

Synchronization between a Vertical Incidence Pulsed Ionospheric Radar and an Ionospheric Echoes Receiver v1

Author: Isaac Tupac
Jicamarca Radio Observatory

6th European GNU Radio days 2023

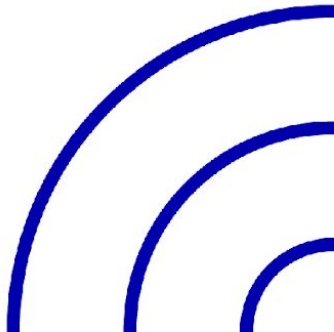




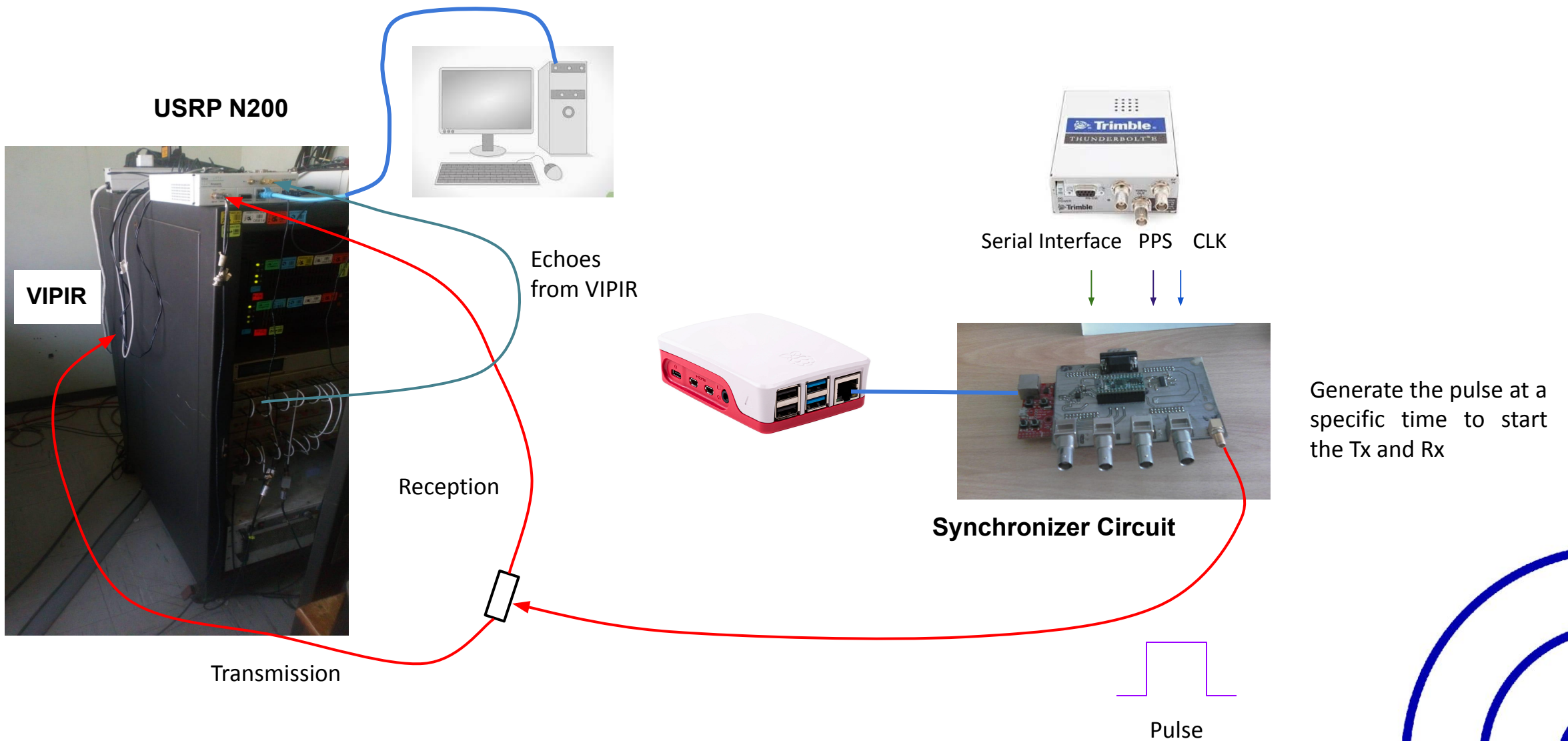
Introduction

In this presentation I will talk about how we got synchronization between a vertical incidence pulsed ionospheric radar (VIPIR) and an ionospheric echoes receiver, version 1, mainly composed by a USRP N200.

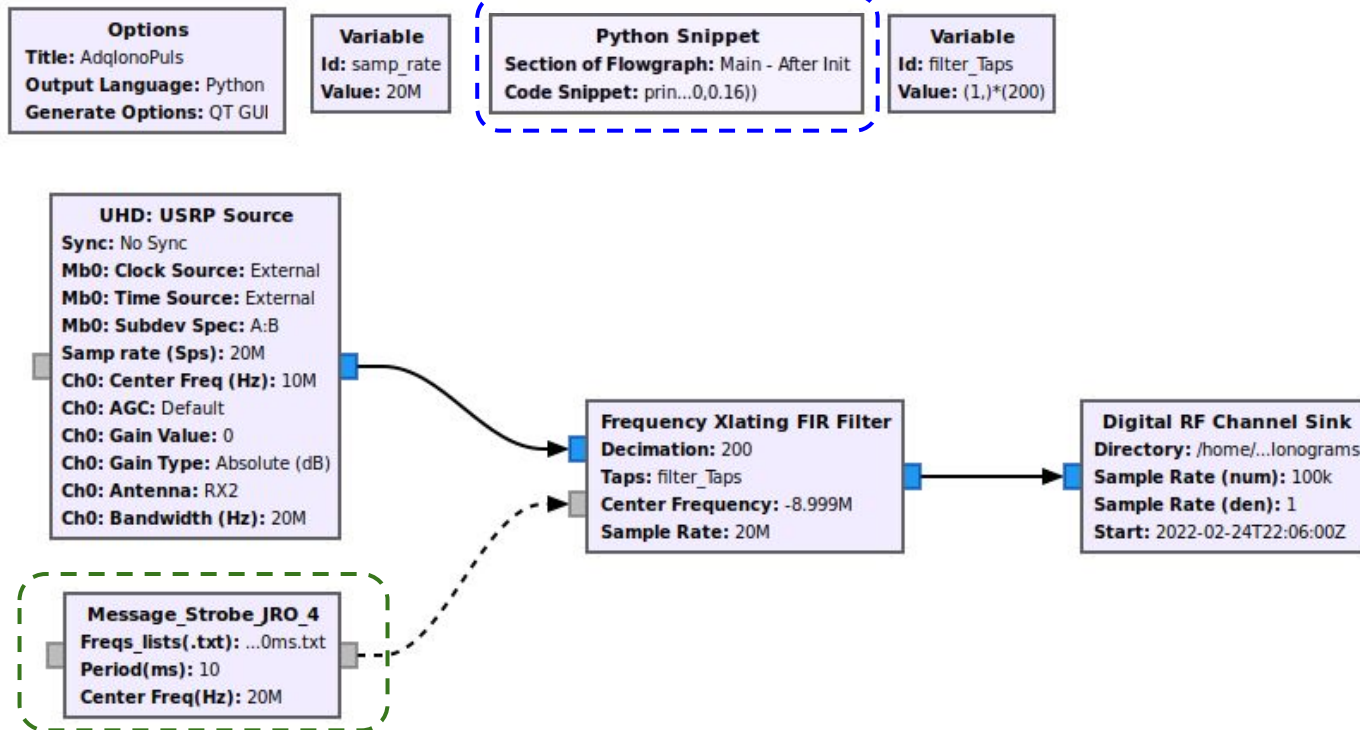
We will see three ways of synchronizations to understand the behavior developed in the acquisition. We will also see how the synchronization could affects the obtained ionospheric profiles and how the using of OOT block and tags could affect the synchronization.



