SIT213 Etape 2
IMT Atlantique

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

12 SEPTEMBRE

Créé par :
LE DUC Elouan
MAQUIN Philippe
LE GRUIEC Clément
LE JEUNE Matthieu
FRAIGNAC Guillaume

Table des matières

Intentions du projet

Il s'agit de réaliser, par équipe de 4 ou 5 élèves, une maquette logicielle (en Java) simulant un système de transmission numérique élémentaire. On intégrera donc dans la chaîne un bloc de modulation numérique.

Le système sera assemblé suivant une bibliothèque de modules comportant des ports d'entrée, des ports de sortie et des paramètres physiques. Ces derniers pourront être déterminés à partir des activités du module SIT 212. Le système global sera mis au point progressivement sur 5 séances au cours desquelles les modules seront raffinés, complétés, validés et connectés selon un schéma de transmission de type « point-à-point ».

Outre la qualité technique de la réalisation, on insistera sur les points suivants :

1. La qualité de documentation de la maquette logicielle (notamment la Javadoc).
2. Les efforts de validation des résultats de simulation produits par la maquette.
3. La maîtrise du processus de travail : gestion des versions successives de la maquette logicielle et du dossier technique afférent, synergie de l'équipe, démarche qualité. Concernant ce tout dernier critère, le respect des exigences de mise en forme du livrable sera primordial.

2ème étape du projet

➢ Objectifs

Dans cette phase du projet nous allons travailler sur une transmission non bruitée d'un signal analogique. On prendra en compte la nature analogique du canal de transmission en faisant évoluer la chaîne de transmission par l'adjonction de deux étages (logique → analogique et analogique → logique), comme indiqué sur le schéma de la figure 1.

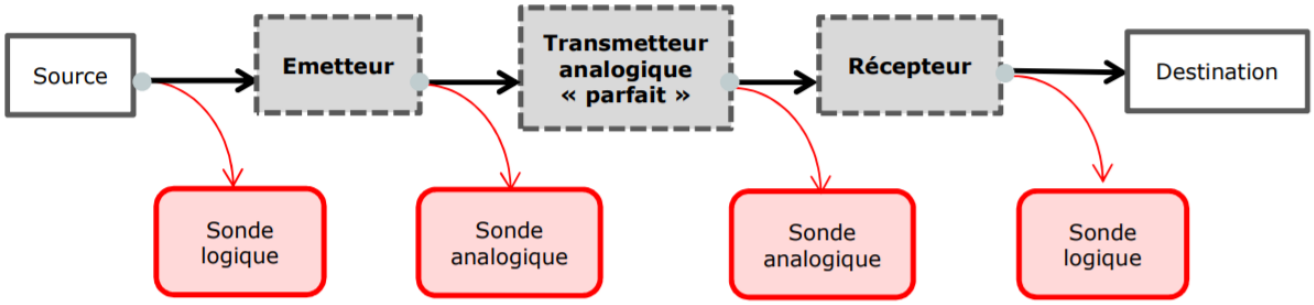

Figure 1 Modélisation de la chaîne de transmission à l'étape 2.

Par défaut le simulateur doit utiliser une chaîne de transmission logique, avec un message aléatoire de longueur 100, sans utilisation de sondes et sans utilisation de transducteur.

L'option *-mess m* précise le message ou la longueur du message à émettre :

- Si *m* est une suite de 0 et de 1 de longueur au moins égale à 7, m est le message à émettre.
- Si m comporte au plus 6 chiffres décimaux et correspond à la représentation en base 10 d'un entier, cet entier est la longueur du message que le simulateur doit générer et transmettre.
- Par défaut le simulateur doit générer et transmettre un message de longueur 100.

L'option *-s* indique l'utilisation des sondes. Par défaut le simulateur n'utilise pas de sondes

L'option *-form f* précise la forme d'onde. Le paramètre f peut prendre les valeurs suivantes :

- NRZ : forme d'onde rectangulaire
- NRZT : forme d'onde trapézoïdale
- RZ : forme d'onde impulsionnelle

L'option *-nbEch ne* en transmission analogique, précise le nombre d'échantillons par bit.

L'option *-ampl min max* en transmission analogique précise l'amplitude min et max du signal

➢ Analyse des actions à mener

Nous allons devoir programmer un simulateur de signaux analogiques sur une machine. La nature du signal ne s'y prête pas. C'est pourquoi nous allons essayer de faire au mieux en suréchantillonnant des signaux logiques. Il faudra transformer une liste de Boolean en une liste de Float. Les True et les False prendront des valeurs différentes en fonctions de la forme et des amplitudes choisies. Cela correspondrait au schéma de la figure 2 ci-dessous.

