Numéros / n° 7-8 - Culture du code

Â«Â Pédale d?effets *open source* pour guitare Â Â»

Joseph Larralde

Résumé

Cet article décrit le processus de conception et de réalisation d?une pédale d?effets pour guitare électrique basée sur des technologies *open source* grand public, destinée aux musiciens et aux développeurs. Il expose tout d?abord le contexte ayant motivé la création d?un tel objet, puis détaille les choix techniques effectués lors de sa mise au point, ainsi que les architectures matérielles et logicielles développées. Les auteurs donnent enfin un bref retour de leur expérience avec le premier prototype réalisé.

Introduction

Les pédales d?effets pour guitares électriques constituent un écosystème riche qui ne cesse de se diversifier depuis les années soixante. Les technologies numériques de traitement du son ont largement contribué à cette diversification dès leur apparition, outre leur impact énorme sur l?évolution des synthétiseurs et autres types d?instruments électroniques. Cependant, pour un inventeur désirant créer ses propres instruments numériques, les moyens employés dans l?industrie restent difficiles d?accès (environnements propriétaires, cartes de programmation onéreuses, connaissances techniques requises complexes). Il faudra attendre l?apparition d?Arduino (Mellis *et al.*, 2007) pour que la notion d?*open source hardware* gagne en popularité et ouvre progressivement la voie à d?autres projets accessibles à un plus large public. Toutefois, le savoir en électronique nécessaire à la conception de pédales purement analogiques n?a pas souffert des mêmes limitations. Il s?est diffusé au fil des années au sein d?une communauté de passionnés échangeant schémas et points de vue sur des sites web et forums spécialisés tels que diyaudio (1), diystompboxes (2), tonepad (3) ou encore runoffgroove (4). Parallèlement, les artistes s?emparent très tôt de l?ordinateur personnel comme outil de création, favorisant l?émergence d?environnements logiciels avancés dédiés à la musique. Ainsi, naissent notamment des environnements de programmation visuelle tels que Max/MSP, Pure Data (Puckette, 1996) ou encore Reaktor, s?inspirant du principe de « patch » issu des premiers synthétiseurs analogiques et se situant à la frontière entre logiciel et langage informatique. Rapidement, des communautés se forment autour de ces outils, qui fournissent un accès direct à l?élaboration de logiciels musicaux temps-réel aux esprits créatifs voulant plonger dans le vif du sujet, sans perdre de temps avec les problématiques liées à la programmation logicielle bas niveau. Bien que certains de ces environnements ne soient pas *open source*, tous constituent des plateformes d?échange d?un savoir plus large, celui de l?audio numérique, et constituent des outils pédagogiques précieux, en plus d?offrir des possibilités de création musicale de qualité professionnelle. Aujourd?hui, la miniaturisation des composants électroniques a gommé les frontières entre systèmes embarqués et ordinateurs personnels, ouvrant le champ à de nouvelles expérimentations. Les auteurs, tous deux guitaristes électriques, ont ainsi eu le loisir d?imaginer une pédale d?effets programmable basée sur un système Linux, capable d?embarquer le fruit de plusieurs années d?expérience avec Pure Data et de concrétiser cette idée en s?appuyant uniquement sur des technologies *open source*.

1. État de l?art

L?idée de créer des instruments électroniques hautement personnalisables par leurs utilisateurs n?est pas vraiment récente : on peut citer en exemple le Nord Modular (5) dont la première version sort en 1997, distribué avec un logiciel d?édition avancé permettant de combiner de nombreux modules et de charger le

