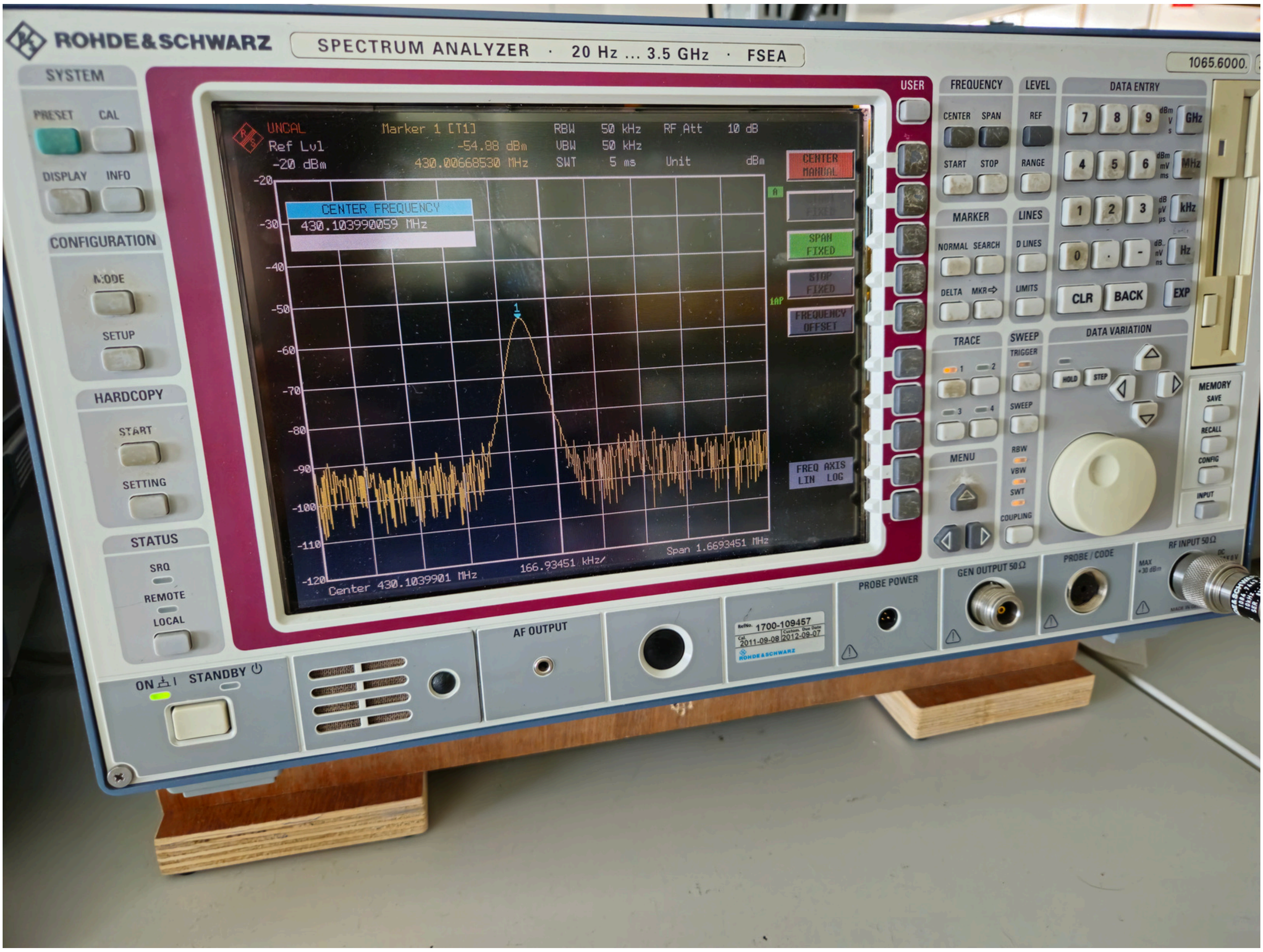
The spectrum analyzer was set to a frequency of 430 MHz, as this was considered the safest option for testing CaribouLite's TX-channel. This frequency falls within a low-risk UHF band commonly used for amateur radio and low-power applications, minimizing the likelihood of interference with critical communication systems. The small blue arrow was set to a 430MHz in the center.



```
05-30 10:54:31.357   802   802 D CARIBOULITE Radio cariboulite_radio_set_modem_state@ca
o 3 (tx prep)
05-30 10:54:31.361   802   802 D CARIBOULITE Radio cariboulite_radio_set_modem_state@ca
 tx_prep
05-30 10:54:31.364   802   802 D CARIBOULITE Radio cariboulite_radio_activate_channel@c
 cw_output
05-30 10:54:31.367   802   802 D CARIBOULITE Radio cariboulite_radio_activate_channel@c
o 3 (tx prep)
05-30 10:54:31.372   802   802 D CARIBOULITE Radio cariboulite_radio_set_modem_state@ca
o 4 (tx)
        Parameters:
        [1] Frequency @ Low Channel [429999968.00 MHz]
        [2] Frequency @ High Channel [2399999744.00 MHz]
        [3] Power out @ Low Channel [-12.00 dBm]
        [4] Power out @ High Channel [-11.00 dBm]
        [5] On/off CW output @ Low Channel [Currently ON]
        [6] On/off CW output @ High Channel [Currently OFF]
        [7] Low Channel decrease frequency (5MHz)
        [8] Low Channel increase frequency (5MHz)
        [9] Hi Channel decrease frequency (5MHz)
        [10] Hi Channel increase frequency (5MHz)
        [99] Return to Main Menu
        Choice: 5
05-30 10:57:11.916   802   802 D CARIBOULITE Radio cariboulite_radio_activate_channel(
l 0, dir = TX, activate = 0
05-30 10:57:11.920   802   802 D CARIBOULITE Radio cariboulite_radio_set_modem_state@
o 2 (trx off)
        Parameters:
        [1] Frequency @ Low Channel [429999968.00 MHz]
        [2] Frequency @ High Channel [2399999744.00 MHz]
        [3] Power out @ Low Channel [-12.00 dBm]
        [4] Power out @ High Channel [-11.00 dBm]
        [5] On/off CW output @ Low Channel [Currently OFF]
        [6] On/off CW output @ High Channel [Currently OFF]
        [7] Low Channel decrease frequency (5MHz)
        [8] Low Channel increase frequency (5MHz)
        [9] Hi Channel decrease frequency (5MHz)
        [10] Hi Channel increase frequency (5MHz)
        [99] Return to Main Menu
        Choice: _
```

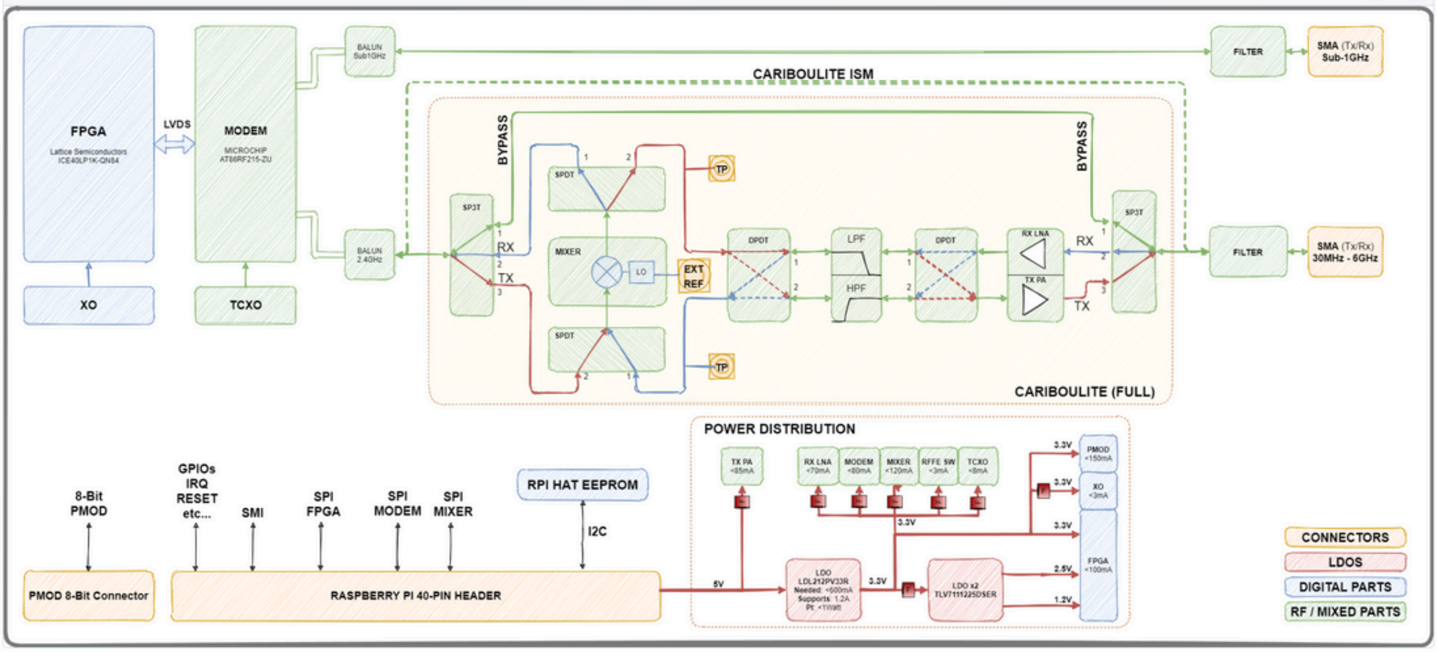
CaribouLite provides a built-in diagnostic tool named cariboulite_test_app, which can be executed from the command line. During testing, I selected the appropriate transmission test mode. The tool enables manual configuration of transmission parameters such as operating frequency, output gain, and other signal characteristics.

