CSUM GS Modules

CSUM

Jul 12, 2022

CONTENTS:

The scripts in this repository can be used to establish a communication link with the Robusta family of satellite missions from the Centre Spatial Universitaire Montpellier (CSUM). The scripts are provided as a set of signal and data processing blocks that encode or decode radio packets that can be sent to or received from Robusta platforms. The CSUM is developing a 1U and a 3U platform. The decoding and encoding process for each platform is explained in the sections below.

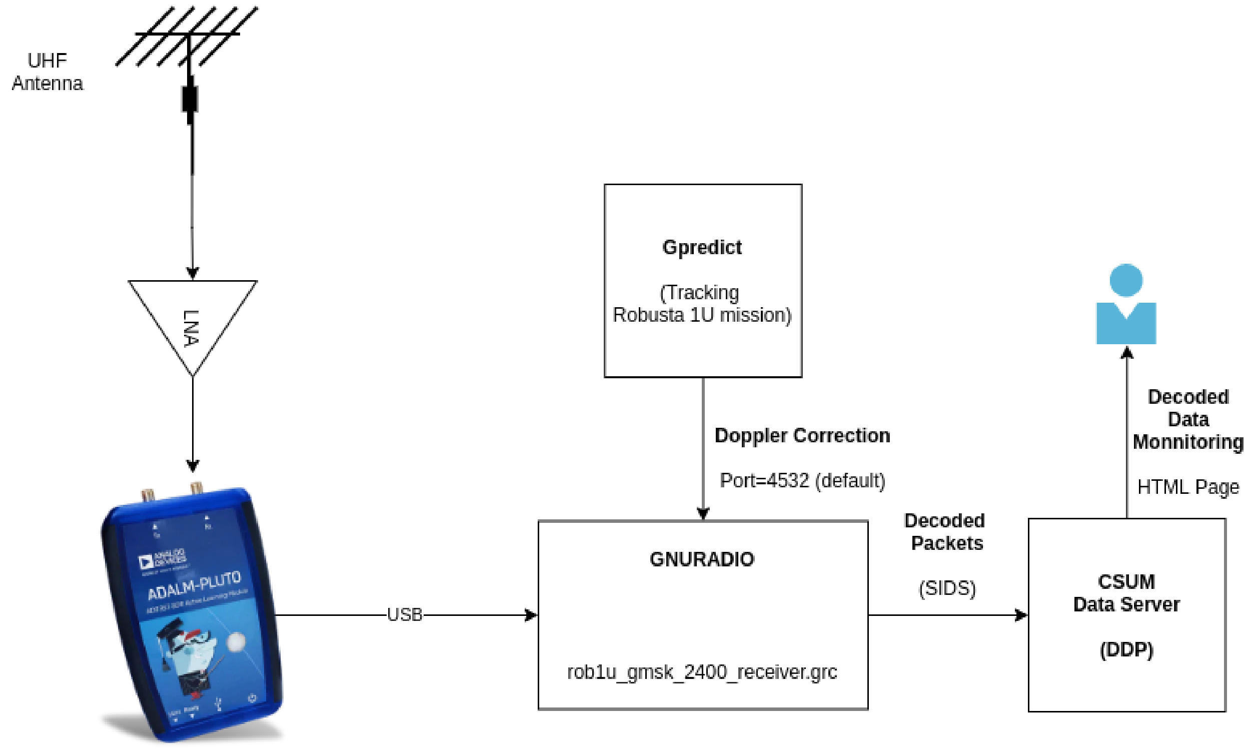
CHAPTER

ONE

1U DECODER

1.1 Reception

The diagram below shows the set of blocks required to receive signals from the 1U satellites. A Pluto SDR is used to implement the digital to Radio-Frequency interface. The signal processing and packet decoding stages can be implemented using GNURadio. Doppler correction on the receiving signal is performed by Gpredict.



