



Department of Electrical and Computer Engineering Technology (ECET)

Division of Engineering, Computer Programming, and Technology (ECPT)

EET 4910

Senior Design Proposal

Smart Electric Guitar (SEG)

Submitted by

Jose Carlo Doliner and Christopher Collins

Supervised by

Dr. Masood Ejaz

November 11, 2023

Abstract

The concept of this project is an electric guitar that utilizes a touchscreen interface that is built into it to serve as a pedal board. This will allow the user to select from various effects on the touchscreen within arm's reach to modify their sound. The touchscreen will be connected to a microcontroller within the guitar that will process various effects. The microcontroller will take the sound input through an ADC, then process the signal for the required effect that the user will select through a touchscreen, and then pass it to the output through a DAC. The user will be able to select from four various effects including distortion, delay, reverb, and chorus.

Effects are used to shape the sound of an instrument, where the sound is dependent on the genre. This sound is achieved using pedals which are small circuit boxes that include analog circuits to reshape the original instrument waveform that is activated by a footswitch. Each pedal contains its effect and therefore creates a large device. Currently, guitar users must carry a separate system when traveling to live performances which makes portability difficult. The user must find various outlets to plug in the equipment which forces them to be limited in where they can play. Having the pedal board integrated into the guitar allows the user to break free from that restraint and play virtually anywhere. The microcontroller and touchscreen will utilize a power bank that can allow it to run for a minimum of four hours before having to plug it in.

The touchscreen Graphical User Interface commands and microcontroller code will work hand in hand in creating effects to the signal before being passed to the output speaker. This allows the user to have effects that can be created with low latency and become extremely customizable. Users can create customized effect banks using presets that allow them to change the sound of the guitar with only a press of a button.

Acknowledgments

We would like to particularly appreciate the support and guidance from our supervisor Dr. Masood Ejaz, program leader of the BSECET program. Notably, we would like to appreciate the support given to us with his understanding and knowledge of digital signal processing as well as his care and attention in the success of the project and our personal development as future engineers.

We would also like to acknowledge the feedback from Neil Bishop and Jason Adams, previous BSECET graduates also passionate about music. They provided initial feedback and suggestions that helped shape the concept of the project. Special acknowledgement is also given to Peter Van Der Sande, Professor of Audio Technology Recording at Valencia College East Campus, who also supported the project by providing initial considerations to be taken before tackling the project itself.

Table of Contents

Cover	<i>i</i>
Abstract	<i>ii</i>
Acknowledgments	<i>iii</i>
Table of Contents	<i>iv</i>
List of Figures	<i>vi</i>
List of Tables	<i>vii</i>
Chapter 1 Introduction	1
1.1 Introduction	2
1.2 Definitions	3
1.2.1 The Distortion Effect	3
1.2.2 Delay Effect	4
1.2.3 Reverb Effect (Echo)	4
1.2.4 Chorus Effect (Modulation)	4
1.2.5 Further Terminology	4
1.3 Hardware Considerations	5
1.4 Potential Solution Approaches to Achieve Digital Effects	6
1.4.1 Analog Effect Circuit Simulation Using MATLAB	6
1.4.2 Creating a Proprietary DSP Interface	6
1.4.3 Using VST effects, our Preferred Approach	7
Chapter 2 Proposed Work	8
2.1 Technical Design and Approach	9
2.1.1 Hardware Design	9
2.1.2 Software Design	10
2.1.3 System High-Level Overview	11
2.2 Engineering Requirements	13
2.2.1 Effects Requirements	14
2.3 Engineering Specifications	14
2.3.1 Processing Module	16
2.3.2 Input/Output Module	17

2.3.3 Electronic Module	17
2.3.4 Power Module	18
2.3.5 Audio Output Module	18
2.4 Proposed Components and Power Budget	18
2.5 Success Criteria	21
2.5.1 Digital Effects Success Criteria	21
2.5.2 Graphical User Interface Success Criteria	23
2.6 Prototype Testing	23
2.6.1 Distortion Effect Prototype Testing	24
2.6.2 GUI Testing	26
2.6.3 Hardware Integration Attempts	28
2.6.4 Guitar Integration Difficulties	30
Chapter 3 Non-Technical Issues	32
3.1 Budget	33
3.2 Proposed Timeline	34
3.2.1 Fall	34
3.2.2 Spring	35
3.3 Issues with Ergonomics	39
3.4 Ethical Concerns	39
Chapter 4 Conclusion	40
4.1 Summary and Conclusion	41
4.2 Suggestions for Future Work	42
References	43
Appendix A: Distortion Effect Code Samples	45
A.1 C++ Code for Audio Processing Block	46
A.2 C++ Code for Parameter Building	47
Appendix B: Graphical User Interface Code Samples	48
B.1 Python Code for Graphical User Interface	49
Group Members	51

List of Figures

Figure 1.1	Project Concept	2
Figure 1.2	3D Rendered Image of the Proposed Project	3
Figure 1.3	Virtual Studio Technology Logo	7
Figure 2.1	Hardware Block Diagram	9
Figure 2.2	Software Block Diagram	11
Figure 2.3	Modular Block Diagram	12
Figure 2.4	CAD Rendering of the SEG (Inner Components Exposed)	20
Figure 2.5	CAD Rendering of SEG Components. (Bottom Side View)	21
Figure 2.6	Arctan Distortion Function	24
Figure 2.7	Prototype UI for the Distortion Effect	25
Figure 2.8	GUI Test Stage 1	26
Figure 2.9	GUI Test Stage 2	27
Figure 2.10	GUI Test Stage 3	27
Figure 2.11	Raspberry Pi 4B and Connections Testing	28
Figure 2.12	Touchscreen PCB for Connection Testing	29
Figure 2.13	Raspberry Pi and Touchscreen Integration	29
Figure 2.14	Raspberry Pi Connections Top Visual	30
Figure 2.15	5” HDMI/USB touchscreen	38

List of Tables

Table 2.1	Engineering Requirements	13
Table 2.2	Digital Effects	14
Table 2.3	Engineering Specifications	15
Table 2.4	Power Budget	19
Table 2.5	Digital Effect Waveforms	22
Table 3.1	Projected Budget	33
Table 3.2	Proposed Timeline Fall Semester	35
Table 3.3	Proposed Timeline Spring Semester	36
Table 3.4	Fall GANTT Chart	37
Table 3.5	Spring GANTT Chart	38

Chapter 1

Introduction

Summary

In this chapter, an in-depth description of the SEG (*Smart Electric Guitar*) will be given. Here we will define guitar effects, effect pedals, as well as the key characteristics and terminology to understand the purpose and design approach of the project.

1.1 Introduction

1.2 Definitions

1.3 Hardware Considerations

1.4 Potential Solution Approaches

1.1 Introduction

When listening to music, either at home or at a live performance, there is an extremely high chance of hearing effects being introduced that have drastically changed the audio you hear. In the music industry, effects (either analog or digital) are utilized to shape the sound of an instrument. Guitarists have the basic requirement of having pedalboards for their live shows and recording sessions. These pedalboards consist of sets of circuit boxes called pedals that modify the audio signal from the guitar to produce their desired sound characteristics. These pedals can range between \$100-\$200 each, posing first a monetary constraint (a set of 4 pedals including their respective cables could range between \$500-\$1000 depending on brand and quality), along with the fact that the musician has to carry this equipment around and likely will have to set up its arrangement for every live performance.

Luckily, there is a way to improve this system to allow for multiple effects to be refactored into a single piece of equipment. This is the primary concept for SEG (*Smart Electric Guitar*), a project that proposes to solve these issues by creating a device built into the guitar itself that will process the audio signal from the guitar using a digital backend for live processing and produce the modified audio signal.

In the guitar world, there is a product similar to this, called Digital Multi-Effect Pedals, which by contrast, creates digital effects on a device that includes several switches controlled by foot. The difference between this and the SEG is that the latter has the advantage of being included in the guitar itself, which makes the system both portable and convenient for the guitarist as it means that only one piece of equipment is needed, that being the instrument itself.

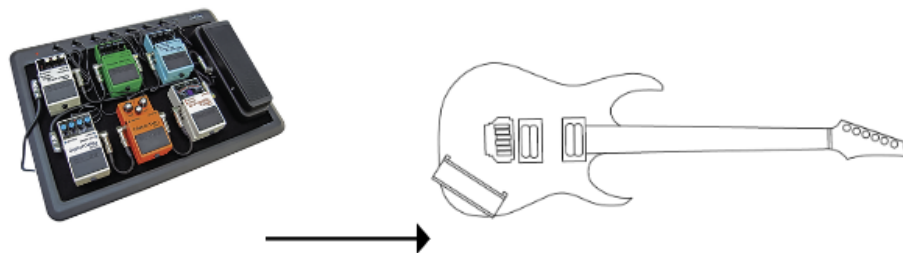


Figure 1.1 Project Concept

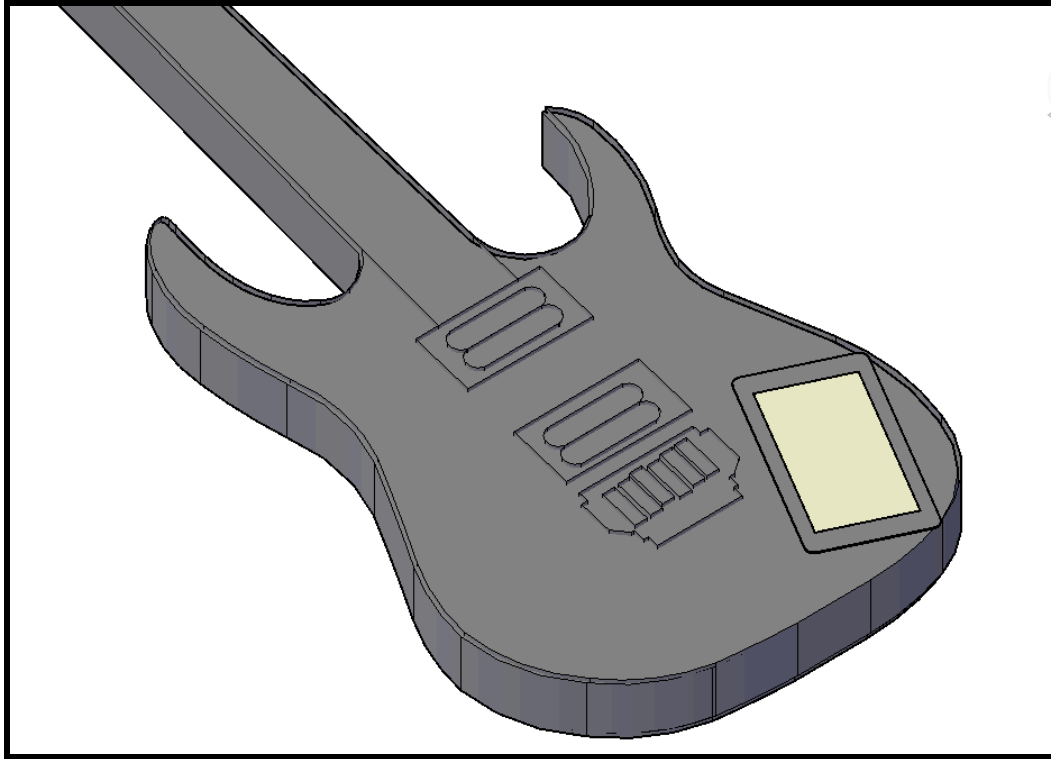


Figure 1.2. 3D Rendered Image of the Proposed Project.

