

Université Sorbonne Paris Nord
Département réseaux et Télécommunications
Parcours: Cybersécurité

Compte rendu du TP GNU Radio :
Mise en œuvre d'un système de transmission radio logicielle

Travail préparé par: Ons AMMAR
Formation initiale : BUT 3 Cybersécurité

Superviseur : Mr Quentin GAIMARD

Année universitaire : 2025/2026

Université Sorbonne Paris Nord

Département réseaux et Télécommunications

Parcours: Cybersécurité

Contexte général

Ce TP fait partie d'un module de transmission radio logicielle (Software Defined Radio, SDR)
L'idée principale est de comprendre et manipuler les signaux radio à l'aide d'outils logiciels, plutôt que de manipuler uniquement des circuits matériels classiques

Le TP utilise :

- **GNU Radio** : qui est un environnement logiciel libre pour concevoir, simuler et tester des systèmes radio. Il permet de construire des chaînes de traitement de signal par blocs (modulation, filtrage, démodulation, etc.)
- **ADALM-PLUTO** : qui est un émetteur-récepteur SDR matériel qui permet de transmettre et recevoir de vrais signaux radio dans certaines bandes de fréquence

Université Sorbonne Paris Nord

Département réseaux et Télécommunications

Parcours: Cybersécurité

Table des matières:

TP 1 : Prise en main de GNU Radio

Contexte spécifique.....	page 4
Conclusion.....	page 15

2ème TP : Transmission numérique

Contexte spécifique.....	page 16
Montage de système AM fonctionnel	
ARCHITECTURE 1 : ÉMETTEUR AM.....	page 18
ARCHITECTURE 2 : RÉCEPTEUR AM.....	page 20
Conclusion.....	page 22

3ème TP (autonomie) : Format de modulation avancée

Contexte spécifique.....	page 23
Montage de système QAM fonctionnel	
ARCHITECTURE 1 : ÉMETTEUR QAM(TX).....	page 25
ARCHITECTURE 2 : RÉCEPTEUR QAM(RX).....	page 27
Conclusion.....	page 29

Université Sorbonne Paris Nord

Département réseaux et Télécommunications

Parcours: Cybersécurité

TP 1 : Prise en main de GNU Radio

Contexte spécifique :

- L'objectif est de **découvrir GNU Radio**, comprendre son interface et ses blocs de traitement du signal
- On apprend à générer des signaux, les visualiser (oscilloscope, spectre), et à appliquer des opérations de base comme le filtrage et l'amplification
- La partie matérielle avec l'**ADALM-PLUTO** permet de connecter le logiciel à un émetteur-récepteur réel, mais l'accent est mis d'abord sur la **simulation logicielle**

GNU Radio est une suite logicielle dédiée à l'implémentation de radios logicielles et de systèmes de traitement du signal

Installation du logiciel

```
Réception de :5 http://security.debian.org/debian-security bullseye-security/main amd64 Packages [422 kB]
Réception de :6 http://security.debian.org/debian-security bullseye-security/main Translation-en [281 kB]
Réception de :7 http://deb.debian.org/debian bullseye-updates/main Translation-en.diff/Index [12,8 kB]
Réception de :8 http://deb.debian.org/debian bullseye-updates/main Translation-en T-2025-07-21-2004.39-F-2025-07-21-2004.39.pdiff [47 B]
Réception de :8 http://deb.debian.org/debian bullseye-updates/main Translation-en T-2025-07-21-2004.39-F-2025-07-21-2004.39.pdiff [47 B]
1 134 ko réceptionnés en 2s (560 ko/s)
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
```

Installation gnradio avec apt

```
etudiant@Marionnet:~$ sudo apt install gnuradio gr-iio libiio-utils iio-sensor-proxy
```

```
etudiant@Marionnet:~$ sudo apt install audacity
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Les paquets supplémentaires suivants seront installés :
  audacity-data libavcodec58 libavformat58 libavresample4 libavutil56
  libchromaprint1 libflac++6v5 libgail-common libgail18 libgme0 libgtk2.0-0
  libgtk2.0-0-bin libgtk2.0-common libid3tag0 liblilv-0-0 libmad0 libopenmp0
  libportsmf0 libqt5x11extras5 librabbitmq4 libserd-0-0 libsound-0-0
  libsoundtouch1 libratom-0-0 libsrtp1.4-gnutls libssh-gcrypt-4 libtsuul-0-0
  libswresample3 libvamp-hostsdk3v5 libwxbase3.0-0v5 libwxgtk3.0-gtk3-0v5
Paquets suggérés :
  ladspa-plugin serdi sordi
Les NOUVEAUX paquets suivants seront installés :
  audacity audacity-data libavformat58 libchromaprint1 libflac++6v5
  libgail-common libgail18 libgme0 libgtk2.0-0 libgtk2.0-0-bin libgtk2.0-common
  libid3tag0 liblilv-0-0 libmad0 libopenmp0 libportsmf0 libqt5x11extras5
  librabbitmq4 libserd-0-0 libsound-0-0 libsoundtouch1 libratom-0-0
  libsrtp1.4-gnutls libssh-gcrypt-4 libtsuul-0-0 libvamp-hostsdk3v5
  libwxbase3.0-0v5 libwxgtk3.0-gtk3-0v5
```

Vérification de la version

```
etudiant@Marionnet:~$ gnuradio-config-info --version
3.8.2.0
etudiant@Marionnet:~$
```

Université Sorbonne Paris Nord

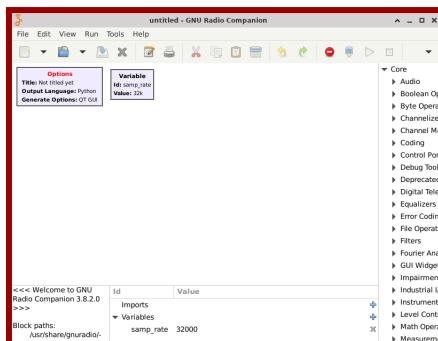
Département réseaux et Télécommunications

Parcours: Cybersécurité

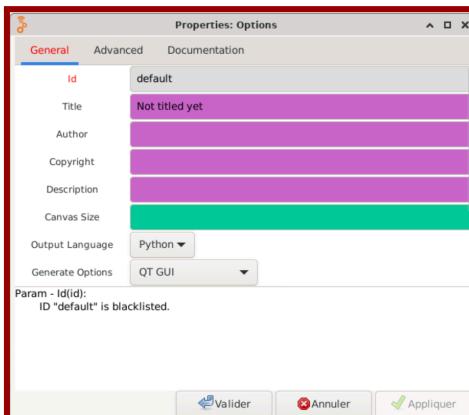
Commande pour lancer GNU

```
etudiant@Marionnet:~$ gnuradio-companion
<<< Welcome to GNU Radio Companion 3.8.2.0 >>>
Block paths:
    /usr/share/gnuradio/grc(blocks
    □
```

Au lancement du logiciel, on obtient cette interface graphique :

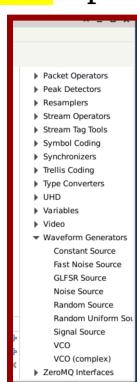


Visualisation des propriétés au niveau du bloc options



Génération d'un signal S de fréquence f

Au niveau des blocs, on choisit waveform generators → puis Signal source

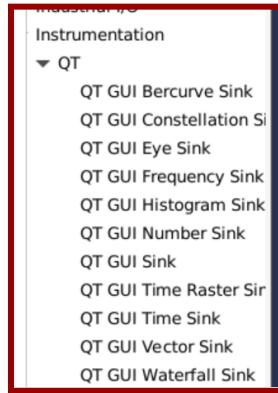


On choisit ensuite QT GUI Sink (c'est l'outil qui nous permettra de visualiser le signal)

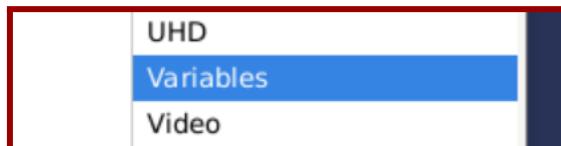
Université Sorbonne Paris Nord

Département réseaux et Télécommunications

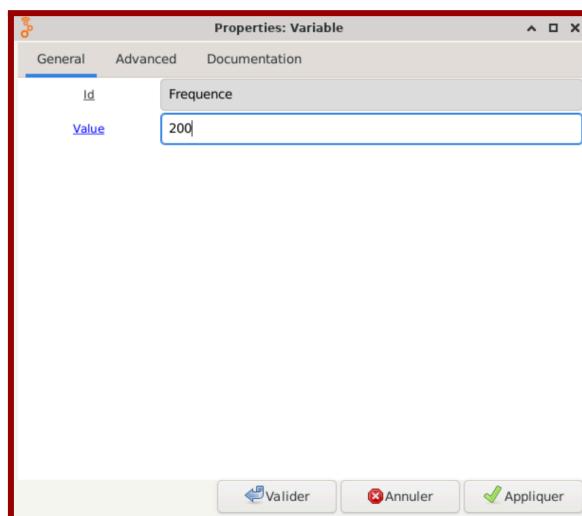
Parcours: Cybersécurité



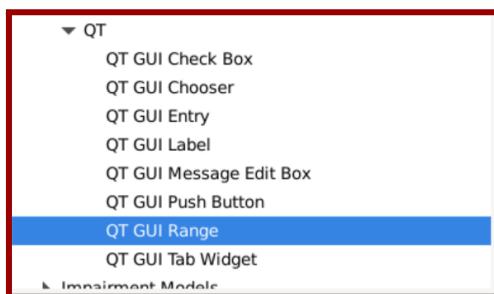
Au niveau des **Variables**, on peut définir n'importe quelle variable de cette manière



exp: J'ai défini une variable (**id=Frequence , Value=200**)



