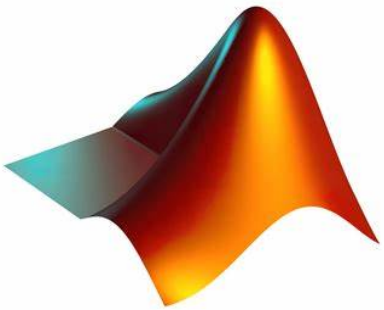
SAE301 METTRE EN OEUVRE UN SYSTÈME DE TRANSMISSION



SOMMAIRE :

Introduction

Ce compte rendu expose les travaux réalisés dans le cadre de la SAE 3.01, intitulée 'Mise en œuvre d'un système de transmission'.

Tout d'abord, une chaîne de transmission regroupe l'ensemble des dispositifs permettant le transfert d'informations, se composant de quatre éléments essentiels : une source, un émetteur, un récepteur et un destinataire.

L'objectif principal de cette SAE est de concevoir un dispositif de transmission à trois caractères, comprenant un émetteur, un récepteur et une trame, et ce, en exploitant l'ADALM-Pluto à travers l'environnement Simulink du logiciel Matlab.

La SAE a été structurée en plusieurs étapes. Initialement, nous avons planifié la conception de la trame, de l'émetteur et du récepteur, en nous appuyant sur divers documents accessibles à la fois sur le site Matlab et sur Internet. Par la suite, nous avons entrepris la construction des blocs constituant l'émetteur-récepteur ainsi que de la trame. Et pour finir nous avons effectué des tests de bon fonctionnement de notre système.

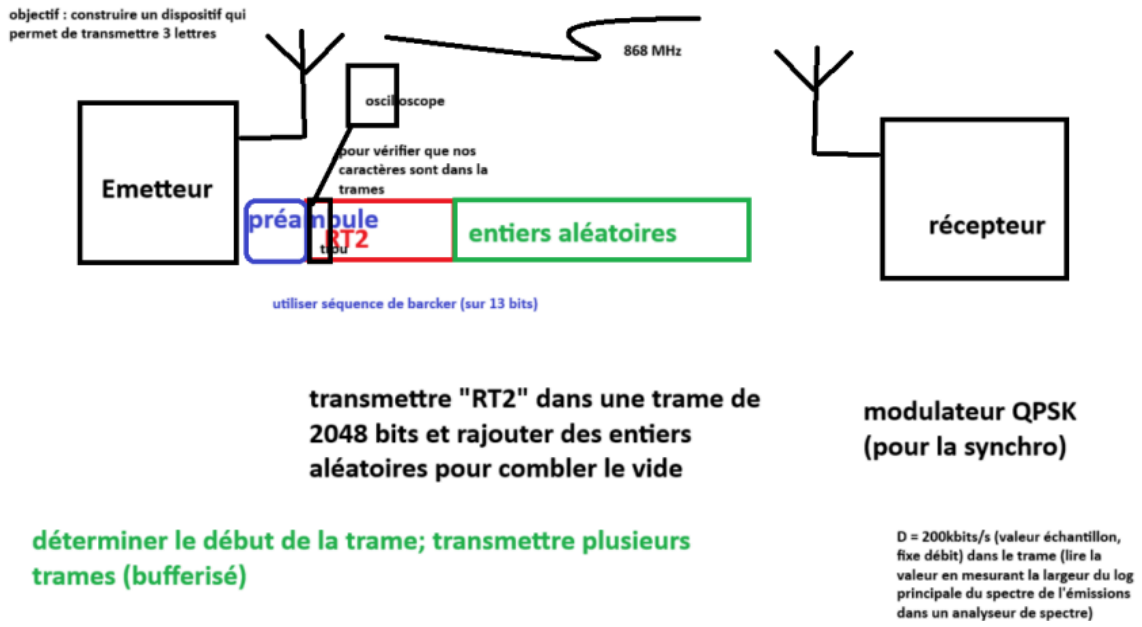
I. Déroulement du projet

1. Architecture initiale

Nous avons mis en place une première architecture dans laquelle nous avons créé une trame de 2048 bits. Celle-ci est composée d'un préambule, d'un trou, du message et d'entiers aléatoires.
Pour le préambule nous avions utilisé une séquence de Barker sur 13 bits que nous avions doublé afin d'avoir des chiffres pairs. Ensuite, avant le message nous avions rajouté un trou de 16 bits afin de vérifier que le message était transmis. Puis, la partie message était composée des caractères "RT2", ce qui faisait 24 bits. Enfin il y avait des entiers aléatoires qui servaient à compléter le vide de la trame.

