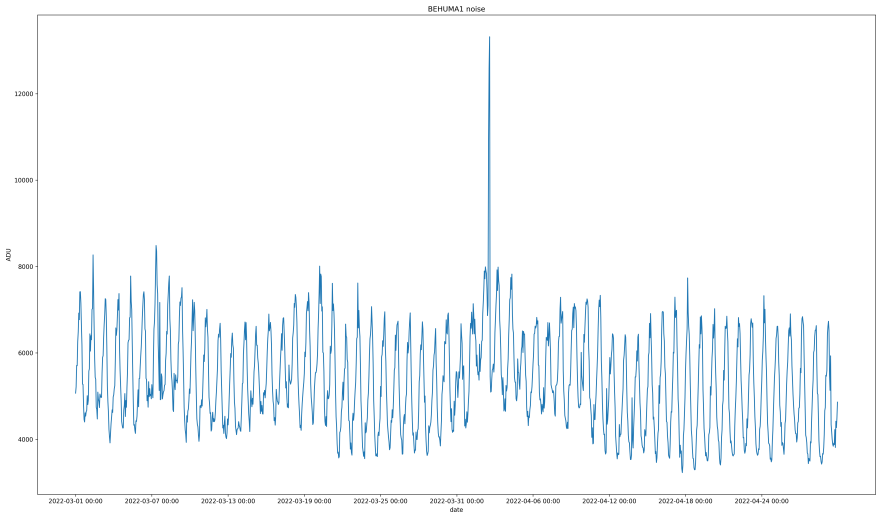
Technologie de l'Informatique
Avenue du Ciseau 15
1348 Ottignies

Monitoring des données BRAMS et détection automatique des échos de météore

Miguel Antoons

Rapporteur
Monsieur Arnaud Dewulf



2021-2022

Remerciements

Je tiens tout d'abord à remercier toutes les personnes qui m'ont aidé à réaliser ce projet de fin d'études.

En commençant par mon professeur rapporteur, à qui j'ai pu poser mes questions en cas de besoin et qui s'est assuré que tout se passe bien tout au long du projet.

Ensuite, je voudrai remercier Mr Hervé Lamy pour avoir proposé ce sujet de fin d'études, mais également pour m'avoir expliqué, de façon claire et précise, toutes les notions qui nécessitaient des explications.

Je tiens également à remercier Mrs Antoine Calegaro et Michel Anciaux, qui m'ont guidé quand c'était nécessaire et qui m'ont conseillé durant le projet de fin d'études.

Enfin, je voudrais exprimer ma reconnaissance envers toutes les personnes qui m'ont conseillé sur, et ont relu ce rapport de projet de fin d'études.

Contents

1 Introduction

1.1 Le projet BRAMS

Lancé en 2010 par Monsieur Hervé Lamy à l'Institut Royal d'Aéronomie Spatiale de Belgique, le projet BRAMS (Belgian RAdio Meteor Stations) à pour but de collecter et stocker des données relatives à des objets rentrant dans l'atmosphère belge et, plus précisément, des météores. Ces données pourront ensuite être analysés afin de retrouver la trajectoire, la vitesse ou encore la masse d'un ou de plusieurs météores.

1.1.1 Le Réseau BRAMS

Afin d'accomplir ce but, le projet BRAMS dispose d'un réseau de stations nommé le réseau BRAMS. Ce réseau est composé d'un ensemble de quarante-deux stations émettrices, situées en Belgique ou dans les pays avoisinants, et d'un émetteur dédié situé à Dourbes, dans le sud de la Belgique.

Il existe 2 types de station de réception :

- Il y a d'abord les stations avec des récepteurs Icom IC-R75, que j'apellerai les récepteurs ICOM dans ce document. Ce sont les premières stations réceptrices mises en service pour le réseau BRAMS. Actuellement, ces stations ne sont plus utilisées suite à l'arrêt de la commercialisation du récepteur Icom IC-R75. De plus, une variation de température pouvait causer une légère déstabilisation en fréquence.
- C'est alors que les stations utilisant le récepteur RSP 2 ont été développés. Ces stations n'éliminaient pas seulement en grande partie les problèmes des anciennes stations, mais sont également plus compacts et plus faciles à installer. Une image des stations RSP 2 peut être trouvé à la figure 1.

1.1.2 Le Format WAV

Le format de fichier WAV (Waveform Audio File) est un format destiné au stockage de signal audio développé par Microsoft et IBM. Il est construit conformément au RIFF (Ressource Interchange File Format) ce qui veut dire que le fichier est organisé en blocs de données, aussi nommés des "chunks" ou "data chunks".

