Bitstream clock synchronization in an ACARS receiver: porting gr-acars to GNU Radio 3.8

T. Lavarenne[1], J.-M Friedt[2]

[1] Lycée Jean Rostand, Villepinte, France
[2] FEMTO-ST/Time & Frequency department, Besançon, France

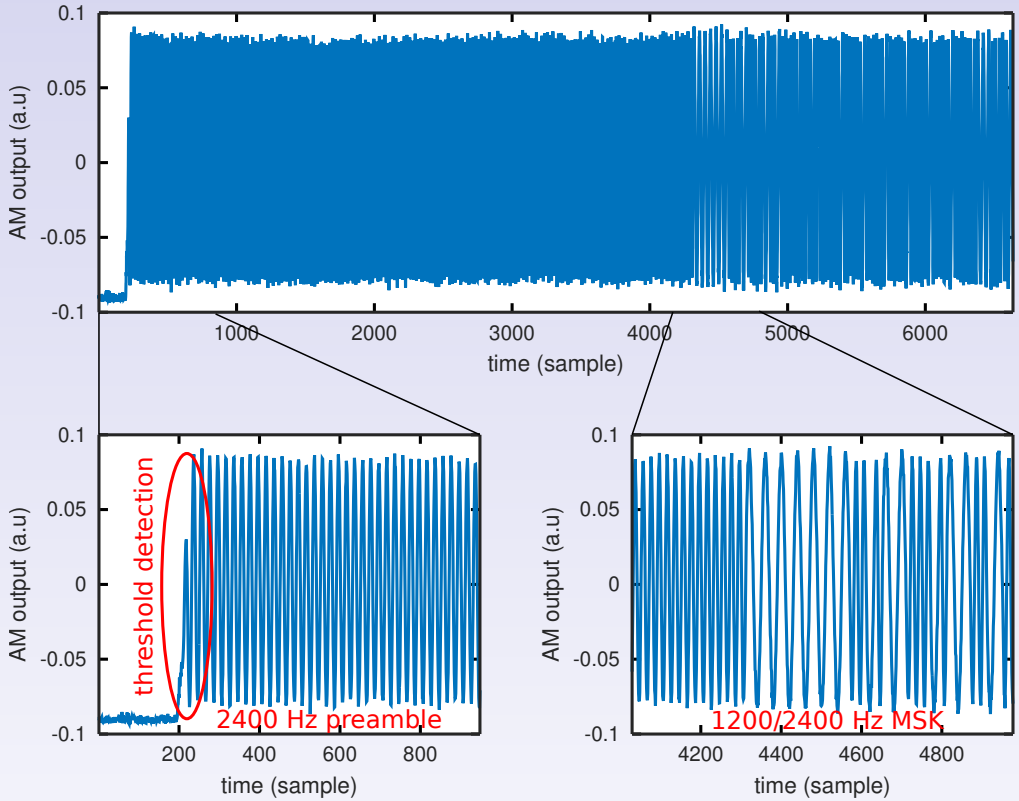
jmfriedt@femto-st.fr

June 27, 2020

ACARS and gr-acars

Aircraft Communications Addressing and Reporting System

- Protocol developed in the 70s for communicating between plane pilots and ground
- Telemetry and free text
- As with all aeronautical protocols, AM mod.→
- Information is MSK modulated at 2400 bps: 0=half 1200 Hz period, 1=full 2400 Hz period
- Differential encoding (0=keep bit state, 1=change bit state)
- Header: synchronization using 16 bytes with ones only (2400 Hz)



- Three channels in European: primary (131.725 MHz), secondary (131.525 MHz), additional (131.825 MHz, 131.850 MHz ?)

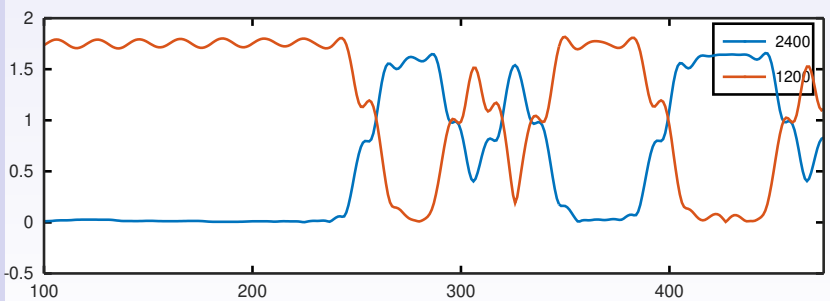
→ very simple protocol, excellent topic for a first GNU Radio OOT block

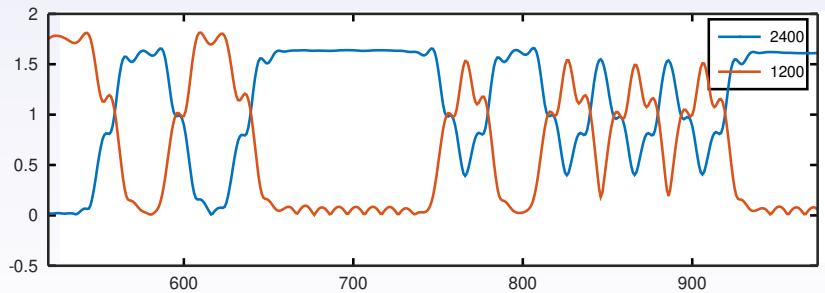
[0] lists.osmocom.org/pipermail/osmocom-sdr/2012-August/000394.html

Bit decoding

- Two tones define the two bit states: convolution[1] with one period = bandpass filter

```
fs=48000;
bps=2400;
f2400=exp(j*2*pi*2400*[0:1/fs:1/bps*2]');f2400=f2400(1:end-1);  % why 2 periods ?
f1200=exp(j*2*pi*1200*[0:1/fs:1/bps*2]');f1200=f1200(1:end);    % why 2 periods ?
% s2400=abs(conv(y,f2400));s2400=s2400(200:end);  % convolution
% s1200=abs(conv(y,f1200));s1200=s1200(200:end);
% b=firls(256,[0 3400 4500 fs/2]*2/fs,[1 1 0 0]); % low pass filter taps
% s2400=filter(b,1,s2400);s2400=s2400(200:end);   % filter (remove 4800 Hz)
% s1200=filter(b,1,s1200);s1200=s1200(200:end);
sf=(fft(y));
f2400f=(fft(f2400,length(sf)));
f1200f=(fft(f1200,length(sf)));
sf2400f=sf.*f2400f;          % convolution
sf1200f=sf.*f1200f;
df=fs/length(sf);            % FFT bin width (in Hz)
fcut=floor(3500/df)          % cutoff bin position
sf2400f(fcut:end-fcut)=0;    % low pass filter/Matlab FFT convention (0 left, fs/2 middle)
sf1200f(fcut:end-fcut)=0;
s2400=abs(ifft(sf2400f));    % back to time domain
s1200=abs(ifft(sf1200f));
plot(s2400,'-');hold on; plot(s1200,'-'); legend('2400','1200')
```