J'ai travaillé avec le module **QT GUI Range**

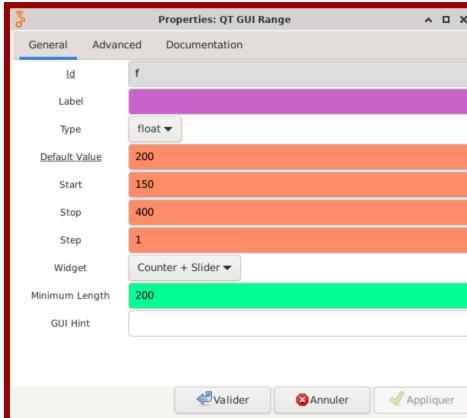


Université Sorbonne Paris Nord

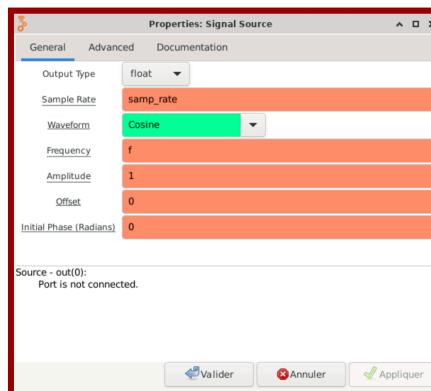
Département réseaux et Télécommunications

Parcours: Cybersécurité

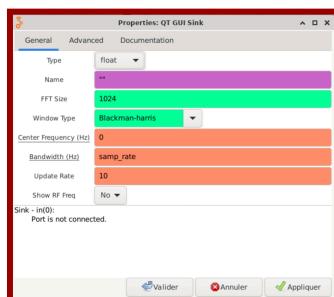
Définition de la variable f



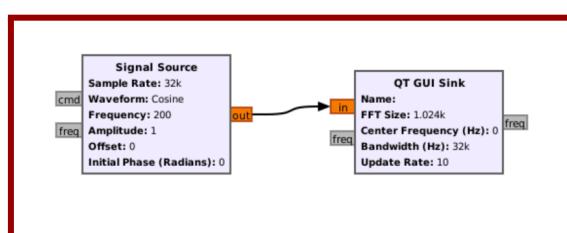
Modification des paramètres au niveau du signal source



Modification au niveau du visualiseur QT GNU Sink (entrée flottant)



Relier les deux blocs

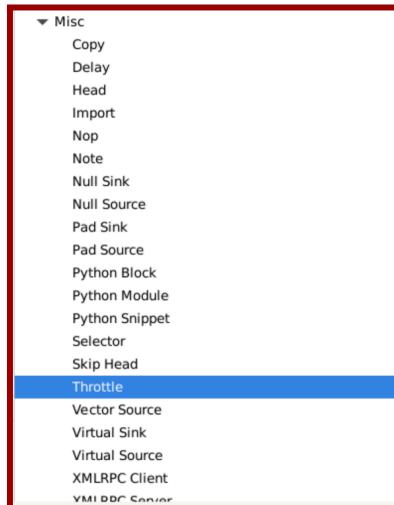


Université Sorbonne Paris Nord

Département réseaux et Télécommunications

Parcours: Cybersécurité

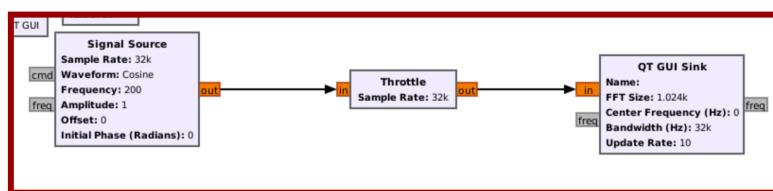
Insertion du bloc **Throttle** (ce bloc permettra de limiter efficacement le débit d'un graphe de flux en limitant le nombre d'échantillons copiés de ses propres tampons d'entrée vers ses tampons de sortie)



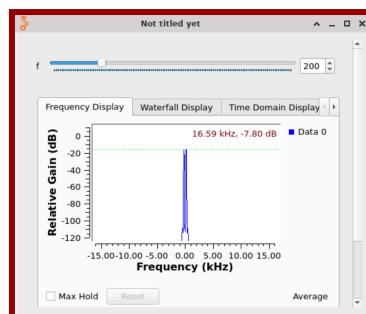
Changer le type en float



Au final on obtient cette architecture



Après exécution, on peut visualiser notre signal



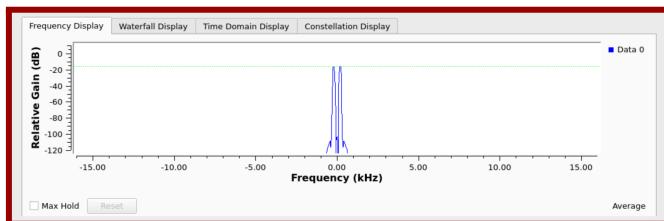
Université Sorbonne Paris Nord

Département réseaux et Télécommunications

Parcours: Cybersécurité

Le signal dans le **domaine temporel** (évolution en fonction du temps), amplitude en ordonnée, temps en abscisse

- On arrive à observer la **forme du signal** : sinusoïdal, carré, triangulaire...
- On peut vérifier la **fréquence** (période du signal)
- On arrive à voir les effets d'un filtre ou d'une addition de signaux



(La densité spectrale de puissance du signal (DSA))

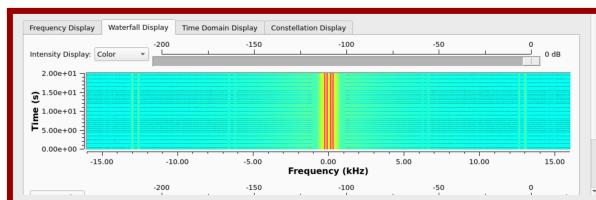
Au niveau du signal dans le domaine fréquentiel on a:

- Sur l'axe horizontal : fréquence
- Sur l'axe vertical : niveau en dB

→ On peut observer quelles **fréquences** sont présentes

On peut voir :

- la fondamentale d'une sinusoïde,
- les harmoniques d'un signal carré,
- l'effet des filtres,
- les pics de fréquence lors de la modulation.



Une représentation temps-fréquence avec des couleurs

Chaque ligne = une **FFT** dans le temps, couleurs = puissance

→ Ici on a réussi à repérer des signaux qui évoluent dans le temps :

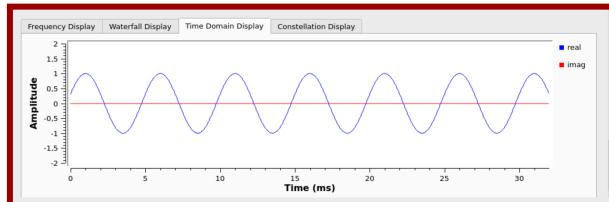
- signal intermittent,
- glissement de fréquence,
- émissions courtes ou brèves

→ On a donc réussi à voir **un signal réel venant d'une radio**

Université Sorbonne Paris Nord

Département réseaux et Télécommunications

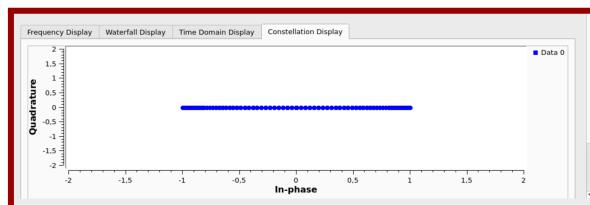
Parcours: Cybersécurité



Le signal affiché dans le plan IQ (réel en x, imaginaire en y)

Habituellement utilisé pour la modulation numérique (QPSK, QAM, etc)

- On voit la forme des symboles numériques
- On peut évaluer le bruit, la dispersion, la qualité de transmission
- Pour un signal réel (float), cet onglet ne montre rien