Tip: In order to use the scripts **as it is** the Pluto SDR needs to be plugged to a USB port of a host computer and be configured with the default dns name: pluto.local.

The script **rob1u_gmsk_2400_receiver.grc** in the **receivers/** folder implements the filter and demodulation stages required to transform the incoming RF signals from the satellite into digital/raw binary data. The decoded packets will be printed in the terminal, but they will also be pushed to CSUM's server using the Simple Downlink Share Convention (SiDS) protocol. Once packets are pushed to CSUM's servers, they can be visualized on these two pages: CELESTA, MTCUBE-2.

1.1.1 Setup

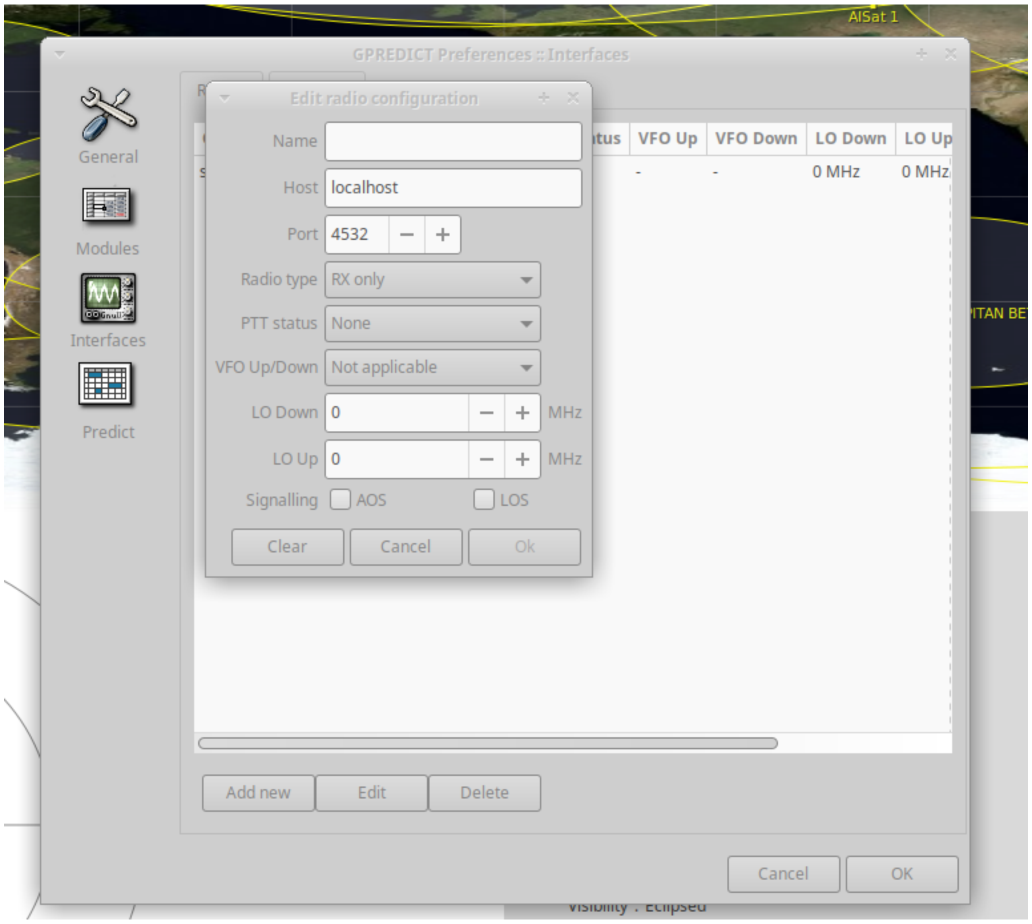
Dependencies

Make sure to have all the following software modules before trying to launch the Gnuradio script

