TP 1 :

R. Zitouni, H. Badis

rafik.zitouni@esiee.fr

TP 1

Objectifs :

1) Découvrir l'environnement GNU Radio Companion

- Encodage d'un signal avec ses composantes **In-phase** et **Quadratique**
- Application d'une fonction FFT à une porteuse
- Interpolation et décimation

2) Créer des émetteurs et récepteur radio FM.

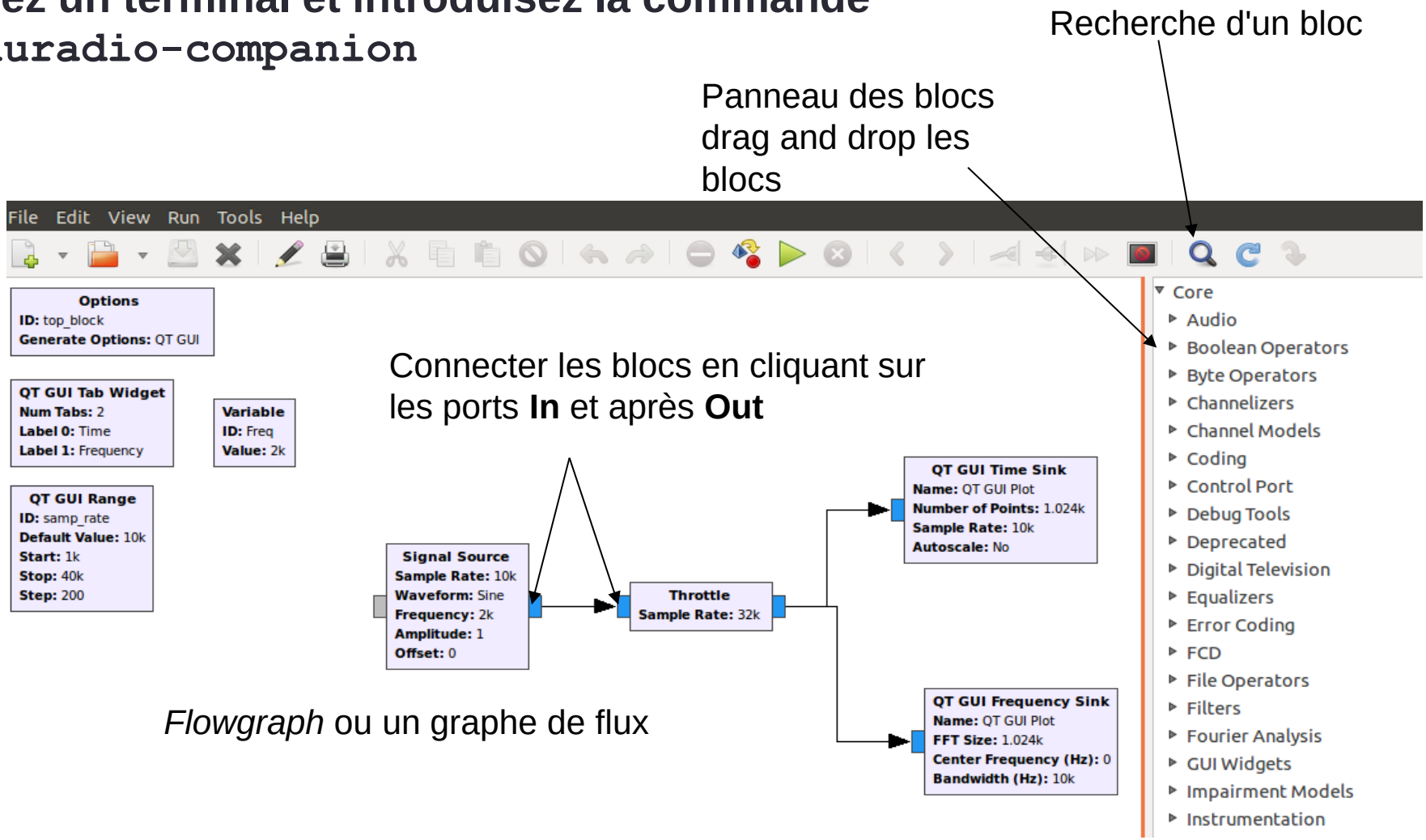
- Comment passer d'une forme d'onde analogique à un décodage numérique et vice versa.

Outils : Environnement Linux, GNU Radio, USRP B210.

Environnement GNU Radio companion

Lancez un terminal et introduisez la commande

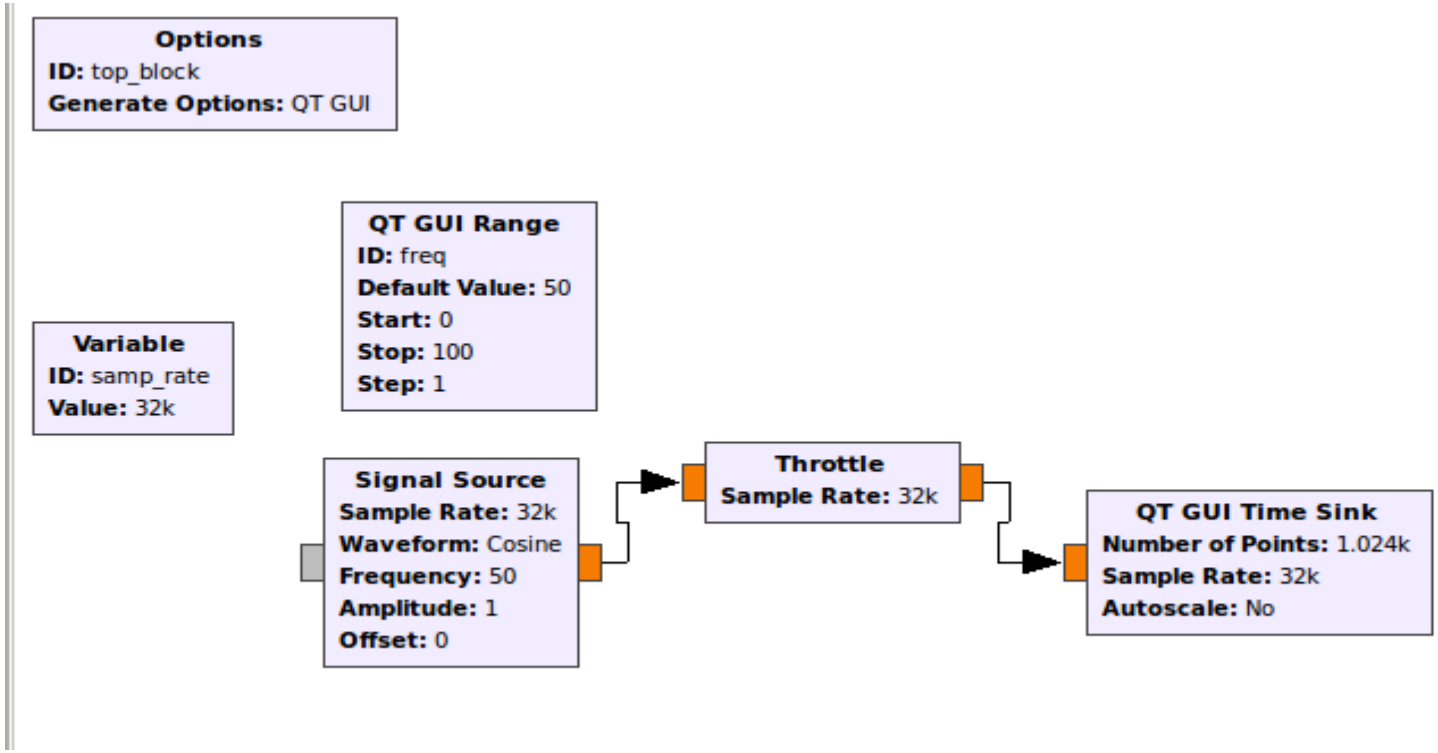
```
$ gnuradio-companion
```



Flowgraph ou un graphe de flux

Environnement GNU Radio companion

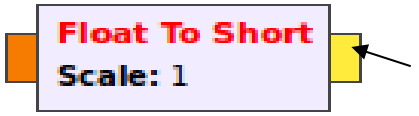
Tutorial 1:
Créez le graphe de flux suivant en reprenant les même paramètres pour chaque bloc :



Sauvegardez et après exécutez votre graphe de flux (**Premier.grc**)
Un fichier **Python** est généré dans le dossier de sauvegarde (**top_block.py**).
Ouvrez votre fichier **top_block.py** avec un éditeur de texte, par exemple (**gedit, vi, vim, emacs .etc**)

