La radio logicielle : une approche flexible pour la réception de signaux

J.-M Friedt

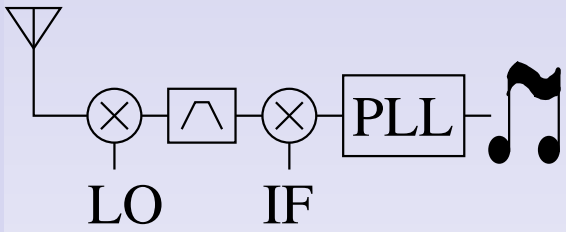
FEMTO-ST Time & Frequency, Besançon, France

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

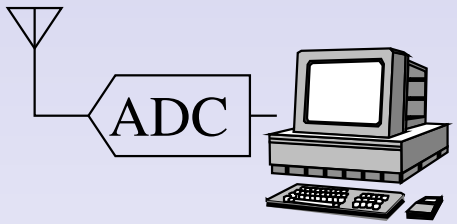
21 mars 2019

Introduction à la radio logicielle

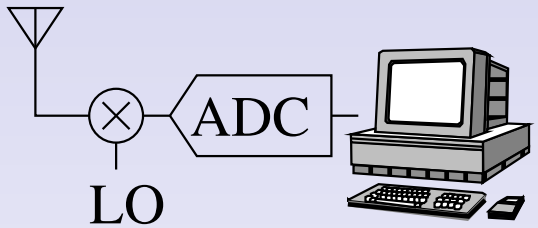
- Approche logicielle, et du traitement numérique du signal v.s analogique
- flexibilité, stabilité, reconfigurabilité [1]



ce qu'on avait



ce qu'on voudrait



ce qu'on a

Flexibilité et mise en œuvre pédagogique : multitude de modes reçus avec du matériel peu coûteux reconfigurable

- bande FM commerciale (réception multicanaux), ADS-B (1090 MHz)
- FM RDS (FDMA, BPSK)
- liaisons spatiales (satellites en orbite basse : bilan de liaison "optimal") analogique & numérique (ISS, LEO)
- POCSAG (décodage de signaux numériques par outils externes)
- GPS (en émission et en réception)
- RADAR (passif) ...

1. D.A. Mindell, *Digital Apollo – Human and Machine in Spaceflight*, MIT Press (2008)

GNU Linux Magazine France :

1. *Leurrage du GPS par radio logicielle*, MISC HS (2019)
2. *Quelques fondements théoriques pour aborder la radio logicielle*, **224** (2019)
3. *Matériel pour la radio logicielle*, **224** (2019)
4. *La peinture sur spectre radiofréquence, et l'effet capture de la modulation en fréquence – ou pourquoi les avions communiquent encore en AM*, **216** (2018)
5. *Quelques éléments de traitement de signaux échantillonnés en temps discret avec GNU Radio*, **221** (2018)
6. *RADAR passif par intercorrélation de signaux acquis par deux récepteurs de télévision numérique terrestre*, **212** pp.36- (2018)
7. *Radio Data System (RDS) – analyse du canal numérique transmis par les stations radio FM commerciales, introduction aux codes correcteurs d'erreur*, **204** (2017)
8. *Exploitation de signaux des satellites GPS reçus par récepteur de télévision numérique terrestre DVB-T*, OpenSilicium **15** (2015)
9. *La réception de signaux venus de l'espace par récepteur de télévision numérique terrestre*, OpenSilicium **13** (Dec 2014/Jan-Fev 2015)
10. *La réception radiofréquence définie par logiciel (Software Defined Radio – SDR)*, **153** 4–33 (2012)

Journaux :

1. W. Feng, J.-M Friedt, G. Cherniak, M. Sato, *Passive bistatic radar using digital video broadcasting–terrestrial receivers as general-purpose software-defined radio receivers*, Rev. Sci. Instrum. **89**, 104701 (2018)
2. J.-M Friedt, C. Eustache, É. Carry, E. Rubiola, *Software defined radio decoding of DCF77 : time and frequency dissemination with a sound card*, Radio Science **53** (1), 48–61 (2018)

Introduction à la radio logicielle

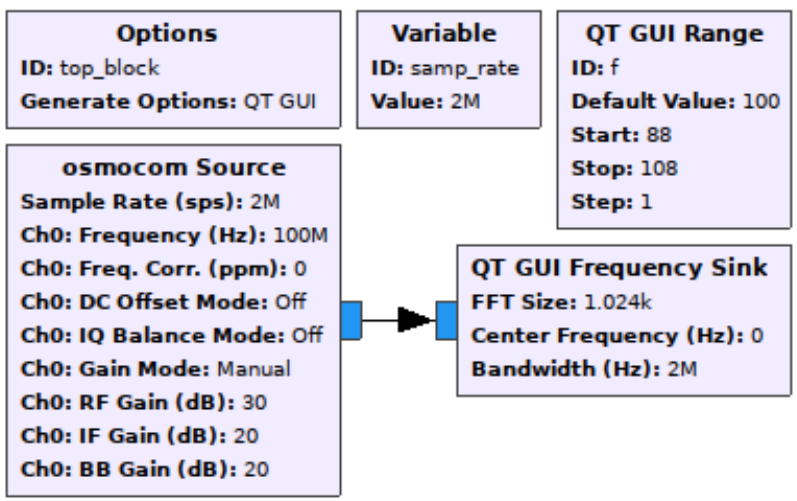
- Divers outils pour l'utilisateur (`gqrx`, `multimon` pour le décodage), `dump1090` ou `rtl*`) ...
- mais nous nous focaliserons ici sur l'aspect développeur [2] [3] [4]

- Bibliothèque de blocs de traitements assemblés en C++ ou Python,
- communication avec une multitude de sources et puits (ou données synthétiques),
- capacité à développer ses propres blocs de traitement (C/C++ ou Python)
- une interface graphique de génération de code Python : `gnuradio-companion`

2. J.-M Friedt & G. Goavec-Merou, *La réception radiofréquence définie par logiciel (Software Defined Radio – SDR)*, **153** 4–33 (2012)
3. J.-M Friedt, *Hacking the radiofrequency spectrum : GNURadio as a signal processing prototyping tool*, OHM 2013 (Observe, Hack, Make) (2013)
4. G. Goavec-Merou, J.-M Friedt, *GNURadio as a general purpose digital signal processing environment*, FOSDEM 2014

Aspects logiciels : GNURadio [5] [6]

- Aspect logiciel : GNURadio pour le traitement numérique du signal
- Logiciel libre disponible sous GNU/Linux, MacOS & MS-Windows
- Générateur de code python : `gnuradio-companion`
- Opensource : **apprendre** par l'exemple et **ajouter ses blocs**

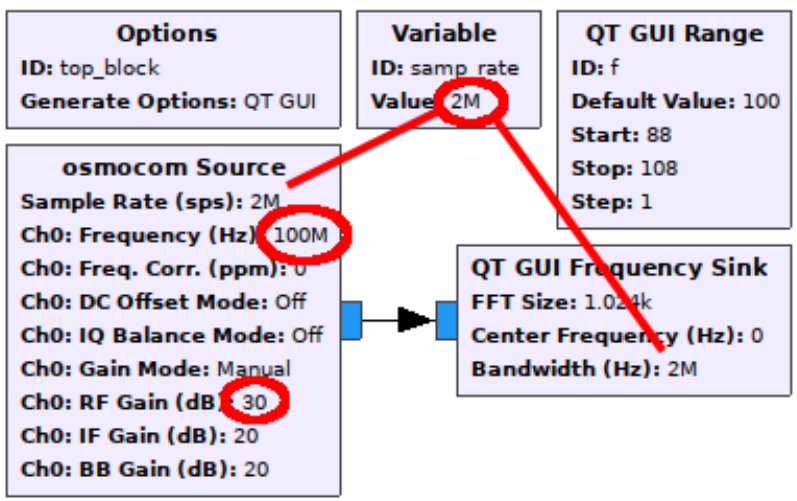



```
from PyQt5 import Qt
from gnuradio import gr
from gnuradio import qtgui
from gnuradio.filter import firdes
from gnuradio.qtgui import Range, RangeWidget
import osmosdr

class top_block(gr.top_block, Qt.QWidget):
  def __init__(self):
    gr.top_block.__init__(self, "Top Block")
    Qt.QWidget.__init__(self)
[...]
    # Variables
    self.samp_rate = samp_rate = 2e6
    self.f = f = 100
    # Blocks
    self._f_range = Range(88, 108, 1, 100, 200)
    self._f_win = RangeWidget(self._f_range, self.→
          ↪set_f, "f", "counter_slider", float)
    self.top_layout.addWidget(self._f_win)
    self.qtgui_freq_sink_x_0=qtgui.freq_sink_c(
        1024, #size
        firdes.WIN_BLACKMAN_hARRIS, #wintype
        0, samp_rate, "", 1 #fc, bw, name, inputs
    )
[... configuration du freq sink]
    self.osmosdr_src_0 = osmosdr.source( args="→
          ↪numchan=" + str(1) + " " + '' )
    self.osmosdr_src_0.set_sample_rate(samp_rate)
    self.osmosdr_src_0.set_center_freq(f*1e6, 0)
[...]
    self.connect((self.osmosdr_src_0, 0), (self.→
          ↪qtgui_freq_sink_x_0, 0))
[...]
def main(top_block_cls=top_block, options=None):
[...]
```

5. https://www.gnuradio.org & https://pubs.gnuradio.org
6. https://fosdem.org/2019/schedule/track/free_software_radio/

Aspects logiciels : GNURadio [5] [6]

- Aspect logiciel : GNURadio pour le traitement numérique du signal
- Logiciel libre disponible sous GNU/Linux, MacOS & MS-Windows
- Générateur de code python : `gnuradio-companion`
- Opensource : **apprendre** par l'exemple et **ajouter ses blocs**




```python
from PyQt5 import Qt
from gnuradio import gr
from gnuradio import qtgui
from gnuradio.filter import firdes
from gnuradio.qtgui import Range, RangeWidget
import osmosdr

class top_block(gr.top_block, Qt.QWidget):
  def __init__(self):
    gr.top_block.__init__(self, "Top Block")
    Qt.QWidget.__init__(self)
[...]
    # Variables
    self.samp_rate = samp_rate = 2e6
    self.f = f = 100
    # Blocks
    self._f_range = Range(88, 108, 1, 100, 200)
    self._f_win = RangeWidget(self._f_range, self.→
        ↪set_f, "f", "counter_slider", float)
    self.top_layout.addWidget(self._f_win)
    self.qtgui_freq_sink_x_0=qtgui.freq_sink_c(
        1024, #size
        firdes.WIN_BLACKMAN_hARRIS, #wintype
        0, samp_rate, "", 1 #fc, bw, name, inputs
    )
[... configuration du freq sink]
    self.osmosdr_src_0 = osmosdr.source( args="→
        ↪numchan=" + str(1) + " " + '' )
    self.osmosdr_src_0.set_sample_rate(samp_rate)
    self.osmosdr_src_0.set_center_freq(f*1e6, 0)
[...]
    self.connect((self.osmosdr_src_0, 0), (self.→
        ↪qtgui_freq_sink_x_0, 0))
[...]
def main(top_block_cls=top_block, options=None):
[...]
```

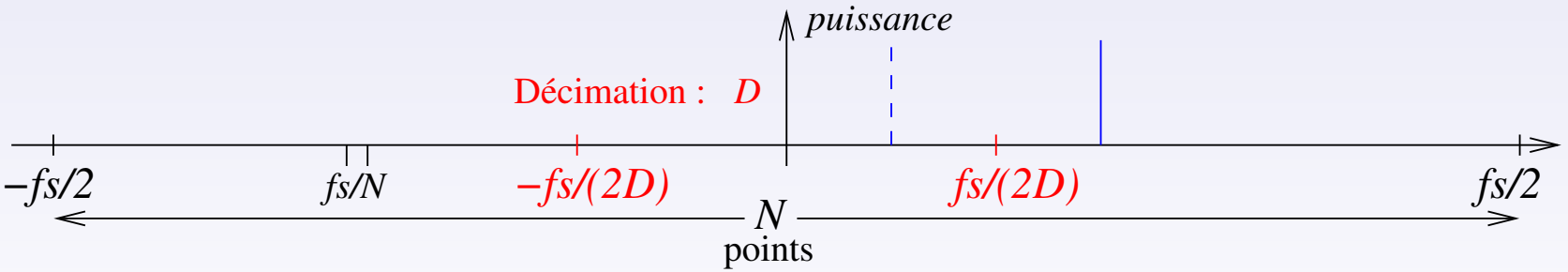
5. https://www.gnuradio.org & https://pubs.gnuradio.org
6. https://fosdem.org/2019/schedule/track/free_software_radio/

Implémentation logicielle de quelques blocs de traitement

Analogique	Numérique
Redresseur (démod. AM)	valeur absolue
Filtre (passe bas)	FIR : $y_n = \sum b_k x_{n-k}$
Mélangeur	$s_n \leftarrow s_n \cdot sin(n/f_s),\ n \in \mathbb{N}$

Mais aussi des mises en œuvre d'algorithmes uniques au numérique (démodulateur FM)

