

Report : Introduction to Software-Defined Radio

Amoür Chadi, Amoür Redouane, Bouisson Arnaud

5ISS, 2021

I – Introduction

Au cours de ces trois TP, nous allons nous concentrer sur la réception de signaux réels. Nous verrons qu'un même récepteur peut enregistrer la quasi-totalité de ces signaux puis les traiter numériquement à l'aide de GNU Radio afin de récupérer toutes les informations.

Dans une première partie, nous travaillerons sur la théorie des émetteurs-récepteurs IQ, la modulation et la démodulation. Dans une seconde partie, nous travaillerons sur la diffusion radio FM et les signaux associés RDS. Nous allons démoduler un signal radio FM en utilisant GNU Radio. Enfin nous implémenterons la couche physique d'un objet communicant en BPSK pour transmettre un message de notre choix (texte, audio, image, etc.).

Les objectifs de ce projet sont :

- faire le lien entre les cours, la théorie et les applications
- comprendre la théorie des télécommunications afin de mettre en œuvre des solutions pour les couches physiques IoT
- comprendre le fonctionnement du SDR (Software-Defined Radio) et ses avantages
- utiliser des outils comme GNU Radio et USRP pour imaginer et implémenter des applications

II – In-phase/Quadrature Software-Defined Radio transceiver

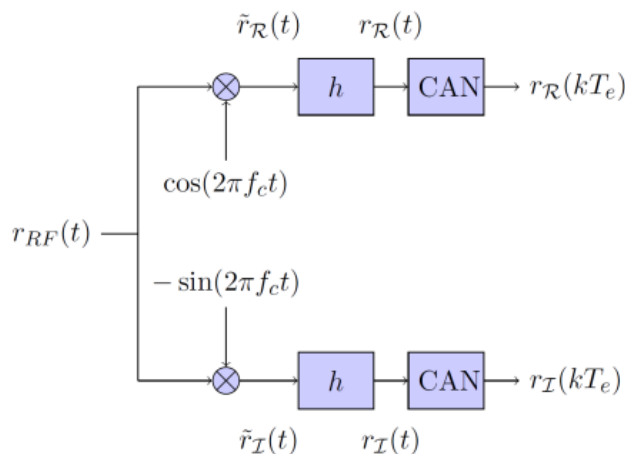


Figure 1 : Block diagram of the receiver

Q1) On exprime les signaux $\tilde{r}_R(t)$ et $\tilde{r}_I(t)$ en fonction de $s_R(t)$, $s_I(t)$, f_0 et f_c .

$$\tilde{r}_R(t) = (s_R(t)/2) * (\cos[2\pi(f_0 - f_c)t] + \cos[2\pi(f_0 + f_c)t]) - (s_I(t)/2) * (\sin[2\pi(f_0 + f_c)t] + \sin[2\pi(f_0 - f_c)t]).$$

$$\tilde{r}_I(t) = (-s_R(t)/2) * (\sin[2\pi(f_0 + f_c)t] + \sin[2\pi(f_c - f_0)t]) + (s_I(t)/2) * (\cos[2\pi(f_0 - f_c)t] - \cos[2\pi(f_0 + f_c)t]).$$

Q2) Si on veut que $r_R(t) = s_R(t)$, il faut que le filtre $H(f)$ soit un filtre passe bas.

$$r_R(t) = h(t) * \tilde{r}_R(t) \text{ ou } R_R(f) = H(f) * \tilde{R}_R(f).$$

Produit de convolution Produit classique

On veut $R_R(f) = S_R(f)$, alors $H(f)$ doit être un filtre passe bas de fréquence de coupure f_{low} tel que $B/2 \leq f_{low} \leq 2f_0 - B/2$. Dans la pratique, on limitera la fréquence de coupure, c'est à dire qu'on choisira une fréquence proche de $B/2$ pour limiter la puissance du bruit. De même avec cette condition, sur la voie I, on aura $R_I(f) = S_I(f)$ et $r_I(t) = s_I(t)$.

