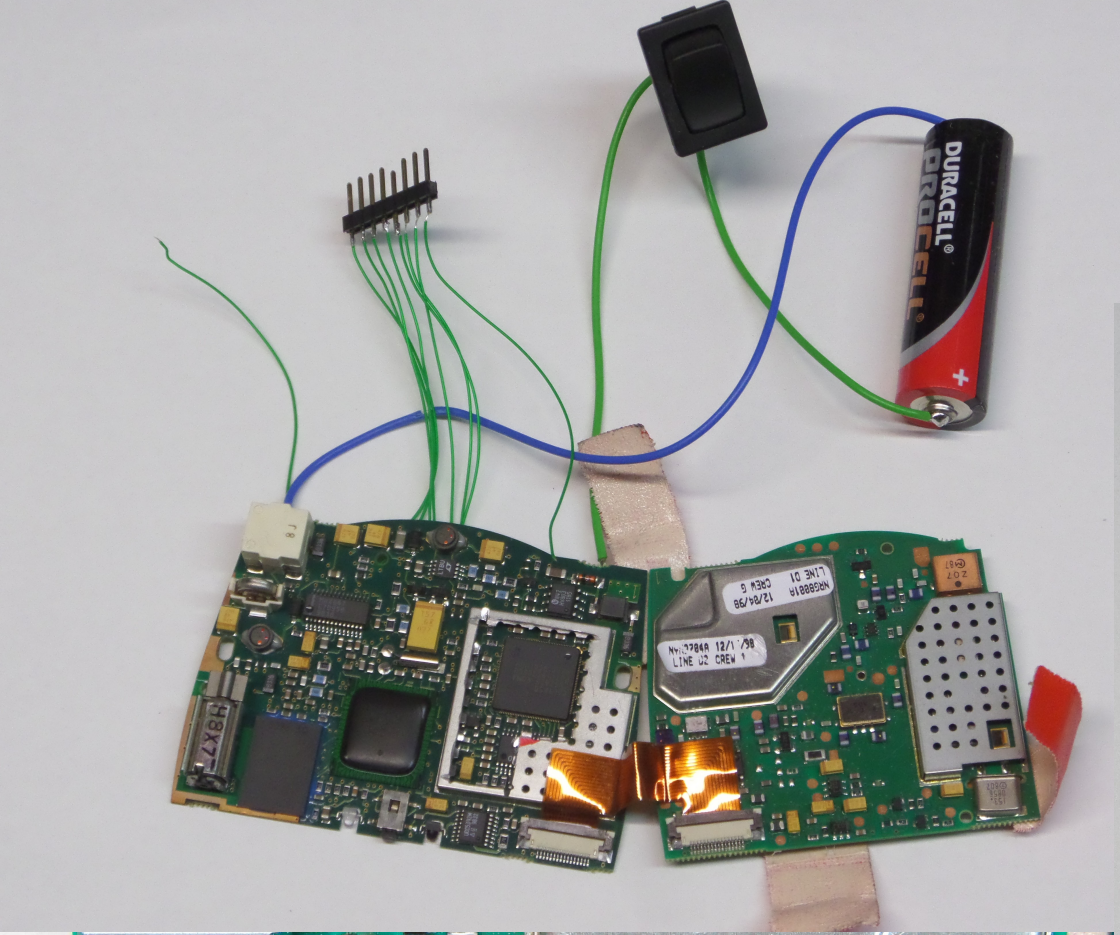
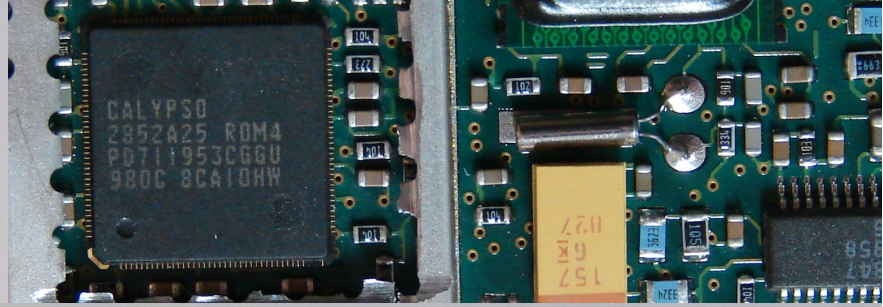
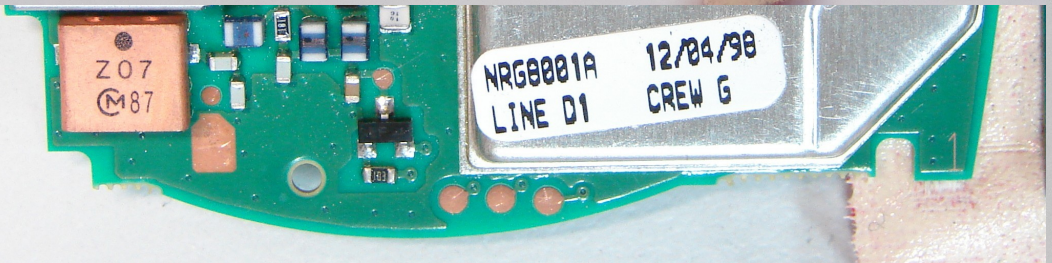
Iridium Pager Hacking

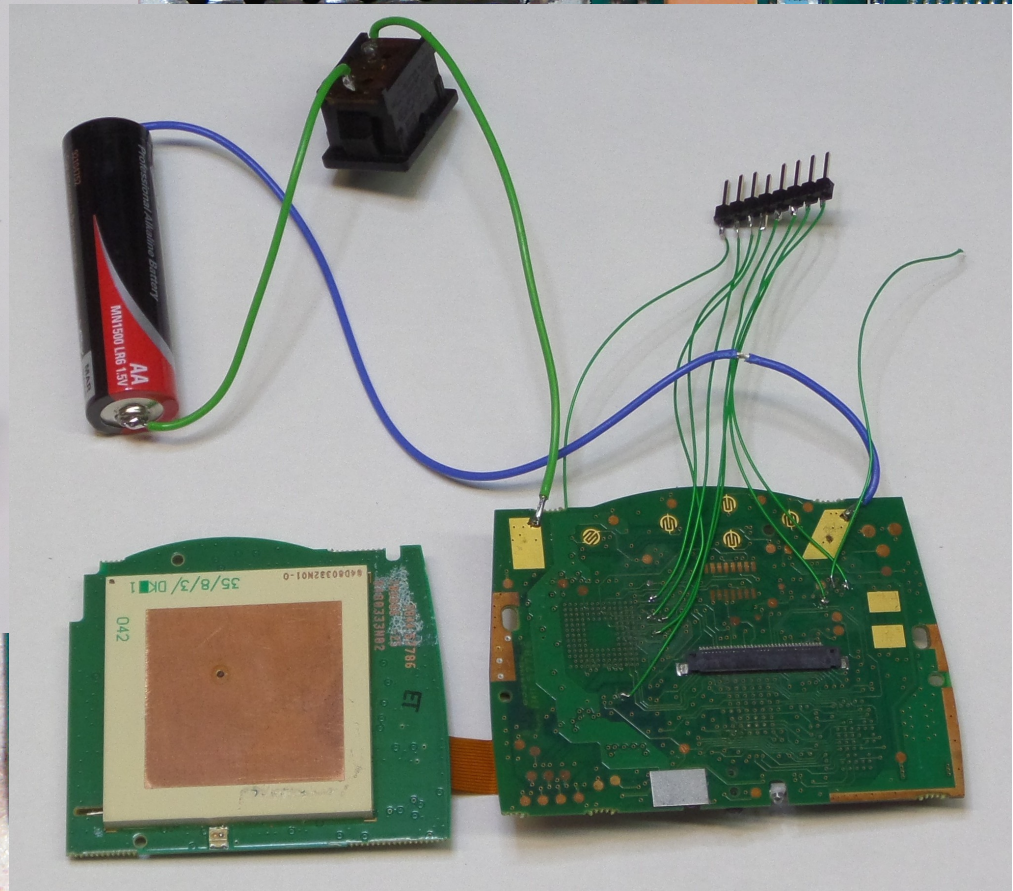
Sec schneider

Image by nibbler


DURACELL
LINE D1 CREW 6


CALYPSO
2852A25 ROM4
PD711953CGGU
980C 8CA1OHW


Z07
NRG8001A 12/04/98
LINE D1 CREW G


AA
MN1500 LR6 1.5V

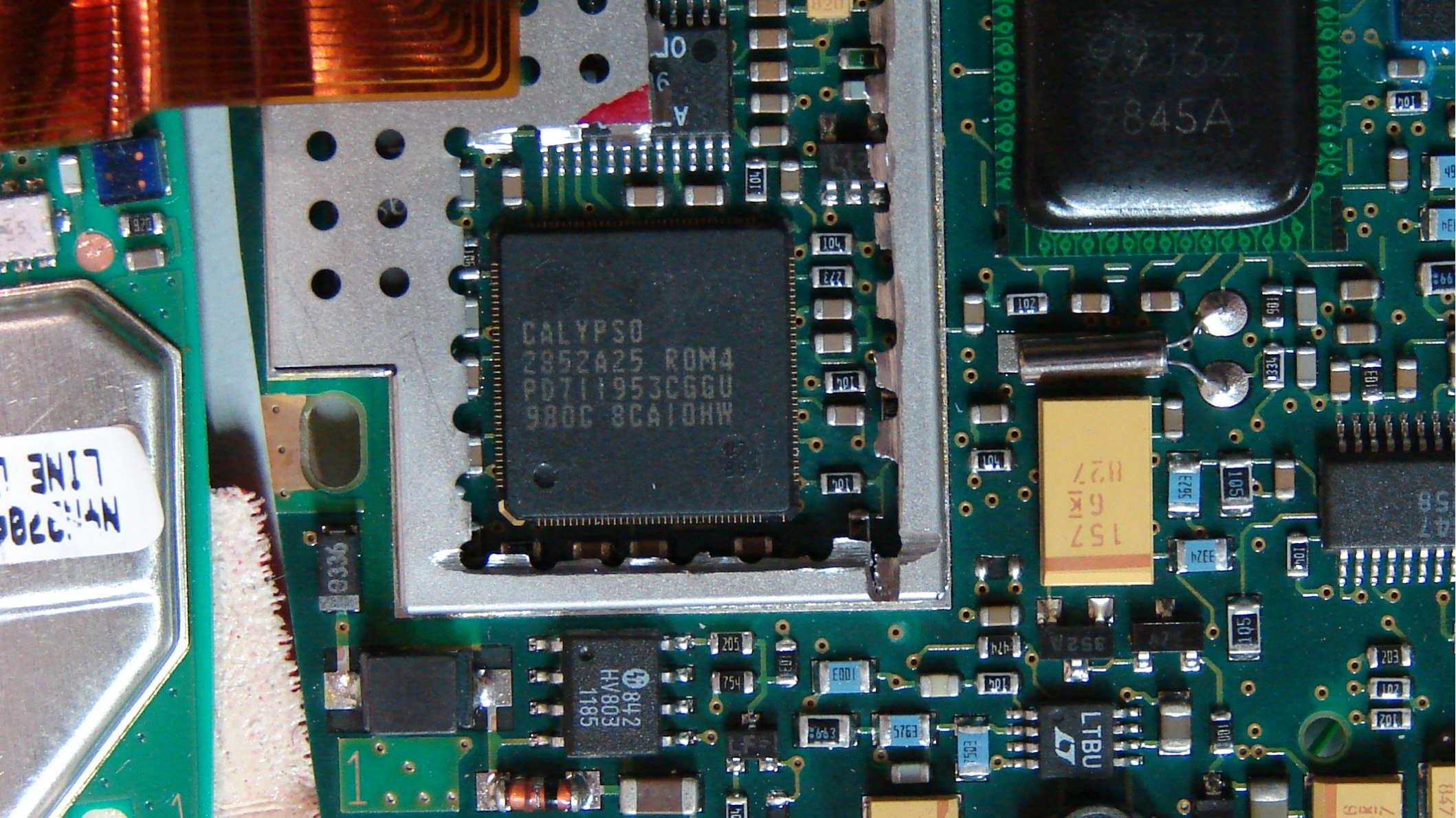
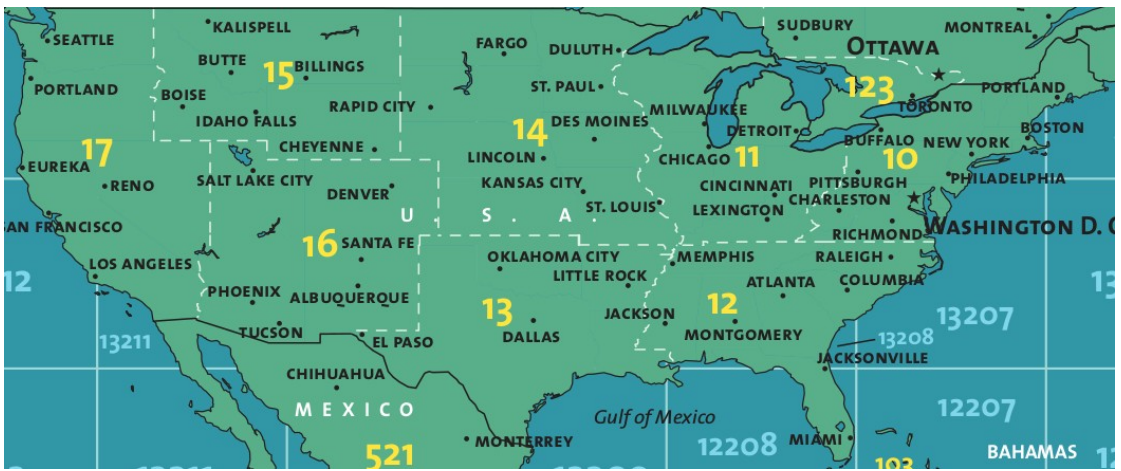

