University of Maine

ECE403 Capstone Report

Guitar Amplifier with Programmable Effects

Author:
Jacob Allenwood

Author:
Travis Russell

May, 2016

Abstract
The Guitar Amplifier with Programmable Effects (GAPE) Unit allows a guitarist to combine digital effects with vacuum tube amplification. The user has the option to select from three effects: delay, compression, or equalization, by means of a Graphical User Interface (GUI). A vacuum tube amplifier acts as the output stage of the effects suite, allowing the user to hear the digitally manipulated guitar signal, while also adding subtle amounts of harmonic distortion. This device gives the guitarist the desired customize-ability to modify the guitar sound to his or her liking, while maintaining the desired tonal qualities of the historically favored vacuum-tube amplifier.

Contents

List of Figures

List of Tables

1 Introduction

This report describes the design of a digital audio effects unit with vacuum-tube amplification. Electric guitars require sound amplification to work as designed. Solid state amplifiers are designed with solid state electronics (diodes, transistors, etc.), and on paper and in theory should yield identical results as vacuum-tube amplifiers, but in reality they fall short in terms of sound quality and appeal [1]. Guitarists also desire more control over their sound than what is offered with just a guitar and amplifier. An effects pedal is widely used to accommodate this desire, and many times uses analog electronic components to achieve effects such as a compressor or delay. This desire for extra functionality in a guitar amplifier, and both author's interest in audio and music, led to the design of the Guitar Amplifier with Programmable Effects (GAPE) Unit. The GAPE allows a user to apply either a delay, compressor, or equalizer digital guitar effect on the signal and then amplify that signal with a traditional vacuum-tube amplifier. This device gives the guitarist the desired customize-ability to modify the guitar sound with an effect, while maintaining the desired tonal qualities of a vacuum-tube amplifier.

Fractal Audio has made a product called the Axe-Fx which has been an inspiration for this project and is similar in certain aspects. The Axe-Fx uses Digital Signal Processing to model different vacuum-tube amplifiers. It gives the user both the ability to manipulate the guitar's sound with digital effects, and a very high quality replication of a vacuum-tube amplifier's sound. The digital effects found on the GAPE Unit fall short compared to the Axe-Fx in many ways, mainly because of resource constraints. The major difference between the Axe-Fx and the GAPE Unit, is that the majority of the software for the Axe-Fx is written to mimic the hardware the GAPE Unit uses.

Because of the clear distinction between the digital and analog portions of the project, the specifications were developed for each portion, rather than in terms of the overall project.

The digital effects specifications include a function to delay the guitar signal by at least 0.5 seconds. A 3 band equalizer with bands: 80Hz to 300Hz, 400Hz to 1000Hz, and 1100Hz to 5000Hz, will be able to boost or attenuate by at least 10dB. Transition bands will have less than a 2dB attenuation when all bands are set to 0dB. Finally for digital effects specifications; a compressor will attenuate the input if the Root Mean Square (RMS) value is above a user-defined threshold.

The pre-amplifier specifications include limiting the minimum gain to 5dB over the 20Hz to 20kHz band. The maximum total harmonic distortion (THD) is limited to 5% in the output signal, with a 1 kHz 500 mV peak to peak input signal. THD is measured as:

$$THD = \frac{\sqrt{V_2^2 + V_3^2 + V_4^2 + ...}}{\sqrt{V_1^2 + V_2^2 + V_3^2 + ...}}$$

where V_n is the RMS voltage of the n^{th} harmonic and $n = 1$ is the fundamental frequency.

The specifications were verified with the oscilloscope and/or network analyzer functions of the Analog Devices Digilent and are shown in Section 5.

An understanding of how an electric guitar works, and the background of amplifiers is helpful in understanding the purpose of the GAPE. Unlike an acoustic guitar, the body of an electric guitar is solid. The body of an acoustic guitar is hollow, which allows for the vibration of the strings to resonate within the body. This can easily be heard by the player and an audience in a small to medium sized room. An electric guitar on the other hand cannot be heard without the use of an amplifier. The vibration of the strings induces a relatively small voltage in the pickups (devices to 'pick up' the sound of the guitar strings) of the guitar, usually on the order of millivolts (a thousandth of a volt). This signal level is far too low in power to drive a speaker. An amplifier is used to increase the voltage level of the guitar signal from a few millivolts up to 40 to 100 volts. A transformer is then used to convert the high voltage, low current signal into a low voltage, high current signal that can easily drive a speaker. This allows the player and an audience to hear what is being played on the electric guitar. The amplifier also has the secondary purpose of "coloring" the sound of the guitar. The amplifier induces a small amount of distortion to the guitar signal that sounds pleasant to some. Vacuum tube amplifiers are often opted for instead of solid state amplifiers that are built with transistors. This is due to the fact that when transistors are used to add distortion to a signal, a large amount of high frequency content is added to the signal which sounds quite harsh and shrill to the ear [2]. Vacuum tube circuits on the other hand induce a very subtle distortion that can be described as warm or rich sounding.

Many guitarists use effects in front of their amplifier to further enhance the sound. Common effects include delays, compressors, and equalizers. The delay effect simply takes what is being played and repeats it back, similar to an echo. Compressors reduce the dynamic range of the signal. If a chord is played that is relatively high in volume, and a single note is played very lightly, they will sound the same in volume level after extreme compression. Equalizers give the player control over the frequency content of the guitar. For example, different guitar body woods sound drastically different from one another. A guitar made out of basswood usually has fairly "muddy" (unclear) low frequency content, so with an equalizer the player can simply reduce it to clean the signal up.

The rest of the document covers the breakdown/overview of the project with a short description of each module, the design details of the digital effects portion, the design details of the amplification portion, the results of the project, and a conclusion.

2 Breakdown

This section provides an overview of the GAPE in terms of inputs and outputs. Each functional block is broken down and explained in more detail, including the GUI, the Display, the General-Purpose Input/Output (GPIO) communication protocol, the Digital Signal Processing (DSP), the power supply, the pre-amplifier, and the power amplifier.

Figure 1 shows the overall signal flow of the GAPE Unit in operation. The input to the GAPE Unit is an analog electronic guitar signal (electric guitar or electronics on an acoustic guitar) and user-input to the GUI. The analog guitar signal is converted to a digital signal by the ADC (Analog to Digital Converter) on the Digital Signal Processor and is fed into the DSP routines to be manipulated; either to delay, compress, or equalize the signal. The guitar effect to run is determined by the user-input to the GUI, and then the GUI communicates to the DSP with a GPIO communication protocol. The GUI also controls a seven-segment display to display which effect was selected and the specific parameters for the effect. The digital guitar signal is then converted back to an analog signal and is fed into the pre-amplifier.

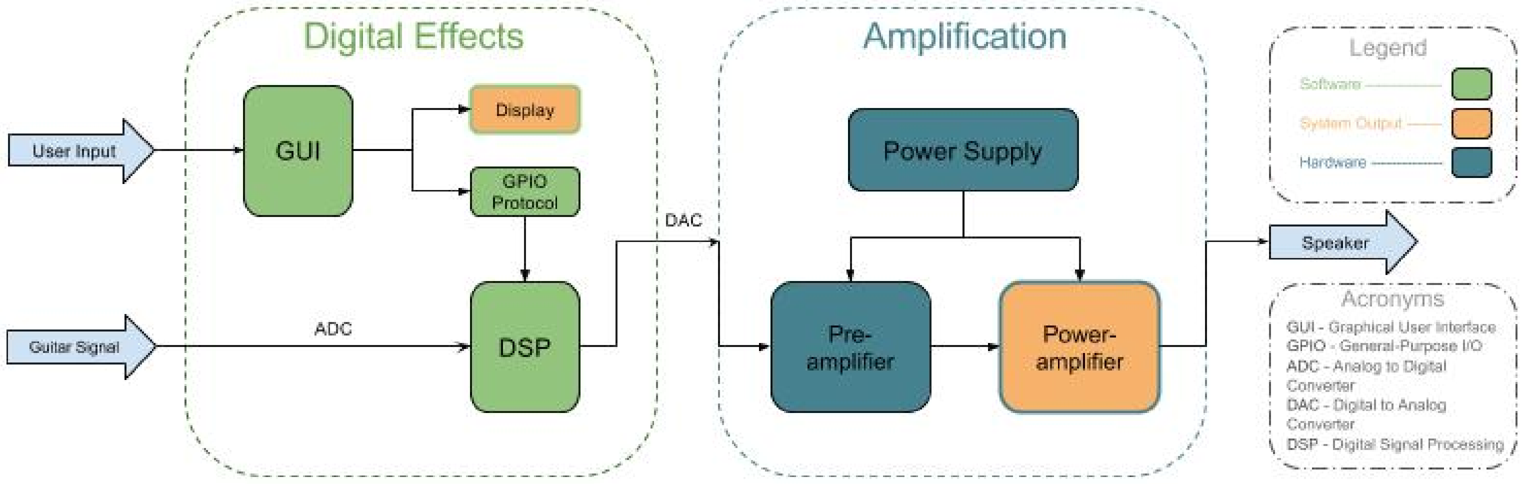


Figure 1: Signal Flow through the GAPE Unit

The vacuum tube pre-amplifier increases the signal level of the modified guitar signal. It also induces a small amount of harmonic distortion. This particular amplifier is what is known as a "clean amplifier". Only a very small amount of distortion is added to the signal. Musically, octaves are added to the original signal. If a 440 Hz tone (the note A), is played through the amplifier, 440 Hz will be heard on the output along with a subtle amount of 880 Hz. The word subtle here is used to describe the significant difference in magnitude between the fundamental frequency and its octave - a 440 Hz signal with a magnitude of 10 dB may have an 880 Hz signal added with a magnitude of -50 dB. The signal is then passed through the power amplifier which allows the signal to drive a 10W loudspeaker.

2.1 Digital Effects

This section breaks down each of the software blocks, including the GUI, the Display, the GPIO protocol, and the DSP. The digital/software portion of the project acts as the user-control, feedback, and calculation of the guitar effects, and the input to the overall system and amplification section. An important note, is that the software portion is split over two different embedded systems (computer with a specified purpose inside of a larger system). The GUI and the display both run on a Raspberry Pi Model 2, and the DSP runs on an STM32F407. The GPIO Protocol is part of both boards as a way for them to communicate. Section 3 will investigate the design details of the software portions.