1.2 Definitions

Let us start by defining digital effects and kinds that are desired for implementation into the SEG. As mentioned earlier, audio effects are present in every live performance you encounter, and they modify the native sound of the instrument. For the guitar, there are thousands of effects with multiple types. For this project, we will focus on developing four standard effects that will come pre-installed into the device itself, being: **Distortion, Delay, Reverb, and Chorus**. The following link will take you to a video demonstration of how the various effects sound:

https://youtu.be/taxik_WI4h8

1.2.1 The Distortion Effect

The distortion effect modifies the original waveform by performing artificial clipping. Specifically, this effect seeks to recreate the effect created by a diode clamping circuit or clipping effects created by vacuum tube amplifiers with too much gain. This effect creates a harsh, more aggressive tone distinctive to many genres such as Rock and Blues.

1.2.2 Delay Effect

The delay effect takes the original sound and plays it back at a fixed rate, where the repetitions eventually fade in amplitude. It is called delay because you can hear a repetition of the notes sometime after they are played.

1.2.3 Reverb Effect (Echo)

This effect simulates the echo and natural decay of sound as it dissipates through a room. This effect's focus is to add space to the sound and provide a more open tone to it. You can think of this effect as if you were standing in an empty room, where your voice constantly bounces back from the walls, therefore having your voice fade over time at a slower rate. This effect is quite common and present in most music genres in some form.

1.2.4 Chorus Effect (Modulation)

The chorus effect seeks to emulate the effect when multiple voices and subtle pitch differences occur when multiple instruments or voices play the same notes at the same time. The term chorus refers to a choir where multiple voices sing the same notes but have distinctive differences in their tones and sounds. On a more objective note, this effect takes a sample of the original sound and doubles it with a preset pitch distinction.

1.2.5 Further Terminology

Across this report we are going to refer to several aspects of guitars and audio processing that are unfamiliar to the average person, therefore we are going to supply more context to these in this section.

For starters let us introduce the device that allows us to get the sound out of an electric guitar. **Magnetic pickups** are devices present in every electric guitar. They take the role of converting the natural sound of the string vibrations into an electrical signal. The output from these pickups is then hardwired into knobs that usually control the volume (reduce signal amplitude) and a simple RC high-pass filter to reduce noise.

Another set of terms used throughout this report is **ADC/DAC** which stands for analog-to-digital and digital-to-analog converters. As their names suggest they are used to convert an analog signal to a discrete binary signal and vice versa. These are used in everyday components to convert voltage values that have an infinite range into a discrete set of values for processing purposes. When an audio signal is processed through an ADC into a microprocessor, the computer will take **samples** of the amplitude of the signal at a rate called the **sample rate**. This sample rate is usually a specification found on ADCs and it is fundamental for audio processing as it dictates how fast we can read any analog audio input passed to the system. This term is accompanied by the **buffer size** which is the number of samples the computer takes to process an audio signal. For audio processing, we would want the highest sample rate with the smallest buffer size. These two terms combined will determine the **latency** of the system, which is the amount of time it takes for an input to be processed and sent to the output. In simple terms and for our purposes, latency is the time it takes for an audio signal to come out of our speakers when we hit the strings of the guitar.

Lastly, **DSP** or Digital Signal Processing is a set of techniques utilized to manage and modify our samples in a way that produces the desired audio effect at hand.

1.3 Hardware Considerations

For the hardware elements of the project, size constraints play a key role in the design choices taken for the project as well as the specifications, due to the proposed approach for dealing with hardware components is to build the electronic components inside the guitar, while exposing the touchscreen as an outside interface. Thankfully, with the use of microprocessors, there is a straightforward approach to achieve this if the guitar is modified to fit the components inside.

There are two ways to interface with a guitar's natural sound. One of them is to create a system independent of the guitar itself that could be detached and attached to the guitar, whose input is the direct output of the guitar itself. This device would then have an output that should resemble the output of the guitar. This approach has the limitation that it would reduce the portability of the system. Every aspect of the design revolves around packaging the system in an enclosure that is small enough to not interfere with the player's ability to perform.

Due to this limitation, our desired approach is to create a device built into the guitar itself. The sound from the guitar is directly gathered by hardwiring the magnetic pickups to an ADC. It converts the analog signal from the guitar into a signal that is processed by the operating system of the processing device. Having this packaged in this arrangement allows for the touchscreen to be the only exposed component of the system, both reducing risks of component failure and allowing for a better weight distribution of the device from the guitar. A more in-depth description of the hardware components is provided in Chapter 2.

1.4 Potential Solution Approaches to Achieve Digital Effects

There are several ways in which digital audio effects can be created as well as ways to integrate them inside of the guitar itself. Section 1.4 will focus on highlighting the creation of digital audio effects and how the SEG will compare against them.

1.4.1 Analog Effect Circuit Simulation Using MATLAB

One approach frequently used to emulate pedal effects is to use MATLAB. This is because MATLAB allows the simulation of analog circuits and transfer functions to create the desired effects [9]. Although simple in theory, over a longer, more expansive project such as this one becomes increasingly complicated, especially due to the issue of high latency and interfacing restrictions with MATLAB. Since this project focuses on a real-time application, the presence of latency poses a substantial problem.

1.4.2 Creating a proprietary DSP interface

Multi-Effect Pedals (described in Section 1.1) are created using a Digital Signal Processor (DSP). These, although powerful and fast, come with the main disadvantage of being complex to work on and difficult to interface with using a touchscreen, especially for a project as ambitious as this one. Although possible to implement, it simply falls outside of the time constraints for the project. Another drawback of creating a proprietary DSP interface is that the user would be restricted to just being able to use the standard four default effects when in theory we could allow

the user to import other effects available on the internet by using a more general solution approach.

1.4.3 Using VST effects, our Preferred Approach.

Our proposed way to achieve this is by creating customized VST Plugins (Virtual Studio Technology) that will allow the user to modify their sound from the guitar itself. The VST format is a widely used software format (developed by Steinberg) for DSP sound effect creation, used in the industry to create digital effects and their respective interfaces for parameter customization. The powerful aspect of VSTs is that they are developed to easily interface with the Operating system's audio devices and create audio samples.



Figure 1.3 Virtual Studio Workstation Logo

Chapter 2

Proposed Work

Summary

In this chapter, we present a detailed explanation and analysis of our proposed project. This includes technical design and approach, engineering requirements and specifications, proposed components and power budget, and the testing performed.

- 2.1 Technical Design and Approach**
- 2.2 Engineering Requirements**
- 2.3 Engineering Specifications**
- 2.4 Proposed Components and Power Budget**
- 2.5 Success Criteria**
- 2.6 Prototype Testing**

2.1 Technical Design and Approach

In this section we will provide a general overview of the technical components proposed to create a guitar digital interface, controlled by a touchscreen, which is capable of handling live effects in arms reach.

2.1.1 Hardware Design

As mentioned briefly in Chapter 1, a technical concern regarding the creation of the SEG system is the size constraint. We seek to eliminate/mitigate these constraints by building a part of the system into the guitar, exposing only the necessary aspects needed by the user on the outside.

The native signal from the guitar pickups will be connected to a bandpass filter that will be controlled by two knobs, one further knob will be implemented to control the volume of the system. The signal coming out of the filter/volume control knobs will be amplified and then passed to an ADC. The ADC will convert the analog electrical signal from the pickups to a digital signal to be interfaced with a microprocessor. This microprocessor serves the purpose of handling the effects by using DSP software developed to create the desired effects. A touchscreen will communicate with the microprocessor to control the parameters of the effects. The modified signal from the microprocessor will be then passed to a DAC that will then be amplified and passed to the output. This output will be at a level that can be safely passed to a standard guitar amplifier. The following block diagram displays the flow of the hardware components:

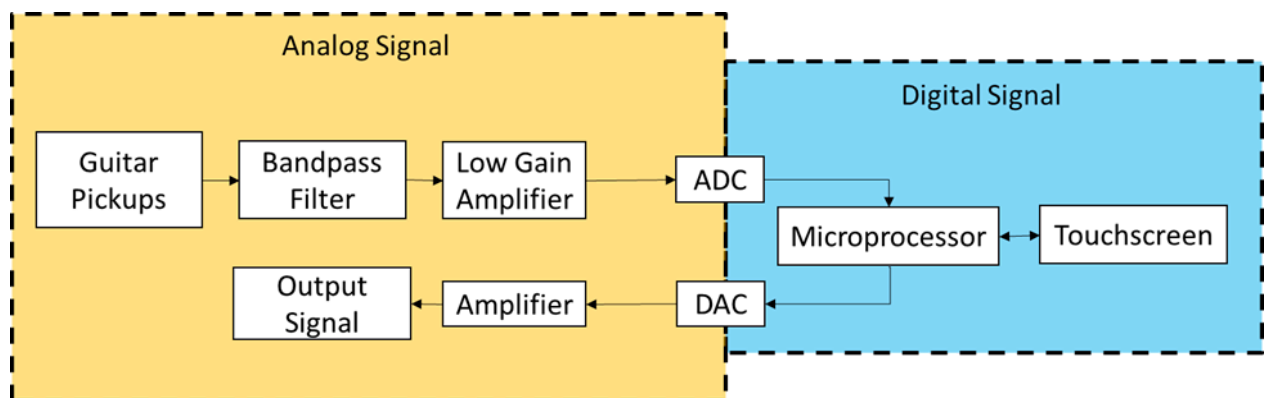


Figure 2.1. Hardware Block Diagram

2.1.2 Software Design

For this project, several considerations need to be made about the software design, as mentioned in Chapter 1. The digital effects will be created by designing VST software packages tailored to the application and format appropriate for a touchscreen display. To make full use of the VST format, a user interface (UI) needs to be developed to interface and manipulate the state and presence of these effects. Even though this poses a great challenge, it also produces a fantastic opportunity to increase accessibility and customization for the user, since it means that the user can import their own VST Plugin into the program, instead of having to rely solely on the effects that we built for the SEG.

Sound Effects are entities that are connected in series. This means that if an effect that changes the signal in a certain way is then passed as the input of a different effect, the latter will modify the signal according to how the original effect handled it in the first place. This means that **order matters**; it is different from having distortion and then delay as having delay and then distortion. This concept is the fundamental aspect of an **Effect Chain**. Since order matters, it is desired to have an interface in which the different effects can be rearranged at will and each effect can be modified independently. This is a concept that will be managed by a designed UI.

