Telecom Bretagne
655 Avenue du Technopole
29200 Plouzané

Promotion 2015 / 2018

Rapport de simulation d'un système de transmission – SIT 213

Auteurs	Marc LEMAUVIEL – Johann CORCUFF-REBECCHI – Arouna KANE – Lim Kévin CHAO – Lahoucine AMHOUCHE
Destinataire	Éric COUSIN, Bruno FRACASSO, Julien MALLET, François-Xavier SOCHELEAU
E-Mails	eric.cousin@telecom-bretagne.eu bruno.fracasso@telecom-bretagne.eu julien.mallet@telecom-bretagne.eu fx.socheleau@telecom-bretagne.eu
Formation suivie	Ingénieur spécialisé en Informatique, Réseaux et Télécommunications
Établissement	Télécom Bretagne
Promotion	2018

Date	Version	Modifié par	Motifs
13/09/2016	1	LEMAUVIEL Marc	Création du document
06/10/2016	2	LEMAUVIEL Marc	MAJ – Organisation travail d'équipe
17/10/2016	3	LEMAUVIEL Marc	MAJ – Etape 2
01/11/2016	4	LEMAUVIEL Marc	MAJ – Etape 3
07/11/2016	5	LEMAUVIEL Marc	MAJ – Etape 4
13/11/2016	6	LEMAUVIEL Marc	MAJ – Etape 5

Rapport de simulation d'un système de transmission – SIT 213

Étudiants : Marc LEMAUVIEL, Johann CORCUFF-REBECCHI, Arouna KANE, Lim Kévin CHAO, Lahoucine AMHOUCHE étudiants en Formation d'Ingénieur en Partenariat

Promotion : 2018

Établissement : Télécom Bretagne

Destinataires : Éric COUSIN, enseignant chercheur au département INFO

Bruno FRACASSO, enseignant chercheur au département OPTIQUE

Julien MALLET, enseignant chercheur au département INFO

François-Xavier SOCHELEAU, enseignant chercheur au département SIGNAL & COM

TABLE DES MATIERES

TABLE DES MATIERES....4

INTRODUCTION....5

ORGANISATION DU TRAVAIL D'EQUIPE....6

BE – TEST ET VALIDATION....7

ETAPE 1: TRANSMISSION « BACK-TO-BACK »....8

ETAPE 2 : TRANSMISSION ANALOGIQUE (NON BRUITEE)....10

ETAPE 3 : TRANSMISSION NON-IDEALE AVEC CANAL BRUITE DE TYPE « GAUSSIEN »....15

ETAPE 4 : TRANSMISSION NON-IDEALE AVEC DIVERS BRUITS « REELS » PLUS LE BRUIT GAUSSIEN DE L'ETAPE 3....22

ETAPE 5 : CODAGE DE CANAL AVEC CANAL BRUITE DE TYPE « GAUSSIEN »....27

CONCLUSION....31

ANNEXES....32

ANNEXE 1 32

INTRODUCTION

L'objectif de ce projet était de nous faire étudier une chaine de transmission (représentée par la figure 1).

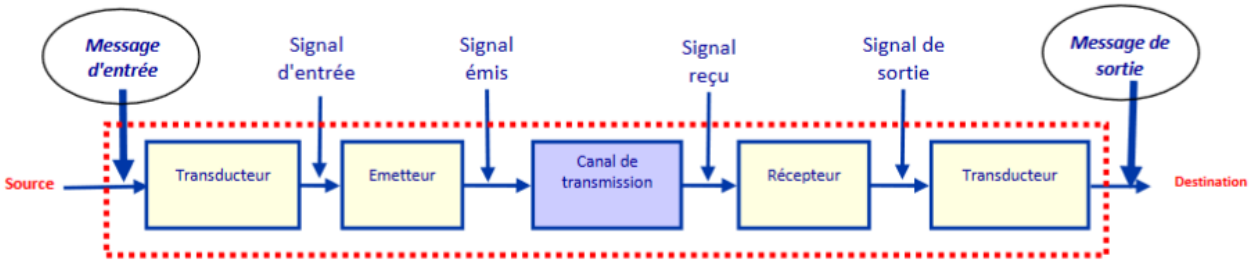

Figure 1 : Représentation d'une chaine de transmission

Pour effectuer cette étude, nous étions composés d'un groupe de 5 personnes. L'avancée du projet s'est faite au travers de 6 étapes qui nous ont permis de simuler pas à pas les différents blocs de la chaine de transmission.

L'étape 1 s'est déroulée entre le 13/09 et le 06/10. Elle consistait à réaliser une transmission élémentaire « back-to-back ».

L'étape 2 a été effectuée entre le 07/10 et le 17/10. Son but était d'effectuer une transmission non bruité d'un signal analogique.

L'étape 3 s'est faites entre le 18/10 et le 01/11. L'objectif était d'ajouter un bruit blanc gaussien au signal afin de simuler des perturbations qui peuvent survenir sur le canal de transmission.

L'étape 4 a été réalisée entre le 02/11 et le 07/11. Elle consistait à ajouter un phénomène de perturbation que nous retrouvons dans la nature : les trajets multiples qui perturbent le signal en réception.

L'étape 5 a été implémentée entre le 08/11 et le 13/11. Le but de cette étape était d'ajouter d'un codage canal pour introduire de la redondance et ainsi réduire le nombre d'erreurs.

ORGANISATION DU TRAVAIL D'EQUIPE

L'organisation a été un point essentiel dans ce projet. Nous étions un groupe de 5 et nous devions donc nous répartir les rôles pour que chacun soit efficace. Pour respecter au mieux les délais et les tâches nous avons réparti des responsabilités pour ce projet :

Chef de projet - Responsable livrables : Marc Lemauviel

Responsable développement : Johann Corcuff

Responsable télécom : Lahoucine Amhouche

Responsable validation : Arouna Kane

Responsable qualité : Lim Kévin Chao

L'objectif principal était que chacun d'entre nous acquière des compétences dans les différents domaines du projet (Info et Telecom). Pour cela nous avons régulièrement organisé des séances de travail en commun pour que chacun puisse comprendre les différentes parties du projet. Nous avons aussi réalisé des documentations (notamment de la javadoc) tout au long du projet pour les différentes étapes du projet.

Afin de garantir un bon niveau de qualité de notre travail et notamment pour notre code, nous y avons appliqué des tests pour nos itérations. La réalisation de ces derniers s'est faite au travers de JUnit qui est un outil simple à maitriser et qui nous a permis d'avancer rapidement sur nos tests. Nous avons également mis un point d'honneur sur la structure et la forme du code afin d'en faciliter la lecture autant que possible.

La rédaction du rapport a été réalisé en majorité par le chef de projet afin de garder la même ligne de conduite. Toutefois, pour garantir la qualité du rapport, chacun des acteurs a effectué une relecture du rapport pour y corriger des éléments et y apporter des améliorations.

Le travail ayant été réalisé en groupe, la mise en commun du code a été un point essentiel. Nous avons choisi d'utiliser un gestionnaire github pour mettre en commun nos différents travaux. Cela nous a ainsi permis de travailler en simultané sur des parties de code différentes.

BE – TEST ET VALIDATION

Les tests sont réalisés pour valider le contrat public des différentes classes et s'assurer du bon fonctionnement du programme au regard du cahier des charges conçu en amont du projet.

Les tests ont différents objectifs :

- Valider les fonctionnalités définies dans le cahier des charges établi par le client,
- Détecter de potentiels bugs et les corriger,
- S'assurer de la qualité du programme développé,

Quelles parties peut-on vraiment tester avec un programme de test ?

Les parties que l'on peut vraiment tester avec un programme de tests sont :

- Les contrats publics que sont censés remplir les différentes classes (tests unitaires),
- Le bon fonctionnement global du programme (tests d'intégrations).

Comment améliorer la testabilité ?

L'idéal est de coder les tests avant l'implémentation des différentes classes. Cela nécessite d'avoir une architecture où les contrats publics des différentes classes sont déjà définis. Dans la réalité, il se peut que l'architecture évolue ce qui peut amener à modifier les tests précédemment réalisés.

Comment faire du test de non régression à chaque nouvelle itération du projet ?

Pour cela il faut tester les parties du code les plus stables ainsi que les tests sur le fonctionnement général du programme qui ne sont pas censés changer au cours des itérations. Si du code mutable est mélangé avec du code stable il est bon de mettre une couche d'abstraction entre les deux. Pour cela on peut utiliser une interface, une classe abstraite ou des design patterns comme le pattern delegation.

Quelle confiance accorder aux résultats de simulation, et comment accroître cette confiance ?

La confiance accordée aux résultats de simulation dépend de la manière dont ils ont été réalisés. Si c'est le même développeur qui code à la fois les tests et la fonctionnalité correspondante, on pourra alors lui accorder une confiance moindre que si ces deux parties avaient été réalisées par deux personnes différentes (regards croisés). C'est pour cela que les entreprises possèdent des personnes spécialisées dans les tests et validation des programmes qui sont indépendantes des équipes de développement afin d'accroître cette confiance dans les résultats.

Comment comparer des simulations lorsqu'elles reposent sur des comportements aléatoires (message, bruit, …) ?

Pour tester les simulations malgré des comportements aléatoires, on peut utiliser des outils statistiques permettant de mesurer un degré de corrélation suffisant entre une entrée et la sortie attendue (calcule du Taux d'Erreur Binaire, TEB). La prédétermination théorique du TEB (en utilisant MATLAB par exemple) et sa comparaison avec le TEB calculé dans le programme est un bon moyen de vérifier la véracité du code réalisé.

ETAPE 1: TRANSMISSION « BACK-TO-BACK »

L'application réalisée pour l'étape 1 correspond à la figure 2 :

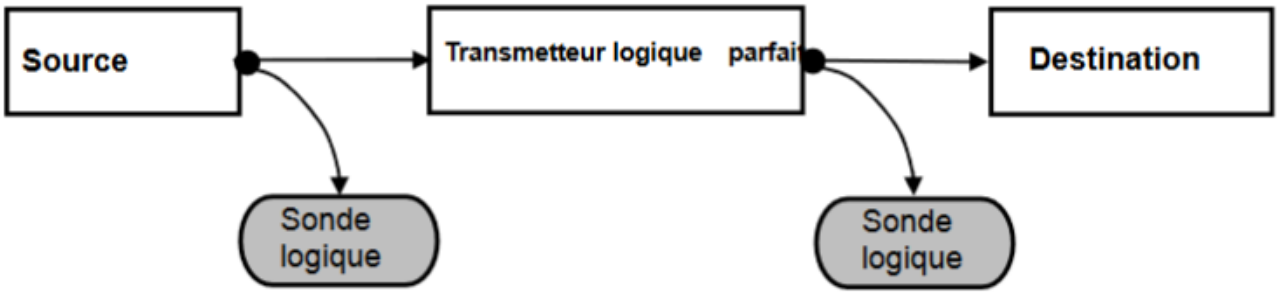

Figure 2 : Modélisation de la chaine de tranmission à l'étape 1

La source émet une séquence booléenne fixée ou aléatoire. Le transmetteur logique parfait se contente, à la réception d'un signal, de l'émettre tel quel vers les destinations qui lui sont connectées. La destination se contente de recevoir le signal du composant sur lequel elle est connectée. Des sondes logiques permettent de visualiser les signaux émis par la source et le transmetteur parfait. L'application principale calcule le taux d'erreur binaire (TEB) du système.

Au cours de cette étape, nous avons utilisé le code fourni pour réaliser l'application. Nous avons ainsi lié entre elles les différentes classes :

- Source,
- Transmetteur,
- Destination,
- Sonde.

Une fois ces éléments liés, nous avons implémenté les éléments de la commande unique :

- - mess m (qui génère un message aléatoire de taille m si m représente un entier décimal de longueur inférieur à 6 ou un message binaire fixé par l'utilisateur si la taille de m est supérieur à 7).
- - s (permet d'ajouter des sondes pour visualiser le message au niveau de la source et du transmetteur logique parfait, puis au niveau de l'émetteur et du récepteur à partir de l'étape 2).
- -seed v (qui permet de répéter le signal aléatoire à l'identique si la semence v est identique entre les 2 simulations).

La figure 3 illustre la réalisation d'une simulation avec les paramètres indiqués en légende.

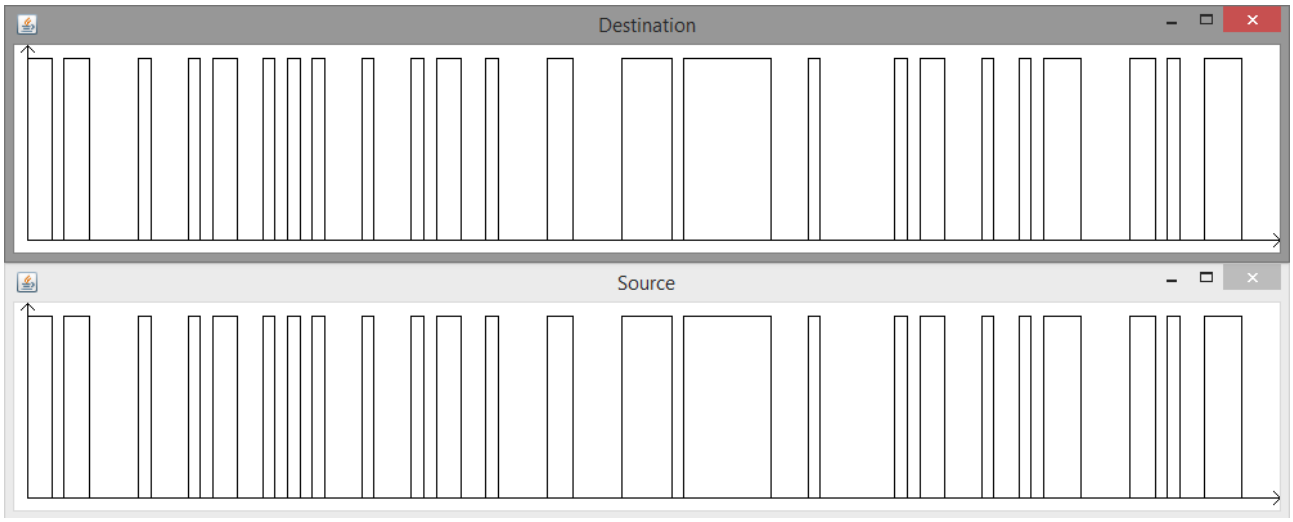

Figure 3 : Résultat de la simulation : java Simulateur -s => TEB : 0.0

Le paramètre -mess n'est pas renseigné, le simulateur va donc générer et transmettre un signal de longueur 100. En relançant la même simulation, nous obtenons un résultat différent puisque nous n'avons pas utilisé le paramètre -seed pour avoir le même signal.