résultat en mémoire. Plus récemment, l?Axoloti [(6)] reprend ce principe et propose une carte électronique accompagnée d?un logiciel d?édition similaire à celui du Nord Modular, laissant le soin aux utilisateurs de concevoir le reste de leur instrument. D?autres produits, comme la pédale Owl (Webster, LeNost et Klang, 2014) de Rebel Technology [(7)] ou les cartes pedalSHIELD et Pedal Pi d?Electrosmash [(8)], visent directement un public de guitaristes et permettent de charger des programmes audio écrits en C++. Le fabricant MOD Devices (Ceccolini et Germani, 2013) va plus loin en basant ses pédales sur un système Linux embarqué et le format de *plugins open source* LV2 [(9)], permettant la combinaison d?un grand nombre de modules existants, tout en laissant la possibilité aux développeurs de créer de nouveaux *plugins*. MOD-UI, l?éditeur *open source* de MOD Devices distribué sous licence GPL-v3, se présente sous la forme d?une application web accessible *via* une connexion locale et permet d?importer des *plugins* LV2, de les connecter selon des graphes audio complexes et de sauvegarder ces différentes configurations. Un projet récent basé sur Linux et plutôt destiné aux *makers* est Zynthian [(10)], un kit constitué d?un boîtier, d?une carte Raspberry Pi [(11)] et d?un ensemble de composants *open source*. Il fonctionne par défaut avec MOD-UI, ce qui lui permet d?émuler les produits MOD Devices. Enfin, un projet similaire incontournable est Bela (McPherson, 2017), une famille de cartes d?extension pour les ordinateurs miniatures BeagleBone [(12)], annonçant des performances inégalées en termes de latence audio et de connectivité avec le monde physique. Également basés sur un système Linux et destinés aux *makers*, les produits Bela se programment grâce à un éditeur web accessible *via* une connexion locale, en C++ ou *via* le chargement de patchs Pure Data. Examinons à présent les possibilités d?utilisation de ces périphériques dans la chaîne typique de traitement électroacoustique d?une guitare électrique (voir la thèse de Benoît Navarret soutenue en 2013, aux pages 7 à 28, pour une description détaillée de cette chaîne). Seules les pédales Owl, Electrosmash et MOD Devices proposent une entrée instrument [(13)] et un boîtier de type *stompbox* [(14)] leur permettant de s?insérer directement dans un *pedalboard* [(15)]. Cependant, après écoute des démonstrations disponibles en ligne, seuls les produits MOD Devices semblent satisfaisants au niveau de la qualité sonore. Tous les autres systèmes cités précédemment ne présentent que des entrées et sorties audio au niveau ligne ; il n?est donc possible de les insérer que dans la boucle d?effets d?un amplificateur, c?est-à-dire après l?étage de préamplification. De plus, ils ne sont pas conçus pour être utilisés avec les pieds, ce qui limite leur utilisation pendant le jeu. Du point de vue du degré de personnalisation offert aux utilisateurs, le Nord Modular et l?Axoloti proposent des librairies de modules conséquentes mais non extensibles, limitant leur utilisation à des combinaisons de ces modules. Tous les autres systèmes permettent à un développeur d?embarquer ses propres algorithmes de traitement audio. Concernant ceux basés sur Linux, leurs documentations respectives ne font toutefois que mentionner un accès possible en SSH et ont plutôt tendance à encourager l?utilisation de leurs propres outils logiciels dédiés, occultant toute une partie du potentiel créatif offert par le système d?exploitation.

2. Formalisation du projet

Les auteurs souhaitent favoriser une appropriation plus profonde de leur projet par les développeurs familiers de Linux, tout en s?adressant à un public de guitaristes engagés dans une recherche de leur propre son à travers la constitution d?un *pedalboard* idéal. Pour atteindre ce but, et afin de garantir la pérennité du projet, ils ont formulé quatre principes guidant sa réalisation :

- Les technologies utilisées doivent être *open source*, bien documentées, avoir une large communauté d?utilisateurs et rester accessibles au plus grand nombre, autant au niveau financier que du point de vue des connaissances techniques requises ;
- La qualité sonore de la pédale doit être équivalente à celle de produits professionnels, c?est-à-dire utilisable en situation de concert par un musicien exigeant ;
- L?interface de contrôle physique doit être simple mais polyvalente, c?est-à-dire capable de s?adapter à une variété de scénarios d?utilisation et s?inspirer des pratiques courantes des constructeurs du marché ;
- Le produit doit proposer plusieurs niveaux d?appropriation par les développeurs en s?adressant aussi bien aux débutants en Pure Data qu?aux experts de Linux.