Direct synchronization

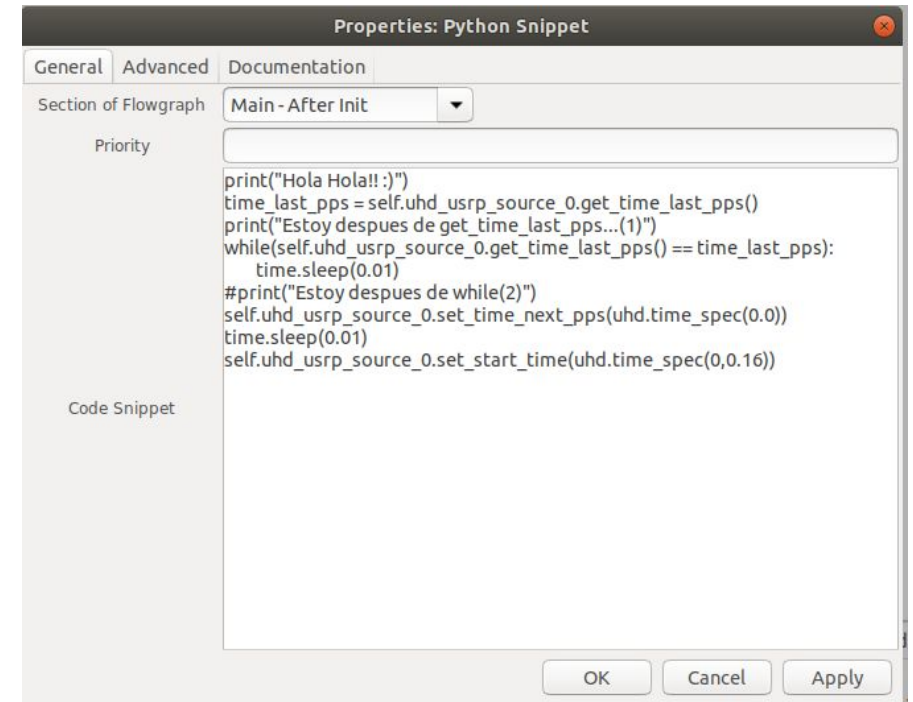


Direct synchronization



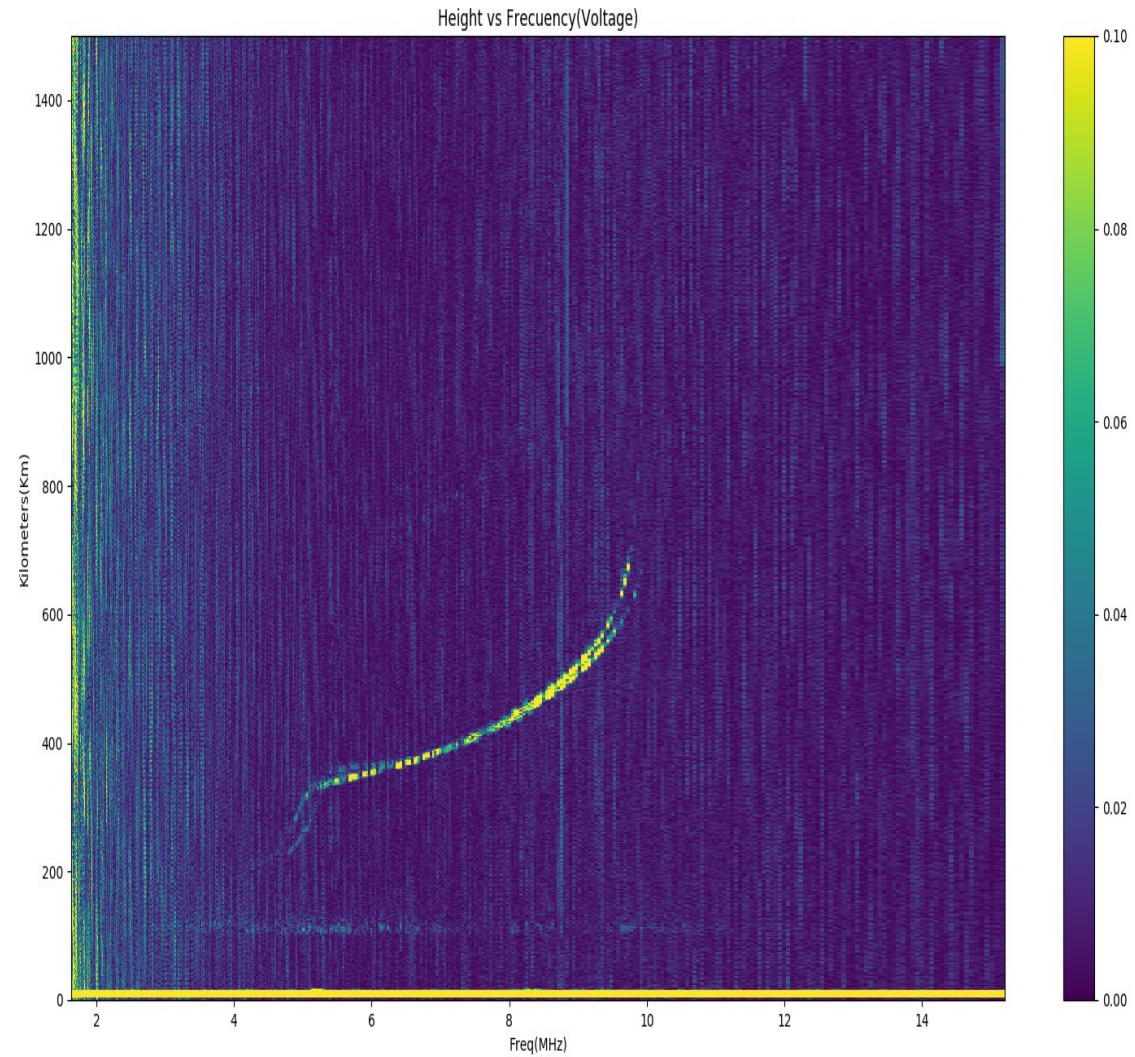
Out of tree block which uses message passing to change the center frequency value.

The sending of the new frequency value and the time between sended values is programming in a second thread.



The flowgraph will wait until the Pulse is received, will set the time in 0 and start the acquisition at 160ms.

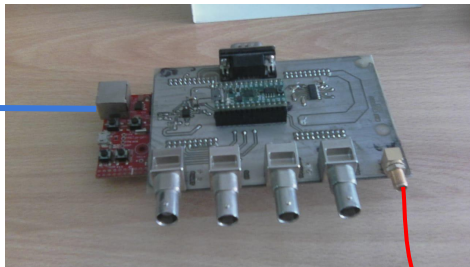
Direct synchronization



Indirect synchronization: time method



Serial Interface PPS CLK



Synchronizer Circuit

Generate the pulse at a specific time to start the Tx



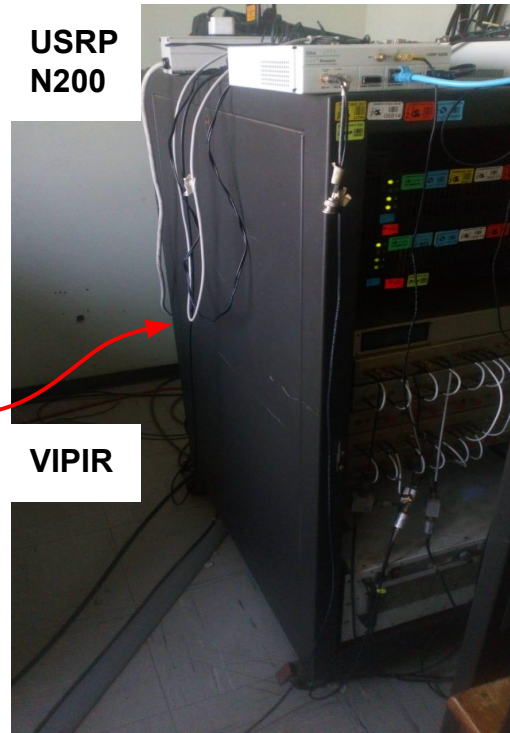
Pulse

GPSDO



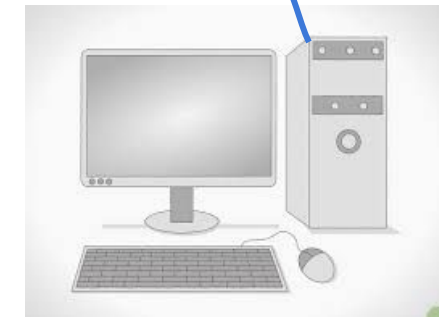
Clock ↓ ↓ PPS

USRP N200

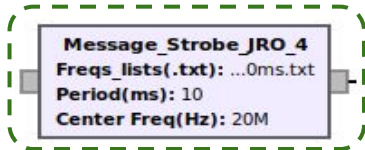
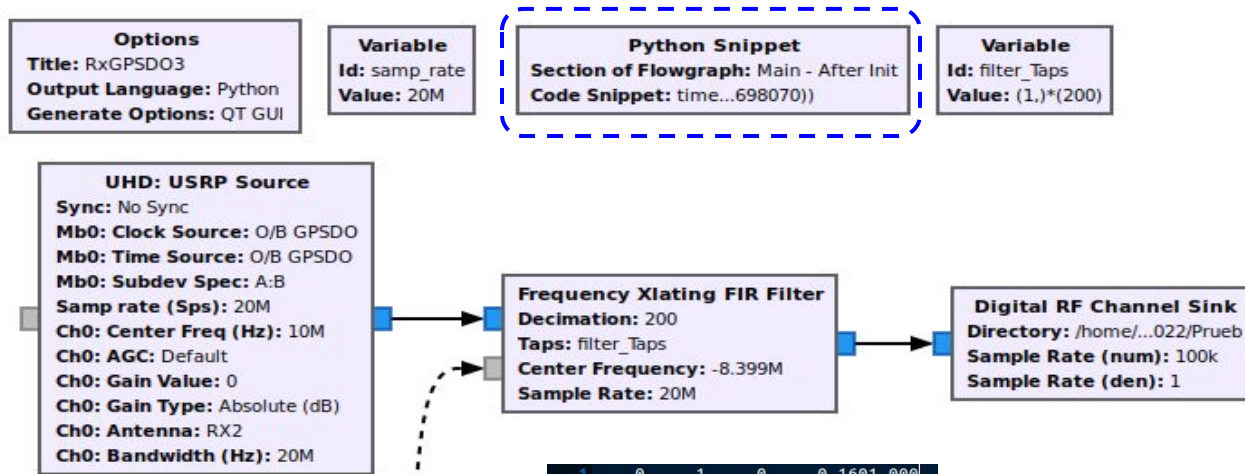


