empreinte acoustique sur ESP32

identification de chansons avec le projet open source Olaf

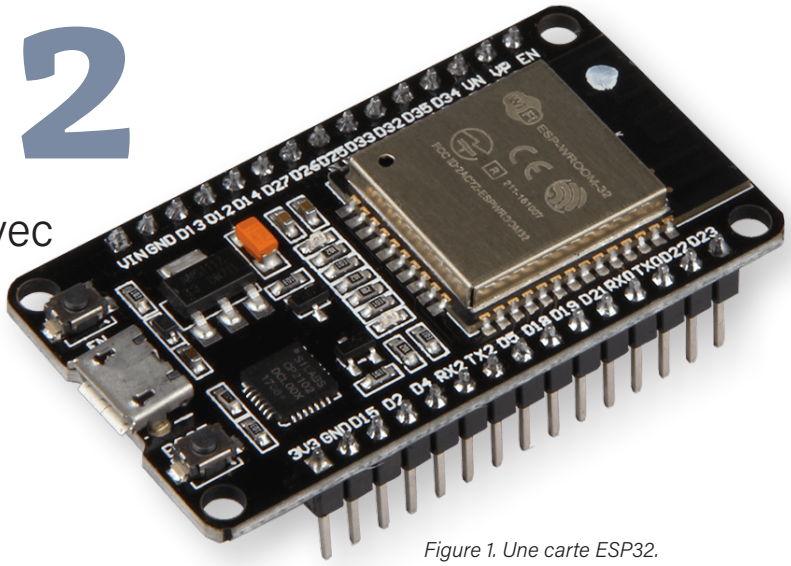


Figure 1. Une carte ESP32.

Joren Six, université de Gand (Belgique)

Il y a quelques années, Joren Six a été chargé de concevoir un dispositif d'informatique vestimentaire capable de reconnaître une chanson. Le code qu'il écrivit pour un module ESP32 donna naissance à Olaf, un projet multiplateforme et open source. Olaf, pour *Overly Lightweight Acoustic* Fingerprinting, est une bibliothèque d'identification musicale facile à utiliser et adaptée aux systèmes embarqués à mémoire et puissance de calcul limitées. Alors tous en rythme avec Olaf !

J'étais informaticien à l'université belge de Gand lorsque j'ai été chargé de mettre au point une technologie de reconnaissance audio pour un costume. Le cahier des charges impliquait de synchroniser les LED du costume sur le rythme d'une certaine chanson : seule cette chanson devait activer les LED, toute autre musique devant être ignorée. La reconnaissance et la synchronisation musicales sont généralement réalisées à l'aide de techniques d'empreintes audio. Le défi consistait à les implanter sur un microcontrôleur peu coûteux, disposant d'une puissance de traitement et d'une mémoire limitées. J'étais parvenu à créer un prototype fonctionnel, mais l'envie d'y revenir un jour ne me quitta plus vraiment. Nous étions en 2019.

L'occasion s'est présentée récemment, à l'approche du quatrième anniversaire de ma fille. Je me suis dit que je pourrais transformer cet ancien prototype en cadeau d'anniversaire extravagant : une robe « Elsa » qui réagit à la chanson *Libérée, délivrée* du film d'animation *La Reine des neiges* [1]. J'ai alors commandé une bande de LED RVB, une batterie Li-Ion, un microphone numérique I²S et, bien sûr, une robe Elsa. Je disposais déjà d'un microcontrôleur ESP32 (**figure 1**) et m'en suis donc servi. Il prend en charge I²S, comprend une unité à virgule flottante (FPU), dispose d'une mémoire vive suffisante, et est facile à relier à une bande de LED. L'unité FPU permet d'éviter les calculs en virgule fixe et facilite ainsi la réutilisation du code à la fois sur PC et sur dispositifs embarqués.

Assemblage de la version à ESP32

La version initiale du projet reposait sur une carte *ESP32 Thing* de Sparkfun – choisie pour son connecteur de batterie pratique – à laquelle était relié un microphone de type MEMS (un I*NMP441*) et une bande de LED adressables (**figure 2**) – un modèle générique, facilement trouvable sur eBay ou AliExpress.

Le câblage de ces éléments relève de la routine. Pour le microphone, il s'agit d'en relier les broches d'E/S *SDA*, *SCK* et *WS* à des broches appropriées du ESP32, par exemple les broches GPIO 32, 33 et 35. Sans oublier, bien sûr, l'alimentation du microphone.