The UI will have two distinct modes to manage the guitar's signal. One mode is the "Edit Mode" which will be seen by the user to modify the state and order of the effects in the chain. This mode will give the option for the user of having different presets, or sound effect banks, which then can be modified due to specific parameters within the effect itself. Here the user can select to add or remove effect pedals as well as change the order of the pedals. The other mode allowed by the UI is the Performance/Play mode. This mode, as the name suggests, is the desired mode for the user to use during a performance. During a performance, the guitarist will enable or disable an effect pedal depending on the arrangement of the song performed. Therefore, it is necessary to provide the user with a way to swiftly change the activeness of a guitar pedal. Whatever effect the user selects in the edit mode, they will then have it available in their performance mode. The user also has the option to change the preset selected in this mode, which is useful as it avoids the user having to go to the edit mode in between songs. The following block diagram describes the behavior of the UI:

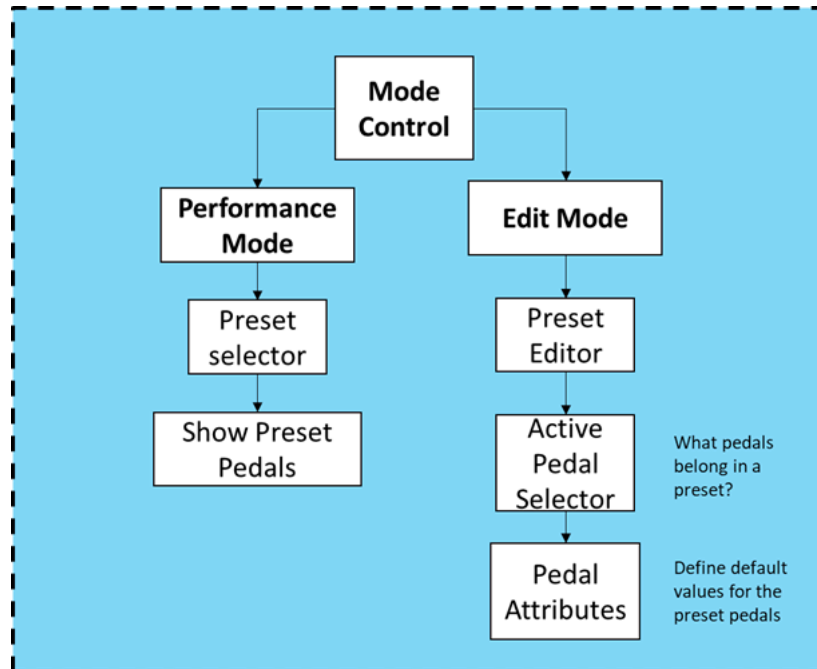


Figure 2.2 Software Block Diagram

Another major software component of the system is the creation of the VST effects themselves. The desired approach taken to develop these interfaces will be using a set of libraries called JUCE [5]. These libraries provide a clear path to develop the GUI interface for the parameters needed for each of the effects as well as for the digital signal processing needed to interface with the individual samples to generate the desired effect. As mentioned earlier, these effects need to be developed independently from one another to allow for the modularity required to create the desired effect chain.

2.1.3 System High-Level Overview

This section covers the specific modules required for the system to properly function. The Electronic Module of the system will oversee managing the input signal from the guitar and providing it in an appropriate format ready for processing by the microprocessor. It consists of guitar pickups, a bandpass filter, an amplifier circuit, and an ADC. The output module is interconnected with the Processing Module. This module consists of a Microprocessor that has the responsibility of overseeing the effect chain. This module relies on the I/O Module which consists of a touchscreen that is going to be the main interface for the user to modify the effect

chain as well as the specific parameters of the effects. After the audio is processed by the microcontroller in the processing module, this digital signal will then pass to the Audio Output Module which has the responsibility of faithfully reproducing the modified signal from the processing module into an analog form ready to pass to an amplifier or speaker. This module consists of a DAC, an amplifier circuit that brings the analog signal to the standard line level signal (-30Db), and a switch. This switch is directly connected to the guitar pickups in the Electronic Module and serves as a bypass for the effects. This means that when the switch is active the microprocessor will handle the audio signal and when inactive will then provide the clean guitar signal straight from the pickups. All modules receive power from the Power Module consisting of a single portable battery and a power switch.

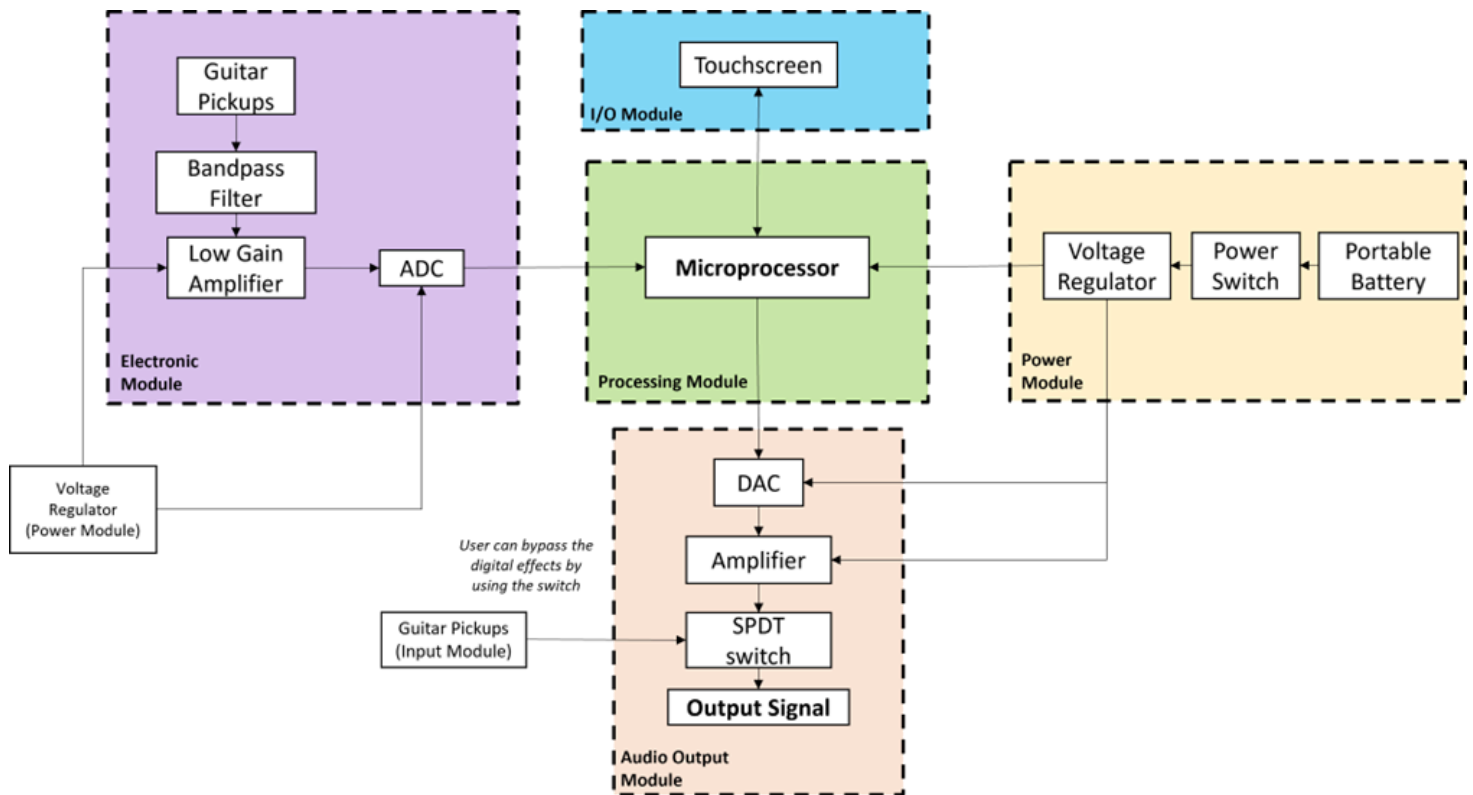


Figure 2.3 Modular Block Diagram

2.2 Engineering Requirements

At a high level, the project and modules have design considerations and guidelines that need to be covered to supply an acceptable product. These considerations are made such that the user has the appropriate output from the system. This project also seeks to provide an enjoyable experience, meaning that the product will not harm but only aid the performance of the guitarist. The following table describes the proposed Engineering requirements that the product shall follow to achieve the goal of the project.

Table 2.1 Engineering Requirements		
Level	Requirements (The Product Shall...)	Verification and Success Criteria
High	Be able to produce the following Guitar Audio Effects: Distortion, Delay, Reverb, and Modulation	The output waveform of each of these effects will be compared to the generic waveform expected for them. Each effect definition can be found in the Effects Description Table below.
	Correctly process input sound from the guitar pickups and be available at the output with the same level as the pickup input (instrument level of -30 dB)	Scope measurements of the input waveform from the pickups will be compared with the output waveform, where the same level must be experienced as well as the same waveform shape (Ignoring resolution loss due to the ADC/DAC since it is unavoidable)
	Have no more than 10ms latency between guitar input from pickups and output signal	Probing the input signal coming from the pickups against the output signal of the output amplifier should yield a latency below 10 ms
Medium	Be able to bypass the guitar pickup signal to the output.	The guitar input signal and the output signal will be compared while using a bypass switch, in this test both signals shall be identical.
Low	Be able to remain powered on for 4 hours with the display active	With a fully loaded effects chain (maximum power consumption) the time taken for the battery to run out will be measured.

2.2.1 Effects Requirements

The main goal of this project is to provide predetermined effects to the user. The effects, Distortion, Delay, Reverb, and Chorus, each have characteristic parameters and waveforms that are desired to be replicated by the processing module of the system. These effects will be required to follow the description seen below (Table 2.2):

Table 2.2 Digital Effects	
Effects	Waveform Description
Distortion	This effect modifies the original waveform by adding nonlinear harmonic distortion. The waveform flattens along positive and negative peaks which can either soft or hard clip the signal depending on the distortion parameter set by the user. This increases the amplitude of even harmonics (for the soft clip) or odd harmonics (for the hard clip).
Delay	This effect plays back a copy of the original waveform for a set amount of time. Some parameters controlled by this effect are time (controls the amount of time between the repetitions of the copy), the volume of the copies (called mix), and the feedback on the amount of time that the copies will play back.
Reverb	The wave has an added tail that emulates the audio bouncing out of the walls of a room. Two parameters are important in this effect which are the room size (dictates how long the reverb tail is active) and the effect's mix (how active the effect is with respect to the original wave)
Chorus (Modulation)	The main modulation type selected for this project is Chorus. This functions similarly to the delay effect with the difference that the pitch of the copies of the original signal is changed by a preset amount determined by the depth parameter. A rate parameter also controls how fast the modulation occurs.