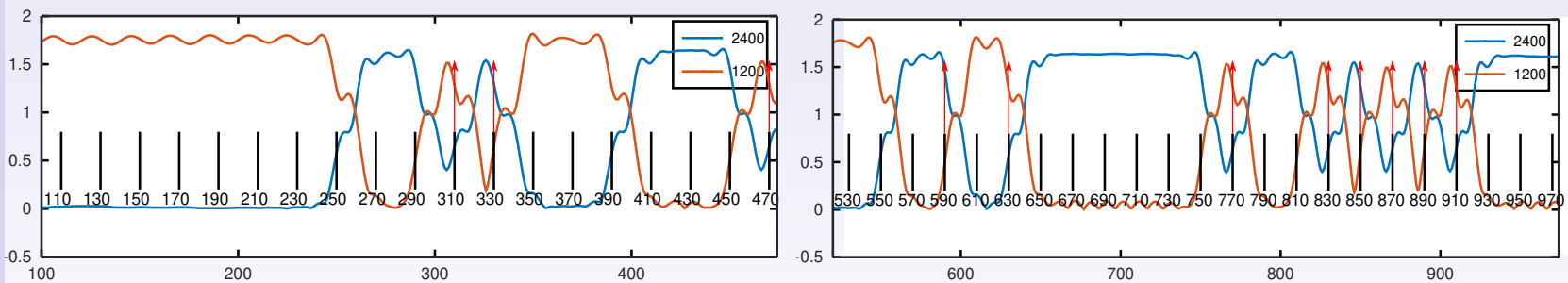




[1] $FFT(x \circledast y) = FFT(x) \cdot FFT(y)$

Bit decoding

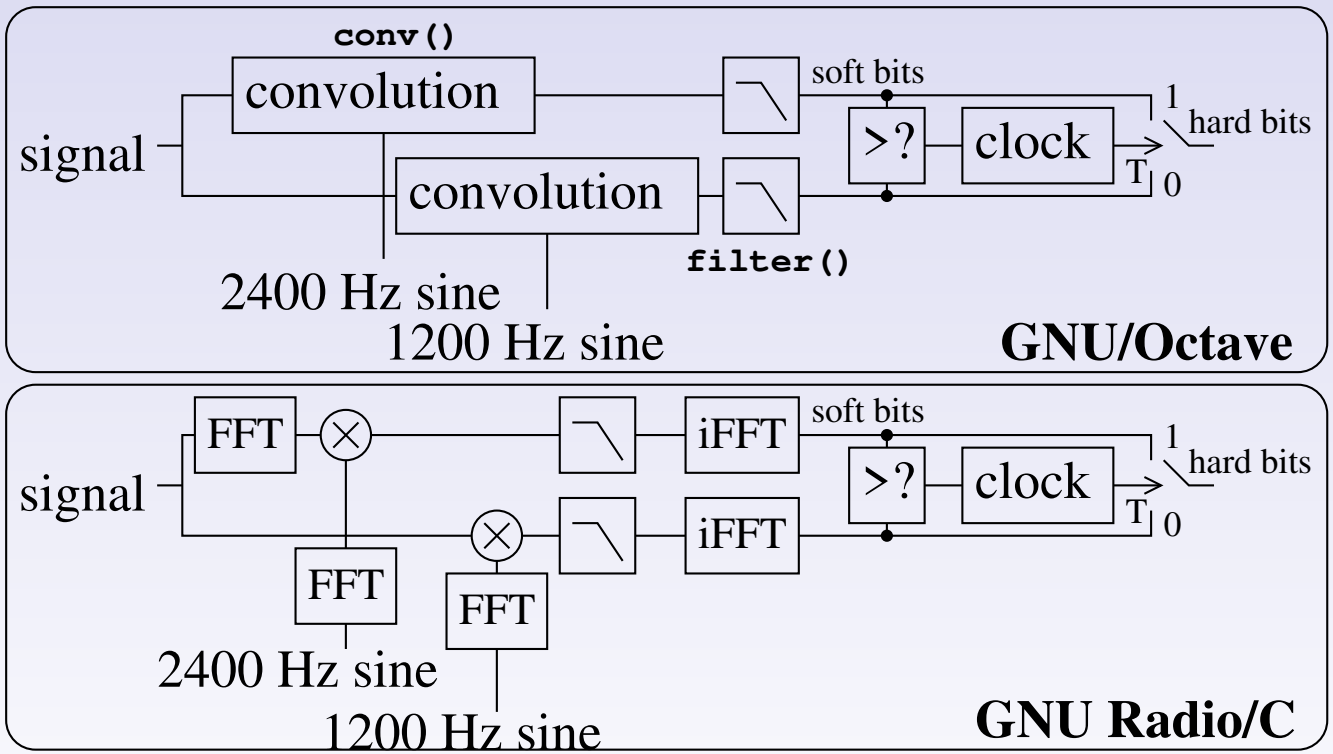
- Detect message as **rise in the standard deviation** of the received signal to avoid continuously running FFTs (might need refinement)
- Two tones define the two bit states: convolution with one period = bandpass filter
- Check every bit period the **most probable bit state**: soft bit $\rightarrow$ hard bit by threshold detection
- Sampling @ 48 kS/s and 2400 bps message: check state every 20 samples.



Soft-bit sequences following convolution with 2400 Hz (blue) and 1200 (red) bandpass filters. Red arrows: **poor sampling** time due to initial poor threshold detection.

