SIT213 Etape 1
IMT Atlantique

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

7 SEPTEMBRE

Créé par :
LE DUC Elouan
LE GRUIEC Clément
DEMOLIN BENJAMIN

Table des matières

Objectifs généraux

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.

1ère étape du projet

➢ Cible

Transmission élémentaire "back-to-back". On introduira le premier modèle de transmission schématisé sur la figure 1. Il vérifie les propriétés suivantes :

- La source émet une séquence booléenne soit fixée, soit 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.

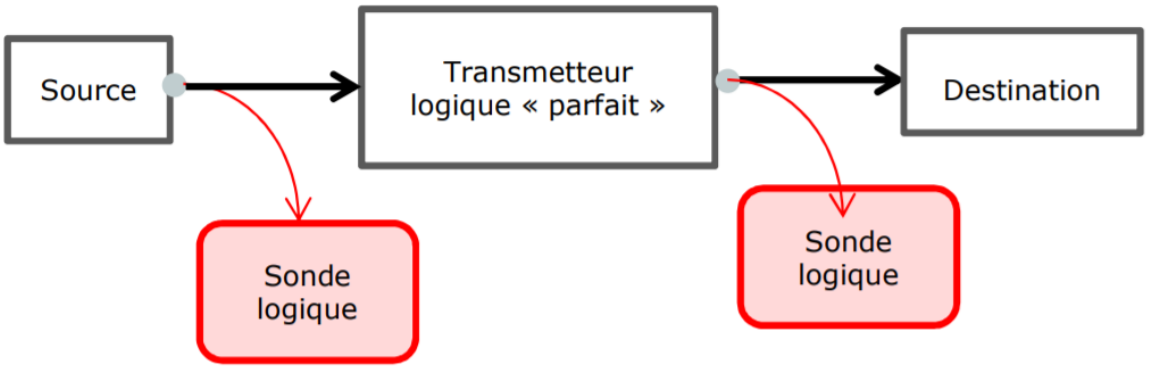

Figure 1 Modélisation de la chaîne de transmission à l'étape 1.

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

➢ Analyse des actions à mener

Pour guider le développement de notre simulateur, un diagramme de classe (figure 2) nous a été fourni. Cela permet d'avoir une vue globale sur la structure à mettre en œuvre ainsi que sur les parties spécifiques à développer dans le cadre de l'étape 1 du projet.