CALYPSO
2852A25 ROM4
PD7Ii953CGGU
980C 8CAIOHW
LTBU

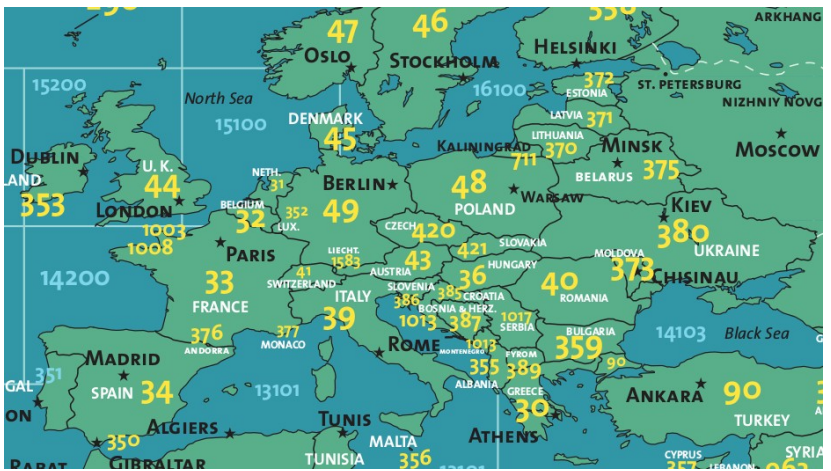
Image by Eric Long
National Air and Space Museum, Smithsonian Institution

Why look at Iridium pagers?

Cell phones **are** tracked. Who wants to be tracked?

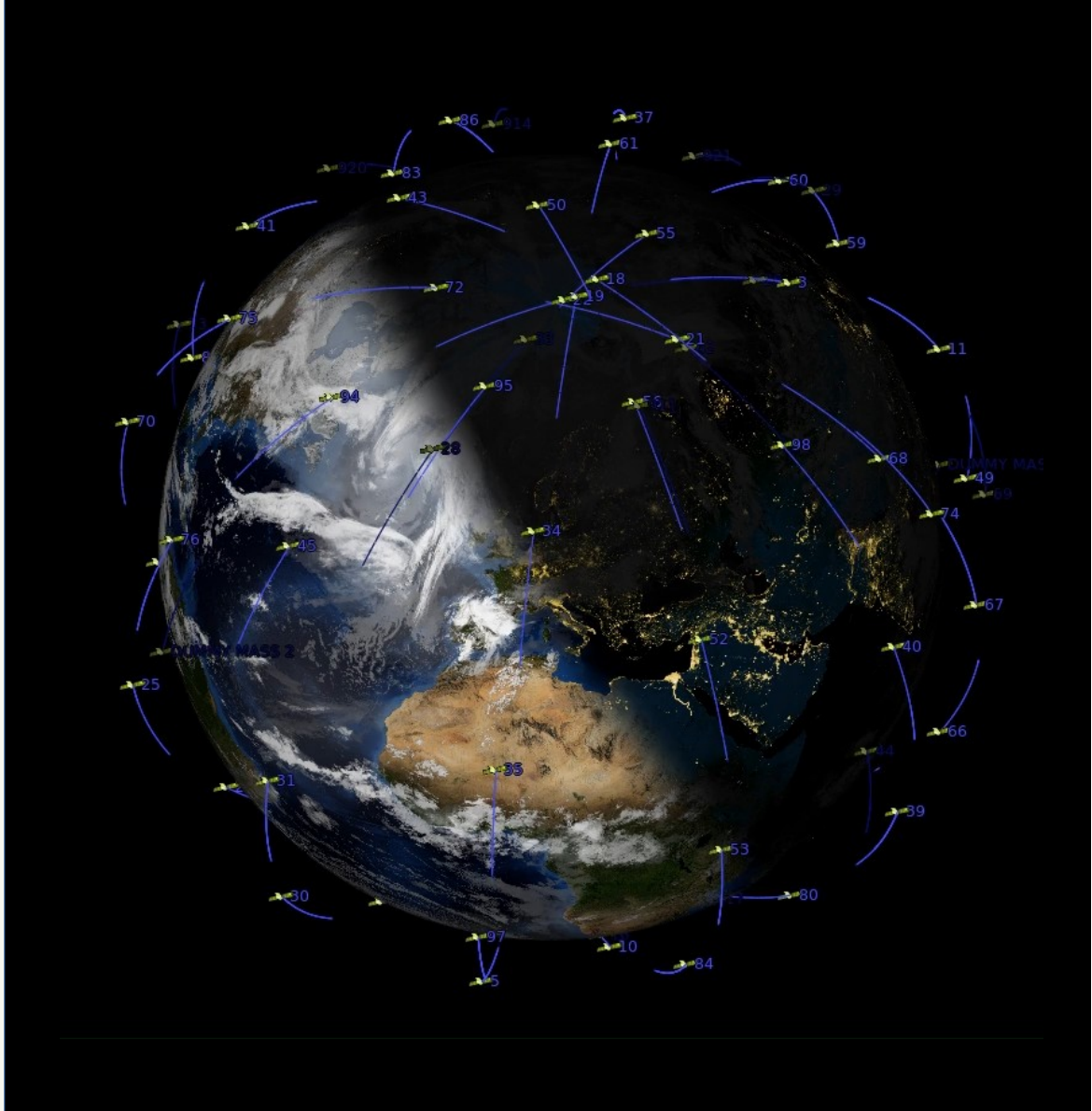
- Iridium pagers are completely passive
- Iridium only needs to know roughly where a pager is





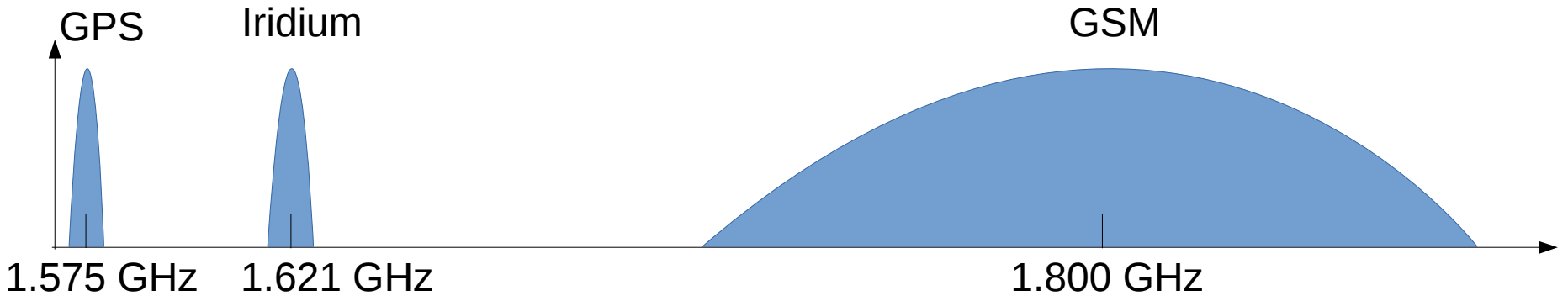
Iridium Basics

- 66 active low earth orbit satellites with inter-satellite links
- One satellite is visible for roughly 8 minutes
- Highly inclined orbits go over the poles

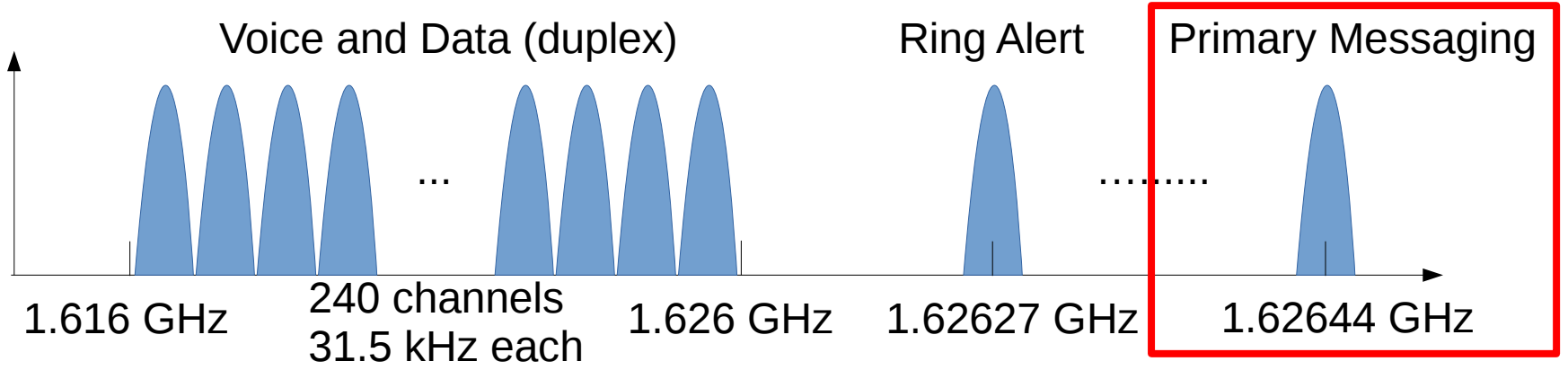


Iridium Basics

- Frequency range: 1.616 GHz to 1.6265 GHz

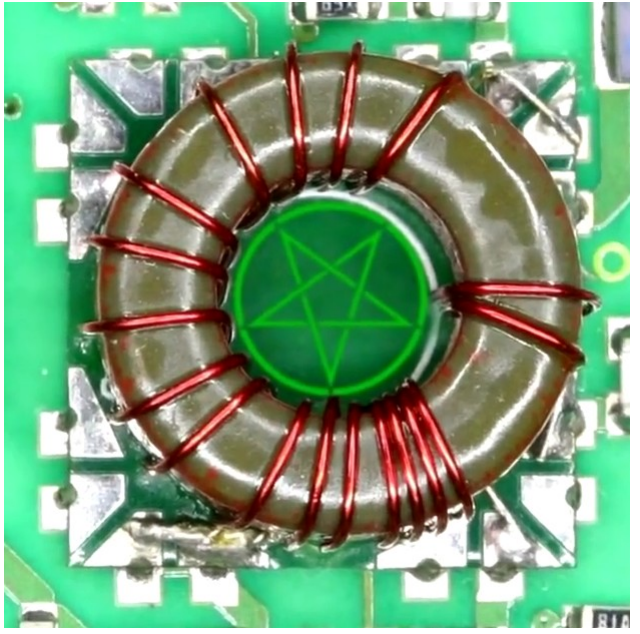


- Multiple small channels across the band



SDR and RF myths

- RF is black magic
- Signal processing is hard math
- Receiving satellites is too much effort
  - You need a large antenna
  - The antenna needs to track the satellite
- Everything is encrypted anyways



From “Anritsu spectrum analyser teardown” by mikeselectricstuff

Why even bother?