On va faire varier le type de données en complexe

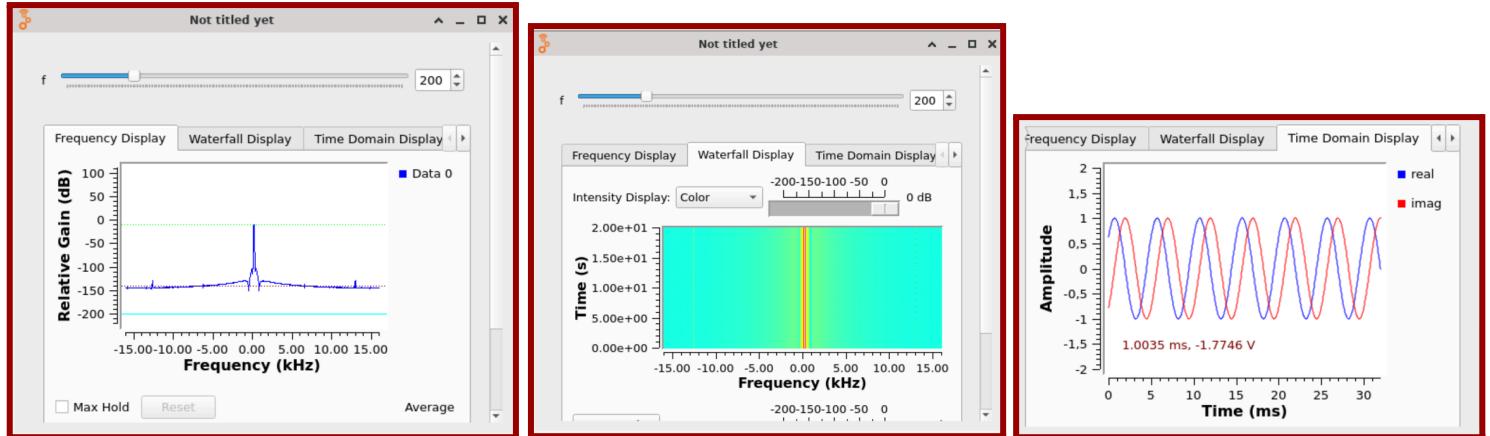
QT GUI Sink Properties	Throttle Properties	Signal Source Properties
Type: complex	Type: complex	Type: complex
Name: **	Sample Rate: samp_rate	Sample Rate: samp_rate
FFT Size: 1024	Vec Length: 1	Waveform: Cosine
Window Type: Blackman-harris	Ignore rx_rate tag: True	Frequency: f
Center Frequency (Hz): 0		Amplitude: 1
Bandwidth (Hz): samp_rate		Offset: 0
Update Rate: 10		Initial Phase (Radians): 0
Show RF Freq: No		
Plot Frequency: On		
Plot Waterfall: On		
Plot Time: On		
Plot Const: On		

Université Sorbonne Paris Nord

Département réseaux et Télécommunications

Parcours: Cybersécurité

Visualisation avec type complexe



Conséquences :

Time Sink nous affiche deux courbes : I et Q

Frequency Sink nous donne une FFT plus précise, car phase et amplitude sont connues

Alors que waterfall nous donne une représentation correcte

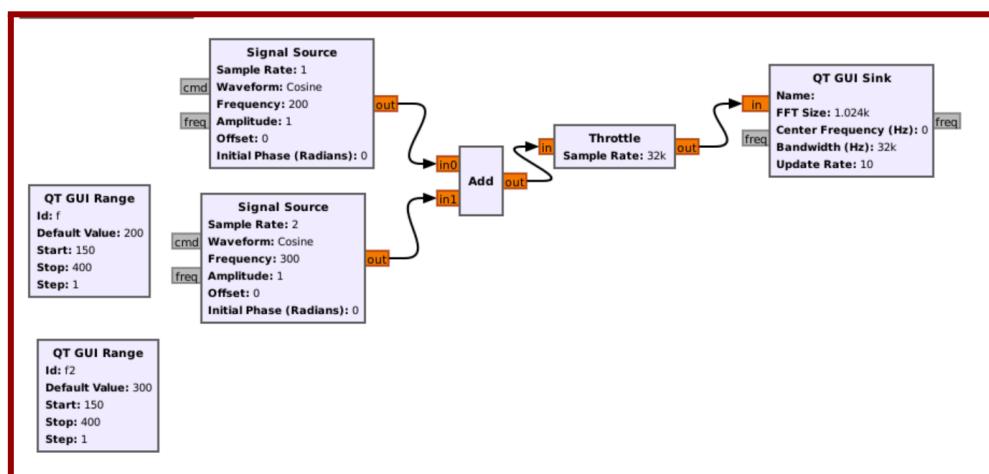
Constellation Sink fonctionne

- pour un sinus complexe pur , nous donne alors un cercle
- pour un vrai signal modulé, et nous donne des symboles dispersés

Traitement de signal

J'ai ajouté un deuxième signal S2

J'ai ajouté une deuxième fréquence f2, avec une valeur maximale 400, valeur par défaut 300, minimale 150

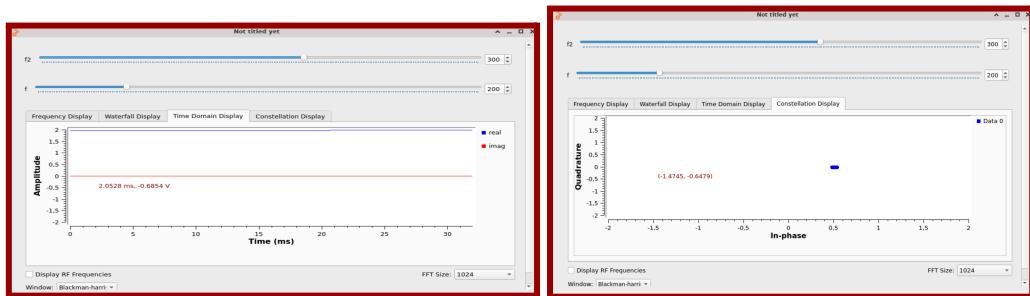
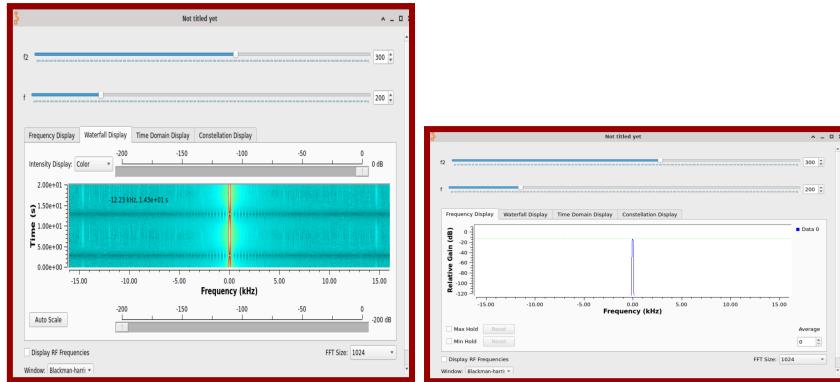


Université Sorbonne Paris Nord

Département réseaux et Télécommunications

Parcours: Cybersécurité

En utilisant le module mathématique ADD, on peut voir l'addition des deux signaux

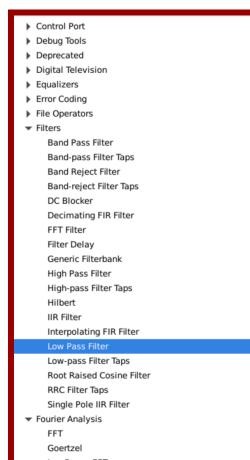


Ajout d'un filtre passe-bas:

Pourquoi ?

→ En réalité, ce filtre passe-bas nous permettra :

- d'éliminer les harmoniques indésirables
- Prévenir les interférences
- Protéger les équipements en aval
- D'améliorer la qualité du signal pour les mesures

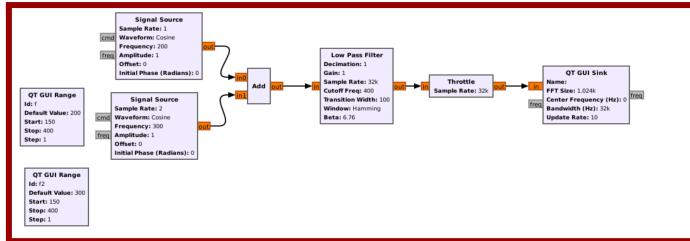


Université Sorbonne Paris Nord

Département réseaux et Télécommunications

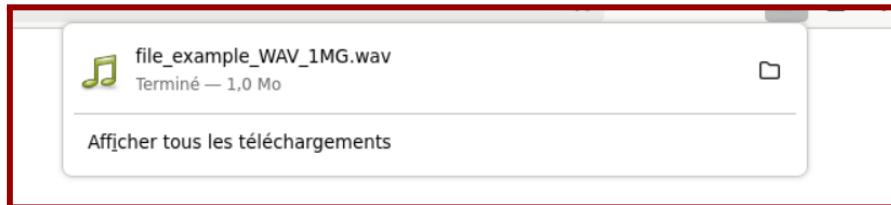
Parcours: Cybersécurité

Fréquence d'échantillonnage $f_c=400\text{hz}$

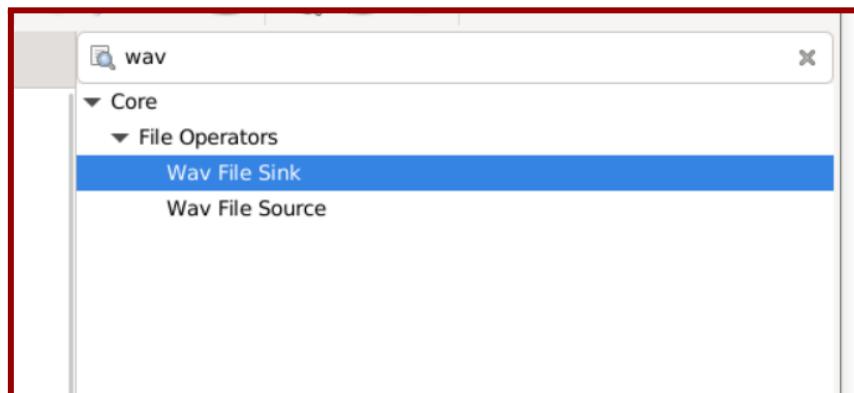


Signal réel:

