



Technologie de l'Informatique

Avenue du Ciseau 15

1348 Ottignies

---

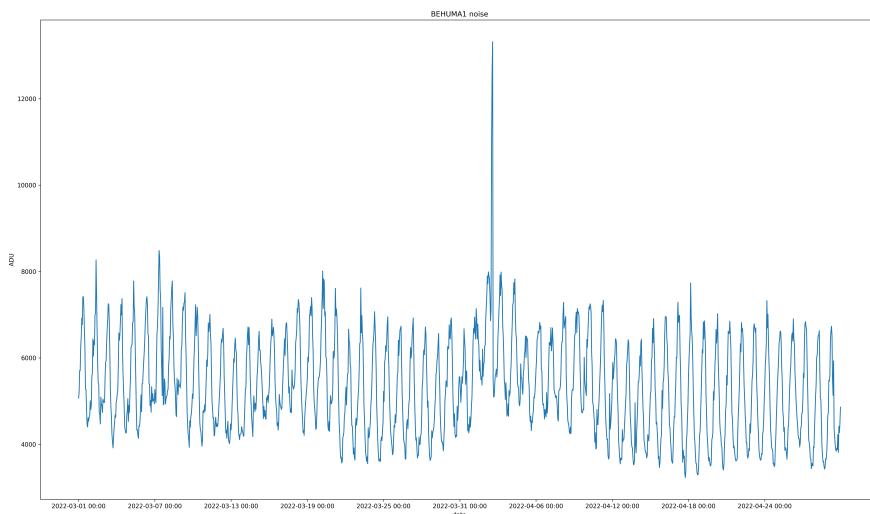
## Monitoring des données BRAMS et déttection automatique des échos de météore

---

*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 voudrais remercier Mr Hervé Lamy d'avoir proposé ce sujet, mais également d'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.*

## Table des matières

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Détection des météores par ondes radios</b>	<b>5</b>
<b>3</b>	<b>Le projet BRAMS</b>	<b>7</b>
3.1	L'Émetteur . . . . .	7
3.2	Les Stations de Réception . . . . .	8
3.2.1	Les Stations V1 . . . . .	8
3.2.2	Les Stations V2 . . . . .	9
3.3	Les Données BRAMS . . . . .	11
3.4	L'Archive BRAMS . . . . .	14
<b>4</b>	<b>Problématique liée à l'étude des données BRAMS</b>	<b>16</b>
<b>5</b>	<b>Méthodologie</b>	<b>17</b>
<b>6</b>	<b>Technologies utilisées</b>	<b>18</b>
6.1	Python . . . . .	18
6.2	MariaDB . . . . .	18
<b>7</b>	<b>Monitoring des Données BRAMS</b>	<b>19</b>
7.1	Déetecter une anomalie sur une station . . . . .	19
7.1.1	Le Bruit . . . . .	20
7.1.2	Le Signal Calibreur . . . . .	21
7.2	Fonctionnement du Logiciel de Surveillance . . . . .	22
7.2.1	Analyse des Fichiers WAV . . . . .	22
7.2.2	L'avertissement d'une anomalie . . . . .	24
7.3	L'interface sur le site web BRAMS . . . . .	26
<b>8</b>	<b>Logiciel de Détection des Météores</b>	<b>29</b>
8.1	Déterminer si un Écho Appartient à un Météore . . . . .	29
8.2	Fonctionnement du Logiciel . . . . .	29
<b>9</b>	<b>Conclusion</b>	<b>35</b>
<b>10</b>	<b>Perspectives</b>	<b>36</b>
<b>11</b>	<b>Annexes</b>	<b>41</b>
<b>A</b>	<b>Images de l'interface web du monitoring</b>	<b>41</b>

<b>B Kernel de Convolution Permettant d'Amplifier les Signaux de Météores</b>	<b>42</b>
<b>C Résultat du filtre pour amplifier les échos de météores</b>	<b>43</b>
<b>D Résultat du filtre à percentile</b>	<b>43</b>
<b>E Résultat des éléments étant trop petits pour pouvoir être un écho</b>	<b>44</b>

# 1 Introduction

Chaque jour, des milliers d'objets passent tout près de l'atmosphère terrestre. Parmi ces objets, on retrouve les météoroïdes : des corps pierreux ou métalliques d'une largeur pouvant varier de quelques millimètres à un mètre. Un météoroïde, une fois rentré dans l'atmosphère, devient un météore et peut créer un phénomène lumineux connu sous le nom d' "Étoile filante". Contrairement à ce que l'on peut penser, un météoroïde qui entre dans l'atmosphère terrestre est un événement qui se produit des milliers de fois par jour.

L'étude de ces météores permet de retrouver différentes informations telles que la masse ou encore la trajectoire de ceux-ci. Ce travail de fin d'étude a pour objectif de faciliter cette étude. Cet objectif sera accompli en permettant une visualisation facile de la qualité des données étudiées et en automatisant une partie de la détection des météores. Mais, afin de pouvoir les étudier, il est nécessaire de les détecter avant. Actuellement, différentes techniques existent pour détecter des météores dans l'atmosphère.

L'une d'entre elles est la détection à l'aide de caméras. Ceci a l'avantage de directement voir la trajectoire du météore et facilite donc l'étude. Cependant, elle a un grand défaut : lorsque le ciel est nuageux ou qu'il fait jour la technique est moins efficace. De plus, lorsqu'un petit météoroïde entre dans l'atmosphère, elle ne produit pas assez de lumière pour pouvoir être détecté par une caméra. C'est alors qu'une autre technique, celle par détection à l'aide d'ondes radio devient intéressante.

Ce travail de fin d'études s'appliquera sur cette deuxième technique. Il accomplira son objectif en automatisant et facilitant deux étapes du traitement des données produites par la détection de météores à l'aide d'ondes radios. Notamment la détection des météores et la détection de données erronées.

À titre plus personnel, ce travail me permettra de prouver mes capacités à réaliser un projet d'une grande ampleur en autonomie. De plus, il aidera également à mieux m'orienter dans les années à suivre.

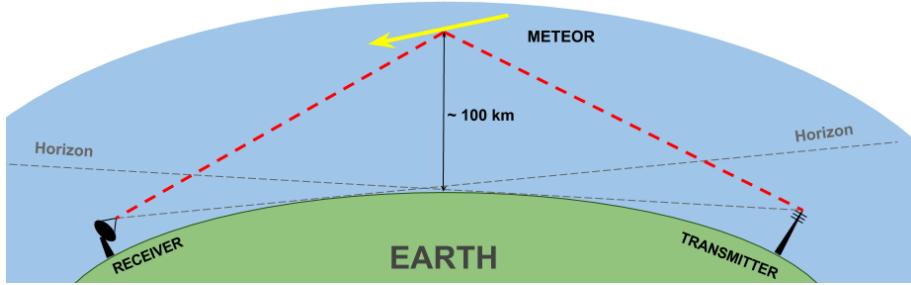


FIGURE 1 – Détection d'un météore à l'aide d'ondes radios.

## 2 Détection des météores par ondes radios

Quand un météoroïde entre dans la partie haute de l'atmosphère (approximativement à 80 - 120 km de la surface terrestre), il laisse derrière lui une trainée ionisée. Cette trainée a la propriété de réfléchir les ondes radio. On peut donc détecter un météore à l'aide de sa trainée ionisée.

Afin d'exploiter cette réflexion, il nous faut un émetteur dont le signal radio est réfléchi à l'aide de la trainée d'un météore et est ensuite enregistré par un récepteur. Cette procédure est illustrée à la figure 1, où la flèche jaune représente la trainée ionisée produite par le météore. Le signal reçu par l'émetteur est alors appelé un écho de météore. Un écho de météore peut durer entre 1 et 10 secondes, selon la durée d'existence de la trainée ionisée.

Une caractéristique importante de cette technique de détection est la réflexion spéculaire. Ceci veut dire que la trainée ionisée du météore agit comme un miroir sur lequel la réflexion se produit uniquement à un point précis, appelé le point de réflexion spéculaire. Le point de réflexion spéculaire dépend de la position de l'émetteur, la position du récepteur et la trajectoire du météore. La conséquence est que les données reçues par un récepteur particulier sont relatives qu'à une partie précise de la trainée.

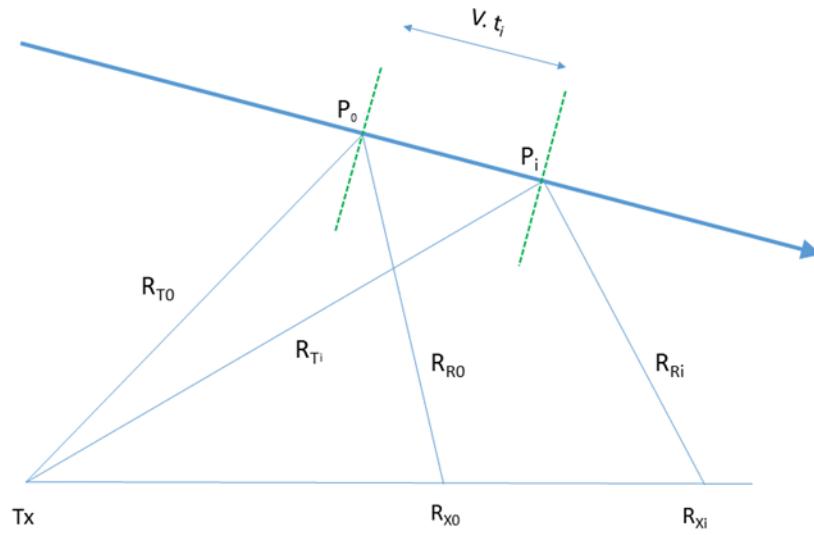


FIGURE 2 – Le point de réflexion spéculaire est différent pour chaque station réceptrice.

De plus, deux récepteurs, situés à des endroits différents, enregistreront un écho de météore à des instants différents puisque leurs points de réflexion spéculaire sont situés à des endroits différents sur la trajectoire. Ceci est illustré à la figure 2 où le point de réflexion  $P_0$  entre le transmetteur  $Tx$  et le récepteur  $Rx_0$  se situe à un endroit différent que le point de réflexion  $P_1$  entre le transmetteur et le récepteur  $Rx_1$ .

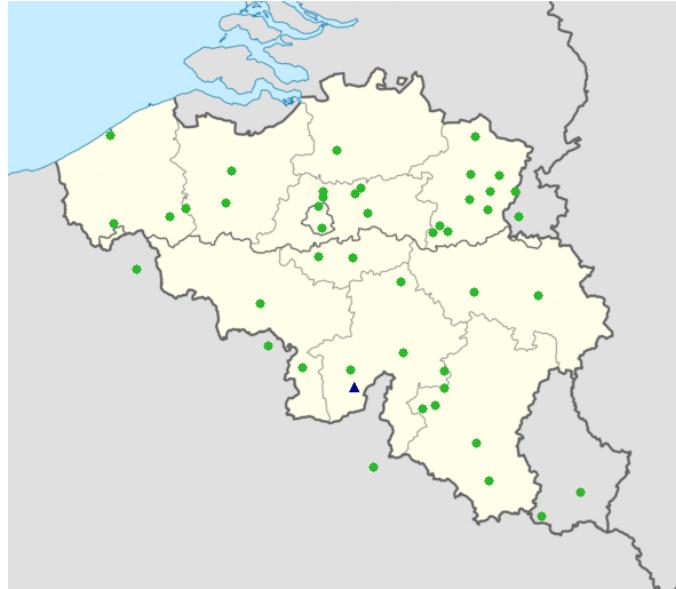


FIGURE 3 – Carte montrant l'emplacement de l'émetteur et récepteurs.