Les auteurs sont convaincus que le moyen le plus évident de pallier le problème de la qualité sonore est de concevoir un circuit analogique qui adapte correctement la tension et l?impédance des signaux d?entrée/sortie de la pédale au niveau ligne typique des entrées/sorties de la plupart des cartes son. De plus, la création d?un tel circuit faciliterait l?introduction dans la chaîne de traitement analogique de composants supplémentaires qui permettraient d?affiner la couleur [(16)] du son de la pédale. La carte pourrait servir de support à d?autres portions de circuit nécessaires à la réalisation du système, comme le

routage des contrôleurs physiques.

3. Réalisation

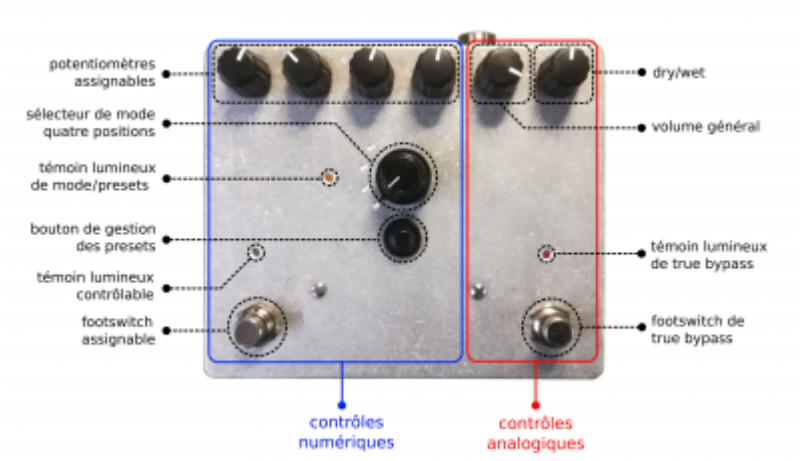
Partant de la simple idée d?une pédale d?effets programmable, la réalisation du premier prototype a consisté à imaginer une architecture matérielle, sélectionner ou créer ses différents modules électroniques, lui imaginer une interface de contrôle physique, puis à mettre au point un environnement de développement logiciel tirant parti de cette interface et enfin à programmer un ensemble d?effets à tester. Des essais ont été effectués par les auteurs sur deux configurations : une guitare type Les Paul avec un ampli à lampes Marshall et une guitare baryton type stratocaster avec un ampli à lampes « fait maison » et ce, en différents points d?un *pedalboard* constitué d?une réverbération TC electronics, une distortion Fuzz Factory, et une distortion plus « classique ». Ces tests ont été menés tout au long de la réalisation du prototype, faisant partie intégrante de son cycle de développement. La qualité sonore y a été appréciée à chaque fois de manière complètement subjective, en prenant soin toutefois de juger séparément la qualité esthétique des effets et la qualité sonore du système en lui-même.

3.1. Choix des composants

Suivant le premier des quatre principes énoncés précédemment, le choix du processeur audio s?est naturellement porté sur une carte Raspberry Pi. Le modèle exact retenu pour le projet est la Raspberry Pi Zero [(17)] ; ses avantages étant sa petite taille, son prix très abordable et sa faible consommation électrique. Pour les mêmes raisons, le microcontrôleur retenu pour gérer l?interface physique de contrôle des effets est une carte Arduino Nano [(18)]. Suivant le deuxième principe, les auteurs ont décidé d?équiper la Raspberry Pi d?une carte son Audio Injector Zero [(19)], sélectionnée parmi les autres modèles disponibles sur le marché pour sa taille, son coût relativement bas, son design minimaliste et sa qualité sonore. Les composants utilisés pour la réalisation du circuit audio analogique ont également été choisis pour leurs qualités sonores, sur la base de connaissances empiriques acquises avec l?expérience, notamment à travers la réalisation de « clones » de pédales célèbres.

