Noise RADAR implementation using software defined radio hardware

J.-M Friedt[1], W. Feng[2]

[1] FEMTO-ST/Time & Frequency department, Besançon, France
[2] National Laboratory of Radar Signal Processing, Xidian University, Xi'an, China

jmfriedt@femto-st.fr

Slides at `http://jmfriedt.free.fr/sdra_radar.pdf`

June 27, 2020

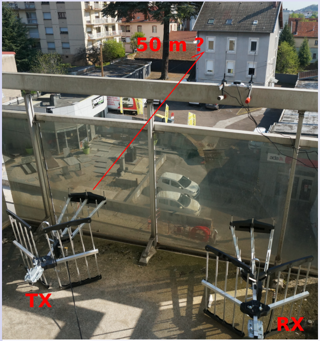
Why a SDR-based RADAR ?

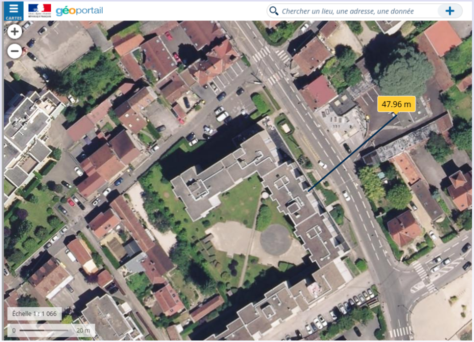
Can we range the house opposite to our balcony using available hardware, namely

- two DVB-T antennas,
- PlutoSDR + Ettus Research B210
- splitter and attenuator

} ⇒ UHF noise, frequency swept, RADAR[1]

[1]RAdio Detection And Ranging





View from the balcony

Aerial map of the area: balcony to house=48 m ?

French Geographic Institute (`geoportail`) claims 48 m: is that correct ?

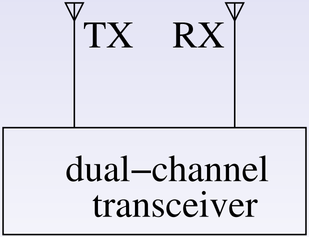
RADAR design – general principles

Emitter and receiver on same board:

- range limited by isolation [1] between TX and RX

$$\frac{P_R}{P_E} = \frac{G^2}{(4\pi)^3} \cdot \frac{\sigma\lambda^2}{d^4}: \; P_R \ll P_E \text{ overwhelmed by direct coupling}$$

- AD936x RF frontend uses different TX and RX LO [2] $\Rightarrow$ carrier frequency drift (might be possible with LimeSDR or debug mode)



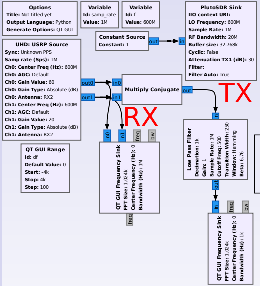
[1] 60–70 dB at best, `https://ez.analog.com/wide-band-rf-transceivers/design-support/f/q-a/114155/ad936n-rx-tx-isolation`

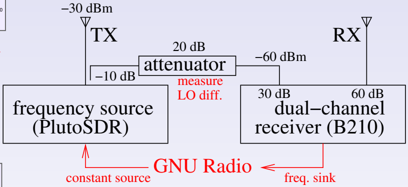
[2] `https://wiki.analog.com/resources/eval/user-guides/ad-fmcomms5-ebz/multi-chip-sync`: "Unfortunately, the AD9361 does not include internal RF synchronization. The ability to synchronize the internal RF local oscillators is not available without some external assistance."