SDR facts

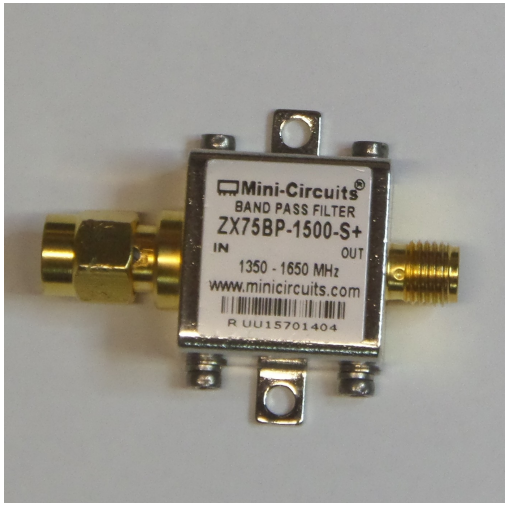
- You can do it with almost no knowledge about HF stuff
  - Use a HackRF or USRP to generate signals and test your setup
- SDR does away with lots of old restrictions:
  - Capture once, decode often
  - Oversample and look for the right spot later
  - Simulate signals to get to know your tools
- There are lots of legacy systems which are completely unencrypted or use weak encryption

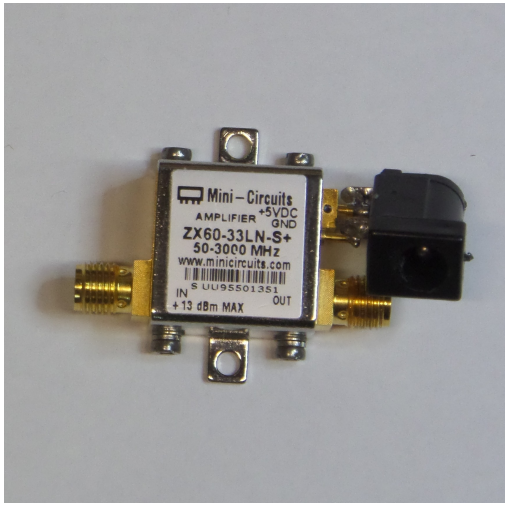
Satellites: How hard can it be?

It can't be that hard?

- Local hackerspace had an USRP with a GSM daughter board lying around
- Log periodic is too directional
  - Satellites move too fast
  - No reproducible setup
- Iridium uses circular polarization

Let's buy some blocks of gold

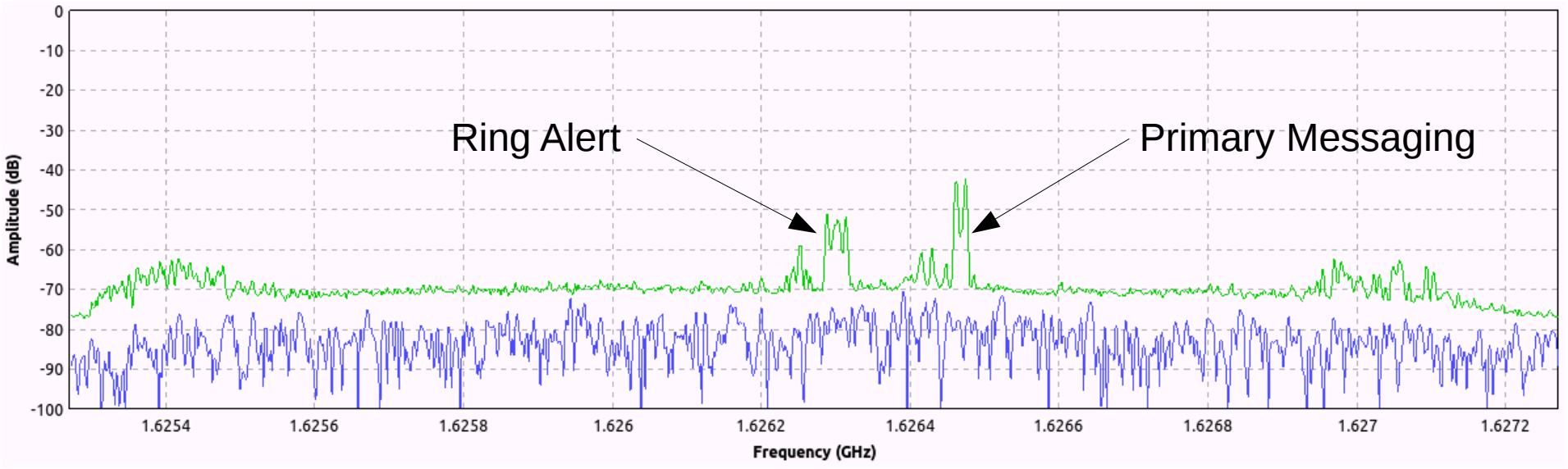




- We started throwing money at the problem
  - Bought a proper antenna, a bandpass and a low noise amplifier.

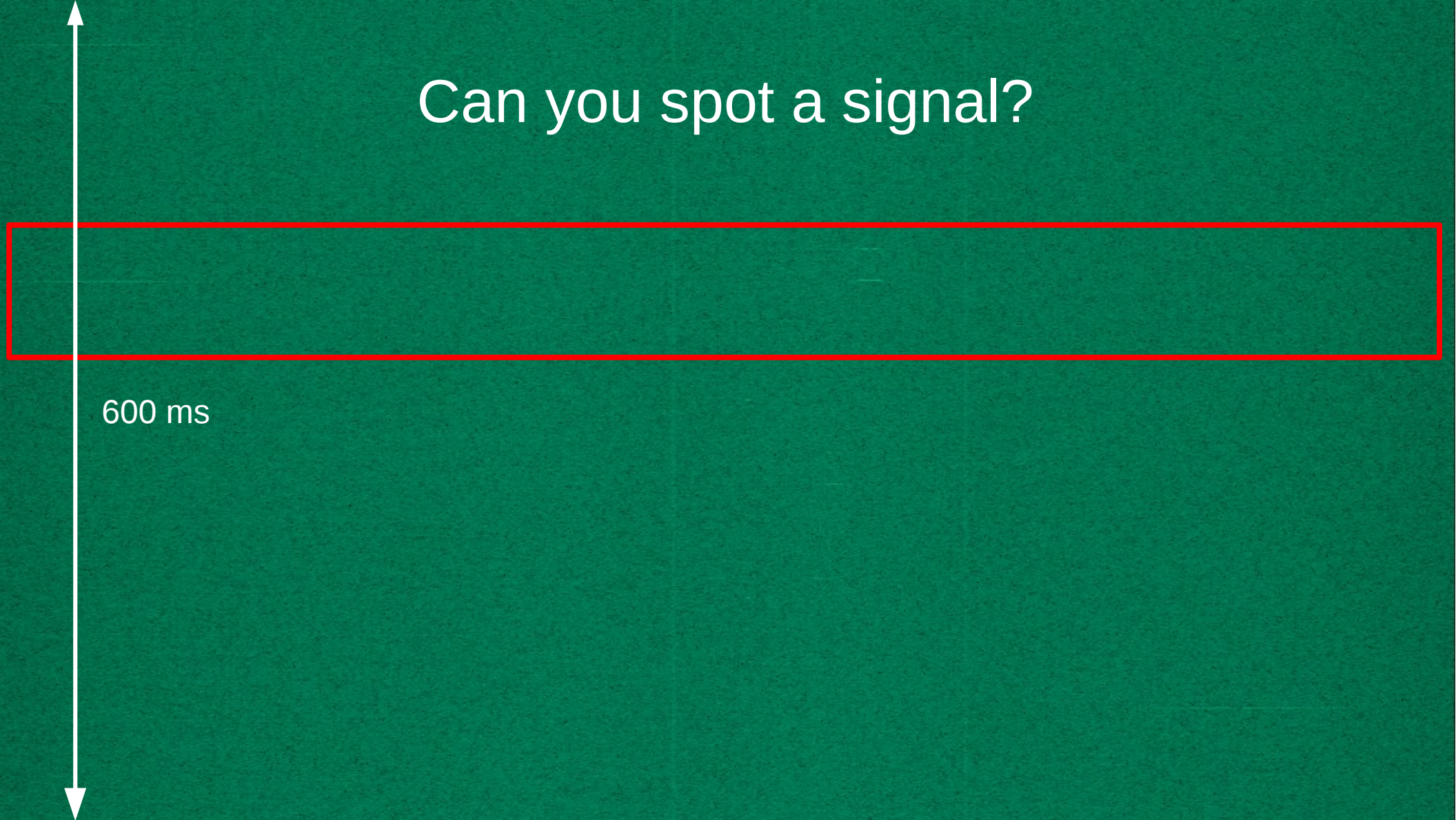

To the roof!

Found something!



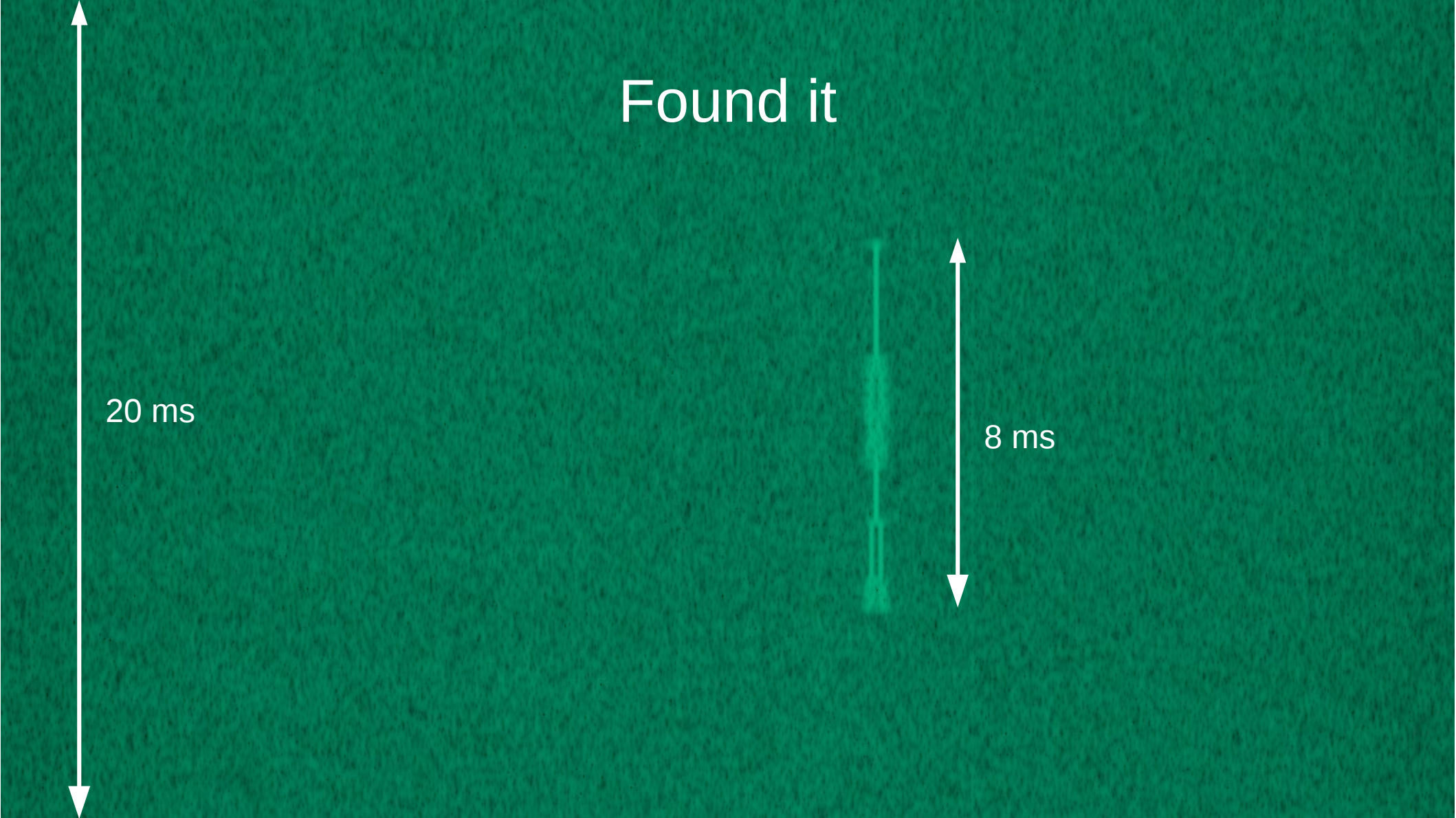
- The GnuRadio FFT sink started to show large peaks at the right places

Can you spot a signal?

