Reprogrammable Effects Pedals on the Daisy Seed Platform

First Author
First Affiliation
first@email.com

Second Author
Second Affiliation
second@email.com

ABSTRACT

Many creative audio effects are available only as plug-ins to digital audio work stations, or as boutique hardware devices. We describe the implementation of three realtime digital effects algorithms – a strobe tuner, spectral processor (with denoising & phase vocoder frequency estimation), and realtime granular synthesizer – designed to run on an inexpensive open-source hardware platform powered by the ElectroSmith Daisy Seed microcontroller. The platform has the form factor of a guitar stompbox; we discuss the merits of portable, reprogrammable hardware audio processing solutions for live performance, and future directions for this project.

1. INTRODUCTION

Since the 1970s, digital audio processing has become a significant part of music production & performance [1]. Digital audio environments provide canvases on which to superimpose, rearrange, and effect recorded sounds. These effects may be subtle – mixing, compression, equalization, reverberation – or significant alterations – distortion, delay, pitch-shifting, to name a few [2]. Digital audio effects are used both "offline" (in non-realtime settings like composition & production) and "online," for live performance.

Many consumer-grade computers & soundcards cannot guarantee "realtime" processing (sub-10 ms latency) without audio dropouts [3, Chapter 4]. A high-performance laptop with an external audio interface can attain round-trip latencies around 3 ms (assuming input and output buffer sizes of 64 samples at 48 kHz), assuming no latency is introduced by the audio interface or its communication protocol with the host. By comparison, an embedded digital effects unit can, in theory, handle much smaller audio buffers and thus deliver lower latencies. Projects such as Bela [4], Pisound [5], and Norns [6] (and its open-source cousin Shield) implement realtime Linux kernels for single board computers. These hardware solutions offer the flexibility of a DAW with the portability and immediacy of a live instrument. These products require various amounts of assembly and configuration, with Norns requiring the least (it also being the only one that ships with a dedicated enclosure).

This paper describes the implementation of three DSP effects on top of an inexpensive open-source hardware platform, which has the latency profile and form factor of a dedicated effects stompbox with the reconfigurability of a DAW (we measured the roundtrip latency at under two milliseconds). The motivation of the project was to provide open-source versions of processing methods typically found only in higher-end boutique pedals: an accurate strobe tuner, spectral processor, and granular synthesizer. The total materials cost per unit, $140, is comparable to the cost of a standalone strobe tuner pedal, and much less than comparable hardware for granular synthesis.

Copyright: ©2024 First Author et al. This is an open-access article distributed under the terms of the Creative Commons Attribution License 3.0 Unported, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Figure 1. The three effects discussed, running on the Daisy Seed pedal hardware: from left to right, granular processor, spectral processor, and strobe tuner.

2. THE DAISY ECOSYSTEM & PEDAL HARDWARE CAPABILITIES

The Daisy Seed is a microcontroller for audio applications [7]. It is powered by an STM32H7 chip (clocked at 480 MHz), with integrated high-quality audio codec and onboard persistent storage. The manufacturer, ElectroSmith, provides an open-source library for interfacing with and configuring the hardware [8].

Firmware is compiled on a computer & sent over micro-USB to the Seed's 128 KB flash memory; the Seed has about an additional megabyte of high-speed onboard RAM, 64 MB of SDRAM, and 8 MB of persistent QSPI storage. ElectroSmith and its community have also developed tools to port DSP code written in Max Gen~, Faust, and Pure-Data to the Seed.

Our code runs on Keith Shepherd's open-source 125b-size guitar pedal hardware for the Seed [9], which provides a rotary encoder, six potentiometers, two momentary footswitches, a 128×64 OLED, stereo audio i/o, and eighth-inch TRS MIDI i/o.

3. THE ALGORITHMS

The hardware repository [9] includes firmware for certain idiomatic guitar DSP effects, such as reverb, distortion, chorus & tremolo. The algorithms we chose to implement,

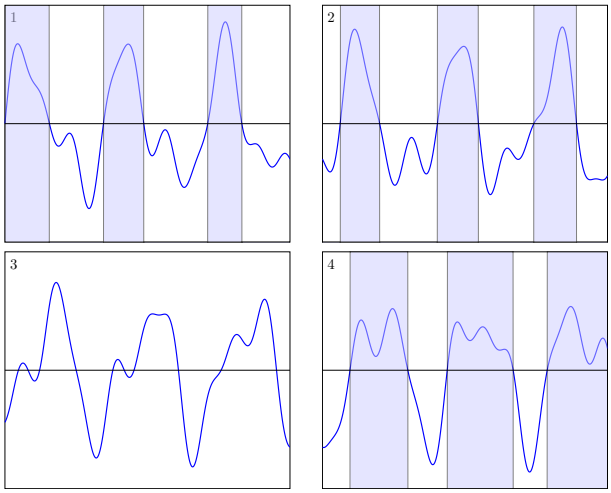

Figure 2. Four frames of the strobe tuner (numbered chronologically), displaying the waveform and barcode pattern. The input signal is an inharmonic sum of sinusoids with frequencies $0.96 \cdot 3f$, $1.85 \cdot 3f$ and $3.11 \cdot 3f$, and with amplitudes 1, 0.5, and 0.25. The overall shape of the barcode phases to the right since the fundamental is slightly flat of a harmonic of f, even though one of the high partials is sharp.

described in this section, are meant to demonstrate the platform's diverse capabilities for time and frequency-domain tasks. All three projects require the firmware to store & manipulate reasonably large buffers of audio; the strobe tuner and granular synth also make extensive use of the screen.