I selected the Low Channel Frequency (Option 1) and entered 430 MHz as the desired frequency. After that, I chose option 5, which enables continuous wave (CW) output on the low channel.



And it works! The CaribouLite successfully transmitted a CW output signal, as clearly depicted in the image.

CARIBOULITE’S TX-CHANNEL

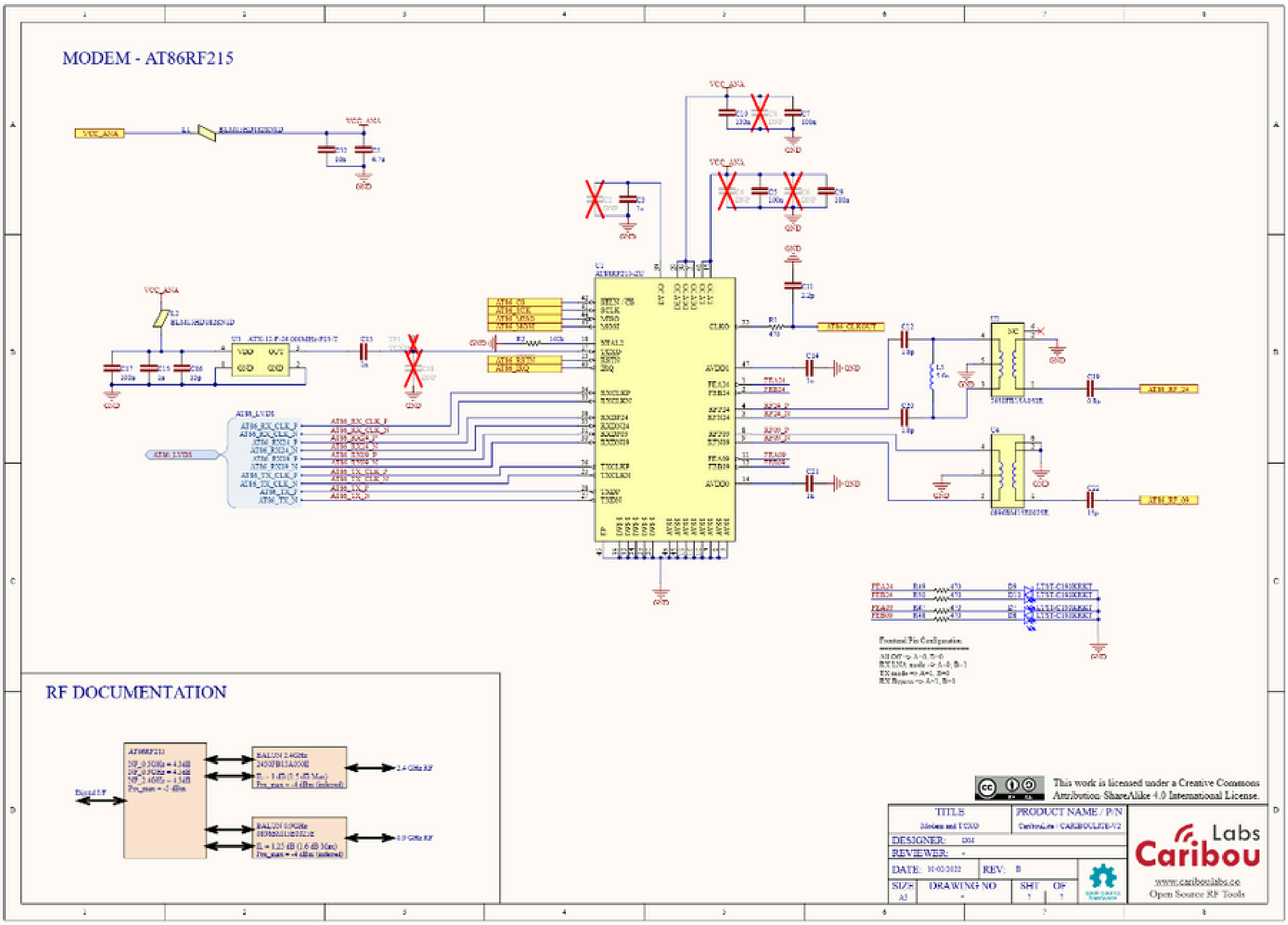


CaribouLite SDR System Diagram

Next on the job was to figure out how the modem and FPGA works on CaribouLite.

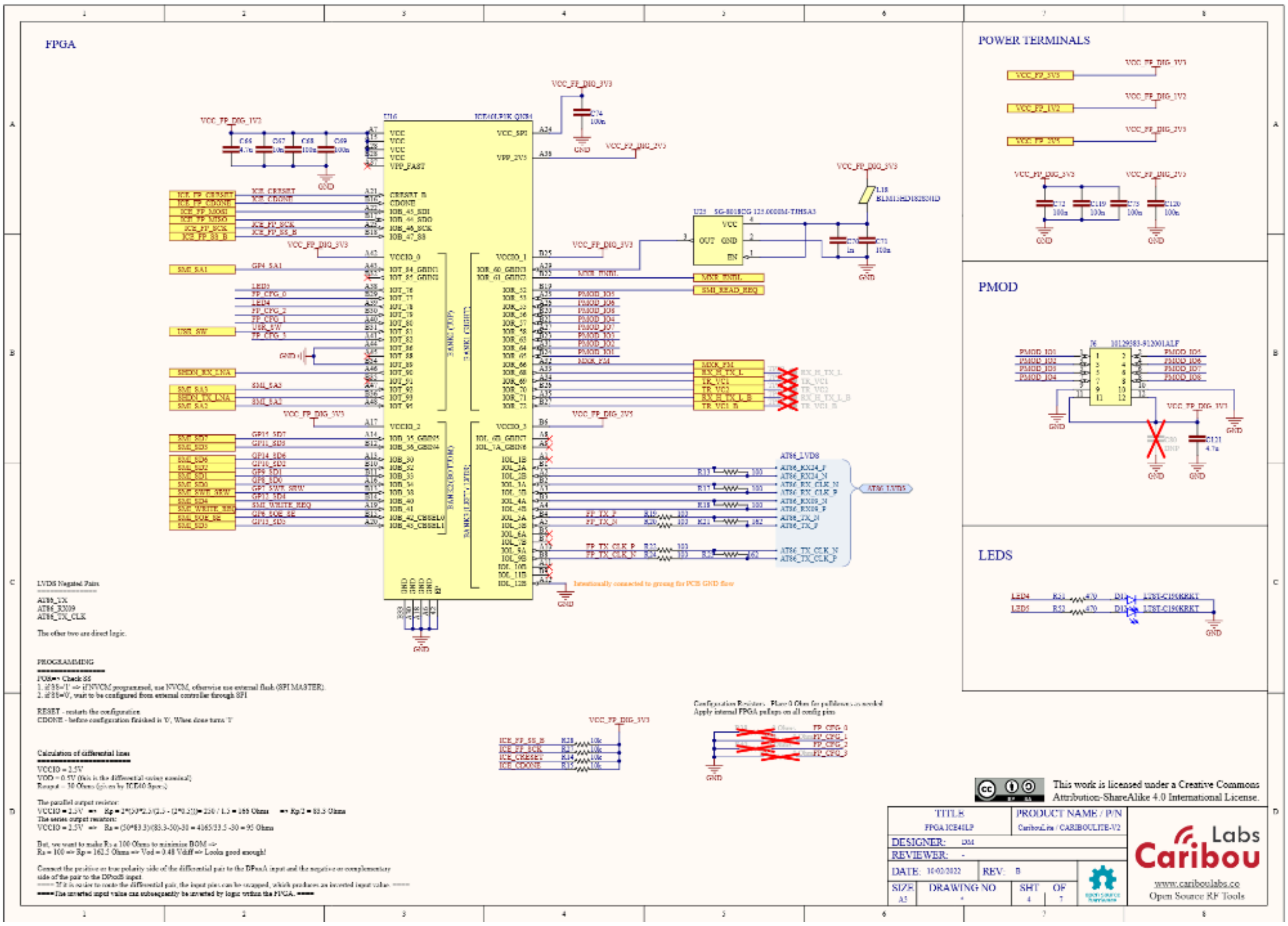
Specs about the modem:

