University of Michigan

College of Engineering

Gesturally Controlled Guitar Pedal for Real-Time Effects

EECS 452: Team 1

Nick West, Hypatia Magyar, Leah Kim, Eric Cooley, Guthrie Tabios

Introduction

A common addition to playing the electric guitar is an effect pedal, which is a small device activated with a footswitch and modified with knobs that changes some aspect of the guitar's sound. Pedals usually use a hybrid of solid state electronics and digital signal processing to modify the electrical guitar signal transmitted through an instrument cable. The limiting factor of these pedals is that their settings are not easily modified and that it is hard to switch effects. The guitarist cannot physically adjust the knobs while playing their instrument or unplug the current effect to switch it out. Our project is a guitar effect suite that frees performers from this constraint by using gestures to change the effect's parameters and by incorporating multiple effects onto one unit. To adjust the effect, the performer could tilt their guitar or change an easy to access slider on the body of the guitar. To change effects completely, the performer would press a push button, also on the body of the guitar. We implemented three effects for this project: tremolo, delay, and wah. This new and easy way to change effects while playing opens new avenues in performance art, where expressive gestures made by performers literally affect the music they are producing.

A comparable product to our project that is already on the market is a line selector. This is a pedal that has several inputs, and you can control which inputs get fed into the output. If you connect other pedals to the line selector, you can easily change between different effects. This product does not have the full functionality and convenience of our idea. Firstly, the line selector requires the player to have other effect pedals already, and all the pedals take up space and can be complicated to set up. Our product is one compact unit with 3 built in effects. It doesn't require additional purchases. Secondly, although the line selector makes choosing effects easy, changing the parameters of those effects would still require the musician to take their hands off the guitar to turn a knob.

Design Constraints and System Overview

Some important constraints we had to make this a good product were size, ease of use, signal integrity, and the subjective sound quality. The product needed to be small enough in area and weight to be attached to the body of an electric guitar. It also needed to not get in the way of the guitarist's normal playing, and the buttons and slider needed to be in a convenient position. The unit needed to not distort or delay the guitar signal unintentionally, and the effects needed to sound good to a listener or to the performer.

Most of our system's components, besides the power source, are contained in one 3D printed box. As shown in Figure 1, an instrument cable feeds into the printed box and another cable connects the output of the system to a speaker or amp. The box is located at a place where the guitar has available flat space to adhere the box to, is out of the way of the guitarist's strumming, and is easy to reach.

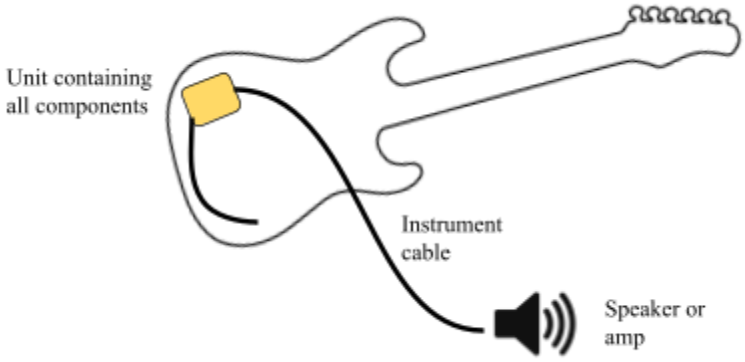

Figure 1: Diagram of effect pedal placing on the front of an electric guitar

The signal path starts with an input signal from the guitar, which goes through an ADC and our sound effect processing, and then into a DAC and amplification. We chose to implement this with five primary components: a Teensy audio shield, Teensy microcontroller, inertial measurement unit (IMU), slide potentiometer, and push buttons. Our materials list with links is detailed in the BOM in the design folder. As shown in Figure 2, the guitar, IMU, slide potentiometer, and buttons input into the Teensy and Audio Shield, which outputs to the amplifier.