- gr-gpredict-doppler
- gr-satellites

Enabling Doppler Correction

- **Adding a new TLE to Gpredict:**
  You can add new TLE coordinates to GPREDICT by clicking on *Edit* -> *Update TLE from Local Files*. You can check this reference for more information about Gpredict.
- Activating the frequency tracker on Gpredict:
  - Add a new radio interface to Gpredict. Click on *Edit* -> *Preferences*. Then in the left panel, go to *Interfaces*, in tab *Radios* click on *Add new*. You should see the following screen:



Name

Host localhost

Port 4532

Radio type RX only

PTT status None

VFO Up/Down Not applicable

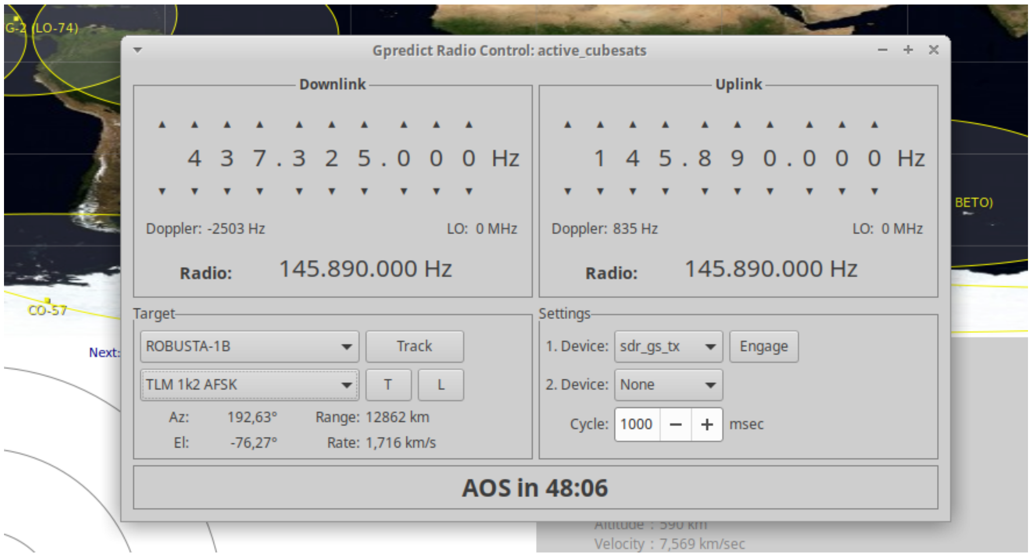
LO Down 0 MHz

LO Up 0 MHz

Signalling ☐ AOS ☐ LOS

tus	VFO Up	VFO Down	LO Down	LO Up
	-	-	0 MHz	0 MHz

  - Define a name for your interface (ie: *Radio CSUM*), set *Host: localhost* and *Port: 4532* as this will match the values in the Gnuradio Script. Click on OK to confirm.
  - On the right hand side of the main screen, click on the down arrow -> *Radio Control*. You should see the following screen:



