LAJUGIE Rodolphe
CORNATON Maxime
GUZELIAN Raphaël
BARRIQUAND Valentin

3ETI

Compte-rendu Projet modulation SSL

1. Cahier des charges

L'objectif est de transmettre simultanément sur un même canal (par voie filaire) 4 messages vocaux différents émis chacun par une source (ordinateur) et reçu par le récepteur du groupe correspondant (ordinateur). De plus la bande passante est limitée à 96kHz.

2. Mise en œuvre

2.1. Bureau d'études

Tout d'abord nous procédons par modulation d'amplitude sans porteuse.

On va d'abord analyser notre système afin de comprendre le phénomène de modulation démodulation et le choix des fréquences que nous utiliserons pour notre travail.

Notre signal à transmettre sera noté $s(t)$ et la porteuse sera sous la forme d'un cosinus telle que $p(t) = \cos(2 \times \pi \times \nu_0 \times t)$, ν_0 étant une fréquence choisis plus tard. Le signal modulé que l'on notera $s_{mod} = s(t) \times p(t)$

La transformée de Fourier de ce dernier signal nous donne :

$$\mathcal{F}(s_{mod}(t))(f) = \mathcal{F}(s(t).p(t))(f) = (S * P)(f) = \left(S * \frac{1}{2}(\delta_{\nu_0} + \delta_{-\nu_0})\right)(f)$$
$$= \frac{1}{2}.(S(f + \nu_0) + S(f - \nu_0))$$

Ensuite lors de la modulation on multiplie à nouveau ce signal par la même porteuse puis on applique un filtre passe-bas pour ne garder que le centre du motif de base (Cf schéma 1).

Ce signal démodulé mais non filtré noté $x(t)$ dont on va calculer la transformée de Fourier :

$$\mathcal{F}(x(t))(f) = \mathcal{F}(s(t).p^2(t))(f) = \left(\frac{1}{2}.(S(f + \nu_0) + S(f - \nu_0))\right) * \left(\frac{1}{2}(\delta_{-\nu_0} + \delta_{\nu_0})\right)$$
$$= \frac{1}{4}(S(f - 2\nu_0) + S(f + 2\nu_0) + 2S(f))$$

On fait donc le tracé des fonctions de f en bleu et rouge.

