J3eA, Journal sur l'enseignement des sciences et technologies de l'information et des systèmes
J3eA 24 (2025) 10
https://doi.org/10.1051/j3ea/20250010
https://www.j3ea.org

Conception d'un synthétiseur audio compatible MIDI sur STM32 pour l'enseignement des systèmes embarqués et du traitement du signal numérique*

Vincent Choqueuse[1*], Vincent Kerhoas[2], Yann Le Masson[3], Johny Kessler[4] Marc Le Roy[5]

[1] *Lab-STICC, UMR CNRS 6285, ENIB, Bretagne INP, Brest*
[2] *Département Électronique, ENIB, Bretagne INP, Brest*
[3] *Arturia, Grenoble*
[4] *STMicroelectronics, Le Mans*
[5] *Lab-STICC, UMR CNRS 6285, IUT Brest-Morlaix, UBO, Brest*

*Auteur de correspondance : choqueuse@enib.fr

Résumé : *Cet article présente un projet pédagogique destiné aux étudiants de dernière année du BUT GEII (parcours électronique et systèmes embarqués) à l'IUT de Brest-Morlaix. L'objectif est de les amener à mobiliser et articuler plusieurs compétences — traitement du signal, électronique, systèmes embarqués et programmation en C. Le projet s'appuie sur une carte STM32 Discovery Kit et se déroule en deux étapes. Durant les 16 premières heures, les étudiants développent les briques fondamentales d'un synthétiseur audio numérique (oscillateurs numériques avec tables d'ondes, filtre numérique paramétrique, modulateurs et interface MIDI via USB). Les 4 dernières heures sont consacrées à un mini-projet au cours duquel chaque étudiant conçoit et met en œuvre son propre synthétiseur « customisé ». Les retours recueillis lors de l'évaluation de l'enseignement mettent en avant l'intérêt et la richesse de ce projet pluridisciplinaire.*

Mots clés : Traitement numérique du signal, Système embarqué, Électronique.

Abstract: **Article title translated in English**
This article presents an STM32-based project designed for final-year GEII BUT students in Electronics and Embedded Systems at the IUT of Brest-Morlaix. The objective of this project is to encourage students to integrate multiple skills in signal processing, electronics, embedded systems, and C programming during a 20-hour project. The project is implemented on an STM32 Discovery kit and is divided into two main phases: the first 16 hours are dedicated to

developing the fundamental building blocks of a digital sound synthesizer (digital oscillators with interpolated lookup wavetables, parametric digital IIR filter, modulators, and MIDI over USB connection). During the remaining four hours, students design and implement their own synthesizer concepts based on their interests, under the supervision and guidance of the instructor. Student feedback highlights the project's effectiveness in bridging theoretical knowledge and hands-on implementation.

Keywords: Digital Signal Processing, Embedded System, Electronics

1. Introduction

Dans le cadre des enseignements en électronique et en informatique, la conception d'un synthétiseur audio constitue un exemple particulièrement pertinent de projet transdisciplinaire. D'une part, ce type de projet permet aux étudiants de mettre en pratique, de façon à la fois concrète et ludique, des notions fondamentales telles que l'utilisation de VCO, la mise en œuvre de filtres et de modulateurs analogiques, ou encore l'analyse temporelle et fréquentielle des formes d'onde à l'aide d'un oscilloscope. D'autre part, les projets en lien avec la musique suscitent généralement un fort intérêt et favorisent l'engagement des étudiants.

Sur le plan pédagogique, les projets de synthèse sonore sont le plus souvent introduits en début de cursus, dans le cadre de cours d'électronique ou de traitement du signal. Ces projets se focalisent principalement sur la conception de synthétiseurs analogiques, puisque l'étude du domaine analogique précède généralement celle du numérique [8]. Plusieurs travaux confirment l'intérêt de cette approche. Par exemple, Martin et al. [3] décrivent un projet de synthétiseur analogique mené par deux étudiants du département GEII de l'IUT de Brest-Morlaix, tandis qu'un projet similaire a été proposé à des étudiants de première année de l'ENIB (Bretagne INP) dans le cadre d'un cours d'électronique. Dans ces deux cas, l'application musicale a été particulièrement appréciée. Concernant l'intégration du numérique, de nombreux établissements utilisent des outils de simulation tels que Matlab, Python [1, 7] ou LabVIEW pour illustrer des concepts comme le filtrage numérique, l'échantillonnage, etc. Bien que ces outils soient très efficaces pour des démonstrations ou du prototypage rapide, ils restent généralement mal adaptés aux applications audio temps-réel. De plus, certains de ces outils (par exemple Matlab et LabVIEW) sont coûteux pour les établissements et peuvent nécessiter des renouvellements de licence. Pour la réalisation de synthétiseurs audio-numériques, une autre approche consiste à utiliser des environnements de programmation visuelle optimisés pour l'audio temps-réel tels que Pure Data, Max/MSP ou Reaktor. Cependant, ces environnements visuels sont principalement destinés aux sound designers et ne conviennent pas nécessairement aux étudiants suivant des cursus orientés en électronique ou en informatique car ces environnements de haut niveau font abstraction de la plupart des concepts sous-jacents de traitement du signal. Certaines API ou environnements de codage spécifiques (par ex. : l'API Web Audio, SuperCollider ou Sonic Pi) présentent le même inconvénient.