3.2. Interface de contrôle

Figure 1. Description de l?interface de contrôle



Source : Joseph Larralde, 2020

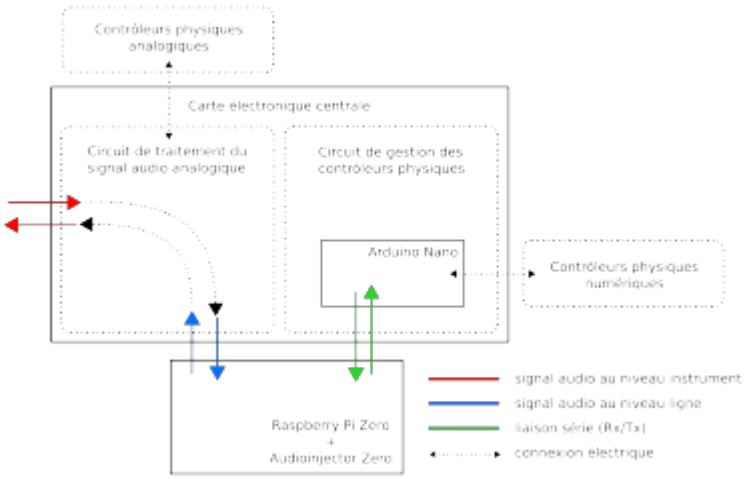
L?interface de contrôle (Figure 1) se veut simple et intuitive. L?absence totale d?écran est délibérée. Elle s?inspire de systèmes existants, en particulier de certaines pédales Electro-Harmonix [(20)] populaires. Une

partie de cette interface est dédiée au contrôle du signal analogique. Elle comprend un potentiomètre de *dry/wet* [21], un potentiomètre de contrôle du volume général, un *footswitch* [22] de *true bypass* [23] et un témoin lumineux de l?état du *true bypass*. L?autre partie est directement reliée à l?Arduino Nano et sert à contrôler la partie logicielle : quatre potentiomètres et un *footswitch* avec témoin lumineux sont assignables aux paramètres de contrôle des effets, un sélecteur rotatif quatre positions est dédié à la gestion de quatre « modes », un bouton est dédié à la gestion de *presets* [24], et un témoin lumineux sert à informer l?utilisateur sur l?état du système. Les auteurs portent également un intérêt particulier au contrôle gestuel des processus sonores. L?un d?entre eux a déjà mis au point une guitare électrique augmentée par des capteurs embarqués avec laquelle il a présenté une création [25] lors d?un concert du Scrime [26] en 2014. Les instruments augmentés constituent un champ de recherche passionnant : on pourra notamment se référer à la thèse d?Otso Lähdeoja (2010, p. 154-288) pour une étude approfondie du cas de la guitare électrique augmentée. Néanmoins, dans le cadre de ce projet, il a été jugé important de rester focalisé sur la qualité sonore et l?ergonomie de la pédale. C?est pourquoi, celle-ci a été sobrement dotée d?une prise jack trois points [27] sur l?un des côtés du boîtier, permettant de raccorder un capteur analogique externe assignable aux paramètres de contrôle des effets, au même titre que les quatre potentiomètres et le *footswitch* numériques. Il est par exemple possible d?y brancher une pédale d?expression, un capteur de distance, un capteur de pression, etc. Un seul degré de liberté est donc disponible pour le contrôle gestuel des effets, mais le choix du capteur (et donc du type de geste capté) est entièrement laissé à l?utilisateur d?autant que la plupart des capteurs compatibles avec Arduino peuvent être directement raccordés à cette entrée supplémentaire.

3.3. Architecture matérielle

La pédale est construite autour d?une carte électronique centrale embarquant tous les circuits nécessaires à son bon fonctionnement et communiquant avec la Raspberry Pi en audio et *via* une liaison série (Figure 2).

Figure 2. Schéma global de l?architecture matérielle



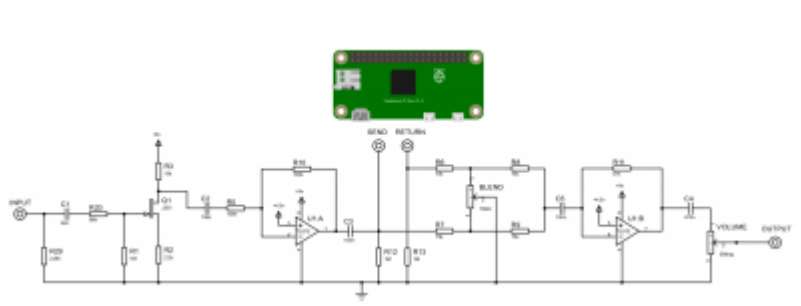
Source : Joseph Larralde, 2020

3.3.1. Description de la carte électronique centrale

Le carte électronique centrale comprend deux circuits alimentés indépendamment :