### 3 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) a pour but de détecter et d’étudier les météores en utilisant la détection des météores par ondes radios. Il dispose pour ceci d’un réseau d’un émetteur et de quarante-deux récepteurs situés dans la Belgique et dans les pays avoisinants. Les emplacements de ces stations peuvent être visualisé sur la figure 3, où le triangle bleu représente l’émetteur et les boules vertes sont des réceptrices. Dans cette section, ce réseau et son fonctionnement seront expliqués.

#### 3.1 L’Émetteur

Le réseau BRAMS dispose d’un émetteur unique situé à Dourbes, dans le sud de la Belgique. Cet émetteur transmet de façon continue un signal à la fréquence 49.970 MHz et d’une puissance d’approximativement 120 W. Ce signal sera réfléchi sur d’éventuelles trainées de météores et pourra être détecté par des récepteurs.

### 3.2 Les Stations de Réception

Les stations réceptrices permettent donc de récupérer le signal de l'émetteur, réfléchi par les échos de météores. Durant la période d'existence du projet BRAMS, plusieurs stations réceptrices ont été développées. Actuellement, le réseau BRAMS est composé de deux types de stations différentes. Ces deux stations réceptrices et leurs différences seront expliquées ci-dessous.

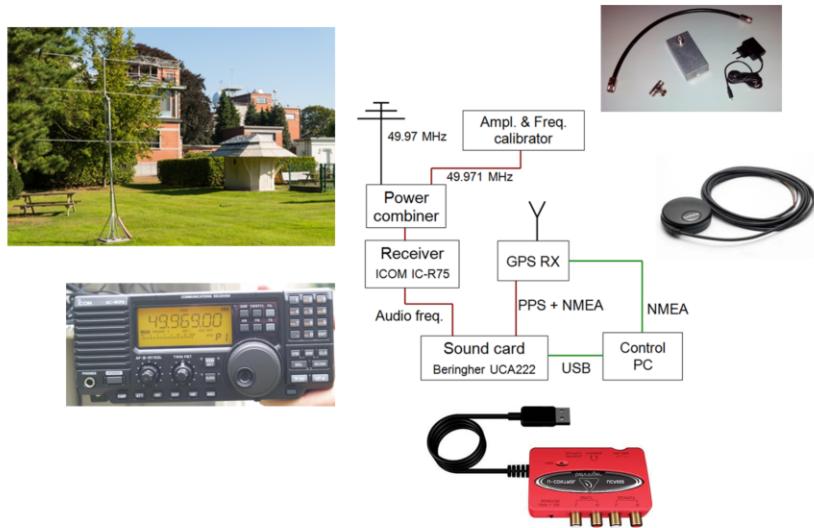


FIGURE 4 – Station v1 du réseau BRAMS

#### 3.2.1 Les Stations V1

Les premières stations utilisent des récepteurs analogiques ICOM-R75. Dans ce récepteur sont injectés le signal venant de l'antenne à 49,97 MHz et le signal calibreur généré par un appareil présent dans la station réceptrice. Le signal calibreur est constant et dispose d'une fréquence de 49,975 MHz (500 Hz au-dessus du signal de l'antenne). Elle est additionnée au signal de l'antenne à l'aide d'un simple T. Ce signal calibreur a un but bien précis : permettre la conversion des valeurs reçues par le récepteur en unités Watts. Comme on connaît sa puissance, elle sert de référence et nous permet de calculer la puissance des autres signaux captés par l'antenne.

Le récepteur décale le signal reçu par l'antenne et le calibreur à 1 kHz à l'aide de l'oscillateur local réglé à 49,969 MHz. Le signal passe ensuite par une carte son externe, où il est échantillonnée à un taux de 22050 Hz. Cette carte son est également connectée à une horloge GPS<sup>1</sup> Garmin. L'horloge ajoute un signal de 1 PPS<sup>2</sup> suivi de données NMEA<sup>3</sup> au signal reçu par l'antenne. Les données NMEA contiennent, entre autres, les informations à propos de la seconde associée au dernier PPS (horodatage). Ceci permet d'avoir un temps très précis pour chaque échantillon du signal.

La carte son est ensuite connectée à un ordinateur faisant tourner le logiciel 'Spectrum Lab'. Ce logiciel pilote la station et gère la procédure permettant de détecter des échos de météores. Afin de synchroniser l'horloge de l'ordinateur, celui-ci est également connecté à l'horloge GPS. Les données récupérées sont enregistrées toutes les cinq minutes dans des fichiers de format WAV<sup>4</sup>. La figure 4 montre un schéma de la station v1.

Ces stations ont parfaitement fonctionné pendant longtemps. Cependant, un problème avec le récepteur ICOM s'est manifesté de façon récurrente à partir de 2017. Remplacer ces récepteurs s'est avéré compliqué comme le récepteur ICOM n'était plus produit. D'autres récepteurs analogiques pouvaient servir comme remplaçant, mais ceux-ci étaient bien plus chers. De plus, la station v1 était confrontée à plusieurs autres problèmes comme une faible portabilité ou encore une installation complexe. Ceci a poussé les membres du projet BRAMS au développement d'une nouvelle solution permettant de remplacer les stations v1.

### 3.2.2 Les Stations V2

En 2017 un étudiant venant de l'Ephec, nommé Antoine Calegaro, a créé un prototype d'une nouvelle station. À partir de ce prototype Michel Anciaux, membre du projet BRAMS, a développé la station v2, destiné à remplacer les stations v1

La station v2 est piloté par un Raspberry PI dont l'horloge est synchronisé avec le signal PPS du même GPS Garmin utilisé sur la station v1. Son récepteur est un RSP2 qui à l'avantage d'avoir une portée dynamique plus

- 
1. Global Positioning System
  2. Pulse Per Second
  3. National Marine Electronics Association
  4. Waveform Audio File

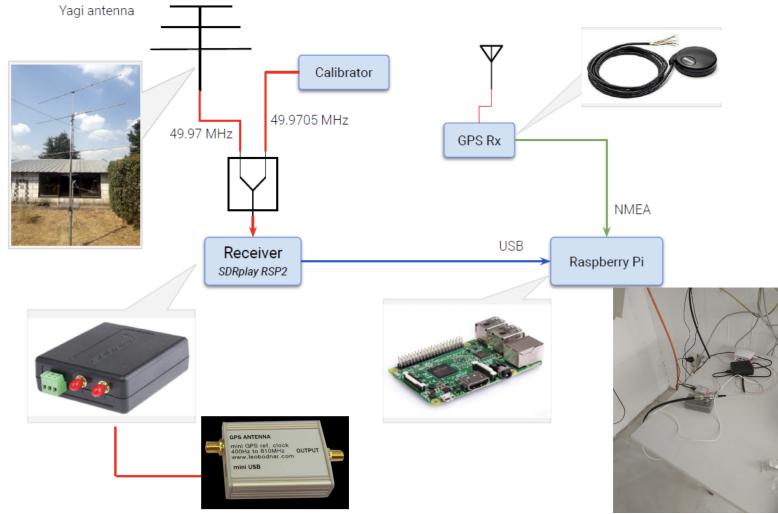


FIGURE 5 – Station v2 du réseau BRAMS.

grande et une sensibilité supérieure par rapport au récepteur ICOM. Il est également moins cher que son prédecesseur. Cependant, il ne permet pas l'échantillonnage du signal de l'antenne ensemble avec le signal GPS de 1 PPS. La fréquence du récepteur est stabilisé à l'aide d'une fréquence de référence externe à 24 MHz donné par un GPSDO<sup>5</sup>.

L'utilisation de deux horloges GPS (un pour le timing sur le Raspberry et un pour la fréquence sur le récepteur) permet d'avoir la fréquence de l'émetteur exactement à 1 kHz et le signal calibreur exactement 500 Hz au-dessus. Ceci était un problème sur les anciens récepteurs qui ne permettaient pas de stabiliser la fréquence à l'aide d'une source extérieure, causant des dérives de fréquence selon la température de celui-ci. La figure 5 montre un schéma des composants de la nouvelle station réceptrice. En 2020, la station fut placée dans une boîte métallique et la fréquence de référence du GPSDO fut également utilisé pour stabiliser l'oscillateur local du calibreur.

Cette station est donc beaucoup plus simple à installer puisqu'elle ne nécessite plus d'ordinateur externe pour son fonctionnement. En plus d'être plus compact et facile à installer que la station v1, la station v2 améliore la qualité des données et est moins cher.

5. GPS Disciplined Oscillator

### 3.3 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 pour les stations v1 ou 6048 Hz pour les stations v2. Cette fréquence permet d'enregistrer des données dans une bande de fréquences allant jusqu'à respectivement 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 qui est décalé 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 signaux capturés par la station. Dans ce bloc, on retrouve quelques informations relatives à la station de réception, la station émettrice, le signal GPS ou encore, la fréquence d'échantillonnage.

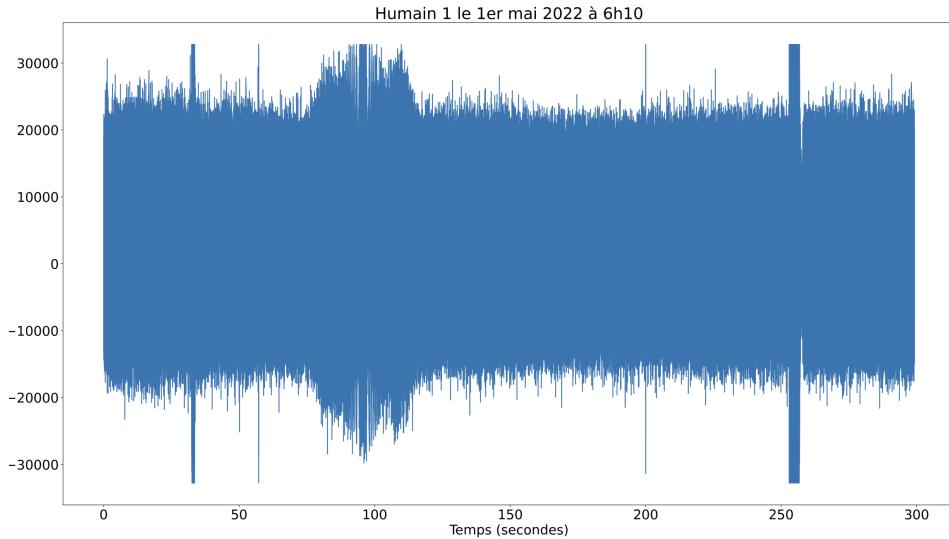


FIGURE 6 – Données brutes venant d'un fichier audio d'une station réceptrice.

