Jared Talon Holton, Rauly Baggett

Dr. Kepuska

ECE 3551

11 December 2021

Digital Guitar Effects Pedal

For this project, we intend on implementing 4 different guitar pedal effects—Fade, Delay, Reverb, Distortion in software on a raspberry pi.

To begin modifying audio with the raspberry pi, we decided on using the PyAudio library in Python. This library helps simplify opening and writing to the input and output devices as streams. The data is read from the single channel analog input with a specified frame size, usually between 1024 and 4096 samples. Each sample of the input is a single 16-bit integer. After the effects are applied to the inputted bytes, they are duplicated to create a two-channel output that can be immediately written to the output stream. This will introduce a fixed delay, as the entire frame size must be filled before the frame can be modified and written to the output.

The first effect used is Fade. Fade is a simple effect of moving the input audio from one channel to the other. In this project, the input is only 1 channel, so the input can be differentially broadcast to either or both output channels. Doing so creates a fade effect, allowing different volumes in both thew left and right channels.

The second and third effects, Delay and Reverb, were created using a ring buffer implementation of an IIR filter. To produce the effects in python, we created a buffer with a large, fixed size and a pointer set to 0. Iterating over the samples read in, each sample is added together with the value stored at the pointer location (initially 0). The value in the buffer is then set to the dry value of the sample. This creates a simple single delay. The reverb effect can be produced by setting the value of the buffer to not just the input sample, but the sample plus the value reduced by some coefficient less than 1. The pointer is then incremented by 1 and set to the remainder of itself divided by the reverb duration (modulo). Altogether, this produces the simple reverb IIR filter:

.

The Distortion effect is created using signal clipping. Clip distortion is the process of making the input waveform squarer, which adds higher frequency elements to the sound. A simple way to perform this, was to overdrive the input signal and clip the values over a threshold. In our implementation, the volume levels were normalized to between -1 and 1, so the overdrive can be applied simply by scaling the input by some factor greater than 1 and clipping the output. This leads to more steep sinewaves with flat tops, with the sharp corners of the wave in traducing the high frequency components.

When attempting to run our code on the raspberry pi, we discovered that the pi’s clock speed was inefficient at reading, modifying, and writing the buffer in a time frame that creates clear audio output. The output graphing functionality also had to be set aside to prioritize the primary objective off the project, which is the audio buffer manipulation. We then decided to attempt writing the code in C with purely audio manipulation to see if we could make the code efficient enough to be handled by pi.

Microcomputer Systems was a class heavily focused on signal manipulation and buffer interpretation and processing. This being the case, it was very fitting to choose this idea for the final project, as well as implement a little real-life application using an electric guitar. We began by writing a python script using libraries: pyaudio, scipy, matplotlib, and numpy. In the testing environment this code ran beautifully, but when applying it to the pi we discovered the lower clock speed issue which made the sound very choppy. It was then decided to attempt to rewrite the code in C with less visual aid and less complexity in order to reduce the execution time. Obviously, sound quality was sacrificed for the necessary speed, but the pi was able to handle the C version of the code.

Works Cited:

[1] Hubert Pham, 2006, “PyAudio v0.2.11: Python Bindings for PortAudio,” <https://pypi.org/project/PyAudio/>

[2] Port Audio Community, “PortAudio Documentation,” <http://portaudio.com/docs.html>

Code repo: <https://github.com/TheUnity42/MicroFinalProject>