GNU Octave to GNU Radio

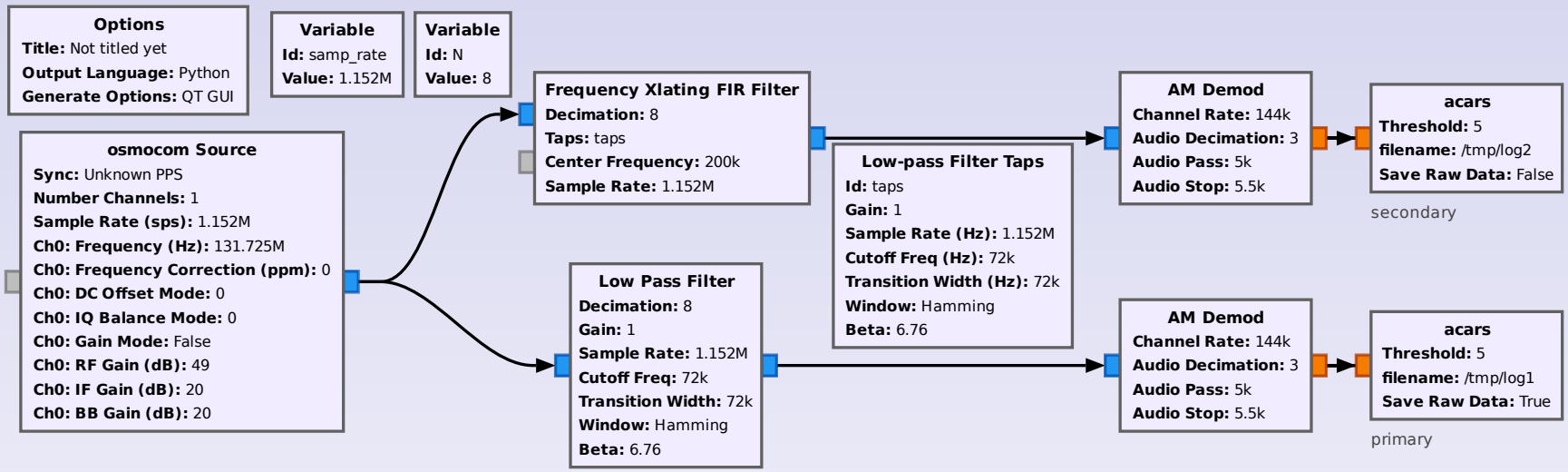
- Convolution as FFT product
- Filtering as FFT weighing



$\Rightarrow$ insert filtering before inverse Fourier transform

Bit decoding: FFT implementation

FFTW → GNU Radio FFT



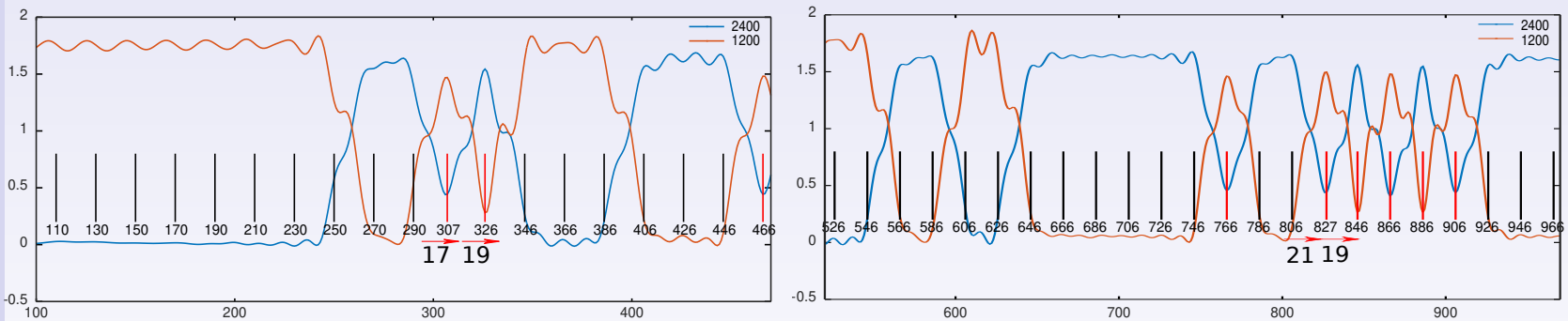
The FFTW initialization is **not** thread safe:
`http://www.fftw.org/fftw3_doc/Thread-safety.html` *planner routines share data (e.g. wisdom and trigonometric tables) between calls and plans ... Neither strategy works, however, in the following situation. The "application" is structured as a set of "plugins" which are unaware of each other, and for whatever reason the "plugins" cannot coordinate on grabbing the lock. (This is not a technical problem, but an organizational one ..."*

Clock synchronization

- Challenge in accurately finding the beginning of the sentence (sync. word $\rightarrow$ payload)
- Differential: any synchronization loss will last until end of sentence
- Challenge to keep long sentences consistent

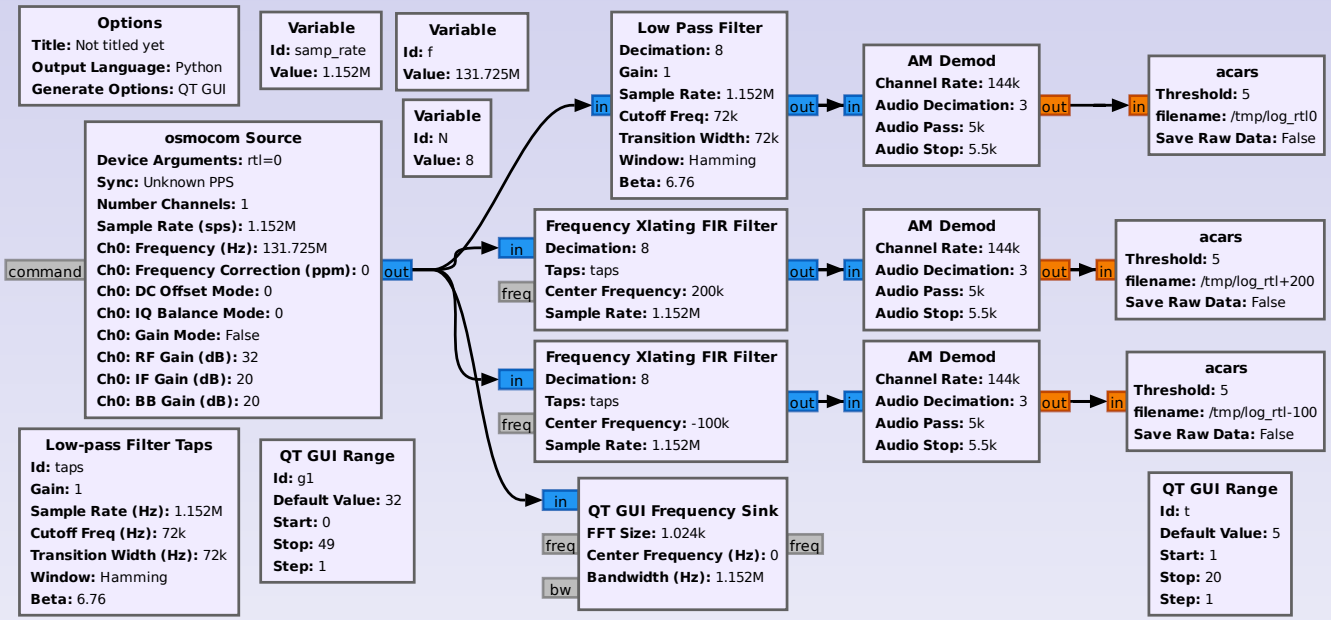
$\Rightarrow$ need for **clock synchronization**