- un circuit pour la gestion de l?audio analogique (Figure 3) : la chai?ne de traitement du signal analogique est une compilation de sche?mas se?lectionne?s sur diffe?rents sites web de la « communaute? DIY », puis adapte?s pour le projet. Elle est compose?e d?un e?tage d?entre?e, d?un de?part de boucle d?effet, d?un circuit mixeur actif et d?un volume de sortie.
- un circuit pour la gestion de l?interface de contrôle numérique (Figure 4) embarquant l?Arduino Nano et se chargeant également d?alimenter la Raspberry Pi et d?adapter les tensions de la liaison série entre l?Arduino et la Raspberry Pi.

Figure 3. Circuit de traitement du signal audio analogique

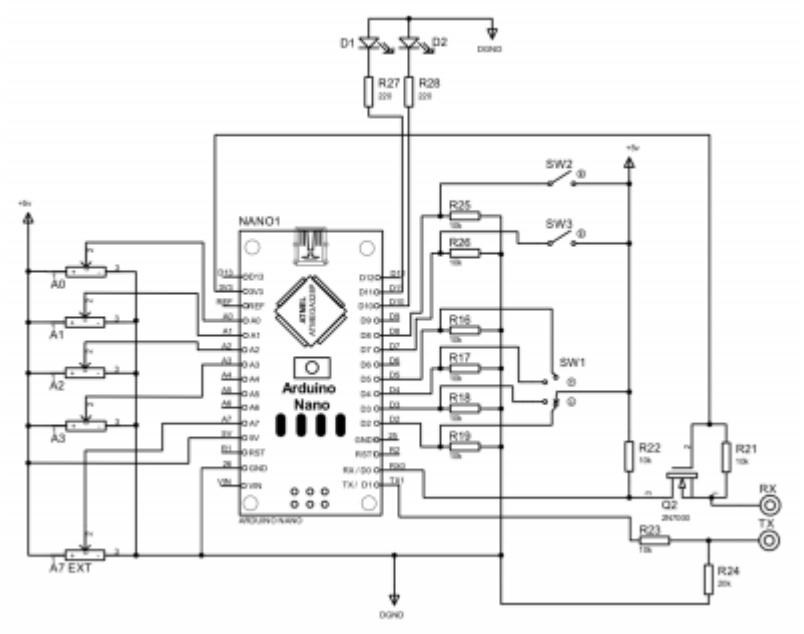


Source : Gabriel Larralde, 2020

3.3.2. Considérations sur l?architecture matérielle

Cette architecture constitue déjà en elle-même une plateforme offrant un premier niveau d?appropriation, c?est-à-dire un ordinateur sous Linux embarqué dans une pédale avec un accès direct aux entrées/sorties audio *via* le driver système et à une interface de contrôle physique versatile par le port série, accessible en SSH à travers une connexion réseau USB. À partir de cette plateforme, un développeur suffisamment expérimenté peut créer son propre système, en programmant éventuellement la carte Arduino pour la détourner de sa fonction initiale. L?architecture logicielle développée au cours du projet et présentée dans la partie suivante propose un deuxième niveau d?appropriation, ne nécessitant que la connaissance de Pure Data et le développement de patchs d?effets se conformant à une spécification déterminée par les auteurs.