Environnement GNU Radio companion

Assurez vous d'avoir connecter les blocs avec une correspondance entre les sorties et après les entrées des blocs



short

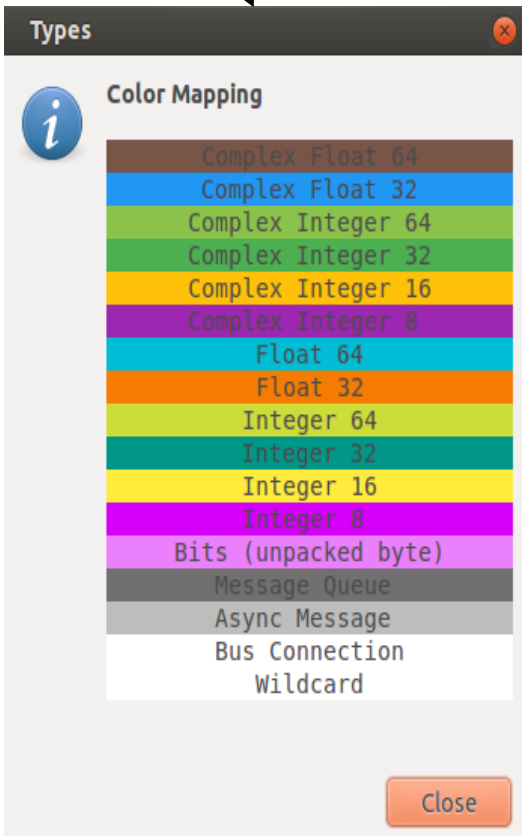
complex

float

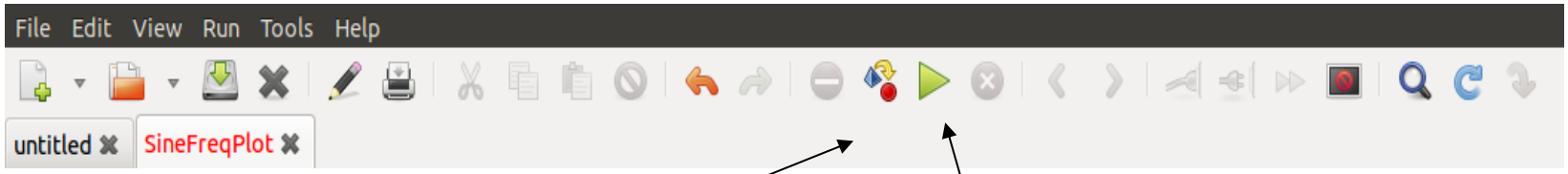
Cette fenêtre montre les message suite à l'exécution d'un graphe de flux

```
Generating: '/home/rafik/enseignement/ESIEE/Tutorials/NyquistShannon/top_block.py'

Executing: /usr/bin/python2.7 -u /home/rafik/enseignement/ESIEE/Tutorials/NyquistShannon/top_block.py

gr::log :INFO: controlport - Apache Thrift: -h rafik-Precision-7510 -p 9090

>>> Done
```

Types correspondants aux couleurs des ports **In** et **out** d'un bloc



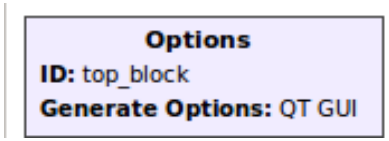
Environnement GNU Radio companion

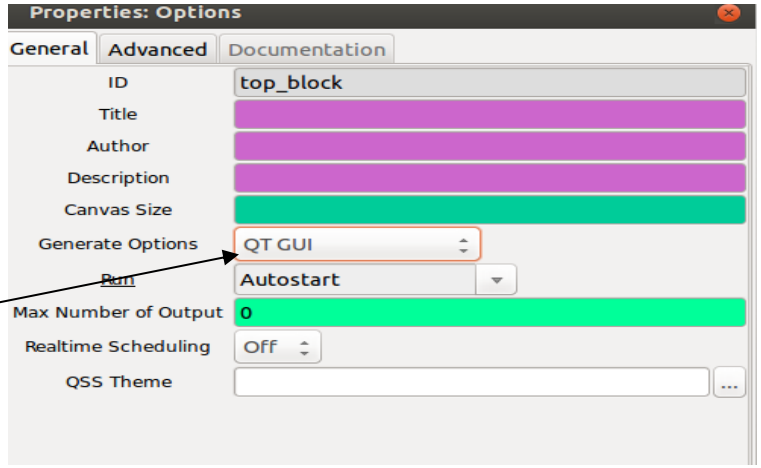


Génération du code python équivalent au graphe de flux **top_block.py**

Exécution d'un graphe de flux et génération d'un code python

Double click

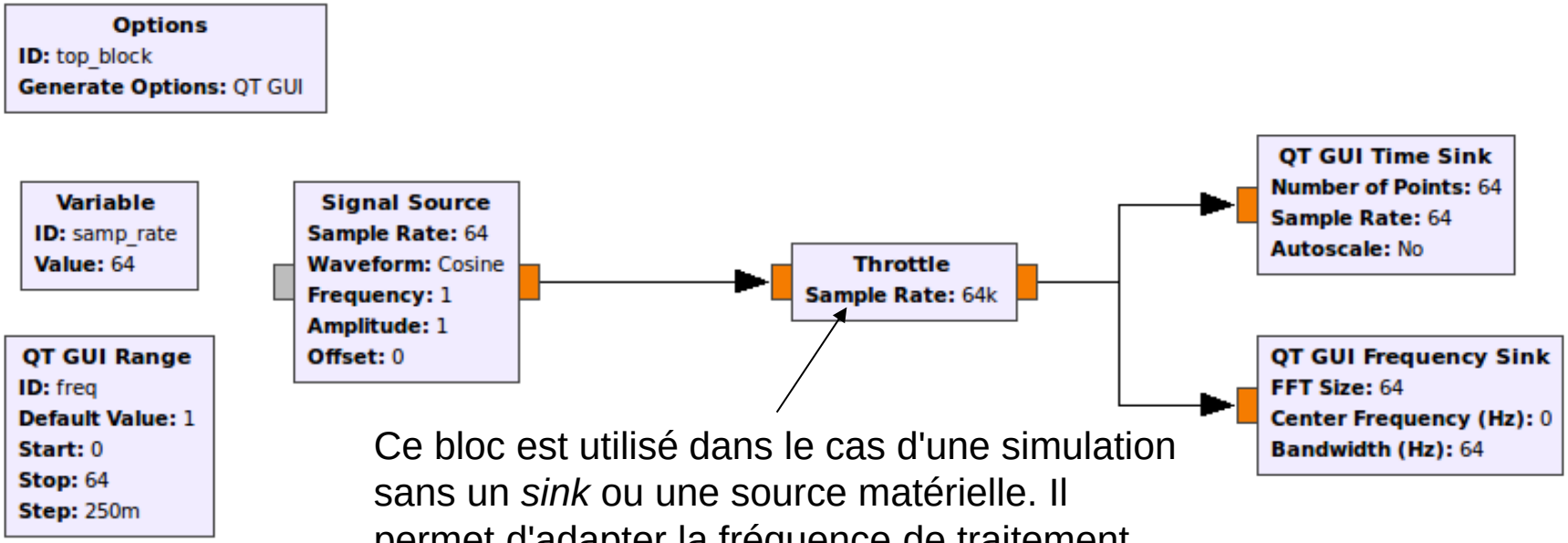




Initialisez les paramètres globales d'un graphe de flux. QT GUI ou WX GU comme *Library* graphiques pour votre GUI. Nous utilisons QT pour la stabilité de cette *Library*.

Environnement GNU Radio companion

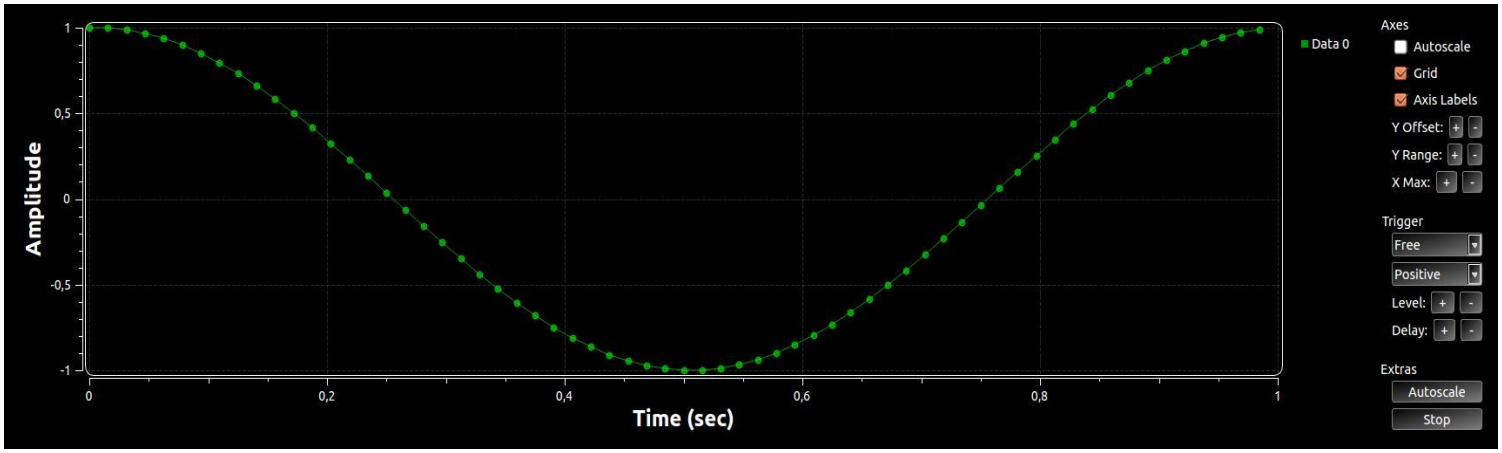
Exercice 1 :
Créez le graphe de flux suivant :

