# Developing a Hybrid Guitar Amplifier Using Tubes, VST Plugins and Elk Audio OS

Nikhil Saxena, Frie Roest, Viktor Shopov, Rick Veldkamp and Wouter van Abeelen

*Fontys University of Applied Sciences*
*Eindhoven, Netherlands*

n.saxena@student.fontys.nl
f.roest@student.fontys.nl
v.shopov@student.fontys.nl
r.veldkamp@student.fontys.nl
w.vanabeelen@student.fontys.nl

*Abstract*— **This paper discusses the design choices and the challenges of developing an innovative amplifier for guitarists and bassists. Our design utilizes vacuum tubes for pre-amp and the power amp, and uses a Raspberry Pi running Elk Audio OS for the digital processing. The processing involves using plugins made using the JUCE framework.**

***Keywords***— **Amplifier, Elk Audio OS, JUCE, Vacuum Tube, VST**

## I. Introduction

For a long time, there was little change in guitar amplifiers. With our new and innovative view on sound engineering we wanted to combine the true sound from the tube amplifiers with the amazing new technologies in digital signal processing. With the DSP as central unit we will have a tube-preamp, providing the amplitude suitable for the ADC and a power-amp to create enough power to be live on stage.

## II. Full System

The amplifier will be a combination of a traditional tube amp and a digital sound processor. To be able to couple these parts we could not use a traditional amplifier architecture. We chose to design a separate preamp, coupled to the digital sound processor which then gives a signal to the power amp.

## III. Pre Amp

The functions of the preamp are to amplify the signal to a level that is within the range of the digital sound processor input and give the sound character that is typical for tube amps. To achieve this the preamp was designed along load graphs belonging to the used tube out of a book about designing preamps [1]. In the preamp a gain control has been inserted, to give the user control over the amount of distortion and to give the amp the flexibility to work with a wide range of input amplitudes.

## IV. Power Amp

The power amplifier has the function of increasing the signal power, as the name suggests, unlike the preamp, which just increases voltage and adds character to the sound. It contains the final stages of the Arvium amplifier – the Phase Splitter, the Push-Pull amplifier, and the output transformer. The output signal from the transformer is routed to the speaker module.

The purpose of the Phase Splitter is to take the audio signal from the DAC as input and turn it into 2 signals with 180° difference in phase, as well as provide voltage amplification. It is required for the correct functioning of the Push-Pull amplifier. For its implementation, we chose the Schmitt design as it offers excellent linearity, stability, and low distortion [2]. The valve used is a twin-triode 12AX7 tube. The signals exhibit the characteristic soft clipping of the tube, which will translate into warm and pleasing distortion.

The power amplifier itself is Class AB and uses 2 EL84 pentodes in a push-pull configuration. This allows for better efficiency than Class A, and eliminates crossover distortion, associated with Class B. It is capable of outputting 18W of peak power (9W RMS power). The general component configuration can be seen in Figure 1 [3].

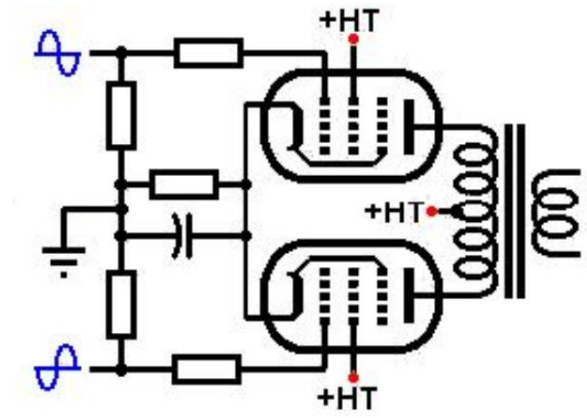


Figure 2: Push-Pull Amplifier Design

The output from the secondary of the output transformer is 12Vpp / 1.5A, which is compatible with a wide range of speakers.

## V. Power Supply

The power supply provides power to all modules and consists of a power transformer with 2 secondary windings – 230V for the valves themselves, and 6.3V for the valve heaters. The supplied 230V are rectified using a diode bridge. A big reservoir capacitor is used for power storage, while 3 decoupling capacitors before every stage smooth the residual AC ripple. The main supply has 3 output stages – 303V, 214V, and 200V respectively. The 6.3V heater supply does not require rectification. The general component configuration can be seen in Figure 1.

