iridium update

news since 2017

Just updates

Not a complete overview

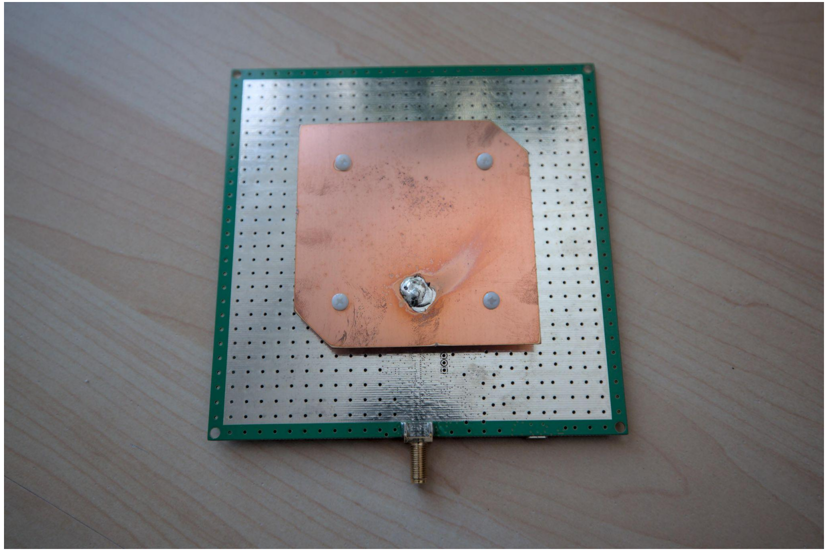
Not a howto

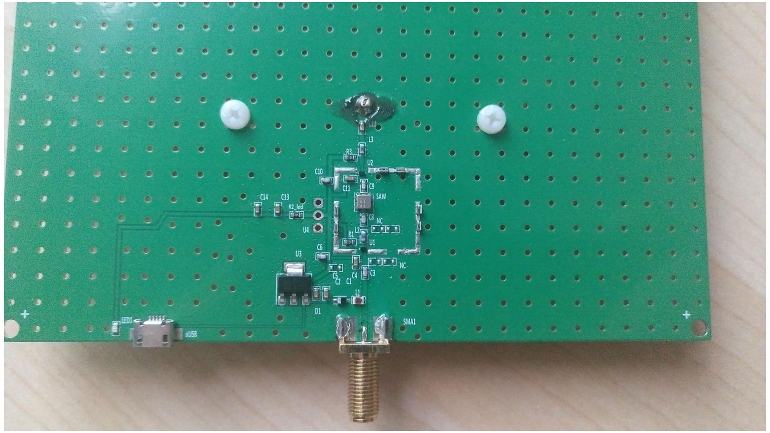
join us in #iridium on IRC (irc.blafasel.de) if you want to hack

Applications

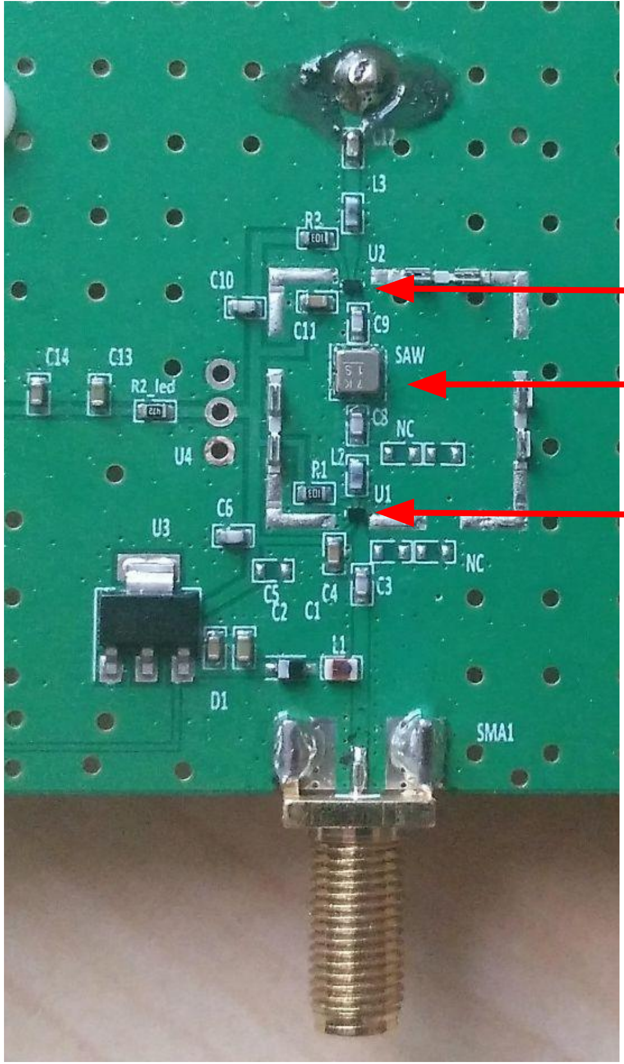
- Tracking
- Fleet management
- Mobile Data/Voice
- Emergency services
- Maritime sensors
- Aircraft comms
- Covert operations

RTL-SDR blog antenna

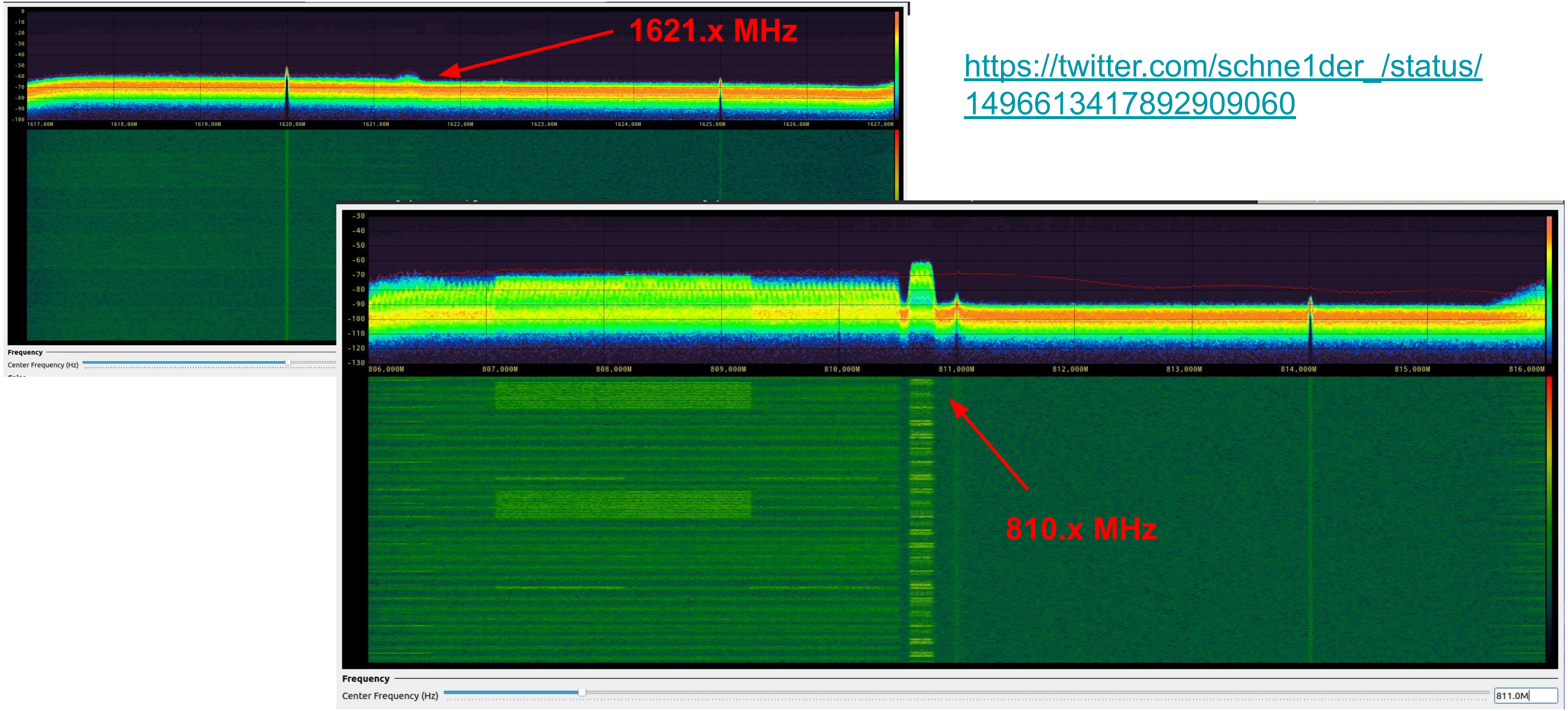




RTL-SDR blog antenna

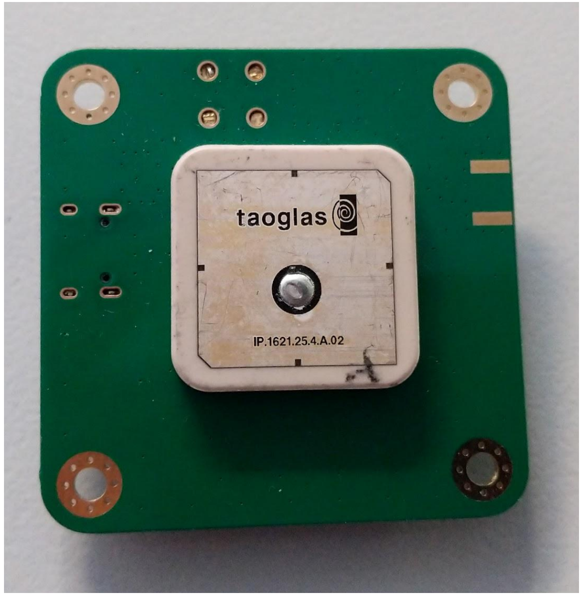


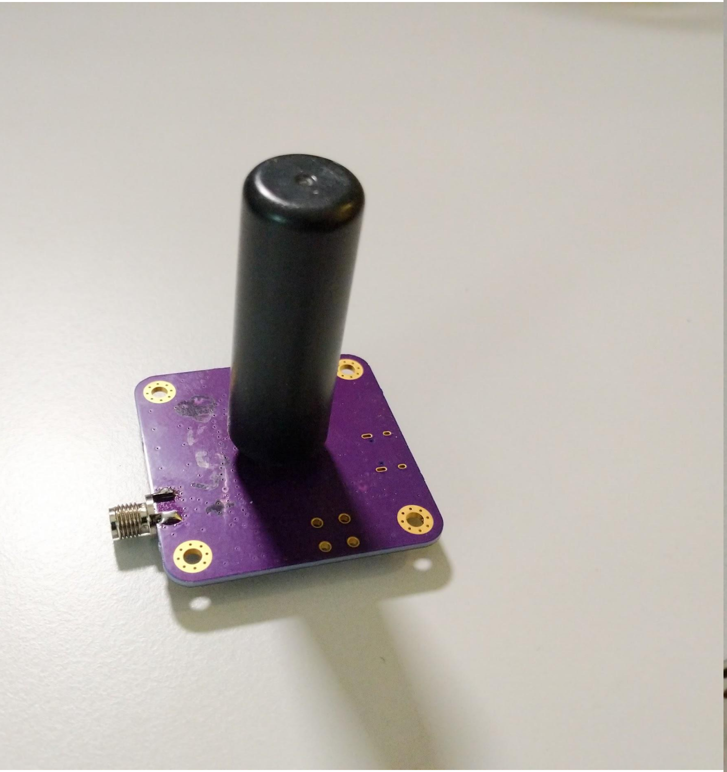
RTL-SDR blog antenna has bad out of band rejection

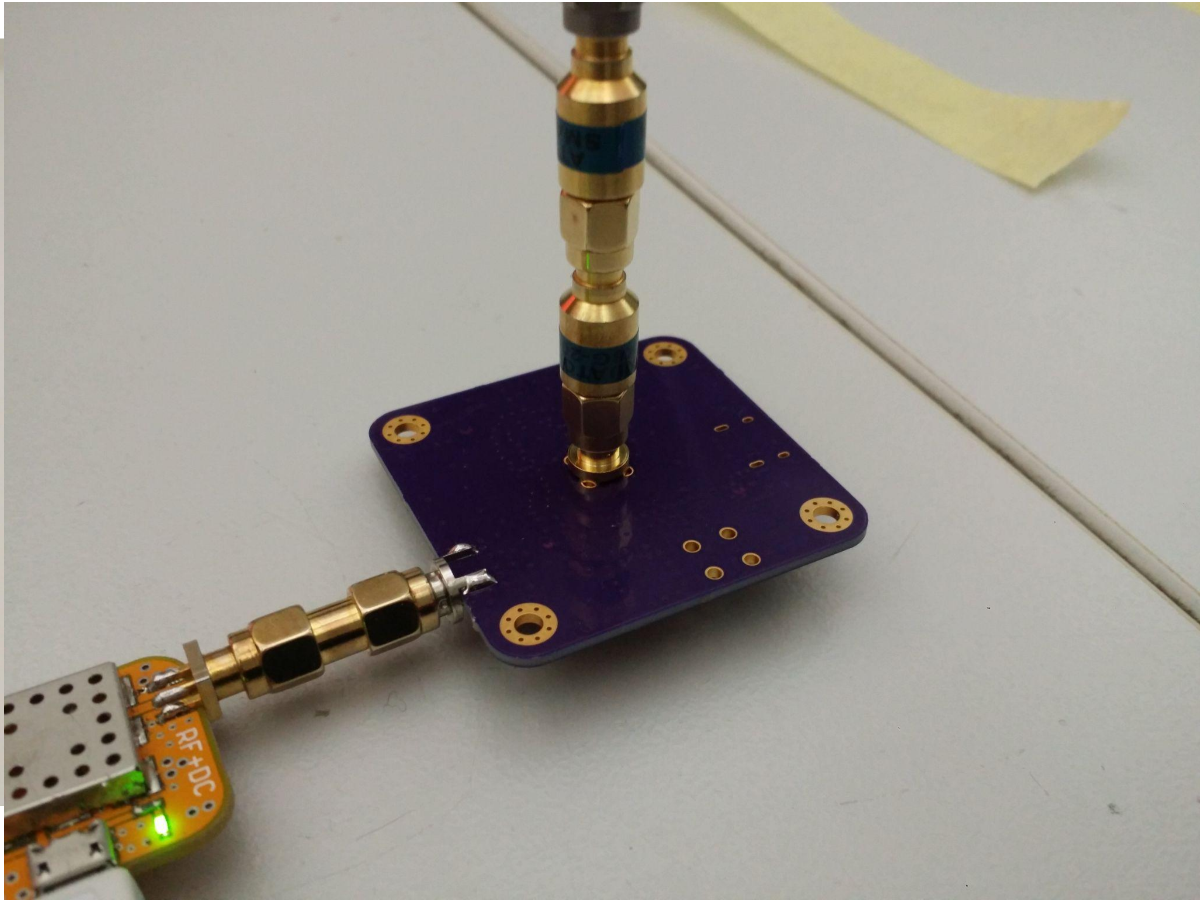


https://twitter.com/schne1der_/status/1496613417892909060

Own design based on tnt's Smart L-Band Antenna






RF+DC

Tallysman Antennas

HC610 Active Iridium Helical Antenna

Antennas

Coverage	Mount	Amplifier Gain	Connector Options
Iridium	Surface Mount / Direct Screw	28 dB typ.	SMA (male)

TW2643A Single Band GNSS Antenna with Receive-Only Iridium

Antennas

Coverage	Mount	Passive	Connector Options
GPS L1, GLONASS G1, Galileo E1, BeiDou B1, Iridium	Surface Mount / Magnet / Direct Screw / Adhesive		Many Options, Please Inquire

Recording at the ATA (Allen Telescope Array)

SETI/Gnuradio cooperation

Nice Infrastructure:

16/64-core 128G RAM

USRP N320 2 TX/RX
200 MHZ BW

Thanks:
Derek Kozel
Wael Farah

Polarization Test

Each antenna Feed with two linear polarizations X / Y

Iridium is right-hand circular polarized.