- Model: Microchip AT86RF215-ZU
    - This is a dual-band RF transceiver that supports both sub-1 GHz (389.5–510 MHz and 779–1020 MHz) and 2.4 GHz bands. It supports multiple modulation schemes such as FSK, OFDM, and O-QPSK. The chip integrates both a baseband modem and an RF front end and offers sensitivity down to -123 dBm.
    - Direct interface to the FPGA
    - Digital baseband I/Q signal processing



Specs about the FPGA:

- Lattice Semiconductors ICE40LP1K-QN84
  - A small, low-power FPGA that serves as the bridge between the Raspberry Pi and the AT86RF215 transceiver. It handles timing, buffering, and digital interfacing, as well as custom logic that may assist with signal routing or format conversion.
  - Manages the data/control flow between the Pi and AT86RF215-ZU



LVDS Negated Pairs
================

AT86_TX
AT86_RX09
AT86_TX_CLK

The other two are direct logic.

PROGRAMMING
================

POR => Check SS
1. if SS='1' => if NVCM programmed, use NVCM, otherwise use external flash (SPI MASTER).
2. if SS='0', wait to be configured from external controller through SPI

RESET - restarts the configuration
CDONE - before configuration finished is '0', When done turns '1'

Calculation of differential lines
=======================

VCCIO = 2.5V
VOD = 0.5V (this is the differential swing nominal)
Routput = 30 Ohms (given by ICE40 Specs)

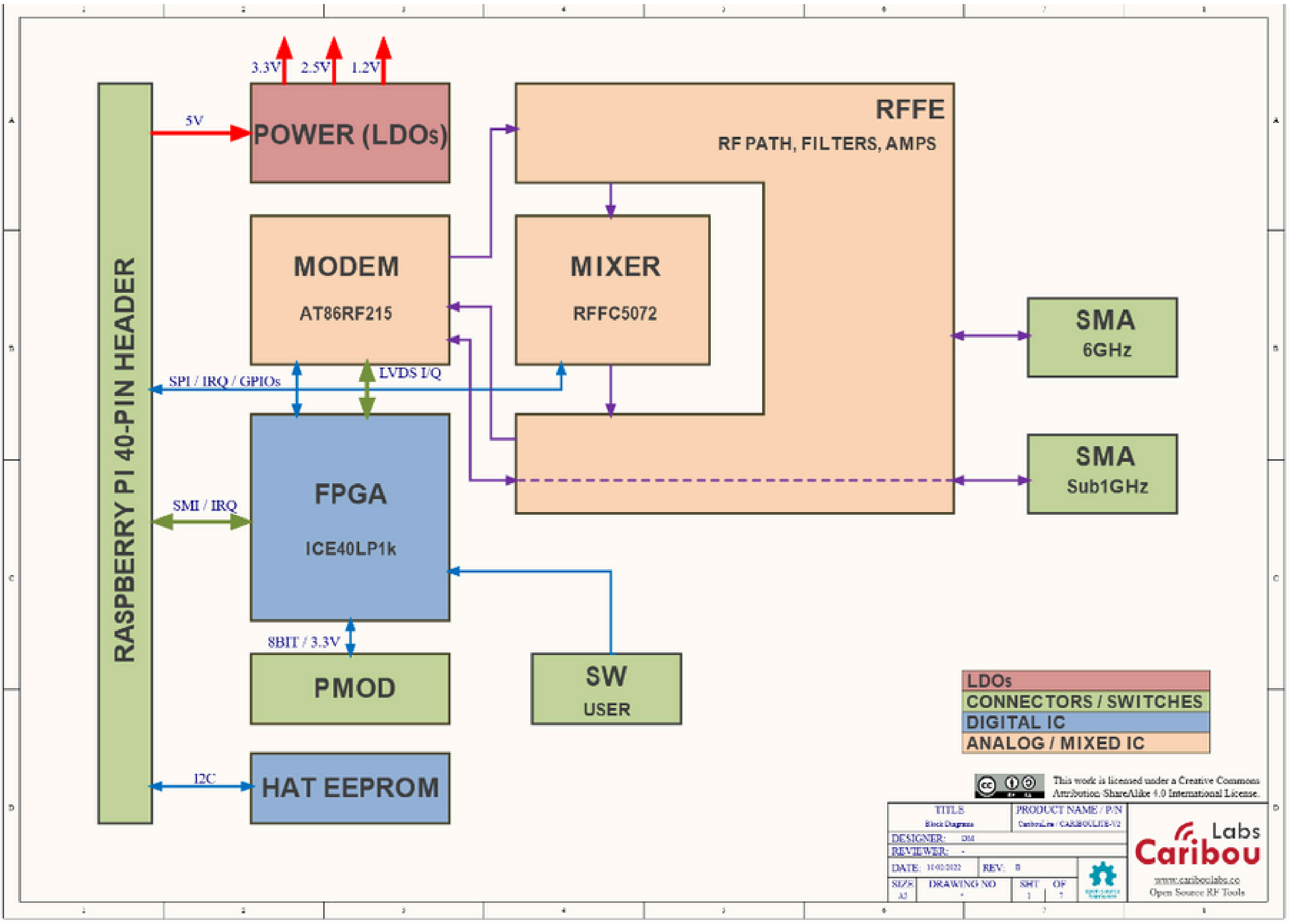
The parallel output resistor:
VCCIO = 2.5V => Rp = 2*(50*2.5/(2.5 - (2*0.5)))= 250 / 1.5 = 166 Ohms => Rp/2 = 83.3 Ohms
The series output resistors:
VCCIO = 2.5V => Rs = (50*83.3)/(83.3-50)-30 = 4165/33.3 -30 = 95 Ohm

But, we want to make Rs a 100 Ohms to minimize BOM =>
Rs = 100 => Rp = 162.5 Ohms => Vod = 0.48 Vdiff => Looks good enough!

Connect the positive or true polarity side of the differential pair to the DPxxA input and the negative or complementary side of the pair to the DPxxB input.
===== If it is easier to route the differential pair, the input pins can be swapped, which produces an inverted input value. =====
=====The inverted input value can subsequently be inverted by logic within the FPGA. =====

TITLE	PRODUCT NAME / P/N
FPGA ICE40LP	CaribouLite / CARIBOULITE-V2
DESIGNER: DM	
REVIEWER: -	
DATE: 10/02/2022	REV: B
SIZE A3	DRAWING NO -	SHT 4	OF 7