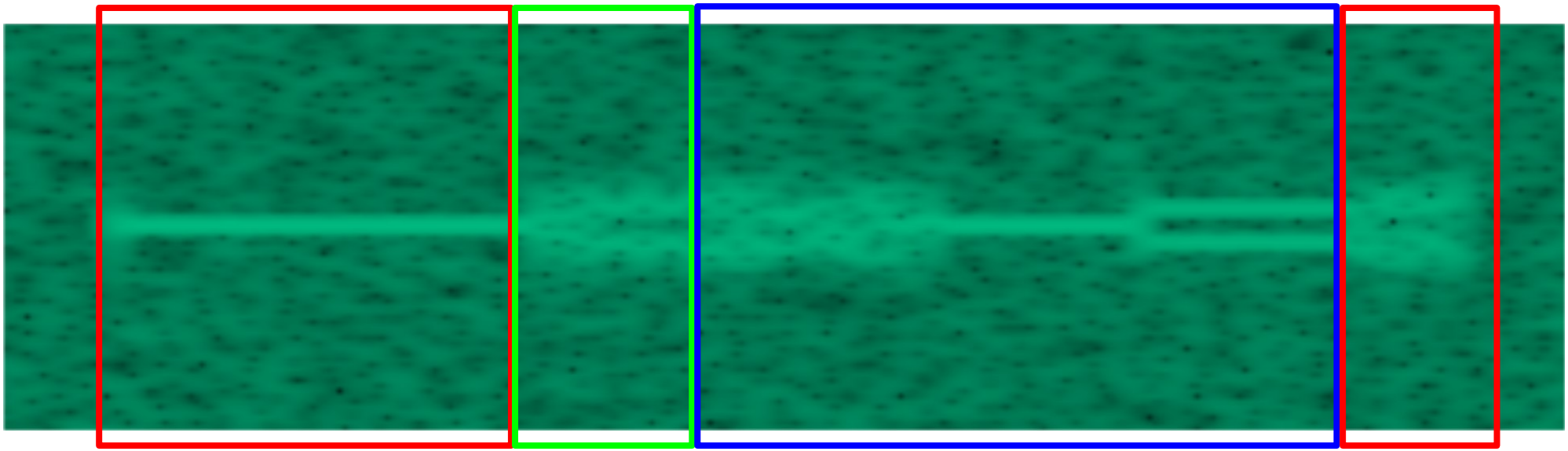


Maybe now?

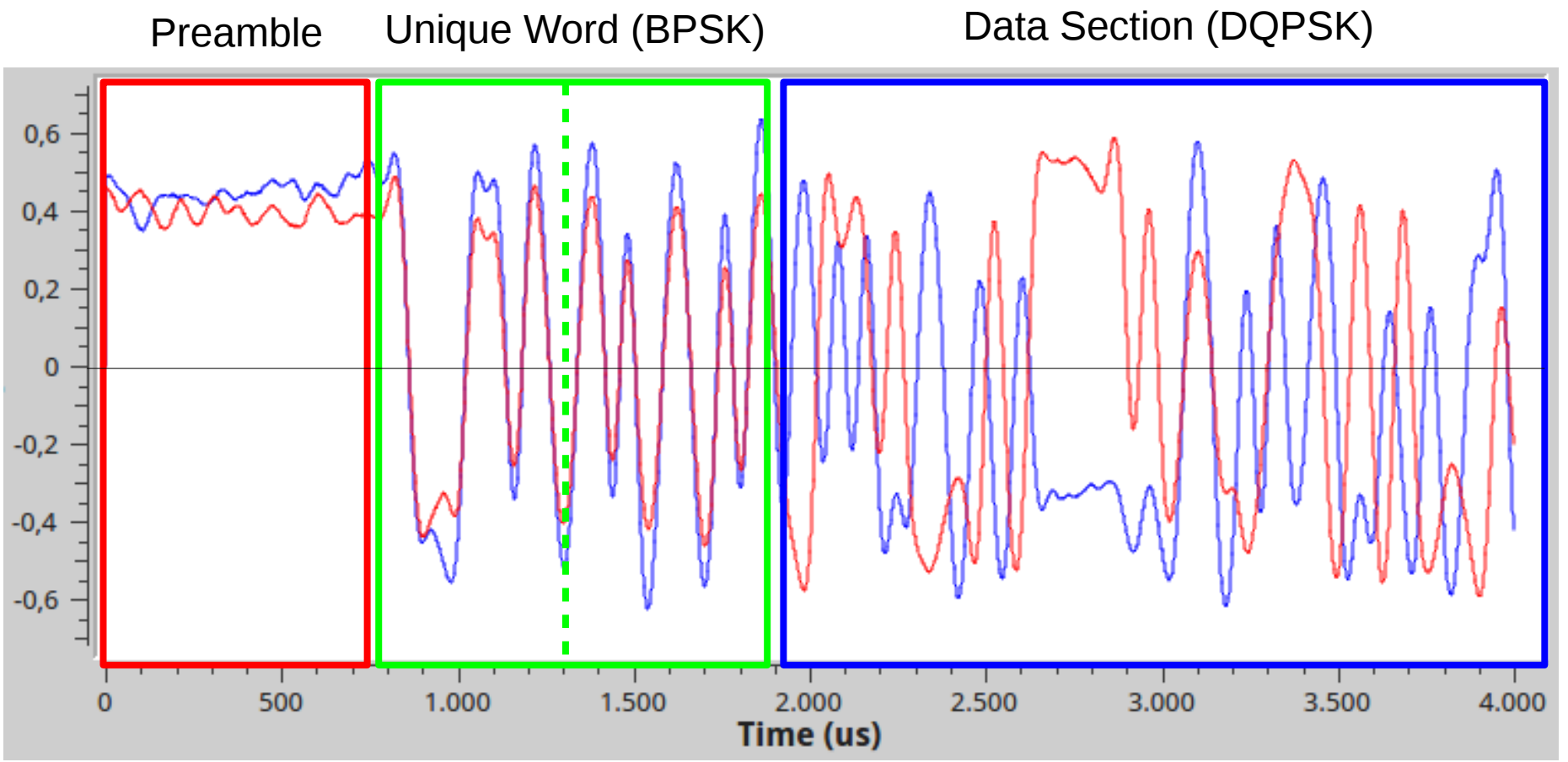
Found it



A packet dissected



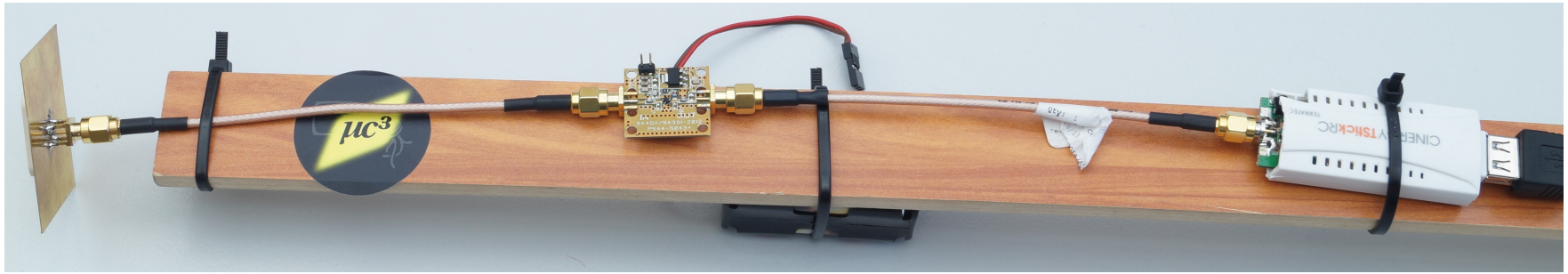
A packet dissected



How much do you have to invest?

- USRP B200 (+ LNA4ALL) + commercial Iridium antenna
  - Around 700 (+ 20) + 50 = 750 – 770 Euro

- E4000 DVB-T dongle + LNA4ALL + DIY Iridium antenna
  - About 20 + 20 + 10 = 50 Euro



L-Band Channel Security

- To monitor an L-band channel,
  - Located within the transmit range of the ISU being monitored (10 to 30 km)
  - ISU downlink L-Band transmissions could be received over a much wider area but within the coverage area of a common beam
- The complexity of the Iridium air interface makes the challenge of developing an Iridium L-Band monitoring device very difficult and probably beyond the reach of all but the most determined adversaries.
- Among the complications are
  - Large, continually changing Doppler shifts
  - Frequent inter-beam and inter-SV handoffs
  - Time-division multiplexed burst mode channels
  - Complicated modulation, interleaving and coding

L-Band Channel Security

- To monitor an L-band channel,
  - Located within the transmit range of the ISU being monitored (10 to 30 km)
  - ISU downlink L-Band transmissions could be received over a much wider area but within the coverage area of a common beam
- The complexity of the Iridium air interface makes the challenge of developing an Iridium L-Band monitoring device very difficult and probably beyond the reach of all but the most determined adversaries.
- Among the complications are
  - Large, continually changing Doppler shifts
  - Frequent inter-beam and inter-SV handoffs
  - Time-divisi
  - Complicate