Etant donné que nous utilisons un transmetteur logique parfait, nous obtenons un TEB nul. Il n'y a aucune perturbation pour le moment qui s'applique sur le signal. Le signal est donc reçu à l'identique par rapport au signal généré.

L'étape 1 nous a ainsi permis de construire la première brique de la chaine de transmission. L'étape 2 nous permettra ensuite de passer d'une transmission logique à une transmission analogique grâce à l'utilisation d'un émetteur et d'un récepteur.

ETAPE 2 : TRANSMISSION ANALOGIQUE (NON BRUITEE)

La seconde étape du projet reprend le travail effectué lors de l'étape 1. Cependant nous y ajoutons un émetteur, un récepteur et changeons le canal de transmission (analogique) comme indiqué dans la figure 4 :

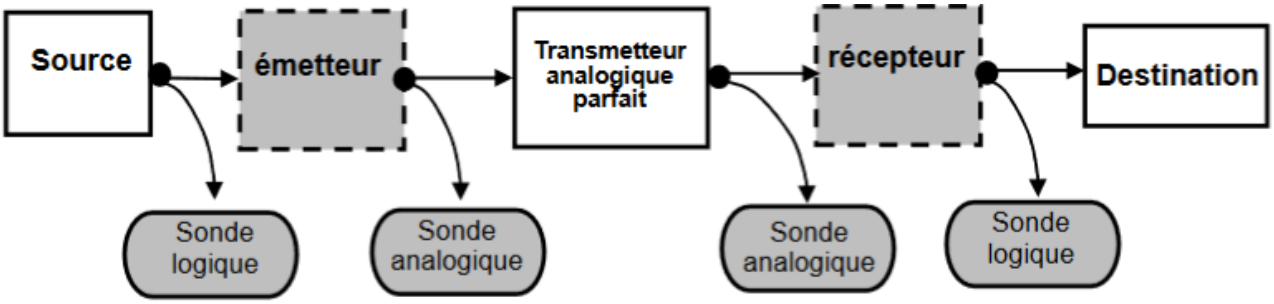

Figure 4 : Modélisation de la chaine de transmission à l'étape 2

L'ajout de l'émetteur et du récepteur permet de réaliser la conversion « logique => analogique » puis « analogique => logique ». Grâce à cela, le signal peut être émis au travers d'un canal analogique (dans notre cas, c'est le transmetteur analogique parfait).

Cela correspond à ce qu'il se passe dans la nature. Nous disposons d'un signal numérique (composé de 0 et de 1 logiques) que nous souhaitons transmettre au travers une chaine de transmission. Pour ce faire, nous convertissons ce message logique en un signal analogique grâce à l'émetteur. L'intérêt de cette pratique est de pouvoir adapter le signal au canal de transmission (hertzien, guidé optique, guidé électrique, …). Au bout du canal de transmission, le récepteur reçoit le signal analogique puis le converti en signal numérique (le but étant que le message en sortie soit identique à celui en entrée).

La transmission de l'information se fait au travers de l'utilisation de l'un de ces 3 signaux dont les caractéristiques sont décrites ci-après :

- NRZ : forme d'onde rectangulaire, cf figure 5,
- NRZT : forme d'onde trapézoïdale (temps de montée ou de descente à 1/3 du temps bit), cf figure 6,
- RZ : forme d'onde impulsionnelle (amplitude min sur le premier et dernier tiers du temps bit, impulsionnelle sur le tiers central avec un max au milieu du temps bit égal à l'amplitude max), cf figure 7.

Nous avons donc utilisé ces caractéristiques pour coder notre émetteur. Notre simulateur récupère les informations nécessaires à la génération du signal (nombre d'échantillons, amplitudes min et max, et la forme du signal désirée) puis génère le signal en fonction de ces informations. Pour NRZT, le front montant est réalisé grâce à une droite linéaire. Pour le signal RZ, c'est une sinusoïde qui est utilisé pour générer le front montant.

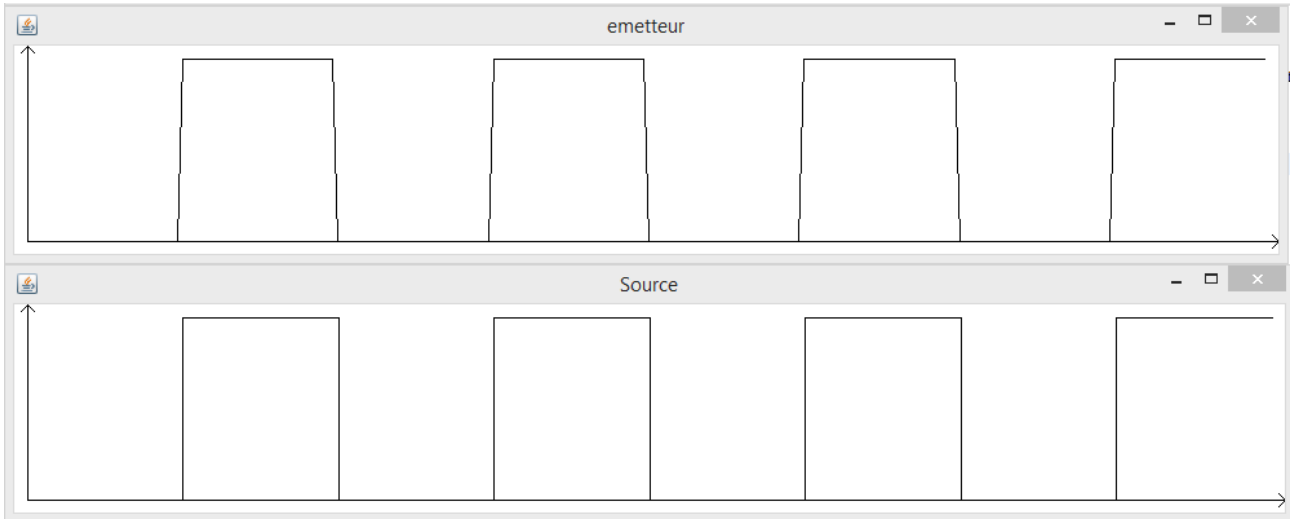

Figure 5 : Simulation pour le signal NRZ

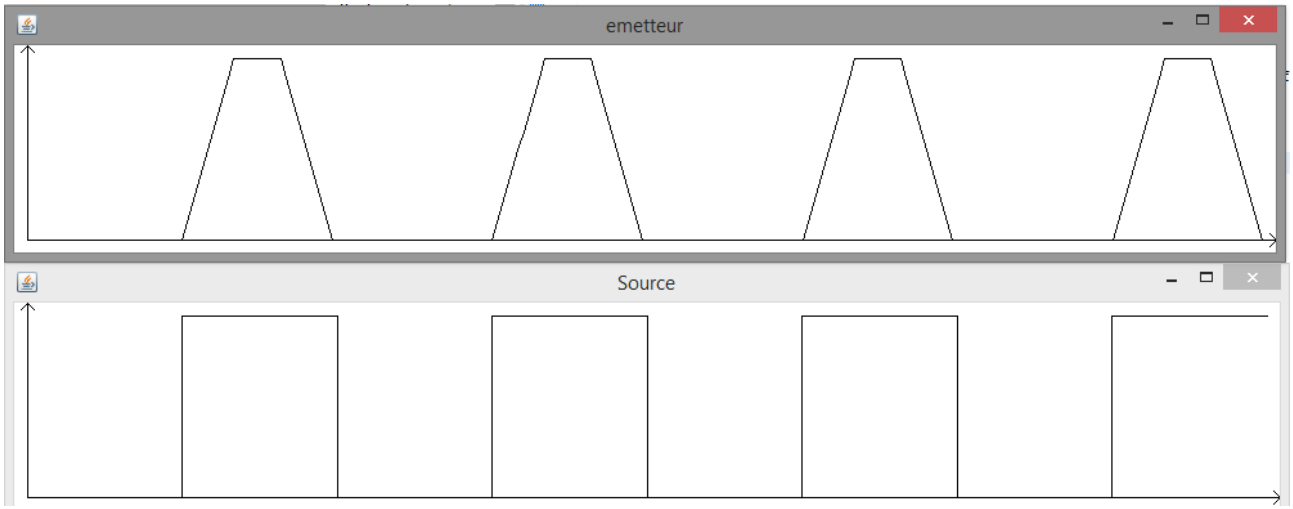

Figure 6 : Simulation pour le signal NRZT

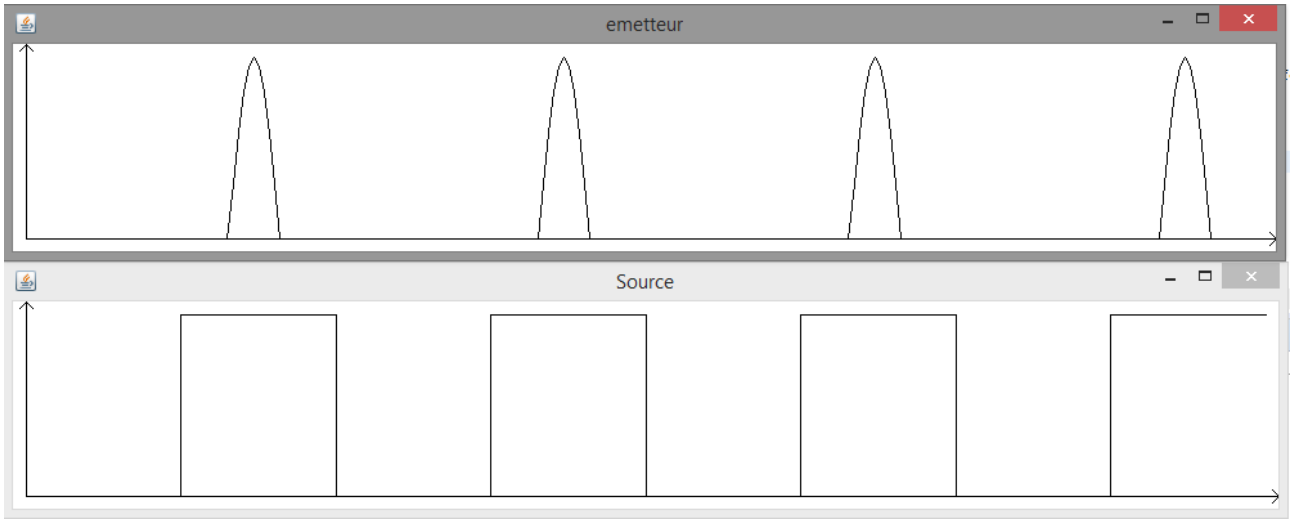

Figure 7 : Simulation pour le signal RZ

D'un point de vue implémentation, nous sommes tout d'abord partis sur une première architecture où nous utilisions l'héritage afin de spécialiser les émetteurs et les récepteurs selon le type de signal transmis (NRZ, NRZT ou RZ). Cela nous a amené à avoir un grand nombre de classes et une relative redondance du code. C'est pourquoi nous avons opté pour une seconde architecture utilisant le pattern delegation. Nous avons ainsi créé le package « codeur » qui contient des encodeurs et des décodeurs pour chacun des types de signaux. Il suffisait ensuite de passer en argument l'encodeur (respectivement le décodeur) désiré à l'émetteur (au récepteur). Ainsi nous n'avions plus qu'une seule classe émetteur (et récepteur) ce qui nous a permis de factoriser le code. Néanmoins, nous avons dû modifier nos tests unitaires afin de les adapter à cette nouvelle architecture.

Pour réaliser le récepteur et la partie reconnaissance du symbole reçu, nous avons effectué une opération de comparaison entre un seuil et la moyenne des échantillons pour un même temps bit.

Le seuil est déterminé en réalisant l'opération suivante : $Seuil = \frac{Moyenne\ de\ toutes\ les\ valeurs\ du\ signal}{Nombre\ d'échantillon\ total}$

Nous calculons ensuite la moyenne des échantillons pour un temps bit en suivant les principes suivants :

- Signal NRZ : moyenne totale des échantillons pour un temps bit (30 échantillons par défaut),
- Signal NRZT : moyenne des échantillons correspondant au tiers du temps bit central (les 10 échantillons centraux par défaut),
- Signal RZ : moyenne des échantillons correspondant au tiers du temps bit central (les 10 échantillons centraux par défaut).

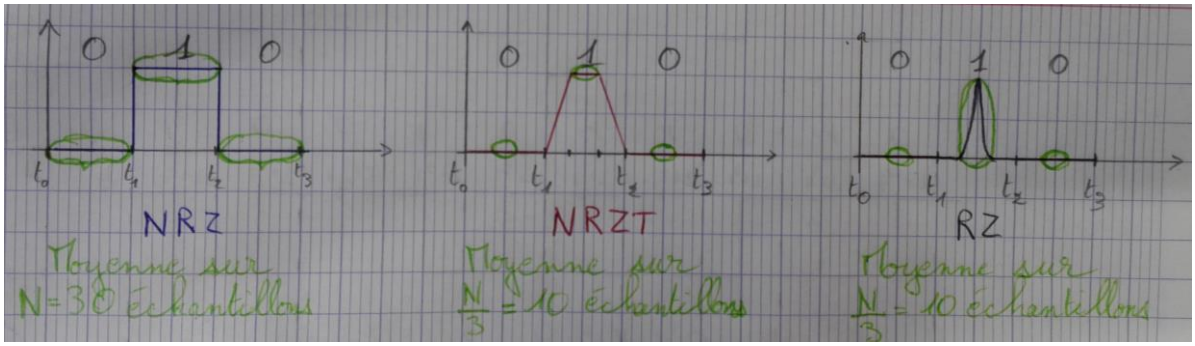

Figure 8 : Illustration de la méthode de moyennage pour la réception des signaux

Enfin, nous comparons la moyenne calculée sur le temps bit avec le seuil pour déterminer la nature du symbole reçu.

Au niveau de la validation, nous avons voulu tester si les signaux produits par les encodeurs à partir d'une suite binaire connue correspondaient bien aux exigences du cahier des charges. Nous avons donc analysé si le signal en sortie de l'encodeur puis du décodeur correspondait bien aux attentes.

Voici les différents résultats que nous avons obtenus :

Figure 9 : Résultat de la simulation : java Simulateur -s -form NRZ => TEB : 0.0

La forme d'onde qui résulte du signal NRZ correspond à une forme rectangulaire générée par une impulsion.