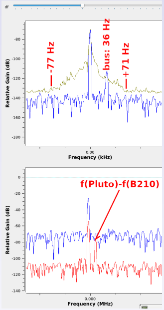
RADAR design – general principles

TX from PlutoSDR and RX with B210

- sampling the PlutoSDR TX on one RX channel and antenna on second RX channel get rid of B210 LO impact
- *but* RADAR range resolution: $\Delta R = \frac{c}{2B}$, B measurement bandwidth
- CW RADAR only detects motion and no range resolution ($B \simeq 0$)







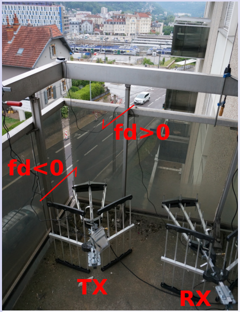
$$\underbrace{\exp(j(\omega_{Pluto} - \omega_{B210})t)^*}_{\text{reference channel}} \cdot \underbrace{\exp(j(\omega_{Pluto} - \omega_{B210} + \omega_{Doppler})t)}_{\text{RX measurement channel}} = \exp(j\omega_{Doppler}t)$$

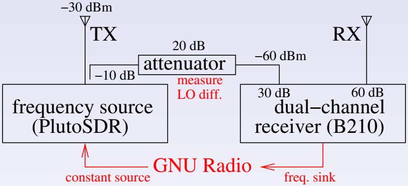
$f_{Doppler} = 2 f_{LO} \times \frac{v}{c}$: at 15 m/s=54 km/h, $f_{Doppler}$ = 60 Hz @ 600 MHz

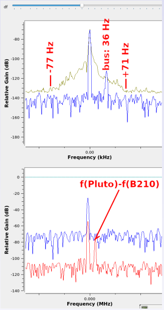
RADAR design – general principles

TX from PlutoSDR and RX with B210

- sampling the PlutoSDR TX on one RX channel and antenna on second RX channel get rid of B210 LO impact
- *but* RADAR range resolution: $\Delta R = \frac{c}{2B}$, B measurement bandwidth
- CW RADAR only detects motion and no range resolution ($B \simeq 0$)







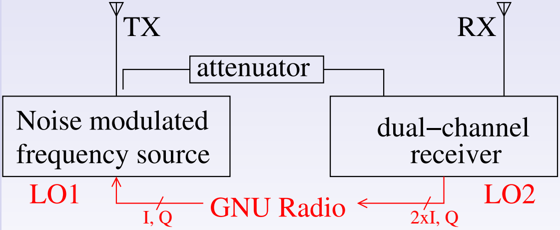
$$\underbrace{\exp(j(\omega_{Pluto} - \omega_{B210})t)^*}_{\text{reference channel}} \cdot \underbrace{\exp(j(\omega_{Pluto} - \omega_{B210} + \omega_{Doppler})t)}_{\text{RX measurement channel}} = \exp(j\omega_{Doppler}t)$$

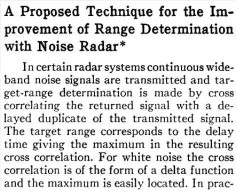
$f_{Doppler} = 2f_{LO} \times \frac{v}{c}$: at 9 m/s=32 km/h, $f_{Doppler} = 36$ Hz (bus speed)

RADAR design – general principles

Spectrum spreading the emission

- could be chirp (LO frequency sweep),
- OFDM [1] (sum of well defined adjacent frequency components),
- here noise RADAR: pseudo random phase fluctuation while keeping amplitude constant (see GPS/CDMA for PRN phase spread spectrum)





A Proposed Technique for the Improvement of Range Determination with Noise Radar*

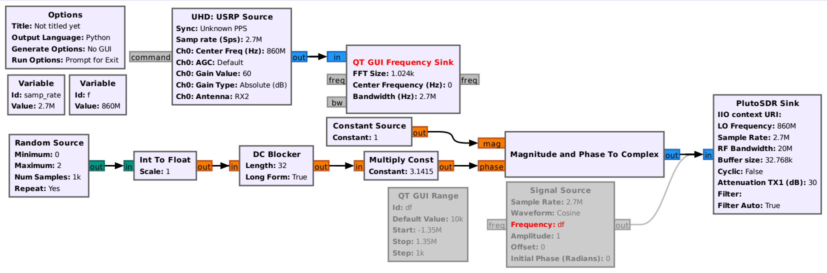
In certain radar systems continuous wide-band noise signals are transmitted and target-range determination is made by cross correlating the returned signal with a delayed duplicate of the transmitted signal. The target range corresponds to the delay time giving the maximum in the resulting cross correlation. For white noise the cross correlation is of the form of a delta function and the maximum is easily located. In prac-