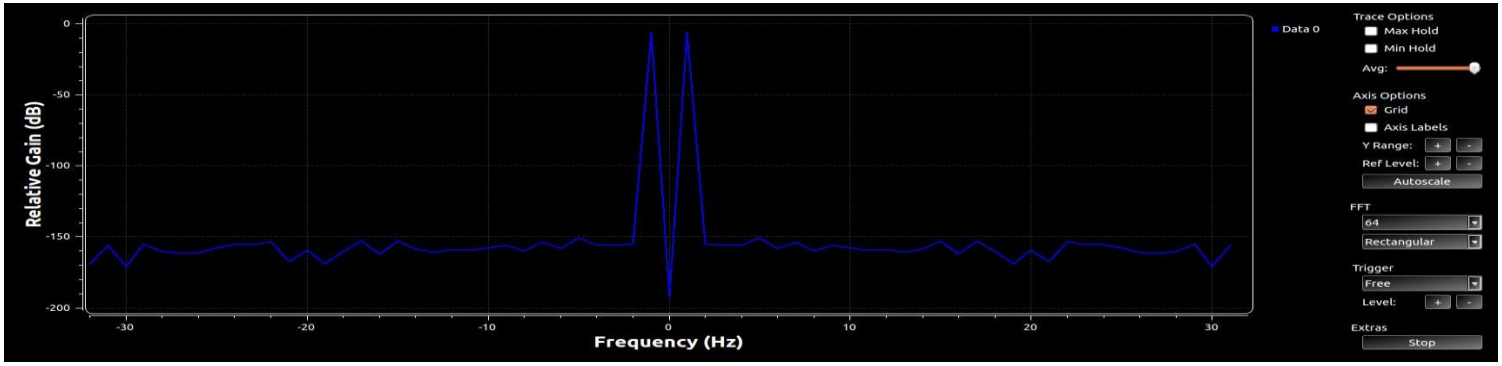


Ce bloc est utilisé dans le cas d'une simulation sans un *sink* ou une source matérielle. Il permet d'adapter la fréquence de traitement des échantillons (*samples*) à un taux inférieur à celui du CPU de l'ordinateur (*host*).

Environnement GNU Radio companion

Le résultat de l'exécution de votre graphe de flux devrait être comme suit :





Le fond noir peut être configuré pour toutes les interfaces QT. Il s'agit de modifier le fichier de configuration **~\.gnuradio\config.conf** en rajoutant la ligne suivante:
[QTGUI] qss = $prefix/share/gnuradio/themes/alt.qss

Environnement GNU Radio companion

Questions :

Q1) Configurez les paramètres du dernier graphe de flux afin d'avoir un affichage des formes d'ondes semblable à celui présenté dans le précèdent diaporama.

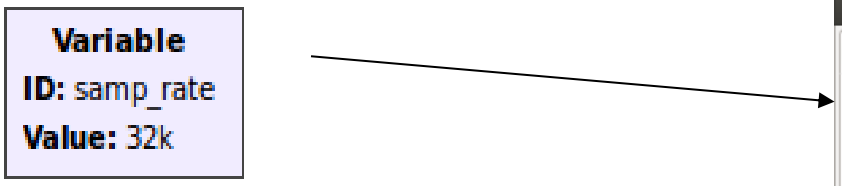
Q2) Si vous augmentez la fréquence à 32Hz, quelle est votre observation et votre commentaire par rapport à la forme de votre onde dans le domaine fréquentiel et temporel ?

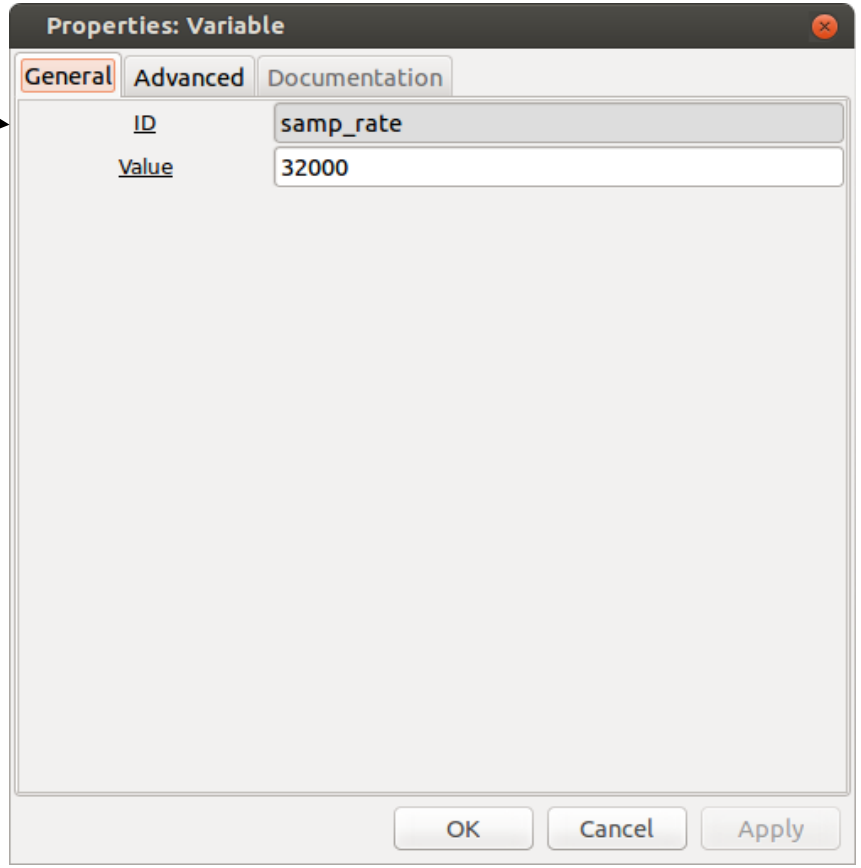
Q3) La même question quand la fréquence est égale à 63Hz ?

Q4) Utilisez le bloc « *Delay* » afin de changer la phase de votre forme d'onde (prise en compte d'un signal *Sine* et *Cosine*) → Sauvegarder ce changement dans un graphe de flux .grc

Environnement GNU Radio companion

Sample Rate ou Taux d'échantillonnage (DSP vs *Hardware*)





ID : (Python) nom de variable
Valeur : Expression en python
32000 → Entier
32e6 : 32000.0 (nombre en virgule flottante)
int(32e6) : 32000 (*cast* de nombre en virgule flottante à entier).

Environnement GNU Radio companion

Sample Rate (DSP)