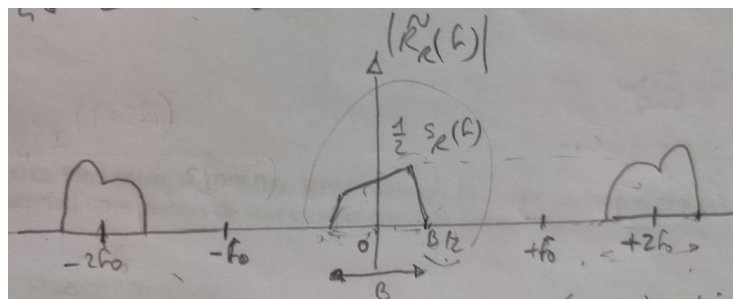


Figure 2 : Representation of $\tilde{R}_R(f)$

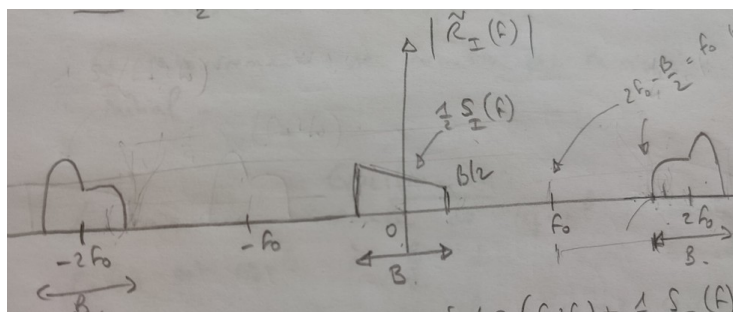


Figure 3 : Representation of $\tilde{R}_I(f)$

Q3) Le récepteur de la figure 1 ne pourra pas traiter des signaux large bande au-delà de la fréquence f_{\max} supérieure à $2f_0 - B/2$ car sinon, il y aura du recouvrement de spectre, et ce recouvrement ne permettra pas de récupérer le signal initial (respect du théorème de Shannon).

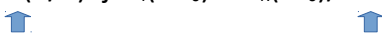
Q4) La fréquence maximale du signal en bande de base $r_R(t)$ est égale à $B/2$ donc d'après le théorème de Shannon, on peut échantillonner ce signal à $2*(B/2) = B = f_e$ donc $T_e = 1/f_e = 1/B$.

Q5) Si on intervertit la transposition et le CAN, alors la fréquence maximale du signal vu par le CAN est égale à $f_0 + B/2$. Il faudrait alors échantillonner ce signal, d'après le théorème de Shannon, à la fréquence $f_e = 2*(f_0 + B/2) = 2f_0 + B$.

Q6) $S_a(t) = S_{RF}(t) + j*H[S_{RF}(t)]$: signal analytique de $S_{RF}(t)$.

$S_a(f) = S_{RF}(f) + j*[-j*\text{sgn}(f)]*S_{RF}(f)$: spectre du signal analytique de $S_{RF}(t)$, avec $\text{sgn}(f) = +1$ si $f > 0$
 -1 si $f < 0$
 0 si $f = 0$

$$S_{RF}(f) = S_R(f+f_0)/2 - (1/2)*j*S_I(f+f_0) + S_R(f-f_0)/2 + (1/2)*j*S_I(f-f_0).$$



Spectre centré autour de $-f_0$ Spectre centré autour de $+f_0$

Donc après développement, $S_a(f) = S_R(f-f_0) + j*S_I(f-f_0)$.

III – Reception of frequency modulation (FM) broadcasting

Q7) Le bloc Variable permet d'associer un nom à une valeur. Le bloc File Source permet de récupérer et importer un fichier binaire pour ensuite faire du traitement dessus. On peut également choisir de répéter le son (nécessaire). Le bloc Throttle permet d'imposer une fréquence d'échantillonnage et limiter l'utilisation du processeur pour avoir un traitement continu d'information et avoir l'impression d'avoir du temps réel. Le bloc Frequency Sink permet de faire une FFT du signal et de l'afficher. Waterfall permet une représentation temporelle et fréquentielle. Time Sink permet une représentation temporelle, et Frequency Sink permet une représentation fréquentielle.