Caribou Labs
www.cariboulabs.co
Open Source RF Tools



- 🟥 LDOs
- 🟩 CONNECTORS / SWITCHES
- 🟦 DIGITAL IC
- 🟧 ANALOG / MIXED IC

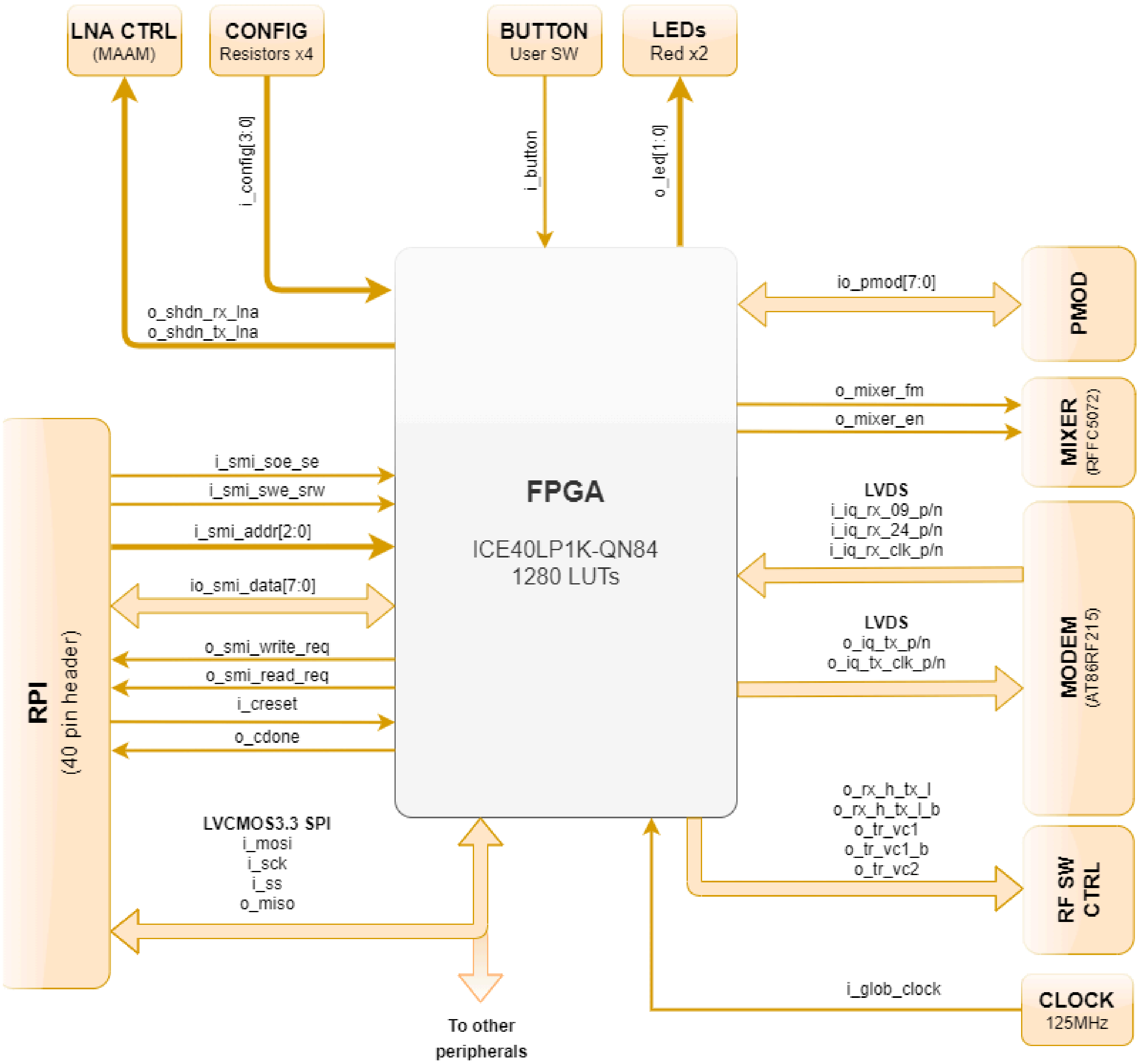
TITLE	PRODUCT NAME / P/N
Block Diagram	CaribouLite / CARIBOULITE-V2
DESIGNER: DM	
REVIEWER: -	
DATE: 10/02/2022	REV: B
SIZE A3	DRAWING NO -	SHT 1	OF 7

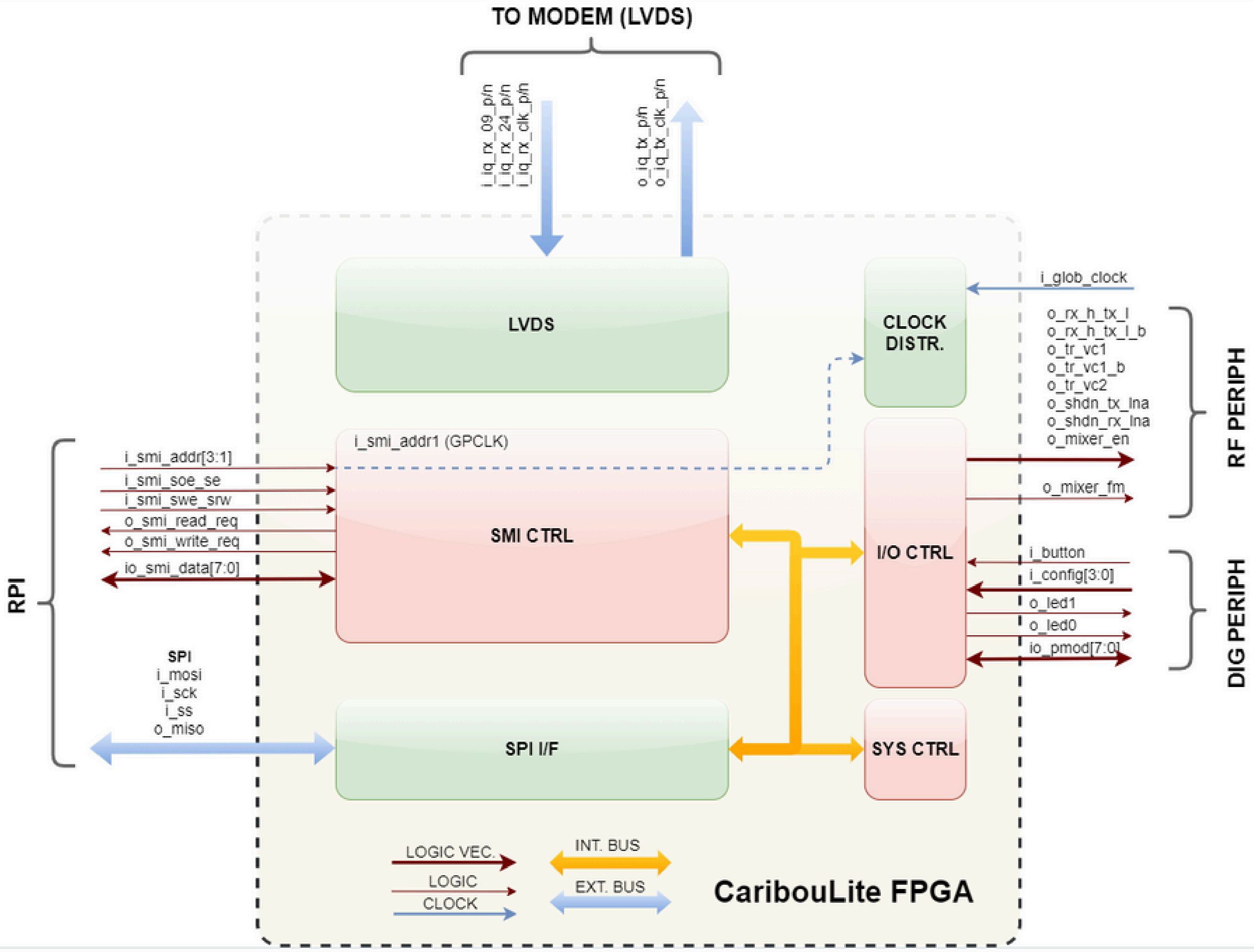
Caribou Labs
www.cariboulabs.co
Open Source RF Tools

All of the pictures can be found on CaribouLite’s Github-page on the hardware-section!

FPGA Firmware Comparison: CaribouLite Original vs. Matteoserva’s Modified Version

…was a bust. There was nothing to compare as the firmware is the identical in both. I found two pictures about FPGA in both repositories.