- **Proposed algorithm**: identify unique bit state (1 framed with 0s or 0 framed with 1s) and lock on signal maximum



Black vertical lines indicate when successive hard bits are expected to be the same, **red vertical lines indicate when a hard bit is framed with two different hard-bit states and are hence used for clock synchronization**

Results: single receiver, multiple channels



Message structure[2]:

bits:	128	16	16	8	8	56	8	16	8	8	32	48	???	8	128	8
name	pre-key	bit sync	char sync	SOH	mode	addr	(n)ack	label	block ID	STX	Seq No.	Flight No.	Text	ETX	CRC	BCS
Eg.	010101 ...	0x2b 0x2a	0x16 0x16	0x01	0x45	.A7-...				02				03		

Example (hex and ASCII) ...

```
2b 2a 16 16 01 45 2e 41 37 2d ...
+*E.A7-BAKB02L82AQR3256/LSAG.AFN/FMHQTR3256,.A7rBAK,06A078,144616/FPON47427E007102,1/FCOADS,01/FCOATC,01247A
```

... interpreted as:

```
Aircraft=.A7-BAK
STX
Seq. No=4c 38 32 41 L82A
Flight=QR3256
/LSAG.AFN/FMHQTR3256,.A7-BAK,06A078,144616/FPON47427E007102,1/FCOADS,01/FCOATC,01247AETX
```

[2] http://www.scancat.com/Code-30_html_Source/acars.html

Sentences and parity bit

8th bit is odd-parity bit

Constant bit rate clock increment:

```
+*2.F-GZNOH12D23CAF4195#DFB 965  965000011 50 50
 60 62318319148145 07 08 00 26 -06-310-308-309-310-...
-239-239-239-310-310-311-309-310 016 017-392-391-026
 000 00Q.09..*POU/*: P/0 P/P_/P/ P/P_/PQP|uU
```

the messages ends up garbled

```
CRC:00000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000
00000000000000000000000000011000011011100110011101 1
1011101101011
```

parity bit: desynchronization shown as non-0 parity bits.

Clock synchronization

```
+*2.F-GZNOH12D23CAF4195#DFB 965  965000011 50 50
 60 62318319148145 07 08 00 26 -06-310-308-309-310-...
-239-239-239-310-310-311-309-310 016 017-392-391-026
 000 001109113005029 000 000 000 000 00102
```

full message properly decoded

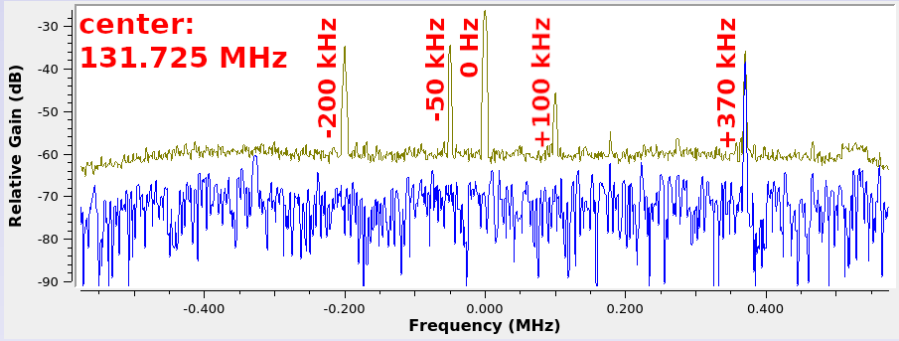
```
CRC:00000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000
0000000000011
```

parity bit state

42 sentences collected over 6 minutes from a location a few kilometers from Roissy Charles de Gaulle airport in Paris: 15 would have been corrupted if decoded without clock synchronization $\Rightarrow$ **35% success rate improvement**

