La Revue 3E.I
Culture Sciences de l'Ingénieur

La radio-logicielle

Jean-Pierre BARBOT [1]

Édité le 15/07/2026

école normale supérieure paris-saclay

[1] *Enseignant-chercheur à l'ENS Paris-Saclay, DER SIEN et au laboratoire SATIE (UMR 8029 CNRS / ENS Paris-Saclay)*

Cette ressource fait partie du N°120 de La Revue 3EI du 3ième trimestre 2026.

La radio-logicielle (SDR en anglais pour Software Defined Radio) est un système de transmission de l'information utilisant une onde radioélectrique comme support de transmission. Du point de vue des signaux échangés entre les antennes d'émission et de réception, rien ne distingue les signaux issus d'un système SDR de ceux d'un système radio « conventionnel », ce sont les mêmes signaux répondant aux mêmes normes. La différence fondamentale réside dans leur élaboration et dans leur traitement. Si pour les systèmes de radio usuels, ces opérations sont réalisées avec des composants analogiques : amplificateur, multiplieur, additionneur, soustracteur, filtre, boucle à verrouillage de phase, oscillateur…, elles sont effectuées numériquement dans les systèmes SDR.

Pour cela, les systèmes SDR sont composés d'une chaine d'acquisition numérique, d'une bibliothèque de programmes de traitement numérique du signal et de quelques composants analogiques permettant la réception et/ou l'émission physique de signaux hautes fréquences.

Cet article, qui vise à présenter et illustrer les principes de fonctionnement de la radio-logicielle, est organisé en quatre parties. La première partie s'attachera à présenter l'architecture matérielle permettant la réalisation physique d'un système SDR. Dans une deuxième partie, nous présenterons les composants logiciel nécessaires à sa mise en œuvre et en particulier la suite logicielle GNU Radio. Enfin, en section trois et quatre, et puisque de radio il s'agit, nous présenterons deux exemples de réalisation : l'un relatif à la radio FM stéréo, le second portant sur la radio DAB+.

1 - Radio-logicielle : les équipements

1.1 - Présentation

Les premières manifestations d'intérêt pour la radio-logicielle datent des années 1970, mais les équipements disponibles ne permettaient pas de les mettre en œuvre. Il aura fallu attendre pour cela le début des années 80 et la commercialisation des premiers DSP (DSP pour Digital Signal Processor) : le TMS320C10 de Texas Instrument n'est sorti qu'en avril 1983, approximativement en même temps que les premiers FPGA.

C'est en 1990 que le terme « software radio » apparait pour la première fois dans une publication scientifique [1]. En 1992, suite au lancement du programme de recherche européen RACE pour la définition du système de téléphonie UMTS (3G), des travaux de recherche portant explicitement sur la radio-logicielle sont menés. En effet, sachant que les normes 2G et 3G allaient coexister, l'objectif était de déterminer s'il fallait concevoir des téléphones intégrant deux équipements distincts, l'un pour la 2G et le second pour la 3G, ou si une plateforme matérielle commune,

reconfigurable et reprogrammable, pouvait s'envisager. Malheureusement, la complexité numérique requise est telle que même actuellement, cette solution reste matériellement impossible.

Toutefois, les progrès réalisés en matière de DSP, FPGA, Convertisseur Analogique-Numérique (CAN) et Numérique-Analogique (CNA) sont tels que beaucoup des systèmes de radio peuvent être réalisés à partir d'une solution radio-logicielle, et non plus seulement avec un équipement totalement analogique.

1.2 - Acquisition, traitement du signal et restitution numérique du signal

Avant d'envisager un traitement numérique, il est impératif de transformer le signal en une suite d'échantillons numériques, c'est-à-dire une suite de nombres. Pour cela on met en œuvre une chaine d'acquisition numérique du signal dont le principe est présenté figure 1.

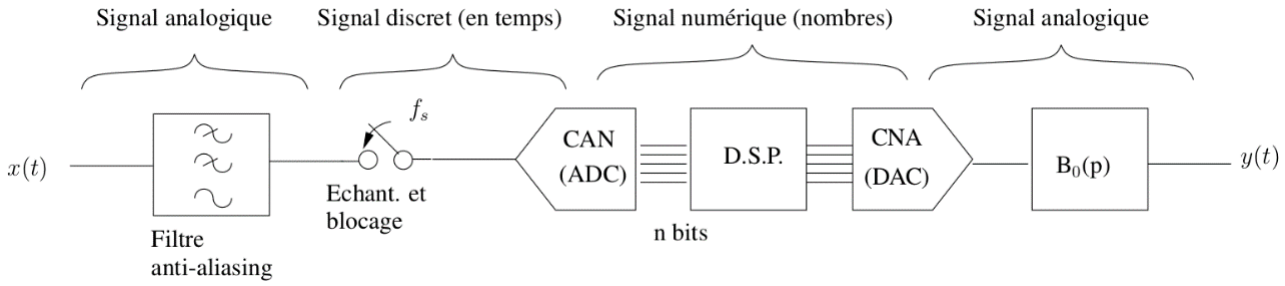


Figure 1 : Schéma d'une chaine d'acquisition numérique de l'information

Le signal d'entrée, $x(t)$, passe au préalable à travers un filtre passe-bas, garantissant que le spectre de $x(t)$ est de type passe-bas. Le signal filtré est ensuite échantillonné temporellement à la fréquence f_s, puis ces échantillons sont convertis en nombres codés sur n bits par un CAN (pour Convertisseur Analogique-Numérique). Cette série de nombres est ensuite traitée par un processeur dédié : un DSP. Dans le cas où la carte d'acquisition permet également la restitution d'un signal analogique, le DSP est suivi d'un CNA (pour Convertisseur Numérique-Analogique) et d'un bloqueur (ou interpolateur) d'ordre zéro. Le schéma de principe représenté figure 1 est typiquement celui de la carte son de la plupart des ordinateurs.

Les deux paramètres permettant de choisir une carte d'acquisition sont la fréquence d'échantillonnage, f_s, et le nombre de bits, n.

Si l'on suppose que le spectre de $x(t)$ est de type passe-bas, strictement borné par une fréquence maximale f_{max}, la fréquence d'échantillonnage, établie d'après le théorème de Shannon-Nyquist [2][3] est $f_s \geq 2f_{max}$.

Le nombre de bits n est quant à lui déterminé par le rapport signal sur bruit de quantification : $RSB_q = 6n + 1{,}8\ [dB]$.

Des fréquences d'échantillonnage allant jusqu'à $10\ GHz$ sont maintenant possibles, de même qu'un nombre de bits allant jusqu'à $n = 32$ possible (typiquement $n \leq 16$).

1.3 - Modulateur et démodulateur

Afin de pouvoir émettre une onde radioélectrique, les systèmes radio utilisent une tension de forme sinusoïdale, appelée porteuse. Appelons $e(t)$ cette porteuse avec :

$$e(t) = a(t)\cos\underbrace{\left(2\pi f_c t + \emptyset(t)\right)}_{\phi_i(t)} (1)$$

$$= \underbrace{a(t)\cos\left(\emptyset(t)\right)}_{I(t)}\cos(2\pi f_c t) - \underbrace{a(t)\sin\left(\emptyset(t)\right)}_{Q(t)}\sin(2\pi f_c t)$$

La porteuse $e(t)$ peut également s'écrire sous forme complexe [3]:

$$e(t) = Re\left(a(t).\exp\left(j\emptyset(t)\right).\exp(j2\pi f_c t)\right)$$

$$= Re\left(\left(a(t)\cos\left(\phi(t)\right) + ja(t)\sin\left(\phi(t)\right)\right).\exp(j2\pi f_c t)\right)$$

$$= Re\left(\left(I(t) + jQ(t)\right).\exp(j2\pi f_c t)\right) (2)$$

avec :

- $a(t)$ l'amplitude instantanée de la porteuse,
- $\phi_i(t)$ la phase instantanée de la porteuse,
- f_c la fréquence de la porteuse,
- $\phi(t)$ le terme de modulation angulaire,
- $I(t)$ le signal en phase avec la porteuse (porté par $\cos(2\pi f_c t)$),
- $Q(t)$ le signal en quadrature avec la porteuse (et porté par $\sin(2\pi f_c t)$)

Suivant l'équation (2), Les composantes en phase et en quadrature, $I(t)$ et $Q(t)$, peuvent se représenter dans le plan complexe, voir figure 2.

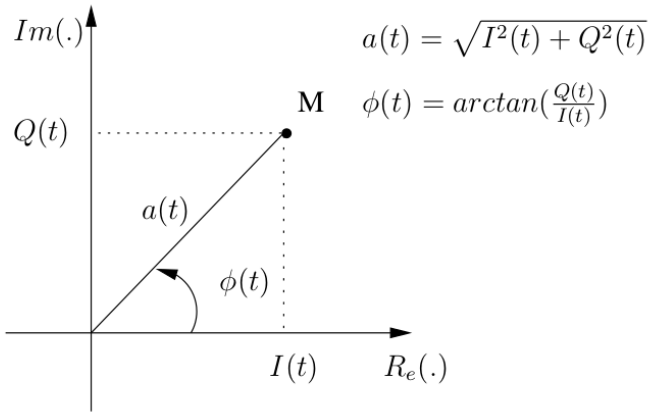


Figure 2 : Représentation dans un plan complexe des voies I(t) et Q(t)

Il est à noter que, d'après l'équation (1), $I(t)$ est portée par la porteuse $\cos(2\pi f_c t)$, alors que $Q(t)$ est portée par $-\sin(2\pi f_c t)$, la « porteuse en quadrature », du fait de son déphasage de $\pi/2$ avec $\cos(2\pi f_c t)$. Cette remarque est importante car d'un point de vue réalisation, les Oscillateurs Locaux (OL) savent produire des tensions sinusoïdales telles que $\cos(2\pi f_c t)$ et/ou $\sin(2\pi f_c t)$, mais en aucun cas la tension complexe $\exp(j2\pi f_c t)$ de l'équation (2).

La figure 3 présente un exemple de réalisation de modulateur IQ.

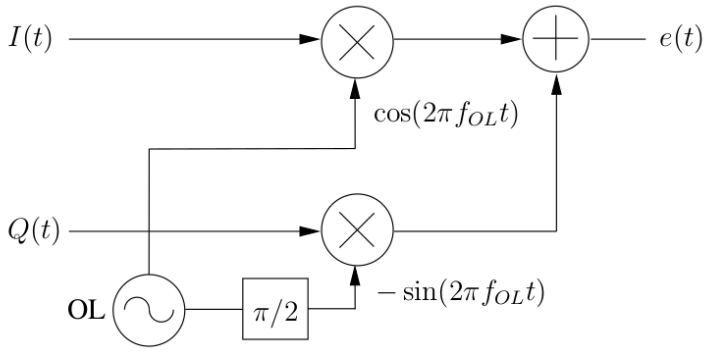


Figure 3 : Principe d'un modulateur I Q

Moduler la sinusoïde porteuse consiste à modifier son amplitude et/ou sa phase instantanée au rythme du message à transmettre. Quand ce message, dont les variations sont lentes devant f_c, modifie légèrement la porteuse, on dit qu'il module la porteuse et on l'appelle modulant [4].

Dans le cas où c'est l'amplitude instantanée $a(t)$ qui est modulée et que $\phi(t) = \phi_0$ reste constante, on parle d'une modulation d'amplitude (AM). Dans le cas où le modulant fait évoluer $\phi(t)$ et que $a(t) = a$ reste constante, on parle de modulation angulaire, modulation dont font partie la modulation de fréquence (FM) et la modulation de phase. Dans le cas des modulations MAQ, le modulant agit à la fois sur $a(t)$ et sur $\phi(t)$ [5].

La tension $e(t)$ issue de cette modulation (1) et appliquée à l'antenne d'émission, produit une onde électromagnétique de fréquence f_c dont la propagation permet le transport l'information. Cette fréquence porteuse f_c prend typiquement des valeurs comprises entre $87.5\ MHz$ et $108\ MHz$ pour un signal de norme FM (en France), $199.36\ MHz$ pour un signal DAB+ (canal 8C) et au-delà de $1\ GHz$ pour un signal de norme WiFi, 3G, 4G, 5G.

Bien qu'il soit devenu possible de générer directement $e(t)$ à partir d'un signal échantillonné et d'une conversion numérique-analogique, procéder ainsi serait inutilement coûteux. Emettre ainsi un signal WiFi dans la bande des $5\ GHz$ nécessiterait des équipements fonctionnant à $f_s \geq 10\ GHz$, ce qui du point de vue du coût n'est pas envisageable.

En revanche, les signaux $I(t)$ et $Q(t)$ de la figure 3, qui permettent de produire n'importe quel type de modulation, occupent une bande de fréquence raisonnable. Ce sont eux qui seront générés par un CAN. Par exemple, la fréquence maximale du modulant d'une modulation de fréquence avoisine $f_{max} \approx 150\ kHz$, une fréquence d'échantillonnage $f_s \geq 2f_{max} = 300\ kHz$ conviendrait pour produire $I(t)$ et $Q(t)$. La transposition en fréquence autour de la fréquence porteuse est réalisée par un multiplieur (appelé mélangeur ou mixer en anglais) et un oscillateur local (OL), comme l'illustre la figure 3.

Inversement, le signal capté par l'antenne de réception, $r(t)$, voir figure 4, pourrait être directement échantillonné et numérisé après amplification par un LNA (Low Noise Amplifier), comme proposé dans [1].

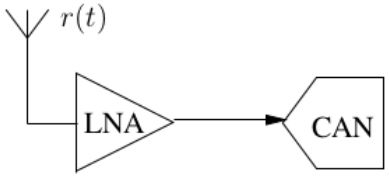


Figure 4 : Récepteur SDR théorique selon [1]

Pour les mêmes raisons et bien que technologiquement réalisable, la solution proposée figure 4 serait trop coûteuse. De fait, ce sont les signaux $I(t)$ et $Q(t)$ de la figure 5 qui seront numérisés.

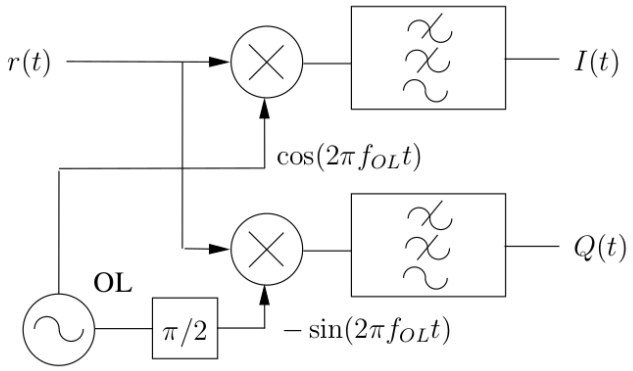


Figure 5 : Principe d'un démodulateur IQ

Les fabricants d'équipements destinés à la radio-logicielle ont adopté cette stratégie : dans le cas d'un récepteur, la carte d'acquisition numérique sera précédée par un démodulateur IQ (figure 5) et suivie d'un modulateur IQ dans le cas d'un émetteur. On notera enfin que l'oscillateur local présent dans ces équipements est un synthétiseur de fréquence ce qui permet de faire varier f_{OL} dans des plages de fréquences parfois importantes. Cette architecture est représentée figure 6.

A ce stade, il est important de souligner le caractère universel de l'architecture matérielle et logicielle adoptée par la radio-logicielle. En effet, dans un équipement analogique, les caractéristiques du modulateur comme celles du démodulateur sont figées de par leur réalisation matérielle. Dans le cas de la radio-logicielle, si la fréquence de l'oscillateur local f_{OL} est modifiable, l'équipement pourra émettre et recevoir tous types de signaux. Par exemple, le même équipement, un dongle RTL-SDR (figure 7), a permis, dans la partie 3 de cet article, de recevoir et de démoduler des signaux dans la bande FM, et des signaux de norme DAB+ dans la partie 4.

Ainsi la radio-logicielle procure une grande agilité tant à la réception qu'à l'émission. Avec des équipements adaptés et si les cartes d'acquisition de I et Q le permettent, il devient possible de tout recevoir mais également de tout émettre, pour peu que les normes en vigueur le permettent.

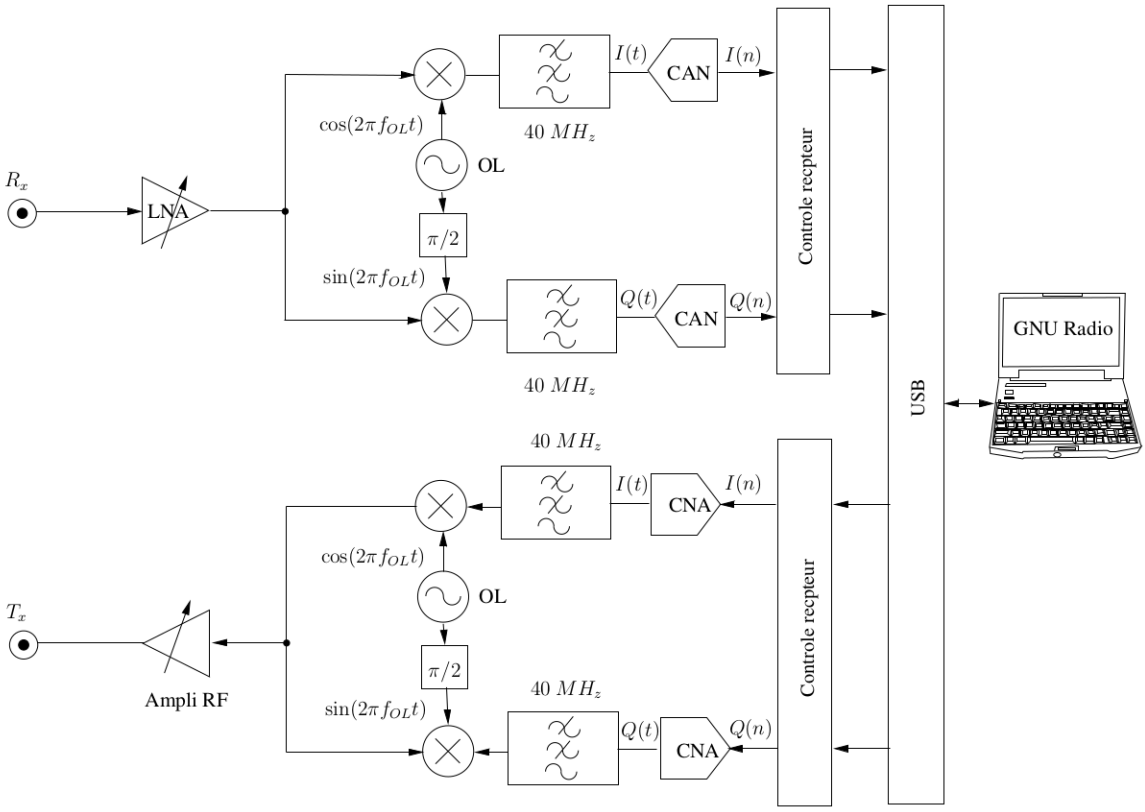


Figure 6 : Principe d'un émetteur/récepteur SDR (USRP B210, ADALM-PLUTO

1.4 - Exemples de réalisations industrielles

Les équipements de téléphonie mobile moderne fonctionnent suivant les principes de la radio-logicielle, mais leurs paramètres de réglage ne sont ni accessibles ni modifiables. En revanche, certains équipements possèdent un bus d'échange de données (généralement de type USB ou ethernet) qui permet à la fois de les paramétrer : fréquence porteuse f_c, gain des amplificateurs, fréquence d'échantillonnage f_{es}, ainsi que d'échanger des signaux numériques vers des CAN, dans le cas d'un équipement récepteur, ou des CNA dans le cas d'un dispositif émetteur.

La figure 6 schématise l'architecture d'un équipement émetteur/récepteur de type radio-logicielle, il s'agit de l'association des dispositifs représentés dans les figure 1, figure 3 et figure 5.

Très loin d'être représentative de toutes les réalisations matérielles compatibles avec la radio-logicielle, la figure 7 présente plusieurs équipements utilisés à l'ENS Paris-Saclay dans le cadre de travaux pratiques et de montages sur la radio-logicielle.

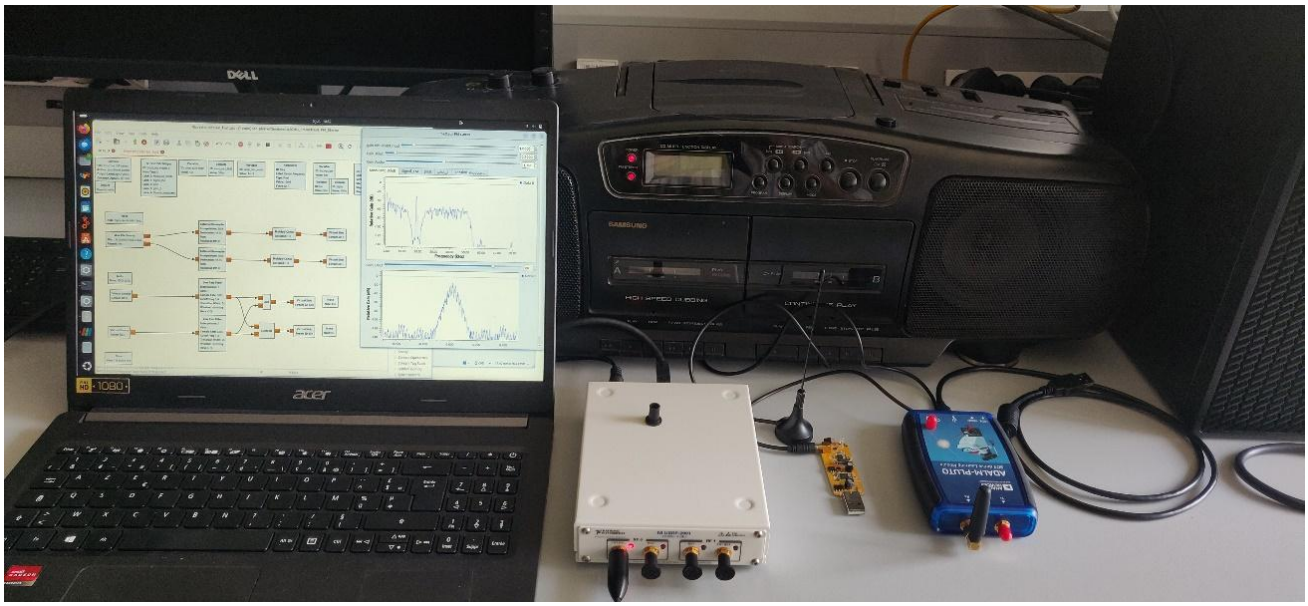


Figure 7 : Quelques équipements de type radio-logicielle utilisables en émission-réception (USRP-B210, ADALM-PLUTO) ou en réception uniquement (dongle RTL-SDR)

Le boitier USRP B210, fabriqué par ETTUS Research (National Instruments), est un émetteur-récepteur. Celui présent sur la figure 7 peut émettre et recevoir dans la bande de fréquences $f_c \in [70\ MHz\ ;\ 6\ GHz]$. Il est relié à l'unité de contrôle, dans le cas de la figure 6 à un ordinateur PC, par un câble USB.

Le dongle RTL-SDR est un récepteur fonctionnant dans la bande de fréquences $24\ MHz - 1766\ MHz$, ce qui est remarquable compte tenu de son faible prix (approximativement 20 Euros). Il se connecte également à un ordinateur PC grâce à un connecteur USB.

Enfin, le boitier émetteur-récepteur ADAL-PLUTO Rev C, d'Analog-Devices, permet d'émettre et de recevoir un signal dans la bande $345\ MHz - 3{,}8\ GHz$. Il est relié à un ordinateur PC à l'aide d'un câble USB.

Les caractéristiques principales de ces 3 équipements sont résumées dans le tableau 1.

		f_c	Bp (MHz)	f_s(MS/s)	CAN	CNA	Ps (dBm)
USRP-B210	Rx	$70\ MHz\ ;\ 6\ GHz$	56	61,44	$12\ bits$	$12\ bits$	
	Tx	$70\ MHz\ ;\ 6\ GHz$	56	61,44	$12\ bits$	$12\ bits$	$> 10\ dBm$
RTL-SDR	Rx	$24\ MHz\ ;\ 1{,}766\ GHz$	2,4	3,2	$8\ bits$		
ADALM-PLUTO	Rx	$345\ MHz\ ;\ 3{,}8\ GHz$	20	61,44	$12\ bits$	$12\ bits$	
	Tx	$345\ MHz\ ;\ 3{,}8\ GHz$	20	61,44	$12\ bits$	$12\ bits$	$7\ dBm$

Tableau 1 : Caractéristiques des équipements USRP-B210, RTL-SDR et ADALM-PLUTO (Rx pour Receiver, Tx pour Transmeitter)

Il est à noter que ces 3 équipements, d'architectures voisines, réalisent en réception une transposition en Fréquence Intermédiaire (FI) de valeur nulle. Ainsi, le signal capté est directement transposé en bande de base, c'est-à-dire autour de $f = 0\ Hz$, sans passage en FI.

2 - Radio-logicielle : la programmation

2.1 - Principes

Les avancées technologiques en matière d'électronique numérique sont telles que remplacer les composants analogiques des équipements radio par des fonctions numériques est devenu possible, tout du moins en grande partie. Mais cette apparente simplicité de mise en œuvre ne doit pas masquer la réelle difficulté qu'impose un tel changement de paradigme : il faut exécuter un programme animant tout cet ensemble, et donc, en premier lieu, il faut l'écrire.

Dans le cas d'un récepteur de type radio-logicielle, il faut configurer les équipements analogiques de réception (amplificateurs, oscillateurs local) à l'aide d'un bus informatique, fixer la fréquence d'échantillonnage, acheminer les flux de données échantillonnées vers des fonctions de traitement numérique du signal et enfin transférer leurs résultats vers des cartes dédiées, telle qu'une carte son d'ordinateur par exemple. De tels programmes sont souvent écrits en langage C/C++, en Python, voire en assembleur pour certains DSP. Ils intègrent les drivers des différents équipements matériels ainsi que des bibliothèques de programmes mettant en œuvre des algorithmes de traitement numérique du signal. Il y a peu, l'écriture et la mise au point de tels exécutables étaient réservées à des spécialistes en informatique et en électronique numérique.

Fort heureusement, des environnements de développement graphiques ont permis d'alléger la programmation de tels systèmes. Les équipements matériels d'entrée sortie, les fonctions mathématiques, les outils de visualisation (oscilloscope, analyseurs de spectres, ...) se transforment en composants logiciels faciles à paramétrer et à associer. Ainsi, la connectivité entre équipements devient intuitive et la programmation informatique quasi-transparente.

Parmi ces outils de développement graphique et permettant de réaliser une « radio-logicielle », nous pouvons citer Simulink, fonctionnant avec MATLAB, LabVIEW de National Instruments et GRC (pour GNU Radio Companion) fonctionnant avec GNU Radio. C'est l'outil GRC (et GNU Radio) que nous présenterons et utiliserons dans la sous-section suivante.

2.2 - GNU Radio

Le projet GNU est un projet informatique dont les premiers travaux ont été réalisés en janvier 1984 par Richard Stallman afin développer le système d'exploitation GNU. Le projet GNU perdure et développe maintenant des logiciels libres, en plus de systèmes d'exploitation tel que GNU/Linux.

Dédié à la radio-logicielle, GNU Radio est une boîte à outils de développement regroupant des logiciels libres. Cette boîte à outils contient des blocs logiciels de traitement du signal destinés à la réalisation de système radio. GNU Radio peut être utilisé avec du matériel radiofréquence (RF) externe afin de créer des radios logicielles, ou sans matériel afin de réaliser des simulations.