VIPIR

GPSDO will ask for the current gps time and if we are after the transmission start time, we will begin the reception



Indirect synchronization: time method



1	0	1	0	0	1601.000
2	0	0	1	0	1609.000
3	0	0	2	0	1617.039
4	0	0	3	0	1625.120
5	0	1	0	0	1601.000
6	0	1	1	0	1609.000
7	0	1	2	0	1617.039
8	0	1	3	0	1625.120
9	0	2	0	0	1601.000
10	0	2	1	0	1609.000
11	0	2	2	0	1617.039
12	0	2	3	0	1625.120
13	0	3	0	0	1601.000
14	0	3	1	0	1609.000
15	0	3	2	0	1617.039
16	0	3	3	0	1625.120
17	4	0	0	0	1633.240
18	4	0	1	0	1641.402
19	4	0	2	0	1649.604
20	4	0	3	0	1657.847
21	4	1	0	0	1633.240
22	4	1	1	0	1641.402
23	4	1	2	0	1649.604
24	4	1	3	0	1657.847
25	4	2	0	0	1633.240
26	4	2	1	0	1641.402
27	4	2	2	0	1649.604
28	4	2	3	0	1657.847
29	4	3	0	0	1633.240
30	4	3	1	0	1641.402
31	4	3	2	0	1649.604
32	4	3	3	0	1657.847

Properties: Python Snippet

General | Advanced | Documentation

Section of Flowgraph: Main - After Init

Priority:

```
time_gps=self.uhd_usrp_source_0.get_mboard_sensor("gps_time").to_real()
while(time_gps < 1651698070):#16:01:10
    time_gps=self.uhd_usrp_source_0.get_mboard_sensor("gps_time").to_real()
self.uhd_usrp_source_0.set_start_time(uhd.time_spec_t(1651698070))
```

Experiment:

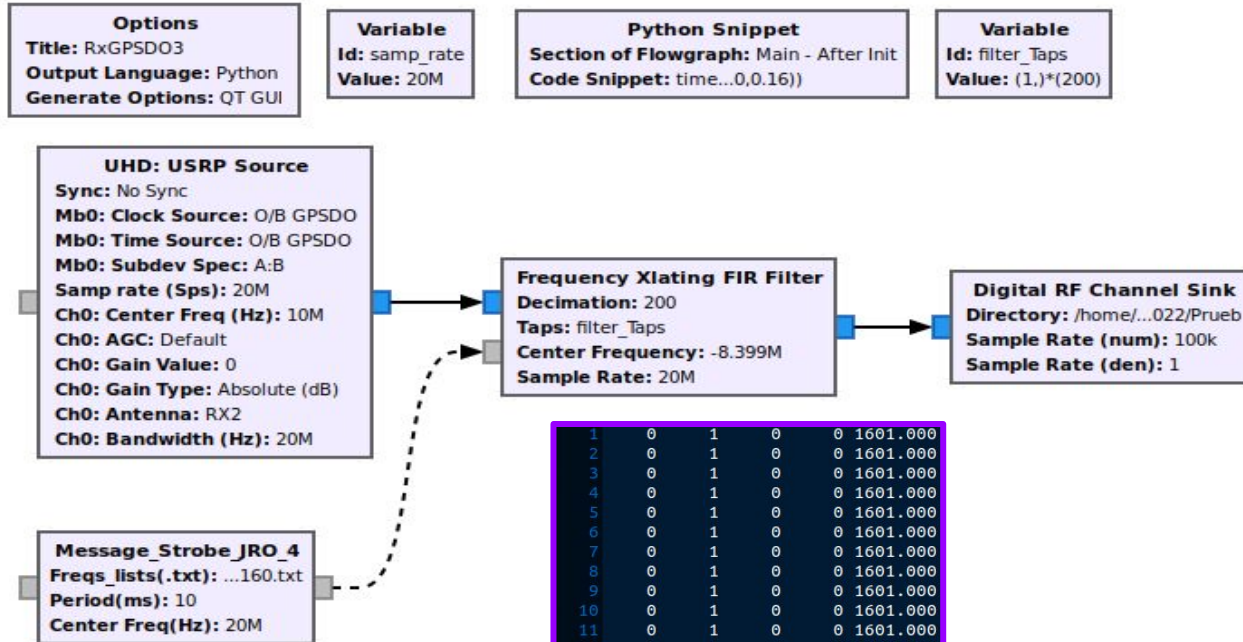
Vipir received the transmission pulse at 16:01:10
 USRP began the acquisition at 16:01:10

Result:

Acquisition mistake: The sample acquire with time 16:01:10 got before the beginning of the acquisition so for that GNU Radio show a mistake.

Indirect synchronization: time method

+160ms



1	0	1	0	0	1601.000
2	0	1	0	0	1601.000
3	0	1	0	0	1601.000
4	0	1	0	0	1601.000
5	0	1	0	0	1601.000
6	0	1	0	0	1601.000
7	0	1	0	0	1601.000
8	0	1	0	0	1601.000
9	0	1	0	0	1601.000
10	0	1	0	0	1601.000
11	0	1	0	0	1601.000
12	0	1	0	0	1601.000
13	0	1	0	0	1601.000
14	0	1	0	0	1601.000
15	0	1	0	0	1601.000
16	0	1	0	0	1601.000
17	0	1	0	0	1601.000
18	0	0	1	0	1609.000
19	0	0	2	0	1617.039
20	0	0	3	0	1625.120
21	0	1	0	0	1601.000
22	0	1	1	0	1609.000
23	0	1	2	0	1617.039
24	0	1	3	0	1625.120
25	0	2	0	0	1601.000
26	0	2	1	0	1609.000
27	0	2	2	0	1617.039
28	0	2	3	0	1625.120
29	0	3	0	0	1601.000
30	0	3	1	0	1609.000
31	0	3	2	0	1617.039
32	0	3	3	0	1625.120

We had to add 16 frequencies (160ms) because the Message Strobe JRO_4 block generated the frequency change in a second processing thread parallel to the acquisition.

```

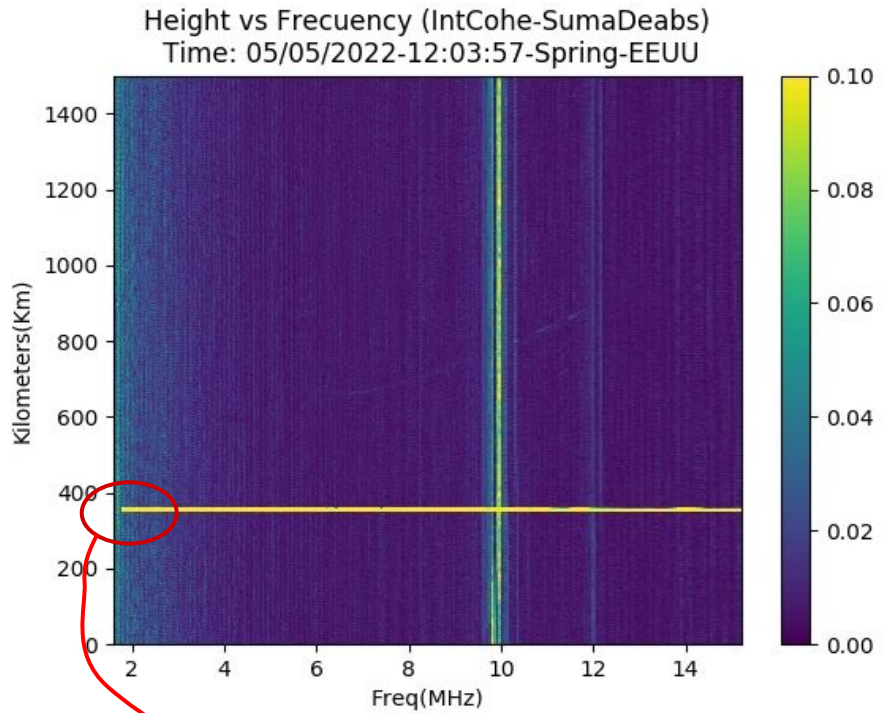
Properties: Python Snippet
General | Advanced | Documentation
Section of Flowgraph: Main - After Init
Priority:
time_gps=self.uhd_usrp_source_0.get_mboard_sensor("gps_time").to_real()
while(time_gps < 1651698070): #16:01:10
    time_gps=self.uhd_usrp_source_0.get_mboard_sensor("gps_time").to_real()
    self.uhd_usrp_source_0.set_start_time(uhd.time_spec_t(1651698070,0.16))
    
```