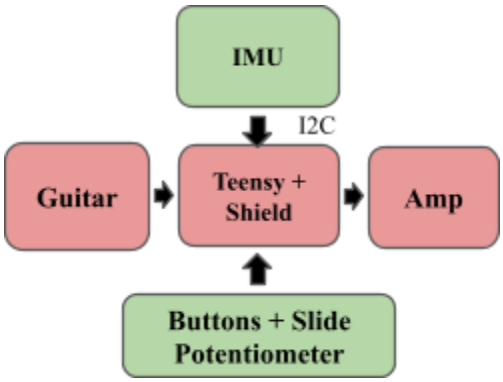

Figure 2: Block diagram of system

Hardware Implementation

As one of our main constraints was size, we wanted the hardware of our system to be as small as possible. While we initially planned on ordering a PCB, many of the parts of our system, specifically the IMU, shield, and Teensy, already had their own PCBs. Therefore we decided it was more efficient in effort and cost to make a protoboard instead. Our final protoboard, pictured in Figure 3, routes together all the components of our system. The only component outside this unit is the power supply, as the method for powering the system was variable. To power the system for our expo, where outlets weren't necessarily

close by, we used a battery adhered with velcro to the back of the guitar. During testing and in our vision for the product, we used a long power cable going to a wall outlet.

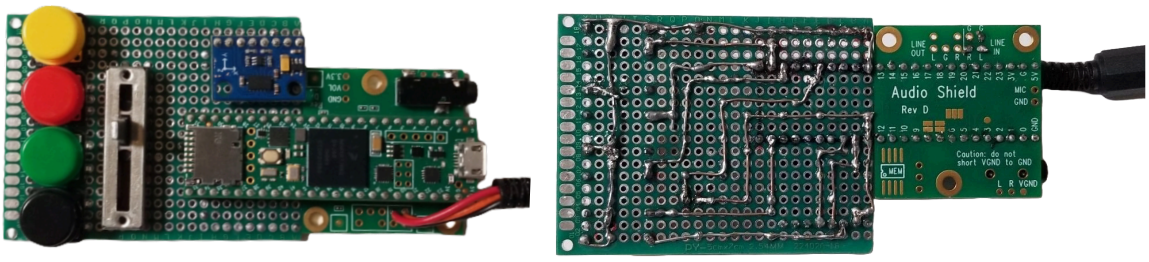

Figure 3: Image of the front and back of our final protoboard, which houses all system components

To house the board, we designed a box, shown in Figure 4, with a small, slide-on lid that covers the processing units and leaves the buttons and slider exposed. The box is small, 11 x 6.4 cm, and has holes in the back of the box for the aux and power connections.

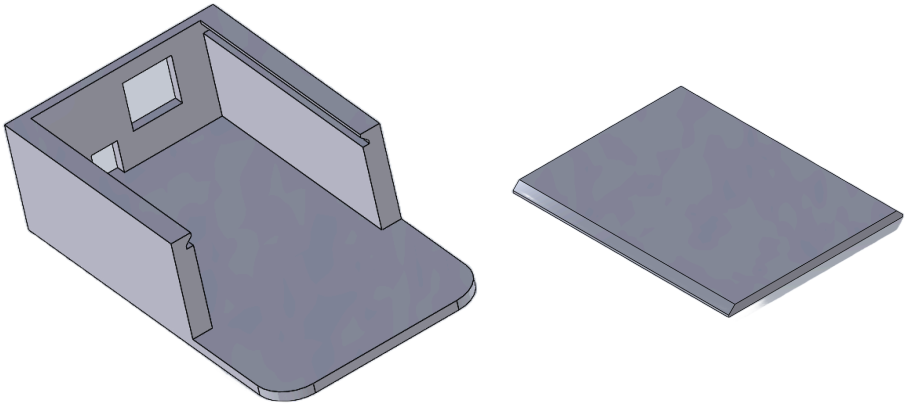

Figure 4: Image of box design with lid

Digital Signal Processing

The three main effects: tremolo, delay, and wah, were each implemented through specific digital signal processing techniques, making use of the Teensy Audio library, which samples at 44100 Hz. Each effect is changed by data from the IMU, slide potentiometer, and buttons. The IMU, sampled at a rate of 30 Hz, reads the degrees of tilt in the Z direction. The potentiometer, sampled at the same rate, reads a value from 0 to 100 after ADC and mapping. This information is accessible by the effects, and used to modify its parameters. The buttons are triggered with interrupts and activate an effect change. Figure 5 shows which

button corresponds to which effect, and what parameters of that effect are changed by the tilt and potentiometer respectively.