A cette boîte à outils est associée l'environnement de développement graphique GNU Radio Companion (GRC), permettant d'assembler simplement ces blocs. L'éditeur graphique GRC est écrit en Python. Les blocs de calcul de GNU Radio sont en revanche majoritairement écrits en C/C++, ce qui leur confère une plus grande rapidité de calcul. Lorsque le flowgraph (voir figure 12) est exécuté, GRC génère automatiquement un script Python : *top_block.py*, sauvegardé dans le répertoire de travail, qui instancie les blocs et lance le traitement. Il est à noter qu'il est possible d'exécuter directement *top_block.py* avec Python sans avoir à réouvrir GRC.

L'étape préliminaire à toute mise en œuvre pratique est bien évidemment l'installation de GNU Radio et de GRC. Le site https://www.gnuradio.org détaille comment procéder. Un tutoriel, dédiée à l'apprentissage de GRC est accessible sur le site wiki.gnuradio.org https://wiki.gnuradio.org/index.php/Guided_Tutorial_GRC. Il est à signaler que l'utilisation de GRC est très intuitive et les paramétrages qu'il permet simples à effectuer.

A l'ouverture de du logiciel GRC apparait l'environnement de développement graphique affiché figure 8.

Un premier bloc, en haut à gauche, appelé « Options » permet de fixer les options et les informations générales relatives au projet : titre, auteur, interface graphique utilisateur (ici QT). Le second bloc : « Variable », permet de fixer la variable dont l'identifiant (ID) est *samp_rate*. Cette variable est la fréquence d'échantillonnage à laquelle seront cadencés, par défaut, l'ensemble des blocs du projet. A l'ouverture de tout nouveau projet, *samp_rate* vaut $32\ kHz$, mais cette variable est bien sur modifiable. La variable *samp_rate* apparait comme paramètre par défaut dans chacun des composants logiciels. Suivant la fréquence à laquelle doit être cadencée ce composant, si nécessaire le paramètre correspondant peut-être modifié en imposant une valeur numérique ou par le biais d'une opération du type *samp_rate/4,* par exemple. Le bandeau vertical situé à droite regroupe l'ensemble des composants logiciels utilisables dans le projet. Ces composants logiciels sont regroupés par thématique : *Instrumentation, IQ Correction, Coding, Filters, etc*. Il est à noter qu'il est possible d'ajouter de nouveaux composants aux composants présents lors de l'installation de GNU Radio. Par exemple, dans la quatrième partie de cet article, nous avons importé le package *gr_dab* afin de réaliser un récepteur radio-logicielle DAB+.

Comme nous l'avons mentionné auparavant, deux types de projets peuvent être mis en œuvre. Dans un premier cas, il peut s'agir d'une simulation pure : les signaux d'entrée sont générés numériquement puis sont traités par de outils de traitement du signal et/ou affichés, avec les composants logiciels de QT GUI du framework GNU QT, par exemple. Une seconde possibilité, la plus motivante, consiste à utiliser des équipements matériels externes tels que, pour les applications radio, les matériels radiofréquence (RF) apparaissant figure 7.

Commençons par un projet de simulation. L'objectif de ce projet est a priori simple : relier un générateur de signaux à un oscilloscope et un analyseur de spectre. Pour cela il faut sélectionner

le générateur de signaux parmi tous les composants logiciels présents (voir tutoriel GRC sur la page wiki de GNU Radio : ce générateur s'appelle *Signal Source*. Une fois positionné dans l'espace de travail la page de projet, il faut sélectionner puis positionner l'oscilloscope : *QT GUI Time Sink* et l'analyseur de spectre *QT GUI Frequency Sink*.

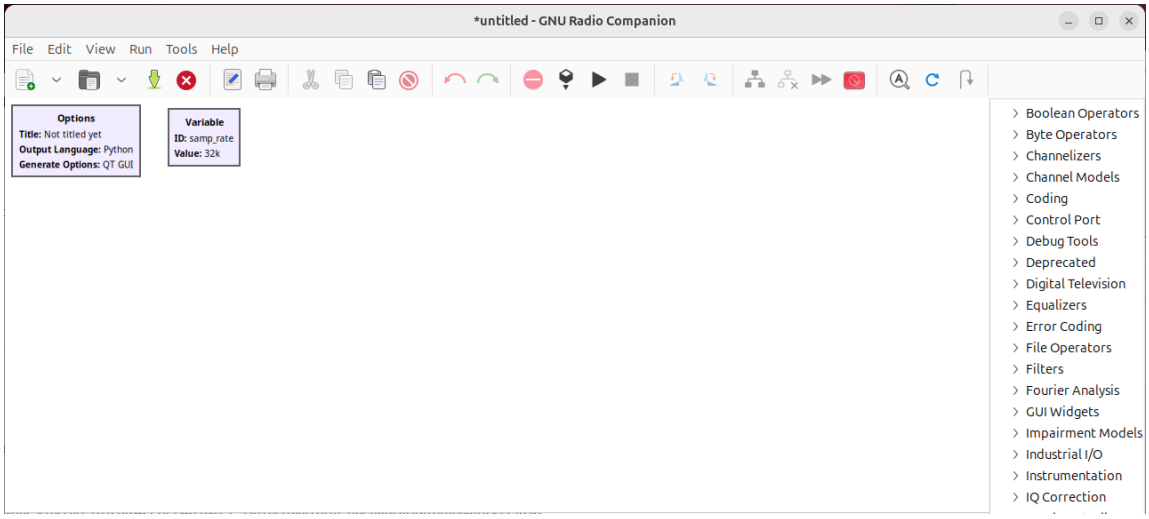


Figure 8 : Principe d'un modulateur IQ et d'un démodulateur IQ

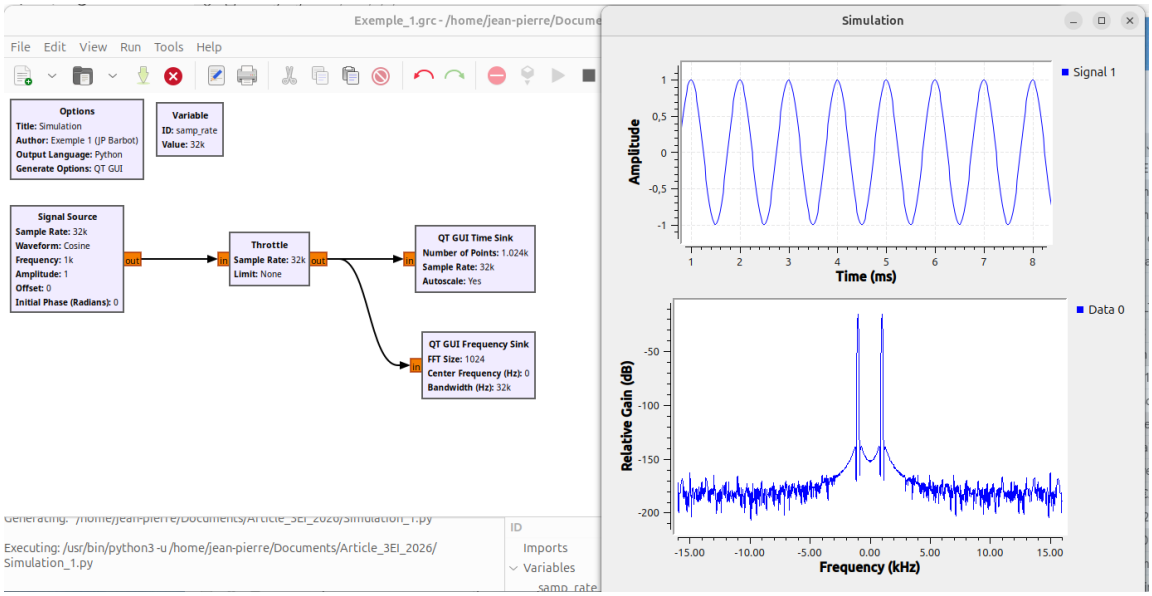


Figure 9 : Simulation d'un générateur de fonction relié à un oscilloscope et un analyseur de spectre à FFT (cas d'une sinusoïde réelle)

Le type des données par défaut de ces 3 blocs est « complex ». Pour obtenir le même résultat de simulation que figure 7, il convient de modifier les types de ces 3 blocs, en remplaçant « complex » par « float ». L'exécution du flowgraph fait apparaitre deux fenêtres de visualisations : l'une correspondant à l'oscilloscope et la seconde à l'analyseur de spectre. Ces affichages sont cohérents avec les paramètres du générateur de signal « signal source », à savoir une fonction cosinus de fréquence $32\ kHz$, d'amplitude $1\ Volt$ échantillonnée à $f_e = 32\ kHz$.

A noter sur la figure 9 la présence du bloc « throttle ». S'agissant d'une simulation, l'utilisation de ce bloc est nécessaire afin de préserver la puissance de calcul du processeur de votre ordinateur, voir la documentation de GRC pour plus de détail.

On constate figure 9 que le signal est bien de forme sinusoïdale, de fréquence à $f_0 = 1\ kHz$ comme en attestent l'oscilloscope et l'analyseur de spectre. On illustre ainsi la propriété des signaux réels d'avoir un spectre symétrique en fréquence par rapport $f = 0\ Hz$.

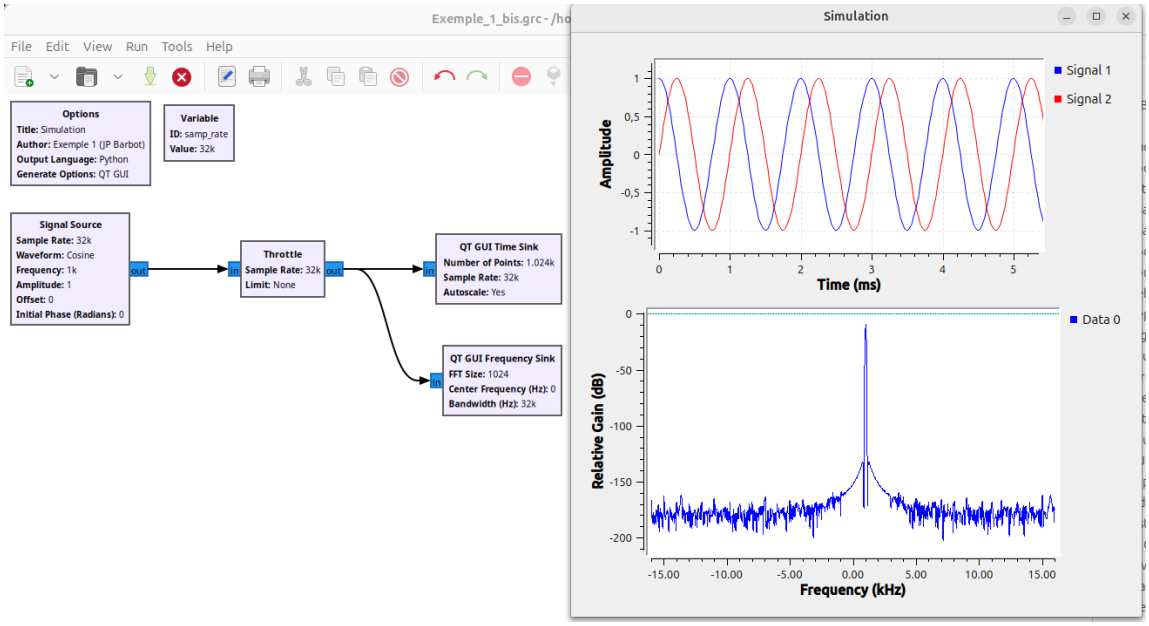


Figure 10 : Simulation d'un générateur de fonction relié à un oscilloscope et un analyseur de spectre à FFT (cas d'une sinusoïde complexe)

Toujours dans le cadre d'une simulation, le signal généré figure 10 est maintenant de type complexe. C'est donc un signal de la forme $\exp(j2\pi f_0 t) = \cos(2\pi f_0 t) + jsin(2\pi f_0 t)$ avec $f_0 = 1\ kHz$ qui est simulé, d'où l'apparition de 2 sinusoïdes sur l'oscilloscope et d'une raie unique à $f_0 = 1\ kHz$ sur l'analyseur de spectre [3].

Le second projet met en œuvre un dongle RTL-SDR (voir sous-section 1.4), c'est à dire un récepteur RF externe. Pour cela, ce dongle, muni de son antenne, est connecté à un des ports USB de l'ordinateur. Dans GRC, ce composant apparait sous 2 formes : « RTL SDR Source » et « Soappy RTLSDR Source ». Nous avons sélectionné la seconde et nous l'avons reliée à un analyseur de spectre *QT GUI Frequency Sink*. Le paramètre « Center Freq » du dongle a été paramétré à $f_{OL} = 199.36\ MHz$, fréquence centrale du canal 8C de la radio DAB+. Afin de pouvoir visualiser entièrement de la densité spectrale du signal DAB+, qui occupe une bande de fréquence de $1536\ kHz$, la variable « samp_rate » a été fixée à $2\ MHz$, ce qui est possible avec ce composant (voir tableau 1).

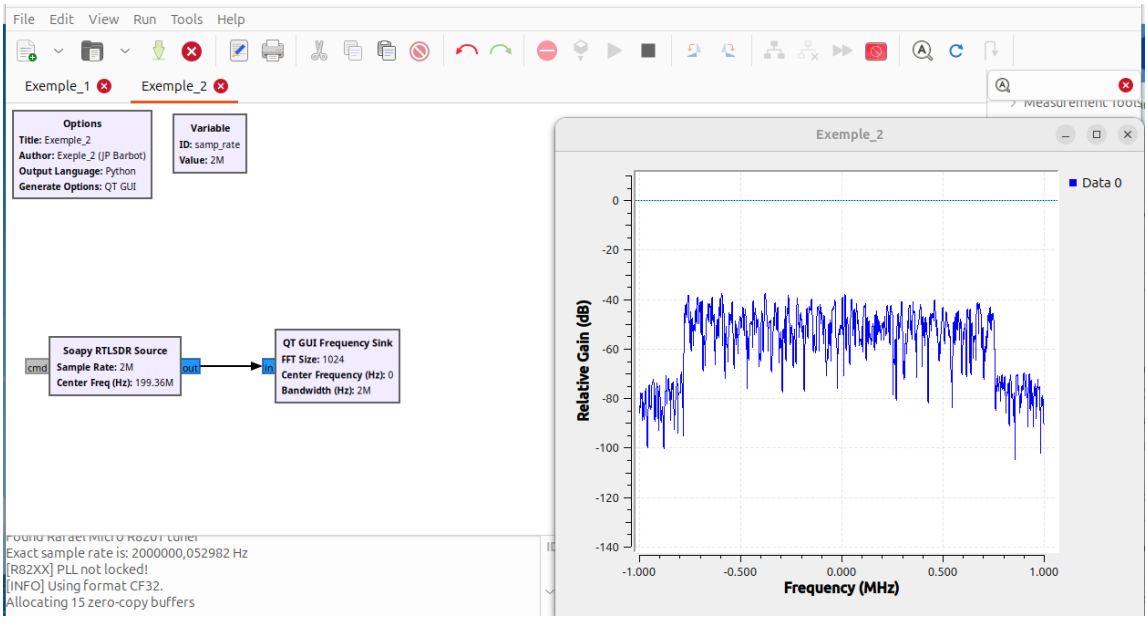


Figure 11 : Principe d'un modulateur IQ et d'un démodulateur IQ

Le flowgraph et le résultat de son exécution apparaissent figure 11. On observe un spectre de forme rectangulaire, typique des modulations numériques OFDM [9] et occupant une largeur de bande de $1536\ kHz$ [10]. Ce spectre est de plus centré sur $f = 0\ Hz$ car $f_{OL} = 199.36\ MHz$.

Ces deux exemples de mise en œuvre de GRC démontrent l'intérêt pédagogique de la radio-logicielle. Il devient en effet simple et quasi-immédiat d'illustrer par des simulations et des mesures sur signaux réels des concepts théoriques compliqués tels que la représentation en enveloppe complexe, le filtrage numérique, la modulation numérique.

Pour approfondir plus avant le concept de radio-logicielle, les références [6], [7], [8] sont d'une grande aide.

3 - Récepteur radio-logicielle FM stéréo

C'est avant tout pour réaliser des radios que la radio-logicielle a été conçue. Nous allons donc dans cette partie mettre en œuvre un récepteur de radio FM.

Pour ce faire nous utiliserons un dongle RTL-SDR comme récepteur et GNU Radio pour opérer les actions de démodulation, de filtrage pour sélectionner la bande de fréquence contenant le signal audio mono (figure 14) et la restitution sonore du signal par la carte audio de l'ordinateur.

Analysons le flowgraph GRC apparaissant figure 12. La variable « samp_rate » a été fixée à $f_s = 1\ MHz$. Le dongle RTL-SDR est paramétré par le bloc « Soappy RTLSDR Source » et le paramètre « Center Freq » du dongle a été fixé à $f_{OL} = 87.8\ MHz$. La sortie du dongle est de type « complex ». En effet, c'est par le traitement du signal $\phi(t)$ apparaissant dans (1) que la démodulation est effectuée. Ainsi ce signal est représenté numériquement sous la forme d'un nombre complexe : $\big(I(t) + jQ(t)\big)$, voir équation (2).

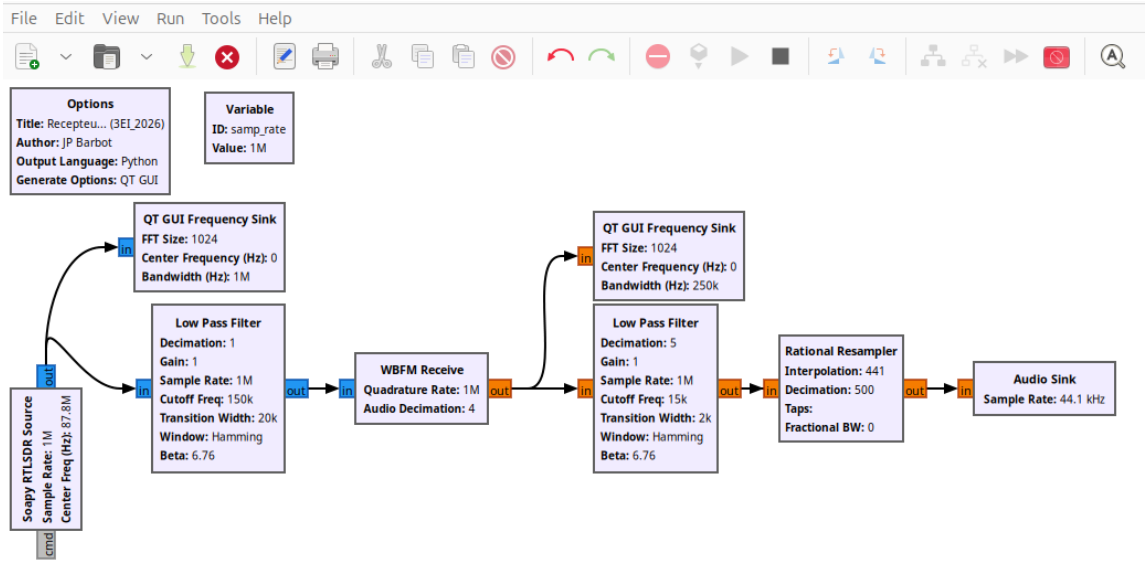


Figure 12 : Flowgraph d'un récepteur FM mono de type radio-logicielle

Ce signal est visualisé avec analyseur de spectre *QT GUI Frequency Sink* et appliqué à l'entrée d'un filtre passe-bas. La fréquence de coupure de ce filtre numérique de type FIR est réglée par « Cutoff Freq » à $150\ kHz$ et sa bande de transition par « Transition Width » à $20\ kHz$. Après ce filtrage, le signal est appliqué à l'entrée du bloc démodulateur « WBFM Receive » qui réalise la démodulation en fréquence. Ce démodulateur est un démodulateur en quadrature dont le principe de son algorithme sera décrit ci-après. Il est à noter qu'après démodulation, le signal de sortie de ce bloc est de type « float » et qu'une décimation « audio decimation » d'un facteur 4 est effectuée. Cette

décimation consiste à ne conserver qu'un échantillon temporel sur 4, faisant ainsi passer la fréquence d'échantillonnage « Samp_Rate » de $1\,MHz$ à $f_s = 250\,kHz$. La bande de fréquence contenant le signal audio mono étant de $15\,kHz$, voir figure 14, il suffit de sélectionner cette bande de fréquence avec un filtre FIR de type passe-bas dont la fréquence de coupure « Cutoff Freq » est paramétrée à $15\,kHz$ et la « Transition Width » à $2\,kHz$. Enfin, il suffit de connecter ce signal filtré à la carte son de l'ordinateur. Cette opération est réalisée par le bloc « Audio Sink » dont le paramètre « Samp_Rate » est fixé à $44.1\,kHz$.

Les paramétrages de ce flowgraph effectués, il ne reste plus qu'à l'exécuter.

Le résultat obtenu suite à l'exécution du flowgraph figure 12 est tracé figure 13. Le graphique du bas représente le spectre du signal modulé en fréquence présent à $87.8\,MHz$ et le graphe du haut ce même signal démodulé. Il est intéressant de constater que l'on peut clairement identifier dans ce « Signal FM démodulé » le pilote en fréquence à $19\,kHz$, ainsi que les composantes spectrales du signal RDS, tels que les décrivent la page Radio FM de wikipedia dont la figure 14 est extraite.

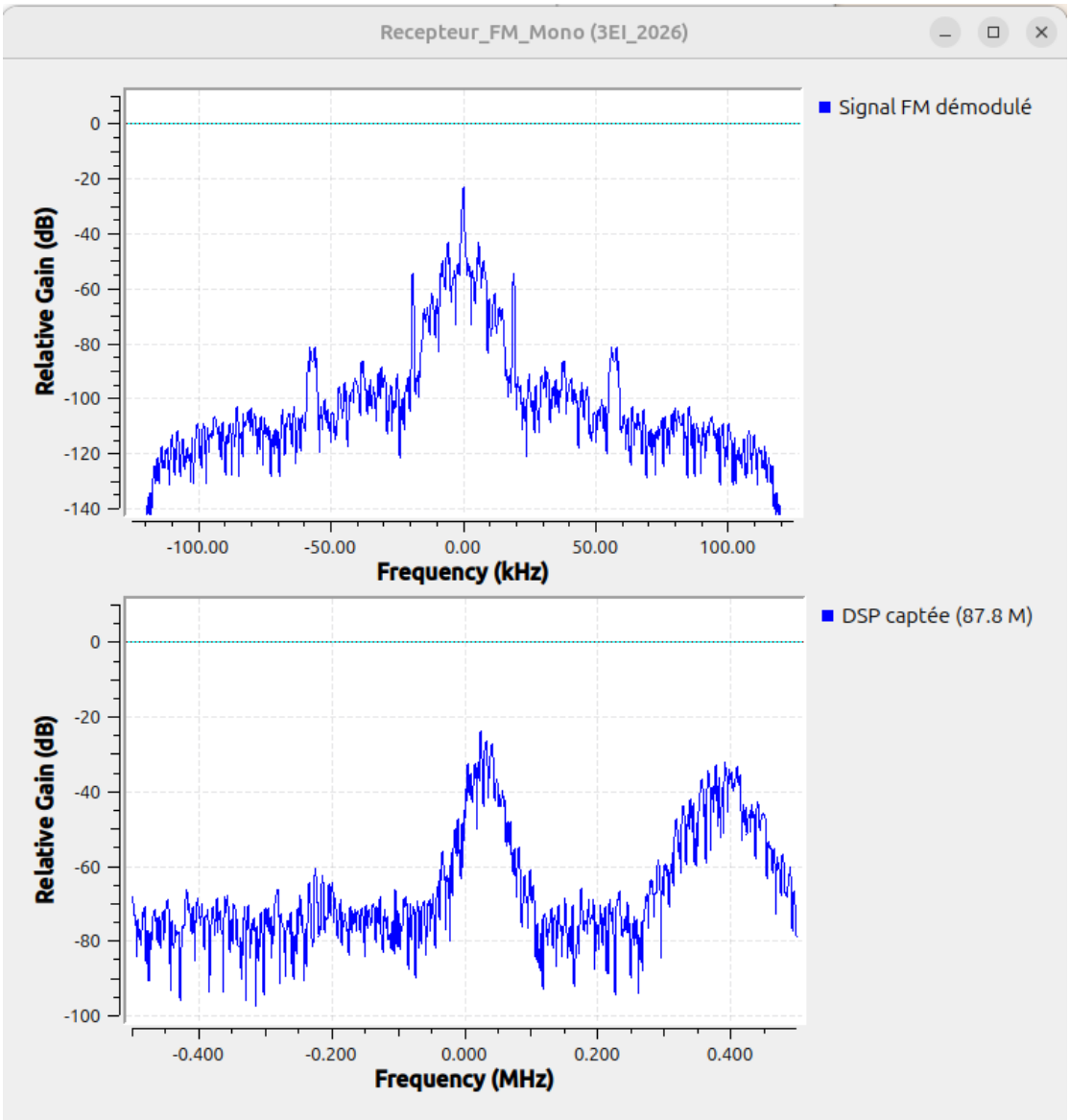


Figure 13 : Spectre d'un signal modulé en fréquence (FM stéréo) capté à $87.8\,MHz$, graphique du bas, et démodulé, graphique du haut

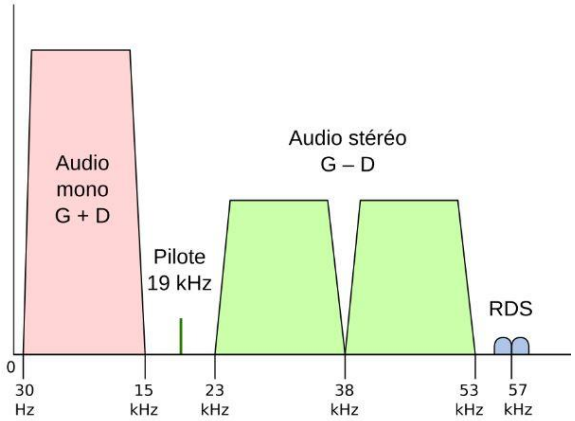


Figure 14 : Principe d'un modulateur IQ et d'un démodulateur IQ

Mais bien plus que la satisfaction de retrouver des signaux tels que prévus par la norme FM, ce qui, plus encore, est enthousiasmant pour les étudiants comme pour leurs enseignants, c'est de pouvoir entendre le résultat de ces traitements.

La simplicité de mise en œuvre du bloc « WBFM Receive » peut sembler déconcertante. L'algorithme utilisé dans ce bloc est codé en C/C++ et les équations du traitement du signal exploitées sont à retrouver dans les commentaires de l'entête du prototype « quadrature_demod_cf.h ». Un extrait de cette documentation est présenté ci-dessous, après une remise en forme car écrite avec le traitement de texte LaTex.

(extrait de `quadrature_demod_cf.h`)

This can be used to demod FM, FSK, GMSK, etc. The input is complex baseband, output is the signal frequency in relation to the sample rate, multiplied with the gain.

Mathematically, this block calculates the product of the one-sample delayed input and the conjugate undelayed signal, and then calculates the argument of the resulting complex number: $y[n] = \arg\left(x[n]\,\bar{x}[n-1]\right)$.