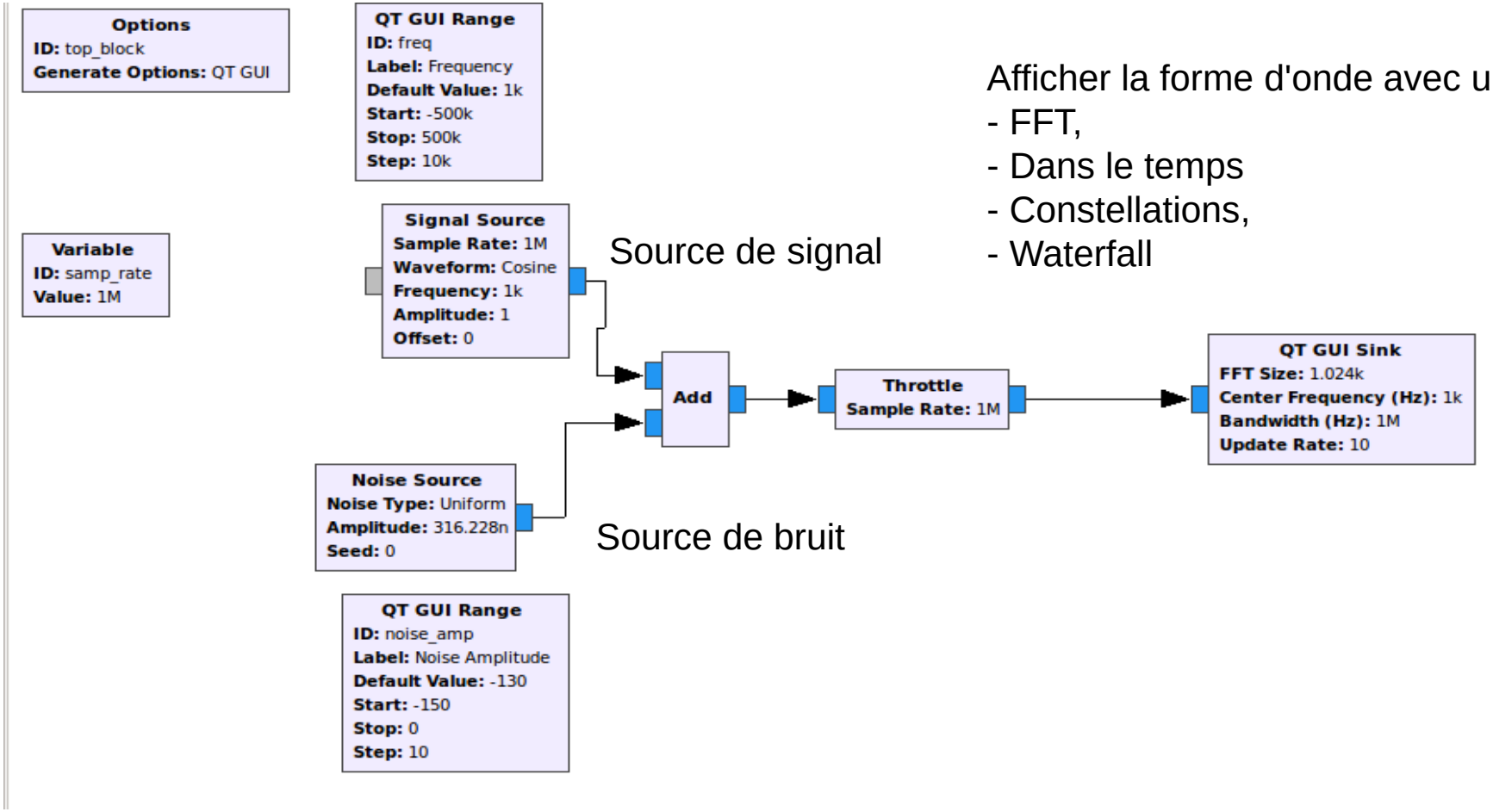
→ Dans le cas d'un calcul d'une forme d'onde, nous devons connaître le taux d'échantillonnage et la fréquence. (théorème de Nyquist)
→ Nous devrions s'assurer dans certains cas que tous les blocs DSP opèrent avec le même taux d'échantillonnage.

Sample Rate (Hardware)

→ Le taux d'échantillonnage réfère aussi au taux de passage des échantillons par les blocs d'un graphe de flux.
→ Sans ce contrôle de taux d'échantillonnage, le CPU traite avec sa vitesse maximale
→ Ce taux fixe aussi l'horloge (**clock**) de certains blocs comme ceux de l'USRP, carte sons ou *Throttle*.
→ Attention à la synchronisation des clocs des différentes sources. Un effet ***overflows/underruns*** sont provoqués dans le cas où la production et la consommation des échantillons n'ont pas la même vitesse.

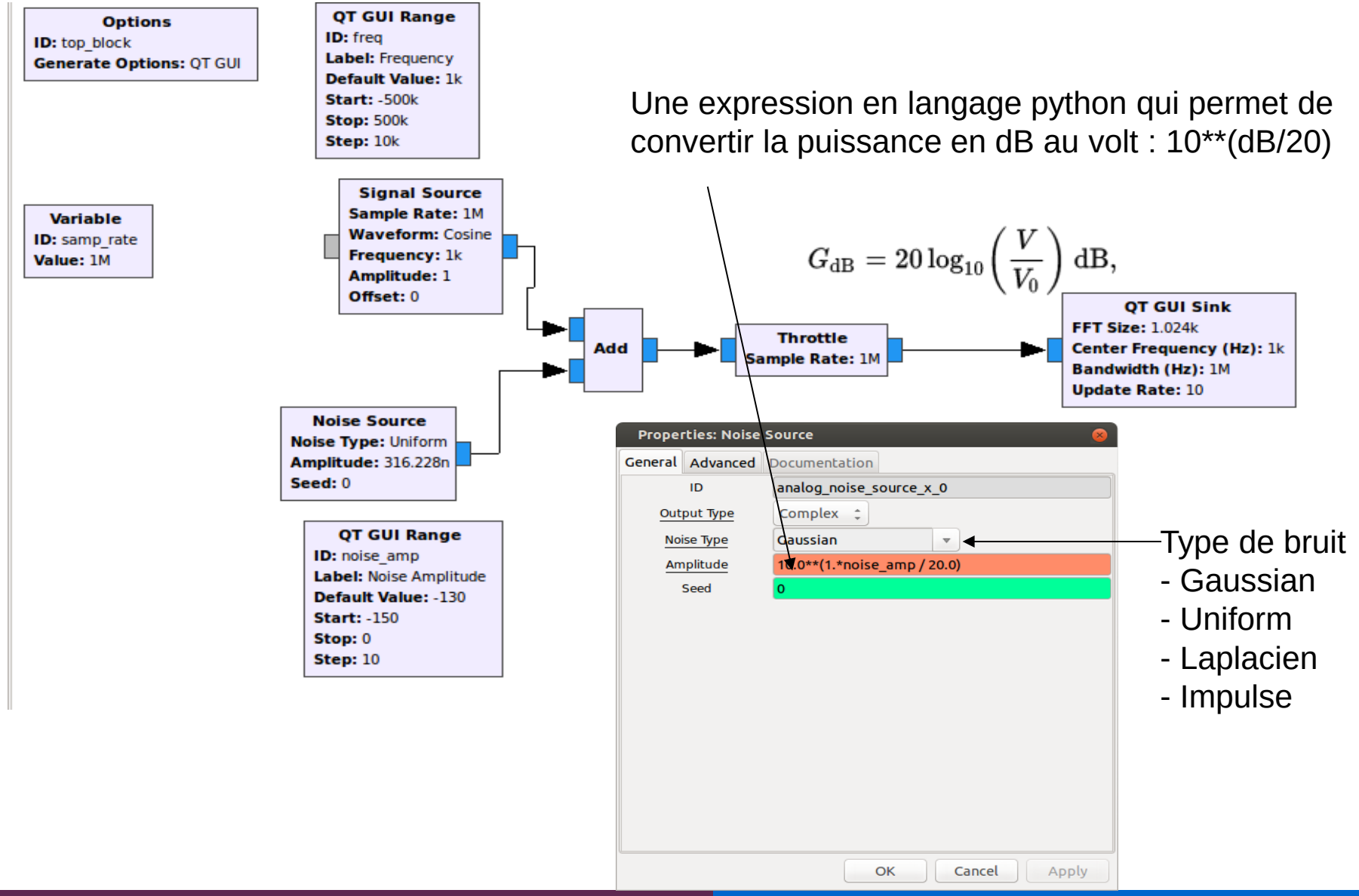
Environnement GNU Radio companion

Générez votre porteuse et ajouter à cette porteuse un bruit et afficher le résultat :



Afficher la forme d'onde avec une
- FFT,
- Dans le temps
- Constellations,
- Waterfall

