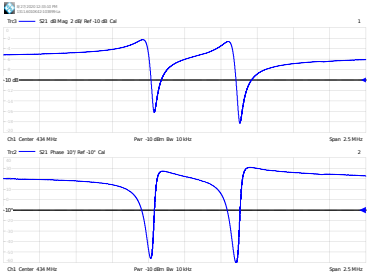
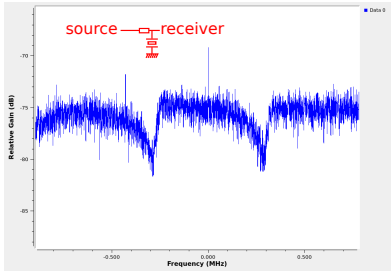
Porting GNU Radio to Buildroot: application to an embedded digital network analyzer

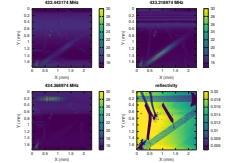


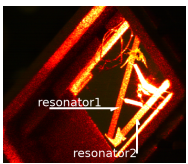
G. Goavec-Merou, J.-M Friedt

FEMTO-ST Time & Frequency,
Besançon, France

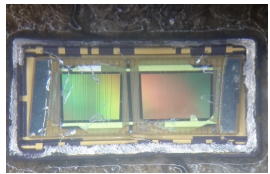


References at `http://jmfriedt.free.fr`





November 9, 2022



Outline

Embedded digital electronics

1. 8-bit microcontroller
2. 32-bit microcontroller (bare metal)
3. 32-bit microcontroller (libraries)
4. 32-bit microcontroller (executive environment)
5. 32/64-bit microcontroller (operating system)
6. 32/64-bit microcontroller (kernel module)
7. 32/64-bit microcontroller (kernel-userspace communication)
8. 32/64-bit microcontroller/FPGA (Redpitaya co-design)
9. 32/64-bit microcontroller/FPGA radiofrequency signal processing

Bottom-up development

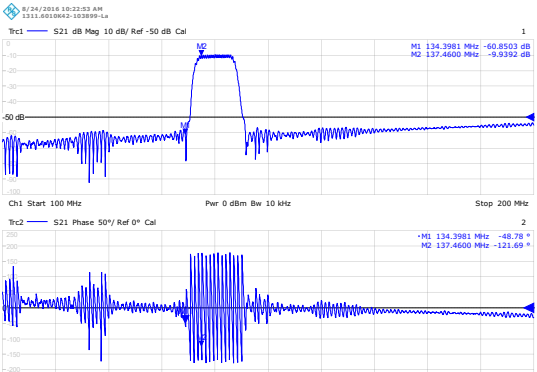
Project

1. Raspberry Pi 4 single board computer (4×1.5 GHz CPUs)
2. Python/GNU Radio userspace applications
3. Communication with personal computer (0MQ)
4. Web-controlled instrument (REMI)
5. Characterization of spectral properties of radiofrequency device
   - frequency sweep network analyzer
   - broadband noise source + FFT
   - pulsed excitation + FFT
6. ringdown measurement of resonant system: Q & f_r