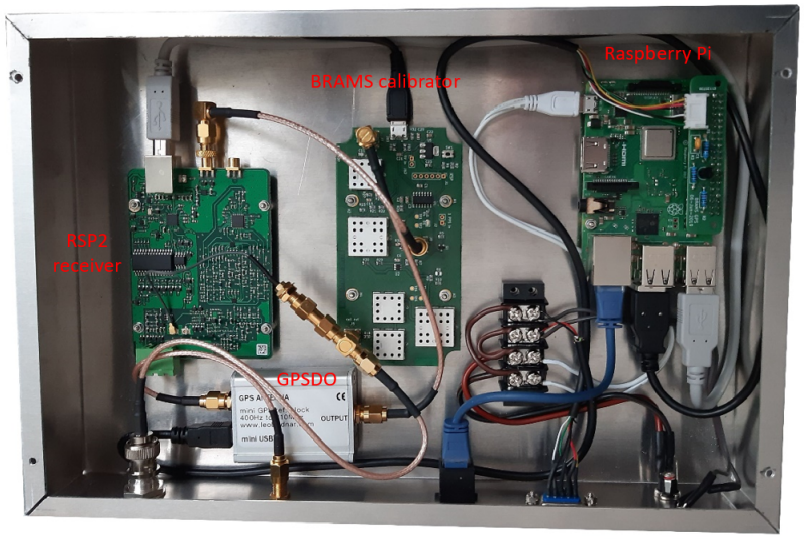


Figure 1: Station RSP 2 du réseau BRAMS

Chaque bloc de données dispose d'un ID codé sur 4 octets, qui représente souvent un mot de 4 lettres. Suit ensuite le champ "ChunkSize" indiquant la taille des données à venir dans l'ensemble du fichier, codé également sur 4 octets. Cette valeur exclut donc les champs "ID" et "ChunkSize" ainsi que la taille de tous les éventuels blocs de données venant avant le bloc courant. Après ces 2 champs, on est libre de rajouter le nombre de champs que l'on souhaite, avec la taille en octets que l'on souhaite.

Pour un fichier WAV, on retrouve typiquement 3 blocs de données. On retrouve premièrement le bloc "RIFF" qui contient la taille de l'entièreté du fichier moins 8 octets (4 octets du champ "ID" et 4 octets du champ "ChunkSize") suivi du champ indiquant le format du fichier. Dans le cas du fichier WAV, ce dernier contient les quatre lettres "WAVE".

Le deuxième bloc, nommé le bloc FMT[1], contenu dans un fichier WAV ordinaire contient toutes les données techniques des données. On y trouve notamment la fréquence d'échantillonnage, le nombre de pistes audio ou encore le nombre d'octets par secondes.

[1]FMT pour format

Nom du Bloc	Nom du Champ	Position (en octets)	Taille (en octets)
Bloc RIFF	ID	0	4
	ChunkSize	4	4
	Format	8	4
Bloc FMT contenant les données techniques du fichier WAV.	ID	12	4
	ChunkSize	16	4
	AudioFormat	20	2
	NumChannels	22	2
	SampleRate	24	4
	ByteRate	28	4
	BlockAlign	32	2
	BitsPerSample	34	2
Bloc de Données	ID	36	4
	ChunkSize	40	4
	Data	44	*

Figure 2: Structure d'un fichier WAV ordinaire

Vient enfin le bloc principal du fichier : le bloc de données. C'est dans ce bloc que se trouvent les données audio brutes. La taille de ce bloc varie bien-entendu selon les caractéristiques techniques des données audio (longueur, fréquence d'échantillonnage, nombre de pistes, etc.).

Une chose à noter à propos des fichiers WAV est que leur structure offre une grande flexibilité. En effet, elle permet non-seulement de lire et interpréter les fichiers facilement, mais également de rajouter des blocs de données personnalisés sans corrompre les autres données.

La figure 2 montre la structure d'un fichier WAV ordinaire.

1.1.3 La Réflexion Spéculaire

Pour comprendre le fonctionnement du réseau BRAMS, la première chose à savoir est : lorsqu'un météore passe dans l'atmosphère, il laisse derrière lui une trainée ionisée. Cette trainée à la propriété de réfléchir les ondes radio. Dans la plus grande partie des cas cette réflexion se fait en un seul point le long de la trajectoire, phénomène qui s'appelle la réflexion spéculaire (illustré à la figure 3). La position de ce point, nommé le point de réflexion spéculaire, dépend notamment du positionnement de l'émetteur, du positionnement du récepteur et de la trajectoire du météore.

Pour exploiter le phénomène de la réflexion spéculaire et réussir à détecter les météores à l'aide d'ondes radios, le réseau BRAMS fonctionne de la façon suivante :

1. Un émetteur, situé à Dourbes, transmet de façon continue un signal à une fréquence de 49.97 MHz. Ce signal est émis en direction du ciel et peut être réfléchi sur des trainées ionisées dans le sillage des météores.

2. Lorsque le signal est réfléchi, il peut être détecté par une ou plusieurs stations réceptrices faisant partie du réseau BRAMS. Un signal calibreur est alors additionné au signal venant du ciel. Ce signal est injecté à 49.9705 MHz, c'est-à-dire 500 Hz plus haut que le signal direct. Il dispose d'une amplitude fixe et sert de référence d'amplitude pour le reste du signal. La station réceptrice décale ensuite le signal direct de 49.97 MHz vers une fréquence de 1 kHz.