En ce qui concerne la transmission, nous avions utilisé un modulateur QPSK. Et en ce qui concerne la réception, nous avions utilisé un démodulateur QPSK.

Ci-dessous, un schéma de notre architecture de base :



Cependant après maintes recherches et calculs, nous n'avions pas réussi à rendre réalisable cette solution.

2. Deuxième tentative

Pour la deuxième tentative, nous nous sommes aidés des documentations Matlab (cf. Annexe).

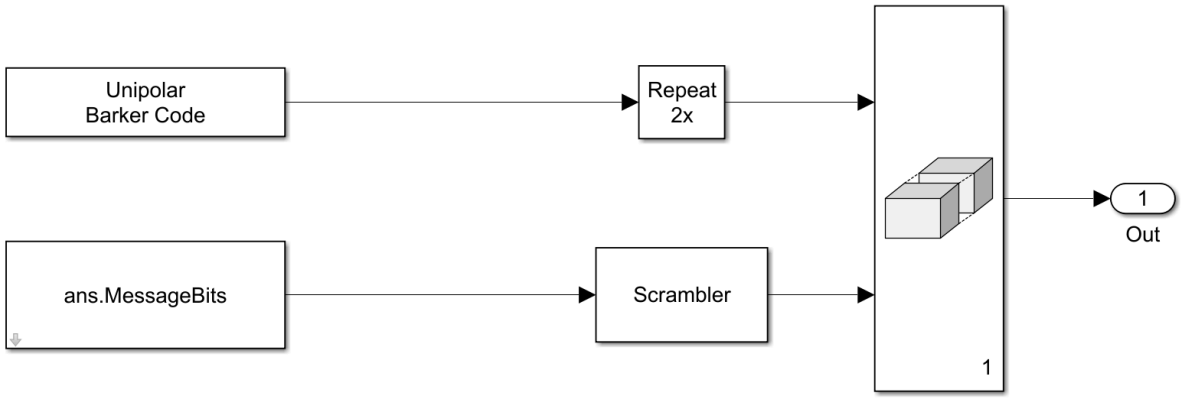
Nous avons donc compris que dans la trame, nous utilisons une séquence de Barker à 13 bits et nous la doublons pour former le préambule, ce qui fait 26 bits. La séquence de Barker est doublée dans le but de faire générer exactement 13 symboles QPSK. Ensuite le message ‘RT2’ d’une taille de 8 bits sera répété 20 fois afin de garantir une distribution équilibrée des bits de données.

Les symboles modulés sont sur-échantillonnés par deux par le filtre d'émission à cosinus surélevé avec un facteur d'affaiblissement de 0,5. Le taux de symboles du système émetteur est de 50 000 symboles par seconde, et le taux d'échantillonnage après le filtre d'émission en cosinus surélevé est de 100 000 échantillons par seconde.

II. Description de chaque bloc

Dans cette partie, nous allons nous concentrer sur la composition de notre architecture. Pour cela nous allons décrire chaque bloc qui la compose.

1. Trame



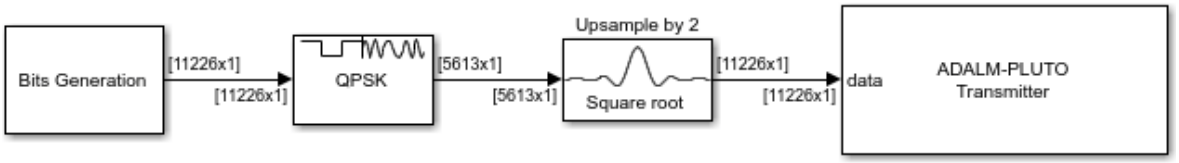
La trame est résumée par le gros bloc **Bit Generation**, qui se compose d’un groupe de blocs fournis par Simulink. Dans ce groupe nous retrouvons :

- Le bloc **Unipolar Barker Code**. C’est là que nous stockons les 13 bits de Barker. Ce bloc est suivi du **Repeat 2x** pour, comme nous le disions précédemment, générer exactement 13 symboles QPSK. Ce qui fait donc une entête de 26 bits

- Ensuite nous avons le bloc **MessageBits** contenant le message ‘RT2’ que nous avons codé, suivi du bloc **Scrambler** qui sert à améliorer l’estimation de la densité de transition des données et du décalage de fréquence.