Top-down development

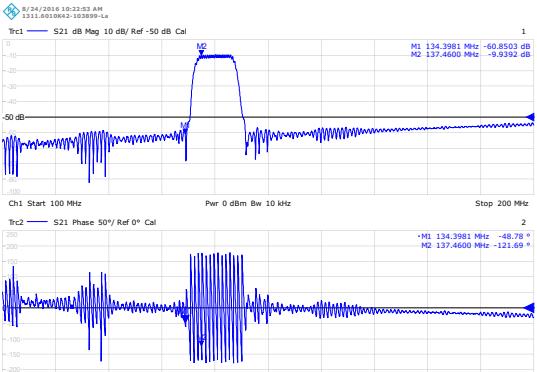
Acoustic transducer

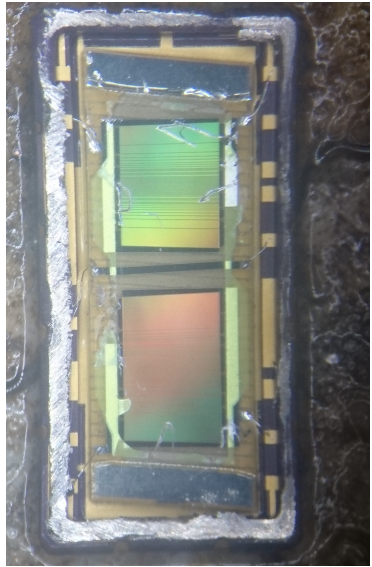
- Electromagnetic wavelength: $\lambda = 300/f_{MHz}$
- At 1 GHz: 30 cm wavelength $\Rightarrow$ signal processing (e.g. cavities, transformer, filters, waveguides) will be $\simeq \lambda$ wavelength
- Acoustic waves propagate $\in$ [500..10000] m/s: at 3000 m/s, 10^5 slower
- Convert between electromagnetic and acoustic wave using piezoelectric substrates: 30 cm become 3 μm wavelength
- 40-50 filters/phone, 1.4 billion phones/year $\Rightarrow$ >50 billion acoustic filters/year [1]



Acoustic transducer

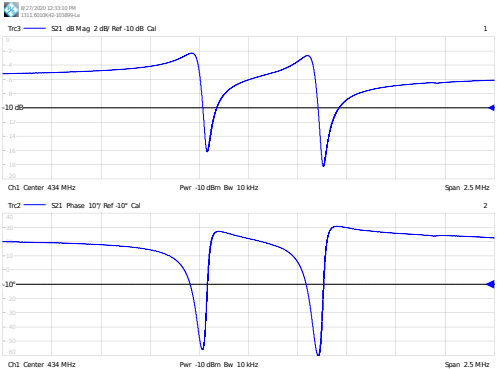
- Electromagnetic wavelength: $\lambda = 300/f_{MHz}$
- At 1 GHz: 30 cm wavelength $\Rightarrow$ signal processing (e.g. cavities, transformer, filters, waveguides) will be $\simeq \lambda$ wavelength
- Acoustic waves propagate $\in$ [500..10000] m/s: at 3000 m/s, 10^5 slower
- Convert between electromagnetic and acoustic wave using piezoelectric substrates: 30 cm become 3 μm wavelength
- 40-50 filters/phone, 1.4 billion phones/year $\Rightarrow$ >50 billion acoustic filters/year [1]



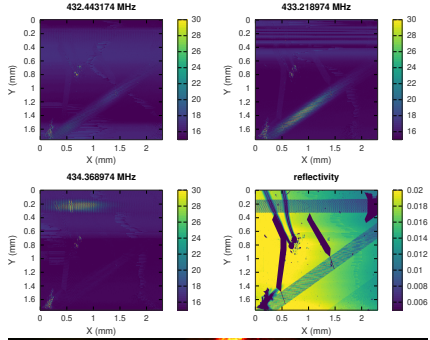


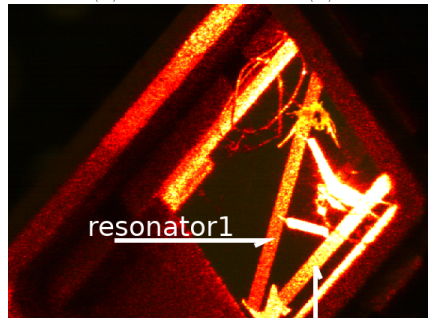
Acoustic resonator

- Dual surface acoustic wave (SAW) resonator operating in the 433.05–434.79 MHz ISM band
- Electromagnetic wave: 70 cm wavelength $\rightarrow$ acoustic wave: 7 μm
- 2-port device used either in reflection or transmission
- $Q \simeq 10^4$ @ 434 MHz



