
Saé 22 : Mesurer et caractériser un signal ou un système

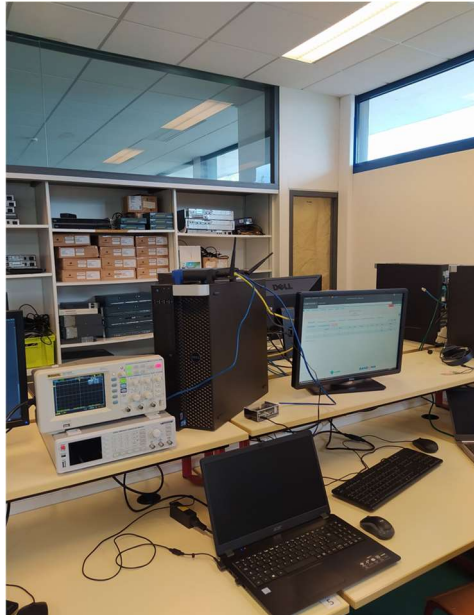


Figure 1: Plan de travail de cette Saé

Revue de projet faite par Zacharie George, Thomas Raynaud et Thomas Mirbey

Introduction :

Le nom complet de Saé est « situation d'apprentissage et d'évaluation ». Son objectif est de valider les nombreuses compétences acquises au fur et à mesure de l'année dans les différentes ressources.

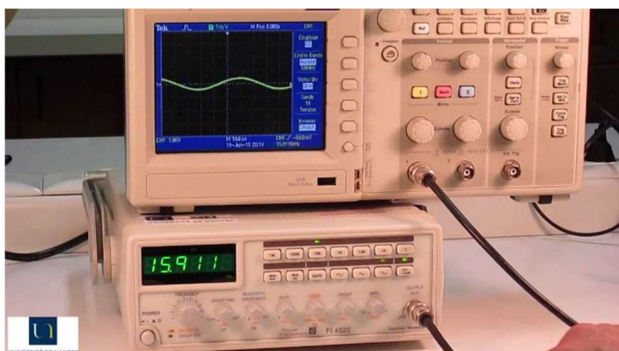
Cette Saé nommée « Mesurer et caractériser un signal ou un système » a débuté le mercredi 4 mai, et s'est finie le vendredi 6 mai 2022.

Il fallait savoir mobiliser les compétences pour analyser des signaux d'un système de transmission, les exploiter, et les présenter dans ce rapport.

Pendant cette Saé, nous avons eu l'occasion de manipuler du matériel omniprésent dans le domaine des télécommunications. Durant ces trois jours, des professeurs nous ont accompagnés afin de nous aider si nous rencontrions des obstacles, pour nous expliquer des choses que nous ne comprenions pas et même pour nous poser certaines questions afin de voir si nous comprenions ce que nous manipulons. Les professeurs étaient Monsieur Vanstraceele, Monsieur Vuillemin et Monsieur Givron.

Notre revue de projet suivra le plan suivant :

1. Vocabulaire technique	3
2. Matériel mis à notre disposition	4
3. Installation de FreePBX	5
4. Configuration de FreePBX	5
5. Configuration de Linphone	7
6. Test de fonctionnement	8
7. Mesures de bandes passantes	9
8. Mesures de rapports signal sur bruit	14
9. Mesures de spectres	16
10. Comparaison audio	18
11. Intérêt du codec	18



1 Vocabulaire technique

Avant de commencer ce rapport, nous devons définir certains termes techniques qui seront abordé lors de l'explication pratique et l'interprétation des résultats de nos travaux :

- **PBX** est un commutateur permettant de d'interconnecter plusieurs postes téléphoniques clients pour permettre des communications.
- **VoIP** ou **Voice Over IP** est une technologie permettant de transmettre la voix sur des réseaux.
- **ToIP** ou **Téléphonie sur IP** utilise la voix sur IP et y ajoute différentes fonctions de la téléphonie (Présentation du numéro appelant, messagerie vocale, transfert d'appel, renvoi d'appel ...).
- **Codec**, dans notre cas, sont des composants logiciels qui permettent de coder et décoder des flux de données. Permet de diminuer fortement le débit des informations tout en gardant une certaine qualité de voix.
- **SIP** ou **Session Initiation Protocol** est un protocole utilisé dans les communications multimédias.
- **GUI** ou **Graphical User Interface** est une interface graphique d'un logiciel pour un utilisateur.
- **Firmware** est un programme intégré à un système informatique nécessaire à son fonctionnement.

VoIP



2 Matériel mis à notre disposition et topologie réseau

Comme énoncé lors de l'introduction, du matériel spécifique aux télécommunications était mis à notre disposition afin de réaliser cette Saé. Le matériel utilisé était :

- Un **Raspberry Pi** 4 de 2Go de RAM ;
- Une **carte micro SD** contenant une image du système d'exploitation (OS) de **FreePBX** et permettant l'accès à un serveur VoIP Asterisk ;
- Un **routeur wifi Linksys** qu'il faut relier aux clients et au serveur **VoIP** ;
- Des **machines clientes** (ordinateurs sous Debian) ;
- Des **câbles** permettant de réaliser les branchements réseau ;
- Un **Oscilloscope** ;
- Un **Générateur Basse Fréquences** (GBF) ;

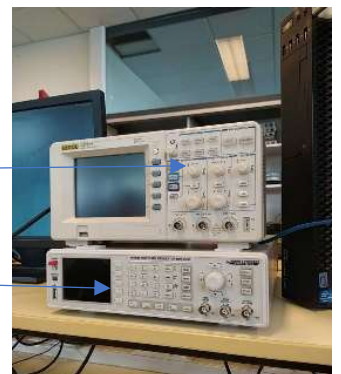


Figure 2: Oscilloscope et GBF

- Des **câbles** permettant de réaliser les branchements électroniques (**cordons Jack mono/banane**, **cordons à reprise arrière**) ;
- **Adaptateurs** pour sonde oscilloscope (connecteur BNC).

