# Traitement du signal - AUDACITY

**« *Audacity****est un logiciel d’édition audio ainsi que d’enregistrement de sons numériques. Totalement gratuit, ce logiciel open-source est disponible pour Windows, MacOS, Linux et d’autres systèmes d’exploitation de type Unix. Audacity fut créé en 1999 par Dominic Mazzoni et Roger Dannenberg puis lancé en mai 2000 dans sa version 0.8.* » **source : Audacity.fr**

**Pour plus d’infos ou pour télécharger le logiciel :**

[**https://audacity.fr/**](https://audacity.fr/)

Il existe aussi une version sous Unix, pour l’installer : « **sudo apt-get install audacity** ».

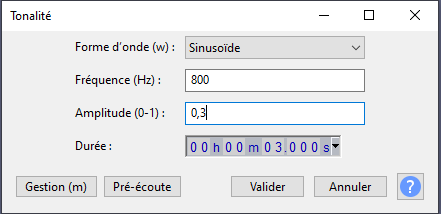
Dans cette étude expérimentale, le logiciel Audacity nous permettra de créer des sons numériques simples pour analyser leur constitution.

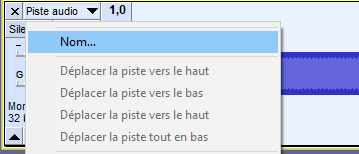
Nous utiliserons aussi un logiciel d’édition binaire et hexadécimal de fichier afin d’observer le codage d’un fichier en hexadécimal. Nous utiliserons « **HexEdit** » sur les ordinateurs du lycée.

Pour les machines sous Unix, nous pouvons utiliser « **Bless** », pour l’installer : « **sudo apt-get install bless**».

## Signal sinusoïdal

* **Ouvrir** Audacity.
* **Cliquer** sur « Générer/Tonalité » pour générer un signal avec les caractéristiques suivantes :

**Sinusoïde de fréquence 800Hz, de durée 3s et d’amplitude 0,3.**

* **Renommer** le signal en haut à gauche par sa fréquence (« 800 »).
* **Écouter** le signal en appuyant sur « Espace » ou sur le bouton « lecture ».
* Avec l'outil loupe ou la molette de la souris**, zoomer** au début du signal (à t = 0s) et **mesurer** la période du signal.

Question 01 : **Calculer** alors la fréquence de ce signal pour vérifier que f = 800Hz.

Ce son a été créé numériquement. Il n’est donc pas continu dans le temps et possède donc un nombre d’échantillon défini. Pour quantifier ce nombre d’échantillons, on parle de fréquence d’échantillonnage. Ici, elle est de 44100Hz, cela signifie qu’il y a 44100 échantillons par seconde.

Question 02 : **Calculer** le nombre d’échantillon présents sur les 3 secondes du signal.

Question 03 : **Calculer** le nombre de période de sinusoïde du signal sur ces 3 secondes.

Question 04 : **Calculer** maintenant le nombre d’échantillon par période du signal.

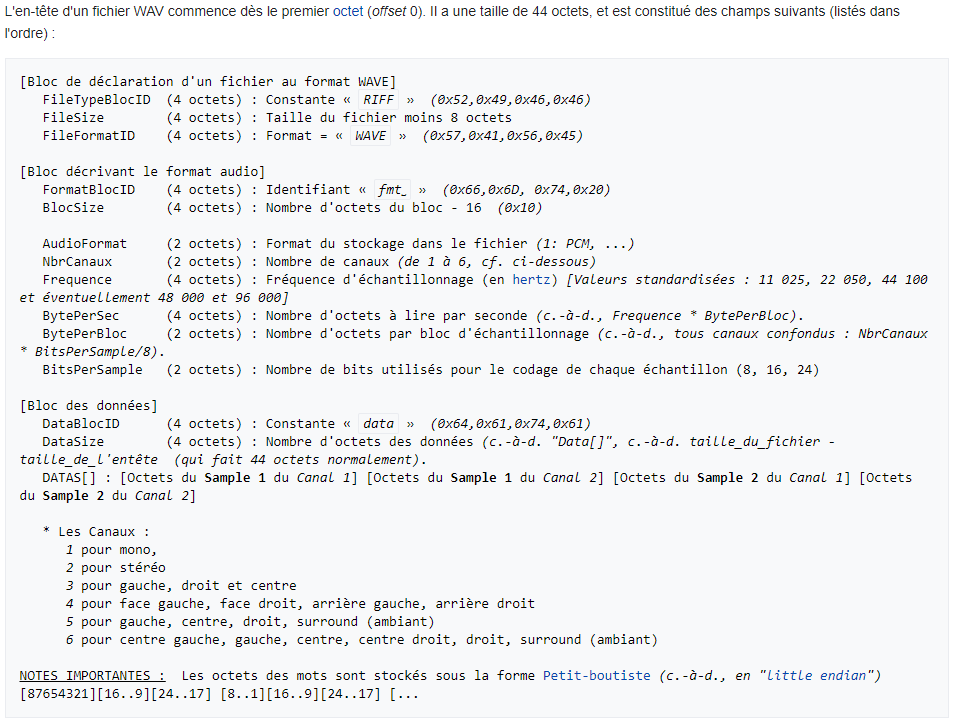
Un échantillon est un morceau de signal où sera codé son amplitude à un instant t. Cette amplitude est alors codée sur un certain nombre de bits (en général 8, 16 ou 32). On parle alors de résolution binaire. Ici, la résolution binaire est de 32 bits, cela signifie que chaque échantillon de ce signal est codé sur 32 bits.