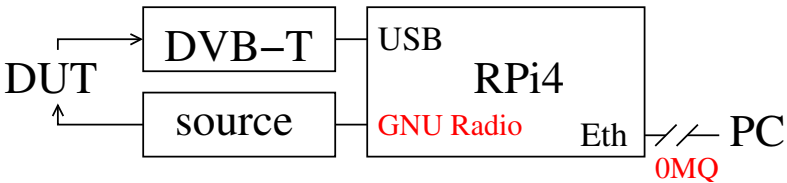
SENSeOR SEAS10 sensor





Principle

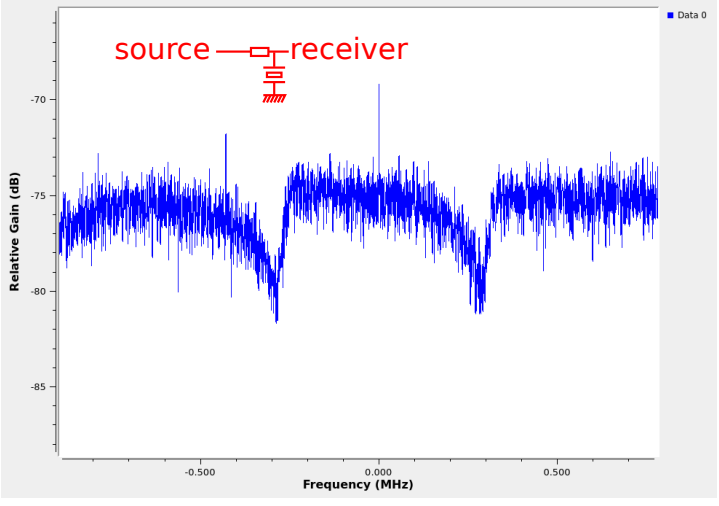
Objective: characterize frequency response of a radiofrequency device under test (DUT)



- Instrument control: GNU/Linux running on Raspberry Pi 4 (RPi4)
- Source: broadband source (noise/pulse) ...
- ... generated by a GPIO of the Raspberry Pi connected to its radiofrequency clock
- Acquisition & sampling: R820T2 based DVB-T receiver as general purpose Software Defined Radio (SDR) receiver
- Processing: GNU/Radio communicating with PC

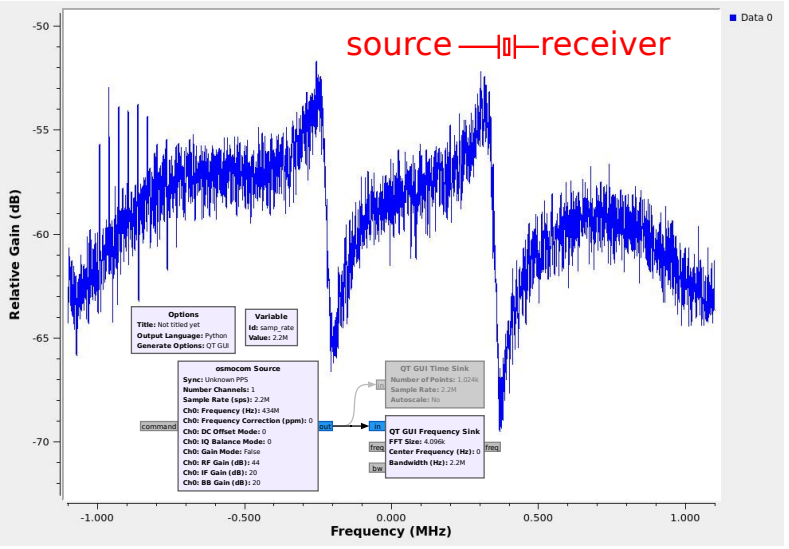
Noise source: reflection mode

- Broadband noise source (reverse-polarized Zener diode + amplifiers)
- Sample 2.2 MHz-wide bandwidth
- Transpose from 434 MHz to baseband

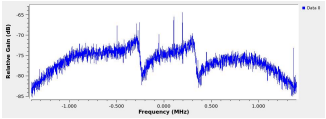


Noise source: transmission mode

Rather than voltage divider bridge to ground, resonator pair in series with signal source

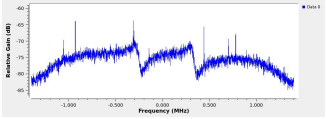


Pulsed source

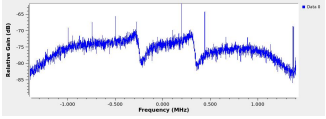


0.5 ns

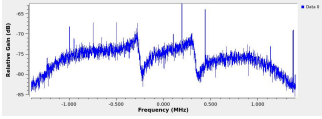
Transmission spectra as a function of pulse duration