Figure 4. Circuit de gestion de l?interface de contrôle numérique

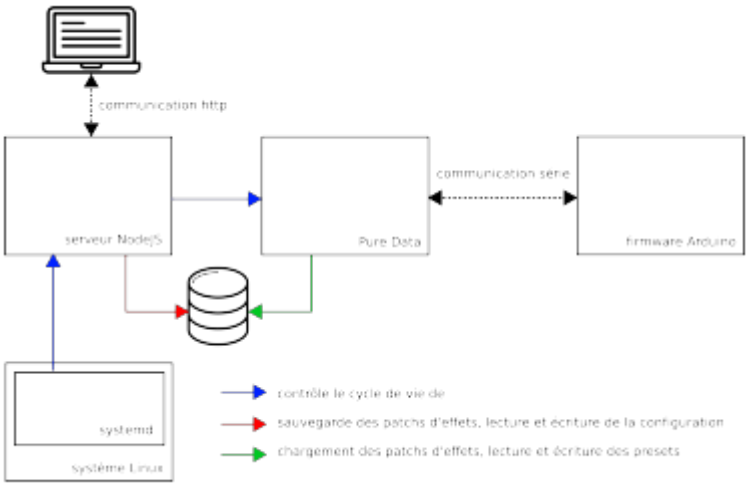


Source : Gabriel Larralde, 2020

3.4. Architecture logicielle

L?architecture logicielle consiste en un ensemble de scripts et de programmes communiquant entre eux. Un script système s?assure qu?un programme NodeJS se lance au démarrage et continue de s?exécuter tant que la pédale est sous tension. Ce programme NodeJS possède une interface web permettant de charger ou de supprimer des effets écrits en Pure Data, depuis un ordinateur connecté en USB, et de lancer l?application hôte pour les effets, également écrite en Pure Data. L?application hôte gère toute la partie audio et communique *via* des messages de haut niveau avec la carte Arduino, qui gère de manière autonome certaines fonctionnalités directement liées à l?interface physique.

Figure 5. Schéma global de l?architecture logicielle



Source : Joseph Larralde, 2020

3.4.1. *Firmware* Arduino

La carte Arduino est programmée avec un *firmware* dédié à la gestion de l?interface de contrôle et communique avec la Raspberry Pi *via* une liaison série (*pins* Rx et Tx). Le *firmware* présente une API simple : il envoie les valeurs des contrôles physiques, précédées d?un identifiant, dès qu?elles changent et accepte des messages de contrôle de la del [(28)] située au-dessus du *footswitch* gauche. Il contrôle également la del située à côté du sélecteur quatre positions de manière autonome, pour indiquer diverses informations liées à l?état courant du système, y compris pour notifier à l?utilisateur que la Raspberry Pi a fini de démarrer après la mise sous tension de la pédale.

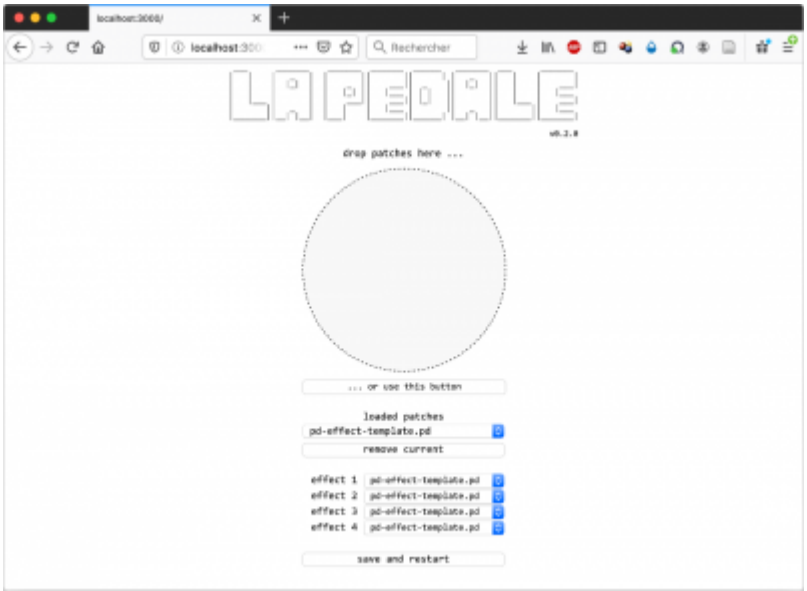
3.4.2. Système d?exploitation

Le système d?exploitation utilisé sur la Raspberry Pi est Raspbian Lite. Un *script shell* simplifie sa configuration initiale et l?installation de toutes les dépendances logicielles. Le script installe notamment Pure Data Vanilla et quelques *externals* [(29)], ainsi que NodeJS. Certaines étapes, comme la configuration de la carte son, doivent encore être effectuées manuellement et sont documentées dans le projet. Raspbian est utilisé dans sa version « de base », c?est-à-dire que le noyau Linux n?est pas recompilé pour optimiser les processus temps-réel, contrairement à la plupart des projets du même type. Le but, ici, est encore de garder le projet accessible au plus grand nombre en utilisant une version du système d?exploitation maintenue la plus standard possible.