- 3db loss when receiving with linear polarization.

“Reconstruct” RHCP with

```
x + i * y
```

Polarization Test

Actual angle between x/y is unknown

- Physical angle at feed (most likely quite good)
- Cable length differences between x/y are unknown (esp. signal path to USRP) and will show up as angle difference
- if sending antenna is not dead-on, signal will not be exactly circular polarized

Polarization Test

We can vary the recombination “angle”:

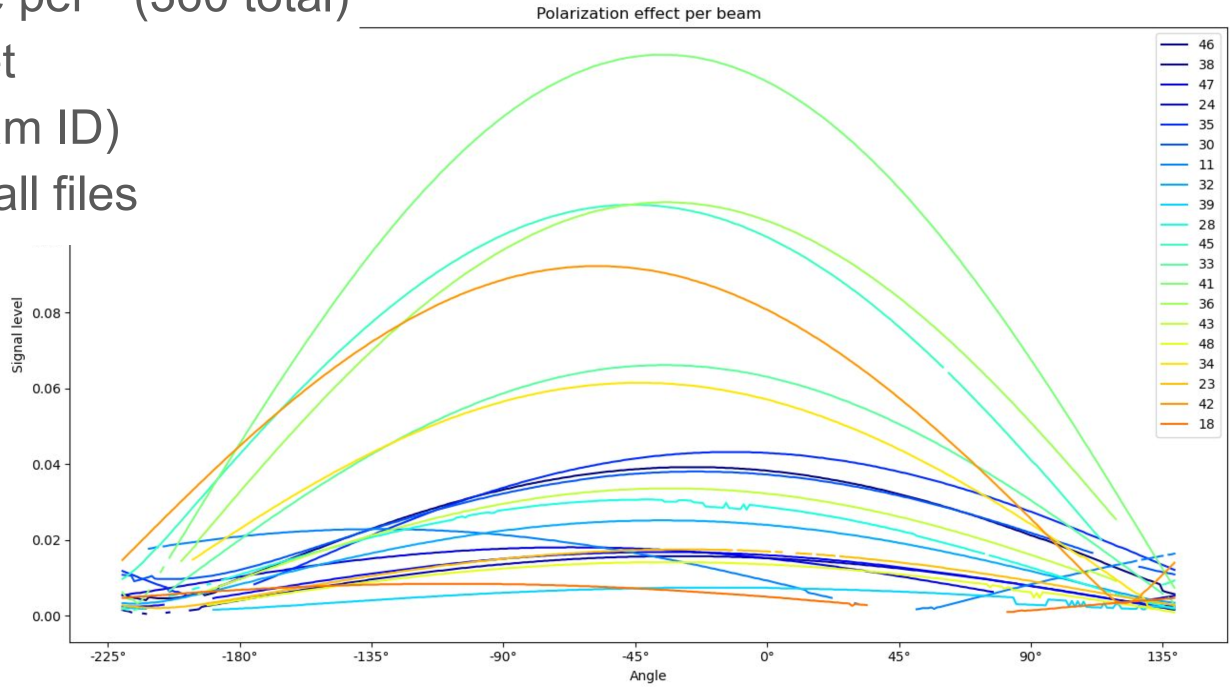
```
x+ cmath.rect(1,angle)*y
```

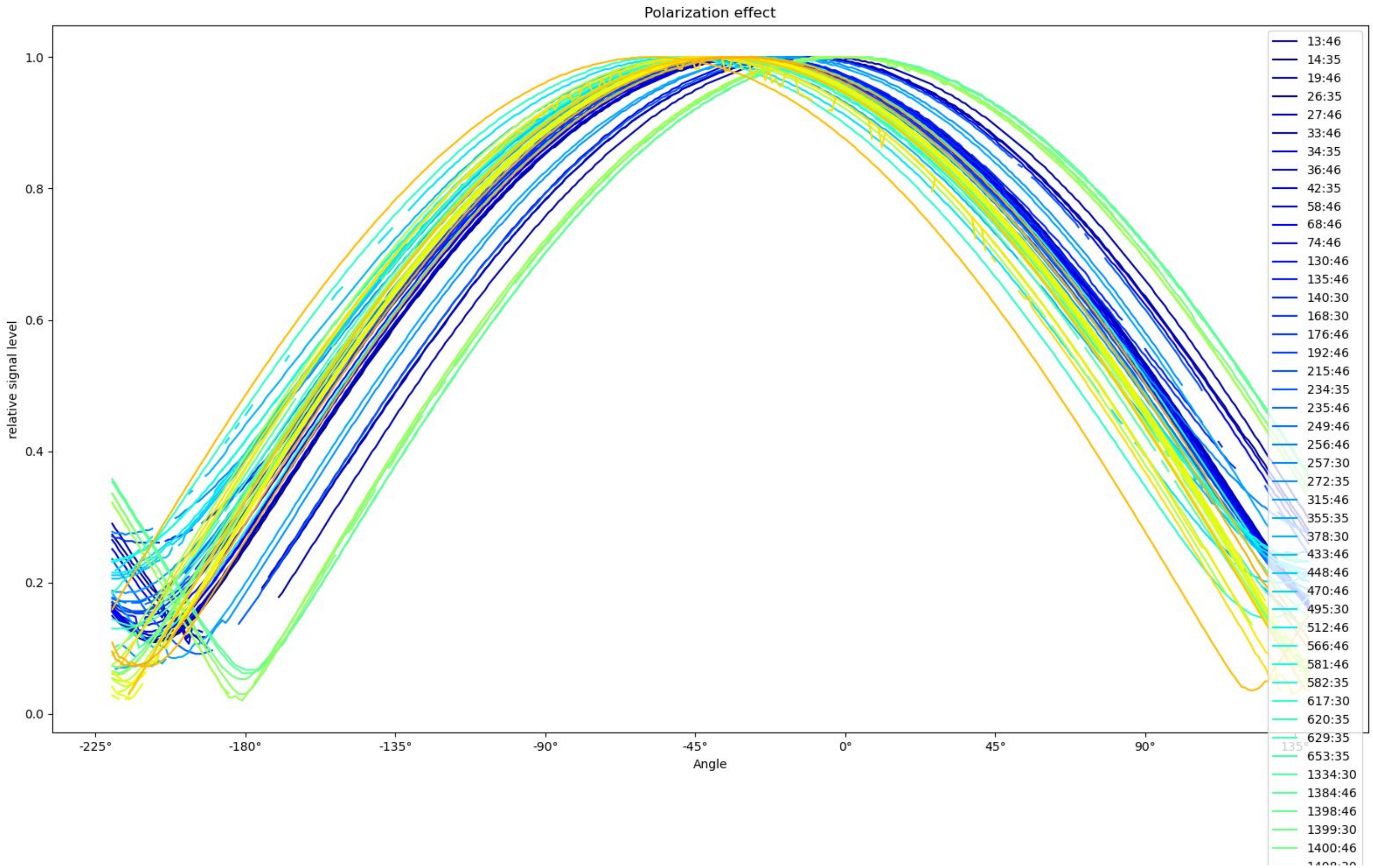
- Max signal strength should be about twice of one signal
  - 3dB gain over linear
- Min signal strength should be ~0 (LHCP / 180° opposite of max signal)

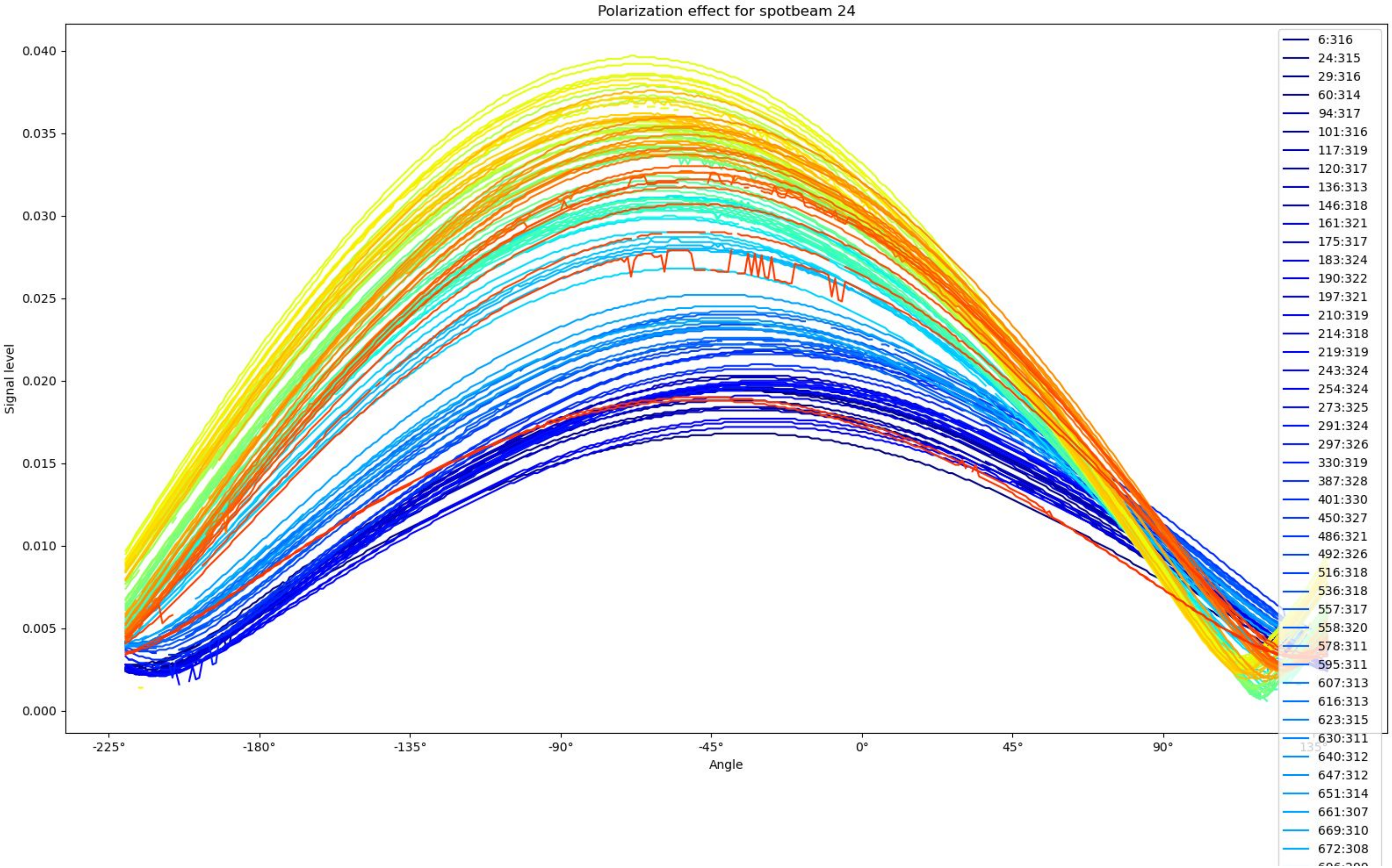
Polarization Test

Simple test:

- Generate one output file per ° (360 total)
- Pick a strong IBC packet (containing the spot beam ID)
- Search same packet in all files & note signal strength




Polarization effect
relative signal level
1.0
0.8
0.6
0.4
0.2
0.0
-225°
-180°
-135°
-90°
-45°
0°
45°
90°
135°
Angle
13:46
14:35
19:46
26:35
27:46
33:46
34:35
36:46
42:35
58:46
68:46
74:46
130:46
135:46
140:30
168:30
176:46
192:46
215:46
234:35
235:46
249:46
256:46
257:30
272:35
315:46
355:35
378:30
433:46
448:46
470:46
495:30
512:46
566:46
581:46
582:35
617:30
620:35
629:35
653:35
1334:30
1384:46
1398:46
1399:30
1400:46



Polarization effect for spotbeam 24
Signal level
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
-225°
-180°
-135°
-90°
-45°
0°
45°
90°
Angle
6:316
24:315
29:316
60:314
94:317
101:316
117:319
120:317
136:313
146:318
161:321
175:317
183:324
190:322
197:321
210:319
214:318
219:319
243:324
254:324
273:325
291:324
297:326
330:319
387:328
401:330
450:327
486:321
492:326
516:318
536:318
557:317
558:320
578:311
595:311
607:313
616:313
623:315
630:311
640:312
647:312
651:314
661:307
669:310
672:308