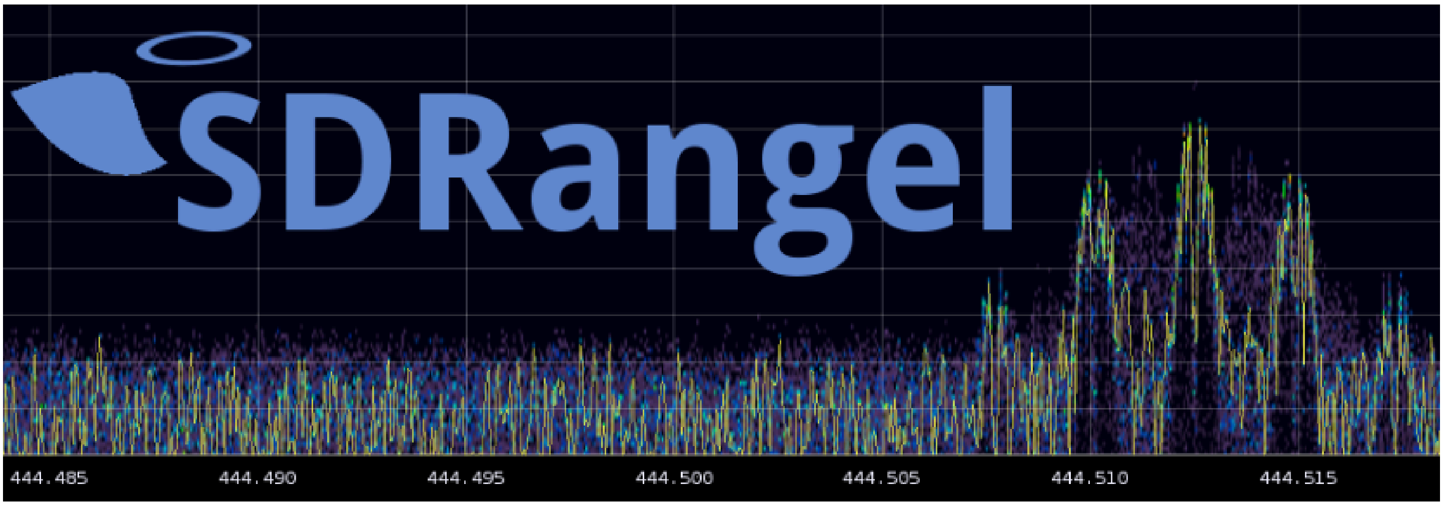

LNA CTRL
(MAAM)
CONFIG
Resistors x4
BUTTON
User SW
LEDs
Red x2
i_config[3:0]
i_button
o_led[1:0]
o_shdn_rx_lna
o_shdn_tx_lna
io_pmod[7:0]
PMOD
o_mixer_fm
o_mixer_en
MIXER
(RFFC5072)
RPI
(40 pin header)
i_smi_soe_se
i_smi_swe_srw
i_smi_addr[2:0]
io_smi_data[7:0]
o_smi_write_req
o_smi_read_req
i_creset
o_cdone
FPGA
ICE40LP1K-QN84
1280 LUTs
LVDS
i_iq_rx_09_p/n
i_iq_rx_24_p/n
i_iq_rx_clk_p/n
LVDS
o_iq_tx_p/n
o_iq_tx_clk_p/n
MODEM
(AT86RF215)
LVCMOS3.3 SPI
i_mosi
i_sck
i_ss
o_miso
o_rx_h_tx_l
o_rx_h_tx_l_b
o_tr_vc1
o_tr_vc1_b
o_tr_vc2
RF SW
CTRL
i_glob_clock
CLOCK
125MHz
To other
peripherals


TO MODEM (LVDS)
i_iq_rx_09_p/n
i_iq_rx_24_p/n
i_iq_rx_clk_p/n
o_iq_tx_p/n
o_iq_tx_clk_p/n
LVDS
CLOCK DISTR.
i_glob_clock
o_rx_h_tx_l
o_rx_h_tx_l_b
o_tr_vc1
o_tr_vc1_b
o_tr_vc2
o_shdn_tx_lna
o_shdn_rx_lna
o_mixer_en
o_mixer_fm
RF PERIPH
i_smi_addr1 (GPCLK)
i_smi_addr[3:1]
i_smi_soe_se
i_smi_swe_srw
o_smi_read_req
o_smi_write_req
io_smi_data[7:0]
SMI CTRL
I/O CTRL
i_button
i_config[3:0]
o_led1
o_led0
io_pmod[7:0]
DIG PERIPH
RPI
SPI
i_mosi
i_sck
i_ss
o_miso
SPI I/F
SYS CTRL
LOGIC VEC.
LOGIC
CLOCK
INT. BUS
EXT. BUS
CaribouLite FPGA

CARIBOULITE’S TX-CHANNEL

Transmission chain trial run using SDRAngel

Having previously verified that the CaribouLite is capable of transmitting basic CW signals using CaribouLites own test app, I proceeded to experiment with sending custom signals using SDRAngel.

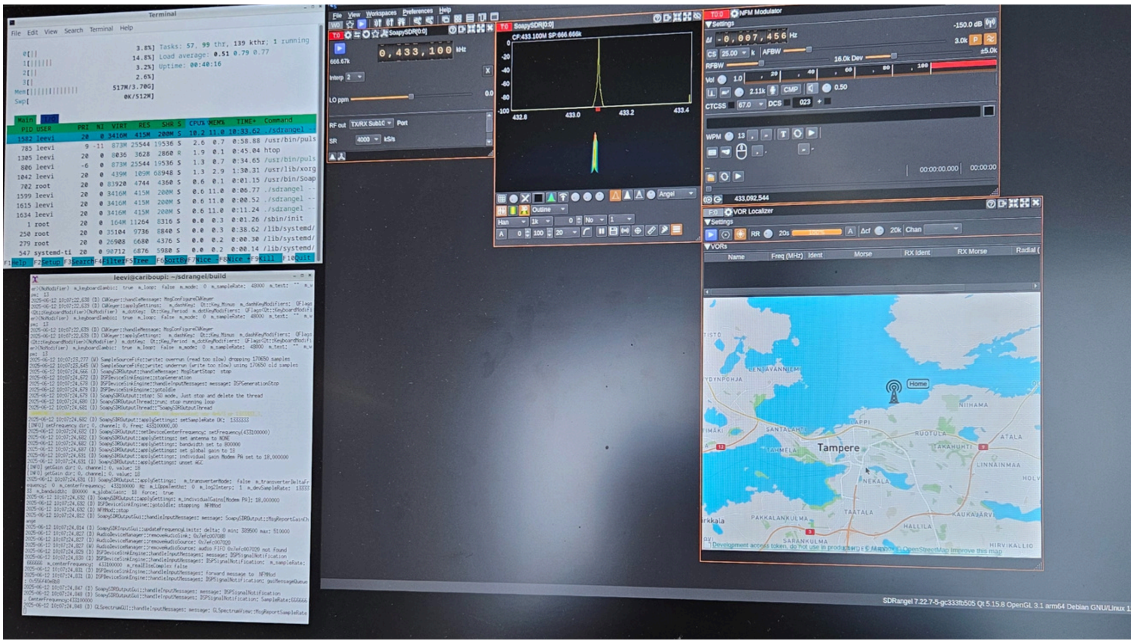


Yet another SDR-based software? **Yes**.

Earlier, while using the SDR++ software, I confirmed that the CaribouLite is capable of effectively receiving radio frequencies. Unlike SDR++, which is receive-only, SDRAngel supports both receiving and transmitting signals.

You can find the installation guides on Github here! The process is quite long but it is really heavy software.

After the succesfull installation, i chose the SoapySDR module with CaribouLite’s S1G-channel. This channel supports the ISM frequency band, making it ideal for testing purposes.