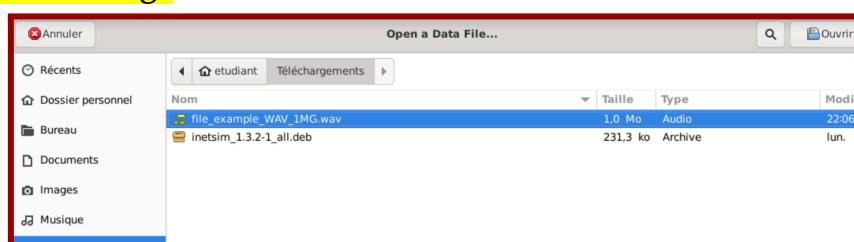
On commence par le téléchargement du fichier .wav



Puis, au niveau de wav , on choisit **wav file source** ou on va importer le fichier .wav



On choisit le fichier .wav téléchargé

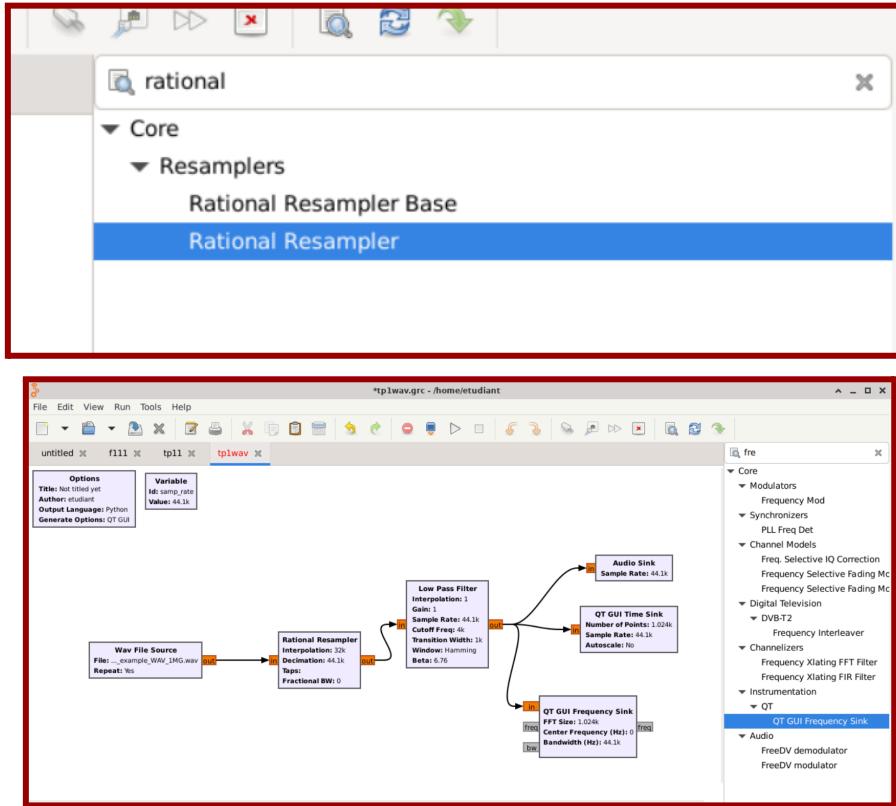


Puis on choisit le Rational Resampler(pour changer le taux d'échantillonnage)

Université Sorbonne Paris Nord

Département réseaux et Télécommunications

Parcours: Cybersécurité



Attribution d'une adresse IP (à l'interface connectée au Pluto)

On vérifie tout d'abord les périphériques USB, puis on vérifie quelle interface est connectée à notre Pluto

```

etudiant@Marionnet:~$ lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 003: ID 0456:b673 Analog Devices, Inc. PlutoSDR (ADALM-PLUTO)
Bus 001 Device 002: ID 80ee:0021 VirtualBox USB Tablet
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
etudiant@Marionnet:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:a9:83:9c brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp0s3
        valid_lft 86382sec preferred_lft 86382sec
        inet6 fe80::a9:83ff:fe00:27ff:fe/64 scope link
            valid_lft forever preferred_lft forever
3: enx00e022824f86: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN group default qlen 1000
    link/ether 00:e0:22:82:4f:86 brd ff:ff:ff:ff:ff:ff
    inet 192.168.2.10/24 brd 192.168.2.255 scope global dynamic enx00e022824f86
*
```

On installe le package “libiio-utils et iio-sensor-proxy”

```

etudiant@Marionnet:~$ sudo apt install libiio-utils iio-sensor-proxy
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
iio-sensor-proxy est déjà la version la plus récente (3.0-2).
libiio-utils est déjà la version la plus récente (0.21-2+b1).
0 mis à jour, 0 nouvellement installés, 0 à enlever et 73 non mis à jour.
etudiant@Marionnet:~$ 
```

Université Sorbonne Paris Nord

Département réseaux et Télécommunications

Parcours: Cybersécurité

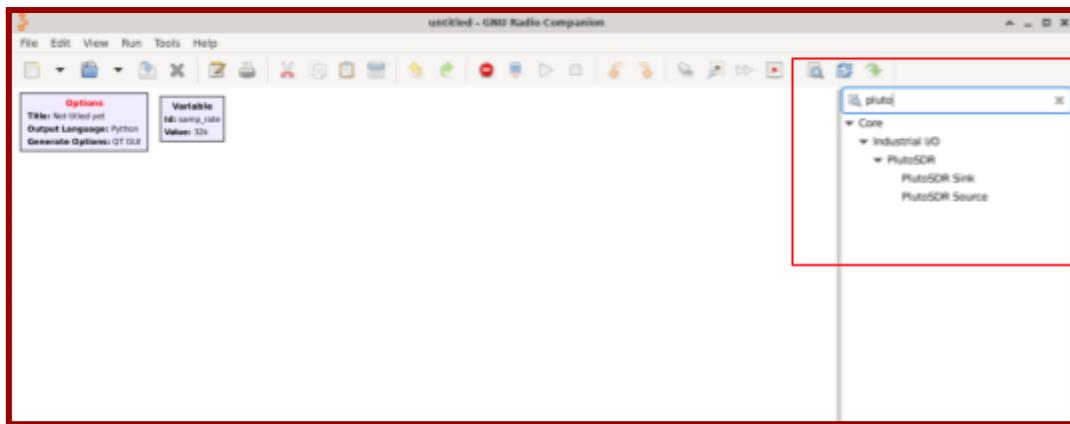
```
étudiant@Marionnet:~$ iio_info -n 192.168.2.1
Library version: 0.21 (git tag: v0.21)
Compiled with backends: local xml ip usb serial
IIO context created with network backend.
Backend version: 0.21 (git tag: v0.21 )
Backend description string: 192.168.2.1 Linux (none) 4.19.0-119999-g6edc6cd #319
SMP PREEMPT Mon Jul 6 15:45:01 CEST 2020 armv7l
IIO context has 9 attributes:
    hw_model: Analog Devices PlutoSDR Rev.C (Z7010-AD9363A)
    hw_model variant: 0
    hw_serial: 1044732a98110015f0ff31006821e58383
    fw_version: v0.32
    ad9361-phy,xo_correction: 40000137
    ad9361-phy,model: ad9363a
    local_kernel: 4.19.0-119999-g6edc6cd
    uri: ip:192.168.2.1
    ip,ip-addr: 192.168.2.1
IIO context has 4 devices:
    iio:device0: ad9361-phy
        9 channels found:
```

iio_info : est une commande qui nous permet d'obtenir des informations sur les périphériques IIO connectés

-n adresse : option qui permet de se connecter directement à un périphérique réseau en précisant son adresse IP

Pour notre cas : **iio_info -n 192.168.2.1** va lister toutes les caractéristiques de ce périphérique IIO accessible via l'IP 192.168.2.1.

Installation réussie



Conclusion :

Le TP1 permet de maîtriser les fondamentaux de GNU Radio et de se familiariser avec le traitement de signaux avant d'aborder la transmission réelle. Les étudiants apprennent à concevoir des chaînes de traitement simples et à observer les effets de filtres et transformations sur un signal. Cette étape est essentielle pour pouvoir manipuler des systèmes radio plus complexes par la suite.