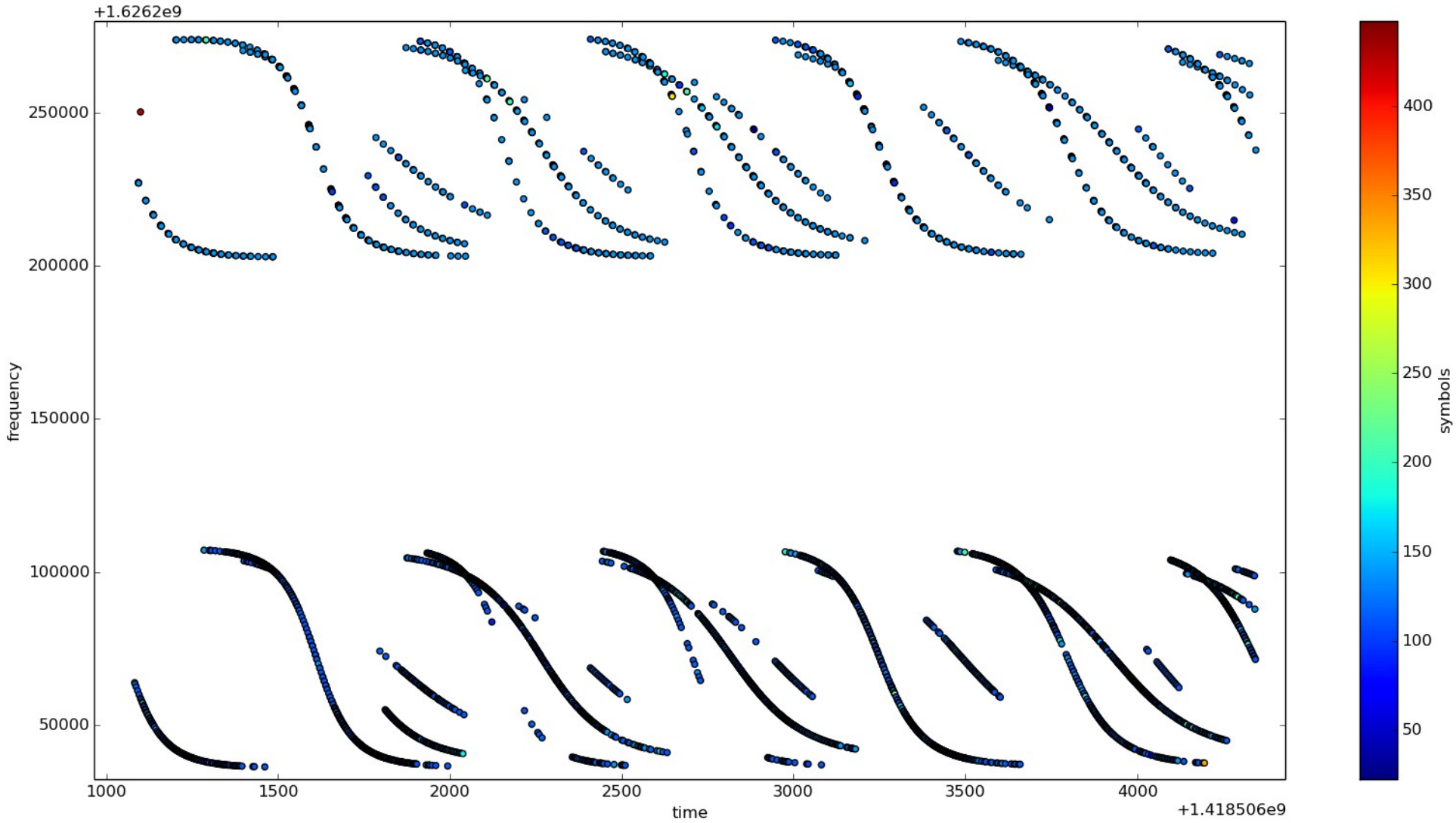
iridium confidential

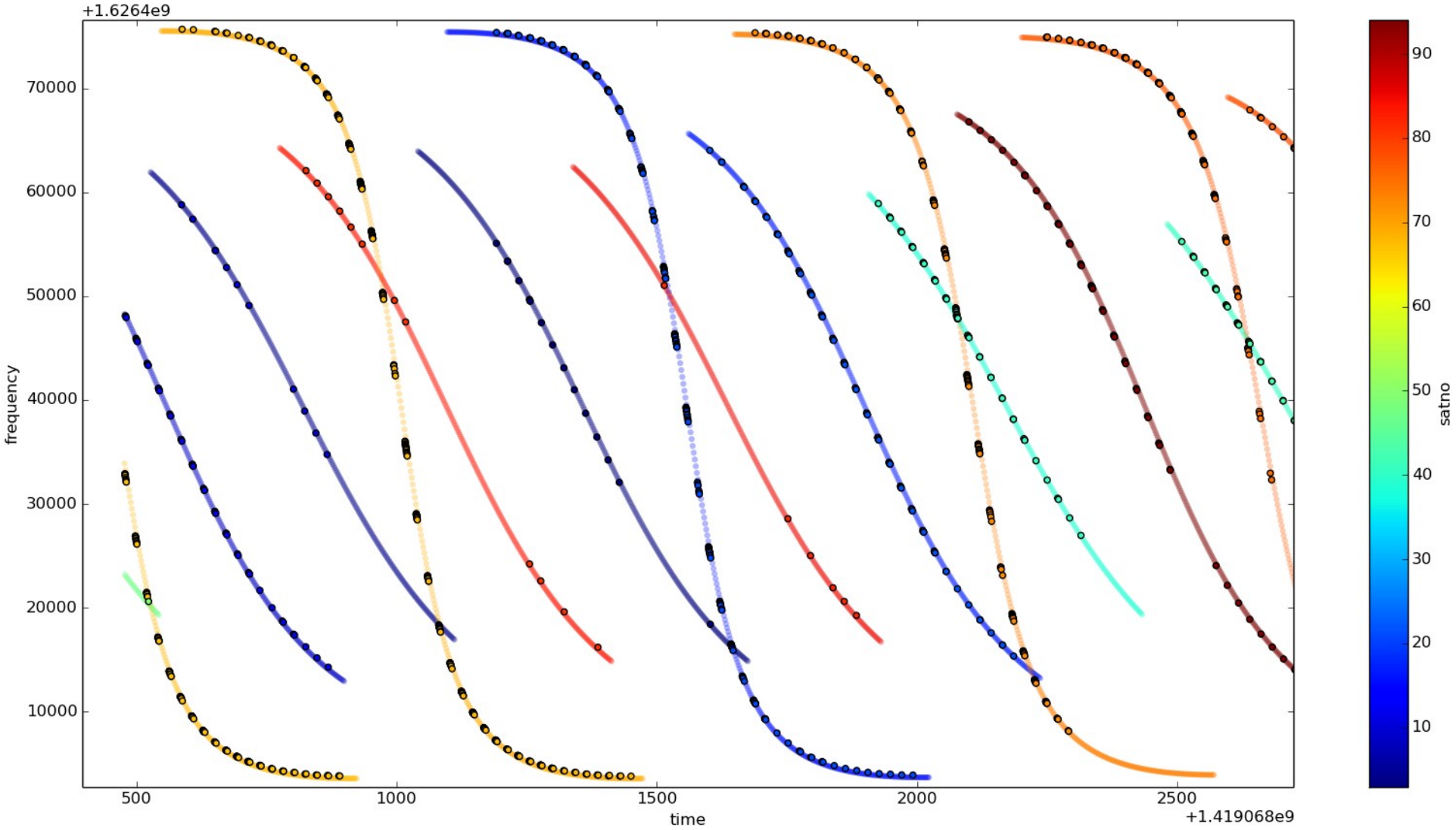
L-Band Channel Security

- To monitor an L-band channel,
  - Located within the transmit range of the ISU being monitored (10 to 30 km)
  - ISU downlink L-Band transmissions could be received over a

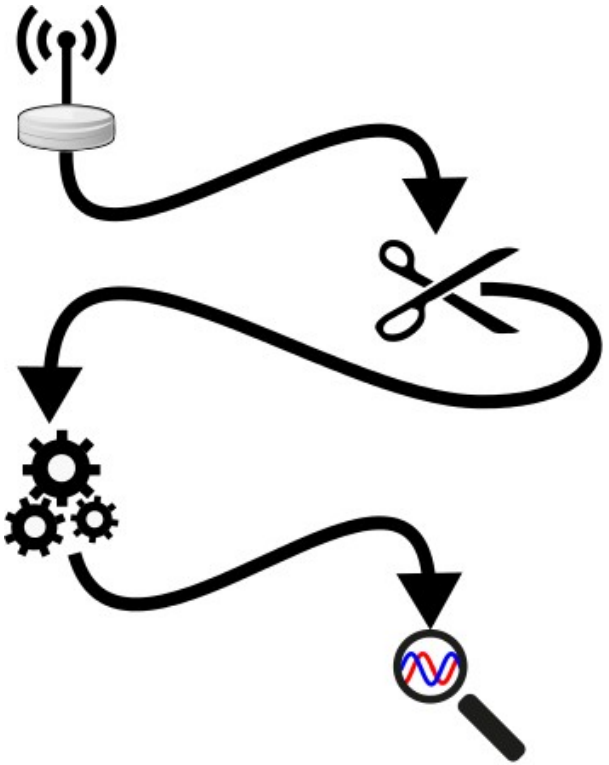
The complexity of the Iridium air interface makes the challenge of developing an Iridium L-Band monitoring device very difficult and probably beyond the reach of all but the most determined adversaries.

- Among the complications are
  - Large, continually changing Doppler shifts
  - Frequent inter-beam and inter-SV handoffs
  - Time-division multiplexed burst mode channels
  - Complicated modulation, interleaving and coding


+1.6262e9
250000
200000
150000
100000
50000
frequency
1000
1500
2000
2500
3000
3500
4000
time
+1.418506e9
400
350
300
250
200
150
100
50
symbols


+1.6264e9
70000
60000
50000
40000
30000
20000
10000
frequency
500
1000
1500
2000
2500
time
+1.419068e9
90
80
70
60
50
40
30
20
10
satno

Tools workflow



runner
(uhd_rx_cfile)

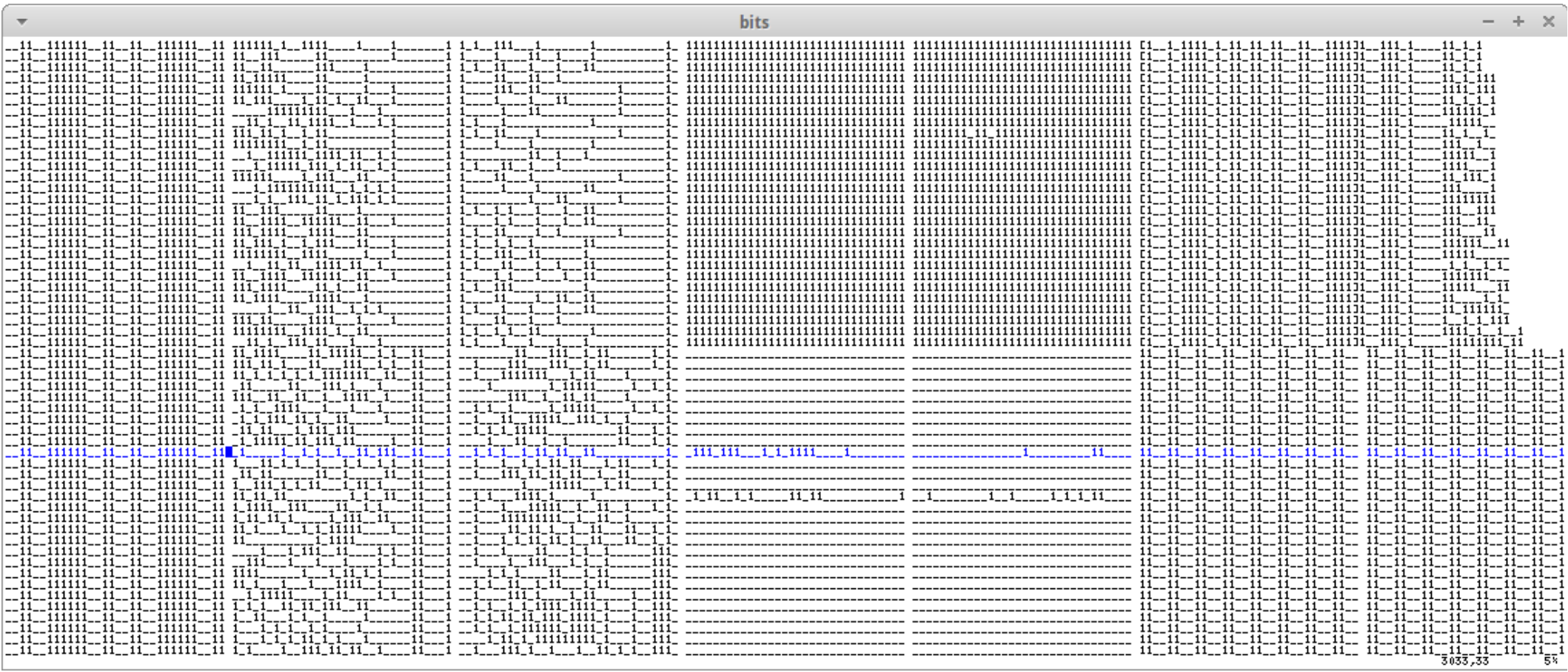
detector-fft.py

cut-and-downmix-2.py

demod.py

doit.pl
(multi-cpu)

Staring at bits



Error Correction

Table 3-1
Subscriber Unit Summary

Coded Data Rate: Uplink Downlink	180 Kbps 400 Kbps
Error Correction Coding	Convolutional, Rate = 3/4, K = 7
Modulation	QPSK
Frequency Band	1610.0-1626.5 MHz

Error Correction

Table 3-1

The frame format for the iridium packets is illustrated in Figure 6. The Frame length is 90 milliseconds, with each transmit burst time of 8.2 milliseconds. There are 8 traffic slots per frame with 4 for uplink and 4 for downlink for duplex operation. The coded data rate is 50 Kbps, and the coding is convolutional FEC, r=3/4,k=7.