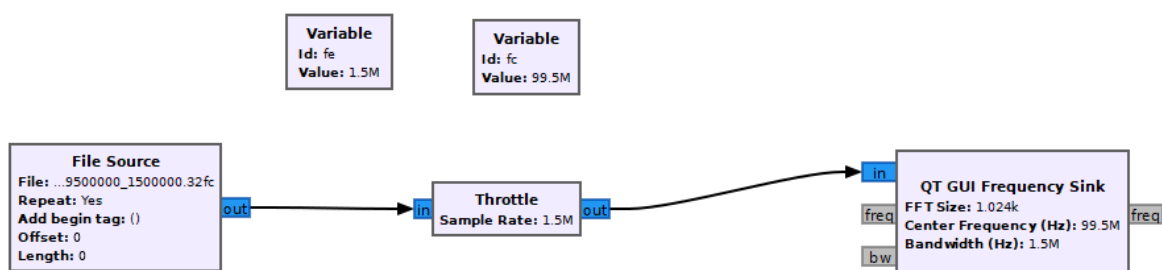


Figure 4 : Frequency analysis processing chain on GRC

Q8) On réglera la fréquence d'échantillonnage du Throttle sur la même fréquence d'échantillonnage du signal initial. On réglera la fréquence centrale du QT GUI à 99,5MHz et la largeur de bande égale à la fréquence d'échantillonnage (d'après Shannon).

Q9)

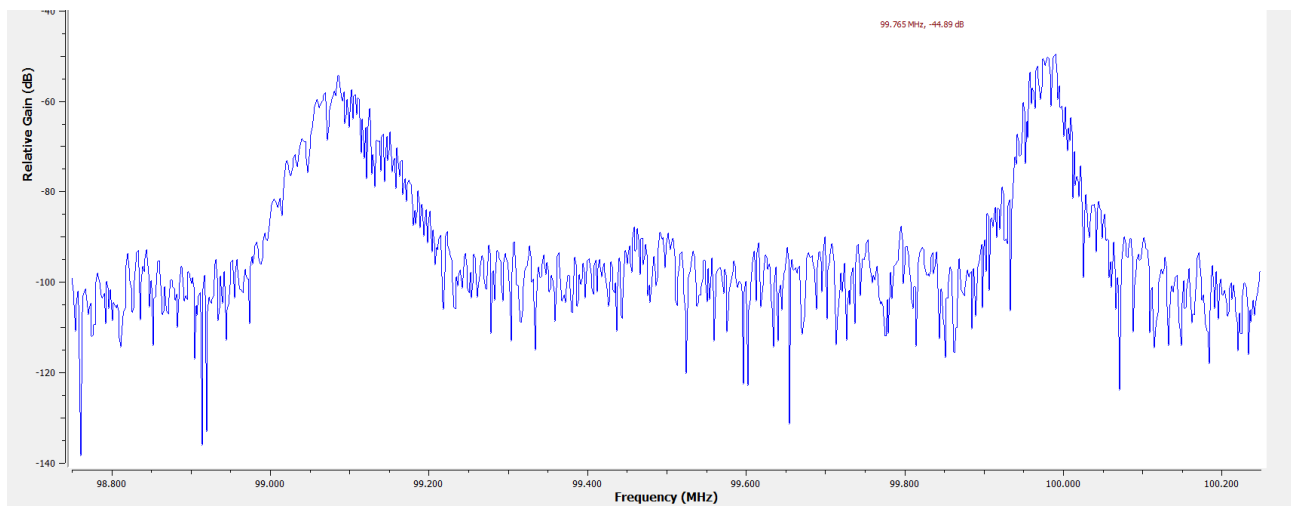


Figure 5 : Représentation fréquentielle du signal initial

On observe trois fréquences 99,1MHz / 99,5MHz / 100MHz : RFM Toulouse, Nostalgie Toulouse, et Skyrock, grâce à l’affichage Waterfall, qui nous permet également de confirmer que c’est une modulation en fréquence.

Q10) RFM / Skyrock / Nostalgie.

Min : 25dB / 15dB / 6dB.

Max : 37dB / 44dB / 10dB.

Average : 37dB / 37dB / 6dB.

Si l’on considère le SNR en mesure moyenne, la puissance reçue pour RFM et Skyrock égale à 37dB sera largement suffisante pour démoduler le signal. Pour Nostalgie (6dB), cela peut éventuellement être un problème.