Gpredict Radio Control: active_cubesats

Downlink

4 3 7 . 3 2 5 . 0 0 0 Hz

Doppler: -2503 Hz LO: 0 MHz

Radio: 145.890.000 Hz

Uplink

1 4 5 . 8 9 0 . 0 0 0 Hz

Doppler: 835 Hz LO: 0 MHz

Radio: 145.890.000 Hz

Target

ROBUSTA-1B Track

TLM 1k2 AFSK T L

Az: 192,63° Range: 12862 km

El: -76,27° Rate: 1,716 km/s

Settings

1. Device: sdr_gs_tx Engage

2. Device: None

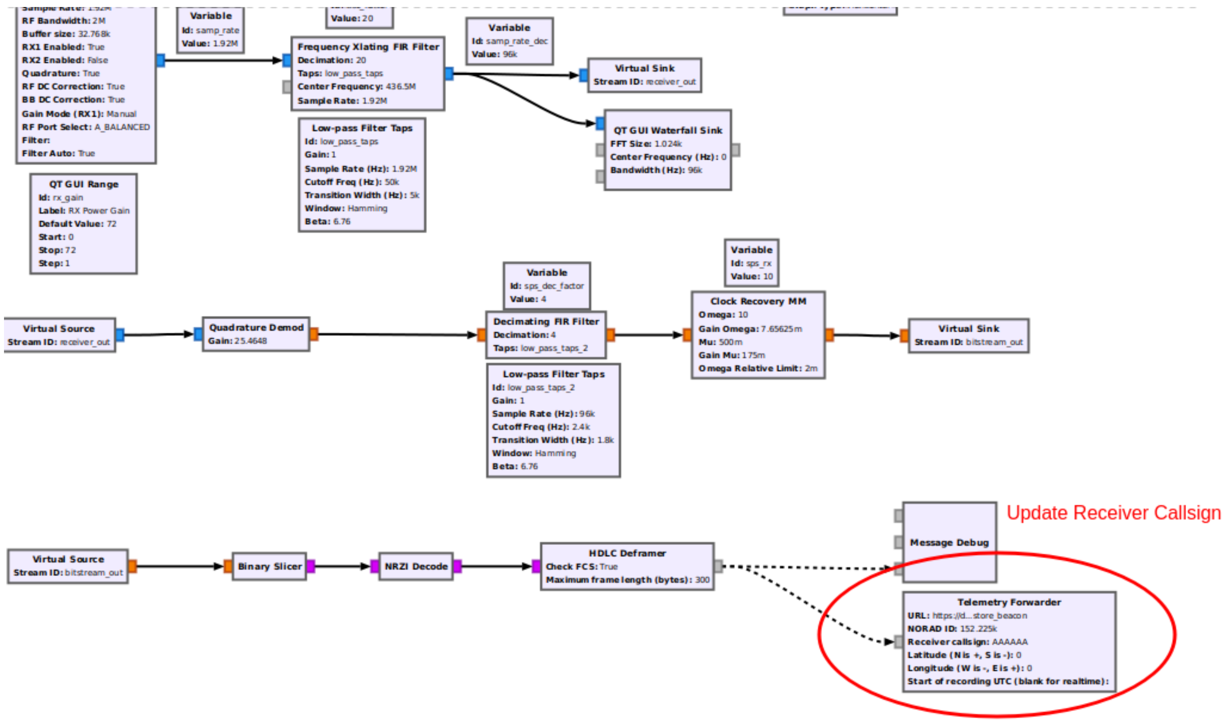
Cycle: 1000 msec

AOS in 48:06

- Under *Settings* -> *Device*, select the Radio interface you just created.
- Under *Target*, select the satellite that you want to track (the one you just added the TLE)
- Set the right downlink frequency for the satellite you want to track
- Click on **Engage** then **Track**. The doppler correction should be updated in your Gnuradio script.