3. Ensuite, la station réceptrice enregistre l'ensemble du signal dans un fichier audio de type WAV. Le signal est échantillonné à une fréquence de 5512,5 Hz pour les stations ICOM et 6048 Hz pour les stations RSP2. Si tout se passe bien, un fichier est généré toutes les cinq minutes et chaque fichier devrait commencer et terminer à un temps prédéfini (par exemple : 16 h 00 à 16 h 05, 16 h 05 à 16 h 10, etc.).

4. Tous les fichiers générés par les stations réceptrices sont envoyés à intervalle régulier aux serveurs utilisés pour le projet BRAMS par internet.

5. Une fois sur le serveur, les fichiers WAV sont archivés, manipulés et étudiés par les scientifiques du projet BRAMS afin d'en extraire les informations utiles.

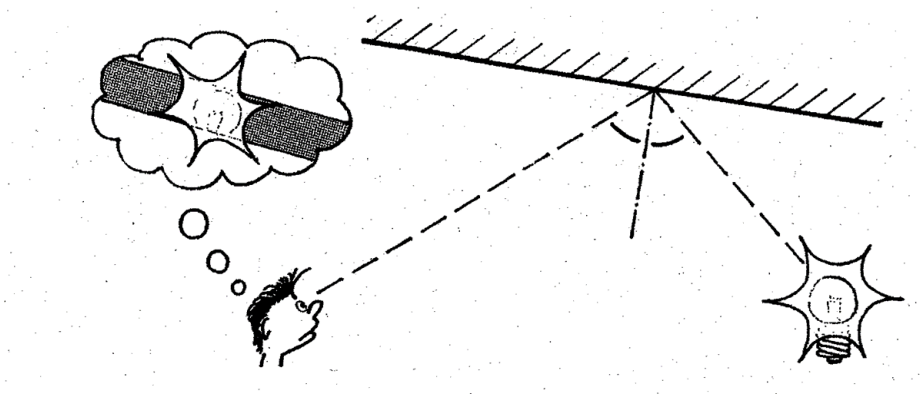


Figure 3: Représentation de la réflexion spéculaire.

1.1.4 Les Données BRAMS

Un fichier WAV venant d'une station réceptrice contient donc le signal capté par l'antenne ensemble avec un signal calibreur. Il est composé d'une seule piste audio, échantillonnée à une fréquence de 5512.5 Hz ou 6048 Hz. Cette fréquence permet d'enregistrer des données dans une bande de fréquences allant jusqu'à 2756.25 Hz ou 3024 Hz (ou la moitié de la fréquence d'échantillonnage), selon le théorème de Nyquist. Sachant que les échos de météores apparaissent typiquement dans une bande de fréquence de 100 Hz autour du signal de l'émetteur à 1000 Hz, cette bande de fréquence couvre l'ensemble des signaux utiles à l'étude des météores.

À chaque fichier WAV est rajouté un bloc de données (data chunk) conçu pour faciliter l'étude des données. Dans ce bloc, on retrouve quelques informations relatives à la station de réception, la station émettrice, le signal GPS[2] ou encore, la fréquence d'échantillonnage. Ce bloc de données se trouve entre le bloc FMT et le bloc de données.

Ces fichiers audio, comme montrée à la figure 4, sont très difficiles à interpréter. Ils sont composés de bruits et de parasites sur l'entièreté de leur bande de fréquence. Par contre, lorsque l'on calcule le spectrogramme du fichier WAV, les données deviennent beaucoup plus simples à lire.

Un spectrogramme est une représentation différente des données contenues

[2]Global Positioning System

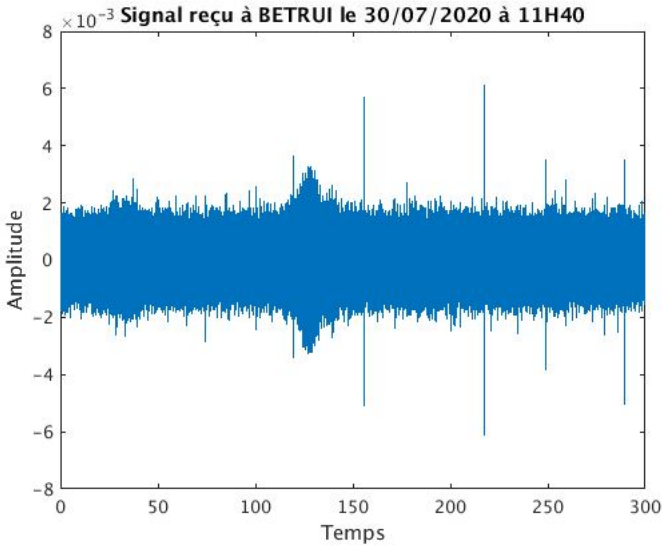


Figure 4: Données brutes venant d'un fichier audio d'une station réceptrice.

dans un fichier audio. Au lieu de représenter la puissance sur l'ordonnée et le temps sur l'abscisse, un spectrogramme représente la répartition des fréquences au cours du temps. Il contient donc trois axes :

- Sur l'abscisse on retrouve toujours le temps.
- Sur l'ordonnée on retrouve la fréquence.
- Sur la cote, on retrouve la puissance. Cet axe est souvent représenté par un code couleur.