Coded Data Rate: Uplink Downlink	180 Kbps 400 Kbps
Error Correction Coding	Convolutional, Rate = 3/4, K = 7
Modulation	QPSK
Frequency Band	1610.0-1626.5 MHz

Error Correction

Table 3-1

The frame format for the iridium packets is illustrated in Figure 6. The Frame length is 90 milliseconds, with each transmit burst time of 8.2 milliseconds. There are 8 traffic slots per frame with 4 for uplink and 4 for downlink for duplex operation. The coded data rate is 50 Kbps, and the coding is convolutional FEC, r=3/4,k=7.

Coded Data Rate: Uplink	180 Kbps
Downlink	400 Kbps
Error Correction Coding	Convolutional, Rate = 3/4, K = 7

In the downlink burst, the supported vocoder information bit rate is 2.4 kbps for digital voice, fax, and data. With rate 3/4 forward error correction (FEC) coding this becomes 3.45 kbps, which includes overhead and source encoding, exclusive of

Error Correction

into the Satellite Data Link Standard (SDLS)). In general, convolutional encoding with Viterbi decoding will continue to be used in earth-orbiting satellite communication systems well into the next century. The Globalstar and Iridium systems use $K = 9$, rate 1/2 and $K = 7$, rate 3/4 convolutional codes, respectively. The rationale for the differing constraint

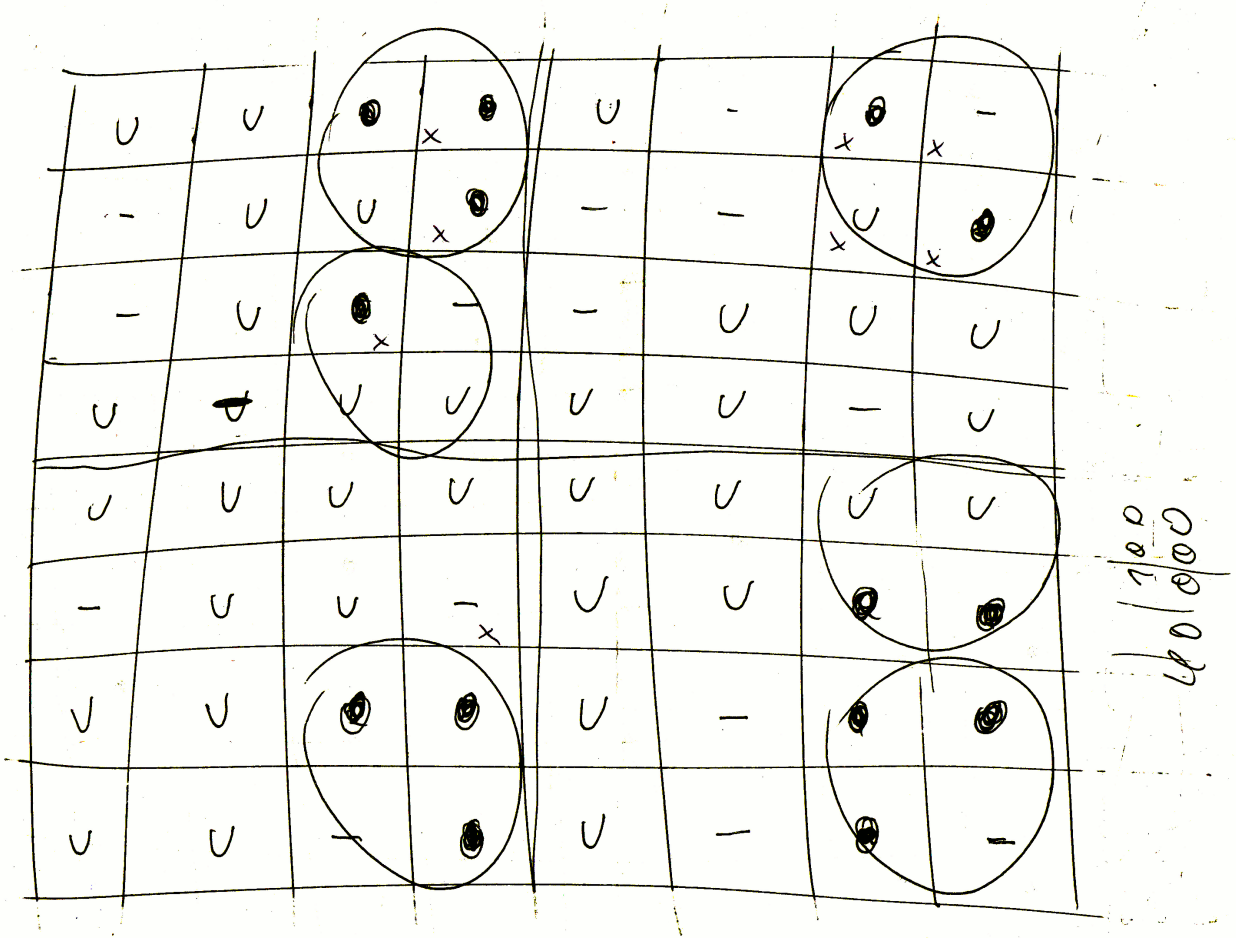
gure 6. The Frame length is 90 milliseconds, with ots per frame with 4 for uplink and 4 for downlink ing is convolutional FEC, r=3/4,k=7.

	ps
Downlink	400 Kbps
Error Correction Coding	Convolutional, Rate = 3/4, K = 7

In the downlink burst, the supported vocoder information bit rate is 2.4 kbps for digital voice, fax, and data. With rate 3/4 forward error correction (FEC) coding this becomes 3.45 kbps, which includes overhead and source encoding, exclusive of

FEC/Interleaving

- Send Messages
  - PPPPPPPPPP
  - PPPQPPPPPP
  - PPPRPPPPPP
  - PPPPQPPPPP
- Find differences
  - Where
  - How many



3 Months later...

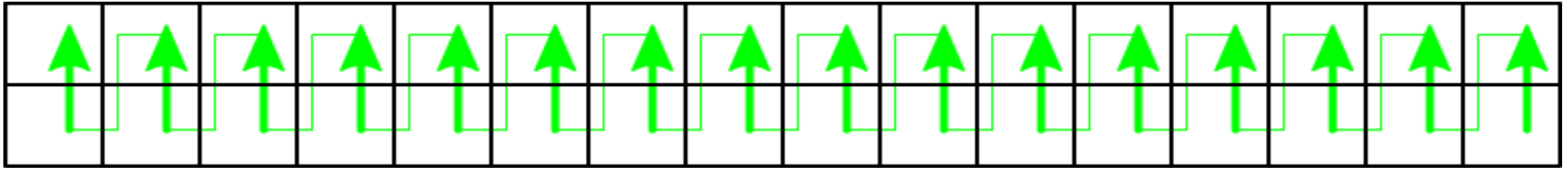
- No convolutional code a.k.a. „Voyager Code“
  - Lots of code written which did not work
- Just interleaving/scrambling
  - 64 bit blocks

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

One box per Symbol (i.e. 2 bits)

3 Months later...

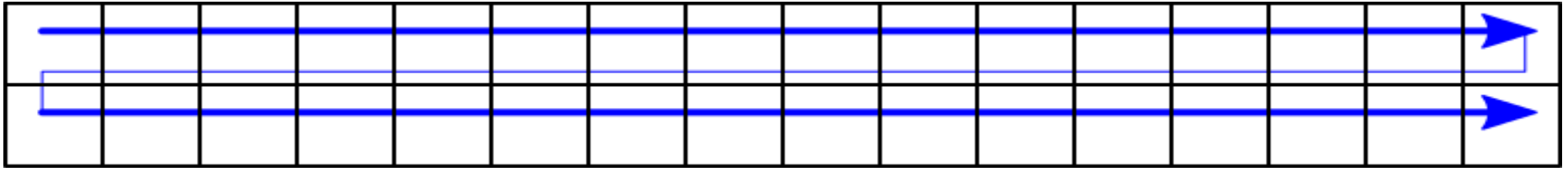
- No convolutional code a.k.a. „Voyager Code“
  - Lots of code written which did not work
- Just interleaving/scrambling
  - 64 bit blocks



One box per Symbol (i.e. 2 bits)

3 Months later...

- No convolutional code a.k.a. „Voyager Code“
  - Lots of code written which did not work
- Just interleaving/scrambling
  - 64 bit blocks

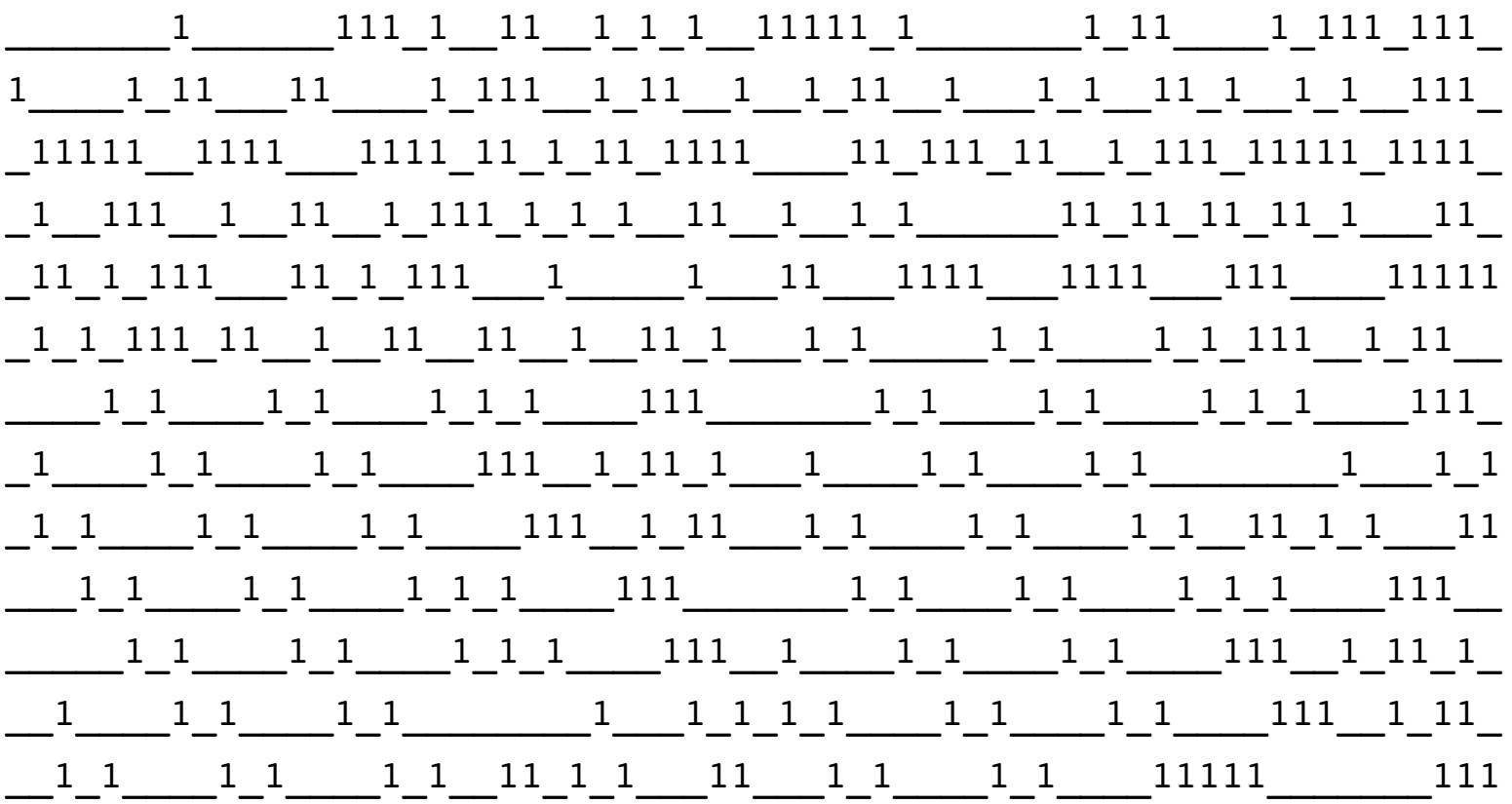


One box per Symbol (i.e. 2 bits)