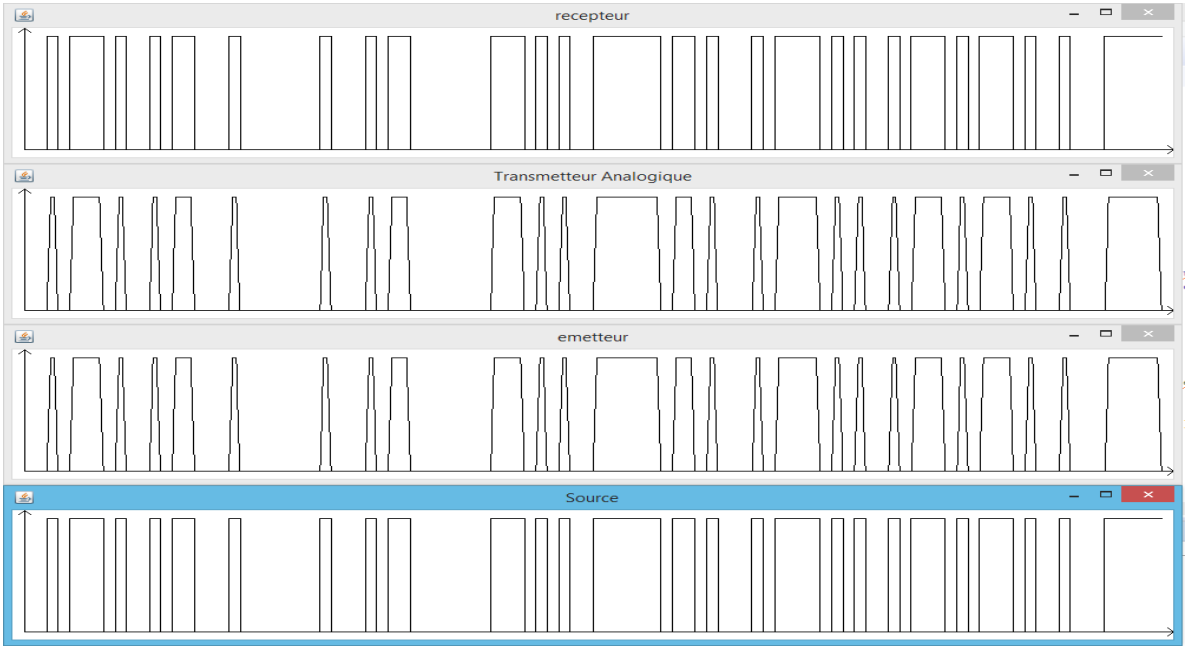

Figure 10 : Résultat de la simulation : java Simulateur -s -form NRZT => TEB : 0.0

La forme d'onde du signal NRZT est similaire à celle du signal NRZ mais cette fois ce ne sont plus des rectangles mais des trapèzes qui sont observés. Les temps de montée et de descente correspondent à 1/3 du temps bit ce qui donne cette forme trapézoïdale.

Figure 11 : Résultat de la simulation : java Simulateur -s -form RZ => TEB : 0.0

La forme d'onde du signal RZ correspond à une impulsion sur le tiers central du temps bit (le premier et le dernier tiers étant à la valeur min) avec un max atteint à la moitié du temps bit.

Les différentes formes d'ondes correspondent donc bien aux attentes exigées dans le cahier des charges.

Tout comme pour l'étape 1, nous obtenons un TEB de 0 ce qui correspond à une transmission parfaite. En effet, la chaine de transmission utilise un canal parfait pour le moment. Lors de l'étape 3, nous modifierons le canal de transmission pour ajouter un bruit blanc additif gaussien ce qui entrainera des variations du TEB.

ETAPE 3 : TRANSMISSION NON-IDEALE AVEC CANAL BRUITE DE TYPE « GAUSSIEN »

La troisième étape du projet reprend le travail effectué lors des deux précédentes étapes. Cependant nous modifions le transmetteur analogique parfait pour y introduire un bruit blanc additif gaussien. La figure 14 représente la nouvelle modélisation de la chaine de transmission :

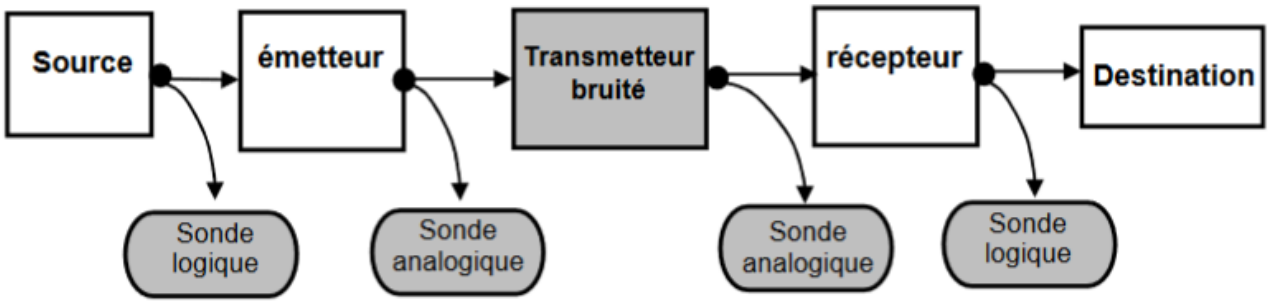

Figure 12 : Modélisation de la chaine de transmission à l'étape 3

Un bruit blanc correspond à la réalisation d'un processus aléatoire dans lequel la densité de puissance est la même pour toutes les fréquences de la bande passante. Le bruit blanc gaussien correspond à un bruit blanc qui suit une loi normale de moyenne et de variance données.

On utilise ce bruit blanc pour modéliser un canal de transmission en télécom. En effet tous les canaux de transmissions ajoutent du bruit au signal émis. Ainsi lorsque l'on reçoit le signal, du bruit s'est ajouté au signal émis. Le bruit blanc n'est qu'un type de perturbation qui se passe dans la nature, nous en verrons d'autres dans les prochaines étapes.

La figure 13 illustre la modélisation du canal de transmission auquel nous ajoutons un bruit blanc gaussien. A la réception, nous obtenons donc le signal émis s(n) plus le bruit b(n) avec b(n) qui suit une loi normale de moyenne nulle et de variance σ_b^2.

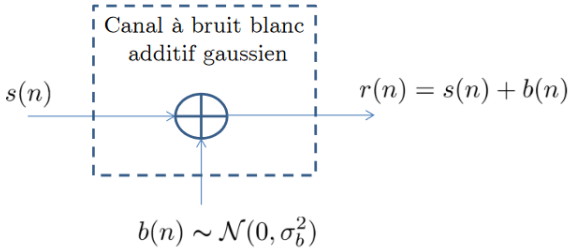

Figure 13 : Modélisation du canal de transmission avec un bruit blanc additif gaussien

La première étape de l'implémentation a donc consisté à réaliser ce bruit blanc gaussien. Pour cela nous avons utilisé la formule suivante :

$$b(n) = \sigma_b \sqrt{-2\ln(1 - a_1(n))}\cos(2\pi a_2(n)) \qquad \begin{matrix} a_1(n) \sim \mathcal{U}[0,1[\text{ (loi uniforme)} \\ a_2(n) \sim \mathcal{U}[0,1[\end{matrix}$$

Figure 14 : Formule pour générer un bruit blanc gaussien

Afin de vérifier que nous générions bien un bruit blanc gaussien, nous avons utilisé Matlab pour réaliser un histogramme et le comparer à une Normale de même paramètre : $N(0, \sigma_b^2)$. Voici la courbe que nous avons obtenue :

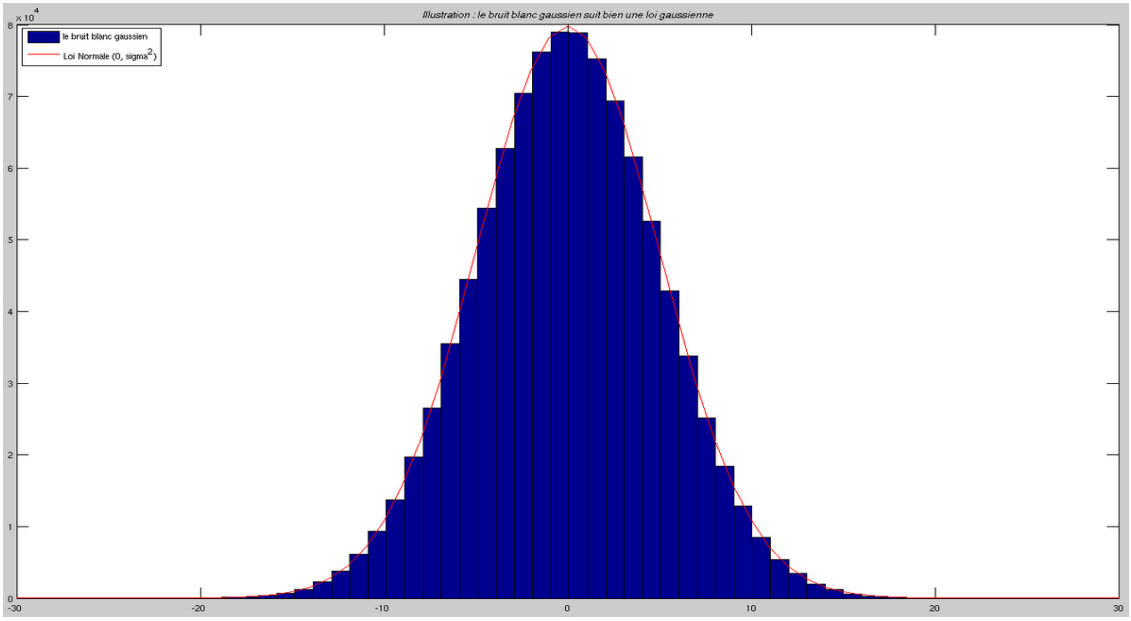

Figure 15 : comparaison du bruit blanc généré par rapport à la loi Normale $N(0, \sigma_b^2)$

Suite à l'implémentation du bruit, nous avons ajouté ce bruit à l'information émise. Lors de la première réalisation de l'étape nous avions eu un problème par rapport au bruit. Ce dernier était trop important par rapport au SNR rentré. Le calcule n'était pas bon, il manquait un rapport 10 entre les deux. Suite à la modification, les résultats sont désormais plus cohérents. Les figures suivantes illustrent l'impact d'un bruit dix fois inférieur à la puissance du signal sur la forme d'onde pour les différents signaux :

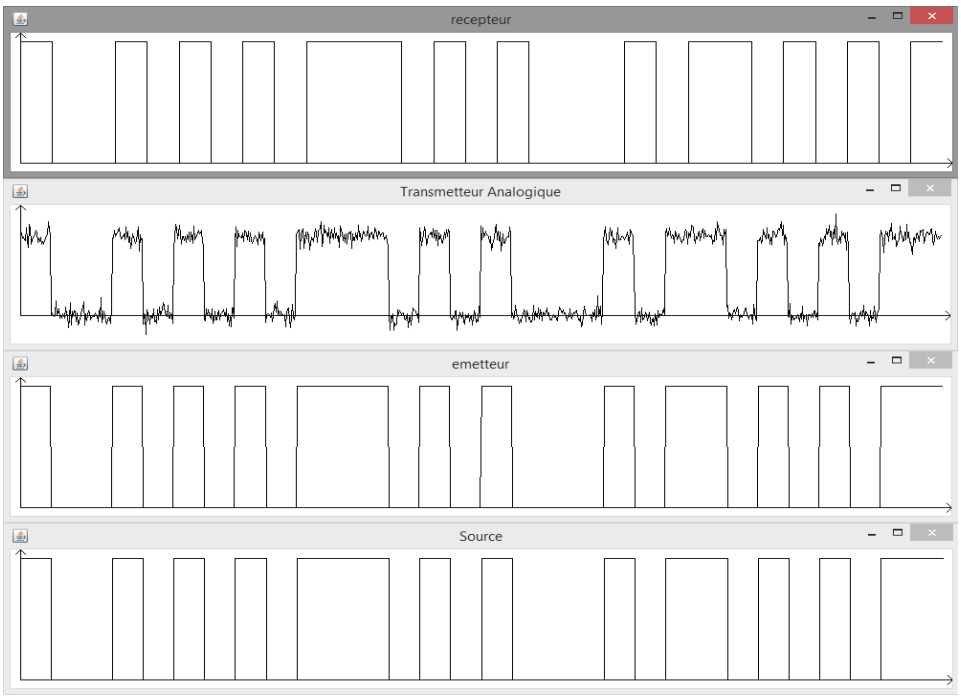

Figure 16 : Résultat de la simulation : java Simulateur -s -mess 1001010101110101000101101010 11 -form NRZ -snr 10 => TEB : 0.0

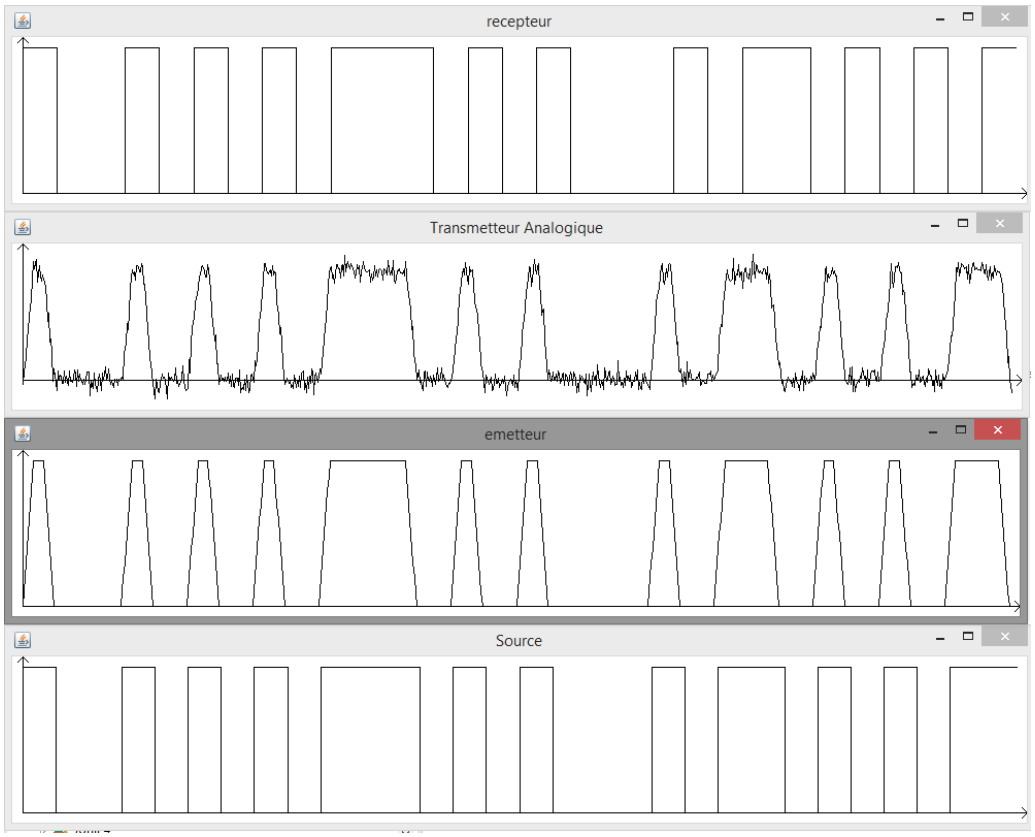

Figure 17 : Résultat de la simulation : java Simulateur -s -mess 1001010101110101000101101010 11 -form NRZT -snr 10 => TEB : 0.0