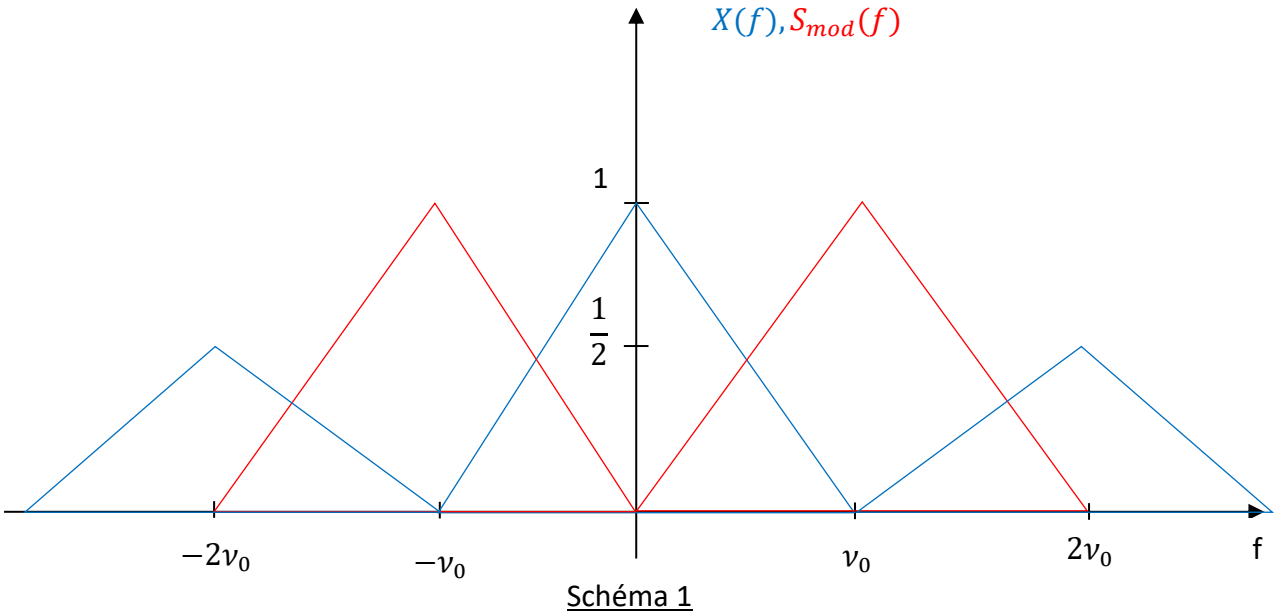


Schéma 1

Pour la suite nous devons choisir la fréquence ν_0 de la porteuse de telle sorte que le signal ne sorte pas de la bande passante. Etant donné que nous avons notre fréquence d'échantillonnage $F_e = 96kHz$ donc d'après le critère de Shannon-Nyquist notre bande passante doit se limiter à 48kHz. De plus comme nous le voyons sur le schéma 1 lors de la démodulation (fonction bleu) le signal est étalé jusqu'à $2\nu_0$ donc on se limitera finalement à 24 kHz afin de ne pas rendre impossible la reconstruction du signal. Enfin on divise par trois cette bande passante (3 groupes) pour obtenir 3 fréquences pour notre porteuse. Notre groupe travaillera donc entre 16kHz et 24kHz et notre porteuse sera à la fréquence $\nu_{03} = 20kHz$. Nous avons aussi choisi arbitrairement la durée du message à 5 secondes.

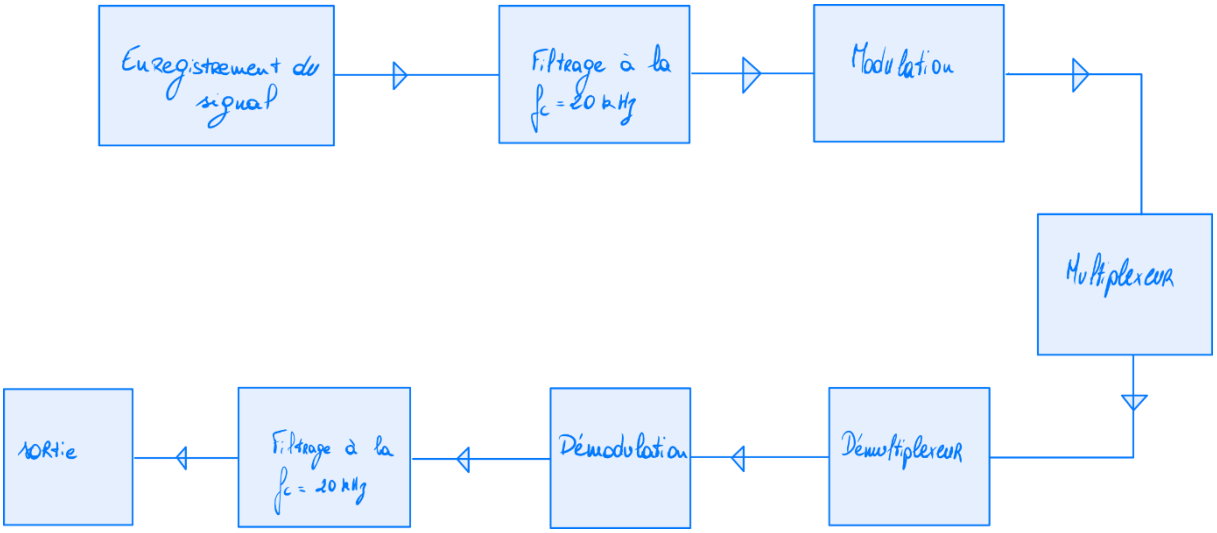


Schéma fonctionnel du principe de fonctionnement de la modulation démodulation

2.2. Programmation

Sur la base du cahier des charges, il nous faut programmer sous Matlab, deux fonctions qui réaliseront les opérations suivantes :

TRANSMISSION : Enregistrement du message. Filtrages éventuels nécessaires. Modulation. Emission sur le canal.