Ces fichiers audio, comme montré à la figure 6, 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 dans un fichier audio. Au lieu de représenter l'intensité sur l'ordonnée et le temps sur l'abscisse, un spectrogramme représente la répartition des fréquences au cours du temps. Il contient donc trois dimensions :

- Le temps sur l'abscisse.
- La fréquence sur l'ordonnée.
- L'intensité représentée par un code couleur.

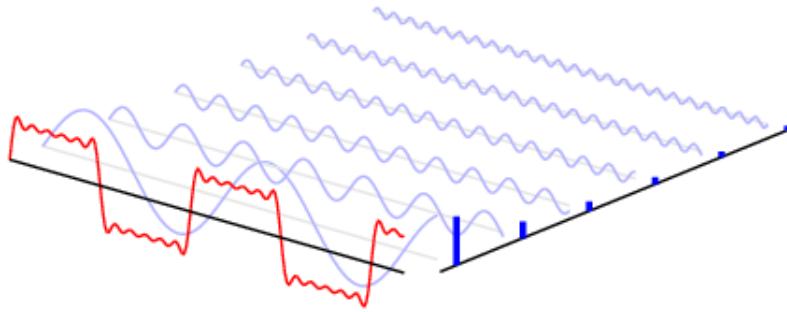


FIGURE 7 – Représentation visuelle de la Transformée de Fourier.

Pour générer un spectrogramme, il faut calculer un ensemble de Transformées de Fourier. La Transformée de Fourier permet de représenter la répartition de puissance entre les fréquences contenues dans un signal ou une partie d'un signal temporel. Elle produit donc ce qu'on appelle le spectre du signal. Elle se calcule sur un nombre quelconque d'échantillons qui se suivent dans un signal temporel. Ce spectre est souvent calculé à l'aide de la FFT<sup>6</sup> qui est plus rapide, mais qui exige un nombre d'échantillons  $2^n$ .

Sur la figure 7, on peut observer en rouge un signal temporel et le spectre de ce même signal en bleu. Les signaux en mauve clair sont les signaux à une fréquence, dont le signal rouge est composé.

---

6. Fast Fourier transform

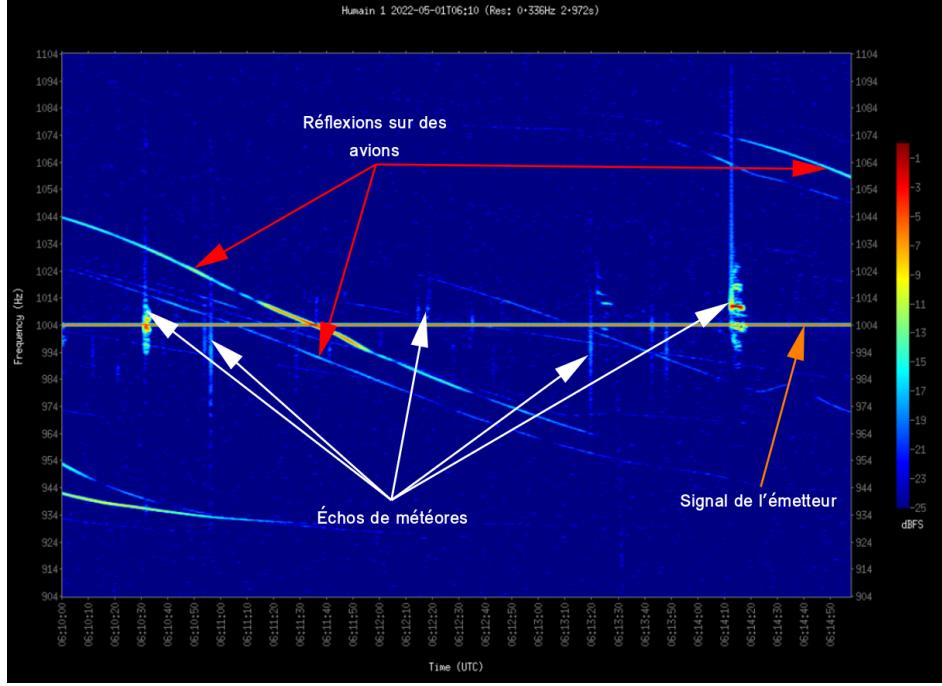


FIGURE 8 – Exemple d'un spectrogramme du projet BRAMS de 900 Hz à 1100 Hz. Les données représentées sont les mêmes qu'à la figure 6

Dans le cas des spectrogrammes pour le projet BRAMS, les FFT 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 v1 et 111 pour les stations v2. Ce nombre est obtenu avec la formule ci-dessous, où  $F_s$  est la fréquence d'échantillonnage,  $T$  la durée en secondes du signal et  $nfft$  le nombre d'échantillons utilisés pour générer un spectre.

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

Le spectrogramme généré aura une résolution fréquentielle, donnée par la formule  $F_s/nfft$ , de 0.34 Hz pour les stations v1 et 0.37 Hz pour les stations v2. Sa résolution temporelle, donnée par la formule  $nfft/F_s$ , est de 2.97 secondes pour les stations v1 et de 2.7 secondes pour les stations v2. Un exemple de spectrogramme venant d'une station de réception est affiché aux figures 8 et 9. Sur la figure 8, on voit la partie du spectrogramme entre les

fréquences de 900 Hz et 1100 Hz. Dans cette partie sont contenues les échos de météores, le signal direct de l'émetteur à environ 1000 Hz et des parasites qui sont typiquement des réflexions sur les avions. On y retrouve donc tout le signal utile. Sur la figure 9 on voit l'entièreté (0 Hz - 3000 Hz) du même spectrogramme affiché à la figure 8. Le contenu de cette dernière se trouve alors entre les deux lignes rouges. On retrouve, sur la figure 9 le signal du calibreur à environ 1500 Hz. En-dehors des lignes orange, on peut observer que le signal est filtré. Ce filtrage est automatique et est appliquée sur tous les fichiers WAV du projet BRAMS.

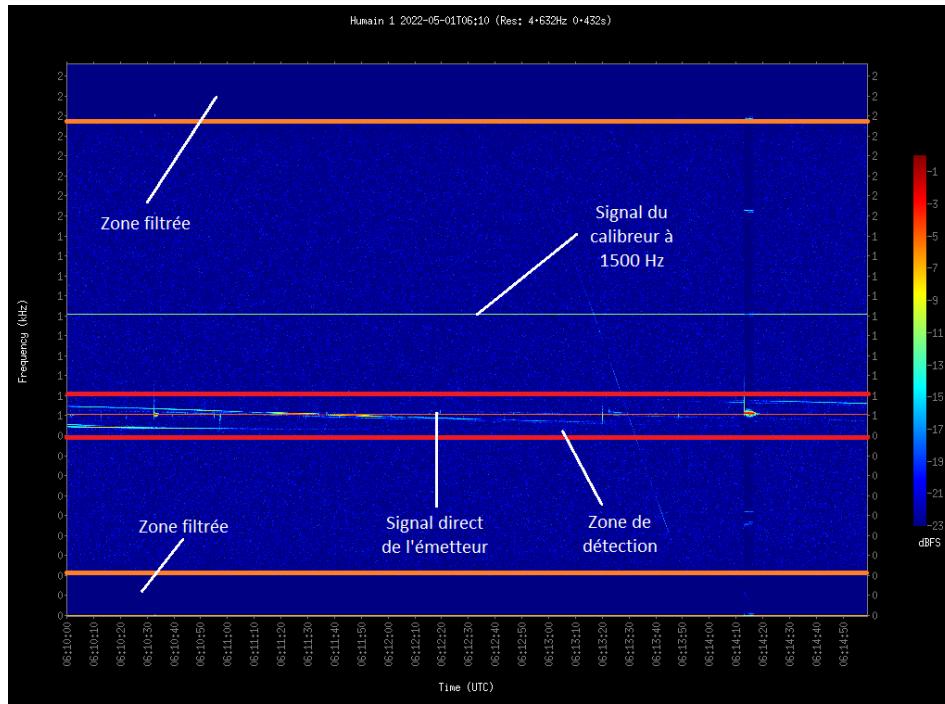


FIGURE 9 – Exemple d'un spectrogramme du projet BRAMS de 0 Hz à 3000 Hz. Les données représentées sont les mêmes qu'à la figure 6

### 3.4 L'Archive BRAMS

Les stations réceptrices produisent donc, en théorie, un fichier WAV toutes les cinq minutes. Ces fichiers ne sont bien évidemment pas stockés sur les stations mêmes, mais sont envoyés à intervalle régulier aux serveurs

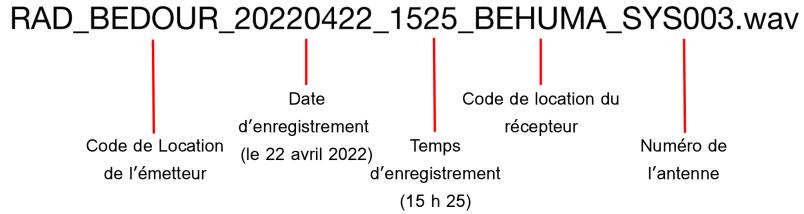


FIGURE 10 – Exemple de nom de fichier BRAMS.

dédiés du projet BRAMS. Les fichiers qui arrivent aux serveurs sont archivés une fois par jour. Cette procédure d’archivage est activée manuellement en lançant un script bash.

Quand les fichiers venant des stations de réception sont archivés, ils sont placés dans une structure de répertoires spécifique. Le chemin pour accéder à un fichier WAV spécifique et son nom dépendent de la station d'où vient le fichier ainsi que la date de début d'enregistrement du fichier. C'est grâce à ces noms de fichiers, dont vous trouverez un exemple à la figure 10, et cette structure fixe qu'on peut retrouver facilement chaque fichier au sein de l'archive.

## 4 Problématique liée à l'étude des données BRAMS

Le but du projet BRAMS est donc d'étudier les fichiers WAV venant des stations réceptrices afin de retrouver, entre autres, la trajectoire des météores. Pour identifier ces trajectoires, il est d'abord nécessaire de récupérer tous les fichiers contenant un écho de ce météore. C'est une action qui demande beaucoup de temps puisque l'utilisateur doit pour cela ouvrir chaque fichier de toutes les stations et vérifier s'il y a bien écho venant du météore.

De plus, actuellement il n'existe aucune façon de détecter une station produisant des données erronées. Ceci veut dire qu'une station pourrait produire des fichiers inutilisables pendant plusieurs mois sans que personne ne s'en aperçoit. Ces fichiers prendraient non-seulement de l'espace de stockage inutilement, mais dans le cas où une étude nécessiterait les données de cette station, elle seraient inutilisables.

C'est pour ces deux raisons que les membres du projet BRAMS souhaitent faciliter et automatiser certaines étapes dans l'étude des données BRAMS. Ils ont donc demandé à réaliser une ou plusieurs solutions permettant les actions suivantes :

- La détection la plus précise des échos d'un météore dans les fichiers de stations réceptrices différentes.  
Le résultat du programme doit être un fichier de format CSV<sup>7</sup> qui indique chaque écho dont le programme pense qu'il vient du météore recherché. Pour chaque écho retenu il faut bien entendu également ajouter des informations telles que la station où il a été détecté ou encore le temps de détection.
- Le monitoring constant des données des différentes stations à l'aide d'une interface sur le site web du projet BRAMS. Ce monitoring doit permettre aux scientifiques du projet de détecter facilement une anomalie dans les données.
- L'avertissement par mail si une anomalie est détectée dans les données BRAMS. Cette action implique également la détection automatique d'anomalies dans les données venant des stations réceptrices.