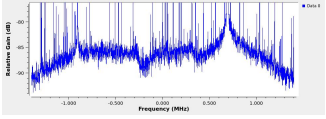


0.625 ns

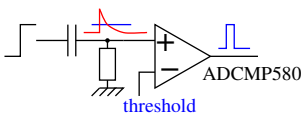


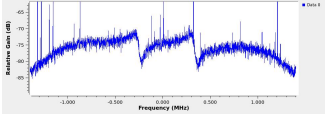
1.0 ns





1.25 ns

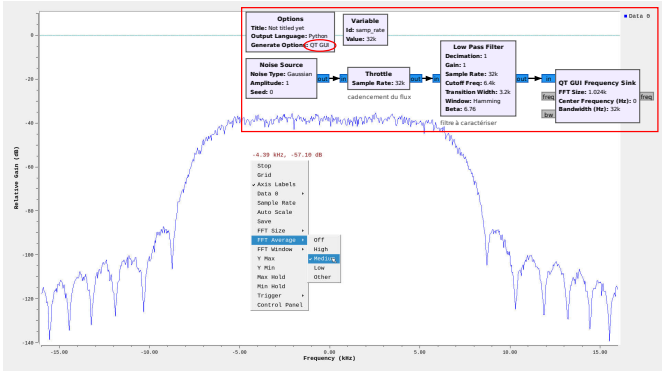




2.22 ns

GNU Radio running on the Raspberry Pi4

Characterization of a band pass filter using micro-HDMI output of RPi4

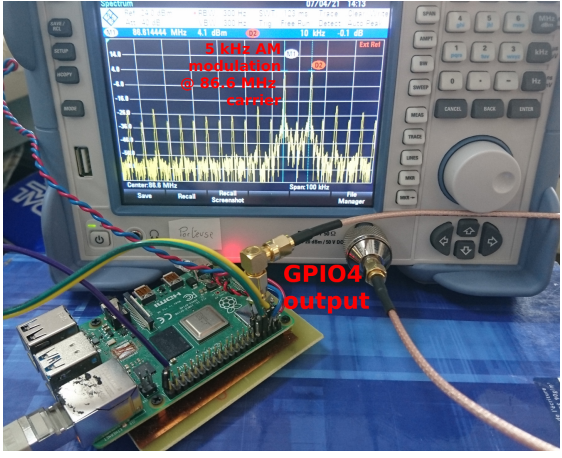



```
#!/usr/bin/env python3
from PyQt5 import Qt
from gnuradio import qtgui
from gnuradio.filter import firdes

class rpi(gr.top_block, Qt.QWidget):

 def __init__(self):
  gr.top_block.__init__(self, "Title")
  Qt.QWidget.__init__(self)

  self._qtgui_freq_sink_x_0_win = sip.wrapinstance(self.→
        ↪qtgui_freq_sink_x_0.pyqwidget(), Qt.QWidget)
  self.low_pass_filter_0 = filter.fir_filter_ccf(1, firdes.→
        ↪low_pass( 1, samp_rate, samp_rate/5, samp_rate/10, →
        ↪firdes.WIN_HAMMING, 6.76))
  self.blocks_throttle_0 = blocks.throttle(gr.sizeof_gr_complex→
        ↪*1, samp_rate,True)
  self.analog_noise_source_x_0 = analog.noise_source_c(analog.→
        ↪GR_GAUSSIAN, 1, 0)
  ##################################################
  # Connections
  ##################################################
  self.connect((self.analog_noise_source_x_0, 0), (self.→
        ↪blocks_throttle_0, 0))
  self.connect((self.blocks_throttle_0, 0), (self.→
        ↪low_pass_filter_0, 0))
  self.connect((self.low_pass_filter_0, 0), (self.→
        ↪qtgui_freq_sink_x_0, 0))
[...]
```

Using a GPIO as radiofrequency source

- PiFM: route an internal radiofrequency phase locked loop to a GPIO pin
- Radiofrequency source, generalized to a generic IQ stream by F5OEO...
- ... now as a GNU Radio sink block[2]