Let x be a complex sinusoid with amplitude $A > 0$, (absolute) frequency $f \in \mathbb{R}$ and phase $\phi_0 \in [0; 2\pi]$ sampled at

$f_s > 0$ so, without loss of generality, $x[n] = Ae^{j2\pi(\frac{f}{f_s}n+\phi_0)}$

then

$$\begin{aligned}
y[n] &= \arg\left(Ae^{j2\pi\left(\frac{f}{f_s}n+\phi_0\right)}\overline{Ae^{j2\pi\left(\frac{f}{f_s}(n-1)+\phi_0\right)}}\right)\\
&= \arg\left(A^2e^{j2\pi\left(\frac{f}{f_s}n+\phi_0\right)}e^{-j2\pi\left(\frac{f}{f_s}(n-1)+\phi_0\right)}\right)\\
&= \arg\left(A^2e^{j2\pi\left(\frac{f}{f_s}n+\phi_0-\frac{f}{f_s}(n-1)-\phi_0\right)}\right)\\
&= \arg\left(A^2e^{j2\pi\left(\frac{f}{f_s}n-\frac{f}{f_s}(n-1)\right)}\right)\\
&= \arg\left(A^2e^{j2\pi\left(\frac{f}{f_s}(n-(n-1))\right)}\right)\\
&= \arg\left(A^2e^{j2\pi\frac{f}{f_s}}\right)\\
&= \arg\left(e^{j2\pi\frac{f}{f_s}}\right) \text{ because } A \text{ is a real}\\
&= \frac{f}{f_s}
\end{aligned}$$

Signalons qu'il s'agit là d'un des blocs démodulateur fournis par GNU Radio pour la démodulation de signaux modulés en fréquence.

Afin de ne pas alourdir cette partie, nous avons volontairement limité cet exemple à celui d'une démodulation mono. Une démodulation stéréo est bien sûr possible et fait l'objet de travaux pratiques au DER SIEN (ex. département EEA) de l'ENS Paris-Saclay.

4 - Récepteur radio-logicielle DAB+

La norme DAB (pour Digital Audio Broadcasting) [10] a été déployée en France à partir de 2013 sous le nom initial RNT (Radio Numérique Terrestre).

Le principe de modulation mis en œuvre par ce système radio est de type OFDM, pour Orthogonal Frequency Division Multiplexing. Il s'agit d'une technique de modulation numérique à haute efficacité spectrale exploitée par l'ensemble des systèmes de télécommunications numériques modernes : 4G, 5G, WiFi, DVB-T, DAB+. Les fondements théoriques de l'OFDM sont « fort élégamment » exposés dans [9].

Pour mettre en œuvre la réception et la démodulation de signaux DAB+, il conviendra d'adjoindre le package *gr-dab* à GNU-Radio, ce package n'en fait pas partie par défaut.

La construction du récepteur DAB+ avec GNU-Radio est représentée figure 15. Le flowgraph établi par GRC fait de nouveau appel au dongle SDR-RTL. La variable « samp_rate » a été fixée à $f_s = 2.048\ MHz$ et la fréquence de réception du dongle RTL-SDR a été fixée par « Center Freq » à $f_{OL} = 199.36\ MHz$.

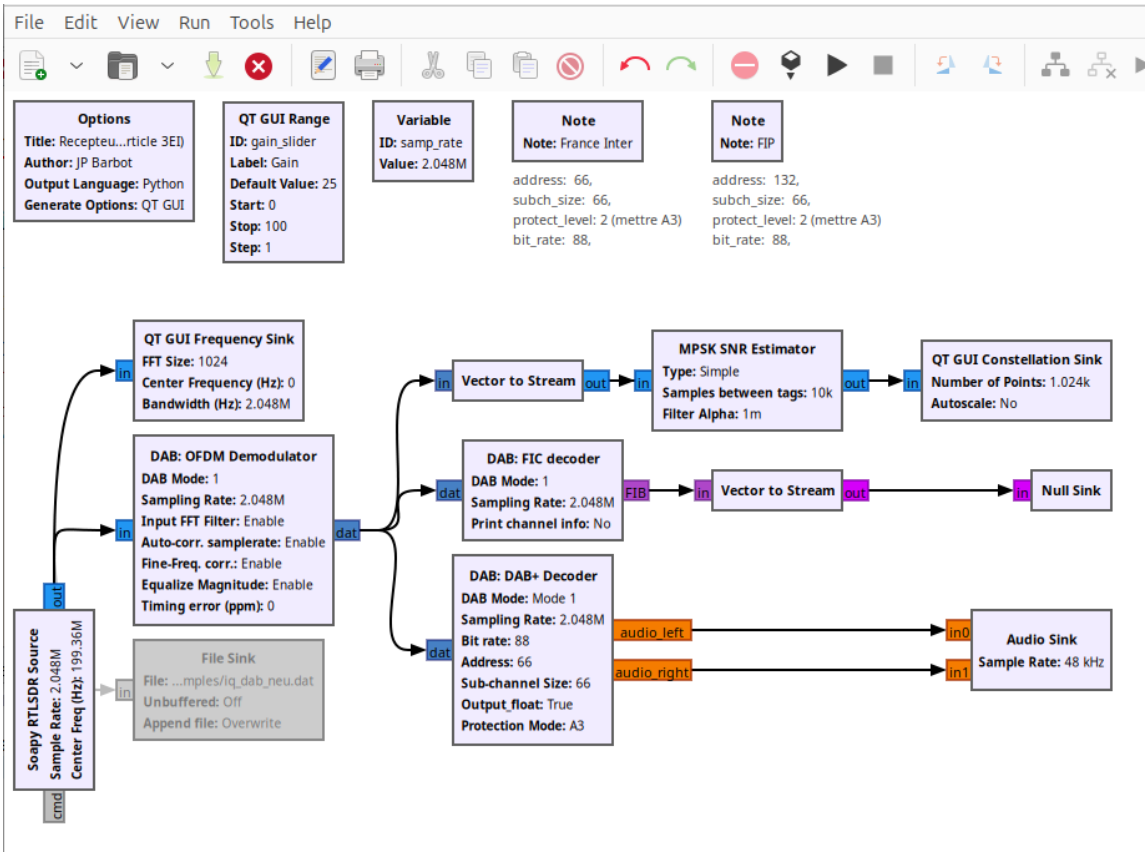


Figure 15 : Principe récepteur DAB+ implémenté avec GNU Radio

L'exécution de l'ensemble de ces blocs permet de visualiser le spectre d'un signal DAB+, voir figure 16, graphique du haut. Ce spectre est constitué de 1536 sous-porteuses, espacées de $1\ kHz$, le contenu spectral est donc de largeur $1.536\ MHz$, comme observé [10]. Chacune de ces sous-porteuses est modulée par une modulation de type MAQ-4 [5]. La constellation correspondante, pour une de ces 1536 sous-porteuses est tracée figure 16, graphique du bas. Le décodeur DAB+ « DAB : DAB+ Decoder » est relié figure 15 au bloc « Audio Sink ». L'exécution de ce flowgraph, en plus des tracés de la figure 16, permet une écoute en stéréo du programme radio diffusé.

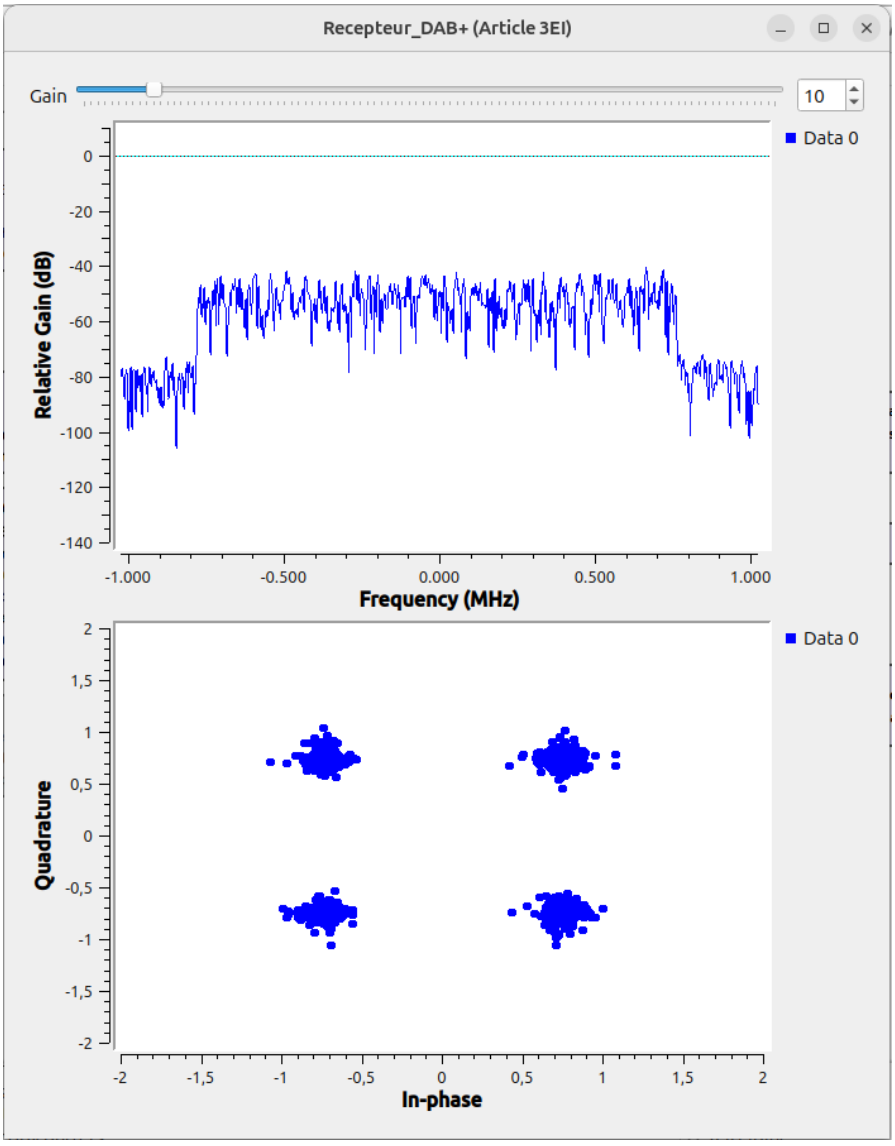


Figure 16 : Principe d'un modulateur IQ et d'un démodulateur IQ

5 - Conclusion

La radio-logicielle a bénéficié des avancées de l'électronique numérique, tant du point de vue matériel (DSP, FPGA, CAN, CNA), que logiciel. Dans un système radio, remplacer en grande partie les fonctions réalisées avec des composants analogiques par des fonctions numériques est devenu possible.

De plus, l'architecture matérielle et logicielle adoptée par la radio-logicielle lui confère un caractère universel. Un même équipement pourra permettre la réception (voire l'émission) de signaux FM et de signaux DAB+, comme l'illustrent les parties 3 et 4 de cet article. En cas d'émission, signalons que la simplicité de mise en œuvre ne doit pas occulter les réglementations en vigueur dans ce domaine.

Les exemples de réalisation basés sur la suite logicielle GNU Radio sont multiples et s'étendent bien au-delà de la radio FM et DAB+, citons par exemple la réception et le décodage de signaux radio ADS-B, émis par les transpondeurs des avions, et utilisés par exemple par le site internet flightradar24.com.

La communauté française a été pionnière dans le développement de la radio-logicielle et reste très active dans le domaine. Certains exemples de réalisation sont à retrouver dans [6], [7], [8], [11], [12], tant du point de vue théorique que pratique.

D'un point de vue pédagogique enfin, la radio-logicielle démontre que « rien n'est plus pratique qu'une belle théorie ». La mise en pratique de concepts parfois arides issus du traitement du signal, de la théorie du codage et des communications numériques pour réaliser un système de télécommunication aide à leur compréhension, et peut susciter l'envie de les explorer plus avant.

Références :

[1] J. Mitola, "Software radios-survey, critical evaluation and future directions," *[Proceedings] NTC-92: National Telesystems Conference*, Washington, DC, USA, 1992, pp. 13/15-13/23.

[2] Hélène Horsin Molinaro, Eric Vourc'h, Jean-Pierre Barbot, "Capteurs et chaîne d'acquisition", CultureSciences de l'ingénieur, 2015, https://sti.eduscol.education.fr/si-ens-cachan/ressources_pedagogiques/capteurs-et-chaine-dacquisition

[3] Frédéric de Coulon, "Théorie et traitement des signaux", Presses Polytechniques et Universitaires Romandes, Traité d'électricité, Vol. VI, Dunod. (téléchargeable).

[4] Dominique Ventre, "Communications analogiques", Collection pédagogique des Télécommunications, Ellipses.

[5] Michel Joindot, Alain Glavieux, Introduction aux télécommunications numériques, Collection Sciences Sup, Ed. Dunod.

[6] J.-M. Friedt, Quelques fondements théoriques pour aborder le radio-logicielle, GNU/Linux Magazine, Numéro 224

[7] J.-M. Friedt, H. Boeglen, Communication Systems Engineering with GNU Radio: A Hands-on Approach, Wiley (2025)

[8] T. Collins & al., Software-Defined Radio for Engineers, (2018) at https://www.analog.com/en/education/education-library/software-defined-radio-for-engineers.html

[9] Xavier Lagrange, Principe de la transmission OFDM - Utilisation dans les systèmes cellulaires, Techniques de l'Ingénieur, TE 7372.

[10] Marian Oziewicz, Digital Radio DAB+, Broadcasting Multimedia System, Ed. Springer.

[11] Jacques Palicot, Christophe Moy, Sufi Tabassum Gul, Merouane Debbah, Romain Couillet, et al. Radio Engineering: From Software Radio to Cognitive Radio. Edited by Jacques Palicot. Wiley-ISTE, 378 p., 2011. ⟨hal-00657728⟩

[12] Raymond Knopp, Carina Schmidt-Knorreck, Renaud Pacalet. Hardware Optimized Sample Rate Conversion for Software Defined Radio. *Frequenz*, 2010. ⟨hal-02893096⟩

Filtrage numérique avec GNU Radio : approche expérimentale

La Revue 3EI
Culture Sciences de l'Ingénieur

Édité le 08/06/2026

école normale supérieure paris—saclay

Thomas LAVARENNE[1]

[1] *Enseignant de Sciences Physiques en BTS CIEL option IR – Académie de Créteil.*

Cette ressource fait partie du N°120 de La Revue 3EI du 3ème trimestre 2026.

Le filtrage numérique constitue un élément central du traitement du signal moderne. Il est présent dans un grand nombre de systèmes, allant des communications radio aux applications audio, en passant par l'instrumentation et l'analyse de données. Malgré cette importance, son enseignement reste souvent très théorique, ce qui peut rendre difficile l'appropriation concrète des concepts.

GNU Radio permet de dépasser cette difficulté. Cet environnement offre la possibilité de manipuler des signaux en temps réel, de visualiser leur évolution et de tester directement l'effet des paramètres d'un filtre. Cette approche expérimentale permet de relier de manière immédiate les équations de filtrage à leur impact réel.

Dans cet article, nous proposons une progression basée sur des manipulations simples. L'étude est ensuite approfondie en explorant le lien entre sélectivité fréquentielle et coût de calcul, avant d'introduire les outils de conception de filtres.

1 - Filtres numériques : interprétation et mise en œuvre

1.1 - Présentation

Un filtre numérique est défini par une relation entre un signal d'entrée $x[n]$ et un signal de sortie $y[n]$.

Dans le cas des filtres FIR (Finite Impulse Response), filtres à réponse impulsionnelle finie, dont les échantillons de sortie dépendent uniquement des échantillons d'entrée :

$$y[n]=\sum_{k=0}^{N} b_k \cdot x[n-k]$$

Chaque échantillon de sortie est obtenu comme une combinaison pondérée d'échantillons passés. Cette structure est simple à comprendre et garantit un comportement stable, au contraire des filtres IIR (Infinite Impulse Response), filtres à réponse impulsionnelle infinie dont les échantillons de sortie dépendent à la fois des échantillons d'entrée et des échantillons de sortie précédemment calculés et qui ne seront pas évoqués ici.

Dans GNU Radio, la relation de récurrence donnée ci-dessus se traduit directement par des blocs élémentaires. Cela permet de visualiser concrètement le rôle de chaque coefficient.

1.2 - Construction de filtres FIR simples

Moyenne glissante

Considérons le filtre :

$$y[n]=0{,}5 \cdot x[n]+0{,}5 \cdot x[n-1]$$

Que nous pouvons représenter par la structure suivante dans Gnuradio Companion :

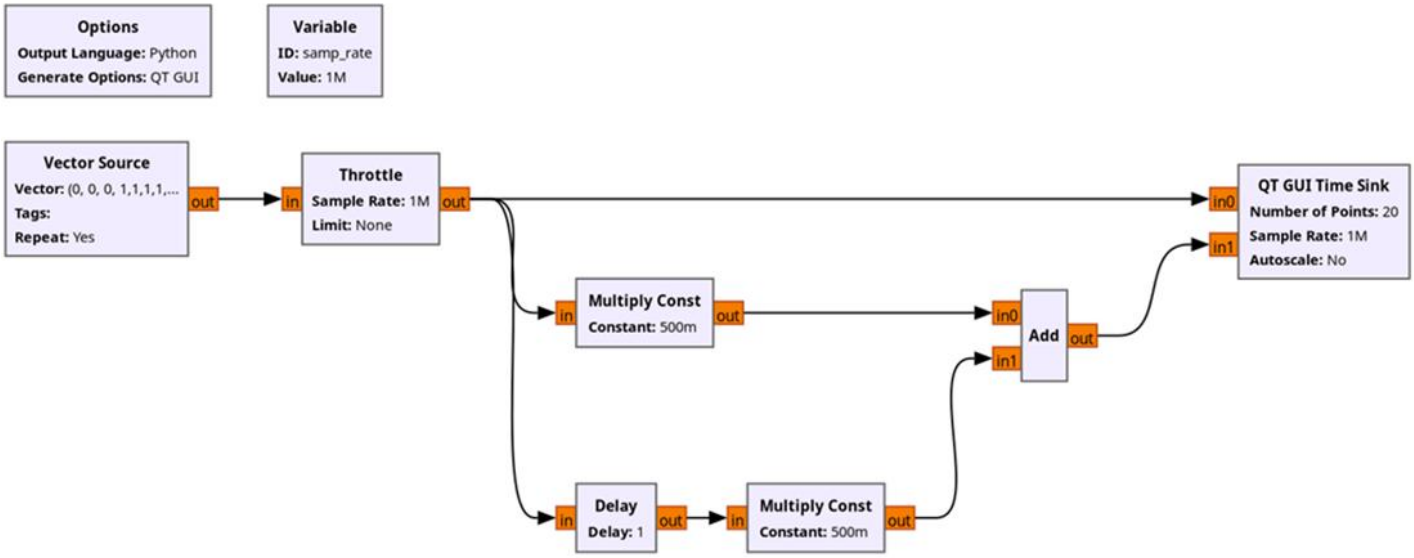


Figure 1 : Graphe GNU Radio Companion représentant un filtre à moyenne glissante soumis à un échelon en entrée

Tous les blocs utilisés sont configurés en type *Float* (ports de couleur orange), ce qui permet de manipuler directement des signaux réels. Le paramètre d'échantillonnage *samp_rate* est fixé à 1MHz, ce qui correspond à un intervalle de 1µs entre deux échantillons.

Le signal d'entrée du filtre est généré à l'aide d'un bloc *Vector Source*, produisant la séquence suivante :

(0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)

Ce signal correspond à un échelon discret, utilisé pour observer la réponse indicielle du filtre.

L'entrée et la sortie du filtre sont visualisées simultanément à l'aide d'un bloc *QT GUI Time Sink* configuré avec deux voies. Le déclenchement est réglé en mode automatique, avec un niveau de 0,1 et un délai de 2 µs, ce qui permet de stabiliser l'affichage autour du front de l'échelon.

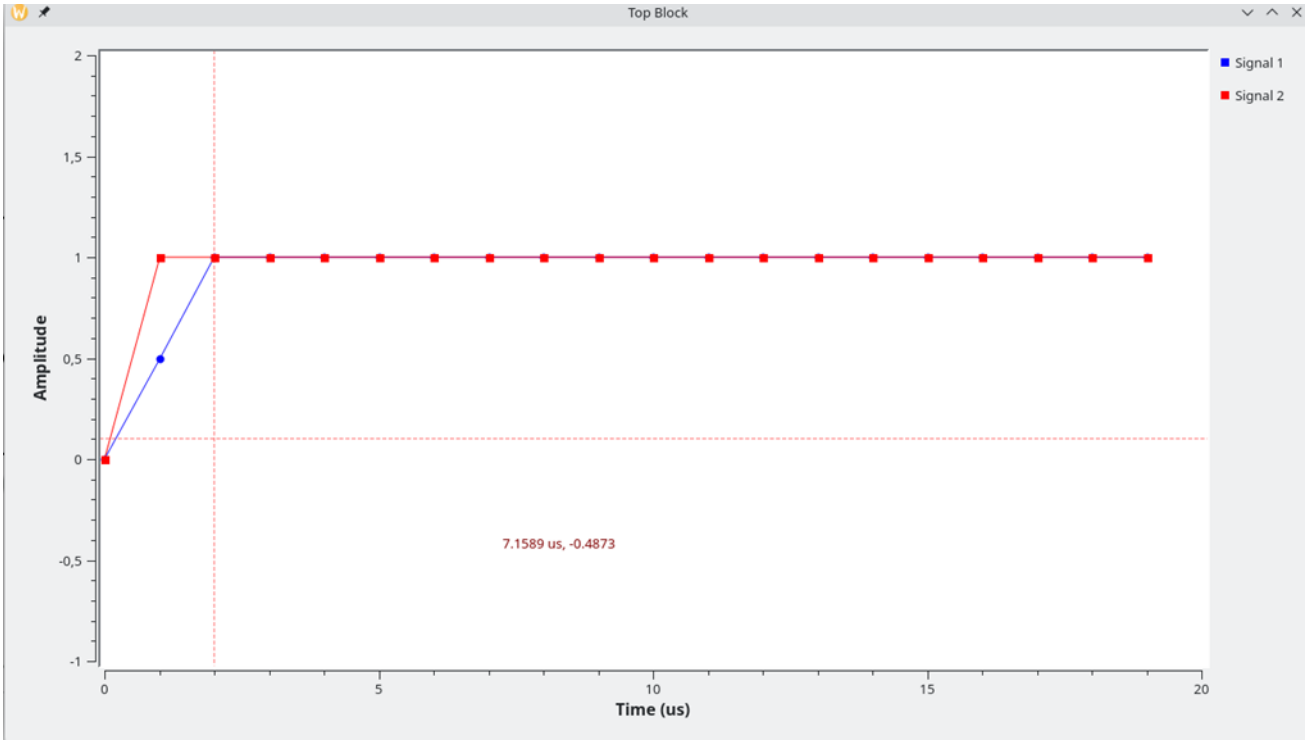


Figure 2 : Réponse indicielle du filtre à moyenne glissante

On observe sur la figure 2 que la sortie ne passe pas instantanément de 0 à 1, mais présente une transition progressive sur deux échantillons. Cette évolution est caractéristique du filtre utilisé, qui réalise une moyenne entre l'échantillon courant et le précédent. Cette observation permet de relier directement l'équation du filtre à son comportement temporel. Ce filtre réalise ainsi une moyenne entre deux échantillons successifs. Il atténue les variations rapides et agit donc comme un filtre passe-bas.

De la même façon, il est possible d'observer la réponse impulsionnelle du filtre en remplaçant l'échelon en entrée par un indice. Toutefois, une approche encore plus riche consiste à étudier son comportement lorsqu'il est excité par un signal sinusoïdal.

Pour cela, remplaçons le signal d'entrée précédent par un bloc *Signal Source* (Figure 3). Ce bloc permet de générer un signal périodique simple, ici un sinus, dont on peut régler facilement les paramètres principaux. L'amplitude est fixée à 1 afin de disposer d'une référence claire pour l'observation du gain du filtre, tandis que la fréquence devient le paramètre variable.

Afin de pouvoir faire varier cette fréquence de manière interactive, on introduit une variable nommée *freq*, associée à un bloc *QT GUI Range*. Ce bloc correspond à un curseur affiché dans la fenêtre de GNU Radio. En déplaçant ce curseur, on modifie en temps réel la fréquence du sinus généré, ce qui permet d'observer progressivement le comportement du filtre sur toute la bande de fréquences.

L'entrée et la sortie du filtre sont ensuite visualisées simultanément à l'aide d'un bloc *QT GUI Time Sink*. Pour améliorer la lisibilité des signaux, notamment lorsque les variations deviennent rapides ou que les déphasages sont faibles, chaque voie est suréchantillonnée à l'aide d'un bloc *Rational Resampler* configuré avec un facteur d'interpolation égal à 10. Cette opération consiste à insérer des échantillons intermédiaires entre les échantillons existants, sans modifier le contenu fréquentiel du signal, ce qui permet d'obtenir une représentation temporelle plus fine, notamment lorsque la fréquence des signaux se rapproche de la fréquence de Nyquist.

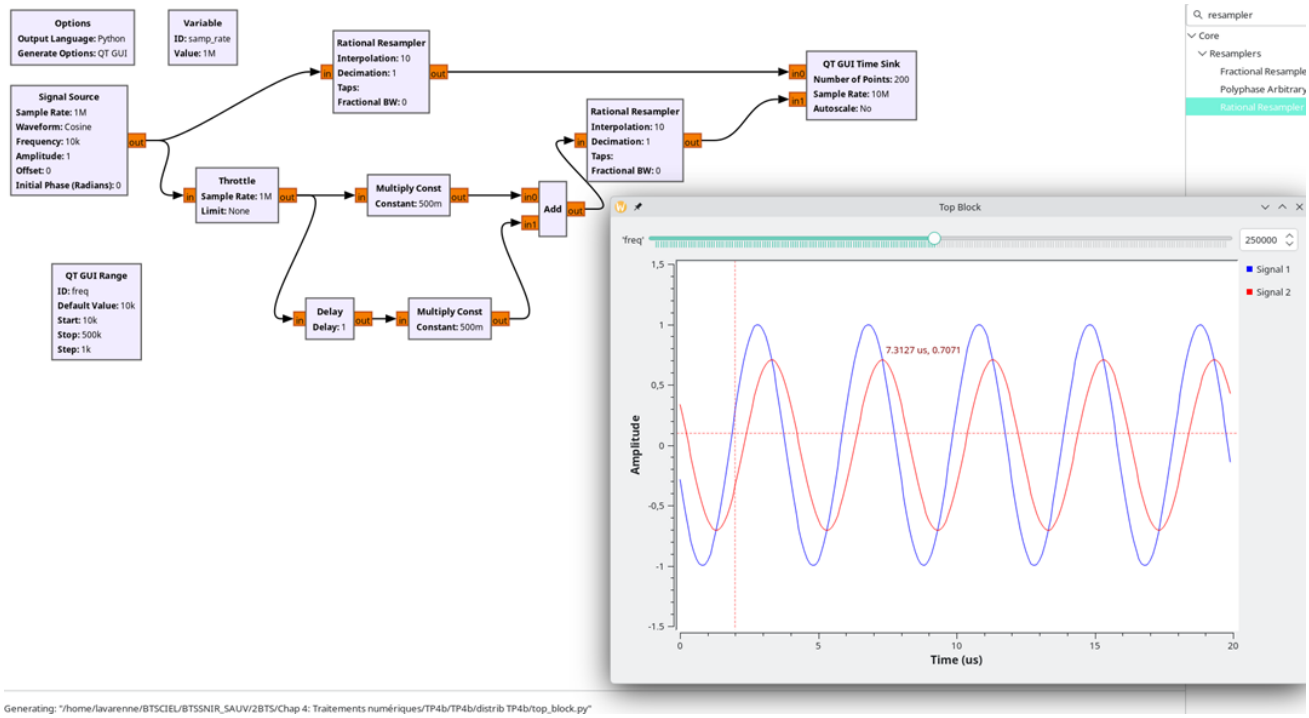


Figure 3 : Réponse du filtre à un signal sinusoïdal de fréquence 250 kHz. En bleu le signal d'entrée et en rouge le signal filtré en sortie

Comme le nombre d'échantillons par seconde est ainsi multiplié par 10, il est nécessaire d'adapter le paramètre *Sample Rate* du bloc *QT GUI Time Sink*, qui est ici réglé à 10 MHz. Cela garantit que l'axe temporel reste cohérent avec le signal affiché.