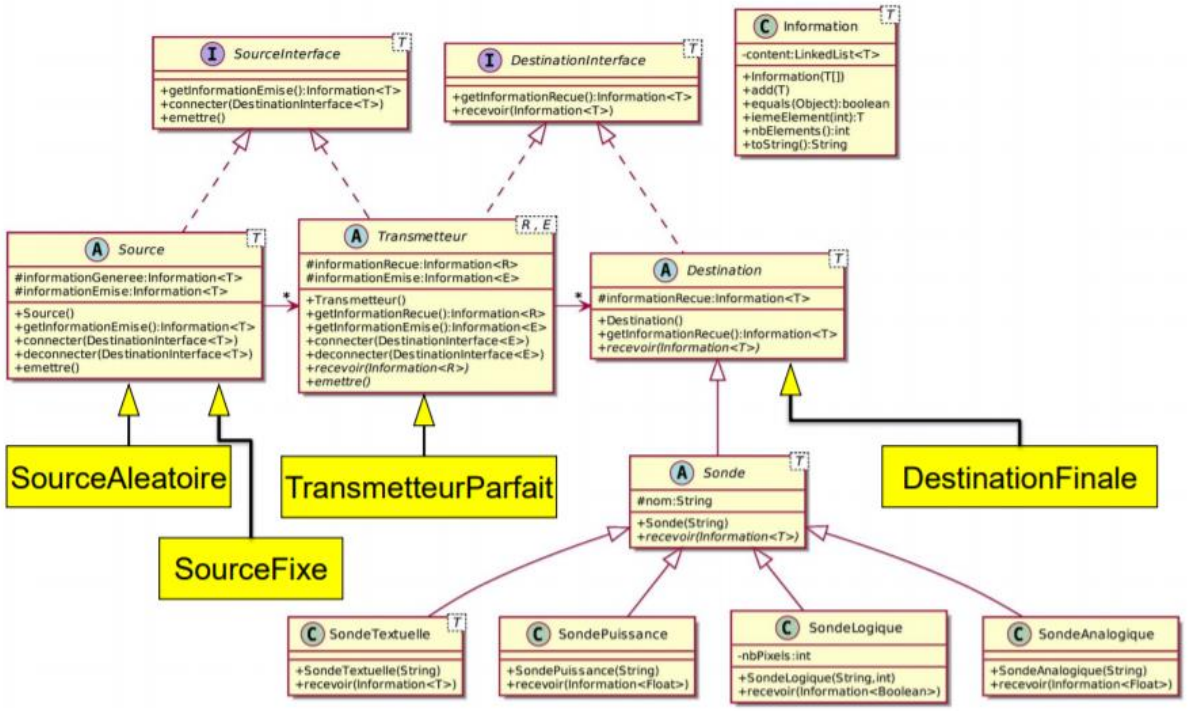

Figure 2 Diagramme de Classe du projet simulateur

De manière général chaque classe correspond à une fonctionnalité spécifique de notre chaine de transmission. Des classes mères abstraites et des interfaces ont déjà été développées au préalable par l'équipe enseignante. Nous pouvons ainsi avoir une base commune sur le projet. Les sous-blocs restent à développer en prenant en compte l'existant et les objectifs.

D'après le diagramme de classe nous avons 4 blocs à implémenter, cela correspond aux éléments en jaune vif sur la figure 2. Pour nous donner des indications sur les méthodes à écrire et sur les interactions entre les différents éléments du programme nous disposons également d'un diagramme de séquence. Les nouveaux blocs seront orchestrés depuis la classe *Simulateur*, le « main ».

Les conditions et indications décrites dans la cible peuvent s'apparenter à un cahier des charges. Ceux sont des éléments à prendre impérativement en compte dans le développement de l'ensemble des fonctionnalités.

➢ Programmation et connexion des blocs

Dans cette partie nous allons parler brièvement des éléments qui ont été programmés. Nous nous concentrerons sur la classe *Simulateur* afin de ne pas alourdir le compte-rendu. Les codes sources des classes développées sont disponibles dans l'archive fournit avec ce document. Une JavaDocs est également disponible afin d'aider à la compréhension des méthodes implémentées.

✓ Mise en place des liaisons entre les blocs

Le constructeur de la classe *Simulateur* est programmé selon les indications que nous avons eues. C'est-à-dire que l'ensemble des blocs de la chaine de transmission doivent être instanciés et connectés ensemble. Des « sondes » permettent de pouvoir visualiser graphiquement l'entrée ou la sortie d'un bloc. C'est une fonctionnalité qui sera très utile pour les tests à réaliser.

Le code permettant de mettre en œuvre la chaine selon le schéma de la figure 1.

```
public Simulateur(String[] args) throws ArgumentsException {

        //Analyse des arguments
        analyseArguments(args);

        //Instanciations des differents blocs de traitement
        if (messageAleatoire) {
            source=new SourceAleatoire(nbBitsMess);
        } else {
            source=new SourceFixe(messageString);
        }

        transmetteurLogique = new TransmetteurParfait();
        destination = new DestinationFinale();

        //Instanciations des differentes sondes
        SondeLogique viewSrc = new SondeLogique("ViewSrc", 720);
        SondeLogique viewTransmit = new SondeLogique("ViewTransmit", 720);


        //connexion des blocs ensembles
        source.connecter(transmetteurLogique);
        if(affichage) source.connecter(viewSrc);
        transmetteurLogique.connecter(destination);
        if(affichage) transmetteurLogique.connecter(viewTransmit);

}
```

✓ Calcul du TEB

Le programme doit calculer le TEB (Taux d'Erreur Binaire) du système. Pour calculer ce taux nous utilisons la formule suivante :

$$TEB = \frac{Nombre\ de\ bits\ en\ erreur\ recue}{Nombre\ de\ bits\ total\ emis}$$

Pour l'étape 1 ce TEB est optionnel puisque nous avons affaire à un transmetteur parfait. Si tout à bien été fait nous devons avoir un résultat final à 0.0 quel que soit le signal en entrée du système.