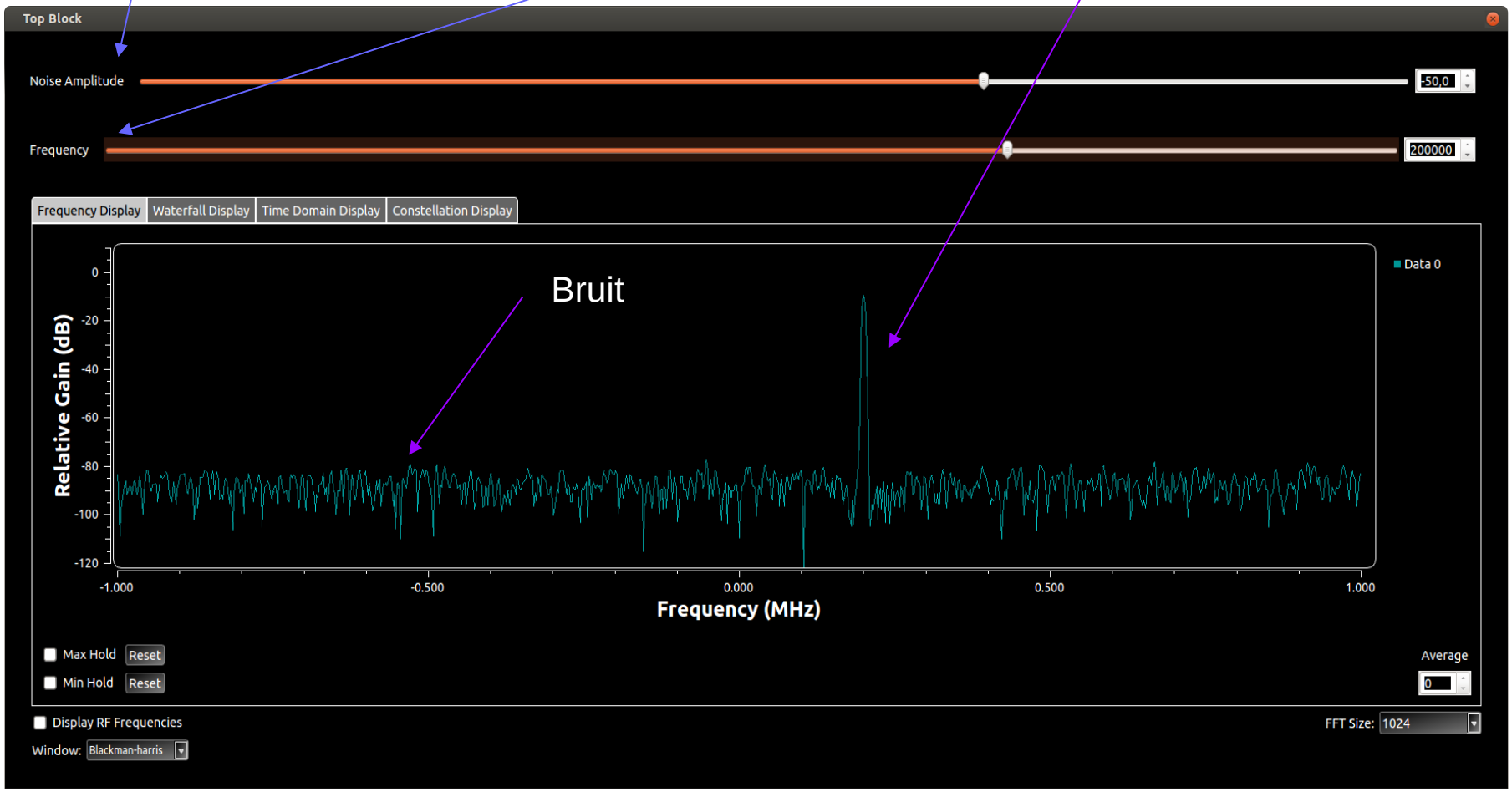
Environnement GNU Radio companion



Environnement GNU Radio companion

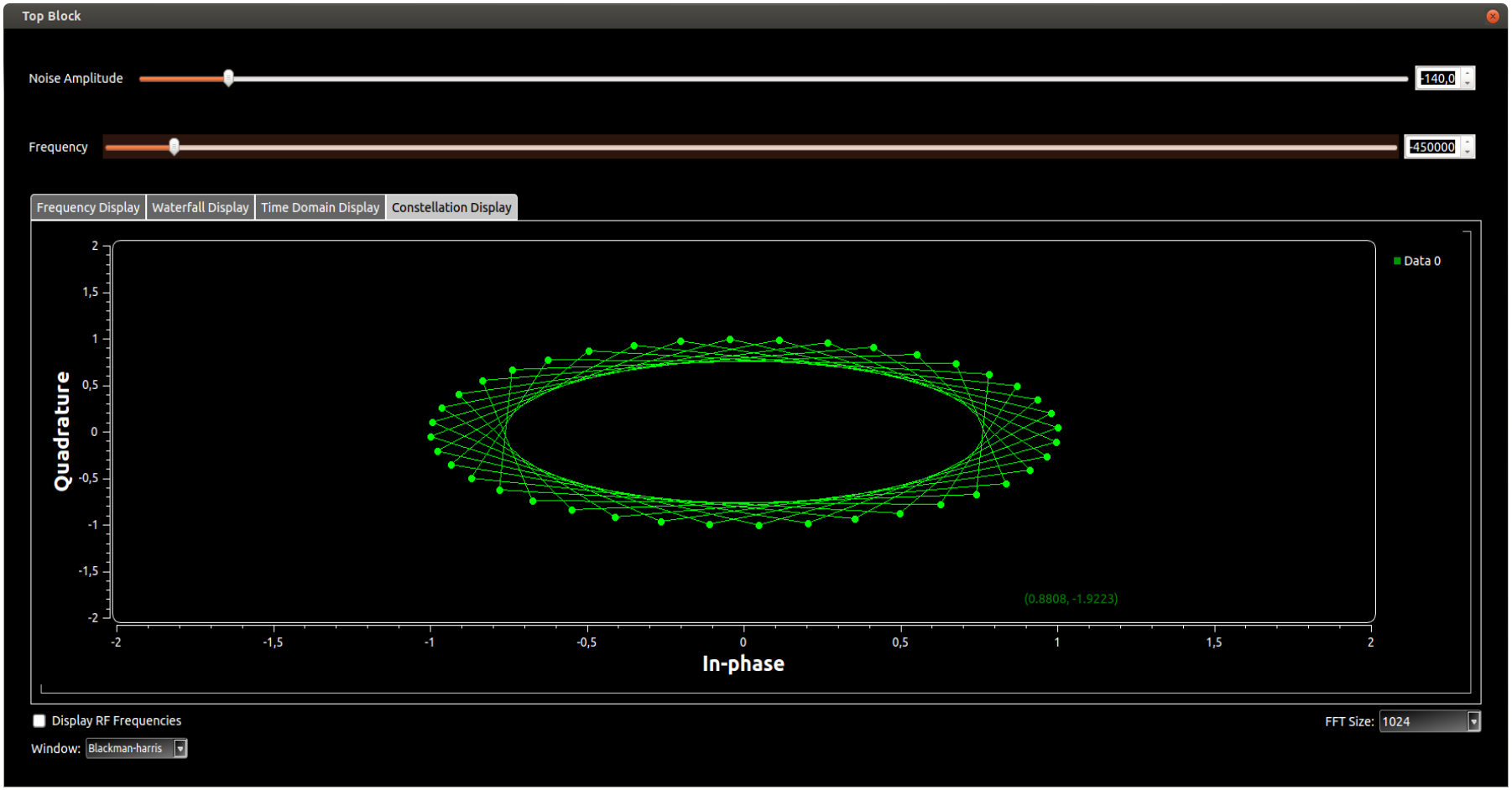
Amplitude du bruit ainsi que la fréquence

Pique correspondant à 1KHz *ton* (porteuse)