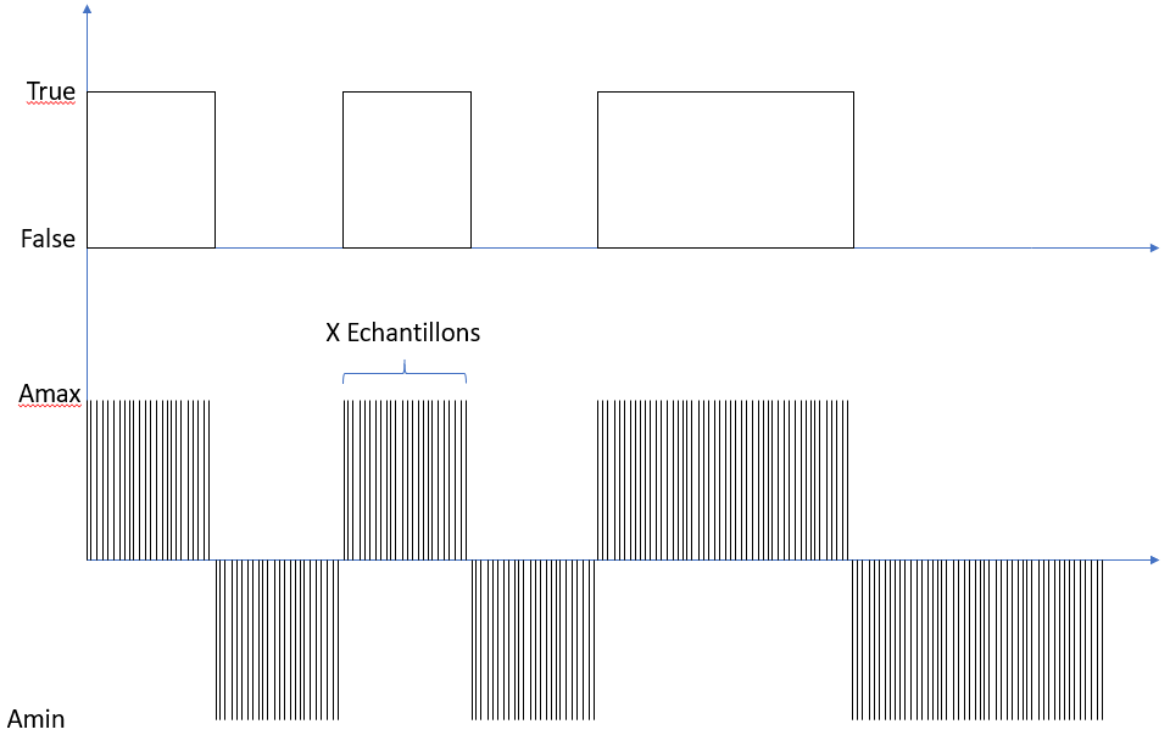

Figure 2 Schéma de transformation de Boolean en Float

Nous avons 3 formes possibles à prendre en compte :

- NRZ

C'est le type de forme la plus simple. Chaque bit correspond à une valeur, il suffit donc d'insérer dans l'information X valeurs correspondant au bit à convertir. Par exemple si on tombe sur un 1 (True), on mettra 30 Float de valeur 5.0 dans l'information, et pour le cas d'un 0, 30 Float de valeur -5.0. Cela ne sera pas forcément très réaliste car nous aurons un signal analogique très carré en sortie. Cela correspond au schéma de la figure 1.

- NRZT

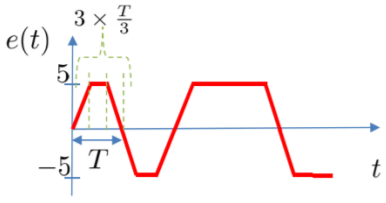

Figure 3 NRZT avec 10110

Une forme dérivée du NRZ et plus réaliste car on intègre une pente sur les fronts montants et descendants. Chaque symbole sera en tiers de période afin de prendre en compte la phase montante, stabilisée et descendante. Quand 2 symboles identiques se suivent il ne faudra pas réaliser l'ensemble des phases et garder le signal sur son seuil (Amax ou Amin).