R. Bourret, *A proposed technique for the improvement of range determination with noise radar*, Proc IRE **45** (12) 1744–1744 (1957)

[1] M. Braun, *OFDM Radar Algorithms in Mobile Communication Networks* (2015) at `publikationen.bibliothek.kit.edu/1000038892/2987095`

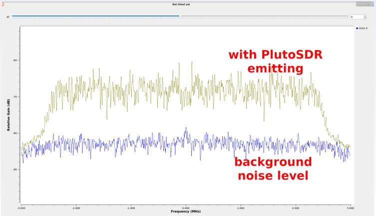
RADAR design – general principles

Fixed magnitude (power) and pseudo-random definition of phase in the `Magnitude and Phase to Complex` block



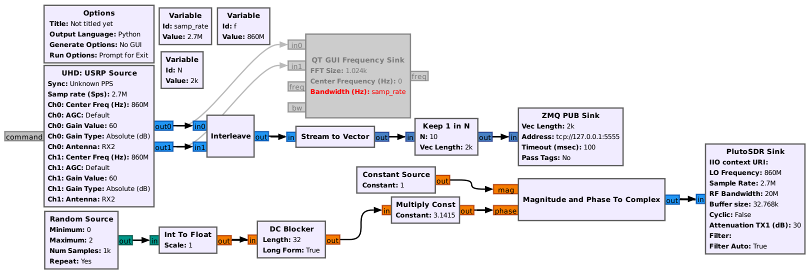
↑ Flowchart for checking the spectrum spreading capability of the PlutoSDR and simultaneous data collection by the B210 (⇒ max `samp_rate` with no sample loss)

Spectrum collected by the B210 ⟶



RADAR design – GNU Radio implementation

B limited by USB to about 2.7 MHz=56 m range resolution: **sweep LO** to concatenate multiple spectra and extend B following multiple sequential sweeps



- Collect time series → program next LO frequency → repeat until full B has been swept
- No known way of synchronizing data collection with LO sweep in GNU Radio Companion

⇒ benefit from external data collection and processing program (GNU Octave): streaming using ZeroMQ from GNU Radio to Octave

RADAR design – GNU Radio implementation

Python generated by GNU Radio Companion

```
#!/usr/bin/env python3
class t(gr.top_block)
[...]
    def set_f(self, f):
        self.f = f
        self.iio_pluto_sink_0.set_params(self→
            ↪.f, self.samp_rate, →
            ↪20000000, 30.0, '', True)
        self.uhd_usrp_source_0.→
            ↪set_center_freq(self.f, 0)
        self.uhd_usrp_source_0.→
            ↪set_center_freq(self.f, 1)

def main(top_block_cls=t, options=None):
    tb = top_block_cls()
    def sig_handler(sig=None, frame=None):
        tb.stop()
        tb.wait()
        sys.exit(0)

    signal.signal(signal.SIGINT, sig_handler)
    signal.signal(signal.SIGTERM, sig_handler→
        ↪)

    tb.start()
```

Callback functions defining PlutoSDR and B210 LO frequency `f`

GNU Octave & ZeroMQ toolbox

```
total_length=140000; % [samples/measurement]
Nfreq=50;
pkg load zeromq
for frequency=1:Nfreq
 sock1 = zmq_socket(ZMQ_SUB);
 zmq_connect   (sock1,"tcp://127.0.0.1:5555")→
       ↪;
 zmq_setsockopt(sock1, ZMQ_SUBSCRIBE, "");
 recv=zmq_recv(sock1, total_length*8*2, 0);
 value=typecast(recv,"single complex");
 x(:,frequency)=value(1:2:length(value));
 m(:,frequency)=value(2:2:length(value));
 zmq_close (sock1);
 % send(sck,'+'); % TCP command: next slide
end
```