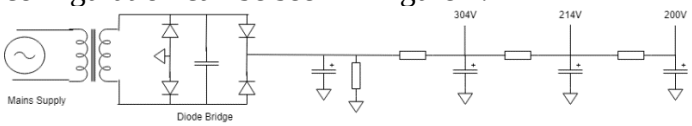


Figure 1: Power Supply Main Circuit

## VI. Digital Signal Processing (DSP)

### A. *Overview*

The DSP module is the "heart" of the system because it is responsible for a lot of the main functionalities. Analog audio in inputted into the module as a digital signal which can then be processed via plugins or in the analog domain (via the EFX Loop). The user can control the plugin order and signal flow to their liking using a GUI on a touchscreen. The digital output is then converted to analog to be sent to the power amplifier.

### B. *Hardware*

The hardware part of the DSP module only consists of a Raspberry Pi 4 with the HiFiBerry DAC+ ADC Pro hat. This hat contains the necessary ADCs and DACs. Figure 3 shows the block diagram with all the inputs and outputs of the DSP module.

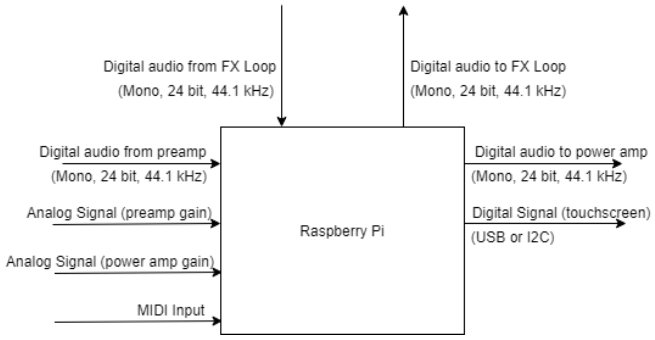


Figure 3: DSP Block Diagram

Notice that the audio is processed in the digital domain, so ADCs and DACs are used. These are not shown in the block diagram for clarity. The ones we use are part of the HiFiBerry DAC+ ADC Pro hat with a minimum and maximum output voltage of 2.1Vrms and a signal-to-noise ratio of 110dB, making it decent for our system.

### C. *Plugins and JUCE*

JUCE is a framework for programming VST plugins and is used by a lot of professional audio companies. It gives you the building blocks for making the audio processing and the user interface at the same time, making it much quicker than traditional methods. It also allows the source code to compile and run identically on Windows, Mac OS X, and Linux platforms.

JUCE plugins are divided into two parts: a processor which handles the audio calculations (signal processing) and an editor or GUI which lets the user control certain parameters the plugin. The functions will not be explained here in a lot of detail, but the most important part is the processBlock(). To obtain blocks of audio samples, plugins implement a callback function which is called by the DAW whenever the DAW is ready to send new data. This means that the sampling rate of the plugin is set by the DAW and not pre-programmed in the code [4].

The distortion plugin we made using JUCE and can be seen below and has 4 basic parameters: Drive, Range, Blend, and Volume.

Drive: This controls the amount of distortion to the signal.
Range: This controls the range of the drive knob for easier use of the plugin. The higher the range, the higher is the maximum value of the drive knob.
Blend: Also known as the 'mix' knob, it controls the amount of dry signal passing through as compared to the wet signal.
Volume: This controls the output volume of the plugin.

The mathematical equation used for distortion is a sigmoid function.
$g(x) = \frac{2}{\pi}\arctan(x)$, where x is the discrete signal value, and g(x) is the distorted signal.

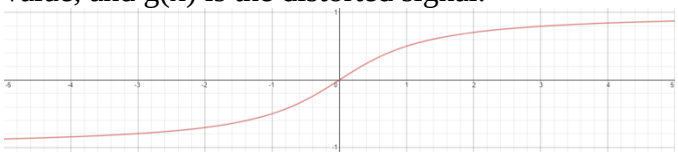

Figure 4: Graph of the Sigmoid Function

All the audio processing in the code is done inside something called the processBlock. The main calculations are shown in Figure 5

```
for (int sample = 0; sample < buffer.getNumSamples(); sample++) {

    float cleanSig = *channelData;

    *channelData *= drive * range;

    *channelData = (((((2.f / float_Pi) * atan(*channelData)) * blend) + (cleanSig * (1.f - blend))) / 2) * volume;

    channelData++;
}
```

*Figure 5: The "Gut" of the Distortion Plugin*

The 'for' loop is where the calculations happen for each individual audio sample.

### *D. Elk Audio OS*

Using a Raspberry Pi with a normal OS would introduce a lot of latency, which can be used for many applications but live music. Elk Audio OS [5] does exactly what we need, it is an OS that is developed for low latency and it makes it possible to run VST plugins without a delay of 1ms round-trip which is inaudible for any musician or audience. Elk Audio OS is made to simplify the development of hardware instruments that want to have digital effects. This came in prefect for the Arvium project as it is brand-new on the market and this will make our project more innovative. The source code of the project can be found on their Github [6].

