Electric guitar distortion effects unit using a Raspberry Pi

Renato Santos Pereira[1] **, Rodrigo Varejão Andreão**[2]

[1]Centro de Pós-Graduação – Centro Universitário Espírito-Santense (FAESA)
Av. Vitória, 2220 – 29053-360 Vitória, ES

[2]Departamento de Engenharia Elétrica – Instituto Federal do Espírito Santo
Av. Vitória, 1729 – 29040-780 Vitória, ES

renato_stosp@gmail.com, rodrigova@ifes.edu.br

Abstract. *With the advance of electronics, techniques and algorithms for digital signal processing, digital equipment has been gaining more and more space in the music scene. Micro-processed tools now generate several effects such as modulation, echo, and distortion of sounds generated by musical instruments, previously obtained only by analog units. In this context, this study aimed to develop a prototype of distortion effects unit using a Raspberry Pi (a low-cost small single-board computer) and affordable electronic components. Therefore, five nonlinear functions were used, four of which are present in the literature and one of them was originally developed by the authors. These functions model the behavior of an active element (such as transistors, valves, and operational amplifiers), which when they exceed their amplification thresholds produce distortions in the audio signals. Throughout this article, all the steps in the development of the analog circuits for signal acquisition and output will be presented, as well as the simulation and implementation of the functions in the microcontroller. At the end, with the finished prototype, the frequency response analysis is performed and the sound results achieved by the algorithms is compared with each other and with other distortion units.*

1. Introduction

Until the emergence of modern science, musical instruments and theaters were built based only on empirical knowledge [1]. Scientific studies in this area, over time, directly influenced musical production. And from the intersection between science and music, the guitar emerged.

The electric guitar had its origin in the 1930s. The first solid-body guitar model, known as "Frying Pan", was developed by Adolph Rickenbacker and George Beauchamp, the latter being the inventor of the magnetic pickup [2]. Distortion, one of the most used effects by guitarists, originated with tube amplifiers. When they became popular, amplifiers for electric instruments used thermionic valves in their circuits.

When tube amplifiers operate beyond their maximum amplification capacity (nonlinear operation) they promote signal distortion. The characteristic distortion of this equipment became attractive to musicians in the 1960s and 1970s because it reinforced the harmonic content of chords and notes in a peculiar way [3]. With the emergence of new musical styles from the late 1960s on, additional distortion became a necessity for guitarists. Thus, distortion pedals were developed - electronic equipment used to add more distortion to the electric guitar signal [3]. With the advancement of digital signal processing techniques, starting in the 1990s, techniques for simulating amplifiers and distortion units were used.

2. Nonlinear Distortions

Physical systems can be mathematically modeled as linear or nonlinear systems [3]. Linear systems are those that satisfy the Superposition Principle [4]. In addition, they do not change the frequency response characteristics of their inputs, in other words, they do not add harmonics to the original signal.

The distortion effect is a widely used effect in the musical world, which is an example of a nonlinear system. Although its most common use is in guitar effects, it is also commonly applied in electric bass guitars, keyboards and even vocals [5]. One way to obtain the distortion effect is by increasing the gain of the input signal, which causes the saturation of the amplifier element. Such a process is called clipping [5]. Another possibility to generate the distortion effect of an audio signal can be modeled mathematically, as a time series generated by some nonlinear function [3], which adds odd or even harmonic components, depending on the type of distortion. This form is known as waveshaping [5].

Waveshaping functions are usually static and although it does not exactly match the physical characteristics of real circuits, this approach can be considered satisfactory and is used in commercial amplifier simulators. The use of static nonlinear functions has the purpose of reducing computational effort and, consequently, making real-time simulation feasible [3].