2.3 Engineering Specifications

This system can be categorized into five main modules: processing, input/output, electronic, power, and audio output, as seen in the block diagram of the system (Figure 2.3) under Section 2.1.3. These modules work together simultaneously and need certain requirements to allow the system to properly perform. Each of the modules has components that make up certain requirements that are needed to perform their duty. Each component will then be broken down to its specific specifications for proper performance. If these specifications are not properly met when gathering materials, the system will not function properly which makes it essential to follow these specifications. A summary of the Engineering Specifications for this project can be seen in Table 2.3:

Table 2.3 Engineering Specifications				
Block Name	Component	Engineering Specification	Justification and Verification	Responsibility
Processing Module	Microcontroller	64-bit Processor	Maximize Data that can be used and can run the programs designated	Chris
		8 Gb RAM	Needed to run the programs at an efficient rate to allow the program to maximize latency	
		GPIO HAT	To connect the ADC/DAC to the microcontroller	
		minimum 32 Gb HDD	To allow space for OS and effects application	
I/O Module	Touchscreen	minimum 5-inch display	Big enough to display the different effects	
		HDMI and USB inputs	Be compatible with the microcontroller	
		LCD color display 800x480 pixels	Capable to display the interface	
		Touch compatible	Allow the user to select the effect	
Electronic Module	Bandpass filter	High pass and Low pass filter cutoff frequencies of 90Hz-500Hz and 2.5 khz-10khz.	Filter out inaudible frequencies and unnecessary noise before entering the ADC stage. Also, it provides users with ways to equalize signals further without digital effects.	Jose

	Low Gain Amplifier	Supply 0.8 Vrms (2.26 Vpp) to the ADC	Provide standard audio input voltage level to the ADC for efficient conversion without distortion	
	ADC	Signal to Noise Ratio 100 dB	Minimize the background noise	
		24-bit Resolution	Allow sufficient resolution for acceptable audio quality. (For a 2.26Vpp input it provides 135nV/Step)	
		GPIO Connection	Allows efficient transfer of data to the microprocessor	
		Minimum sample rate 44.1 kHz	Minimum requirement for CD audio quality	
Power Module	Portable Battery	Supply 18W to the system	Requirements to run the microcontroller and satisfy Electronic and Audio output modules power draw	Group
		15 Ah	Run the microcontroller for around 4 hours	
Audio Output Module	Output Amplifier	Bring output voltage from DAC to a maximum of 0.4 Vrms (1 Vpp).	Provide the proper voltage output level for a guitar	
	DAC	Signal to Noise Ratio of 100 dB	Minimize the background noise	
		24-bit Resolution	Allow sufficient resolution for acceptable audio quality.	
		GPIO Connection	Allows for sufficient sound quality at the output.	

2.3.1 Processing Module

The processing module is the brains of the operation and allows the system to run programs in the background to add effects to the audio signal and run a Graphical User Interface for the user's convenience. This is a large load set on the processing module which means a fast, reliable processor will be needed for this system. The processing module component will be a microcontroller. A microcontroller is a mini, credit card-sized computer. This will allow the system to be compact and fit inside an electric guitar. A 64-bit processor will be needed to maximize the data that can be used and run the programs that are designed for the project. A minimum of 8 GB of RAM will be needed to run the programs at an efficient rate to allow the program to minimize latency issues on the signal as well as the interface. A minimum of 32 GB

of memory in the hard drive is needed to allow space for the operating system and applications for the effects. The module will also need an HDMI port to connect the ADC/DAC to the system to convert the incoming and outgoing signals.

2.3.2 Input/Output Module

The input/output (I/O) module is the backbone of the project. This module allows the user to select from various effects and speaks to the processing module on what needs to be implemented. The I/O module will consist of a touchscreen as its component. The touchscreen needs to have a screen that is a minimum of four inches to be big enough to display the various effects to be chosen but five inches is the largest to fit the guitar. The touchscreen will need to have HDMI and USB inputs that will make it compatible with the microcontroller. The touchscreen will need a minimum LCD color display of 800x400 pixels to display the proper interface on the touchscreen, larger screens will need different pixel dimensions. The screen must be touch-compatible to allow the user to select the effect that is desired without other peripheral devices.

2.3.3 Electronic Module

The electronic module is the muscle of the operation. This module will be broken down into three components: a bandpass filter, a low-gain amplifier, and an ADC.

The bandpass filter will need high pass and low pass filter cutoff frequencies between 90 Hz – 500 Hz and 2.5 kHz -10 kHz. This component will filter out inaudible frequencies and unnecessary noise before entering the ADC stage. Also, this component will allow the user to further equalize the signal without the digital effects.

The low-gain amplifier will need to supply 0.8V (RMS) or 2.26V(pp) to the ADC. This specification is needed to provide a standard audio input voltage level to the ADC for efficient conversion without distortion.

The final component is the Audio to Digital Converter (ADC) which is responsible for taking incoming audio signals and converting them to digital signals to allow the computer to

read and edit them. This will need a signal-to-noise ratio of 100 dB to minimize the background noise. It will need 24-bit resolution to allow sufficient precision for acceptable audio quality. For a reference voltage of 5V, it will provide around 300nV per step. It will need a GPIO connection to allow efficient transfer of data to the microprocessor. Lastly, it will need a minimum sampling rate of 44.1 kHz, which is the minimum requirement for CD audio quality.

2.3.4 Power Module

The power module is responsible for supplying the system with power to perform its required tasks. The power module's main component will be a portable battery. This battery will need to supply 18W to the system which is the requirement to run the microcontroller and satisfy electronic and audio output modules power draw. The battery will need a minimum of 15Ah to efficiently run the microcontroller for approximately 4 hours.

2.3.5 Audio Output Module

The audio output module is responsible for getting the signal that has been altered to the proper output peripheral. This system will contain two components, an output amplifier, and a digital-to-analog converter. The output amplifier will need to bring output voltage from DAC to a maximum of 0.4V(rms) or 1V(pp), which is required to provide the proper voltage output level for a guitar. The DAC will need a Signal-to-Noise ratio of 100dB to minimize the background noise. It will require a 24-bit resolution to allow sufficient precision for acceptable audio quality. The DAC will need GPIO connections to allow for sufficient sound quality at the output.

2.4 Proposed Components and Power Budget

After analyzing the engineering specifications that will be required to properly perform this project, the components can be easily chosen. When finding components for the project it is required to read all data sheets to ensure the component fits the specifications that were described in section 2.3 of the report. Table 2.4 summarizes the power budget for this project:

Table 2.4 Power Budget				
Module	Item	Voltage (V)	Current (A)	Power(W)
Processing Module	Microcontroller	5 V	3 A	15 W
I/O Module	Touchscreen	5 V	550 mA	2.75 W
Electronic Module & Output Audio Module	ADC+DAC + I/O Amplifier	5 V	60 mA	0.3 W
Power Module	Portable Battery			-22W
Total				-4 W

For the processing module, a microcontroller will be needed to run the programs at the scale that is required. The microcontroller that will be used is a Raspberry Pi 4B with 8 GB of RAM. [7] This microcontroller has 8 GB of DDR4 SDRAM, HDMI ports, a 64-bit processor, and 128 GB of storage. This microcontroller contains all the minimum requirements stated in Chapter 2.4 and can confidently say will satisfy the needs of the project. This module will use 5 V and 3 A which equates to a power draw of 15 W.

For the I/O module, a touchscreen will be needed to allow the user to select from various effects seamlessly. The touchscreen that fits the required specifications is the Raspberry Pi's official 7-inch touchscreen. This touchscreen has a 7-inch display and is manufactured for the sole purpose of connecting to the Raspberry Pi 4. This touchscreen will allow enough room on the screen for the user to comfortably select its effect. This touchscreen uses 5 V and 0.55 mA which is a total of 2.75 W.

For the Electronic Module as well as the Output Audio module the ADC+DAC + I/O amplifier that will be used is the DAC + ADC from Hifiberry. [8] This ADC+DAC is compatible with the Raspberry Pi 4 and contains the ADC, DAC, and amplifier that are required to run this project. After reviewing the datasheet [9], provides all requirements that were set out in the specifications and more. This module uses 5 V and 60 mA for a total of 0.3 W.

For the Power Module, the portable power bank that matches the required specifications is the KEOLL Power Bank.[10] This power bank has a 25.8 Ah, 22.5 W battery. This has USB

ports to allow all components to be efficiently plugged in. This gives us more than enough wattage and amp hours to power this system for the allotted time. The wattage needed to run this system is 18 W, the power bank has a wattage of 22.5 W and allows 4.5 W leftover. The 25.8 Ah allows the system to run longer than the intended 4 hours when running at full power.

In Figure 2.4, a 3D rendering of these parts integrated within the guitar can be appreciated. The image is in scale with the dimensions of the guitar as well as the components used such as the Raspberry Pi, the ADC/DAC, the touchscreen as well and a placeholder box for the battery. Note that the face of the guitar is excluded from this image to appreciate the components inside of it.

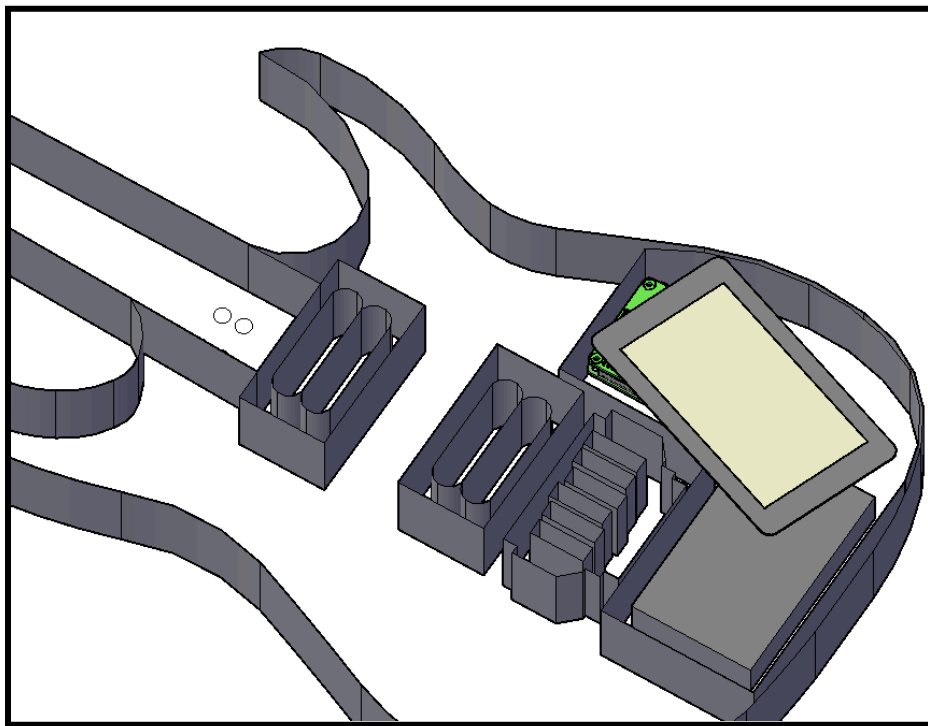


Figure 2.4 CAD Rendering of the SEG (Inner Components Exposed)

In the following CAD Rendering (Figure 2.5) a specific description of the parts integrated within the guitar can also be seen. Note that Figure 2.5 is a visual from the underside of the guitar.