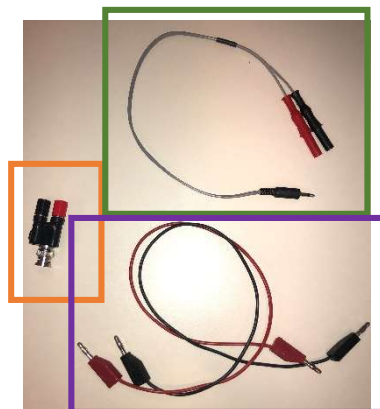


Figure 3: Cordons à branchements électrique

3 Installation de FreePBX

La configuration du matériel était assez simple, nous possédions une carte micro SD contenant l'image de l'OS FreePBX (Aussi appelé RaspPBX). Il nous suffisait donc de mettre cette dernière dans le connecteur de carte micro SD du Raspberry Pi 4 puis de le mettre sous tension afin de le démarrer.



Le Raspberry Pi jouait le rôle d'un serveur VoIP Asterisk. Il agissait comme un commutateur privé (PBX) en créant un réseau local physique permettant de connecter les lignes téléphoniques entre-elles.

4 Configuration de FreePBX

Dans cette partie, l'objectif était d'accéder à la page de connexion FreePBX GUI sur un navigateur afin de réaliser les manipulations demandées dans le document « Présentation » de cette Saé. Il faudra utiliser les ressources mises à notre disposition pour atteindre cet objectif.

Afin de configurer FreePBX nous avons réalisé différentes étapes dans l'ordre suivant :

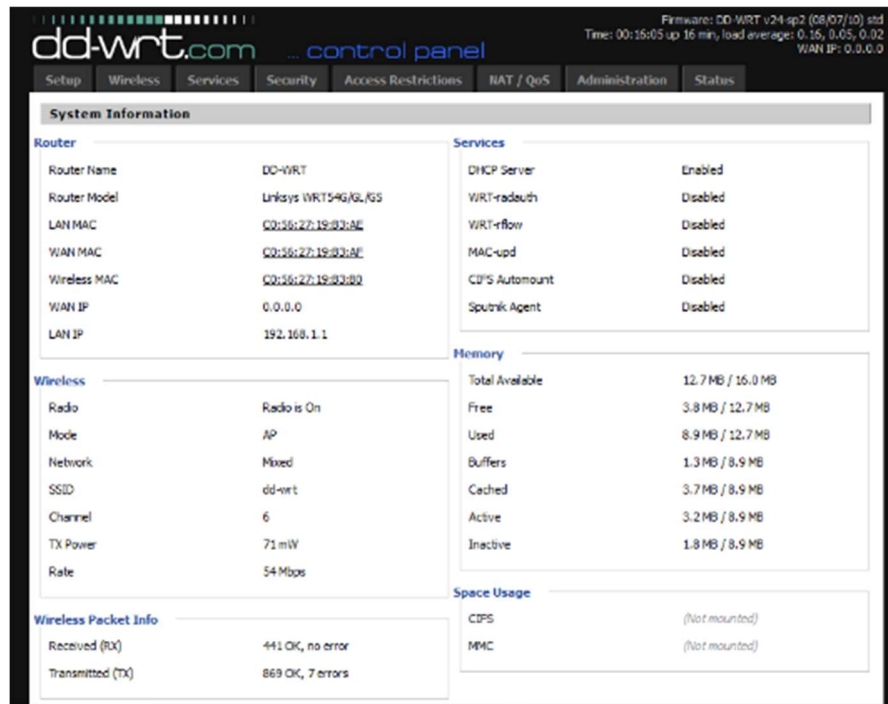
- 1 Réinitialisation de la borne Linksys en suivant des instructions précises (appuyer sur le bouton reset, puis mettre sous tension en maintenant le bouton 5 secondes) ;
- 2 Réalisation de la topologie réseau en branchant les différents appareils à la borne wifi Linksys à l'aide de câbles Ethernet (présentée dans la partie « Matériel mis à notre disposition et topologie réseau ») :
 - Raspberry Pi ;
 - PC1 sur la deuxième carte réseau (eth0) ;
 - PC2 (ordinateur portable personnel) ;

- Configuration au routeur Linksys via un navigateur depuis le PC1 (Connexion sur l'IP 192.168.1.1 => IP du routeur).

Une fois dans le firmware, nous avons entré notre identifiant et mot de passe suivants :

Identifiant : groupe3

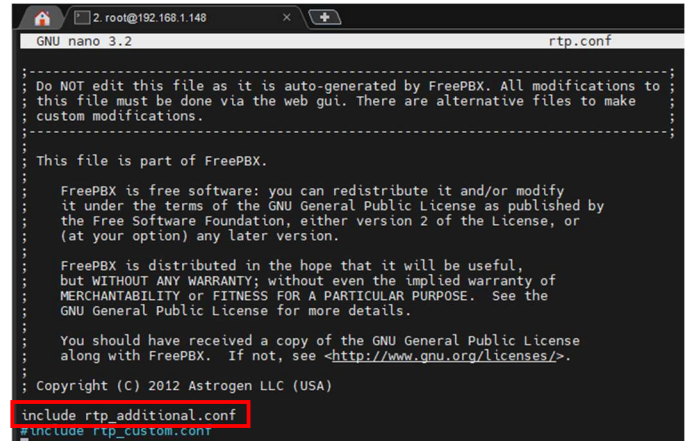
Mot de passe : TTZ



- Nous avons recherché de l'IP du Raspberry Pi depuis le **firmware** dans la liste des adresses distribuées par le **serveur DHCP** du routeur Linksys. On trouve l'adresse IP suivante : 192.168.1.148.