Experiment:

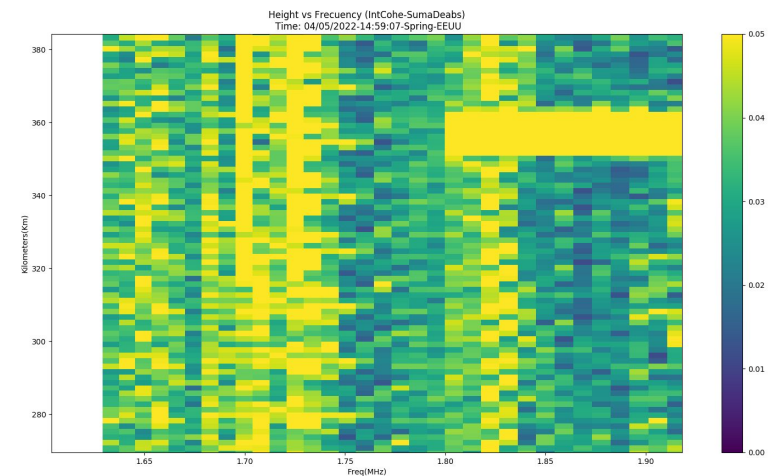
Vipir received the transmission pulse at 16:01:10
 USRP began the acquisition at 16:01:10+160ms

Ipp: 10ms
 freq: 1601 MHz - 15.171 MHz
 Step:4
 Rep:4

Indirect synchronization: time method

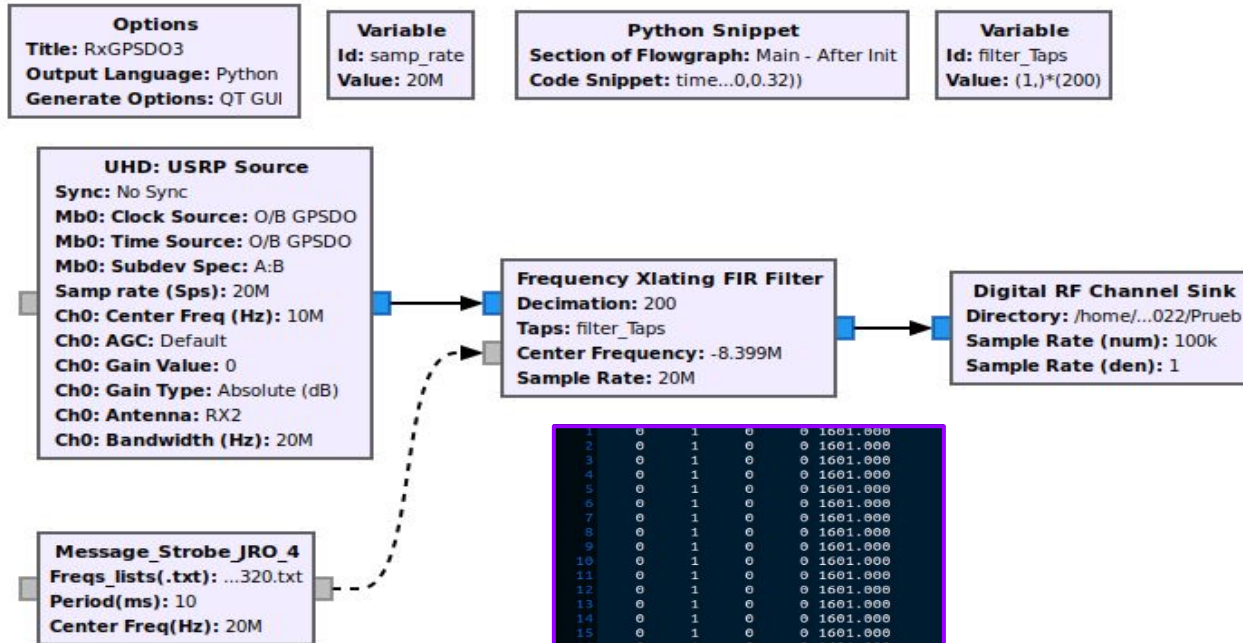


We saw a profile 160ms after the beginning of VIPIR transmission



Indirect synchronization: time method

+320ms



1	0	1	0	0	1601.000
2	0	1	0	0	1601.000
3	0	1	0	0	1601.000
4	0	1	0	0	1601.000
5	0	1	0	0	1601.000
6	0	1	0	0	1601.000
7	0	1	0	0	1601.000
8	0	1	0	0	1601.000
9	0	1	0	0	1601.000
10	0	1	0	0	1601.000
11	0	1	0	0	1601.000
12	0	1	0	0	1601.000
13	0	1	0	0	1601.000
14	0	1	0	0	1601.000
15	0	1	0	0	1601.000
16	0	1	0	0	1601.000
17	0	1	0	0	1601.000
18	0	1	0	0	1601.000
19	0	1	0	0	1601.000
20	0	1	0	0	1601.000
21	0	1	0	0	1601.000
22	0	1	0	0	1601.000
23	0	1	0	0	1601.000
24	0	1	0	0	1601.000
25	0	1	0	0	1601.000
26	0	1	0	0	1601.000
27	0	1	0	0	1601.000
28	0	1	0	0	1601.000
29	0	1	0	0	1601.000
30	0	1	0	0	1601.000
31	0	1	0	0	1601.000
32	0	1	0	0	1601.000
33	0	1	0	0	1601.000
34	0	0	1	0	1609.000
35	0	0	2	0	1617.039
36	0	0	3	0	1625.120
37	0	1	0	0	1601.000
38	0	1	1	0	1609.000
39	0	1	2	0	1617.039
40	0	1	3	0	1625.120
41	0	2	0	0	1601.000
42	0	2	1	0	1609.000
43	0	2	2	0	1617.039
44	0	2	3	0	1625.120

We had to add 32 frequencies (320ms) because the Message Strobe JRO_4 block generated the frequency change in a second processing thread parallel to the acquisition.

```

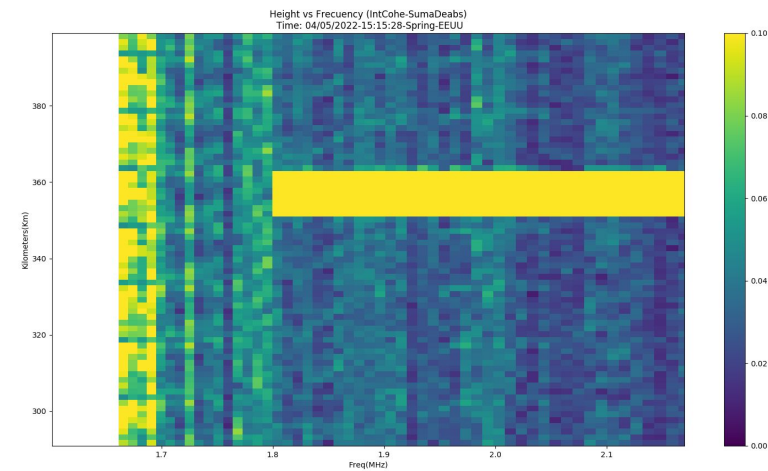
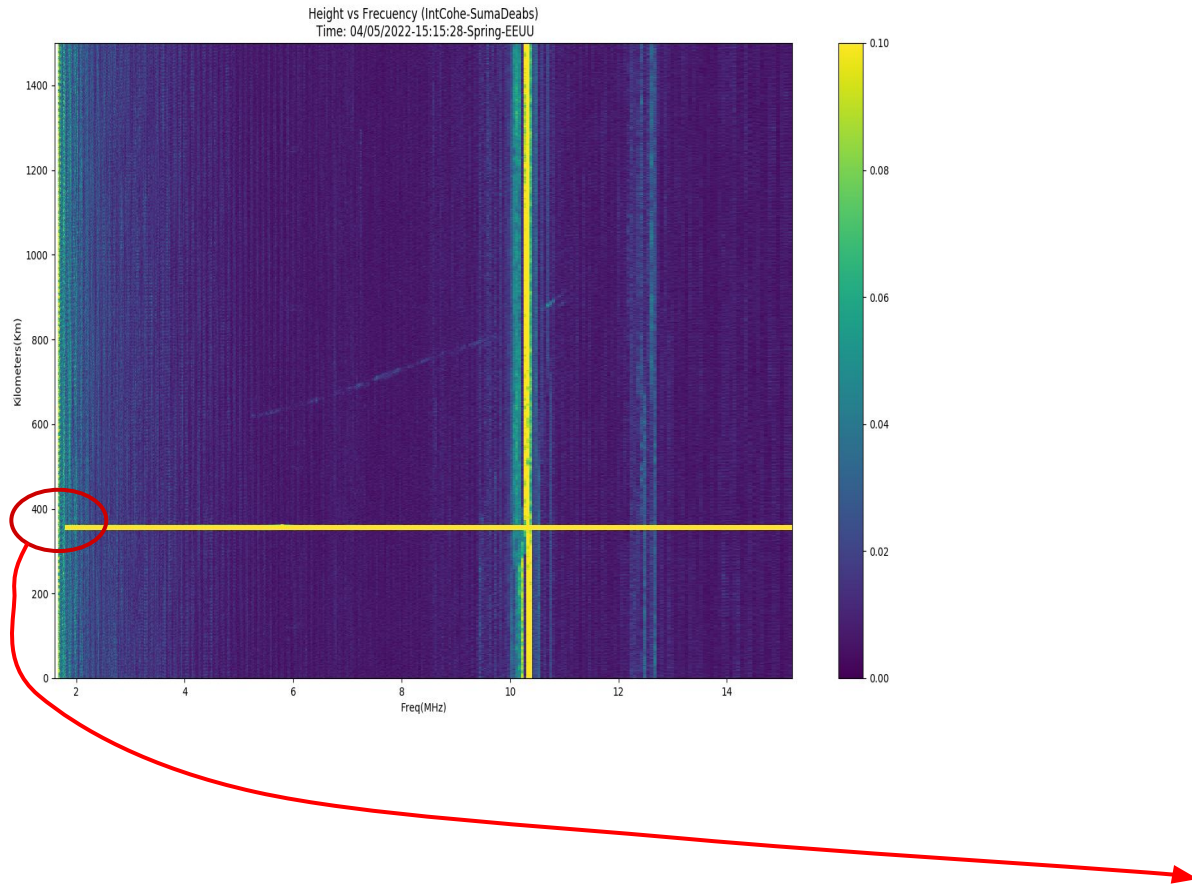
Properties: Python Snippet
General | Advanced | Documentation
Section of Flowgraph: Main - After Init
Priority:
time_gps=self.uhd_usrp_source_0.get_mboard_sensor("gps_time").to_real()
while(time_gps < 1651698070): #16:01:10
    time_gps=self.uhd_usrp_source_0.get_mboard_sensor("gps_time").to_real()
    self.uhd_usrp_source_0.set_start_time(uhd.time_spec_t(1651698070,0.32))
    
```