Question 05 : **Calculer** la taille que le fichier du signal prend sur 3 secondes puis **convertir** le résultat en octets.

* Dans le menu « Fichier », **exporter** le signal en « .WAV » avec un encodage « Signed 32-bits PCM » et, en cliquant droit sur le fichier exporté aller dans les propriétés.

Question 06 : **Relever** la taille du fichier en octets observé dans les propriétés.

Question 07 : **Calculer** la différence en octet entre la taille du fichier relevée (question 06) et celle calculée (question 05).



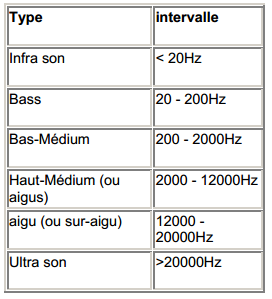
Question 08 : Grâce au détail ci-dessus de la constitution d’un fichier WAV, **expliquer** d’où provient la différence de taille du fichier trouvé à la question précédente.

* En gardant le signal précédent, **générer** les signaux suivants sur Audacity en les renommant par leur fréquence :
  + Sinusoïde, 5 Hz, 3s, amplitude 0,6.
  + Sinusoïde, 20000 Hz, 3s, amplitude 0,6.
* **Ecouter** les signaux un par un en activant le mode solo pour chaque piste avant de l’écouter.

Question 09 : **Expliquer** ce qu’il se passe lors de l’écoute de ces sons.

* **Ecouter** maintenant le signal « 5 » et le signal « 20000 » en même temps en rendant silencieux le signal « 800 ».

Question 10 : **Expliquer** ce qu’il se passe lors de l’écoute des deux sons en même temps.



* **Créer** des signaux sinus avec différentes fréquences et **écouter** leur combinaison (par exemple 2Hz, 5Hz et 200Hz). Il est aussi possible d’**observer** la forme du signal combiné en sélectionnant les pistes à combiner et allant dans « Piste/Mix/Mix et rendu vers une nouvelle piste ».

## Signal carré

* **Supprimer** tous vos signaux.
* **Générer** un signal carré de fréquence 4410 Hz, d’amplitude 0.6 et d’une durée de 3 secondes.
* **Zoomer** pour observer les échantillons.

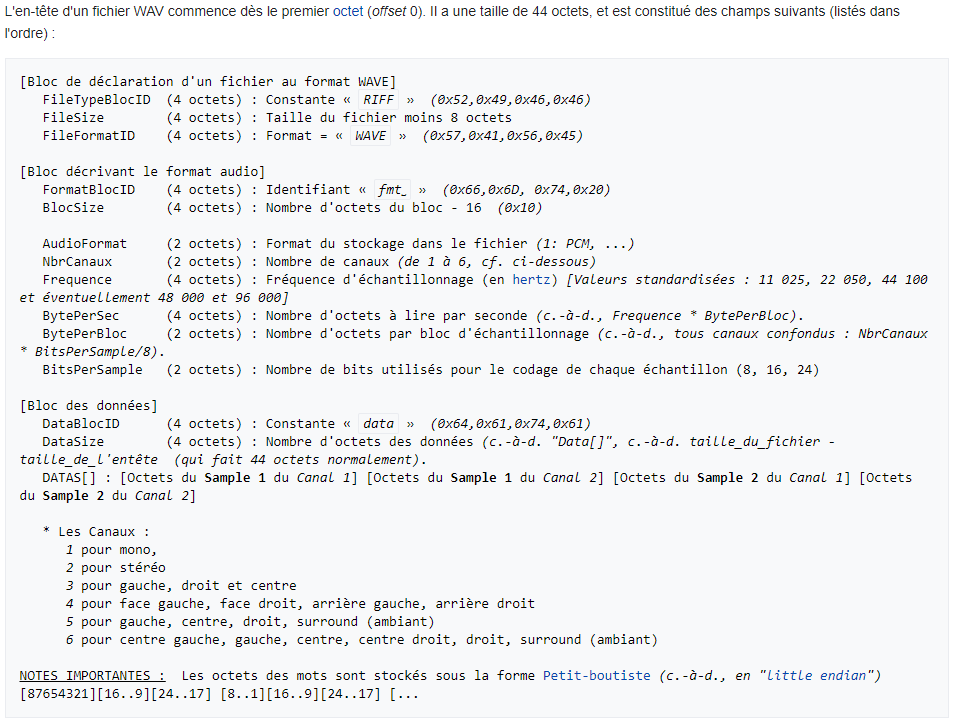
Question 11 : **Calculer** le nombre d’échantillon par période.