Et ces 2 chaînes de blocs sont reliées par un bloc **Matrix concatenate**.

2. Emetteur



L'émetteur se compose donc de la trame précédemment décrite et des blocs suivants :

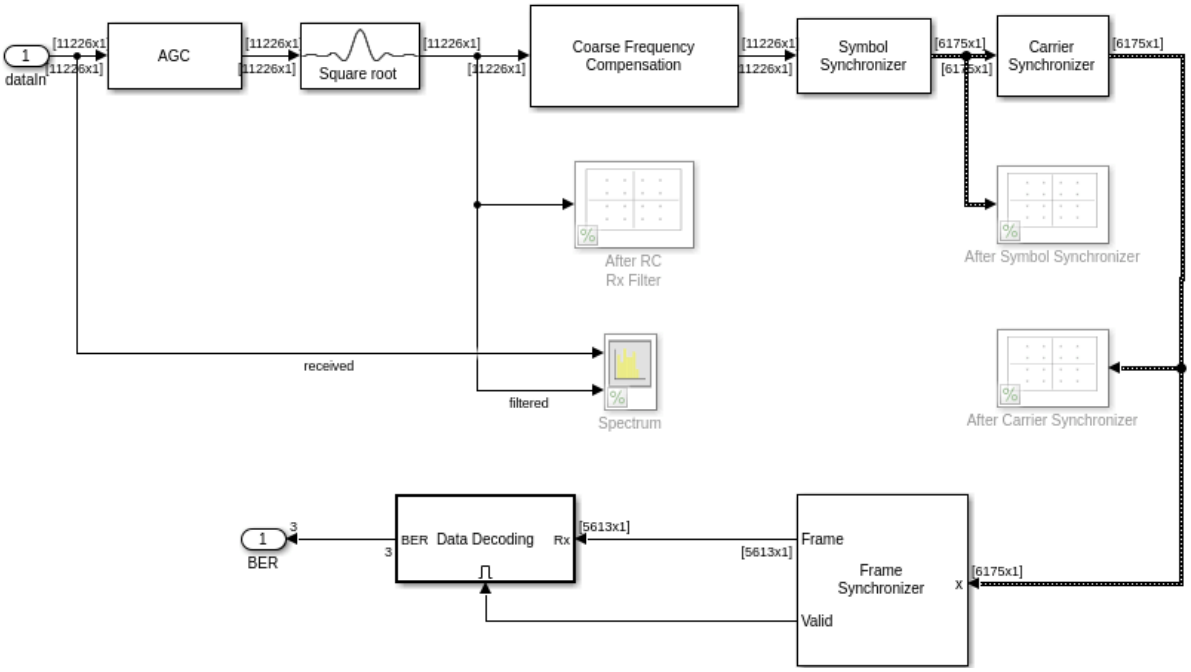
- **QPSK Modulator Baseband :** Le modulateur QPSK Modulator baseband prend les bits générés et les transforme en signaux modulés.

- **Square Root :** Le bloc Square Root est utilisé pour concevoir des filtres de mise en forme de signal ou des filtres de réception, préparant ainsi les signaux pour la transmission ou la réception, notamment pour les signaux modulés.

- **ADALM-PLUTO Transmitter :** Ce bloc permet la transmission à travers le boîtier branché à notre ordinateur.

3. Récepteur

Le récepteur est un peu plus compliqué à faire car il faut qu'il se synchronise avec l'émetteur de sorte à bien recevoir le message transmis.



- **AGC (Automatic Gain Control) :** C'est un système qui s'assure que le signal que nous recevons n'est ni trop fort ni trop faible. Il ajuste la puissance du signal pour que les appareils qui l'utilisent puissent le comprendre correctement.

- **Square root :** Comme dans l'émission, ce bloc est utilisé pour concevoir un filtre spécial qui s'assure que le signal que nous recevons est propre et facile à comprendre.

- **Coarse Frequency Compensation :** C'est un système qui corrige légèrement les erreurs de fréquence dans le signal que nous recevons. Parfois, même après cette correction, il peut rester une petite erreur, mais un autre système la corrige encore mieux.

- **Symbol Synchronizer :** Ce bloc permet d'assurer que les différents symboles dans le signal sont bien compris et arrivent au bon moment.