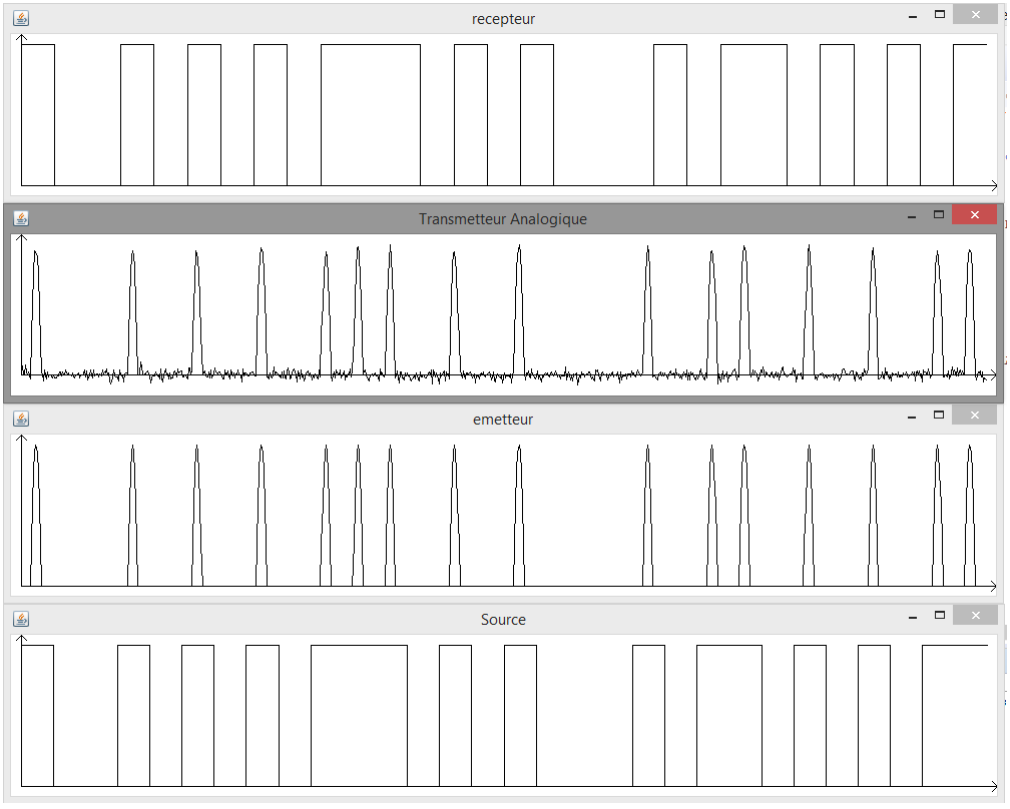

Figure 18 : Résultat de la simulation : java Simulateur -s -mess 1001010101110101000101101010 11 -form RZ -snr 10 => TEB : 0.0

On constate sur la figure 18 que le bruit impact moins la forme du signal RZ par rapport au signal NRZ et NRZT. L'intérêt du signal RZ est qu'il change plus souvent d'état par rapport aux deux autres signaux. Il est donc plus facile de se synchroniser avec un signal RZ qu'avec les signaux NRZ et NRZT qui ne varient pas tous les tiers de temps bit.

Afin de mieux visualiser l'évolution du TEB, nous avons tracé la courbe du TEB en fonction du SNR par bit (Eb/N0). Pour ce faire, nous avons utilisé la formule suivante :

$$\left(\frac{Eb}{N0}\right)(dB) = 10\log\left(\frac{Ps * N}{2\sigma b^2}\right) = 10\log(\frac{Ps}{\sigma b^2}) + 10\log(\frac{N}{2}) = SNR\ (dB) + 10\log(\frac{N}{2})$$

Avec :

- Eb : énergie par bit (Joule: Watt. sec)
- N0 : densité spectrale du bruit (Watt/Hz)
- P_s : puissance du signal
- σ_b^2 : puissance bruit
- N : Nombre d'échantillon par bit

Pour obtenir cette courbe, nous effectuons plusieurs simulations en faisant varier le SNR entre -20 et +15 dB par pas de 0,1. De plus, nous effectuons 100 fois chaque simulation avec le même SNR pour obtenir une valeur moyenne de TEB plus précise. Nous avons réalisé cette simulation avec le signal NRZT pour une longueur de message de 1 000 bits. Pour des problèmes de performances sur les machines linux et l'école, nous n'avons pas essayé de monter au-delà de 1 000 bits.

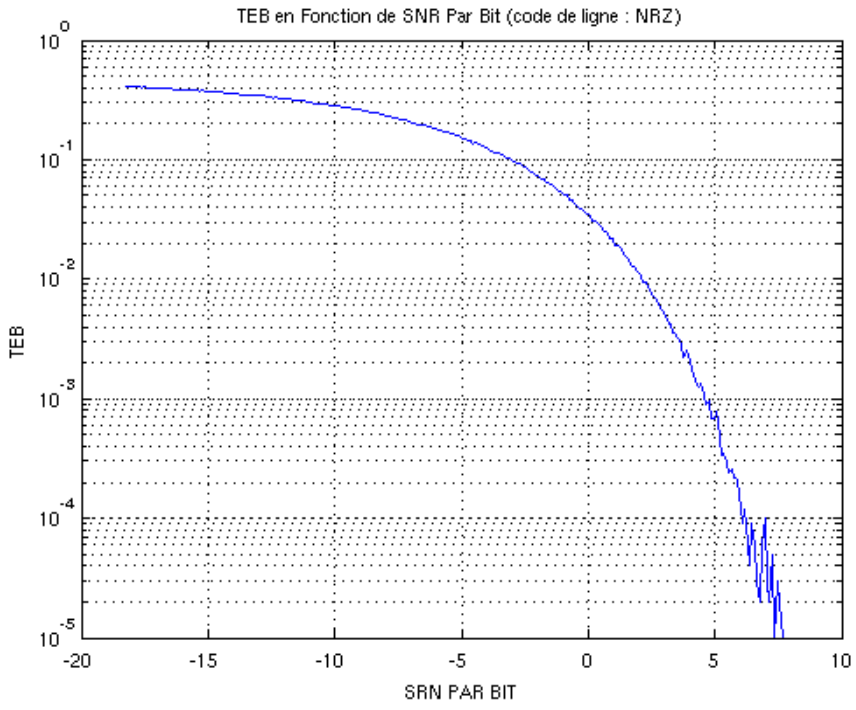

Figure 19 : Evolution du TEB en fonction du SNR pour le signal NRZ

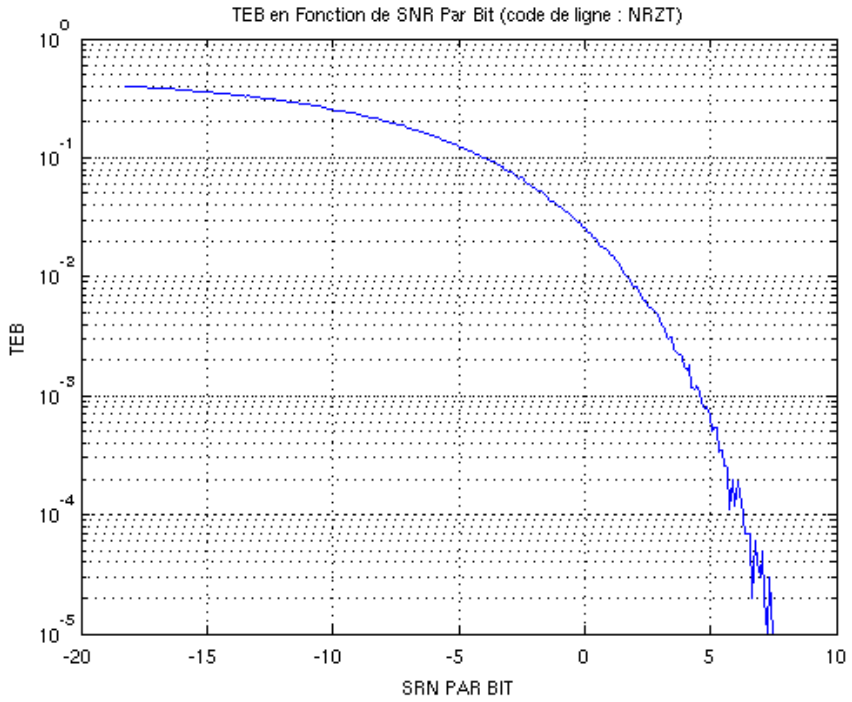

Figure 20 : Evolution du TEB en fonction du SNR pour le signal NRZT

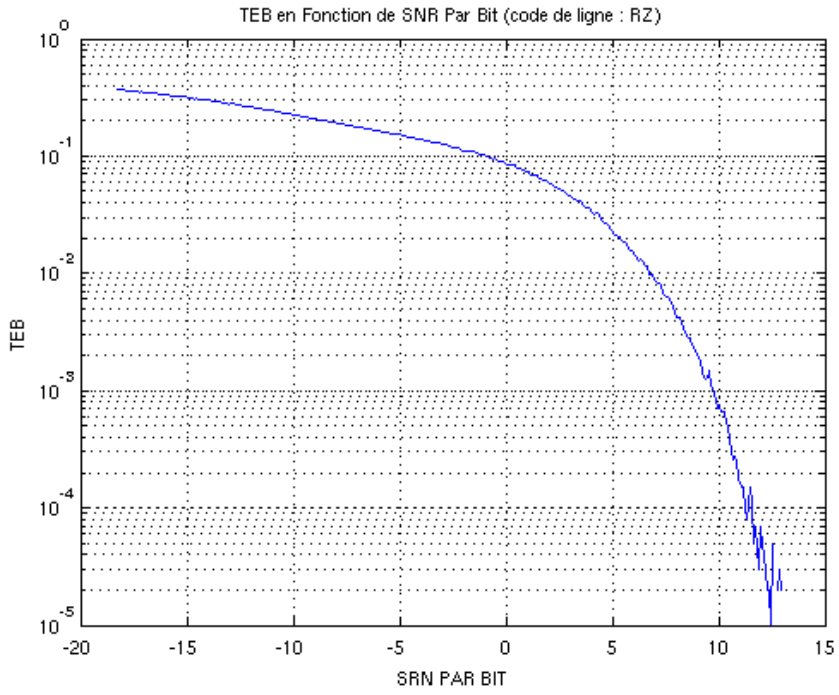

Figure 21 : Evolution du TEB en fonction du SNR pour le signal RZ

Nous constatons que c'est le signal RZ qui est le moins performant au niveau du TEB. L'évolution du TEB est moins importante pour le signal RZ qui est plus impacté par le bruit que les deux autres signaux. Cela provient du calcul de la moyenne que nous effectuons sur les 10 échantillons centraux du temps bit. Une amélioration pourra consister à réduire le nombre d'échantillons pris en compte pour n'en garder que 4 ou 5 qui seront plus proches du max à la moitié du temps bit.

Une autre façon de constater l'impact du bruit est d'observer les diagrammes de l'œil des trois signaux. Ces derniers nous renseignent sur la qualité de la transmission. Plus le diagramme de l'œil est ouvert, meilleure sera la transmission.

Les figures suivantes montrent l'impact du bruit sur les diagrammes de l'œil pour les signaux NRZ et NRZT (la représentation du diagramme de l'œil du signal RZ n'est pas correctement fonctionnel).