Le développement d'applications audio temps réel pose de nombreux défis, en particulier en matière de latence et de contraintes de calcul. Sous Windows, Linux ou macOS, les entreprises du secteur de l'informatique musicale s'appuient généralement sur **JUCE**, un framework C++ spécialisé dans le développement d'applications audio multiplateformes (instruments virtuels et plugins). JUCE fournit une API de haut niveau qui masque les spécificités des systèmes

d'exploitation et propose des modules DSP intégrés, facilitant ainsi le développement. Plusieurs études ont d'ailleurs exploré son utilisation dans des programmes de premier cycle en génie électrique et en informatique industrielle [5]. Néanmoins, bien que ce framework puisse présenter un intérêt pour des enseignements en informatique, il est moins adapté aux étudiants suivant des cursus en électronique car la plupart des algorithmes DSP sont déjà implémentés. Une autre approche consiste à utiliser une carte de développement et à implémenter les algorithmes DSP *from scratch* dans un langage de bas niveau tel que le C. Cette méthode offre plusieurs avantages par rapport à JUCE : une latence réduite et l'absence de surcoûts liés au système d'exploitation. A titre d'exemple, l'étude [2] présente un projet audio basé sur une STM32 Discovery, un microphone MEMS Adafruit I2S Breakout et un décodeur stéréo I2S. Ce projet se concentre sur la mise en œuvre de divers effets audio DSP, tels que la transformation de voix, la réalisation de filtres numériques et la réalisation d'algorithmes de *pitch-shifting.*

Dans cet article, nous présentons un projet de réalisation d'un synthétiseur audio mené avec une classe de 12 étudiants de dernière année du BUT GEII. Contrairement à l'étude décrite dans [2], notre projet porte sur la mise en œuvre d'un synthétiseur numérique contrôlé par un clavier/contrôleur MIDI, développé à l'aide d'une STM32 Discovery Kit qui intègre nativement l'ensemble des éléments nécessaires à ce type de réalisation. L'article est structuré comme suit : la section 2 décrit le contexte pédagogique, les acquis d'apprentissage visés ainsi que le matériel et les logiciels utilisés ; la section 3 présente un exemple de synthétiseur conçu par les étudiants ; enfin, la section 4 propose une analyse des retours recueillis lors de l'évaluation de l'enseignement.

2. Objectifs de l'Enseignement et Mise en Œuvre

Cette section présente le contexte et les objectifs de l'enseignement ainsi que les outils matériels et logiciels utilisés.

2.1. Contexte et Objectifs

Ce projet est proposé dans un enseignement intitulé « Traitement numérique du signal audio » dispensé en troisième année du cursus BUT GEII à l'IUT de Brest-Morlaix [4]. Cet enseignement est uniquement suivi par les étudiants de la filière Électronique et Systèmes Embarqués (ESE) [4]. Dans la maquette du GEII, cet enseignement fait partie du module R5.ESE.10, qui s'inscrit dans la suite du module R4.ESE.07 (Électronique spécialisée avec une section consacrée aux bases des filtres numériques) de deuxième année. L'objectif de cet enseignement est principalement de faire le lien entre les concepts théoriques de l'électronique classique (synthèse de formes d'onde et analyse dans le domaine temporel et fréquentiel, synthèse numérique directe du module R3.ESE.15) et le filtrage numérique avec une mise en œuvre pratique sur un système temps-réel. En termes d'organisation, le volume d'enseignement programmé est de 20 h (5 fois 4h). Lors des différentes séances, les étudiants travaillent en binôme. Le cours comprend 16 h de tutoriels guidés, suivies de 4 heures de conception libre, pendant lesquelles les étudiants doivent réaliser un synthétiseur « customisé ». L'évaluation repose à la fois sur l'avancement des tutoriels et sur une évaluation par les pairs du synthétiseur customisé de chaque groupe. Dans le programme du GEII, cet enseignement adresse la compétence « Mise en œuvre d'un système matériel ou logiciel » dans le développement de

systèmes embarqués (AC34.ESE « Implanter »). Le tableau 1 présente les acquis d'apprentissage visés du projet.