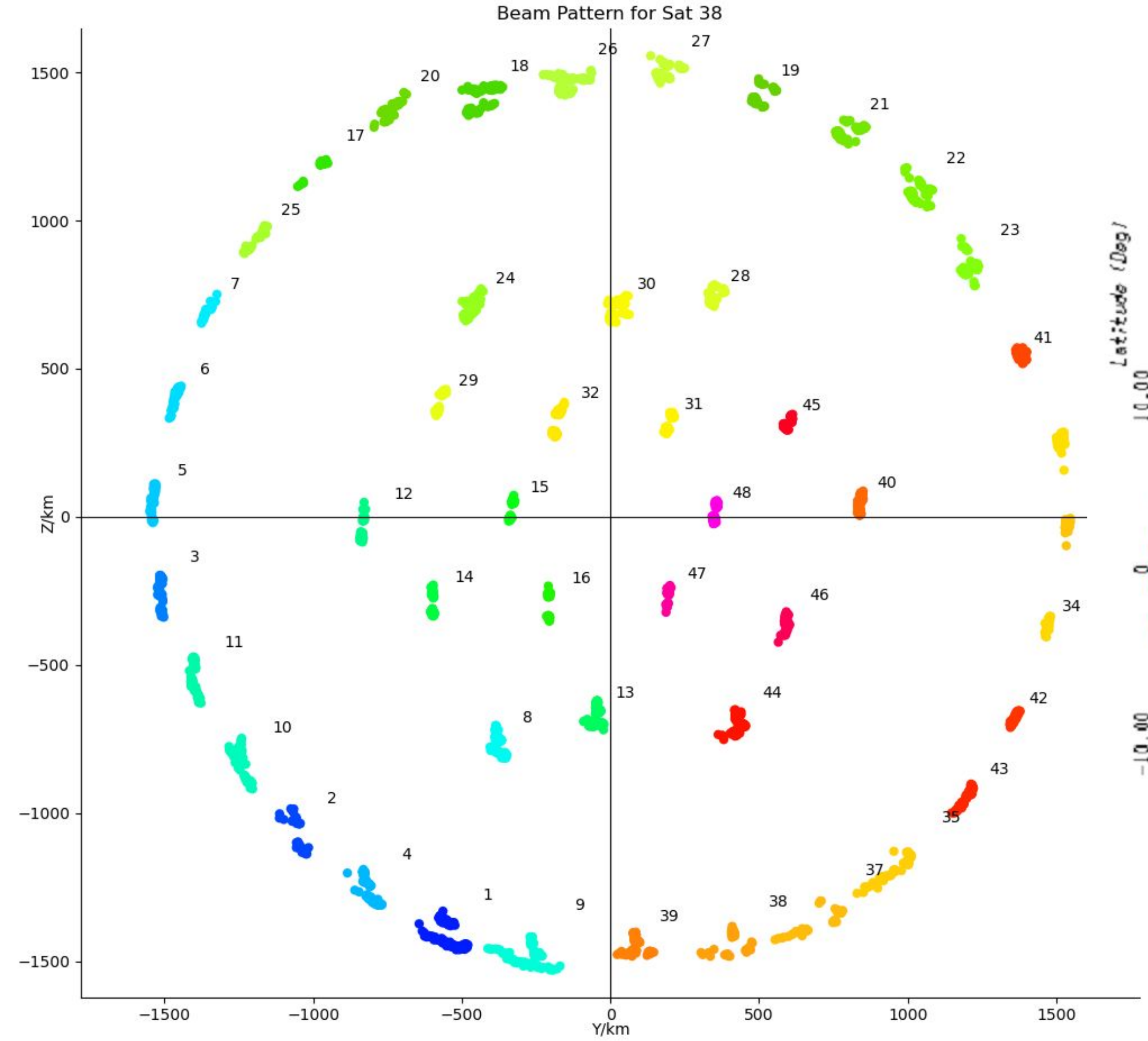


Exhibit A

Space Station Antenna Contours

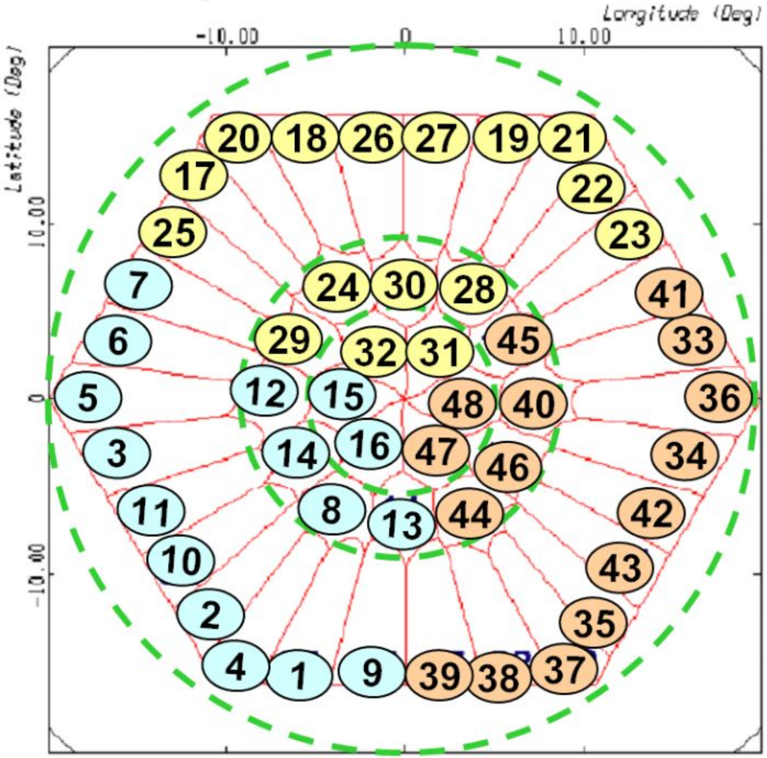
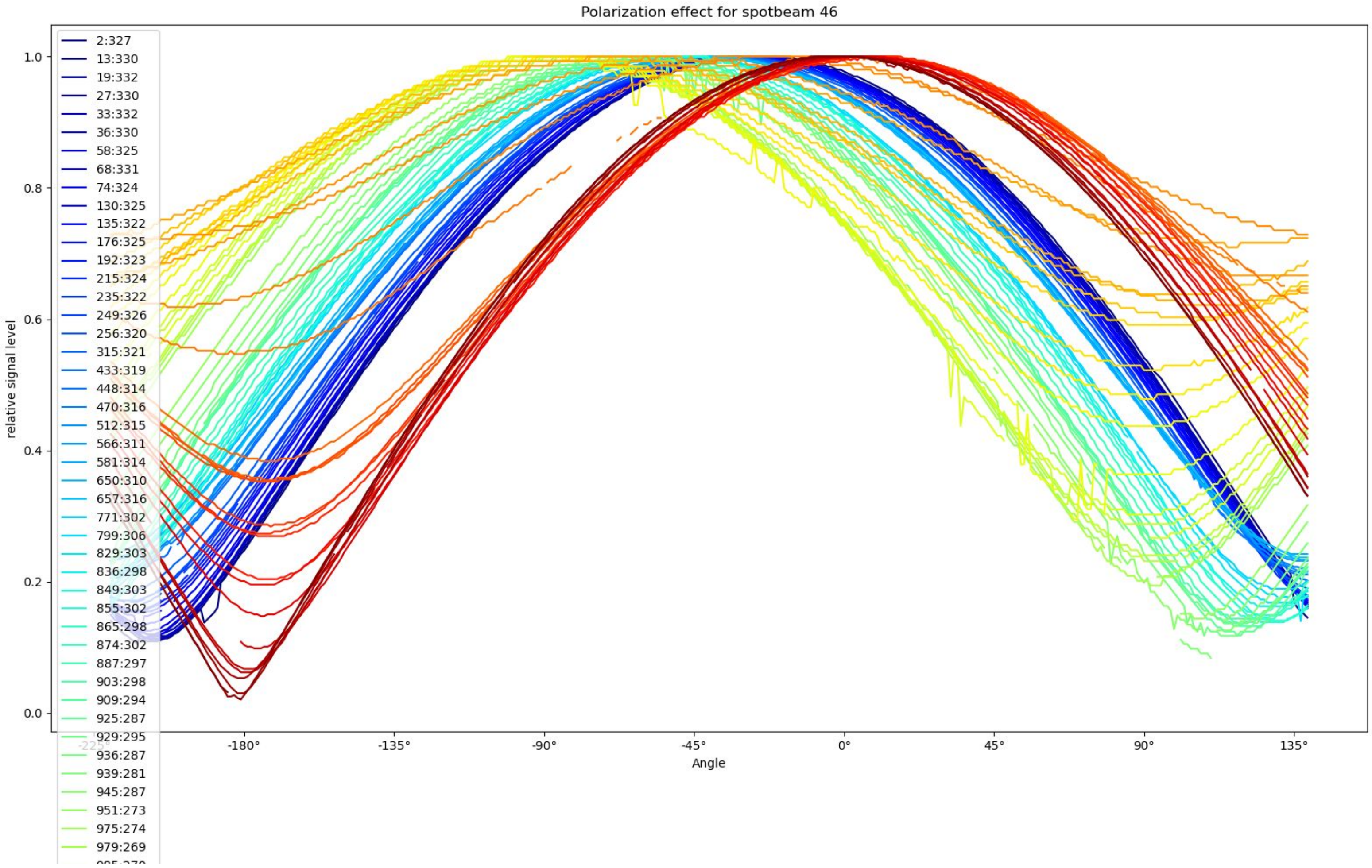
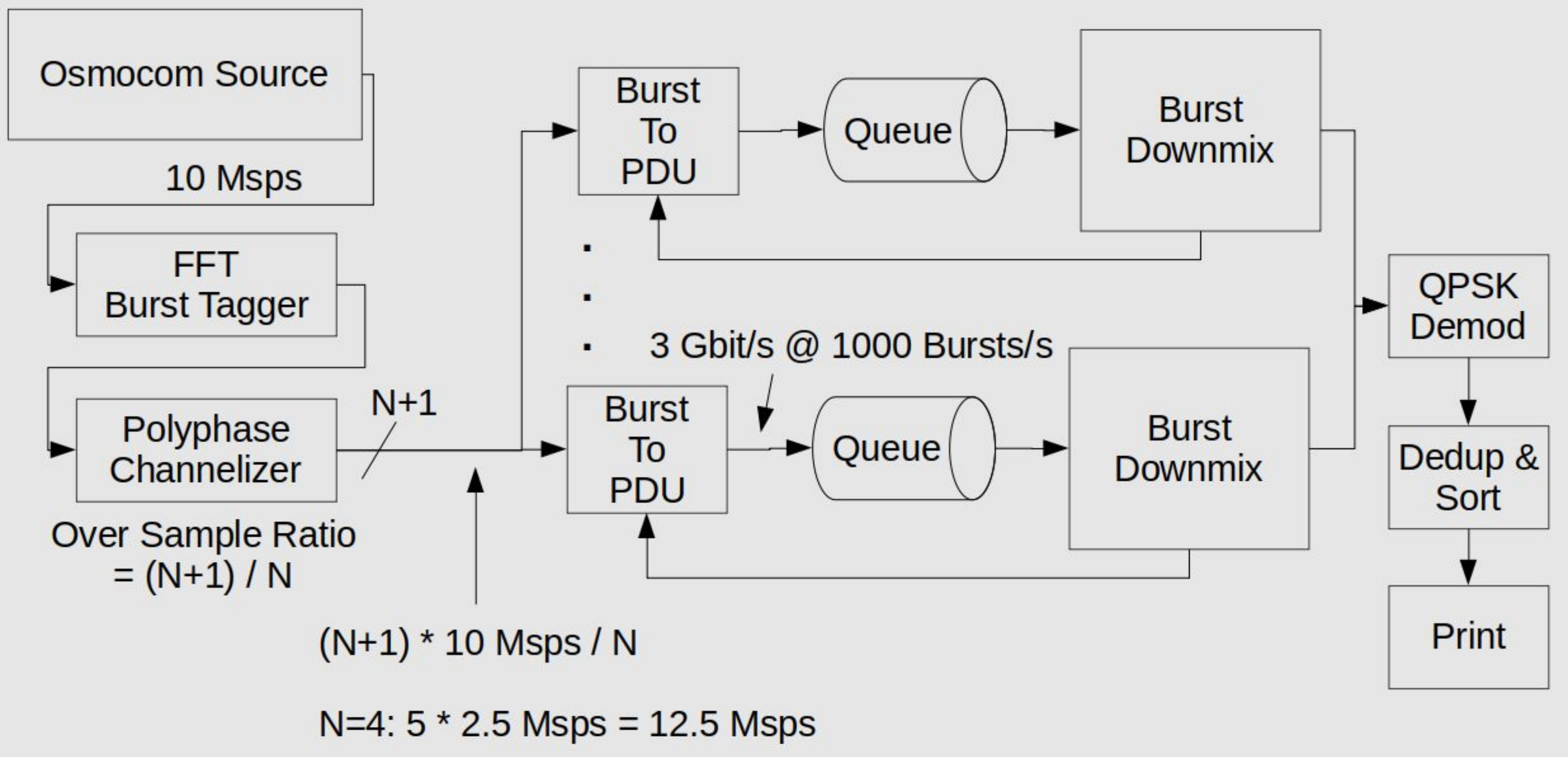