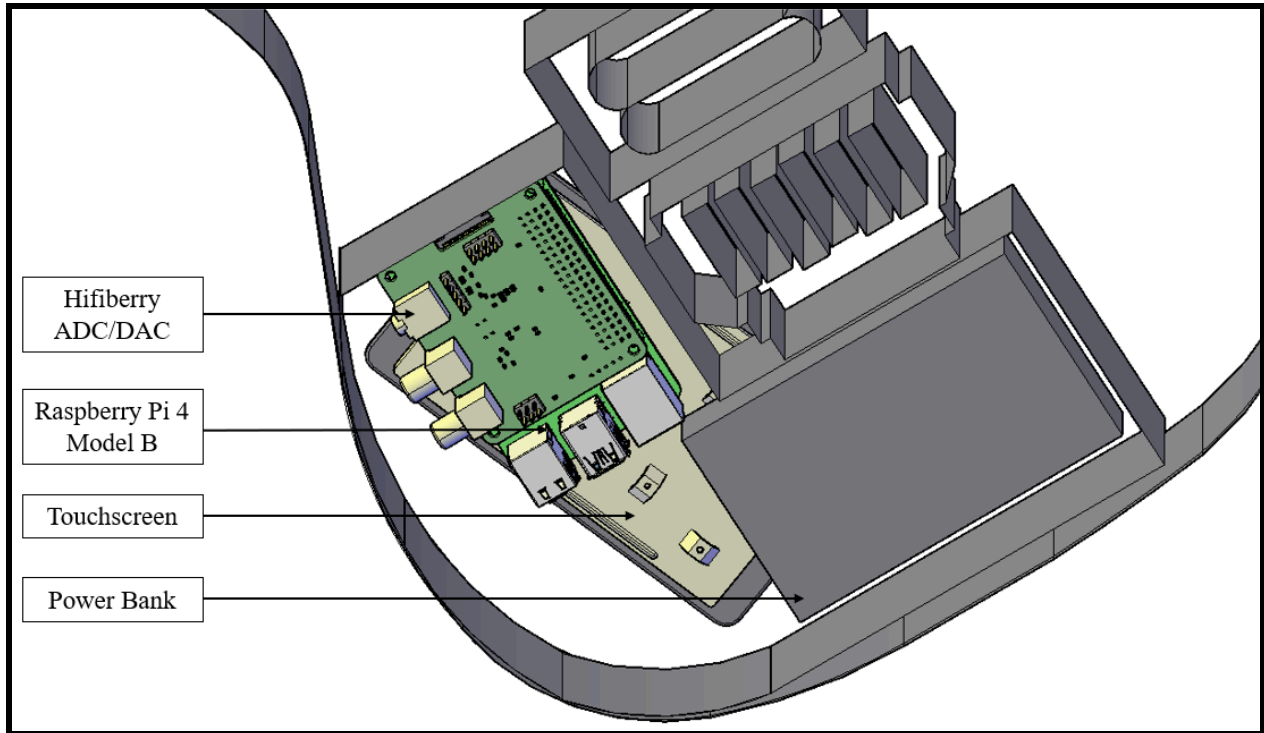


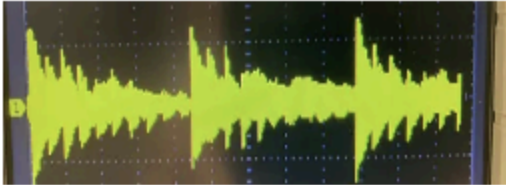
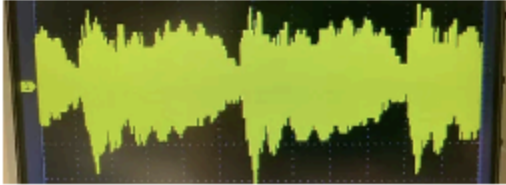



Figure 2.5 CAD Rendering of SEG Components. (Bottom Side View)

2.5 Success Criteria

2.5.1 Digital Effects Success Criteria

The sound quality and aspects of a sound waveform cannot be objectively deduced just by listening to the effect itself. Due to this, the success of the effects being developed will be tested by taking oscilloscope measurements of each effect from the output of the device. The output waveforms shall resemble the expected waveform for the desired effect. **Table 2.5** below displays examples of how the waveform of a clean signal gets modified by a specific effect. The output provided by the effects created in the processing module shall resemble the waveform behaviors seen in Table 2.5. Note that with “resemble” we mean that the characteristic changes from the original waveform to the output waveform shall also be apparent in the VST effects developed for this project.

Table 2.5: Digital Effect Waveforms	
Effects	Sample Waveform to be recreated
No Effects (Clean Signal)	
Distortion	
Delay	
Reverb	
Chorus (Modulation)	

The effects seen in Table 2.5 were gathered by using a Roland SP404A sampler device. The sampler has the main functionality of recording a sound sample and playing it back, with the capability of adding digital effects to the clean sample. The sound sample recorded was a single musical note G_3 (~ 196 Hz). The test was performed by stripping a line cable to reveal its two ground and signal wires, which were attached to a Rigol DS1102D Oscilloscope. This line cable was connected to the output of the sampler. Different waveforms were collected by applying the four different effects. Note that for the chorus effect, the scaling for the view was expanded in the x -axis to better show the effect.

2.5.2 Graphical User Interface Success Criteria

Testing of the Graphical User Interface (GUI) will be conducted in many stages of this project. The initial GUI testing will be issued after the completion of the first GUI. The first GUI will be a simple GUI with four buttons with our desired effects as their names. It can be concluded that the GUI passes the first test when a button is selected, and the program outputs the name of the desired effect. This GUI will undergo many changes throughout the semester to accommodate the ever-changing critiques that will occur during this process. Once the interface reaches the desired orientation and look, it will be tested once more with the program that prints the effect that is selected.

Once the desired look has been achieved, the next step is to make the interface allow the buttons to change the order of the effect positions. In this system, the order of the effects will change the output of the selected effects. This will require the touchscreen to be able to drag and drop these buttons in an “edit” mode. Once the edit mode has been successfully created, the next step of testing the success criteria is to test the commands once more and see if they successfully output the button that is selected.

Once the interface reaches the desired look, the next portion of the success criteria is to incorporate the backend software. This code will need to successfully output the desired effect that has been selected. It will need to select and deselect from various buttons and put them in the order the user desires. This will be considered successful if the GUI can accurately detect what effect and order of effects want to be implemented.

2.6 Prototype Testing

This semester, the system went through many stages of initial testing and configurations. This allowed us, the developers, to successfully improve the ideas that were initially proposed. This testing came with failures from the initial design but has been properly adjusted to create a functioning system.

2.6.1 Distortion Effect Prototype Testing

Throughout this semester several stages of testing were performed in the creation of digital effects. In particular, the distortion effect was the first to be chosen for development.

An approach for the creation of distortion originates by simulating what occurs when a signal clips in a diode circuit. To create a digital effect that produces this characteristic waveform, a mathematical function needs to be created to modify the samples of the waveform. Through our research performed on the distortion effect a function that fits our criteria required is satisfied by a function in the form of [13]:

$$f(s) = \frac{2}{\pi} \arctan(s * g)$$

Where the input parameter is (s) which is the sample amplitude and (g) which is our gain value. The value of the gain is an arbitrary scaling factor to increase the clipping experienced in the curve. The following plot (Figure 2.6) demonstrates the behavior of this function by letting $s = \sin(x)$ and "g" be an arbitrary value of 8.

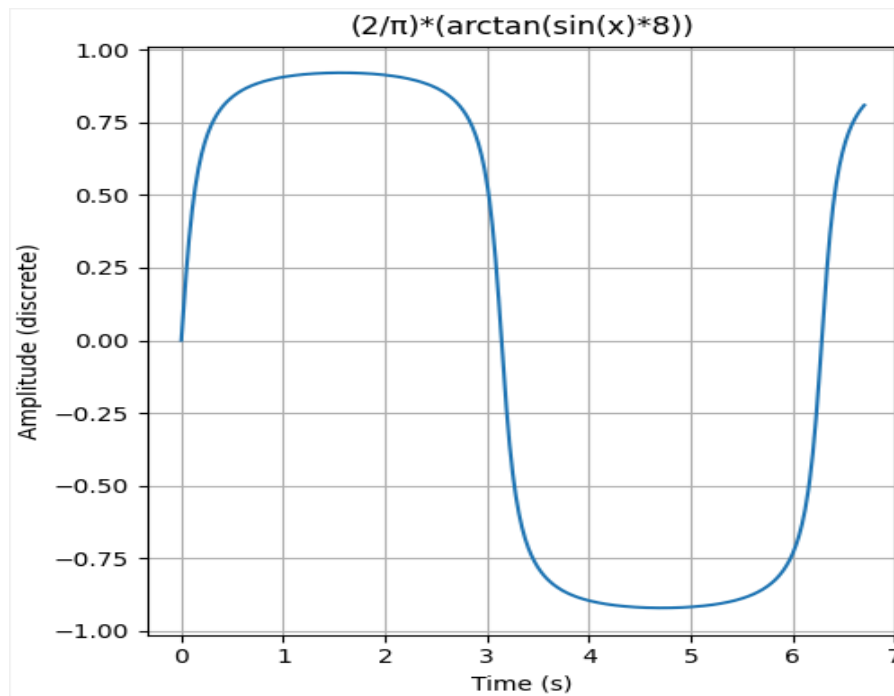


Figure 2.6 Arctan Distortion Function

A couple of parameters were also added to further modify the effect on the signal. A volume control was added to reduce the volumes of the samples (v) at the user's will and a drive parameter was added as a division factor for the gain, this dictates the slope in which to reach the clipping point in the signal.

In practice, the full formula implemented by the effect becomes:

$$f(s) = v(\frac{2}{\pi} \arctan(sgd) + \frac{s}{2})$$

The parameter for gain was chosen to be from [1-150] based on the testing performed, which showed these values to create sufficient distortion. The distortion and volume factors were set from [0,1] as they serve as scaling/reducing factors. A simple GUI for this effect is shown below and was used to test the performance of the effect.

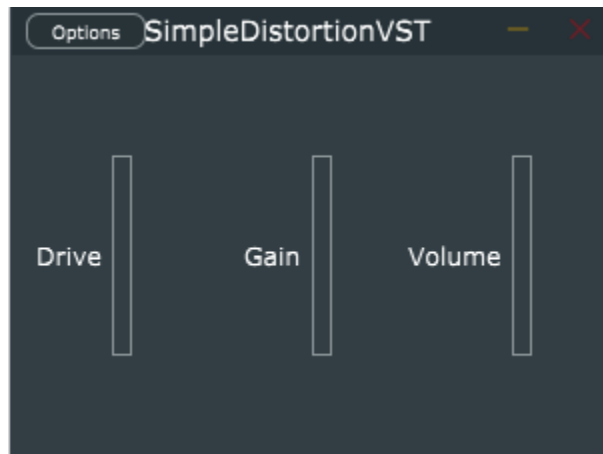


Figure 2.7 Prototype UI for the Distortion Effect

A code sample used to create the above effect can be seen in Appendix A. Along with this, the sound produced by this effect can be listened to with the following link provided:

https://static.wixstatic.com/mp3/f9d080_806e78a820304751b535648f57154b5f.wav

2.6.2 GUI testing

Throughout the first semester the Graphical User Interface (GUI) went through many stages of prototype testing. This semester, three GUIs were successfully created and critiqued to acclimatize the ever-changing system requirements and aesthetics. Stage 1, shown below, was a trial to understand the functionality of GUI design. This design implements buttons to select the various effects. This GUI design resembled a childhood game of four squares with individual effects in their respective corners of the screen.