RECEPTION : Réception du signal de mélange. Filtrages éventuels nécessaires. Démodulation. Ecoute du message reconstruit.

Dans un premier temps, nous avons testé les fonctions sans passer par les (dé-)multiplexeurs analogiques, en connectant directement les postes SOURCE et RECEPTION.

Nous avons ensuite

transmission.m

Les objectifs de ce fichier sont les suivants :

1. Enregistrer un signal
2. Appliquer un filtre Passe-bas à celui-ci
3. Multiplier par la porteuse
4. Sauvegarde/Emission du signal modulé

Plus en détail :

1. Le fichier appelle la fonction RecordModulation pour enregistrer un signal, d'une durée de 5s.
2. Le message enregistré est filtré à une fréquence de coupure de 20kHz pour éviter que celui-ci dépasse de notre bande de fréquence de groupe, grâce à la fonction PasseBas.
3. Le signal est ensuite modulé (modulation d'amplitude sans porteuse) en multipliant le signal par la porteuse $\cos(2.\pi.\nu_{03}.t)$.
4. Nous avons ensuite enregistré le signal (pour le comparer avec celui après démodulation) et grâce à la fonction Sound, nous avons lu en boucle le signal modulé.

demodulation.m

Les objectifs de ce fichier sont les suivants : Elle va dans un premier temps lire le fichier son contenant le signal démodulé à la fréquence d'échantillonnage de 96kHz. Le signal est ensuite multiplié par une porteuse de fréquence 20kHz. On peut alors peut alors récupérer, lire le

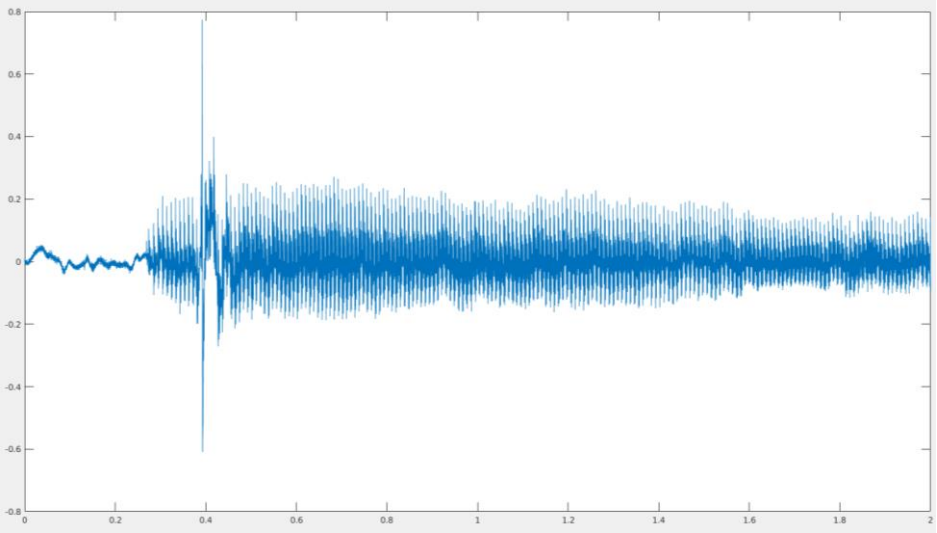
signal démodulé et afficher sa transformée de Fourier grâce à un filtre passe bas comme présenté ci-dessous.

PasseBas.m (fourni)

Prenant en entrée un signal et une fréquence de coupure, il renvoie un signal filtré à cette dernière.