Côté bande de LED, le câblage dépend du modèle utilisé. Pour un ruban à LED *WS2812B*, trois fils de connexion sont nécessaires (alimentation, masse, données). Je me suis appuyé sur la bibliothèque *FastLED* pour l'écriture du code, donc assurez-vous que votre ruban à LED est pris en charge par cette bibliothèque si vous souhaitez reproduire ce projet. L'ensemble une fois soudé et en place, restait à coudre la bande de LED sur le costume, tâche que je laissai aux mains expérimentées de ma meilleure moitié. La vidéo du lien [1] montre le résultat final.

Figure 2. Ruban à LED RGB.

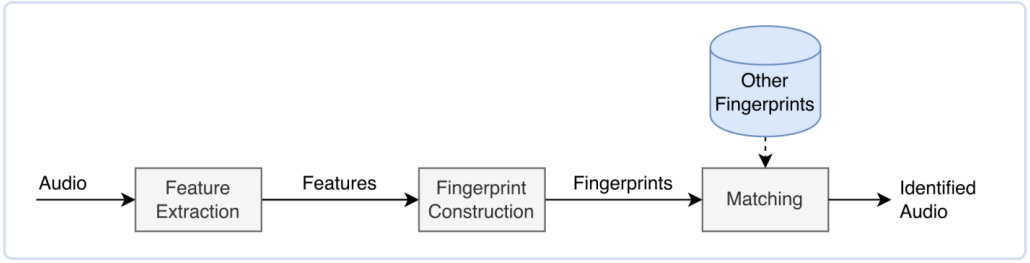


Figure 3. Les étapes de l'identification par empreintes acoustiques.

La première chanson n'est pas reconnue et n'active donc pas les LED, mais ces dernières entament bien leur sarabande lumineuse lorsque *Let it go* (*Libérée, délivrée*) se fait entendre. Les LED restent allumées un bref instant lorsque la chanson est mise sur pause, une caractéristique choisie à dessein pour tenir compte des « trous » durant la reconnaissance. La chanson est correctement identifiée lorsqu'elle reprend. Vous trouverez sur [2] les actualités de ce projet.
Bien entendu, comme souvent avec les projets à microcontrôleur, c'est du code que provient toute la magie.

Comment ça marche ?

Comment apprendre à un ordinateur à identifier une chanson ou de la musique ? Plusieurs techniques sont possibles. L'une d'elles, courante et sur laquelle reposent Olaf et le service d'identification musicale *Shazam*, est la reconnaissance par crête spectrale.
L'idée est simple : l'appli transforme le fichier audio enregistré sur le téléphone en un format qu'un processeur saura aisément comparer à d'autres chansons (**figure 3**). Ce processus repose sur l'empreinte acoustique de la chanson, c'est-à-dire sur un condensé numérique de ladite chanson en une signature unique, identifiable même parmi un bruit de fond sonore intense.
Une telle signature repose sur le spectrogramme de la chanson, autrement dit sur son diagramme temps-fréquence. Un spectrogramme indique aussi le niveau de puissance du signal audio au moyen de couleurs. Cette puissance reflète le niveau sonore perçu par l'oreille. Toutefois, si un service comme *Shazam* comparait directement plusieurs spectrogrammes entre eux pour identifier une chanson, le résultat n'arriverait que bien après la fin de celle-ci. Une meilleure approche consiste à comparer les pics du spectrogramme puisque ces pics contiennent des informations essentielles, y compris pour le cerveau humain.
Olaf enregistre donc un échantillon sonore, en calcule le spectrogramme, puis simplifie celui-ci en un diagramme de dispersion des pics de fréquence – surlignés par des points verts sur le spectrogramme de la **figure 4**.
Ces diagrammes de dispersion, qui pour l'essentiel représentent les signaux les plus caractéristiques du spectrogramme, doivent ensuite être comparés aux entrées d'une base de données contenant de nombreuses chansons ou, dans le cas du costume d'Elsa, être comparés à une seule chanson.
Plutôt que de rechercher dans une base de données de chansons la suite de points correspondante, il s'avère plus astucieux de relier par paires les pics les plus proches. Si en parcourant la base de données l'algorithme trouve suffisamment de paires correspondantes ayant un alignement temporel identique, Olaf (ou *Shazam*) peut identifier la chanson.