2.1.1 GUI

The purpose of the Graphical User Interface is to allow the user to interface with the project, selecting which guitar effect to run on the guitar signal. Once the effect is selected with the parameters desired by the user, the GUI sends output to both the display block and the DSP block. This communication is done by having the GUI run two different C programs upon effect submission that set specific GPIO pins according to the selected effect and parameters. The GPIO protocol will be explained in depth in Section 3.1, but the most significant feature of the GUI is the transition between a user selecting an effect and the DSP knowing which effect to run. The way that this works, in terms of the GUI itself, is that the submit button is tied to many different functions that run once it is clicked.

The work-flow of this submit process is as follows:

1. Submit is clicked on the GUI
2. A check is made that an effect was selected, and that the parameter values are appropriate
3. A check is made to see if this is the first effect submission
4. Effect and parameter values are sent to both the DSP and Display programs

The first two checks are to make sure that values are selected, and that the values make sense for the DSP. If that check fails, then an error window appears and alerts the user to make sure to select appropriate effects and parameters. This blocks any bad input from entering the display and DSP. The next check is there to see if the GUI needs to halt the display and DSP programs that are already running. Finally, the parameter values are scaled to make them easier to transmit through the GPIO protocol (no negative numbers, etc.) and are sent along with the effect value as command line arguments to the display and DSP programs. Command line arguments are values passed to a program from the command line when running a program from the shell (user interface for a computer's operating system).

2.1.2 Display

The purpose of the display is to give constant user feedback on the effect and parameters currently running on the guitar signal. The display program accepts the effect and parameter values from the GUI and converts them into the hexadecimal values used by the 7-segment display. This program continuously writes the effect and then the effect parameters in an infinite loop, with 2.5 seconds of delay between switching values. This program works by comparing each value it received from the GUI with every digit from 0 to 9. When it finds a match to the digit, it then sets the hexadecimal value to represent that digit into the output buffer. Once all the values are matched accordingly, the output buffer is written to the display, which actually shows the numbers on the display.

2.1.3 GPIO Protocol

The GPIO protocol is the link between every piece of software. It is the language that each piece uses to understand the others. This is a custom and basic serial communication protocol.

Two major pieces to this protocol:

1. Setting the GPIO pins on the Raspberry Pi
2. Reading the GPIO pins on the STM32F407

The GPIO pins on the Pi are connected to the GPIO pins (exact pin setup is shown in Table 1) on the STM board with jumper cables; so when specific pins are set on the Pi, then the pins connected on the STM will also be set. The idea is if the STM board knows what it means when certain pins are set, then it can know what kind of values to run the DSP with. Figure 2 shows the communication process at a high level.

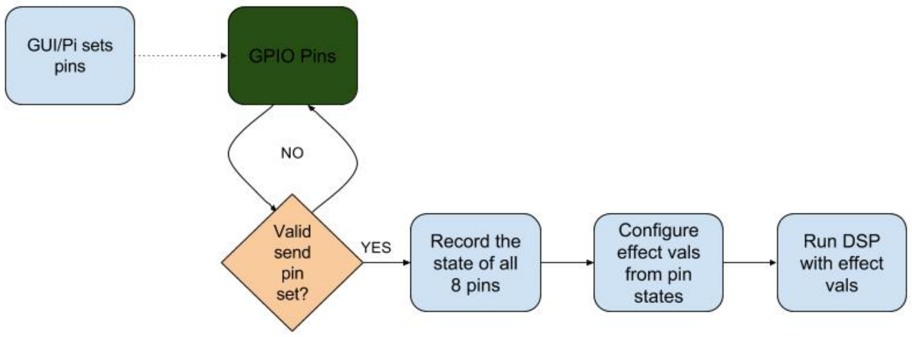


Figure 2: The process of the DSP handling GPIO data from the GUI/Raspberry Pi

When an effect is selected from the GUI and the GUI runs the GPIO-to-DSP program, it passes along the values from the GUI so that the program can set the GPIO pins accordingly. The DSP program waits until the 'valid send' pin is set high by the Pi, and then it knows that it has other data to read on the other pins. These other pins tell the DSP program which effect algorithms to run and with what parameters.

The details of this protocol are explained in Section 3.3.

2.1.4 DSP

The purpose of the Digital Signal Processing portion is to actually manipulate the guitar signal and apply the effect selected by the user. Three different algorithms are used to run the three different effects. The first delays the signal using a circular buffer. The second compresses the signal by scaling down the amplitude if it breaches an Root-Mean-Square (RMS, which is a type of averaging) threshold defined by the user. The third equalizes the signal by attenuating or boosting the signal between the 3 different frequency bands. Design details of these DSP algorithms are explored in Section 3.

The input to the DSP is the guitar signal converted to a digital signal with the Analog-to-Digital Converter (ADC). Once the signal goes through the DSP routines and has been manipulated to the user's liking, it then passes through the Digital-to-Analog Converter (DAC) to be converted back to an analog signal to proceed to the amplifier.

2.2 Vacuum Tube Amplifier

The purpose of the vacuum tube amplifier is to drive a loudspeaker so that the guitar may be heard.

2.2.1 Pre-amplifier

The purpose of the pre-amplifier is to provide a significant voltage gain to the guitar signal. The overall gain from the pre-amplifier is divided between two vacuum tube gain stages. In addition to amplifying the input signal, each stage also induces a subtle amount of distortion to the signal. Two types of distortion are introduced: harmonic distortion and intermodulation distortion.

Harmonic distortion is induced by clipping, which adds a subtle amount of second harmonic distortion to the signal. This is musically equivalent to adding octaves to the fundamental frequencies of the input signal. If for example a 440 Hz note is played through the amplifier, the output signal will consist of the 440 Hz tone, along with a subtle amount of 880 Hz.

Intermodulation distortion is induced by the nonlinear nature of vacuum tubes. If for example a 440 Hz note and a 660 Hz note are played simultaneously through the pre-amplifier, they will be mixed together. The sum and difference frequencies are created by this phenomena – 1100 Hz and 220 Hz, respectively. Musically, these frequencies relate to 440 Hz in an incredibly interesting way. The 1100 Hz tone is a major third away from 440 Hz, and the 220 Hz tone is a sub-octave of 440 Hz. This phenomena allows frequencies to be generated that did not originally exist in the input signal.

2.2.2 Power Supply

The purpose of the power supply is to supply the required power to the pre-amplifier and power amplifier. Vacuum tubes require high voltages, which in this case is 300 V. Vacuum tubes also require a large current to be passed through a heater, which heats the cathode in order to allow electrons to boil off more easily.

2.2.3 Power Amplifier

The purpose of the power amplifier section is to drive the loudspeaker, so that the guitar signal can be audibly heard. A loudspeaker has a fairly low impedance, which in this case is 8 Ω. The high voltage, low current signal from the pre-amplifier is converted into a high current, low voltage signal by means of a transformer, which drives the loudspeaker.

This particular power amplifier operates in Class A mode, meaning that it is always on. This is the least efficient type of power amplification, with an efficiency of around 50% to 70%. It was chosen over a more efficient class for the reason that it preserves harmonic distortion. Class B amplifiers, commonly known as push-pull amplifiers, have a tendency to cancel even harmonic distortion generated by the pre-amplifier. A particularly rich sound was sought after, so a Class A stage was chosen.

3 Digital Effects/Software Design

This section explains the design decisions and details for designing the GUI, the Display program, the GPIO serial protocol, and the DSP algorithms.

The extent of the software design involved in this project was completely new in almost every aspect; from developing custom DSP algorithms, to creating a custom serial GPIO communication, to making a GUI to control multiple distinct software elements, to interfacing each large module together to create a cohesive digital design.

Although C code was used to write multiple programs that did very different things, some design decisions were made for all C code programs in terms of data representation, prototype and implementation separation, variable scoping, and program flow.

Function prototypes and definitions were kept separated using '.h' (header) and '.c' files respectively. Each '.c' file included the appropriate '.h' files if they were using the functions defined in that '.h' file. This separation helps keep each part of the overall design clear, and to keep the main program as uncluttered and readable as possible.

With DSP especially, variable typing is significant and presents a balancing act between accuracy of the sample values/minimization of rounding errors, and memory efficiency of the program.

Data that needed to be passed around within programs was kept in data structures, which kept each function call straight-forward with only a pointer to the structure being returned if the function manipulated or created any data. With data structured in such a way, it made it possible to keep all variables' scope within functions, and avoid using global variables (in most cases avoiding globals is considered best practice [3]).
The basic program flow was simple:

1. Initialize the data structures with the required data buffers and/or variables needed for the calculation routine.
2. Pass a pointer to the data structure to the calculation routine to perform the data calculation.
3. Free the memory allocated in the initialization routine if possible. Many times this was not possible as most of the programs were meant to run on a continuous basis, so the data was constantly being used for the next calculation.

3.1 Graphical User Interface Design

This section describes the design decisions and details concerning the GUI development. The GUI acts as the human-to-GAPE interface; it is the 'face' of the GAPE, and the way that a user can communicate with it.

Designing a good UX (User eXperience [4]) is as much an art-form as a science. Admittedly, because the GUI was just one part of a huge project, the GUI design was weighted much more heavily on getting it to work well than making it a beautiful and flawless UX. That being said, the design (written in Python), needed to include some core functionality:

- Represent the effect options and parameters in a logical way
- Accept user input to control the guitar sound
- Check for invalid inputs to make sure nothing that could break the DSP would be passed along (and give the user feedback if they attempted to submit such an input)
- Allow continuous input (change the effect/parameters at any time and re-submit)
- Interface with the Display and DSP programs by passing the selected values

To handle the functionality of representing the options available to the user and accepting input from the user, the interface was designed with buttons. Once the effect is selected, the available options for that are effect are then displayed on the screen. When a different effect is selected, the options from the previous effect are 'deleted' off the screen, and the options for the newly selected effect are displayed. The submit button then passes the selected values onto the Display and DSP programs, after being checked for invalid input. Figure 3 shows a screen shot of the GUI when the 'Delay' effect is selected and the 'Small Room' preset is selected.