Q11) On choisit de calculer la bande passante approximative d’un canal en fonction du seuil -3dB.

On trouve une largeur de bande égale environ à 120kHz.

Q12) On souhaite traiter les signaux autour de la fréquence nulle en radio logicielle. Bande passante égale à f_e donc entre $-f_e/2$ et $f_e/2$ pour notre bande passante. On a une fréquence centrale à 99,5MHz, donc on utilisera des offsets égaux à +500kHz pour ramener le signal centré à 100Mhz en bande de base et -400kHz pour ramener le signal centré à 99,1MHz en bande de base.

Q13)

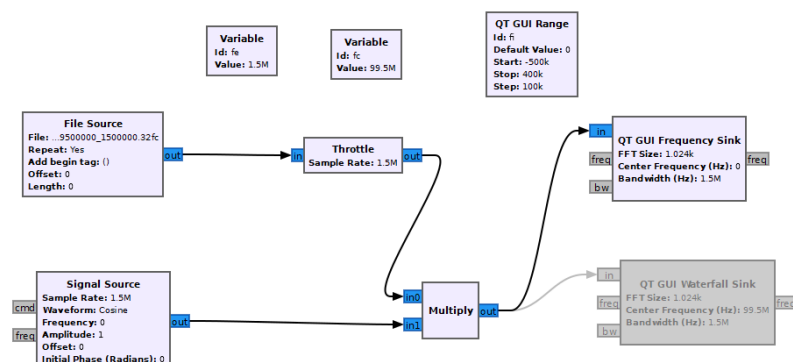


Figure 6 : Channel extraction by frequency transposition

Si notre offset fréquentiel est plus important que la fréquence d'échantillonnage, cela revient à réaliser une permutation circulaire du signal car il a une période en fréquence égale à la fréquence d'échantillonnage f_e .

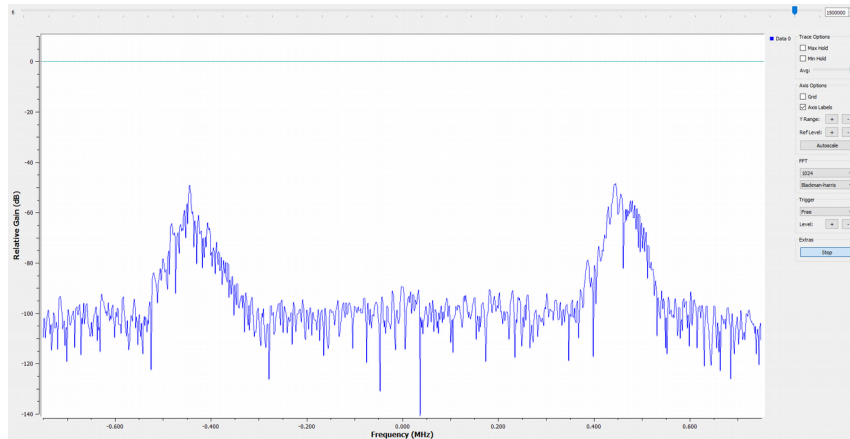


Figure 7 : Circular permutation

Q14) Avant établissement du filtre passe bas :

