TP1 - Transmission de bandes de base

Table de matières :

Rappel du Sujet

Le sujet de ce premier TP est d'implanter un programme JAVA permettant de tracer le signal électrique émis lors de l'envoi d'une chaîne binaire en fonction du code choisi. On travaille avec les 4 codes suivants : NRZ, Manchester, Manchester Différentiel, et Miller.

L'interface doit prendre une chaîne binaire et le code à utiliser en paramètre et afficher le bon signal en fonction de ceux-ci.

Analyse du Sujet

Pour mettre en place un tel programme il faut construire une interface graphique pour répondre aux attentes de ce sujet :
cette interface va nécessiter un menu de choix qui ne permet qu'un seul choix pour les codes (on ne va pas tracer 2 graphes en même temps) et un endroit où écrire pour le code Binaire afin de le récupérer. Ce qui nous amènera forcément à créer un menu.

Nous aurons aussi besoin d’un espace pour dessiner des trames, et donc des fonctions pour les dessiner en respectant les différentes règles de chaque type de codes. Tout les codes fonctionnent avec des tensions positives en négatives et d’un passage de l’un à l’autre en fonction du code donc on devra indiquer le voltage sur la trame et garder une hauteur égale entre nV et -nV en fonction de 0V.
Chaque code a ensuite des contraintes spécifiques :

▼ Code NRZ

Le code NRZ utilise la tension positive ou négative pour coder les bits : le bit 1 est codé par un signal de n volts et le bit 0 par un signal de -n volts.

▼ Code de Manchester

Le code de Manchester utilise la différence de signal pour coder les bits au lieu de la tension : le bit 1 est codé par un passage du positif au négatif et inversement pour le bit 0.

▼ Code de Manchester Différentiel

Le code de Manchester Différentiel utilise la différence de signal quelque soit le bit codé mais ajoute un changement de tension en début d’horloge pour coder le bit 0.

▼ Code de Miller

Le code de Miller est caractérisé par la présence de transitions en milieu d’horloge sur le codage du bit 1 et l’absence de transition sur le codage du bit 0 sauf en cas de chaînes de 0, on aura alors une inversion de la tension en milieu d’horloge.

On devra aussi gérer les erreurs, on ne pourra pas générer de trame si aucun type n’est sélectionné et si aucun code binaire n’est donné, et on ne peut pas non plus donner autre chose qu’un code binaire.

Choix techniques effectués

Dans la partie précédente on a déterminé qu’en résumé il fallait :

- Un menu avec :
  - une sélection de type de transmission qui désélectionne les autres quand on en choisi un.

    - un espace pour donner le code binaire qui empêche de valider autre chose que le binaire
- Un espace de dessin :
    - Qui dessine une seule trame à la fois
    - Qui respecte les conventions de chaque type de trames.

J’ai décidé de séparer ma fenêtre de cette manière pour créer mon interface :

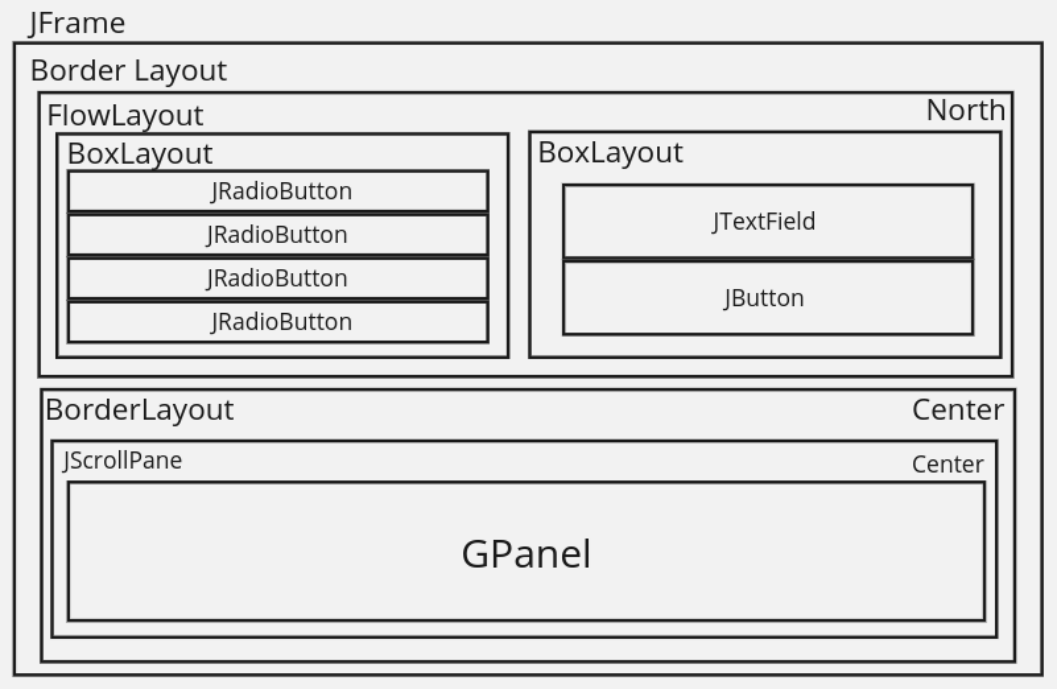

Fig 1 - Schéma de la disposition des composants dans la fenêtre

Le menu est composé de 2 JPanel avec chacun BoxLayout eux-même contenus dans un JPanel avec un FlowLayout, comme ça le menu reste centré au redimensionnement de la fenêtre. L’espace de dessin est juste une classe fille de JPanel nommée GPanel qui dessine les Trames en fonction des paramètres donnés. Ce GPanel est contenu dans un JScrollPane qui est lui même dans un autre JPanel. Ces 2 JPanel sont mis dans le ContentPane de la fenêtre qui est un BorderLayout, le menu est mis dans la zone Nord et la zone de dessin dans la zone centrale.