On peut alors faire varier progressivement la fréquence du sinus à l'aide du curseur. Pour les basses fréquences, le signal de sortie est quasiment identique au signal d'entrée. En revanche, lorsque la fréquence augmente, on observe une atténuation de l'amplitude ainsi qu'un déphasage entre les deux signaux. Lorsque l'amplitude du signal de sortie devient environ égale à 0,7 fois celle du signal d'entrée, on atteint la fréquence de coupure du filtre, caractérisée par une atténuation de -3 dB (Figure 3). En effet, pour le filtre défini par :

$$y[n]=0{,}5 \cdot x[n]+0{,}5 \cdot x[n-1]$$

La réponse fréquentielle vaut :

$$H(f)=0{,}5 \cdot \left(1+e^{-j2\pi f/f_e}\right)$$

Son module est :

$$|H(f)|=cos(\pi\, f/f_e)$$

La fréquence de coupure est définie par :

$$|H(f_c)|=\frac{1}{\sqrt{(2)}}$$

On obtient alors :

$$cos(\pi\, f_c/f_e)=\frac{1}{\sqrt{(2)}}$$

D'où :

$$f_c=\frac{f_e}{4}$$

On observe ainsi sur la Figure 3 que la fréquence pour laquelle l'amplitude du signal de sortie vaut environ 0,7 fois celle du signal d'entrée correspond bien à un quart de la fréquence d'échantillonnage, conformément à la théorie.

Pour finir, il est également possible de visualiser directement la réponse en fréquence du filtre en utilisant un bruit aléatoire en entrée et un bloc *QT GUI Frequency Sink* en sortie. Cette approche permet d'observer en une seule fois le comportement du filtre sur l'ensemble du spectre, sans avoir à faire varier la fréquence du signal d'entrée (Figure 4).

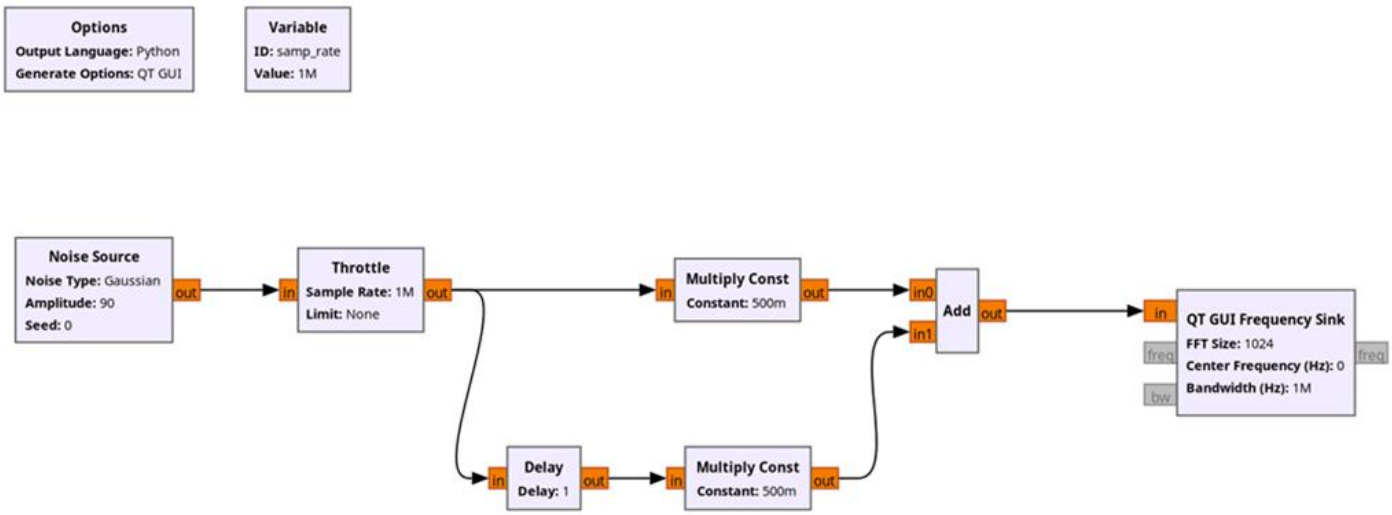


Figure 4 : Graphe GNU Radio Companion représentant un filtre à moyenne glissante soumis à un échelon en entrée

Le signal injecté est généré à l'aide d'un bloc *Noise Source*, configuré en bruit gaussien. Son amplitude est fixée à une valeur élevée (par exemple 90), ce qui ne modifie pas la forme de la réponse fréquentielle du filtre, mais permet d'obtenir un niveau de signal suffisamment élevé pour que le spectre affiché soit lisible.

En sortie, le bloc *QT GUI Frequency Sink* permet d'afficher le spectre du signal filtré. Il est important de régler correctement ses paramètres pour une interprétation claire. Le type de données est ici en *Float*, ce qui correspond à des signaux réels. Dans ce cas, le spectre affiché est symétrique, avec des fréquences positives et négatives. Il est donc pertinent d'activer l'affichage en mode « *Half* », afin de ne visualiser que la partie positive du spectre, ce qui simplifie la lecture.

Dans cette configuration, on observe clairement sur la Figure 5 que les basses fréquences sont transmises sans atténuation, tandis que les hautes fréquences sont progressivement atténuées. La transition entre ces deux zones correspond à la fréquence de coupure du filtre, que l'on retrouve autour d'un quart de la fréquence d'échantillonnage.

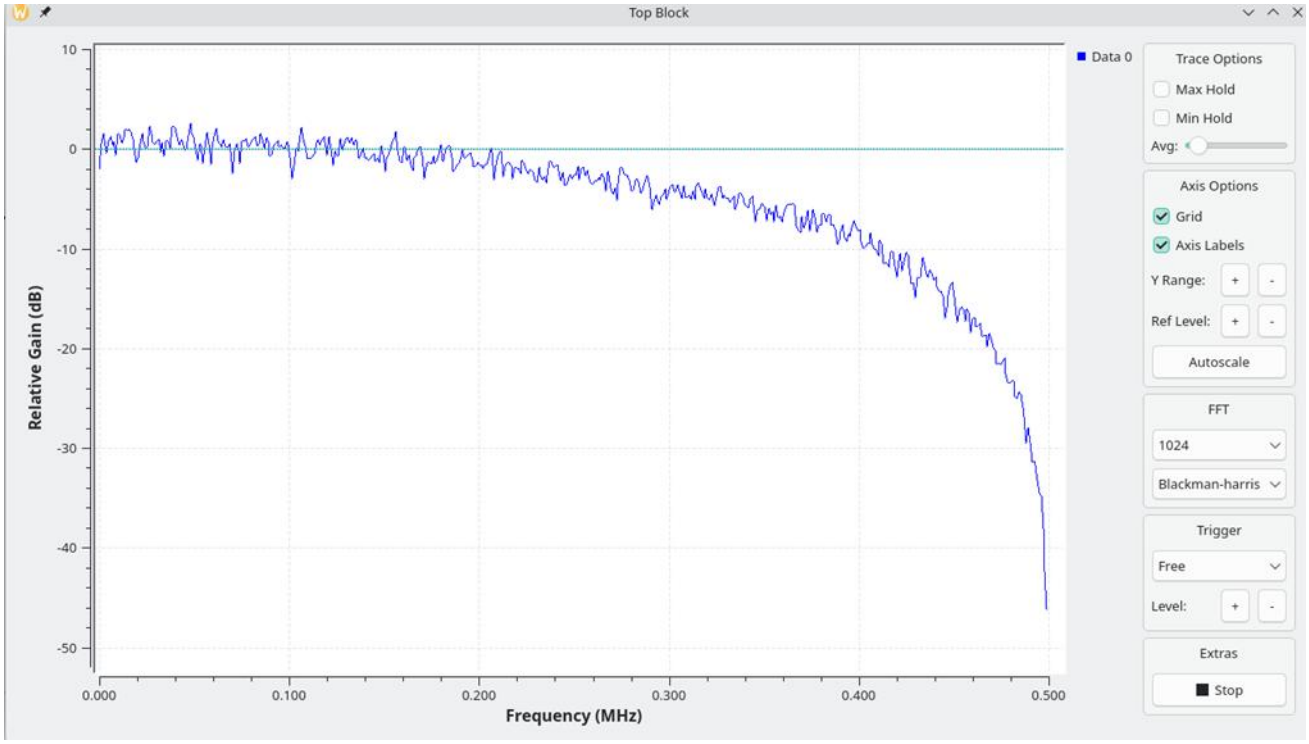


Figure 5 : Réponse en fréquence du filtre à moyenne glissante

1.3 - Filtre passe-bande

Soit le filtre caractérisé par l'équation suivante (Figure 6) :

$$y[n] = -0{,}2 \cdot x[n] + 0{,}05 \cdot x[n-1] + 0{,}7 \cdot x[n-2] + 0{,}05 \cdot x[n-3] - 0{,}2 \cdot x[n-4]$$

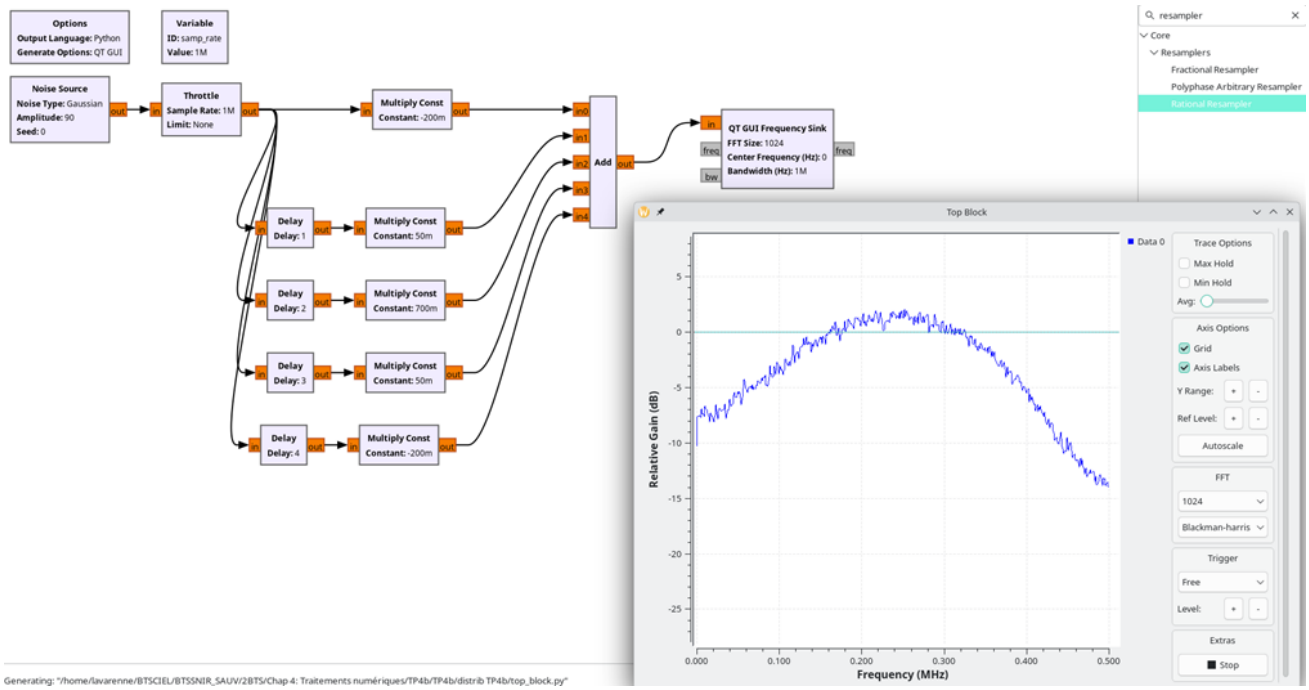


Figure 6 : Filtre passe bande à 5 coefficients

Contrairement aux exemples précédents, les coefficients ont ici été choisis de manière à privilégier une certaine bande de fréquences. La visualisation de la réponse fréquentielle, obtenue à l'aide d'un bruit en entrée et d'un bloc *QT GUI Frequency Sink*, met clairement en évidence un comportement de type passe-bande : les basses fréquences sont atténuées, le gain devient maximal autour d'une fréquence centrale, puis baisse à nouveau pour les hautes fréquences.

Cet exemple illustre un point important en filtrage numérique : la nature du filtre dépend directement des coefficients utilisés. Une simple modification de ces derniers permet de passer d'un filtre passe-bas à un filtre passe-haut, ou encore à un filtre passe-bande, sans changer la structure générale du montage. En modifiant en temps réel les coefficients (à l'aide de *QT GUI Range*), il devient possible d'observer immédiatement l'impact sur la réponse fréquentielle, ce qui permet de relier de manière concrète l'expression mathématique du filtre à son comportement spectral.

2 - Utilisation des blocs de filtrage dans GNU Radio

Après avoir étudié des filtres construits explicitement à partir de leur équation de récurrence, il est naturel de s'intéresser aux outils permettant de concevoir des filtres de manière plus directe. En pratique, on ne manipule généralement pas les coefficients un par un, mais on définit plutôt les caractéristiques fréquentielles souhaitées, telles que la fréquence de coupure ou la sélectivité.

GNU Radio propose pour cela des blocs de filtrage prêts à l'emploi, dont le plus courant est le bloc *Low Pass Filter*. Celui-ci permet de réaliser simplement un filtre passe-bas en spécifiant quelques paramètres physiques, les coefficients étant ensuite générés automatiquement par une fonction python.

2.1 - Paramétrage du bloc Low Pass Filter

Le bloc *Low Pass Filter* permet de réaliser un filtre passe-bas en spécifiant directement ses caractéristiques fréquentielles. Contrairement à l'approche précédente, il n'est plus nécessaire de définir explicitement les coefficients : ceux-ci sont calculés automatiquement à partir des paramètres fournis.

Les principaux paramètres à renseigner sont les suivants :

- La fréquence d'échantillonnage (**samp_rate**), identique à celle utilisée dans le reste de la chaîne de traitement ;
- La fréquence de coupure (**cutoff frequency**), qui définit la limite entre les fréquences transmises et celles qui seront atténuées ;
- La largeur de transition (**transition width**), qui correspond à la zone dans laquelle le gain du filtre passe progressivement de la bande passante à la bande coupée ;
- Le gain du filtre, généralement fixé à 1.

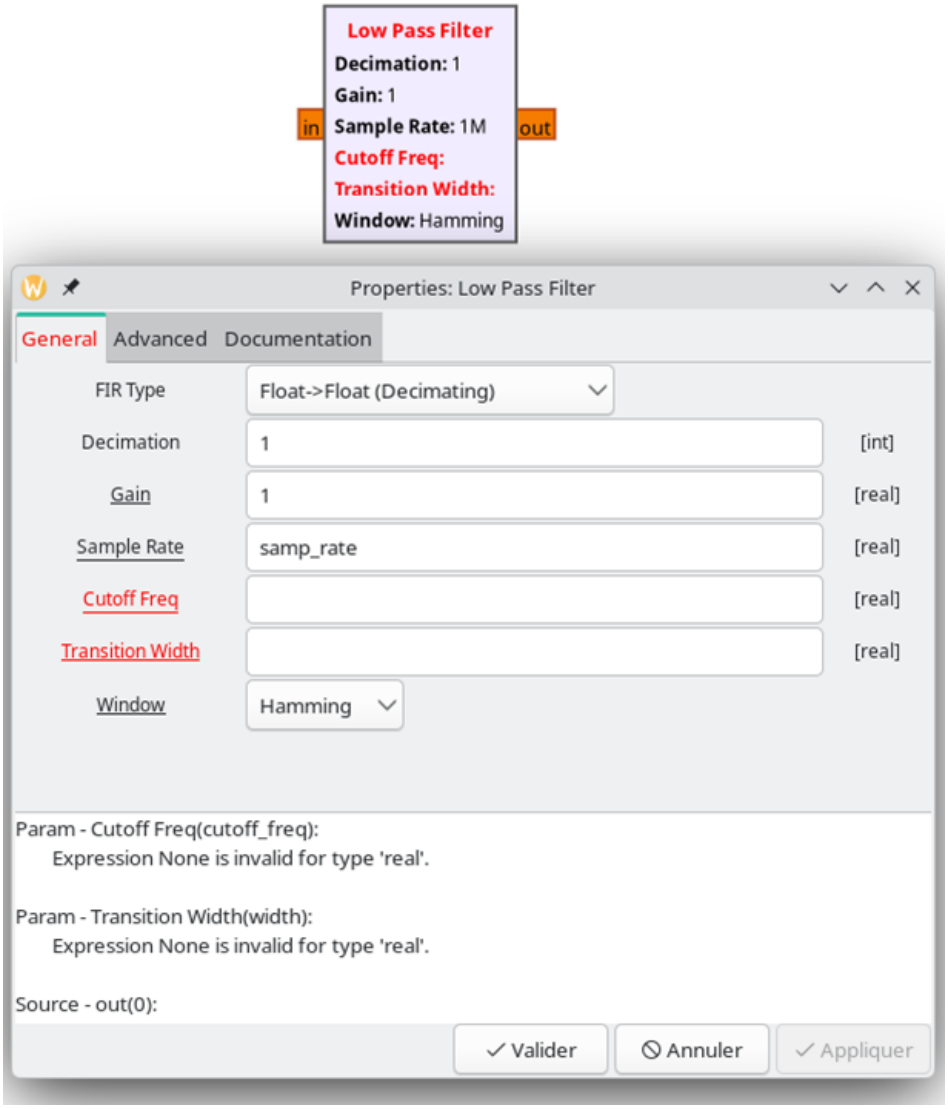


Figure 7 : Configuration du bloc Low Pass Filter

Cette manière de paramétrer le filtre est particulièrement simple et intuitive. On raisonne directement en termes de fréquences, ce qui facilite le lien avec le signal traité. Par exemple, pour un signal échantillonné à 1 MHz, une fréquence de coupure fixée à 100 kHz signifie que les composantes fréquentielles inférieures à cette valeur seront transmises, tandis que les composantes plus élevées seront progressivement atténuées. Une fois ces paramètres définis, GNU Radio génère automatiquement les coefficients du filtre FIR correspondant. L'utilisateur peut ainsi se concentrer sur le comportement du filtre plutôt que sur son implémentation.

2.2 - Observation de la réponse fréquentielle

Comme dans la partie précédente, la réponse fréquentielle du filtre peut être observée en injectant un bruit en entrée et en visualisant le spectre du signal en sortie à l'aide d'un bloc *QT GUI Frequency Sink*.

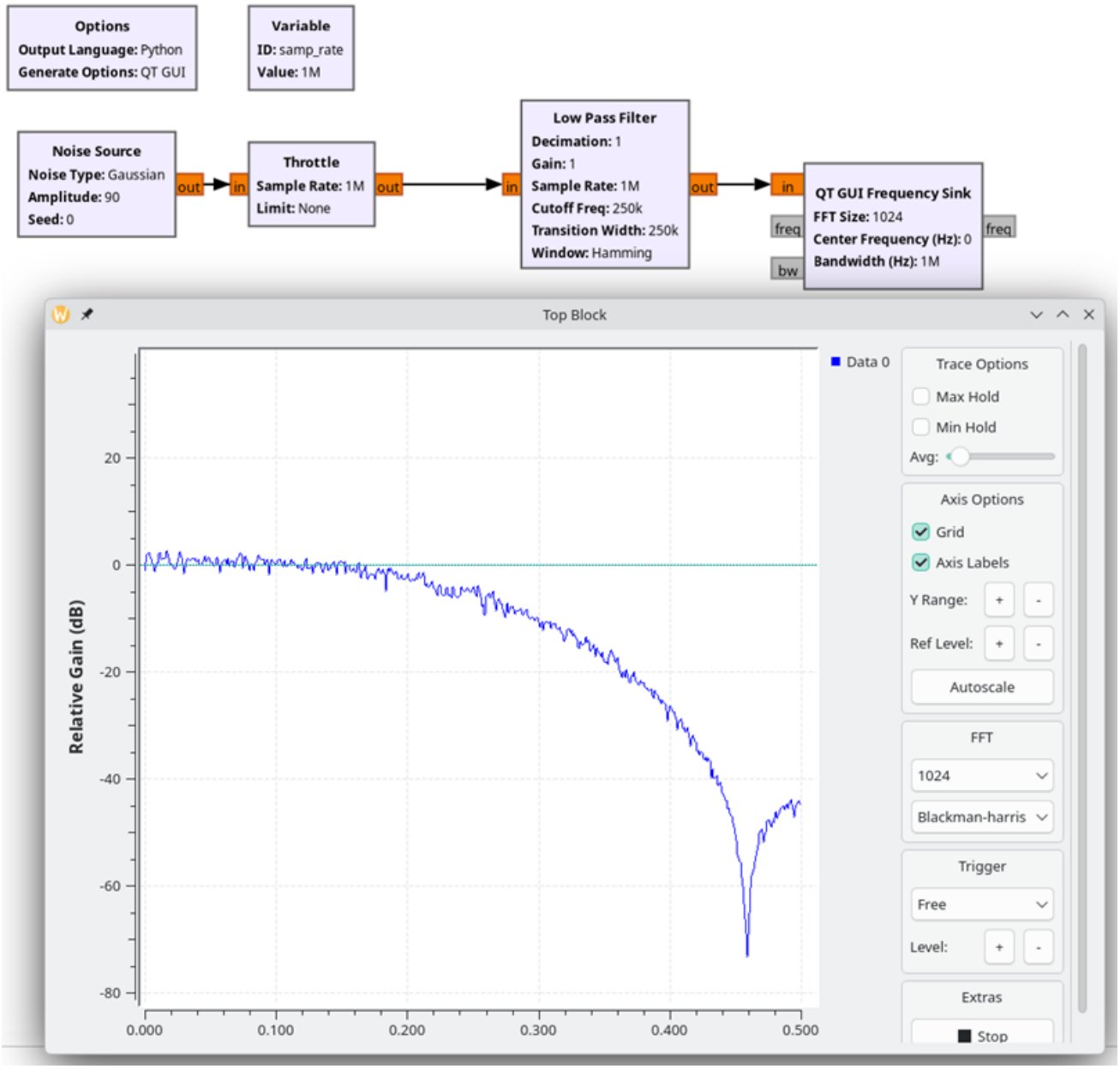


Figure 8 : Réponse en fréquence du Low Pass Filter avec cutoff = 250 kHz et Transition = 250 kHz

La réponse du filtre apparaît alors clairement. Les basses fréquences sont transmises, tandis que les hautes fréquences sont progressivement atténuées. La zone de transition entre ces deux comportements correspond à la fréquence de coupure définie dans le bloc *Low Pass Filter*.

Cette méthode permet d'observer en une seule fois l'ensemble de la réponse fréquentielle du filtre, sans avoir à faire varier la fréquence du signal d'entrée et constitue ainsi un outil simple et efficace pour analyser le comportement d'un filtre numérique.

2.3 - Influence de la largeur de transition

Un des paramètres importants du bloc *Low Pass Filter* est la largeur de transition (**transition width**). Elle définit la zone dans laquelle le gain du filtre passe progressivement de la bande passante à la bande atténuée.

Pour mettre en évidence son rôle, on peut associer ce paramètre à un bloc *QT GUI Range* afin de le faire varier en temps réel. L'observation de la réponse fréquentielle montre alors immédiatement l'effet de ce réglage.

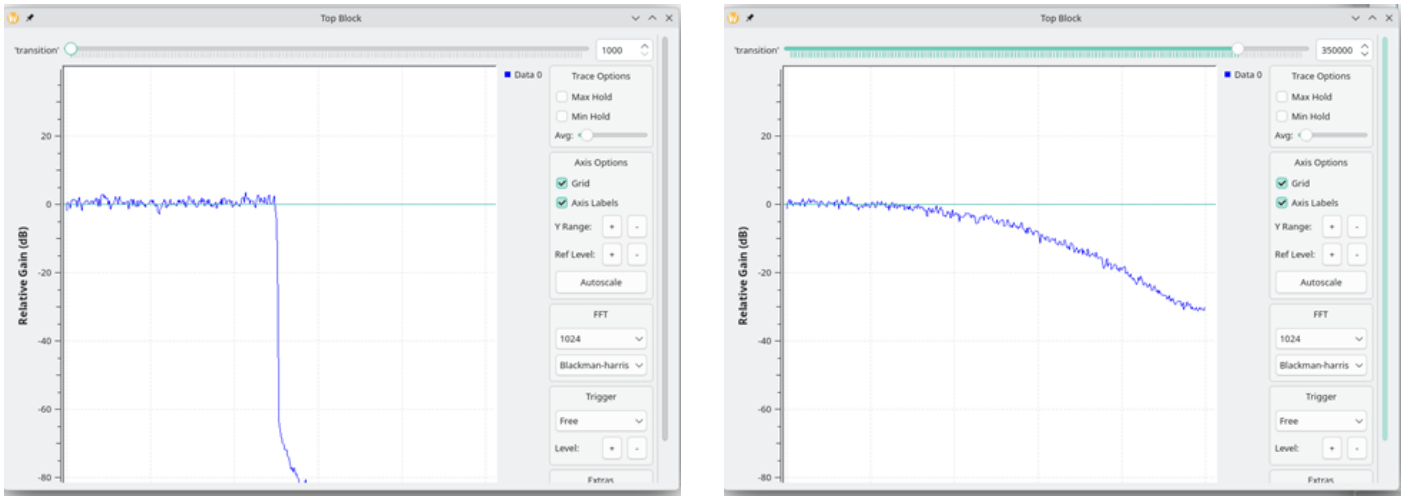


Figure 9 : Réponse en fréquence avec Transition = 1 kHz (à gauche) et Transition = 350 kHz (à droite)

Lorsque la largeur de transition est importante, la décroissance du gain est lente. Le filtre est peu sélectif : il laisse passer une partie des fréquences situées au-delà de la fréquence de coupure.

À l'inverse, lorsque la largeur de transition est réduite, la coupure devient beaucoup plus abrupte. Le filtre est alors plus sélectif, ce qui permet de mieux séparer les différentes composantes fréquentielles du signal. Cette amélioration a cependant un coût, car elle nécessite un filtre avec davantage de coefficients, comme nous allons le montrer dans le paragraphe suivant.

Cette observation met en évidence un compromis fondamental en traitement du signal numérique : il n'est pas possible d'obtenir simultanément une transition très abrupte et un filtre de faible complexité. Plus la transition est étroite, plus le nombre de coefficients nécessaires est important.

2.4 - Nombre de coefficients et coût de calcul

Le bloc *Low Pass Filter* masque volontairement la complexité du filtrage en générant automatiquement les coefficients nécessaires. Toutefois, ces coefficients existent bien, et leur nombre dépend directement des paramètres choisis, en particulier de la largeur de transition. Lorsque la transition est large, le filtre nécessite peu de coefficients. À l'inverse, lorsque la transition devient plus étroite, le nombre de coefficients augmente rapidement. Cette augmentation se traduit directement par une complexité de calcul plus importante.