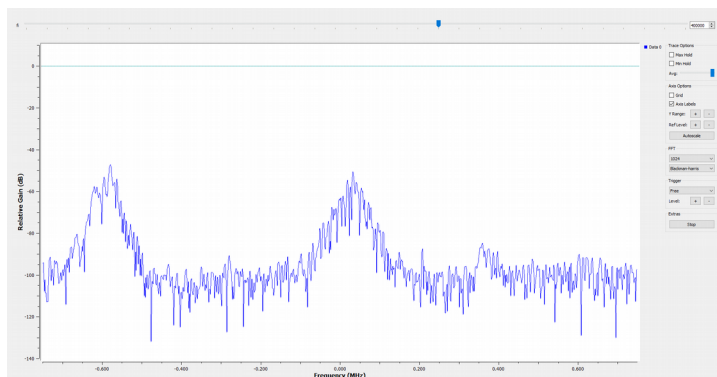


Illustration 2: Figure 8 : Offset -400kHz

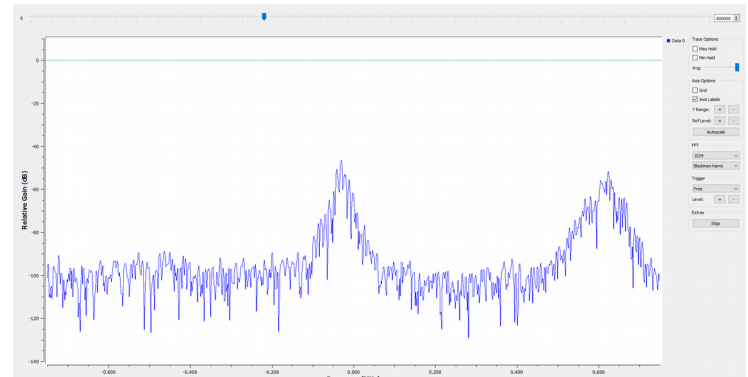


Illustration 1: Figure 9 : Offset 500kHz

Le bloc Signal Source génère une fréquence $-f_i$, avec f_i la fréquence d'offset réglée dans le bloc QT GUI Range.

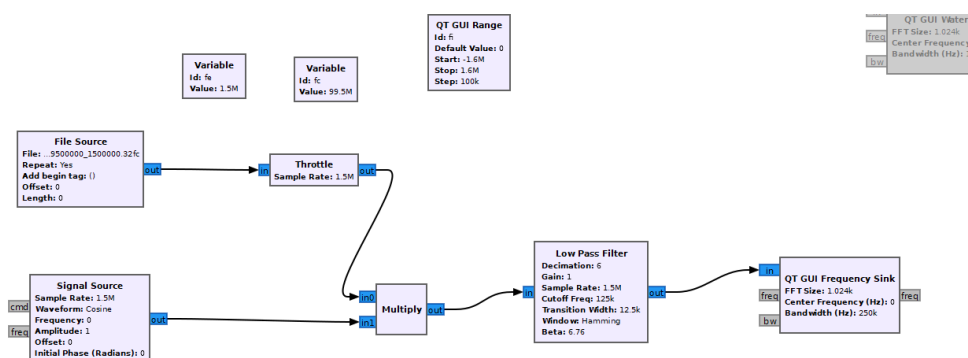


Figure 10 : Channel extraction by frequency transposition and low-pass filtering

On utilise donc un filtre passe bas de fréquence de coupure égale à 125kHz, fréquence d'échantillonnage égale à 1,5MHz, largeur de transition égale à 12,5kHz, et décimation égale à 6.

15) Pour récupérer le signal monophonique, il suffit de filtrer les hautes fréquences pour garder uniquement notre signal utile. Si on applique donc la règle de Carson :

$$B_{FM} = 2(\Delta f + f_m) = 2(75\text{kHz} + 53\text{kHz}) = 256\text{kHz}.$$

16) On module le signal $m(t)$ en fréquence de sorte que le signal radio centré sur f_0 en sortie de l'émetteur pour obtenir le signal $S_{RF}(t)$.

$$y_l[k] = Ae^{jk_f \sum_{i=0}^k m[i]} + b[k]$$

17)

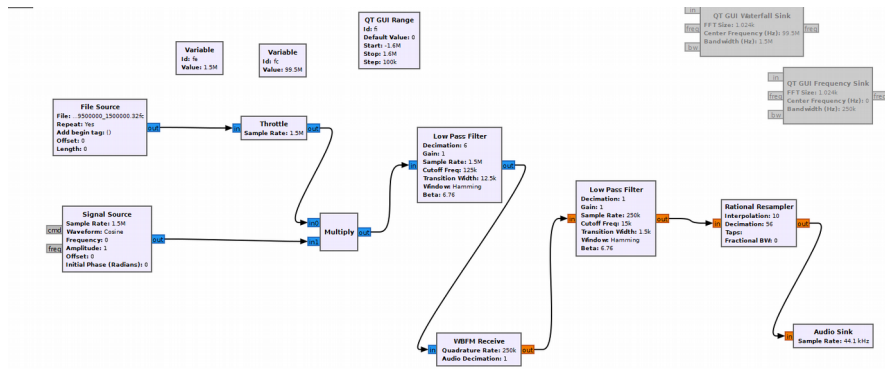


Figure 11 : Frequency demodulation and restitution

Le filtre en sortie en démodulateur est caractérisé par une fréquence d'échantillonnage égale à 250kHz et sa fréquence de coupure est 15kHz (signal monophonique), et largeur de transition égale à 1,5kHz.