3.4.3. Processus maître NodeJS

Le processus principal est un serveur NodeJS qui contrôle le cycle de vie de Pure Data. Il est lancé au démarrage par un service systemd et exécute à son tour Pure Data avec la dernière configuration sauvegardée. S?inspirant de l?état de l?art, il sert une page web (Figure 6) permettant la gestion d?une librairie de patchs d?effets Pure Data et leur affectation à quatre emplacements correspondant aux quatre positions du sélecteur rotatif central. Cette page web est accessible depuis tout périphérique connecté au réseau local par la prise USB.

Figure 6. Interface web de configuration de la pédale



Source : Joseph Larralde, 2020

Le *mapping* des contrôleurs physique est entièrement défini dans les patchs d?effets. Dès que l?utilisateur clique sur le bouton « save and restart », le serveur NodeJS sauvegarde les éventuels changements d?affectation de patchs aux quatre emplacements, copie les patchs correspondants dans un dossier spécifique en les renommant respectivement « effect-1.pd », « effect-2.pd », etc., puis il relance

Pure Data. Cette nouvelle configuration sera utilisée tant qu?elle n?est pas à nouveau modifiée.

3.4.4. Fonctionnement du patch Pure Data hôte

Un patch principal fait office d?hôte pour les quatre patchs d?effets sélectionnés depuis l?interface web. Il gère aussi un certain nombre de tâches, comme la communication série avec l?Arduino, la sauvegarde et le rappel du *preset* associé à chaque effet (ici, un *preset* est défini comme une configuration particulière des positions des potentiomètres numériques), l?activation de l?effet correspondant au mode sélectionné (les autres effets sont alors complètement désactivés pour économiser les ressources) et le routage des données de contrôle physique vers l?effet actif. Les patchs d?effets chargés depuis l?interface web doivent satisfaire certaines contraintes pour pouvoir fonctionner correctement au sein de l?hôte. Ils doivent comporter trois entrées : une entrée pour l?audio, une entrée pour les messages de contrôle issus de la liaison série et une entrée servant à notifier un changement d?état (actif ou inactif), permettant ainsi de gérer des problématiques spécifiques aux processus mis en ?uvre, comme vider la mémoire d?une ligne à retard dans un effet de *delay*, lorsque celui-ci est désactivé. Ils doivent également comporter deux sorties : une sortie pour l?audio et une sortie pour les messages de contrôle transmis à la liaison série. Un patch d?exemple minimal est inclus dans le projet et récapitule ces prérequis.

4. Retour d?expérience

La finalisation du premier prototype étant très récente au moment de la rédaction de cet article, les auteurs l?ont seulement testé eux-mêmes en l?insérant en divers points d?un *pedalboard* existant. Les quatre effets chargés dans la pédale lors des tests étaient un *delay* granulaire, un *bounce delay* (un *delay multi-tap* permettant de générer des rythmiques évoquant le rebondissement d?une balle), un effet de destruction numérique du son (dégradation de la quantification et de la fréquence d?échantillonnage du signal) et un enregistreur granulaire (enregistreur audio permettant de rejouer le son de manière non linéaire par synthèse granulaire). Les auteurs ont jugé le résultat très satisfaisant au niveau de la qualité sonore : le système semble transparent, le caractère numérique du son non discernable et une brillance subtile est apportée sur le canal *wet* par l?utilisation d?un transistor JFET. Ils ont également trouvé l?utilisation du système fluide. Les points faibles identifiés sont le temps de démarrage du système d?exploitation (environ 30 secondes avant que la pédale ne soit fonctionnelle), une légère latence audio perceptible par le musicien (fixe, de l?ordre d?une vingtaine de millisecondes, correspondant probablement aux latences d?entrée et de sortie cumulées du système) et la faible puissance du processeur limitant les types d?algorithmes de traitement utilisables (30).