Tableau 1**.** Compétence « Mise en œuvre d'un système matériel ou logiciel ».

Catégorie	Acquis d'Apprentissage Visés
Compétences Techniques	• Implémenter des algorithmes DSP en temps réel en C. • Développer un système embarqué à l'aide de microcontrôleurs STM32. • Gérer la communication USB MIDI pour le traitement audio en temps réel. • Mettre en œuvre des oscillateurs numériques, des filtres et des enveloppes.
Maîtrise de la programmation	• Renforcer les compétences en programmation C, notamment en matière de gestion de la mémoire et d'optimisation des performances. • Travailler avec la programmation embarquée de bas-niveau et la configuration des périphériques. • Mettre en œuvre une conception logicielle embarquée structurée et modulaire.
Conception et Intégration de Systèmes	• Comprendre l'architecture d'un synthétiseur numérique. • Intégrer les composants matériels (contrôleur MIDI, carte STM32, sortie audio). • Dépanner et déboguer des applications de traitement audio en temps réel
Ingénierie pratique, Ingénierie et outils	• Utiliser STM32CubeIDE pour le développement embarqué. • Analyser des signaux à l'aide de Python (NumPy, SciPy, Matplotlib). • Travailler avec des oscilloscopes et des outils de mesure des signaux analogiques

2.2. Matériels utilisés

Dans ce projet, les équipements les plus coûteux sont le microcontrôleur et le contrôleur MIDI. Pour choisir ces équipements, nous avons préféré opter pour des produits de fabrication française (Arturia et STMicroelectronics).

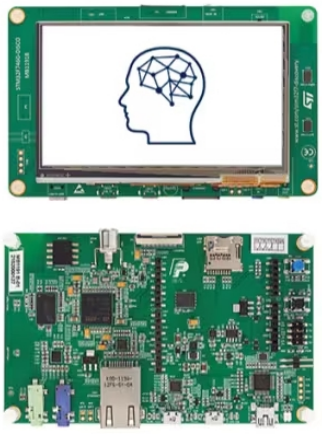

Figure 1. STM32F476G-DISCO

Figure 2. Contrôleur MIDI Arturia MKII

2.2.1. Carte STM32F746G-DISCO

Concernant le choix de la carte, le projet est basé sur une STM32F746G-DISCO (voir figure 1), dont le prix est d'environ 66,70 €. Nous avons choisi l'écosystème STM32 car il est utilisé

par de nombreux partenaires industriels de l'IUT de Brest-Morlaix et de l'ENIB. Le microcontrôleur STM32F746G-DISCO est équipé d'un processeur Arm® Cortex®-M7, de 1Mo de mémoire Flash, de 340 Ko de RAM et de 128 Mo de SDRAM. La carte comprend plusieurs périphériques adaptés à la réalisation d'un synthétiseur audio. En particulier, elle dispose de ports USB OTG HS/FS permettant la réception de messages MIDI via USB, d'un codec audio SAI avec une sortie analogique stéréo et d'un écran LCD-TFT RVB de 4,3 pouces (480×272) pour l'affichage de diverses informations. Enfin, elle dispose de connecteurs pour cartes microSD et d'un port Ethernet RJ45. Notons que, bien que les cartes microSD et le port Ethernet ne soient pas directement utilisés dans notre projet, ils offrent une flexibilité pour de futures extensions, telles que la réalisation d'un échantillonneur audio ou d'une boîte à rythme.

2.2.2. Contrôleur MIDI Arturia Minilab MKII

Pour générer les messages MIDI, le projet utilise un clavier/contrôleur MIDI Arturia Minilab MKII (voir figure 2), disponible pour environ 69 €. Ce contrôleur est doté d'un clavier à 25 touches sensibles à la vélocité (messages *Note On* / *Note Off*), ainsi que d'une surface de contrôle composée de 16 boutons rotatifs assignables (messages *Control Change*). Les encodeurs 1 et 9 peuvent également être utilisés comme des boutons, ce qui permet d'implémenter des fonctionnalités telles que la sélection de formes d'onde. Le Minilab MKII comprend également deux bandes tactiles pour la modulation et huit pads. Le port USB Type-B du Minilab est utilisé à la fois pour la transmission des messages MIDI et pour l'alimentation du contrôleur. Nous avons également testé la version MKIII du Minilab, mais celle-ci est actuellement incompatible avec notre projet en raison du passage du port USB Type-B à l'USB-C. Le contrôleur MIDI est connecté au port micro-USB de la carte STM32 via un câble Micro-USB OTG compatible (≤ 3 €). La sortie audio analogique de la STM32 est envoyée simultanément à des enceintes audio et à un oscilloscope via un *splitter* analogique conçu par le département GEII de l'IUT de Brest-Morlaix.