18) On peut bien écouter la radio. Jordy a gagné l'album de Sam Smith.

IV – Implementation of the physical layer of a communicating object in BPSK to transmit a message (audio file)

On choisit donc ici d'implémenter sur GNU Radio la couche physique d'un objet communicant à l'aide de la modulation BPSK sur la transmission d'un message.

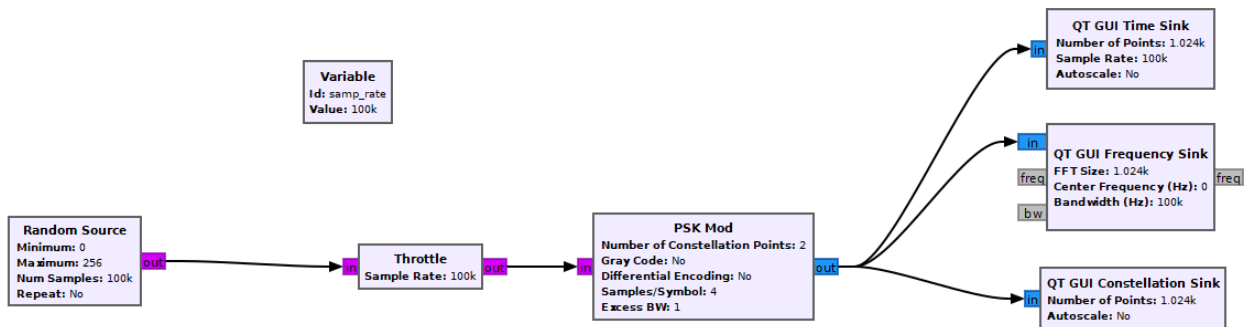


Figure 12 : Implementation of simple transmission chain managing BPSK modulation

On génère un flux d'octets de 100000 échantillons grâce au bloc Random Source qu'on fait passer dans un bloc Throttle (utilisation du processeur), flux qui sera les données en entrée de notre bloc PSK Mod.

Afin d'avoir une modulation BPSK (soit 2 états, et donc un symbole codé sur 1 bit), on règle le nombre de points de la constellation, puis on désactive le codage de Gray (inutile dans le cas d'une BPSK puisque 2 symboles codés chacun sur un bit) et le codage différentiel. Ensuite, on fixe un nombre de 4 échantillons par symbole et un facteur de retombée du filtre en racine de cosinus surélevé égal à 1.

Enfin, en sortie du bloc PSK Mod pour vérifier le bon fonctionnement de notre modulation, on fixe les blocs Time Sink (affichage temporel), Frequency Sink (affichage fréquentiel), et Constellation Sink (diagramme de constellation).