MAC Address	Host Name	IP Address	Enable WOL?
48:9E:BD:DB:51:E7	v1-HP-245-G8-Notebook-PC	192.168.1.100	<input type="checkbox"/>
E8:94:F6:02:07:81	rt	192.168.1.102	<input type="checkbox"/>
DC:A6:32:2B:2F:6E	raspbx	192.168.1.148	<input type="checkbox"/>
08:97:98:94:5B:E5	LAPTOP-M49NSQUA	192.168.1.147	<input type="checkbox"/>

5. Nous nous sommes connectés au Raspberry Pi en **SSH** pour **modifier le fichier de configuration** sous `/etc/asterisk/rtp.conf`
Les identifiants du Raspberry Pi pour se connecter étaient : root (identifiant), raspberry (mot de passe).



```
GNU nano 3.2 rtp.conf
Do NOT edit this file as it is auto-generated by FreePBX. All modifications to
this file must be done via the web gui. There are alternative files to make
custom modifications.

This file is part of FreePBX.

FreePBX is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 2 of the License, or
(at your option) any later version.

FreePBX is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with FreePBX. If not, see <http://www.gnu.org/licenses/>.

Copyright (C) 2012 Astrogen LLC (USA)

include rtp_additional.conf
#include rtp_custom.conf
```

6. Connexion à la page FreePBX GUI du Raspberry Pi en se connectant avec l'IP de ce dernier (trouvé dans les étapes précédentes) nous avons créé un identifiant et mot de passe :

Identifiant : groupe3

Mot de passe : RT2022SAE

IP du Raspberry Pi : 192.168.1.148

7. Puis, nous avons réalisé les configurations demandées sous :

- Sous Settings/Asterisk SIP Settings/General SIP Settings;
- Sous Settings/Asterisk Settings/SIP Settings [chan_pjsip];
- Applications/Extensions/Add Extension/Add New SIP [chan_pjsip] Extension;

Cette étape de configuration de FreePBX ne nous a pas posé de problèmes et a été réalisée assez rapidement le mercredi après-midi.

5 Configuration de Linphone

Pour faire la configuration de Linphone il faut tout d'abord télécharger le paquet sur les deux PC grâce à la commande : **sudo apt-get install linphone**.

Nous avons eu plusieurs **problèmes** à cette étape.

Le PC que nous avons ne pouvait pas accéder au portail captif ce qui empêchait de télécharger les paquets. Nous nous sommes rendu compte que le câble réseau de notre PC n'était pas connecté à l'armoire de brassage, mais était branché directement à un switch dans un autre étage. Nous avons donc rebranché le câble sur l'armoire de brassage puis récupéré une IP à l'aide de la commande **dhclient**. Nous avons finalement pu réaliser l'installation de Linphone.

Nous avons par la suite rencontré des difficultés quant à la création des téléphones IP. Une fois le gestionnaire pris davantage en main nous avons pu contourner cette difficulté et mettre en place les deux téléphones IP.

Le second PC que nous avons utilisé ne nous permettait pas d'accéder au portail captif malgré de nombreuses manipulations (renouvellement de l'IP avec la commande **dhclient**, recâblage,

remise en place du proxy ...). Pour contourner ce problème nous avons utilisé un de nos PC personnels tournant sur UBUNTU et nous n'avons plus rencontré de difficultés lors de installations suivantes.

Configuration dans le logiciel :

- Création de notre identité SIP en utilisant plusieurs informations : numéro de téléphone, IP du serveur VoIP (Raspberry Pi) et le port SIP. Il fallait aussi renseigner l'adresse du proxy SIP avec certaines des informations précédentes.
- Réalisation de cette manipulation sur les deux machines séparément avec les numéros 300 et 301 qui ont été réservés lors de la configuration de FreePBX. Il fallait suivre l'exemple présent dans les diaporamas de consignes.
- Plusieurs problèmes ont été rencontrés, les téléphones ne communiquaient pas, nous avons alors :
 - Vérifié les configurations pour voir si elles comportaient des erreurs.
 - Redémarré le Raspberry Pi.
 - Mis le même codec dans les logiciels client (Linphone).
- Ce n'a malgré tout pas réglé le problème, les appels ne passaient que dans un sens.

Monsieur Vanstraceele nous a conseillé de réaliser à nouveau un appel et d'utiliser la commande **asterisk -rvvvv** en console sur le serveur VoIP pour vérifier si le serveur nous retournait des erreurs mais ce n'a pas été le cas. Nous avons alors démarré à nouveau le Raspberry Pi et les communications ont fonctionnées.

6 Test de fonctionnement

Les appels passent dans les deux sens. Nous pouvons nous connecter au Raspberry Pi en SSH et pinguer les différents équipements.

Nous avons rencontré des difficultés à prendre en main Freepbx pour mettre en place les clients. Nous avons aussi eu du mal à télécharger les paquets nécessaires pour Linphone.

Nous avons eu plusieurs soucis de configurations
Un de nos PC n'obtenait pas d'IP via le serveur de l'IUT en raison de mauvais branchements.

L'autre PC ne nous permettait pas d'accéder au portail captif nécessaire pour se connecter à Internet, nous avons donc utilisé notre PC personnel afin de contourner cette difficulté.

Nous avons dû obtenu des erreurs "service unavailable" à chaque changement de codec. Nous avons redémarré le Raspberry Pi à chaque modification de la configuration. Ce qui nous a fait perdre pas mal de temps lors du changement de codec principalement.

7 Mesures de bandes passantes

Après avoir mis en place la configuration nécessaire avec le Générateur Basses Fréquences et l'oscilloscope, il fallait passer à la pratique.

Mais l'utilisation du GBF nous a posé des problèmes.

C'est pour cela que Monsieur Givron est intervenu pour nous faire une **courte explication**, nous n'utilisons pas la bonne sortie du GBF et il nous a montré comment mesurer des valeurs avec l'oscilloscope.

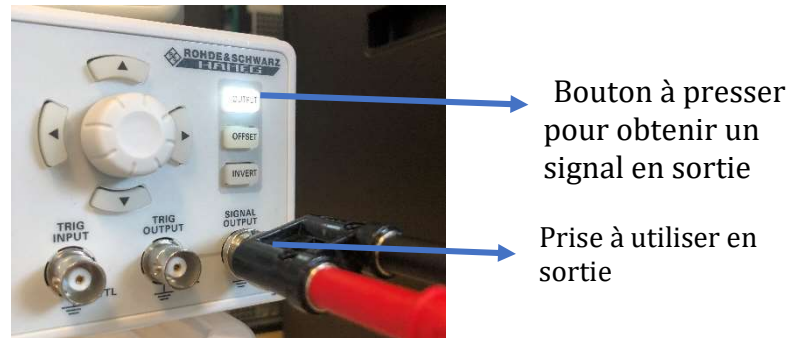


Figure 4: Sortie du GBF

Il en a profité pour ajouter des câbles pour comparer le signal en entrée du GBF et en sortie de l'ordinateur recevant l'appel ce qui nous a facilité le travail par la suite car on a pu isoler les problèmes. Si aucun signal ne sort du GBF, on peut deviner que le problème n'est pas informatique. Cependant dans le cas contraire, si aucun signal n'apparaît on peut en déduire que le problème est informatique.