Results: single receiver, multiple channels

← 131.725 MHz

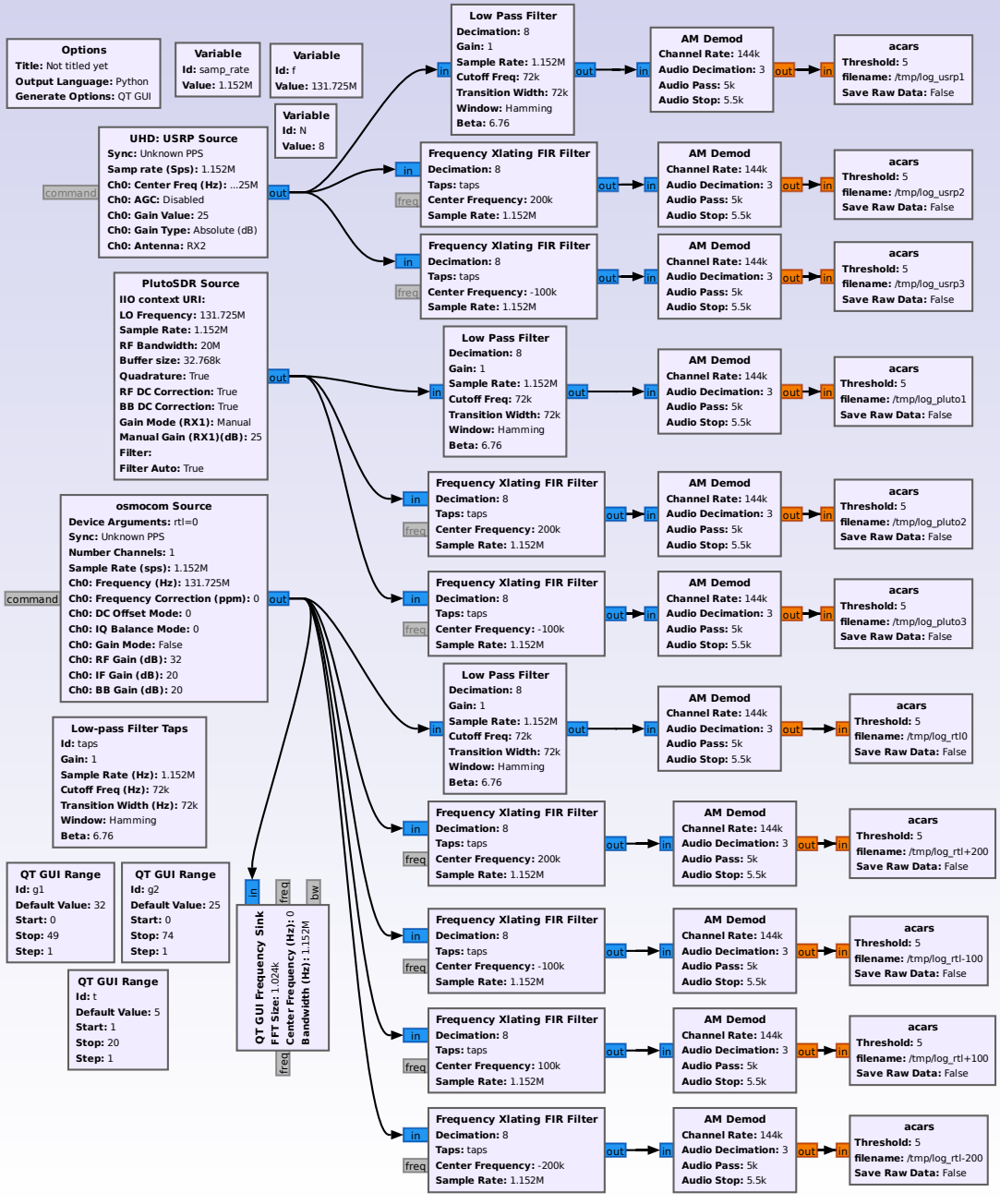


↓ 131.825 MHz

```
Fri May  1 17:53:22 2020
Aircraft=.HB-JMI
Seq. No=53 34 36 41 S46A
Flight=LX0018
ETX

Fri May  1 17:53:27 2020
Aircraft=.HB-JMI
Seq. No=53 34 37 41 S47A
Flight=LX0018
ETX

Fri May  1 19:32:18 2020
Aircraft=.B-5921
Seq. No=44 35 37 41 D57A
Flight=MU7080
#DFBR04/A33004,1,1C1,.B-5921,
20MAY01,17.19.51,LSGG,ZSPD,
CES7080   ,4000,013C2,016,
05.0,000000,D1333RR06C0MU12,
000,012,012C3,015.0,01962,
0.252,111,1.53,111,10,0,01,
111,1.61,XC4,015.0,01859,
0.252,111C5,!:


Fri May  1 19:32:23 2020
Aircraft=.B-5921
Seq. No=44 35 37 42 D57B
Flight=MU7080
#DFB042335,07008,12610,
01345,00104,00016,00000,
00802,06066C6,042188,09517,
32847,01911,00104,00016,
00000,00802,16066C7,01,00,
00,00,00,00,01,00C8,01,00,
00,00,00,00,01,00N1,1.458,
1.459,1.458,1.611,082.08,
0,d

Fri May  1 19:32:30 2020
Aircraft=.B-5921
Seq. No=44 35 37 43 D57C
Flight=MU7080
#DFB92.6,093.9,0704,
21.152N2,1.459,1.459,1.458,
1.611,082.17,092.1,093.8,
0725,21.121S1,08895,14.35,
015.0,097.7,241.4,418.2,
546.4,08.9,078.0,078.0S2,
08886,14.36,015.0,097.3,
240.5,414.4,546.9,09.6,
077.9,077.F

Fri May  1 19:32:54 2020
Aircraft=.B-5921
Seq. No=44 35 37 47 D57G
Flight=MU7080
#DFB7,0487,0428T4,023,45,
151,073,097,119.8,022.6,
047.0,047,0488,0424V3,
0.15,0.05,0.09,042,0.25,
008.3V4,0.15,0.08,0.02,
191,0.19,008.0X1,1.477,
012.5,03050,0.237,2,20X2,
1.527,004.5,10025,0.395,0,
80X3,1.

Fri May  1 19:33:00 2020
Aircraft=.B-5921
Seq. No=44 35 37 48 D57H
Flight=MU7080
#DFB.567,-03.5,20011,0.660,
0,80X4,1.584,-12.7,25031,
0.731,0,80:ETX
```

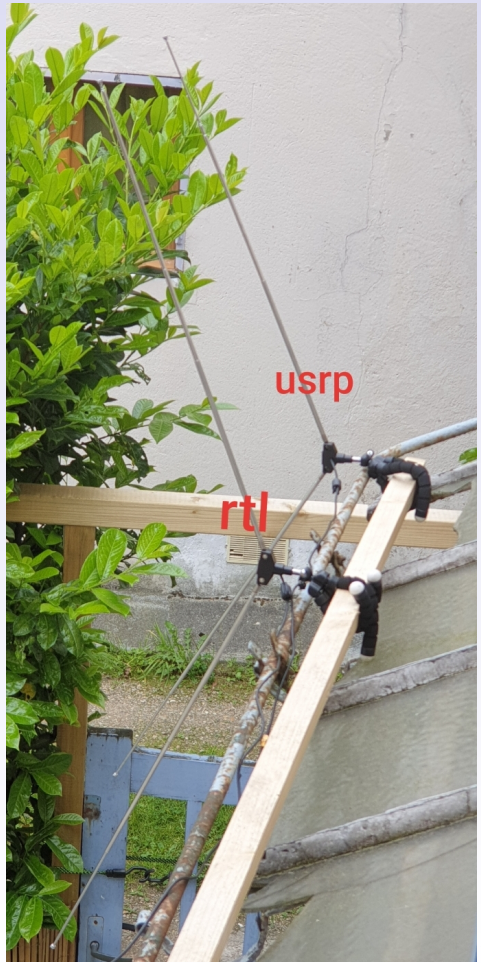
```
Fri May  1 12:09:16 2020
Aircraft=..N22UB
Seq. No=53 30 32 41 S02A
Flight=ZD0001
ETX

Fri May  1 14:32:07 2020
Aircraft=.LX-YCV
Seq. No=53 32 30 41 S20A
Flight=CV7944
ETX

Fri May  1 17:52:30 2020
Aircraft=.F-GUOC
Seq. No=44 38 31 41 D81A
Flight=AF6746
#DFBOP DATAFUEL1552 33997
2.33 -0.3 6200       0  6500
12700 0.816 0.814  0.816 73600
72360TRUE   TRUE    -11 669 789
ETX

Sat May  2 16:46:21 2020
Aircraft=.A7-BAK
Seq. No=4c 38 32 41 L82A
Flight=QR3256
/LSAG.AFN/FMHQTR3256,
.A7-BAK,06A078,
144616/FPON47427E007102,1/FCOADS,
01/FCOATC,01247AETX

Sun May  3 07:34:07 2020
Aircraft=.LX-YCV
Seq. No=53 31 38 41 S18A
Flight=CV062M
ETX
```