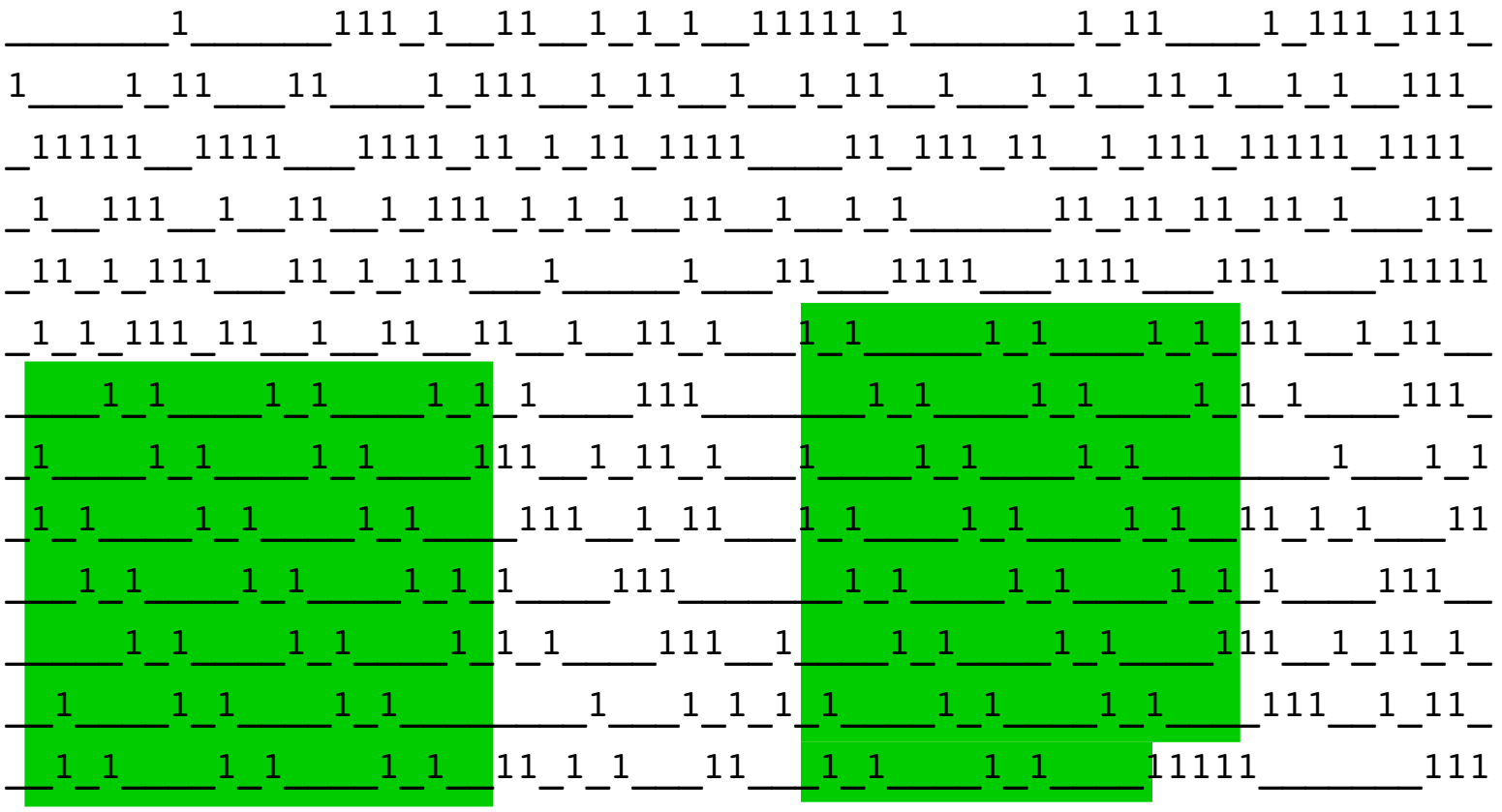
Spot the Ps (1010000)

00000001000000111010011001010100111110100000001011000010111011101000010110001100001011100101100100101100100010100110100101001110011111001111000111101101011011110000110111011001011101111101111001001110010011001011101010100110010010100000011011011011010001100110101110001101011100010000010001100011110001111000111000011111010101110110010011001100100110100010100000101000010101110010110001010111001101011100010000010001100011110001111000111000011111010101110110010011001100100110100010100000101000010101110010110000101000010101000011100000001010000101000010101000011100101101000100001010000101000000000100010100101110010110001010000101000010100110101001000010100001010000111001011010000101000010101000011100000001010000101000010101000011100101101000010100001010000000010001010101000010100001010000111001011000010100001010000111001011010000101000010101000011100100001010000101000010100001110010110100001010000101000011111000000011100101000010100011010100011000101000010100001111100000001110010100001010001101010001100010100001010000111110000000111

Spot the Ps (1_1____)



Found them



Error correction

- 20 bit „Data“ leaves 12 bit for checksum
- Could be Reed-Solomon/BCH
- BCH requires „generator polynomial“
  - Many possibilities, hard to generate
  - Is actually a (n+1)-digit binary number
  - Bruteforce!
- 1897 a.k.a.
  - BCH(31,21)= $x^{10}+x^9+x^8+x^6+x^5+x^3+1$
  - 32nd bit is Parity

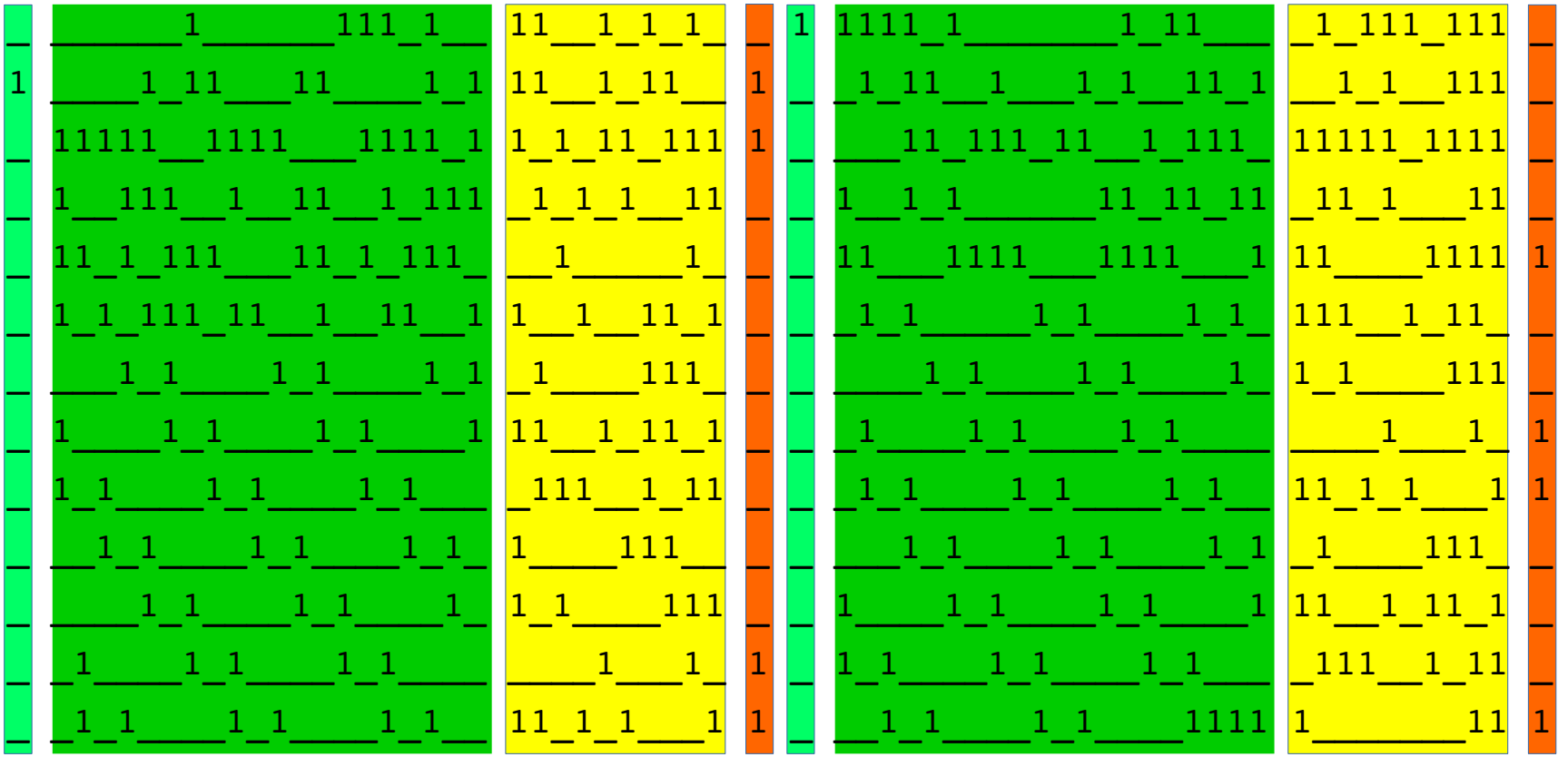
Error correction

- 20 bit „Data“ leaves 12 bit for checksum
- Could be Reed-Solomon/BCH

data calls. This protocol results in each Iridium TDMA burst consisting of 160 SBD information bits, 20 SBD header bits and 234 other overhead bits. These 160 information bits are protected by a BCH(31,20) FEC code. This error correction coded data is then protected by a 16-bit CRC error detection code that is used in conjunction with a

  - Bruteforce!
- 1897 a.k.a.
  - BCH(31,21)= $x^{10}+x^9+x^8+x^6+x^5+x^3+1$
  - 32nd bit is Parity

Packet Layout



Payload BCH Parity

Decoded Message

0000 0010 000011 0101 00	Zero, block=2, frame=3, len=5, group=0
111101000000010110000 00101	RIC, msg_fmt=5
100011 0000 0110000010	msg_seq=35, Zero, unknown
1 0001 1 1 0 0110111	msg_idx=2/2, Zero, msg_csum=55
1010000 1010000 1010000 1010000	P P P P
1010000 1010000 1010000 1010000	P P P P
1010000	P
0000011 0000011 0000011 0000011	ETX ETX ETX ETX
0000011	ETX
11111111111111111111	Filler

Decoded Message

0000 0010 000011 0101 00	Zero, block=2, frame=3, len=5, group=1
111101000000010110000 00101	RIC, msg_fmt=5
100011 0000 0110000010	msg_seq=35, Zero, unknown
1 0001 1 1 0 0110111	msg_idx=2/2, Zero, msg_csum=55
1010000 1010000 1010000 1010000	P P P P
1010000 1010000 1010000 1010000	P P P P
1010000	P
0000011 0000011 0000011 0000011	ETX ETX ETX ETX
0000011	ETX
11111111111111111111	Filler

Decoded Message

0000 0010 000011 0101 00	Zero, block=2, frame=3, len=5, group=1
111101000000101100000 00101	RIC, msg_fmt=5
100011 0000 0110000010	msg_seq=35, Zero, unknown
1 0001 1 1 0 0110111	msg_idx=2/2, Zero, msg_csum=55
1010000 1010000 1010000 1010000	P P P P
1010000 1010000 1010000 1010000	P P P P
1010000	P
0000011 0000011 0000011 0000011	ETX ETX ETX ETX
0000011	ETX
11111111111111111111	Filler