Le premier menu (à gauche) est composé de 4 boutons radios synchronisés par un ButtonGroup, j’ai choisi de faire ça en dehors d’une barre de menu pour plus de lisibilité. Le deuxième menu (à droite) est composé d’un champ de texte et d’un bouton pour valider.

Dans la zone de dessin on aura les graduations, les bits et l’entête qui se dessinent en noir quelque soit la trame choisie et les lignes de la trame ont ajoutées en rouge

par dessus.

Résultats et tests

Quand on lance le programme on obtient ceci :

Fig 2 - Capture d'écran du programme a son lancement

Si on ne sélectionne pas de type de code, on a un dialogue d'erreur :

Fig 3 - Capture d'écran du programme avec dialogue d'erreur sur le type du code

Si on n'écrit rien dans le champ de texte, on a un dialogue d'erreur

Fig 4 - Capture d'écran du programme avec dialogue d'erreur sur champ de texte vide

Si on écrit autre chose que du binaire dans le champ de texte, on a un dialogue d'erreur :

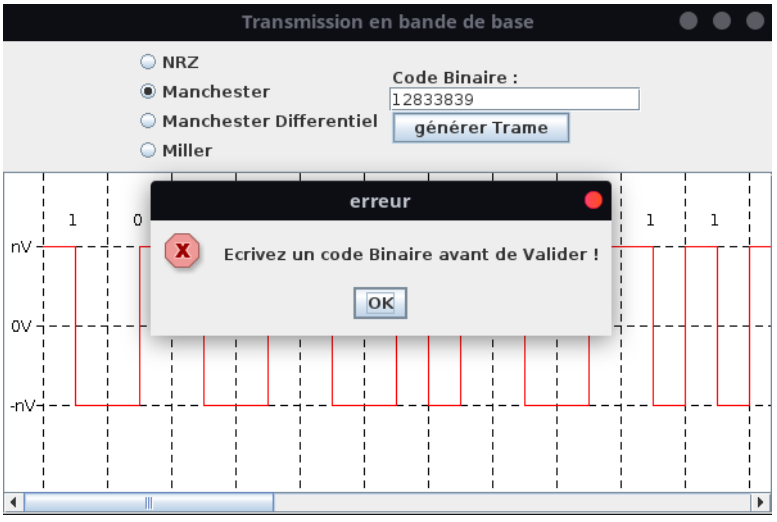

Fig 5 - Capture d'écran du programme avec dialogue d'erreur sur chanp de texte invalide

▼ Tests avec le code binaire 1101000101

On vérifie avec le code donné en TD pour être sûr d'avoir le bon tracé.

▼ NRZ

Dans le TD on avait :

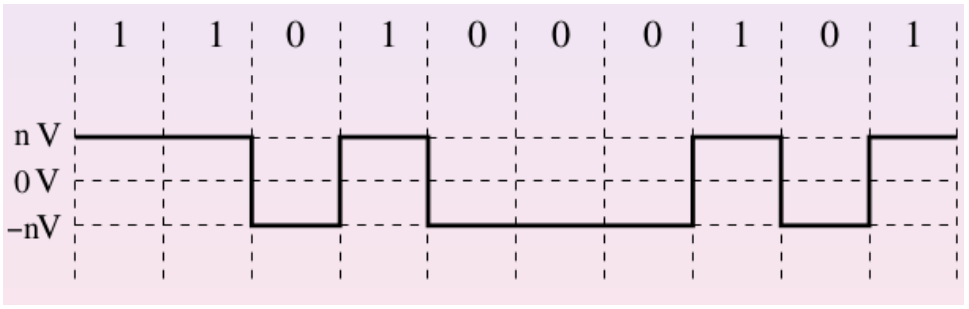

Fig 6 - Trame NRZ du code 1101000101

Et le programme nous donne :