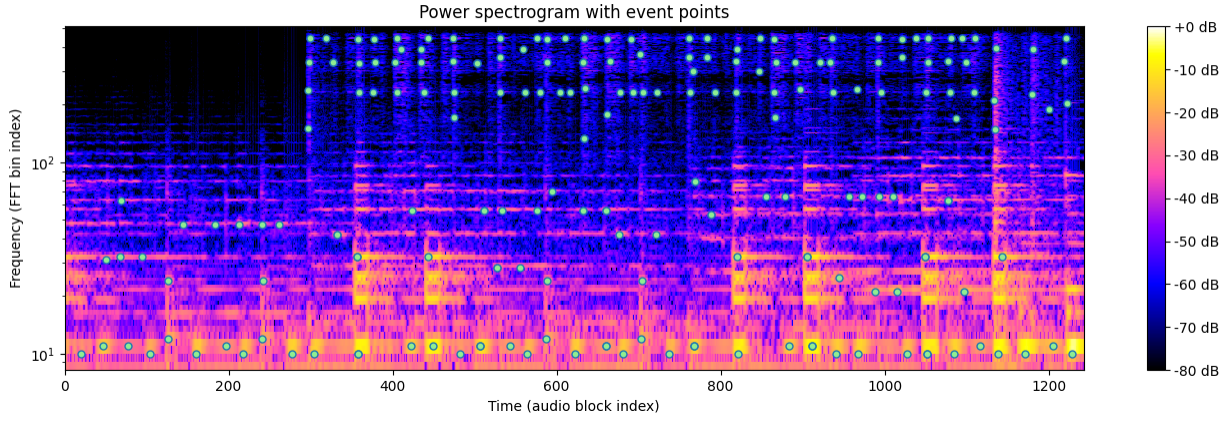


Figure 4. Un spectrogramme représente les fréquences sur un intervalle de temps donné. Les points verts sont extraits par Olaf et indiquent les pics de fréquence.

Évolution du projet

Le code du prototype initial de 2019 n'était pas très bien organisé, mais celui de la seconde version était bien plus clair, au point qu'il ne me parut pas déraisonnable de le partager sur GitHub. C'est à ce moment-là que j'ai baptisé le projet Olaf, pour *Overly Lightweight Acoustic Fingerprinting* [3].

J'ai progressivement étoffé le code pour l'amener à faire bien plus que son objectif premier. Le projet est ainsi devenu un système d'empreinte acoustique polyvalent, paré pour de multiples applications. Olaf fait montre de performances impressionnantes, grâce notamment à une exploitation frugale des ressources matérielles et à son recours à la bibliothèque *PFFFT*.

Le nom de cette bibliothèque ne vous dit sans doute pas grand-chose, surtout si, comme Bob Pease, votre langage de programmation préféré est le fer à souder ! PFFFT [4] est l'acronyme de *Pretty Fast FFT*, une bibliothèque conçue pour être plus légère que la renommée *FFTW*. Elle est de fait compacte et rapide, ce qui la rend idéale pour le ESP32.

Sur un dispositif embarqué, les empreintes de référence sont stockées en mémoire, ce qui élimine le besoin d'une base de données. Sur un ordinateur classique, ces empreintes sont stockées dans une base de données très performante de type *LMDB* – la décrire en détail nous entraînerait trop loin, je vous encourage à en découvrir vous-même les avantages.

À l'origine simple gadget, Olaf est devenu une application/bibliothèque complète pour l'extraction, le stockage et la recherche d'empreintes acoustiques. Olaf sait extraire ces empreintes d'un flux audio de façon efficace, les stocker dans une base de données et, comme expliqué plus haut, rechercher leur correspondance à partir de paires de pics (technique appelée *landmark-based acoustic fingerprinting*).

J'ai aussi écrit Olaf car il ne semblait pas y avoir beaucoup de bibliothèques semblables, à la fois légères, faciles à déployer sur un système embarqué et ne requérant guère de mémoire et de ressources de calcul. Olaf est écrit en C et se veut aussi portable que possible. Il vise principalement les dispositifs à ARM 32 bits (comme divers modèles *Teensy*), certaines cartes Arduino et le module ESP32. D'autres plateformes de spécifications similaires pourraient être compatibles. Nul doute qu'Olaf, qui est open source, stimulera la création de projets innovants combinant reconnaissance musicale/sonore et dispositifs pour l'IdO !