Results: multiple receivers

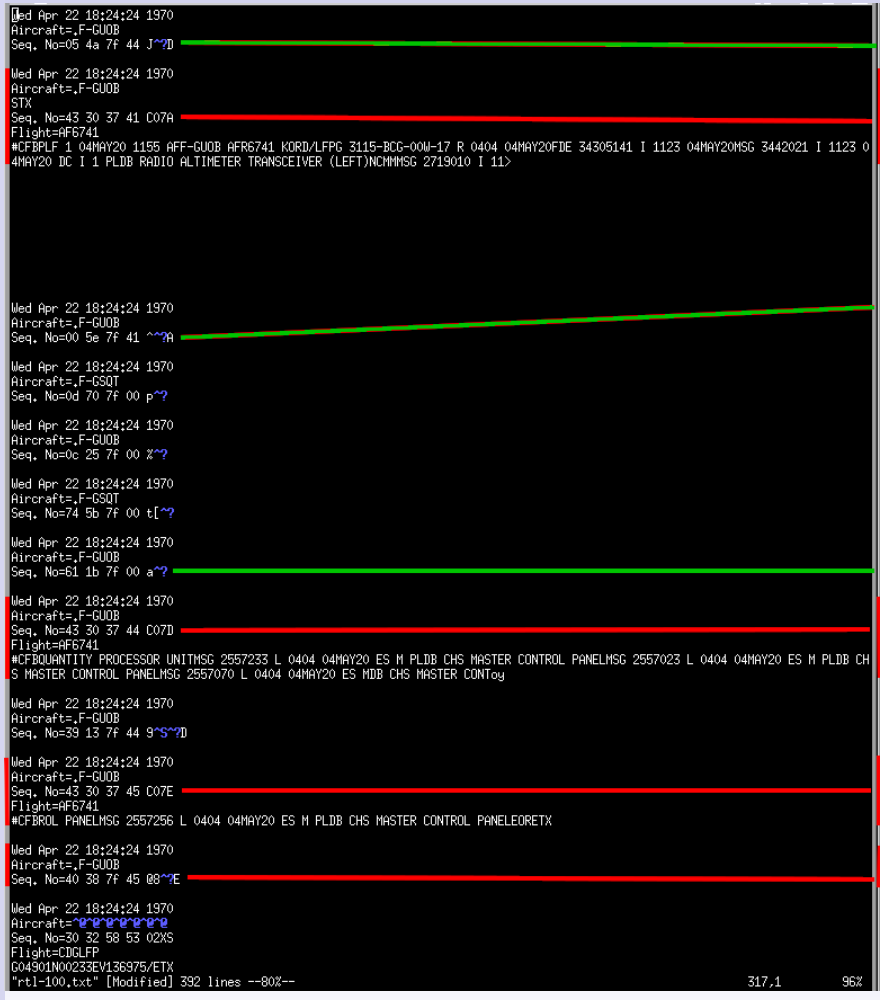


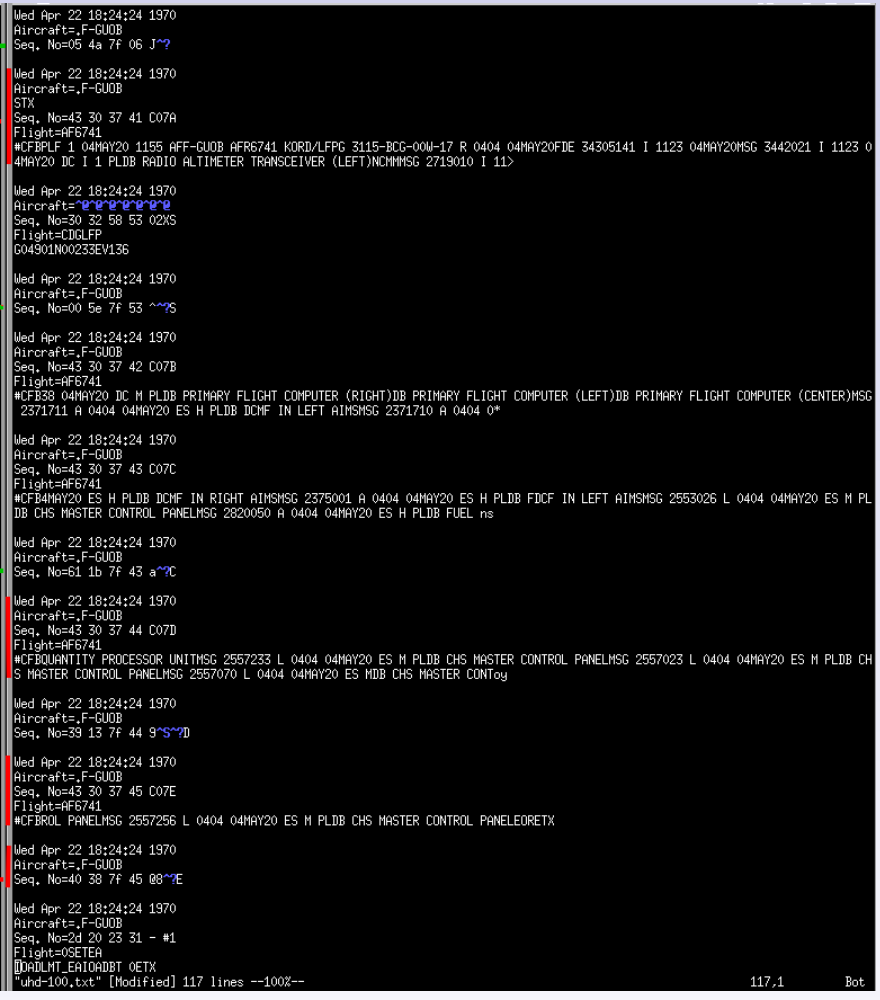
Three sources (RTL-SDR, UHD for B200 and PlutoSDR) all connected to three ACARS channels