Conclusion et perspectives

Les auteurs ont présenté une pédale d?effets réalisée uniquement à partir de technologies *open source*, constituant un compromis entre une qualité d?expérience et une accessibilité technique à un large public, sans toutefois faire de concession sur la qualité sonore. Le point faible le plus contraignant constaté est actuellement la latence, celle-ci impactant directement l?expérience du jeu musical. Dans plusieurs projets similaires, elle est généralement minimisée par l?utilisation d?un noyau Linux temps-réel. Certains de ces projets sont notamment basés sur la distribution Xenomai (31) dédiée aux applications temps-réel. Il était tentant d?utiliser un système Bela, lui aussi basé sur Xenomai et mettant en ?uvre des optimisations matérielles supplémentaires, mais son coût étant plus élevé et sa communauté plus restreinte, les performances de base du couple Raspberry Pi/Audio Injector Zero ont été jugées acceptables pour le développement d?un premier prototype. De plus, le *dry*/*wet* analogique permet de contourner le problème de la latence dans un grand nombre de cas et aucune optimisation du système spécifique à l?audio n?a encore été tentée avant de considérer l?utilisation d?un noyau temps-réel. Les auteurs envisagent donc pour l?instant de poursuivre le développement de cette version et commencent à étudier d?autres choix technologiques qui leur permettraient de conserver la même dynamique de travail et de rester fidèle aux quatre principes déterminés dans la formalisation du projet.

---

1. https://www.diyaudio.com/ [consulté le 18/09/2020]

2. https://www.diystompboxes.com [consulté le 18/09/2020]

3. http://www.tonepad.com/ [consulté le 18/09/2020]

4. http://www.runoffgroove.com/ [consulté le 18/09/2020]

5. https://www.nordkeyboards.com/products/nord-modular [consulté le 18/09/2020]

6. http://www.axoloti.com [consulté le 18/09/2020]

7. https://www.rebeltech.org/product/owl-pedal/ [consulté le 18/09/2020]

8. https://www.electrosmash.com [consulté le 18/09/2020]

9. https://lv2plug.in/ [consulté le 18/09/2020]

10. https://zynthian.org/ [consulté le 18/09/2020]

11. https://www.raspberrypi.org/ [consulté le 18/09/2020]

12. https://beagleboard.org/bone [consulté le 18/09/2020]

13. entrée conçue pour brancher un instrument dont le son est produit par un micro électromagnétique

14. boîtier conçu pour être utilisé avec les pieds

15. support facilement transportable, destiné à accueillir un ensemble de pédales de type *stompbox*

16. identité sonore, déterminée perceptivement par une certaine enveloppe spectrale et éventuellement par une légère distorsion du signal

17. https://www.raspberrypi.org/products/raspberry-pi-zero/ [consulté le 18/09/2020]

18. https://store.arduino.cc/arduino-nano [consulté le 18/09/2020]

19. http://www.audioinjector.net/rpi-zero [consulté le 18/09/2020]

20. https://www.ehx.com/ [consulté le 18/09/2020]

21. mixage du son brut de l?instrument et du son passé à travers l?effet

22. interrupteur à pied

23. configuration du circuit dans laquelle le son ne fait que passer à travers la pédale sans passer par le circuit numérique

24. configurations particulières des paramètres des effets pouvant être sauvegardés et rappelés

25. https://vimeo.com/102871277 [consulté le 18/09/2020]

26. https://scrime.u-bordeaux.fr/ [consulté le 18/09/2020]

27. prise permettant de raccorder un câble composé de trois fils généralement utilisée pour transmettre un signal stéréo ou symétrique, utilisée ici pour recevoir le signal d?un capteur analogique (alimentation, signal, masse)

28. diode électroluminescente, plus communément appelée led (de l?anglais *light-emitting diode*)

29. objets développés par la communauté non inclus dans la distribution officielle

30. par exemple, un effet de réverbération basé sur un algorithme de convolution partitionnée demande trop de ressources

31. https://xenomai.org [consulté le 18/09/2020]

Pour citer ce document:
Joseph Larralde, Â« Pédale d?effets *open source* pour guitare Â», *RFIM* [En ligne], Numéros, n° 7-8 - Culture du code, Mis Ã jour le 21/12/2020
URL: http://revues.mshparisnord.org/rfim/index.php?id=586