- **Carrier Synchronizer :** C'est un système qui corrige précisément les erreurs de fréquence et de phase dans le signal que nous recevons pour qu'il soit clair et facile à comprendre.

- **Frame Synchronizer :** C'est un système qui repère le début et la fin d'une "trame" ou d'un groupe de signaux pour être sûr que les données sont lues correctement.

- **Data Decoding :** C'est un système qui prend les informations du signal et les transforme en messages que nous pouvons lire, comme des textes ou des images.

III. Phase de test

Pour effectuer les tests de bon fonctionnement, chaque membre de notre binôme s'est muni d'un boîtier ADALM-Pluto. Depuis nos PC respectifs, nous avons lancé, pour l'un l'émetteur, et pour l'autre le récepteur.

Dans le 'View diagnostics' de simulink, dans la partie réceptrice, nous voyons le message transmis :

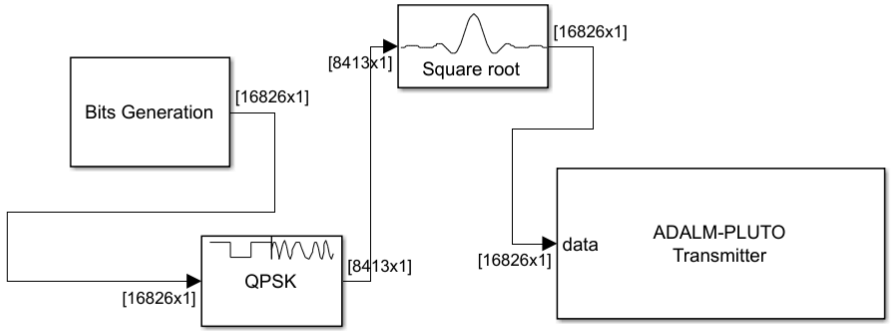
Résultat de la transmission

```
RT2 007
RT2 008
RT2 009
RT2 010
RT2 011
RT2 012
RT2 013
RT2 014
RT2 015
RT2 016
RT2 017
```

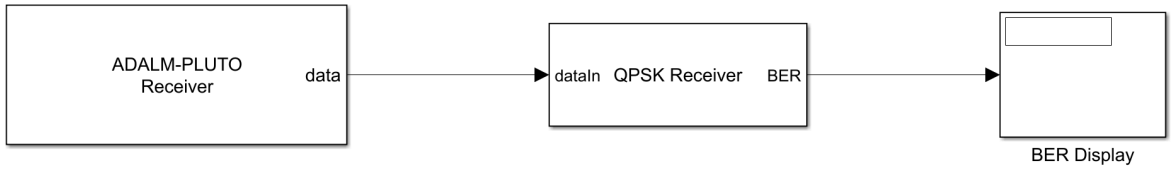
Comme nous pouvons le constater, le message transmis est RT2 avec le nombre de fois que celui-ci est transmis.

IV. Architecture final

Transmetteur



Récepteur



Conclusion

Nous pouvons dire que la construction de cette solution n'a pas été des plus simples.

Nous avons rencontré plusieurs difficultés. Tout d'abord au niveau du matériel de par les lourdes demandes de ressources du logiciel Matlab, nous avons fait face à pas mal de latence. Ensuite, étant un groupe composé d'un FI et d'une FA, il a été difficile de synchroniser notre travail car la FA ne pouvait pas se déplacer à l'IUT durant sa période d'entreprise. Cela ne l'a pas empêché de faire des recherches et d'effectuer des tests durant ses heures de temps libre.

Nos recherches individuelles ont permis de faire des tests concrets, avec le matériel, lors de nos heures de projet et de TPs. Nous nous sommes aussi aidé de l'exemple de Matlab (cf Annexe) pour construire notre solution finale.

En conclusion, nous avons créé une architecture composée d'une trame transmise et réceptionnée par l'ADALM-Pluto. Nous avons donc transmis une trame contenant le message 'RT2'. Et nous avons utilisé le moyen de modulation QPSK pour la transmission et le démodulateur QPSK pour la réception.

Annexe

Exemple QPSK Matlab :

https://fr.mathworks.com/help/supportpkg/plutoradio/ug/qpsk-transmitter-with-adalm-pluto-radio-1.html

https://fr.mathworks.com/help/supportpkg/plutoradio/ug/qpsk-receiver-with-adalm-pluto-radio-1.html

https://fr.mathworks.com/help/comm/ug/qpsk-transmitter-and-receiver-in-simulink.html