Université Sorbonne Paris Nord

Département réseaux et Télécommunications

Parcours: Cybersécurité

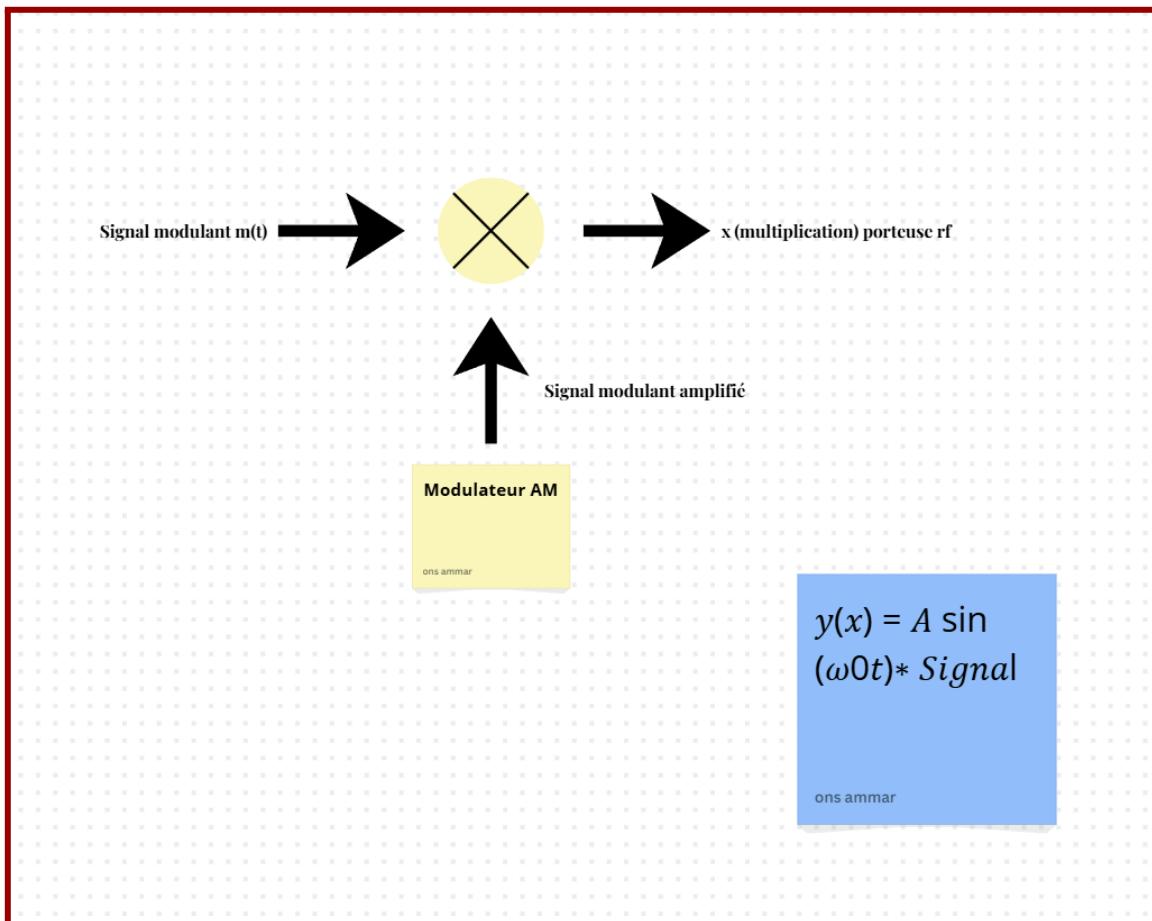
2ème TP : Transmission numérique

Remarque préalable: Dans ce TP on ne peut pas faire de l'AM analogique réelle, car le Pluto génère uniquement des signaux complexes I/Q, pas une amplitude analogique autour d'une porteuse RF réelle

Contexte spécifique :

- Objectif : mettre en œuvre une chaîne complète de transmission et réception de signaux numériques.
- On expérimente :
 - a. La modulation d'amplitude et la démodulation
 - b. La transmission via l'**ADALM-PLUTO**, en utilisant un signal réel
 - c. La réception et l'analyse du signal reçu pour observer l'effet du canal et du matériel
- Ce TP combine simulation et expérimentation réelle, permettant de comprendre comment un signal numérique est affecté lors de sa transmission.

Schéma: Fonctionnement d'un système de transmission AM



Université Sorbonne Paris Nord

Département réseaux et Télécommunications

Parcours: Cybersécurité

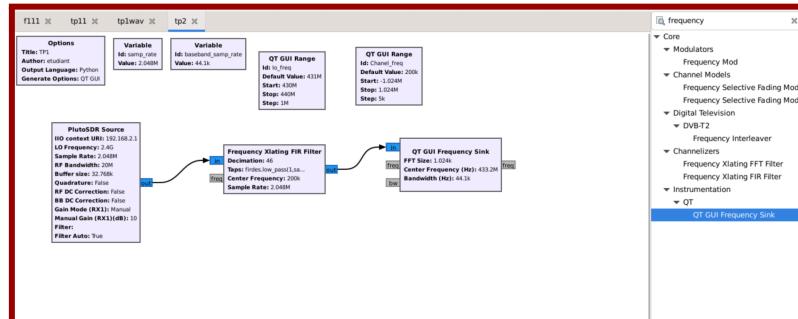
Différence entre AM et ASK

AM (analogique)	ASK (numérique)
L'amplitude varie continuellement selon un signal (audio pour notre cas)	L'amplitude prend deux valeurs : 0 ou A
Signal analogique	Bits (0/1)
Utilisée en radiodiffusion	Utilisée en télécom numérique

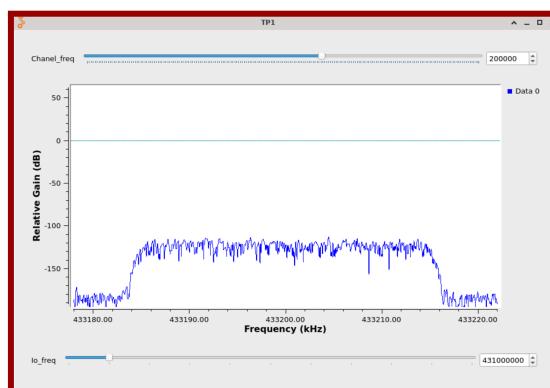
Ping vers l'interface pluto

```
etudiant@Marionnet:~$ ping 192.168.2.1
PING 192.168.2.1 (192.168.2.1) 56(84) bytes of data.
64 bytes from 192.168.2.1: icmp_seq=1 ttl=64 time=1.07 ms
64 bytes from 192.168.2.1: icmp_seq=2 ttl=64 time=0.759 ms
^C
--- 192.168.2.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.759/0.916/1.073/0.157 ms
etudiant@Marionnet:~$
```

Architecture étudiée:



Signal visualisé:



Université Sorbonne Paris Nord

Département réseaux et Télécommunications

Parcours: Cybersécurité

Montage de système AM fonctionnel

ARCHITECTURE 1 : ÉMETTEUR AM

Ici on génère un signal audio, on le module en amplitude (AM), puis on l'envoie sur une fréquence RF via le PlutoSDR

File Source (ou Audio Source)

→ sert à lire un fichier audio ou une trame sonore

Multiply Const

→ son rôle c'est d'ajuster l'amplitude pour éviter la saturation

Signal Source (porteuse)

→ il génère une onde sinusoïdale haute fréquence (RF)

Multiply

→ utilisé pour multiplier (porteuse × signal audio) = modulation AM

Low Pass Filter

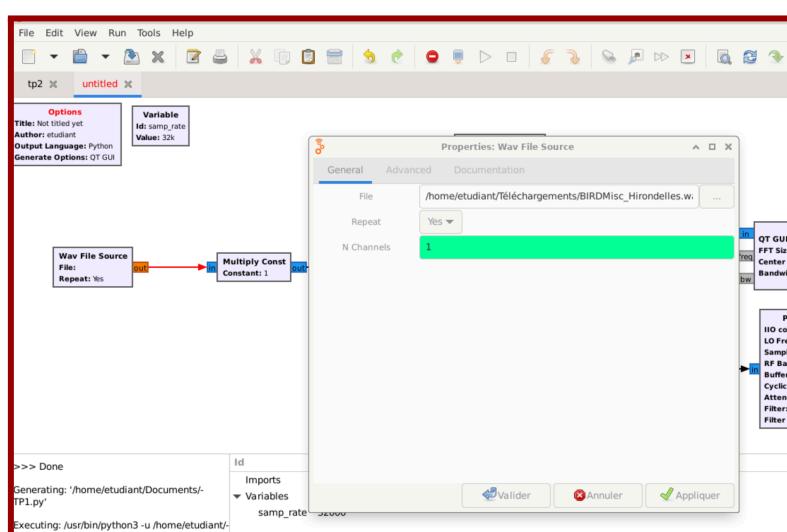
→ sert au nettoyage du signal et évite le repliement spectral.

Rational Resampler

→ adapte la fréquence d'échantillonnage pour le PlutoSDR.