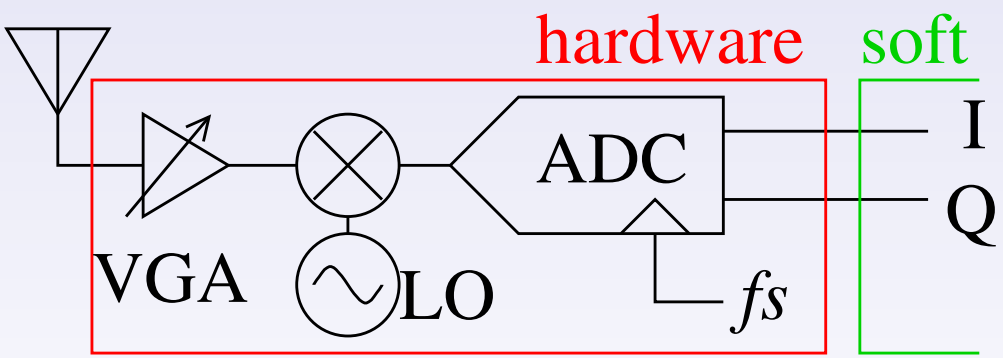
Réduction du débit de données : la **décimation**. Décimer de D permet de faire une homothétie sur l'axe des abscisses



Tout traitement spectral **nécessite de maîtriser la fréquence d'échantillonnage** f_s/D : le spectre s'étend de $-f_s/(2D)$ à $+f_s/(2D)$

Historique sur les récepteurs

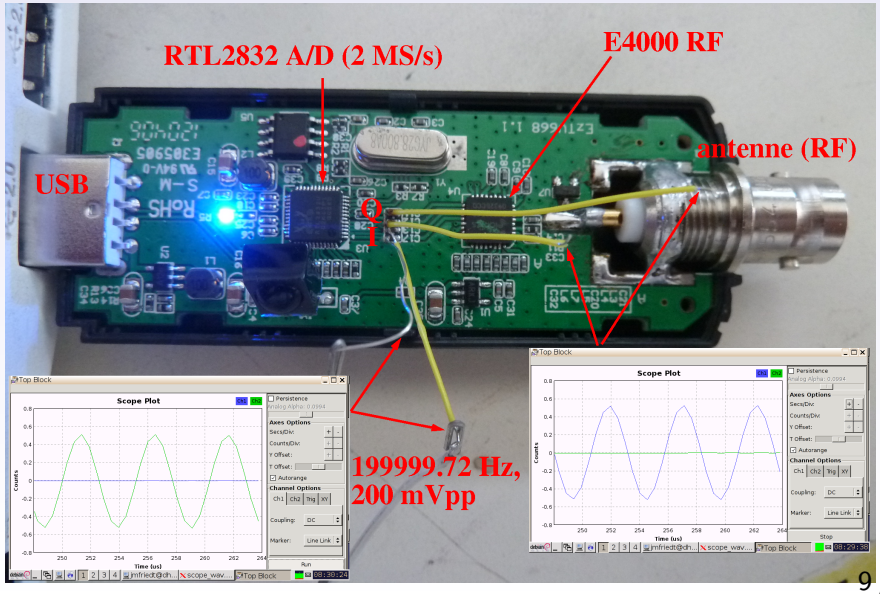
- Historiquement : solution super-hétérodyne pour compenser l'isolation médiocre du mélangeur induisant une fuite LO excessive
- Solution zero IF : échantillonnage direct ...
- ... mais dynamique limitée par la résolution du convertisseur analogique.
- Convergence des étages de réception RF (ampli-mélangeur-passe bas), puissance de calcul des systèmes embarqués, et *outils libres* de traitement



Divers critères de sélection du récepteur :
fréquence de travail (porteuse LO)), bande passante (f_s), sensibilité (ADC)

Le récepteur DVB-T

- Un récepteur de télévision numérique terrestre (DVB-T) à moins de 10 euros
- Récepteur généraliste avec la démodulation prise en charge par le logiciel ...
- ... permet de recevoir bien d'autres modes de communication que la télévision.
- Caractéristiques :
  - porteuse $\in [30 : 1600]$ MHz (inclut FM, aviation & satellites, GPS et Iridium),
  - bande passante $\in [1,0 : 2,4]$ MS/s,
  - convertisseur analogique-numérique 8 bits
- utilisation en bande de base comme oscilloscope en éliminant le R820T2 et se connectant sur le RTL2832U !
- pilote osmosdr et récemment SoapySDR



Applications

Reconfiguration du logiciel : une même plateforme matérielle permet d'appréhender une multitude de modes de communication

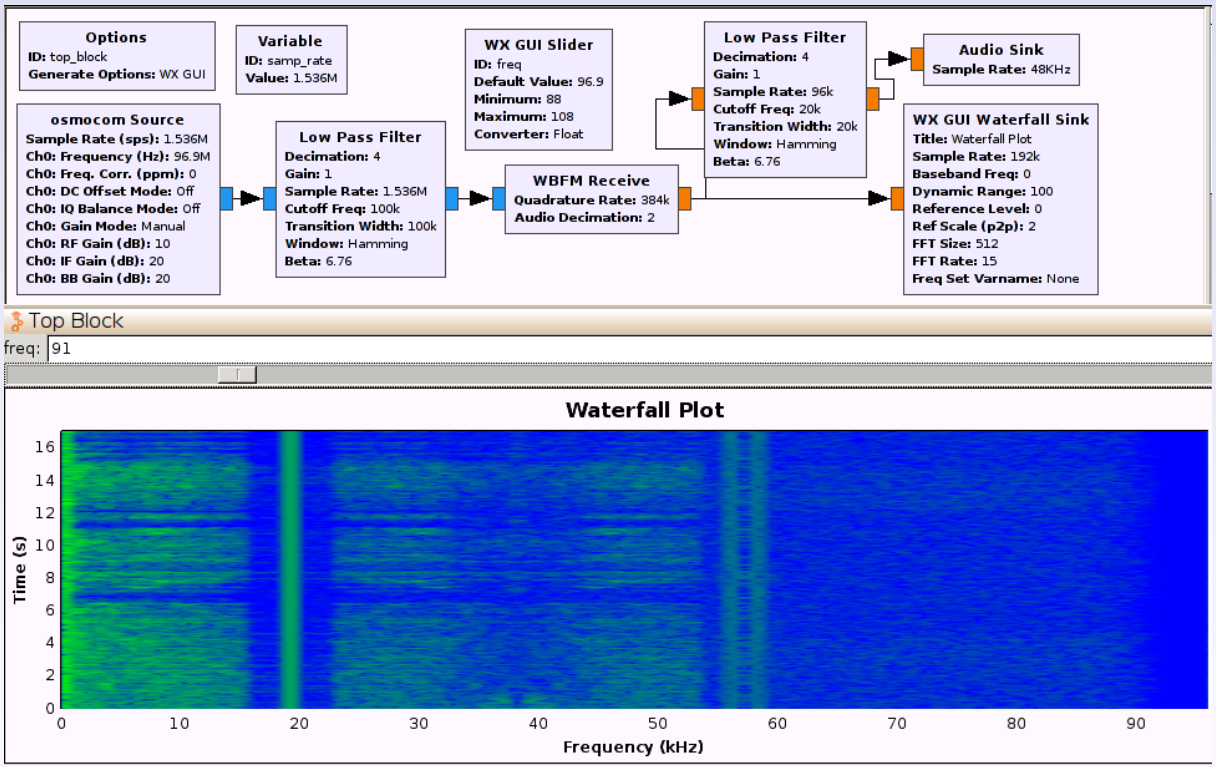
Mode	modulation	porteuse (MHz)	bande passante (kHz)
WBFM	FM analogique	88–108	120
DAB	OFDM (FM num)	175–250	1537
POCSAG	FSK	466	9
ADS-B	ASK	1090	50
XE1203	FSK	434, 868	dépend du débit
ACARS	AM	131,725	5
Iridium	PSK	1620	31,5
GPS	BPSK	1575,42	1000
NOOA (APT)	analog	137	34
Meteor M2 (LRPT)	QPSK	137,9	120
RADAR passif	bruit/xcorr		max $(\Delta R = c_0/(2B))$

Tous[7] ces modes sont accessibles avec un récepteur échantillonnant à $fs \leq 2,4$ MS/s ($f_s \geq 2BW$)

7. https://www.sigidwiki.com/wiki/Signal_Identification_Guide

WBFM

- La France reste figée sur la radio analogique dans la bande FM (88–108 MHz, 250 kHz/station)
- FDMA : extraction des divers canaux contenus dans chaque bande FM (L+R, L-R, RDS),
- décodage multicanaux (importance de la bande passante)

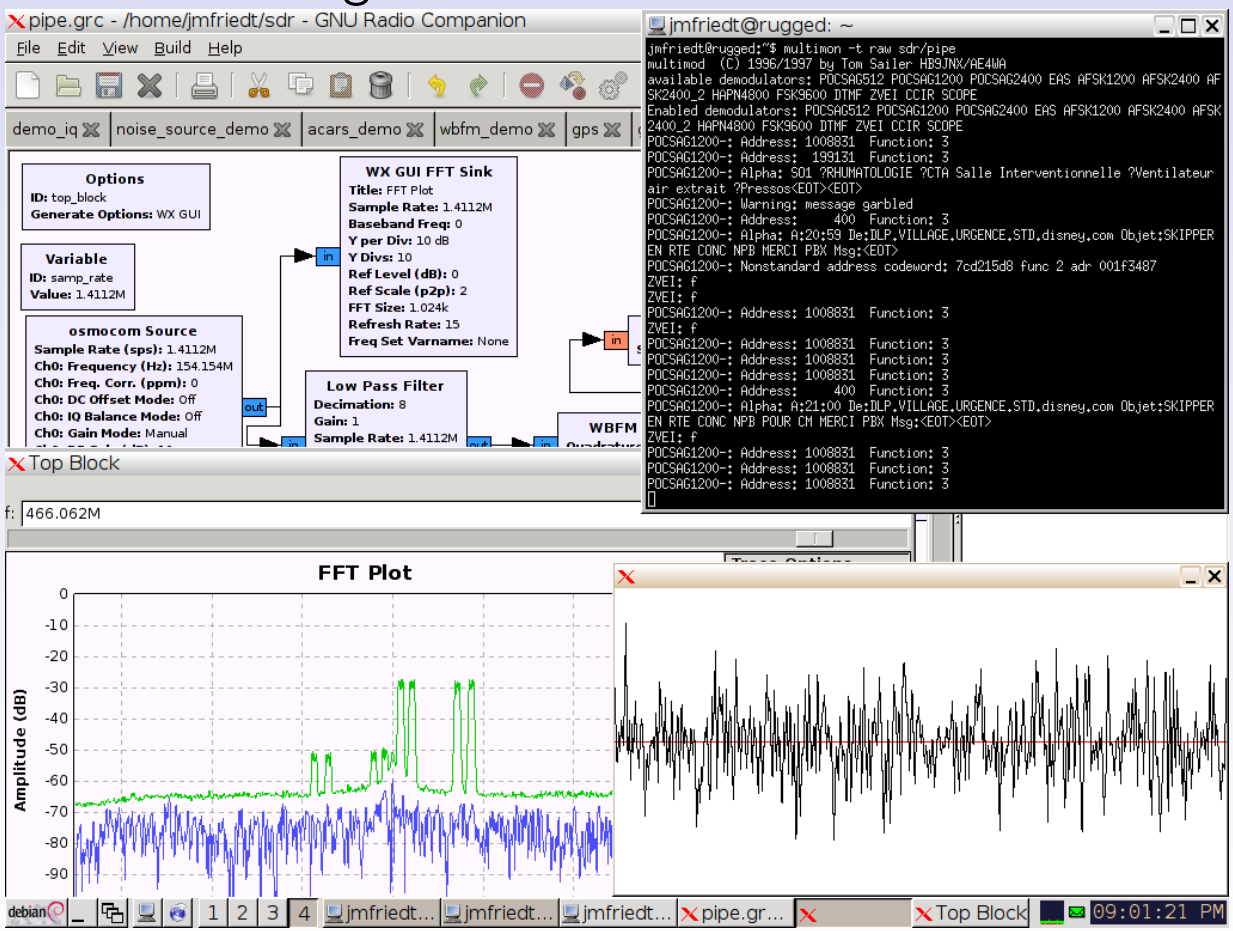


Première chaîne de traitement “triviale” pour se familiariser avec les contraintes de gestion de flux (adapter la fréquence d’échantillonnage d’entrée – souple – à la fréquence d’échantillonnage de sortie – rigide)

POCSAG

Pagers (466 MHz+0,025×N), un mode de communication sécurisé [8]

- FSK (PLL pour conversion fréquence → tension)
- interfaçage avec un outil externe de décodage (`multimon-ng`)
- réseau national d'urgence

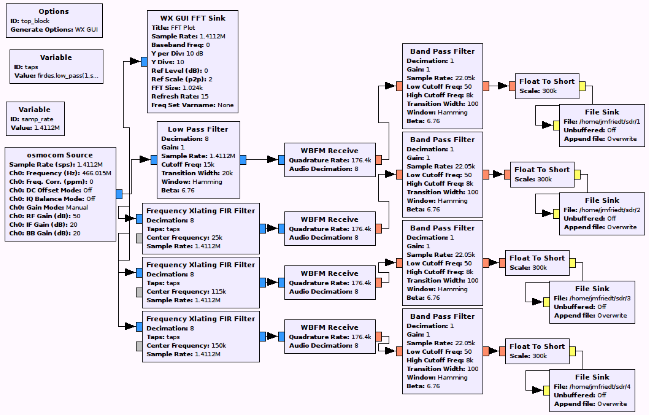


8. Alphapage : "Pour une alerte rapide, sécurisée et fiable, la radiomessagerie d'e*Message"

POCSAG

Pagers (466 MHz+0,025×*N*), un mode de communication sécurisé [8]

- FSK (PLL pour conversion fréquence → tension)
- interfaçage avec un outil externe de décodage (`multimon-ng`)
- réseau national d'urgence