Pour générer un spectrogramme, il faut calculer un ensemble de Transformées de Fourrier. La Transformée de Fourrier permet de représenter la répartition de puissance entre les fréquences contenues dans un signal ou une partie d'un signal temporel. Elle produit donc se qu'on appelle le spectre du signal. Elle se calcule sur un nombre quelconque d'échantillons qui se suivent dans un signal temporel. Dans l'informatique, elle est souvent calculée à l'aide de la FFT[3]. C'est un algorithme qui permet de calculer une Transformée de Fourrier plus rapidement, mais qui exige un nombre d'échantillons 2^n.

[3]Faste Fourier transform

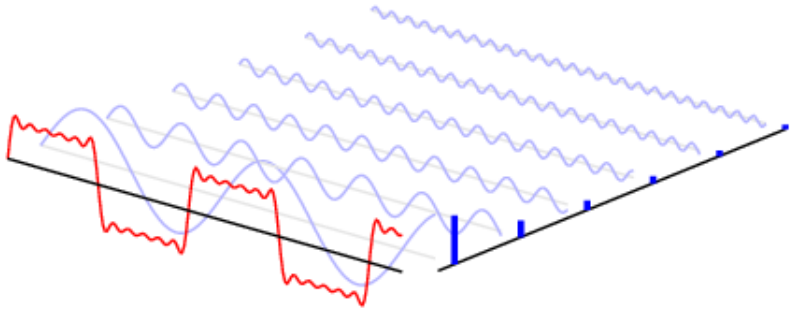


Figure 5: Représentation visuelle de la Transformée de Fourier.

Sur la figure 5, on peut voir en rouge un signal temporel et le spectre de ce même signal en bleu. Les signaux en mauve claire sont les signaux à une fréquence, dont le signal rouge est composé.

Dans le cas des spectrogrammes pour le projet BRAMS, les FFT contenus dans un spectrogramme sont calculés sur 16384 échantillons. Si on suppose qu'un fichier WAV venant d'une station réceptrice dure en moyenne trois-cents secondes (ou cinq minutes), un spectrogramme est composé d'environ 101 spectres pour les stations ICOM et 111 pour les staions RSP2. Ce nombre est obtenu avec la formule ci-dessous, où Fs est la fréquence d'échantillonnage, T la durée en secondes du signal et nfft le nombre d'échantillons utilisés pour générer un spectre.

$$\frac{Fs * T}{nfft}$$

Le spectrogramme généré aura une résolution fréquentielle, donnée par la formule $Fs/nfft$, de 0.34 Hz pour les stations ICOM et 0.37 Hz pour les stations RSP2. Sa résolution temporelle, donnée par la formule $nfft/Fs$, est de 2.97 secondes pour les stations ICOM et de 2.7 secondes pour les stations RSP2. Un exemple de spectrogramme venant d'une station de réception est affiché à la figure 6.

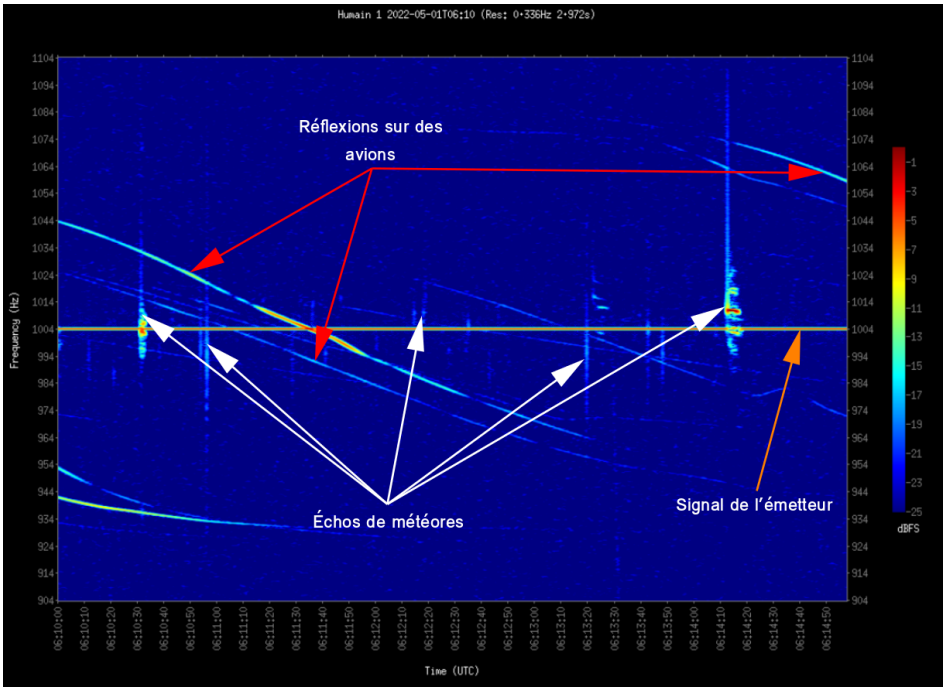


Figure 6: Exemple d'un spectrogramme du projet BRAMS.