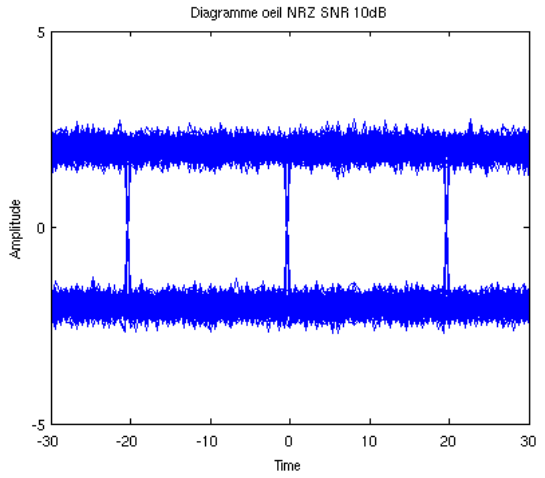

Figure 22 : Diagramme de l'oeil pour le signal NRZ avec -mess 1000 -snr 10

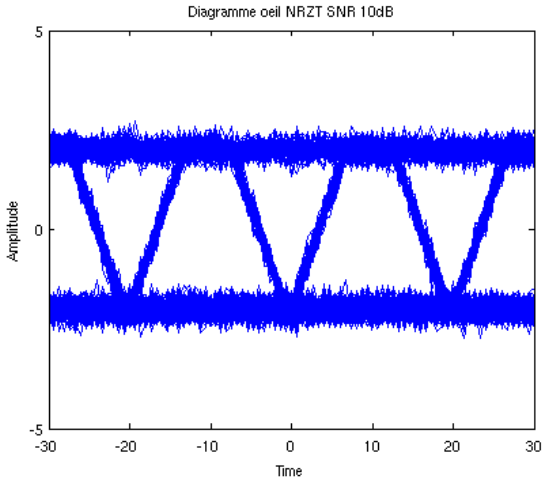

Figure 23 : Diagramme de l'oeil pour le signal NRZT avec -mess 1000 -snr 10

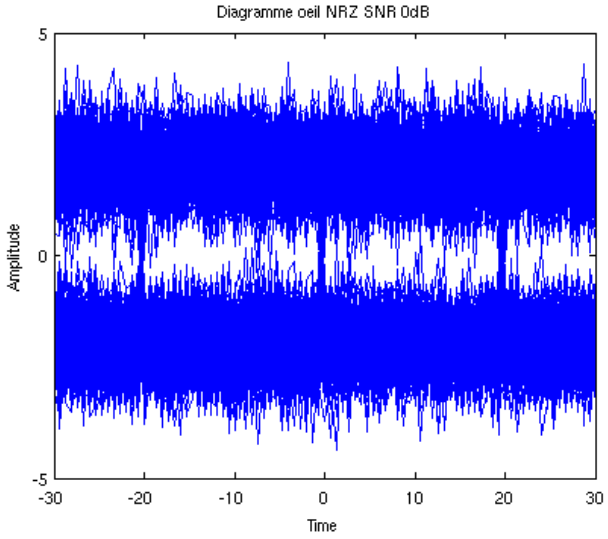

Figure 24 : Diagramme de l'oeil pour le signal NRZ avec -mess 1000 -snr 0

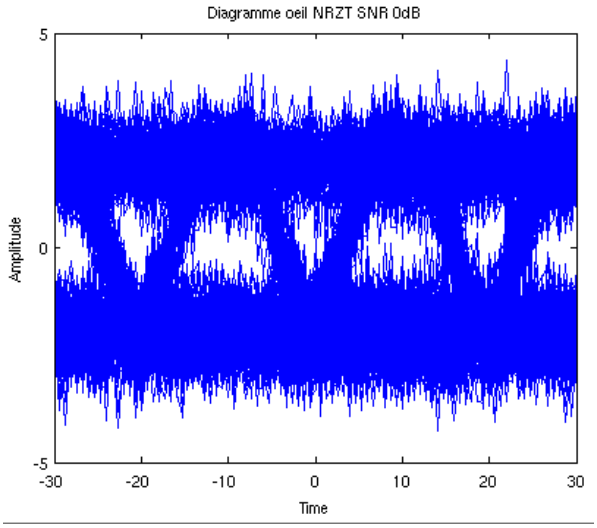

Figure 25 : Diagramme de l'oeil pour le signal NRZT avec -mess 1000 -snr 0

On constate que lorsque la puissance du signal est dix fois supérieure à celle du bruit, le diagramme de l'œil est très ouvert pour les deux signaux. Lorsque la puissance du bruit est équivalente à la puissance du signal, le diagramme de l'œil est beaucoup plus fermé. Le bruit joue principalement sur l'ouverture verticale de l'œil.

Lors de l'étape 2 nous avions rencontré un problème de performance. Afin de le résoudre, nous avons remplacé les linkedlist dans la classe information par des arraylist qui sont plus performantes. Nous avons ainsi réduit le temps d'exécution d'une trentaine de secondes pour transmettre un message de 10 000 bits (avec l'affichage) à seulement 2 ou 3 secondes après la modification.

Le schéma de la figure 26 résume les différentes étapes de fonctionnement que nous avons mis en place pour réaliser l'étape 3 :

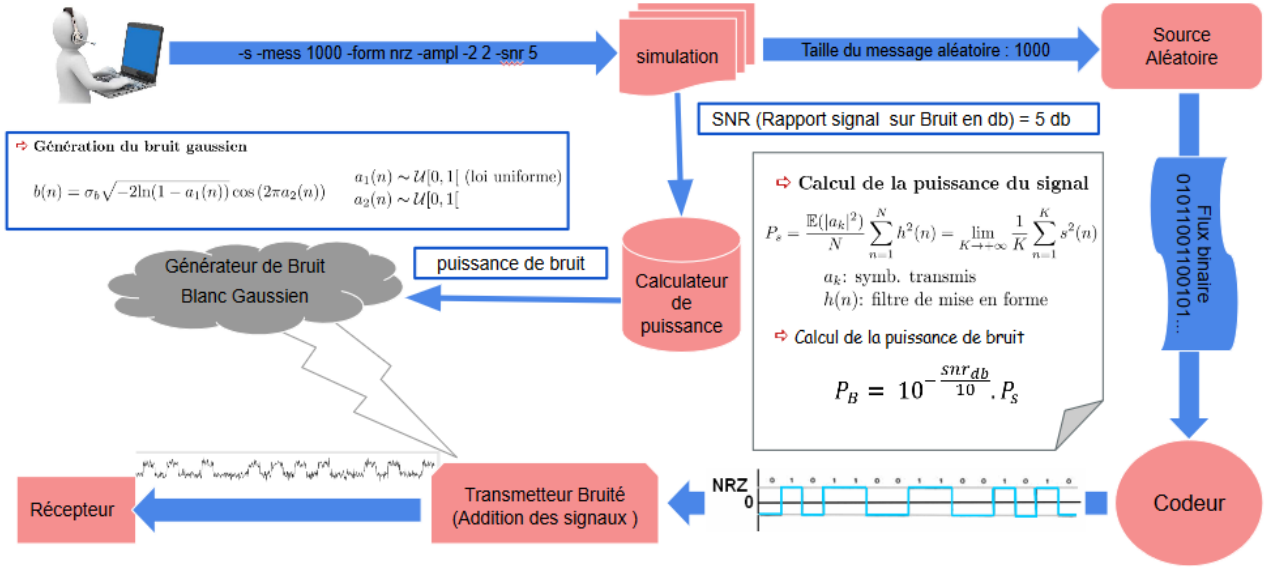

Figure 26 : Schéma de fonctionnement de l'étape 3

Les résultats obtenus à l'étape 3 sont concluants, le calcul du TEB semble correct et son évolution en fonction du SNR correspond à nos attentes. Lors de l'étape suivante, nous ajouterons, en plus du bruit blanc gaussien, un type de perturbation réel à savoir : des trajets-multiples.

ETAPE 4 : TRANSMISSION NON-IDEALE AVEC DIVERS BRUITS « REELS » PLUS LE BRUIT GAUSSIEN DE L'ETAPE 3

L'objectif de l'étape 4 est d'appliquer des modèles physiques sur notre chaine de transmission. Nous allons donc ajouter un phénomène de trajets-multiples tout en ajoutant le bruit blanc gaussien que nous avons réalisé à l'étape précédente. La figure 27 illustre les modifications que nous allons effectuer.

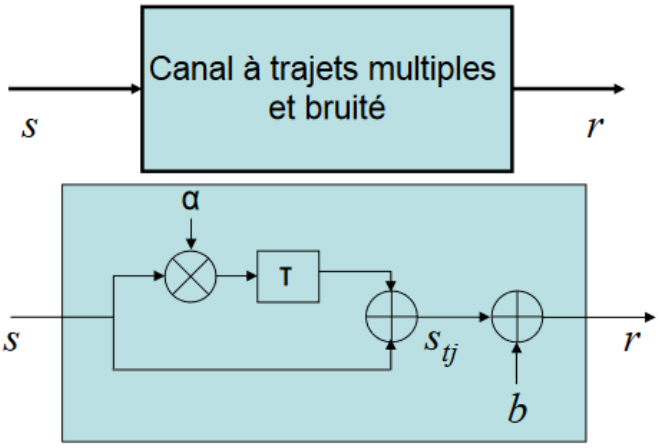

Figure 27 : Ajout d'un phénomène de trajets-multiples et du bruit blanc gaussien

La réalisation du trajet multiple consiste à reprendre le signal émis auquel nous ajoutons un retard T (en nombre d'échantillon) et une atténuation α (comprise entre 0 et 1). Voici un exemple d'un trajet multiple :

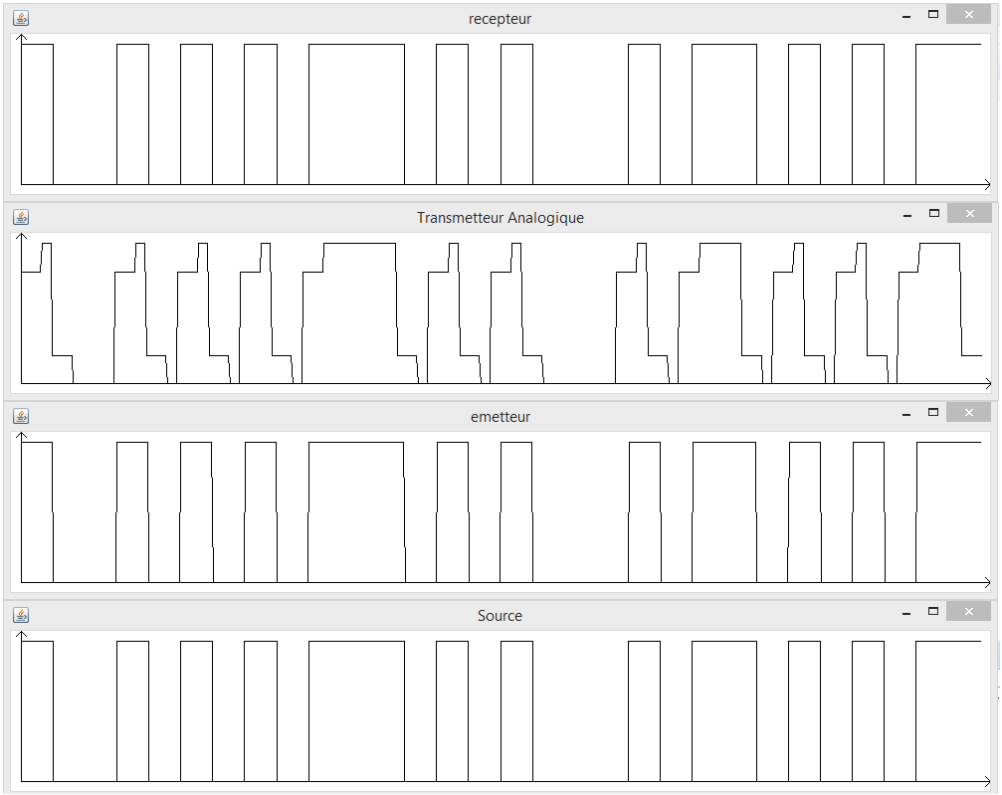

Figure 28 : Illustration d'un trajet multiple pour le signal NRZ avec α = 0,5 et T = 20

Comme nous pouvons le constater, le retard provoqué pour le trajet multiple fausse le signal reçu par le récepteur. Il faut donc réussir à retrouver l'information utile du signal pour pouvoir reconstruire le signal original.

Nous avons tout d'abord tenté d'implémenter un code matlab pour simuler un filtre adapté. Cependant cette méthode n'est pas complètement fonctionnelle, il existe certains cas particuliers pour lesquels cette méthode ne donne pas les bons résultats. L'annexe 1 montre la méthode du filtre adapté dans un cas fonctionnel.

Pour remplacer cette solution, nous avons gardé notre récepteur dont le seuil de décision s'adapte parfaitement à l'atténuation du trajet retardé. Toutefois lorsque le retard est équivalent à un temps bit (30 échantillons par défaut), notre récepteur n'est plus capable de faire la différence entre le signal utile et le signal retardé si la puissance du trajet multiple est équivalente à la puissance du trajet direct (ceci étant un phénomène impossible à retrouver dans la nature, il y aura forcément un peu d'atténuation sur le second trajet).

Les figures suivantes montrent l'impact des trajets multiples sur la forme du signal reçu avec l'ajout d'un peu de bruit. Le message transmis sera toujours le même.

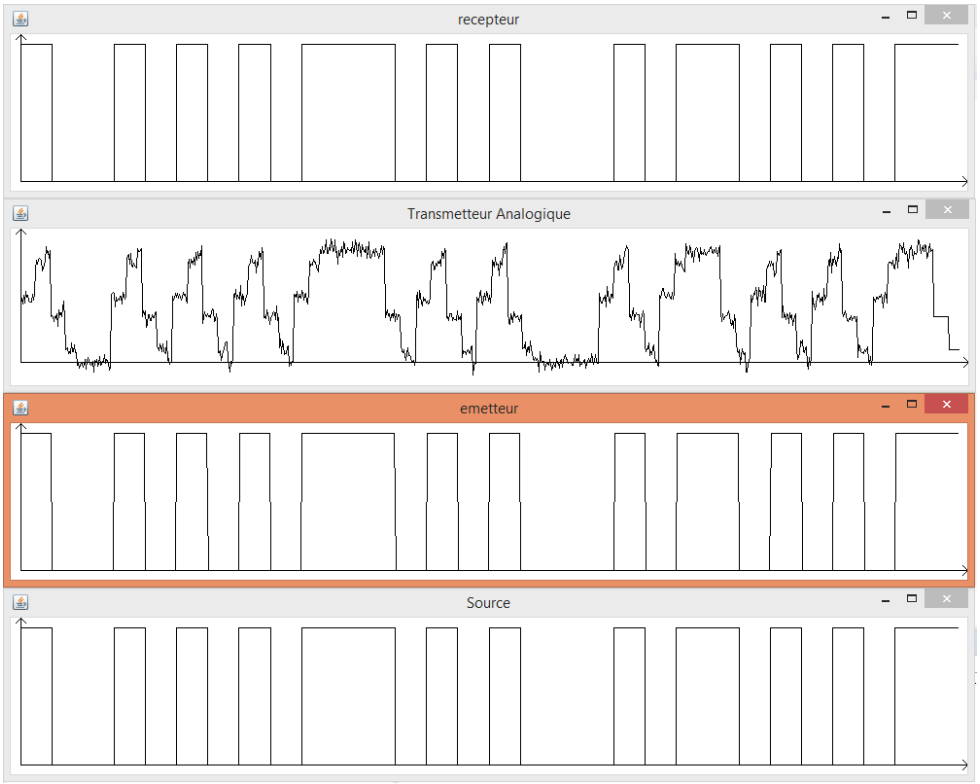

Figure 29 : Résultat de la simulation : java Simulateur -s -mess 1001010101110101000101101010111 -form NRZ -snr 10 -ti 15 0.5 -ti 25 0.2 => TEB : 0.0