En Java le TEB est implémenté de la manière suivante :

```java
public float calculTauxErreurBinaire() {

        int nbErr=0;
        float TEB=0.0f;
        for (int i = 0; i < destination.getInformationRecue().nbElements(); i++) {
            if(destination.getInformationRecue().iemeElement(i)
               !=source.getInformationEmise().iemeElement(i)) nbErr++;
        }
        TEB=(nbErr*1.0f)/(source.getInformationEmise().nbElements());

        return TEB;
}
```

➢ Tests et validations du programme

✓ Vérification du TEB

Dans le cas d'une transmission parfaite le TEB attendu pour tout signal est à 0. Ce test est validé et fonctionne peu importe le signal, qu'il ait été généré de manière aléatoire ou qu'il ait été fixé.

```
java  Simulateur  -mess  10  -s    =>   TEB : 0.0
java  Simulateur  -s    =>   TEB : 0.0
java  Simulateur  -s  -mess  10101010    =>   TEB : 0.0
```

Par acquis de conscience j'introduis volontairement 33 erreurs dans le signal par défaut afin de vérifier le calcul. Je m'attends alors à voir un TEB à 0.33. Et c'est bien le cas :

```
java  Simulateur  -s    =>   TEB : 0.33
```

✓ Vérification des signaux entrée / sortie

Cas avec un signal par défaut (longueur 100, aléatoire) :

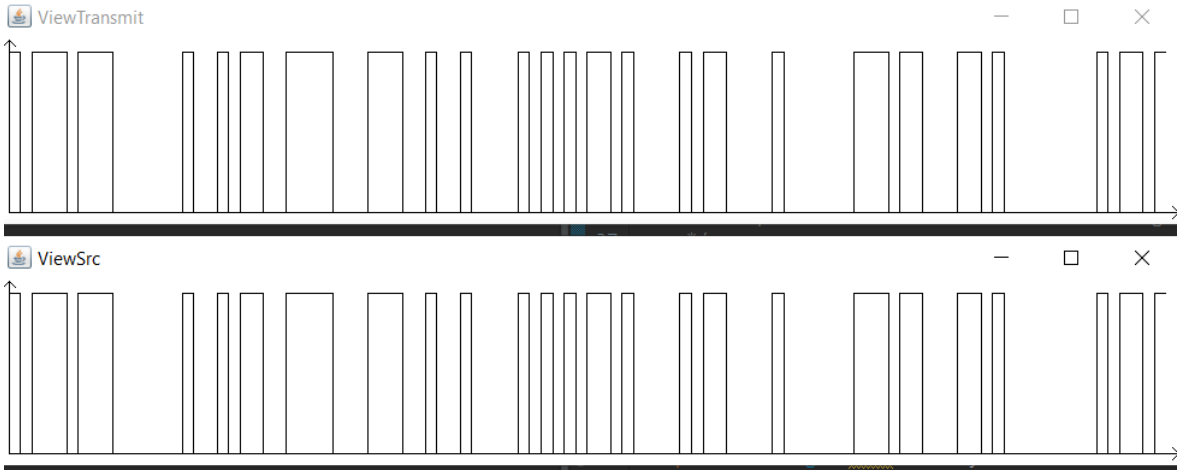

Figure 3 Signal aléatoire par défaut 1

Comme on pouvait s'y attendre le signal est identique en entrée et en sortie. Le TEB à 0 le confirme.