1.1.5 L'Archive BRAMS

Quand les fichiers venant des stations de réception sont archivés, ils sont placés dans une structure de répertoires spécifique. Cette structure dépend de la date de début d'enregistrement des données contenues dans le fichier.

Dans la première couche de l'archive, on retrouve un répertoire par station BRAMS. Chaque nom de répertoire est le code de location de la station dont elle contient les données. Ensuite, on retrouve un nombre de répertoires, chacun contenant les fichiers d'une année spécifique. Le nom de ces répertoires est donc l'année des données qu'elles contiennent. Si on rentre dans le répertoire d'une année, on se retrouve dans la troisième couche de l'archive BRAMS. Ici, les données sont encore une fois séparées selon le mois d'enregistrement du fichier. Finalement, dans la quatrième couche de l'archive, on retrouve un répertoire par jour d'enregistrement d'un fichier.

Dans chaque répertoire représentant un jour spécifique, on retrouve les don-

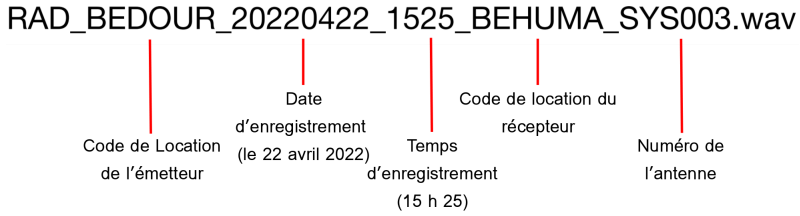


Figure 7: Exemple de nom de fichier BRAMS.

nées BRAMS organisés par heure dans des fichiers de format tar. Ce format permet de regrouper plusieurs fichiers dans un même fichier, qui prend alors le format tar. Le nom des fichiers tar et les fichiers wav qui se trouvent à l'intérieur des fichiers tar suivent également une structure fixe. Il est composé de six informations ordonnées de la manière suivante :

1. Les trois lettres **RAD**.
2. Le code de la station émettrice. Comme il n'y a qu'une station réceptrice à présent, la valeur sera toujours **BEDOUR**.
3. La date du début de l'enregistrement du fichier. Elle respecte le format YYYYMMDD.
4. Le temps du début de l'enregistrement du fichier, précis à la minute. Elle respecte le format HHMM.
5. Le code de location de la station émettrice. Un code de location est composé de deux informations. Les deux premiers caractères représentent le code du pays ou se trouve la station (e.g. BE, FR, ...) et les quatre lettres restantes représentent typiquement les quatre premières lettres de la ville où est situé la station (e.g. HUMA pour Humain, UCCL pour Uccle, ...).
6. L'alias de la station. Elle est composée des trois caractères 'SYS', suivie du numéro de l'antenne qu'a capté les données. Le numéro de l'antenne est toujours écrit sur trois chiffres (e.g. 001, 002, ...).

Toutes ses valeurs sont séparées par un tiret bas. Un exemple de nom de fichier est affichée à la figure 7.

1.2 Problématique liée à l'étude des données BRAMS

Afin de pouvoir trouver la trajectoire d'un météore dans l'atmosphère, il est nécessaire de rechercher par quelles stations réceptrices le météore est détecté. Ceci est une procédure qui prend un temps non négligeable pour les scientifiques du projet. En effet, toutes les stations de réception ne détectent pas tous les météores. De plus, certaines données peuvent être corrompues ou encore, non-existantes.

De plus, savoir, ou au moins avoir une idée, si un météore détecté par une station réceptrice est le même qu'un météore détecté sur une autre station réceptrice serait une grosse économie de temps.

C'est pour cette raison que les membres du projet BRAMS souhaitent disposer d'un programme qui remplit deux fonctions. D'une part, elle doit être capable de vérifier si un fichier contient des données non-corrompues et utilisables. D'autre part, elle devra, sur base d'un écho de météore d'une station de réception, retrouver les échos de ce même météore sur des fichiers d'autres stations de réception.

2 Méthodologie

Afin d'arriver à un résultat final de qualité et qui est conforme aux requis des membres du projet BRAMS, il est important d'utiliser une bonne méthodologie.