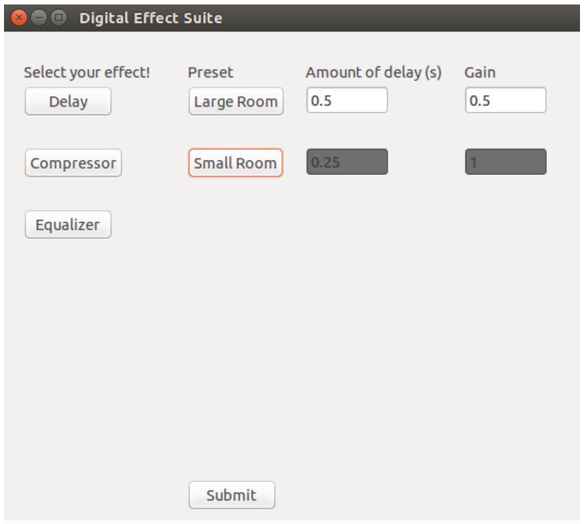


Figure 3: Screen shot of the GUI with the 'Delay' effect selected and 'Small Room' preset selected.

Upon clicking the 'submit' button, the DSP portion would recieve the parameters to run the 'Delay' effect with a delay time of 0.25 seconds and a gain of 1 given the selections shown in Figure 3.

The application was designed with pre-defined selections of the digital effects and parameters to fit with the GPIO protocol. Details of this protocol will be explained in Section 3.3. The idea is similar to the presets found in a car radio. One may see a 'Bass Boost' preset in a radio, which would be selected to increase the gain of the low-end frequencies. 'Bass Boost' is also a preset found in the GAPE, and is available to the user once the 'Equalizer' effect button is selected.

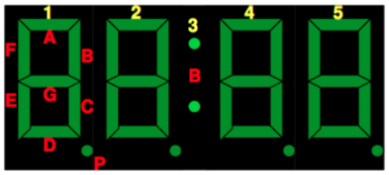
3.2 Display Design

The Display program was designed to take input from the GUI that communicated which effect was selected and what values went along with the effect, and display them on a 7-segment display. The idea of having a smaller and separate feedback mechanism in the form of this display rather than just using the GUI, was the possibility of being able to close the device with the GUI (whether it was an iPhone app, or desktop app, or web app etc.) and have the display close to the guitarist, or possibly even on the guitar itself.

The flow of the Display program is as follows:

1. Initialize I2C protocol (serial communication protocol [5]) to 'talk' to the 7-segment display
2. Initialize the display settings for the 7-segment display to function properly
3. Clear the display segments if the GUI is about to shut down
4. Read through the input values from the GUI and set up the appropriate buffers to display those values
5. Convert the effect and parameter values to the corresponding hexadecimal values used to light up the correct segments on the display
6. Until the GUI is shut down, continuously loop through and display the effect name and effect parameters

Figure 4 [6] shows the value mapping to light up the segments of the display.



byte 0 = 0x00 (display pointer offset)
byte 1 = (1P, 1G ,1F, 1E, 1D, 1C, 1B, 1A)
byte 2 = 0x00
byte 3 = (2P, 2G, 2F, 2E, 2D, 2C, 2B, 2A)
byte 4 = 0x00
byte 5 = (X, X, X, X, X, X, X, 3:, X)
byte 6 = 0x00
byte 7 = (4P, 4G, 4F, 4E, 4D, 4C, 4B, 4A)
byte 8 = 0x00
byte 9 = (5P, 5G, 5F, 5E, 5D, 5C, 5B, 5A)
byte10–byte16 = 0x00

Figure 4: Value mapping to control the 7-segment display

For instance; to light up the display with the 'd' in the first slot, a '01011110' should be written as 'byte 1' to the display. This lights up sections B, C, D, E, and G on the first character, which creates a 'd'. This logic was applied to create a binary representation of every number (1-9) to display the parameter values and every letter needed to display the guitar effect names.

One of the biggest design challenges in terms of actually displaying the values was being able to handle different numbers of parameter values correctly. When the Delay or Compressor is selected, the display will show the effect name (DLAY or COMP) and then show the two values associated with the effect (delay time and delay gain, or threshold and ratio, respectively). When the Equalizer is selected, it has three values associated with it, so the logic of looping through each parameter for any effect is made slightly more complex. Also related, was displaying not only each value for each effect, but displaying each character to make up each value. Basically what resulted was some keeping track of multiple indexes and incrementing or decrementing based on certain states.

3.3 GPIO Communication Protocol Design

This section describes the design decisions and details of the GPIO communication protocol. The purpose of the protocol is to relay the effect and parameter information from the GUI to the DSP. This is done by the GUI setting GPIO pins specific to the selected effect and parameters, and the DSP reading the pin states to determine the effect and parameters.

Eight GPIO pins on each embedded board are used for the communication protocol. One pin on each board is used to signify that valid data are being transmitted. Six pins on each board are used to represent which effect preset was selected. Finally, the last two pins represent which effect was selected. Table 1 shows the GPIO mapping between the Pi and the STM, and depicts what the function is for each set of pins, what preset the states represent, and what effect they represent.

Table 1: *GPIO protocol mapping between the RPi and STM*

	Pin Function:	Valid Send	Preset Select	Preset Select	Preset Select	Preset Select	Preset Select	Preset Select	Effect Select	Effect Select
	RPi:	GPIO25	GPIO21	GPIO20	GPIO16	GPIO26	GPIO19	GPIO13	GPIO6	GPIO5
	STM:	PB3	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
Preset Number	**Effect**									
1	Delay	1	0	0	0	0	0	1	0	1
2	Delay	1	0	0	0	0	1	0	0	1
3	Comp	1	0	0	0	0	0	1	1	0
4	Comp	1	0	0	0	0	1	0	1	0
5	EQ	1	0	0	0	0	0	1	1	1
6	EQ	1	0	0	0	0	1	0	1	1
7	EQ	1	0	0	0	1	0	0	1	1
8	EQ	1	0	0	1	0	0	0	1	1
9	EQ	1	0	1	0	0	0	0	1	1
10	EQ	1	1	0	0	0	0	0	1	1
11	EQ	1	0	0	0	0	1	1	1	1

The idea is to treat these pin statuses as a set of binary numbers, one that signifies a valid send, one that represents the effect, and one that represents the selected preset. For instance: Two pins (GPIO6/PD1 and GPIO5/PD0 for the Raspberry Pi/STM32F407) that act as the effect selection pins. A 1 (01 in binary) means that the effect is a delay. GPIO6/PD1 is low, or 0, and GPIO5/PD0 is high, or 1. Those two pins are sectioned off as the 'Effect Select' pin function. The rest of the pins, except for the 'Valid Send' pin (GPIO25/PB3), are all used to communicate which effect preset was sent.

An example work-flow should solidify the concept:

1. The DSP code on the STM board waits until the 'Valid Send' pin is high, or 1. When the Raspberry Pi sets this pin high, it is telling the STM board that data is available to be read off of the other pins. This ensures that the DSP does not try and run incorrect guitar effects parameters. This pin only gets set high once the user has submitted a valid effect and parameter selection using the GUI
2. The user selects the 'EQ' effect with the 'Bass Boost' preset. This in turn sets the 'Valid Send' pin high, and sets pins GPIO5/PD0, GPIO6/PD1 and GPIO13/PD3 high
3. Now the STM board reads all the other pins and finds the three pins just mentioned all turned high
4. Each preset has predefined values associated with the effect's available parameters. From Table 1 one can see that the STM board knows that the user selected the 'EQ' effect and preset 5, which is the 'Bass Boost' preset. In this case, the DSP now knows to run the EQ algorithm with the 'Bass Boost' parameters, which is a boost of 10dB to the low frequency band, and 0dB to the mid and treble frequency bands.

That is the work-flow from a user selecting an effect with the GUI, and the DSP running the correct algorithm with the correct parameters or effect values.

3.4 Digital Signal Processing Algorithm Design

The basic idea of DSP is to convert real-life continuous-time signals into digital discrete-time signals to be mathematically manipulated and used for another purpose. This is done by 'sampling' (taking a finite number of snapshots of the signal) a continuous-time signal and representing it with '1's and '0's [9].

The limited memory available on the board made it impossible to be able to chain effects together (run multiple effects concurrently) and it made the equalizer design especially difficult because the number of coefficients (values to manipulate the frequency content) available was limited by the memory available. The limited number of filter coefficients made it much harder to meet the stop-band specifications for the equalizer. This will be detailed further in Sections 3.4.3.

The basic flow of the DSP main program is as follows:

1. Initialize clock, timers, etc.
2. Wait for the 'valid send' pin to be set, meaning there is now effect and parameter information to be read
3. Read and update the effect and parameter values for the DSP
4. Initialize the ADC and DAC for streaming input and output data
5. Initialize variables for calculations and input/output buffers
6. Initialize FIR filter for low-pass filtering the guitar signal out to 10kHz (high end of the frequency range of guitars)
7. Initialize the selected effect (set up data structures, etc)
8. Continuously:
    - Get a block of input samples
    - Run an FIR low-pass out to 10kHz on the block
    - Run the selected effect calculation

In the overall project, error checking was a large part of the design, and no exception was made with the DSP. Anywhere memory is allocated, it is checked to be successful before trying to use it. Anywhere variables being used for a calculation, checking is done to make sure the values are valid for the effect. In order to restrict bad code running on the board, or confusion for the user; if an error is caught, then the error (red) LED is set on the board, and the code runs in an infinite loop.

3.4.1 Delay Effect

The delay was designed with a circular buffer, which stores previous samples in a buffer to retrieve at a later time (the desired amount of delay time).

In order to convert the desired amount of time delay into the amount of discrete samples to delay by, Equation 1 is used.

$$N = F_s * t \tag{1}$$

This states that the number of samples (N) is equivalent to the sampling frequency (F_s is 48kHz) multiplied by the amount of time (t) in seconds. This makes sense when one realizes that the sampling frequency is how many samples are taken in one second.

An example of a circular buffer is shown in Figure 5.

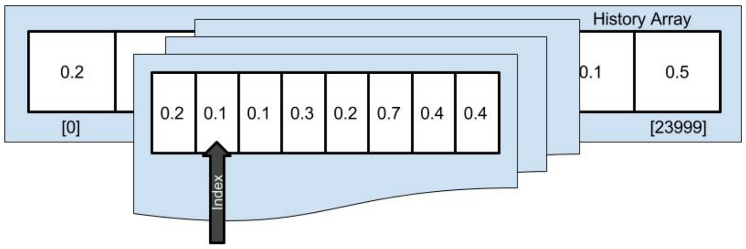
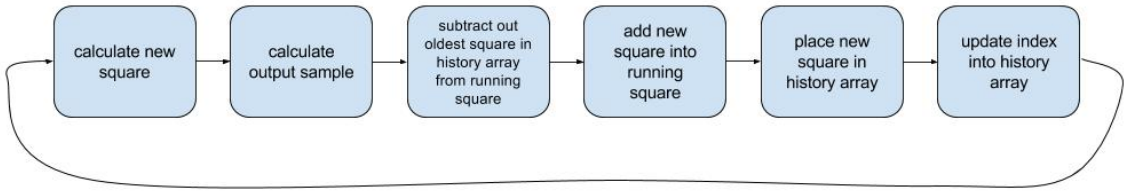


Figure 5: Visualization of how the circular buffer works with the Delay algorithm.