En effet, pour chaque échantillon traité, le filtre doit effectuer une combinaison pondérée de l'ensemble de ses coefficients. Plus le filtre est long, plus le nombre d'opérations à réaliser est élevé. Dans une application en temps réel, ce point peut devenir non négligeable.

Cet aspect peut être mis en évidence en utilisant la fonction python de GNU Radio permettant de générer les coefficients du filtre :

firdes.low_pass(1, samp_rate, cutoff, transition)

Le nombre de coefficients générés dépend principalement de la largeur de transition. Lorsque celle-ci est large, le filtre est court et comporte peu de coefficients. En revanche, lorsque la transition devient très étroite, le nombre de coefficients augmente fortement.

Dans un shell python :

```
>>> from gnuradio.filter import firdes

>>> taps = firdes.low_pass(1, 1e6, 100e3, 100e3)

>>> print(taps)
[0.0020104029681533575, 0.001621020375750959, -1.0999144682209492e-18, -0.0044467085
97242832, -0.011685466393828392, -0.018134258687496185, -0.016773713752627373, 5.118
36987812244e-18, 0.035877108573913574, 0.08697697520256042, 0.14148786664009094, 0.1
8345333635807037, 0.19922685623168945, 0.18345333635807037, 0.14148786664009094, 0.0
8697697520256042, 0.035877108573913574, 5.11836987812244e-18, -0.016773713752627373,
-0.018134258687496185, -0.011685466393828392, -0.004446708597242832, -1.09991446822
09492e-18, 0.001621020375750959, 0.0020104029681533575]

>>> print(len(taps))
25
```

Ainsi, une bande de transition de 100 kHz nécessite la génération de 25 coefficients (la fonction len() en python renvoie le nombre d'éléments d'une liste). On peut diminuer la valeur de la bande de transition et observer l'effet sur le nombre de coefficient générés. Pour une bande de transition de 10 kHz :

```
>>> taps = firdes.low_pass(1, 1e6, 100e3, 10e3)

>>> print(len(taps))
241
```

Le nombre de coefficients est quasiment décuplé. On peut encore continuer, avec une bande de transition de 1 kHz :

```
>>> taps = firdes.low_pass(1, 1e6, 100e3, 1e3)

>>> print(len(taps))
2409
```

Nous constatons que la sélectivité fréquentielle d'un filtre non-récursif est directement liée à sa longueur. Plus la bande de transition est étroite, plus le nombre de coefficients nécessaires augmente.

Dans une implémentation temps réel, cela se traduit par une augmentation du nombre d'opérations à effectuer pour chaque échantillon. Le choix des paramètres du filtre ne relève donc pas uniquement de considérations théoriques, mais doit également tenir compte des contraintes de calcul.

3 - Conclusion

Cette première approche du filtrage numérique avec GNU Radio met en évidence un point essentiel : les concepts théoriques du traitement du signal prennent tout leur sens lorsqu'ils sont manipulés concrètement. La possibilité de visualiser simultanément les signaux temporels et fréquentiels permet d'établir un lien direct entre les équations de récurrence et leurs effets réels.

Les filtres FIR étudiés ici illustrent clairement comment le choix des coefficients influence le comportement du filtre, tandis que les blocs de conception intégrés à GNU Radio facilitent l'accès à des filtres plus complexes en se basant sur des paramètres physiques directement interprétables.

L'étude de la largeur de transition met également en évidence un compromis important entre sélectivité fréquentielle et coût de calcul, notion essentielle dans toute implémentation temps réel.

Analyse et sécurité d'une sonnette sans fil avec GNU Radio

La Revue 3EI
Culture Sciences de l'Ingénieur

Édité le 15/06/2026

école normale supérieure paris—saclay

Thomas LAVARENNE[1]

[1] *Enseignant de Sciences Physiques en BTS CIEL option IR – Académie de Créteil.*

Cette ressource fait partie du N°120 de La Revue 3EI du 3ème trimestre 2026.

Après avoir étudié le filtrage et le traitement de signaux simulés (voir ressource « Filtrage numérique avec GNU Radio : approche expérimentale » [1]), il est intéressant de se confronter à des signaux radio réels issus de systèmes du quotidien. Cette approche permet d'introduire concrètement plusieurs notions fondamentales de la radio logicielle moderne : représentation complexe des signaux, traitement en bande de base, analyse fréquentielle et démodulation numérique.

Contrairement aux signaux purement réels manipulés habituellement en électronique ou en traitement analogique, les récepteurs SDR (Software Defined Radio) fournissent généralement des signaux complexes composés de deux voies : une composante en phase I et une composante en quadrature Q. Cette représentation, très présente dans les systèmes de communication modernes, permet de conserver simultanément les informations d'amplitude et de phase du signal reçu tout en simplifiant de nombreuses opérations de traitement numérique.

L'objectif de cet article est de proposer une approche expérimentale et progressive de ces notions à l'aide de GNU Radio. Cet environnement graphique permet de construire facilement des chaînes de traitement tout en visualisant en temps réel les différentes étapes du traitement du signal. Chaque opération peut ainsi être isolée, observée et reliée directement aux concepts théoriques associés.

Le support choisi ici est une sonnette sans fil fonctionnant dans la bande ISM 433 MHz. Ce type de dispositif utilise généralement une modulation simple de type OOK (On-Off Keying), particulièrement adaptée à une première approche de la démodulation radio. À partir du signal reçu par une clé SDR, nous montrerons comment observer le spectre, extraire l'enveloppe du signal, décoder les informations transmises puis reproduire l'émission à l'aide d'un émetteur SDR.

Au-delà de l'aspect purement technique, ces expériences permettent également d'aborder certaines problématiques concrètes liées à la robustesse et à la sécurité des communications radio : rejeu de trames, sensibilité au brouillage et limites de certains protocoles radio simples encore présents dans des équipements grand public.

1 - Réception d'un signal radio réel avec un dispositif SDR

Avant de rentrer dans le vif du sujet, il est nécessaire de comprendre la nature du signal fourni par un récepteur SDR, et en particulier sa représentation en bande de base sous forme complexe.

1.1 - Passage en bande de base

Lorsqu'un signal radio est capté par une clé SDR, celui-ci n'est pas directement fourni sous sa forme haute fréquence (par exemple 433,92 MHz). Le récepteur effectue en interne une opération de transposition fréquentielle : il sélectionne une fréquence centrale puis ramène le signal autour de cette fréquence vers la bande de base (0 Hz).

Autrement dit, si l'on règle le récepteur sur une fréquence f_c, le signal de sortie ne contient plus directement les composantes autour de f_c, mais les écarts de fréquence par rapport à cette valeur. Une composante située à $f_c+\Delta f$ apparaîtra ainsi comme une composante à $+\Delta f$ en bande de base.

Ce processus repose sur une multiplication par un signal sinusoïdal complexe (c'est-à-dire une voie en cosinus et une voie en sinus), suivie d'un filtrage passe-bas. Il permet de conserver toute l'information utile du signal tout en le ramenant dans une bande de fréquence exploitable numériquement.

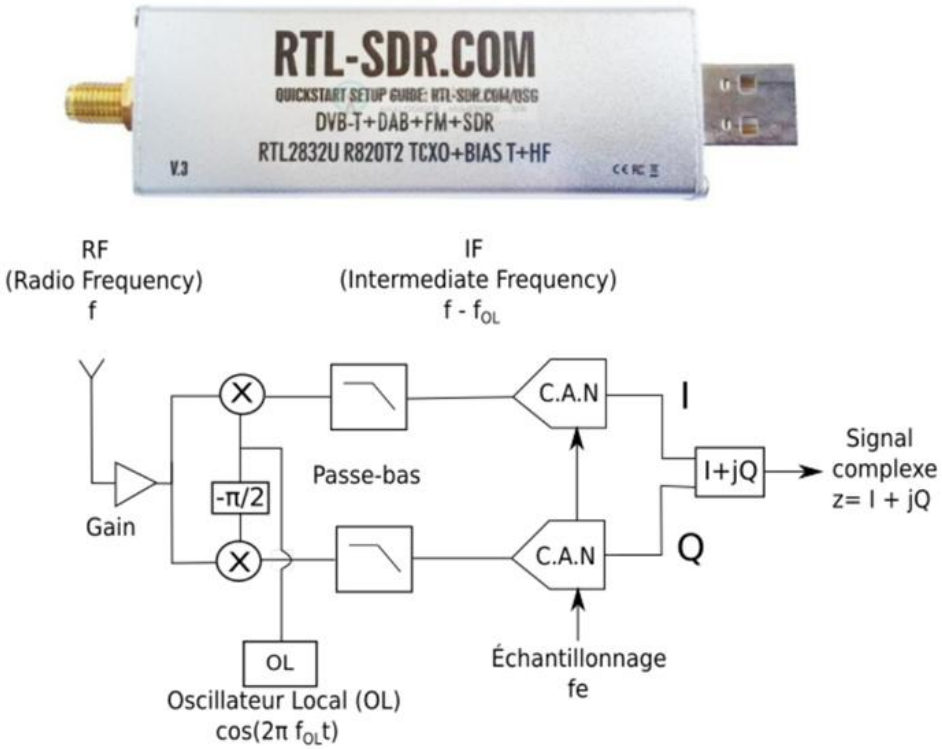


Figure 1 : Principe de fonctionnement d'un récepteur SDR (clé RTL-SDR)

Le signal fourni par un récepteur SDR est un signal complexe, constitué de deux composantes : I (en phase) et Q (en quadrature). Il s'écrit :

$$s(t) = I(t) + jQ(t)$$

Cette représentation permet de conserver à la fois l'amplitude et la phase du signal, ce qui est indispensable pour l'analyse des modulations. Dans GNU Radio, ces signaux sont manipulés sous forme de flux complexes (ports bleus), directement exploitables par les blocs de traitement.

1.2 - Visualisation du spectre

Pour observer le contenu fréquentiel du signal, on utilise le bloc *QT GUI Frequency Sink*. Celui-ci réalise une estimation du spectre à partir de FFT (Transformées de Fourier Rapides) calculées sur des blocs d'échantillons, ce qui permet une visualisation en temps réel proche d'un analyseur de spectre numérique. Dans le cas d'un signal complexe, le spectre s'étend de $-fe/2$ à $+fe/2$, où fe

est la fréquence d'échantillonnage. Le centre du spectre (0 Hz) correspond à la fréquence sur laquelle le récepteur est accordé :

- Les fréquences positives correspondent aux composantes au-dessus de la fréquence centrale ;
- Les fréquences négatives aux composantes situées en dessous.

Cette représentation permet de visualiser directement la position fréquentielle des signaux reçus. Nous allons maintenant appliquer ces principes à un exemple simple : la réception et la démodulation d'un signal de sonnette sans fil.

2 - Démodulation du signal reçu par une sonnette sans fil

Les sonnettes sans fil constituent un excellent support pédagogique pour introduire la démodulation de signaux radio. Leur fonctionnement repose généralement sur une modulation simple de type OOK (On-Off Keying), dans laquelle la présence ou l'absence de porteuse code l'information. L'objectif ici est de montrer comment extraire cette information à partir du signal complexe fourni par le récepteur SDR.

Figure 2 : Sonnette sans fil étudiée

2.1 - Observation du signal reçu

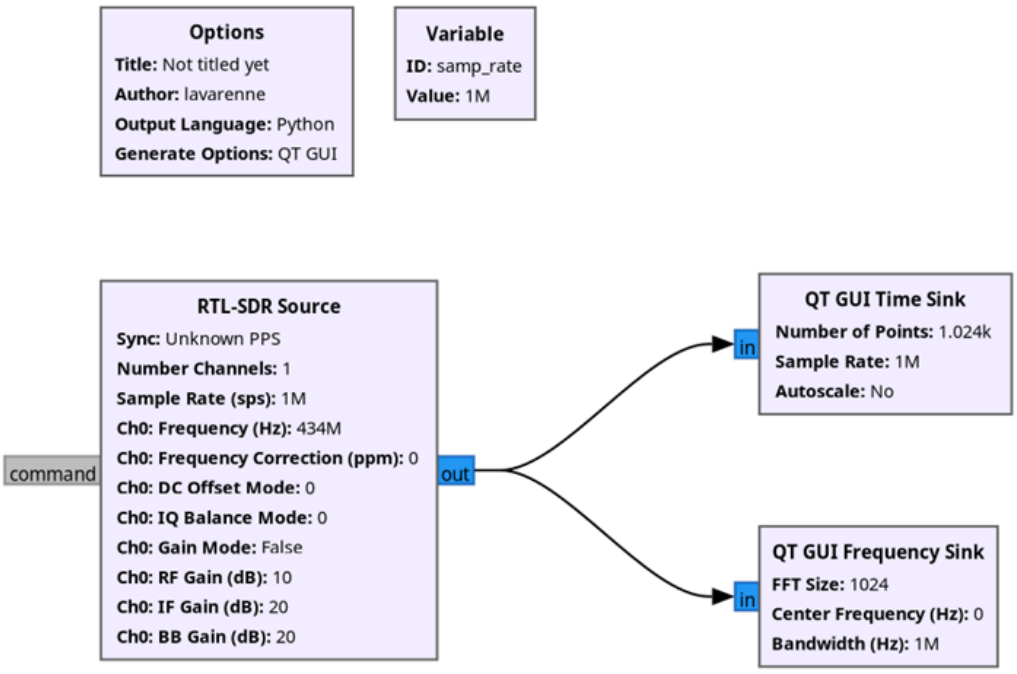


Figure 3 : Blocs GNU Radio pour afficher les représentations temporelles et fréquentielles

Après avoir réglé le récepteur sur la fréquence de la sonnette (typiquement autour de 433,92 MHz), l'appui sur le bouton émet un signal visible dans le spectre. Le paramètre **samp_rate** règle le taux d'échantillonnage qui est appliqué au signal ramené en bande de base. Il est important de le régler afin de ne pas être dans le cas d'un sous-échantillonnage ou d'un suréchantillonnage par rapport au signal utile transporté.

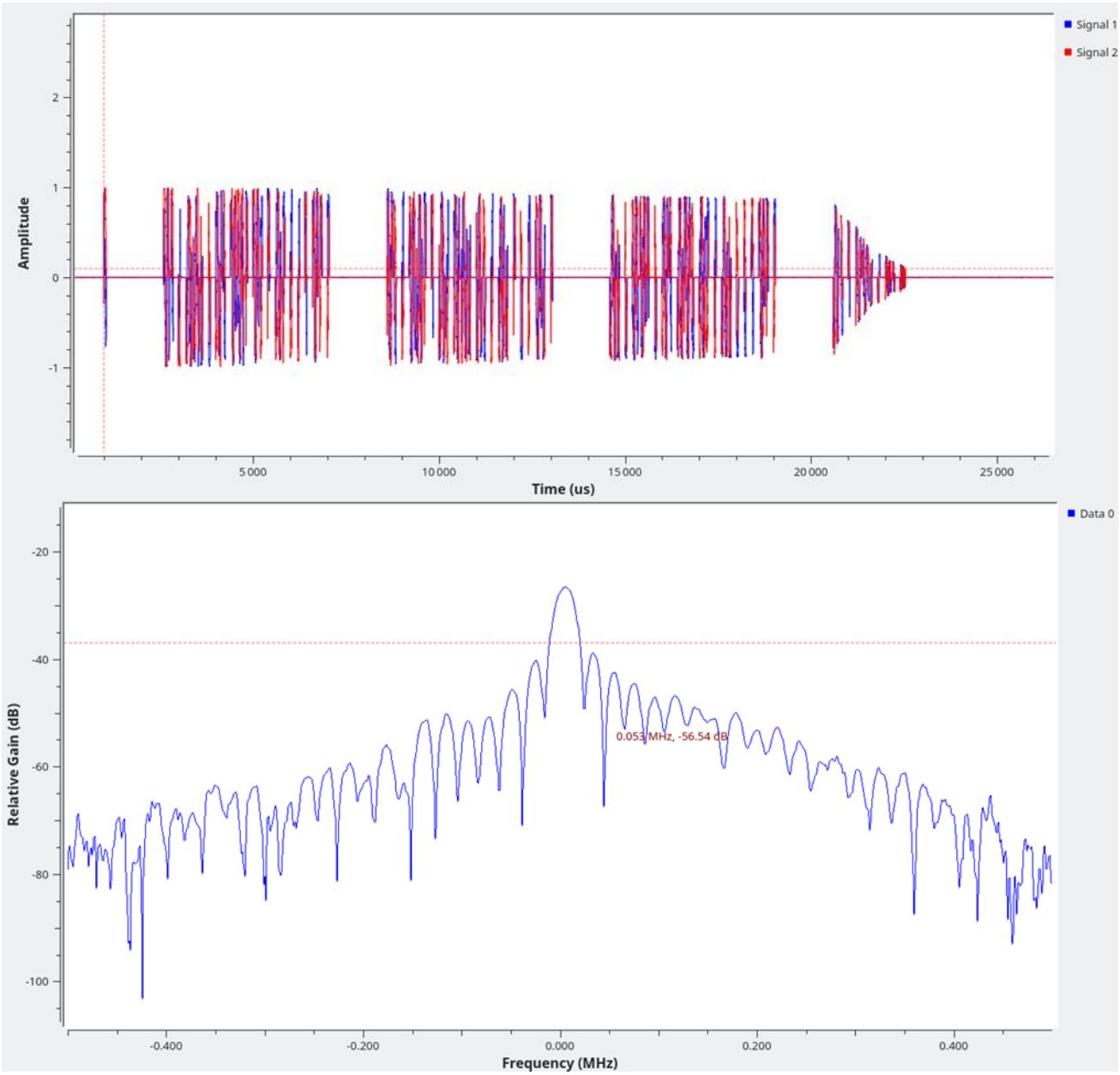


Figure 4 : Représentation temporelle et fréquentielle caractéristiques d'une modulation OOK

2.2 - Enregistrement et rejeu du signal

Pour plus de commodité, il est courant de réaliser une acquisition du signal émis avec un bloc File Sink (sous forme de fichier binaire *sonnette.bin* ici) et de le rejouer ensuite en boucle avec l'association des blocs File Source et Throttle (ce dernier étant nécessaire lorsqu'aucun périphérique matériel ne cadence l'échantillonnage).

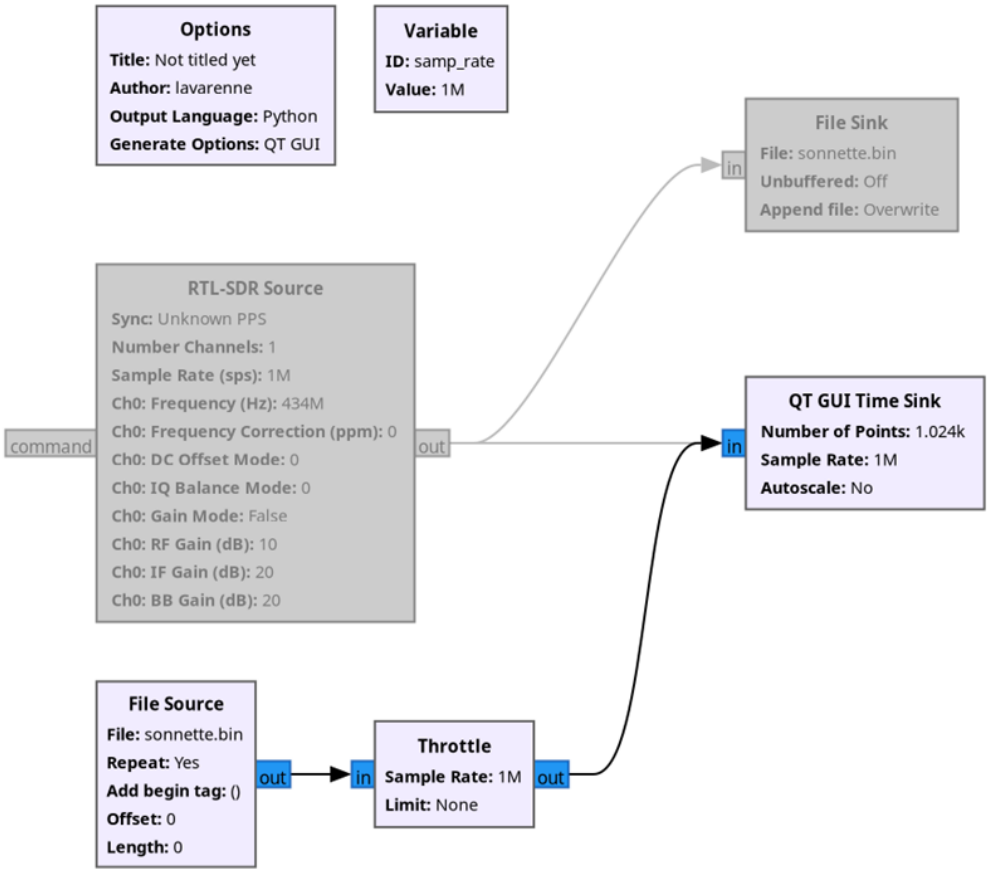


Figure 5 : Enregistrement puis rejeu du signal. En grisé, les blocs File Sink et RTL-SDR Source sont désactivés. Ils étaient actifs lors de l'acquisition

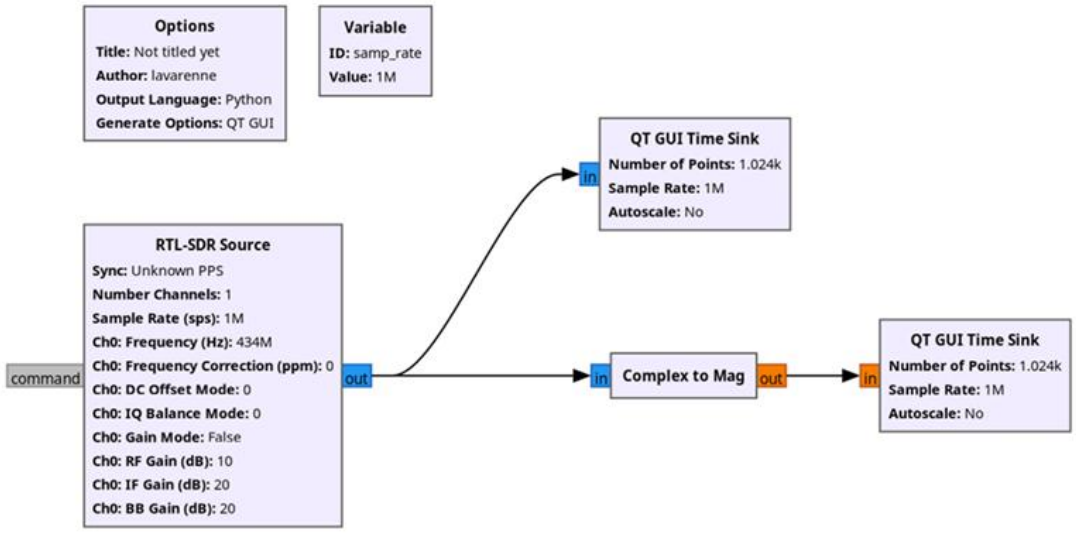
2.3 - Extraction de l'enveloppe

La modulation OOK repose sur la variation de l'amplitude du signal. L'information est donc contenue dans l'enveloppe du signal complexe.

Cette enveloppe peut être obtenue simplement à l'aide du bloc *Complex to Mag* qui calcule le module du signal complexe, c'est-à-dire :

$$|s(t)|=\sqrt{(I^2+Q^2)}$$

Le signal obtenu est réel et correspond à la puissance instantanée du signal reçu.



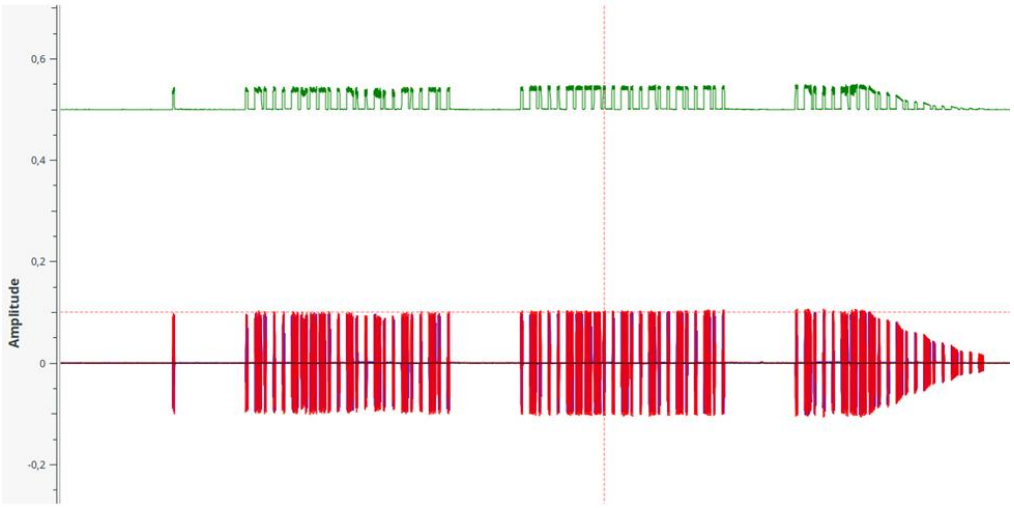


Figure 6 : Extraction de l'enveloppe avec le bloc Complex To Mag

2.4 - Embedded Python Block

Il peut être pédagogiquement intéressant de reconstruire cette opération d'extraction d'amplitude manuellement à l'aide des blocs GNU Radio :

- Un bloc **Complex To Float** pour séparer les deux voies I et Q ;
- Deux blocs **Multiply** pour calculer I^2 et Q^2 ;
- Un bloc **Add** ;
- Puis un **bloc Python personnalisé** pour effectuer la racine carrée.

Cette approche permet de relier explicitement les blocs au calcul mathématique. Pour commencer, réalisons le bloc *Complex To Mag²* qui réalise simplement $I^2 + Q^2$, sans effectuer la racine carrée :

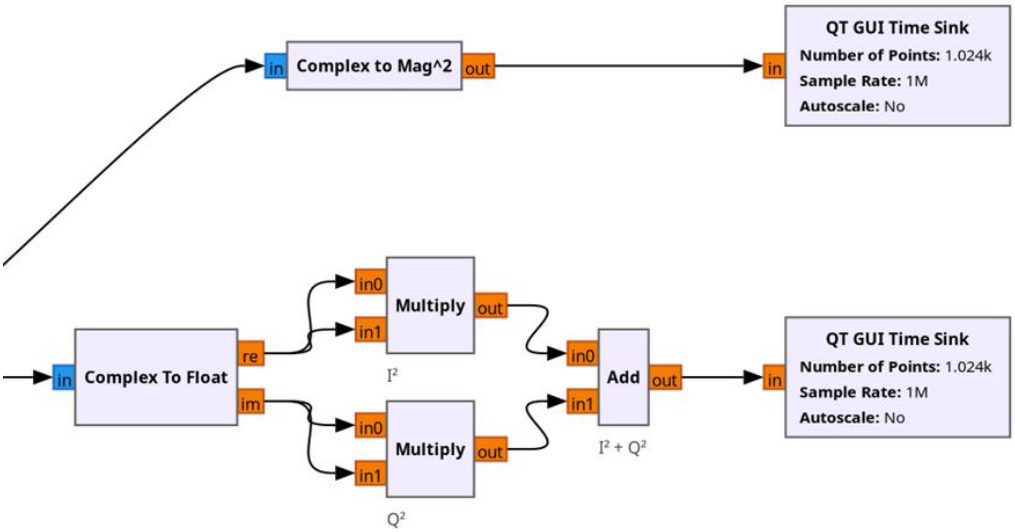


Figure 7 : Traitement GNU Radio pour obtenir $I^2 + Q^2$. Le bloc Complex To Float sépare les voies I et Q.