We can classify distortion effects, in relation to the symmetry of the output waveform, as symmetric or asymmetric. Symmetric distortion occurs when the positive and negative parts of the audio signal are limited at the same amplitude, while asymmetric distortion occurs when the positive and negative parts are limited at different amplitudes. This determines whether the generated harmonics will be only odd (symmetric clipping) or odd and even (asymmetric clipping) [6]. According to the clipping, the effect can be classified as soft clipping or hard clipping. In soft clipping, the output signal starts to be soft clipped as it approaches an amplification threshold. In hard clipping, the output signal is clipped just after it exceeds a certain threshold [7]. This determines whether most of the resulting harmonics are high-order or low-order harmonics [6].

2.1. Distortion Functions

There is a wide variety of distortion function formats, symmetric and asymmetric, smooth and discontinuous functions [1], of which five were chosen for study, simulation and implementation in the microcontroller. The first distortion function, presented in Equation 1 is a soft clipper based on the hyperbolic tangent [8]:

$$f(x) = \tanh x \tag{1}$$

A second distortion function is a soft clipper based on the arc-tangent as shown in Equation 2 [1]. In this model, k_d represents the distortion coefficient. Such a coefficient is directly proportional to the amount of distortion applied.

$$f(x) = \frac{\tan^{-1}(k_d x)}{k_d \pi} \tag{2}$$

A third distortion function is the hard clipper, presented in Equation 3 [9]. In this model, the parameter a represents the value at which the input signal is clipped.

$$f(x) = \begin{cases} x, & |x| \leq a \\ a, & \text{otherwise} \end{cases} \tag{3}$$

A fourth distortion function is the cubic soft clipper, described by Equation 4 [10].

$$f(x) = \begin{cases} -\frac{2}{3}, & x < -1 \\ x - \frac{x^3}{3}, & -1 \leq x \leq 1 \\ \frac{2}{3}, & x > 1 \end{cases} \tag{4}$$

The author in [11] cited the asymmetric clipping equation, described by Equation 5, as proposed to simulate triode distortions. In this model, $dist$ is a distortion tuning parameter and Q simulates the operating point of the triode.

$$f(x) = \frac{x - Q}{1 - e^{-dist(x-Q)}} + \frac{Q}{1 - e^{dist(Q)}}, \quad Q \neq 0,\ x \neq Q \tag{5}$$

The design parameters for the asymmetric distortion simulation are based on the above-mentioned function, in which no distortion should occur when the input level is low (the derivative of $f(x)$ should be $f'(0) = 1$ and $f(0) = 0$).

With a limited sampling rate, a high gain on a sine input can produce an approximately square wave output, regardless of the distortion function. All saturating nonlinearities approach a hard clipper asymptotically as the gain increases. There are spectral differences between these different distortion functions, which are subtle but noticeable at lower gain levels. For high gain levels, the soft clipping and hard clipping functions are very similar in terms of output spectrum. The arc-tangent function also exhibits high saturation at large input levels, but the smooth transition produces a smoother spectral response. Hyperbolic tangent functions saturate more slowly at large input levels, thus reducing high-order distortion and aliasing compared to the arc-tangent function [8]. The asymmetric distortion function must perform clipping and limiting large negative input values, while for positive values the behavior is approximately linear. This equation produces even and odd order harmonics [11].

3. Distortion Functions Simulation

In this step, the five equations presented in Section 2.1 of this paper was simulated by software. Three of them (the soft clipper based in hyperbolic tangent, the cubic soft clipper and the asymmetric clipping function) have been adapted to make the project more functional when implemented in the microcontroller. In the other two functions (the soft clipper based on the arc-tangent and the hard clipper), no adjustment was needed. In order that, in the soft clipper based on the hyperbolic tangent, detailed in Equation 1, a coefficient k was added to modify the amount of distortion. This, when multiplied at the input x, modifies the amount of distortion produced by this function. Therefore, the new function based on this equation is described by Equation 6:

$$f(x) = \tanh kx \tag{6}$$

The soft clipper based on the arc-tangent, detailed in Equation 2, has a coefficient k_d, which controls the amount of distortion applied to the input signal, in a directly proportional manner. In the hard-clipping function, detailed in Equation 3, the coefficient a controls the amount of distortion applied to the input signal. The smaller the modulus of a, the more distortion is obtained. From the cubic soft clipper, detailed in Equation 4, a new polynomial function was developed, with a coefficient k, which controls the amount of distortion applied, in an inversely proportional manner, as follows:

$$f(x) = x - \frac{x^k}{k} \operatorname{sgn} x, \qquad k \in \mathbb{N},\ -1 \leq x \leq 1 \tag{7}$$