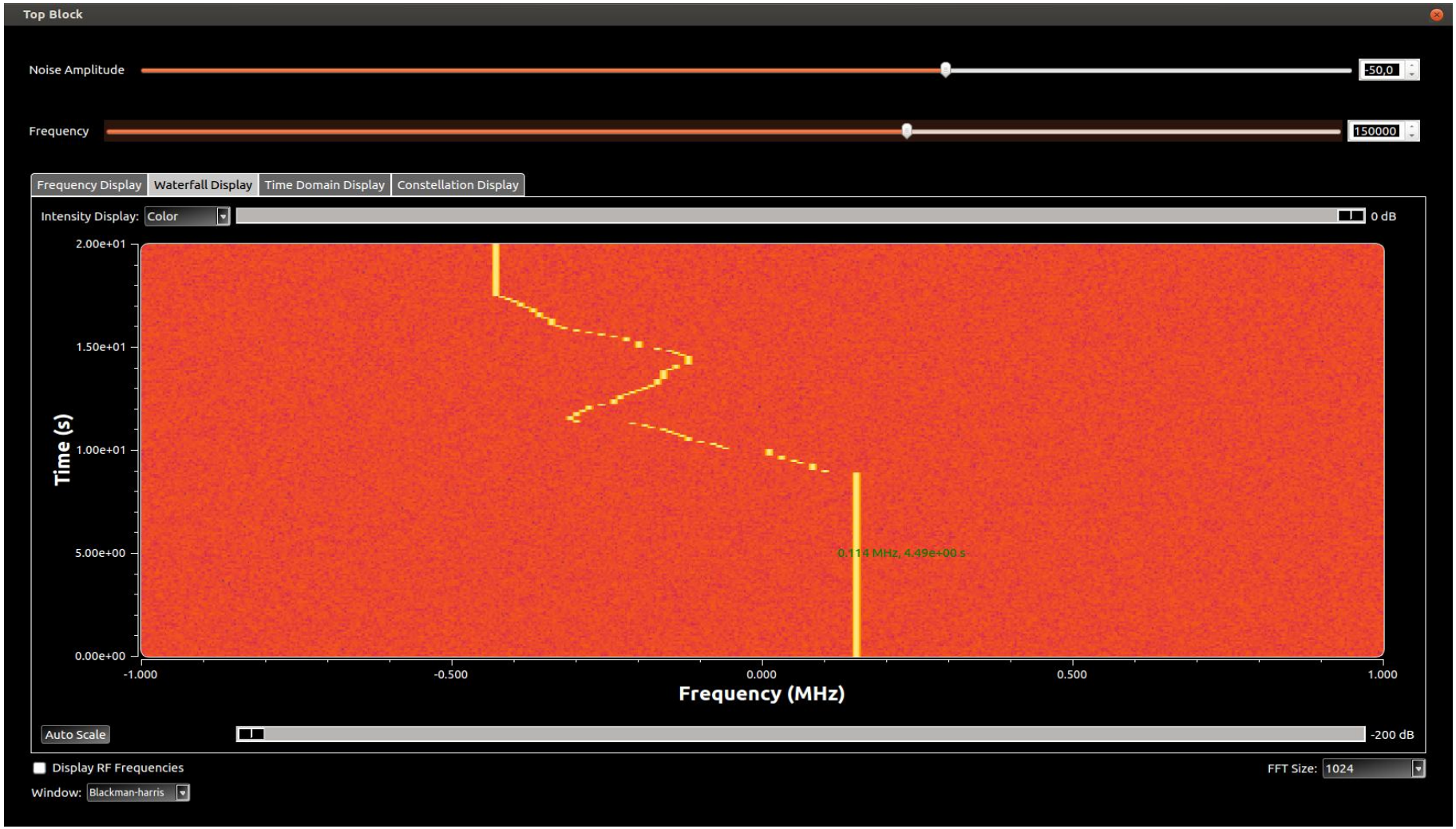
Environnement GNU Radio companion

Affichage d'une constellation du signal avec les deux composantes **Quadratique** et **In-phase.** La rotation est dans le sens inverse des aiguilles d'un montre.



Environnement GNU Radio companion

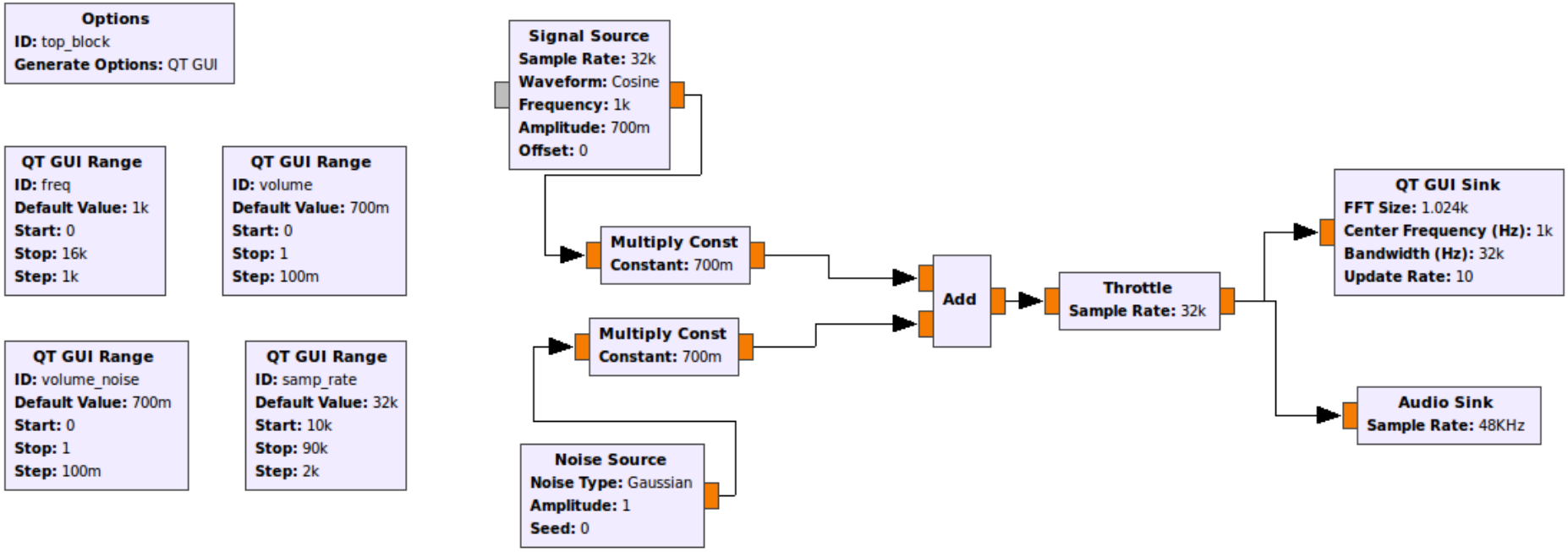
Waterfall Display : Cette affichage donne l'état du spèctre dans le temps.



Encodage pour l'audio

Exercice 2 :
L'objectif de ce graphe de flux est de jouer avec le taux d'échantillonnage, sur-échantillonnage et sous échantillonnage (*Up and Down Sampling*).

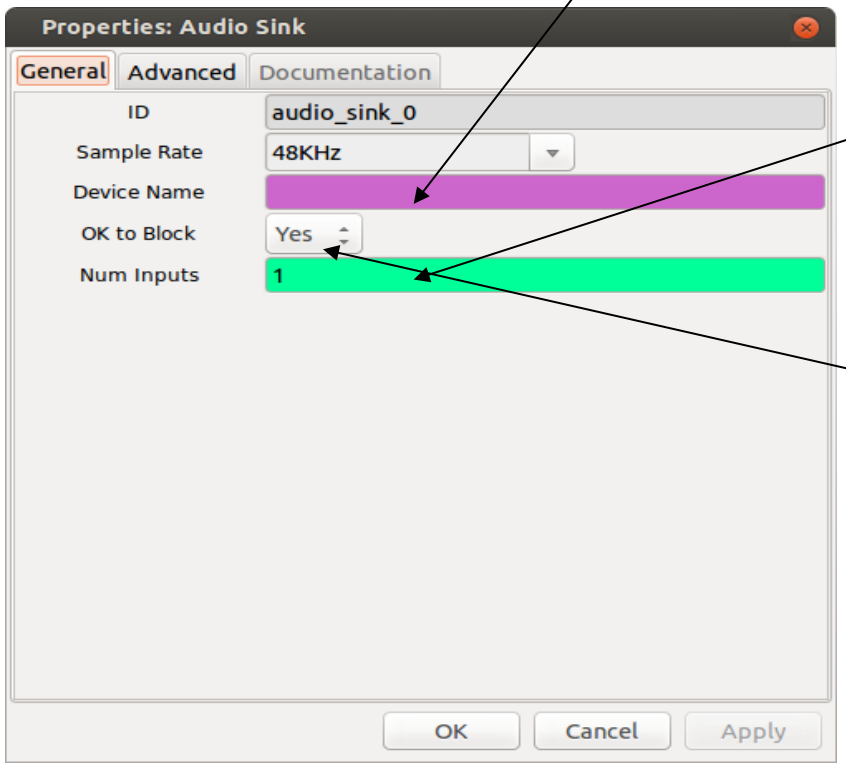
Créez votre graphe de flux suivant :



Q1) Quelle est votre observation quand le samp_rate < 48KHz, = 48 KHz et > 48 Khz ?

Encodage pour l'audio

L'identifiant Vide implique la carte audio par défaut.
Avec une carte ALSA, vous pouvez mettre « *pluse* »



1 pour un seul canal,
2 pour *stereo*

- *Blocking* mode en mettant les échantillons dans le buffer de la carte audio.
- *Non-Blocking* mode avec une attente de la consommation des échantillons.

Il est préférable d'avoir un *blocking mode* dans le cas d'une seul carte matérielle dans un graphe de flux.

Q2) Utilisez le *non-blocking* mode. Quelle est votre remarque par rapport à la consommation des échantillons et la qualité de votre son ?

USRP B210

Paramètres RF de la carte USRP B210

Couverture en bande de fréquence RF	**Fenêtre des fréquences couvertes**	**Fréquence d'échantillonage**
50 MHz – 6000 MHz	200 Khz – 56 MHz	Jusqu'à 61.44MS/s quadrature *

* La fréquence d'échantillonnage physique de la carte (ou **M**aster **C**lock **R**ate) peut être initialisée à des valeurs entre 5 MHz et 61,44 MHz (pour Un mode dual-channel, cette valeur est égale à 30,72 MHz). Notez qu'un taux supérieur ou égale à 56 MHz est possible, mais il est préférable qu'il soit inférieur à cette valeur.