Les deux sources que nous avons sont visibles :

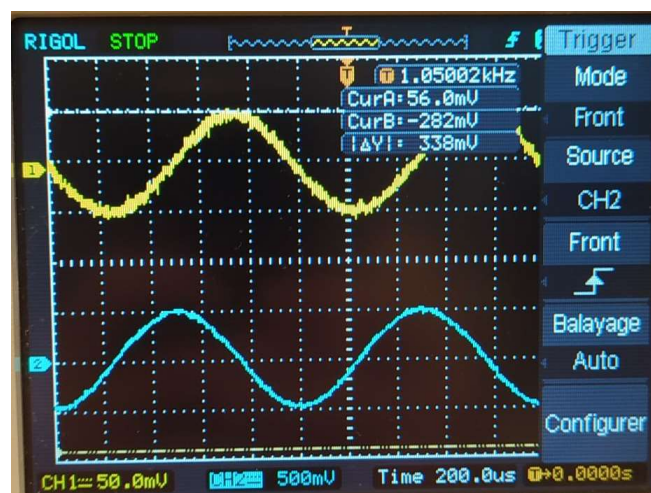


Figure 5: Visualisation des signaux sur l'oscilloscope (bleu = signal en sortie directe du GBF ; jaune = sortie du signal reçu par le PC2)

Le fait de changer de codec après avoir déterminé sa bande passante nous a porté défaveur car la configuration des téléphones logiciel se réinitialisait par moment.

Nous avons donc réussi à prendre les différentes mesures afin de déterminer la bande passante. Dans les tableaux suivants, la case en jaune est la fréquence de fin de la bande passante. Nous avons pu la déterminer grâce à la valeur présente dans la case « Amplitude de la fréquence de coupure ».

Il suffisait de la calculer en utilisant la valeur de l'amplitude en sortie pour 1kHz divisée par $\sqrt{2}$. Nous avons donc mis ce calcul comme témoin de bande passante qui est une valeur qui doit être inférieure à l'amplitude en sortie que l'on mesure pour chaque fréquence. Si l'amplitude de la fréquence de bande passante vient à être supérieur à une amplitude que l'on aurait mesuré en sortie, on pourrait donc déterminer la fin de la bande passante. Il faudra donc réaliser des mesures de plus en plus précises afin de s'approcher le plus possible de la fréquence de coupure. Lors de nos mesures, plusieurs **contraintes** nous ont été rencontrées, notamment lors de la montée en fréquence dans le GBF. Les mesures étaient par moment imprécises et les valeurs en sortie fluctuaient beaucoup.

Monsieur Vanstraceele nous a aussi fait remarquer que les basses fréquences (20Hz par exemple) ne sont pas prises en charge par les différents codecs.

- Utilisation du codec **PCMU**, avec une fréquence d'échantillonnage de 8 kHz et un débit de 80 kbit/s :

Fréquences en entrée en kHz	Amplitude max sortie en mV	Amplitude min sortie en mV	Amplitude en sortie en mV (Us)	Amplitude de la fréquence de coupure
1	124	-124	248	175,3624817
2	124	-124	248	
3	108	-104	212	
3,01	92	-84	176	
3,02	92	-84	176	
3,03	92	-80	172	
3,04	84	-80	164	
3,1	92	-72	164	
3,5	36	-28	64	

Tableau 1 : Présentation de nos résultats de mesures pour le codec PCMU

⇒ D'après ce tableau, on peut en conclure que la bande passante du codec PCMU se situe d'environ 0 à 3.03 kHz. Car l'amplitude en sortie à 3,04 kHz est inférieure à l'amplitude de fréquence de coupure.

D'un point de vue purement théorique on peut également déterminer la fréquence maximale de la fréquence de coupure. Grâce au théorème de Shannon :

$$F_e \geq 2F_{max}$$

Où F_{max} (fréquence de coupure maximale accepté par le codec).

Dans ce cas, sachant que la fréquence d'échantillonnage est de 8kHz, nous pouvons modifier la formule afin de faire : $F_{max} \leq F_e / 2$. On obtient donc : $F_{max} \leq 8/2 = 4 \text{ kHz}$.

- Utilisation du codec **Speex**, avec une fréquence d'échantillonnage de 16 kHz et un débit de 40 kbit/s :

Fréquences en entrée en kHz	Amplitude max sortie en mV	Amplitude min sortie en mV	Amplitude en sortie en mV (Us)	Amplitude de la fréquence de coupure
1	112	-104	216	152,7350647
2	120	-104	224	
3	112	-100	212	
4	108	-108	216	
4,1	88	-76	164	
4,2	100	-84	184	
4,3	92	-76	168	
4,31	72	-64	136	

Tableau 2 : Présentation de nos résultats de mesures pour le codec Speex

⇒ D'après ce tableau, on peut en conclure que la bande passante du codec PCMU se situe d'environ 0 à 4.3 kHz. Car l'amplitude en sortie à 4.31 kHz est inférieure à l'amplitude de fréquence de coupure.

D'un point de vue purement théorique on peut également déterminer la fréquence maximale de la fréquence de coupure. Grâce au théorème de Shannon :

$$F_e \geq 2F_{max}$$

Où F_{max} (fréquence de coupure maximale accepté par le codec).

Dans ce cas, sachant que la fréquence d'échantillonnage est de 16kHz, nous pouvons modifier la formule afin de faire : $F_{max} \leq F_e / 2$. On obtient donc : $F_{max} \leq 16/2 = 8 \text{ kHz}$.