Decoded Message

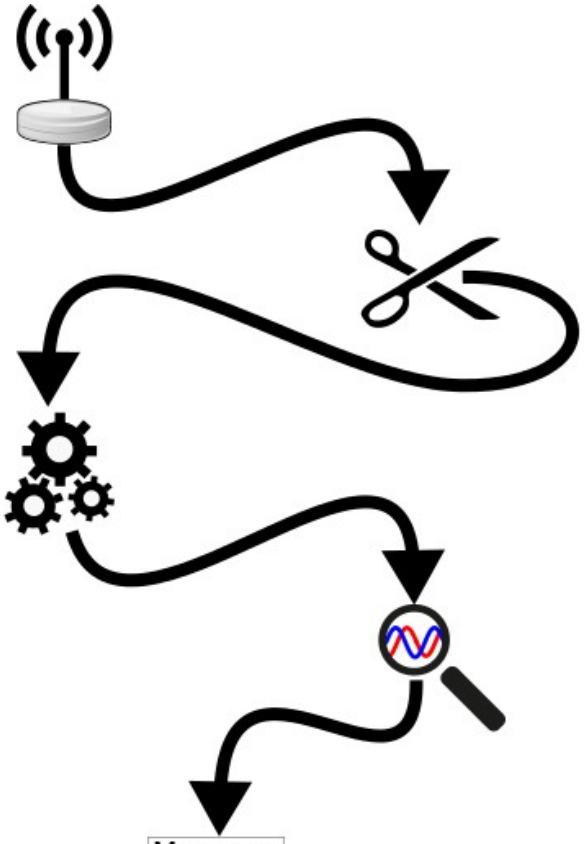
0000 0010 000011 0101 00	Zero, block=2, frame=3, len=5, group=1
11110100000001011000000 00101	RIC, msg_fmt=5
100011 0000 0110000010	msg_seq=35, Zero, unknown
1 0001 1 1 0 0110111	msg_idx=2/2, Zero, msg_csum=55
1010000 1010000 1010000 1010000	P P P P
1010000 1010000 1010000 1010000	P P P P
1010000	P
0000011 0000011 0000011 0000011	ETX ETX ETX ETX
0000011	ETX
11111111111111111111	Filler

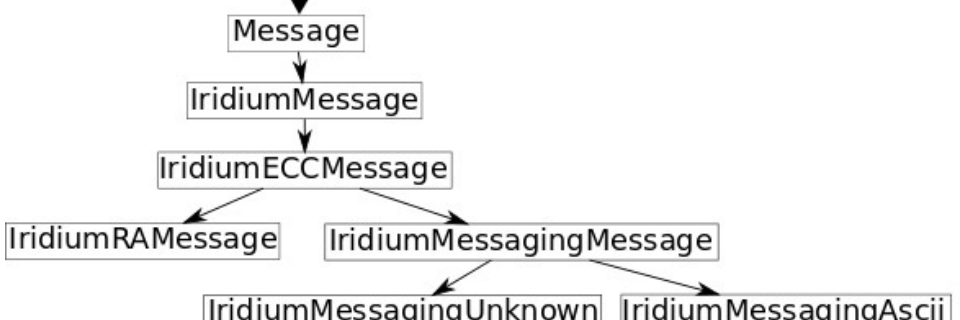
Decoded Message

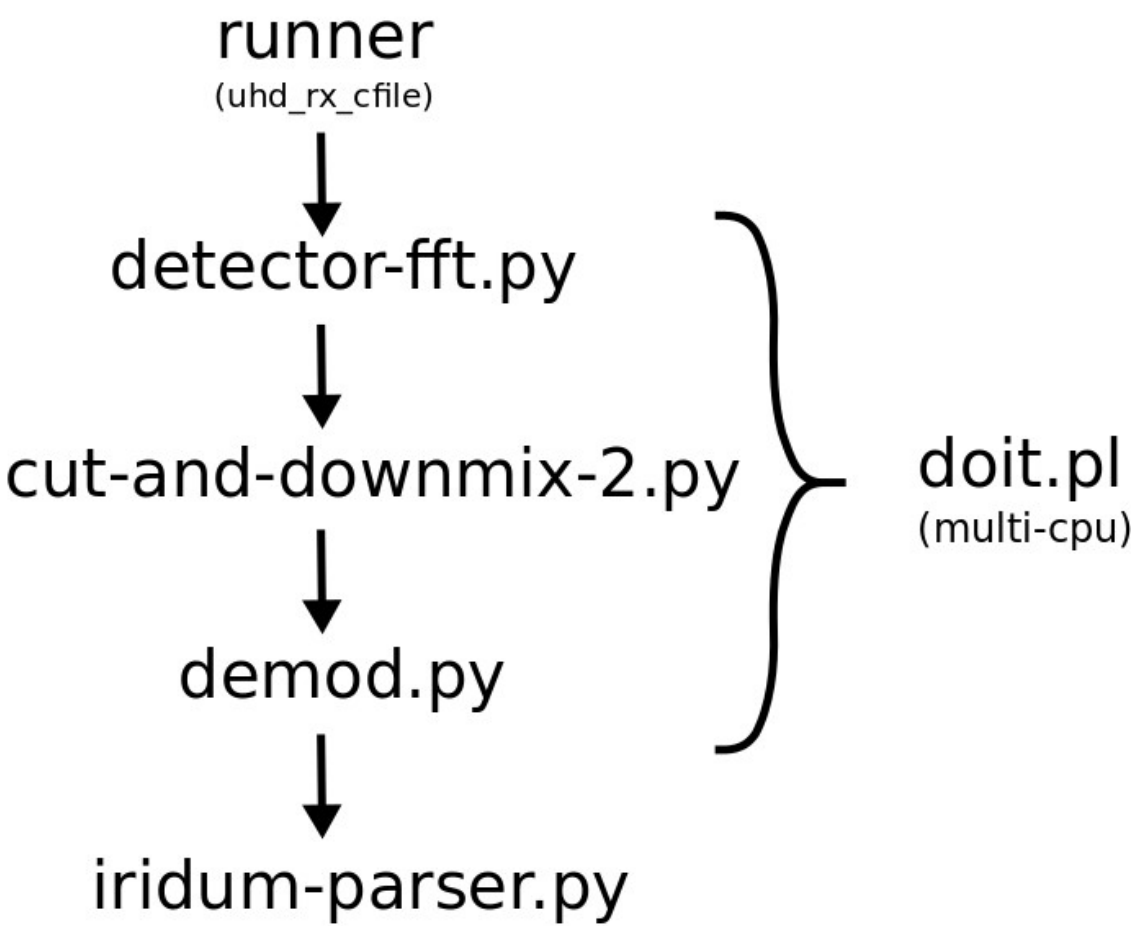
0000 0010 000011 0101 00	Zero, block=2, frame=3, len=5, group=1
11110100000001011000000 00101	RIC, msg_fmt=5
100011 0000 0110000010	msg_seq=35, Zero, unknown
1 0001 1 1 0 0110111	msg_idx=2/2, Zero, msg_csum=55
1010000 1010000 1010000 1010000	P P P P
1010000 1010000 1010000 1010000	P P P P
1010000	P
0000011 0000011 0000011 0000011	ETX ETX ETX ETX
0000011	ETX
11111111111111111111	Filler

Decoded Message

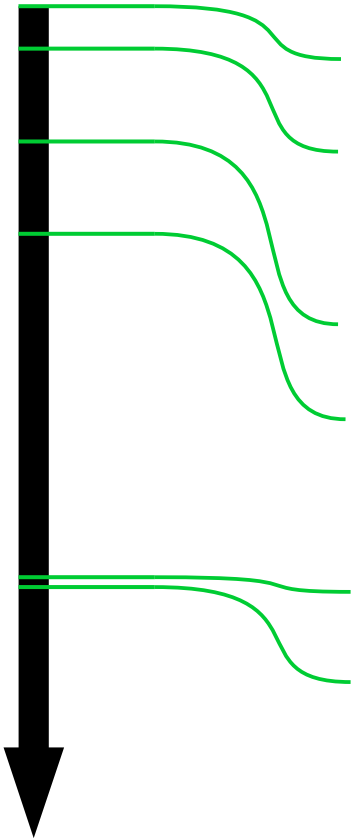
0000 0010 000011 0101 00	Zero, block=2, frame=3, len=5, group=1
111101000000010110000 00101	RIC, msg_fmt=5
100011 0000 0110000010	msg_seq=35, Zero, unknown
1 0001 1 1 0 0110111	msg_idx=2/2, Zero, msg_csum=55
1010000 1010000 1010000 1010000 1010000 1010000 1010000 1010000 1010000	P P P P P P P P P
0000011 0000011 0000011 0000011 0000011	ETX ETX ETX ETX ETX
1111111111111111111	Filler




Message
IridiumMessage
IridiumECCMessage
IridiumRAMessage
IridiumMessagingMessage
IridiumMessagingUnknown
IridiumMessagingAscii


runner
(uhd_rx_cfile)
detector-fft.py
cut-and-downmix-2.py
demod.py
iridum-parser.py
doit.pl
(multi-cpu)

Timeline



2014-04-20 First idea

2014-05-03 Increased team size by 100%

- bought Antenna & LNA

2014-05-31 First signal received

2014-06-28 Identified unique word (& gave up on gnuradio)

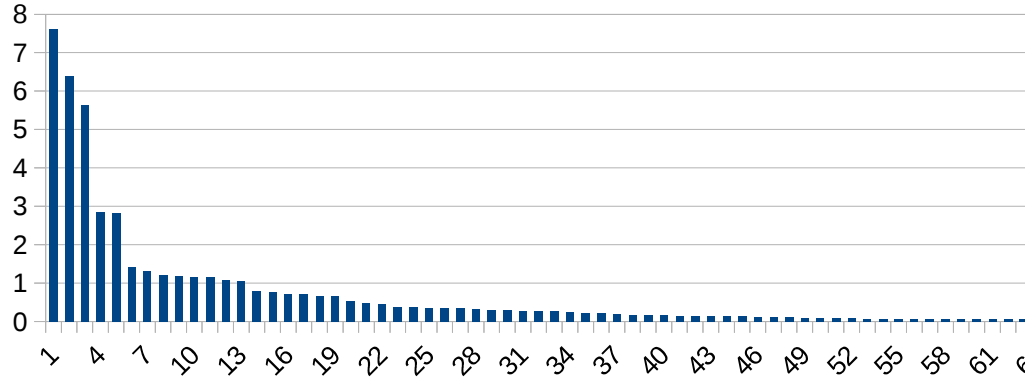
- stared at bits

2014-10-10 turns out k=7 was a lie

2014-10-13 first third party message decoded

Statistics (.de)

- 119 unique recipients
  - #1 receives 16%
  - #1-#5 receive 50% of messages
- ~50-100 Messages per day
  - 36-58% crypto
  - 16% [redacted]
  - 12% testing
  - 3% call me
  - 4% military
  - 6% regular



Lots of interesting stuff left to do

- Understand more protocol
- Voice communication
    - GSM-specified algorithm A3 for authentication security
    - „proprietary“ AMBE-family voice codec (OsmoGMR may be of help)
- SBD: Short Burst Data
    - uses the Iridium signaling channel for data transport
- RUDICS: Ciruit Switched Data over Iridium
- AMS: Aircraft communication

- SDR Workshop in 30 minutes (also maybe tomorrow)
  - 17:45 in Hall 13, Bring Laptop
- SDR-Corner near chaoswelle (near Hall 2)
- We have Equipment @ µc³ Assembly
  - Network Analyzer, USRP B-200
- Code is on github (BSD Licence)
  - https://github.com/muccc/iridium-toolkit
- Iridium System Specification / Iridium Radio Link Protocol Specification
  - We want it
  - Also any other documentation you might have
  - No questions asked

<sec@42.org>

26A5 7E7C A201 73FA 8D90
DD96 B86F 0A34 **AB9E 3213**

<schneider@muc.ccc.de>

A471 3753 2EC1 E5FF A673
812C 5C85 6CAA **96ED 4C12**