La première chose que j'ai fait est : comprendre quoi exactement le programme devrait faire. Pour ce faire, plusieurs réunions ont eu place avant la réalisation du travail. Durant ces réunions, j'ai pu poser mes questions et demander des explications sur les concepts à connaître pour pouvoir réaliser le travail. Ayant reçu des documents qui expliquent de façon claire et précise le fonctionnement du réseau BRAMS, j'ai pu me préparer avant de m'attaquer à l'analyse et la réalisation du travail.

Durant la période d'analyse et de réalisation du TFE, j'ai pu régulièrement demander validation quant à la direction que je prenais pour mon travail. Je présentais régulièrement mes réalisations aux scientifiques du projet BRAMS afin d'avoir un feedback régulier et de pouvoir perfectionner mon programme un maximum.

Dans le but d'être plus efficace lors de l'écriture des fonctionnalités pour le programme, je décrivais avant ce que cette fonctionnalité devait faire. Ensuite, je la divisais en tâches techniques afin de pouvoir m'organiser plus facilement. Chaque tâche technique contenait et expliquait les étapes que le programme devrait exécuter pour accomplir cette tâche technique. Les étapes étaient décrites textuellement, par pseudo-code, par schéma ou encore, par le mélange de deux ou trois des moyens cités.

Durant l'entièreté du projet de fin d'études, j'ai tenté de garder un rythme de travail régulier. Je me suis organisé de telle façon de pouvoir travailler une moyenne de 23 heures par semaine. Une grande partie de ces heures se sont déroulés lorsque je rentrais de mon stage ou pendant le week-end.

3 Technologies utilisées

3.1 Python

Le programme réalisé est entièrement écrit en Python. Ce choix a été pris premièrement pour les librairies performantes et open-source qu'offre Python. En effet, des librairies comme numpy, scipy ou encore matplotib ont été très utiles pour arriver à un résultat performant et fonctionnelle. Ces librairies offrent beaucoup de flexibilité et offrent une haute performance grâce à leur écriture en C, C++ et même en Fortran.

Un autre avantage du Python est sa portabilité. Puisque c'est un langage interprété, largement répandu, quasiment tous les systèmes d'exploitation le supportent. Ceci a facilité aussi bien la phase de développement que la phase de testing du projet de fin d'études.

Un des désavantages souvent évoqués pour Python, est que le langage est lent pour des gros traitement de données. Cependant, dans le cas de ce travail, ceci ne fut pas un problème et les librairies utilisées étaient largement assez rapide. Ceci est dû, entre autres, grâce à la nature open-source de ces librairies ainsi que leur âge. Suite à ces deux facteurs, des milliers de personnes travaillent sur ces librairies depuis plusieurs années dans le but d'optimiser le plus possible ses fonctions et méthodes.

3.2 MariaDB

Afin de sauvegarder les données produites par le programme, une base de données est requise. Les données produites seront toutes liées à un fichier, le choix d'une base de données relationnelle est donc appropriée.

MariaDB est une dérivée de MySQL. Elle a l'avantage d'être gratuite et open-source. Grâce à une large communauté contribuant au développement continu de MariaDB, elle est souvent plus performante que MySQL. De plus, étant open-source, les problèmes sont résolus rapidement par des mises à jour régulières.

Enfin, le projet BRAMS dispose actuellement déjà d'une base de données MariaDB. Elle contient, entre autres, les données de chaque fichier archivé, venant d'une station de réception BRAMS. Pour le programme, il suffit donc de rajouter les champs nécessaires à la table contenant les fichiers. MariaDB est donc le choix logique.

4 Lecture des Fichiers BRAMS

Avant de commencer à développer des outils pour la détection des météores ou pour l'analyse des fichiers BRAMS, il a fallu développer un logiciel permettant de lire ces fichiers. Ce logiciel sera implémenté dans un module Python contenant une classe nommée BramsWavFile. Ceci veut dire qu'il pourra facilement être utilisé par d'autres fichiers Python et qu'il doit donc être flexible. Dans cette section, le fonctionnement de ce module est expliqué en détail.

4.1 Librairies Utilisées

4.1.1 Os

La librairie os de Python contient une suite de fonctions permettant d'interagir avec le système d'exploitation. Cette librairie vient par défaut avec une installation de Python, ce qui évite de devoir l'installer lorsque nécessaire. Pour la classe BramsWavFile, la librairie os sera utilisée afin de récupérer des chemins de dossiers et fichiers, et pour récupérer le contenu d'un dossier.

4.1.2 Numpy

Numpy est une librairie Python utilisée lorsqu'on travaille avec des listes très larges. Il contient de nombreuses fonctions pour l'algèbre linéaire, les Transformées de Fourier et les matrices. Par rapport aux listes classiques de Python, les listes Numpy (aussi appelés des ndarray) sont beaucoup plus rapides.

Ceci est dû à trois raisons principales. La première est que les listes Numpy sont stockés à un endroit continu dans la mémoire RAM[4], ce qui permet un accès et une manipulation plus efficace des données. Ensuite, comme Numpy est une librairie open-source, il est continuellement adapté à des nouvelles architectures de processeurs. Finalement, une grande partie de la librairie Numpy est écrit dans des langages de bas niveau tels que le C, le C++ ou encore le Fortran. Ceci résulte à un code beaucoup plus proche du langage machine et qui est donc plus optimisé.