2.3. Outils Logiciels

Pour faciliter le développement en C sur STM32, le projet repose sur l'environnement de développement STM32CubeIDE. Pour tester et analyser leurs algorithmes hors ligne, les étudiants sont également encouragés à utiliser Python/Conda.

- **STM32CubeIDE** : STM32CubeIDE est un environnement de développement intégré (IDE) développé par STMicroelectronics pour la programmation et le débogage des microcontrôleurs STM32. Cet outil est gratuit et multiplateforme (Windows, macOS, Linux) et utilise une chaîne d'outils basée sur Arm GCC.
- **Python / Conda** : Conda est un gestionnaire de paquets et d'environnements multiplateforme qui permet la création d'environnements Python isolés. Dans ce projet, un environnement Conda dédié intégrant NumPy, SciPy et Matplotlib est utilisé pour tester différents algorithmes en temps différé, tels que l'analyse de filtres et la génération d'oscillateurs par tables d'ondes.

2.4. Support de cours

Le cours est disponible sous la forme d'un site web statique rédigé en français et hébergé sur la plateforme cloud AWS S3[1]. La figure 3 illustre l'interface du site. Le contenu est créé à l'aide du générateur de site statique **VitePress**.

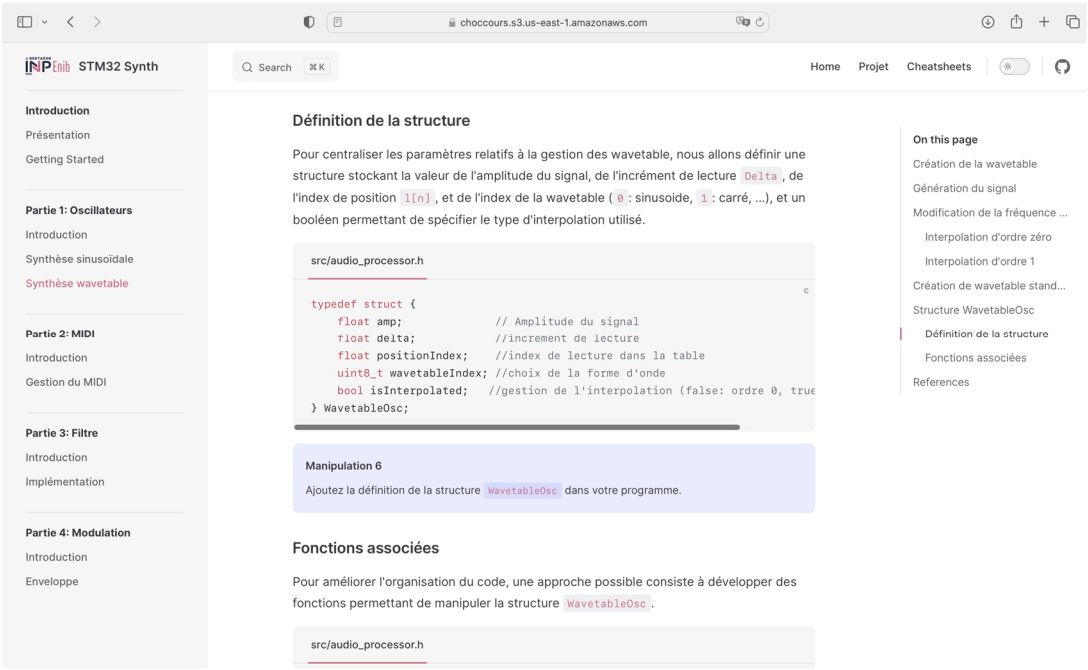

Figure 3. Support de cours VitePress.

Le générateur VitePress offre plusieurs avantages pour la conception des cours/tutoriels : il permet de copier-coller facilement du code C, de naviguer de manière fluide entre les sections et les ressources du site, et d'effectuer des recherches grâce à une barre de recherche intégrée. Le template VitePress de base prend en charge les modes clair/sombre. Pour les enseignants, VitePress présente également plusieurs atouts. D'une part, il permet de créer du contenu facilement en utilisant le format Markdown. D'autre part, il est possible d'utilisation des équations LaTeX ou des schémas-blocs rapidement via des extensions tierces.

Le code source de base du projet est disponible sur GitHub[2]. Il s'inspire du projet Dekrispator[3], basé sur une autre carte STM32. Contrairement au projet Dekrispator, le code disponible sur GitHub est destiné à un usage pédagogique : en particulier, il ne contient de base aucun algorithme de traitement du signal.