- Utilisation du codec **GSM**, avec une fréquence d'échantillonnage de 8 kHz et un débit de 30 kbit/s :

Fréquences en entrée en kHz	Amplitude max sortie en mV	Amplitude min sortie en mV	Amplitude en sortie en mV (Us)	Amplitude de la fréquence de coupure
1	128	-128	256	181,019336
2	128	-128	256	
3	96	-88	184	
3,05	108	-108	216	
3,06	76	-60	136	
3,07	80	-68	148	
3,1	80	-72	152	

Tableau 3 : Présentation de nos résultats de mesures pour le codec GSM

⇒ D'après ce tableau, on peut en conclure que la bande passante du codec PCMU se situe d'environ 0 à 3.05 kHz. Car l'amplitude en sortie à 3,06 kHz est inférieure à l'amplitude de fréquence de coupure.

D'un point de vue purement théorique on peut également déterminer la fréquence maximale de la fréquence de coupure. Grâce au théorème de Shannon :

$$F_e \geq 2F_{max}$$

Où F_{max} (fréquence de coupure maximale accepté par le codec).

Dans ce cas, sachant que la fréquence d'échantillonnage est de 8kHz, nous pouvons modifier la formule afin de faire : $F_{max} \leq F_e / 2$. On obtient donc : $F_{max} \leq 8/2 = 4 \text{ kHz}$.

- Utilisation du codec **Opus**, avec une fréquence d'échantillonnage de 48 kHz et un débit de 50 kbit/s :

Fréquences en entrée en kHz	Amplitude max sortie en mV	Amplitude min sortie en mV	Amplitude en sortie en mV (Us)	Amplitude de la fréquence de coupure
1	108	-100	208	147,0782105
2	104	-92	196	
5	96	-92	188	
8	96	-88	184	
11	104	-84	188	
14	84	-82	166	
17	100	-84	184	
20	80	-82	162	
20,01	84	-72	156	
20,02	56	-40	96	

Tableau 4 : Présentation de nos résultats de mesures pour le codec Opus

⇒ D'après ce tableau, on peut en conclure que la bande passante du codec PCMU se situe d'environ 0 à 20.01 kHz. Car l'amplitude en sortie à 20.02 kHz est inférieure à l'amplitude de fréquence de coupure.

D'un point de vue purement théorique on peut également déterminer la fréquence maximale de la fréquence de coupure. Grâce au théorème de Shannon :

$$F_e \geq 2F_{max}$$

Où F_{max} (fréquence de coupure maximale accepté par le codec).

Dans ce cas, sachant que la fréquence d'échantillonnage est de 48kHz, nous pouvons modifier la formule afin de faire : $F_{max} \leq F_e / 2$. On obtient donc : $F_{max} \leq 48/2 = 24 \text{ kHz}$.

D'après les mesures des différents codecs que nous avons utilisés, nous pouvons en déduire plusieurs choses.

Un écart théorique pratique existe, par exemple pour le codec Opus la fréquence de coupure pratique est de 20.01kHz. Alors que la fréquence de coupure théorique qui est de 24 kHz. Nous pouvons supposer que cette différence peut être expliqué par de nombreux facteurs qui viennent modifier la valeur théorique : perturbations électromagnétique, erreurs de mesures, contrôle automatique de gain.

Cependant la raison principale reste que le codec accepte des fréquences différentes de celles mesurées dans la théorie. Il y a aussi une influence des cartes son de nos ordinateurs. En effet le logiciel ne peut modifier la fréquence d'échantillonnage de la carte son. Donc il enlève des échantillons si la fréquence d'échantillonnage voulue est plus faible. Dans le cas inverse on doit ajouter des échantillons.

Nous pouvons en déduire que pour une fréquence d'échantillonnage d'un codec qui est de $F_e=48\text{kHz}$, on trouve $F_{\text{max}}=24\text{kHz}$ en théorie. Mais si on augmente la fréquence en entrée dans l'ordinateur on se rend compte que vers 20.01kHz , notre machine traite mal les fréquences plus élevées. Notre fréquence de coupure pratique est de 20.01kHz . Au-dessus de ces valeurs le logiciel interprète que le signal n'est plus exploitable pour le codec utilisé à partir d'une certaine fréquence. On ne veut donc pas travailler directement à la limite de Shannon mais on préfère avoir une marge.

Par la suite, nous avons eu l'idée de voir si le changement de débit pouvait influencer la fréquence de coupure pratique et donc la bande passante.

Nous avons donc refait certaines mesures, avec le codec Opus à un débit de 20 kbit/s puis 80 kbit/s . Cependant après relecture des valeurs par Monsieur Givron, les résultats n'étaient pas très cohérent. Dans un premier temps, le tableau que nous avons obtenu est :

Fréquences (kHz)	Amplitude	Objectifs
1	200	141,421356
3	172	
3.5	180	
3.6	168	
3.7	200	
3.8	160	
3.9	124	
4	/	

Tableau 5 : Présentation de nos résultats avec un débit de 20kbit/s

Dans un second temps :

Fréquences (kHz)	Amplitudes (mV)	Amplitude fréquence de coupure
1kHz	190	134,350288
5kHz	192	
8kHz	192	
10kHz	196	
15kHz	200	
18kHz	152	
18.3	148	
18.4	128	
18.5	124	
19kHz	108	

Tableau 6 : Présentation de nos résultats avec un débit de 80kbit/s

⇒ Initialement il ne devrait pas y avoir de modifications car le débit n'exerce aucune influence sur la bande passante.

8 Mesures de rapports signal sur bruit

Nous avons réalisé des mesures sur le rapport signal sur bruit.

Afin de réaliser ces mesures, nous avons injecté un signal sinusoïdal de fréquence 1 kHz. Nous avons réalisé des mesures avec les différents codecs.