Décodage multicanaux tant que la bande d'acquisition est suffisante

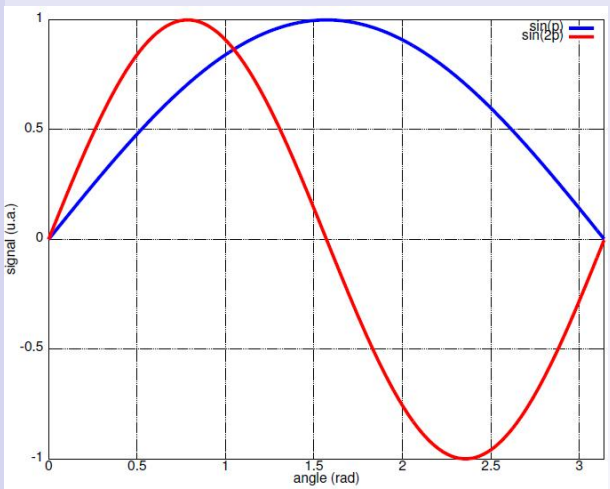
8. Alphapage : "Pour une alerte rapide, sécurisée et fiable, la radiomessagerie d'e*Message"

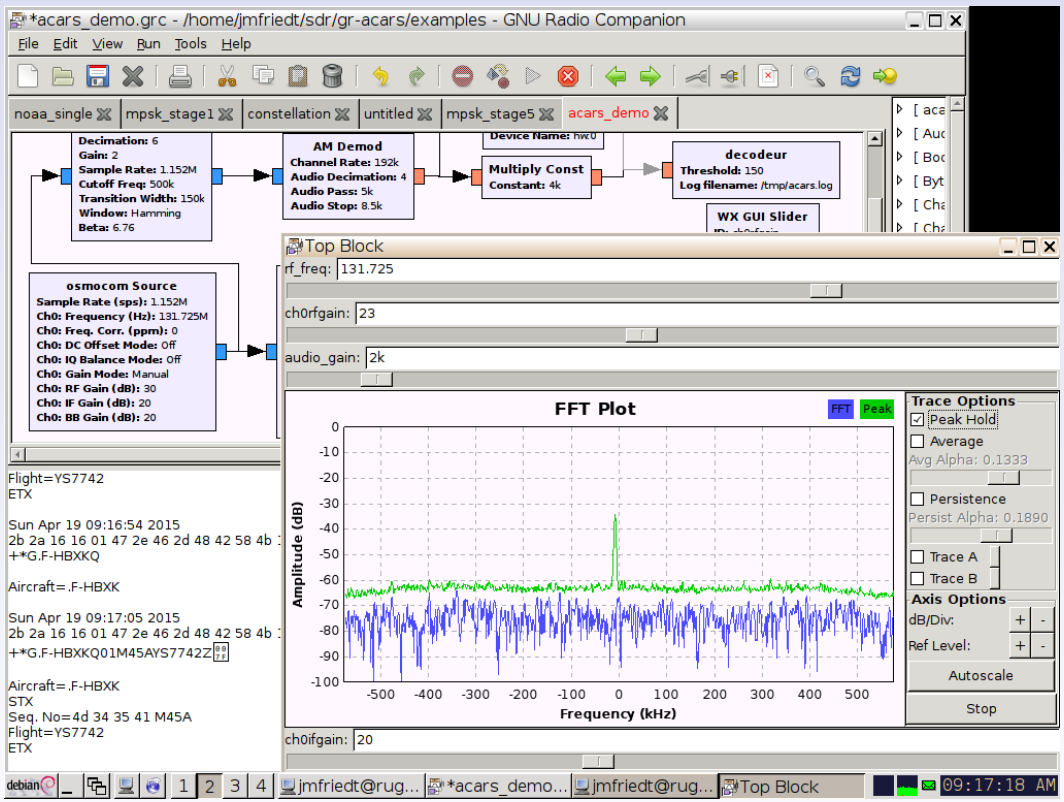
ACARS

Écriture d'un bloc de traitement dédié : identification de la valeurs des bits par corrélation

- 131,725 MHz, AM
- bits encodés par 1200 ou 2400 Hz @ 1200 bps
- texte libre, télémesure
- `gr-acars`

1. enregistrer et prototyper sous GNU/Octave
2. GNU/Octave → C sur fichier enregistré
3. C → GNU Radio (flux de données, FIFO)





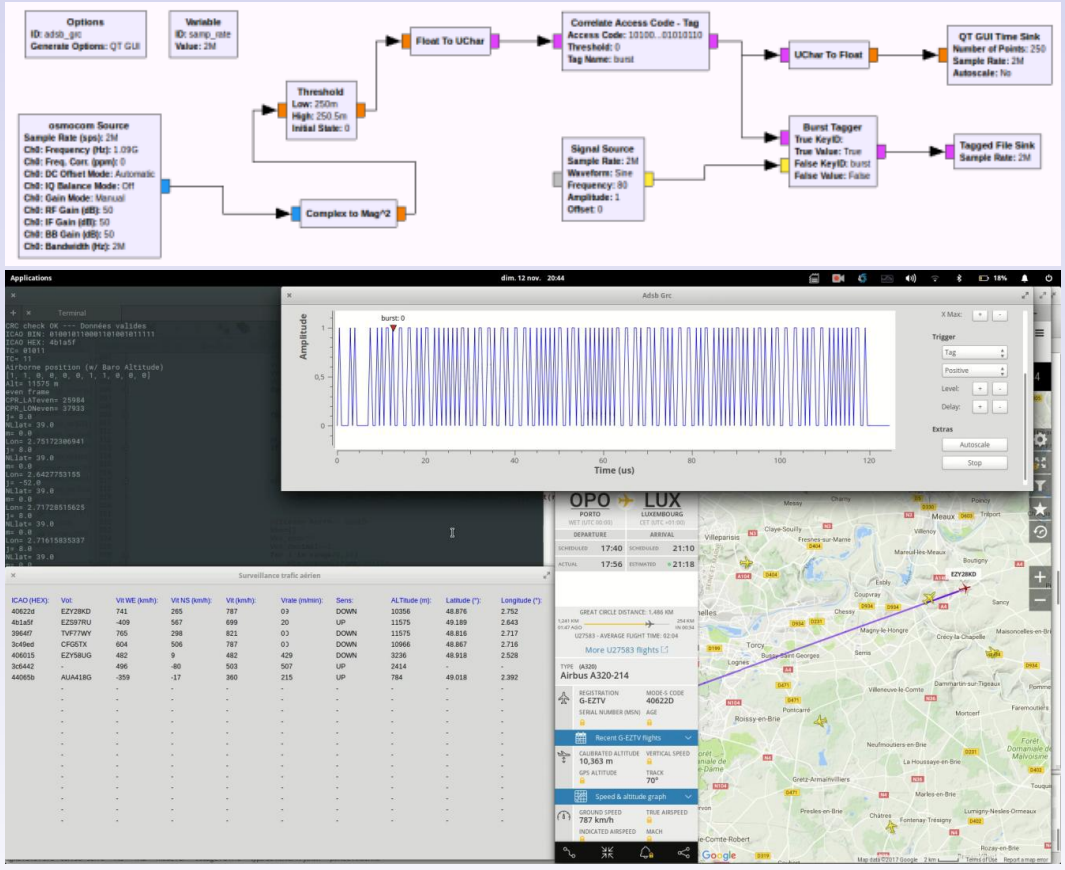
$$\int_0^1 \sin(2\pi t)\sin(\pi t)dt \propto \int_0^1 \cos(3\pi t)-\cos(\pi t)dt = \sin(3\pi)-\sin(0)-\sin(\pi)-\sin(0) = 0$$

$$\int_0^1 \sin(2\pi t)\sin(2\pi t)dt \propto \int_0^1 \cos(4\pi t)-\cos(0)dt = \sin(4\pi)-\sin(0)+1 \neq 0$$

$$\int_0^1 \sin(\pi t)\sin(\pi t)dt \propto \int_0^1 \cos(2\pi t)-\cos(0)dt = \sin(4\pi)-\sin(0)+1 \neq 0$$

ADS-B [9]

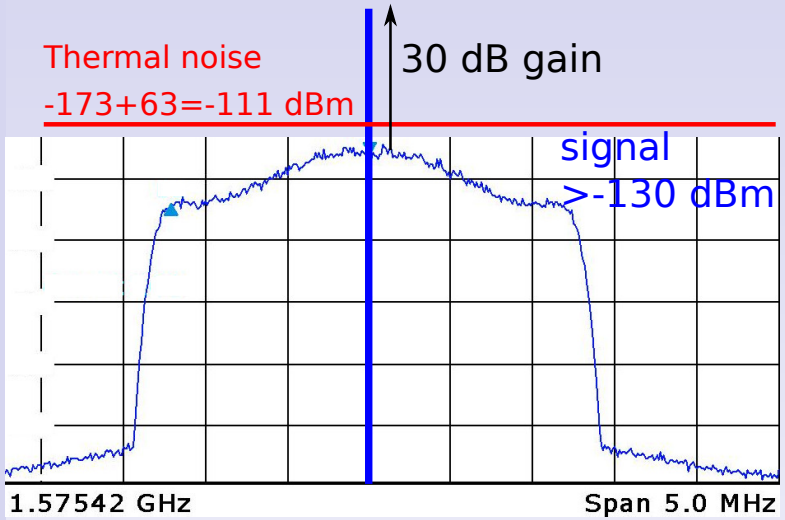
- 1090 MHz
- modulation AM
- position, altitude et vitesse des avions
- recherche de début de trame par corrélation avec un code connu (préambule)
- analyse de la trame en post-traitement (script Python)
- validation de la position des avions avec `flightradar24.com`

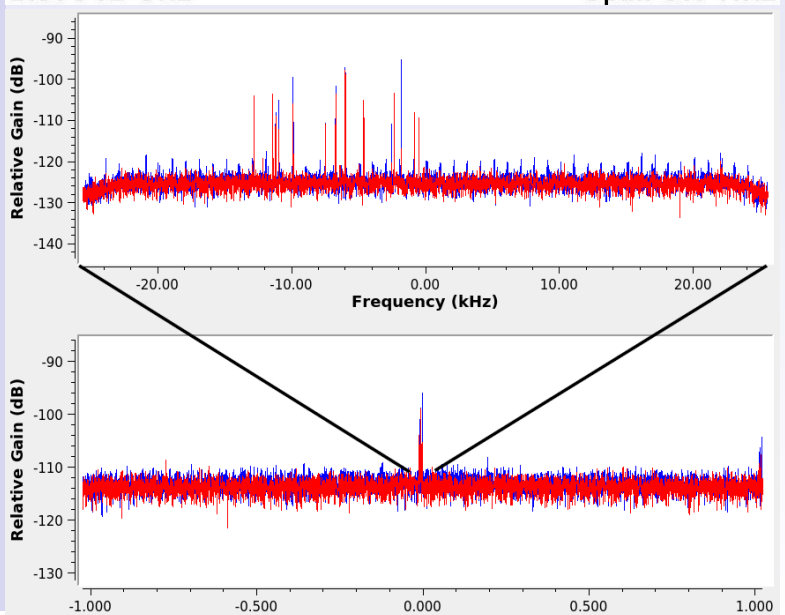


9. T. Lavarenne, *Introduction à la surveillance du trafic aérien – les signaux ACARS et ADS-B*, Bull. Union des Physiciens (BUP) **1005** 831–855 (2018)

GPS

- 1575,42 MHz, BPSK : $s(t) = \exp(j(\omega_c + \varphi))$ avec $\varphi \in [0, \pi]$
- difficulté : le signal est sous le bruit thermique

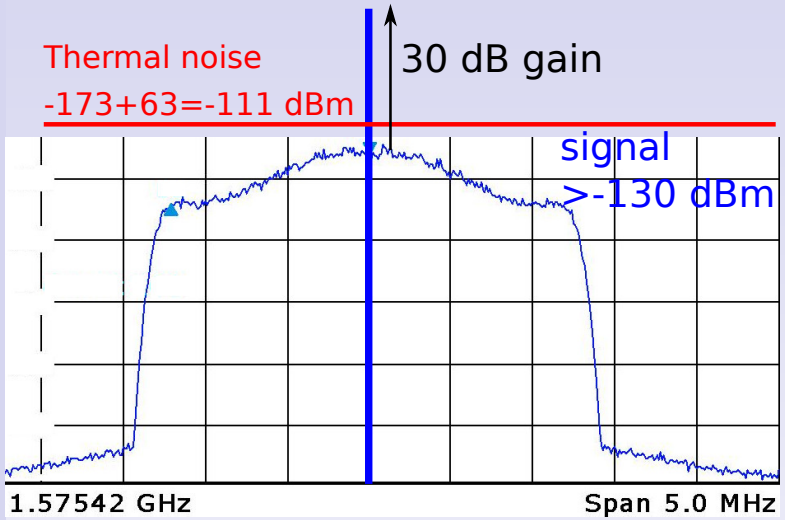


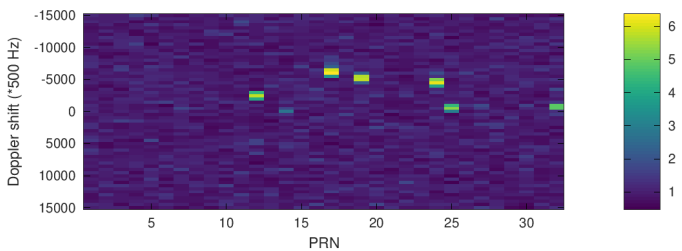


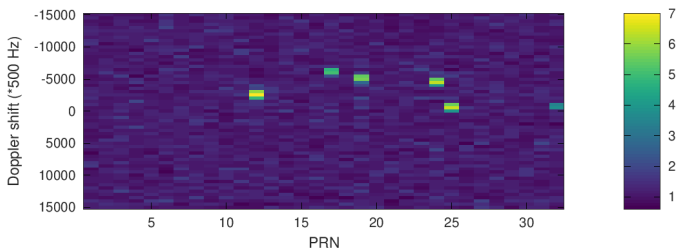
- densité spectrale de bruit : -174 dBm/Hz
- bande d'intégration 2 MHz=63 dBm
- bruit à -111 dBm
- un satellite émet 50 W (17 dBm=47 dBm), pertes de propagation (1.57542 GHz, 20000 km) de 182 dB induisent signal au sol de -122 dBm (13 dBi de gain d'antenne)
- seul le gain de compression sur 1023 bits$\simeq$ 30 dB permet de remonter le signal au dessus du bruit thermique
- élimination de BPSK par mise au carré : $s^2(t) = \exp(2 \times j(\omega_c + \varphi)) = \exp(2\omega_c t)$
- même sans connaîître les codes pseudo-aléatoires identifiant chaque satellite par CDMA, on peut retrouver son décallage Doppler et donc son élévation