3. Conception du Synthétiseur Numérique

Dans le cadre de ce projet, les tutoriels disponibles sur le site permettent aux étudiants d'implémenter un synthétiseur monophonique soustractif inspiré du Minimoog Model D.

[1] https://choccours.s3.us-east-1.amazonaws.com/audio/content/index.html
[2] https://github.com/vincentchoqueuse/STM32F746_Disco_Default_Synth
[3] https://github.com/MrBlueXav/Dekrispator_v2

3.1. Minimoog Model D

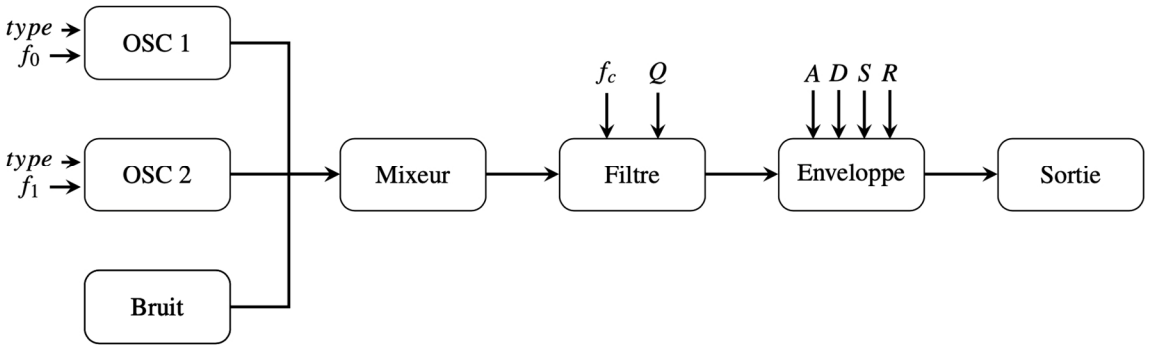

Figure 4. Architecture de base d'un synthétiseur analogique de type Minimoog Model-D. Le Minimoog Model D est composé de deux oscillateurs parallèles capables de générer différentes formes d'ondes (dent de scie, carrée, triangulaire, sinusoïdale), ainsi que d'un générateur de bruit. Leurs sorties sont additionnées et envoyées à l'entrée d'un filtre de fréquence de coupure et de résonance contrôlables. En sortie de filtre, l'amplitude du signal est modulée par une enveloppe ADSR.

Sorti en 1970, le Minimoog Model D fut l'un des premiers synthétiseurs analogiques portables au monde. Son timbre très caractéristique a été utilisé par de nombreux artistes emblématiques en musique électronique, hip-hop et pop music. Parmi les utilisateurs célèbres du Minimoog, nous pouvons citer Herbie Hancock, Kraftwerk, Jean-Michel Jarre, Pink Floyd, Michael Jackson, Radiohead, Dr. Dre, Air, Daft Punk, Deadmau5, etc.

Au niveau sonore, le Minimoog est un synthétiseur monophonique principalement utilisé pour générer des sons de basse ou des sonorités de pad (son évoluant très lentement). L'architecture du Minimoog est présentée dans la figure 4 et repose sur la synthèse soustractive. La synthèse soustractive s'articule autour de deux éléments :

1. Un ou des oscillateurs riches en harmoniques (ondes en dents de scie, triangulaires, carrées, etc.) ;
2. Un filtre, potentiellement variable dans le temps, permettant de modifier le contenu spectral de l'oscillateur.

Sur le plan pédagogique, l'architecture d'un synthétiseur soustractif permet de mobiliser différents concepts clés du programme national GEII : oscillateurs, filtres numériques, machines à états (ADSR) et protocoles de communication (MIDI).

3.2. Éléments de base du synthétiseur.

3.2.1. MIDI via USB

Tableau 2. Structure des messages MIDI (format standard 3 octets et format USB 4 octets). La valeur {x} correspond au numéro de canal.

Message MIDI	Octet 1	Octet 2	Octet 3	Octet 4

Générique	(USB uniquement)	État	Donnée 1	Donnée 2
Note ON	(USB uniquement)	`0x9{x}`	Pitch	Vélocité
Note OFF	(USB uniquement)	`0x8{x}`	Pitch	Vélocité
Control Change	(USB uniquement)	`0xB{x}`	Identifiant	Valeur

Le **MIDI** (*Musical Instrument Digital Interface*) est un protocole de communication permettant à des instruments de musique électroniques, à des ordinateurs ou à d'autres équipements d'échanger des informations simples. Les messages MIDI sont envoyés sous forme séquentielle et transitent via un câble MIDI ou USB. Chaque message se compose d'un octet d'état suivi d'un ou deux octets de données. Le protocole prend en charge 16 canaux MIDI et utilise une résolution de 7 bits (valeurs de 0 à 127) pour la plupart des données. Lorsqu'ils sont transmis via USB, les messages comprennent également un octet de préambule.