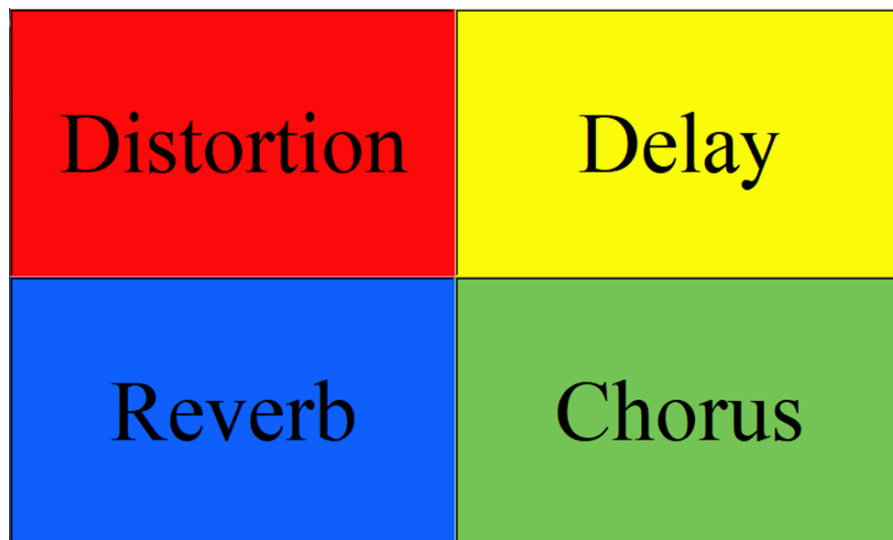


Figure 2.8 GUI Test Stage 1

This design went through testing and successfully printed the command of the selected button, but it does not allow the user to see what effect is currently selected on the screen. This led to the creation of trial two. This uses radio buttons that will allow the user to see what effect is currently being selected and cycle through them.



Figure 2.9 GUI Test Stage 2

This GUI configuration allows the user to see what effect is currently being processed and passes the test that prints the selected effect. When using effects, the order in which the effects are presented matters as stated earlier. The final GUI trial was underway and allowed the user to see what effect was selected and the order in which it was selected. It uses vertical bars rather than the four-square approach to allow the user to create different orders between the effects.

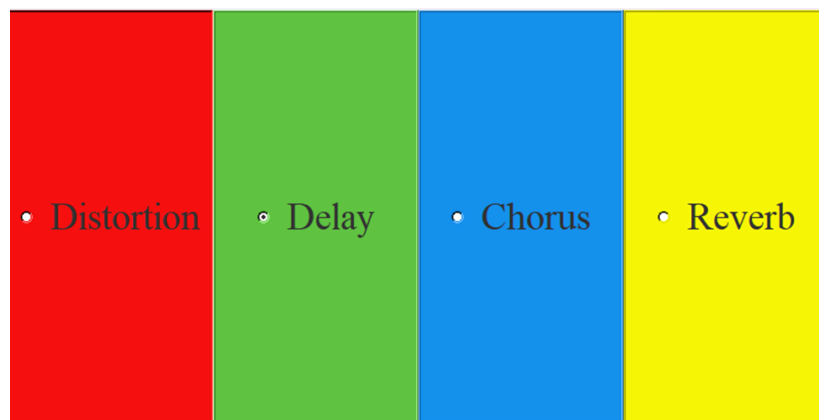


Figure 2.10 GUI Test Stage 3

2.6.3 Hardware Integration Attempts

The Raspberry Pi, shown in Figure 2.11, visualizes the many forms of connectivity that are provided. This Raspberry Pi will utilize the type-c power supply connection to power the microcontroller, the GPIO headers for power to the touchscreen, and the MIPI DSI Display Port for the test. The touchscreen used for this demonstration is a Raspberry Pi 7” standard touchscreen.

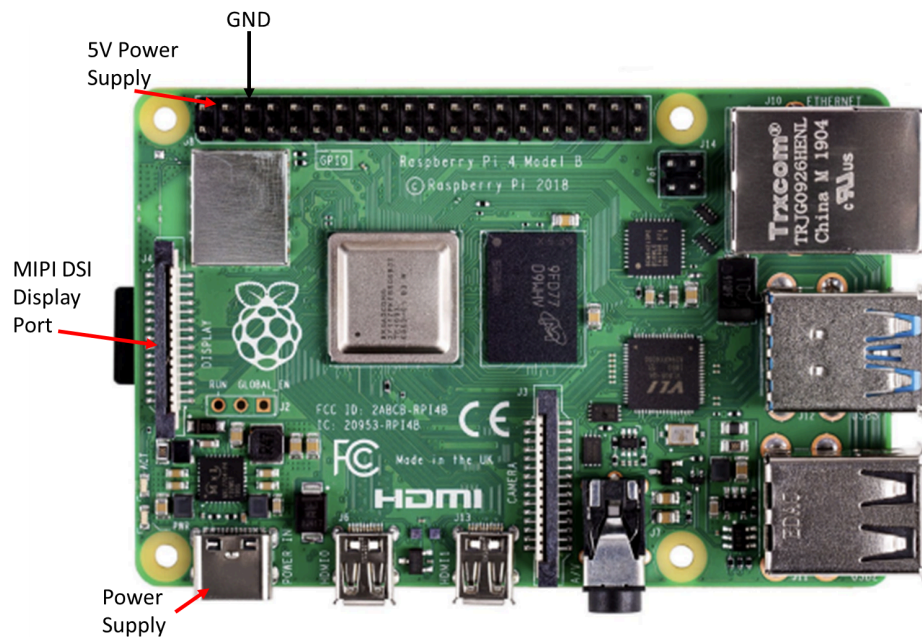


Figure 2.11 Raspberry Pi 4B and Connections Testing

The touchscreen, Figure 2.12, has versatile connections as well. The touchscreen can connect to the Raspberry Pi by a Printed Circuit Board that is mounted on the back. This allows the board to be able to receive what the Raspberry Pi is transmitting by various connections. The connections that this test will utilize are the GPIO headers and the Display Port. The Display Port of the two boards will be connected by an FFC ribbon cable.

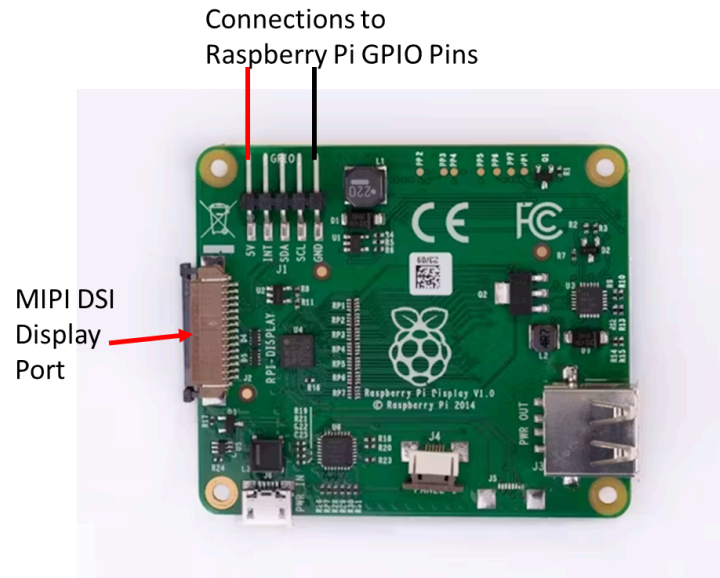


Figure 2.12 Touchscreen PCB for Connection Testing

The PCB is mounted on the back of the touchscreen and offers a mount for the Raspberry Pi. The PCB and Raspberry Pi are mounted onto the board with the connections shown below.

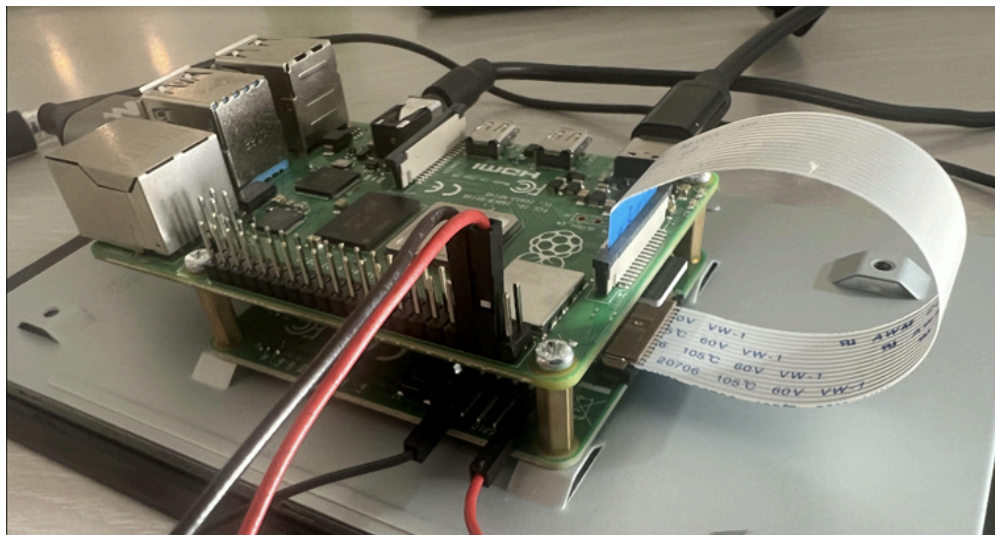


Figure 2.13 Raspberry Pi and Touchscreen Integration

ribbon cable which means that the Raspberry Pi also cannot be placed into the guitar itself since the ribbon connector is very fragile and might suffer damage if left exposed to the outside surface of the guitar.

It was also found that a simpler/smaller touchscreen might be more convenient to use for this project. Specifically, a display within four to five inches has the appropriate dimensions to fit within the guitar body and still provide appropriate user experience in the GUI. Along with this a touchscreen that uses a more user-friendly format such as an HDMI/USB port is more appropriate as they can be left exposed without the threat of damage to the component itself. Due to this, a candidate for the touchscreen became the HDMI/USB-C Raspberry Pi display on Figure 2.15, which ensures the above-mentioned concerns are resolved while allowing acceptable user experience.



Figure 2.15 5” HDMI/USB touchscreen [14]

Chapter 3

Non-Technical Issues

Summary

In this chapter, we will discuss the budget, proposed timeline, concerns ethically, and ergonomic issues of the project.

3.1 Budget

3.2 Proposed Timeline

3.3 Issues with Ergonomics

3.4 Ethical Concerns

3.1 Budget

The components that are needed to complete this project can be found on Amazon and other common retailers. The microcontroller that will be used for this project is a Raspberry Pi 4B which can be purchased at BestBuy for \$180.[7] This microcontroller comes in a kit that includes a case, HDMI cables, and an SD card for storage. The DAC+ADC that will be used for this project is from Hifiberry, a retailer that creates audio add-ons for Raspberry Pi.[8] This DAC+ADC is \$50 on the website but is essential for the project because of the duality of the component and allows the engineers to utilize one component for two various essential needs. The touchscreen that will be used is the ELECROW 5" touchscreen. This touchscreen can be found on Amazon for \$55. [14] The power bank that will be required to be used to run our system is the KEOLL 25800 mAh power bank.[10] This power bank is \$21 and has the specifications that are required to keep the system powered for a significant amount of time. The last components that will be required are electrical components including capacitors, potentiometers, and resistors that will be provided by the engineering lab. The proposed budget is shown in Table 3.1.