PlutoSDR Sink

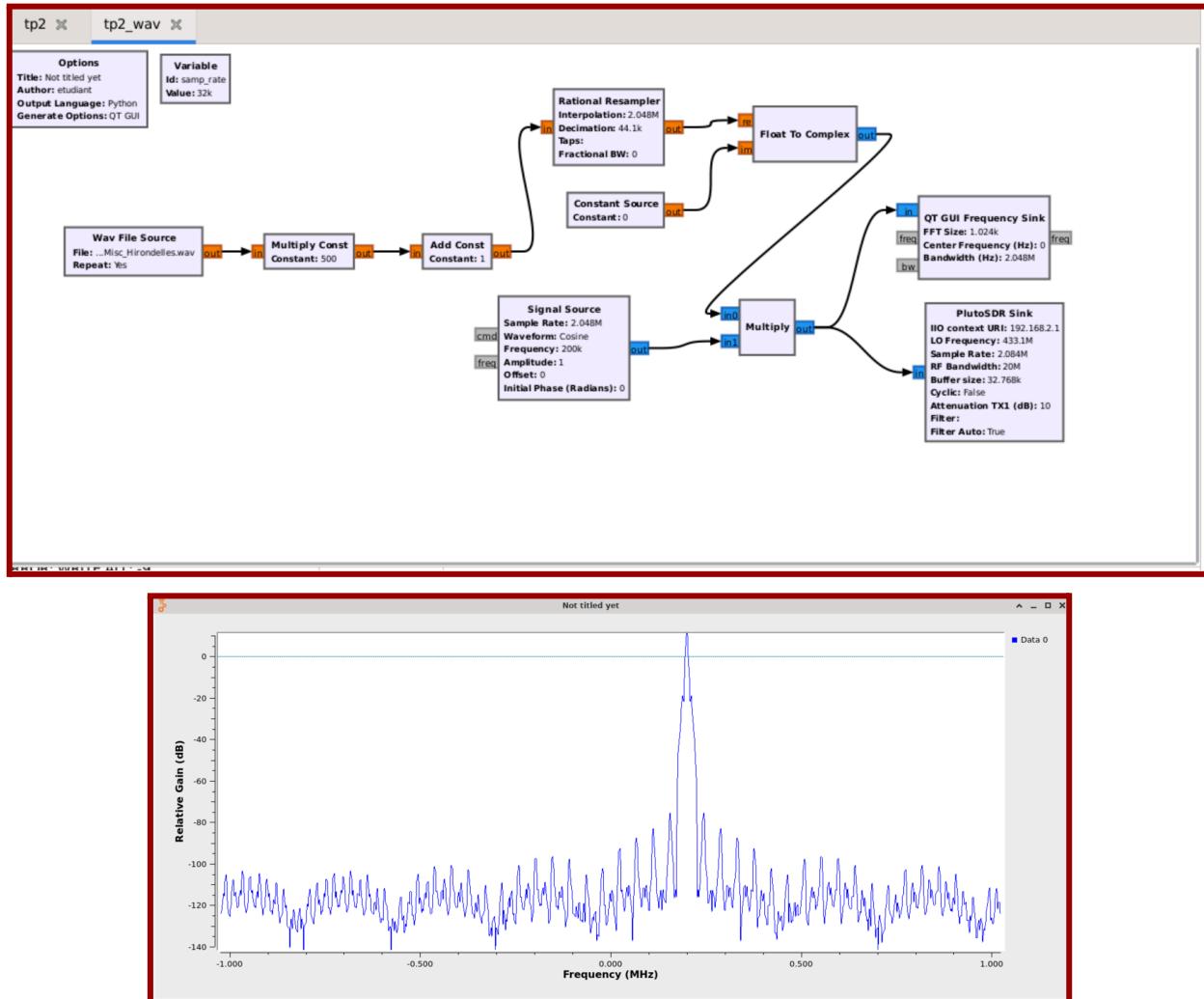
→ va émettre le signal AM sur une fréquence donnée (ex : 100 MHz).



Université Sorbonne Paris Nord

Département réseaux et Télécommunications

Parcours: Cybersécurité



- Elle prend un **signal audio** (bande base)
 - Elle crée une **porteuse RF**
 - Elle applique la modulation AM = variation d'amplitude de la porteuse
 - Elle envoie le signal modulé dans l'air via le PlutoSDR
- C'est une chaîne de transmission analogique.

Pourquoi décimation + interpolation donnent Fs_Out voulu ?

→ Tout simplement parce que le Rational Resampler applique :

$$Fs_{Out} = Fs_{In} \times (\text{Interpolation} / \text{Décimation})$$

Rôle de la multiplication par la sinusoïde complexe

Elle effectue la modulation en fréquence (montée en RF) :

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

$$\text{Signal AM} = s(t) \times \exp(j2\pi fct)$$

→ De ce fait, notre signal est transposé autour de la porteuse RF

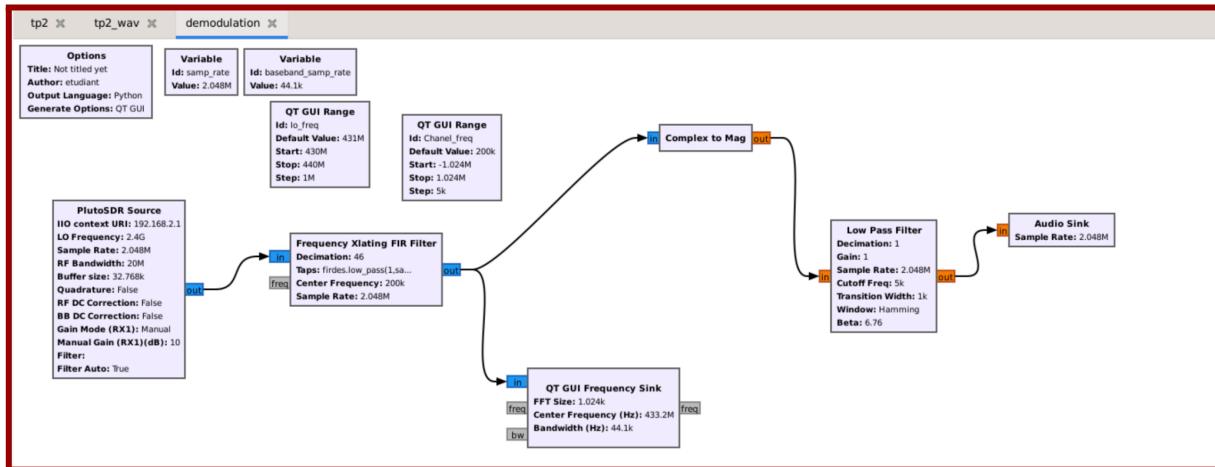
Université Sorbonne Paris Nord

Département réseaux et Télécommunications

Parcours: Cybersécurité

ARCHITECTURE 2 : RÉCEPTEUR AM

Ici on capture un signal AM depuis un autre poste de TP, l'isoler, le démoduler et écouter la voix/le son reçu



PlutoSDR Source

→ Sert à capter le signal RF AM autour d'une fréquence choisie

Frequency Xlating FIR Filter

→ nous permet une sélection du canal, et un décalage de fréquence vers le 0 Hz

Rational Resampler

→ permettra une adaptation du débit pour la carte son (48 kHz)

Complex to Mag

→ servira pour la démodulation AM (module du signal complexe)

Low Pass Filter

→ C'est tout simplement un filtre audio, il supprimera le bruit haute fréquence

Multiply Const

→ Servira pour le réglage du volume audio

Audio Sink

→ lecture du son dé-modulé sur les haut-parleurs

Ce que fait réellement cette architecture AM:

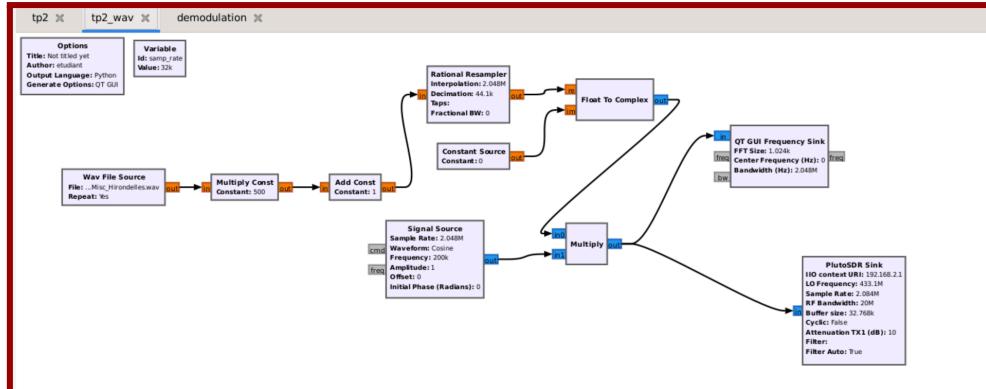
- elle reçoit un **signal RF** via le PlutoSDR
 - puis isole la **bande AM utile** du canal
 - elle démodule l'AM en prenant le **module du signal complexe**
 - appliquera un filtre pour récupérer un **son clair**
 - jouera le signal audio dans les haut-parleurs
- On a réussi à avoir une chaîne de réception analogique

Université Sorbonne Paris Nord

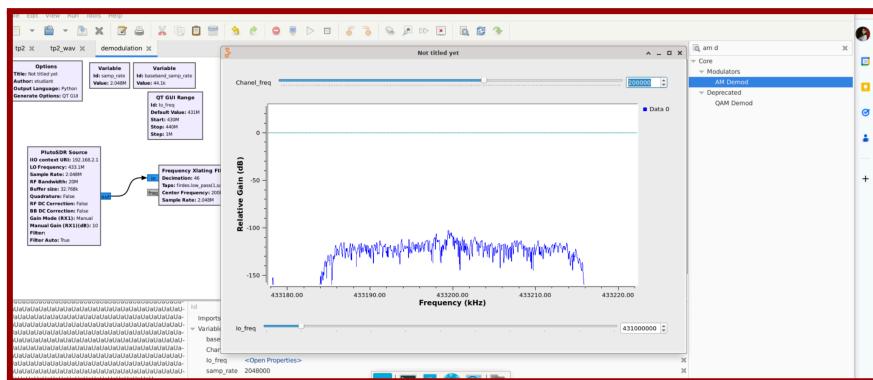
Département réseaux et Télécommunications