All code and supplemental photo & video documentation can be found in the linked Git repository: [10].

3.1 Strobe Tuner

The first algorithm we describe is a simple but accurate strobe tuner. The principle is as follows: we are given an input digital waveform $(n \mapsto x(n))$ and a reference frequency to compare our signal to, f. We set up a phasor $(n \mapsto p(n))$ whose value moves linearly through the interval $[0, 1]$ at frequency f. Given $w = 128$, the width of the screen in pixels, whenever n is such that $w \cdot p(n)$ crosses an integer, we modify a vertical line of pixels at horizontal coordinate $\lfloor w \cdot p(n) \rfloor$, turning the pixels on if $x(n) > 0$ and off otherwise.

If x is harmonic with frequency $f + \delta$, this will produce a barcode-like pattern (capturing the zero-crossings of the signal x) which phases in one direction or the other depending on the sign of δ. If, as in the case of plucked strings, the signal is not harmonic, the barcode represents the phasing of partials relative to fundamental; the fundamental can still be tuned to the reference pitch by eye, as demonstrated in Figure 2, or the user may choose to tune a higher partial to the reference frequency instead.

For simplicity of implementation, our algorithm displays six simultaneous barcodes with reference frequencies tuned to the open strings of a guitar (relative to $A\,440$); in another mode, the reference pitch can be selected from the chromatic scale using the encoder. In future work, tuning will be possible relative to frequencies other than $A\,440$. We also plan to add pitch-tracking so that the tuner can operate in "chromatic mode," comparing the input signal to the nearest note in a chosen reference scale – this will enable the user to tune to scales other than 12 EDO.

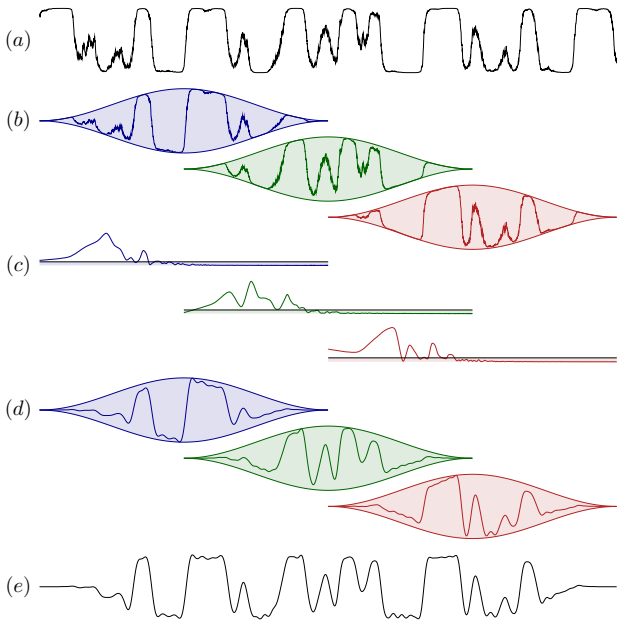

Figure 3. Stages of spectral processing. The input signal (a) is split into overlapping time-domain windows (b), which are then Fourier-transformed (c). In the denoising example we implement, all frequency content with amplitude below the thresholds (represented by shaded rectangles in (c)) is removed. The resulting spectra are inverse-Fourier-transformed, yielding the time-domain signals in (d), which are recombined to yield the output signal (e). Notice the removal of the noise from the original signal, but the global similarity in the shapes of the two waveforms.

3.2 Spectral Processing

The second algorithm is a spectral processor [11]; we define a `Fourier` class which handles windowing and the scheduling of forward- and reverse-FFT methods: an instance of the class is initialized with a callback defining the manipulation to take place in the frequency domain; audio fed to the instance is written to a number of circular buffers (one for each overlapping analysis window). When an input buffer fills up, the class invokes a forward FFT, then the user-defined callback, and then the inverse FFT, writing and windowing the resulting values into a number of overlapping resynthesis buffers, which are summed and returned to the Daisy's audio callback.

In our tests, we succeeded in running FFTs on the Daisy hardware of sizes up to $N = 4096$ with four overlaps or up to $N = 2048$ with eight overlaps. We implement two callbacks: one that denoises the signal by independently mixing the low- and high-amplitude bins, and a second which accurately measures the frequencies present in a signal using the phase vocoder algorithm [11] with bin-to-bin smoothing. A graphical representation of the denoising callback is outlined in Figure 3.