Dans le code du projet, le fichier **midi_processor.c** contient la fonction nommée `process_midi_data(.)`, déclenchée à chaque réception d'un ou plusieurs messages MIDI. De leur côté, les étudiants doivent implémenter l'extraction des messages et mapper les informations vers les paramètres du synthétiseur audio.

- Pour les messages *Note On*, la hauteur (*pitch*) est associée à la fréquence de l'oscillateur et la vélocité à son amplitude.
- Les messages *Note On* et *Note Off* déclenchent respectivement les phases d'attaque et de relâchement de l'enveloppe ADSR.
- Pour les messages *Control Change*, les étudiants sont libres d'attribuer chaque CC à un paramètre du synthétiseur (par ex. : fréquence de coupure du filtre,).

3.2.2. Oscillateurs numériques

L'oscillateur est l'élément central d'un synthétiseur. Il génère un signal audio périodique à une fréquence fondamentale f_o. En programmation, les étudiants sont souvent habitués à générer des signaux périodiques à l'aide de fonctions trigonométriques, telles que la fonction `sin(.)`. Cependant, dans les applications audio temps réel, il est important de souligner que l'évaluation directe de ces fonctions trigonométriques est souvent trop coûteuse en temps de calcul, en particulier lorsqu'elles doivent être appelées fréquemment (par exemple, pour un signal audio échantillonné à $F_s = 44100\ Hz$). Dans ce contexte, la plupart des synthétiseurs numériques utilisent des stratégies alternatives afin de réduire la complexité calculatoire. Dans le cadre de ce projet, les étudiants évaluent l'efficacité et la complexité de différentes approches :

1. **Implémentation naïve** d'un oscillateur sinusoïdal à l'aide de la formule directe $\sin(2\pi f_o t)$ et de la fonction trigonométrique `sin()`.
2. **Implémentation récursive** d'un oscillateur sinusoïdal à l'aide d'un incrément pré-calculé permettant d'évaluer rapidement les composantes en phase et quadrature via un mélange linéaire.
3. **Synthèse par table d'ondes**, utilisant une forme d'onde périodique précalculée et une interpolation d'ordre zéro ou d'ordre un.

Pour chaque méthode, les étudiants sont encouragés à mesurer le temps de calcul nécessaire au remplissage du tampon audio circulaire et à analyser les compromis entre précision (distorsion), efficacité et flexibilité.

3.2.3. Filtres numériques

Pour les applications musicales, un filtre couramment utilisé est le filtre analogique à variables d'état (SVF) de second ordre, dont la structure est illustrée dans la figure 5. Ce filtre présente plusieurs avantages. Tout d'abord, il permet de passer facilement d'un filtre passe-bas à un filtre passe-haut ou passe-bande en sélectionnant simplement la sortie appropriée. Ensuite, il dispose de paramètres de réglage indépendants pour la fréquence de coupure (f_c) et le facteur de qualité (Q). Les étudiants du département GEII de l'IUT de Brest-Morlaix ont déjà étudié les caractéristiques de ce filtre lors des travaux pratiques d'électronique de deuxième année. Dans ce projet, nous proposons de convertir le SVF analogique en son équivalent numérique, en adoptant la version Cytomic du SVF. Cette mise en œuvre offre d'excellentes performances numériques tout en utilisant l'arithmétique en virgule flottante 32 bits, ce qui la rend particulièrement adaptée aux applications de traitement du signal numérique [6]. Ce type de filtre est d'ailleurs utilisé dans de nombreux logiciels audio professionnels (par ex. : Ableton Live). Dans le support de cours, le filtre est décrit sous la forme d'espace d'état. Les étudiants sont invités à analyser la réponse en fréquence théorique du filtre à l'aide de Python/SciPy, avant de procéder à son implémentation sur le microcontrôleur STM32.

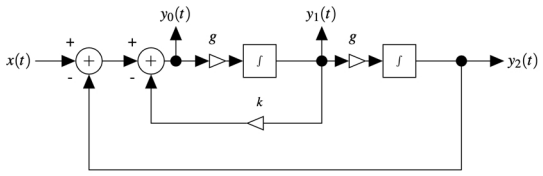

Figure 5. Structure d'un filtre SVF analogique. Un filtre passe-bas, passe-bande ou passe-haut s'obtient en sélectionnant respectivement la sortie $y_2(t)$, $y_1(t)$ ou $y_0(t)$.