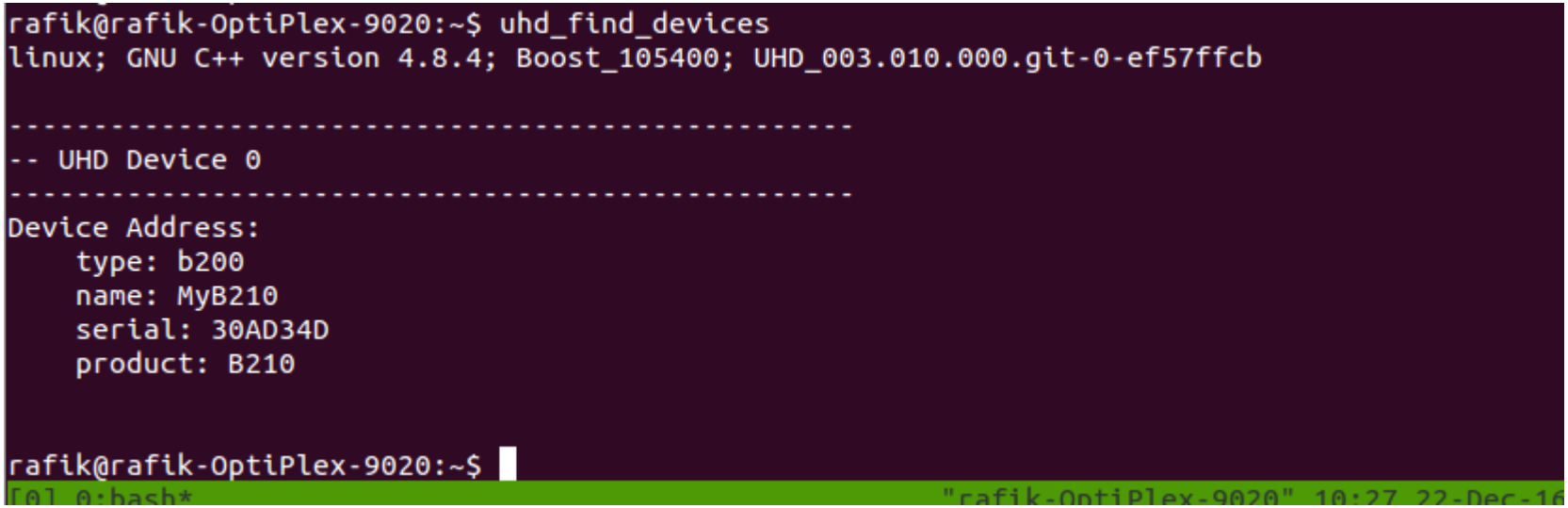
Puissance d'émission (dBm)	**Gain (dB)**
2 TX/RX power +20dBm max 2 RX power -15dBm max	0 – 89,8 dB (pas de 0,2 dB)

Vous pouvez visualisez certains de ces paramètres avec la commande :
$ sudo uhd_usrp_probe

USRP B210

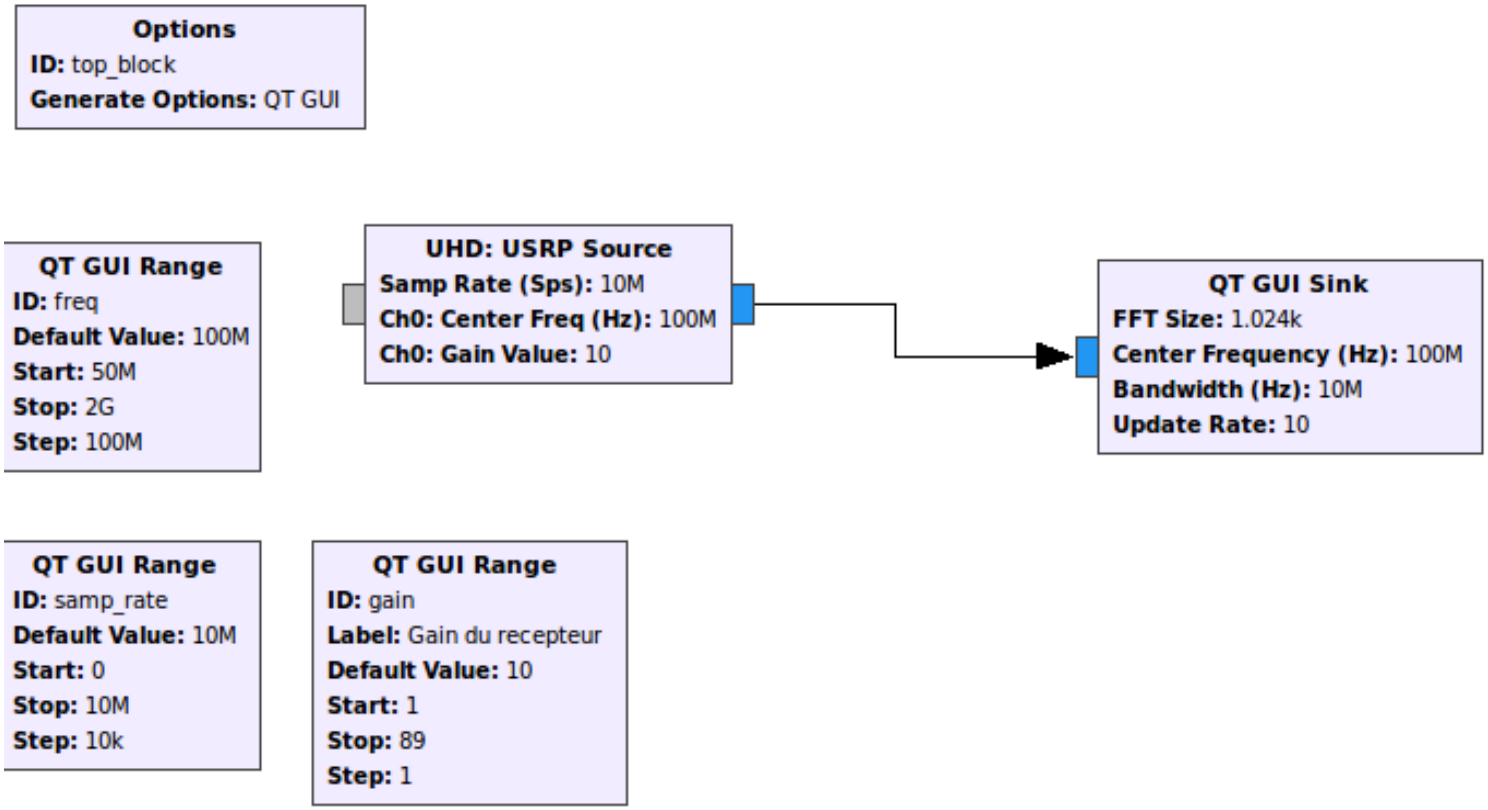
Assurez vous que votre carte USRP B210 est connectée à votre host via l'USB 2. Vérifiez si le driver UHD a *mappé* cette carte. Introduisez la commande suivante : **$ uhd_find_devices**

Vous devriez avoir le message suivant :



Exercice 3 :

Construisez le graphe de flux suivant :



Exercice 3 :

Q1) Quelle est la fréquence d'échantillonnage maximale d'un graphe de flux ?

Q2) Quelle est la bande passante couverte par un graphe de flux ?

Q3) Affichez une capture du *waterfall display* de la bande de fréquence de 95 MHz à 105 MHz. Commentez le résultat obtenu ?

Q4) La même question pour la bande 1800 MHz et 2400 MHz ?

Q5) Donnez une interprétation au message suivant en changeant le taux d'échantillonnage (*samp-rate*).

UHD Warning:
The requested decimation is odd; the user should expect CIC rolloff.
Select an even decimation to ensure that a halfband filter is enabled.
decimation = dsp_rate/samp_rate -> 107 = (32.000000 MHz)/(0.300000 MHz)

Récepteur FM

L'objectif est d'utiliser **des chaines d'encodage/décodage** d'un flux numérique via un modulateur/démodulateur FM.

Nous n'allons pas détaillé la formulation mathématique d'un modulateur/démodulateur FM, par contre, il faut maîtriser les paramètres nécessaires pour décoder et encoder le *Stream* numérique. L'objectif est de fixer des valeurs correctes à vos paramètres.

Quelques valeurs de paramètres :

Une station radio FM occupe un canal d'une bande passante de **250 Khz**.