RecordModulation.m (fourni)

Fonction qui permet de procéder à l'enregistrement du message et le numérise sur 16 bits. Elle reçoit en entrée la fréquence d'échantillonnage du signal, la durée en seconde du message à enregistrer. Elle retourne en sortie le nom (chaine de charactères) du fichier son (.wav) dans lequel a été enregistré le signal ; un vecteur contenant les échantillons du message enregistré ; et le vecteur temporel



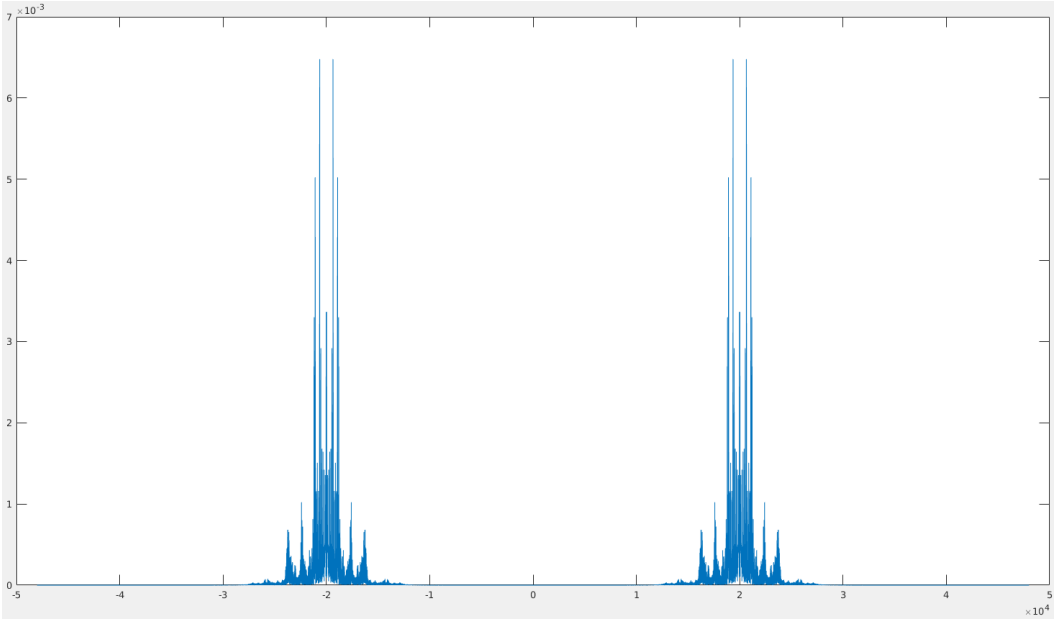
Singal en sortie de RecordModulation.m

TransFourier.m (fourni)

Fonction qui réalise la transformée de fourrier d'une fonction. Elle reçoit en entrée un vecteur S de taille N contenant les N échantillons du signal à analyser ; un vecteur de taille N contenant les instants d'échantillonnage de S. Elle fournit en sortie un premier vecteur de taille N contenant les coefficients de la transformée de Fourrier du signal S et enfin un vecteur de taille N contenant les fréquences correspondant aux coefficients de S.