- `zmq_recv`: number of bytes (*8 complex, *2 interleaved channels)
- `typecast`: char → complex float
- socket-connect-opt-close = 130 us

RADAR design – GNU Radio implementation

Adding TCP server to Python:

```
import threading
import socket
    def jmf_server(self):
        sock=socket.socket(socket.AF_INET, →
              ↪socket.SOCK_STREAM)
        sock.setsockopt(socket.SOL_SOCKET, →
              ↪socket.SO_REUSEADDR, 1)
        sock.bind(('localhost', 5556))
        sock.listen(1)
        conn, addr = sock.accept()
        with conn:
            while True:
                data=conn.recv(1)
                if '+' in str(data):
                    self.f=self.f+1000000
                if '0' in str(data):
                    self.f=int(700e6)
                if 'q' in str(data):
                    sock.shutdown(socket.→
                          ↪SHUT_RDWR)
                    sock.close()
                self.set_f(self.f)

def main(top_block_cls=t, options=None):
[...]
    tb.start()
    threading.Thread(target=tb.jmf_server).→
           ↪start()
    try:
        input('Press Enter to quit: ')
```

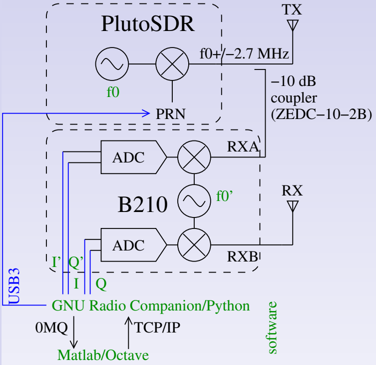
GNU/Octave:

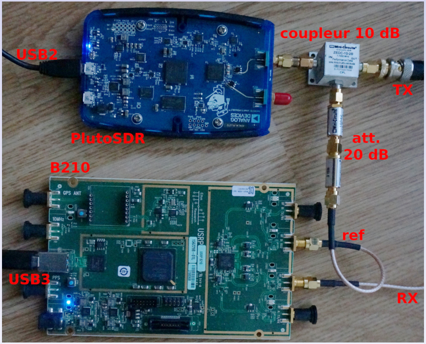
```
pkg load sockets
sck=socket(AF_INET, SOCK_STREAM, 0);
s_info=struct("addr","127.0.0.1","port",5556);
connect(sck,s_info);
send(sck,'0');
pause(1.0)
for frequency=1:Nfreq
 sock1 = zmq_socket(ZMQ_SUB);
 [...]
 zmq_close (sock1);
 send(sck,'+'); % wait PlutoSDR to stabilize
 pause(1.0)     % will be addressed later !
end
send(sck,'q');
```

- TCP client: `sockets` toolbox

Modify Python script generated by GNU Radio Companion by adding TCP server receiving commands from Octave (LO reset, LO increment)

RADAR design – GNU Radio implementation

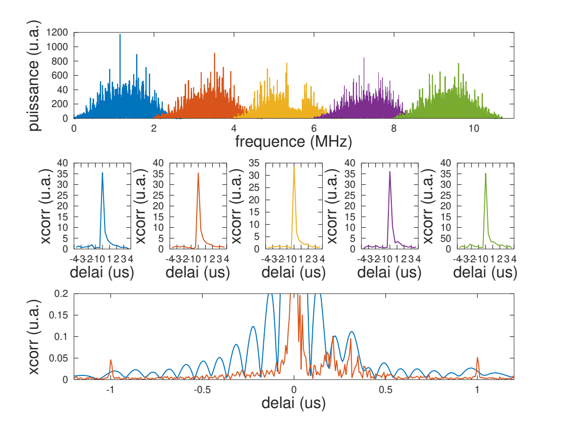




- ⇒ functional frequency sweep ...
- ... but need to **wait** for unknown delay between LO programming and settling ⇒ 1 s between programming the PlutoSDR [1] and recording
- if LO not settled, no pseudo-random modulation ⇒ single carrier carrying all energy with no range resolution (threshold test)