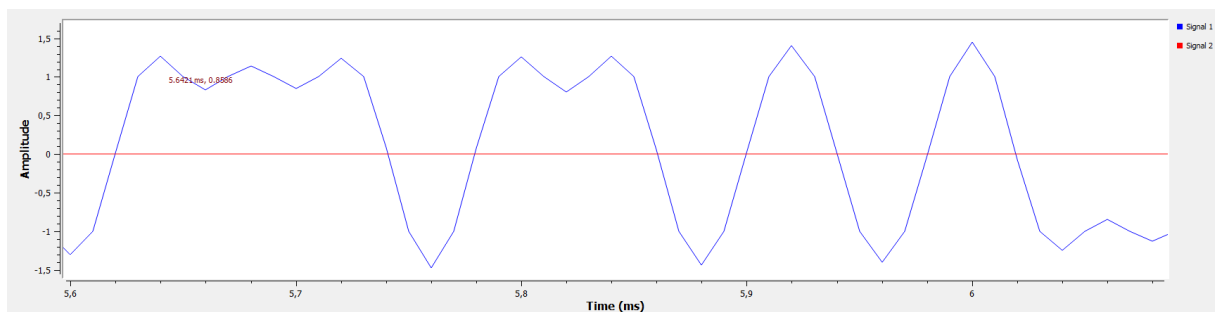


Figure 13 : Time display

Nous avons une fréquence d'échantillonnage fixée à $f_e = 100\text{kHz}$ et 4 échantillons par symbole. Ainsi, nous avons un rythme symbole $R_s = 25\text{kb/s}$ (égal également au rythme binaire puisque BPSK). Nous sommes donc censés obtenir un symbole/bit toutes les $40\mu\text{s}$, comme on peut le voir ci-dessus sur la figure 13.

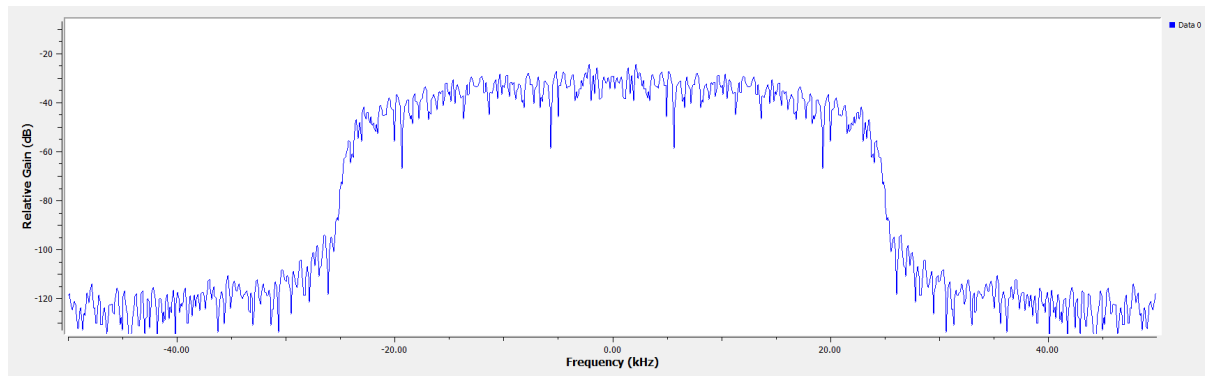


Figure 14 : Frequency display

Avec un rythme symbole égal à 25kBd et un facteur de retombée du filtre égal à 1, d'après la relation $BW = (1 + \text{facteur de retombée}) * R_s$, nous sommes censés obtenir une bande passante de 50kHz, comme on peut le voir ci-dessus sur la figure 14.

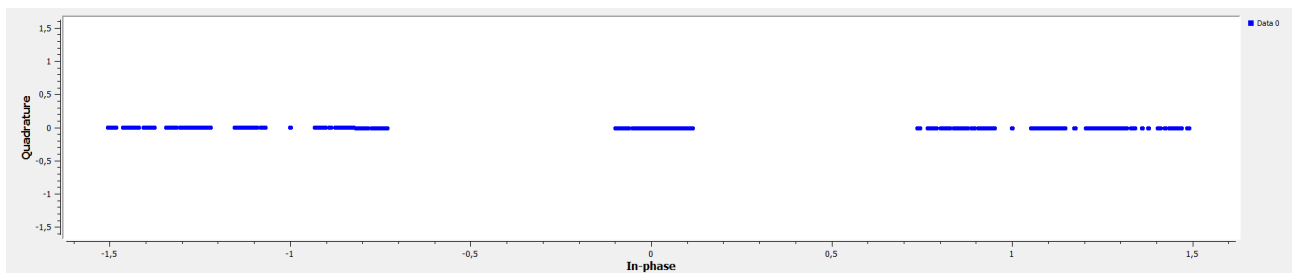


Figure 15 : Constellation diagram

On peut apercevoir sur le diagramme de constellation en sortie du modulateur BPSK (figure 15), tous les états possibles. Avec un démodulateur (sortie du filtre adapté), on se positionne à l'ouverture maximale de l'œil pour décider la valeur du bit.