Figure 5: Diagram of button interface with corresponding effect and effect's changeable qualities

The tremolo effect is a modulation of volume. To do this, we applied a sine wave multiplier to each audio block sample. The frequency of the sine wave multiplier, which affects the rate of the tremolo, is determined by the value received from the IMU, and the amplitude, which affects the depth, is from the slide potentiometer value.

The delay effect plays the original signal along with a series of delayed signals getting quieter and quieter. We implemented this through multiple delay blocks with different gains, as shown in Figure 6. In this effect, the IMU values change the time step length of the delay and the potentiometer values change the number of delay blocks. We use a circular buffer to efficiently access the data from our audio block over multiple passes, which is updated by the audio library every 128 samples.

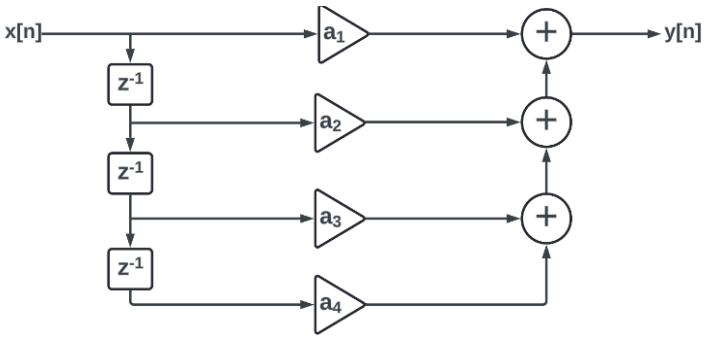

Figure 6: Block diagram of delay implementation

The wah effect changes the timbre of the guitar sound to mimic a "wah" sound. We did this by using a peak filter to highlight a specific frequency. The IMU values change the center frequency of the Wah effect, and the potentiometer values change the gain. We chose to use an infinite impulse biquad filter shown in Figure 7 for our implementation. Our code works by calculating the coefficients of this filter according to already designed equations every time the IMU and potentiometer are sampled. These coefficients are put into the filter transfer function, which uses the current sample and two past samples to calculate the output.

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

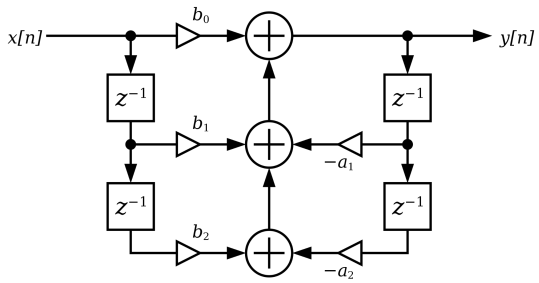

Figure 7: Transfer function and block diagram of Wah effect filter

Problems Faced and Debugging

When building this project we came across some technical issues, which we will detail for replication purposes. One of those was clipping, which affected almost all of our effects. Because most of our effects involve adding to our sound signal in a variable length loop, the numbers in our output would get very high under certain parameters. This caused a signal that was too loud for the system to handle, resulting in unpleasant sounds. We dealt with this by lowering the general output levels.

We encountered a couple other smaller problems, including math errors in our buffer. This was simply an off by one error when indexing through the buffer, but it resulted in indexing out of bounds and creating unpleasant noises. The other problem was the calculation of Wah filter parameters conflicting with scheduled variable updates. Because the filter application is called in the update function of the AudioStream object, it's automatically called on a pre scheduled timer. However, the coefficient calculation is not on the same schedule. This means that the parameters of the filter could change while the filter was being applied, resulting in an output that had a wrong filter. This was solved by disabling interrupts during the IMU and potentiometer value update, as well as during the coefficients calculation.