Each source connected to a different antenna



Results: multiple receivers



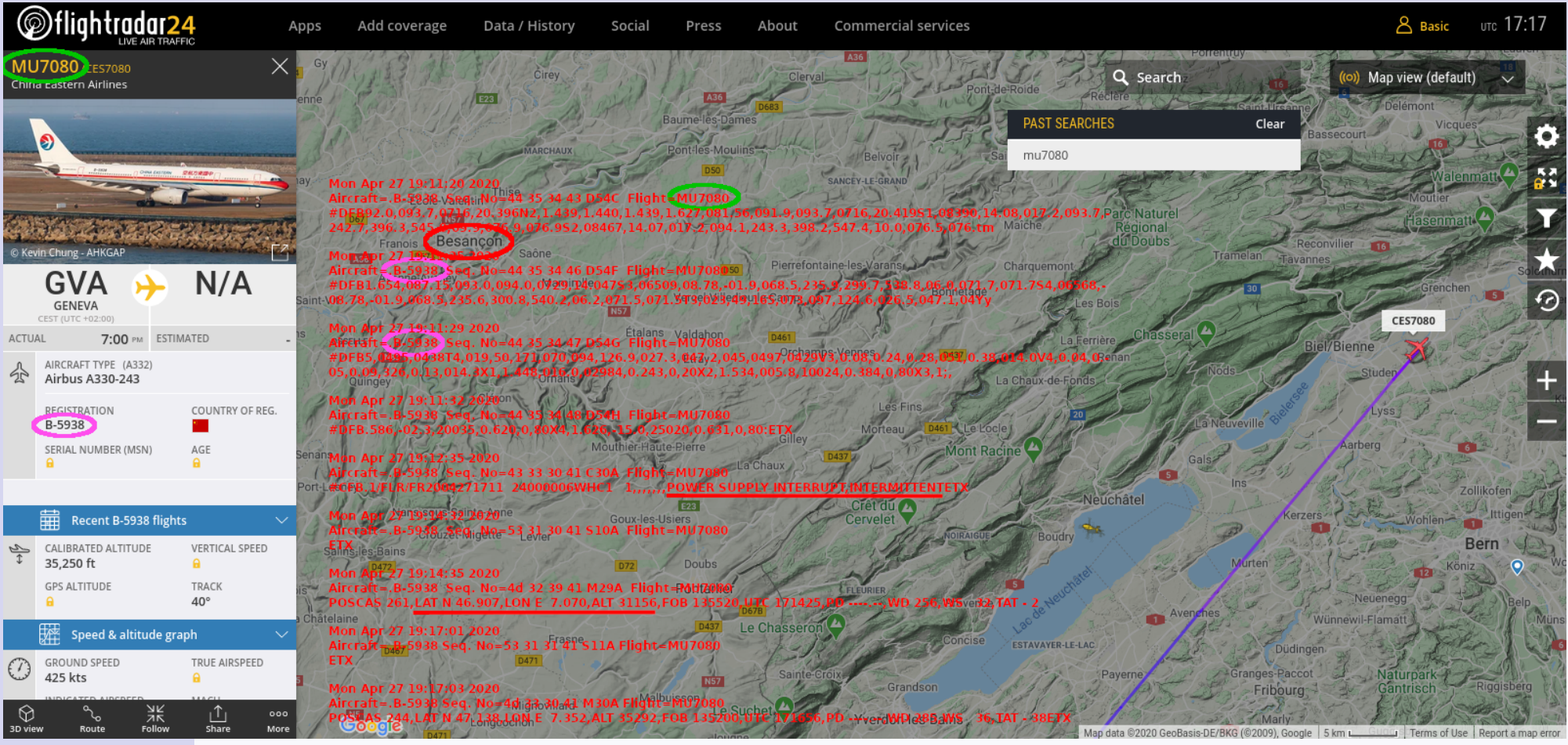


RTL-SDR (131.825 MHz) B200 (131.825 MHz)

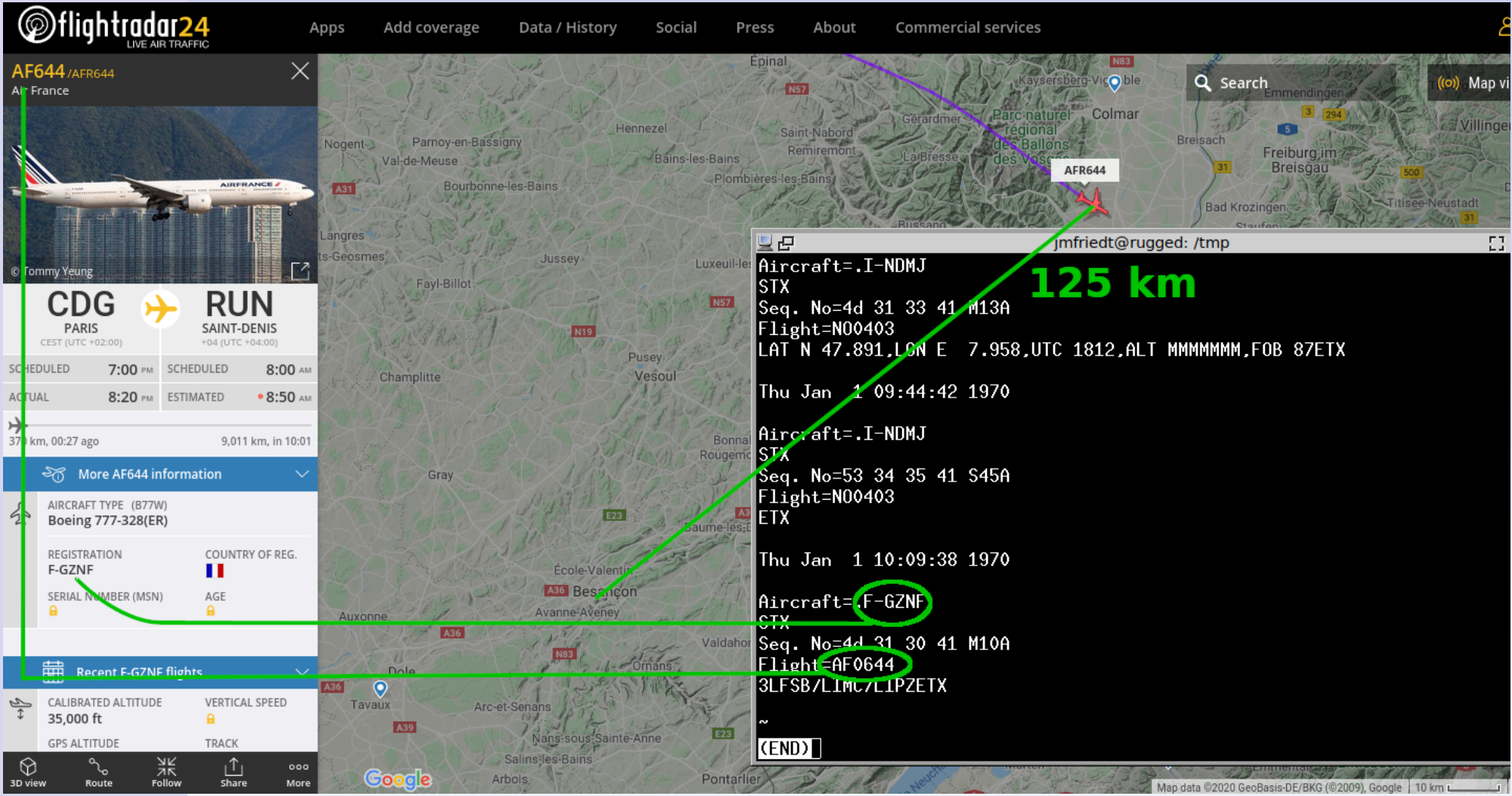
Red: identical messages by RTL-SDR and B200
Green: nearly identical messages by RTL-SDR and B200