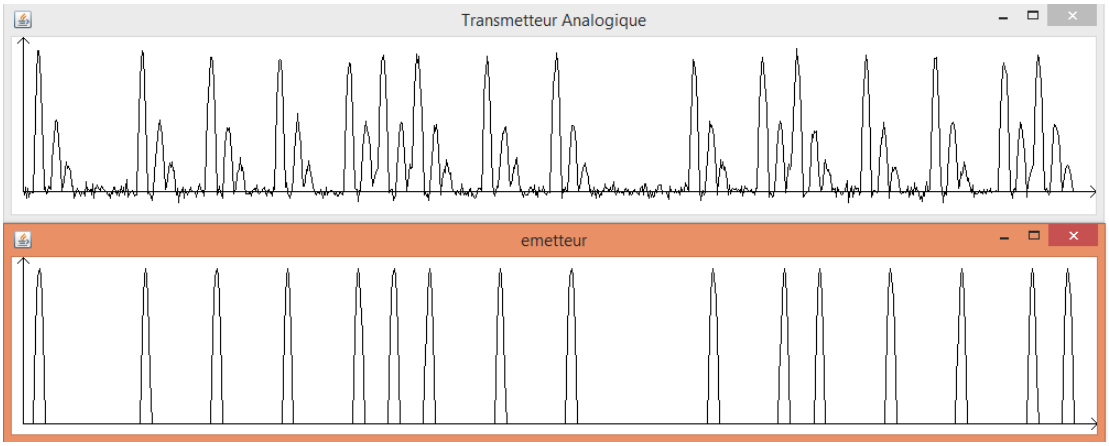

Figure 30 : Résultat de la simulation : java Simulateur -s -mess 1001010101110101000101101010 11 -form RZ -snr 10 -ti 15 0.5 -ti 25 0.2 => TEB : 0.0

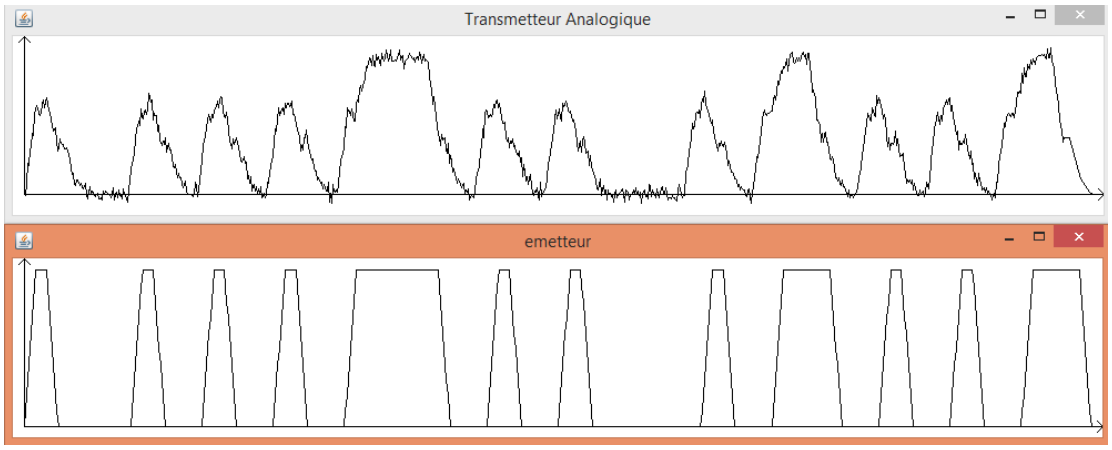

Figure 31 : Résultat de la simulation : java Simulateur -s -mess 1001010101110101000101101010 11 -form NRZT -snr 10 -ti 15 0.5 -ti 25 0.2 => TEB : 0.0

On constate que notre récepteur arrive à retrouver le bon message à chaque fois malgré les trajets multiples. En effet, étant donné que notre seuil de comparaison s'adapte au signal reçu, ce dernier varie et il est donc simple de retrouver le bon symbole.

La courbe suivante représente l'évolution du TEB pour le signal NRZ avec l'utilisation de 5 trajets multiples.

On passe en arguments : -mess 1000 -ti 10 0.5 -ti 15 0.2 -ti 20 0.1 -ti 5 0.6 -ti 25 0.8

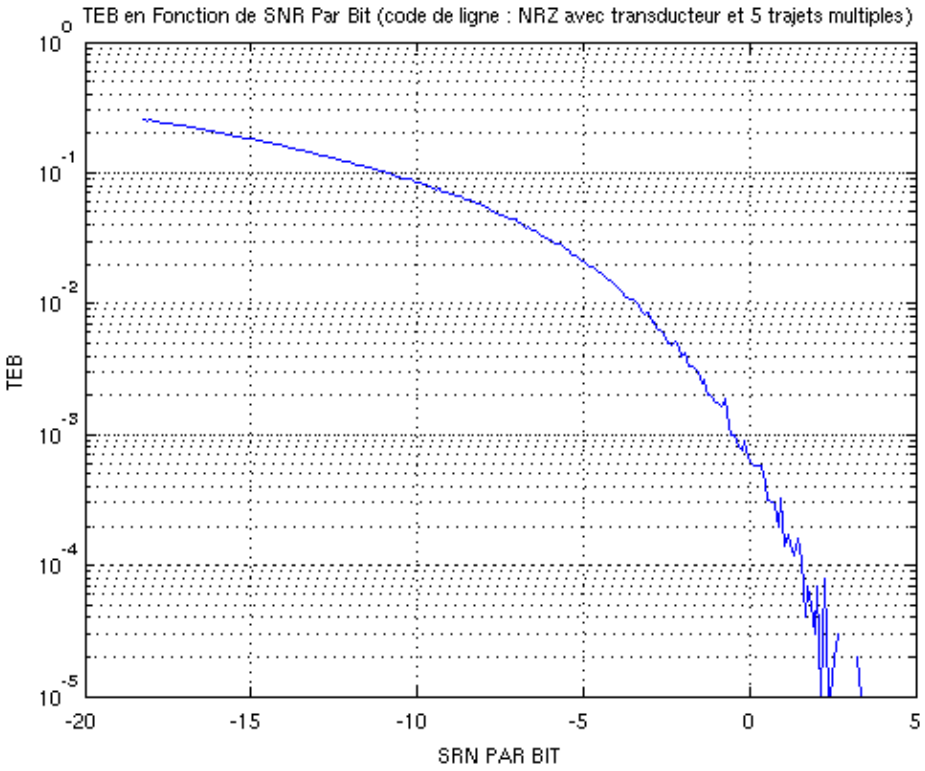

Figure 32 : Evolution du TEB pour le signal nRZ avec 5 trajets multiples

Nous constatons qu'il y a un problème sur cette courbe. En effet, en la comparant avec la même courbe de l'étape 3, nous constatons que l'ajout de trajets-multiples permettrait d'obtenir un meilleur TEB ce qui n'est pas possible…

Les autres courbes de TEB pour les signaux NRZT et RZ présentent le même problème, nous avons donc choisi de ne pas les afficher puisqu'elles sont similaires à celle de la figure 32.

Il est également intéressant d'observer l'influence des trajets multiples sur les diagrammes de l'œil des signaux NRZ et NRZT. On passe en arguments : -mess 1000 -ti 10 0.5 -ti 15 0.2 -ti 20 0.1 -ti 5 0.6 -ti 25 0.8 (et on fait varier la forme du signa, NRZ ou NRZT ainsi que le SNR, 0 ou 10 dB).

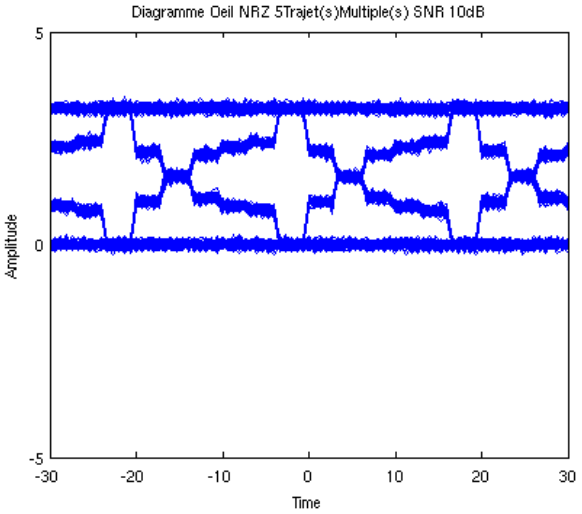

Figure 33 : Diagramme de l'oeil pour 5 trajets-multiples, signal NRZ, snr 10 dB

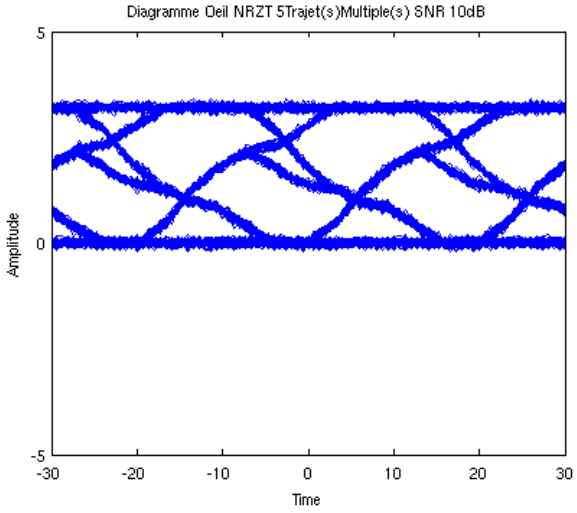

Figure 34 : Diagramme de l'oeil pour 5 trajets-multiples, signal NRZT, snr 10 dB

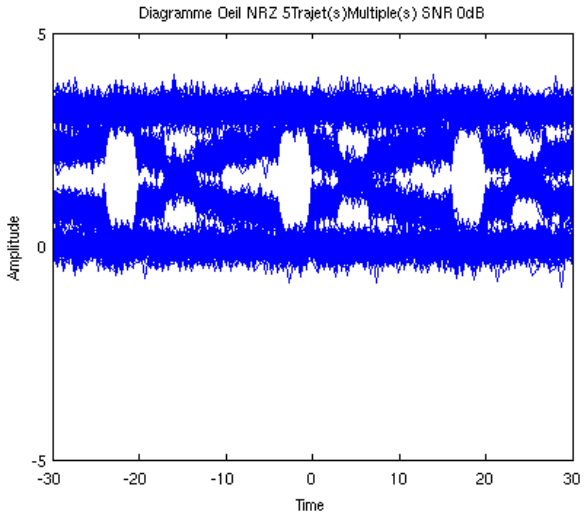

Figure 35 : Diagramme de l'oeil pour 5 trajets-multiples, signal NRZ, snr 0 dB

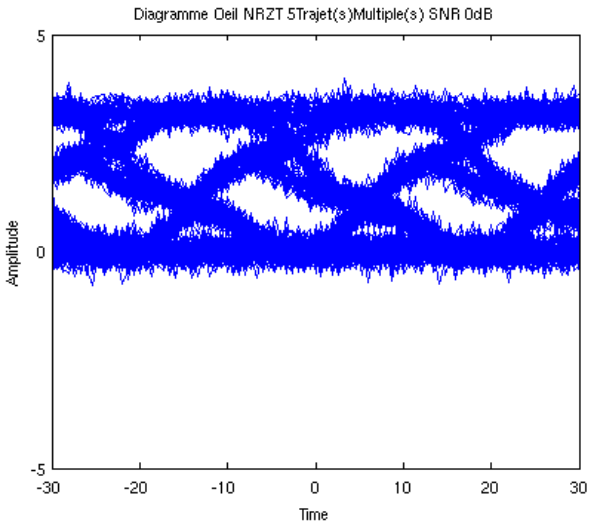

Figure 36 : Diagramme de l'oeil pour 5 trajets-multiples, signal NRZT, snr 0 dB

On constate que l'ajout des trajets-multiples impacte l'ouverture horizontale (le bruit impactant l'ouverture verticale principalement). L'instant de décision pour l'échantillonnage donné par l'ouverture horizontale est plus difficile à déterminer dans ces cas-là. Au niveau du signal NRZ, l'instant d'échantillonnage reste relativement lisible comparé au signal NRZT.

Le phénomène de trajet multiple est rencontré très fréquemment pour les transmissions télécom et surtout pour les transmissions en espace libre. Il est donc important de les prendre en compte et de diminuer au maximum leur impact. Pour cela, nous pouvons introduire de la redondance pour retrouver l'information. C'est ce que nous allons faire au cours de l'étape 5.

ETAPE 5 : CODAGE DE CANAL AVEC CANAL BRUITE DE TYPE « GAUSSIEN »

La cinquième étape consiste à ajouter un codage de canal pour ajouter de l'information redondante au message émis par la source. Le but est de compenser avec le bruit induit par le canal de transmission. Le codeur va donc rajouter ou supprimer de l'information avant de l'envoyer dans le canal puis le décodeur va effectuer l'opération inverse à l'arrivée pour retrouver le signal émis. La figure 37 modélise le montage à réaliser.

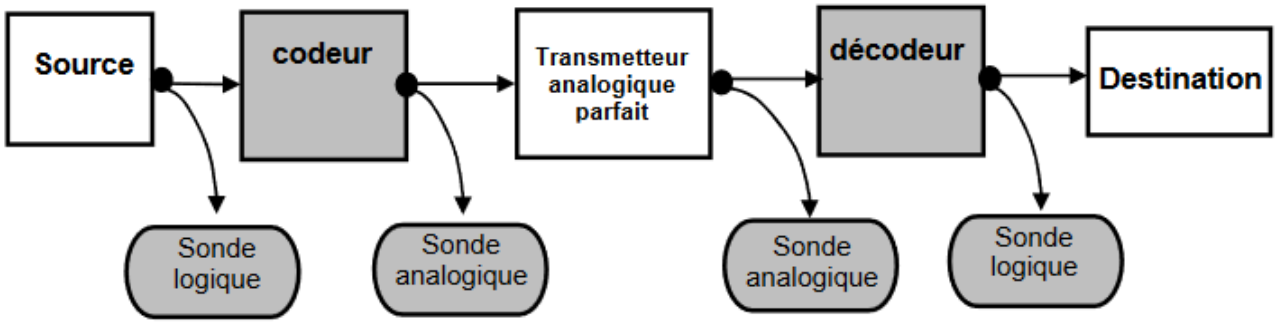

Figure 37 : Modélisation de la chaîne de transmission à l'étape 5

Pour ajouter l'information, nous utilisons un transducteur qui va convertir suivant le code suivant les symboles :

- 0 -> 0 1 0
- 1 -> 1 0 1

Cette méthode possède deux avantages :

- Faciliter la synchronisation émetteur-récepteur car il n'y a jamais 2 bits identiques consécutifs,
- Faciliter la détection des erreurs et la réparer lorsqu'il n'y a qu'un bit erroné sur un paquet de 3 bits.