GPS

- 1575,42 MHz, BPSK : $s(t) = \exp(j(\omega_c + \varphi))$ avec $\varphi \in [0, \pi]$
- difficulté : le signal est sous le bruit thermique





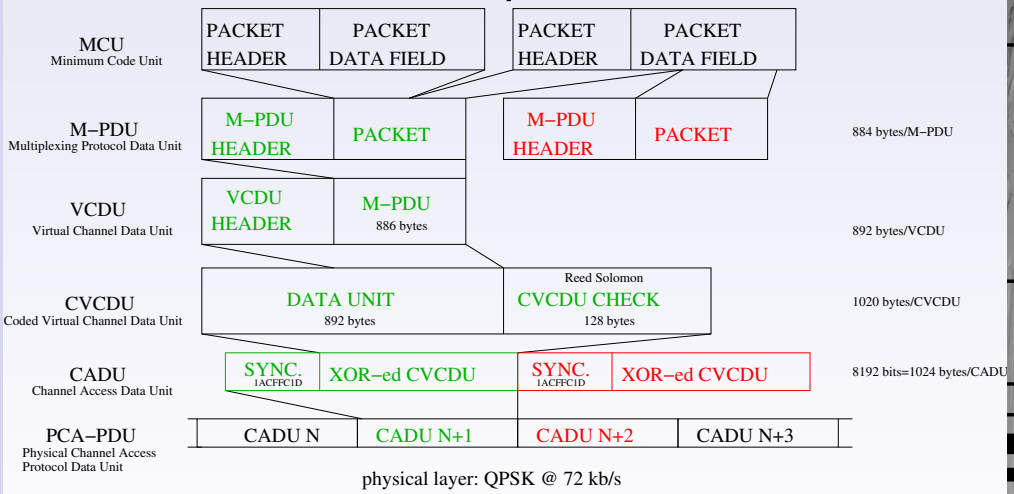


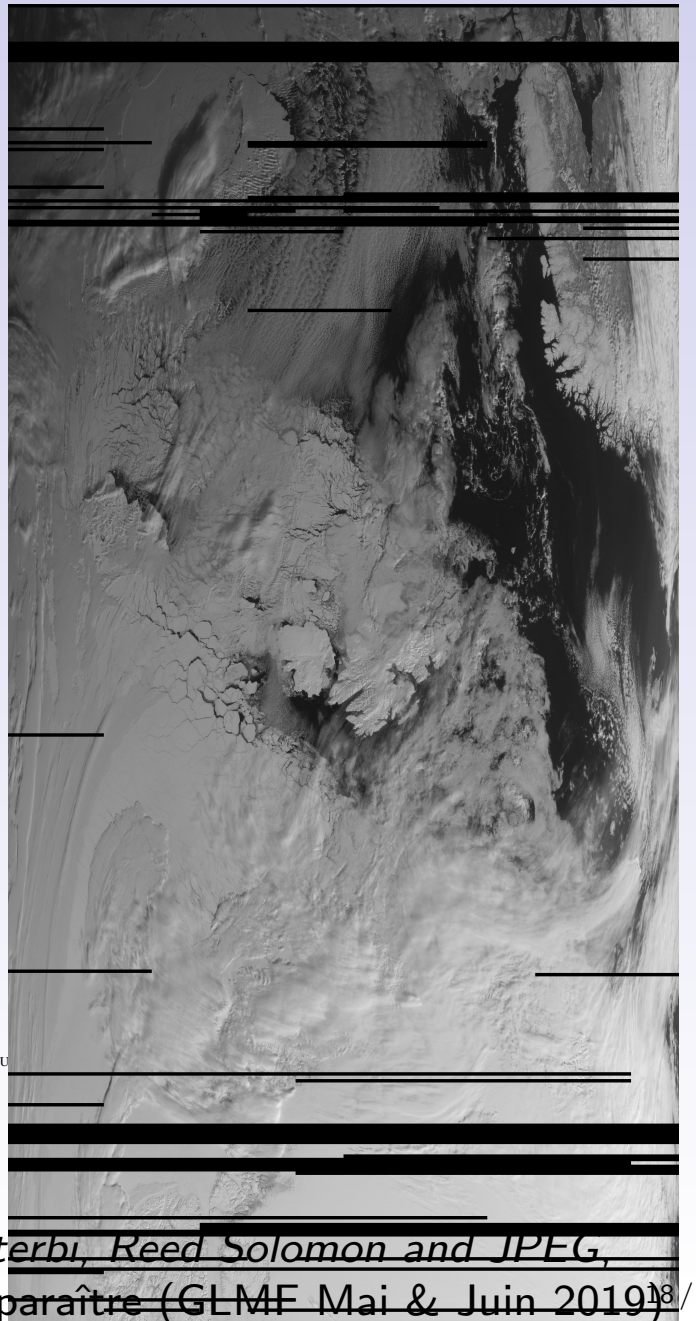
- densité spectrale de bruit : -174 dBm/Hz
- bande d'intégration 2 MHz=63 dBm
- bruit à -111 dBm
- un satellite émet 50 W (17 dBm=47 dBm), pertes de propagation (1.57542 GHz, 20000 km) de 182 dB induisent signal au sol de -122 dBm (13 dBi de gain d'antenne)
- seul le gain de compression sur 1023 bits$\simeq$ 30 dB permet de remonter le signal au dessus du bruit thermique
- élimination de BPSK par mise au carré : $s^2(t) = \exp(2 \times j(\omega_c + \varphi)) = \exp(2\omega_c t)$
- même sans connaîître les codes pseudo-aléatoires identifiant chaque satellite par CDMA, on peut retrouver son décallage Doppler et donc son élévation

Meteor M2 [10]

- 137,9 MHz, QPSK (Costas, clock sync.)
- code convolutif/Viterbi
- (Reed Solomon)
- regroupement des trames en paquets
- JPEG (Huffman, DCT)

Équivalent pour le spatial des couches OSI des réseaux informatiques



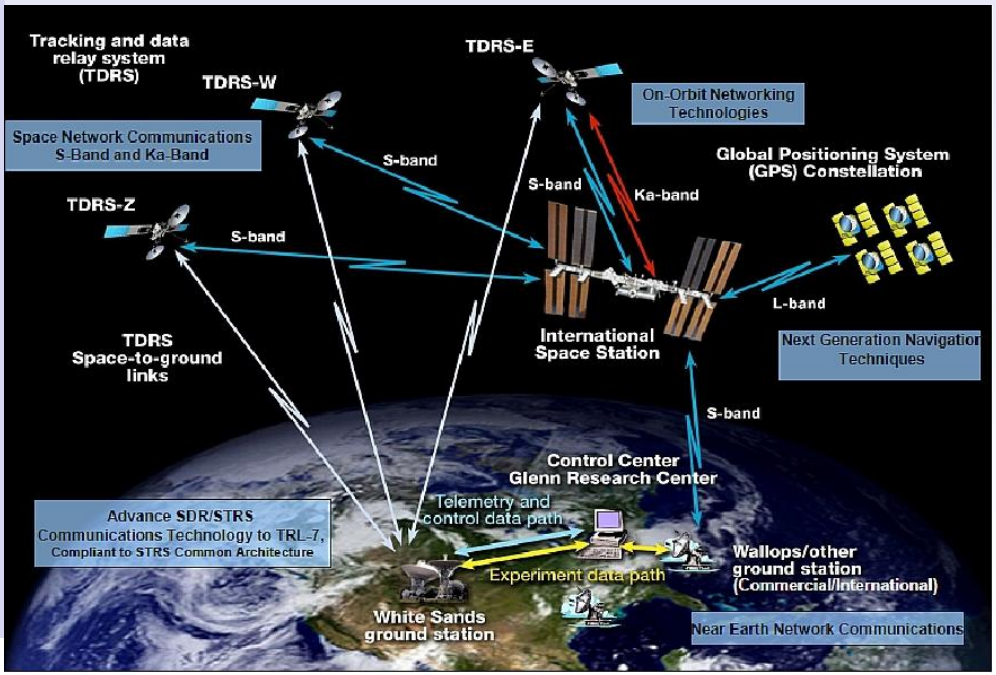


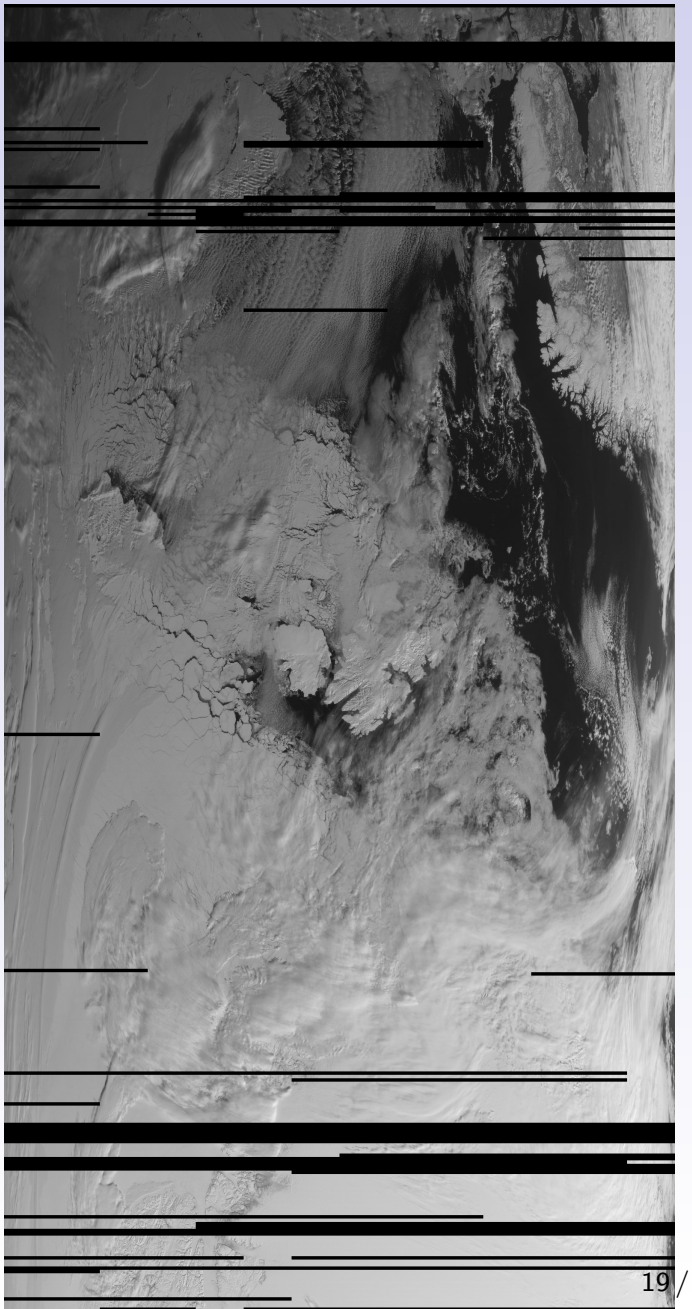
10. J.-M Friedt, *Decoding Meteor M2 : QPSK, Viterbi, Reed Solomon and JPEG*, FOSDEM 2019 (Free Software Radio devroom), à paraître (GLMF Mai & Juin 2019)

Meteor M2

- 137,9 MHz, QPSK (Costas, clock sync.)
- code convolutif/Viterbi
- (Reed Solomon)
- regroupement des trames en paquets
- JPEG (Huffman, DCT)

Équivalent pour le spatial des couches OSI des réseaux informatiques

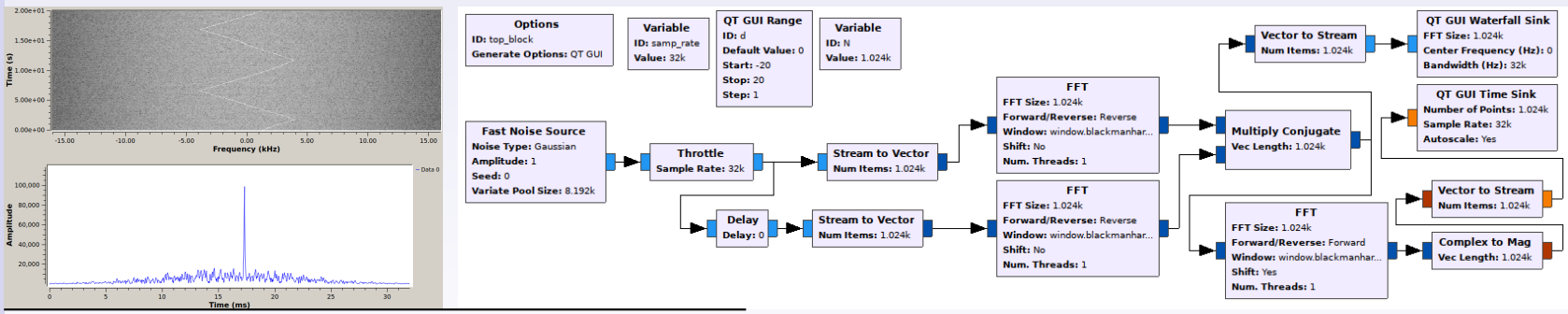




RADAR passif

Intercorrélation d'une voie de référence avec une voie de surveillance pour trouver les copies décallées dans le temps et en fréquence (Doppler) des cibles