Experiment:

Vipir received the transmission pulse at 16:01:10
 USRP began the acquisition at 16:01:10+320ms

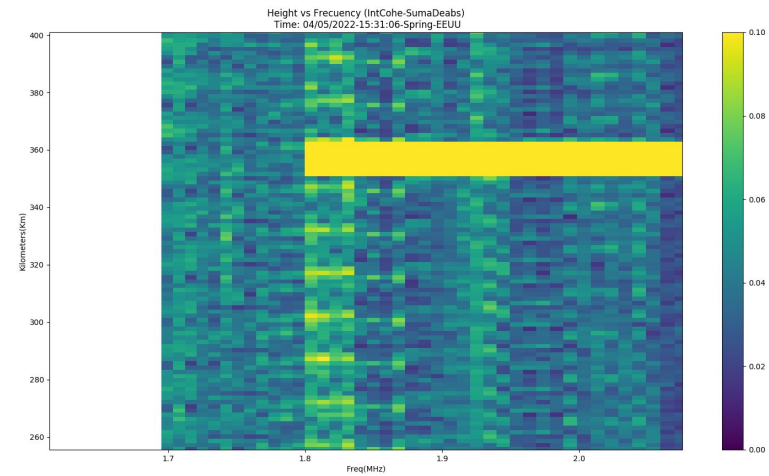
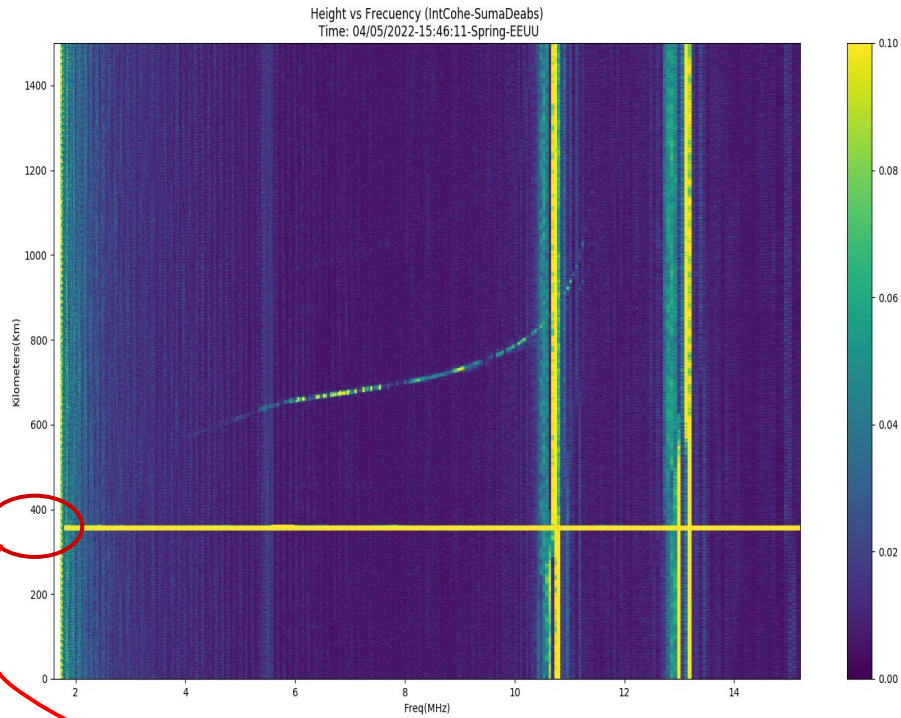
Ipp: 10ms
 freq: 1601 MHz - 15.171 MHz
 Step:4
 Rep:4

Indirect synchronization: time method



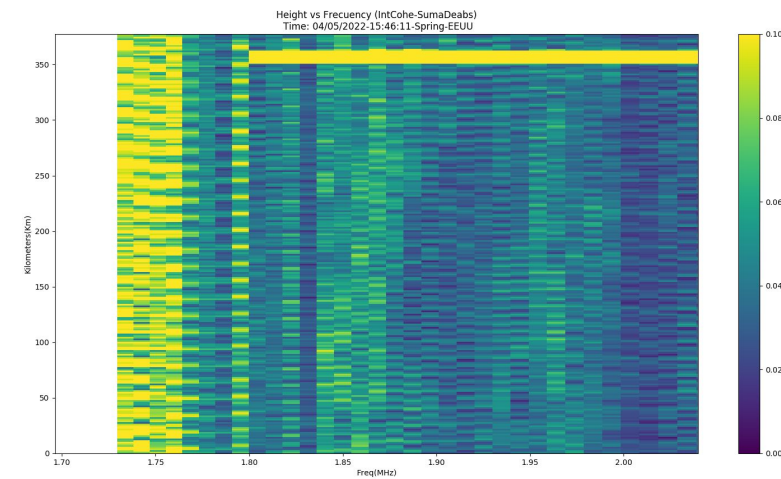
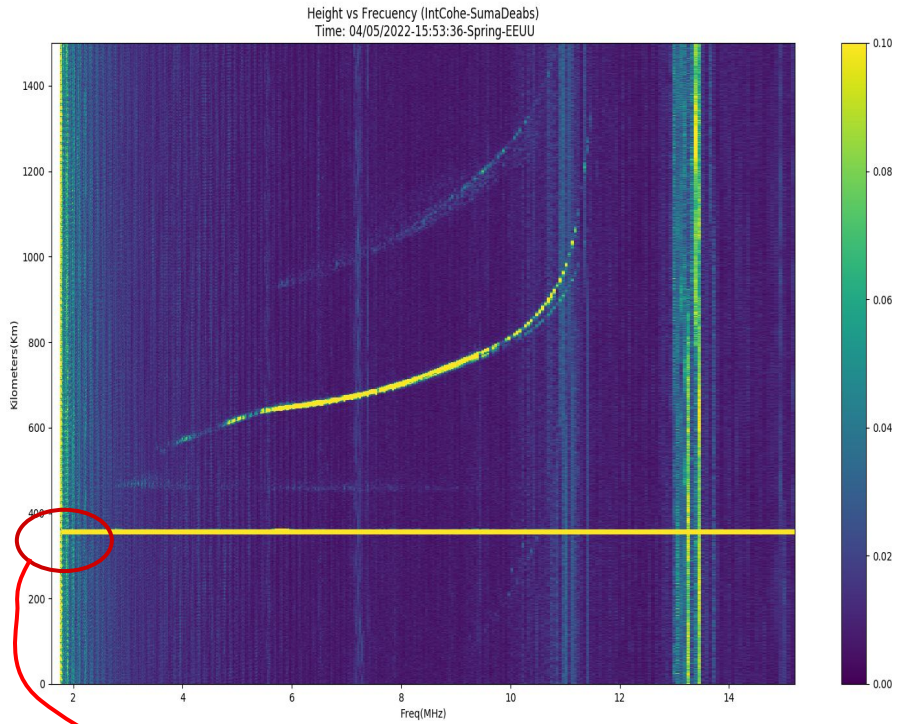
Indirect synchronization: time method