Given Equation 1, it is apparent how receiving the desired amount of delay time from the user/GUI can lead to figuring out how many samples long the history array should be (array holding the previous N amount of samples). Figure 5 shows an example history array full of floating point values that represent the sampled guitar signal. As shown, an index into the history array moves along to retrieve a value from the array, replace the old sample in the array with the current input sample, and then move on to the next spot. With the history array containing the amount of samples needed to delay by the user specified amount of time (N from Equation 1); by the time the index comes around to any given spot in the array (once the entire array is filled with input values initially), the value in that spot will be from t seconds ago. Pulling these old samples from the history array and outputting them results in a time delay of t seconds.

3.4.2 Compressor

The compressor was designed to attenuate the gain of the input signal (or 'compress' the dynamic range of the signal) if the Root-Mean-Square (RMS, a common averaging value found in audio compression) value surpassed a user defined threshold value (specific gain value limit). Obviously, an algorithm to detect the RMS value of the signal was an important part of the design. Figure 6 shows the flow and design of the RMS algorithm. The RMS routine takes a parameter $window\_size$, which is the number of samples to average over. The number of samples to average over is a significant part of the design, as a window too small will not fully represent a signal correctly, but a window too large introduces a large amount of delay in the compressor (which sometimes is desired, but should be controlled with the attack and release parameters [11]). For a guitar, the longest/slowest note is about 80Hz. The period, (how long the note is) is 0.0125 seconds. Using Equation 1, this leads to a sample amount of six-hundred to represent the 80Hz note. This means that in order for the RMS algorithm to correctly average over any guitar note input, the window should be at least six-hundred samples long.



calculate new square
calculate output sample
subtract out oldest square in history array from running square
add new square into running square
place new square in history array
update index into history array

Figure 6: Flow and design of RMS algorithm.

The RMS routine uses a running square so that the entire RMS (square root of the mean of the squares) does not need to be calculated every time for every new sample (meaning summing up all the squares from all of the previous input values back to *window_size* values away, which would be re-calculating all of the same values except the new one); rather there is always a running total of the squares so for each output value, only one new RMS value needs to be calculated.

After the RMS is calculated, varying levels of compression can be achieved with different complexities of compression algorithms [11]. The GAPE's algorithm for compression is very simple, as it just scales down the input value if the RMS value breaches the threshold level input by the user.

3.4.3 Equalizer

An equalizer (EQ, like the one found in a car radio) is used to boost or attenuate specific frequency bands. One may find an equalizer with presets, such as 'Bass Boost', which boosts the bass frequencies of the audio signal. The GAPE design splits up the guitar range into 3 different frequency bands; the low, or bass frequency band (80Hz - 300Hz), the mid frequency band (400Hz - 1kHz), and the high, or treble frequency band (1.1kHz - 5kHz).

The EQ design with FIR filtering is sometimes referred to as "perfect reconstruction" because when the filter outputs are reconstructed for the system output, virtually no distortion is added into the signal. It is important to keep the signal as clean as possible until distortion is actually desired, like in the case of the distortion setting on the vacuum-tube amplifier.

Figure 7 shows an example of the relationship between phase shift and frequency with both an IIR and FIR filter [12].

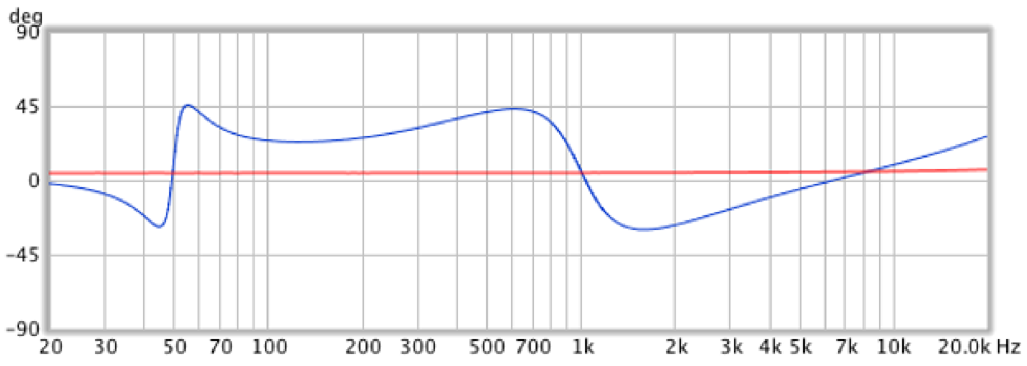


Figure 7: The phase shift related to frequency for an FIR (red) and IIR (blue) filter with nearly identical magnitude responses.

As seen, the FIR filter (red) has a constant shift no matter the frequency, as opposed to the IIR filter (blue). IIR filters have unpredictable behavior of the phase shift (which is like the discrete-time equivalent to continuous-time delay) through the filter [15]. The delay of the IIR filter changes with frequency, rather than having a constant delay (linear-phase) for every frequency like FIR filters. This causes a distortion in the magnitude response of the signal [16], where amplitudes at certain frequencies will be boosted and others will be attenuated because of the different shifts in the delay. This is a significant reason why the GAPE's EQ was realized with FIR filters, to minimize this shift, in turn minimizing undesired distortion in the output signal.

The basic idea of the equalizer algorithm is to use FIR filters to control different frequency bands, and use the delay algorithm (Section 3.4.1) to keep the input signal in phase with each filter output. This combination of filters and delays makes up the EQ design, as shown in Figure 8.

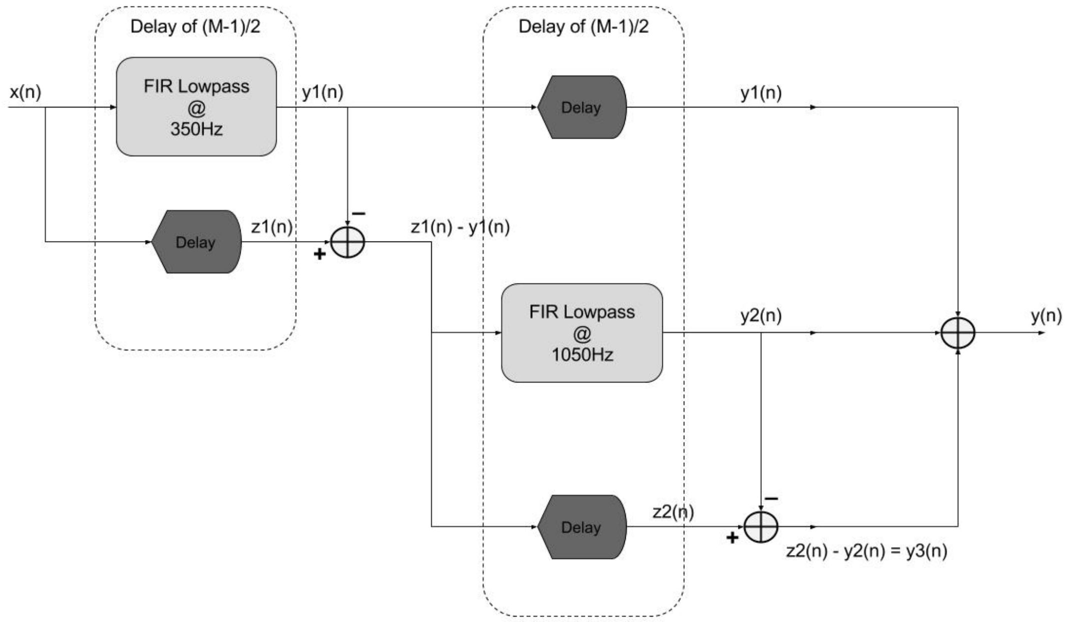


Figure 8: EQ algorithm block diagram, showing the relationship between an input sample and the output sample

FIR filters inherently apply a delay to the guitar signal, which can be calculated by $D = \frac{M-1}{2}$, where M is the number of filter coefficients. The delay blocks themselves also delay the input signal the exact same amount as the FIR filters, to keep each signal in phase with each other so they can be added or subtracted, to achieve the desired band. For instance, the diagram shows that to achieve the mid band of the equalizer, the input is filtered with a cutoff frequency (the frequency at which the input is 'cutoff', meaning the signal is virtually non-existent beyond the cutoff frequency) of 350Hz (this output is the low frequency band). That filter output is then subtracted from the delayed input signal to get the rest of the frequency band that was not filtered. With each filter, the output is the frequency band below the cutoff, and then the rest of the signal beyond the cutoff is retrieved by subtracting the (delayed, or in-phase) input by the filter output.

In the GAPE design, the equalizer is a 3-band equalizer. The 3 bands:

1. Low band; found by low-pass filtering the input out to 350Hz and then delaying to stay in phase for the final addition of each band output
2. Mid band; found by low-pass filtering the in-phase (delayed) input subtracted by the low band, out to 1050Hz
3. High band; found by subtracting the in-phase (delayed, again) signal going into the mid-band FIR filter, by the mid-band filter output (remember that the mid-band filter input is the in-phase input signal subtracted by the low-band filter output)

Since each band is separated and in-phase, then simply scaling the corresponding bands and adding them up will result in the 'equalized' guitar signal. The 'scaling' value is found by using $Gain = 10^{x/20}$, where x is the gain value in decibels, which is selected by the user.

4 Amplifier/Hardware Design

This section covers the design of the vacuum tube amplifier portion of the GAPE unit. Component selection and design choice for each subsection of the amplifier are explained. Note that this amplifier was designed using heuristic techniques used by most guitar amplifier designers, rather than the analytical methods used to design small-signal solid state amplifiers. Although the design approach is different, it is still valid and can be applied to any amplifier design.

4.1 Pre-amplifier Design

The design of the pre-amplifier began by first selecting a vacuum tube. The ECC83 triode was chosen since it is a common choice for many amp designers, and has an amplification factor of 100. The Philips datasheet provides various configurations for a common cathode gain stage, with gains ranging from 30.6 dB to 37.7 dB, which far exceeds the specification of 5 dB. Figure 9 shows the common cathode gain stage which is used widely in guitar amplifier design.