/	Opus	Speex	GSM	PCMU
Amplitude du bruit (mV)	16	10	10	8
Amplitude du signal (mV)	208	216	256	248
SNB (dB)	22,27886705	26,68907502	28,16479931	29,82723388

Tableau 7 : Présentation de l'amplitude du bruit et du signal (en mV) et du SNB (en db)

Les mesures ont été réalisées à l'oscilloscope en mesurant sur le signal en sortie :

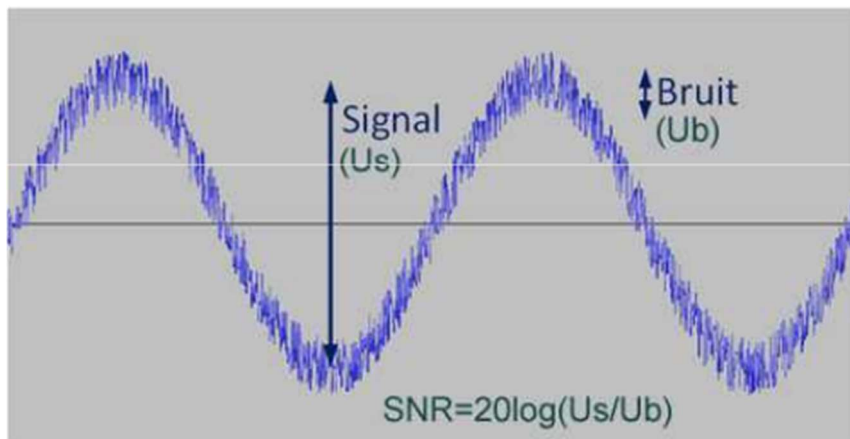


Figure 6: Courbe d'un signal sinusoïdal de fréquence 1 kHz

Puis nous avons comparé ces valeurs à la même mesure mais cette fois-ci analysée sur Audacity à l'aide d'un fichier .wav enregistré sur Linphone.

Nous avons donc importé nos signaux enregistrés sur Audacity et avons analysé des signaux semblables au signal suivant :

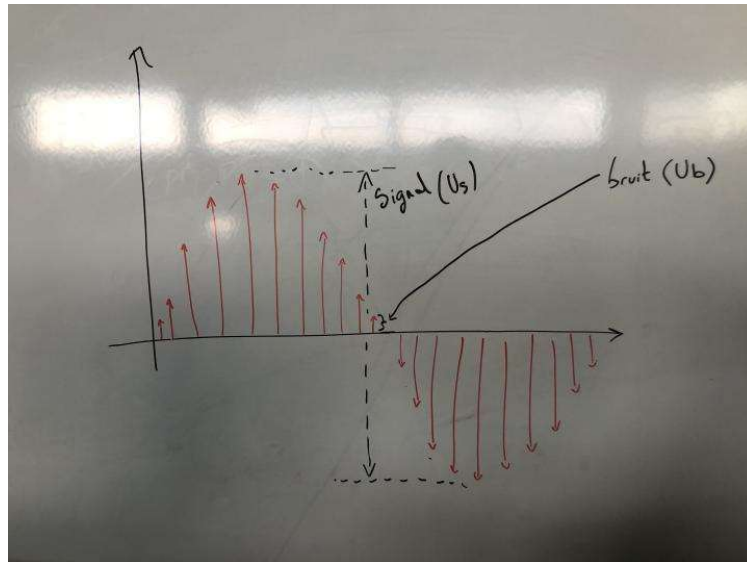


Figure 7: Schéma de la courbe et de l'échantillonnage d'un signal sinusoïdal de fréquence 1 kHz

Le bruit correspond à la valeur du quantum. Cette mesure va nous permettre d'avoir une autre approche du rapport signal sur bruit. On va uniquement voir le bruit de quantification mais pas le bruit blanc qui s'y ajoute lors de la mesure observée à l'oscilloscope.

Dans Audacity, nous avons obtenu :

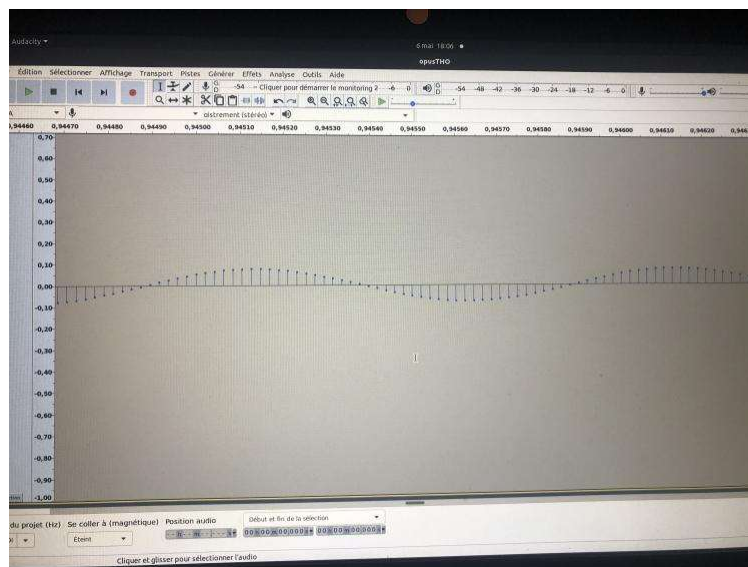


Figure 8: Représentation de la courbe et de l'échantillonnage d'un signal sinusoïdal

Tandis que les mesures que nous avons prises correspondent à :

Codec	Opus	PCMU	GSM	SPEEX 16kHz
Bruit (mV)	2,9	2,8	4	6
Amplitude (mV)	156	144	144	160
SNR (dB)	34,614532	34,2240892	31,12605	28,51937465

Tableau 9: Mesure correspondantes à la courbe d'un signal sinusoïdal en fonction du codec

Nous savons qu'un bruit élevé est préférable afin de distinguer un son de manière optimale. Nous pouvons le prouver avec cette image :