The signal function was inserted into the system because for odd values of k:

$$\begin{cases} \frac{x^k}{k} < 0, & x < 0 \\ \frac{x^k}{k} \geq 0, & \text{otherwise} \end{cases}$$

However, for even values of k, $\frac{x^k}{k} \geq 0$, even for $x < 0$. In this way, the original function, detailed in Equation 4, would not produce the desired effect at the input. Finally, for the asymmetric clipping function, detailed in Equation 5, a new function (Equation 8) was developed, also with a coefficient k, that controls the amount of distortion applied.

$$f(x) = 0,5 + x - e^{-kx} \frac{1}{|\min x - e^{-kx}|} \tag{8}$$

The software used for the analysis of each of the models was MATLAB®. The input signal was a sinusoid,

with unity amplitude and frequency of 110 Hz. The sampling rate used was 8 kHz, the default of the program. In addition to the output waveforms, the frequency response of each function was also generated as a function of the input signal. The method used to obtain the frequency response was the Fast Fourier Transform (FFT). It is also worth noting that after passing through the system, the signal receives a gain, so that its amplitude is unitary. The amplitude and frequency charts of the five functions is shown in Figures 1 to 5.

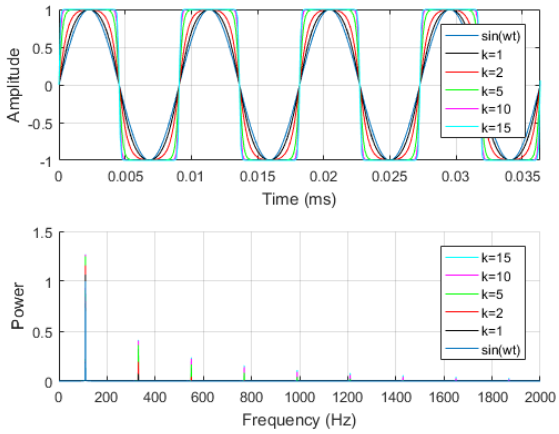

Figure 1: Hyperbolic tangent - Time and frequency domain charts

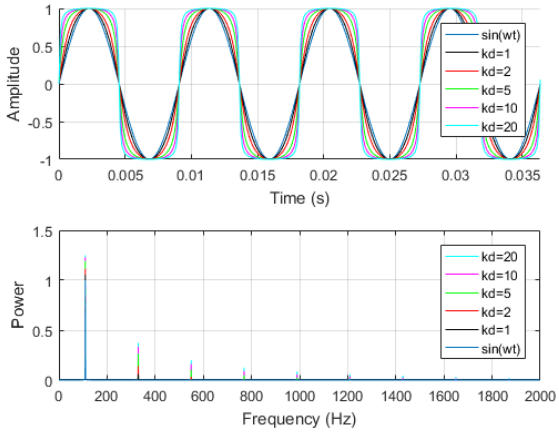

Figure 2: Arc-tangent - Time and frequency domain charts

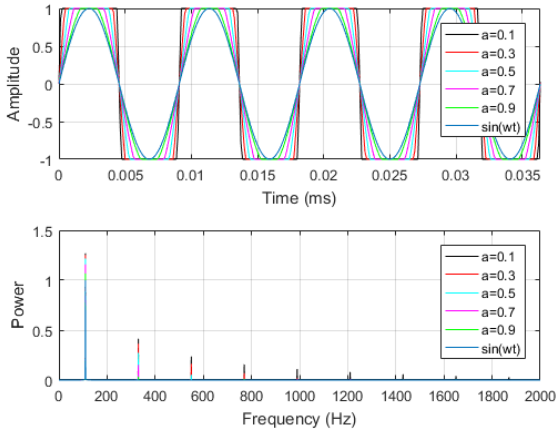

Figure 3: Hard clipper - Time and frequency domain charts

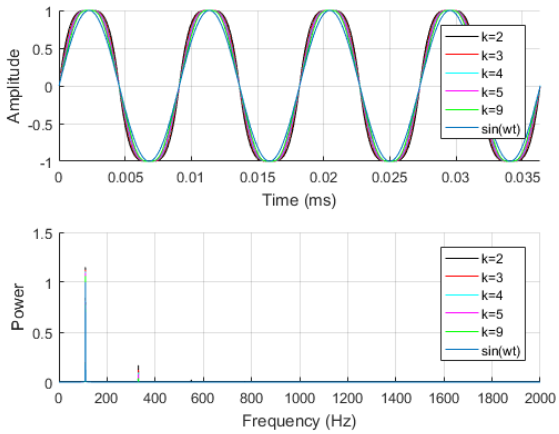

Figure 4: Polynomial Soft Clipper - Time and frequency domain charts

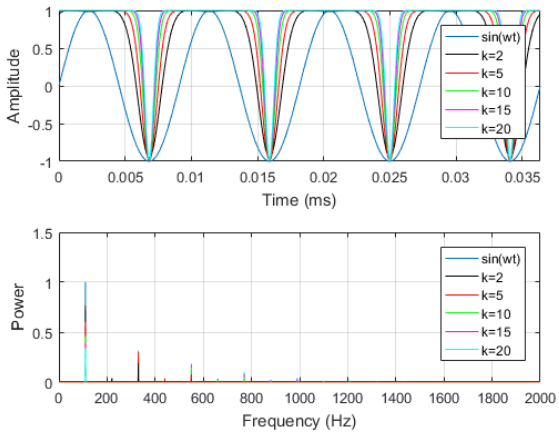

Figure 5: Asymmetric clipping - Time and frequency domain charts