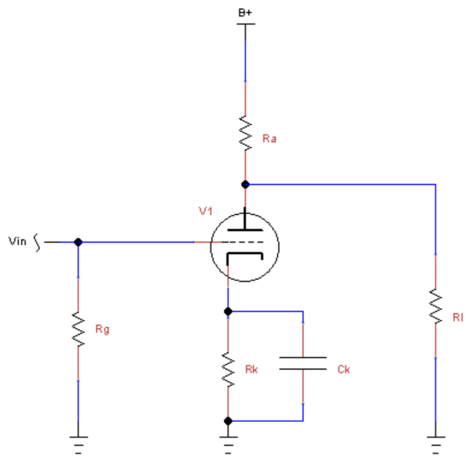


Figure 9: Common cathode gain stage

Unlike MOSFETs and BJTs, vacuum tubes do not have simple equations to relate the anode and grid voltages to the anode current. A graphical approach must therefore be used to select component values for the anode resistor, R_a, and the cathode resistor, R_k.

The maximum rated anode to cathode voltage for the ECC83 tube is 300V. With this in mind, a B+ voltage of 300 V was chosen. This is acceptable, even though the maximum voltage for the ECC83 tube is 300V, because the anode voltage will never actually swing up to this voltage in working operation.

Next, the anode resistor, R_a, was selected. Large anode resistances typically result in lower harmonic distortion, and since specifications call for 5% total harmonic distortion (THD) or under, an arbitrarily large standard resistor value of 220 kΩ was chosen. Figure 10 shows the grid curves for the ECC83 with a 220 kΩ load line drawn.

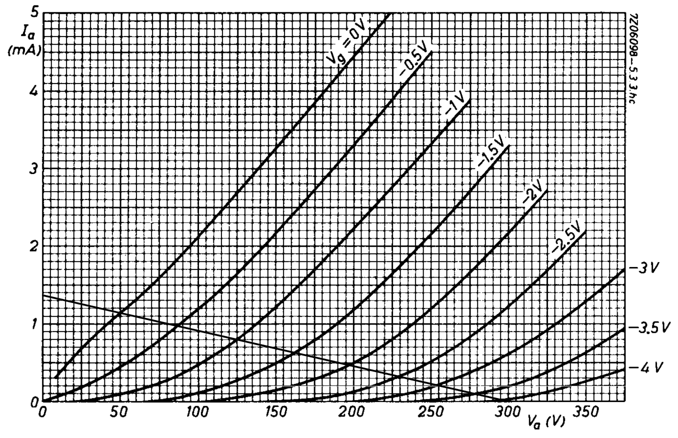


Figure 10: ECC83 grid curves with 220 k Ω load line

Each intersection point between the load line and the grid curves is a possible bias point for the tube. By using a cathode resistor R_k to self-bias the tube, the grid voltage, V_g, can be kept at 0 V with pull-down resistor R_g, while the cathode voltage, V_k, is raised to some positive voltage. For maximum headroom, the bias point was chosen to be $V_{gk} = -2V$. This corresponds to an anode current of 0.47 mA. The value of the cathode resistor, R_k, can then be computed using Ohm's Law.

$$R_k = \frac{2V}{0.47mA} = 4.3k\Omega \tag{2}$$

A cathode bypass capacitor was added in parallel with the cathode resistor in order to increase gain. The value was calculated by choosing an appropriate cut-off frequency for the high pass filter formed from the capacitor and the cathode resistor. To preserve all bass information in the guitar signal, the cut-off was chosen to be 10 Hz. Equation 3 was used to solve for a capacitor value of 3.7 μF. A standard value of 10 μF was used.

$$f_{3dB} = \frac{1}{2\pi R_k C_k} \tag{3}$$

Figure 11 shows the fully designed pre-amplifier. Capacitors C_2 and C_4 were added to filter the DC offset off of the anode of each tube. Potentiometers R_5 and R_9 were added to control the signal levels into the second gain stage and power amplifier, respectively.

Resistors R_2 and R_6 were added to control clipping on the positive half of the input waveform. As the grid voltage approaches the cathode voltage, some electrons begin to flow through the grid. This creates a forward current into the grid. By placing a resistor in series with the grid, a voltage drop is created, effectively dropping the grid voltage so that V_{gk} falls to a voltage less than 0 V. The larger this resistance is, the greater the voltage drop. A value of 10 kΩ was arbitrarily selected.

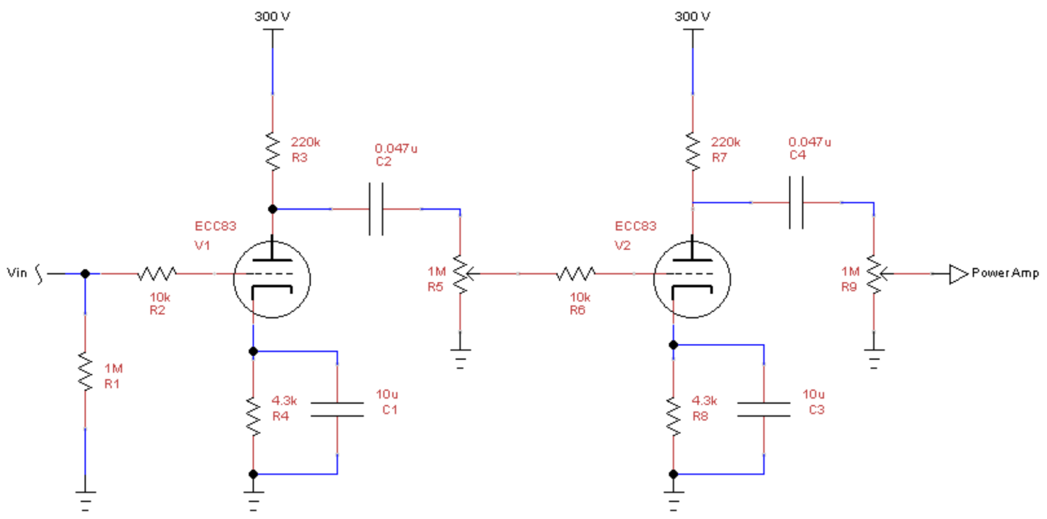


Figure 11: Pre-amplifier Schematic

4.2 Power Amplifier Design

A Class A power amplifier was opted for instead of a Class AB or Class B amplifier. This design choice was made with the desire to produce as much harmonic distortion from the power amplifier as possible. Class B amplification symmetrically amplifies the input signal, producing only odd-order harmonics. Class A amplification asymmetrically amplifies the input signal, producing both even-order and odd-order harmonics. A fuller sound can be achieved as a result, with the cost of efficiency.

The end goal was to supply 10 to 15W of power to an 8 Ω speaker. Since the output power of a Class A amplifier is approximately equal to one half of the anode dissipation, a tube that can handle 20 to 30W of power was needed. The EL34 pentode was chosen, as it can handle 25W of power and has a maximum quiescent anode voltage of 800V.

Figure 12 show a schematic of the fully designed Class A power amplifier.

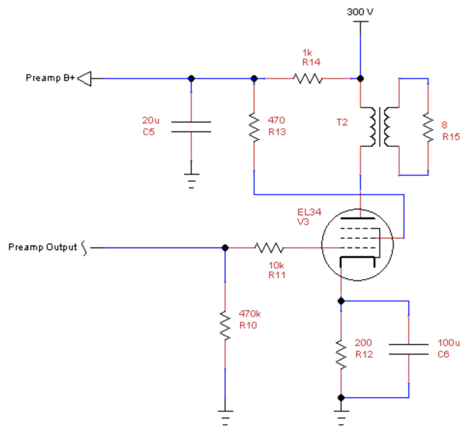


Figure 12: Class A Power Amplifier Schematic

The design began by first selecting an output transformer. The ideal primary input impedance can be approximated by

$$Z_{in} = \frac{V_a^2}{P_a}, \tag{4}$$

where V_a is the anode voltage, and P_a is the anode power dissipation. A V_a of 300V and a P_a of 25W results in an impedance of 3.6 kΩ.

A Hammond 1750B transformer was chosen, shown as T_2, which has a primary impedance of 5.5 kΩ for an 8 Ω load connected to the secondary.

The grid curves for the EL34 were extracted from the datasheet, and a 5.5 kΩ load line was drawn (shown in blue). This can be seen in Figure 13.

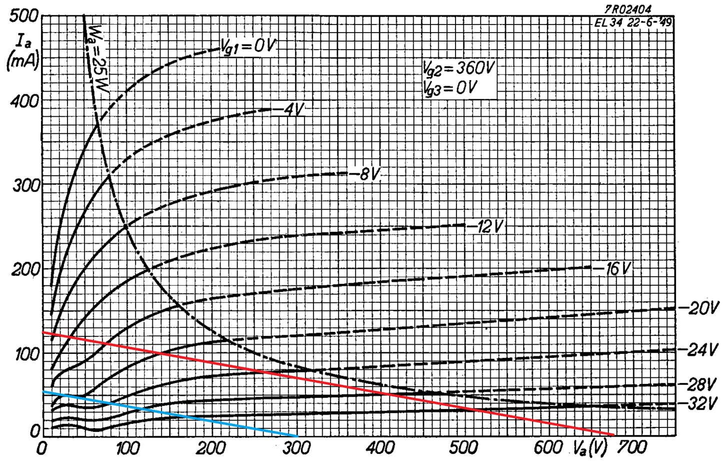


Figure 13: EL34 grid curves

Unlike a traditional resistive load, the primary of the transformer is seen as a short circuit under DC conditions. This results in the DC anode voltage equaling the supply voltage, which in this case is 300 V. This would result in the amplification of only half of the waveform (Class B amplification). To get around this, the slope of the load line is maintained while shifting the load line upwards. This is normally done by hand with a ruler, but for the purpose of generating a figure this was done in Microsoft Paint. This is shown as the red line in Figure 13.

A low-pass filter is formed from resistor R_{14} and capacitor C_5 to further reduce 60 Hz hum on the pre-amplifier rails. A arbitrarily small value of 1 kΩ was selected for resistor R_{14} to prevent the voltage drop across it from being too large. Capacitor C_5 was selected to be 20 μF, resulting in a cut-off frequency of 7.96 Hz using the equation

$$f_{3dB} = \frac{1}{2\pi RC}. \tag{5}$$

From the datasheet, the anode to screen current ratio is roughly 7 to 1. The quiescent anode current at 300 V is equal to 70 mA, as can be seen from the red load line in Figure 13. This results in a screen current of roughly 10 mA.