---

7. Comma Separated Values

## 5 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 étape consistait à assimiler correctement de quoi le programme doit être capable. Pour cela, plusieurs réunions ont eu lieu 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 expliquant 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 régulièrement pu demander validation quant à la direction que je prenais pour mon travail. Je présentais fréquemment mes réalisations aux scientifiques du projet BRAMS afin d'avoir un feedback constant 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 à l'avance ce dont la fonctionnalité devait être capable. 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 même tâche. 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 à pouvoir travailler en moyenne deux jours par semaine. Une grande partie de ces heures se sont déroulés lorsque je rentrais de mon stage ou pendant le week-end.

À la fin de ce projet de fin d'études l'entièreté du code a été donné aux membres du projet BRAMS. Comme demandé, le code est commenté et accompagné d'un mode d'emploi.

## 6 Technologies utilisées

### 6.1 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 matplotlib ont été très utiles pour arriver à un résultat performant et fonctionnel. Ces librairies offrent beaucoup de flexibilité et offrent une performance élevée 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 suffisamment rapide. Ceci est dû, entre autres, à la nature open-source de ces librairies ainsi qu'à 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.

### 6.2 MariaDB

Afin de sauvegarder les données produites par le programme, une base de données est requise. Les données sont enregistrées dans la base de données existant du projet BRAMS. Le type de base de données du projet BRAMS est MariaDB.

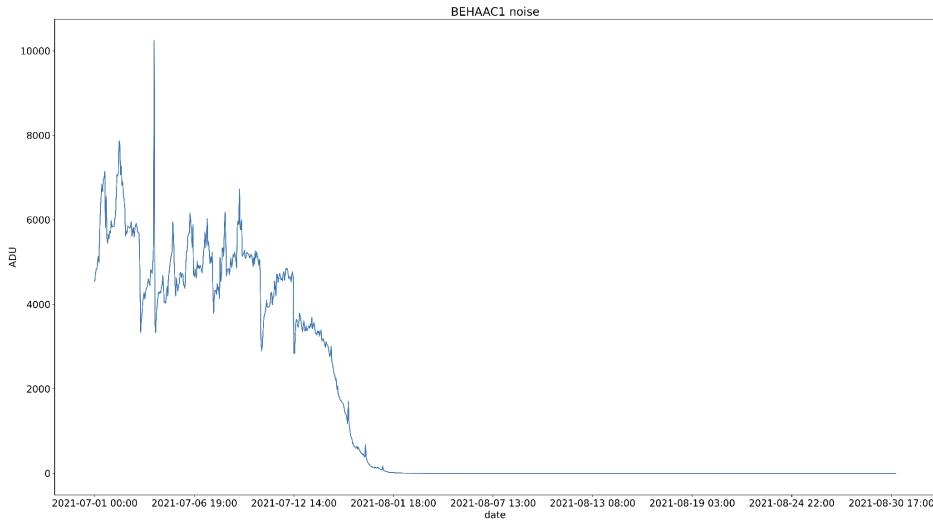


FIGURE 11 – Intensité du bruit de la station BEHAAC, antenne 1 entre le 1<sup>er</sup> juillet 2021 et le 31 août 2021.

## 7 Monitoring des Données BRAMS

Comme évoqué plus haut dans ce rapport, le monitoring des fichiers peut faciliter l'étude des données BRAMS. Le développement d'un bon logiciel de surveillance relève plusieurs défis. Il doit être cohérent, et ceci pour tous les types de stations. De plus, sachant que par jour une grande quantité de fichiers est produit par chaque station, l'optimisation du logiciel est également un facteur important afin de réduire l'utilisation des ressources système et le temps nécessaire pour obtenir les résultats du monitoring. Dans cette section, le développement du logiciel de surveillance ainsi que les démarches prises afin de résoudre ces défis seront expliqués.

### 7.1 Déetecter une anomalie sur une station

Afin de déterminer si un fichier contient une anomalie ou non, il est essentiel de bien choisir les éléments à surveiller. Dans les sections qui suivent, les différents éléments surveillés dans chaque fichier ainsi que leur pertinence seront expliqués.

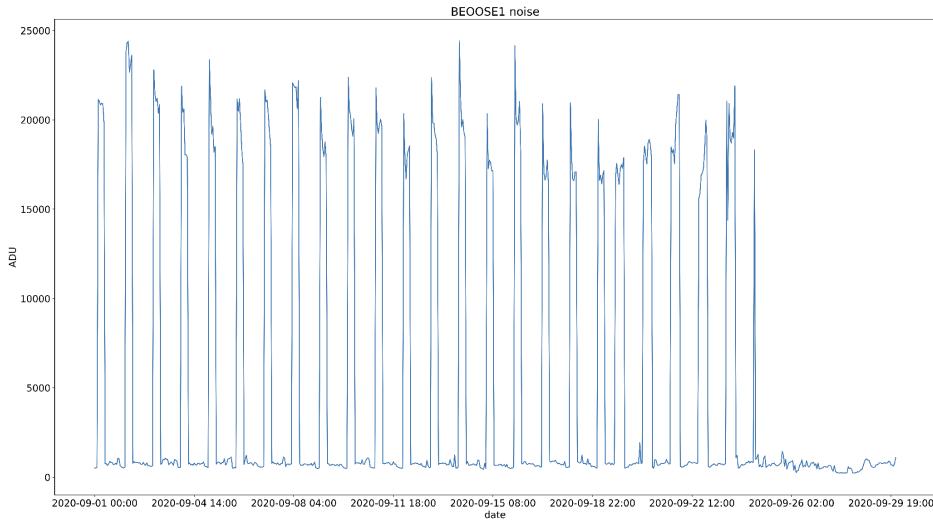


FIGURE 12 – Bruit de la station BEOOSE, antenne 1 entre le 1<sup>er</sup> septembre 2020 et le 30 septembre 2020.

### 7.1.1 Le Bruit

L'intensité du bruit est le premier élément qui sera surveillé. Bien que celui-ci puisse varier légèrement avec le temps, elle devrait rester constante sur une longue période et les grosses variations de bruit sont souvent signe que la station est défectueuse.

Ceci était le cas par exemple pour la station BEAAC lors du mois de juillet 2021. Comme on peut le voir sur la figure 11, on constate une diminution graduelle puis très rapide du bruit. Ce phénomène, pour les stations v1, est signe que le récepteur ICOM doit être remplacé. Il faut savoir qu'à partir du moment où le récepteur ne capte presque plus rien, jusqu'à la solution du problème, les données produites par la station sont inutilisables.

Un autre exemple de grosses variations de bruit est la station de BEOOSE entre le 1er septembre 2020 jusqu'au 30 septembre 2020. Quand on visualise la figure 12, on remarque de nombreux pics d'intensité du bruit. La cause de ce problème peut varier au cas par cas, il peut être un mauvais branchement au niveau de la station, des câbles endommagés ou encore un parasite extérieur à la station réceptrice. Il est important de détecter et résoudre ce

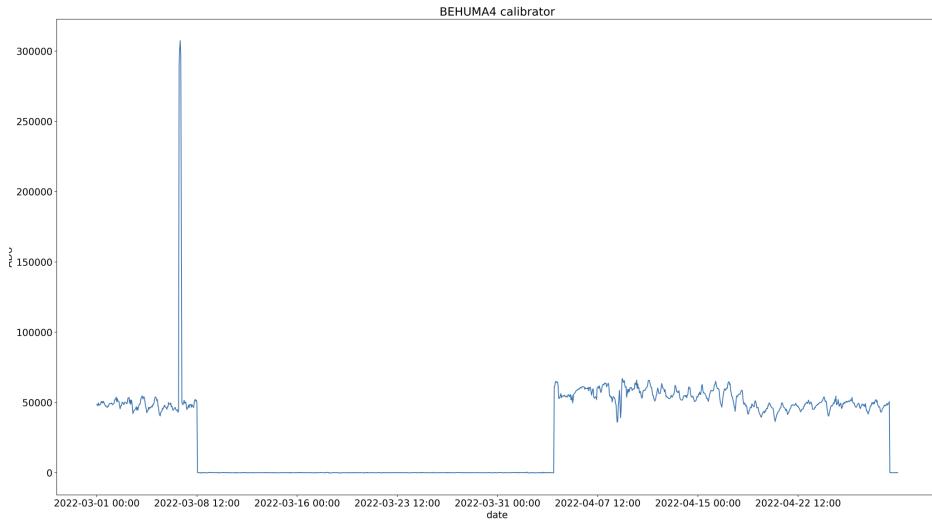


FIGURE 13 – Signal calibreur de la station BEHUMA, antenne 4 entre le 1<sup>er</sup> mars 2022 et le 30 avril 2022.

problème rapidement puisque le bruit est tellement haut qu'il est impossible de détecter autre chose dans le fichier.

Une autre raison de surveiller le bruit est qu'il doit être présent dans chaque fichier. Un fichier avec un bruit nul est impossible et veut dire qu'au moment où la station a généré le fichier, il avait un problème. L'intensité du bruit est donc non-seulement une valeur facile à surveiller, mais elle permet également de détecter toute une série de problèmes qui sont de nature différente.

#### 7.1.2 Le Signal Calibreur

Le second élément que le programme surveille est l'intensité du signal calibreur ainsi que la fréquence à laquelle celle-ci se trouve. Mesurer l'intensité de ce signal permet, comme la mesure du bruit, de détecter si un récepteur de la station v1 est cassé. De plus, comme le bruit, le signal calibreur est un élément qui est censé être présent dans chaque fichier produit par une station réceptrice, s'il est absent cela signifie que la station a un problème ce qui nécessite donc d'être détecté.

Vient ensuite la surveillance de la fréquence à laquelle se trouve le signal calibreur. En théorie, il devrait toujours se trouver à 1500 Hz; en pratique

cette valeur peut varier en particulier avec les anciennes stations où il peut y avoir une translation de fréquence en fonction de la chaleur. Une variation de cette fréquence entre les valeurs de 1350 Hz et 1750 Hz ne pose pas un grand problème et ne doit donc pas être détectée. Par contre, une trop grosse variation indique que la station a un défaut, et selon l'anomalie peut rendre les données inutiles. C'était notamment le cas pour la station BEHUMA4, lors d'une période qui a commencé le 8 mars 2022 et qui s'est étendu jusqu'au 3 avril 2022. Pendant ce temps, toutes les fréquences étaient décalées d'environ 1000 Hz vers le bas dans chaque fichier généré. Ceci peut être visualisé à la figure 13. Bien que ça ne pose pas vraiment de soucis pour le signal calibreur même, qui se trouve alors à environ 500 Hz, toutes les données utiles qu'on retrouve typiquement autour de 1000 Hz se situent maintenant autour de 0 Hz. Sachant qu'en plus les fréquences basses sont filtrés dans les fichiers BRAMS, l'information utile est donc absente.