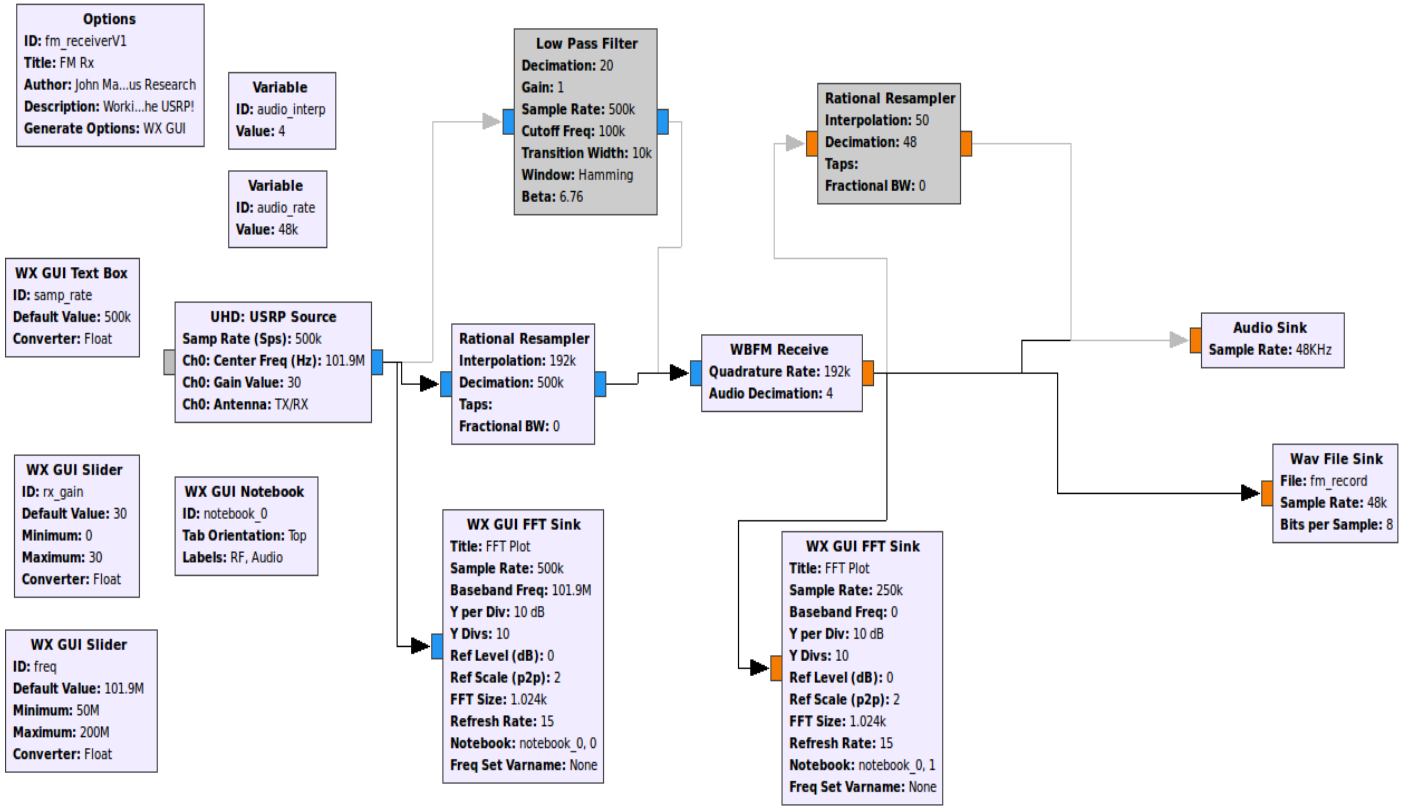
La fréquence d'échantillonnage d'une carte son est de **48 Khz.**

Récepteur FM

Exercice 4 :

Assurez vous que vous écoutez une chaîne radio en utilisant le graphe de flux **fm_receiver.grc.** Par exemple, Europe 1 à la fréquence 99,4 MHz.

Q1) Commentez la fonction de chaque bloc de cette chaîne.



Récepteur FM

Questions :

Q2) *Rational Resampler* utilise deux paramètres **Interpolation** et **décimation**. Illustrez le rôle de ce bloc et de ces paramètres avec un GUI Sink d'une FFT d'un signal sinusoïdal.

Q3) Formulez le rapport entre le taux d'interpolation et de décimation et les taux d'échantillonnage en entrée et en sortie?

Default ***M**aster **C**lock **R**ate (MCR) Sample rate* doit être un entier et il devrait être divisible par **4** pour de meilleures performances. L'initialisation est automatique, mais il peut prendre une valeur *static* "*master_clock_rate=X*" où X est une valeur en Hz.

Q4) Redefinissez un nouveau graphe de flux, mais avec un démodulateur à bande étroite ou Narrow Band FM (NBFM Receiver).

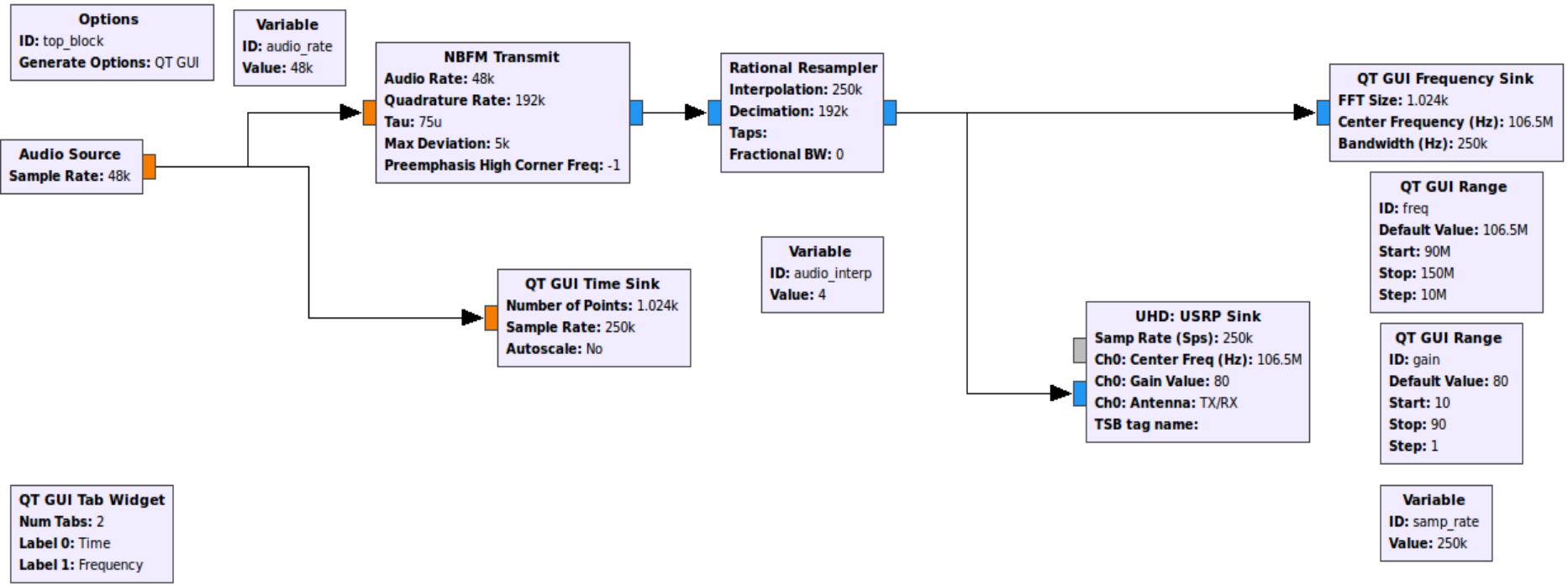
Q5) Donnez l'équivalent de ce récepteur avec une GUI QT.

Q6) Proposez un graphe de flux d'un récepteur FM avec un filtre passe bas, *LowPass Filter*. **(Bonus)**

Récepteur FM

Exercice 5 :

Reconstruisez la chaîne d'encodage d'un modulateur FM.



Récepteur FM

En exécutant ce graphe de flux et après un certain temps, vous recevez le message suivant :

-- Actually got clock rate 32.000000 MHz.
-- Performing timer loopback test... pass
-- Performing timer loopback test... pass
gr::log :INFO: audio source - Audio source arch: alsa
UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU

Q1) Expliquez pourquoi vous avez eu ces UUUUUUU ou *underrun* ?

Q2) Comment éviter ce phénomène *d'underrun* ?

Q3) Utilisez une source de fichier wave pour générer un *Stream* numérique en format .*wav*. Quelle est la portée de votre émetteur radio FM ?

Rendu TP1

Le rendu TP doit être réalisé en équipe de **4 étudiants au maximum**. Le rendu des étudiants doit être compressé dans un seul fichier au format **tar.gz** (Linux). Le nom de ce fichier est composé du nom, du prénom et du numéro de groupe d'un seul étudiant de l'équipe. Par exemple, pour Dupont Maxime du groupe 2 le rendu doit être : **Dupont_Maxime_ 2.tar.gz**

Le rendu doit contenir les fichiers suivants :

- **TP1.pdf ou .doc** : Réponses aux exercices de 1 à 5. Captures d'écran et commentaires

- **Exercice1_Question4.grc**, **Exercice4_Question2.grc**, **Exercice4_Question6.grc** : Graphes de flux correspondant aux réponses.

Date limite des rendus :

Vendredi 13/01/2017 à mi- nuit (00h00)