Testing

To test that our IMU and potentiometer readings were accurate, didn't add delay, and worked at their extremities, we tested them thoroughly. We utilized Python to receive information from the Teensy, parse, and graph it. Figure 8 shows an example of an accelerometer and potentiometer data test. This test shows that the readings we get are accurate and the delay is applied as expected.

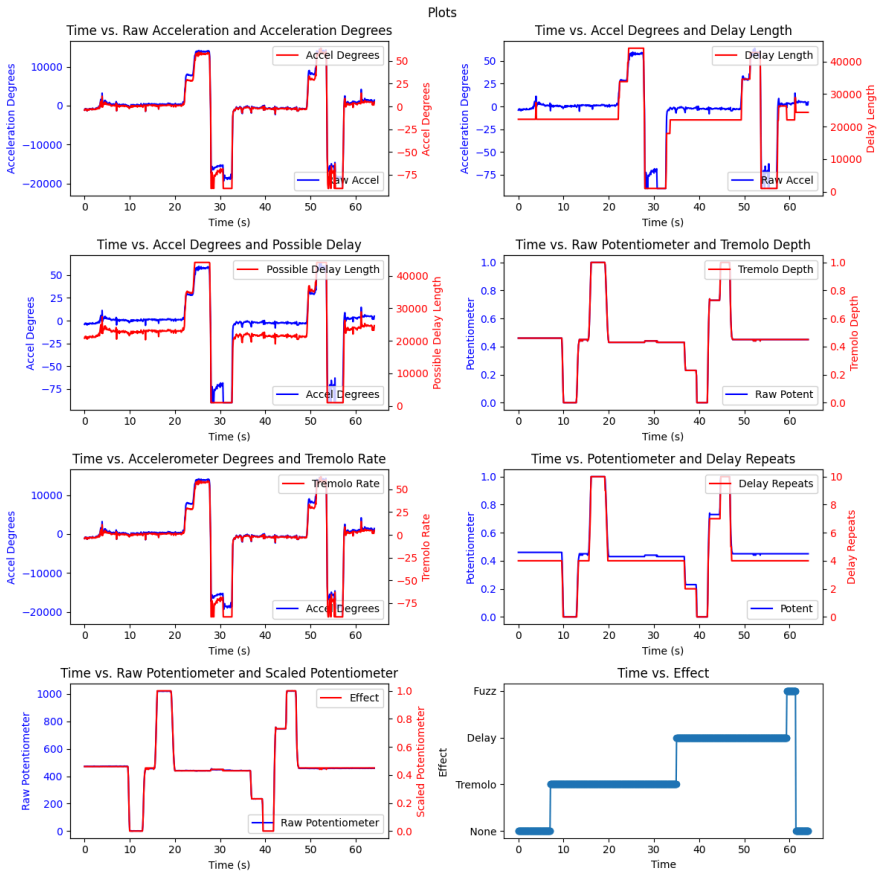

Figure 8: Accelerometer and slide potentiometer data captured over time

Maintaining signal integrity and producing something that sounded good was a very important part of the project. We ensured this with several different methods of testing. First, we tested the system to determine optimal parameter ranges for the different effects. By observing the behavior of the audio output in a spectrogram, we could precisely gauge the effect of parameter adjustments on sound characteristics such as frequency distribution, amplitude modulation, and temporal dynamics. Additionally, subjective listening tests were conducted to evaluate perceptual aspects such as clarity, depth, and overall tonal balance. Utilizing a combination of spectrograms and subjective evaluation, we meticulously assessed the impact of varying tremolo rates (ranging from 1 Hz to 15 Hz) and depths (0 to 0.5), delay repetitions (2 to 10), time step lengths (ranging from 100 to 44100 samples), and wah gain (-30 to 30 dB) and center frequency (500 to 5000 Hz). This comprehensive testing ensured that our system reliably delivered high-quality audio output across a wide range of parameter configurations. One of our spectrogram tests, showing the frequency impact of the delay effect, is below in Figure 9.