Le signal calibreur est donc également un élément, se trouvant dans chaque fichier, qui peut indiquer rapidement s'il y a un problème avec une station. Ensemble avec le bruit, ils forment une bonne base pour déterminer si les données générées sont utilisables et si une station a besoin d'une intervention où non.

## 7.2 Fonctionnement du Logiciel de Surveillance

Ensemble avec les scientifiques du projet BRAMS, nous avons décidé que le programme doit se lancer automatiquement après chaque archivage de données. Le programme se lance en ligne de commande et ne nécessite aucun paramètre pour fonctionner. Une fois que le programme est démarré, il va chercher un par un les fichiers WAV nécessaires dans l'archive BRAMS. Dans le but d'avoir une surveillance suffisante sans utiliser trop de ressources machines, nous avons décidés d'analyser un fichier par heure par station. Ceci revient à l'analyse de 24 fichiers par jour par station.

### 7.2.1 Analyse des Fichiers WAV

Pour chercher et lire un fichier WAV provenant d'une station réceptrice, le logiciel utilise une classe nommée BramsWavFile. La base de cette classe a été écrite par Michel Anciaux. Cependant, ce code a subi deux grandes modifications afin de l'adapter pour le programme de monitoring et pour le rendre plus modulable. Le premier grand changement implique la recherche des fichiers dans l'archive : tandis que le code non modifié nécessitait un

```

1 def get_psd(f, flow=800, fhigh=900):
2     # get fourier tranform from BramsWavFile class
3     freq, S, fbin = f.FFT(f.Isamples)
4     idx = (freq >= flow) * (freq < fhigh)
5
6     # calculate the total power of the wanted frequencies
7     p = (S[idx] * S[idx].conj()).real / 2
8
9     # get a mean normalized to 1Hz
10    psd = p.mean() / fbin
11
12    return psd
13

```

FIGURE 14 – Code de la fonction permettant de calculer la dsp.

chemin pour trouver le fichier, maintenant il est capable de chercher tout seul un fichier dans l'archive BRAMS ou dans un autre dossier sur base de quelques informations (date et heure du fichier, station, numéro d'antenne). Le second changement concerne l'optimisation et donc le temps nécessaire pour lire un fichier WAV. Toutes les étapes non nécessaires ont été effacées et plusieurs librairies ont été testées afin de trouver celle qui était la plus rapide. Pour la méthode FFT de la classe BramsWavFile, qui permet de calculer la transformée de Fourier d'un fichier WAV, la librairie Scipy offrait par exemple des performances supérieures à la librairie Numpy tout en offrant les mêmes fonctionnalités et la même portabilité.

Une fois les valeurs du fichier WAV obtenues, on calcule d'abord la densité spectrale de puissance (dsp) du bruit. La dsp représente la répartition des puissances en fonction des différentes fréquences contenues dans un signal. L'unité de la dsp est exprimé en puissance par Hertz.

Afin d'avoir une estimation précise du bruit sans devoir calculer la dsp de tout le bruit dans un fichier, on prend uniquement la dsp entre 800 Hz et 900 Hz. Entre ces deux fréquences on ne trouve normalement pas d'éléments autres que le bruit. Dans un cas contraire, c'est souvent une indication qu'il y a un problème avec une station. Le code permettant de calculer la dsp du bruit est affiché à la figure 14. Dans un premier temps, on calcule la FFT du fichier WAV. Ce calcul nous donne trois variables : une liste avec les valeurs de l'axe x (freq), un vecteur avec les valeurs de l'axe y (S) en fonction de l'axe x et la résolution fréquentielle de l'axe x (fbin). Ensuite, on calcule le

spectre des puissances en multipliant le vecteur  $S$  par son conjugué, ce qui nous donne son carré et donc la puissance pour chaque fréquence de l'axe  $x$ . Enfin, on fait la moyenne de ce vecteur et on divise cette moyenne par la résolution fréquentielle ce qui nous donne une moyenne par Hz du bruit et donc la dsp.

Après, le programme passe à la recherche de la fréquence du signal calibreur. Il fait ceci en recherchant l'intensité maximale entre les fréquences de 1350 Hz et 1750 Hz dans un fichier. Comme le signal calibreur s'étend sur l'entièreté de la durée du fichier, il doit toujours avoir la valeur maximale entre ces deux fréquences. La fréquence retrouvée nous permet alors de calculer la dsp du signal. Il y a ici deux différences par rapport au calcul de la dsp du bruit. La première concerne la bande de fréquence utilisée pour le calcul : pour le signal calibreur, elle est calculée sur une bande de 10 Hz autour de la fréquence trouvée pour cette dernière. La deuxième différence implique la soustraction de la dsp du bruit une fois la dsp du signal calibreur obtenue. Cette étape est faite afin de disposer d'une estimation plus précise.

Les étapes listées ci-dessus permettent d'avoir les informations nécessaires afin de pouvoir déterminer s'il y a un problème avec un fichier, et donc par conséquence, avec une station. Les deux valeurs de dsp sont enregistrées dans la base de données du projet BRAMS et sont donc réutilisables.

### 7.2.2 L'avertissement d'une anomalie

Avant l'enregistrement, une autre étape est complétée : la détection automatique d'une anomalie. Pour cette étape, le logiciel applique les mêmes actions sur le bruit et le signal calibreur avec une exception, qui sera expliquée plus bas dans cette section. Ces actions sont répétées à chaque fois qu'une nouvelle valeur de dsp est générée.

Afin de détecter une grande variation dans les données, le programme se base sur l'ensemble des valeurs de dsp des deux semaines précédent la nouvelle valeur de dsp. À partir de cet ensemble, il faut définir une frontière haute et une frontière basse. Si la nouvelle dsp se trouve en dehors d'une frontière, il sera considéré comme anormal et les scientifiques du projet BRAMS seront avertis.

Différentes méthodes existent pour définir ces frontières. La méthode utilisée dans ce logiciel, utilisant la gamme interquartile, peut être visualisée à

```

1 def detect_variations(y_data, current_value):
2     q1 = np.percentile(y_data, 25, interpolation='lower')
3     q3 = np.percentile(y_data, 75, interpolation='higher')
4
5     interquartile = q3 - q1
6
7     upper_limit = q3 + (2.4 * interquartile)
8     lower_limit = q1 - (2.4 * interquartile)
9
10    if current_value >= upper_limit:
11        return 1
12    elif current_value <= lower_limit:
13        return -1
14    else:
15        return 0
16

```

FIGURE 15 – Code de la fonction permettant de détecter des variations excessives dans les données.

la figure 15 et est expliquée ci-dessous.

Il faut savoir que les deux paramètres de la fonction affichée à la figure 15, **y\_data** et **current\_value**, sont l'ensemble des valeurs de dsp des deux semaines précédent la nouvelle valeur de dsp et la nouvelle valeur de dsp même. Le programme commence par calculer le 25e et le 75e percentile de l'ensemble des données qui se trouvent dans **y\_data**. Un percentile est une mesure indiquant la valeur en dessous de laquelle se trouvent un pourcentage donné d'observations, dans un ensemble d'observations. Par exemple, le 20e percentile est la valeur en dessous de laquelle se trouvent 20% des observations dans un ensemble d'observations. Ensuite, on calcule la gamme interquartile de **y\_data**. Cette valeur est calculée en faisant la différence entre la 25e et la 75e percentile, que le programme a déjà calculé.

Le logiciel définira à partir de ces trois valeurs calculées, la frontière haute et basse. Les formules utilisées sont  $Q3 + (x * IQR)$  pour la frontière haute et  $Q1 - (x * IQR)$  pour la frontière basse où  $Q1$  et  $Q3$  sont le 25e et le 75e percentile et  $IQR$  la gamme interquartile. Le facteur  $x$  permet de définir la largeur entre les deux frontières, au plus sa valeur est élevée, au plus les frontières sont écartées.

C'est ce facteur qui est différent pour le bruit et le signal calibreur. Tandis

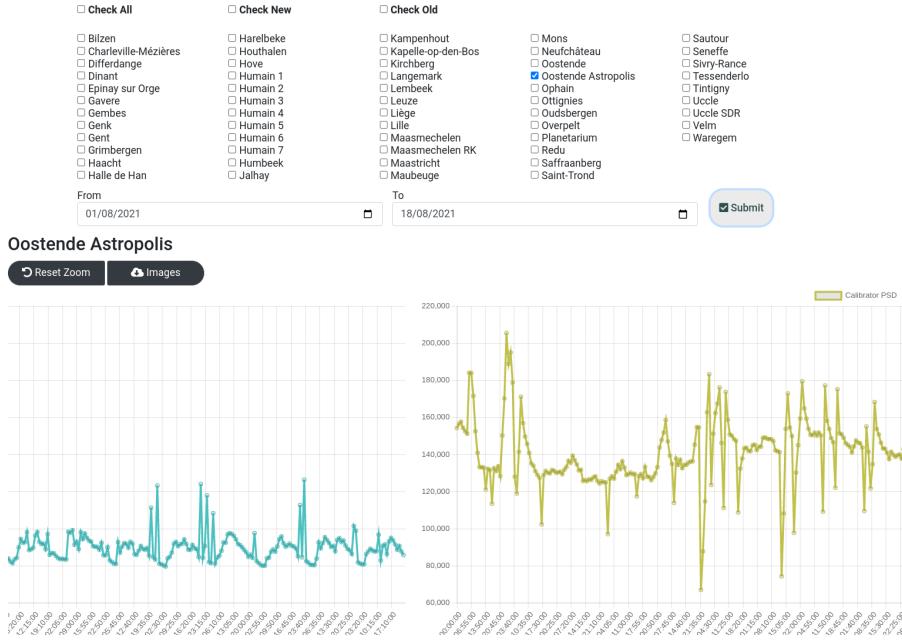


FIGURE 16 – Interface graphique du logiciel de surveillance.

que pour le bruit, il vaut 2.4, il est défini à 2.5 pour le signal du calibreur. La différence entre ces valeurs est que le bruit varie moins et il n'est donc pas nécessaire d'avoir les deux frontières aussi écartées que pour le signal du calibreur.

Finalement, on vérifie si la nouvelle valeur de `dsp (current_value)` dépasse ou non une des frontières. Si c'est le cas, on informe l'utilisateur d'un problème dans les données.

Bien que cette méthode permet bien de détecter des anomalies, elle n'est pas parfaite. En effet, parfois il y a des faux positifs et le programme alerte l'utilisateur alors qu'aucun problème est présent.