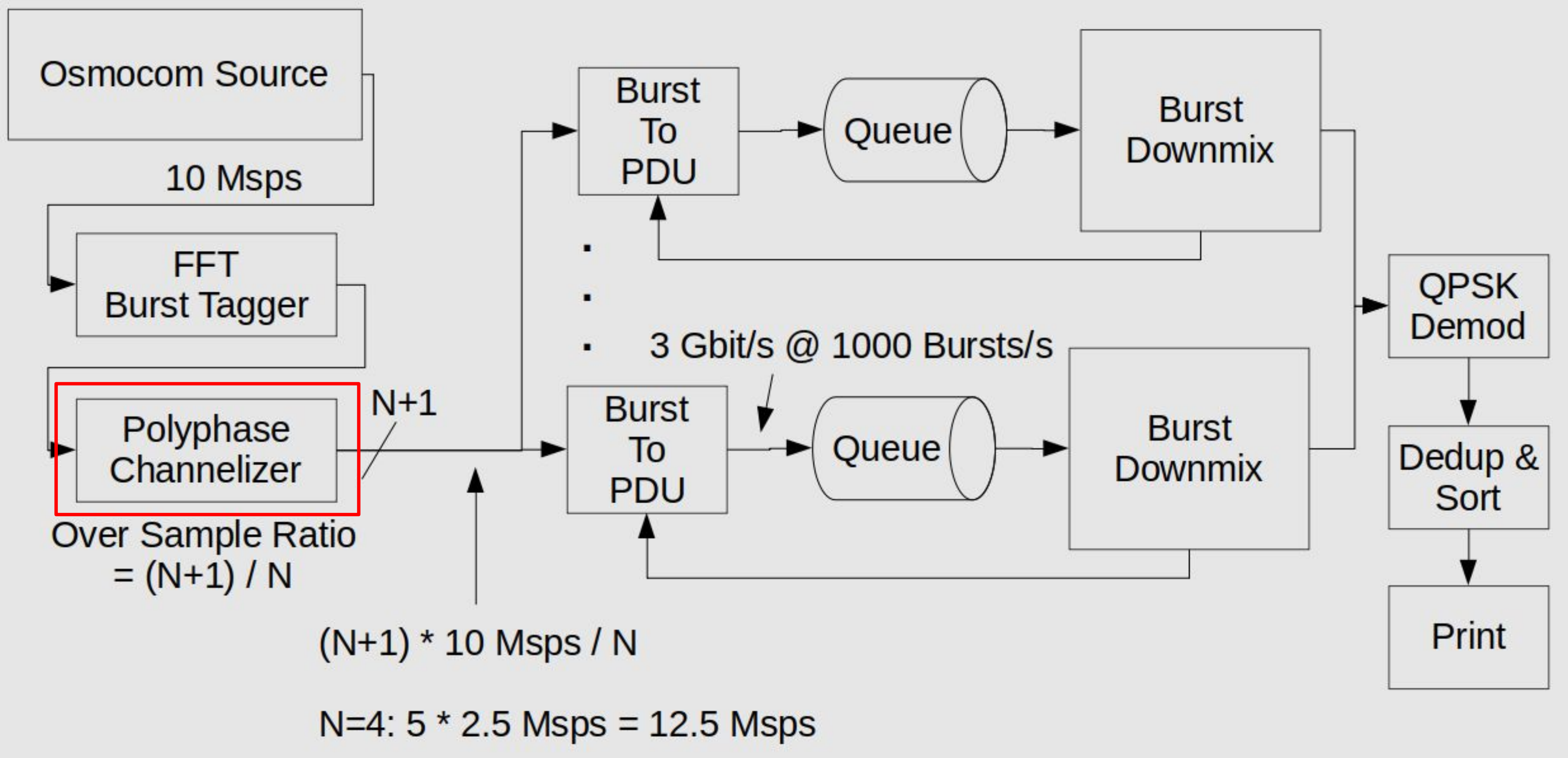
Figure A: Space Vehicle L Band Beam Layout and Numbering



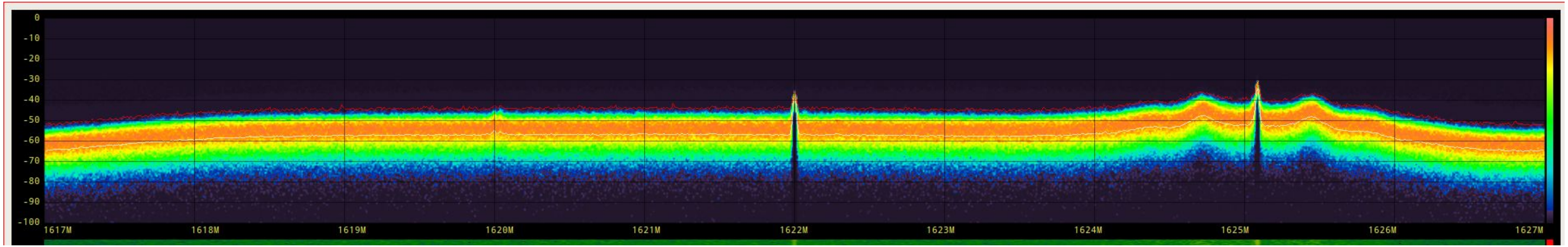
Polarization effect for spotbeam 46
relative signal level
1.0
0.8
0.6
0.4
0.2
0.0
-225°
-180°
-135°
-90°
-45°
0°
45°
90°
135°
Angle
2:327
13:330
19:332
27:330
33:332
36:330
58:325
68:331
74:324
130:325
135:322
176:325
192:323
215:324
235:322
249:326
256:320
315:321
433:319
448:314
470:316
512:315
566:311
581:314
650:310
657:316
771:302
799:306
829:303
836:298
849:303
855:302
865:298
874:302
887:297
903:298
909:294
925:287
929:295
936:287
939:281
945:287
951:273
975:274
979:269



Osmocom Source
10 Msps
FFT
Burst Tagger
Polyphase
Channelizer
N+1
Over Sample Ratio
= (N+1) / N
(N+1) * 10 Msps / N
N=4: 5 * 2.5 Msps = 12.5 Msps
Burst
To
PDU
Queue
Burst
Downmix
3 Gbit/s @ 1000 Bursts/s
Burst
To
PDU
Queue
Burst
Downmix
QPSK
Demod
Dedup &
Sort
Print

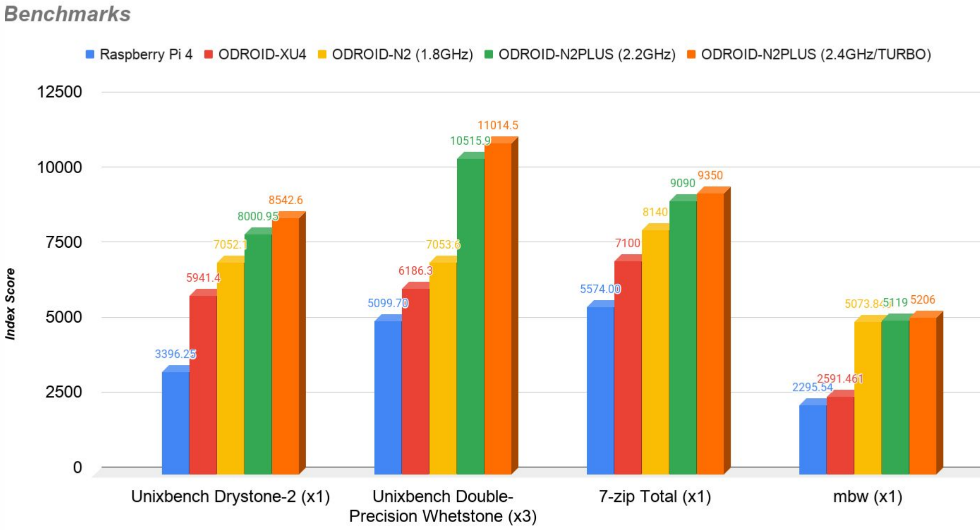

Osmocom Source
10 Msps
FFT
Burst Tagger
Polyphase
Channelizer
Over Sample Ratio
= (N+1) / N
N+1
(N+1) * 10 Msps / N
N=4: 5 * 2.5 Msps = 12.5 Msps
Burst
To
PDU
Queue
Burst
Downmix
3 Gbit/s @ 1000 Bursts/s
Burst
To
PDU
Queue
Burst
Downmix
QPSK
Demod
Dedup &
Sort
Print

RPI4 GbE is noisy :(

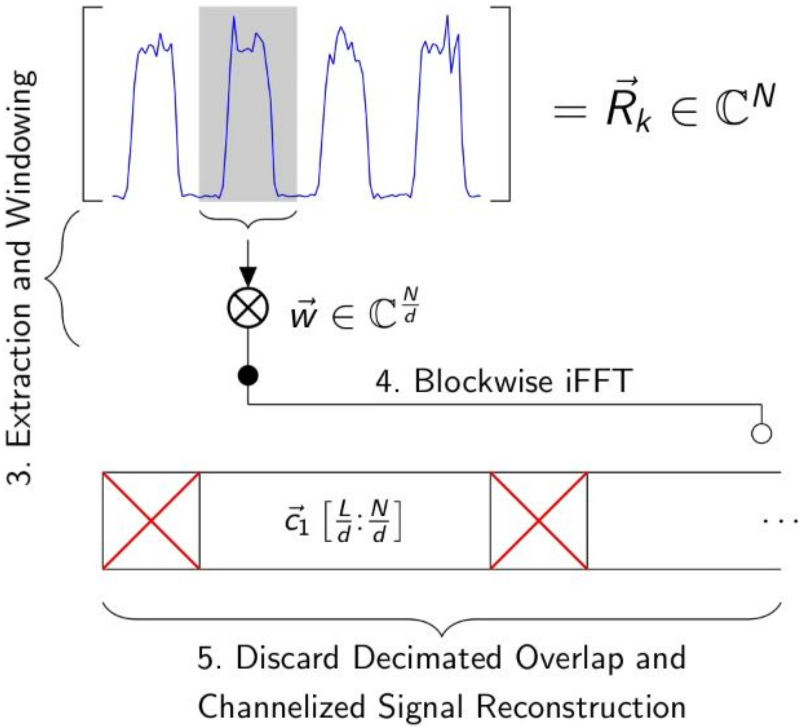


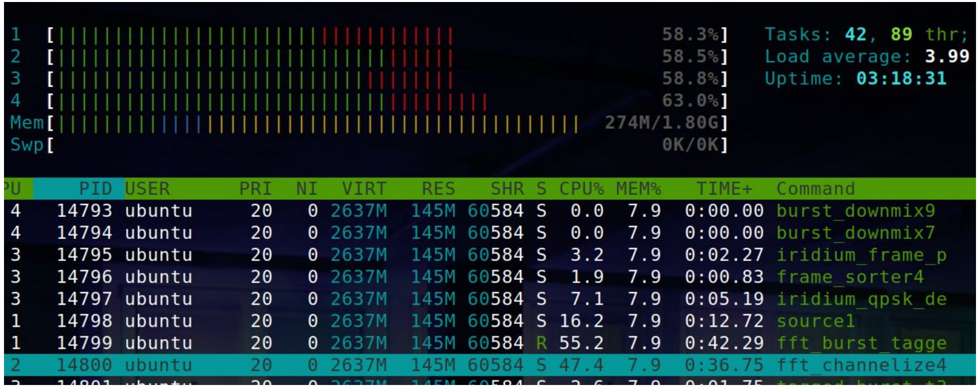
USB3 as well. Make sure to have good USB3 cables and keep your active antenna away from them