Olaf fonctionne également sur PC, et rapidement. Le code peut être compilé et exécuté localement sur une machine, mais aussi être lancé depuis un navigateur web ! L'encadré **Exécuter Olaf depuis un navigateur web** explique comment procéder.

Le code source est sur GitHub, accompagné d'un exemple pour ESP32 et de petits outils de débogage écrits par mes soins. L'archive du lien [6] contient aussi un fichier de projet *PlatformIO*.

Déployer Olaf sur un ESP32

Lors du développement du projet, Olaf avait besoin d'un minimum de RAM pour le traitement audio, ce qui posait souvent problème avec de nombreux microcontrôleurs. La version PC à stockage clé-valeur nécessitait 512 Ko de RAM, alors que la version embarquée n'exigeait que 200 Ko avec le module ESP32. Comme je ne travaille pas souvent avec des microcontrôleurs, il me fallait un environnement de développement intégré facile à configurer et à utiliser. *PlatformIO* et l'EDI *Arduino* ont été à cet égard parfaits pour un utilisateur comme moi. Ils n'accaparaient pas mon processeur et reconnaissaient bon nombre de microcontrôleurs (dont Teensy 3+, RP2040 et les Cortex-M). Une unité FPU peut par ailleurs réduire la consommation d'énergie, un point essentiel lorsqu'on travaille sur batterie.

Je me suis aussi servi de quelques modules *M5StickC* de M5Stack, car ils sont faciles à programmer et ont un microphone intégré – donc parfaits pour une personne comme moi plus intéressée par le logiciel que le matériel, même si je me suis déjà attaqué à quelques projets de domotique (voir mon site web), dont une commande de ventilation et la surveillance du niveau d'un réservoir d'eau de pluie.

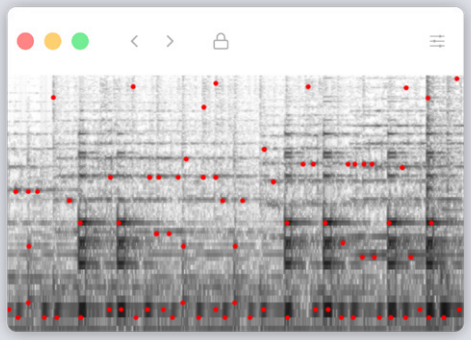
J'ai ajouté pour cet article un code de démo ESP32 à mon dépôt GitHub [7] et complété la documentation. La dernière version d'Olaf donne des résultats fiables sur ESP32 et avec un microphone MEMS – facile à se procurer. J'explique sur mon blog [6] comment utiliser des microphones I²S comme le *INMP441*.

Exécuter Olaf depuis un navigateur

Connaissez-vous WebAssembly, ou WASM [5] ? WASM est un format de code binaire portable, exécutable par un navigateur web. Sa spécification décrit aussi sa représentation au format texte. Son principal objectif est de faciliter le développement d'applications performantes pour le web.

Du code écrit en C peut être compilé en WebAssembly avec Emscripten. Ce compilateur permet, dixit sa documentation, d'exécuter du code C/C++ sur une page web à une vitesse presque native, sans recourir à des extensions. Combiner l'API Web Audio et la version WASM d'Olaf ouvre la voie à des applications d'empreintes acoustiques pour l'internet.

Sur mon blog, vous pouvez tester Olaf depuis votre navigateur – et lire mes derniers articles à ce sujet. Le code qui s'exécute sur l'ESP32 s'exécutera dans votre navigateur, autrement dit Olaf tentera d'identifier la chanson Let It Go de la bande originale de Frozen (La Reine des neiges). Lancez la vidéo YouTube embarquée, puis lancez Olaf (à droite de la vidéo). Olaf analysera le flux audio du microphone de votre PC, calculera sa transformée de Fourier rapide (FFT) et la visualisera avec Pixi.js. Les empreintes rouges (cf. capture d'écran) devraient devenir vertes au bout de quelques secondes si l'identification a réussi. Elles repasseront au rouge lorsque vous stopperez la chanson. Comme dans la démo vidéo mentionnée dans l'article, la transition visuelle entre correspondance et non-correspondance est temporisée pour tenir compte des « trous » durant la reconnaissance.