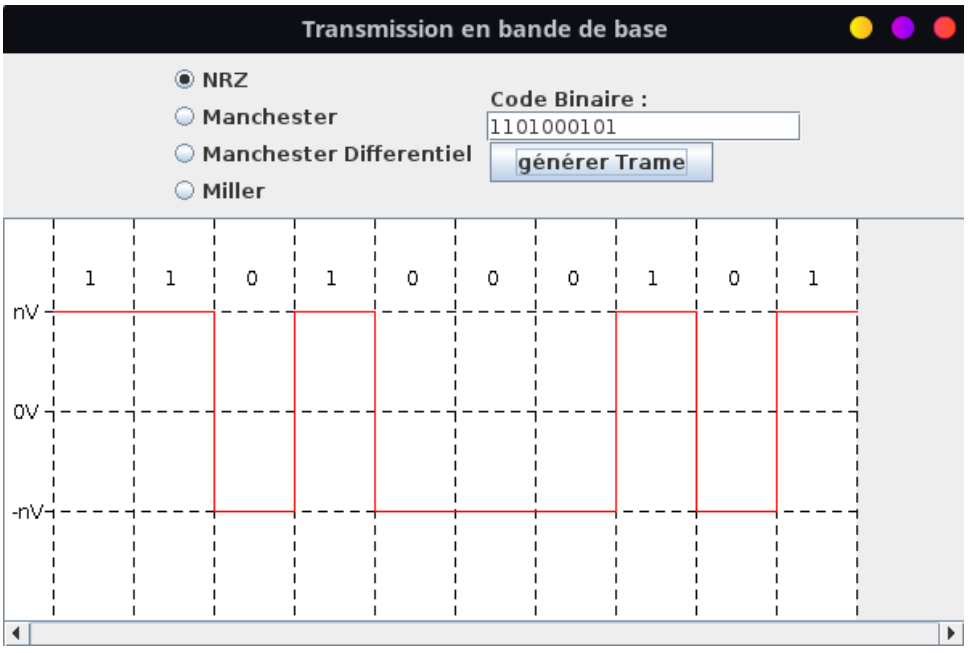

Fig 7 - Trame NRZ du code 1101000101 dans le programme

▼ Manchester

Dans le TD on avait :

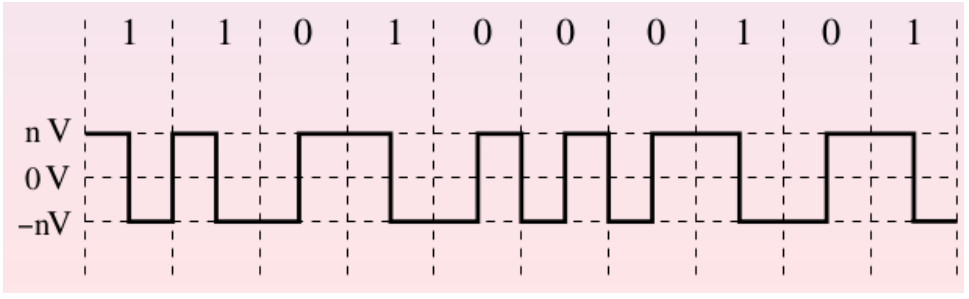

Fig 8 - Trame Manchester du code 1101000101

Et le programme nous donne :

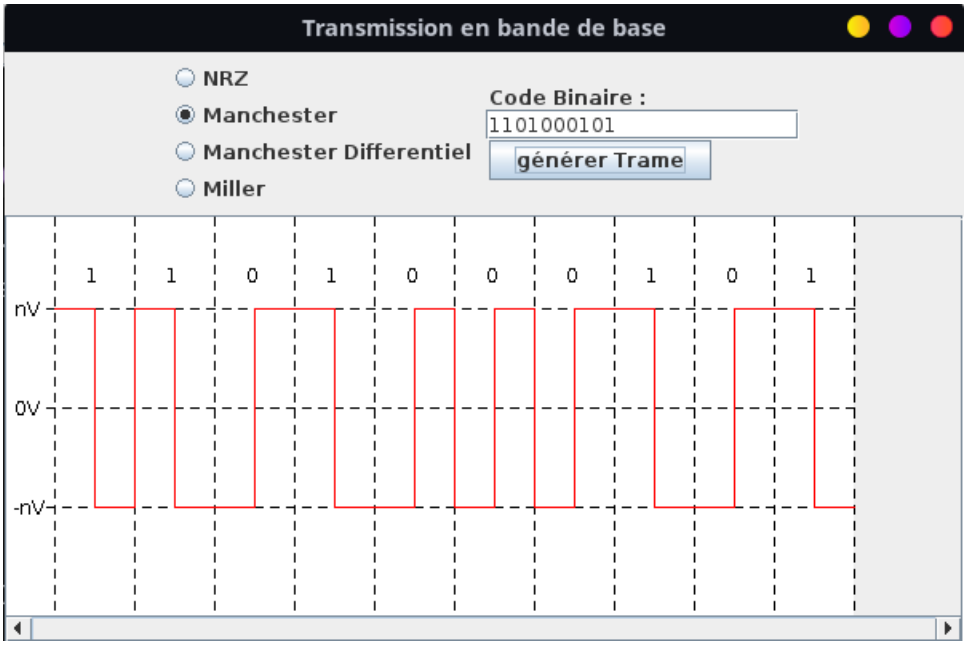

Fig 9 - Trame Manchester du code 1101000101 dans le programme

▼ Manchester Différentiel

Dans le TD on avait :

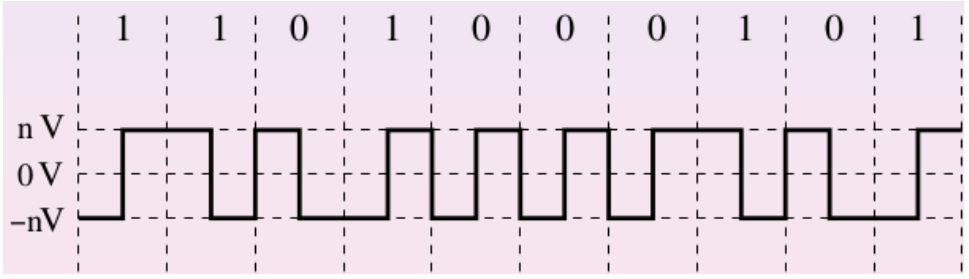

Fig 10 - Trame Manchester Différentiel du code 1101000101

Et le programme nous donne :