Parcours: Cybersécurité

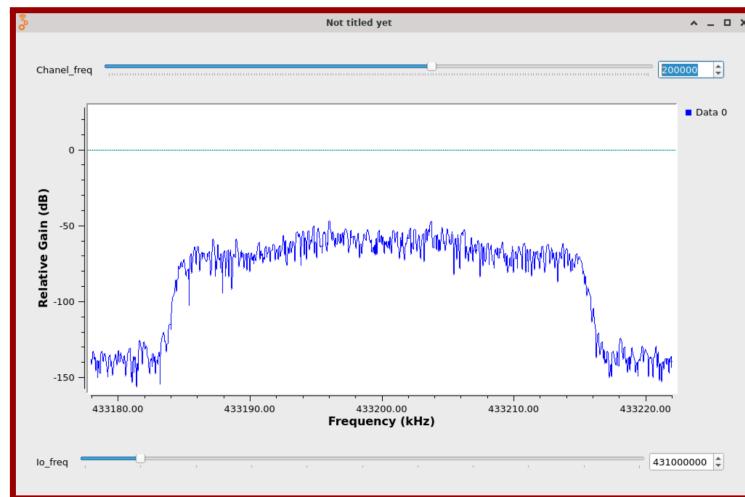
Architecture émission (au niveau du deuxième post de travail)



Adalm-pluto éloignés :



Adalm-pluto rapprochés :

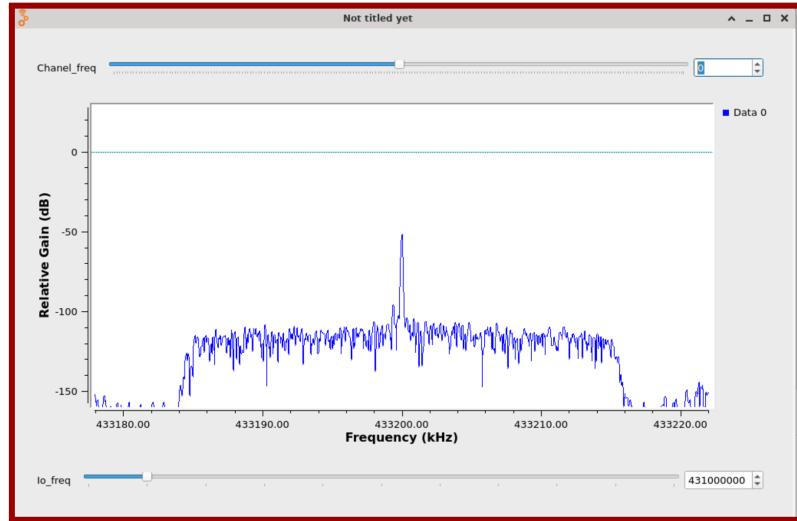


Signal centré au zéro :

Université Sorbonne Paris Nord

Département réseaux et Télécommunications

Parcours: Cybersécurité



Conclusion :

A la fin de cette partie, on arrive à:

- construire un modulateur AM
- reconstruire l'audio à la réception
- utiliser un PlutoSDR pour émettre/réception RF
- filtrer, resampler, centrer une bande RF
- visualiser les signaux en temps, fréquence et constellation (plus tard)
- assimiler les bases du traitement du signal numérique appliquée à la radio

Université Sorbonne Paris Nord

Département réseaux et Télécommunications

Parcours: Cybersécurité

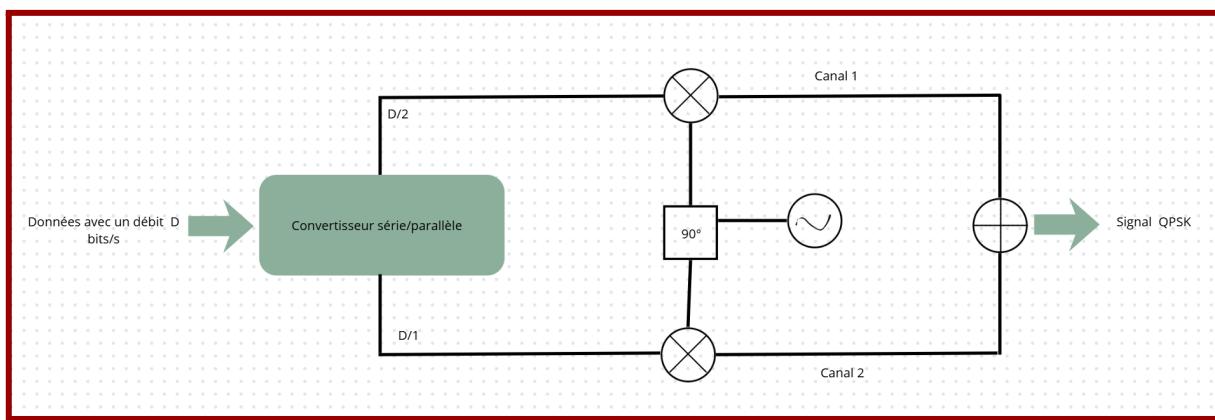
3ème TP (autonomie) : Format de modulation avancée

Remarque préalable : au niveau de la modulation QAM (réception et émission), j'avais quelques difficultés au niveau du réglage du domaine temporelle, malgré l'ajout du module PFB Clock Sync pour la récupération temporelle des symboles et le réglage des décalages horaires entre TX et RX, c'est pour cette raison qu'on arrive pas à visualiser clairement le signal modulé en quadrature

Contexte spécifique :

- Objectif : étudier des techniques de modulation plus complexes pour augmenter le débit et l'efficacité du signal.
- On travaille sur :
 - a. **QAM (Quadrature Amplitude Modulation)**, qui combine amplitude et phase pour transmettre plus d'informations par symbole.
 - b. **Multiplexage**, pour transmettre plusieurs signaux sur une même bande de fréquence.
- Ce TP prépare à des systèmes de communication modernes et plus performants, en simulant et testant des signaux complexes sur GNU Radio et ADALM-PLUTO.

1. le principe d'une modulation QAM (modulation d'amplitude en quadrature)



2. Quel est l'intérêt de la QAM ?

Elle permet de transmettre beaucoup plus de bits par symbole

Exemples :

- QAM-4 → 2 bits symboles
- QAM-16 → 4 bits par symbole
- QAM-64 → 6 bits par symbole

Université Sorbonne Paris Nord

Département réseaux et Télécommunications

Parcours: Cybersécurité

-
- QAM-256 → 8 bits par symbole

→ Plus de débit sans augmenter la bande passante

Elle utilise pleinement les deux dimensions du signal (I, Q)

Contrairement à :

- AM : 1 seule amplitude
- PSK : 1 seule phase
- ASK : 1 seule amplitude (binary)

Dans QAM :

amplitude + phase = 2 dimensions

donc beaucoup plus d'états possibles

Université Sorbonne Paris Nord

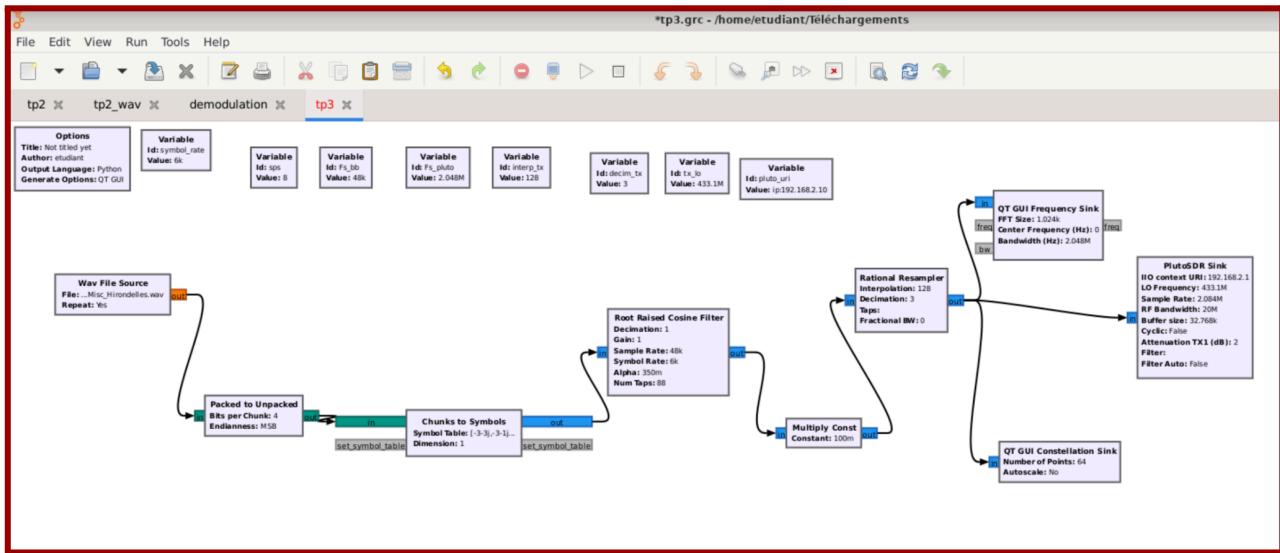
Département réseaux et Télécommunications

Parcours: Cybersécurité

Montage de système QAM fonctionnel

ARCHITECTURE 1: ÉMETTEUR QAM (TX)

Ici, on prend un flux binaire, le mapper en symboles QAM16, former le signal, et l'envoyer en RF via le PlutoSDR



1. Source de données (Random / File Source)

Génère un flux de bits → ce sont les données à transmettre

2. Packed to Unpacked (4 bits / symbole)

16-QAM → 4 bits représentent un symbole

Ce bloc convertit les octets en groupes de 4 bits

3. Constellation Modulator (Chunks to Symbols dans mon cas)

C'est le cœur du modulateur QAM

Grâce à la constellation 16-QAM :