An additional resistor R_{13}, called a screen stopper, is placed in series with the screen to prevent the screen from burning up under clipping conditions. If the 0 V grid curve in Figure 13 were to move down to the position of the -8 V curve, the -8 V curve would approximately move down to the position of the -16 V curve.

The 0 V curve crosses the load line at approximately 110 mA. The red dot on the mutual characteristics graph shown in Figure 14 show the point where the grid voltage equals -8 V and the anode current is 110 mA.

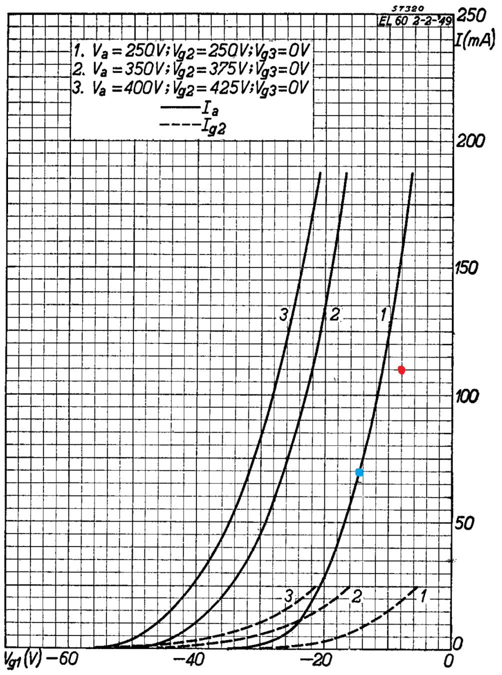


Figure 14: EL34 mutual characteristics

As can be seen, this point is fairly close to line 1 where the screen voltage is equal to 250 V. A 50 V drop from 300 V is therefore needed under peak conditions. Assuming 110 mA hits the screen under these conditions (just to be safe), the screen stopper can be calculated as

$$R_{13} = \frac{50V}{110mA} = 455\Omega. \tag{6}$$

A standard resistor value of 470 Ω is used. Assuming the pre-amplifier draws an additional 5 mA of current, the screen voltage will be

$$V_{g2} = 300V - (10mA)(470\Omega) - (15mA)(1k\Omega) = 280.3V. \tag{7}$$

Approximating the screen voltage as 250 V, the blue dot in Figure 14 shows the point where the quiescent anode current equals 70 mA. As can be seen, V_{g1} equals roughly -14 V. The cathode resistor R_{12} is calculated as

$$R_{12} = \frac{14V}{10mA + 70mA} = 175\Omega. \tag{8}$$

A standard resistor value of 200 Ω is used. A cathode bypass capacitor C_6 is added in parallel with the cathode resistor to increase gain. For a 10 Hz cut-off frequency, C_6 is calculated as

$$C_6 = \frac{1}{2\pi(10Hz)(200\Omega)} = 79\mu F. \tag{9}$$

A standard value of 100 μF is used. Resistor R_{11} is placed in series with the grid to form a low-pass filter with the internal capacitance of the grid to prevent high frequency oscillation. Resistor R_{10} is used to reference the grid to 0 V.

4.3 Power Supply Design

A power supply voltage of 300 V was chosen for the ECC83 triodes used in the pre-amplifier and the EL34 used in the power amplifier. Figure 15 shows the full design of the DC power supply.

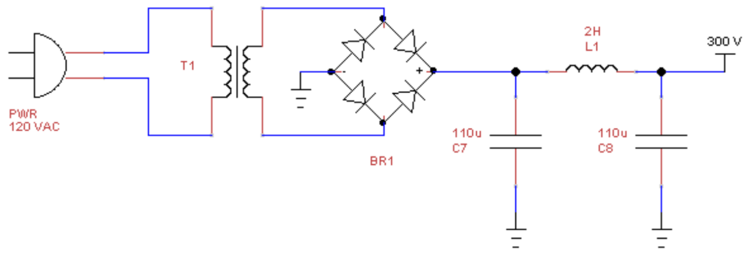


Figure 15: DC Power Supply Schematic

The design of the power supply began after the amplifier had been fully designed and tested. The standby currents of the amplifier were measured to be 78.5 mA for the power amplifier stage, and 10.2 mA for the screen current and pre-amplifier stages. This results in a total of 88.7 mA. The total power required can be computed by

$$P = IV \tag{10}$$

where P is power, I is current, and V is voltage. A current of 88.7 mA and a voltage of 300 V results in a required power of 26.61 W. The design began by first selecting the transformer, T_1. A peak voltage of close to 300 V was desired. A 1:1.83 step up transformer was chosen to step 120 V RMS to 220 V RMS, which roughly equals 311 V peak.

A bridge rectifier is then used to rectify the negative half cycles from the output of the transformer. A smoothing capacitor, shown as C_7 in Figure 15, was selected using

$$C = \frac{I}{2fV_{ripple}}, \tag{11}$$

where I is the current, f is the operating frequency, and V_{ripple} is the voltage ripple. Setting I as 88.7 mA, f as 60 Hz, and V_{ripple} as 5% of 300 V results in a capacitance of 49.3 μF. A standard capacitor value of 110 μF was selected. A second order low-pass filter consisting of inductor L_1 and capacitor C_8 was added to filter any excess 60 Hz hum. The equation

$$f_{3dB} = \frac{1}{2\pi\sqrt{LC}} \tag{12}$$

where f_{3db} is the cut-off frequency, C is the capacitance, and L is the inductance, is used to select the component values. A capacitor value of 110 μF and a cut-off frequency of 10 Hz results in an inductance value of 2 H. The secondary winding of a transformer was used to achieve this inductance.

5 Results

This section discusses the GAPE's test results regarding the contract specifications for both the hardware and the software portions. Overall, the specifications for both the digital portion and the analog portion were met, and exceeded by adding additional functionality and modules to the project.

5.1 Software

Each effect algorithm was tested with streaming input data and the actual results are compared with the desired results for the algorithm. These results include output for the Delay, Compressor, and Equalizer effects. All software measurements were taken with the Analog Discovery Digilent and exported to be graphed in Matlab.

5.1.1 Delay Algorithm Results

The contract stated that an effect would be able to delay the guitar signal by 0.5 seconds. Upon selecting *Delay* followed by *LargeRoom*, the DSP is told by the GUI to run the Delay effect with a delay time of 0.5 seconds. Figure 16 shows the measured output of this effect and preset selection.

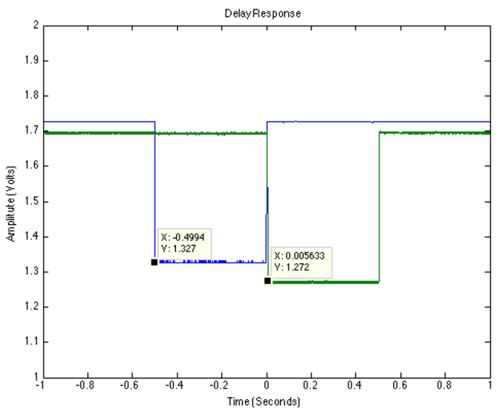


Figure 16: 0.5 second delay response for the Delay: Large Room preset

As shown, the Delay effect meets the specification of a 0.5 second delay.

5.1.2 Compressor Algorithm Results

The contract stated that the compressor would attenuate the signal if the RMS value of the input breached a user-defined threshold level. Upon selecting *Compressor* followed by $CoffeeShop$, the DSP is told by the GUI to attenuate the signal if the input breaches a level of -7dB. This value corresponds to an RMS value calculated by using $dB = 20log_{10}(RMS)$. A -7dB value corresponds to $10^{(-7/20)} = 446.7mV_{RMS}$. For a sine wave, the peak voltage value corresponds to the RMS value by $V_{RMS} = V_{peak}/\sqrt(2)$. As shown in Figure 17, the threshold is being breached between the 700mV and the 750mV inputs. 700mV corresponds to an RMS value of about 490mV, and the 750mV input corresponds to an RMS value of 530mV.

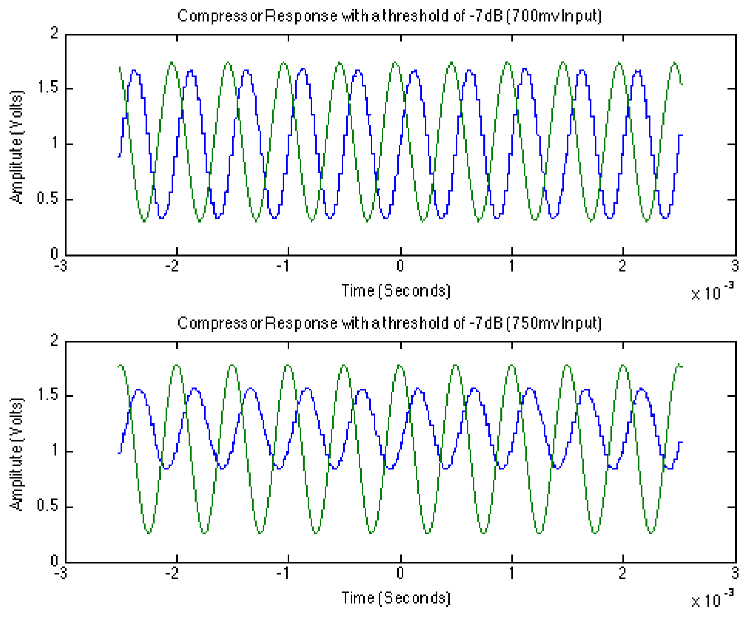


Figure 17: Compressor response given a $700mV_{peak}$ input, and a $750mV_{peak}$ with a threshold of -7dB.

It appears the RMS calculation algorithm is about 50mV off, which can be explained by both the size of the window of samples being averaged and the design of the RMS algorithm. When calculating the RMS value from the peak voltage of a sine wave, it is like having an infinitely large window to encapsulate the entire waveform. Having a finite number of samples in the RMS window with a finite sampling frequency will limit the accuracy of the RMS result. If the window is larger than the RMS calculation would be more accurate, but then the amount of time the compressor takes to attenuate the signal is larger. This is known as the attack, and should be controlled with an attack parameter, rather than by the window size. The design of the algorithm with a running RMS value calculated by adding new values in and subtracting old ones out, can eventually lead to rounding errors as the running average continues over a very large amount of calculations. The important thing to take away is that the compressor still attenuates the signal once it breaches the threshold given the RMS algorithm calculation, however the RMS calculation is not fully comparable to an ideal waveform calculation.