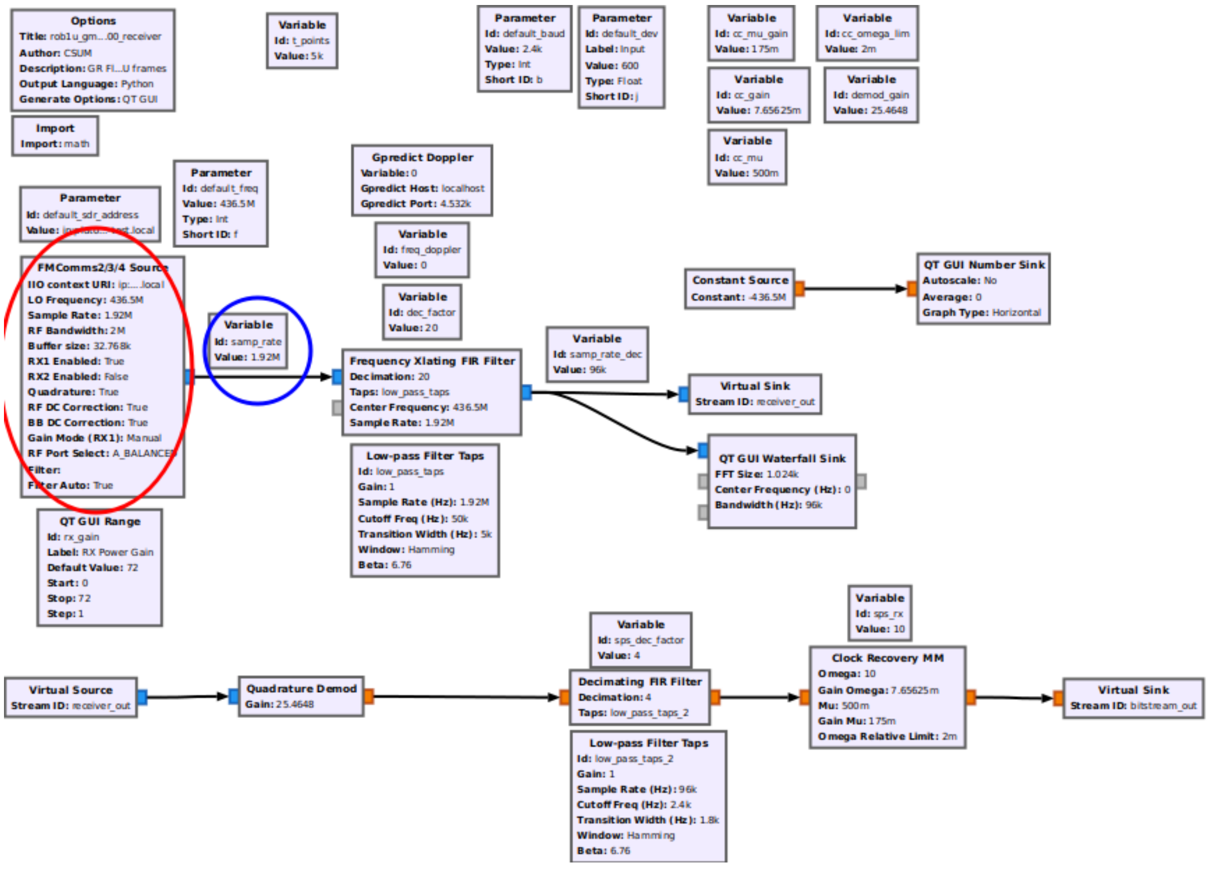
Packet Forwarder

Packets decoded using the provided **rob1u_gmsk_2400_receiver.grc** script are automatically forwarded to CSUM's server. In order for the packets to be properly stored, make sure to update your HAMRADIO callsign in the *Telemetry Forwarder* block. The image below indicates which block to update in the GRC flowgraph.