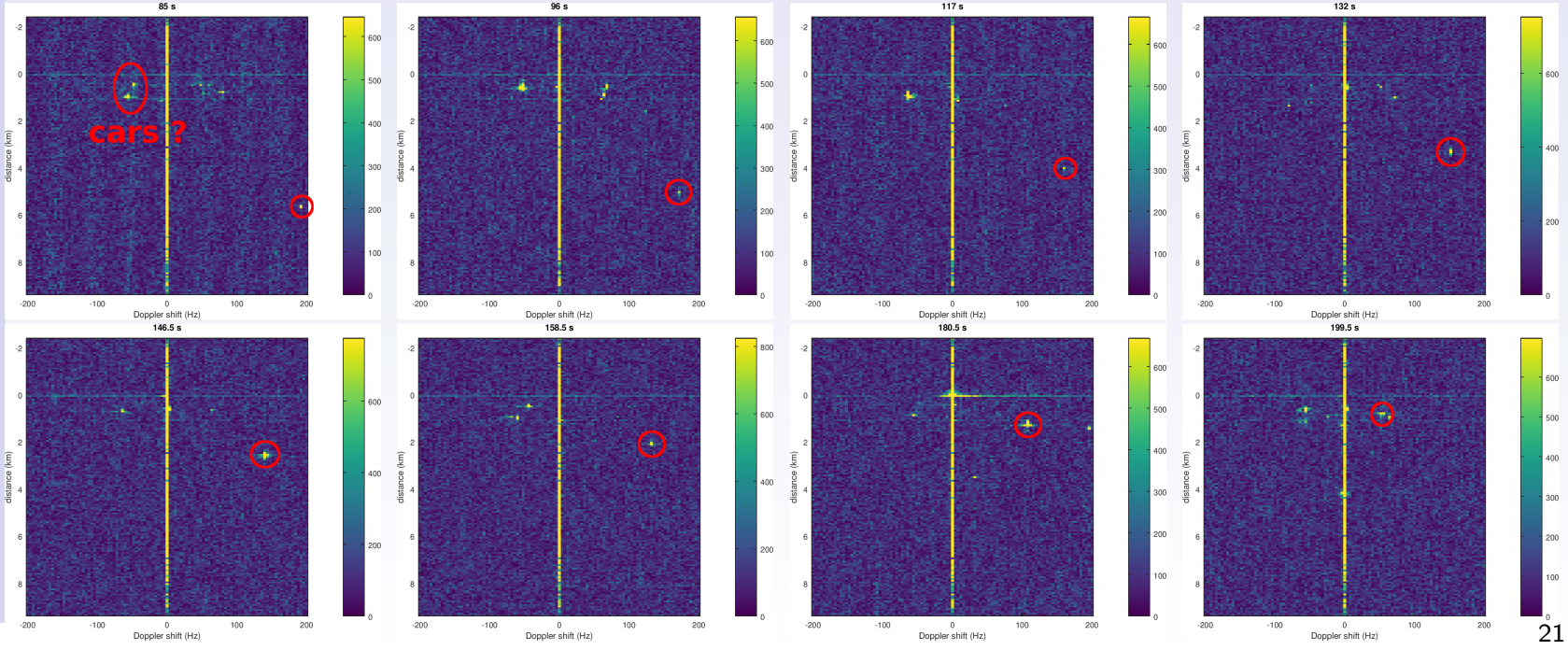
- défi expérimental : synchronisation d'une multitude de récepteurs sur un oscillateur local commun
- résolution en distance donné par la bande passante de mesure
- possibilité de combiner des mesures séquentielles sur des bandes adjacentes pour améliorer la résolution [10]
- bus USB induit un retard non-déterministe mais constant tant que le flux n'est pas coupé : 0MQ ("UDP") pour transmettre en continu et ne recevoir que les trames utiles
- Corrélation en temps-réel exploite $FFT(xcorr(x, y)) = FFT(x) \cdot FFT^*(y)$



10. W. Feng, J.-M. Friedt, G. Cherniak, M. Sato, *Passive bistatic radar using DVB-T receivers as general-purpose software-defined radio receivers*, Rev. Sci. Instrum. **89**, 104701 (Sept. 2018)

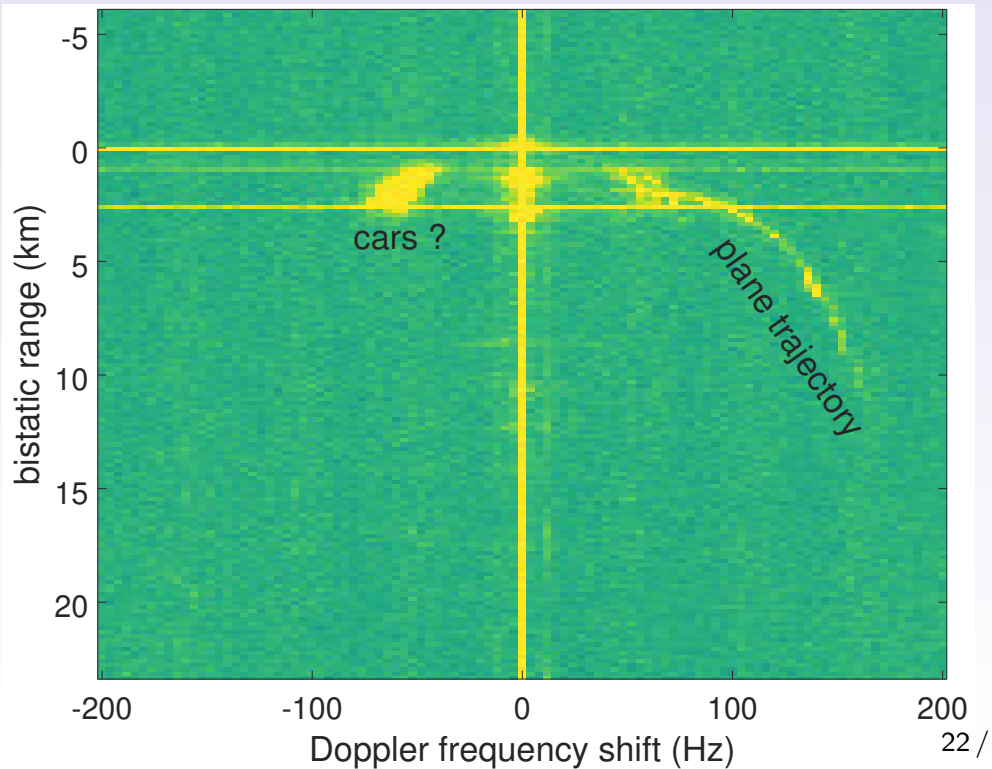
RADAR passif : avions

- Flux de données : 32 MB/s=1.92 GB/min
  2 canaux×I/Q×4 octets/mesure×2 MS/s
- Avions atterissant au port de Sendai (Japon), émission TV à 473 MHz (NHK, 3 kW)
- cartes vitesse-distance : acquisitions de 3 minutes nécessitant plusieurs heures de traitement sous GNU/Octave



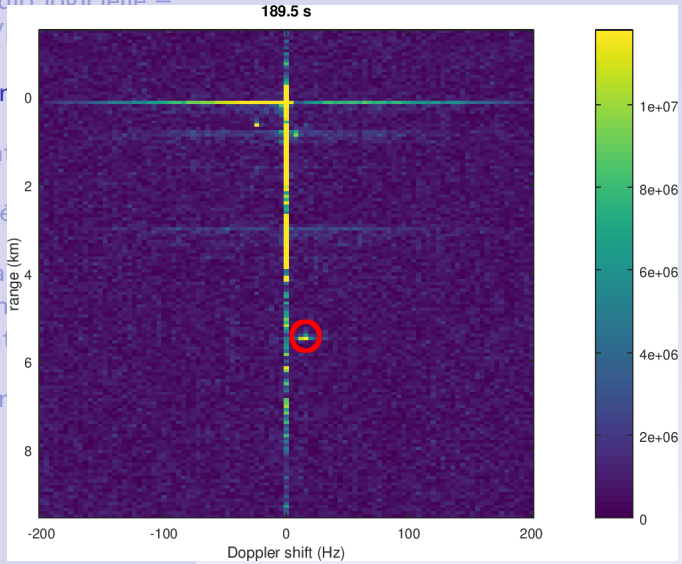
RADAR passif : avions

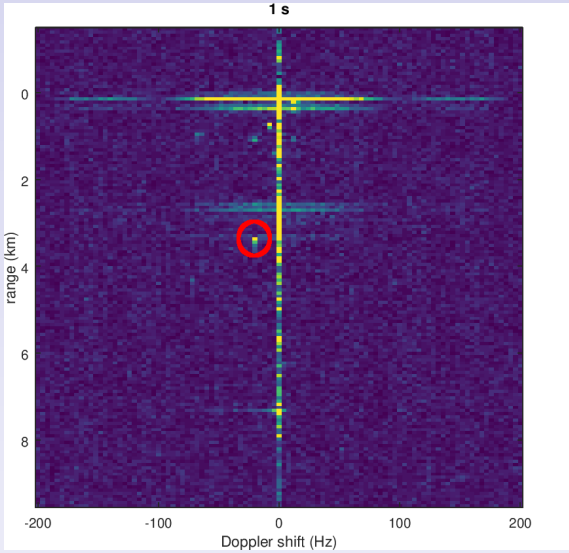
- Flux de données : 32 MB/s=1.92 GB/min
  2 canaux×I/Q×4 octets/mesure×2 MS/s
- Avions atterissant au port de Sendai (Japon), émission TV à 473 MHz (NHK, 3 kW)
- cartes vitesse-distance : acquisitions de 3 minutes nécessitant plusieurs heures de traitement sous GNU/Octave

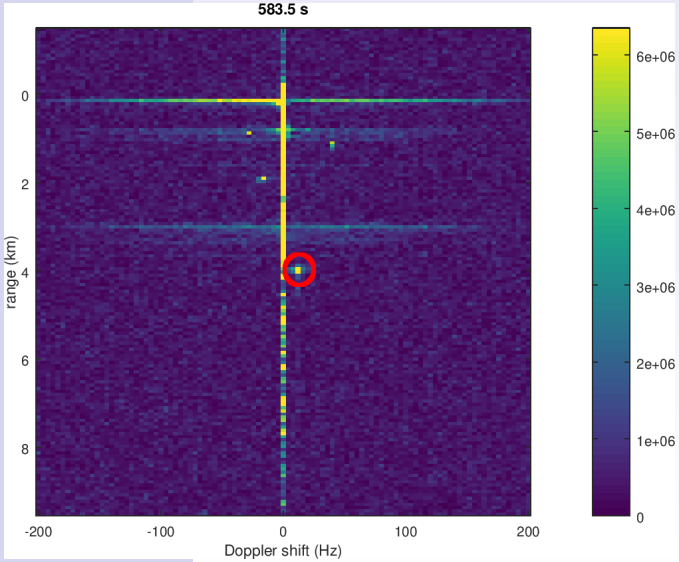


RADAR passif : bateaux

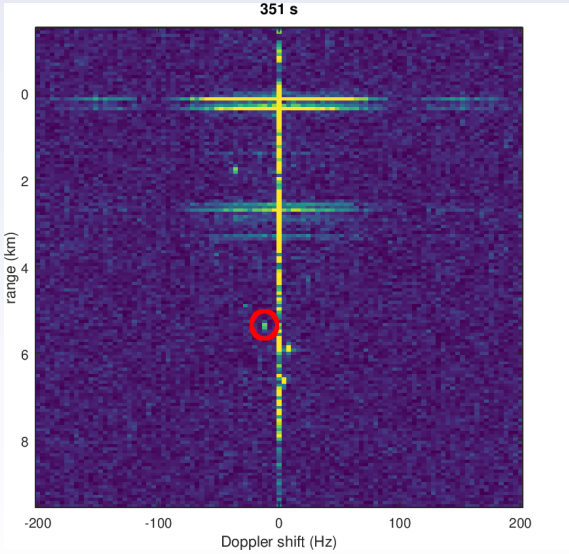
Cartes distance-vitesse (*range-Doppler*) : bateaux (port de Sendai, Japon)