### 7.3 L'interface sur le site web BRAMS

En plus de la détection automatique d'une anomalie, une interface graphique permettant de visualiser l'évolution de la `dsp` du signal calibreur et du bruit a été développée. Il faut noter que cette interface est indépendante

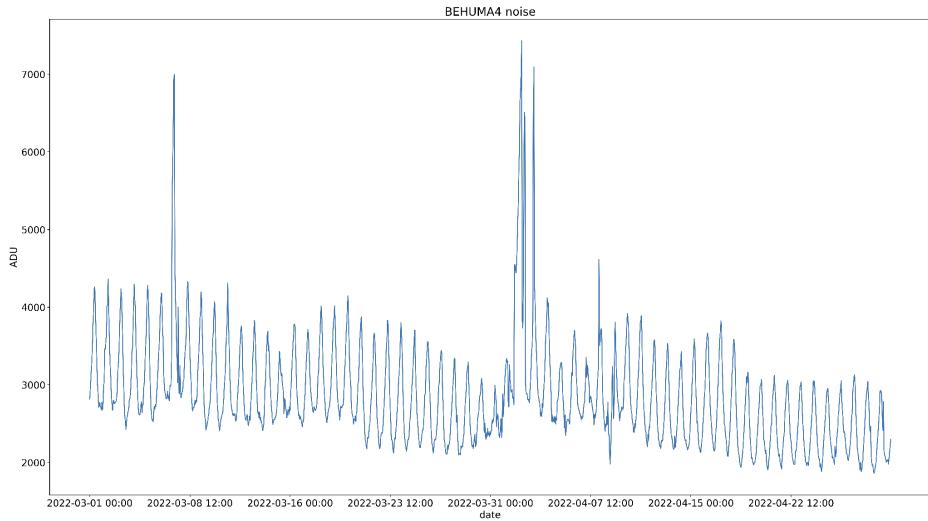


FIGURE 17 – Signal du bruit de la station BEHUMA, antenne 4 entre le 1<sup>er</sup> mars 2022 et le 30 avril 2022.

du programme de monitoring même, elle utilise, cependant, les données générées par ce-dernier. Comme on peut le voir sur la figure 16, l’interface affiche deux graphiques pour chaque station que l’utilisateur a sélectionnée. Le premier représente les valeurs de dsp pour le bruit, pour une période que l’utilisateur a indiquée et le deuxième affiche la même chose, mais cette fois-ci pour le signal calibreur.

Cette interface permet aux scientifiques du projet BRAMS de surveiller ces valeurs et éventuellement de détecter un problème qui ne serait pas détecté par la surveillance automatique. De plus, elle donne la possibilité de mieux comprendre les variations en observant des motifs qui reviennent. Un des motifs retrouvés lors des tests est une diminution de la dsp du bruit lorsque la nuit approche. Ceci est particulièrement visible sur la figure 17 qui représente la dsp du bruit pour la station BEHUMA4 du 1er mars 2022 au 30 avril 2022.

Cette interface graphique, comme l’entièreté du nouveau site web BRAMS utilise Joomla comme back-end. Joomla est un framework gratuit et open-source utilisant le langage PHP. Pour afficher les différents graphiques, le

front-end de l’interface utilise la librairie JavaScript open-source ChartJS<sup>8</sup>. Cette librairie permet d’afficher des beaux graphiques et de les agrandir si nécessaire. Cette interface sera disponible publiquement sur le nouveau site web BRAMS dès le déploiement de cette dernière. Plus d’images de cette interface sont disponibles à l’annexe A.

---

8. <https://www.chartjs.org/>

## 8 Logiciel de Détection des Météores

Une fois que les données sont archivées, les membres du projet BRAMS commencent à les analyser et les interpréter. Un objectif assez important de ce projet est le calcul de la trajectoire d'un météore dans l'atmosphère. Pour retrouver la trajectoire d'un météore à l'aide du réseau BRAMS, il faut qu'un météore soit détecté par au moins huit stations réceptrices. Vérifier manuellement, pour chaque station, si celle-ci a détecté un météore est une tâche prenant un temps non négligeable. Dans cette section, le développement d'un programme permettant de réduire la perte de temps causée par cette procédure est expliquée. Ce programme est entièrement indépendant du programme de monitoring, expliqué dans la section 7.

### 8.1 Déterminer si un Écho Appartient à un Météore

Avant de développer un logiciel, il est nécessaire de savoir comment trouver, sur différentes stations réceptrices, les échos venant d'un seul météore. Différents facteurs peuvent indiquer qu'un écho vient d'un météore spécifique ou non. Parmi les facteurs connus on trouve l'emplacement de la station qui a détecté l'écho de météore et le moment où l'écho est détecté. Dans le cas de ce logiciel, le facteur temps sera utilisé.

Quand on compare plusieurs échos de météores, tous détectés par des stations différentes, le facteur temps permet de donner une idée s'ils proviennent du même météore ou non. En effet, par expérience on sait que lorsqu'un météore est détecté par plusieurs stations, toutes ces détections se font dans un intervalle d'environ trois secondes. Bien que ce facteur ne permet pas d'assurer complètement qu'un écho vient d'un météore spécifique, il permet d'éliminer toutes les stations où aucun écho n'a été détecté pour un météore, réduisant grandement le temps nécessaire pour les scientifiques du projet BRAMS. De plus, si dans l'intervalle recherché on ne trouve qu'un écho, ceci augmente fortement les chances que celui-ci appartient au météore recherché.

### 8.2 Fonctionnement du Logiciel

Le programme développé se lance en ligne de commande et nécessite un paramètre obligatoire : un temps précis à la seconde, autour duquel on va rechercher d'autres échos. Un deuxième paramètre optionnel est le code de location d'une station où on est sûr d'avoir détecté le météore recherché.

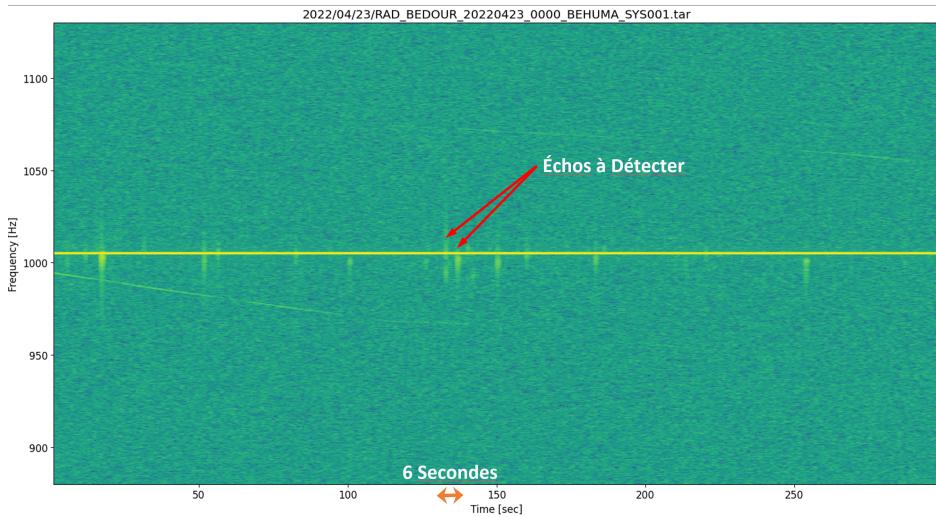


FIGURE 18 – Spectrogramme original de la station BEHUMA 1 le 23 avril 2022 à 00h00

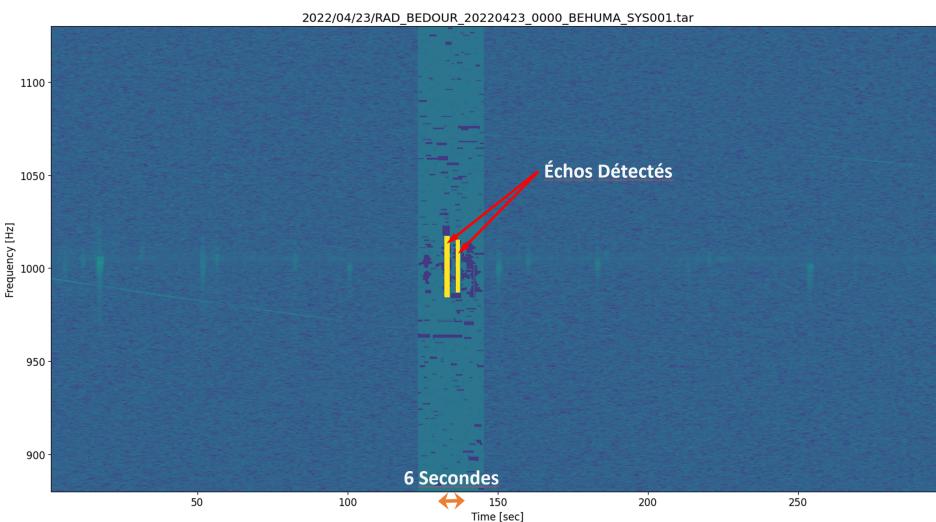


FIGURE 19 – Résultat du système de détection de météores.

Une fois que le programme est lancé, il calcule l'intervalle. Si le temps donné en entrée est nommé  $t_0$  et que  $\Delta t$  vaut trois secondes, l'intervalle calculé vaut  $[t_0 - \Delta t; t_0 + \Delta t]$ . Le programme va ensuite chercher tous les fichiers contenant cet intervalle. Afin de compléter cette action, il utilise la classe BramsWavFile qui est également utilisée pour le programme de monitoring.

Pour chaque fichier récupéré, il calcule le spectrogramme et détecte les météores présents dans l'intervalle calculé. La détection des météores, en ce moment, est fait par un code développé en parallèle avec le logiciel expliqué dans cette section. Cependant, ce code est temporaire et est voué à disparaître pour une solution plus efficace qui est en cours de développement par les membres du projet BRAMS et utilisant le machine learning. Actuellement pour détecter les météores, différentes étapes sont exécutées par le code sur le spectrogramme. Ces étapes sont listées ci-dessous et leur application sera montrée sur base du spectrogram non changé affiché à la figure 18.

```

1 def filter_by_percentile(
2     self,
3     start=0,
4     end=None,
5     percentile=95
6 ):
7     spectrogram_slice = self.__get_slice(start, end)
8
9     for column in spectrogram_slice.T:
10         column_percentile = np.percentile(column, percentile)
11         column[column < column_percentile] = 0
12

```

FIGURE 20 – Code permettant d'appliquer le filtre par percentile.

1. L'amplification des objets ressemblants à des météores par un filtre. Ce filtre a été développé spécialement pour ce but et peut être visualisé à l'annexe B. Le filtre est appliqué à l'aide d'une convolution. Une convolution est une opération mathématique qui prend deux fonctions en entrée et qui produit un nouveau signal. Dans ce cas-ci, les signaux en entrée sont le spectrogramme, ou une partie du spectrogramme, et le filtre. Un résultat de cette convolution sur un spectrogramme est représenté à l'annexe C.