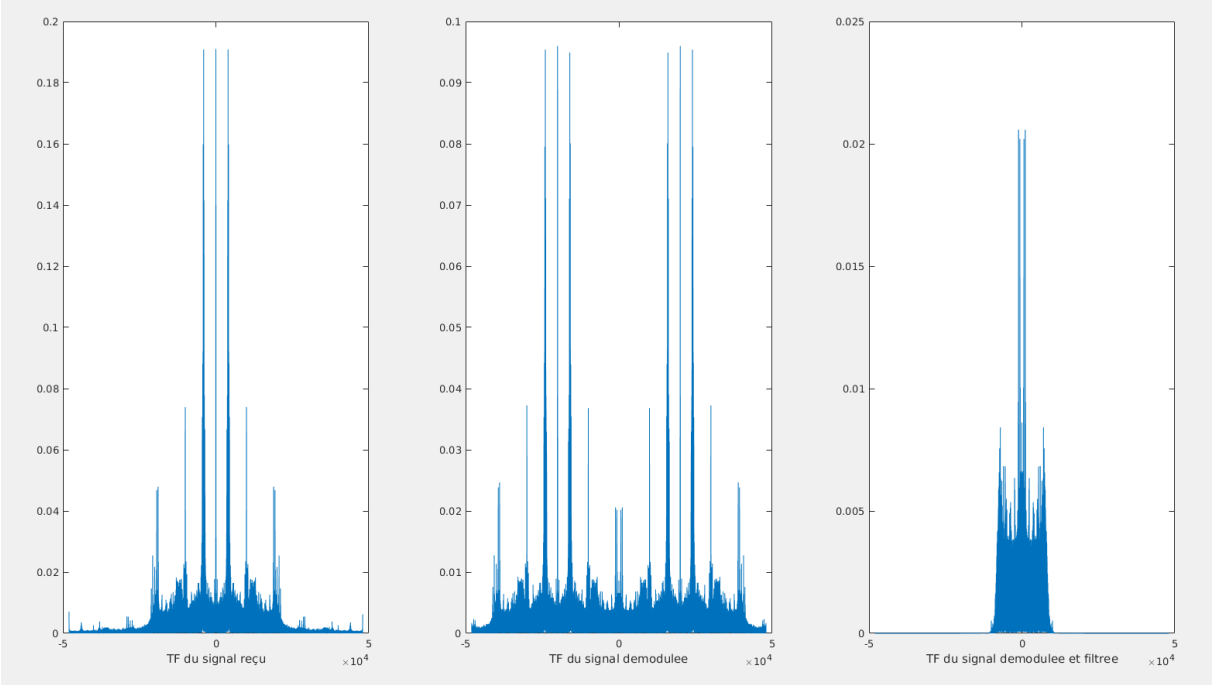
2.3 Manipulation

Ici nous avons mélangé nos signaux avec ceux des autres groupes afin de reconstruire uniquement notre signal. Nous avons calculé les transformées de Fourier à différentes étapes de la modulation et démodulation afin de vérifier si nos signaux ont un comportement correspondant à celui énoncé dans le Bureau d'étude et de comparer le signal envoyé et reçu.



Signaux modulé ($\mathcal{F}(s(t).p(t))$)

Ici on retrouve notre signal $S_{mod}(f)$ comme défini précédemment. On voit bien qu'il a été projeté en -20kHz et 20kHz.



Graphe 1 Graphe 2 Graphe 3

Du côté de la démodulation on a ces 3 graphiques : Graphe 1 : la transformée de Fourier du signal reçu ; Graphe 2 : la transformée de Fourier du signal reçu multiplié par la porteuse ; Graphe 3 : la transformée de Fourier du signal modulé que nous avons filtré pour garder uniquement le motif central qui est notre signal de base.

Pour vérifier si notre modulation/démodulation s'est bien passé il faut regarder que le motif du Graphe 3 correspond à celui de la transformée de Fourier du signal module (Cf screen du dessus).

3. Gain de bande-passante

Il est ensuite question développer une variante du programme pour permettre, dans les mêmes conditions de bandes passantes et de performances, de transmettre non pas 4 mais 8 messages simultanément sur le même canal.

3.1. Bureau d'études

Face à ce problème 2 solutions nous sont venues à l'esprit :

- Redécouper nos bandes de 8kHz en bandes de 4kHz pour envoyer deux fois plus de signaux. Cependant, nous augmentons les risques de ne plus respecter le critère de Shannon ce qui rendra le signal non reconstructible.
- La modulation BLU. Effectivement, notre signal d'entrée étant réel, son spectre est réel est pair. Finalement, on utilise que la moitié du spectre puisque l'autre se retrouve par symétrie.

On utilise la transformée de Hilbert pour la modulation BLU. L'objectif est de retarder le signal de $\frac{\pi}{2}$ avec un filtre de quadrature.

Matlab met à disposition la fonction '$\mathcal{H}$ilbert' dont on garde la partie imaginaire du signal en sortie.

L'idée est donc d'enregistrer 2 signaux $s_1(t)$ et $s_2(t)$ et de là, créer $x_1(t)$ et $x_2(t)$ afin d'envoyer un signal de la forme : $x(t) = x_1(t) + x_2(t)$

$$x_1(t) = s_1(t).cos(2\pi.\nu_0.t) - \mathcal{H}\big(s1(t)\big).sin(2.\pi.\nu_0.t)$$

$$x_2(t) = s_2(t).cos(2\pi.\nu_0.t) - \mathcal{H}\big(s2(t)\big).sin(2.\pi.\nu_0.t)$$

Pour démoduler ce signal, il faudra utiliser un passe-bande dans un premier temps pour récupérer les 2 parties qu'on démodulera comme précédemment (multiplication par la porteuse puis filtrage passe-bas).