En cas d'erreur sur un des bits (celui repéré en gras), voici les décisions qui seront prises :

- 0 **0** 0 -> 0
- **0** 0 1 -> 1
- 0 1 0 -> 0
- 0 1 **1** -> 0
- 1 0 **0** -> 1
- 1 0 1 -> 1
- **1** 1 0 -> 0
- 1 **1** 1 -> 1

Le problème de cette méthode est que l'on rajoute un traitement et on réduit la bande passante utile. Cependant cela permet d'améliorer le TEB.

Voici un exemple de l'effet des transducteurs sur un signal de forme NRZ :

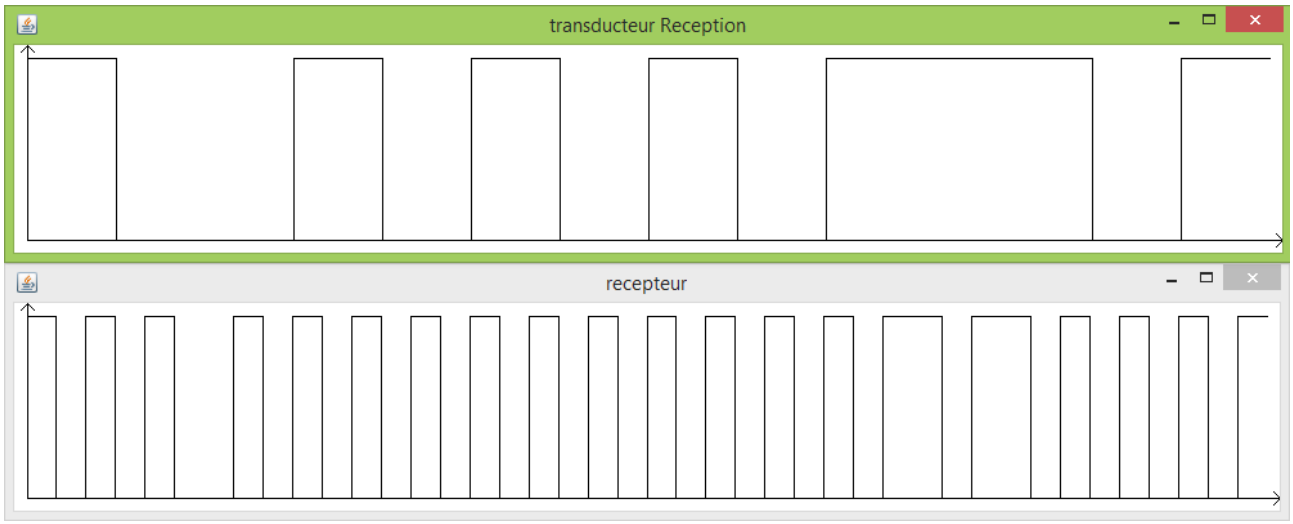

Figure 38 : Résultat de la simulation : java Simulateur -s -mess 10010101011101 -form NRZ -transducteur => TEB : 0.0

La sonde au niveau de la source est identique à celle du transducteur en réception (elle ne s'affiche pas sur tous les écrans en raison de la résolution).

On constate que pour chaque bit de la source (ou du transducteur en réception pour l'observation) correspond 3 bits pour la transmission.

Il est intéressant d'observer la nouvelle courbe d'évolution du TEB avec l'ajout des transducteurs :

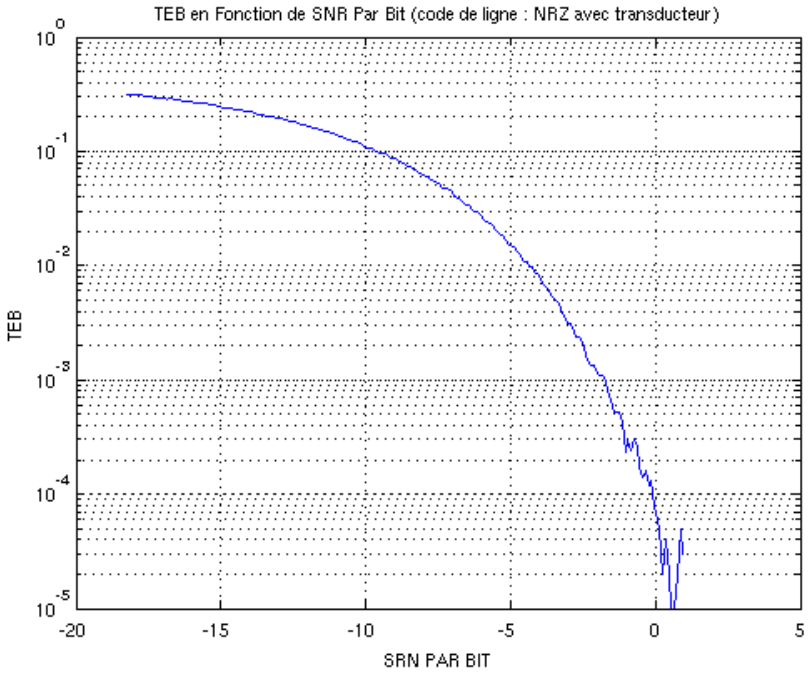

Figure 39 : Evolution du TEB en fonction du SNR pour le signal NRZ avec transducteurs

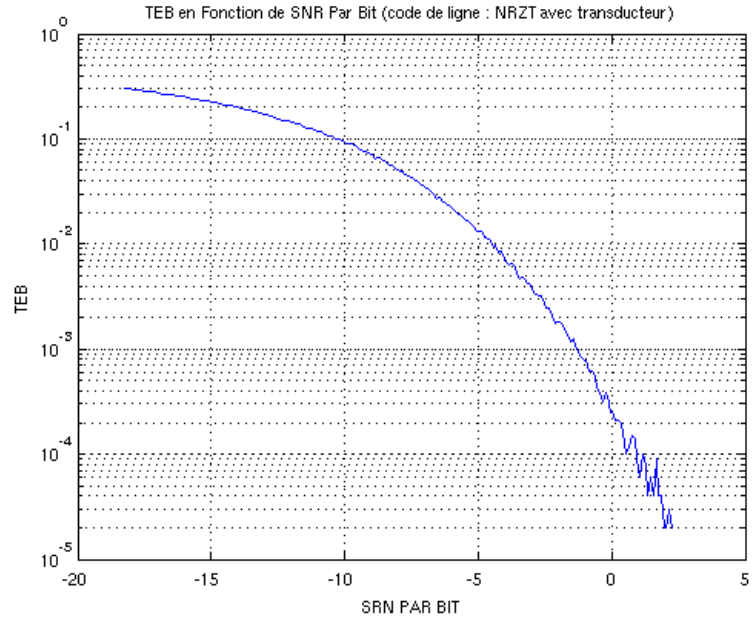

Figure 40 : Evolution du TEB en fonction du SNR pour le signal NRZT avec transducteurs

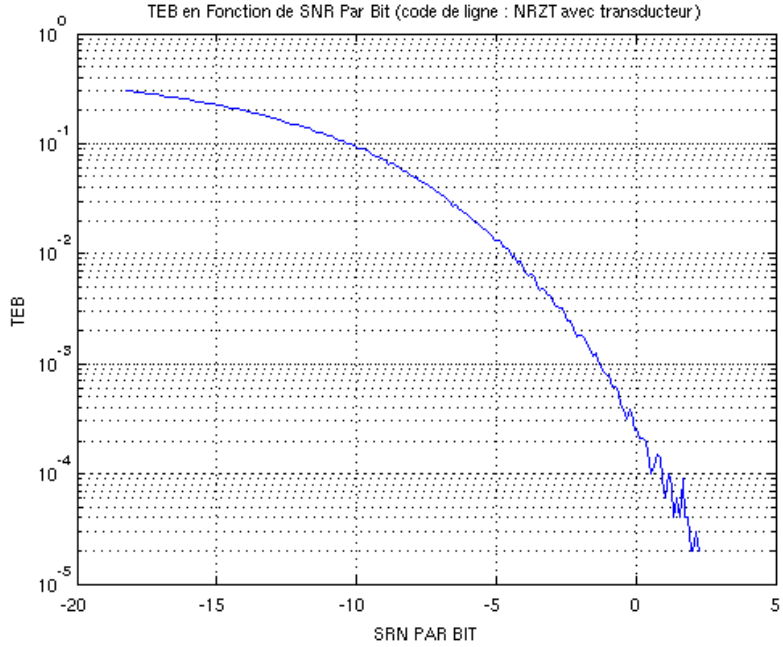

Figure 41 : Evolution du TEB en fonction du SNR pour le signal RZ avec transducteurs

Les courbes de TEB montrent l'influence des transducteurs. En comparant par rapport à l'étape 3, on constate que pour une même valeur de SNR, l'utilisation des transducteurs permet d'obtenir un meilleur TEB (par exemple, pour un SNR de -5 dB sur le signal NRZ, on a un TEB de 0,15 à l'étape 3 et de 0,01 pour l'étape 5 avec les transducteurs).

Nous observons également une modification au niveau des diagrammes de l'œil pour les signaux NRZ et NRZT. On passe en arguments : -mess 1000 – transducteur (et on fait varier la forme du signa, NRZ ou NRZT ainsi que le SNR, 0 ou 10 dB).

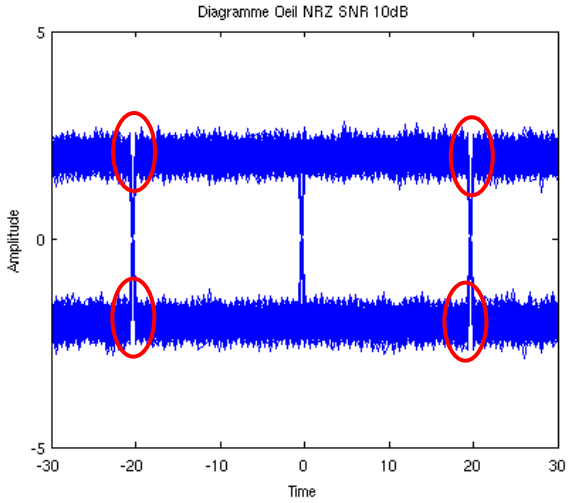

Figure 42 : Diagramme de l'oeil du signal NRZ avec transducteur et SNR 10 dB

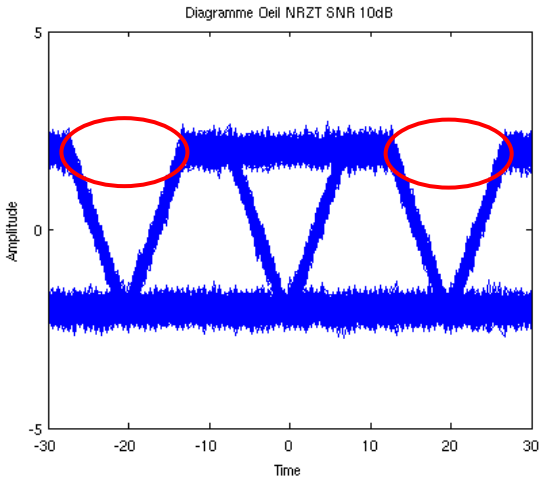

Figure 43 : Diagramme de l'oeil du signal NRZT avec transducteur et SNR 10 dB

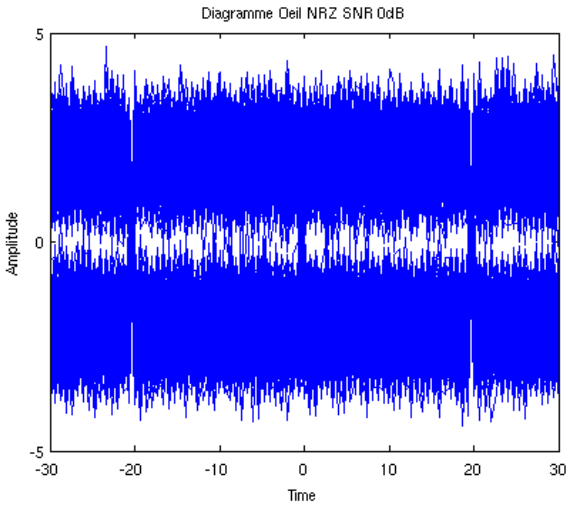

Figure 44 : Diagramme de l'oeil du signal NRZ avec transducteur et SNR 0 dB

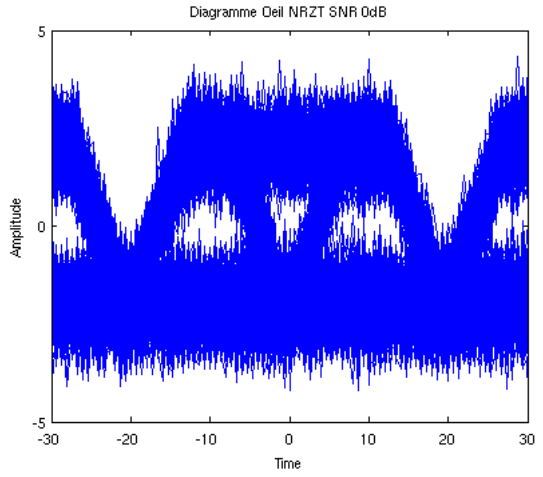

Figure 45 : Diagramme de l'oeil du signal NRZT avec transducteur et SNR 0 dB

On constate que les ouvertures verticales et horizontales sont les mêmes que pour les diagrammes de l'œil de l'étape 3. Ce qui change c'est les espaces vides au niveau du changement des bits (entourés en rouge sur les figures X et X). En effet, l'utilisation des transducteurs empêchent d'avoir deux bits identiques consécutifs, nous avons donc un changement à chaque bit et des valeurs deviennent impossible à avoir (par exemple 1 1 1). Des valeurs telles que 1 1 0 restent possible si le message initial comporte la séquence suivante : 1 1, dans ce cas le résultat après les transducteurs sera : 1 0 1 1 0 1 et nous utilisons le fait que si deux bits consécutifs sont à 1 alors nous ne revenons pas à la valeur min entre les deux.

L'utilisation des transducteurs est intéressante pour synchroniser facilement l'émetteur et le récepteur mais surtout pour détecter et corriger les erreurs dans une transmission (s'il n'y a pas deux bits consécutifs erronés). Cependant cette méthode possède des limites, notamment la réduction de la bande passante. En effet, nous passons d'une transmission d'un seul bit à 3 bits pour transmettre le même symbole.