Je réalise un second lancer afin de vérifier que le signal est bien aléatoire :

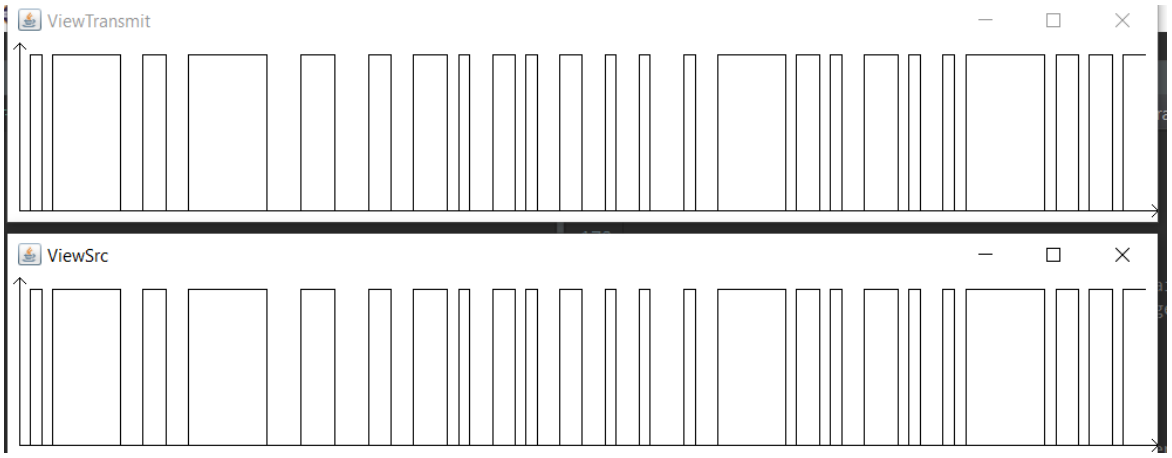

Figure 4 Signal aléatoire par défaut 2

Les 2 signaux sont une nouvelle fois identique (TEB à 0), de plus il y a bien un message différent entre le lancer 1 et le lancer 2. Ces tests sont donc concluants.

Cas avec un signal aléatoire de longueur fixée :

Je lance le Simulateur en indiquant de générer un signal de longueur 10 :

```
java | Simulateur  -s  -mess  10    =>   TEB : 0.0
```

Le message généré doit donc faire une longueur de 10 bits.