Using Another SDR

You can use another SDR device to decode signals from CSUM satellites. For that, you only need to replace the SDR Source block in the Gnuradio script, as shown below:



```
Options
Title: rob1u_gm....00_receiver
Author: CSUM
Description: GR Fl...U frames
Output Language: Python
Generate Options: QT GUI

Import
Import: math

Variable
Id: t_points
Value: 5k

Parameter
Id: default_baud
Value: 2.4k
Type: Int
Short ID: b

Parameter
Id: default_dev
Label: Input
Value: 600
Type: Float
Short ID: j

Variable
Id: cc_mu_gain
Value: 175m

Variable
Id: cc_omega_lim
Value: 2m

Variable
Id: cc_gain
Value: 7.65625m

Variable
Id: demod_gain
Value: 25.4648

Variable
Id: cc_mu
Value: 500m

Parameter
Id: default_sdr_address
Value: ip:pluto....test.local

Parameter
Id: default_freq
Value: 436.5M
Type: Int
Short ID: f

Gpredict Doppler
Variable: 0
Gpredict Host: localhost
Gpredict Port: 4.532k

Variable
Id: freq_doppler
Value: 0

Variable
Id: dec_factor
Value: 20

FMComms2/3/4 Source
IIO context URI: ip:....local
LO Frequency: 436.5M
Sample Rate: 1.92M
RF Bandwidth: 2M
Buffer size: 32.768k
RX1 Enabled: True
RX2 Enabled: False
Quadrature: True
RF DC Correction: True
BB DC Correction: True
Gain Mode (RX1): Manual
RF Port Select: A_BALANCED
Filter:
Filter Auto: True

Variable
Id: samp_rate
Value: 1.92M

Frequency Xlating FIR Filter
Decimation: 20
Taps: low_pass_taps
Center Frequency: 436.5M
Sample Rate: 1.92M

Variable
Id: samp_rate_dec
Value: 96k

Constant Source
Constant: -436.5M

QT GUI Number Sink
Autoscale: No
Average: 0
Graph Type: Horizontal

Virtual Sink
Stream ID: receiver_out

QT GUI Waterfall Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 96k

Low-pass Filter Taps
Id: low_pass_taps
Gain: 1
Sample Rate (Hz): 1.92M
Cutoff Freq (Hz): 50k
Transition Width (Hz): 5k
Window: Hamming
Beta: 6.76

QT GUI Range
Id: rx_gain
Label: RX Power Gain
Default Value: 72
Start: 0
Stop: 72
Step: 1

Variable
Id: sps_dec_factor
Value: 4

Variable
Id: sps_rx
Value: 10

Virtual Source
Stream ID: receiver_out

Quadrature Demod
Gain: 25.4648

Decimating FIR Filter
Decimation: 4
Taps: low_pass_taps_2

Clock Recovery MM
Omega: 10
Gain Omega: 7.65625m
Mu: 500m
Gain Mu: 175m
Omega Relative Limit: 2m

Virtual Sink
Stream ID: bitstream_out

Low-pass Filter Taps
Id: low_pass_taps_2
Gain: 1
Sample Rate (Hz): 96k
Cutoff Freq (Hz): 2.4k
Transition Width (Hz): 1.8k
Window: Hamming
Beta: 6.76
```

If you do so, make sure to update the sampling rate accordingly (blue circle).

1.1.2 Launching Procedure

In order to start the reception flow, the following steps are required:

1. Open GnuRadio, set the right downlink frequency (Variable **default_freq**)
2. Open one of the receiver GRC scripts inside the **receivers** folder
3. Execute the flowgraph (Start button in gnuradio companion). You should see a simple waterfall display.
4. Open Gpredict, click on the down arrow on the right hand side of the main screen -> *Radio Control*. Select the proper radio interface and set the downlink frequency. Click on **Engage** then **Track**.
5. Wait for the satellite to arrive