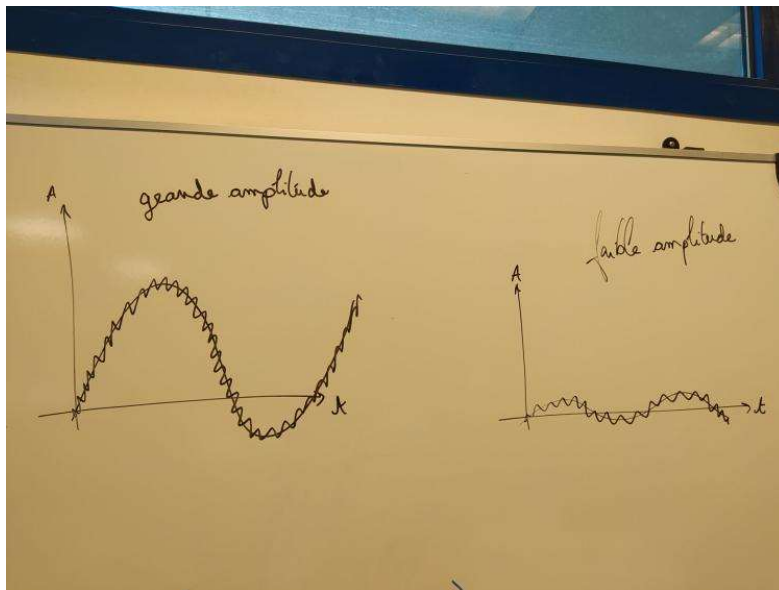


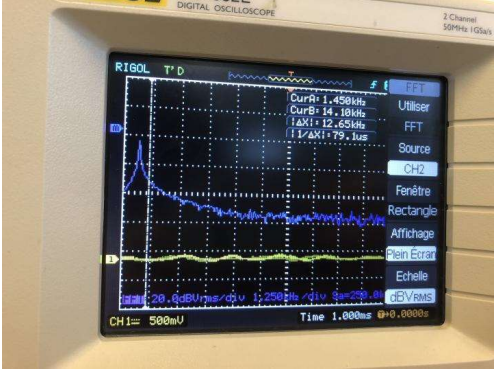
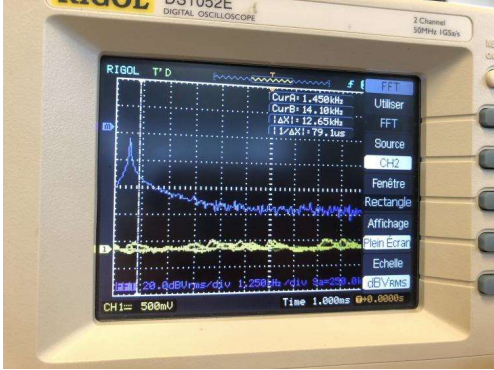
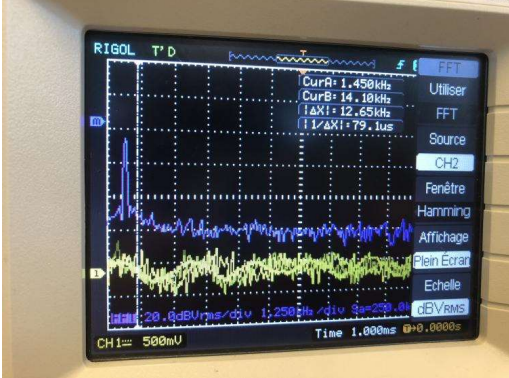
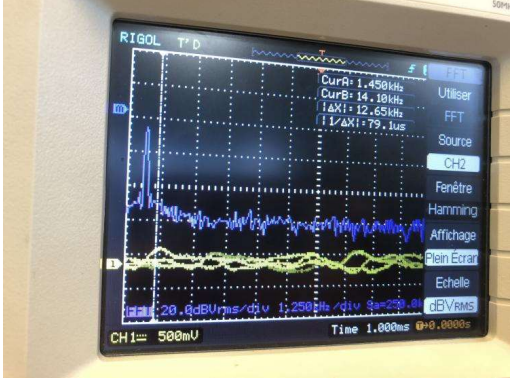
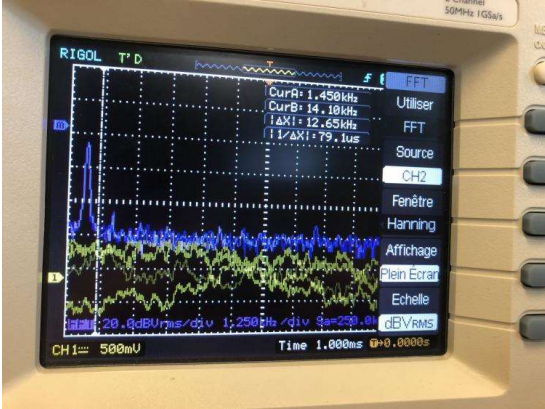
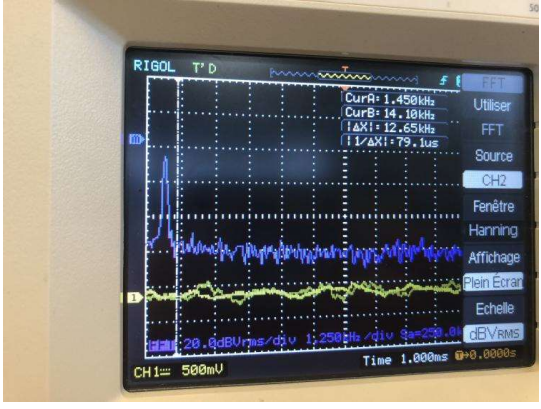
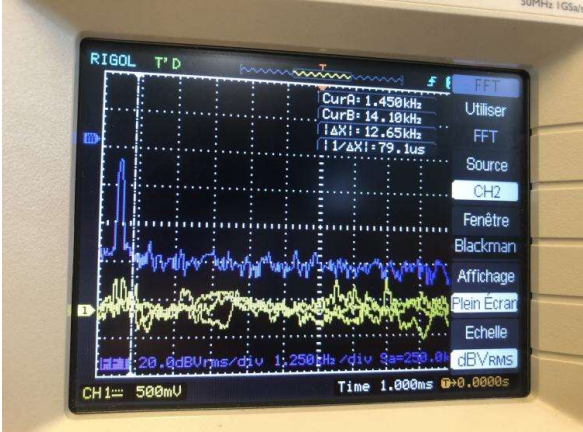
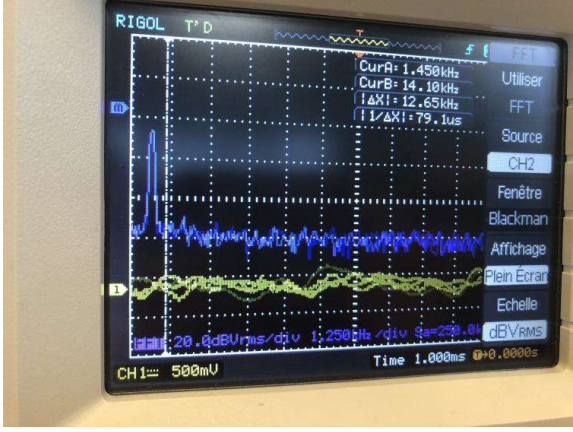
Figure 10: Schéma de deux courbes d'amplitudes différentes comportant du bruit

En effet on voit que si l'amplitude est faible, on ne peut presque pas distinguer le signal. Dans ce cas on aura un rapport signal sur bruit faible ce qui n'est pas préférable. A contrario, si l'amplitude est forte, on aura un rapport signal sur bruit élevé.