[4]Random Access Memory

4.1.3 Scipy

Scipy est une librairie qui est basé sur la librairie Numpy. Elle partage donc de nombreuses fonctions et méthodes avec cette dernière et fonctionne également avec des ndarray. Cependant, cette librairie ajoute des fonctions pour la science des données. De plus, Scipy optimise des fonctions déjà présentes dans la librairie Numpy. Ceci permet de construire un programme qui demande moins de ressources machine et qui est donc plus rapide.

4.1.4 Tarfile

Le module Python Tarfile permet de lire et écrire des fichiers du format tar. Cette librairie est incluse par défaut avec Python et ne doit donc pas être installé séparément.

4.1.5 Datetime

La librairie Datetime offre une suite de fonctions et de classes permettant la manipulation et la conversion de temps et de dates. Cette librairie fait partie des modules standards de Python et ne doit donc pas être installé.

4.2 Fonctionnement du Module

Le contenu principal du module est donc la classe BramsWavFile. Cette classe permet de lire et de récupérer des informations sur les fichiers WAV du réseau BRAMS de manière efficace. Elle est basée sur un code écrit par Michel Anciaux, membre du projet BRAMS. Cependant, elle a fortement été retravaillé afin de l'adapter aux besoins du programme et de l'optimiser. La classe prend six paramètres en entrée :

- Le paramètre **date_time** indique le temps et la date du fichier recherché. Ce paramètre est obligatoire et doit être de type datetime, type qui est défini dans le module datetime.
- Le paramètre **station** indique le code de location de la station recherchée (e.g. BEHUMA, BEHAAC, BEUCCL). Les deux premières lettres de ce code représentent le code pays (e.g. BE, FR). Typiquement, les quatre lettres suivantes sont les quatre premières lettres de la ville ou se trouve la station. Ce paramètre est obligatoire et doit être de type str (chaine de caractères).

- Le paramètre **alias** indique l'antenne, se trouvant à la location spécifiée, dont on veut récupérer un fichier. En effet, certaines stations disposent de plusieurs antennes. Les antennes sont spécifiées à l'aide de six caractères : les trois premiers contiennent les lettres 'SYS' et les trois suivants contiennent le numéro de l'antenne (e.g. 001, 002, ...). La valeur par défaut de ce paramètre est 'SYS001', c'est donc un paramètre optionnel. Le type du paramètre est str (chaine de caractères).

- Le paramètre **is_wav** est un booléen indiquant à la classe si le fichier recherché se trouve dans un fichier de format tar (dans quel cas la valeur devrait être 'faux') ou pas (dans quel cas la valeur devrait être 'vrai'). Ce paramètre est optionnel et prend la valeur 'faux' par défaut.

- Le paramètre **parent_directory** indique le répertoire parent ou il faudra commencer la recherche du fichier demandé. C'est un paramètre optionnel de type str (chaine de caractère) qui prend le répertoire de l'archive BRAMS comme valeur par défaut.

- Le paramètre **from_archive** est un booléen indiquant si le fichier recherché se trouve dans un répertoire respectant la structure de l'archive BRAMS (dans quel cas la valeur est 'vrai') ou non (dans quel cas la valeur est 'faux'). Elle prend la valeur 'vrai' par défaut et est donc optionnel.

Lorsque la classe est initialisée, elle vérifie d'abord si le fichier recherché existe. Si le paramètre 'from_archive' contient la valeur 'vrai', la classe rajoute, au chemin spécifié par le paramètre 'parent_directory', les dossiers nécessaires pour que le chemin respecte la structure de l'archive BRAMS. Ensuite, le programme essaye de lister les contenus du dossier. Dans le cas ou le dossier n'est pas trouvé, l'initialisation de la classe est interrompue et l'exception 'DirectoryNotFoundError' est levée.

La classe recherche ensuite pour le bon fichier dans le contenu du répertoire en comparant la date et le temps dans le nom du fichier avec la date et le temps reçu comme paramètre. Quand les fichiers wav sont contenus dans une archive tar, l'étape précédente est répétée pour trouver le bon fichier dans l'archive. Si aucun fichier correspondant à la date et la station reçue en paramètres est trouvé, l'initialisation de la classe est interrompue et l'exception 'BramsError' est levée. Une fois le bon fichier trouvé, toutes les données brutes sont lues et enregistrées dans un buffer (variable).

La prochaine étape est la séparation des différents blocs de données (ou chunk) contenus dans le fichier wav. Pour ça, la classe parcourt le buffer contenant les données du fichier WAV. Il utilise un pointeur de lecture indiquant la position en octets du prochain bloc de données dans le buffer. Ce pointeur est calculé à partir du champ 'subChunkSize' indiquant la longueur du bloc de données courant.