2. La suppression du bruit en appliquant un filtre par percentile. Cette étape est effectuée par la fonction affichée à la figure 20. Ce code débute en prenant l'intervalle dans lequel on recherche des échos (ligne 7). Une fois cet intervalle récupéré, on calcule pour chaque colonne le 95e percentile. Toutes les valeurs de cette colonne inférieures à ce percentile sont ensuite mises à zéro (lignes 9 à 12). On peut observer l'effet de ce filtre à l'annexe D, où l'on voit bien que l'intervalle dans lequel on recherche des échos de météores est bien plus propre qu'à l'annexe C.
3. L'élimination de tous les objets qui sont trop petits pour pouvoir être l'écho d'un météore. Pour cette étape, une labellisation du spectrogramme est appliquée. Une labellisation, dans ce cas, est une action où l'on retrouve tous les objets séparés, non-nuls sur le spectrogramme. On vérifie alors leur taille et on les compare à la taille minimale qu'un écho de météore peut avoir. Si la taille de l'objet est inférieure à la taille minimale, elle est mise à zéro. Ceci peut être visualisé à l'annexe E où l'on peut observer, dans l'intervalle où on recherche des échos de météores, des tâches bleues. Ces tâches bleues sont des éléments qui se sont fait éliminer par cette étape.
4. La suppression des échos d'avions. En effet, les météores ne sont pas les seuls objets qui traversent l'atmosphère et qui sont détectés par les stations réceptrices. Les échos d'avions sont des parasites dans les fichiers BRAMS et le programme ne peut pas les confondre avec des échos de météores. C'est pour cette raison que le logiciel vérifie, pour chaque objet restant sur le spectrogramme à cette étape, si cet objet n'est pas trop large pour être un météore.

Après toutes ces actions, les objets restants sur le spectrogramme sont considérés comme étant des échos de météores et le programme récupère leurs coordonnées. Un exemple de résultat final de la détection des météores est affiché à la figure 19. Notez que cette image n'est pas la sortie du programme et qu'elle ne s'affiche pas lors du déroulement du programme.

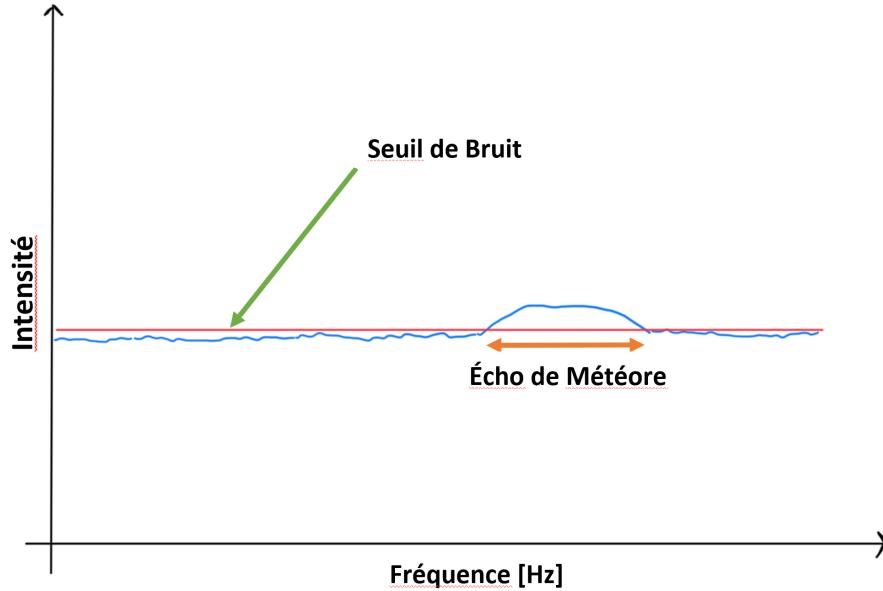


FIGURE 21 – FFT contenant un écho de météore.

Il ne reste plus qu'à rechercher la fréquence minimale et la fréquence maximale sur lequel s'étend l'écho. Ceci est fait en calculant la transformée de Fourier de l'intervalle pendant laquelle l'écho de météore dure. Une fois qu'on a la FFT, le logiciel détermine un seuil de bruit à la 85e percentile. Comme on peut le voir à la figure 21, ceci met en évidence l'écho de météore et permet de retrouver facilement sa fréquence minimale et maximale.

```

1 location_code,antenna_id,file_start,meteor_count,meteor_time,fmin,fmax,distance_km
2 BEHUMA,1,202204230000,2,2022-04-23T00:02:13:801995,1000.2845764160156,1015.7615661621094,0.0 km
3 BEHUMA,1,202204230000,2,2022-04-23T00:02:17:585397,987.835693359375,1010.7147216796875,0.0 km
4 BEHUMA,3,202204230000,2,2022-04-23T00:02:17:585397,987.4992370605469,1009.0324401855469,0.0 km
5 BEHUMA,3,202204230000,2,2022-04-23T00:02:13:801995,987.1627807617188,1012.7334594726562,0.0 km
6 BEHUMA,4,202204230000,2,2022-04-23T00:02:17:585397,988.5086059570312,1010.0418090820312,0.0 km
7 BEHUMA,4,202204230000,2,2022-04-23T00:02:13:801995,989.8544311523438,1012.7334594726562,0.0 km
8 BEHUMA,5,202204230000,2,2022-04-23T00:02:13:801995,987.835693359375,1012.7334594726562,0.0 km
9 BEHUMA,5,202204230000,2,2022-04-23T00:02:17:241451,988.1721496582031,1011.7240905761719,0.0 km
10 BEHUMA,6,202204230000,2,2022-04-23T00:02:13:801995,990.52734375,1002.6397705078125,0.0 km
11 BEHUMA,6,202204230000,2,2022-04-23T00:02:17:585397,990.52734375,1009.368896484375,0.0 km

```

FIGURE 22 – Exemple de fichier CSV généré par le programme de détection de météores.

Enfin, le programme crée un fichier CSV qui contiendra une ligne par écho de météore détecté. Chaque ligne contiendra :

1. Le code de location de la station où est détecté l'écho.
2. Le numéro de l'antenne qui a détecté l'écho.
3. La date et le temps de début d'enregistrement du fichier d'où vient l'écho.
4. Le nombres d'écho comptés dans le même fichier que celui d'où vient l'écho.
5. Le temps, précis à la microseconde, de détection de l'écho.
6. La fréquence minimum de l'écho.
7. La fréquence maximum de l'écho.
8. La distance entre la station où est détecté l'écho et la station reçue en entrée au lancement du programme.

Un exemple de fichier CSV généré par le programme peut être visualisé à la figure 22.

## 9 Conclusion

Le but principal de ce projet de fin d'études était de faciliter et automatiser une partie du travail des scientifiques du projet BRAMS. Premièrement il devait permettre aux scientifiques de détecter rapidement une anomalie dans les données produites par les stations réceptrices. Si un défaut n'est pas détecté rapidement, la station défectueuse continue à produire des données inutilisables. Ce problème a été résolu grâce au logiciel de monitoring qui permet non-seulement de détecter une anomalie dans les données, mais également de visualiser la qualité des données grâce à son interface intégrée au site web BRAMS. En effet, même si le logiciel alerte parfois l'utilisateur d'un problème non-existant et que des améliorations sont encore possibles, ces cas sont rares et le programme reste utilisable.

Deuxièmement, le travail devait faciliter la recherche de tous les échos de météores pouvant correspondre à un météore spécifique. Le programme de détection de météores permet aux scientifiques de grandement réduire le temps nécessaire pour chercher ces échos en retrouvant automatiquement tous les échos dans l'intervalle de temps nécessaire.

Ce travail m'a également permis d'étendre mes connaissances dans le domaine du traitement du signal en mettant en pratique plusieurs notions théoriques vues durant les cours. Le mélange de ces connaissances avec des nouvelles notions, que je n'avais pas encore appris, afin d'arriver à un résultat final a été un vrai défi pour moi.

Enfin, ce projet m'a permis de contribuer à un projet scientifique de grande ampleur et m'a appris comment un tel projet fonctionne.

## 10 Perspectives

Suite à une réflexion sur les programmes réalisés, j'ai listé ci-dessous quelques pistes d'améliorations intéressantes pour le futur :

- Bien que cela ait déjà été dit, le remplacement du système de détection de météores, dans le programme qui retrouve les échos d'un météore, par le système plus avancé en développement par le projet BRAMS représenterait une réelle amélioration.
- Pour le programme qui retrouve les échos d'un météore, on pourrait, à terme, ajouter le facteur d'emplacement de la station pour retrouver les échos appartenant à un météore. Cependant, cela demande beaucoup de recherche et d'étude et prendrait donc un temps considérable.
- On pourrait ajouter la possibilité de manipuler l'entièreté du programme de monitoring à partir de son interface en ligne, sur le site de BRAMS.
- Actuellement, le programme de monitoring surveille les fichiers de façon à détecter des erreurs connues. Cependant, il est possible que des problèmes non connus surviennent, qui ne sont alors pas détectés par le logiciel de surveillance. Une amélioration intéressante serait donc de réfléchir à des potentiels erreurs qui ne seraient pas détectés et en tenir compte dans le programme.
- Il serait intéressant d'avoir une interface graphique sur le site web de BRAMS permettant de manipuler le programme qui recherche les échos venant d'un météore. Cette interface pourrait alors afficher ces résultats sur un spectrogramme en plus de générer un fichier CSV.
- Pour le système de détection d'anomalies dans le programme de monitoring, il serait intéressant d'effectuer plus de tests et de recherches afin d'améliorer la méthode utilisée. En effet, comme expliqué avant, il n'est pas parfait et alerte parfois l'utilisateur d'un problème alors qu'il n'y en a pas.

## Références

- [1] *Site internet du projet BRAMS*, consulté en janvier 2022  
<https://brams.aeronomie.be/>
- [2] *Documentation de la fonction scipy.signal.spectrogram*, consulté en février 2022  
<https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.spectrogram.html>
- [3] *Hands-On Tutorial on Visualizing Spectrograms in Python by Yugesh Verma*, consulté en février 2022  
<https://analyticsindiamag.com/hands-on-tutorial-on-visualizing-spectrograms-in-python/>
- [4] *Cutting unused frequencies with specgram, matplotlib by uwii*, consulté en février 2022  
<https://stackoverflow.com/questions/19468923/cutting-of-unused-frequencies-in-specgram-matplotlib>
- [5] *Documentation Matplotlib*, consulté en février 2022  
[https://matplotlib.org/stable/api/mlab\\_api.html#matplotlib.mlab.specgram](https://matplotlib.org/stable/api/mlab_api.html#matplotlib.mlab.specgram)
- [6] *Set spectrogram Parameters by rayryeng*, consulté en février 2022  
<https://stackoverflow.com/questions/29321696/what-is-a-spectrogram-and-how-do-i-set-its-parameters>
- [7] *Documentation de la fonction numpy.convolve*, consulté en mars 2022  
<https://numpy.org/doc/stable/reference/generated/numpy.convolve.html>
- [8] *How to convolve two 2-dimensional matrices in python with scipy by Benjamin H.G. Marchant*, consulté en mars 2022  
<https://moonbooks.org/Articles/How-to-do-a-simple-2D-convolution-between-a-kernel-and-an-image-in-python-with-scipy-/>
- [9] *Slice 2D array in smaller 2D arrays by unutbu*, consulté en mars 2022  
<https://stackoverflow.com/questions/16856788/slice-2d-array-into-smaller-2d-arrays>
- [10] *Compute a confidence interval from sample data by shasan*, consulté en mars 2022  
<https://stackoverflow.com/questions/15033511/compute-a-confidence-interval-from-sample-data>