### *E. Sushi*

Sushi is a headless Digital Audio Workstation (DAW), for the system it will work as a plugin host that supports the VST plugins. Sushi is designed specifically for ELK OS and meet the requirements of the low latency. Sushi can be controlled with through MIDI or OSC. Because the project is very innovative, we are choosing to go with OSC as it is a much higher precision of sending data so you can control your plugins more accurate. It stands for Open Sound Control. The plugins will be loaded by loading a configuration file on Sushi and this loads the selected plugins. The parameters of these plugins will be changed by the incoming OSC messages such as:

1) /parameter/plugin_name/parameter_name, float_value
2) /bypass/plugin_name

These can control the individual parameters of the plugins and can also bypass (ignore) the plugin entirely.

### *F. TouchOSC*

To send the Open Sound Control messages we make use of an Android / iOS application called TouchOSC [7]. On this application you can create your own interfaces and you can program the according messages to each button, slider, or potentiometer. For each different plugin, a new tab is created so it becomes very user-friendly. A small downside on the current concept is the inability to change the order of the plugins but this can be improved in later versions.

## VII. *Design and Housing*

### A. *Overview*

The function of the housing is tying all the different parts together, while keeping the user experience in mind. It also acts as an interface for the user to configure different parts of the amplifier to their own liking. Examples of this are the Gain and Volume knobs, as well as the touchscreen and the connection options.

B. *Touchscreen*

The touchscreen is the main configuration and information hub to the user. Here he or she will configure and adjusts their own settings or presets. Readability and screen real estate are important issues to consider. The 21:9 aspect ratio will help keep the size of the amplifier down, while providing enough on-screen space to easily navigate and configure the amplifier.

C. *Speakers*

Research showed that more and more musicians choose for 12” neodymium speakers. Thanks to their roundedness and overall light weight. The two 12” inch neodymium magnet speakers incorporated in the design provide a well-balanced sound-profile for both guitar and bass.

D. *Inputs & Outputs*

A good amplifier has options, this is the main reason we incorporated two inputs, one normal, the other one -6dB. There is also a USB-connector on the back of the amplifier to connect a computer to configure the amplifier via software. On the back of the amplifier, you have options to replace the main speaker cabinet, or add another speaker cabinet, take a DI OUT signal, connect a foot controller, or swap out fuses quickly.

E. *Module bay & modules*

A module bay is incorporated in the design, where modules (analog sound effect cartridges) will slot into. This allows the user to further customize his or her sound.

F. *User Interface*

The main UI on the left side of the touchscreen consists of big volume a gain knobs, to increase visibility, the two inputs written about earlier, and two switches to toggle the power and standby mode. These switches are equipped with LED-indicators to provide feedback to the user.

G. *Front & back panel*

The Arvium amp has protective panels in the front and back side to protect its hardware from being damaged. These panels consist of a metal mesh, along with a thin foam layer. This way, the parts are protected without negatively affecting the acoustics.

H. *Overall Design*

The Arvium amp has a contemporary look. Its covered mostly in black tolex, with certain parts highlighted in an orange hue. Its name situated front and center in big, bold, white letters.

## VIII. Conclusions

This project unfortunately did not culminate into a fully working total prototype, but it has been a success despite the current situation. Instead of producing one complete prototype we did produce the separate modules as working individual prototypes – such as for the distortion plugin, the TouchOSC interface and the Elk OS – as well as working simulations of the analog electronics and a beautiful finished casing design.

## Acknowledgment

We would like to thank Fontys and the Be Creative department for giving us the opportunity to work on this project and bring Rick Veldkamp’s vision to life. Thanks to the feedback from our mentor Ralph Goes, we were able to stay on track. Thank you to the engineers of Elk Audio for their prompt and elaborate support on their forums.

## References

[1] M. Blencowe, Designing Tube Preamps for Guitar and Bass, Wem Publishing, 2012.

[2] W. G. Morley, "Understanding Audio," *The Radio Contructor*, 1967.

[3] M. Blencowe, "Valve Wizard," 2020. [Online]. Available: http://www.valvewizard.co.uk.

[4] A. P. R. J. D. McPherson, Audio Effects Theory, Implementation and Application, 2014.

[5] Modern Ancient Instruments Networked AB, "Elk OS," 2019. [Online]. Available: https://elk.audio/audio-os/.

[6] "GitHub," [Online]. Available: https://github.com/elk-audio/.

[7] Hexler Limited, "Touch OSC," [Online]. Available: https://hexler.net/products/touchosc.

[8] R. Veldkamp, "Marktonderzoek Gitaar&Basversterkers," -, Eindhoven, 2020.