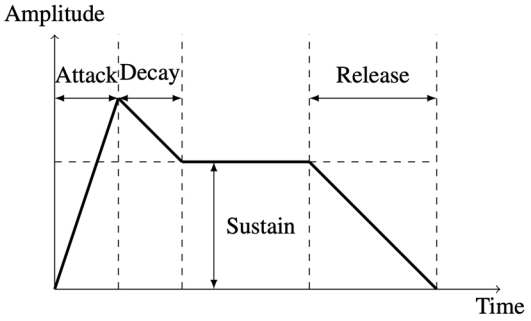

Figure 6. Enveloppe ADSR. La phase d'attaque est déclenchée par un message MIDI Note On, tandis que la phase de relâchement commence à la réception d'un message MIDI Note Off.

3.2.4. Enveloppe ADSR

La plupart des sons produits par les instruments de musique varient dans le temps. Pour reproduire ces variations, une approche classique consiste à utiliser des oscillateurs basse fréquences ou des enveloppes temporelles. Dans ce projet, les étudiants doivent d'abord implémenter une enveloppe ADSR (Attack, Decay, Sustain, Release). La structure d'une enveloppe ADSR est illustrée à la figure 6 et peut être implémentée à l'aide d'une machine à états finis. Lors de la modulation d'un paramètre, une problématique importante concerne la fréquence à laquelle le paramètre modulé doit être mis à jour. Bien qu'une mise à jour à la fréquence d'échantillonnage F_s puisse paraître naturelle, cette stratégie augmente fortement la

charge calculatoire. Dans ce contexte, les étudiants sont encouragés à expérimenter avec des fréquences de mise à jour plus faibles et à en évaluer l'impact sur la qualité du signal audio.

4. Commentaires des Étudiants et Impact Pédagogique

Une évaluation de l'enseignement a été menée au moyen d'un formulaire en ligne auprès de deux promotions (promotion 1 : 12 répondants, promotion 2 : 16 répondants). Le questionnaire comprenait deux questions ouvertes, l'une portait sur les points positifs et l'autre sur les aspects à améliorer.

Aspects positifs

Les étudiants ont particulièrement apprécié le caractère **interactif et engageant** du cours. Le **site web de support** a été régulièrement mentionné comme une ressource utile et moderne pour suivre et approfondir le projet. Le projet a également été jugé **innovant et motivant**, car il permettait de mettre en pratique des notions théoriques dans une application concrète et créative. Plusieurs retours ont aussi souligné l'impact technique du projet. L'un d'eux indiquait par exemple :*« J'ai plus appris en C dans ce module qu'en deux ans de cours d'info »*. Enfin, l'aspect amusant et ludique du projet a été relevé à plusieurs reprises.

Points à améliorer

Les retours négatifs ont été plus limités, mais certains étudiants ont exprimé des difficultés liées à la gestion du temps. Trois étudiants ont estimé que la durée totale du projet était insuffisante et qu'il serait bénéfique d'y consacrer davantage de séances. D'autres ont jugé certaines sessions trop longues (8 h d'affilée) et ont suggéré un format plus équilibré (séances de 4 h). Sur le plan technique, un étudiant a signalé des problèmes de configuration lors de la première session, ce qui a pu freiner la prise en main initiale. Quelques retours indiquent également que le contenu peut sembler complexe ou difficile à assimiler, notamment pour les étudiants moins expérimentés en programmation.

Dans l'ensemble, l'évaluation montre que le projet est jugé utile, motivant et innovant. La combinaison d'un sujet concret et créatif (synthétiseur), d'un support en ligne efficace et d'une mise en application progressive favorise un fort engagement des étudiants. Les principales pistes d'amélioration concernent la répartition temporelle des séances et un accompagnement renforcé pour faciliter l'assimilation de certaines notions.

5. Conclusion

Ce projet de réalisation d'un synthétiseur audio s'est révélé efficace pour impliquer les étudiants dans une démarche pluridisciplinaire, alliant électronique, programmation embarquée et traitement numérique du signal. En travaillant sur une application concrète, les étudiants approfondissent leur compréhension des concepts théoriques tout en acquérant une expérience pratique du traitement audio en temps réel. Les retours des étudiants mettent en évidence le fort engagement suscité par l'application musicale.

À l'avenir, ce projet pourrait être enrichi par l'intégration de nouvelles techniques de synthèse sonore, telles que la synthèse FM ou la modélisation physique. De plus, l'exploration des bibliothèques **CMSIS-DSP** constituerait une piste intéressante pour optimiser les performances et accélérer les opérations de traitement du signal.