CONCLUSION

Au cours de ce projet, nous avons pu simuler pas à pas une chaine de transmission basique. Nous avons ainsi implémenté les différents éléments qui constituent une chaine de transmission (émetteur, récepteur, canal, encodeur, décodeur, …). Tout au long du projet, nous avons essayé d'apporter des améliorations pour les problèmes que nous avons rencontrés (optimisation, bruit trop important, etc.). Nous avons ainsi acquis de nouvelles compétences d'un point de vue informatique (codage sur un projet complexe et répartition des différentes parties du code) et télécom (analyse des problématiques et des résultats).

Ce projet nous a ainsi apporté des compétences et connaissances diverses. Nous avons ainsi appris à maitriser des outils de travail collaboratif (notamment GitHub), des compétences en télécom (notions de bruit blanc, de trajets-multipes, …), des compétences en informatique (structure et architecture du code) et des compétences relationnelles pour le travail d'équipe.

Dans l'ensemble notre solution est fonctionnelle mais elle possède malgré tout quelques limites. En effet notre solution ne serait pas utilisable à l'état actuel. Notre calcule de seuil est une première limite puisque nous le calculons une fois l'ensemble du signal reçu. Dans le cas où nous recevons un flux constant de données alors nous ne pourrions pas utiliser notre formule pour calculer le seuil. Une solution serait alors de calculer le seuil régulièrement (par exemple on calcule le seuil pour les 100 premiers bits puis on l'applique aux bits suivants qui arrivent et on re-calcule le seuil à intervalle régulier). Un second point qui pourrait être améliorer est le moyennage pour le signal RZ qui est fait sur 10 échantillons mais qui pourrait être restreint aux échantillons centraux proches du max.

Lors des présentations des autres groupes, nous avons relevé quelques points forts que nous n'avons pas mis en place dans notre projet :

- Groupe de Souraya CHAWKI : le calcule théorique pour avoir une idée des résultats attendus, circuit de validation pour le rapport et les tests du code.
- Groupe de Clément-Alan Girard : l'apprentissage entre l'émetteur et le récepteur pour gérer les trajets-multiples.
- Groupe de Bastien Sanchez : l'utilisation réussi du filtre adapté.

ANNEXES

ANNEXE 1

Cette annexe illustre l'utilisation de matlab pour effectuer un filtre adapté et résoudre le problème de trajets multiples. Toutefois, cette solution ne fonctionne pas dans tous les cas, il existe quelques cas particuliers (nous ne les avons pas tous identifié) pour lesquels cette méthode ne fonctionne pas.
Voici l'explication de la méthode :

La figure 46 représente le message à émettre (101000) pour un signal RZ avec un temps bit de 30 échantillons.

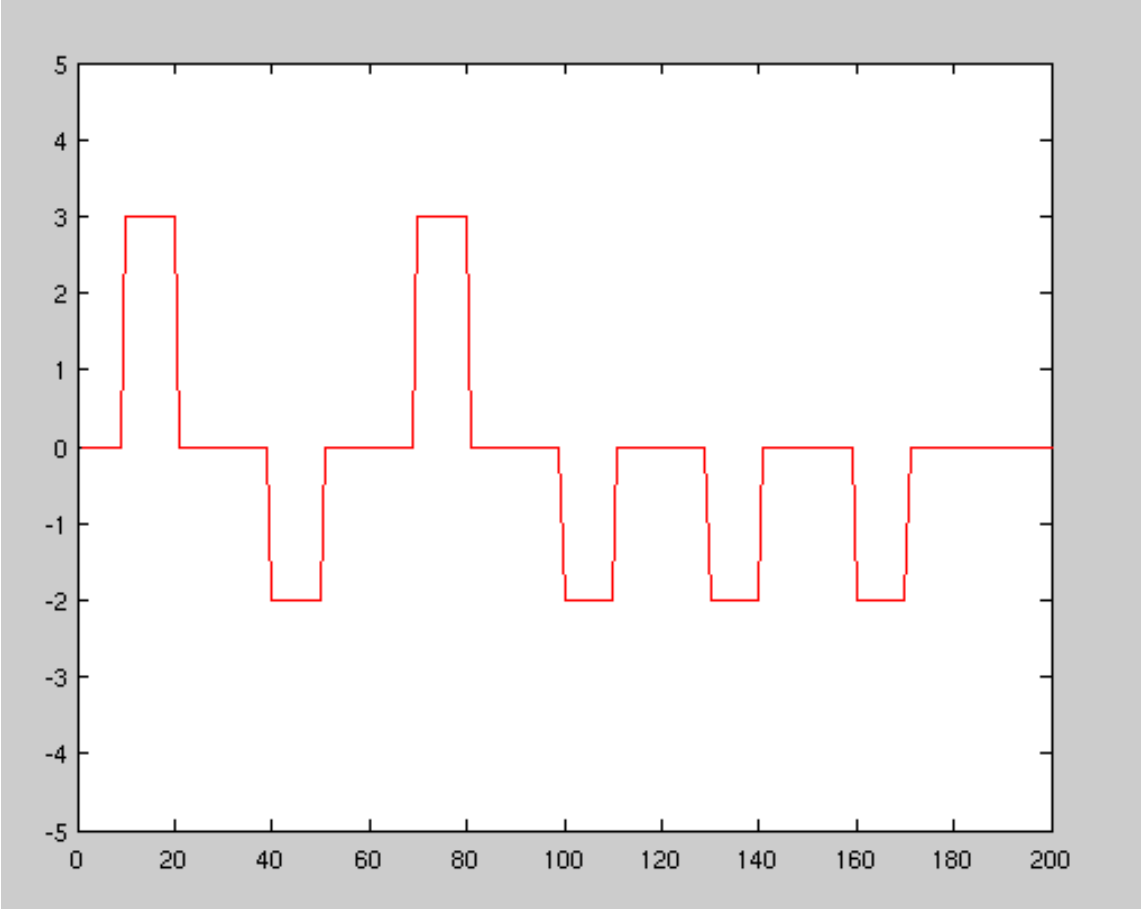

Figure 46 : Signal à émettre : 101000

A ce signal RZ, on ajoute un trajet multiple décalé de T = 20 échantillons et avec une amplitude α = 0,5. L résultat est illustré sur la figure 47.

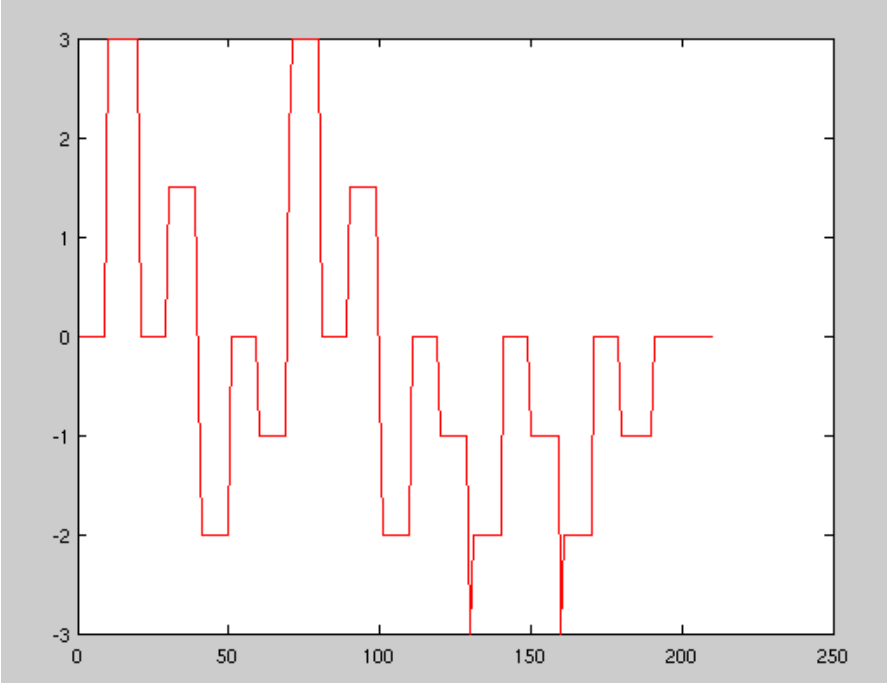

Figure 47 : Signal émis + trajets-multiples

On convolue ensuite le signal reçu (signal émis + trajets-multiples) que l'on a recentré (pour garder un seuil à 0 au moment de la prise de décision) avec le filtre adapté de réponse impulsionnelle illustré en figure 48 :

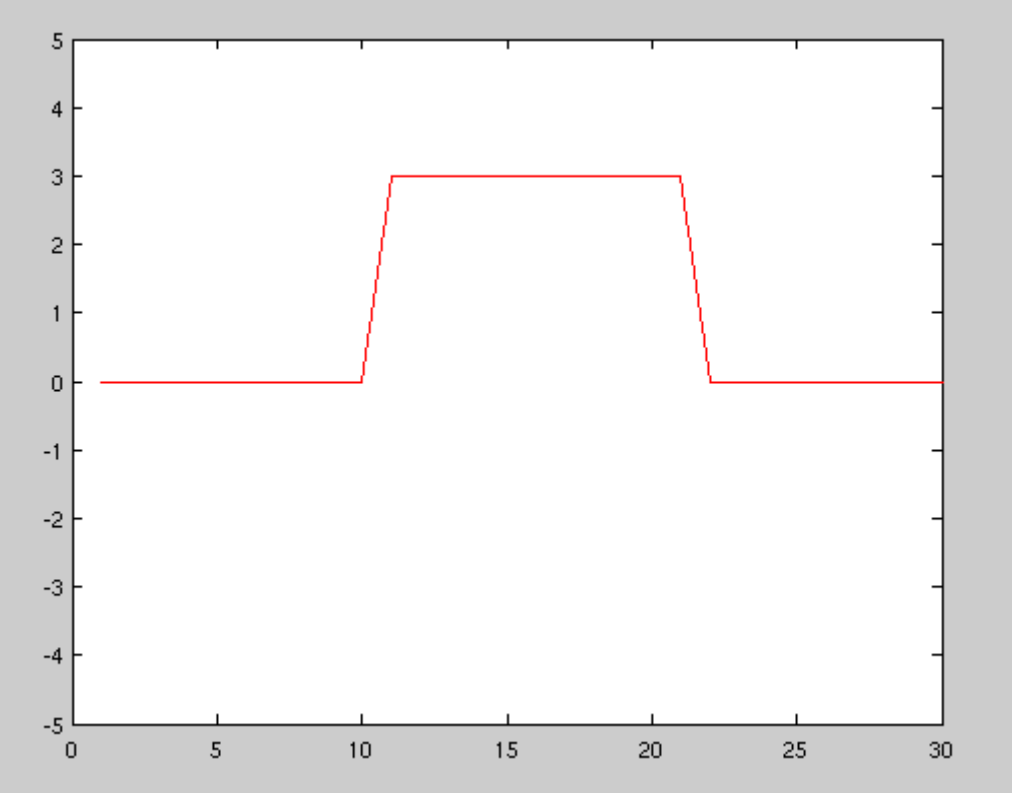

Figure 48 : Réponse impulsionnelle du filtre adapté

On récupère ensuite le signal en sortie du filtre (représenté en figure 49). On prend ensuite une décision tous les 30 échantillons suivant ce principe :

- Si le signal est supérieur au seuil alors c'est un 1 logique,
- Si le signal est inférieur au seuil alors c'est un 0 logique.

Ainsi pour le signal illustré par la figure 49, on reconnait le message d'origine à savoir 101000.

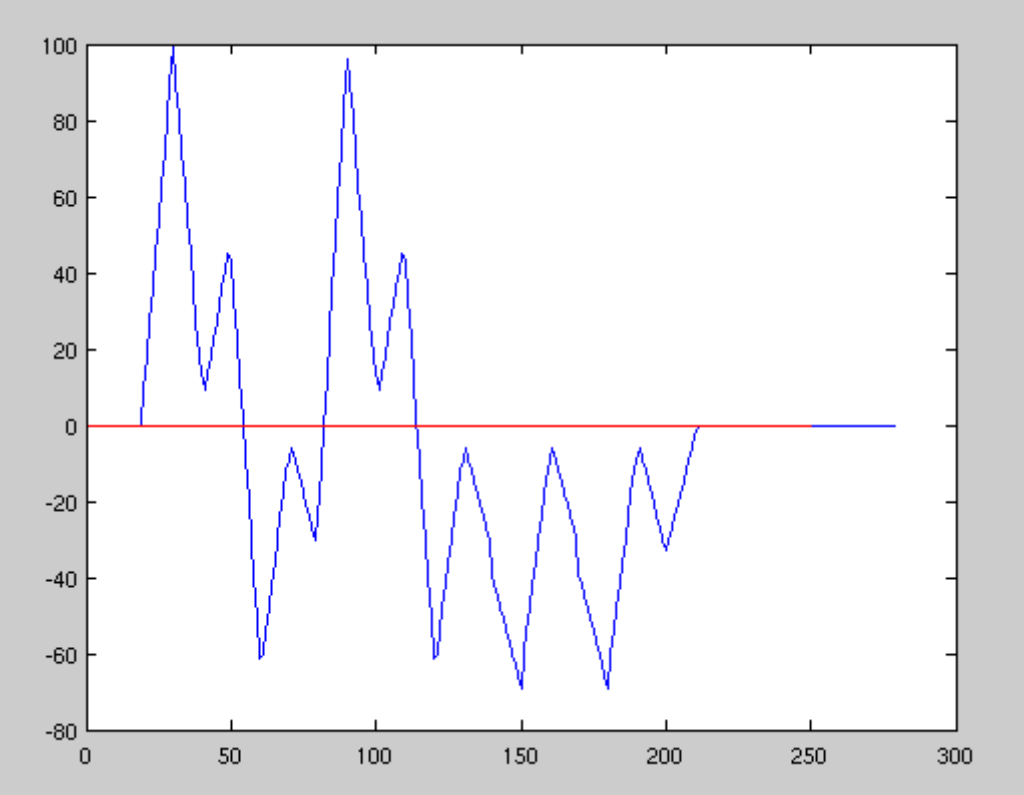

Figure 49 : Sortie du filtre adapté avec échantillonnage et seuil de détection

Cette solution présente l'avantage d'avoir un seuil à 0 à chaque fois (on recentre le signal reçu juste avant le filtre adapté pour garder un seuil constant) et on peut donc analyser tous les symboles dans un flux continu de données.

Cette solution est utilisable dans la nature contrairement à celle que nous avons implémenté qui requiert de connaitre l'ensemble du signal pour calculer le seuil d décision.

$$Seuil = \frac{Moyenne\ de\ toutes\ les\ valeurs\ du\ signal}{Nombre\ d'échantillon\ total}$$