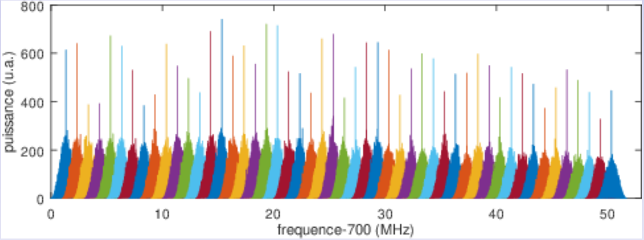
[1]R. Bell, *Maximum Supported Hopping Rate Measurements using the Universal Software Radio Peripheral Software Defined Radio*, Proc. GRCon 2016

Results: spectra concatenation

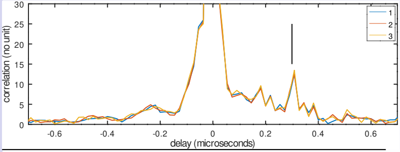


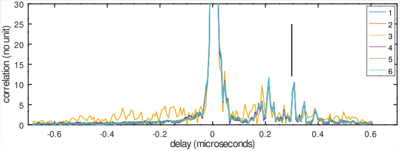
- Each individual (complex) correlation exhibits more energy in the future (right from 0 delay) than in the past
- Concatenation of spectra for improved range resolution:
- $B = 50$ MHz $\Rightarrow \Delta R = 3$ m but **1 min sweep**
- $B = 150$ MHz $\Rightarrow \Delta R = 1$ m but 3 min sweep + limited antenna bandwidth

Results: broadband correlation



- Collect (complex) I, Q samples from both channels in each (2.7 MHz-wide) frequency band
- Fourier transform and concatenate spectra (=oversampling)
- product of reference spectrum with complex conjugate of measurement spectrum [2]
- inverse Fourier transform for time-domain correlation ↓ (line=50 m)

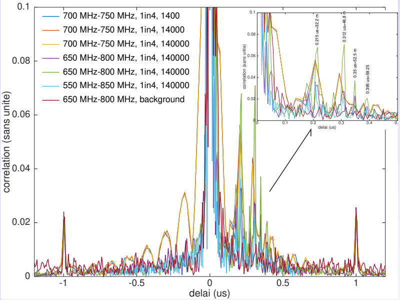


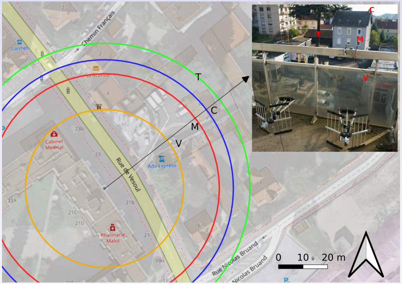


[2] remember: FFT(xcorr(x,y))=FFT(x)$\cdot$ FFT*(y)

Analysis: echo interpretation

- Improved resolution with increasing bandwidth, restricted by antenna
- Emitted power: -30 dBm
- Interpretation: 32 m is a car **V**, 47 m is the house **M**, 52 m is the rooftop with a chimney **C**, 59 m is a tree trunk **T**