Question 12 : Ce signal carré à la moitié de sa période en amplitude maximale et l’autre moitié en amplitude minimale. **Donner** la valeur de l’amplitude maximale et l’amplitude minimale.

* **Exporter** le son en fichier WAV avec un encodage « signed 16-bits PCM ».
* **Ouvrir** le fichier exporté avec le logiciel d’édition hexadécimal (HexEdit ou Bless pour Raspberry).

Question 13 : Par une capture d’écran, **relever** les 10 premières lignes de code hexadécimal et ASCII du fichier qui apparait sur le logiciel d’édition hexadécimal.

Question 14 : **Entourer** en rouge sur la capture d’écran les octets affichés en hexadécimal et en ASCII qui composent l’en-tête du fichier WAV.



Question 15 : Grâce au détail ci-dessus de la constitution d’un fichier WAV, **relever** les octets en hexadécimal qui codent le nombre d’octets des données du fichier (DataSize).

Question 16 : **Convertir** ces octets en décimal en prenant soin de **remettre** les octets dans le bon ordre.

**Attention :** Pour lire correctement une suite d’octet dans le codage d’un fichier WAV il faut effectuer quelques manipulations.

Par exemple :

* Un nombre codé sur 32 bits soit 4 octets « 89 AB CD EF » devra se lire « EF CD AB 89 ».
* Un nombre codé sur 16 bits soit 2 octets « 89 AB » devra se lire « AB 89 ».

Question 17 : **Relever** la taille du fichier dans ses propriétés et la **comparer** avec le résultat précédent.

Le son a été exporté en encodage « signe 16-bits PCM », cela veut dire qu’un échantillon est codé sur 16 bits soit 2 octets.

Question 18 : **Relever** les octets des 10 premiers échantillons.

Question 19 : **Expliquer** ce qu’il est possible de remarquer sur ces octets de donnée.

Question 20 : **Convertir** les 2 octets du premier échantillon en décimal en prenant soin de **remettre** les octets dans le bon ordre.

Question 21 : Chaque échantillon étant codé sur 16 bits, **calculer** le nombre de valeur qu’un échantillon peut prendre.

Un fichier encodé en « signed 16-bits PCM » veut dire que chaque échantillon est codé sur 16 bits et qu’il peut avoir une valeur négative (« signed » veut dire « signé », un nombre signé est un nombre avec un signe positif ou négatif).

Ce codage de nombre signé permet donc d’aller d’une valeur négative à une valeur positive. Par exemple en encodage « signed 8-bits » on code des valeurs allant de -128 à +127 (soit 28 = 256 valeurs possibles).

Question 22 : Avec un encodage « signed 16-bits », donner la valeur minimale et la valeur maximale qu’il est possible de coder.

Les valeurs que prend un échantillon correspond à l’amplitude du signal à l’instant où se situe l’échantillon. Sur Audacity, l’amplitude du signal peut varier entre un minimum : -1 et un maximum : 1.

Question 23 : En regardant le premier échantillon sur Audacity, **donner** son amplitude.

En binaire signé, on reconnait un nombre positif si le bit de poids fort est à 0.

Question 24 : **Retrouver** (en effectuant un produit en croix par rapport aux amplitudes maximales et aux amplitudes mesurées) la correspondance de l’amplitude trouvée à la question 20 et celle trouvée à la question 23.

Question 25 : **Relever** les 2 octets codant l’amplitude du sixième échantillon.

Question 26 : En regardant le sixième échantillon sur Audacity, **donner** son amplitude.

En binaire signé, on reconnait un nombre négatif si le bit de poids fort est à 1. Il est possible d’observer le code binaire du fichier dans le logiciel d’édition hexadécimal.

Si le bit de poids fort est à 1 il faut alors effectuer plusieurs opérations pour trouver le nombre décimal correspondant et pas seulement le convertir. Il faut :

* Prendre le nombre binaire de la donnée.
* Effectuer l’opération « NOT » sur ce nombre (inverser chaque bit).
* Soustraire 1 au nombre binaire.
* Convertir le nombre binaire obtenu en décimal.

On obtient une valeur décimal positive, il suffit alors de rajouter un signe négatif devant ce nombre.

Cette méthode est appelée : « complément à deux ».

Question 27 : Grâce à la méthode précédente, **convertir** les 2 octets codant l’amplitude du sixième échantillon en indiquant chaque étape.

Question 28 : **Retrouver** la correspondance de l’amplitude trouvée à la question 25 et celle trouvée à la question 26.