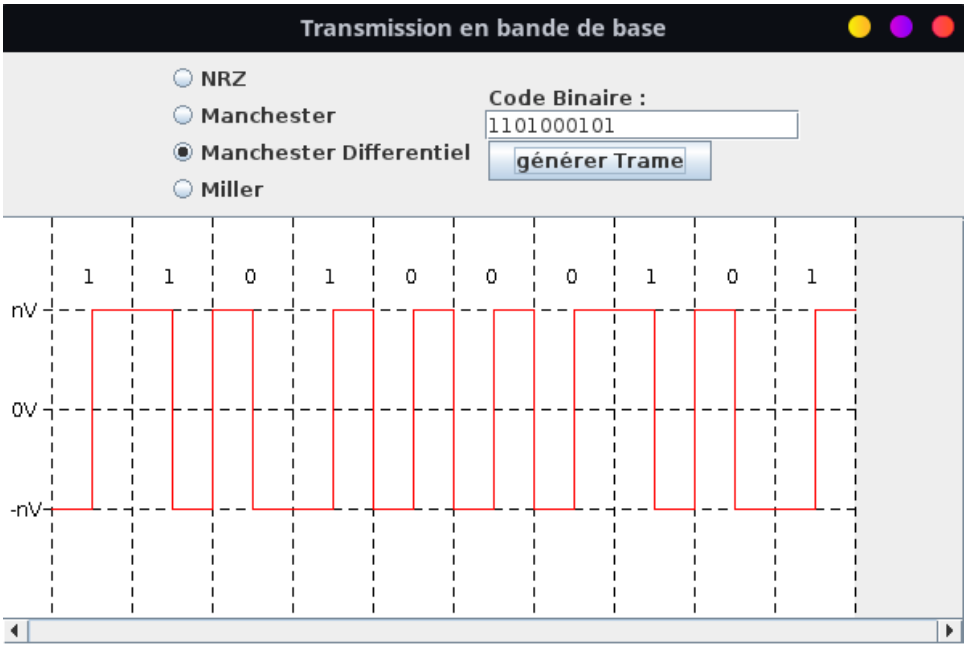

Fig 11 - Trame Manchester Différentiel du code 1101000101 dans le programme

▼ Miller

Dans le TD on avait :

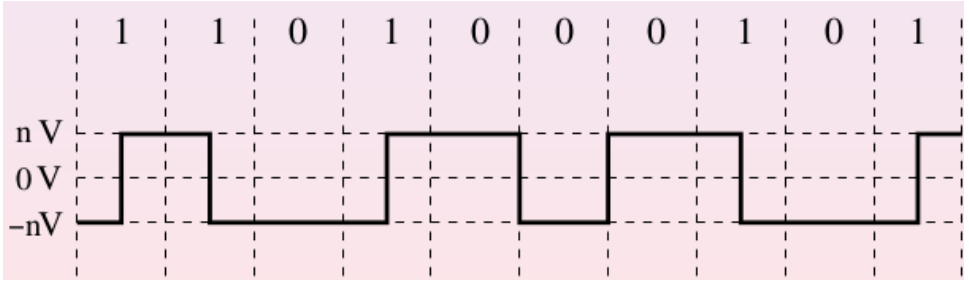

Fig 12 - Trame Miller du code 1101000101

Et le programme nous donne :

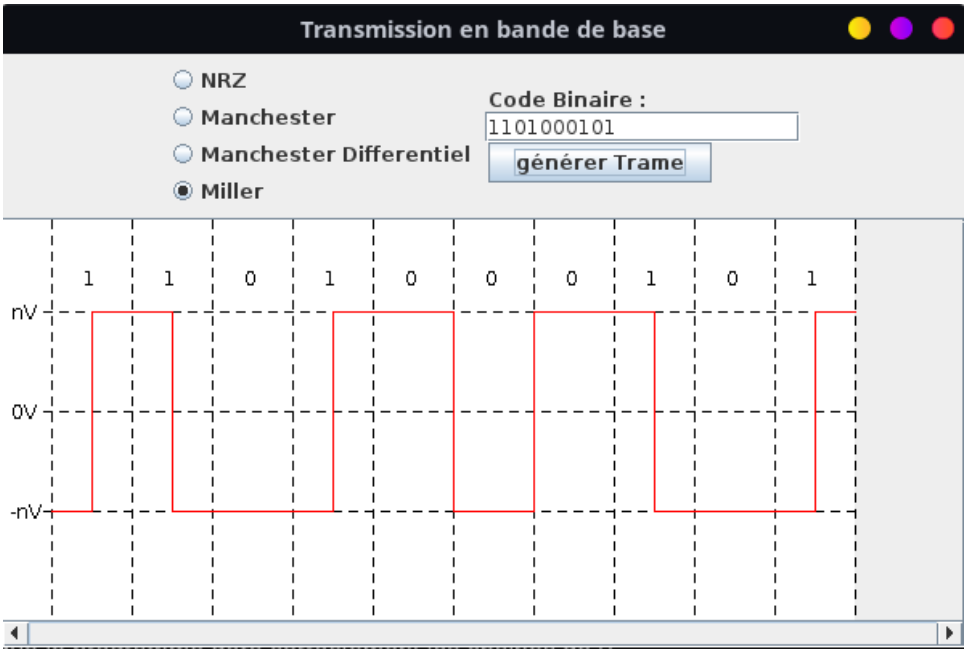

Fig 13 - Trame Miller du code 1101000101 dans le programme

▼ Tests avec le code binaire 0000000 :

Ce test sert juste à vérifier que le programme gère correctement les chaines de 0 (il ne servirait à rien autrement).