The denoiser separates the harmonic from the wideband elements of a sound; when in the configuration that zeros all frequencies with low amplitude, it smooths out transients. The sounds produced are especially interesting when the input is nearly (but not exactly) harmonic, as when processing, for example, a distorted dyad or triad. The distortion adds harmonics of the pitches present, as well as of

common undertones. The spectral processor removes the wideband elements, often leaving behind crisp & distinct high pitches & chirps. These sounds are demonstrated in the video linked in the codebase: [10].

In future work, we plan to implement an accurate polyphonic pitch-tracking algorithm so that the pedal can be used as an online audio-to-MIDI device. We also plan to implement the resynthesis portion of the phase vocoder, for cross-synthesis and frequency-domain pitch shifting [11].

3.3 Granular Synthesizer

The third algorithm we implement is a granular synthesizer. In the taxonomy of Roads' book [12, Chapter 5], our algorithm is "granulation in real time": audio is written into a fifteen-second circular buffer, from which grains are read. Four independently addressable metronomes request playback of grains; each metronome has its own parameters for the size, density, transposition and direction of playback of grains, as well as control over randomization of these parameters (and of grain predelay). For example, the density parameter determines how frequently the metronome asks for grains, while the spray parameter randomizes the regularity at which these grains are triggered (while maintaining the requested density, on average). The Seed can handle about forty voices of grain polyphony without audio dropouts.

The parameters of the granulation are stored to the Daisy Seed's persistent memory; for difference choices of parameters, the granular synth can be configured as a transposition effect, chorus, reverse delay, or can turn input sounds into textures. These configurations are demonstrated in the video attached to the software repository.

In future work, we plan to include an option to use the effect statically, without constantly overwriting the source audio, to implement "freeze" capabilities, and to make the granulation sensitive to acoustic features of the input signal (like pitch & rhythm).

4. FUTURE WORK

Besides the algorithms we've presented, the Seed guitar pedal could be used for a number of practical musical settings. In future work we plan to implement firmware that performs live beat tracking & pitch detection for coordination of other audio equipment.

The Seed has enough computational power to evaluate moderately-sized neural networks: a community project for Daisy Seed has emulated a number of guitar amplifiers & distortion pedals [13]. In further work we also plan to port machine learning algorithms, trained on a computer, to the Seed. For example, the pedal could run an algorithm which would perform auditory feature extraction and neural audio synthesis, as in IRCAM's RAVE project [14].

5. CONCLUSIONS

We have detailed the implementation of three real-time digital effects algorithms – a strobe tuner, spectral processor, and granular synthesizer – for the ElectroSmith Daisy Seed. The aim of this work is to publish open-source software versions of audio effects that usually are available only either as software plugins for DAWs, or inside specialized or boutique effects hardware. We hope also to generate interest in the further development of effects and synthesizers for the Daisy Seed guitar pedal platform.

6. REFERENCES

[1] J. Watkinson, *The Art of Digital Audio*. Focal Press, 2001. [Online]. Available: https://books.google.com/books?id=eVpITJfPxMEC

[2] U. Zölzer, *DAFX: Digital Audio Effects*, 2nd ed. Wiley Publishing, 2011.

[3] Y. Wang, "Low latency audio processing," Ph.D. dissertation, Queen Mary University of London, 2018.

[4] A. McPherson, "An environment for submillisecond-latency audio and sensor processing on BeagleBone Black," in *Audio Engineering Convention 138*, 2015. [Online]. Available: http://www.eecs.qmul.ac.uk/~andrewm/mcpherson_aes2015.pdf

[5] Blockas. Pisound – Raspberry Pi Sound Card & MIDI Interface. [Online]. Available: http://blokas.io/pisound/

[6] Monome. Norns. [Online]. Available: http://monome.org/docs/norns/

[7] ElectroSmith. Daisy Seed. [Online]. Available: http://electro-smith.com/products/daisy-seed

[8] ——. libDaisy: Hardware Library for the Daisy Audio Platform. GitHub. [Online]. Available: http://github.com/electro-smith/libDaisy/

[9] K. Shepherd. (2023) DaisySeedProjects. GitHub. [Online]. Available: http://github.com/bkshepherd/DaisySeedProjects/

[10] F. Author. (2024) GitFront. [Online]. Available: http://gitfront.io/r/alo-bu/jzfyZttkVw63/main/

[11] J. O. Smith, *Spectral Audio Signal Processing*, 2011. [Online]. Available: http://ccrma.stanford.edu/~jos/sasp/

[12] C. Roads, *Microsound*. MIT Press, 2004.

[13] K. Bloemer. (2023) NeuralSeed. GitHub. [Online]. Available: http://github.com/GuitarML/NeuralSeed

[14] A. Caillon and P. Esling, "Rave: A variational autoencoder for fast and high-quality neural audio synthesis," 2021. [Online]. Available: http://arxiv.org/abs/2111.05011