⇒ **flexible** signal source for generating a single tone, a frequency swept tone or a broadband noise (pseudo-random generator)

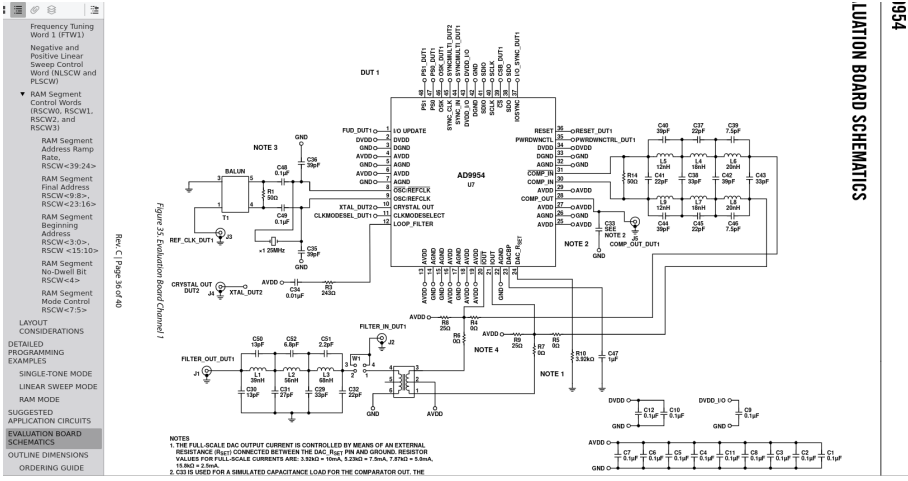


[2] `https://github.com/jmfriedt/gr-rpitx` and J.-M Friedt, É. Courjaud, *gr-rpitx: GNU Radio compatible general purpose SDR emitter using the Raspberry Pi(4) internal phase locked loop*, European GNU Radio Days (2021)

Designing a dedicated radiofrequency source

Design (schematic) and realization (board) of a radiofrequency signal source for driving the DUT

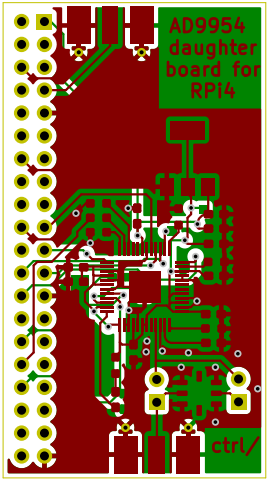
- daughterboard to the Raspberry Pi 4 (analyze the signals on the RPi4 pins)
- radiofrequency circuit: grounding and decoupling considerations
- selection of a signal source: PLL v.s DDS (Analog Devices) v.s clock generator (Skyworks)

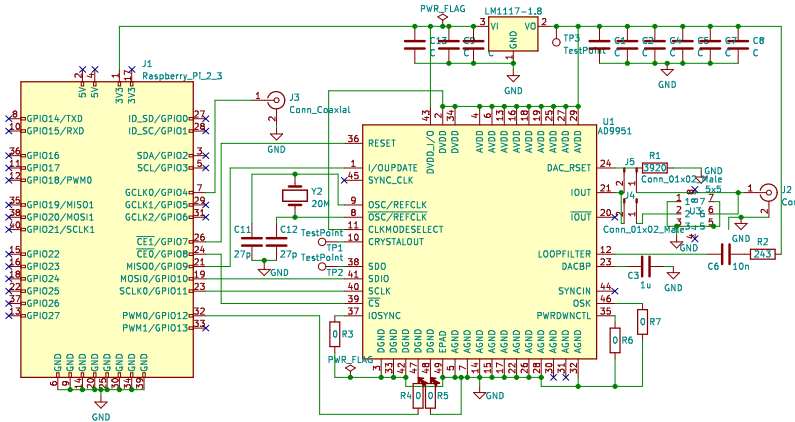


Designing a dedicated radiofrequency source

Design (schematic) and realization (board) of a radiofrequency signal source for driving the DUT