3.2. Programmation

transmissionBLU.m

Les objectifs de ce fichier sont les suivants :

1. Enregistrement de deux signaux
2. Filtrage des deux signaux par un passe-bas
3. Modulation du signal filtré
4. Sauvegarde/Emission du signal modulé

demodulationBLU.m

Les objectifs de ce fichier sont les suivants :

1. Filtrage des deux signaux par un passe-bande pour isoler les messages
2. Démodulation des signaux filtrés
3. Filtrage de signal démodulé par un passe-bas
4. Sauvegarde/Emission des signaux démodulés

PasseBande.m (fourni)

Cette fonction prend en entré le signal z sur lequel effectuer le PasseBande, la fréquence d'échantillonnage de z : F_e , F_{cLow}, F_{cHigh} respectivement les fréquences de coupure basse et haute à -3dB. Et elle renvoie s le vecteur d'échantillon temporel du signal z filtré.

Malheureusement, nous n'avons pas pu faire fonctionner ces parties du projet sur le logiciel Octave, mais les codes se trouvent en annexe.

4. Conclusion

Ainsi, après avoir analysé le projet à l'étape de bureau d'études, nous avons pu transmettre simultanément sur un même canal (par voie filaire) 4 messages vocaux différents émis chacun par une source (ordinateur) et reçu par le récepteur du groupe correspondant (ordinateur) par le principe de modulation/démodulation des signaux. Néanmoins nous n'avons pas réussi à faire fonctionner la partie modulation/démodulation par bande latérale unique pour transmettre deux fois plus de signaux.

5. Annexes

Fonction Transmission

```
function [s_module] = transmission()

Fe = 96000;
T = 2;
nu03 = 20000;
Fc = 8000;
t=0:1/Fe:T-1/Fe;


%enregistrement du son
[nomfic,Signal,t] = RecordModulation(Fe,T);

%filtrage du son enregistré
[s]=PasseBas(Signal,Fe,Fc);

%création de la porteuse
porteuse = cos(2*pi*nu03*t);

%modulation du signal
s_module = porteuse.*s;

%sauvegarde du signal modulé
audiowrite('signalmodule.wav',s_module,Fe);

%transformée de Fourier du signal modulé
[S,f] = TransFourier(s,t);

%affichage de la transformée de Fourier
plot(f,abs(S));
size(t)
size(s)

end
```

Fonction Demodulation

```
function [s_demodulee] = demodulation(s,f)

x=s';
Fe = 96000;
Fp = 20000;
T = 5;
t=0:1/Fe:T-(1/Fe);

%création de la porteuse
p = cos(2*pi*Fp.*t);

%démodulation du signal transmis
x1 = x.*p;

%transformée de Fourier des signaux modulé et démodulé
[tfs,f1] = TransFourier (s,t);
[tfx1,f3] = TransFourier (x1,t);

%filtrage du signal démodulé
s_demodulee = PasseBas(x1,Fe,8000);

%transformée de Fourier du signal démodulé filtré
[tfsm,f2] = TransFourier (s_demodulee,t);

%affichage des transformées de Fourier
figure(2);
subplot(132);plot(f3,abs(tfx1));xlabel('TF du signal demodulee');
subplot(133);plot(f2,abs(tfsm));xlabel('TF du signal demodulee et filtree');
subplot(131);plot(f1,abs(tfs));xlabel('TF du signal reÃ§u');

%lecture du signal démodulé
while 255 == 255
    soundsc(s_demodulee,Fe);
    pause(6);
end
end
```