5.1.3 Equalizer Algorithm Results

The contract stated that a 3 band equalizer with bands: 80Hz to 300Hz, 400Hz to 1000Hz, and 1100Hz to 5000Hz would have each band able to boost or attenuate by at least 10dB. Transition bands were to have less than a 2dB attenuation when all bands are set to 0dB. Figure 18 shows the equalizer response when all bands are set to 0dB.

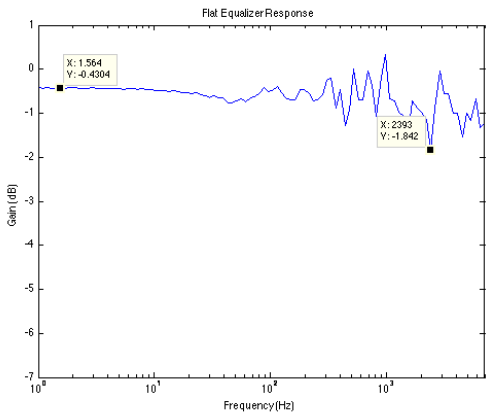


Figure 18: Equalizer response with all bands set to 0dB of boost and 0dB of attenuation.

As seen, the largest attenuation in the response is -1.2dB at about 2.2kHz, which is not even part of the transition band.

Figure 19 shows the equalizer response with the *BassBoost* preset selected, giving the frequency band of 80Hz to 300Hz a gain boost of about 10.6dB.

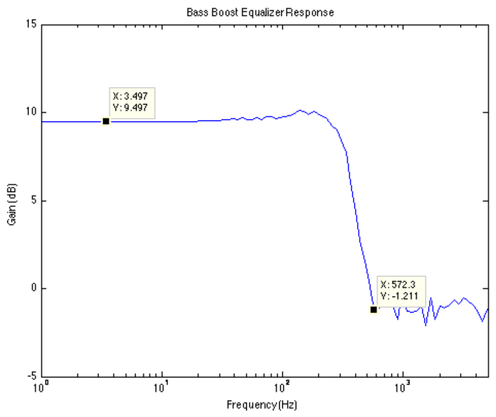


Figure 19: Equalizer response with the Bass Boost preset selected.

One final example of the equalizer response is found in Figure 20, which shows the attenuation of the treble frequency band (1.1kHz to 5kHz).

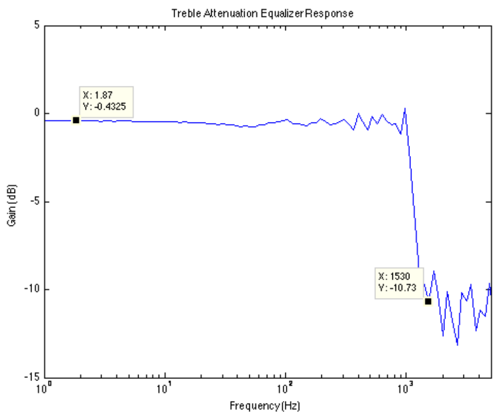


Figure 20: Equalizer response with the Treble Attenuation preset selected.

As shown, the equalizer met each specification, where the flat response would not attenuate more than 2dB in the transition bands, and that each frequency band would be able to boost or attenuate by 10dB.

5.2 Hardware

All hardware measurements were taken with an Analog Discovery Digilent. The total harmonic distortion of the pre=amplifier was measured using a spectrum analyzer. Figure 21 shows a measured THD of -62.68 dBc. This corresponds to 0.073%, which is well below the specified 5%.

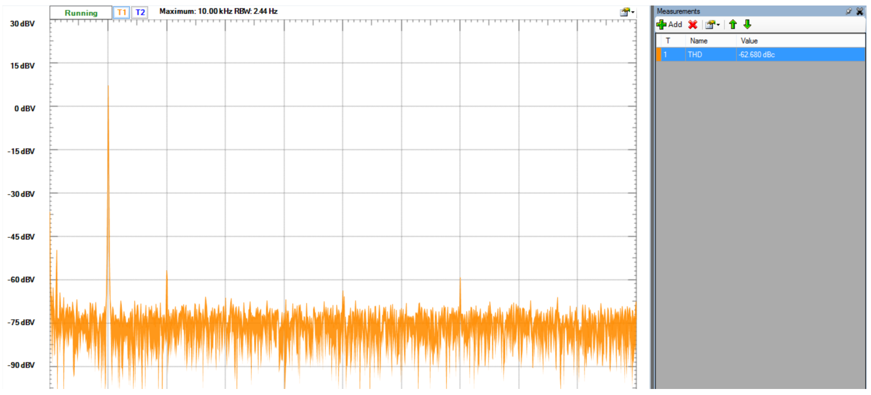


Figure 21: Total Harmonic Distortion

A network analyzer was used to determine the gain from 20 Hz to 20 kHz. Figure 22 shows the gain magnitude and phase response of the pre-amplifier. A mid-band gain of 23.2 dB was achieved, which is well above the specified 5 dB.

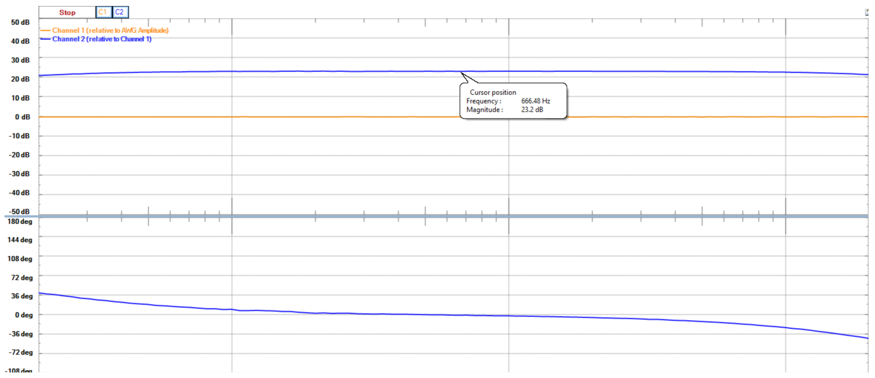


Figure 22: Pre-amplifier gain

6 Conclusion

The Guitar Amplifier with Programmable Effects (GAPE) Unit allows a user to apply either a delay, compressor, or equalizer digital guitar effect on a guitar signal with its selection using a Graphical User Interface (GUI), and then amplify the guitar signal with a vacuum-tube amplifier. This device gives the guitarist the desired customize-ability to modify the guitar sound with an effect, while maintaining the desired tonal qualities of the historically favored vacuum-tube amplifier.

The GAPE Unit was successfully designed and constructed to meet all contract specifications. The Delay effect delays the guitar signal by 0.5 seconds, the Compressor effect attenuates the guitar signal once the RMS value breaches a user-defined threshold value, and the Equalizer has less than a 2dB attenuation in the transition bands with a flat response selected, and also is able to boost or attenuate each of the three frequency bands by +/-10dB. The vacuum tube amplifier was measured to have a THD of 0.07%, with a mid-band gain of 23.2 dB.

7 References

[1] Dave Eichenberger, *CAGE MATCH: TUBES VS. SOLID STATE & MODELING* Available at `http://www.seymourduncan.com/blog/the-tone-garage/cage-match-tubes-vs-solid-state-modeling`.

[2] Wayne Storr, *Amplifier Distortion*
Available at `http://www.electronics-tutorials.ws/amplifier/amp_4.html`.

[3] Philip Koopman, *Better Embedded System Software (Ch. 19)* Available at `http://koopman.us`.

[4] Jacob Gube, *What is User Experience Design? Overview, Tools and Resources* Available at `https://www.smashingmagazine.com/2010/10/what-is-user-experience-design-overview-tools-and-resources/`.

[5] Jean-Marc Irazabal, Steve Blozis, *I2C Manual* Available at `http://www.nxp.com/documents`.

[6] Vincent Weaver, *ECE471: Embedded Systems – Homework 5* Available at `http://web.eece.maine.edu/ vweaver/classes/ece471_2015f/ece471_hw5.pdf`.

[7] Microchip Technology Inc, *Section 21. UART* Available at `http://ww1.microchip.com/downloads/en/DeviceDoc/39708B.pdf`.

[8] Kerry Redshaw, *Binary - So Simple a Computer Can Do It* Available at `http://www.kerryr.net/pioneers/binary.htm`.

[9] Analog Devices, Inc., *A Beginner's Guide to Digital Signal Processing* Available at `http://www.analog.com/en/design-center/landing-pages/001/beginners-guide-to-dsp.html`.

[10] Elena Punskaya, *Design of FIR Filters* Available at `http://www.vyssotski.ch/BasicsOfInstrumentation/SpikeSorting/Design_of_FIR_Filters.pdf`.

[11] DIMITRIOS GIANNOULIS, MICHAEL MASSBERG, JOSHUA D. REISS, *Digital Dynamic Range Compressor Design — A Tutorial and Analysis* Available at `https://www.eecs.qmul.ac.uk/ josh/documents/GiannoulisMassbergReiss-dynamicrangecompression-JAES2012.pdf`.

[12] miniDSP Ltd., *FIR vs IIR filtering* Available at `https://www.minidsp.com/applications/dsp-basics/fir-vs-iir-filtering`.

[13] Unknown Author, *Digital Filtering* Available at `http://users.abo.fi/htoivone/courses/sbappl/asp_chapter1.pdf`.

[14] Julius O. Smith III, *Filter Stability* Available at `https://ccrma.stanford.edu/j̃os/fp/Filter_Stability.html`.

[15] Massachusetts Institute of Technology Department of Electrical Engineering and Computer Science, *DT Filter Design: IIR Filters* Available at `http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-341-discrete-time-signal-processing-fall-2005/lecture-notes/lec08.pdf`.

[16] Iowa Hills Software, *What is Group Delay* Available at `http://iowahills.com/B1GroupDelay.html`.

8 Appendix A: Project Contract

Name of Project

GAPE (Guitar Amplifier with Programmable Effects)

Project Summary

This project proposes the design and construction of a vacuum tube preamplifier with a programmable digital effects unit. The output of an electric guitar will be manipulated digitally and fed into a preamplifier to add color to the sound.