Table 3.1 Projected Budget			
Quantity	Item	Cost	Where to buy
1	Raspberry Pi 4B kit includes Power Supply, SD Card, HDMI Cords, Case, Heat Sinks, Cooling Fan	\$180	BestBuy
1	DAC+ADC	\$50	Hifiberry
1	5" Touchscreen Display	\$55	Amazon
1	Power Bank	\$50	Amazon
1	Electrical Components (Capacitors, Potentiometers, Resistors, etc.)	\$0	Engineering Lab
	Total:	\$335	

3.2 Proposed Timeline

This project will be broken down into two phases, Fall semester and Spring semester. The Fall semester will focus on initial functionality whereas the spring will focus on upgrading these features to create a more advanced, customizable pedalboard for the user.

3.2.1 Fall

The Fall semester contains basic functionality procedures for the project. The focus of this semester is to gather materials, create a Graphical User Interface, and work on the code for the first audio signal effect. For weeks one through three, the objective includes gathering materials, setting up the OS for the microcontroller, beginning the first stages of GUI design, creating a Band-pass filter, and researching transfer functions for digital system processing effects. The objective for weeks four through six will be to test and document waveforms, create a rough draft GUI, work on the first effect module being distorted, and work on the proposal. Weeks seven through ten will carry an objective to finish the proposal report, GUI module, and distortion effects. The final week, week eleven, will integrate the distortion effects into the GUI that was created. This integration will allow the user to select the distortion effect on the touchscreen and distort the voice of the user in a microphone. This will also be the week of the proposal presentation and the submission of the proposal report. The documented timeline is presented below in Table 3.2.

Table 3.2 Proposed Timeline Fall Semester	
Week	Objective
1 (9/27)	Purchase Raspberry Pi and set up the OS. Band-pass Filter Rough Draft Design, Testing.
2 (10/4)	Begin creating the GUI. Researched Transfer Functions for DSP effects
3 (10/11)	Select and purchase touchscreen
4 (10/18)	Abstract and Chapter 1 for the Report.
5 (10/25)	Test using existing effects and document their waveforms. Rough Draft for the GUI Play Module on PyQT.
6 (11/1)	Work on the Distortion Effect Module and refine GUI Play Mode. Work on the Proposal Report
7 (11/8)	Wrap up Report Chapter 2.
8 (11/15)	Finish Report Rough Draft (Chapters 3/4)
9 (11/22)	Proposal rough draft finished and ready for revision
10 (11/29)	GUI play module finished and ready for audio code. Work on Presentation
11 (12/6)	GUI on touchscreen with a basic voice modulator finished. Have the presentation ready for review.

3.2.2 Spring

The spring semester will focus on creating a more advanced, customizable system for the user. The first three weeks of the Spring semester will focus on creating the second module of the GUI, the edit module. The edit module will allow the user to move the various effects on the screen in order of importance. The second effect will be created during these weeks as well, the Reverb effect. Once Reverb is created it will be integrated into the GUI like the distortion effect. Weeks four through six will be to create the Delay effect and implement that into the GUI as well. These weeks will also focus on the integration of hardware into the guitar and begin initial testing. Weeks eight through eleven will conclude the creation of the effects and integration of

those programs into the GUI. These weeks will begin high-level testing of the system and identify any errors made in hardware and software integration. The final weeks, twelve through sixteen, will focus on cleaning up any of these errors that were found. These weeks will have a final product that will need to undergo low-level testing to ensure there are not any bugs or power issues. These weeks will also be the finalization of the report and presentation. Week sixteen will hold the submission of the Report and the presentation of the project. The proposed timeline for the spring semester is shown below in Table 3.3.

Table 3.3 Proposed Timeline Spring Semester	
Week	Objective
12 (1/08)	Begin editing the module of the GUI. Wrap up Distortion Effect VST.
13 (1/15)	Continue editing Module on the GUI. Work on Reverb Effect VST.
14 (1/22)	Begin VST integration in the GUI. Finish Reverb Effect. Work on Delay Effect VST.
15 (1/29)	Continue VST integration in GUI. Finish Delay Effect VST.
16 (2/5)	Assemble the Microprocessor into the Guitar Finish Delay Effect VST. Work on Chorus Effect VST.
17 (2/12)	Hardware integration, Band-pass Filter Building, and testing. Work on Chorus Effect VST.
18 (2/19)	Start the Report. Finish Chorus Effect VST.
19 (2/26)	Work on Report, Continue Integration of GUI and VST effects.
20 (3/4)	Finish Report Finish Hardware integration
21 (3/11)	GUIs completed play and edit
22 (3/18)	Audio effects working with GUIs
23 (3/25)	Hardware integration testing

24 (4/1)	Report finished
25 (4/8)	Begin testing
26 (4/15)	Finish testing
27 (4/22)	Presentation

A GANTT chart was created to allow the engineers to keep track of progress and ensure that they are on track to complete the project by the proposed deadline. The GANTT chart is broken into two semesters shown below.

Table 3.4 Fall GANTT Chart											
Week	1	2	3	4	5	6	7	8	9	10	11
Date	9/27	10/4	10/11	10/18	10/25	11/1	11/8	11/15	11/22	11/29	12/6
Acquire Materials											
Setup Device OS											
GUI Design (Play Mode)											
Connect Code and GUI											
Test Code and GUI Functionality											
Bandpass Filter design											
Test effect waveforms											
Distortion VST Code											

Table 3.5 Spring GANTT Chart																
Week	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
Date	1/8	1/15	1/22	1/29	2/5	2/12	2/19	2/26	3/5	3/12	3/19	3/26	4/2	4/9	4/16	4/23
Acquire Materials																
GUI Design (Edit Mode)																
Connect Code and GUI																
Test Code and GUI Functionality																
Reverb VST Code																
Delay VST Code																
Chorus VST Code																
High-Level Testing																
Hardware integration																
Battery integration																
Final Fit and Finish																
Low-Level Testing																
Final Testing																
Writing Report																
Submit Report																
Presentation																

3.3 Issues with Ergonomics

Working in the confined spaces of the boundaries of the guitar plays a challenge but making it intuitive and easy to use for the guitarist but also comfortable. Since changing effects is usually done by using a footswitch, this might be difficult to get used to for the guitarist. Therefore, it would be advantageous for the placement of this device to be closer to where the guitarist is playing. For a right-hander guitar, (which is the one where the prototype will be built) the proposed placement is in the lower right corner. This placement presents a comfortable placement for the user as it allows for quick access during a performance due to the fingers of the guitarist being close to the device itself. Initially the decision was made to make the touchscreen bigger and retractable for a more flexible experience, but portability issues discussed in Section 2.6.5 made it difficult to accomplish. Therefore, the preferred design decision taken was to place the touchscreen flat on top of the guitar itself and have all components within the guitar to make the system more portable. (this way the device is also able to fit in standard guitar cases)

3.4 Ethical Concerns

The Industry of Electrical and Electronic Engineers produced a Code of Ethics that engineers are expected to follow with the utmost regard in all aspects of their work. This ethical code helps to ensure the safety of people and the protection of people's work and ideas. This system ensures the Code of Ethics is followed due to the simple Graphical User Interface that has been produced. This ensures the user follows the proper guidance of the intentions of the system. All references are properly documented in the references portion of the report. This ensures all work that was used in the research portion of this project is properly credited to the engineers who contributed to the work. The electrical system will be properly analyzed and calculated to ensure it is a system that is safe and will not cause harm to any users. [12]

Chapter 4

Conclusion

Summary

In this section we will summarize our current findings and work moving forward. An extensive overview of the system will be also given as well as future considerations for the project.

4.1 Summary and Conclusion

4.2 Suggestions for Future Work

4.1 Summary and Conclusion

In this report we have introduced the concept of SEG. A project in which we intend to create a more user-friendly environment for guitar effects by allowing the effects to be built into the guitar itself. This system seeks to modernize an unexplored territory within the guitar industry in a fashion in which most guitarists can become accustomed to and quickly pick up.

In Chapter 2 we have been introduced to core concepts to understand the creation of digital effects as well as how to integrate them into a guitar efficiently and comfortably for the user. We have discussed the core principles of digital effect design and the considerations needed for the creation of the SEG project.

The overall requirements of the project were discussed as well as the different modules of the system which each contribute to specific aspects required for the correct functionality of the system. Throughout the report we also addressed the different modules used within the system regarding their hardware and software elements, by which we defined the specifications required for those components to successfully follow the design characteristics of the system.

Prototyping work done during the proposal stage was also presented in Section 2.6, which showed proof of concept for the design as having a functional user interface as well as the functionality for one of the sound processing modules implemented in code.

Within chapter 2 we also discussed some of the design considerations regarding the software use, in which we concluded that the most optimal approach to achieve the concept was to utilize and develop standardized VSTs as they allow fast processing and modularized interface that allows for expansion of the project via third party created effects. (which means that the end user will not be limited to use the VSTs build by default in the device)

4.2 Suggestions for Future Work

For the future progression of the system, one of the suggestions is for the Graphical User Interface. As of right now, this system can only show the effects in a vertical bar notation and does not allow the user to change the order of the effects. It has been proposed for the future to create a GUI that can change the order of the effects in an “edit” mode in a drag-and-drop orientation. The future shall hold the integration of hardware into the guitar and create a seamless system. It is also recommended that the buttons of the GUI have a more relaxed color scheme that is not as vibrant and looks more professional. There should also be an encasing for the backside of the touchscreen to enclose the PCBs.

References

- [1] Sobot Peter, “*PedalBoard*”, Zenodo, 2021, Available at:
<https://doi.org/10.5281/zenodo.7817838>
- [2] AutoDITex, “*Anti-Lock Braking System (ABS)*” Anti-lock braking system (ABS), Available at:
<https://autoditex.com/page/anti-lock-braking-system-abs-9-1.html>
- [3] Dr. Lois Berg, “*Soil and Plant Nutrition: A Gardener’s perspective*”, Garden and Yard, Available at: <https://extension.umaine.edu/gardening/manual/soils/soil-and-plant-nutrition/>
- [4] O. Das Biee, “*DIGITAL AUDIO EFFECTS.*” Available:
<https://ccrma.stanford.edu/~orchi/Documents/DAFx.pdf>
- [5] “JUICE | JUICE,” juice.com. <https://juice.com/>
- [6] E. Zeki, “DIGITAL MODELLING OF GUITAR AUDIO EFFECTS A THESIS SUBMITTED TO THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES OF MIDDLE EAST TECHNICAL UNIVERSITY.” Accessed: Oct. 18, 2023. [Online]. Available: etd.lib.metu.edu.tr/upload/12618426/index.pdf
- [7] “CanaKit Raspberry Pi 4 Extreme Kit 8GB RAM Black PI4-8GB-EXT128EWF-C4-WHT-RT,” Best Buy.
<https://www.bestbuy.com/site/canakit-raspberry-pi-4-extreme-kit-8gb-ram-black/6429585.p?skuId=6429585> (accessed Nov. 06, 2023).
- [8] “HiFiBerry DAC+ ADC | HiFiBerry.”
<https://www.hifiberry.com/shop/boards/hifiberry-dac-adc/> (accessed Nov. 06, 2023).
- [9] “Datasheet DAC+ ADC | HiFiBerry.”
<https://www.hifiberry.com/docs/data-sheets/datasheet-dac-adc/>