Fonction TransmissionBLU

```
function [s_moduleBLU] = transmissionBLU()
Fe = 96000;
T = 2;
nu03 = 20000;
Fc = 8000;
t=0:1/Fe:T-1/Fe;


%enregistrement des 2 sons
[nomfic1,Signal1,t1] = RecordModulation(Fe,T);
[nomfic2,Signal2,t2] = RecordModulation(Fe,T);

%filtrage des 2 sons enregistrés
[s1]=PasseBas(Signal1,Fe,Fc);
[s2]=PasseBas(Signal2,Fe,Fc);

%création des signaux x1 et x2
y1=hilbert(s1);
y2=hilbert(s2);

TransHilbert1=imag(y1);
TransHilbert2=imag(y2);

x1=s1.*cos(2*pi*nu03*t) - TransHilbert1.*sin(2*pi*nu03*t);
x2=s2.*cos(2*pi*nu03*t) - TransHilbert2.*sin(2*pi*nu03*t);

%création du signal final x1+x2
s_moduleBLU= x1 + x2;

audiowrite('signalmoduleBLU.wav',s_moduleBLU,Fe);

%transformées de Fourier des signaux
[X1,f1] = TransFourier(x1,t);
[X2,f2] = TransFourier(x2,t);
[S_ModuleBLU,f] = TransFourier(s_moduleBLU,t);

%affichage des transformées de Fourier
figure(1);
subplot(1,3,1);plot(f1, abs(X1));xlabel('TF du signal x1');
subplot(1,3,2);plot(f2, abs(X2));xlabel('TF du signal x2');
subplot(1,3,3);plot(f, abs(S_ModuleBLU));xlabel('TF du signal émis');

end
```

Fonction DemodulationBLU

```
function [signal_demodulee_BLU] = demodulation(s_moduleBLU,f)

x=s';
Fe = 96000;
Fp = 20000;
Fc = 8000;
T = 5;
t=0:1/Fe:T-(1/Fe);

%filtrage du signal émis par transmissionBLU
[s_moduleBLU_filtree1]=PasseBande(s_moduleBLU,Fe,0,Fc);
[s_moduleBLU_filtree2]=PasseBande(s_moduleBLU,Fe,Fc,2*Fc);

%démodulation du signal filtré
s_demoduleeBLU1 = s_moduleBLU_filtree1.*cos(2*pi*Fp*t)*2;
s_demoduleeBLU2 = s_moduleBLU_filtree2.*cos(2*pi*Fp*t)*2;

%filtrage du signal démodulé
[s_demoduleeBLU_filtree1] = PasseBas(s_demoduleeBLU1,Fe,Fc);
[s_demoduleeBLU_filtree2] = PasseBas(s_demoduleeBLU2,Fe,Fc);

%lecture et sauvegarde des sons démodulées par BLU
soundsc(s_demoduleeBLU_filtree1, Fe);
audiowrite('signaldemoduleBLU1.wav'.wav', s_demoduleeBLU_filtree1, Fe);

soundsc(s_demoduleeBLU_filtree2, Fe);
audiowrite('signaldemoduleBLU2.wav'.wav', s_demoduleeBLU_filtree2, Fe);

%transformées de Fourier des signaux
[S1,f1] = TransFourier(s_demoduleeBLU_filtree1,t);
[S2,f2] = TransFourier(s_demoduleeBLU_filtree2,t);

%affichage des transformées de Fourier
figure(1);
subplot(1,2,1);plot(f1, abs(S1));xlabel('TF du signal 1 démodulé par BLU');
subplot(1,2,2);plot(f2, abs(S2));xlabel('TF du signal 2 démodulé par BLU');

end
```

Fonction Passe-Bas

PasseBas.m

```
function [s]=PasseBas(z,Fe,Fc)

% [s]=PasseBas(z,Fe,Fc)
% Filtrage passe-bas du signal temporel z.
% Inputs
% z  : vecteur des echantillons temporels du signal a filtrer
% Fe : frequence d'echantillonnage de z
% Fc : frequence de coupure a -3dB du filtre passe-bas
% Outputs:
% s  : vecteur des echantillons temporels du signal filtre

% PG : 2017

Wc = 2*Fc/Fe ;
ordre = 9 ;
[B,A] = butter(ordre,Wc) ;
s = filtfilt(B,A,z) ;
```