+480ms



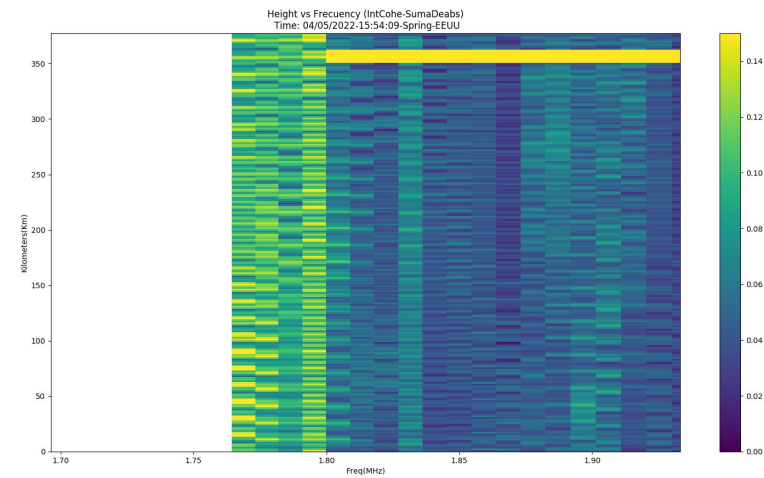
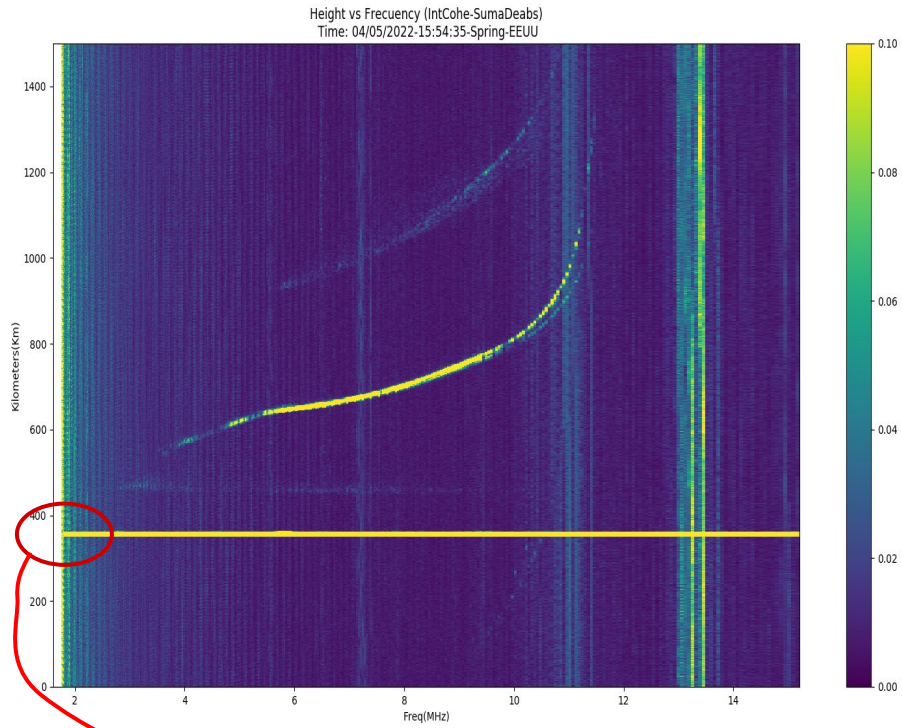
Indirect synchronization: time method

+640ms



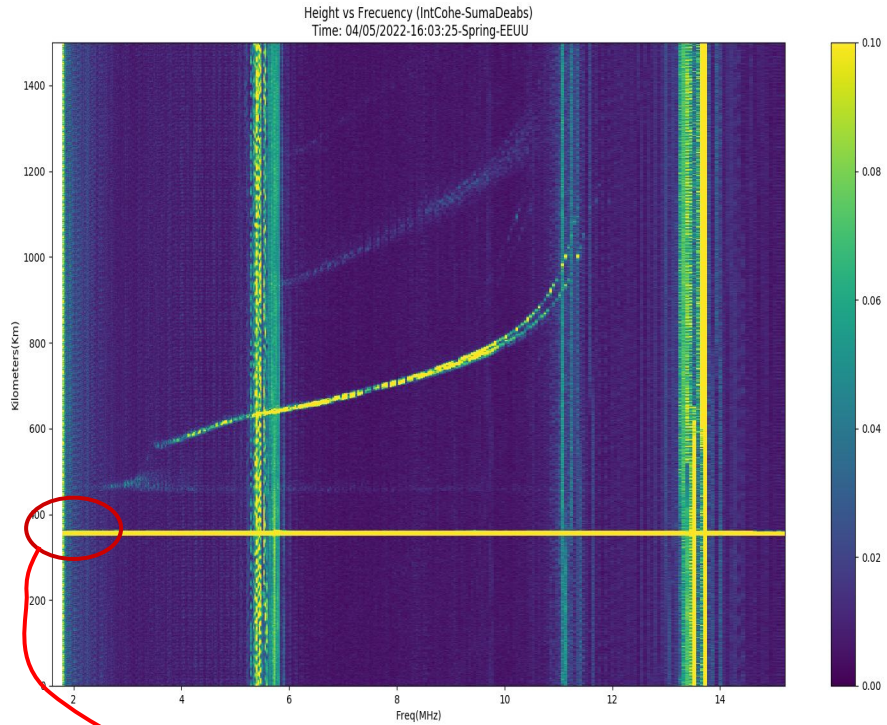
Indirect synchronization: time method

+800ms

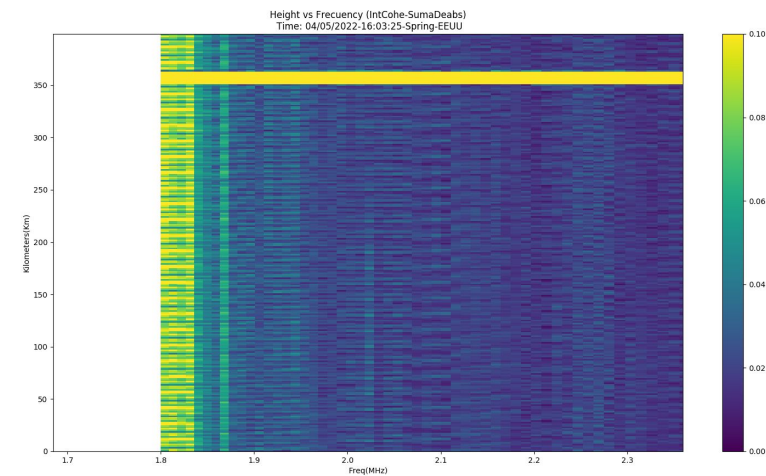


Indirect synchronization: time method

+960ms

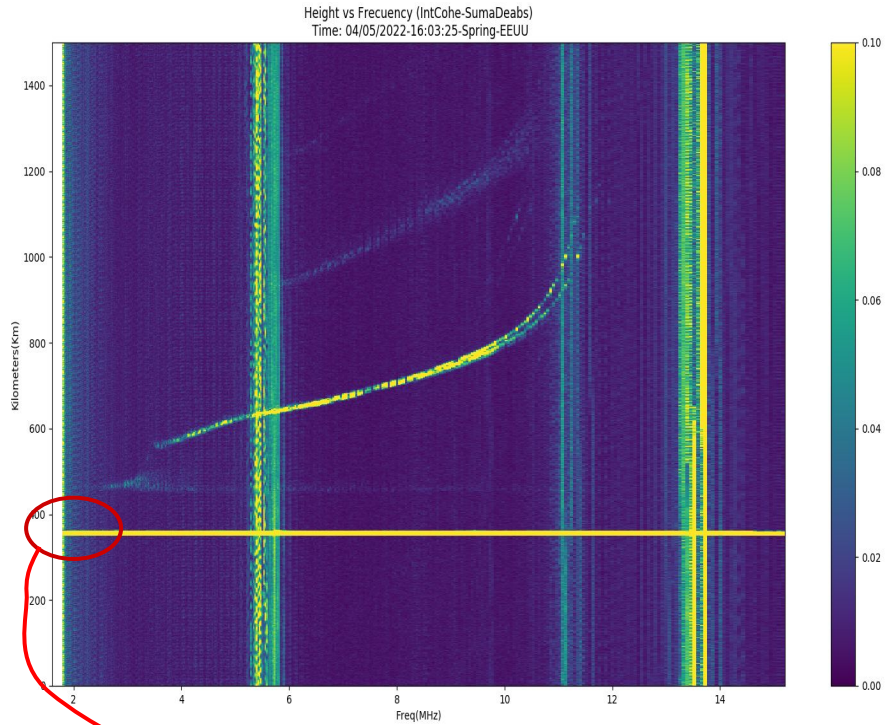


We see that with 960ms added to the reception start time, we could synchronize with the transmission but we see the reference in the ionogram is not at zero level.