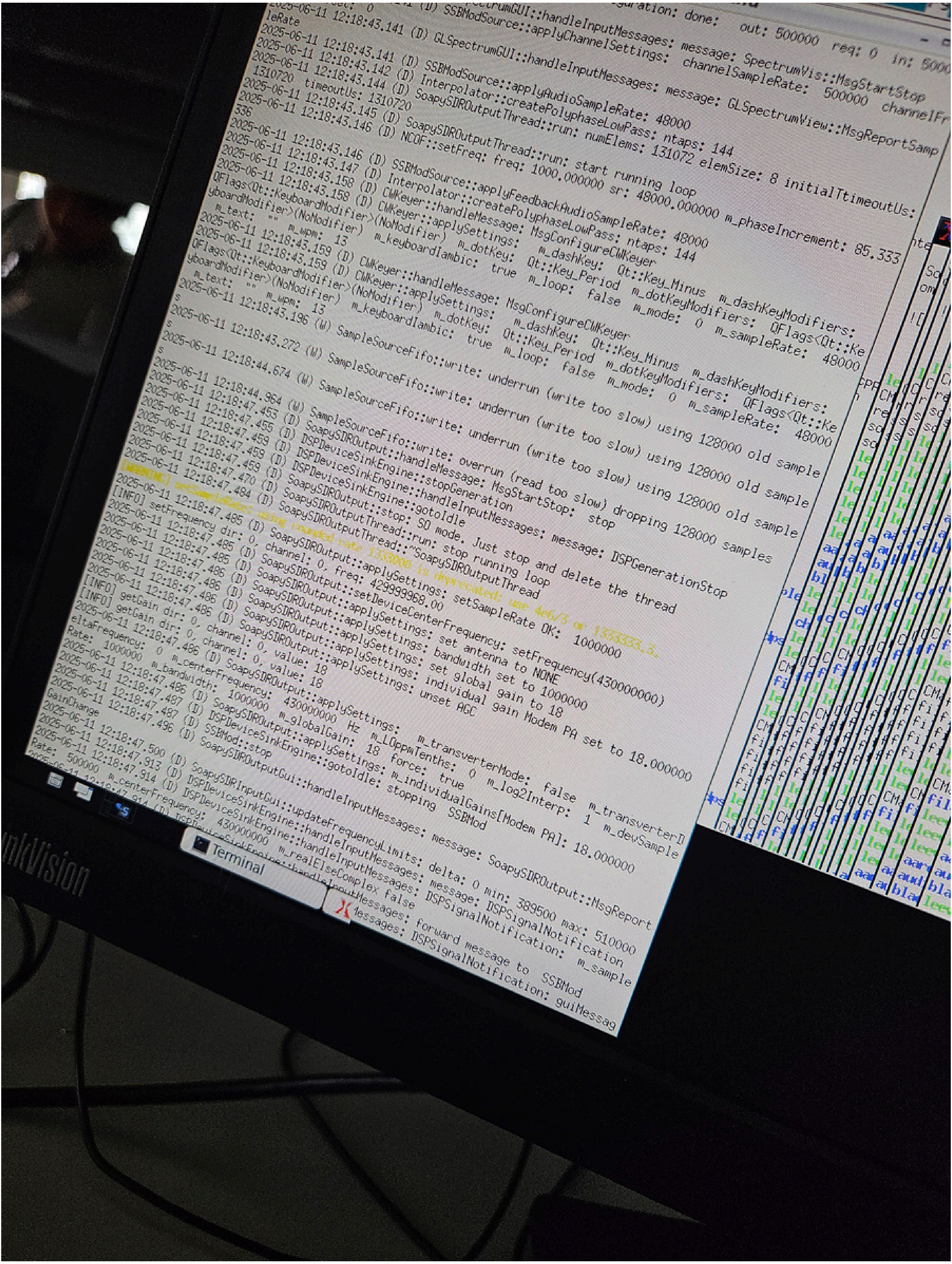
```
Terminal
File  Edit  View  Search  Terminal  Help

    0[||                          3.8%]  Tasks: 57, 99 thr, 139 kthr; 1 running
    1[||||||                     14.8%]  Load average: 0.51 0.79 0.77
    2[||                          3.2%]  Uptime: 00:40:16
    3[|                           2.6%]
  Mem[|||||||||||||        517M/3.70G]
  Swp[                        0K/512M]

  Main  I/O
    PID USER       PRI  NI  VIRT   RES   SHR S  CPU% MEM%   TIME+  Command
   1582 leevi       20   0 3416M  415M  200M S  10.2 11.0 10:33.62 ./sdrangel --
    785 leevi        9 -11  873M 25544 19536 S   2.6  0.7  0:58.88 /usr/bin/puls
   1305 leevi       20   0  8036  3628  2860 R   1.9  0.1  0:45.04 htop
    806 leevi       -6   0  873M 25544 19536 S   1.3  0.7  0:34.65 /usr/bin/puls
   1042 leevi       20   0  439M  109M 68948 S   1.3  2.9  1:30.31 /usr/lib/xorg
    702 root        20   0 83920  4744  4360 S   0.6  0.1  0:01.15 /usr/bin/Soap
   1599 leevi       20   0 3416M  415M  200M S   0.6 11.0  0:06.77 ./sdrangel --
   1615 leevi       20   0 3416M  415M  200M S   0.6 11.0  0:00.52 ./sdrangel --
   1634 leevi       20   0 3416M  415M  200M S   0.6 11.0  0:11.24 ./sdrangel --
      1 root        20   0  164M 11264  8316 S   0.0  0.3  0:01.26 /sbin/init
    250 root        20   0 35104  9736  8840 S   0.0  0.3  0:38.62 /lib/systemd/
    279 root        20   0 26908  6680  4376 S   0.0  0.2  0:00.30 /lib/systemd/
    547 systemd-ti  20   0 90712  6876  5980 S   0.0  0.2  0:00.14 /lib/systemd/
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

```
leevi@caribоupi: ~/sdrangel/build
er>(NoModifier) m_keyboardIambic: true m_loop: false m_mode: 0 m_sampleRate: 48000 m_text: "" m_w
pm: 13
2025-06-12 10:07:22.638 (D) CWKeyer::handleMessage: MsgConfigureCWKeyer
2025-06-12 10:07:22.639 (D) CWKeyer::applySettings:  m_dashKey: Qt::Key_Minus m_dashKeyModifiers: QFlags
<Qt::KeyboardModifier>(NoModifier) m_dotKey: Qt::Key_Period m_dotKeyModifiers: QFlags<Qt::KeyboardModifi
er>(NoModifier) m_keyboardIambic: true m_loop: false m_mode: 0 m_sampleRate: 48000 m_text: "" m_w
pm: 13
2025-06-12 10:07:22.639 (D) CWKeyer::handleMessage: MsgConfigureCWKeyer
2025-06-12 10:07:22.639 (D) CWKeyer::applySettings:  m_dashKey: Qt::Key_Minus m_dashKeyModifiers: QFlags
<Qt::KeyboardModifier>(NoModifier) m_dotKey: Qt::Key_Period m_dotKeyModifiers: QFlags<Qt::KeyboardModifi
er>(NoModifier) m_keyboardIambic: true m_loop: false m_mode: 0 m_sampleRate: 48000 m_text: "" m_w
pm: 13
2025-06-12 10:07:23.277 (W) SampleSourceFifo::write: overrun (read too slow) dropping 170650 samples
2025-06-12 10:07:23.645 (W) SampleSourceFifo::write: underrun (write too slow) using 170650 old samples
2025-06-12 10:07:24.666 (D) SoapySDROutput::handleMessage: MsgStartStop:  stop
2025-06-12 10:07:24.672 (D) DSPDeviceSinkEngine::stopGeneration
2025-06-12 10:07:24.678 (D) DSPDeviceSinkEngine::handleInputMessages: message: DSPGenerationStop
2025-06-12 10:07:24.679 (D) DSPDeviceSinkEngine::gotoIdle
2025-06-12 10:07:24.679 (D) SoapySDROutput::stop: SO mode. Just stop and delete the thread
2025-06-12 10:07:24.680 (D) SoapySDROutputThread::run: stop running loop
2025-06-12 10:07:24.681 (D) SoapySDROutputThread::~SoapySDROutputThread
[illegible]
2025-06-12 10:07:24.682 (D) SoapySDROutput::applySettings: setSampleRate OK:  1333333
[INFO] setFrequency dir: 0, channel: 0, freq: 433100000.00
2025-06-12 10:07:24.682 (D) SoapySDROutput::setDeviceCenterFrequency: setFrequency(433100000)
2025-06-12 10:07:24.682 (D) SoapySDROutput::applySettings: set antenna to NONE
2025-06-12 10:07:24.682 (D) SoapySDROutput::applySettings: bandwidth set to 800000
2025-06-12 10:07:24.687 (D) SoapySDROutput::applySettings: set global gain to 18
2025-06-12 10:07:24.691 (D) SoapySDROutput::applySettings: individual gain Modem PA set to 18.000000
2025-06-12 10:07:24.691 (D) SoapySDROutput::applySettings: unset AGC
[INFO] getGain dir: 0, channel: 0, value: 18
[INFO] getGain dir: 0, channel: 0, value: 18
2025-06-12 10:07:24.691 (D) SoapySDROutput::applySettings:  m_transverterMode: false m_transverterDeltaFr
equency: 0 m_centerFrequency: 433100000 Hz m_LOppmTenths: 0 m_log2Interp: 1 m_devSampleRate: 13333
33 m_bandwidth: 800000 m_globalGain: 18 force: true
2025-06-12 10:07:24.692 (D) SoapySDROutput::applySettings: m_individualGains[Modem PA]: 18.000000
2025-06-12 10:07:24.692 (D) DSPDeviceSinkEngine::gotoIdle: stopping  NFMMod
2025-06-12 10:07:24.692 (D) NFMMod::stop
2025-06-12 10:07:24.812 (D) SoapySDROutputGui::handleInputMessages: message: SoapySDROutput::MsgReportGainCh
ange
2025-06-12 10:07:24.814 (D) SoapySDRInputGui::updateFrequencyLimits: delta: 0 min: 389500 max: 510000
2025-06-12 10:07:24.827 (D) AudioDeviceManager::removeAudioSink: 0x7efc007088
2025-06-12 10:07:24.827 (D) AudioDeviceManager::removeAudioSource: 0x7efc007020
2025-06-12 10:07:24.827 (W) AudioDeviceManager::removeAudioSource: audio FIFO 0x7efc007020 not found
2025-06-12 10:07:24.829 (D) DSPDeviceSinkEngine::handleInputMessages: message: DSPSignalNotification
2025-06-12 10:07:24.830 (D) DSPDeviceSinkEngine::handleInputMessages: DSPSignalNotification:  m_sampleRate:
 666666  m_centerFrequency:  433100000  m_realElseComplex false
2025-06-12 10:07:24.831 (D) DSPDeviceSinkEngine::handleInputMessages: forward message to  NFMMod
2025-06-12 10:07:24.831 (D) DSPDeviceSinkEngine::handleInputMessages: DSPSignalNotification: guiMessageQueue
: 0x556f40e8b8
2025-06-12 10:07:24.847 (D) SoapySDROutputGui::handleInputMessages: message: DSPSignalNotification
2025-06-12 10:07:24.848 (D) SoapySDROutputGui::handleInputMessages: DSPSignalNotification: SampleRate:666666
, CenterFrequency:433100000
2025-06-12 10:07:24.848 (D) GLSpectrumGUI::handleInputMessages: message: GLSpectrumView::MsgReportSampleRate
```

And there it was! On my screen, I had HTOP, Xterm, and SDRAngel all running. After selecting the correct channel, I tuned to the desired frequency and began exploring what the software is truly capable of. I added the NFM module directly to the S1G channel and hit the “Run” command—but nothing happened.

I also uploaded a WAV file directly to the Raspberry Pi and attempted to modulate it using SDRAngel.

I was able to see a signal spectrum on the SDRAngel but did not receive any signals on the spectrum analyzer. In other words, this confirmed that no signals were being transmitted from the CaribouLite.