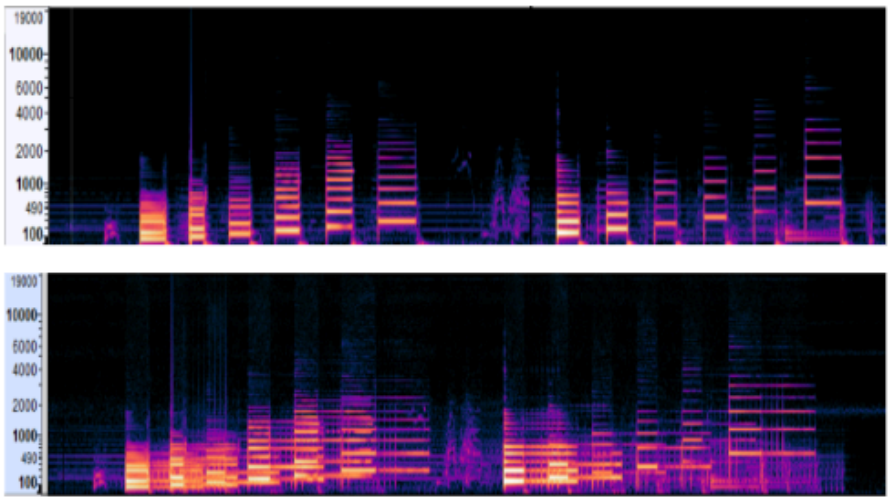

Figure 9: Spectrogram of single guitar notes played with no effect (top) and delay effect (bottom)

To test the effects without having to have the physical system, we also made a Python simulation. The code mimics the Teensy's processing on inputs of MP3 files, complete with a graphical interface. This made it possible and easy to test from afar and refine the system. The simulation is shown in Figure 10

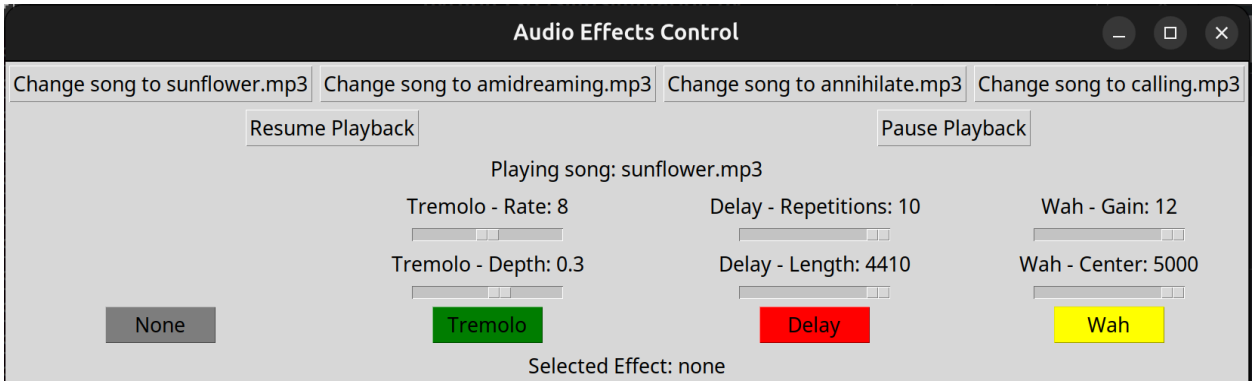

Figure 10: Simulation used to mimic the physical system.

Future Work

Moving forward, the path to enhancing the gesturally controlled guitar pedal involves several considerations. Firstly, expanding the range of effects to include additional options such as reverb, chorus, and distortion could amplify possibilities for musicians. Simultaneously, refining the implementation of existing effects, particularly the wah, by exploring methods to reduce the required angle for activation, would enhance usability and accessibility during live performances. Furthermore, the development of a prototype without reliance on an audio shield could streamline the design, potentially reducing costs and

simplifying the manufacturing process. Addressing packaging challenges to create a more compact and visually appealing unit, with minimized exposed wires and components, would improve aesthetics and user experience. By pursuing these avenues, future iterations of the gesturally controlled guitar pedal can achieve greater functionality, and usability, positioning it as a tool for musicians seeking innovative ways to shape their creative landscapes.

-