À ce stade, le signal est déjà démodulé et exploitable et le bloc *Complex To Mag²* est tout à fait utilisable en lieu et place du *Complex To Mag*. Néanmoins, c'est l'occasion de présenter une autre fonctionnalité de GNU Radio très intéressante : la possibilité d'implémenter soi-même ses propres blocs de traitement en Python.

En effet, il n'existe pas de blocs dans la librairie standard de GNU Radio qui applique l'opération racine carrée au flux d'échantillon. Qu'à cela ne tienne, nous allons donc le fabriquer nous-même !

Pour ce faire il existe des modèles de *Python Block* que nous allons utiliser et modifier pour réaliser l'opération voulue. Lorsque nous choisissons d'ajouter un tel bloc, nous avons la possibilité d'accéder au code source du bloc (*Open in editor*).

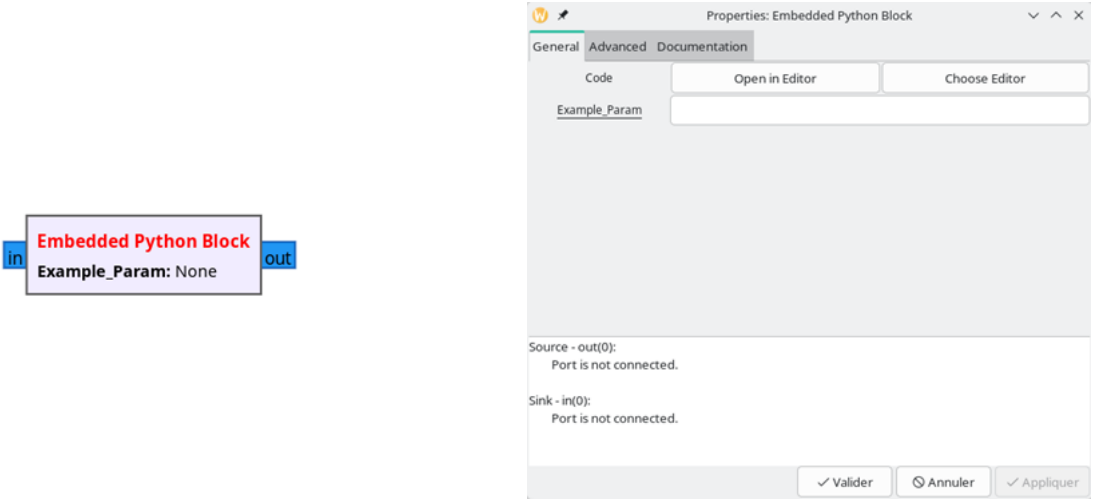


Figure 8 : Python bloc personnalisable

L'exemple qui s'ouvre par défaut est assez facile à comprendre. Il s'agit ici d'une simple multiplication par une constante. Adaptons donc cet exemple pour créer un bloc puissance qui prend en paramètre la valeur de la puissance à appliquer au flux (nous prendrons 0,5 pour appliquer la racine carrée).

```
"""
Embedded Python Blocks:

Each time this file is saved, GRC will instantiate the first
class it finds
to get ports and parameters of your block. The arguments to
__init__  will
be the parameters. All of them are required to have default
values!
"""

import numpy as np
from gnuradio import gr


class blk(gr.sync_block):  # other base classes are basic_block,
decim_block, interp_block
    """Embedded Python Block example - a simple multiply const"""

    def __init__(self, example_param=1.0):  # only default
    arguments here
        """arguments to this function show up as parameters in
        GRC"""
        gr.sync_block.__init__(
            self,
            name='Embedded Python Block',   # will show up in GRC
            in_sig=[np.complex64],
            out_sig=[np.complex64]
        )
        # if an attribute with the same name as a parameter is
        found,
        # a callback is registered (properties work, too).
        self.example_param = example_param

    def work(self, input_items, output_items):
        """example: multiply with constant"""
        output_items[0][:] = input_items[0] * self.example_param
        return len(output_items[0])
```

```
"""
Embedded Python Blocks:

Each time this file is saved, GRC will instantiate the first
class it finds
to get ports and parameters of your block. The arguments to
__init__  will
be the parameters. All of them are required to have default
values!
"""

import numpy as np
from gnuradio import gr


class blk(gr.sync_block):  # other base classes are basic_block,
decim_block, interp_block
    """Embedded Python Block example - a simple multiply const"""

    def __init__(self, puissance=1.0):  # only default arguments
    here
        """arguments to this function show up as parameters in
        GRC"""
        gr.sync_block.__init__(
            self,
            name='Puissance',   # will show up in GRC
            in_sig=[np.float32],
            out_sig=[np.float32]
        )
        # if an attribute with the same name as a parameter is
        found,
        # a callback is registered (properties work, too).
        self.puissance = puissance

    def work(self, input_items, output_items):
        """example: multiply with constant"""
        output_items[0][:] = input_items[0] * self.puissance
        return len(output_items[0])
```

Figure 9 : À gauche le bloc Python exemple qui multiplie tous les échantillons d'entrée par une constante. À droite, le bloc modifie pour appliquer une puissance à tous les échantillons d'entrée.

Sans entrer trop dans les détails expliquons les grandes lignes de ce traitement : le flux d'échantillon étant continu, l'ordonnanceur GNU Radio décide d'envoyer en entrée de chaque bloc un nombre fini d'échantillon (les **input_items**) à chaque appel de la méthode **work()** qui réalise le traitement souhaité et produit les échantillons de sortie (les **output_items**) si au moins une sortie est déclarée dans le constructeur. Mettons en évidence les parties modifiées afin de réaliser le traitement souhaité :

- Dans le constructeur, **example_param** change de nom et devient **puissance** ;
- Le nom du bloc passe de « **Embedded Python Block** » à « **puissance** » ;
- Attention aux types d'entrées et sorties (in_sig et out_sig) : dans l'exemple le type est **np.complex64,** nous les passons en **np.float32** ;
- **self.example_param = example_param** devient **self.puissance = puissance** afin de correspondre à ce que l'on souhaite effectuer ;
- Dans la méthode **work(),** appelée régulièrement par l'ordonnanceur de GNU Radio (tant qu'il y a des échantillons d'entrée) la ligne permettant de calculer les échantillons de sortie est modifiée : **output_items[0][:] = input_items[0]**self.puissance** ou ** représente l'opérateur puissance en Python.

Nous n'avons donc plus qu'à ajouter ce bloc à la suite du traitement et à appliquer une puissance 0.5 (racine carrée). Le résultat obtenu, présenté sur la figure ci-dessous, est identique en tous points à celui fourni par le bloc *Complex to Mag*.

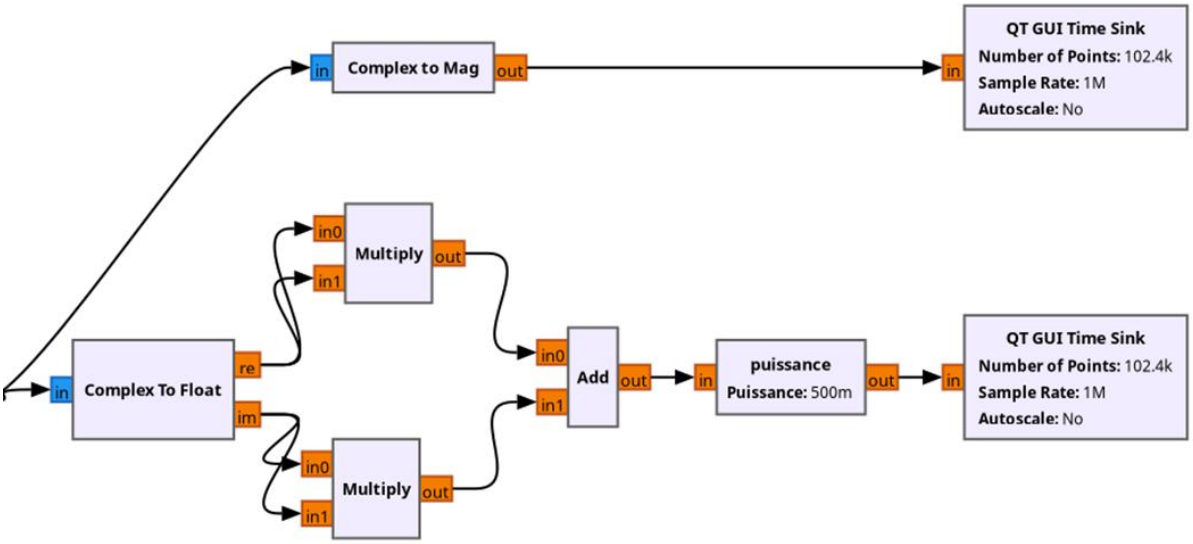


Figure 10 : Structure complète permettant de reproduire le traitement du bloc Complex To Mag

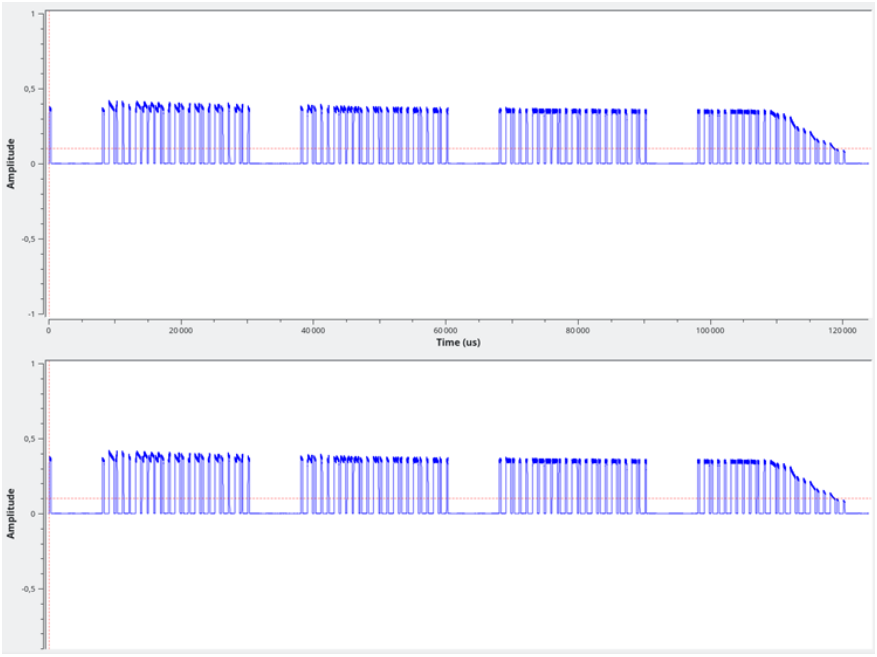


Figure 11 : Comparaison du résultat de notre traitement avec celui du Complex To Mag. Les deux signaux sont strictement identiques.

De cette manière on montre que les blocs de GNU Radio ne sont pas de simples « boîtes noires ». Leur fonctionnement repose sur des opérations mathématiques plus ou moins simples, que l'on peut reconstruire et comprendre pas à pas. Cette démarche permet de relier directement les outils logiciels aux concepts théoriques du traitement du signal, et contribue ainsi à une meilleure appropriation des notions étudiées.

3 - Analyse du signal et prolongements

3.1 - Décodage et caractérisation du signal

Une fois l'enveloppe extraite, le signal obtenu peut être observé dans le domaine temporel afin d'identifier la structure des trames émises par la sonnette. On distingue alors une succession d'impulsions correspondant aux bits transmis. La durée des états hauts et bas permet généralement de retrouver le codage utilisé par le constructeur.

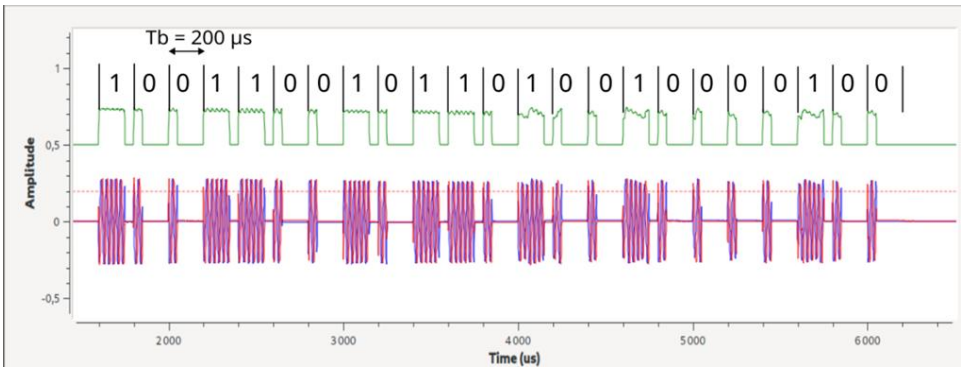


Figure 12 : Décodage du signal. Les informations binaires sont encodées en Pulse Length Encoding : un « 1 » correspond à un état haut de 150 µs puis un état bas de 50 µs et inversement pour le « 0 ».

Dans de nombreux systèmes simples de télécommande radio, l'information est transmise à l'aide de codages élémentaires reposant sur des variations de durée d'impulsion (PLE, Pulse Length Encoding) ou sur des transitions d'état.

GNU Radio permet alors d'aller plus loin en ajoutant des blocs de seuillage, de synchronisation ou encore de décodage personnalisé afin de reconstruire directement la suite binaire transmise.

3.2 - Reproduction et rejeu

Une fois le signal enregistré, il est possible de le rejouer à l'identique à l'aide d'un émetteur SDR compatible émission (HackRF One, USRP B206 mini-i, Pluto SDR, …).

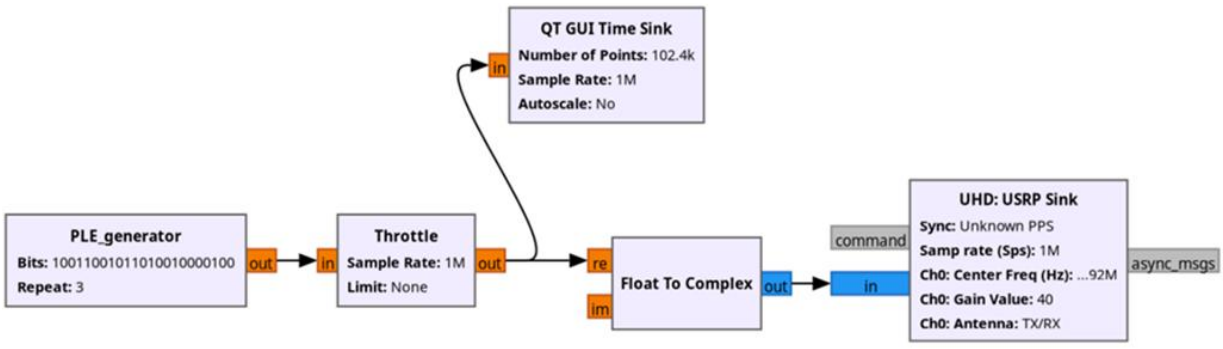
Figure 13 : À gauche : usrp b206 mini-i, au milieu : HackRF One, à droite : Pluto SDR. Trois dispositifs SDR compatibles émission.

Avertissement : *même dans la bande libre ISM 433, l'émission radiofréquence est strictement contrôlée. Avant de tenter une quelconque expérience de diffusion, assurez-vous d'être dans des conditions respectant les normes de l'ANFR (Agence Nationale des Fréquences) afin de ne perturber aucun système en fonctionnement au voisinage.*

La création d'un **Embedded Python Block** permettant de générer le signal en fonction de la suite binaire est un bon exercice. On déclare ainsi un nouveau bloc sans aucune entrée et avec une seule sortie (np.float32) qui génère le signal via deux fonctions **set_one(self)** et **set_zero(self)** puis l'envoie vers la sortie grâce à la fonction **work()**.

```python
import numpy as np
from gnuradio import gr

class blk(gr.sync_block):
    def __init__(self, bits="10011001011010010000100", repeat=3):
        gr.sync_block.__init__(
            self,
            name='PLE_generator',
            in_sig=None,
            out_sig=[np.float32])
        self.bits = bits
        self.repeat = repeat
        self.signal = []

        for _ in range(repeat):
            for b in bits:
                if b == "1":
                    self.set_one()
                else:
                    self.set_zero()

            self.signal += [0]*1000    # espace entre répétitions
        self.signal += [0]*20000      # espace entre trames
        self.signal = np.array(self.signal, dtype=np.float32)
        self.index = 0

    def set_one(self):        # HIGH 150 us + LOW 50 us
        self.signal += [1]*150
        self.signal += [0]*50

    def set_zero(self):         # HIGH 50 us + LOW 150 us
        self.signal += [1]*50
        self.signal += [0]*150

    def work(self, input_items, output_items):
        out = output_items[0]
        for i in range(len(out)):
            out[i] = self.signal[self.index]
            self.index += 1
            if self.index >= len(self.signal):
                self.index = 0
        return len(out)
```

Figure 14 : Bloc permettant de générer le signal dans GNU Radio



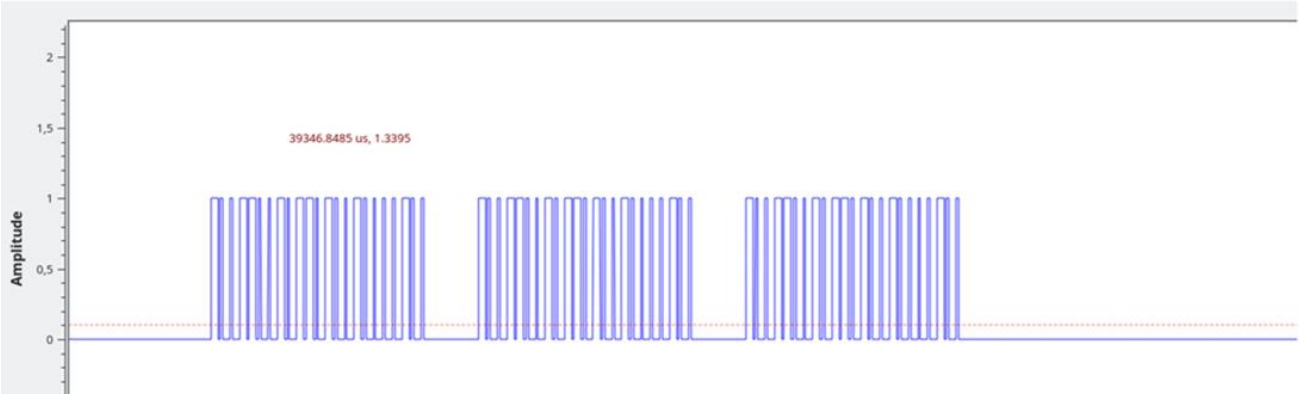


Figure 15 : Émission du signal avec un USRP. En haut, la structure permettant d'envoyer le signal sur l'usrp. En bas, le résultat observer dans le Time Sink.

Cette manipulation met en évidence le principe des attaques par rejeu (« replay attack »), consistant à réémettre un signal légitime précédemment capturé. Dans le cas de systèmes simples ne mettant en œuvre aucun mécanisme cryptographique ou code tournant (« rolling code »), cette réémission peut suffire à reproduire l'action d'origine. Cette expérience constitue une introduction intéressante aux problématiques de cybersécurité appliquées aux systèmes radiofréquences.

3.3 - Brouillage du système

Il est également possible d'étudier expérimentalement la sensibilité du système au brouillage radio.

Figure 16 : Principe du brouillage : on approche un HackRF émettant dans la bande ISM 433

Pour générer le bruit large bande, on utilise GNU Radio et le bloc Noise Source et on fait passer le flux dans un filtre passe-bas comme indiqué sur la figure 17.

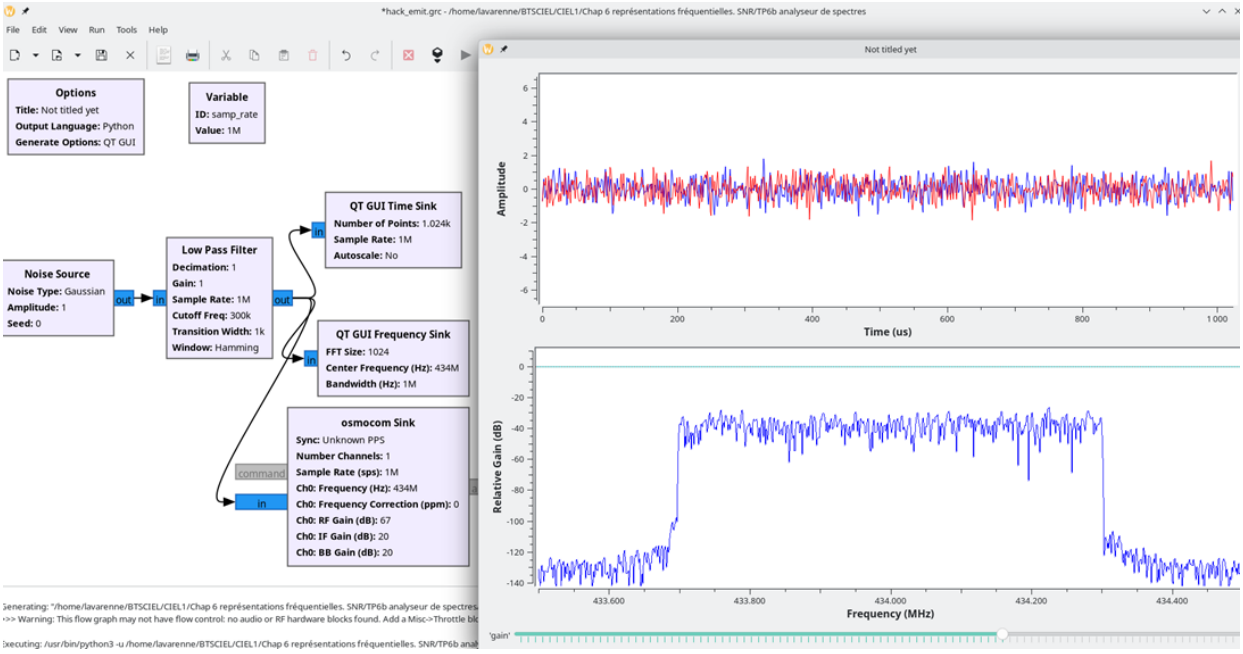


Figure 17 : Génération du bruit large bande et émission avec le HackRF (bloc Osmocom Sink).

Un analyseur de spectre peut être utile pour visualiser les niveaux avec et sans brouillage. Sur la figure 18 ci-dessous on constate bien que le signal du bouton de la sonnette est complètement noyé dans le bruit généré par le HackRF.

L'ajout d'un signal parasite dans la bande ISM 433 MHz dégrade rapidement la réception et peut empêcher complètement le décodage des trames de la sonnette. Ces expériences permettent d'illustrer concrètement l'importance du rapport signal sur bruit ainsi que la vulnérabilité potentielle des systèmes radio simples face aux perturbations volontaires ou non.

Bien entendu, le brouillage d'une simple sonnette sans fil présente ici surtout un intérêt pédagogique. Néanmoins, cette expérience met en évidence une problématique bien réelle : certaines alarmes domestiques bas de gamme, détecteurs sans fil ou automatismes utilisent encore des protocoles proches de ceux étudiés ici. Dans ce contexte, des dispositifs SDR peu coûteux associés à un générateur de bruit peuvent suffire à perturber localement les communications radio entre les différents équipements.

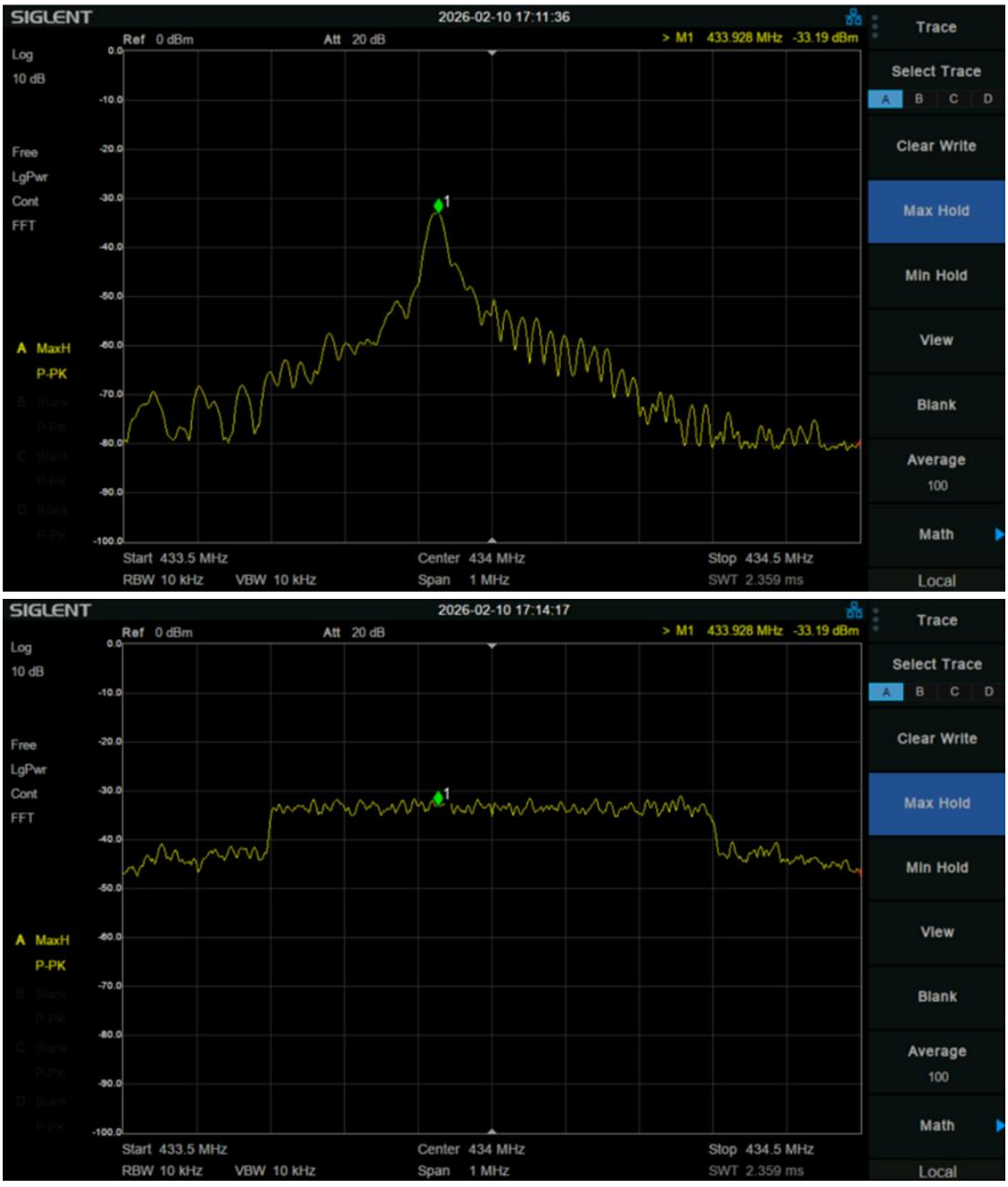


Figure 18 : Visualisation sur l'analyseur de spectre à proximité de la sonnette : en haut, sans brouillage, en bas avec le brouillage activé.