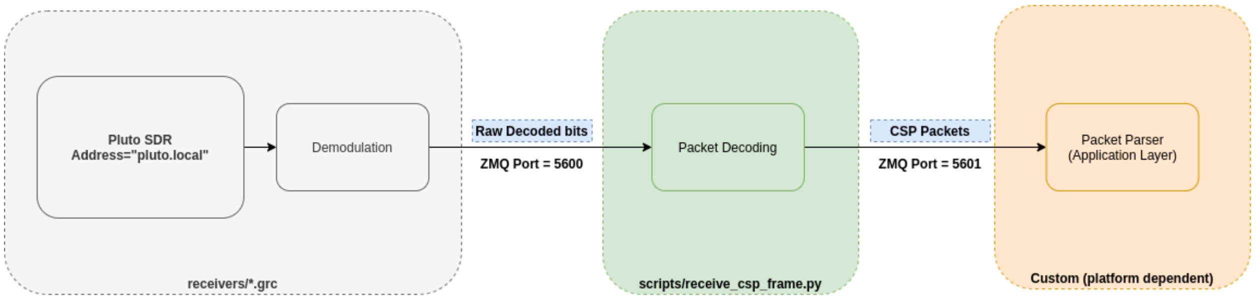
CHAPTER

TWO

3U DECODER

2.1 Reception

The diagram below shows the set of blocks required to receive signals from the satellite. A Pluto SDR is used to implement the digital to Radio-Frequency interface. The set of scripts provided in the **receivers/** folder implement the filter and demodulation techniques required to transform the incoming RF signals from the satellite into digital/raw binary data. These decoded bits are then forwarded to another processing stage through a ZMQ Interface (**Port 5600**). The **receive_frame** python scripts inside the **scripts** folder can be then used to subscribe to this port and build the incoming CSP packets from this raw bit stream. The decoded packets will be printed in the terminal, but they will also be pushed to another ZMQ interface (**Port 5601**) so that the user can build its custom application (not provided in this repo), that will be in charge of further processing the incoming data.



Tip: In order to use the scripts **as it is** the Pluto SDR needs to be plugged to a USB port of a host computer and be configured with the default dns name: pluto.local.

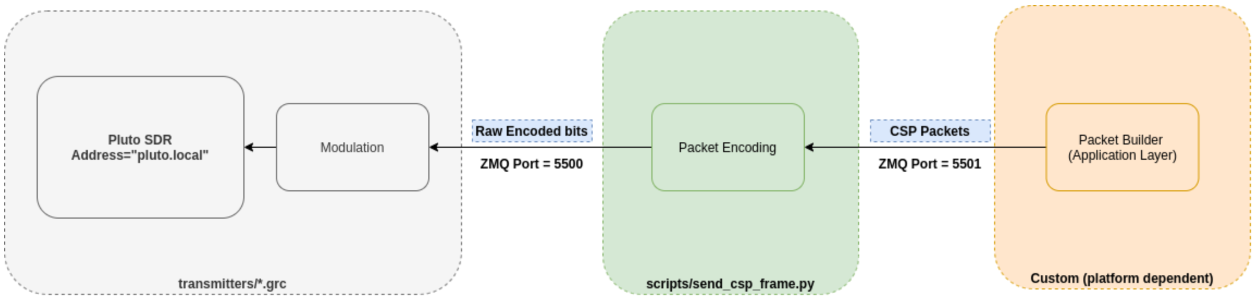
In order to start the reception flow, the following steps are required:

1. Open GnuRadio
2. Open one of the receiver GRC scripts inside the **receivers** folder
3. Execute the flowgraph (Start button in gnuradio companion). You should see a simple waterfall display.
4. Open a terminal in the **scripts** folder.
5. Start the **receive_csp_frame.py** script: *python3 receive_csp_frame.py*

Tip: By default, the receiver gnuradio radio scripts are configured to receive signals at 437 MHz. Verify if that is the same transmission frequency of the satellite.

2.2 Transmission

The transmission flow works in a similar way, but on the opposite direction. The packets can be built using a Custom script at the application layer (not provided in this repository). Then, these packets are pushed via ZMQ to **Port 5501**. The *send_csp_frame.py* script takes this packets, builds the proper radio packets and forward them to the ZMQ Interface (**Port 5500**) of the Gnuradio transmitter script, as shown in the picture below.



In order to start the reception flow, the following steps are required:

1. Open GnuRadio
2. Open one of the receiver GRC scripts inside the **transmitters** folder
3. Execute the flowgraph (Start button in gnuradio companion). You should see a simple waterfall display.
4. Open a terminal in the **scripts** folder.
5. Start the **send_csp_frame.py** script: *python3 send_csp_frame.py*

This script will keep listening to port ZMQ 5500 for new packets. Any data that is pushed to this port will be immediately encapsulated with a Radio Packet and dispatched to the gnuradio script and hence to the RF channel.

Tip: By default, the receiver gnuradio radio scripts are configured to receive signals at 435.8 MHz. Verify if that is the same transmission frequency of the satellite.