▼ NRZ

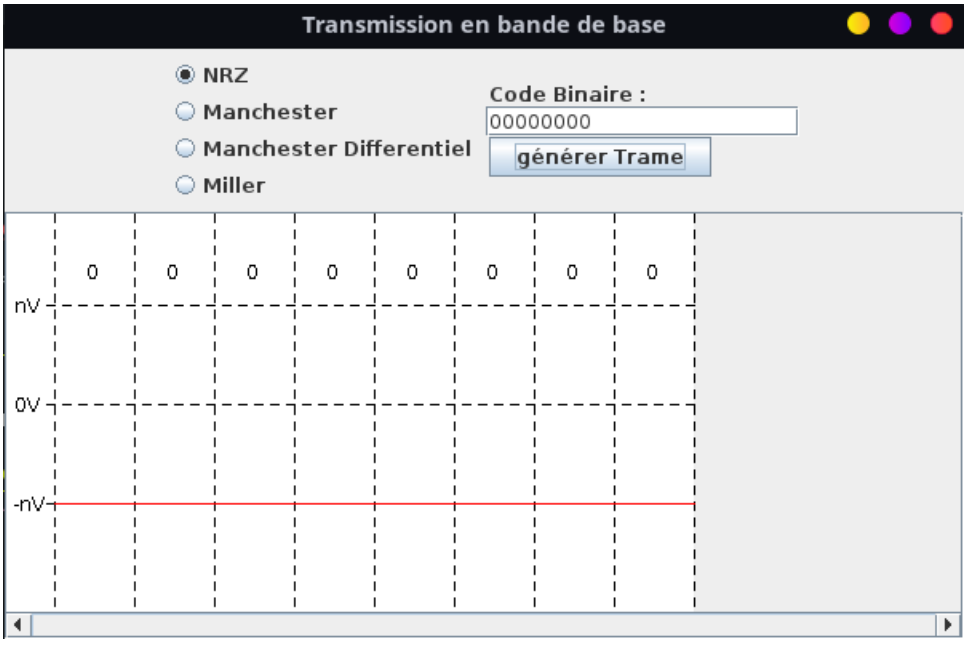

Fig 14 - Trame NRZ du code 00000000

▼ Manchester

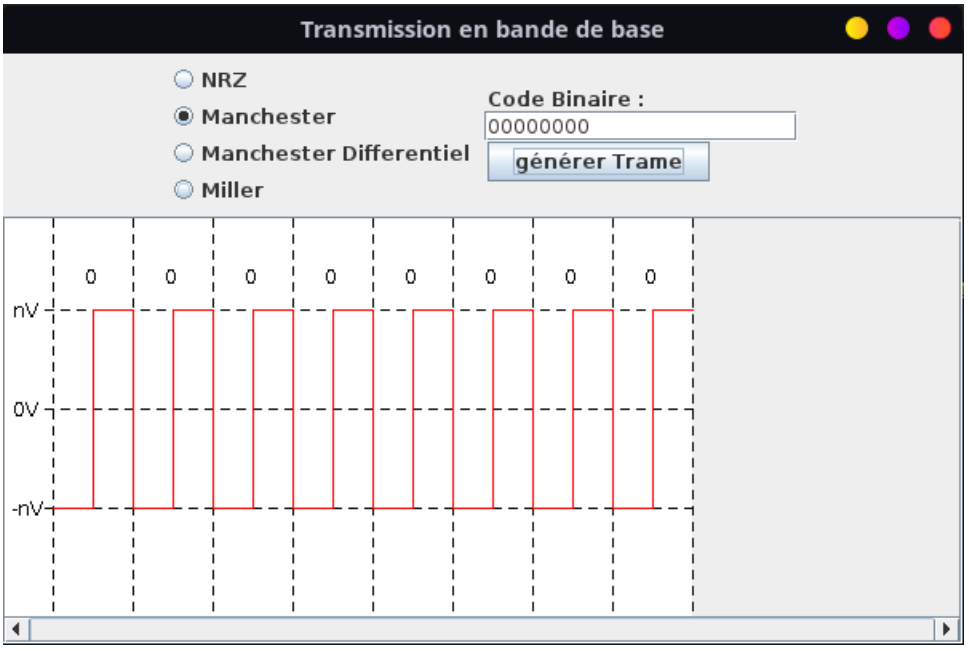

Fig 15 - Trame Manchester du code 00000000

▼ Manchester Différentiel