Results

Decoded sentences (over the ether)

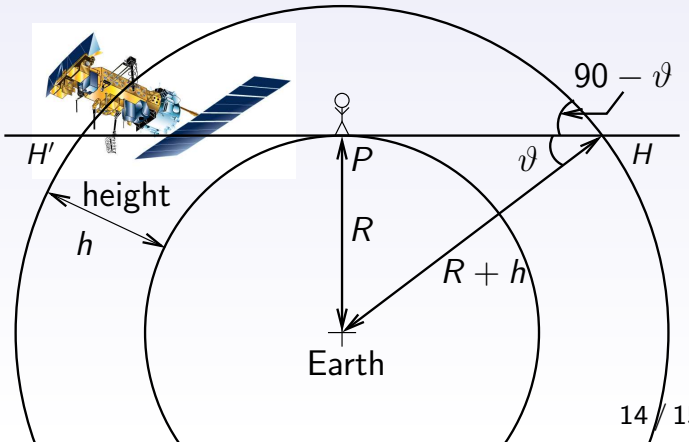


consistent with ADS-B messages collected from `flightradar24.com`



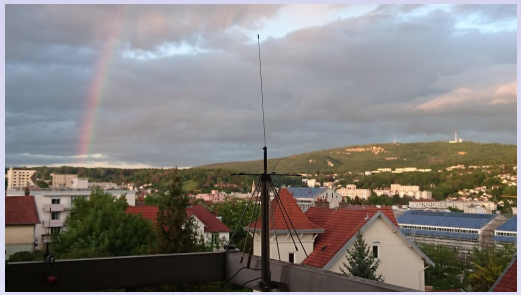
An emitter flying at altitude h over the Earth with radius $R = 6400$ km can be detected (line of sight) at range $d = R \cdot \arccos\left(\frac{R}{R+h}\right)$

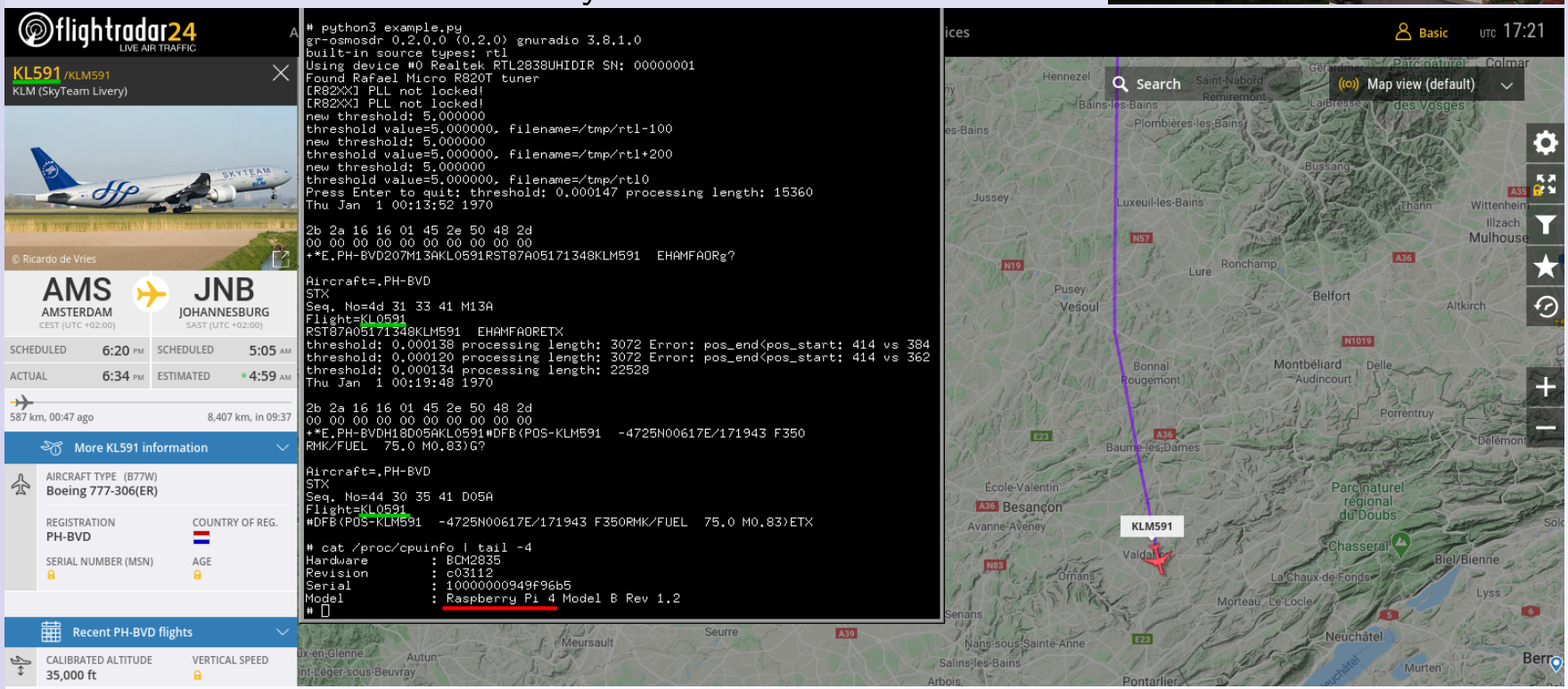
$\Rightarrow$ if $h = 10$ km: $d \simeq 350$ km (Paris !)



Conclusion

- gr-acars ported to GNU Radio 3.8
- opportunity to add clock synchronization for better decoding capability ...
- ... and switching from FFTW to GNU Radio FFT class allowing for multiple blocks to run in parallel.
- Lighter transmission detection using standard deviation analysis, avoiding continuous FFTs ...
- ... thanks to narrowband analysis of each ACARS channel.





Embedded: gr-acars+GNU Radio on RaspberryPi4 (Buildroot & RTL-SDR)=7.5 W

gr-acars source code at `https://sourceforge.net/projects/gr-acars/` with the clock correcting algorithm in the 3.8ng directory