Next best thing: Odroid N2+



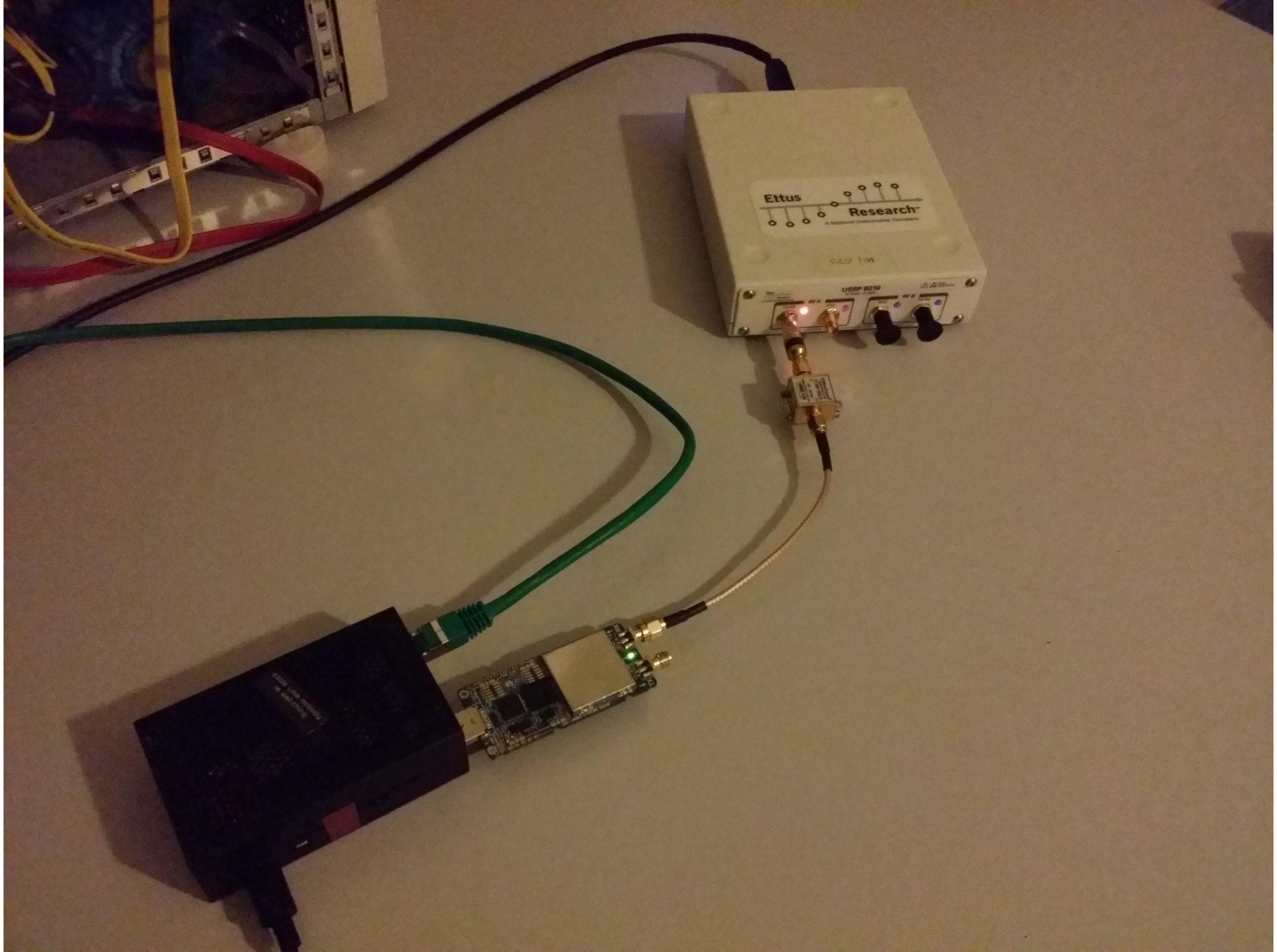
Most recent: Lazy FFT based channelizer



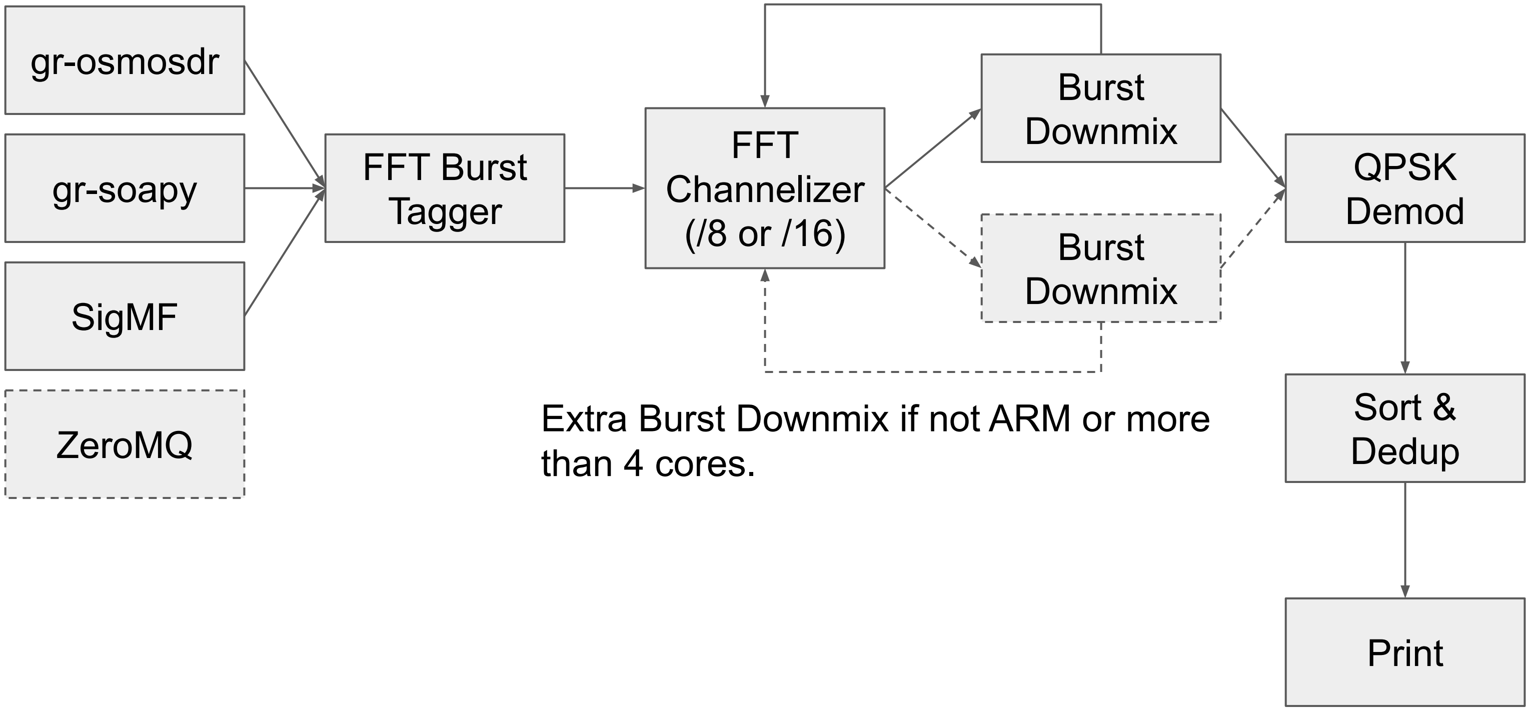


50% of a single core on a RPI4 @ 8 MSPS, channelizing into 17 channels.

https://media.ccc.de/v/gpn18-15-channelizing-with-gnuradio


Ettus
Research
USRP B210

gr-iridium current state



iridium-extractor with ZMQ as sample source

Experimental. Goal is to feed one machine with samples from two antennas connected to an USRP B210 to compare them.

6 examples/zeromq-sub.conf

```
@@ -0,0 +1,6 @@
+[zeromq-sub-source]
+
+sample_rate=10000000
+center_freq=1622000000
+address=tcp://127.0.0.1:5000
+pass_tags=True
```

SigMF support

Enhanced offline mode of iridium-extractor

```
iridium-extractor -c 1626000000 -r 2000000 -f float
name-f1.626000e+09-s2.000000e+06-t20160401000000.cfile >
output.bits
```

New:

```
iridium-extractor recording-test.sigmf-data > output.bits
```

“proper” SigMF support

```
iridium-extractor recording-test.sigmf
```

Problem: It’s a .tar file

Gnuradio can’t read a tar file.

Similar issue with .wav

```
iridium-extractor -c 1622000000 baseband.wav
```

file_object source

Solution: (python) source block that accepts a file-like object

```
def work(self, input_items, output_items):
        items=len(output_items[0])
        count=items*self.itemsize
        buf = self.fileobject.read(count)
        …
```

A little slower, but works.

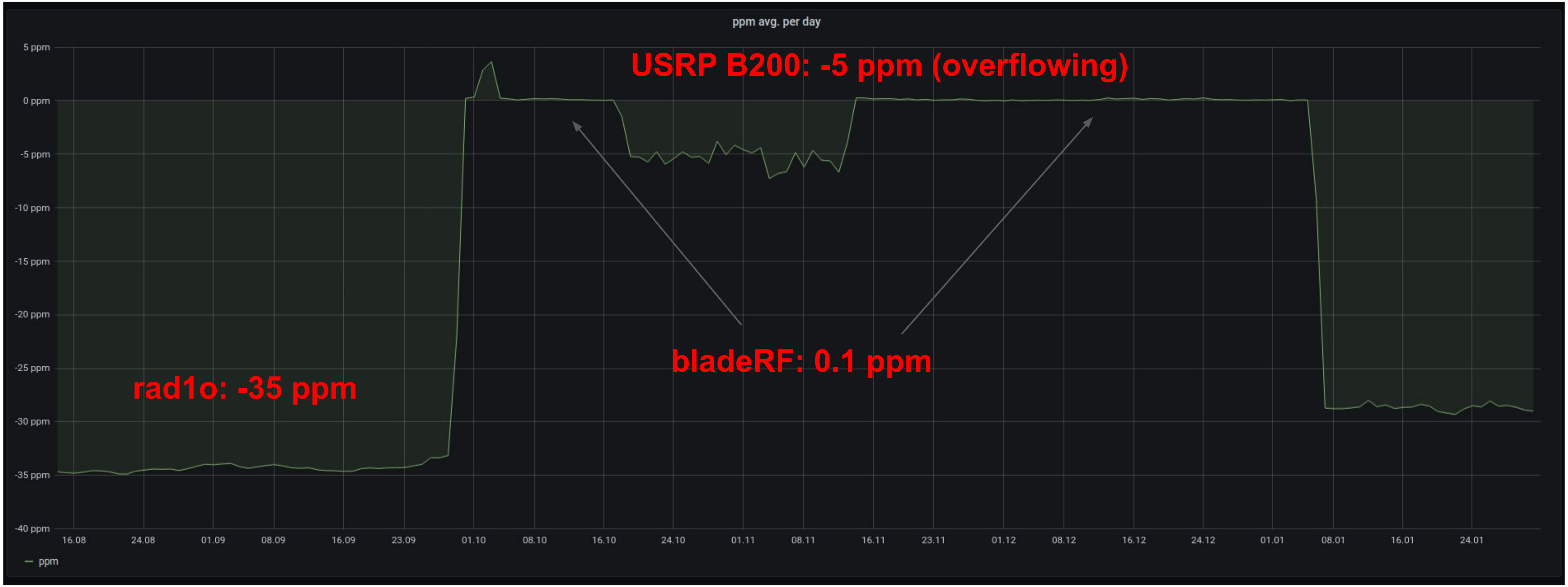
SigMF support (2)

Also for output files:

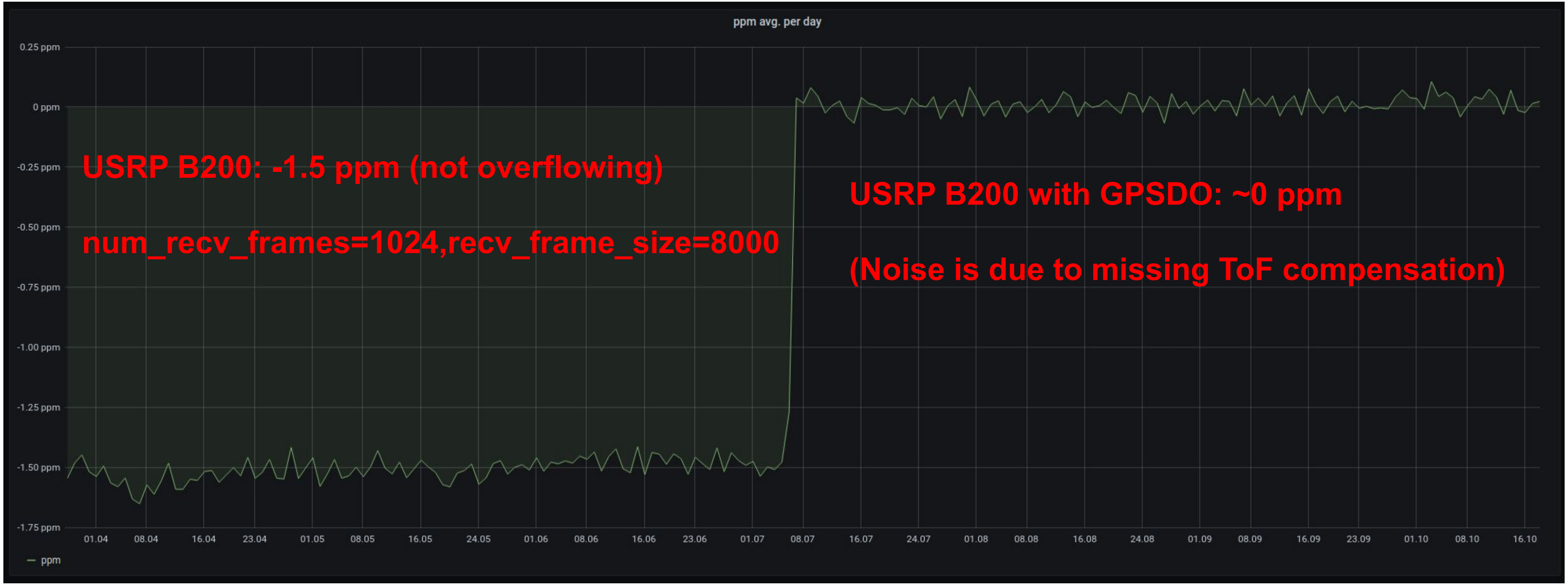
```
iridium-extractor --raw-capture debugfile
```

Now creates proper .sigmf-meta with sample_rate etc.

Frequency stability of SDRs



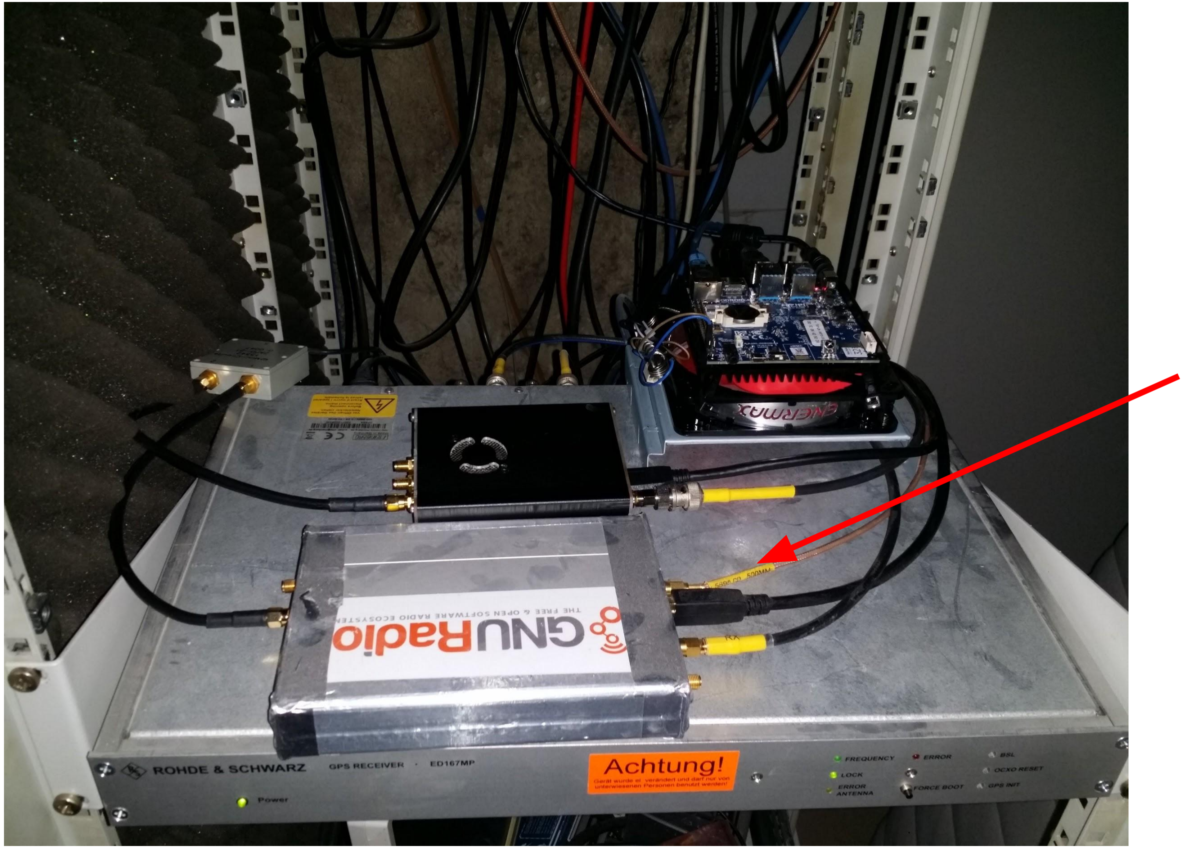
GPSDO


ppm avg. per day
0.25 ppm
0 ppm
-0.25 ppm
-0.50 ppm
-0.75 ppm
-1.00 ppm
-1.25 ppm
-1.50 ppm
-1.75 ppm
01.04
08.04
16.04
23.04
01.05
08.05
16.05
24.05
01.06
08.06
16.06
23.06
01.07
08.07
16.07
24.07
01.08
08.08
16.08
24.08
01.09
08.09
16.09
23.09
01.10
08.10
16.10
ppm
USRP B200: -1.5 ppm (not overflowing)
num_recv_frames=1024,recv_frame_size=8000
USRP B200 with GPSDO: ~0 ppm
(Noise is due to missing ToF compensation)

Timestamping

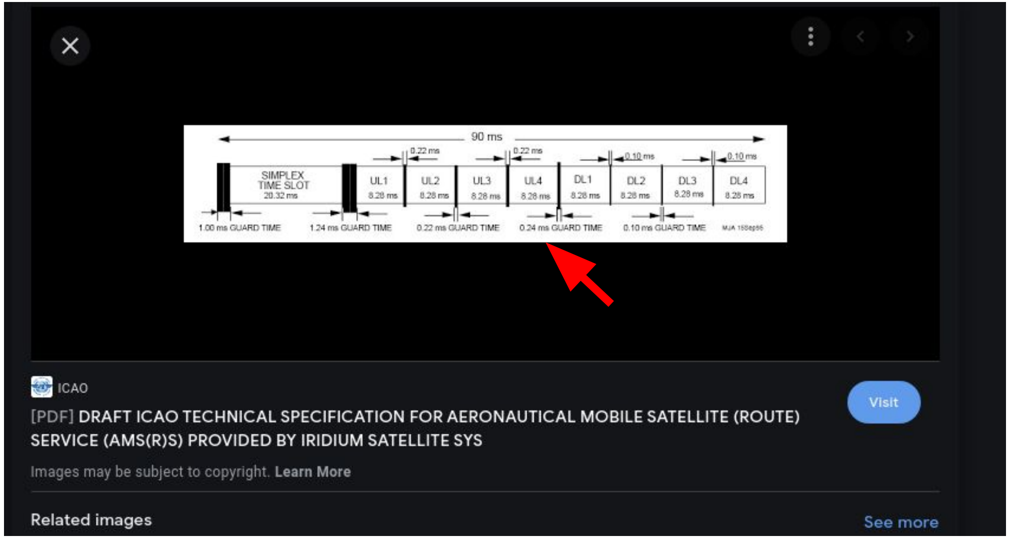
Spent significant amount of effort to timestamp Iridium frames to sub symbol accuracy.