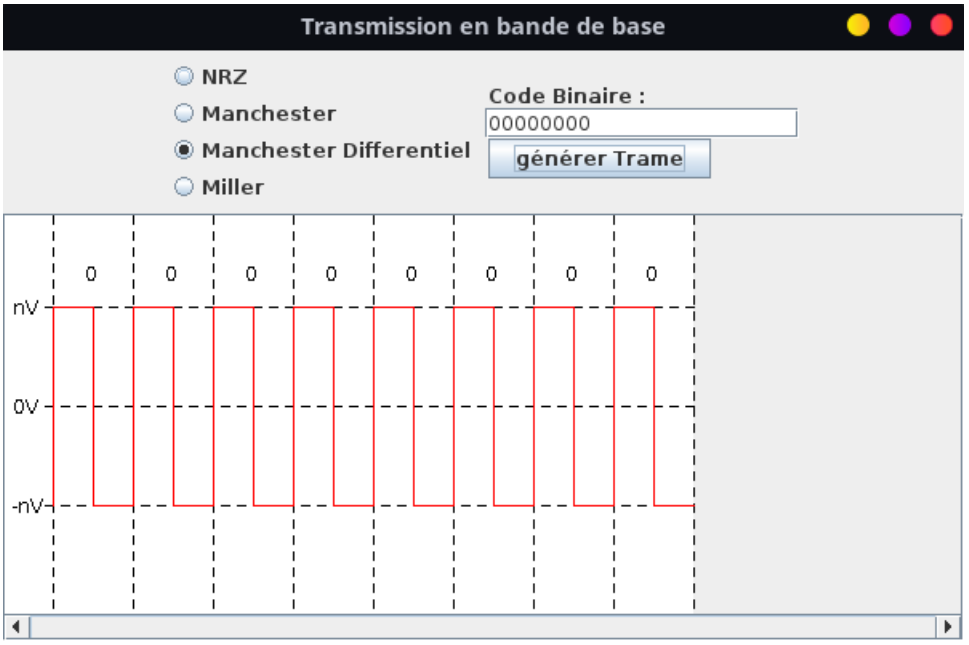

Fig 16 - Trame Manchester Différentiel du code 00000000

▼ Miller

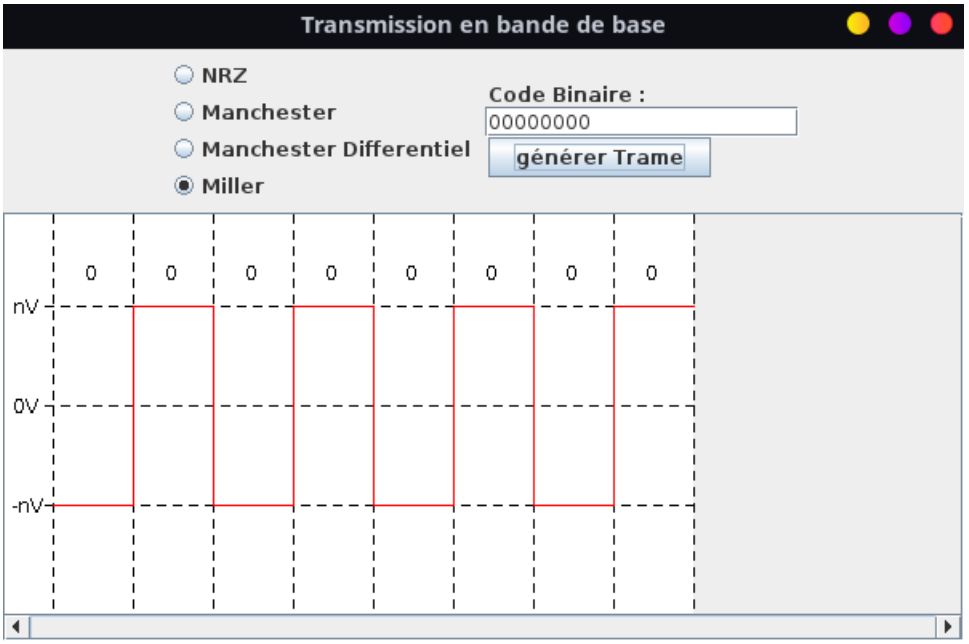

Fig 17 - Trame Miller du code 00000000

▼ Tests avec le code binaire 11111111 :