Références

[1] Thomas A Baran, Richard G Baraniuk, Alan V Oppenheim, Paolo Prandoni, et Martin Vetterli. Mooc adventures in signal processing: Bringing DSP to the era of massive open online courses. *IEEE Signal Processing Magazine*, 33(4):62–83, 2016.

[2] Eric Bezzam, Adrien Hoffet, et Paolo Prandoni. Teaching practical dsp with off-the-shelf hardware and free software. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7660–7664. IEEE, 2019.

[3] Pierre-Marie Martin, Jean-Baptiste Fumey, Gabriel Pedenaud, Stéphane Girod, Philippe Abollivier, Florian Cassol, et Vincent Choqueuse. Mise en œuvre d'une maquette didactique en électronique sur un synthétiseur analogique, dans le cadre d'un projet tutoré de 1ère année de DUT GEII. CETSIS *2014, 2014*

[4] Ministère de l'Enseignement Supérieur, de la Recherche et de l'Innovation. Programme National du Bachelor Universitaire de Technologie en Génie Électrique et Informatique Industrielle.

[5] Kyle Andrew Sellers. An interactive software tool to assist DSP education of electrical and computer engineering undergraduates through audio. 2021.

[6] Andrew Simper. Solving the continuous SVF equations using trapezoidal integration and equivalent currents. *Cytomic, Retrieved from the Internet*, url : https://www.cytomic.com/files/dsp/SvfLinearTrapOptimised2.pdf.

[7] Nikolay Smagin et Lynda Chehami. Développement d'une interface utilisateur interactive pour l'enseignement du traitement du signal. *CETSIS 2023*, 2023

[8] Eddie Smigiel et Guillaume Chevereau. Enseigner la transformation de Fourier: une approche innovante. *CETSIS 2018*, 2018.

Biographie des auteurs

Vincent Choqueuse : Vincent Choqueuse est diplômé ingénieur de l'UTT (2004) et titulaire d'un master en statistique en optimisation des systèmes de l'UTT (2005). Il a obtenu son doctorat en 2008 à l'ENSTA sur « l'interception des systèmes de communications MIMO ». Il a été maître de conférences à l'IUT de Brest-Morlaix de 2009 à 2018, puis a rejoint l'École Nationale d'Ingénieurs de Brest (ENIB, Bretagne INP) en 2018. Ses recherches portent sur le traitement du signal et l'apprentissage automatique appliqués aux communications numériques, notamment optiques.

Vincent Kerhoas : Vincent Kerhoas est PRAG à l'École Nationale d'Ingénieurs de Brest (ENIB, Bretagne INP) depuis 2005. Diplomé de l'INSA de Lyon et titulaire de l'agrégation de Génie Electrique, ses activités d'enseignement sont consacrées aux systèmes embarqués (Electronique Numérique et Architecture des processeurs, Traitement du Signal, Objets Connectés, Systèmes Robotiques). Les ressources pédagogiques correspondantes sont disponibles sur : https://web.enib.fr/~kerhoas

Yann Le Masson : Ingénieur en développement logiciel chez Arturia. Il a contribué à des recherches sur l'implémentation du protocole MIDI 2.0 et au développement du logiciel *Analog Lab*. Il est diplômé de l'École Nationale d'Ingénieurs de Brest (ENIB, Bretagne INP).

Johny Kessler : Johny Kessler est ingénieur diplômé de l'École Nationale d'Ingénieurs de Brest (ENIB, Bretagne INP) en 2016. Après plusieurs années d'expérience dans la sécurité des systèmes embarqués chez Thales, il a rejoint en 2022 STMicroelectronics. Il travaille sur l'intégration d'un framework de sécurité et développe des exemples pratiques pour faciliter la prise en main de la technologie par les clients. Ses missions incluent la gestion de releases itératives, avec à chaque étape des améliorations progressives visant à optimiser les fonctionnalités et l'expérience utilisateur, contribuant ainsi à l'adoption efficace de solutions sécurisées dans des applications industrielles et grand public.

Marc Le Roy : Marc Le Roy est professeur dans l'équipe DH (Dispositifs Hyperfréquence : https://labsticc.fr/fr/annuaire/le-roy-marc (https://labsticc.fr/fr/annuaire/le-roy-marc) du pôle MatRF au Lab-STICC (UMR CNRS 6285) et enseigne au département GEII de l'IUT de Brest-Morlaix (UBO - Université de Brest). Ses activités de recherche portent sur la modélisation et la conception de topologies innovantes de composants passifs hyperfréquences tels que des filtres, des lignes non-uniformes de formes quelconques ou périodiques (EBG), déphaseurs analogiques et numériques, lignes à retard, antennes, diviseurs de puissance, baluns et aussi sur des composants actifs (amplificateurs, circuits à TPG négatif et non-Foster).