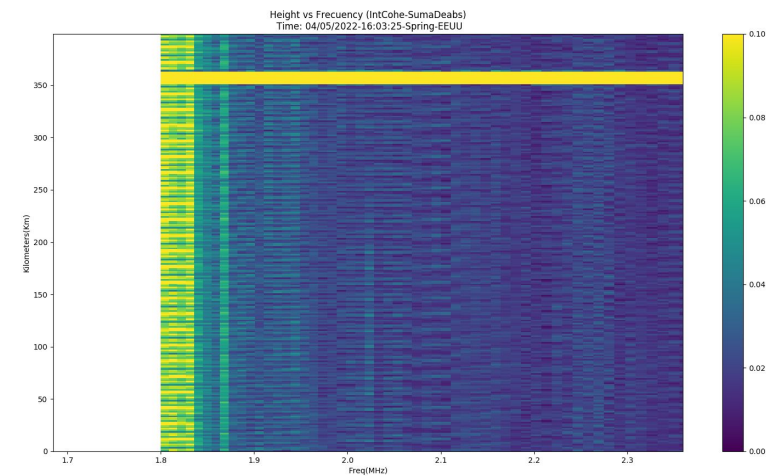


Indirect synchronization: time method

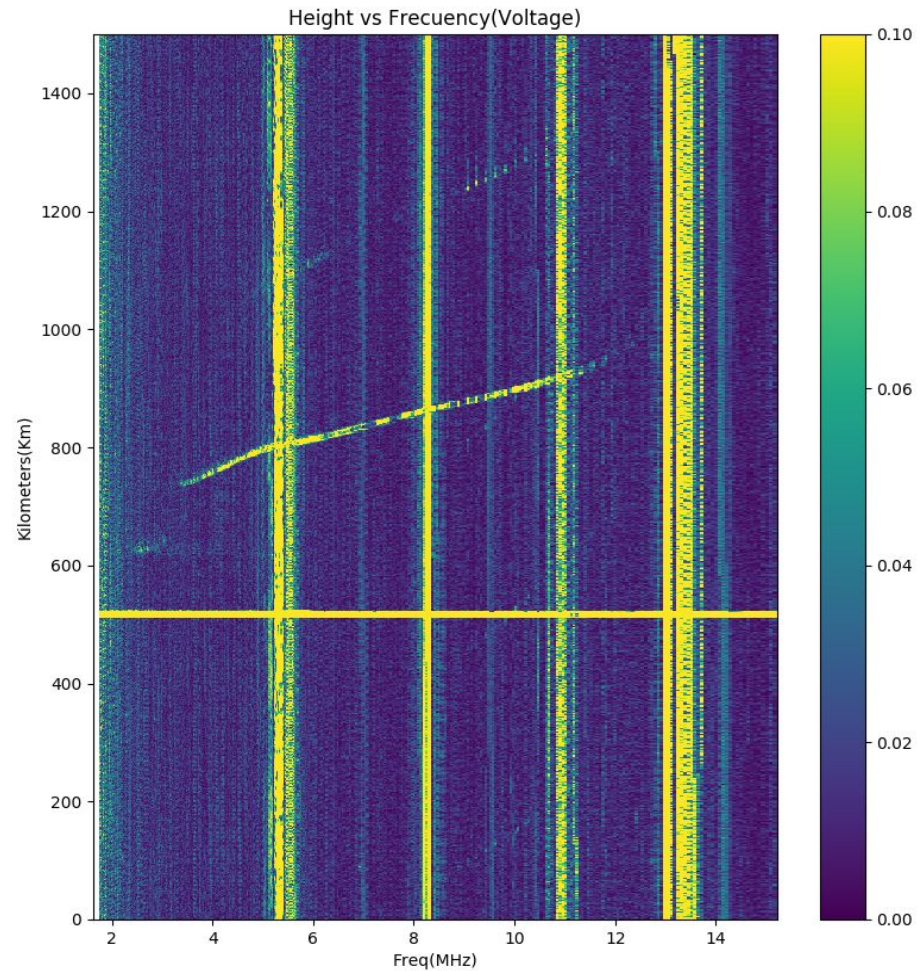
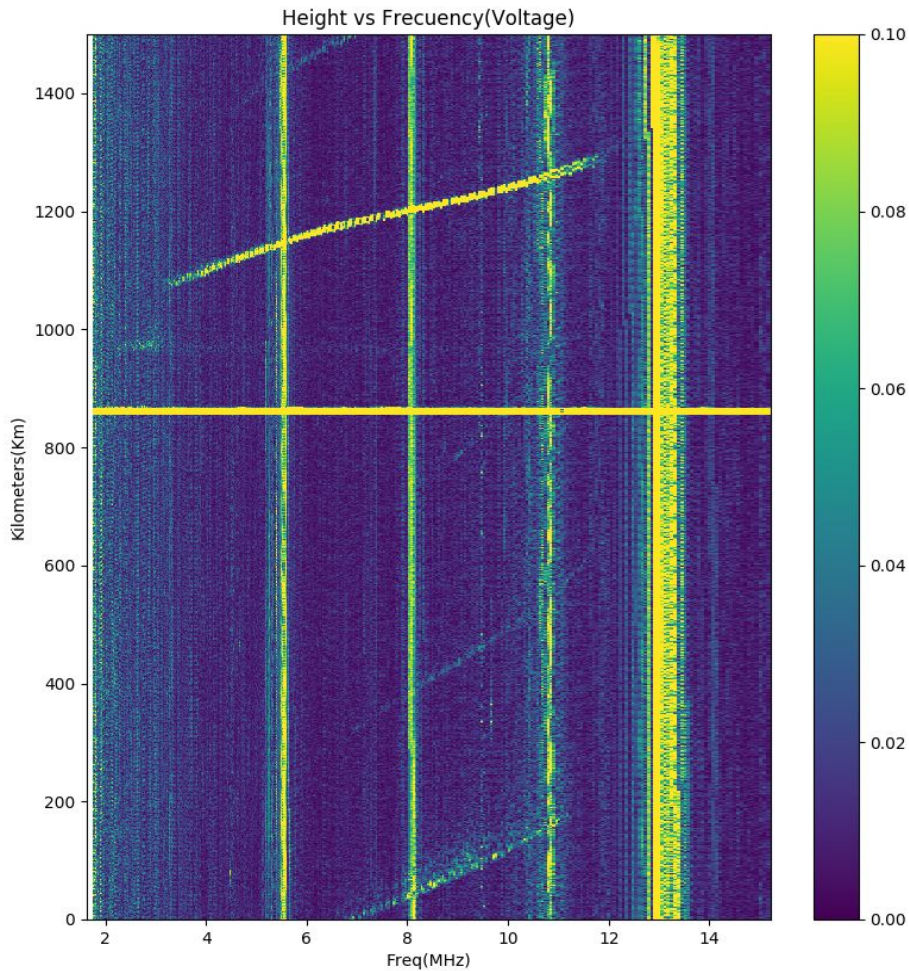
+960ms



We see that with 960ms added to the reception start time, we could synchronize with the transmission but we see the reference in the ionogram is not at zero level.