- 4 bits → 1 symbole complexe ($I + jQ$)
- crée les points de la constellation

Paramètres essentiels :

- Samples/symbol (sps) = 8
- Constellation = **constel_16qam** (un objet constellation que j'ai créé)
- Rolloff (excess BW) = 0.35

4. Filtre Root Raised Cosine (RRC – shaping filtre)

- Applique la mise en forme des impulsions
- Réduit l'interférence inter-symboles
- Donne une bande occupée et contrôlée

Université Sorbonne Paris Nord

Département réseaux et Télécommunications

Parcours: Cybersécurité

5. Multiply Const (normalisation)

Ici on l'utilise pour éviter que le signal dépasse la valeur 1

Sinon on aura une saturation du Pluto (distorsion)

6. Rational Resampler (Baseband 48 kHz → RF 2.048 MHz)

Le Pluto doit fonctionner à haute fréquence d'échantillonnage

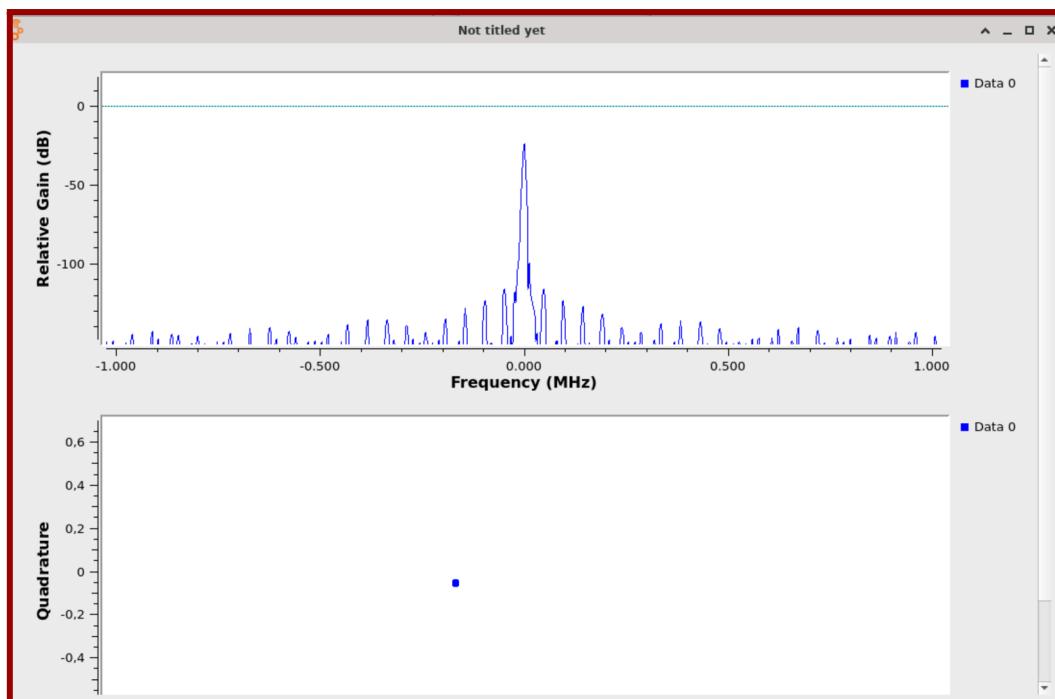
On applique alors :

- interpolation = 128
- décimation = 3

→ nouveau $F_s = 2.048 \text{ MHz}$

7. PlutoSDR Sink

Émet réellement en RF



Ce que fait réellement cette architecture TX :

- elle transforme des bits en symboles QAM
- elle met en forme le signal
- le suréchantillonné
- l'envoie par ondes radio via le PlutoSDR

→ On arrive à concevoir une chaîne de modulation numérique complète

Université Sorbonne Paris Nord

Département réseaux et Télécommunications

Parcours: Cybersécurité

ARCHITECTURE 2 : RÉCEPTEUR QAM (RX)

Ici on reçoit le signal RF QAM, on corrige les erreurs du canal, on récupère les symboles puis les bits

1. PlutoSDR Source

Capture le signal QAM RF

Fréquence = 433.1 MHz

Sample rate = 2.048 MHz

2. Frequency Xlating FIR Filter

Isoler le canal utile

→ centrer la bande de notre signal autour de **0 Hz**

3. Rational Resampler (RF 2.048 MHz → Baseband 48 kHz)

Même conversion que dans l'autre sens au TX, on ramène la bande de base à la bonne fréquence d'échantillonnage

4. Root Raised Cosine filter (RRC – matched filter)

Filtre adapté au filtre TX

Permet de :

- enlever le bruit
- reconstruire correctement les symboles
- améliorer le rapport signal/bruit (SNR)

5. PFB Clock Sync

Récupération temporelle des symboles

Il corrige les décalages d'horloge entre TX et RX

Paramètres que j'ai réglé :

- Samples per symbol = **8**
- Loop bandwidth = **0.001**
- Taps = même RRC que TX

→ Sans ce bloc, les symboles sont mal alignés, on aura donc une constellation floue

6. Costas Loop (Order = 4 pour QAM16)

Récupération de la phase porteuse

Il corrige :

- les rotations de constellation
- les erreurs de fréquence résiduelles

→ C'est lui qui stabilise la constellation QAM

Université Sorbonne Paris Nord

Département réseaux et Télécommunications

Parcours: Cybersécurité

7. Constellation Decoder (Hard Bits)

Son rôle c'est de transformer chaque point IQ en symbole, puis en série de bits

Configuration :

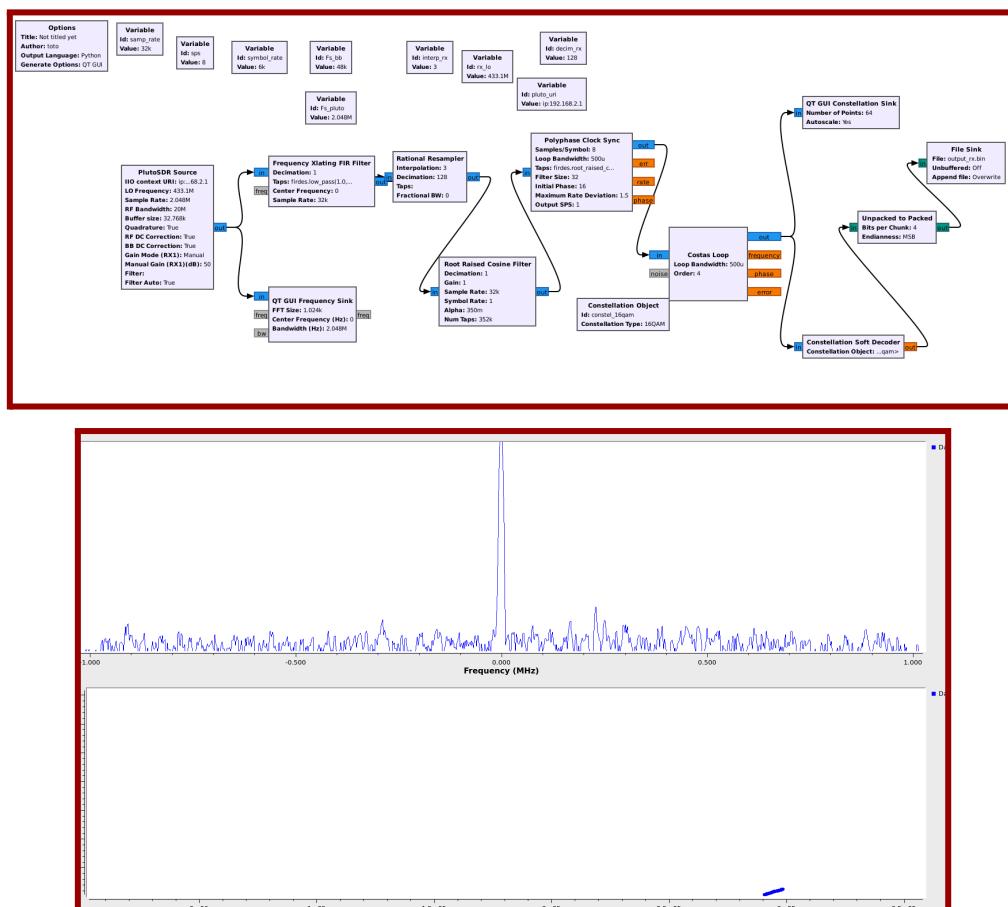
- constellation = **constel_16qam** (l'objet constellation que j'ai créé)
- décision = Hard bits

8. Unpacked to Packed

Convertit les 4 bits en 1 octet

9. File Sink

Enregistre les données reçues



Ce que fait réellement cette architecture RX :

- récupère les symboles QAM depuis l'air
- corrige le timing
- corrige la phase
- trie les points dans 16 cases
- récupère les bits d'origine

C'est une chaîne complète de **démodulation numérique QAM**

Université Sorbonne Paris Nord

Département réseaux et Télécommunications

Parcours: Cybersécurité

Conclusion :

A la fin de cette partie, on apprend :

- une construction d'un modulateur QAM réel
- une analyse d'une constellation
- l'importance du filtre adaptatif RRC
- la synchronisation temporelle (Clock Recovery)
- la synchronisation de phase (Costas Loop)
- la transmission/réception numériques avec PlutoSDR
- comment reconstruire un flux binaire transmis par radio
- une bonne approche d'une chaîne de communication numérique