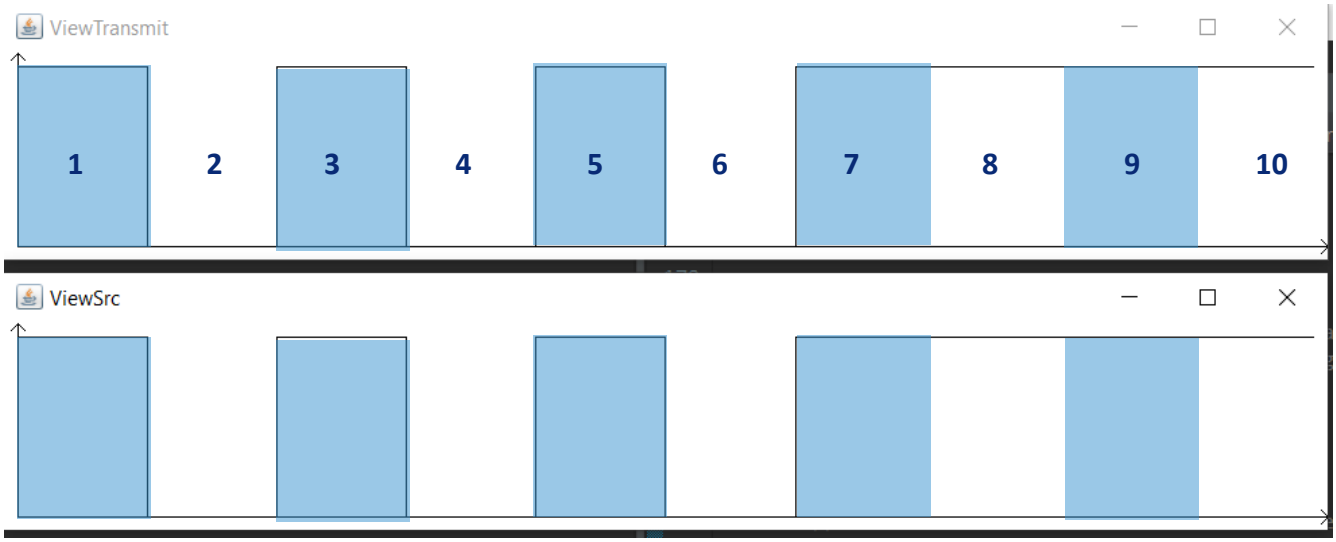

Figure 5 signal aléatoire de longueur fixée à 10

Le résultat obtenu est bien celui attendu. Le test est concluant.

Cas avec un signal fixé :

Je décide de mettre en paramètre le message fixe suivant : 10101010

```
java  Simulateur  -s  -mess  10101010    =>   TEB : 0.0
```

Le message transmis doit donc correspondre à ce message, une alternation de niveaux logiques haut et bas graphiquement.

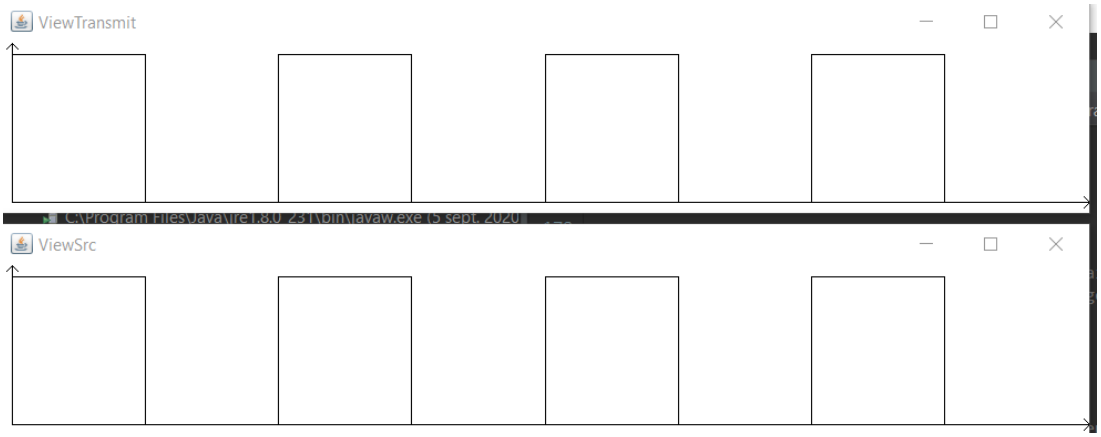

Figure 6 signal fixé sur 01010101

Le résultat obtenu est bien celui attendu. Le test est concluant.

Capitalisation du travail réalisé

C'est un projet qui est réalisé en groupe. Dans l'intérêt de chacun et pour disposer d'un système de versionnage nous utilisons les outils git et un dépôt GitHub.

Cette partie du programme a été déposée et versionnée dans un répertoire GitHub commun à tous les membres de nos groupes finaux. Ainsi nous pourront mettre en commun nos aboutissements sur cette première étape.

Conclusion

Cette première étape nous a permis d'appréhender le projet en réalisant une chaine de transmission simple dites « Back to Back ».

Au travers d'une analyse, puis en passant par le développement et pour finir sur une batterie de tests, nous sommes arrivés à mettre en place un système de communication. Il met en évidence les éléments importants d'une chaine de transmission comme la source, le transmetteur et le récepteur (appelé destination ici).

Dans l'étape 2 nous devrons implémenter un système de communication analogique non bruitée. L'approche sera certainement différente, il faudra de nouveau effectuer les étapes d'analyse avant de commencer à programmer les nouveaux blocs.

L'approche des télécommunications par la programmation est très intéressante. C'est une approche pratique qui permet de bien comprendre l'utilité et l'organisation des système de communication. De plus c'est un projet pluridisciplinaire qui fait appel à diverses connaissances et qui permet bien sûr d'en développer des nouvelles.

Table des illustrations