Inputs and Outputs

IN - The input to the system will be the output of an electric guitar. Depending on the pickups, this signal can range from 100 mV to 500 mV RMS. The signal contains frequencies within the audible spectrum, 20 Hz to 20 kHz.
OUT - The output of the system will be a digitally processed signal with analog amplification.

Specifications

Digital Effects
- A function to delay the signal by at least 0.5 seconds.
- A 3 band equalizer with bands: 80 to 300 Hz, 400 Hz to 1 kHz, and 1.1 kHz to 5 kHz. Each band will be able to boost or attenuate by at least 10 dB. Transition bands will have less than a 2 dB attenuation when all bands are set to 0 dB.
- A compressor that will attenuate the input if the RMS value is above a user defined threshold.

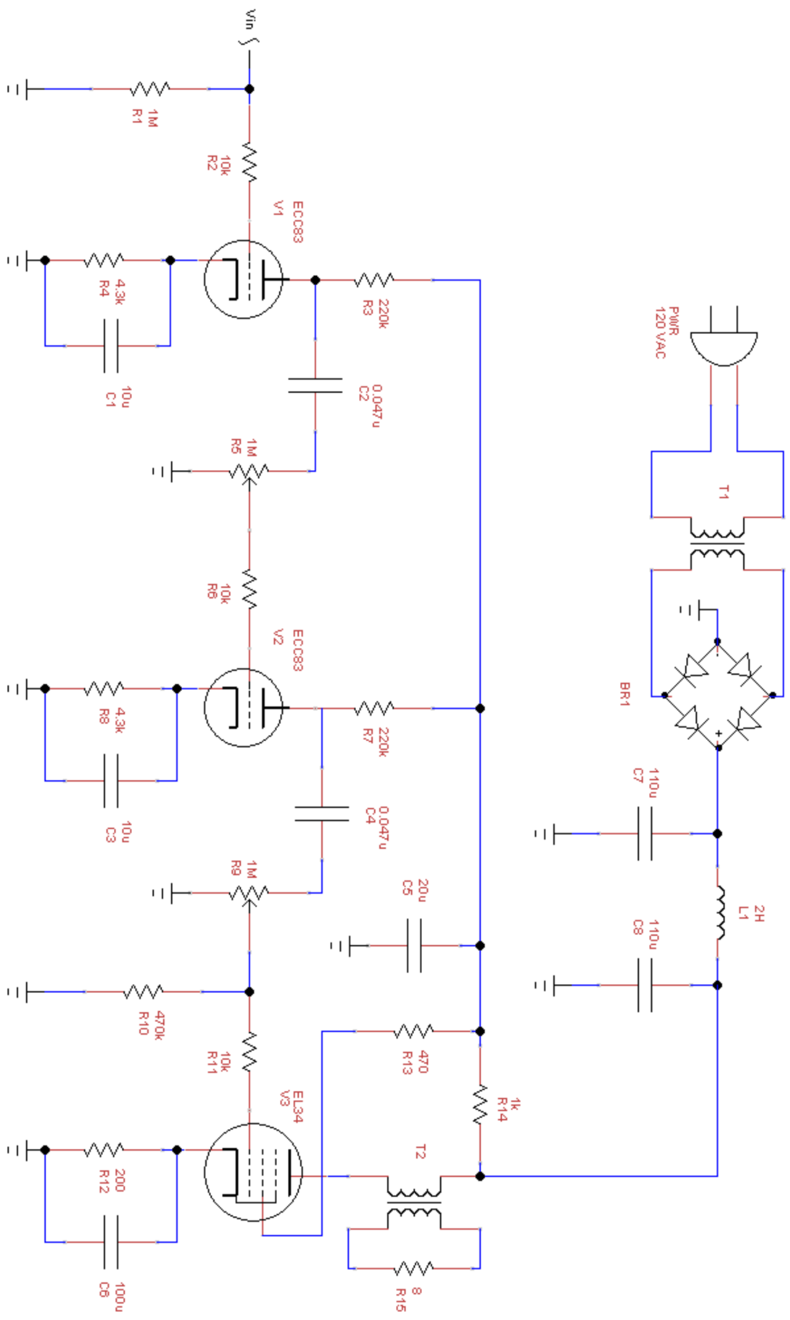
Preamplifier
- A minimum gain of 5 dB over the 20 Hz to 20 kHz band.
- A maximum total harmonic distortion (THD) of 5% in the output signal, with a 1 kHz 500 mV peak to peak input signal, measured as

$$THD = \frac{\sqrt{V_2^2 + V_3^2 + V_4^2 + ...}}{\sqrt{V_1^2 + V_2^2 + V_3^2 + ...}}$$

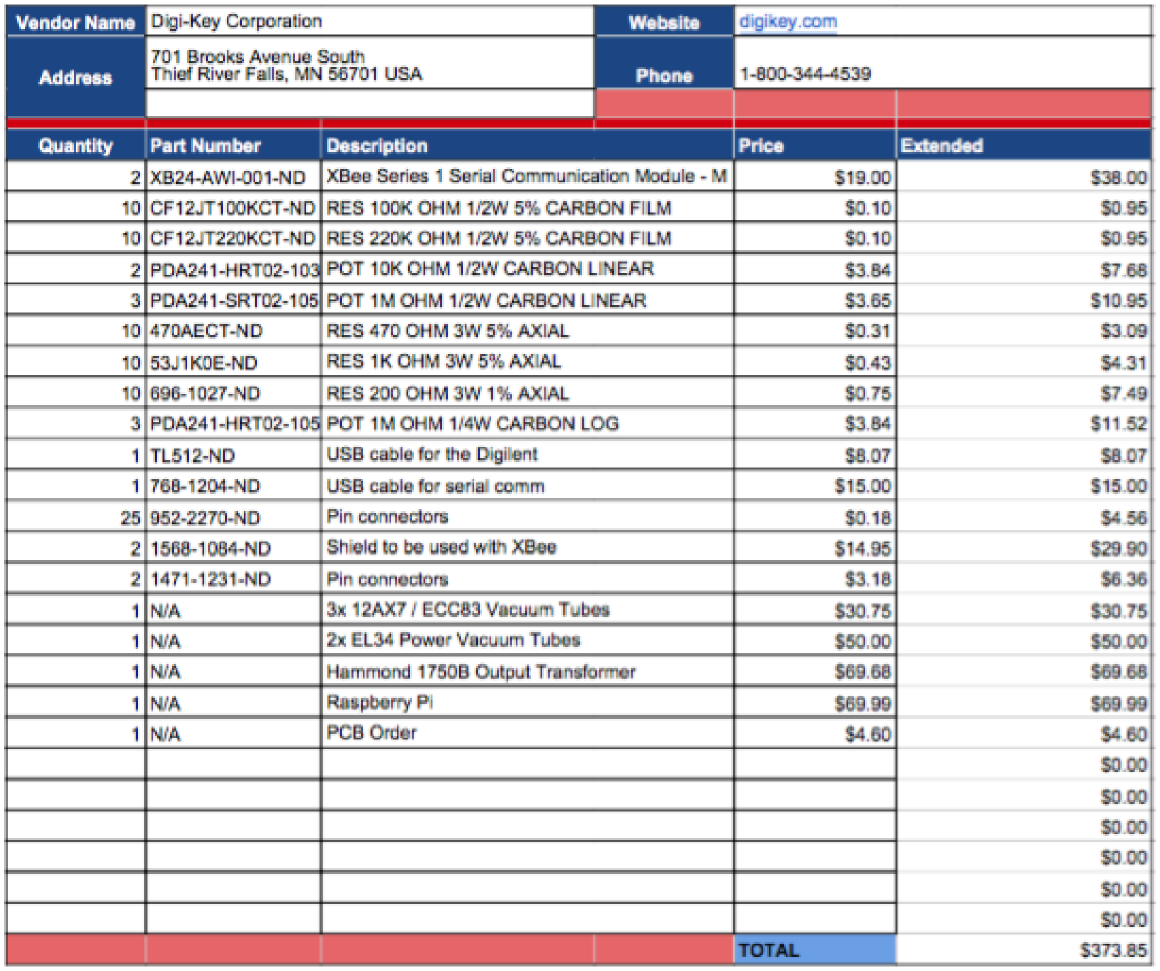
where V_n is the RMS voltage of the n^{th} harmonic and $n = 1$ is the fundamental frequency.

Travis S. Russell Date Jacob A. Allenwood Date

9 Appendix B: Project Schematic



10 Appendix C: Parts List



Vendor Name	Digi-Key Corporation	Website	digikey.com
Address	701 Brooks Avenue South Thief River Falls, MN 56701 USA	Phone	1-800-344-4539

Quantity	Part Number	Description	Price	Extended
2	XB24-AWI-001-ND	XBee Series 1 Serial Communication Module - M	$19.00	$38.00
10	CF12JT100KCT-ND	RES 100K OHM 1/2W 5% CARBON FILM	$0.10	$0.95
10	CF12JT220KCT-ND	RES 220K OHM 1/2W 5% CARBON FILM	$0.10	$0.95
2	PDA241-HRT02-103	POT 10K OHM 1/2W CARBON LINEAR	$3.84	$7.68
3	PDA241-SRT02-105	POT 1M OHM 1/2W CARBON LINEAR	$3.65	$10.95
10	470AECT-ND	RES 470 OHM 3W 5% AXIAL	$0.31	$3.09
10	53J1K0E-ND	RES 1K OHM 3W 5% AXIAL	$0.43	$4.31
10	696-1027-ND	RES 200 OHM 3W 1% AXIAL	$0.75	$7.49
3	PDA241-HRT02-105	POT 1M OHM 1/4W CARBON LOG	$3.84	$11.52
1	TL512-ND	USB cable for the Digilent	$8.07	$8.07
1	768-1204-ND	USB cable for serial comm	$15.00	$15.00
25	952-2270-ND	Pin connectors	$0.18	$4.56
2	1568-1084-ND	Shield to be used with XBee	$14.95	$29.90
2	1471-1231-ND	Pin connectors	$3.18	$6.36
1	N/A	3x 12AX7 / ECC83 Vacuum Tubes	$30.75	$30.75
1	N/A	2x EL34 Power Vacuum Tubes	$50.00	$50.00
1	N/A	Hammond 1750B Output Transformer	$69.68	$69.68
1	N/A	Raspberry Pi	$69.99	$69.99
1	N/A	PCB Order	$4.60	$4.60
				$0.00
				$0.00
				$0.00
				$0.00
				$0.00
				$0.00
			TOTAL	$373.85