- daughterboard to the Raspberry Pi 4 (analyze the signals on the RPi4 pins)
- radiofrequency circuit: grounding and decoupling considerations
- selection of a signal source: PLL v.s DDS (Analog Devices) v.s clock generator (Skyworks)

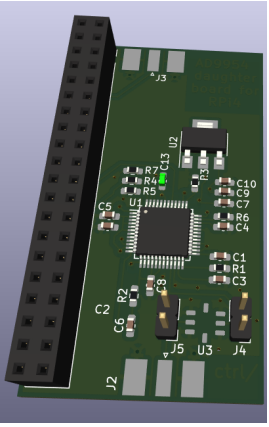


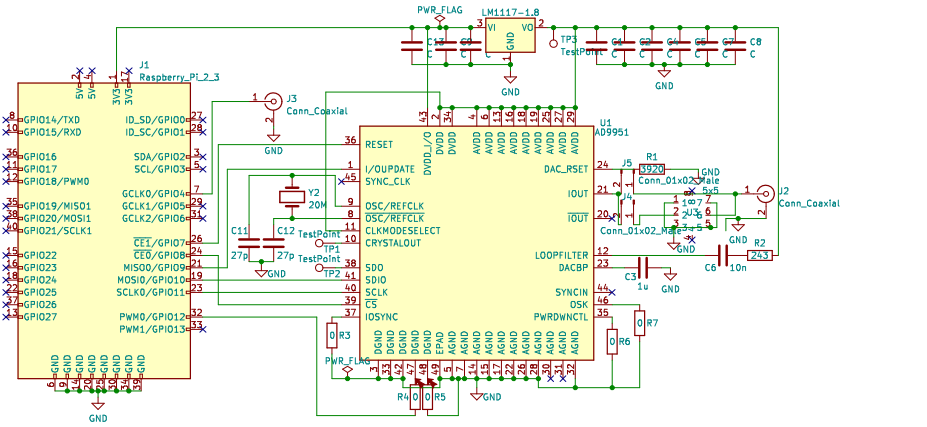


Designing a dedicated radiofrequency source

Design (schematic) and realization (board) of a radiofrequency signal source for driving the DUT

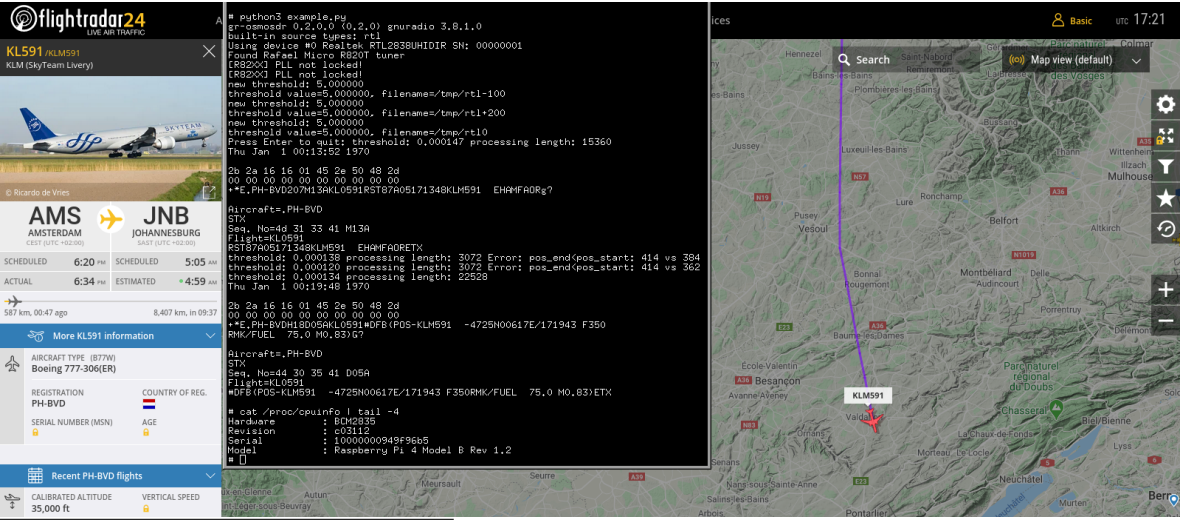
- daughterboard to the Raspberry Pi 4 (analyze the signals on the RPi4 pins)
- radiofrequency circuit: grounding and decoupling considerations
- selection of a signal source: PLL v.s DDS (Analog Devices) v.s clock generator (Skyworks)