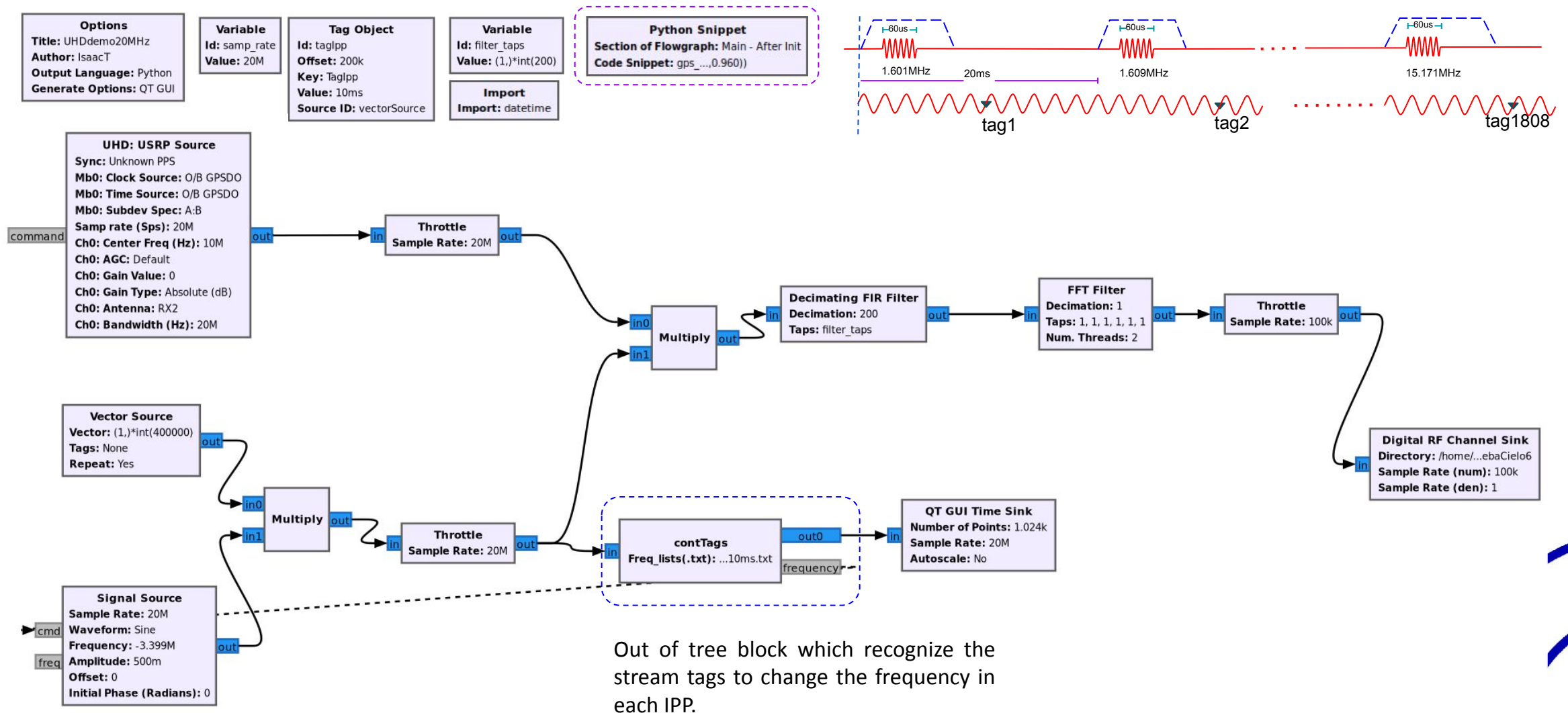


Indirect synchronization: time method



The reference level varied with the reception repetitions.

Indirect synchronization: PPS method



Indirect synchronization: PPS method

Properties: Python Snippet

General Advanced Documentation

Section of Flowgraph Main - After Init

Priority

```
gps_time=uhd.time_spec_t(self.uhd_usrp_source_0.get_mboard_sensor("gps_time").to_int()+1.0) #Adquiere la hora gps local
print('El valor de gps_time es:',gps_time)
self.uhd_usrp_source_0.set_time_next_pps(gps_time); # setea el tiempo gps en el siguiente PPS
```

```
Full_second=gps_time.get_full_secs() #Adquiere el epoch en seg.
UTC_time= datetime.datetime.fromtimestamp(Full_second) # Adquiere el tiempo en formato de UTC

year=int(UTC_time.strftime("%Y"))
print('El valor de year es:',year)
month=int(UTC_time.strftime("%m"))
print('El valor de month es:',month)
day=int(UTC_time.strftime("%d"))
print('El valor de day es:',day)
hour=int(UTC_time.strftime("%H"))
print('El valor de hour es:',hour)
minute=int(UTC_time.strftime("%M"))
print('El valor de minute es:',minute)
second=int(UTC_time.strftime("%S"))
print('El valor de second es:',second)
```

Code Snippet

```
if(hour>=5):
    dayN=day

if (hour==0):
    hour=24
    dayN=(day-1)
elif (hour==1):
    hour=25
    dayN=(day-1)
elif (hour==2):
    hour=26
    dayN=(day-1)
elif (hour==3):
    hour=27
    dayN=(day-1)
elif (hour==4):
    hour=28
    dayN=(day-1)
date_time = datetime.datetime(year, month, dayN, (hour-5), (minute+1), 4) # Se resta 5h para ponerlo en hora local (gps)
unix_time=time.mktime(date_time.timetuple())# epoch time en el cual comenzara la adquisicion por USRP
self.uhd_usrp_source_0.set_start_time(uhd.time_spec_t(unix_time,0.960))
```

UTC hour to
local time
one minute
before the
acquisition
second of the
acquisition

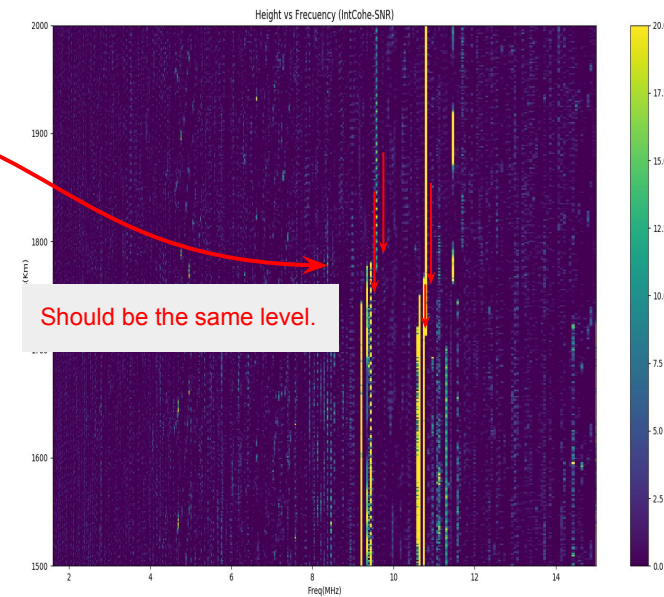
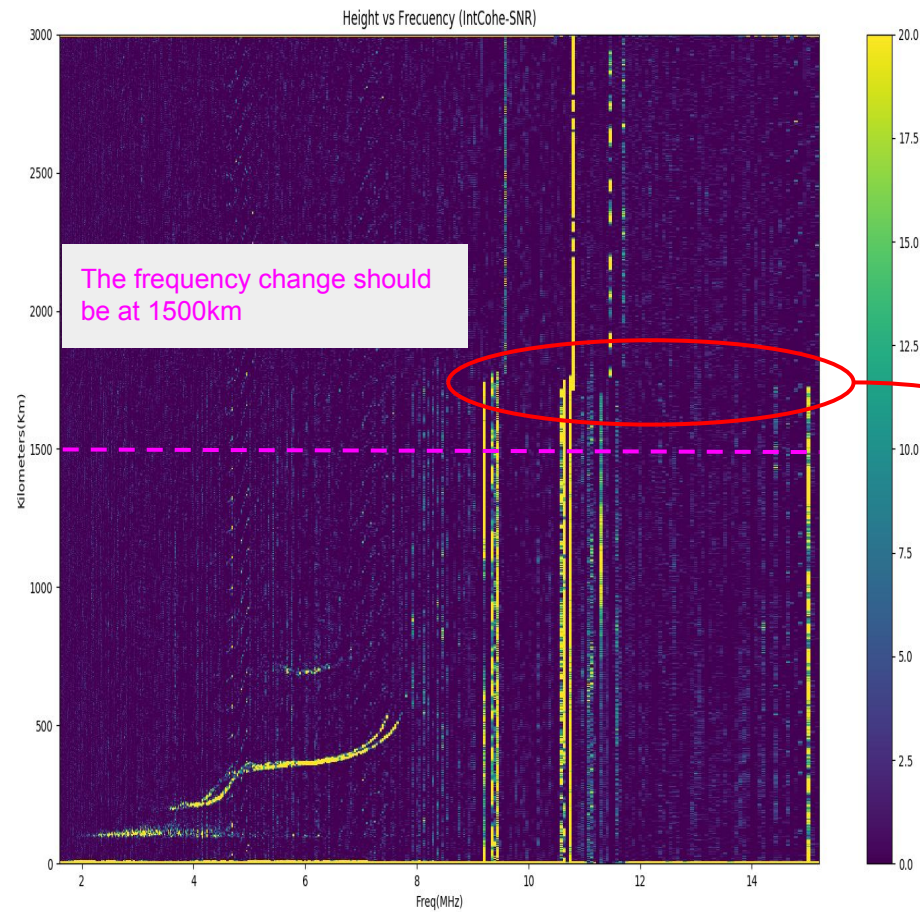
Set the USRP time registers by the GPSDO at the next pps

Acquire the UTC time parts by the epoch

Define the exact local time to begin the acquisition

As it was seen, we have to add 960ms to be synchronized with VIPIR

Indirect synchronization: PPS method



Conclusions

- We could make direct synchronization making the USRP N200 recognizes one external pulse generated at a specific time.
- In the indirect synchronization time method if we only consider the time given from the GPSDO to start the reception and we don't set the USRP N200 time, the ionogram reference level was not at 0 and it varied with reception repetitions.
- We had to add frequencies values in the file read by the message_strobe_jro_4 OOT block to be synchronized because of this block starts the sending of new frequency values by a second thread.
- In the indirect synchronization PPS method we set the GPSDO time to the USRP N200 and then started the reception.
- We created a new OOT block which recognized the tags and sent the new frequency values. Despite this block didn't include a second thread, we saw different levels in the ionogram where the frequency change was produced.

References

Device Synchronization

https://files.ettus.com/manual/page_sync.html

Message Passing

https://wiki.gnuradio.org/index.php/Message_Passing

Polymorphic Types (PMTs)

[https://wiki.gnuradio.org/index.php/Polymorphic_Types_\(PMTs\)](https://wiki.gnuradio.org/index.php/Polymorphic_Types_(PMTs))

Python Block Tags

https://wiki.gnuradio.org/index.php?title=Python_Block_Tags

Stream Tags

https://wiki.gnuradio.org/index.php?title=Stream_Tags

Contact:

isacctd92@gmail.com