Ce test sert à vérifier que le programme gère correctement les chaines de 1

▼ NRZ

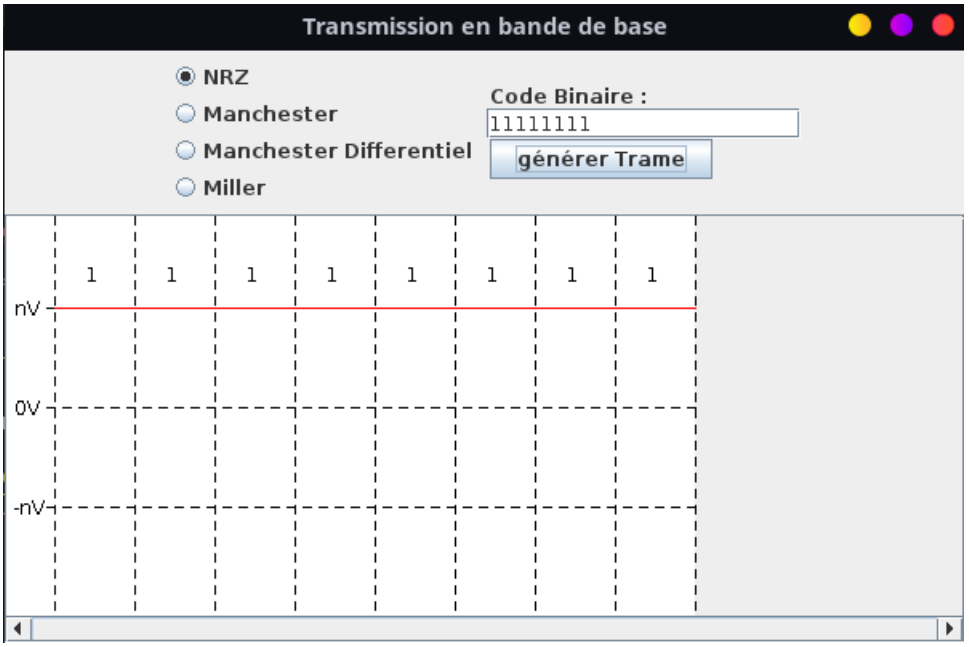

Fig 18 - Trame NRZ du code 11111111

▼ Manchester

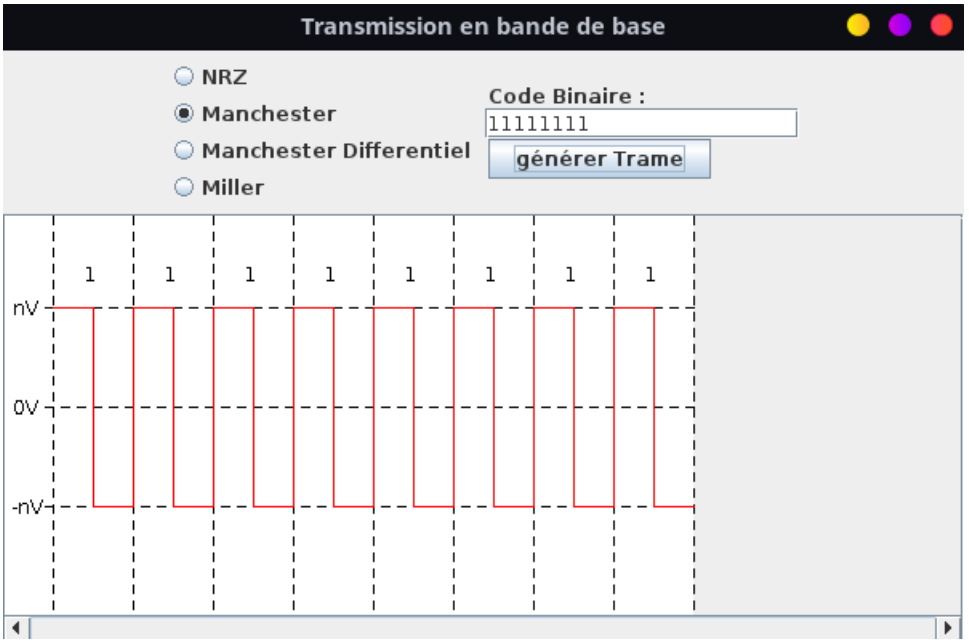

Fig 19 - Trame Manchester du code 11111111

▼ Manchester Différentiel

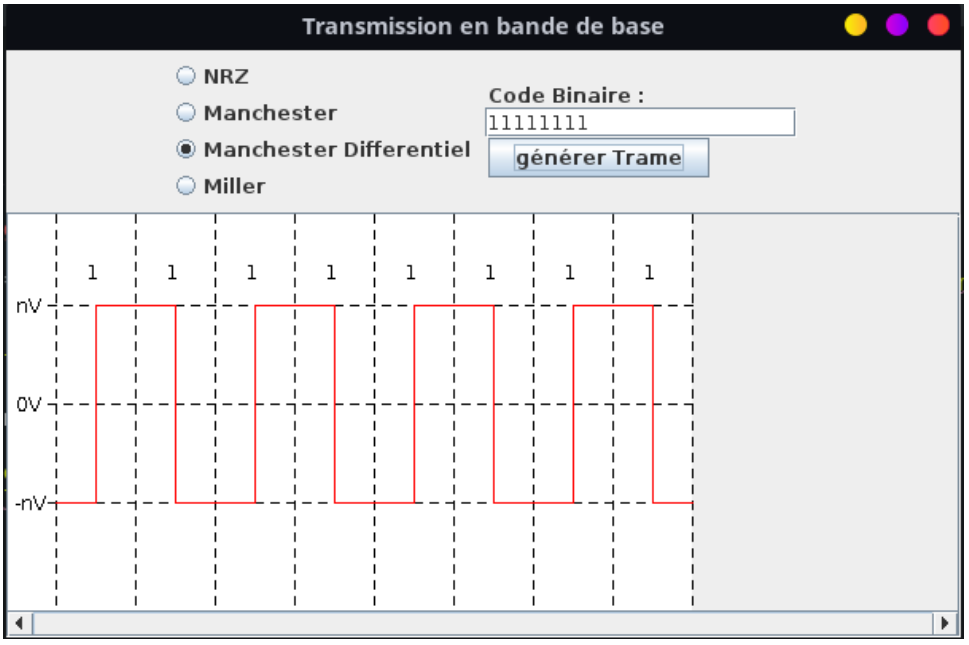

Fig 20 - Trame Manchester Différentiel du code 11111111

▼ Miller

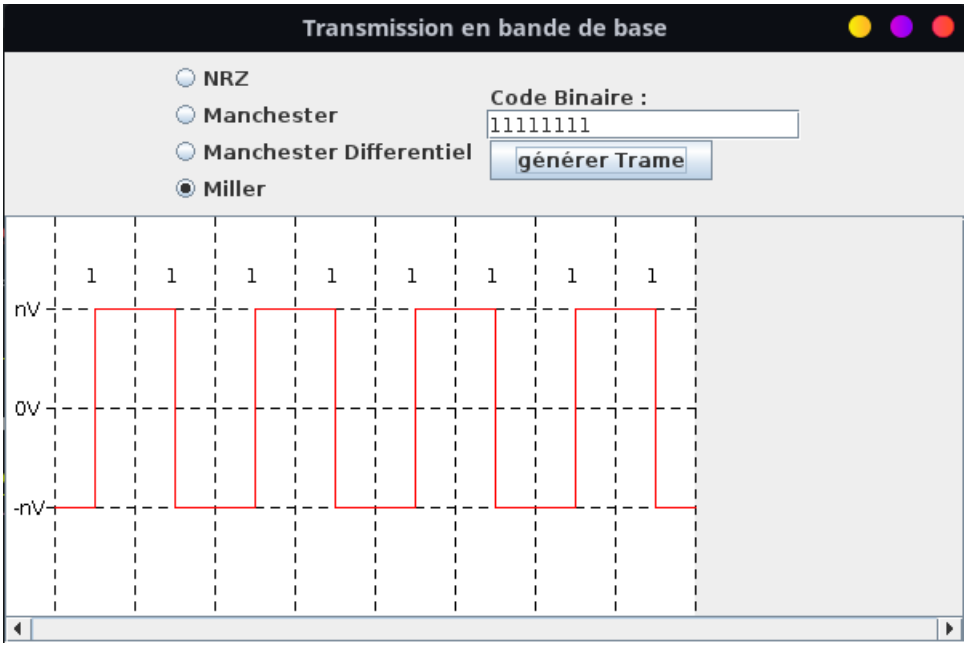

Fig 21 - Trame Miller du code 11111111

▼ Test avec le code binaire 10101001011110010010101001010010 :

Ce teste sert à vérifier comment le programme gère les chaines plus longues.

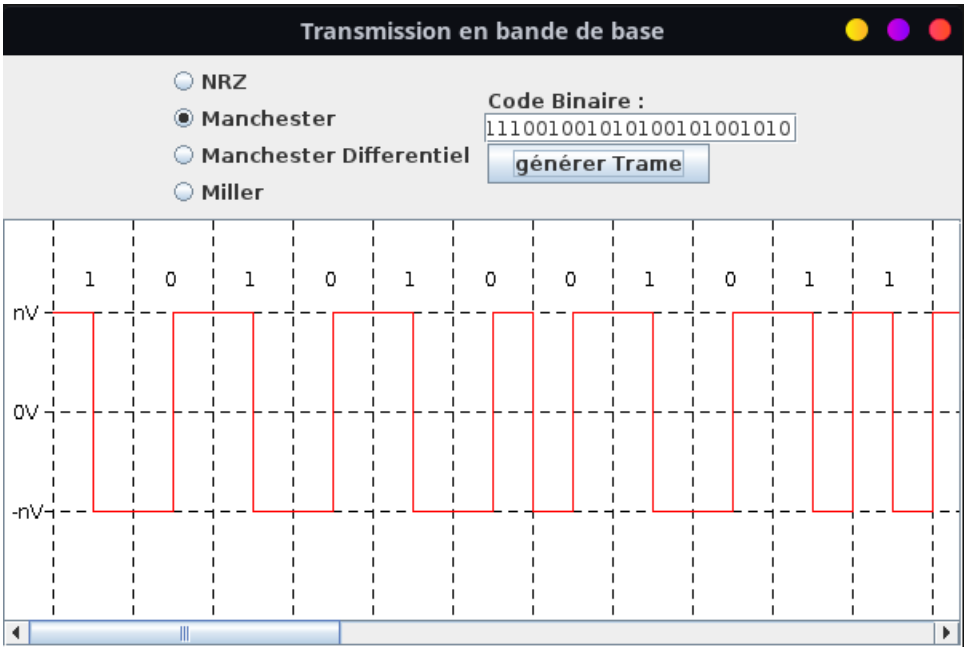

Fig 22 - Trame avec une barre de défilement qui montre une bonne gestion des codes longs

Tout les tests semblent concluants, on n'a aucune erreur et les trames sont justes.

Conclusion

Ce programme fonctionne parfaitement mais il mériterait d'être optimisé notamment au niveau des fonctions pour tracer les graphiques qui sont très redondantes, ce qui n'a pas été fait par manque de temps.

Je n'ai pas trouvé le sujet difficile, je ne savais juste pas encore comment utiliser les outils d'interface et de dessin de Java avant de commencer le TP et j'ai donc eu un peu de mal à trouver comment fonctionnait chaque Classe utilisée.

On pourrait y ajouter quelques fonctionnalités intéressantes pour aller plus loin :

- le type de code NRZI puisque il suffirait d'inverser celui du NRZ pour l'obtenir.
- on pourrait ajouter une fonctionnalité qui permet d'enregistrer l'image.
- on pourrait aussi ajouter un sens de début pour savoir si le bit précédent était un 1 ou un 0 et dessiner la trame en fonction de ça.
- Il y a aussi une possibilité de modifier le programme pour tracer plusieurs trames sur un seul graphe, ou encore d'ajouter un bouton pour réinitialiser la zone de dessin.