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

- Unknown LNA
- Standard SAW filter (3 dB insertion loss)
- Unknown LNA
RTL-SDR blog antenna has bad out of band rejection

[Image of spectrum analyzer with frequencies 810.x MHz and 1621.x MHz highlighted]

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

[Image of spectrum analyzer with frequency 810.x MHz highlighted]
Own design based on tnt’s Smart L-Band Antenna
Tallysman Antennas

HC610 Active Iridium Helical Antenna

- Coverage: Iridium
- Mount: Surface Mount / Direct Screw
- Amplifier Gain: 26 dB typ.
- Connector Options: SMA (male)

TW2643A Single Band GNSS Antenna with Receive-Only Iridium

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

\[ x + i \times y \]
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”:

\[ x + \text{cmath.rect}(1, \text{angle}) \times y \]

- Max signal strength should be about twice of one signal
  - 3dB gain over linear
- Min signal strength should be \( \sim 0 \) (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
Osmocom Source

10 Msp

FFT Burst Tagger

Polyphase Channelizer

Over Sample Ratio = \( \frac{(N+1)}{N} \)

\( (N+1) \times 10 \text{ Msp} / N \)

\( N=4: 5 \times 2.5 \text{ Msp} = 12.5 \text{ Msp} \)

Burst To PDU

Queue

Burst Downmix

QPSK Demod

Dedup & Sort

Print
Over Sample Ratio = \frac{(N+1)}{N}

\[(N+1) \times 10 \text{ Msp} / N\]

\[N = 4: 5 \times 2.5 \text{ Msp} = 12.5 \text{ Msp}\]
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.

gr-iridium current state

Extra Burst Downmix if not ARM or more than 4 cores.
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.

```bash
-v 6  examples/zeromq-sub.conf

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>@@</td>
<td>0,0</td>
<td>+1,6@@</td>
</tr>
</tbody>
</table>

1  +[zeromq-sub-source]
2  +
3  +sample_rate=10000000
4  +center_freq=1622000000
5  +address=tcp://127.0.0.1:5000
6  +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
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

rad1o: -35 ppm

bladeRF: 0.1 ppm

USRP B200: -5 ppm (overflowing)
GPSDO

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:
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.
- 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 000059.9996 1622600000 N:32.12-80.05 I:0600000000 100% 0.13551 179 0110000011011011101100000006

- 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 000059.9996 1622600000 N:32.12-80.05 I:0600000000 100% 0.13577 179 0110000011011011101100000006

- 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 000059.9996 1622600000 N:32.12-80.05 I:0600000000 100% 0.13643 179 0110000011011011101100000006

- name: Test raw samples
  run:
  - ln -s prbs15-2M-20dB.sigmf-data test-data/prbs15-2M-20dB.fc32
    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


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

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

\[
PSD_j = \frac{s_j^2}{\Delta f \cdot ENBW} \quad (1)
\]
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
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.
Live processing

Pipes are versatile, but are difficult for unix newbies.

```
pv -l -c -N bits -t -r -b | \npy3 ~/iridium-toolkit/iridium-parser.py --harder --uw-ec
--errorfile /dev/null | \n```

```
pv -l -c -N parsed -D 1 -t -r -a -b | ...
```

Can not easily add/remove consumers
ZMQ

Support for ØMQ X PUB/X SUB:

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 :-}
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: VgBnUyZ1ApRCrTGE1xdgAeBSI7wCCF9Df76yhoagtNcV1bxTvF15nyizmmhr
3526206 ... OK: VgBnYCZhXIgxVctXT5BaTwSEGExGxDIZR4hxEGH4sMHTbEKzLce9vmAZB7yt
3526188 ... OK: VgBnBiYQtI+RMXLDJkM9*IgBMd+iMdLQD7jGVDKmjy8GGYO78W2My5xAPvkC

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

8 different RICs, up to 4k bytes content.
SBD reassembly

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

Second-layer reassembly (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:

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

```python
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

```python
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
```

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 :-(

[Signal level dBFS graph]
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

```python
>>> 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)