- RZ

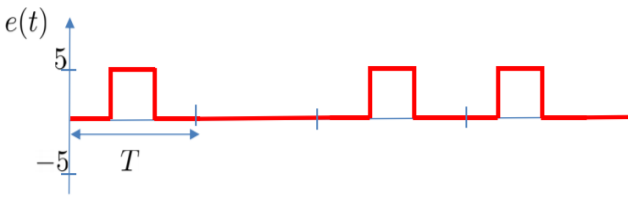

Figure 4 RZ avec 10110

C'est une forme qui est particulièrement lisible. Lorsque l'on a un 1 (True), elle maintiendra la valeur Amax sur un tiers de sa période seulement et reviendra à 0 sur les deux autres tiers. La valeur max se situe sur le tier central. Pour un 0 (False), elle reste à 0. Avec cette forme, même si 2 bits de mêmes valeurs se suivent, les valeurs retourneront à 0 sur 2 tiers de période.

D'après le schéma de la figure 1 nous devons programmer 3 nouveaux blocs : un émetteur, un transmetteur analogique parfait et un récepteur. Les sources et destinations sont les mêmes qu'à l'étape une. Quand on y regarde de plus près, l'émetteur et le récepteur sont des transmetteurs qui encodent ou décodent nos informations. Ils ont tous deux, une entrée et une sortie, l'une pour des Float, l'autre pour des Boolean. Nous pourrons donc faire hériter ces 3 nouveaux blocs de la classe Transmetteur et bénéficier des fonctionnalités déjà programmées. Pour ce qui concerne les sondes, elles ont déjà été programmées par les enseignants.

Nous devons également programmer la gestion des paramètres du logiciel afin de pouvoir configurer notre simulateur facilement sans avoir à retourner dans le code source. Certaines fonctions avaient déjà été réalisées par les enseignants, d'autres restent à programmer.

Nous devons implémenter les paramètres suivants : *-form f, nbEch ne, -ampl min max, -seed v.*

Pour chacune de ces fonctionnalités nous devons détecter les paramètres du programme et s'assurer de leur(s) conformité(s) en utilisant des Regex par exemple.

La fonctionnalité -seed pour paramétrer une graine dans le cadre des générations aléatoires avait été omis dans l'étape 1. Nous l'avons donc ajouté.

➢ Tests et validations du programme

À ce stade du projet nous n'avons pas encore automatisé les tests mais nous pouvons déjà en réaliser grâce aux données tel que le TEB et l'affichage des sondes.

✓ Résultats attendus

Nous sommes dans le cas d'une chaine de transmission avec un transmetteur analogique parfait. Le signal émis doit être identique au signal reçu qu'il soit converti sous une forme NRZ, NRZT ou encore RZ.

Sur les sondes les signaux sources et destinations sont identiques, les signaux en sorti de l'émetteur et du transmetteur sont également identiques. Dans tous les cas le TEB doit être à 0 à la fin de la chaine de transmission si tout se passe bien. Les formes d'ondes choisies doivent être utilisées selon leur spécifications.

Pour les tests nous utiliserons un signal avec les paramètres suivants :

- Amplitude max : 5V
- Amplitude min : -5V
- Nb échantillon : 60
- Seed : 40
- Longueur (mess) : 20

Le signal aléatoire généré sera égal à 1101 0011 0100 1001 1111

Ce message est intéressant, il permet de tester différents cas de figure, par exemple deux 1 qui se suive, le passage d'un 0 à un 1 etc…