Beyond RF device characterization

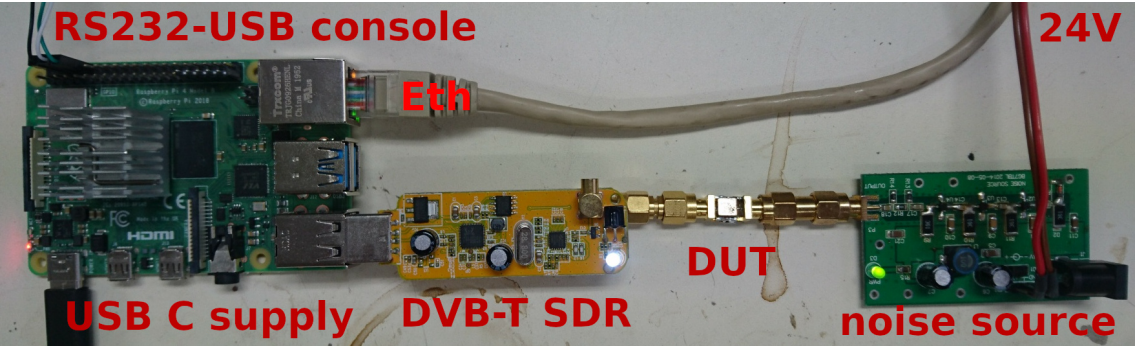
Autonomous ACARS[3] (plane-to-ground communication) receiver running on RPi4



[3] https://sourceforge.net/projects/gr-acars/

Useful tools

- Buildroot for developing on the RPi4 or `qemu`
- GNU Radio running on RPi4
- ZeroMQ or UDP socket to stream from RPi4 to PC
- Python TCP server on RPi4 to receive commands from PC (telnet or Python client)
- Include TCP server in GNU Radio (Python Module)
- Launch TCP server from GNU Radio (Python Snippet) with access to the callback functions
- Display on PC stacked spectra with broader bandwidth than the DVB-T 2.8 MHz bandwidth

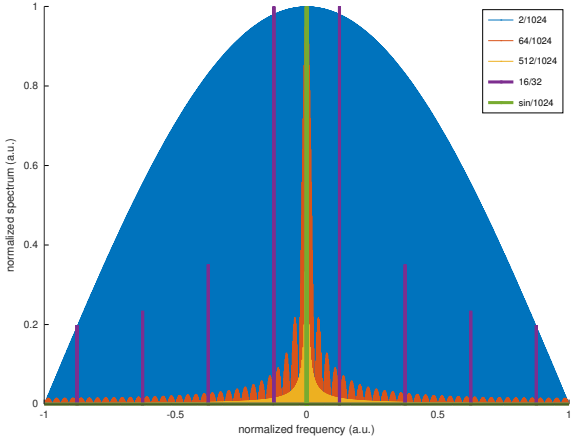


Preliminary investigations

Why a pulse and what properties are wanted ?

1. What is the spectrum of a Dirac function ?
2. What is the spectrum of a square function with period T ?
3. What is the spectrum of a pulse with rise time τ and duration T ?
4. What is the spectrum of a pulse with duration T gating a sine wave with frequency f, $f \gg 1/T$?

```
x=zeros(1024,1); x(1:2)=1;
x=[x ;x; ;x; ;x; x]; x=x-mean(x);
f=linspace(-1,1,length(x));
y=fftshift(abs(fft(x))); plot(f,y/max(y))

x=zeros(1024,1); x(1:64)=1;
x=[x ;x; ;x; ;x; x]; x=x-mean(x);
y=fftshift(abs(fft(x))); plot(f,y/max(y))

x=zeros(1024,1);x(1:512)=1;
x=[x ;x; ;x; ;x; x]; x=x-mean(x);
y=fftshift(abs(fft(x))); plot(f,y/max(y))

x=zeros(1024,1);
for k=1:1024
 if mod(k,16)<8 x(k)=1;end
end
x=[x ;x; ;x; ;x; x]; x=x-mean(x);
y=fftshift(abs(fft(x))); plot(f,y/max(y))

x=sin([0:1023]/1024*2*pi)';
x=[x ;x; ;x; ;x; x];
y=fftshift(abs(fft(x)));
```