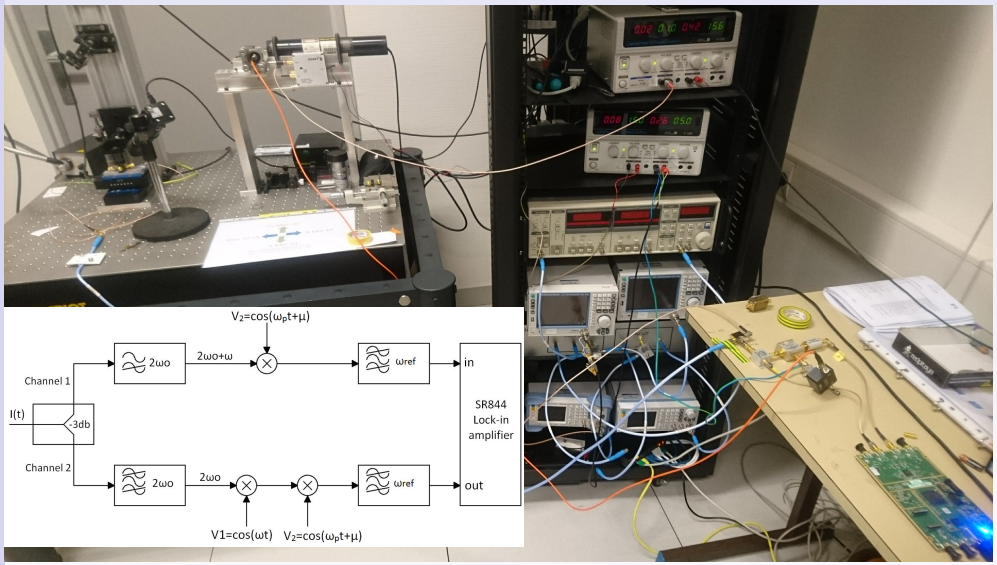


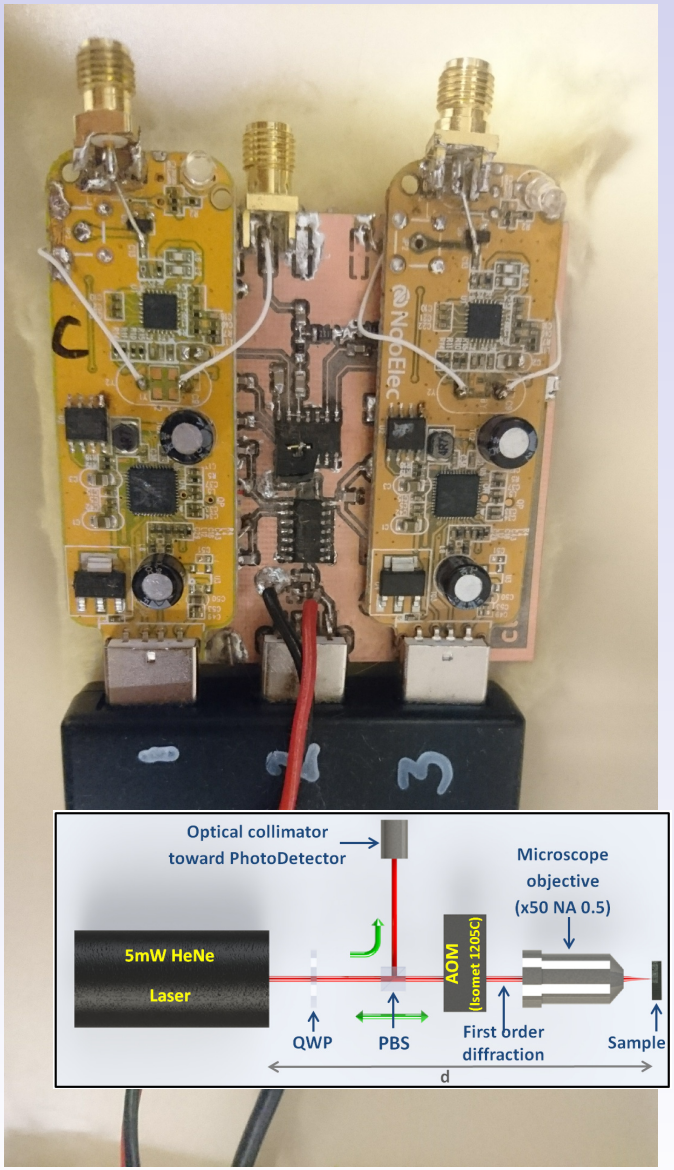
$f_D \simeq 12$ Hz$\simeq$ 3.6 m/s$\simeq$ 13 km/h$\simeq$ 7 nœuds, cohérent avec AIS ($\simeq$12 nœuds vitesse max.)



Détection synchrone

- Synchronisation d'une multitude de récepteurs (oscillateur commun distribué)
- $\Rightarrow$ détecteur synchrone
- Remplace deux analyseurs de spectre et détection synchrone dédiée par une Ettus Research B210
- Caractérisation de filtres radio SAW†

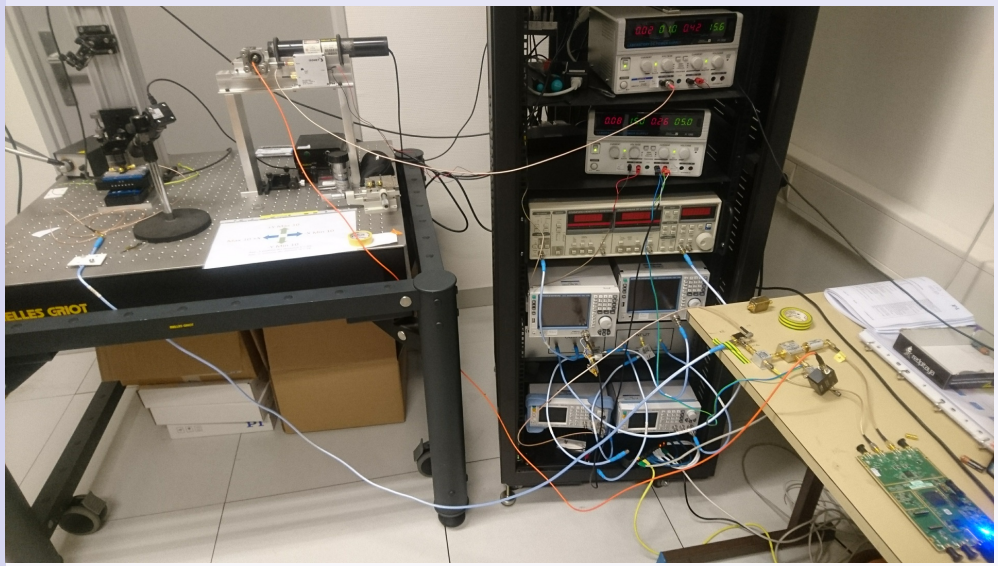


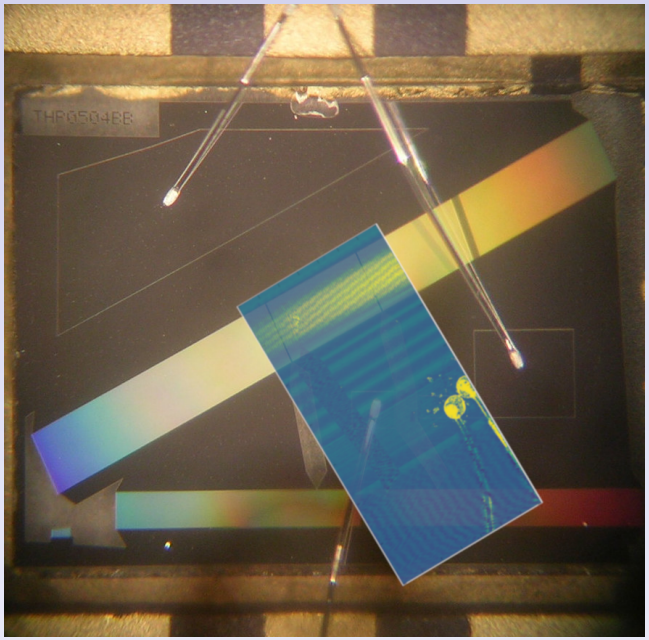


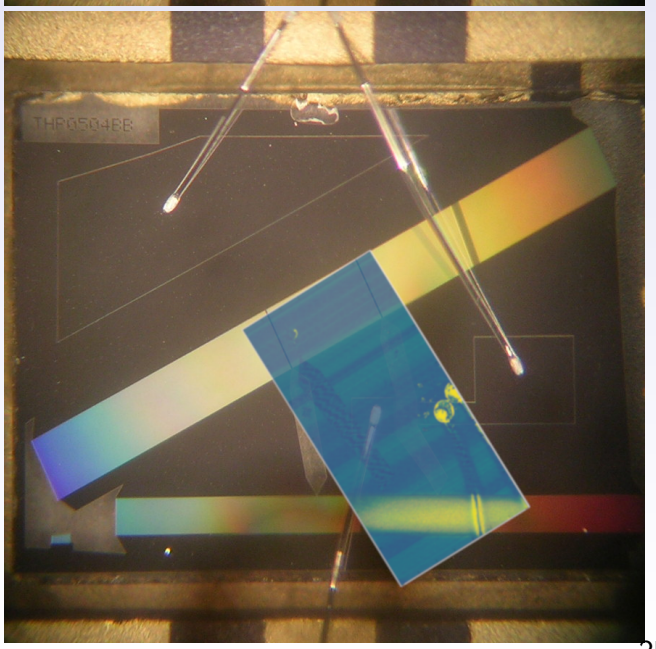
† D. Teyssieux, T. Baron, J.-M. Friedt, G. Martin, P. Vairac, *Absolute phase and amplitude mapping of surface acoustic wave fields*, Proc. 2013 IFCS

Détection synchrone

- Synchronisation d'une multitude de récepteurs (oscillateur commun distribué)
- $\Rightarrow$ détecteur synchrone
- Remplace deux analyseurs de spectre et détection synchrone dédiée par une Ettus Research B210
- Caractérisation de filtres radio SAW†







† D. Teyssieux, T. Baron, J.-M. Friedt, G. Martin, P. Vairac, *Absolute phase and amplitude mapping of surface acoustic wave fields*, Proc. 2013 IFCS

Pour aller plus loin ...

- Du matériel plus performant, mais au prix plus élevé
- Inutile d'investir dans des plateformes couteuses, mais si on a besoin de bande passante, elle se paie
- RX vs TX/RX

... liste non-exhaustive de matériel [11]

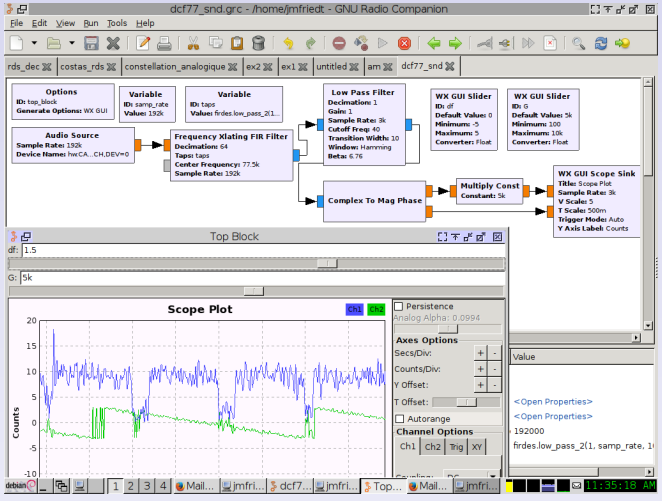
	Bande passante (MHz)	Coût	Gamme porteuse (MHz)	Émission ?
carte son	0,192	inclus	$\simeq$DC–0.096	full-duplex
Redpitaya	125	200/260 euros	DC–62,5	full-duplex
Oscilloscope				non
R820T2+RTL2832U	2,4	$<$10$	24–1700	non
AirspyMini (R820T)	3, 6, 10	99$	24–1700	non
PlutoSDR	20 (USB 2)	149$	70–6000	full-duplex
Lime SDR	61,44 (USB 3)	299$	0.1–3800	MIMO
HackRF One	2–20 (USB 2)	300$	1–6000	half-duplex
BladeRF 2.0	61,44 (USB 3)	480$	47–6000	MIMO
B210	61,44 (USB 3)	1200$	70–6000	MIMO

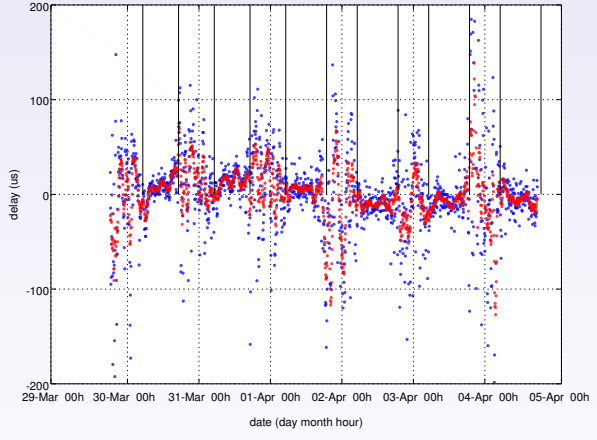
11. J.-M Friedt, *Matériel pour la radio logicielle*, GNU Linux Magazine France **224** (2019)

Même une carte son permet de faire de la science

DCF77/TDF : carte son pour décoder les signaux de dissémination de temps très basse fréquence (VLF) [12]

- 77,5 kHz, Mainflingen (Allemagne)
- dépendance du temps de propagation avec les conditions ionosphériques
- codage en amplitude *et en phase*
- mesure de tmeps de vol référencée au 1 PPS de GPS
- extension à diverses autres sources (MSF, TDF, eLORAN) pour analyser la relation de dispersion