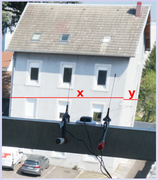


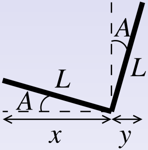


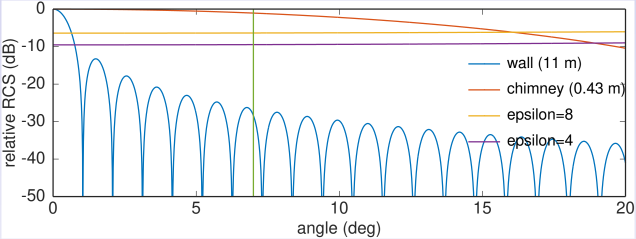
- Symmetric correlation peaks at ± 1 μs: artefacts introduced by sweeping the spectra with 1 MHz steps [3]

[3] periodic stucture every 1 MHz becomes periodic structure every 1 μs

Analysis: RADAR cross section of targets [4]



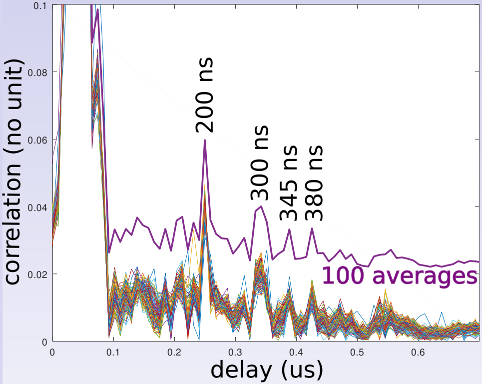




- $\bigtriangledown$ A flat wall reflects the incoming wave away from the receiver (principle of stealth planes/ships)
- $\uparrow$ green vertical line = angle between illuminating beam & receiver: 7°
- *sinc*() transfer function: strong reflection from the λ-wide chimney
- Low permittivity concrete reflects only a fraction of incoming wave
- Received signal/thermal noise floor=10–20 dB SNR, in agreement with experiment, with **similar levels** returned by wall and chimney

[4] Analysis driven by F. Daout, IUT Ville d'Avray, France

Sweep rate improvement



- **All** parameters of the AD936x are updated simultaneously by `gr-iio`: waste of time when only LO settling time is needed
- Update the GNU Radio 3.8 port of `gr-iio`[5] with single parameter update[6]

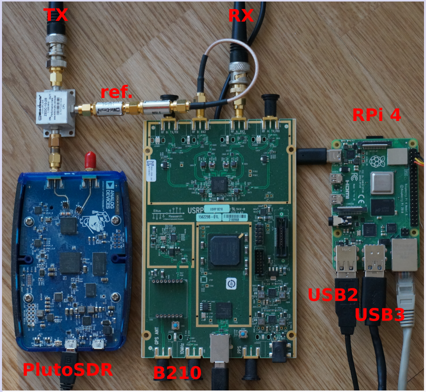
- now sweep rate only limited by data collection time: $1.4 \cdot 10^5$ samples @ 2.7 MS/s: 52 ms/freq. step=2.6 s/50 MHz sweep
- Becomes fast enough to allow time resolved measurements or averaging

[6] https://ez.analog.com/adieducation/university-program/f/q-a/91477/adalm-pluto-how-to-change-settings-with-quick-settling/199287

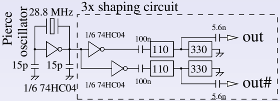
[6] https://github.com/analogdevicesinc/gr-iio/tree/single-param

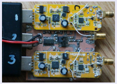
Conclusion & perspective

- Software Defined Radio for active RADAR prototyping
- **separate functions** and delegate to most efficient framework (GNU Radio, Python, Octave)



- Replace high grade B210 with synchronized DVB-T receivers (software update[7] to stream through 0-MQ continuously even when changing LO): 2.4 MS/s bandwidth