Lorsque le programme retrouve un bloc de données avec un champ ID correspondant à 'fmt ', 'BRA1' ou 'data', il extrait toutes ces données et les place dans un ndarray. Ceci est fait à l'aide de la fonction 'frombuffer' venant de la librairie NumPy. Cette fonction offre la possibilité de convertir, de façon très rapide, des données vers un ndarray. Une des contraintes pour utiliser cette fonction est l'indication du type de données qu'on veut placer dans un ndarray.

Les données contenues dans la classe seront la fréquence d'échantillonnage, extrait du bloc de données 'BRA1', et la piste audio du fichier WAV, extrait du bloc de données 'data'.

4.2.1 La méthode FFT

La classe BramsWavFile dispose d'une méthode publique. Cette méthode se nomme 'FFT' et permet, comme son nom l'indique, de récupérer la Transformée de Fourier de l'entièreté de la piste audio du fichier wav. Ceci est fait à l'aide de la fonction 'rfft' de la librairie Scipy, multipliée par une fenêtre de Hann. La fonction 'rfft' existe également dans la librairie NumPy, cependant dans ce cas de figure la librairie SciPy permettait de calculer la Transformée de Fourrier plus rapidement. Enfin, la fonction FFT retourne la Transformée de Fourier normalisée, son abscisse et sa résolution fréquentielle.

5 Monitoring des Données BRAMS

Comme indiqué précédemment, les fichiers ne contiennent pas toujours des données utilisables. Pour des diverses raisons, il peut parfois arriver que le bruit est trop fort pour pouvoir détecter des météores. Ou encore, lorsque les stations ICOM arrivent en fin de vie, la puissance des données détectées décroît de telle sorte qu'on détecte du bruit seulement. Bref, de nombreux problèmes peuvent survenir avec les données. Pour optimiser la détection de météores, les données corrompus doivent être détectés avant celle-ci. C'est pourquoi, un outil permettant le monitoring des données a été développé.

Cet outil se base sur deux éléments dans les données BRAMS indiquant si un fichier est utile ou non. Le premier est le bruit. En effet, lorsque le bruit augmente ou diminue beaucoup d'un coup ou graduellement sans revenir à un niveau normal, il est fort probable qu'il y ait un problème avec la station réceptrice ou son environnement.

Le deuxième élément permettant de détecter un problème est le signal du calibreur. Le signal calibreur devrait toujours être présent dans un fichier produit par une station réceptrice. Typiquement elle devrait se trouver autour de la fréquence 1500 Hz. Cependant, avec les anciennes stations, elle peut varier jusqu'à 250 Hz dépendant de la chaleur. Dans le cas où elle n'est pas aux alentours de 1500 Hz, la probabilité qu'il y ait une erreur avec la station est grande.

5.1 Librairies Utilisées

5.1.1 Argparse

Argparse est une librairie Python qui permet de facilement créer une interface ergonomique en ligne de commande. On spécifie les paramètres nécessaires et optionnels pour le programme qu'on crée, et lorsqu'on lance le programme, Argparse pourra retrouver les paramètres entrés par l'utilisateur. Il indiquera également une erreur si un paramètre obligatoire manque, ou qu'un type de paramètre n'est pas respecté. Argparse est une librairie installée par défaut avec Python.

5.1.2 Simplejson

Simplejson est une librairie Python offrant la possibilité de facilement écrire et lire des fichiers JSON[5]. Elle vient par défaut avec Python sous le nom 'json'. Cependant, installer Simplejson séparément ajoute l'avantage d'avoir des mises à jour plus régulières et de disposer des dernières fonctionnalités plus rapidement.

5.1.3 Matplotlib

La librairie Python Matplotlib permet de générer des graphiques et de les enregistrer en tant qu'image.

5.1.4 Tqdm

Tqdm est une librairie Python permettant d'afficher des barres de chargements en ligne de commande. Elle permet également de donner une estimation de temps de chargement restant.

5.1.5 Decimal

Le module Decimal contient le type de variable décimale, comme son nom l'indique.

5.1.6 NumPy

Voir chapitre 4.1.2.

5.1.7 Datetime

Voir chapitre 4.1.5.

5.2 Fonctionnement du Programme

Le programme permet donc de mesurer le niveau du bruit et le signal calibreur d'un ou de plusieurs fichiers WAV du projet BRAMS. Il fait ceci en calculant la densité spectrale de puissance de ceux-ci. La densité spectrale de puissance montre la distribution des puissances à travers les différentes fréquences contenues dans le signal de base. Ceci permet de comparer ces différentes valeurs et de remarquer des variations non typiques.

[5] JavaScript Object Notation

6 Détection des Météores

6.1 Librairies Utilisées

6.1.1 Math

6.1.2 Geopy

6.1.3 Arparse

6.1.4 Datetime

6.1.5 Matplotlib

6.1.6 Numpy

6.1.7 Scipy

6.2 Fonctionnement du Programme

7 Améliorations Possibles

8 Conclusion