```
        et: 0          ...ctrumGUI::handleInputMessages: message: SpectrumVis::MsgStartStop   ...guration: done:   out: 500000  req: 0  in: 5000
2025-06-11 ... (D) SSBModSource::applyChannelSettings:  channelSampleRate:  500000  channelFr
leRate
2025-06-11 12:18:43.141 (D) GLSpectrumGUI::handleInputMessages: message: GLSpectrumView::MsgReportSamp
2025-06-11 12:18:43.141 (D) SSBModSource::applyAudioSampleRate: 48000
2025-06-11 12:18:43.142 (D) Interpolator::createPolyphaseLowPass: ntaps: 144
2025-06-11 12:18:43.144 (D) SoapySDROutputThread::run: numElems: 131072 elemSize: 8 initialTtimeoutUs:
1310720  timeoutUs: 1310720
2025-06-11 12:18:43.145 (D) SoapySDROutputThread::run: start running loop
2025-06-11 12:18:43.146 (D) NCOF::setFreq: freq: 1000.000000 sr: 48000.000000 m_phaseIncrement: 85.333
336
2025-06-11 12:18:43.146 (D) SSBModSource::applyFeedbackAudioSampleRate: 48000
2025-06-11 12:18:43.147 (D) Interpolator::createPolyphaseLowPass: ntaps: 144
2025-06-11 12:18:43.158 (D) CWKeyer::handleMessage: MsgConfigureCWKeyer
2025-06-11 12:18:43.158 (D) CWKeyer::applySettings:  m_dashKey:  Qt::Key_Minus  m_dashKeyModifiers:
QFlags<Qt::KeyboardModifier>(NoModifier)  m_dotKey:  Qt::Key_Period  m_dotKeyModifiers:  QFlags<Qt::Ke
yboardModifier>(NoModifier)  m_keyboardIambic:  true  m_loop:  false  m_mode:  0  m_sampleRate:  48000
  m_text:  ""  m_wpm:  13
2025-06-11 12:18:43.159 (D) CWKeyer::handleMessage: MsgConfigureCWKeyer
2025-06-11 12:18:43.159 (D) CWKeyer::applySettings:  m_dashKey:  Qt::Key_Minus  m_dashKeyModifiers:
QFlags<Qt::KeyboardModifier>(NoModifier)  m_dotKey:  Qt::Key_Period  m_dotKeyModifiers:  QFlags<Qt::Ke
yboardModifier>(NoModifier)  m_keyboardIambic:  true  m_loop:  false  m_mode:  0  m_sampleRate:  48000
  m_text:  ""  m_wpm:  13
2025-06-11 12:18:43.196 (W) SampleSourceFifo::write: underrun (write too slow) using 128000 old sample
s
2025-06-11 12:18:43.272 (W) SampleSourceFifo::write: underrun (write too slow) using 128000 old sample
s
2025-06-11 12:18:44.674 (W) SampleSourceFifo::write: underrun (write too slow) using 128000 old sample
2025-06-11 12:18:44.964 (W) SampleSourceFifo::write: overrun (read too slow) dropping 128000 samples
2025-06-11 12:18:47.453 (D) SoapySDROutput::handleMessage: MsgStartStop:  stop
2025-06-11 12:18:47.455 (D) DSPDeviceSinkEngine::stopGeneration
2025-06-11 12:18:47.459 (D) DSPDeviceSinkEngine::handleInputMessages: message: DSPGenerationStop
2025-06-11 12:18:47.459 (D) DSPDeviceSinkEngine::gotoIdle
2025-06-11 12:18:47.459 (D) SoapySDROutput::stop: SO mode. Just stop and delete the thread
2025-06-11 12:18:47.470 (D) SoapySDROutputThread::run: stop running loop
2025-06-11 12:18:47.484 (D) SoapySDROutputThread::~SoapySDROutputThread
[WARNING] setSampleRate: using rounded rate 1333000 is deprecated; use 4e6/3 or 1333333.3.
2025-06-11 12:18:47.485 (D) SoapySDROutput::applySettings: setSampleRate OK:  1000000
[INFO] setFrequency dir: 0, channel: 0, freq: 429999968.00
2025-06-11 12:18:47.485 (D) SoapySDROutput::setDeviceCenterFrequency: setFrequency(430000000)
2025-06-11 12:18:47.485 (D) SoapySDROutput::applySettings: set antenna to NONE
2025-06-11 12:18:47.486 (D) SoapySDROutput::applySettings: bandwidth set to 1000000
2025-06-11 12:18:47.486 (D) SoapySDROutput::applySettings: set global gain to 18
2025-06-11 12:18:47.486 (D) SoapySDROutput::applySettings: individual gain Modem PA set to 18.000000
2025-06-11 12:18:47.486 (D) SoapySDROutput::applySettings: unset AGC
[INFO] getGain dir: 0, channel: 0, value: 18
[INFO] getGain dir: 0, channel: 0, value: 18
2025-06-11 12:18:47.486 (D) SoapySDROutput::applySettings:  m_transverterMode:  false  m_transverterD
eltaFrequency:  0  m_centerFrequency:  430000000  Hz  m_LOppmTenths:  0  m_log2Interp:  1  m_devSample
Rate:  1000000  m_bandwidth:  1000000  m_globalGain:  18  force:  true
2025-06-11 12:18:47.486 (D) SoapySDROutput::applySettings: m_individualGains[Modem PA]: 18.000000
2025-06-11 12:18:47.487 (D) DSPDeviceSinkEngine::gotoIdle: stopping  SSBMod
2025-06-11 12:18:47.487 (D) SSBMod::stop
2025-06-11 12:18:47.496 (D) SoapySDROutputGui::handleInputMessages: message: SoapySDROutput::MsgReport
GainChange
2025-06-11 12:18:47.500 (D) SoapySDRInputGui::updateFrequencyLimits: delta: 0 min: 389500 max: 510000
2025-06-11 12:18:47.913 (D) DSPDeviceSinkEngine::handleInputMessages: message: DSPSignalNotification
2025-06-11 12:18:47.914 (D) DSPDeviceSinkEngine::handleInputMessages: DSPSignalNotification:  m_sample
Rate:  500000  m_centerFrequency:  430000000  m_realElseComplex false
2025-06-11 12:18:47.914 (D) DSPDeviceSinkEngine::handleInputMessages: forward message to  SSBMod
                                                    Messages: DSPSignalNotification: guiMessag
```

After some troubleshooting, I initially suspected that the issue was caused by the SoapySDR plugin in SDRAngel. As you can see under the yellow text, the problem appeared to be that the *SoapySDROutput::applySettings* function was setting the **antenna to NONE**. However, I later discovered that this actually wasn't the root cause.

It turns out that when selecting a TX or RX device and the correct channel in SDRAngel, the appropriate antenna is automatically selected. In this case, it was the TX/RX 1GHz antenna, which was also correctly identified by the **SoapySDRUtil --probe** command. So no luck there. Will continue to test the TX-side on CaribouLite's own libraries!

Problems?

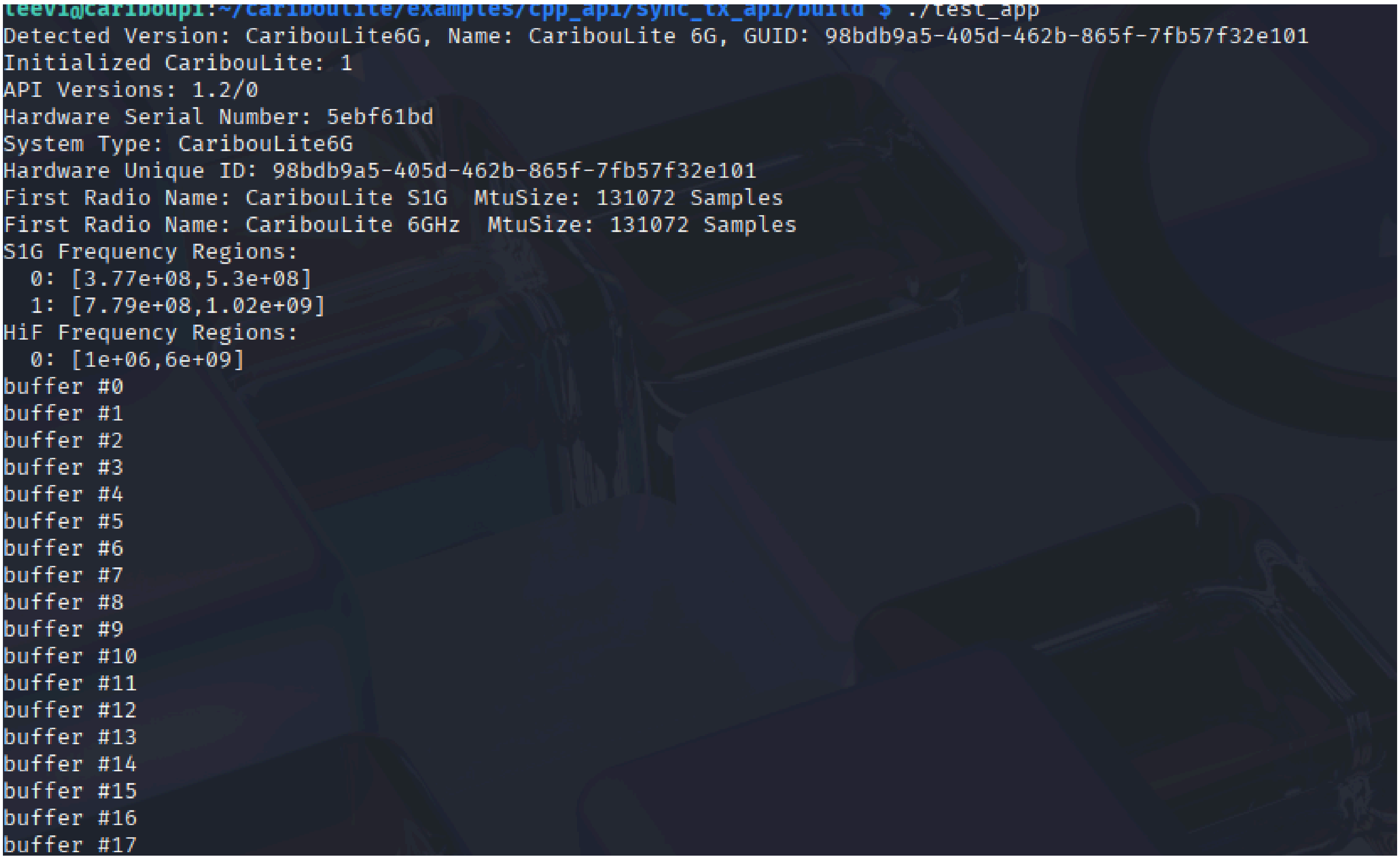
There was a quite a bit of heating happening on the Raspberry Pi. I noticed that when using SDRAngel, the system heated up quite quickly, so keep that in mind if you want to use a heavy softwares for these kind of projects. HTOP is your friend here and i highly recommend using that before trusting your devices.

CARIBOULITE’S TX-CHANNEL

Transmission trial run using CaribouLite’s own libraries

CaribouLite’s Github page has a library called **libcariboulite**. This library contains the core drivers and APIs required to interface with the CaribouLite SDR hardware. It manages hardware initialization, communication over the Raspberry Pi 4’s SMI interface, and provides functions to configure frequency, gain, sample rate, and antenna paths. It also includes functionality to transmit and receive IQ samples, and offers integration with SoapySDR, enabling use with popular SDR tools like GNU Radio and SDRAngel.

I found a program called sync tx api on the examples section. I compiled the program using CMake and Make.