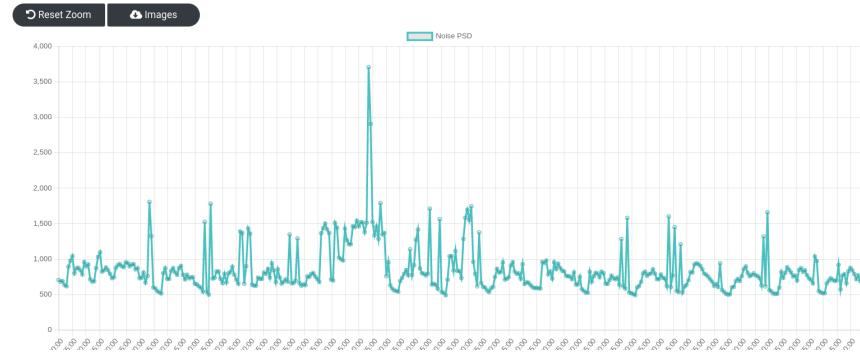
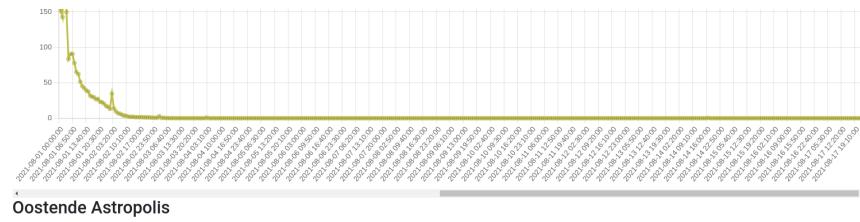
- [11] Microsoft WAVE soundfile format by craig@ccrma.stanford.edu, consulté en mars 2022  
<http://soundfile.sapp.org/doc/WaveFormat/>
- [12] Python - Sending Email using SMTP, consulté en mars 2022  
[https://www.tutorialspoint.com/python/python\\_sending\\_email.htm](https://www.tutorialspoint.com/python/python_sending_email.htm)
- [13] The fastest 2D convolution in the world by Laurent Perrinet, consulté en mars 2022  
<https://laurentperrinet.github.io/sciblog/posts/2017-09-20-the-fastest-2d-convolution-in-the-world.html>
- [14] Documentation de la fonction `scipy.ndimage.convolve`, consulté en mars 2022  
<https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.convolve.html>
- [15] Power Spectral Density - an overview by John Dempster, consulté en avril 2022  
<https://www.sciencedirect.com/topics/computer-science/power-spectral-density>
- [16] What is a Power Spectral Density by peter.schaldenbrand@siemens.com, consulté en avril 2022  
<https://community.sw.siemens.com/s/article/what-is-a-power-spectral-density-psd>
- [17] How to add time onto a datetime object in Python by Adam Smith, consulté en avril 2022  
<https://www.adamsmith.haus/python/answers/how-to-add-time-onto-a-datetime-object-in-python>
- [18] Fourier Transforms With `scipy.fft` : Python Signal Processing by Cameron MacLeod, consulté en avril 2022  
<https://realpython.com/python-scipy-fft/#why-would-you-need-the-fourier-transform>
- [19] Documentation de la fonction `scipy.fft.rfft`, consulté en avril 2022  
<https://docs.scipy.org/doc/scipy/reference/generated/scipy.fft.rfft.html>
- [20] Reading and Writing CSV Files in Python - Real Python by Jon Fincher, consulté en mai 2022  
<https://realpython.com/python-csv/>

- [21] *A Fast, Extensible Progress Bar for Python and CLI - TQDM*, consulté en mai 2022  
<https://github.com/tqdm/tqdm>
- [22] *Documentation du module tarfile*, consulté en mai 2022  
<https://docs.python.org/3/library/tarfile.html>
- [23] *Documentation de la librairie simplejson*, consulté en mai 2022  
<https://pypi.org/project/simplejson/>
- [24] *Python Command Line Arguments by Andre Burgaud*, consulté en mai 2022  
<https://realpython.com/python-command-line-arguments/>
- [25] *Documentation du module argparse*, consulté en mai 2022  
<https://docs.python.org/3/library/argparse.html>
- [26] *Wikipedia - Étoile filante*, consulté en juin 2022  
[https://fr.wikipedia.org/wiki/%C3%89toile\\_filante](https://fr.wikipedia.org/wiki/%C3%89toile_filante)
- [27] *Wikipedia - Meteoroid*, consulté en juin 2022  
<https://en.wikipedia.org/wiki/Meteoroid>
- [28] *Wikipedia - Pulse-per-second signal*, consulté en juin 2022  
[https://en.wikipedia.org/wiki/Pulse-per-second\\_signal](https://en.wikipedia.org/wiki/Pulse-per-second_signal)
- [29] *What Exactly Is GPS NMEA Data ? by Eric Gakstatter*, consulté en juillet 2022  
<https://www.gpsworld.com/what-exactly-is-gps-nmea-data/>
- [30] *Dynamically creating charts of each row in an HTML table with chart.js by Naga Sai A*, consulté en juillet 2022  
<https://stackoverflow.com/questions/49881981/dynamically-creating-charts-of-each-row-in-an-html-table-with-chart-js>
- [31] *Documentation de ChartJS*, consulté en juillet 2022  
<https://www.chartjs.org/chartjs-plugin-zoom/latest/guide/developers.html>
- [32] *Variability / Calculating Range, IQR, Variance, Standard Deviation by Pritha Bhandari*, consulté en août 2022  
<https://www.scribbr.com/statistics/variability/>
- [33] *MySQL - select interval of every 2 hours from timestamp column by Lukasz Szozda*, consulté en août 2022  
<https://stackoverflow.com/questions/34270918/mysql-select-interval-of-every-2-hours-from-timestamp-column>

- [34] *Percentile explications*, consulté en août 2022  
<https://pallipedia.org/percentile/>
- [35] *How to Find Outliers | 4 Ways with Examples & Explanation by Pritha Bhandari*, consulté en août 2022  
<https://www.scribbr.com/statistics/outliers/>

## 11 Annexes

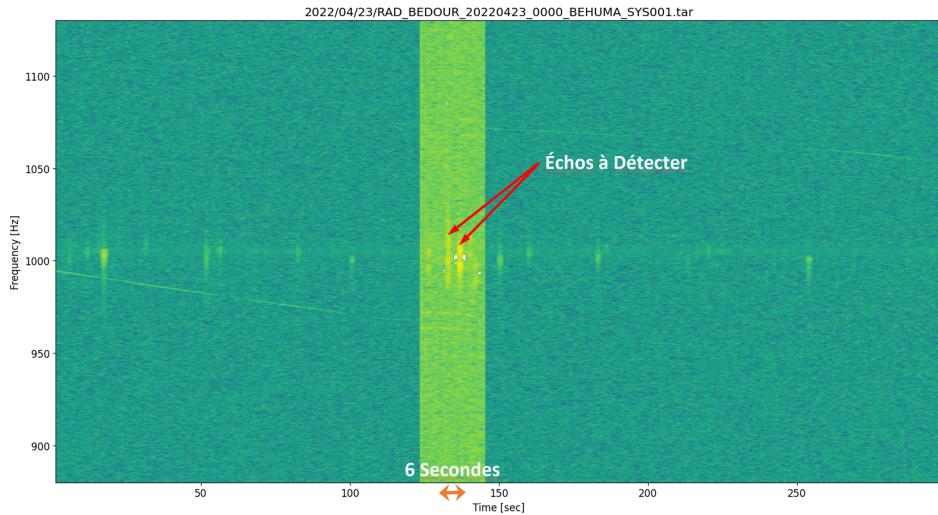
### A Images de l'interface web du monitoring



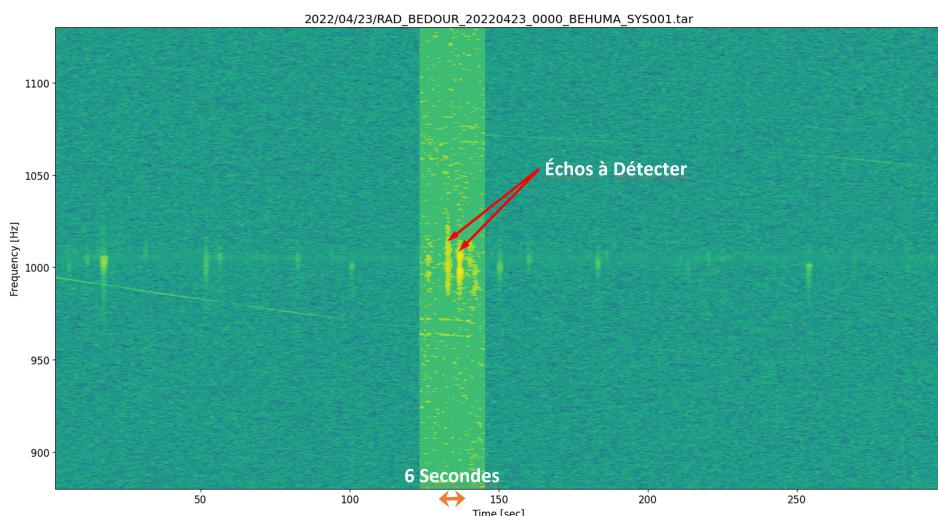
## B Kernel de Convolution Permettant d'Amplifier les Signaux de Météores

```
1      [[ 0.   0.   0.   50.   0.   0.   0.   ]
2      [ 0.   0.   0.   50.   0.   0.   0.   ]
3      [ 0.   0.   0.   0.   0.   0.   0.   ]
4      [ 0.   0.   0.   0.   0.   0.   0.   ]
5      [ 0.   0.   0.   0.   0.   0.   0.   ]
6      [ 0.   0.   0.   0.   0.   0.   0.   ]
7      [ 0.   0.   0.   0.   0.   0.   0.   ]
8      [ 0.   0.   0.   0.   0.   0.   0.   ]
9      [ 0.   0.   0.   0.   0.   0.   0.   ]
10     [ 0.   0.   0.   0.   0.   0.   0.   ]
11     [ 0.   0.   0.   0.   0.   0.   0.   ]
12     [ 0.   0.   0.   0.   0.   0.   0.   ]
13     [-1.5  0.   0.   0.   0.   0.   -1.5]
14     [-1.5  0.   0.   0.   0.   0.   -1.5]
15     [-1.5  0.   0.   0.   0.   0.   -1.5]
16     [ 0.   0.   0.   0.   0.   0.   0.   ]
17     [ 0.   0.   0.   0.   0.   0.   0.   ]
18     [ 0.   0.   0.   0.   0.   0.   0.   ]
19     [ 0.   0.   0.   0.   0.   0.   0.   ]
20     [ 0.   0.   0.   0.   0.   0.   0.   ]
21     [ 0.   0.   0.   0.   0.   0.   0.   ]
22     [ 0.   0.   0.   0.   0.   0.   0.   ]
23     [ 0.   0.   0.   0.   0.   0.   0.   ]
24     [ 0.   0.   0.   0.   0.   0.   0.   ]
25     [ 0.   0.   0.   0.   0.   0.   0.   ]
26     [ 0.   0.   0.   50.   0.   0.   0.   ]
27     [ 0.   0.   0.   50.   0.   0.   0.   ]]
```

## C Résultat du filtre pour amplifier les échos de météores



## D Résultat du filtre à percentile



## E Résultat des éléments étant trop petits pour pouvoir être un écho