Conclusion & perspectives

IA, ROBOTIQUE & SCIENCE — SDR

ANALYSE ET RÉALISATION D'UN
RADAR À BRUIT PAR
RADIO LOGICIELLE

J.-M FRIEDT
[FEMTO-ST, département temps-fréquence, Besançon, France]

W. FENG
[Xidian University, National Laboratory of Radar Signal Processing, Xi'an, Chine]

MOTS-CLÉS : RADAR, GNU/OCTAVE, TRAITEMENT DU SIGNAL

Pour paraphraser l'introduction d'une vidéo de chanteurs de rap français [1], « on a 3 minutes, un appart' rempli de bordel, une vieille émission pour les gosses dans l'ordi, on peut sûrement y aller au bluff ». Dans notre cas, nous n'avons pas 3 minutes, mais 5 semaines de confinement, on a bien un appartement rempli de bordel, et plutôt que l'émission de télévision, nous avons des antennes et des circuits de radio logicielle. Pouvons-nous mesurer la distance de la maison en face dudit appartement en réalisant un RADAR (*RAdio Detection And Ranging*) actif avec les moyens du bord (figure 1) ?

Le traitement du signal, domaine abstrait par excellence, devient très amusant lors de sa mise en pratique. Nous nous proposons de démontrer expérimentalement un RADAR à bruit mis en œuvre par radio logicielle.

NOTE
Cet article est le premier d'une série de trois articles sur le sujet.

- Opportunity to deploy an embedded system for a practical application
- Opportunity to learn how to properly use GNU/Linux targeted to an embedded environment (Buildroot – **never compile on the target system**)
- Reproducible experiment: <50 euros (RPi4+DVB-T)
- Radiofrequency signal processing ...
- for characterizing a (RF) microsystem.
- RADAR range resolution: $\Delta R = c/(2B)$ ...
- ... can be reached with many sources other than a pulse generator to achieve large B.

References

- C. Simmonds, *Mastering Embedded Linux Programming*, Packt (2015)
- K. Yaghmour *& al.*, *Building Embedded Linux Systems: Concepts, Techniques, Tricks, and Traps*, O'Reilly (2008)
- P. Ficheux & É. Bénard, *Linux embarqué*, Eyrolles (2012)
- P. Ficheux, *Linux embarqué – Mise en place et développement*, Eyrolles (2017)
- P. Ficheux, *Introduction à Buildroot*, GNU/Linux Magazine France (Avril 2010)
- M. Corbin, *Buildroot for RISC-V*, FOSDEM 2019 at `https://archive.fosdem.org/2019/schedule/event/riscvbuildroot/`
- Buildroot Git repository: `https://git.busybox.net/buildroot/`
- Experiment ... even without hardware: `qemu` emulating a Raspberry Pi {3,4}/64 bits


```
processor       : 2
BogoMIPS        : 125.00
Features        : fp asimd evtstrm aes pmull sha1 sha2 crc32 cpuid
CPU implementer : 0x41
CPU architecture: 8
CPU variant     : 0x0
CPU part        : 0xd03
CPU revision    : 4

processor       : 3
BogoMIPS        : 125.00
Features        : fp asimd evtstrm aes pmull sha1 sha2 crc32 cpuid
CPU implementer : 0x41
CPU architecture: 8
CPU variant     : 0x0
CPU part        : 0xd03
CPU revision    : 4

Hardware        : BCM2835
Model           : Raspberry Pi 3 Model B
# export QT_QPA_PLATFORM=linuxfb:fb=/dev/fb0
# python3 rpi.py
[   47.274329] random: python3: uninitialized urandom read (24 byte
s read)
Warning: failed to XInitThreads()
[   53.230974] random: crng init done
```