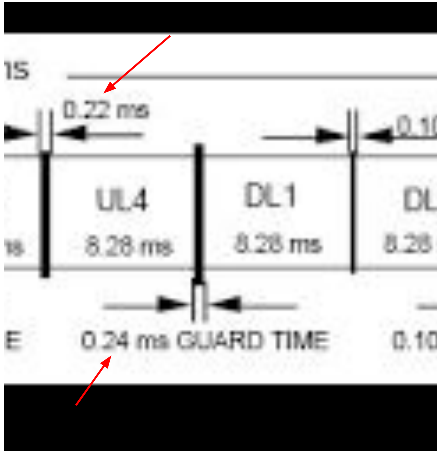
USRP B200 together with PPS from a Mainberg GPS receiver is used to timestamp.



Timestamping

Timestamps of IBC packets did not match expectation. Eventually figured out that the frame structure as repeated over the Internet is not accurate:
Guard time after last uplink slot is 0.24 ms and not 0.22 ms:

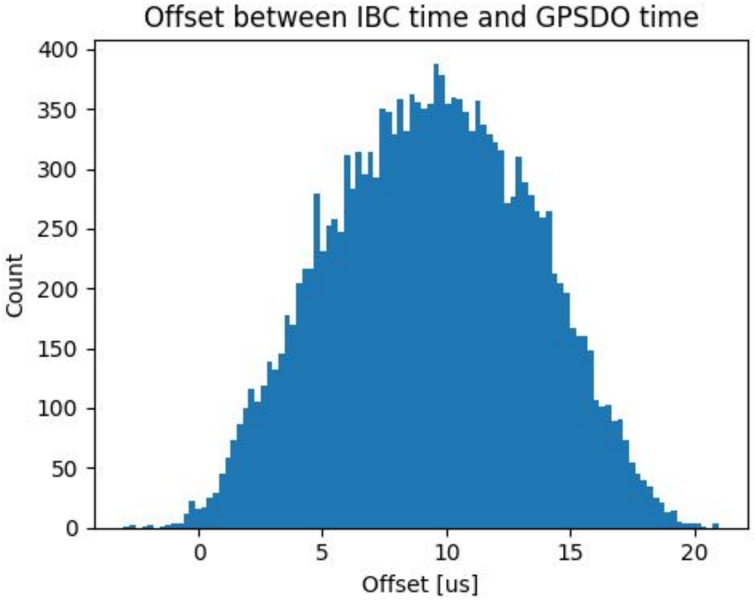




http://www.decodesystems.com/iridium.html now carries the correct timings.

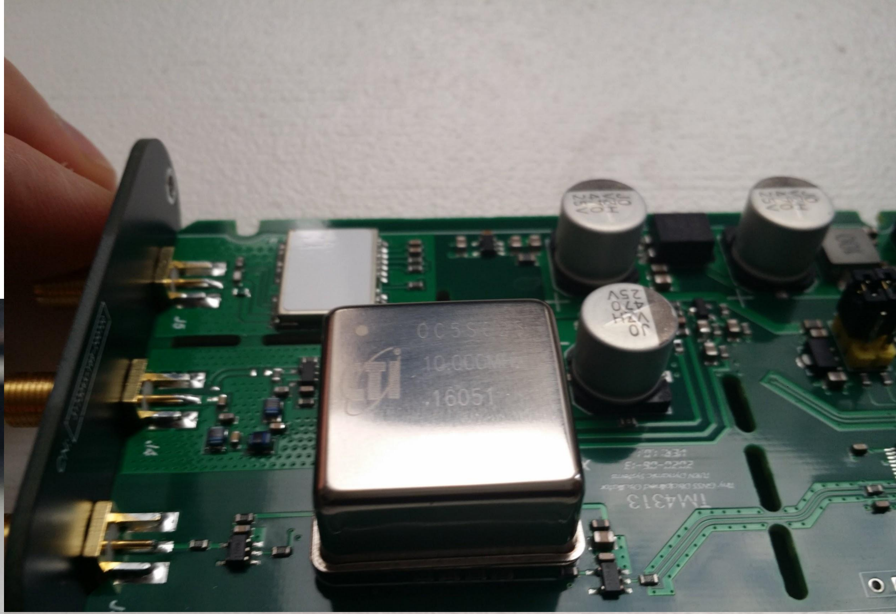
Timestamping

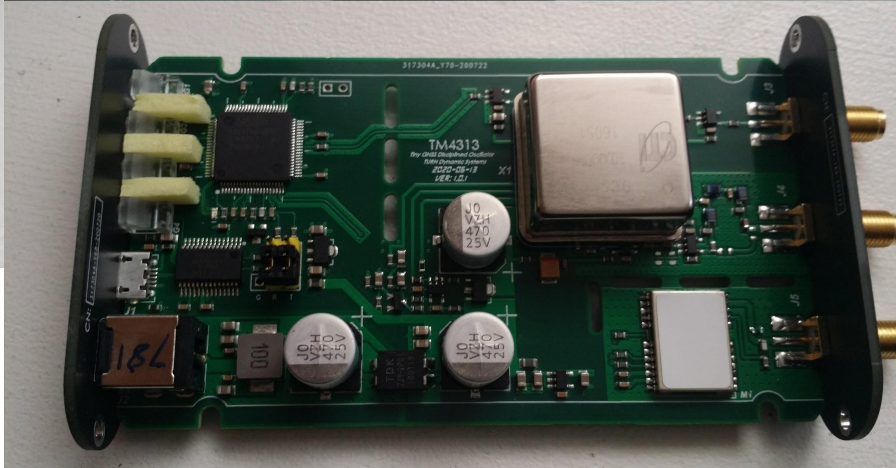
We still have an unexplained offset to ToF compensated Iridium time of around 9.5 us:




```
Blob:
- Start Itime  : 2022-03-24T23:00:38.516014589
- End   Itime  : 2022-03-25T03:02:17.363448704
- Start Utime  : 2022-03-24T23:00:38.516019456
- End   Utime  : 2022-03-25T03:02:17.363459328
- Runtime      : 4:01:38
- PPM          : 0.000
rec.tmin -0.000003
rec.tmax 0.000021
rec.ppm 0.000
dist_min: 781565.1924488014
t_min: 2022-03-24T23:44:41.834300
delta_min: 0.002623392
median 9.468
average 9.468
```

Cheap GPSDO with PPS

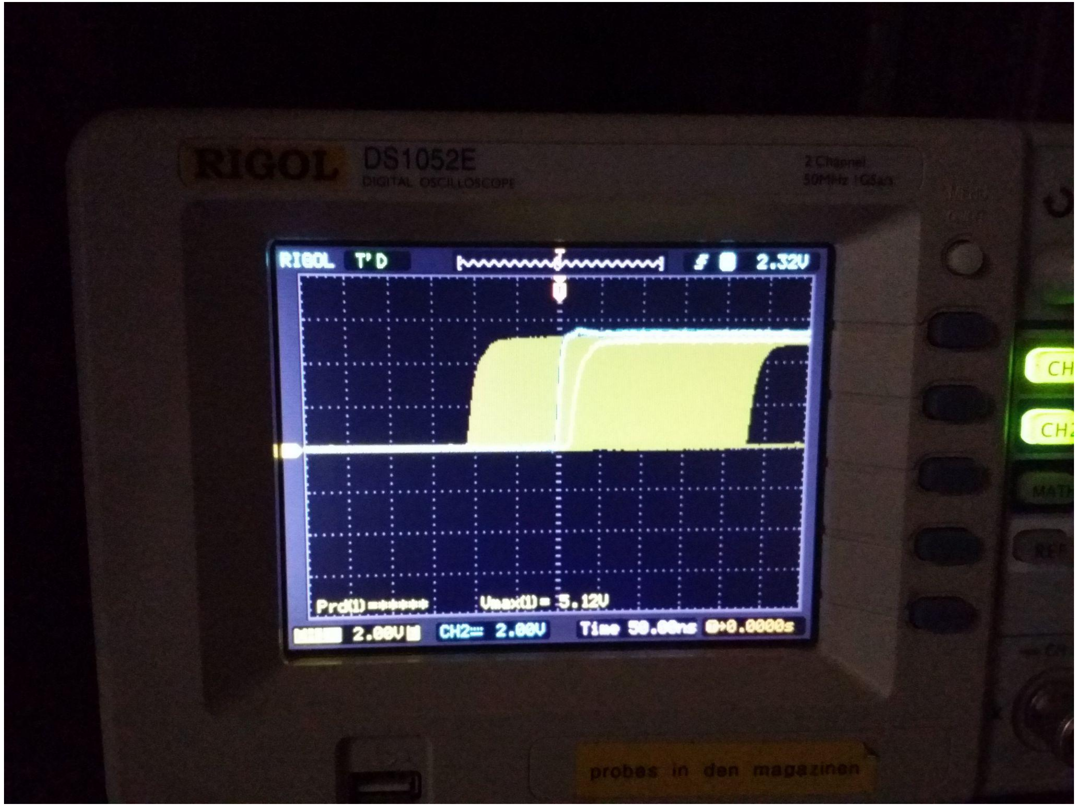




Cheap GPSDO with PPS

Some jitter between the Mainberg GPSDO and the cheap GPSDO. Needs further investigation.

Picture shows around 350 ns jitter.



CI

```yaml
- name: Unit Tests
  run: |
   cd build
   make test
- name: Demod PRBS15 SigMF
  run: |
   iridium-extractor test-data/prbs15-2M-20dB.sigmf-meta |grep ^RAW > prbs15-2M-20dB.bits
   grep "RAW: prbs15-2M-20dB 0000599.9996 1622000000 N:32.12-80.05 I:00000000000 100% 0.13551 179 0011000000110000111100111000000000
- name: Demod with decimation 4
  run: |
   iridium-extractor -D 4 test-data/prbs15-2M-20dB.sigmf-meta |grep ^RAW > prbs15-2M-20dB-D4.bits
   grep "RAW: prbs15-2M-20dB 0000599.9996 1622000000 N:32.12-80.05 I:00000000000 100% 0.13577 179 0011000000110000111100111000000000
- name: Demod with decimation 8
  run: |
   iridium-extractor -D 8 test-data/prbs15-2M-20dB.sigmf-meta |grep ^RAW > prbs15-2M-20dB-D8.bits
   grep "RAW: prbs15-2M-20dB 0000599.9996 1622000000 N:32.12-80.05 I:00000000000 100% 0.13643 179 0011000000110000111100111000000000
- name: Test raw samples
  run: |
   ln -s prbs15-2M-20dB.sigmf-data test-data/prbs15-2M-20dB.fc32
   iridium-extractor --offline -c 1622000000 -r 2000000 -f float test-data/prbs15-2M-20dB.fc32 |grep ^RAW > prbs15-2M-20dB.bits.raw
   cmp prbs15-2M-20dB.bits prbs15-2M-20dB.bits.raw
- name: Test SigMF Archive support
  run: |
   tar cf test-data/prbs15-2M-20dB.sigmf test-data/prbs15-2M-20dB.sigmf-*
   iridium-extractor test-data/prbs15-2M-20dB.sigmf |grep ^RAW > prbs15-2M-20dB.bits.archive
   cmp prbs15-2M-20dB.bits prbs15-2M-20dB.bits.archive
```

SNR Estimation

https://www.sjsu.edu/people/burford.furman/docs/me120/FFT_tutorial_NI.pdf

https://www.ap.com/blog/fft-spectrum-and-spectral-densities-same-data-different-scaling/

Now applying scaling factors and effective noise bandwidth inside burst tagger to get a better SNR estimate from the burst tagger.

QPSK demod creates a signal level. Can be used with channel noise estimate from burst tagger.

Table 3. Correction Factors and Worst-Case Amplitude Errors for Windows

Window	Scaling Factor (Coherent Gain)	Noise Power Bandwidth	Worst-Case Amplitude Error (dB)
Uniform (none)	1.00	1.00	3.92
Hann	0.50	1.50	1.42
Hamming	0.54	1.36	1.75
Blackman-Harris	0.42	1.71	1.13
Exact Blackman	0.43	1.69	1.15
Blackman	0.42	1.73	1.10
Flat Top	0.22	3.77	< 0.01

$$PSD_j = \frac{s_j^2}{\Delta f \cdot ENBW} \qquad (1)$$

Analysis

Lots of packets received per day.

Unwieldy to look at / search for something.

No insight how changes affect setup

Statistics server

Separate host (no interference with recording).