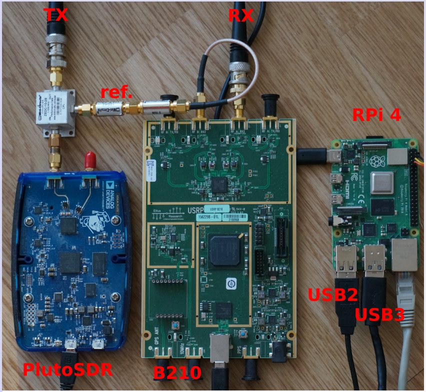


- Replace dedicated laptop with USB-3 fitted Raspberry Pi4 single board computer: GNU Radio, UHD, gr-iio and 0-MQ all ported to Buildroot

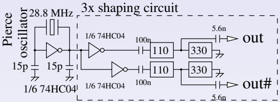
[7] S. Scholl, DC9ST *Introduction and Experiments on Transmitter Localization with TDOA*, Software Defined Radio Academy (2017) at `https://www.youtube.com/watch?v=Km4TU17b05s` and detailed description at `http://www.panoradio-sdr.de/tdoa-transmitter-localization-with-rtl-sdrs/`

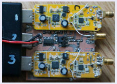
Conclusion & perspective

- Software Defined Radio for active RADAR prototyping
- **separate functions** and delegate to most efficient framework (GNU Radio, Python, Octave)



- Replace high grade B210 with synchronized DVB-T receivers (software update to stream through 0-MQ continuously even when changing LO): 2.4 MS/s bandwidth





- Replace dedicated laptop with USB-3 fitted Raspberry Pi4 single board computer: GNU Radio, UHD, gr-iio and 0-MQ all ported to Buildroot[7]

Archive of control software and datasets: `github.com/jmfriedt/active_radar`

Slides at `http://jmfriedt.free.fr/sdra_radar.pdf`

[7] `https://github.com/oscimp/PlutoSDR/tree/for_next`: BR2_EXTERNAL addition to mainline Buildroot (including Raspberry Pi4) by G. Goavec-Merou

Backup: emitted power estimate

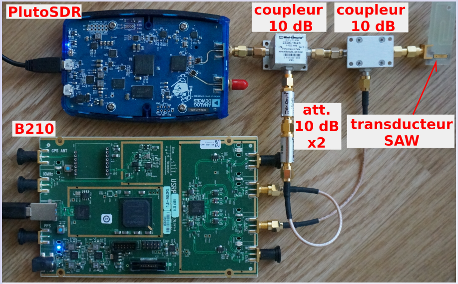
Emission in the DVB-T frequency band (470–790 MHz): did we prevent neighbours from watching TV ?

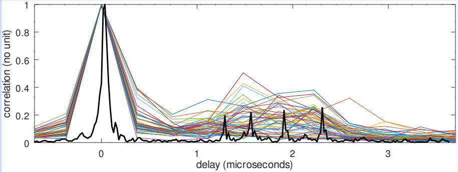
- 25 kW DVB-T emitter located in Montfaucon, 4850 m from the emission site
- Friss (Free Space Propagation Loss) [8]: 74 dBm - 103 dB at 700 MHz = **-29 dBm** received power
- we emit -30 dBm, not accounting for spreading power in a 2 MHz bandwidth.
- target is at 50 m so received power is -30-63=**-93 dBm**

[8] $FSPL = 20\log_{10}(d) + 20\log_{10}(f) - 147.55$ dB with f in Hz and d in m

Backup: SAW reflective delay line as reference echo generator

Means to reproducibly generate echoes in the 100 to 2000 ns delay (15–300 m targets) during development stage ?





- Coaxial cable: velocity 66% from velocity in vacuum, so 20 to 400 m long cable. Neither prectical nor compact.
- Acoustic waves propagate 10^5 times slower than electromagnetic waves: dimensions shrink by the same factor (200 to 4000 μm=0.2 to 4 mm acoustic path)
- piezoelectric substrate (quartz, lithium niobate or lithium tantalate) readily convert electromagnetic to acoustic wave (principle of SAW filters)
- operate: 100 to 2500 MHz range

Reflective SAW delay line as compact ($<$5 mm long) echo reference: here 1.5 to 2.5 μs