Les systèmes modernes cherchent généralement à se prémunir contre ces vulnérabilités grâce à différentes solutions :

- Détection de brouillage radio ;
- Changement dynamique de fréquence ;
- Redondance des transmissions ;
- Augmentation de la robustesse des protocoles ;
- Ou encore utilisation de liaisons filaires pour les fonctions critiques.

L'objectif de cette manipulation n'est donc évidemment pas de fournir une méthode d'attaque, mais de sensibiliser aux limites de certains systèmes radio simples et à l'importance de concevoir des architectures de communication robustes face aux perturbations intentionnelles ou non.

4 - Conclusion

L'utilisation de GNU Radio associée à un récepteur SDR permet de mettre en œuvre rapidement des expériences concrètes de réception et de démodulation radio. À travers l'exemple simple d'une sonnette sans fil, il devient possible d'introduire progressivement plusieurs notions fondamentales du traitement numérique du signal : représentation complexe I/Q, transposition en bande de base, analyse fréquentielle, extraction d'enveloppe et démodulation.

L'approche graphique de GNU Radio présente un intérêt pédagogique particulièrement important : chaque étape du traitement peut être isolée, visualisée et modifiée simplement, ce qui facilite fortement la compréhension des phénomènes étudiés et permet de relier directement les concepts théoriques aux signaux réels. Les expériences proposées montrent également qu'il est relativement simple, à l'aide de matériels SDR désormais accessibles, d'analyser, reproduire ou perturber certaines communications radio grand public. Ces manipulations permettent ainsi d'aborder concrètement plusieurs problématiques modernes liées à la robustesse et à la sécurité des transmissions sans fil. Au-delà de cet exemple volontairement simple, les mêmes principes se retrouvent dans de nombreux systèmes modernes de communication radio. Cette approche constitue ainsi une excellente porte d'entrée vers l'étude plus avancée des modulations numériques, des protocoles radio et de la radio logicielle.

Référence

[1]: Filtrage numérique avec GNU Radio : approche expérimentale, T. Lavarenne, juin 2026, https://sti.eduscol.education.fr/si-ens-paris-saclay/ressources_pedagogiques/filtrage-numerique-avec-gnu-radio-approche-experimentale

Le décodage ADS-B comme observable des propriétés physiques d'une antenne

La Revue 3EI
Culture Sciences de l'Ingénieur

Édité le 07/07/2026

école normale supérieure paris—saclay

Thomas ORTEGA[1]

[1] *Lycée Militaire de Saint-Cyr l'Ecole - BTS CIEL IR option spécifique cyber-défense*

Cette ressource fait partie du N°120 de La Revue 3EI du 3ème trimestre 2026.

Cet article a pour objet d'apporter les éléments d'une activité pédagogique pratique autour de la réception et du décodage des signaux ADS-B (Automatic Dependent Surveillance–Broadcast) émis par les aéronefs civils à 1090 MHz. Dans le contexte de l'enseignement en BTS CIEL, l'étude de ces signaux en radio logicielle avec GNU Radio permet aux étudiants de confronter les concepts enseignés en physique avec le concret.

L'idée centrale est ici d'utiliser le nombre d'avions détectés comme indicateur observable et motivant de la performance de chaque antenne réalisée.

En utilisant un récepteur de radio logicielle peu onéreux et le logiciel open-source GNU Radio, les étudiants construisent leurs propres antennes, reçoivent en temps réel les trames de position des avions, les décodent et comparent leurs résultats avec les données publiées par exemple par Flightradar24. Cette approche développe simultanément des compétences en propagation radio, en traitement du signal numérique, en protocoles de communication, en programmation Python et en cybersécurité.

La radio logicielle (Software Defined Radio, SDR) offre un terrain d'exploration exceptionnel pour le co-enseignement Physique–Informatique en BTS CIEL.

Cet article repose sur la mise en œuvre de GNU Radio en enseignement de physique par Thomas LAVARENNE qu'il a présentée en particulier à l'European GNU Radio Days [1].

1 - Intérêt pédagogique : pourquoi l'ADS-B en co-enseignement BTS CIEL ?

Le BTS CIEL (Cybersécurité, Informatique et réseaux, Électronique), dans ses deux options – option A (Informatique et réseaux) et option B (Électronique et réseaux) – place les enseignants devant un défi structurant : co-animer, à deux enseignants, des contenus de physique appliquée (propagation des ondes, antennes, chaîne de réception radio) avec des contenus informatiques (traitement numérique du signal, protocoles, programmation) autour des mêmes compétences. Le co-enseignement constitue une réponse naturelle, à condition de disposer d'un objet d'étude commun, concret et motivant pour les étudiants, et accessible aux deux enseignants.

L'ADS-B répond parfaitement à ces exigences, pour plusieurs raisons :

- accessibilité et gratuité du signal : émis avec une puissance de l'ordre de 200W, le signal est assez facile à recevoir pour tout avion commercial en vol au-dessus du lieu

d'enseignement. Le signal est non chiffré, non soumis à autorisation de réception, et disponible sans abonnement.

- résultat visible et vérifiable en temps réel : les avions décodés s'affichent sur une carte et peuvent être immédiatement comparés avec Flightradar24, ce qui procure une satisfaction immédiate et valide concrètement le travail des étudiants.
- fil directeur unique : le nombre d'avions détectés. Ce compteur synthétise en une seule grandeur l'ensemble de la chaîne (qualité de l'antenne, placement, gain RF, seuil de décodage) et constitue une métrique de performance objective et concrète accessible à tous les étudiants, quel que soit leur niveau.
- coût réduit : un clef USB RTL-SDR (~40 €) suffit pour une version allégée et des antennes fabriquées par les étudiants à partir d'un connecteur SMA, de câble coaxial et de fil de cuivre (~4 € chacune).
- richesse des contenus pluridisciplinaires : le TP mobilise la transmission des ondes électromagnétiques, la caractérisation temporelle et fréquentielle des signaux, les caractéristiques des antennes et le lien entre longueur d'onde et longueur de l'antenne, la transmission numérique en bande de base et le codage Manchester, les protocoles de communication, la vérification CRC, la programmation Python — autant de points inclus dans le référentiel BTS CIEL.

Plusieurs compétences du BTS CIEL option A ou option B sont susceptibles d'être mises en œuvre :

• Supports de propagation dans les réseaux informatiques (compétences C04 analyse et C05 conception).

• Caractéristiques des communications présentes dans les systèmes informatiques étudiés (compétences validation C06, installation C09 et exploitation C10).

• Principes de transmission en espace libre ou guidée dans les réseaux informatiques (compétences analyse C04 et conception C05).

La documentation réglementaire et technique peut aussi faire l'objet d'une exploitation en co-enseignement anglais (ESLA) en exploitant les documents techniques ou vidéo [2] en langue anglaise.

2 - Régulation du trafic aérien et transpondeurs

2.1 - Contexte historique : du radar primaire à l'ADS-B

La surveillance du trafic aérien repose historiquement sur deux technologies complémentaires :

- Le radar primaire (PSR, Primary Surveillance Radar) détecte les aéronefs par réflexion d'une impulsion radio (principe de l'écho). L'onde est émise par une antenne souvent en rotation continue. La différence de temps entre l'émission et la réception fournit la position angulaire et la distance — sans identification.
- Le radar secondaire (SSR, Secondary Surveillance Radar) interroge activement les transpondeurs des aéronefs à 1030 MHz en émettant deux ou trois impulsions radio. Ceux-ci répondent au code d'interrogation à 1090 MHz avec leur code d'identification (squawk ident en Mode A) et leur altitude telle que mesurée dans l'aéronef (par incrément de 100 pieds en mode C). La séparation temporelle entre les impulsions émises détermine le mode utilisé et donc la question posée.

Le Mode S (Mode Select Beacon System, développé au MIT Lincoln Laboratory dans les années 1970 et déployé dans les années 1990, améliore le SSR en ajoutant la fonctionnalité d'interrogation sélective de chaque aéronef via une adresse unique de 24 bits, identifiant fixé par l''Organisation de l'aviation civile internationale (OACI) (ICAO en anglais), capable d'adresser jusqu'à 16 777 216 aéronefs distincts. Les codes ICAO sont accessibles sur des bases en lignes comme OpenSky [3].

L'ADS-B (Automatic Dependent Surveillance–Broadcast) représente l'étape suivante : l'aéronef diffuse spontanément ses données de vol à intervalles réguliers, sans interrogation, sur la fréquence 1090 MHz. C'est précisément sur cette émission radio que repose la suite de cet article : il n'est pas nécessaire d'être à proximité d'un aéroport pour l'étudier.

L'acronyme ADS-B se décompose ainsi :

- *Automatic* : aucune action du pilote ni du contrôleur n'est requise ;
- *Dependent* : le système dépend des récepteurs GNSS embarqués par les aéronefs pour calculer la position ;
- *Surveillance* : il remplit une fonction de surveillance du trafic ;
- *Broadcast* : la diffusion est omnidirectionnelle, à destination de toute station capable de recevoir.

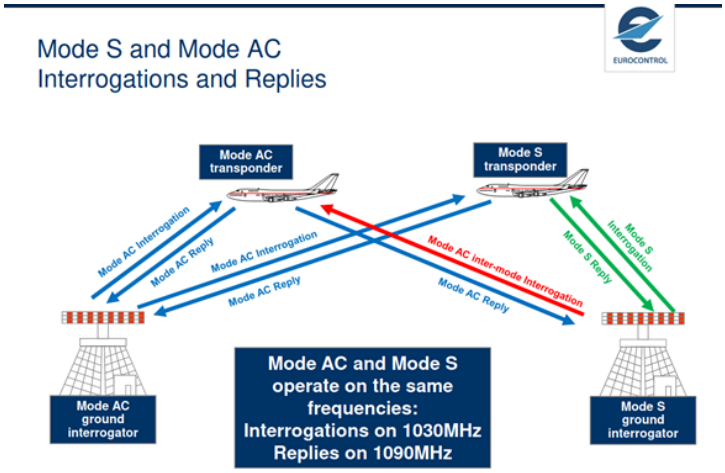


Figure 1 : Modes S et A/C, source ICAO [4]

En utilisant l'interrogation sélective, le protocole améliore considérablement la capacité de transfert de données sur le canal de la porteuse (1090MHz). Mais ce qui nous intéressera dans cet article, c'est une approche passive basée sur le décodage des informations émises en broadcast par les aéronefs.

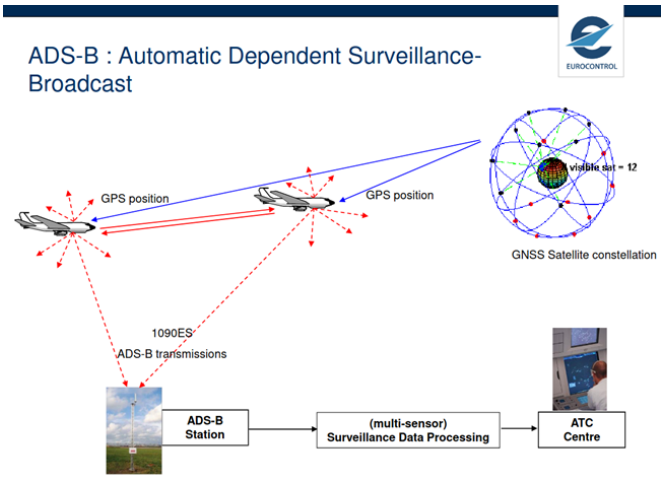


Figure 2 : ADS-B, source ICAO [4]

Encadré - Extrait de The 1090MHz Riddle (Second Edition) - Junzi Sun [5]

Automatic Dependent Surveillance-Broadcast (ADS-B) is a surveillance technology designed to allow aircraft to broadcast their flight state periodically without the need for interrogation.

The word automatic refers to the fact that no inputs from controllers or pilots are required.

The word dependent indicates this technology depends on information from other onboard systems, such as air data systems and navigation systems.

ADS-B only become possible because of Global Navigation Satellite Systems (GNSS), such as Global Positioning System (GPS), which is the first GNSS system made available for civil usage by the USA after the Korean Air Lines Flight 007 accident in 1983.

Common aircraft state parameters included in ADS-B are position, altitude, and speed.

The position is determined by GNSS and air data systems.

The velocity is derived from the GNSS position and the inertial measurement system.

The altitude information includes both barometric altitude and GNSS altitude. The barometric altitude is provided by the air data system.

In addition to these primary state parameters, ADS-B also allows other information to be broadcast, for example, aircraft call-sign, accuracy indicators, integrity indicators, and operational status.

It is worth emphasizing that ADS-B is a broadcast-based surveillance system. Unlike other types of Mode S surveillance, no surveillance interrogation is required. The decoding of ADS-B messages is also easier than other Mode S messages because the message type and structure are clear for each individual message.

Different types of ADS-B messages are identified by Type Codes. The structures of these messages are all defined in ICAO documents.

2.2 - Déploiement mondial et obligation réglementaire

Depuis le 7 juin 2020, tout aéronef civil qui navigue dans l'espace aérien européen au-dessus du plancher de vol FL245 (environ 7 500 m) doit être équipé d'un transpondeur ADS-B opérationnel (règlement d'exécution UE 2017/386) (https://eur-lex.europa.eu/eli/reg_impl/2017/386/oj/eng). Aux États-Unis, la réglementation FAA impose l'ADS-B Out depuis janvier 2020 pour les espaces aériens contrôlés. La quasi-totalité des avions commerciaux émet donc des signaux ADS-B en continu.

2.3 - Les sites de surveillance participative

Plusieurs sites dont Flightradar24, FlightAware ou OpenSky illustrent parfaitement le potentiel de l'ADS-B : il agrège les données de milliers de stations de réception réparties dans le monde entier (dont beaucoup sont des particuliers utilisant un simple récepteur radio logiciel RTL-SDR) pour afficher en quasi-temps-réel la position de tous les avions visibles. Cette infrastructure participative repose entièrement sur la réception passive du signal ADS-B - exactement ce que les étudiants mettent en œuvre dans ce TP. Ils constituent un outil externe de validation des montages des étudiants.

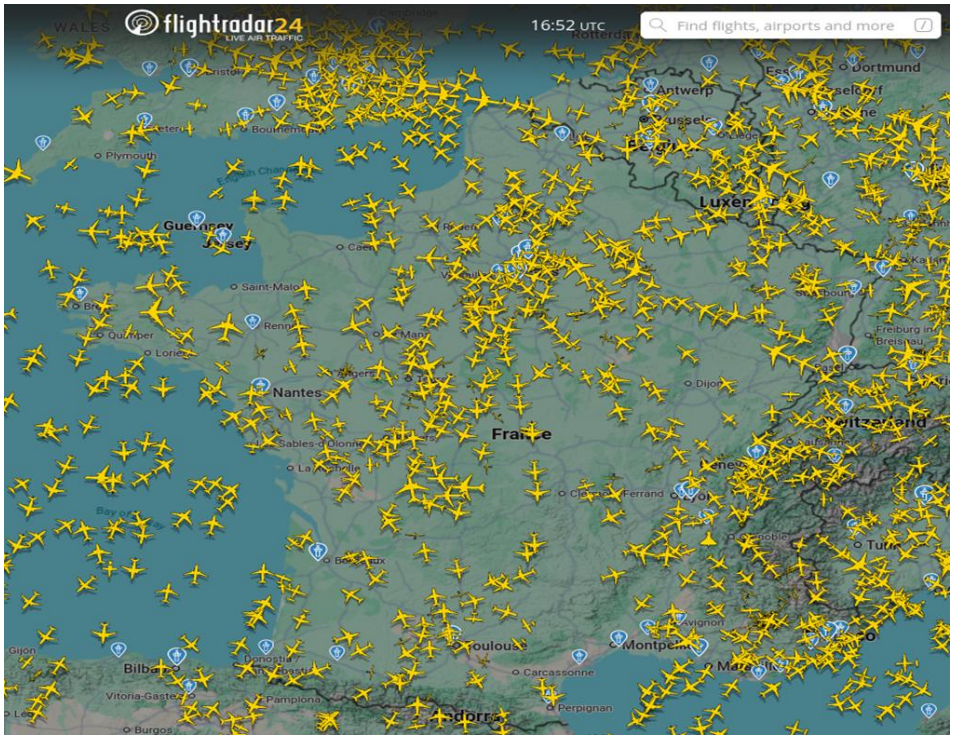


Figure 3 : Capture d'écran de FlightRadar24 montrant les aéronefs au-dessus de la France, source www.flightradar24.com

3 - Structure des trames ADS-B

3.1 - Les sites de surveillance participative

Le signal ADS-B est transmis à 1090 MHz avec une modulation PPM (*Pulse Position Modulation*) :

- Un bit «1» est représenté par une impulsion dans la première moitié de la période, suivie d'un silence ;
- Un bit «0» est représenté par un silence dans la première moitié, suivi d'une impulsion.

Le débit binaire est de 1 Mbit/s, soit une durée de bit T_b = 1 µs. Chaque trame est précédée d'un préambule fixe de 8 µs (séquence binaire 10100000 01000000, A040 en hexadécimal) permettant la synchronisation. Ce préambule sera utilisé ultérieurement avec GNU Radio pour identifier le début des trames à capturer.

Encadré - Codage Manchester et PPM

Le signal ADS-B utilise la Modulation d'impulsions en position ou PPM (Pulse Position Modulation), parfois assimilée au codage Manchester car les deux codages encodent chaque bit par une transition. En PPM stricte, c'est la position de l'impulsion au sein de la période bit qui porte l'information (première ou deuxième moitié), alors qu'en Manchester c'est la direction de la transition (montante ou descendante). Les deux approches nécessitent 2 échantillons par bit pour être décodées correctement, ce qui explique le taux d'échantillonnage de 2 MHz utilisé dans GNU Radio.

3.2 - Structure d'une trame ADS-B

Une trame ADS-B complète comporte **112 bits** organisés comme suit :

DF (5 bits)	CA (3bits)	ICAO (24 bits)	ME / DATA (56)	PI (24bits)

Avec la décomposition suivante :

Champ	Bits	Description
DF	1–5	*Downlink Format* : pour les trames ADS-B mode S Extended Squitter des aéronefs civils, DF vaut 17 (10001 en binaire)
CA	6–8	*Capability* : capacité du transpondeur (0 à 7)
ICAO	9–32	Adresse unique de l'aéronef (24 bits, hexadécimal)
ME	33–88	*Message Extended squitter* : données de position, vitesse, identification
PI	89–112	*Parity/Interrogator* : CRC-24 pour contrôle d'intégrité

On retiendra de la valeur de CA :

CA=0 : transpondeur type 1

CA=4 : transpondeur type 2, l'aéronef est au sol

CA=5 : transpondeur type 2, l'aéronef est en vol

CA=6 ou 7 : l'aéronef est en vol ou au sol

Le détail complet des trames est détaillé dans l'ouvrage *The 1090 Megahertz Riddle [5].*

Les 5 premiers bits du champ ME constituent le Type Code (TC), qui identifie le type de message :

TC	Contenu
1–4	Identification et catégorie (indicatif)
5–8	Position en surface
9–18	Position en vol (altitude barométrique)
19	Vitesse en vol
20–22	Position en vol (altitude GNSS)
28	Statut aéronef
31	Statut opérationnel

3.3 - Contrôle d'intégrité : le CRC-24

Le champ PI de 24 bits est un **CRC** (*Cyclic Redundancy Check*) calculé sur les 88 premiers bits de la trame. Le polynôme générateur utilisé est :

$$G(x) = x^{24} + x^{23} + x^{22} + x^{17} + x^{10} + x^3 + x + 1$$

Ce CRC permet de détecter (mais pas de corriger) les erreurs de transmission. Un message dont le CRC ne correspond pas doit être rejeté. En pratique, avec la modulation PPM et le rapport signal/bruit typique en réception, le taux d'erreur sur les trames reçues peut être significatif, surtout pour les aéronefs lointains.

3.4 - Exemple de décodage du début d'une trame

Prenons la trame binaire reçue par un récepteur :

10001 101 010010000100 000011010110 00100 0000010110011 00001101110001110000 110010110011100000 | 010101110110 000010011000 |

soit en héxadécimal : 8D4840D6 2202CC37 1C32CE0 576098

- **DF** = 10001 = 17 en décimal→ Extended Squitter ADS-B ✓
- **CA** = 101 = 5 en décimal → transpondeur niveau 2 et l'aéronef est en vol
- **ICAO** = 4840D6 → adresse ICAO hexadécimale 24bit de l'aéronef.
  La consultation de la base OpenSky Network, Flightradar24 ou mieux planespotter via l'URl : https://www.planespotters.net/hex/4840D6 permet de confirmer qu'il s'agit bien d'un vol KLM , appareil Fokker 70 dont l'indicatif est PH-KZD
- TC (5 premiers bits de ME) = 00100 = 4 → message d'identification

Le décodage de cette partie sera suffisant pour l'activité proposée. Elle pourra toutefois être poursuivie avec le décodage de la position, l'altitude et la vitesse.

3.5 - Position, altitude, vitesse

Le décodage de la position géographique d'un avion à partir des trames ADS-B est un point intéressant qui peut être exploité en co-enseignement mathématique-informatique. En effet les messages de type TC=9 à 22 transmettent latitude et longitude non pas directement en degrés décimaux, mais dans un format compact appelé CPR (Compact Position Reporting).

Encoder une latitude avec une précision de 5 m au niveau mondial nécessiterait 26 bits pour chaque coordonnée, soit 52 bits pour la paire (latitude, longitude). Le format CPR permet d'atteindre la même précision avec seulement 17 bits par coordonnée (34 bits au total), en exploitant le fait que des messages successifs d'un même avion sont reçus à intervalles rapprochés et qu'on n'a donc pas besoin d'un repère absolu et univoque à l'échelle de la planète. On alterne deux grilles de codage légèrement différentes : la trame paire (F=0) et la trame impaire (F=1). La résolution globale résulte de la combinaison des deux trames [6].

Deux sources d'altitudes sont transmises : l'altitude calculée par l'aéronef à partir des constellations satellites GNSS et l'altitude barométrique issue des mesures de pression par l'aéronef. Ces deux modes de calcul de l'altitude seront utilisés aussi pour calcul la vitesse ascensionnelle dans la partie vitesse.

En ce qui concerne la vitesse, trois composants sont transmis par ADS-B :

DEWS (10 bits) : composante Est-Ouest de la vitesse sol
DNS (10 bits) : composante Nord-Sud de la vitesse sol
VR (10 bits) : taux de montée/descente vertical

Il est donc aussi possible d'en faire calculer le cap de l'aéronef.

Le détail complet des trames est détaillé dans l'ouvrage *The 1090 Megahertz Riddle* [5].

4 - Evaluation de la réception théorique

4.1 - Propagation des ondes de la bande L

La fréquence 1090 MHz appartient à la bande L (définie par les fréquences de 1 à 2 GHz). Cette bande de fréquence est utilisée par des service de radio astronomie, des usages militaires, de la téléphonie mobile (LTE 1800), quelques radars spécifiques, une partie de radio numérique (DAB), les constellations satellite de positionnement GNSS (GPS/Galileo/GLONASS/Beidou) et ADS-B.

Pour la fréquence 1090MHz, la longueur d'onde est $\lambda = \frac{c}{f} = \frac{3\times10^8}{1{,}09\times10^9} \approx 27{,}5cm$

Cette bande est caractérisée par propagation quasi-optique. La pluie et les nuages ont une atténuation négligeable à cette fréquence. Les obstacles (collines, bâtiments) créent des zones d'ombre importantes. C'est pourquoi l'élévation de l'antenne est le paramètre le plus critique pour maximiser le nombre d'avions détectés. Il faudra donc veiller à placer les antennes sur un même niveau pour avoir des résultats significatifs.

4.2 - Portée géométrique maximale

Le signal ADS-B utilise une porteuse à 1090 MHz. Cette fréquence est située dans la bande L qui est connue pour se propager en ligne de vue. En supposant l'absence d'obstacles et une puissance d'émission suffisante, la portée maximale est alors géométriquement limitée par la courbure de la Terre :

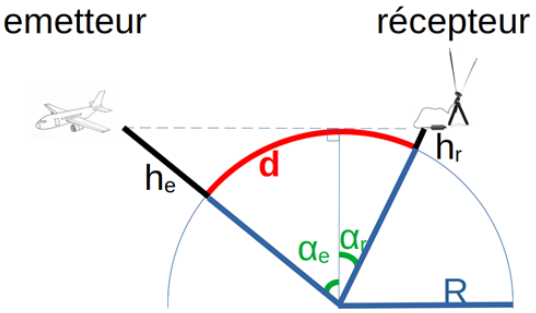


Figure 4 : Positions de l'émetteur et du récepteur sur la courbure de la Terre

Avec : R = 6,371.106 m (rayon terrestre moyen)

he = altitude de l'aéronef au-dessus du niveau de la mer (en m)

hr = altitude de l'antenne réceptrice (en m)

Longueur l'arc de cercle $d = (\alpha_e + \alpha_r) \times R$

$$d = \left(arcos\left(\frac{R}{R+h_r}\right) + arcos\left(\frac{R}{R+h_e}\right)\right) \times R$$

Ainsi pour un récepteur placé à 50 m d'altitude et un avion à 10 km d'altitude, la distance maximale serait de l'ordre de 400km :

$$d = \left(arccos\left(\frac{6{,}371.10^6}{6{,}371.10^6+50}\right) + arccos\left(\frac{6{,}371.10^6}{6{,}371.10^6+10^6}\right)\right) \times 6{,}371.10^6 \approx 397.10^3 \approx 397km$$

Cette portée théorique n'est cependant qu'un maximum géométrique. En conditions réelles, le modèle de Friis impose que la portée dépende aussi de la puissance d'émission, des gains d'antenne et des pertes de trajet.

4.3 - Bilan de liaison selon Friis

La formule de transmission de Friis donne la puissance reçue en espace libre :

$$P_r = P_e + G_e + G_r - \alpha$$

avec :

- Pr = Puissance reçue (dBm)
- Pe = Puissance émise (dBm)
- Ge = Gain d'antenne émission (dBi)
- Ge = Gain d'antenne réception (dBi)
- α = perte de propagation (dB) en espace libre $\alpha = 20 \times log_{10} \frac{4\pi\delta}{\lambda}$

avec δ = Distance entre l'émetteur et le récepteur (en mètres)

λ = Longueur d'onde (en mètres)

4.4 - Application numérique

Puissance émise (P_e)

Figure 5 : Façade du transpondeur Garmin GTX™ 345

La fiche technique du transpondeur Garmin GTX 345 indique une puissance d'émission du signal de 200W soit, (en utilisant la formule de conversion P(dBm)=10log10(1000xP(W)) Pe=+53dBm.

Gain des antennes

On peut dans un premier temps prendre comme approximation qu'une antenne d'émission a un gain nul (Ge=3dBi) car incluse dans le système du transpondeur et donc dans ses caractéristiques. On peut aussi prendre en approximation qu'une antenne de réception type spider λ/4 a un gain de 3dBi (Gr=3dBi). La qualité des antennes de réception pourra faire l'objet de mesures fines grâce à un équipement NanoVNA (Network Vector Analyzer de poche, accessible chez des fournisseurs habituels pour moins d'une centaine d'euros).

Sensibilité du récepteur

Des mesures effectuées sur les clefs RTL-SDR montrent une sensibilité de l'ordre de -129dBm à 1GHz. (cf blog RTL SDR https://www.rtl-sdr.com/measurements-on-rtl-sdr-e4000-and-r820t-dvb-t-dongles-image-rejection-internal-signals-sensitivity-overload-1db-compression-intermodulation/).