Fonction Passe-Bande

PasseBande.m

```
function [s]=PasseBande(z,Fe,FcLow,FcHigh)

% [s]=PasseBas(z,Fe,FcLow,FcHigh)
% Filtrage passe-bande du signal temporel z dans la bande
% [-FcHigh,FcLow] U [FcLow,FcHigh]
% Inputs
% z  : vecteur des echantillons temporels du signal a filtrer
% Fe : frequence d'echantillonnage de z
% FcLow  : frequence de coupure Basse a -3dB du filtre passe-bande
% FcHigh : frequence de coupure Haute a -3dB du filtre passe-bande
% Outputs:
% s  : vecteur des echantillons temporels du signal filtre

% PG : 2019

WcLow = 2*FcLow/Fe ;
WcHigh = 2*FcHigh/Fe ;
ordre = 9 ;
[B,A] = butter(ordre,[WcLow WcHigh]) ;
s = filtfilt(B,A,z) ;
```

Fonction TransFourier

TransFourier.m

```
function [S,f] = TransFourier(s,t)

% [S,f] = TransFourier(s,t)
% Transformee de Fourier d'un signal s.
% Input:
% - s: vecteur de taille N contenant les N echantillons s[n] du signal a
% analyser
% - t: vecteur de taille N contenant les instants d'echantillonnage de s.
% s[n] = s(t[n]).
% Outputs:
% - S : vecteur de taille N contenant les coeffeciens de la transformee de
% Fourier du signal s
% - f : vecteur de taille N contenant les frequences correspondant aux
% coefficients de S : S[n] = S(f[n])

s = s(:)' ;
N = length(s) ;

switch nargin
    case 1
        t  = 1:N ;
end

if N ~= length(t)
    error('Les vecteurs "s" et "t" doivent etre de meme longueur')
end
if std(diff(t)) > 1000*eps
    error('Le vecteur ''t'' doit etre lineairement croissant et a pas constant')
end

dt = t(2)-t(1) ; Fe = 1/dt ;
sshift = [s(t>=0) s(t<0)] ;

M = N ;
S = fft(sshift,M) ;
S = fftshift(S) ;
S = S.*dt ;
f = linspace(-Fe/2,Fe/2,M+1) ;
f = f(1:M) ;
```

Fonction RecordModulation

RecordModulation.m

```
function [nomfic,Signal,t] = RecordModulation(Fe,T)
% [nomfic,Signal,t] = RecordModulation(Fe,T)
% Permet de proceder a l'enregistrement audio d'un message.
% Numerisation sur 16 bits en mode mono.
%
% Inputs:
%   Fe      : frequence d'echantillonnage (multiple de 8kHz et inferieure ou
%   egale a 96kHz)
%   T       : dur?e (en secondes) du message a enregistrer
%
% Outputs:
%   nomfic  : Nom (chaine de characteres) du fichier .wav dans lequel a ete
%   enregistre le message
%   Signal  : vecteur contenant les echantillons du message enregistre
%   t       : vecteur temporel correspondant au vecteur Signal

% Version Projet Modulation de la fonction 'enregistrer.m' standard
% PG 2017
        T=5;  % Duree d'enregistrement.
bits=16;
Canaux=1;
sauve = 0 ;

while sauve == 0 ;
% Cree objet audiorecorder.
Obj_Rec=audiorecorder(Fe,bits,Canaux);
disp('------  Parlez !!!  ------')
recordblocking(Obj_Rec, T); % Enregistre durant un temps T.
disp('------  STOP !!!  ------');
% Joue l'enregistrement.
%play(Obj_Rec);
% disp('appuyer sur la barre d''espace une fois l''ecoute terminee')
% pause;
Signal=getaudiodata(Obj_Rec);

N=length(Signal);
t=(0:N-1)/Fe;
%figure(1);
t = t(:) ;  Signal = Signal(:) ;
%plot(t,Signal);

sauve=input('Sauvegarde dans un fichier ? oui=1 non=0 : ');
if sauve==1
    nomfic=input('Nom du fichier (entre cotes):  ');
    nomficwav=[nomfic,'.wav'];
    audiowrite(nomficwav,Signal,Fe);
end
end
```