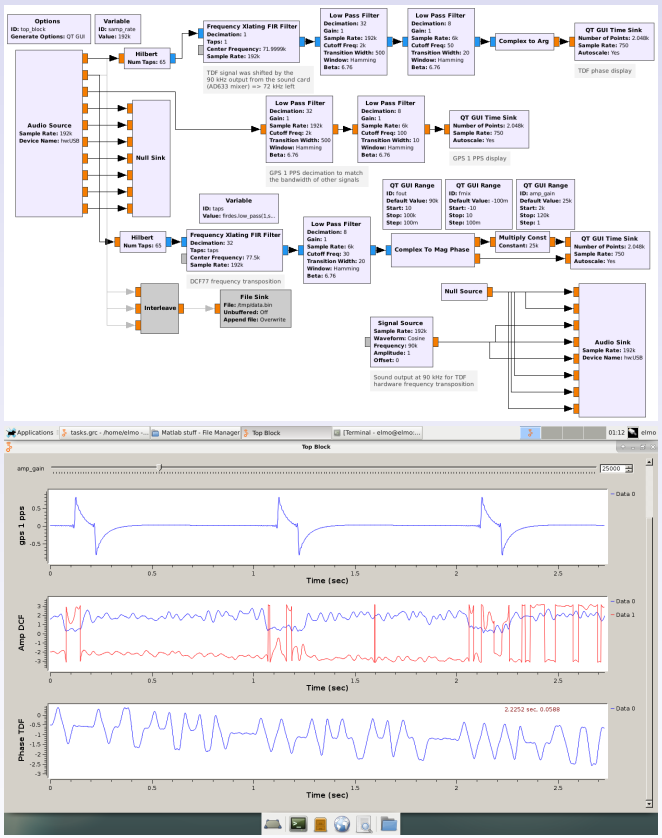


12. J.-M Friedt, C. Eustache, É. Carry, E. Rubiola, *Software defined radio decoding of DCF77 : time and frequency dissemination with a sound card*, Radio Science **53** (1), pp.48-61 (Jan 2018)

Même une carte son permet de faire de la science

DCF77/TDF : carte son pour décoder les signaux de dissémination de temps très basse fréquence (VLF) [12]

- 77,5 kHz, Mainflingen (Allemagne)
- dépendance du temps de propagation avec les conditions ionosphériques
- codage en amplitude *et en phase*
- mesure de tmeps de vol référencée au 1 PPS de GPS
- extension à diverses autres sources (MSF, TDF, eLORAN) pour analyser la relation de dispersion

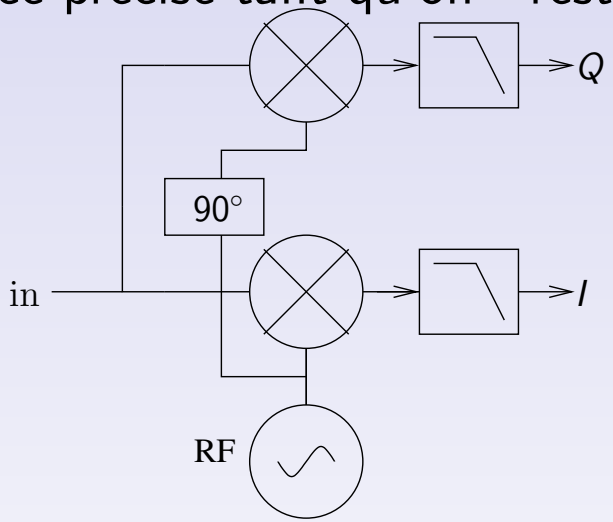


12. J.-M Friedt, C. Eustache, É. Carry, E. Rubiola, *Software defined radio decoding of DCF77 : time and frequency dissemination with a sound card*, Radio Science **53** (1), pp.48-61 (Jan 2018)

Fondements théoriques

Traitement de signaux échantillonnés en temps discret

- expression **complexe** des signaux en sortie de I/Q
- problème de la cohérence (décimation, nécessité de filtrer, transposition de fréquence)
- Xlating FIR : inutile d'avoir une source précise tant qu'on reste dans la bande
- coefficients **I**(dentité) et **Q**(uadrature) : $A \simeq \sqrt{I^2 + Q^2}$, $\varphi \simeq \arctan(Q/I)$
- signal réel (porteuse modulée) est devenu complexe après transposition



- mélanger pour transposer une fréquence :

$$A_1 \sin(\omega_1 t+\varphi)\cdot A_2 \cos(\omega_2 t) \propto A_1\cdot A_2\cdot\sin(\omega_1+\omega_2+\varphi)+\sin(\omega_1-\omega_2+\varphi)$$
$$\propto A_1 \cdot A_2 \cdot \sin(\varphi) \text{ si } \omega_1 = \omega_2$$

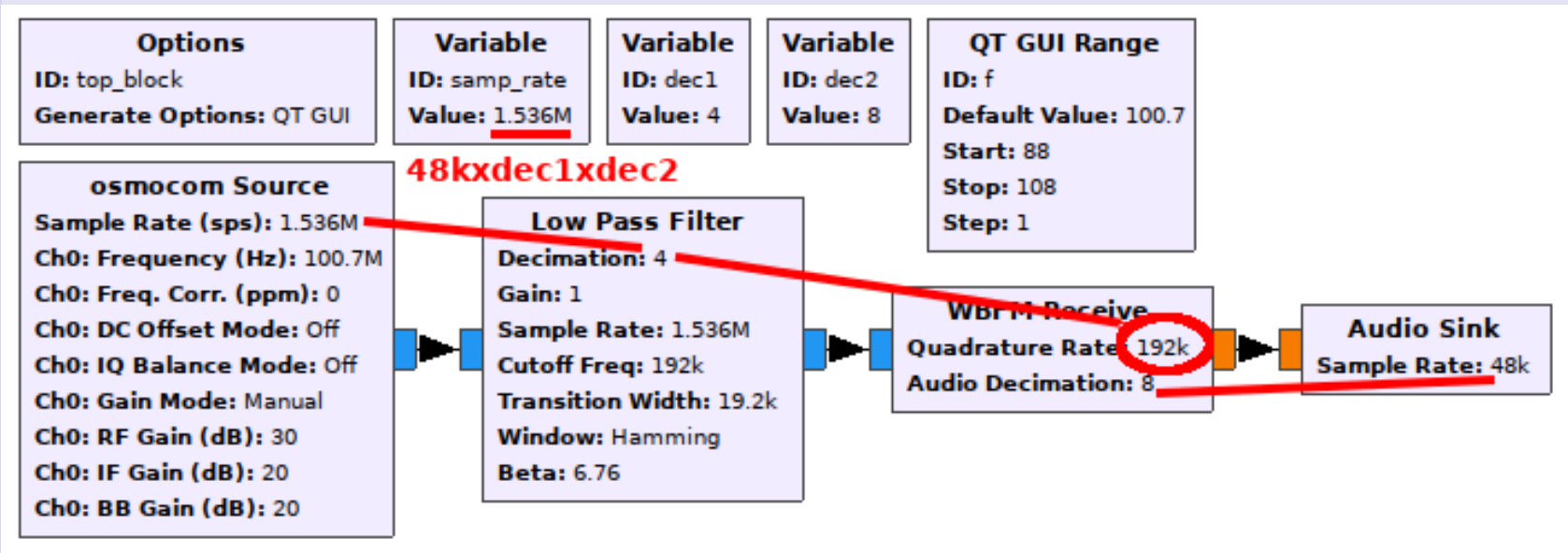
- si $\varphi = 0$, la sortie est toujours nulle quelquesoient A_1, A_2
- utiliser la version transposée de 90^o du signal de référence : cos $\rightarrow$ sin :

$$A_1 \sin(\omega_1 t + \varphi) \cdot A_2 \sin(\omega_2 t) \propto A_1 \cdot A_2 \cdot \cos(\varphi) \text{ si } \omega_1 = \omega_2$$

Fondements théoriques

Traitement de signaux échantillonnés en temps discret

- expression complexe des signaux en sortie de I/Q
- problème de la **cohérence du flux** (décimation, nécessité de filtrer, transposition de fréquence)
- Xlating FIR : inutile d'avoir une source précise tant qu'on reste dans la bande

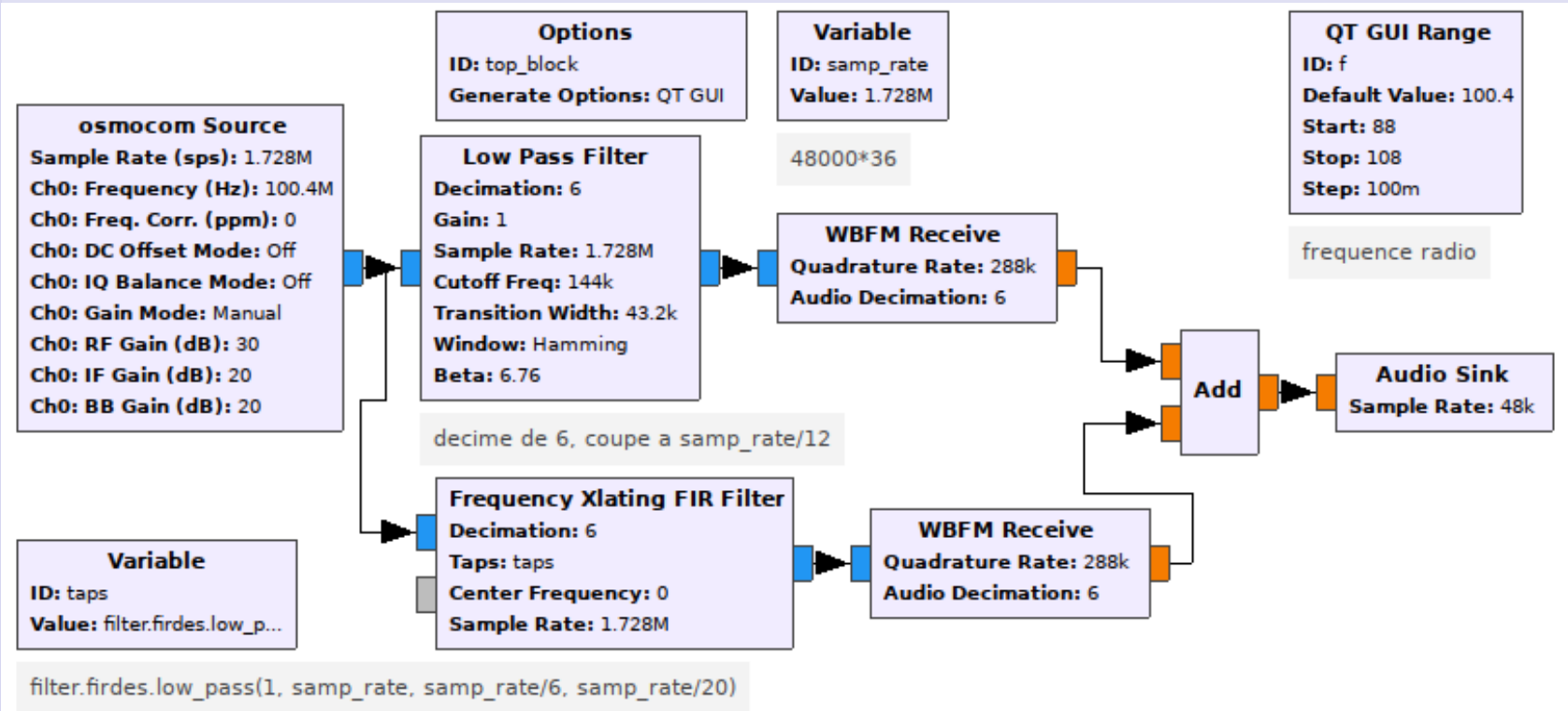


Le flux d'entrée d'un bloc doit être le flux de sortie du précédent, **tenant compte de la décimation**

Fondements théoriques

Traitement de signaux échantillonnés en temps discret

- expression complexe des signaux en sortie de I/Q
- problème de la cohérence (décimation, nécessité de filtrer, transposition de fréquence)
- Xlating FIR : inutile d'avoir une source précise tant qu'on **reste dans la bande**

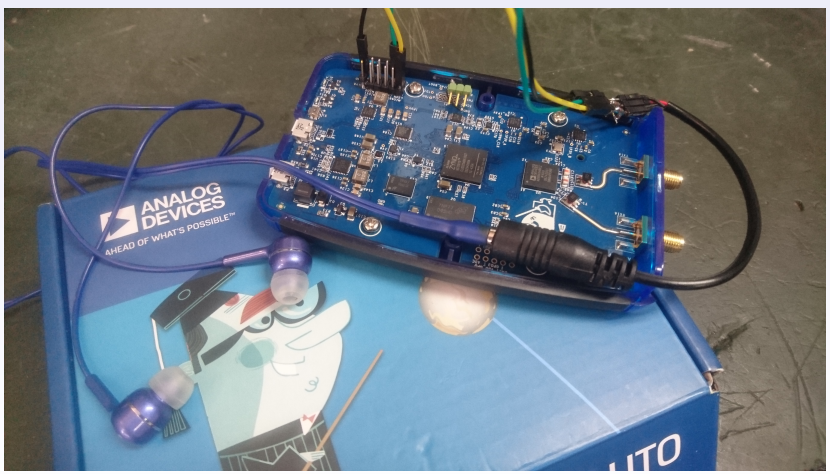


Exemple de deux bandes FM décodées simultanément

$\Rightarrow$ ajustement fin de l'écart de l'oscillateur physique par mélange avec un second oscillateur numérique, ou sauts rapides de fréquence (FHSS)

GNU Radio : un environnement libre de radio logicielle

- blocs de traitement en Python ou C++
- flux de données entre blocs défini en Python ou C++
- un environnement graphique pour assembler des blocs de traitement : gnuradio-companion (générateur de code Python ou C++)
- l'interface graphique est optionnelle : une solution **embarquable** sans interface graphique en exécutant du code Python (PlutoSDR [13])
- portage de gnuradio et ses bibliothèques à buildroot pour une utilisation sur tout système embarqué supportant GNU/Linux (Redpitaya, APF, PlutoSDR)
- au-delà des blocs de traitement existant : développer ses propres blocs (Octave → C(++) → GNU Radio)
- co-design CPU+FPGA