Le dépôt GitHub contient un programme de démonstration qui transmet le son du microphone à un PC par Wi-Fi. Il permet de s'assurer que le microphone fonctionne et que les échantillons audio sont bien interprétés. Taille des tampons, fréquences d'échantillonnage, formats audio et autres paramètres stéréo et mono sont conformes au protocole I²S. Un autre code d'exemple montre comment se fait la correspondance entre l'audio d'un microphone *INMP441* et les empreintes acoustiques. Contrairement à la version PC qui repose sur une base *LMDB* à stockage clé-valeur, la version embarquée d'Olaf utilise une liste de hachages stockée dans le fichier d'en-tête *src/olaf_fp_ref_mem.h*. Ce fichier est l'index des empreintes acoustiques.

Le fichier d'en-tête *src/olaf_fp_ref_mem.h* est par défaut inclus dans le code de l'ESP32. Pour le tester et le déboguer, utilisez la version *mem* d'Olaf sur votre ordinateur : `bin/olaf query olaf_audio_your_audio_file.raw "arandomidentifier"`. La version ESP32 est identique à la version *mem*, si ce n'est que l'audio provient de l'entrée d'un microphone MEMS et non d'un fichier.

Une fois testés avec succès la version *mem* d'Olaf et le microphone *INMP441*, vous pouvez déployer la version ESP32 sur le microcontrôleur à l'aide de l'EDI *Arduino*.

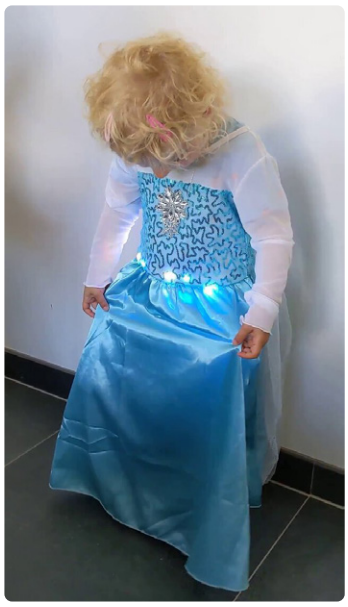


Figure 5. Le costume alimenté par ESP32.

Olaf et vous

J'espère que cet article suscitera en vous des idées de projets reposant sur l'identification acoustique : savoir que l'amélioration progressive d'un petit projet au départ purement ludique peut vite conduire à des résultats impressionnants est toujours motivant. Olaf est ainsi passé d'un premier prototype qui ne répondait guère à mes attentes à un second prototype à ESP32 largement amélioré, beaucoup plus satisfaisant pour moi et… ma fille (**figure 5**).

Olaf est devenu au fil du temps et d'itérations du code une application (et une bibliothèque) complète, open source, portable, multi-plateforme et très performante. J'ai écrit deux articles universitaires sur le sujet afin qu'Olaf soit reconnu et apprécié par la communauté des chercheurs. Et vous, lecteurs et lectrices : quels projets passionnants avez-vous en tête pour votre ESP32 ?

VF : Hervé Moreau — 230578-04

Questions ou commentaires ?

Contactez l'auteur (joren.six@ugent.be) ou Elektor (redaction@elektor.fr).

À propos de l'auteur

Joren Six est informaticien et chercheur en informatique musicale, recherche d'informations musicales et ethnomusicologie informatique. Il est titulaire d'un doctorat de l'université de Gand (Belgique) et travaille actuellement sur plusieurs projets mêlant informatique et acoustique.

Produits

- **Carte de développment ESP32 JOY-iT NodeMCU**
  www.elektor.fr/19973
- **Carte ESP32-DevKitC-32E**
  www.elektor.fr/20518
- **Carte ESP32-C3-DevKitM-1**
  www.elektor.fr/20324

LIENS

[1] Premier prototype du projet : https://0110.be/posts/Olaf_-_Acoustic_fingerprinting_on_the_ESP32_and_in_the_Browser
[2] Actualités du projet Olaf : https://0110.be/search?q=olaf
[3] Dépôt du projet Olaf : https://github.com/JorenSix/Olaf
[4] Bibliothèque Pretty Fast FFT : https://bitbucket.org/jpommier/pffft/src/master/
[5] WebAssembly sur Wikipédia : https://en.wikipedia.org/wiki/WebAssembly
[6] Fichiers à télécharger : https://0110.be/files/attachments/475/ESP32-Olaf.zip
[7] Version ESP32 d'Olaf sur GitHub : https://github.com/JorenSix/Olaf/tree/master/ESP32