✓ Tests avec signal de type NRZ

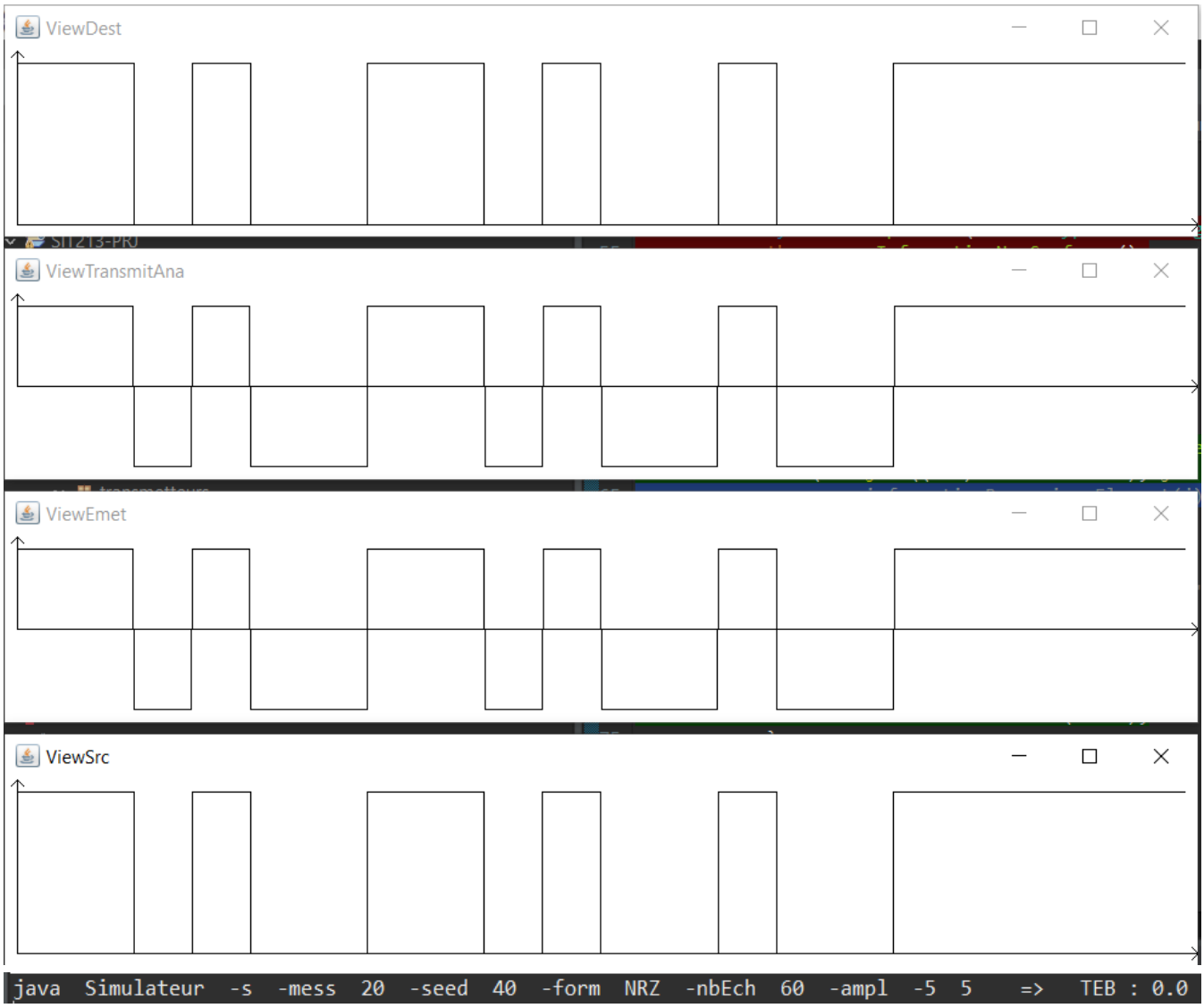

Figure 5 Tests du signal de type NRZ

Le signal émis correspond bien au signal reçu. Le TEB le confirme et est bien égal à 0. La conversion analogique s'est réalisée comme prévu, 5V pour un bit à 1 et -5v pour un bit à 0. Le test est validé.

✓ Tests avec signal de type NRZT

Figure 6 Tests du signal de type NRZT

Le signal émis correspond bien au signal reçu. Le TEB le confirme et est bien égal à 0. La conversion analogique s'est réalisée comme prévu, 5V pour un bit à 1 et -5v pour un bit à 0. Que cela soit un 0 ou un 1 qui est transmis, il y a des pentes. Lorsque deux symboles identiques se suivent, le niveau est maintenu, il n'y a pas de retour à l'amplitude min. Le test est validé.

✓ Tests avec signal de type RZ

Figure 7 Tests du signal de type RZ

Le signal émis correspond bien au signal reçu. Le TEB le confirme et est bien égal à 0. La conversion analogique a réagi comme prévu, 5V pour un bit à 1 et 0V pour un 0. Pour un 0 il n'y a pas de changement de niveau, pour un 1 nous avons bien 2 tiers du symbole à 0 et 1 tiers du symbole à 5V. Quand deux 1 se suivent il y a quand même un retour à 0. Le test est validé.

Conclusion

Nous avons ainsi pu simuler une conversion numérique à analogique et inversement, analogique à numérique. Les tests se sont passés comme prévus, les résultats sont donc satisfaisants.

Dans l'étape 3 nous devrons implémenter une transmission non-idéale avec canal bruité de type « gaussien ». Il faudra programmer un ou plusieurs nouveaux blocs afin d'y arriver. Le travail réalisé lors de cette seconde étape sera une base importante pour la suite du projet.

Table des illustrations