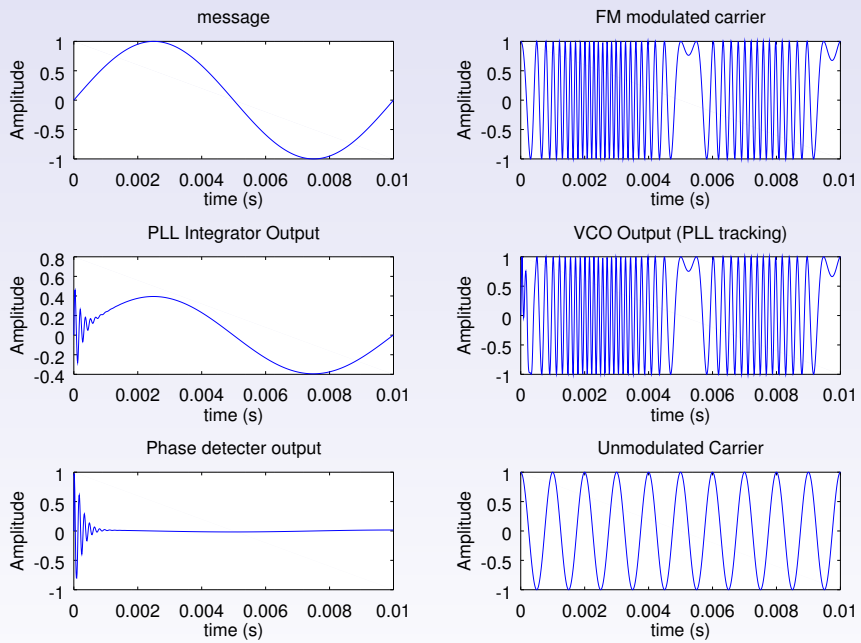


13. https://github.com/oscimp/PlutoSDR & github.com/oscimp/oscimpDigital

Traitement *numérique* du signal

Exemple de la démodulation FM (PLL)

- $y(t) = \sin\left(2\pi \cdot f_c \cdot t + 2\pi f_\Delta \int_0^t x(\tau)d\tau\right)$
- si $x(\tau) = \cos(2\pi \cdot f_m t)$:
  $y(t) = \sin\left(2\pi \cdot f_c \cdot t + \frac{f_\Delta}{f_m}\cos(2\pi f_m t)\right)$
- indice de modulation $kf = \frac{f_\Delta}{f_m}$: $kf \ll 1 \Rightarrow$NFM, $kf > 1 \Rightarrow$WBFM




```
f=1000;                 % Carrier frequency
fs=100000;              % Sample frequency
N=1001;                 % Number of samples
t=[0:1/fs:(N/fs-1/fs)];

%Create the message signal
f1=100;                 % Modulating frequency
msg=sin(2*pi*f1*t);

kf=.0628;               % Modulation index
S=exp(j*(2*pi*f*t+2*pi*kf*cumsum(msg)));   14
C=exp(j*(2*pi*f*t));% Unmodulated carrier

kp=0.15; ki=0.1;        % Loop P & I constants
phi(1)=30; e(1)=0; phd(1)=0; vco(1)=0;

for n=2:length(S)       % PLL implementation
 vco(n)=conj(exp(j*(2*pi*n*f/fs+phi(n-1))));
 phd(n)=imag(S(n)*vco(n)); % VCO x S input
 e(n)=e(n-1)+(kp+ki)*phd(n)-ki*phd(n-1);% PI
 phi(n)=phi(n-1)+e(n);      % Update VCO
end;
```

14. http://fr.mathworks.com/matlabcentral/fileexchange/24167-simple-pll-demostration/content/simple_PLL.m

Intermède mathématique : FM

En bande de base (sortie de mélangeur RF) : le signal I/Q est

$$X(t) = A \cdot \exp(j\Delta\omega \cdot t + j\varphi(t)) \text{ avec } \varphi(t) = D \int_{-\infty}^{t} m(\tau)d\tau$$

alors

$$X_n \cdot X_{n+1}^* = A \cdot \exp\left(j\Delta\omega \cdot T_s + jD \int_{n\cdot T_s}^{(n+1)\cdot T_s} m(\tau)d\tau\right)$$

et

$$\int_{n\cdot T_s}^{(n+1)\cdot T_s} m(\tau)d\tau \simeq T_s \cdot m(n \cdot T_s) \text{ (rectangle)}$$

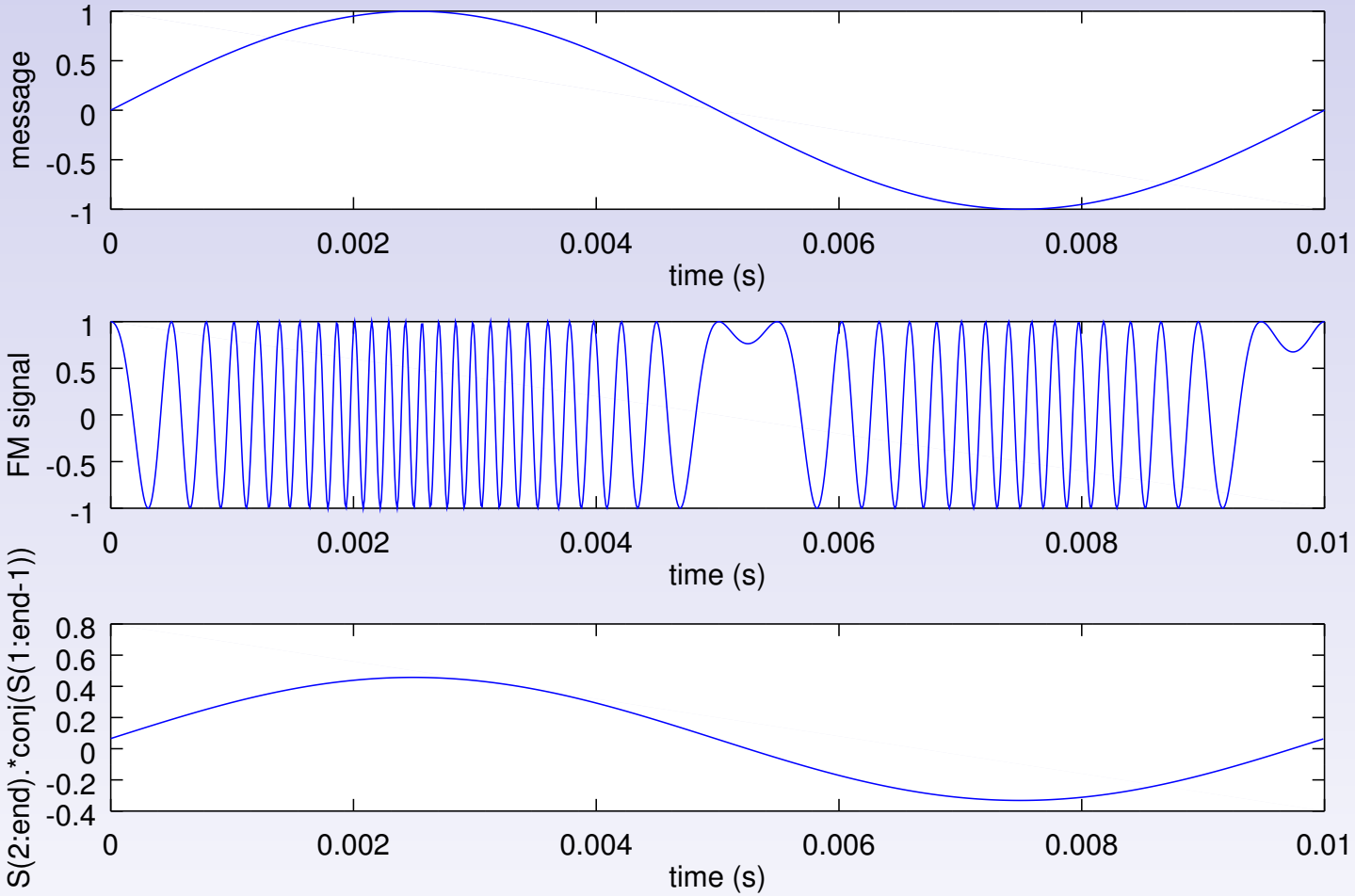
donc

$$\arg(X_n \cdot X_{n+1}^*) = \underbrace{\Delta\omega \cdot T_s}_{offset} + \underbrace{D \cdot T_s}_{gain} \cdot m(n \cdot T_s) \text{ qu'on multiplie par } \frac{1}{T_s}$$

Il s'agit du *quadrature FM demodulator* de GNURadio [15]

15. D. Bederov, *Arithmetic based implementation of a quadrature FM Demodulator*, à `https://fosdem.org/2015/schedule/event/sdr_arithmetic/`

Démodulation FM (sans PLL)



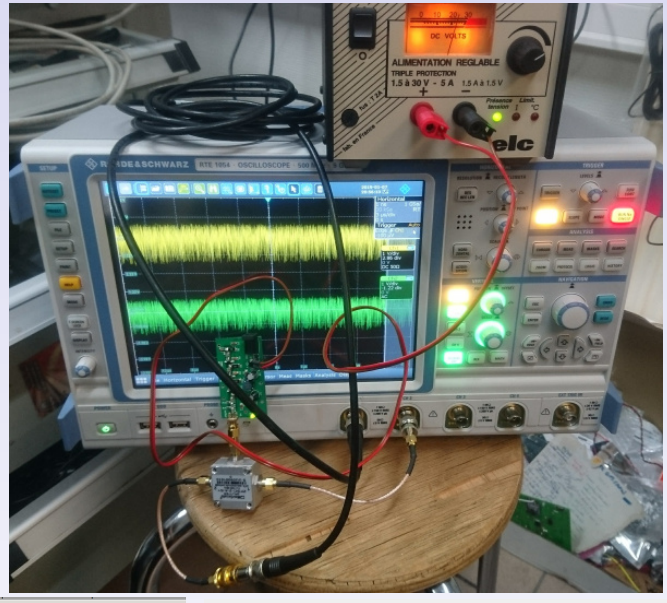

```
f=1000; fs=100000; N=1001;
t=[0:1/fs:(N/fs-1/fs)];
f1=100;                 % Modulating frequency
msg=sin(2*pi*f1*t);
kf=.0628;               % Modulation index
S=exp(j*(2*pi*f*t+2*pi*kf*cumsum(msg)));

plot(angle(S(2:end).*conj(S(1:end-1))))
```

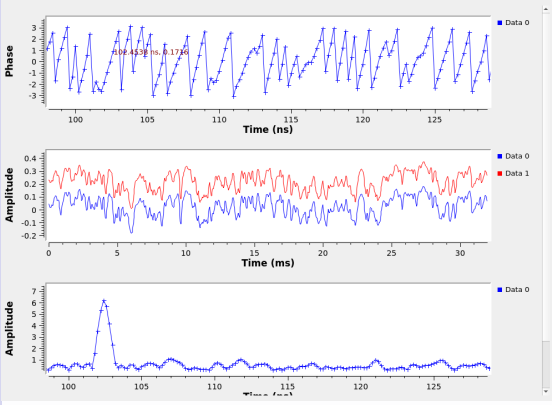
- La démodulation FM se résume en 1 ligne de code !
- Problème du calcul de `angle` qui nécessite une **fonction trigonométrique**.

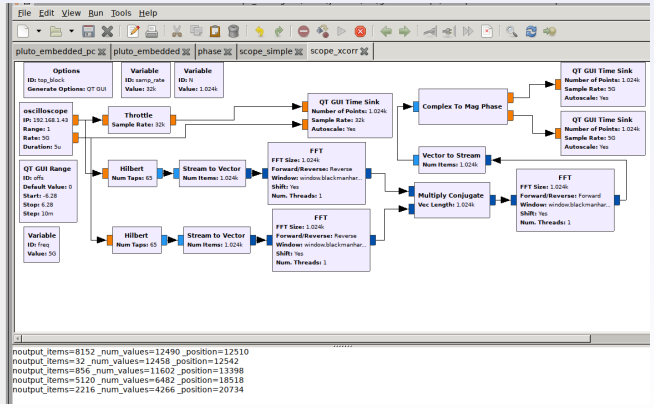
Développer ses propres blocs de traitement

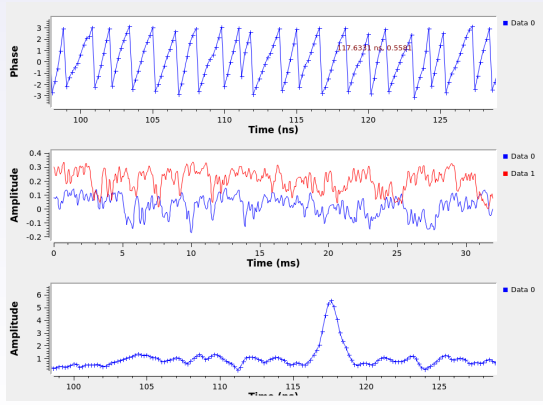
Un oscilloscope comme source de signaux radiofréquences (cas de l'application RADAR) : `github.com/jmfriedt/gr-oscilloscope`

- Exemple de la mesure de longueur d'un cable
- Une source de bruit large bande alimente une voie de référence puis une voie de mesure au travers d'un cable
- Intercorrélation entre les deux voies donne le retard pour lequel les deux voies se ressemblent
- 15 ns à 66%c_0 (200 m/μs) indique 3 m qui est bien la longueur géométrique
- résolution spatiale $c_0/(2B) = 3$ cm @ 5 GHz









Conclusion

- La radio logicielle comme une approche **pédagogique** à l'enseignement du **traitement du signal numérique** échantillonné en temps discret
- La radio logicielle comme une approche **flexible** au traitement de signaux radiofréquences
- Disponibilité d'un environnement **libre** de prototypage pour se familiariser avec les concepts
- Domaine abordable pour un coût démarrant à moins de 10 euros

- Perspectives **multiples de développements** (communications numériques ou analogiques, sécurité, caractérisation de canaux de propagation, liaisons spatiales, RADAR ...) – voir `websdr`