- [10] “Amazon.com: KEOLL Portable Charger 25800mAh Power Bank, 22.5w Fast Charging Battery Pack, LED Display USB C Backup Battery, Slim Portable Phone Charger for iPhone 15/14/13 Pro Samsung Galaxy iPad AirPods: Cell Phones & Accessories,”
www.amazon.com.
https://www.amazon.com/KEOLL-Portable-Charger-25800mAh-Charging/dp/B0C8SBKVN?ref=sr_1_3?crd=19R7XGR2MY36&keywords=keoll%2Bpower%2Bbank&qid=1699292564&srefix=KEOLL%2BPOWER%2B%2Caps%2C104&sr=8-3&th=1 (accessed Nov. 06, 2023).
- [11] “Amazon.com: Raspberry Pi Official Touch Screen,” Amazon.com.
https://www.amazon.com/Raspberry-Pi-Official-Touch-Screen/dp/B073S3LQ6Q/ref=sr_1_4?adgrpid=1345803589006750&hvadid=84112918724436&hvbmt=be&hvdev=c&hvlocphy=45058&hvnetw=o&hvqmt=e&hvtargid=kwd-84113613943057%3Aloc-190&hydadcr=18064_13443538&keywords=raspberry+pi+touchscreen+display&qid=1697492190&s=electronics&sr=1-4 (accessed Nov. 06, 2023).
- [12] “IEEE Code of Ethics.” Available at:
<https://www.ieee.org/content/dam/ieee-org/ieee/web/org/about/corporate/ieee-code-of-ethics.pdf>
- [13] D. Yeh, J. Abel, and J. Smith, “SIMPLIFIED, PHYSICALLY-INFORMED MODELS OF DISTORTION AND OVERDRIVE GUITAR EFFECTS PEDALS,” 2007. Available:
https://ccrma.stanford.edu/~dtyeh/papers/yeh07_dafx_distortion.pdf
- [14] ELECROW 5 inch Monitor for Raspberry Pi Monitor Touchscreen Display Compatible with Raspberry Pi 5 4 3B+ 3B BB Black Banana Pi Jetson Nano Windows PC
https://www.amazon.com/dp/B07FDYXPT7/ref=sspa_dk_detail_1?psc=1&pd_rd_i=B07FDYXPT7&pd_rd_w=dt0pU&content-id=amzn1.sym.f734d1a2-0bf9-4a26-ad34-2e1b969a5a75&pf_rd_p=f734d1a2-0bf9-4a26-ad34-2e1b969a5a75&pf_rd_r=XDNPVWMHCSN5XH8BJW2H&pd_rd_wg=L2bxA&pd_rd_r=414b807f-a0f2-4d11-9b9d-f595c02aab0c&s=pc&sp_csd=d2lkZ2V0TmFtZT1zcF9kZXRhWw

Appendix A

Distortion Effect Code Samples

Summary

This Appendix serves as reference to display the work done on the distortion effect in the JUCE C++ framework. Due to the modularity and extensiveness of the classes that JUCE provides as a template for the VST plugins, only samples of methods used will be covered.

A1. C++ Code for Audio Processing Block

A2. C++ Code for Parameter Building

A1. C++ code for Audio Processing Block

This block of code serves as a reference to the work done within JUCE to create the distortion effect from the description given in Chapter 2 section 2.6. Here the AudioProcessor class has a baseline method called the “process block” which is used to manipulate the state of the samples within the buffer. Here the arctangent function described in Chapter 2 is implemented.

```
void AudioProcessor::processBlock (juce::AudioBuffer<float>&
buffer, juce::MidiBuffer& midiMessages)
{
    juce::ScopedNoDenormals noDenormals;
    auto totalNumInputChannels  = getTotalNumInputChannels();
    auto totalNumOutputChannels = getTotalNumOutputChannels();

    for (auto i = totalNumInputChannels; i <
totalNumOutputChannels; ++i)
        buffer.clear (i, 0, buffer.getNumSamples());

//gather all input parameters

    float drive = *state->getRawParameterValue("drive");
    float gain = *state->getRawParameterValue("gain");
    float volume = *state->getRawParameterValue("volume");

//we want to iterate over all of the buffer's channels

    for (int channel = 0; channel < totalNumInputChannels;
++channel)
    {
        auto* channelData = buffer.getWritePointer (channel);

//we want to iterate over all of the buffer's samples

        for (int sample = 0; sample < buffer.getNumSamples();
sample++) {
            //channel data in this case is the individual sample
            obtained within the buffer.
            float cleanSig = *channelData;
            *channelData *= drive*gain;
            *channelData = ((volume) * (((PIOVER2) *
(atanf(*channelData))) + (cleanSig / 2))));

            channelData++;
        }
    }
}
```

A2. C++ code for Parameter Building

JUCE provides a class to interface with the parameters created in the GUI which is the “AudioProcessorValueTreeState”. We can create an instance of this class called “state” in which we can provide values for the parameters accessed in the GUI. In this case, the three parameters described in Chapter 2 are defined with their specific ranges and step sizes. These values are passed as pointers to improve performance, as these values need to be constantly accessed in our process block to create the effects in real-time when changing their parameters.

```
AudioProcessor::AudioProcessor()
// JUCE Code Template
#ifdef JucePlugin_PreferredChannelConfigurations
    : AudioProcessor (BusesProperties()
                      #if ! JucePlugin_IsMidiEffect
                      #if ! JucePlugin_IsSynth
                        .withInput  ("Input",
juce::AudioChannelSet::stereo(), true)
                      #endif
                        .withOutput ("Output",
juce::AudioChannelSet::stereo(), true)
                      #endif
    )
#endif
{
    // We define the state as a pointer variable containing all of
    // the parameter information such as ranges and default values
    state = new juce::AudioProcessorValueTreeState(*this,
nullptr);
    state->createAndAddParameter("drive", "Drive", "Drive",
juce::NormalisableRange<float>(0.f, 1.f, 0.01f), 0.f, nullptr,
nullptr);
    state->createAndAddParameter("gain", "Gain", "Gain",
juce::NormalisableRange<float>(1.f, 150.f, 0.01f), 0.f, nullptr,
nullptr);
    state->createAndAddParameter("volume", "Volume", "Volume",
juce::NormalisableRange<float>(0.01f, 1.f, 0.01f), 0.f, nullptr,
nullptr);

    state->state = juce::ValueTree("drive");
    state->state = juce::ValueTree("gain");
    state->state = juce::ValueTree("volume");
}
```

Appendix B

Graphical User Interface Code Samples

Summary

This Appendix serves as reference to display the work done on the Graphical User Interface that was built using Python coding. This Graphical User Interface went through many stages of trial and error and only the result will be displayed.

B1. Python Code for Graphical User Interface

B1. Python Code for Graphical User Interface

This code was built on the Spyder Integrated Development Environment. This Graphical User Interface utilizes the Tkinter module to build the buttons and framework. This was made possible using the website visualtk.com to build the GUI using a drag-and-drop representation with the buttons. After dragging and dropping, it outputs the Python program. This allows the user to fine-tune the end programming code while saving time from building the framework. This GUI utilizes radio buttons to show what button is being used at the time. The commands that are processed when the button is pressed will be inserted into the functions corresponding to their button name.

```
import tkinter as tk
import tkinter.font as tkFont
class App:
    def __init__(self, root):
        #setting title
        root.title("undefined")
        #setting window size
        width=800
        height=400
        screenwidth = root.winfo_screenwidth()
        screenheight = root.winfo_screenheight()
        alignstr = '%dx%d+%d+%d' % (width, height, (screenwidth - width) / 2,
(screenheight - height) / 2)
        root.geometry(alignstr)
        root.resizable(width=False, height=False)

        GRadio_825=tk.Radiobutton(root)
        GRadio_825["activebackground"] = "#f8484"
        GRadio_825["activeforeground"] = "#f9f1f1"
        GRadio_825["anchor"] = "center"
        GRadio_825["bg"] = "#f81111"
        GRadio_825["cursor"] = "arrow"
        GRadio_825["disabledforeground"] = "#fbefef"
        ft = tkFont.Font(family='Times',size=28)
        GRadio_825["font"] = ft
        GRadio_825["fg"] = "#333333"
        GRadio_825["justify"] = "center"
        GRadio_825["text"] = "Distortion"
        GRadio_825["relief"] = "sunken"
        GRadio_825.place(x=0,y=0,width=200,height=400)
        GRadio_825["value"] = "100"
        GRadio_825["command"] = self.GRadio_825_command

        GRadio_642=tk.Radiobutton(root)
        GRadio_642["anchor"] = "center"
        GRadio_642["bg"] = "#60c441"
        ft = tkFont.Font(family='Times',size=28)
        GRadio_642["font"] = ft
        GRadio_642["fg"] = "#333333"
```



```

GRadio_642["justify"] = "center"
GRadio_642["text"] = "Delay"
GRadio_642["relief"] = "ridge"
GRadio_642.place(x=200,y=0,width=200,height=400)
GRadio_642["value"] = "90"
GRadio_642["command"] = self.GRadio_642_command

GRadio_65=tk.Radiobutton(root)
GRadio_65["anchor"] = "center"
GRadio_65["bg"] = "#f9f906"
ft = tkFont.Font(family='Times',size=28)
GRadio_65["font"] = ft
GRadio_65["fg"] = "#333333"
GRadio_65["justify"] = "center"
GRadio_65["text"] = "Reverb"
GRadio_65["relief"] = "ridge"
GRadio_65.place(x=600,y=0,width=200,height=400)
GRadio_65["value"] = "80"
GRadio_65["command"] = self.GRadio_65_command

GRadio_331=tk.Radiobutton(root)
GRadio_331["bg"] = "#1691ef"
ft = tkFont.Font(family='Times',size=28)
GRadio_331["font"] = ft
GRadio_331["fg"] = "#333333"
GRadio_331["justify"] = "center"
GRadio_331["text"] = "Chorus"
GRadio_331["relief"] = "ridge"
GRadio_331.place(x=400,y=0,width=200,height=400)
GRadio_331["value"] = "20"
GRadio_331["command"] = self.GRadio_331_command
def GRadio_825_command(self):
    print("command")
def GRadio_642_command(self):
    print("command")
def GRadio_65_command(self):
    print("command")
def GRadio_331_command(self):
    print("command")
if __name__ == "__main__":
    root = tk.Tk()
    app = App(root)
    root.mainloop()

```

Group Members



Jose Carlo Doliner

- Bachelor of Science Student in Electrical and Computer Engineering Technology with Computer Systems concentration (current GPA of 4.0)
- Dean's List recognition holder since Fall 2019 at Valencia College.
- Passionate about Digital Audio Design and the Music Industry.
- Current R&D intern at Siemens Energy, supporting the development of software analysis tools of large gas turbine components.



Christopher Collins

- Bachelor of Science Student in Electrical and Computer Engineering Technology concentrating in Computer Systems (current GPA of 3.52)
- Currently focused on Computer and Network Engineering
- Currently a Launch Control System Computer Engineer Intern at NASA. Working to support the development and testing of the Spaceport Command and Control System in the Launch Control Center.
- Six-time President and two-time Dean's list recognition