Dans des conditions de réception difficile, ou à titre d'extension de TP pour les élèves qui ont avancé plus rapidement, elle peut être augmentée d'environ 6 dB par l'ajout d'un amplificateur LNA (low-noise amplifier) externe généraliste, ou 16 dB avec un amplificateur LNA et un filtre passe bande centré sur 1090 MHz (ces équipements accessibles chez les fournisseurs habituels pour quelques dizaines d'euros).

Pour une distance émetteur-récepteur δ = 200 km :

$$\alpha = 20 \times log_{10} \frac{4 \times \pi \times 200{,}000}{0{,}275} = 20 \times log_{10}(9{,}12.10^6) \approx 139{,}2dB$$
$$P_r = 53 + 0 + 3 - 139{,}2 = -83{,}2dBm$$

La puissance du signal reçu est bien au-dessus du plancher de tous les récepteurs la détection est assurée.

Pour une distance émetteur-récepteur δ = 500 km :

$$\alpha = 20 \times log_{10} \frac{4 \times \pi \times 500000}{0{,}275} \approx 147{,}2dB$$
$$P_r = 53 + 0 + 3 - 147{,}2 = -91{,}2dBm$$

La puissance du signal reçu est supérieure au seuil de sensibilité des récepteurs mais elle sera conditionnée par le dégagement de l'antenne car on est à la limite géométrique pour un aéronef volant à 10 km d'altitude.

Conclusion du bilan de liaison de Friis : la portée Friis est presque toujours inférieure à la portée géométrique (sauf pour les avions très proches). Pour les avions en croisière (10 km d'altitude), la limite pratique avec un RTL-SDR et une antenne de type *spider* est de l'ordre de 350–400 km.

4.5 - Exploitation pédagogique

Les étudiants comprennent ici que la chaîne de réception a deux limites indépendantes — géométrique (courbure de la Terre) et énergétique (Friis) — et que c'est la plus contraignante qui s'impose. Pour les avions en croisière, c'est presque toujours la limite géométrique qui domine.

Les étudiants peuvent, à partir de la formule géométrique, prédire la portée maximale attendue depuis leur lycée en fonction de l'altitude des avions en croisière (typiquement 200km à 12200km). Ils tracent ensuite la courbe théorique et la comparent avec les données réellement reçues (position GPS des avions décodés). L'écart entre théorie et mesure ouvre la discussion sur les obstacles locaux, le relief et la qualité de l'antenne.

5 - Importance des antennes

5.1 - Pourquoi l'antenne est-elle le facteur clef ?

Dans la chaîne de réception ADS-B, le récepteur SDR est suffisamment performant pour ne pas être le maillon limitant. La **qualité de l'antenne** et son **placement** déterminent à eux seuls la majorité de la performance : une antenne bien accordée à 1090 MHz capte des avions bien au-delà de l'horizon, là où une antenne TNT désaccordée ou un simple fil ne reçoit que les avions proches.

C'est l'idée centrale du TP : **mesurer le nombre d'avions détectés sur une fenêtre de temps fixe** (par exemple 15 minutes) permet de comparer objectivement et de façon reproductible les performances de différentes antennes réalisées par les étudiants.

5.2 - Appel : longueur d'onde à 1090 MHz

$$\lambda = \frac{c}{f} = \frac{3 \times 10^8}{1{,}09 \times 10^9} \approx 27{,}5cm$$

Les longueurs caractéristiques des antennes accordées à 1090 MHz sont donc :

λ/2 = **13,75 cm** (demi-onde, dipôle ou monopole)

λ/4 = **6,875 cm** (quart d'onde, élément d'une antenne planaire)

5.3 - Types d'antennes réalisables par les étudiants

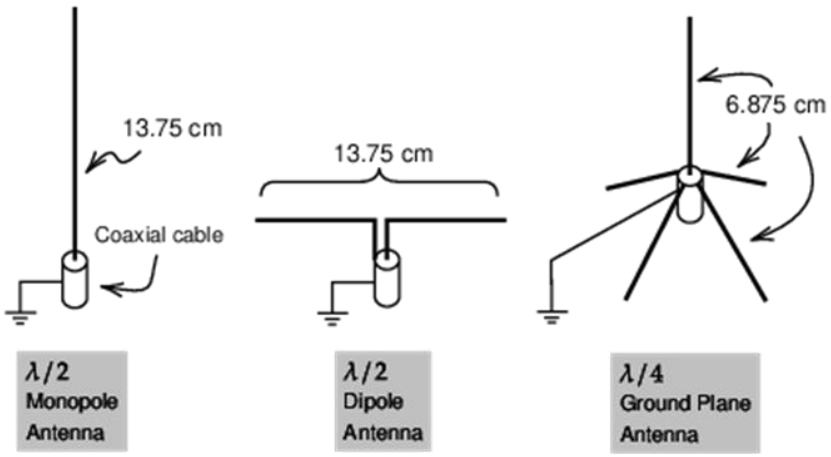


Figure 6 : Schéma des de différentes formes d'antenne, source The 1090MHz Riddle [5]

Antenne monopole λ/2 : un simple fil de cuivre 13,75 cm soudé sur un connecteur SMA. Facile à réaliser, gain nul (0 dBi), directivité quasi-omnidirectionnelle dans le plan horizontal. C'est la référence de base.

Antenne dipôle λ/2 : deux brins de 13,75 cm montés en opposition sur un câble coaxial, le brin central et la tresse formant les deux éléments. Gain théorique ≈ 2,15 dBi. Légèrement supérieure au monopole. L'utilisation d'un double domino à vis ou de deux connecteurs rapides type Wago simplifie le montage mécanique.

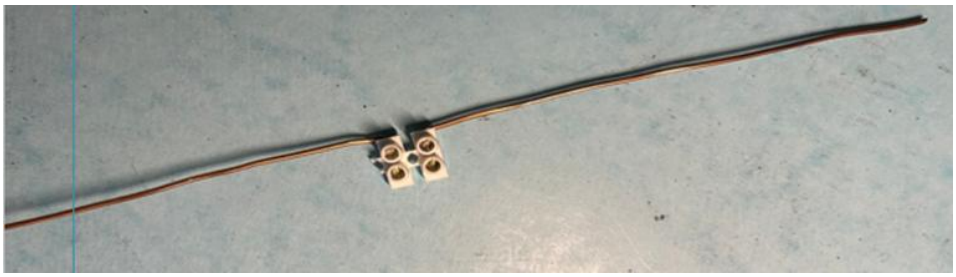


Figure 7 : Montage d'un dipôle à l'aide d'un domino

Antenne planaire quart d'onde (*Spider*) : un élément vertical central de λ/4 = 6,875 cm entouré de 4 à 6 radians horizontaux de même longueur. C'est l'antenne de référence pour la réception ADS-B. Gain ≈ 2–3 dBi, omnidirectionnel dans le plan horizontal, cette antenne est très adaptée à la réception depuis toutes les directions.

Antenne Yagi TNT de récupération : une antenne de télévision terrestre peut être détournée pour 1090 MHz si ses éléments sont approximativement accordés. Gain élevé (6–10 dBi) mais directionnelle : efficace uniquement dans la direction pointée, ce qui réduit le nombre d'avions vus. Les fréquences de la TNT sont comprises entre 470MHz et 790MHz ; ces antennes comportent souvent un filtre démontable plus ou moins facilement et dont on peut étudier l'influence.

Antenne planaire demi-onde : variante de l'antenne type *spider* avec des radians plus longs (λ/2), légèrement différente en termes de diagramme de rayonnement.

5.4 - Réalisation pratique par les étudiants

Matériaux nécessaires par groupe :

- 1 mètre de câble coaxial (pour limiter les soudures, on peut prendre un cordon avec un connecteur SMA déjà serti à une extrémité)
- 1 connecteur SMA mâle à souder par type d'antenne
- Fil de cuivre rigide 1,5 mm² (électricité)
- Étain et fer à souder
- Mètre ruban, pince coupante, dénudeur

Conseils pratiques :

- Les tolérances de ±2 mm sont acceptables à 1090 MHz.
- Un multimètre en mode « continuité » permet de vérifier qu'il n'y a pas de court-circuit entre l'âme et la tresse.
- L'utilisation d'un analyseur d'antenne type NanoVNA, (*Network Vector Analyzer* de poche, accessible chez des fournisseurs habituels pour moins d'une centaine d'euros) permet de mesurer le taux d'onde stationnaire (TOS ou SWR) et les pertes par réflexion (Return Loss) d'une antenne en fonction de la fréquence. Pour les étudiants BTS CIEL, c'est un outil de validation expérimentale direct entre le calcul théorique et la performance mesurée.
- Dans notre contexte pédagogique, le NanoVNA valide le dimensionnement (cotes correctes) de l'antenne, et le comptage d'avions mesure la performance globale incluant le gain.
  Le NanoVNA est une aide précieuse pour l'accord des antennes :

  - Résonance trop haute (ex. : minimum SWR à 1150 MHz)

  → l'antenne est trop courte. Allonger légèrement les brins (+1 à 2 mm).

  - Résonance trop basse (ex. : minimum SWR à 1040 MHz)

  → l'antenne est trop longue. Raccourcir les brins.

  - SWR minimum > 2 à 1090 MHz

  → vérifier la qualité des soudures, la continuité et l'absence de court-circuit partiel.

On peut fixer comme objectif pour d'obtenir une mesure SWR < 1,5 à 1090 MHz

6 - Mise en œuvre de GNU Radio pour la réception des trames

6.1 - Présentation de GNU Radio

GNU Radio est un framework open-source de traitement du signal, disponible sous licence GPL. Il permet de construire des diagrammes de flux (*flowgraphs*) en connectant graphiquement des blocs de traitement dans l'interface GNU Radio Companion (GRC). Chaque diagramme génère automatiquement le code Python correspondant, exécutable en ligne de commande de manière indépendante.

GNUR Radio permet de mettre à disposition de manière simple et utilisable des outils puissants du traitement du signal.

Son installation sous Linux Ubuntu/Debian est simple : sudo apt-get install gnuradio gr-osmosdr

6.2 - La chaîne de traitement GNU Radio

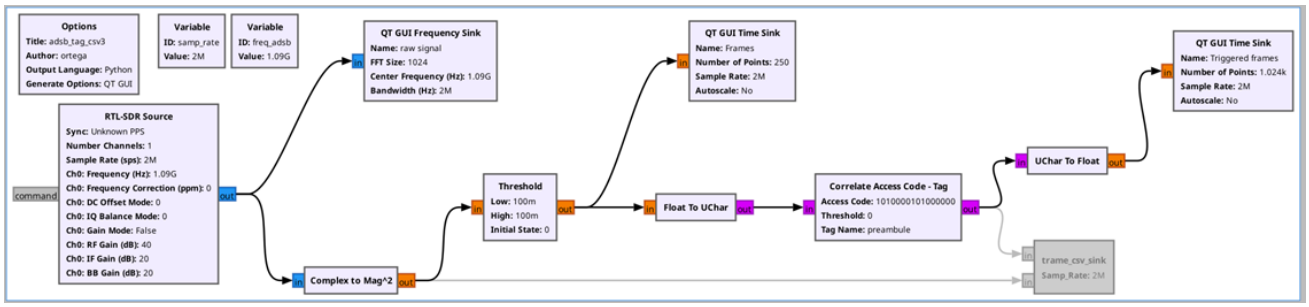


Figure 8 : Flowchart de la réception du signal sous GNU Radio Companion

La chaîne de traitement ADS-B dans GNU Radio comprend les étapes suivantes :

- Variables : sample_rate : le taux d'échantillonage : 2.10^6
- Variables : freq_adsb : fréquence de la porteuse ADSB : 1090.10^6

Étape 1 - Source SDR (bloc osmocom Source) :

Fréquence centrale : freq_adsb
Taux d'échantillonnage (samp_rate) : samp_rate
Gain RF : 20-40 dB (à ajuster selon la configuration)
Gain IF : 20 dB
Gain BB : 20 dB

Étape 1bis - Affichage brut du signal reçu (bloc QT GUI Frequency Sink) :

On affiche le signal brut.
FFT Size 1024
Center Frequency : freq_adsb
Bandwidth : 2MHz

Étape 2 - Calcul de l'amplitude (bloc Complex to Mag²) :

Transforme le signal complexe I+jQ en amplitude instantanée

C'est l'enveloppe du signal, qui révèle les impulsions PPM.

Étape 3 - Seuillage (bloc Threshold) :

Compare chaque échantillon à un seuil (typiquement 100 pour un niveau normalisé) et produit un signal binaire : 1 si l'amplitude est dans l'intervalle, 0 sinon. Cet intervalle doit être réglé expérimentalement selon le niveau de bruit ambiant. Cela permet de filtrer et d'extraire les bits d'information du bruit.

Étape 3bis - Affichage brut des trames reçues (bloc QT GUI Time Sink) :

Number of points 250 (on affiche 250 bits)
Sample Rate : samp_rate

Étape 4 - Conversion de type (Float to Uchar) :

Conversion des flottants en caractères non signés pour le bloc de corrélation suivant.

Étape 5 - Corrélation avec le préambule (Correlate Access Code - Tag) :

Ce bloc central recherche dans le flux binaire la séquence fixe du préambule ADS-B : 10100000 01000000 (en représentation binaire sur 2 échantillons/bit : 1010000001000000...)

Lorsque la séquence est reconnue, il attache un **tag** nommé « preambule» à la position correspondante du flux, signalant le début d'une trame.

Access Code : 1010000101000000
Treshold : 0
Tag Name : preamble

Étape 6 - Conversion de type (Uchar to Float) :

Conversion des caractères non signés en flottants pour le bloc d'affichage suivant.

Étape 7 - Visualisation (QT GUI Time Sink) :

Affiche en temps réel le signal après seuillage, avec un déclenchement synchronisé sur les tags «burst». L'étudiant peut observer visuellement la structure de la trame (préambule, champs DF/CA/ICAO, données).

Number of points 1024 (on affiche 1024 bits)
Sample Rate : samp_rate
Sample Rate : samp_rate
Trigger / Trigger Tag Keu : « Preamble »

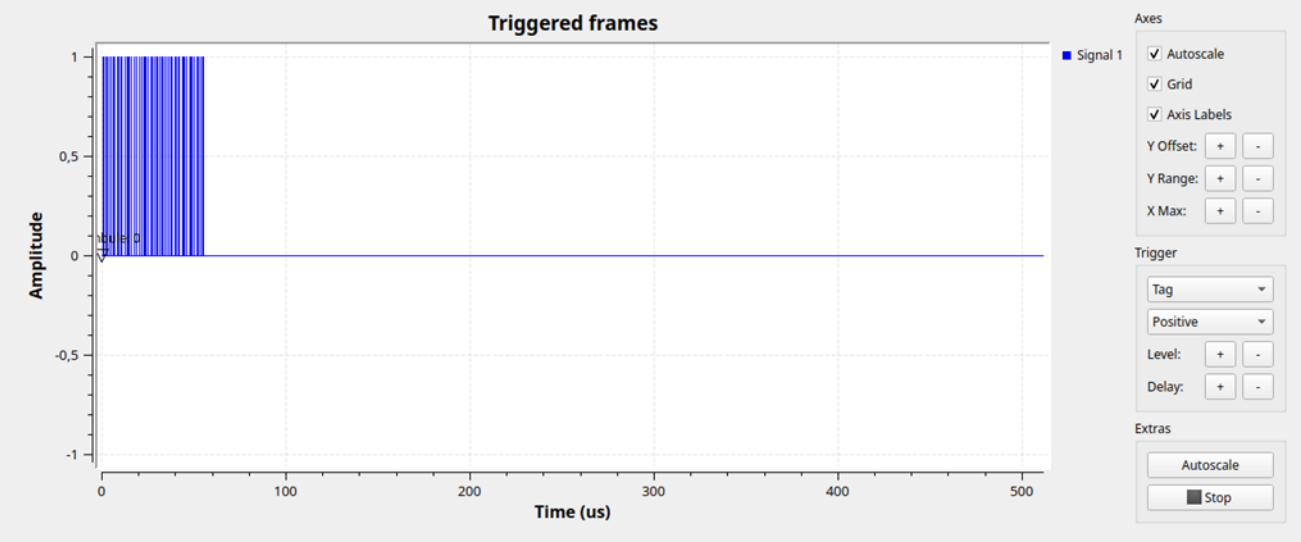


Figure 9 : Affichage des trames ADS B sous GNU Radio

6.3 - Réglage du gain et du seuil

Le réglage du gain est une étape empirique importante pour les étudiants : **trop peu de gain** → signal trop faible, peu de trames détectées ; **trop de gain** → saturation de l'ADC, distorsion, fausses détections. La procédure recommandée est :

- Démarrer GNU Radio avec le gain RF à 0.
- Observer le niveau de bruit de fond sur le QT GUI Time Sink.
- Augmenter progressivement le gain RF par pas de 10 dB jusqu'à observer des impulsions clairement au-dessus du bruit.
- Ajuster le seuil du bloc Threshold à environ 2–3 fois le niveau de bruit moyen.
- Vérifier que les préambules sont bien détectés dans l'affichage.

6.4 - Du signal reçu au décodage de la trame en Python

Les fichiers binaires enregistrés par GNU Radio sont ensuite traités par un programme Python dans le bloc trame_csv_sink (grisé dans le flowchart ci-dessus). Ce code spécifique peut être fourni aux étudiants ou faire l'objet d'un mini-projet.

Ce code spécialement développé :

- Surveille les tags `preambule` ;
- capture les échantillons suivant chaque tag,
- décode la modulation PPM du Mode S (1 bit = 1 µs ; impulsion dans la première demi-microseconde = 1, dans la deuxième = 0) ;
- lit les 112 bits de chaque trame enregistrée ;
- extrait les champs DF, CA, ICAO, TC selon la structure du tableau §2.2 ;
- vérifie le CRC-24 et ajoute crc_ko en cas d'erreur, crc_ok si la trame est valide et donc exploitable ;
- selon le Type Code et si le CRC est correct, converti la trame binaire en hexadécimal et affiche l'identification ICAO en hexadécimal ;
- stocke le résultat dans un fichier a format csv avec un horodatage.

```
horodatage;binaire;hexadecimal;df;icao;crc
20260705_114515;01010101010100101...10110010001001011;;10;;crc_ko
20260705_114515;10001101011100111...01010010111001011;8D7380C3EA25385D15
3F8C5D4B97;17;7380C3;crc_ok
20260705_114515;01011101011100101...01011010010011 10;;11;;crc_ko
20260705_114516;10101001010100100...10001110110010100 1;;21;;crc_ko
20260705_114516;10101011110110100...01100010111001000 1;;21;;crc_ko
```

Figure 10 : Extrait du fichier csv de trames généré

7 - Mesure de la qualité de réception : le comptage d'aéronefs

7.1 - Principe de la métrique

L'idée centrale de ce TP est d'utiliser le nombre d'aéronefs ICAO distincts détectés sur une fenêtre de temps fixe comme indicateur de performance de la chaîne de réception. Cette métrique présente plusieurs avantages pédagogiques :

- Simple à mesurer : un programme Python comptant les adresses ICAO uniques suffit.
- Objective : elle intègre toute la chaîne (antenne, placement, gain, seuil, décodage).
- Reproductible : en testant les antennes dans les mêmes conditions de trafic (même créneau horaire, même durée), la comparaison est équitable.
- Motivante : les étudiants « voient » directement l'impact de leur réalisation d'antenne.

7.2 - Protocole de mesure

Il convient de fixer des conditions de mesure explicites pour tous les étudiants :

- Durée de mesure : 20 minutes minimum par antenne (pour obtenir une statistique représentative)
- Créneau horaire : 9h–11h ou 14h–16h (trafic commercial dense)
- Position du récepteur : identique pour toutes les antennes testées
- Gain RF GNU Radio : fixe (même valeur pour toutes les antennes)
- Seuil Threshold : fixe

7.3 - Evaluation

Une note peut être élaborée à partir des résultats des mesures selon les critères établis et annoncés par l'enseignant. L'évaluation sommative permettra de prendre en compte l'ensemble du travail réalisé par l'étudiant et son efficacité concrète.

7.4 - Extension : cartographie des avions détectés

La visualisation sur carte constitue une extension naturelle pour les étudiants avancés. La bibliothèque Python folium permet de générer une carte HTML interactive :

La comparaison de la carte obtenue avec celle de Flightradar24 au même instant valide le bon fonctionnement du décodage. Les écarts (avions présents sur Flightradar24 mais non détectés) permettent d'identifier les zones d'ombre liées aux obstacles locaux.

Il peut être aussi opportun d'intégrer une partie base de données pour exploiter le code OACI aéronef sur 24 bits et rechercher le numéro de vol, ses aéroports de départs et destination, …

8 - Aspects cybersécurité

8.1 - Vulnérabilités inhérentes au protocole

L'ADS-B a été conçu pour la disponibilité et l'interopérabilité, sans considération initiale pour la sécurité. Cette réalité, discutée ouvertement dans la communauté de sécurité depuis les conférences DEFCON et Black Hat, présente plusieurs vulnérabilités :

- Absence d'authentification : n'importe quel émetteur SDR peut envoyer des trames ADS-B valides avec une adresse ICAO et une position arbitraire. Il n'existe aucun mécanisme cryptographique pour vérifier que le message provient bien de l'aéronef qu'il prétend être.
- Absence de chiffrement : toutes les données sont en clair, lisibles par quiconque dispose d'un récepteur. Les informations diffusées (position, vitesse, indicatif) peuvent révéler des informations sensibles sur certains vols.

Ces deux faiblesses du protocole ouvrent de nombreuses possibilités d'attaque :

- *Ghost aircraft injection* (injection de faux avions) : créer des trafics fantômes qui apparaissent sur les écrans de contrôle. Présenté à DEFCON 20 par Andrei Costin et Augustin Francillon [7]
- *Aircraft spoofing* (usurpation d'identité) : émettre avec l'adresse ICAO d'un vrai avion pour induire en erreur les systèmes de surveillance. [8],[9]

- *Replay attack* : rejouer des trames enregistrées précédemment.
- *DoS par saturation* : inonder la fréquence 1090 MHz de trames, saturant les récepteurs.

Avertissement légal : toute émission sur la fréquence 1090 MHz sans autorisation est strictement interdite en France par article L. 33-3-1 du Code des Postes et Communications Électroniques. Ce TP utilise exclusivement le mode réception passive (les clefs RTL SDR ne font que de la réception).

8.2 - Contre-mesures et mécanismes de détection

Plusieurs approches sont à l'étude ou déjà déployées pour sécuriser l'ADS-B :

- MLAT (*Multilateration*) : en combinant les temps d'arrivée du même signal sur plusieurs récepteurs géographiquement distants, il est possible de recalculer indépendamment la position de l'émetteur. Si la position calculée par MLAT ne correspond pas à la position déclarée dans le message, le message est suspect. [10]
- Détection d'anomalies par apprentissage machine : des travaux récents appliquent des réseaux de neurones récurrents (LSTM) pour détecter les séquences de messages incohérentes physiquement (vitesse impossible, trajectoire erratique).
- ADS-B version 2 : la version 2 du standard introduit des indicateurs d'intégrité (NIC, NAC, SIL) plus robustes, mais ne résout pas fondamentalement l'absence d'authentification.

9 - En 2026 : entretien avec un contrôleur aérien

En situation opérationnelle normale, ADS-B améliore-t-il votre travail de contrôleur aérien au quotidien ?

Le territoire français métropolitain (et plus largement le territoire européen) est assez bien couvert par de nombreux radars secondaires. Il persiste quelques zones blanches, notamment en zone montagneuse, mais les données radars sont celles privilégiées pour le contrôle aérien français (notamment pour des raisons d'intégrité). Ainsi, je travaillais avant dans un centre en route et je ne crois pas que les données ADS-B y étaient exploitées. Je crois cependant que cette technologie est aujourd'hui plus largement exploitée en Outre-Mer, où les radars sont moins nombreux et où la maintenance est plus compliquée. Les données ADS B permettent de travailler plus efficacement là où le contrôle aux procédures était la norme, et où le contrôle radar était l'exception.

Je crois aussi que des projets d'installation de Multilatération utilisant l'ADS-B sont en cours sur des grandes approches telles que l'aéroport Marseille, où le radar sol primaire ne suffit souvent pas à donner une image précise de la situation au sol. Ce serait alors un grand plus pour les contrôleurs aériens qui y travaillent.

Comment évaluez-vous la précision et la fiabilité des données ADS-B (position, vitesse, altitude) par rapport aux radars secondaires classiques ? Quels sont les avantages et les limites que vous observez ?

Les données ADS-B sont généralement considérées comme plus précises que les données radars, car provenant directement des systèmes de navigation bord. Et ces données sont également mises à jour plus régulièrement. Cependant, sans infrastructure sol pour vérifier l'information, c'est par sa fiabilité que le système ADS-B est limité. On est en effet dépendant de la qualité des données fournies, qui peut varier suivant des dysfonctionnements systèmes, des interférences, ... C'est pour ça que le système ADS-B n'est à ma connaissance pas utilisé comme seule source d'information et qu'il est redondé par d'autres infrastructures sol.

Bref, système précis, efficace et beaucoup plus économique à entretenir, mais qui n'est pas totalement fiable s'il n'est pas redondé par d'autres moyens de détection

Les pilotes remontent des problématiques de guerre électronique affectant la réception des signaux GNSS, avez-vous connaissance d'incidents similaires avec ADS-B ?

Il me semble qu'un vol en provenance de Beyrouth et à destination de Marseille avait été victime de leurrage et de brouillage GNSS à son départ (zone de conflits). Cela avait entraîné des perturbations sur la suite de son vol, notamment des alarmes injustifiées à bord. C'est probablement le cas de nombreux autres vols, et cela illustre le risque quant à la fiabilité de cet outil si utilisé seul.

Références :

[1]: Utilisation des "Tags" pour synchroniser un flux de données ; Application au décodage des signaux ADS-B. Thomas Lavarenne, https://gnuradio-fr-18.sciencesconf.org/214676.html

[2]: Document video : " PROGRESS IN AIR TRAFFIC CONTROL " 1973 U.S. AIR FORCE ATC EDUCATIONAL FILM XD94765, https://youtu.be/OTmwe3Vf_Y0?si=PlFZvfAoDS6PECOe

[3]: OpenSky https://opensky-network.org/data/aircraft

[4]: ICAO Technical Provisions for Mode S Services and Extended Squitter, https://www.icao.int/MID/Documents/2019/MICA/MICA-MID%20-%20WP%2002%20-%20Mode%20S%20Surveillance%20Principle.pdf

[5]: The 1090MHz Riddle. An open-access book about decoding Mode-S and ADS-B data by Junzi Sun, http://mode-s.org/decode/

[6]: Formal analysis of the compact position reporting algorithm, Aaron Dutle, Laura Titolo, Mariano M. Moscato, César A. Muñoz, Gregory Anderson, Francois Bobot, https://cea.hal.science/cea-04491533v1

[7]: Andrei Costin and Aurélien Francillon. "Ghost is in the Air(traffic): On insecurity of ADS-B protocol and practical attacks on ADS-B devices". In: Black Hat USA. July 2012, pp. 1–12.

[8]: ADS-B spoofing attack detection method based on LSTM Jing Wang, Yunkai Zou, Jianli Ding

[9]: Detecting ADS-B spoofing attacks – using collected and simulated data Joakim Thorn, Alex Wahlgren

[10]: Non-Radar Surveillance ADS-B/MLAT/WAM Products -THALES, https://www.icao.int/SAM/Documents/2017-ADSB/14%20THALES_%20NRS_Products_SEP_2017.pdf

Ressource publiée sur Culture Sciences de l'Ingénieur : https://sti.eduscol.education.fr/si-ens-paris-saclay