D'après nos mesures à l'oscilloscope, on peut noter que le meilleur rapport sur bruit est obtenu avec le codec PCMU, puis le GSM, Speex et enfin Opus. On remarque lors de l'utilisation de Audacity que nos mesures précédentes changent. En effet, le rapport signal sur bruit a augmenté puisque ce logiciel supprime le bruit blanc. Il ne reste donc que le bruit de quantification. Cependant après ces mesures, le classement des codecs change, le plus grand signal sur bruit sera le codec Opus, puis le PCMU, GSM et enfin Speex. En utilisant des supports de transmission et avoir un meilleur son en sorti lors d'appels par exemple. D'après nos mesures, nous pouvons en conclure que le meilleur codec à utiliser est le codec Opus.

9 Mesures de spectres

Nous avons utilisé un oscilloscope pour analyser les mesures d'un signal du codec Opus que nous avons envoyé à l'aide d'un téléphone dans la prise micro de l'ordinateur utilisé comme appelant. Nous avons récupéré ce signal directement en sortie du téléphone et nous l'avons comparé avec le signal obtenu en sortie par l'ordinateur appelé.

Types de fenêtre	En sortie directe d'un téléphone portable	En sortie d'un l'appel téléphonique sur l'ordinateur appelé
Rectangulaire		
Hamming		
Hanning		
Blackman		

Nous pouvons remarquer l'utilisation d'une FFT sur un signal. Le type de fenêtre diffère et chacune à un avantage particulier. On peut se rendre compte que la fenêtre rectangulaire affaiblit les raies harmoniques et nous pouvons bien étudier la fondamentale avec sa largeur.

Tandis que pour les autres types de fenêtres (Hamming, Hanning, Blackman) la fondamentale à d'être moins haute, on amplifie les harmoniques qui sont en grande partie dues aux choix de la fenêtre.

⇒ On se rend compte si l'on compare les résultats des fenêtres en sortie directe d'un téléphone portable ou à la sortie d'un appel téléphonique il n'y a que peu de différences. Le signal est passé par deux ordinateurs et a subi des modifications. On se rend peu compte de changements dans le domaine fréquentiel.

10 Comparaison audio

Dans cette partie, nous avons noté sur 5 points la qualité audio de chaque codec en écoutant la même musique. Nous avons travaillé avec un autre groupe (groupe 2) pour avoir un avis plus précis sur chacun des différents codecs et nous avons recensés ces notes dans le tableau suivant.

Codecs/numéro étudiant	1	2	3	4	5	6	Moyenne
Opus	4	4	4	2	4	3	3,5
Speex-16k	2	2	2,5	2	2	2	2,08333333
PCMU	2	2	2	2	2	1	1,83333333
GSM	1	1	1	1	1	1	1

Tableau 9: Notes données après l'écoute d'une même musique en fonction du codec

D'après ces moyennes, le codec présentant la meilleure qualité audio est le code opus.

11 Intérêt du codec

Données :

Codec	Echantillonnage par seconde
GSM	8000
Opus	48000
PCMU	8000
Speex	16000

Tableau 10: Echantillonnage par seconde en fonction du codec

Nous savons aussi que le MTU (Maximum Transmission Unit) est de 1500 octets, ce qui fait 12000 bits (1500*8).

L'interface travaille à 100Mbps/s.

Une trame IP à une taille totale de 1526 octets, ce qui donne 12208 bits = 0.012208 Mbits.

Une trame est transmise à 100Mbits/s (débit demandé dans l'énoncé).

Application numérique :

- Délais : temps d'envoi d'une seule trame = taille de paquet divisé par le débit. Ce qui donne :
 $= 0,012208 / 100 = 0.00012208 \text{ s} = 122.08 \mu\text{s}$
- Calcul du nombre de paquets nécessaires pour une seconde échantillonnée = $F_e / \text{taille MTU (en octets)}$ = $F_e / 1500$ (il faut arrondir au-dessus) = $8000 / 1500 = 5,3333$ trames.
- On fait ensuite une division euclidienne entre le total échantillonné et le MTU. On obtient 5 trames complètes de 4000 bits restant.
- On multiplie le nombre de trames complètes par 1526 (nbr total d'octets) puis par 8 pour avoir le nombre total de bits $1526 * 8$, soit 12 208 bits. Et on y ajoute la trame restante (4000 bits + $26 * 8 = 4208$ bits), soit 14416 bits.
- On cherche maintenant le temps d'envoi du total en divisant le poids total en MB par le débit ;
Ce qui nous donne un temps de $0,00014416 \text{ s} = 0.144 \text{ ms} = 144 \mu\text{s}$.

On applique la même méthode pour les autres codecs

- Pour Opus 48kHz:
On a 32 trames.
On obtient donc $32 * 1526 * 8 = 390656$ bits.
Ce qui donne 0.390656 Mbits.
Et pour obtenir le temps en secondes pour envoyer les 390626 bits, on fait $0.390626 / 100$ ce qui fait 3,9 ms.
- On obtient le même résultat pour le codec PCMU que pour le codec Opus.
- Pour Speex 16kHz en faisant la même méthode, on obtient 1,3ms.
 - ⇒ Pour conclure, on peut dire que Plus l'échantillonnage du codec est élevé, plus le temps d'envoi est élevé.
 - ⇒ L'intérêt du codec est de moduler le temps d'envoi en fonction de l'échantillonnage, ce qui est important lors d'un appel téléphonique pour avoir un bon débit. Plus l'échantillonnage sera bas plus on perdra des données.