```
teevi@caribopi:~/cariboulite/examples/cpp_api/sync_tx_api/build $ ./test_app
Detected Version: CaribouLite6G, Name: CaribouLite 6G, GUID: 98bdb9a5-405d-462b-865f-7fb57f32e101
Initialized CaribouLite: 1
API Versions: 1.2/0
Hardware Serial Number: 5ebf61bd
System Type: CaribouLite6G
Hardware Unique ID: 98bdb9a5-405d-462b-865f-7fb57f32e101
First Radio Name: CaribouLite S1G  MtuSize: 131072 Samples
First Radio Name: CaribouLite 6GHz  MtuSize: 131072 Samples
S1G Frequency Regions:
  0: [3.77e+08,5.3e+08]
  1: [7.79e+08,1.02e+09]
HiF Frequency Regions:
  0: [1e+06,6e+09]
buffer #0
buffer #1
buffer #2
buffer #3
buffer #4
buffer #5
buffer #6
buffer #7
buffer #8
buffer #9
buffer #10
buffer #11
buffer #12
buffer #13
buffer #14
buffer #15
buffer #16
buffer #17
```

Here you can see the program in action. It initializes the hardware, detects available radio interfaces and prints the gathered information. It also allocates sample buffers to test basic functionality. I tried changing the frequency in the program to the desired tuning value, but I still don't see any signal on the spectrum analyzer.

END OF A PROJECT

Final words

The combination of CaribouLite and Raspberry Pi 4 is a good demonstration how the SDR works. Using CaribouLite as a receiver works perfectly. It can receive a wide range of radio frequencies, and with the right software, it can demodulate and play them back accurately. So cool, right?

The real problem for this project was the transmission(TX) side. I tried multiple approaches, including two different software applications, various signal modulation types, and custom programs written in C. However, I was unable to confirm any signal transmission on the spectrum analyzer. The only confirmed transmission was observed using CaribouLite's own test application.

This really was a fun and interesting project on SDR-devices. I enjoyed working with Raspberry Pi and CaribouLite, as I have an interest in both the SDR world and Raspberry Pi's products.

Fear not my little Pi warrios, as this project will continue with another person as their personal project!

By,

Jaakko Oja

SOURCES

Links

Cariboulite / Github

Cariboulite - MatteoServa’s fork / Github

SoapySDR / Github

SDR++

Raspberry Pi / Github

Raspberry Pi / Official Website

SDRAngel / Github