- Automated process copies file after nightly rotation
- parses, runs different reassembler modes & some basic “grep” statistics
- pushed into grafana to visualize

One data point each per day

- Not very detailed, but good to see changes over time


Received frames
40 Mil
30 Mil
20 Mil
10 Mil
0
2017-01
2018-01
2019-01
2020-01
2021-01
2022-01
total
Recording time / day
27:46:40
22:13:20
16:40:00
11:06:40
05:33:20
00:00:00
2017-01
2018-01
2019-01
2020-01
2021-01
2022-01
length
% unparsed packets
40%
30%
20%
10%
0%
2017-01
2018-01
2019-01
2020-01
2021-01
2022-01
asPercent(unparsed,frames)
% uplink packets
2.0%
1.5%
1.0%
0.5%
0%
2017-01
2018-01
2019-01
2020-01
2021-01
2022-01
asPercent(uplink,parsed)

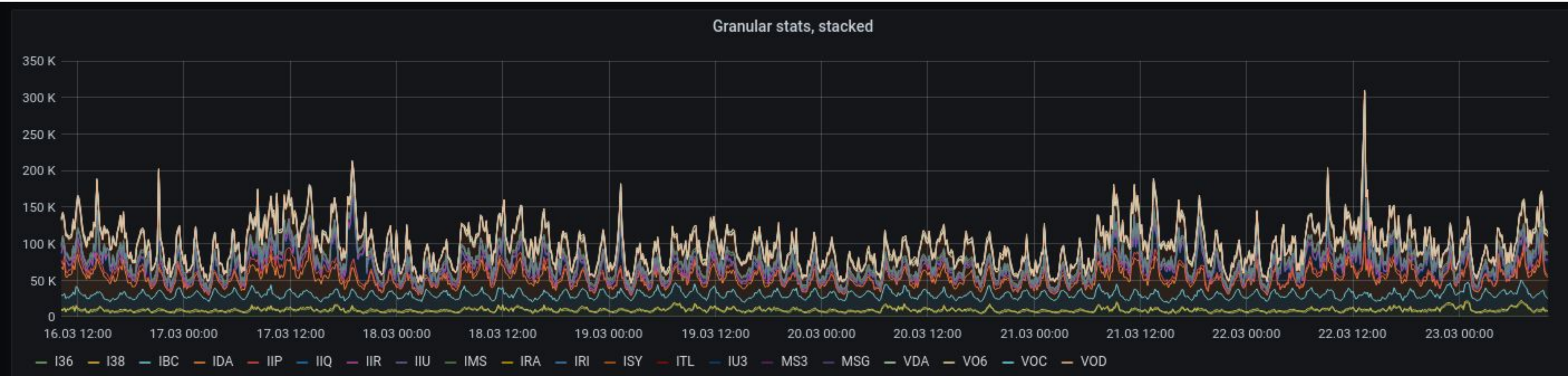
Statistics server

Issues

- about 1-4h per day
- helps find unexpected bit combinations
- due to low disk space, parsed output is not kept
- not obvious which are parser changes and recording changes

Statistics server

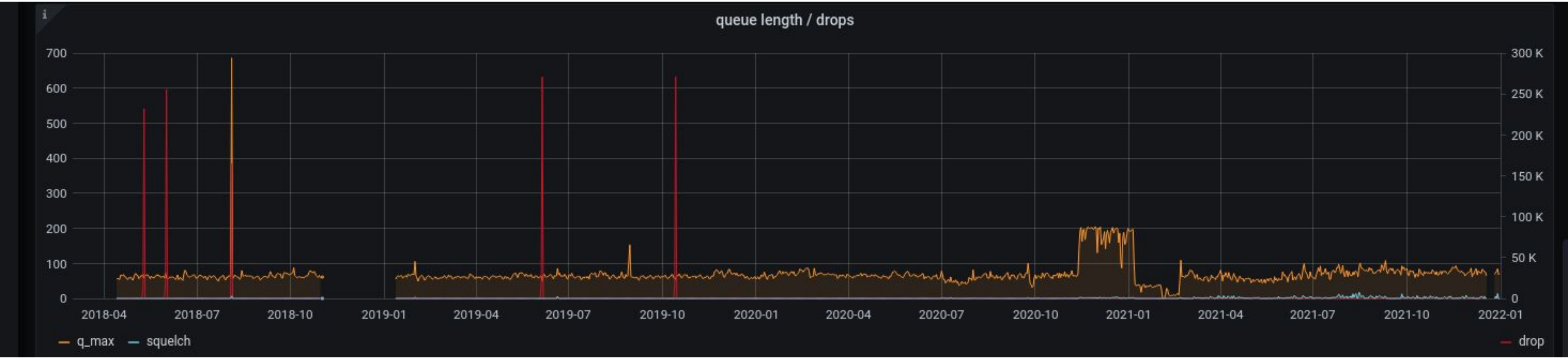
- added “live” graph (per 10 minutes)
- only count of packet types



Statistics server

- added logfiles
- overview of gr-iridium / host “health”

Also alerting (email) if no frames are coming in



Live Map

Previously generated .kml files to view in google-earth (pro)

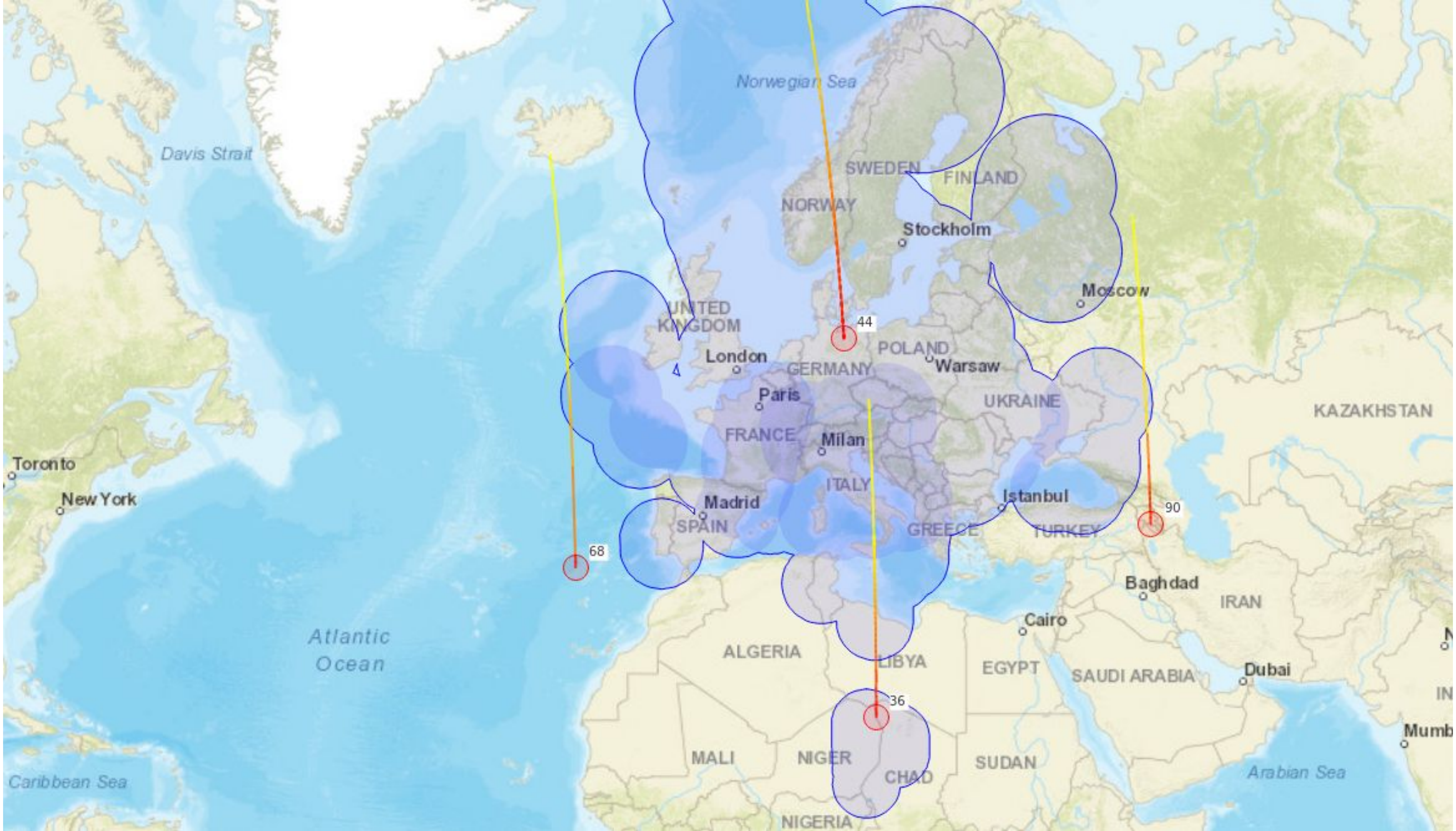
Was difficult to run, requires non-free software

“Quick” proof of concept with openlayers / jsts (javascript topology suite)

Polls sats.json every 5 min

Kind of inefficient

TBD: merge multiple receivers.



Norwegian Sea
Davis Strait
SWEDEN
FINLAND
NORWAY
Stockholm
Moscow
UNITED KINGDOM
44
POLAND
London
Warsaw
GERMANY
Paris
UKRAINE
KAZAKHSTAN
FRANCE
Milan
Toronto
ITALY
New York
Istanbul
Madrid
90
SPAIN
GREECE
TURKEY
68
Baghdad
IRAN
Cairo
Atlantic Ocean
ALGERIA
LIBYA
EGYPT
SAUDI ARABIA
Dubai
36
MALI
NIGER
SUDAN
CHAD
Caribbean Sea
Arabian Sea
NIGERIA

Live processing

Pipes are versatile, but are difficult for unix newbies.

```
pv -l -c -N bits -t -r -b | \
pypy3 ~/iridium-toolkit/iridium-parser.py --harder --uw-ec
--errorfile /dev/null | \
pv -l -c -N parsed -D 1 -t -r -a -b | …
```

Can not easily add/remove consumers

ZMQ

Support for ØMQ XPUB/XSUB:

ZMQ “Topics” (i.e. beginning of each line) corresponds perfectly with default `iridium-parser` output format.

```
iridium-parser.py --harder --uw-ec --errorfile /dev/null
--stats -o zmq
```

and

```
reassembler.py -a perfect -m live-map -i zmq: -o sats.json
--stats
```

ZMQ

ØMQ has no support for any security

Intentionally hardcoded to `127.0.0.1` for now

No solution yet for transporting bits from extractor to parser.

```
tail -F | ssh ‘cat > pipe’
```

still TBD – Maybe mqtt?

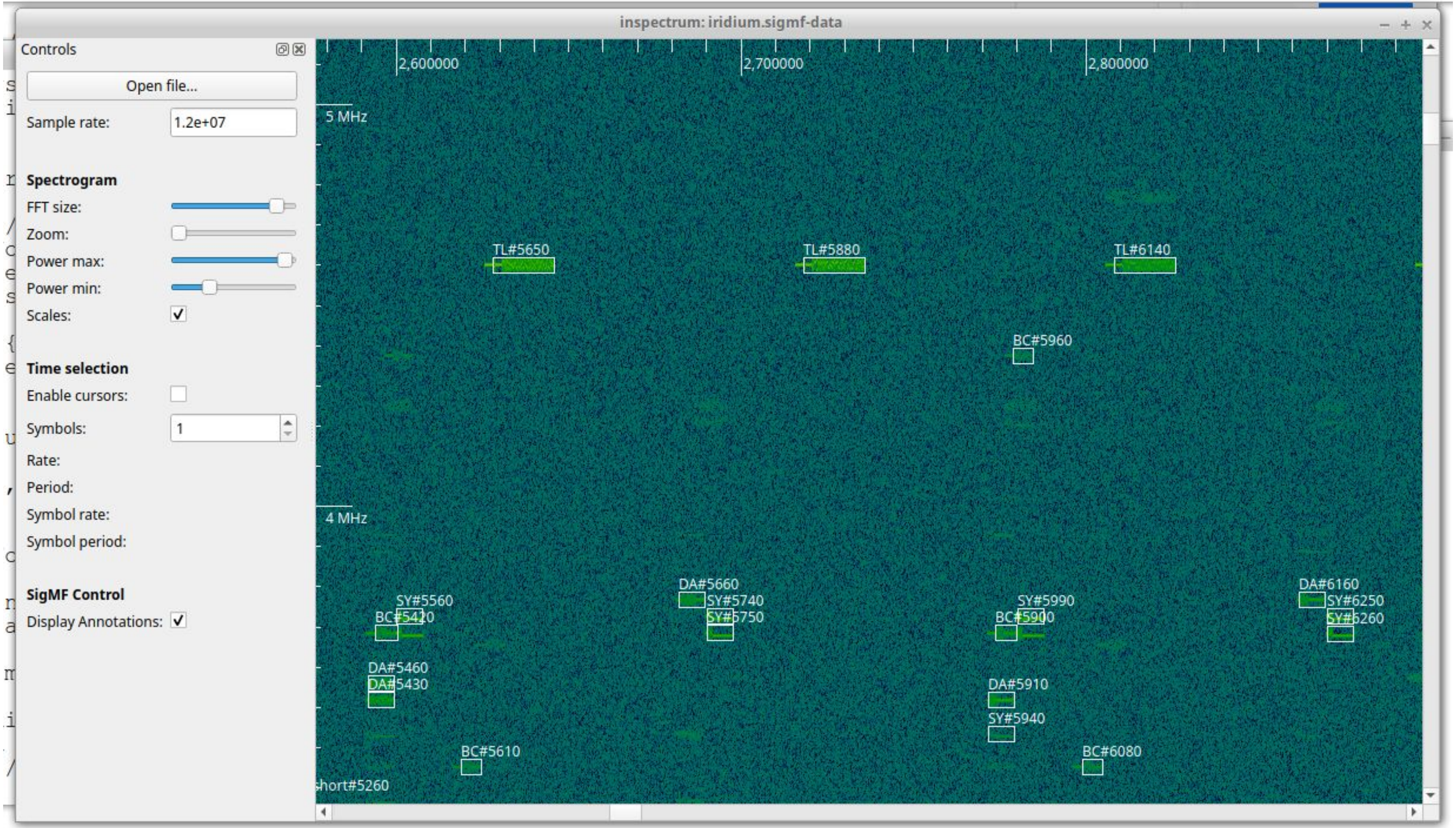
SigMF annotation support

`iridium-parser.py --sigmf iridium.sigmf-meta -o sigmf iridium.bits`

Creates annotations in existing sigmf metafile

`inspectrum` has support.

– Thanks to schneider :-)


inspectrum: iridium.sigmf-data
Controls
Open file...
Sample rate:
1.2e+07
Spectrogram
FFT size:
Zoom:
Power max:
Power min:
Scales:
Time selection
Enable cursors:
Symbols:
1
Rate:
Period:
Symbol rate:
Symbol period:
SigMF Control
Display Annotations:
2,600000
2,700000
2,800000
5 MHz
4 MHz
TL#5650
TL#5880
TL#6140
BC#5960
DA#5660
SY#5740
SY#5750
SY#5560
BC#5420
SY#5990
BC#5900
DA#6160
SY#6250
SY#6260
DA#5460
DA#5430
DA#5910
SY#5940
BC#5610
BC#6080
short#5260

Soapy source

People were having trouble with the 3.8/3.9 incompatibilities with gr-osmosdr packages.

Soapy is there by default in GR 3.9+

- no timestamp support
- no sdr autodetection
- gain names differ from gr-osmosdr
- doesn’t print warnings when a limesdr loses samples
- missing some lime specific stream args to improve performance

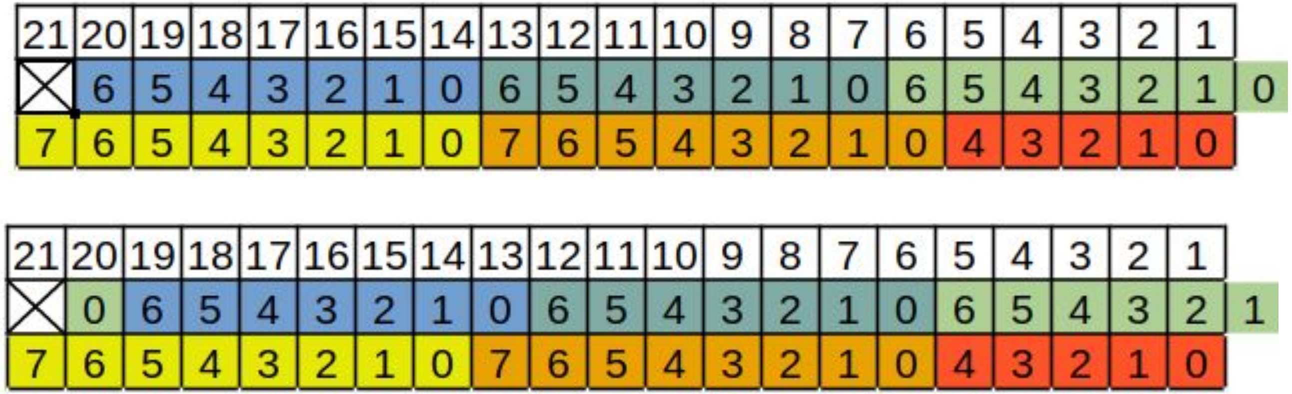
Pager Messages

New (old) checksum revisited.

reversed & added to parser

10-bit sum of message blocks

took a while due to strange/unexpected ordering



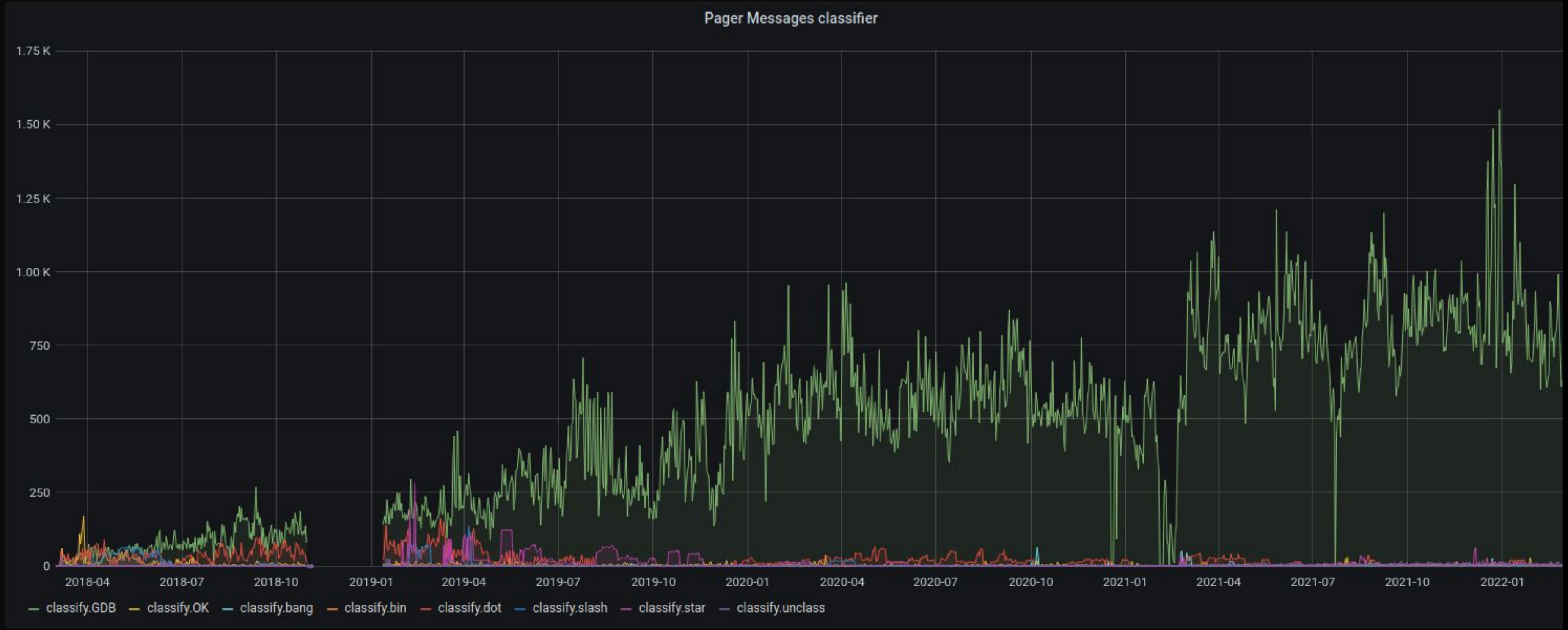
Iridium Data Burst?

We called them “encrypted” in the past

```
3526197 … OK: VgBnUyZ1ApRCrTGE1xdgAeBSI7wCCF9Df76yhoagtNcV1bxTvFl5nyizmmhr
3526206 … OK: VgBnYCZhXIgxVctXT5BaTwSEGExGxDIZR4hxEGH4sMHTbEKzLce9vmAZB7yt
3526188 … OK: VgBnBiYQtI+RMsXDJkM9*IgBMd+iMdLQD7jGVDKmjy8GGYO78W2My5xAPvkC
```

Turned out to be “basically” Base64 (* -> /, and missing end padding)

8 different RICs, up to 4k bytes content.



Pager Messages classifier
1.75 K
1.50 K
1.25 K
1.00 K
750
500
250
0
2018-04
2018-07
2018-10
2019-01
2019-04
2019-07
2019-10
2020-01
2020-04
2020-07
2020-10
2021-01
2021-04
2021-07
2021-10
2022-01
classify.GDB
classify.OK
classify.bang
classify.bin
classify.dot
classify.slash
classify.star
classify.unclass

SBD reassembly

Part of LAPDm messages, type 0x06 and 0x76(?)
(0x06 would be Radio Resource Management)

Second-layer reassmbly (first LAPDm, then SBD)

./reassembler.py -m sbd < muccc-2022-03-11.parsed

Supports streaming

Mostly (unknown) semi-binary protocols.

ACARS (over SBD)

7bit ASCII, odd parity. Kind of hard to find official spec.

Works quite ok.

```
Dir:UL Mode:2 REG:TC-LYC  NAK   Label:Q0 (Link Test) bID:7 SEQ:
S25A, FNO: TK04RT
Dir:DL Mode:2 REG:TC-LYC  ACK:7 Label:_? (Demand mode) bID:Y
```

Call Handover Tracking

voc-cluster.py tracks call handovers:

```
LCW(2,T:hndof,C:handoff_resp[cand:P,denied:0,ref:1,slot:2
,sband_up:7,sband_dn:7,access:4],01)
```

```
if "handoff_resp" in sl[8]:
    fields = sl[8].split(',')
    sband_dn = int(fields[7].split(':')[1])
    access = int(fields[8].split(':')[1].split(']')[0])
    #print sband_dn, access
    frame.f_alt = (1616000000 + 333333 * (sband_dn - 1) + 41666 * (access - 0.5) + 52000)
```

Localization

Did some experiments to localize a receiver using Iridium IRA and IBC frames:

- IRA: Gives us SV position
- IBC: Gives us the Iridium time at start of of 90 ms frame

Challenge: Must interpolate the SV position as it is only +- 4 km in X/Y/Z

Simple numpy based minimization lead to ~1000m accuracy if averaged over time.

Localization

```
python3 iridium-parser.py -p --filter=IridiumRAMessage,'q.ra_alt>7100'
--format=globalns,ra_sat,ra_cell,ra_alt,ra_pos_x,ra_pos_y,ra_pos_z iridium.bits > iridium.ira

python3 iridium-parser.py -p --filter=IridiumBCMessage+iri_time_ux
--format=globalns,iri_time_ux,slot,sv_id,beam_id iridium.bits > iridium.ibc
```

python3 **ibc_position_interpolator.py** iridium.ibc iridium.ira > iridium.ibc_pos_interp

python3 **locator.py** iridium.ibc_pos_interp

```
Error: 798 ( -250 ) [4177181.76787259  854743.75425443 4728026.74910621]
Error: 798 ( -250 ) [4177181.76787259  854743.75425443 4728026.74910621]
Error: 880 ( -454 ) [4177057.61037911  854689.9149959  4727871.68599473]
Error: 1121 ( -1005 ) [4176470.5561329   854603.08676267 4727664.05743592]
Error: 1003 ( -867 ) [4176554.37046054  854639.4218041  4727768.68231563]
Error: 1377 ( -1290 ) [4176296.71157331  854527.71533911 4727447.04489401]
Error: 1761 ( -1698 ) [4176048.49026336  854420.08190372 4727137.17863116]
Error: 854 ( -681 ) [4176667.84455307  854688.61158214 4727910.3295299 ]
Error: 1018 ( -885 ) [4176543.72162508  854634.80545156 4727755.38950141]
Error: 1027 ( -899 ) [4176528.70321852  854632.59237939 4727750.10430602]
Error: 846 ( -674 ) [4176665.60743175  854691.91002168 4727920.90775398]
Error: 562 ( -96 ) [4177017.95643079  854844.54980148 4728360.48947208]
Error: 676 ( -422 ) [4176819.34382826  854758.51409141 4728112.70777421]
Error: 841 ( -668 ) [4176669.64708987  854693.66023717 4727925.94763803]
Error: 1004 ( -871 ) [4176545.53087894  854639.88373443 4727771.09886043]
Error: 834 ( -207 ) [4177255.4970728   854744.13968235 4728019.87929082]
Error: 1019 ( -629 ) [4176998.0393332   854632.65706191 4727698.84607364]
good 1047 bad 10 known_bad 75
average cartesian error: 964.4526122742765 ( -170.21[illegible]70892 )
average cartesian position: [4177193.360[illegible]  854772.82723399 4728119.2772457 ]
average cartesian position error: 742.6931948335987
average cartesian position height error: -170.2139186663553
average cartesian position to lla 48.147622500510515 11.564697999994626 369.5867578834295
```

Make sure to have recent astropy and pymap3d installations.

Iridium Time & Location

Very long, distinct header (UW + “11” + “00” * 94)

Changed format @ 2017-10

Significantly stronger …

Until recently :-(

Iridium Time & Location

Decode I & Q separately.

`I`: 16byte “constant” + 32byte “plane” (6 values)

`Q`: 4*12byte “PRS” (Pseudo-random-sequence)

=> Total of 512 different PRS. 4 sets of 128.

Each single message uses all sets: ABCD, BADC, CDAB or DCBA.

Map each PRS set to 0-127, but in which order?

Iridium Time & Location

If we look at messages where the first byte is constant, we can find in some of these a field which counts up synchronous with the LBFC.

But with different moduli [123,125,127,128,31].

This helps identify the 0. So we can “easily” create a complete mapping.

\o/

Iridium Time & Location

Why the same info (LBFC) with different moduli?

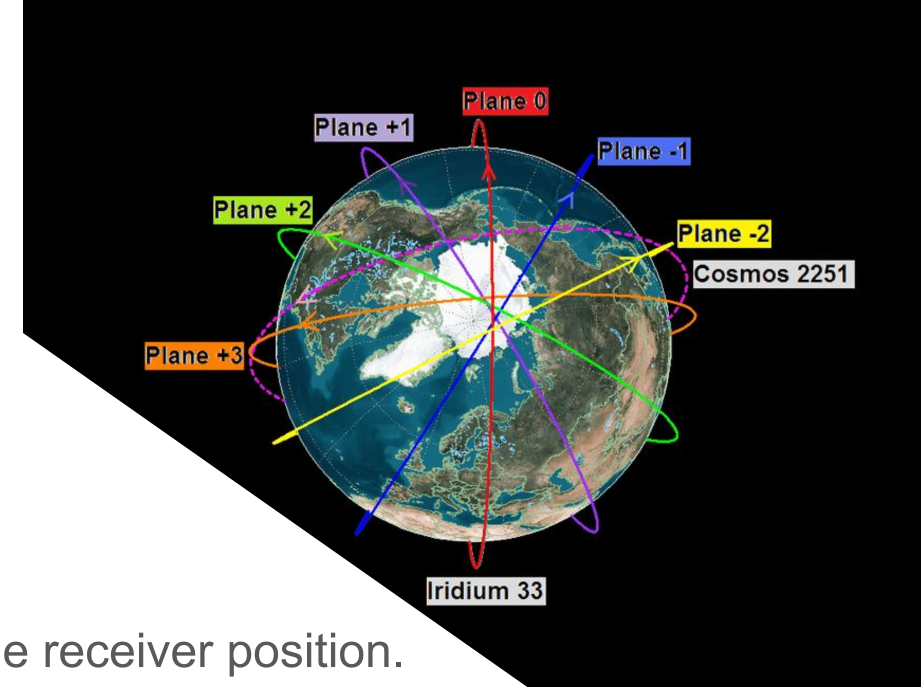
Chinese Remainder Theorem

LBFC is 32 bit

```
>>> log(128*127*125*123*31,2)
32.85117978716037
```

We can reconstruct the LBFC, i.e. time down to 90ms

Iridium Time & Location



Location is (still) unknown.

Each ITL identifies current plane(1-6) and index within plane(1-11).

Time together with sat position would enable receiver position.

So far could not find such correlation in the remaining unknown bits

Would already work if online - i.e. current TLEs are available

(& mapping of plane/index to NORAD name)