Note that in the models characterized by symmetrical clipping (hyperbolic tangent - Figure 1[12], arc-tangent - Figure 2[12], hard clipper - Figure 3[12] and soft clipper - Figure 4[12]), there is the presence of only the odd harmonics, especially the third and fifth harmonics. Furthermore, it can be observed that the increase in distortion applied to the signal implies an increase in power of the higher harmonics. As the function based on the cu-

bic soft clipper applies a softer distortion, the presence of these harmonics becomes less evident. It can also be seen that, except for the asymmetric distortion function (Figure 5[12]), the output signals of the system have higher power than the power of the input signal. In the asymmetric distortion function, both odd and even harmonics are present. The power of the fundamental harmonic decreases and this power is distributed among the other harmonics.

4. Analog Circuit Development

In this step, the input stage circuit (responsible for acquiring the guitar signal to be processed by the Raspberry Pi) and the output stage circuit (in which the already processed signal is prepared for be received by another effect unit, an amplifier or a soundboard) was simulated and mounted on a breadboard (Figure 6). These circuits were built based on the Pedal Pi project, developed by ElectroSmash [13]. This project was chosen as a basis for the relative simplicity of the circuit, as well as the ease of obtaining the electronic components used in the prototype. The software used for the circuit simulation was the Multisim®.

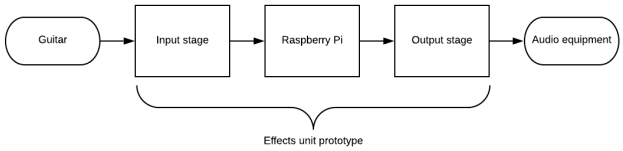

Figure 6: Distortion effects unit - block diagram

4.1. Input Stage Development

The signal acquisition circuit (Figure 7[12]) consists of an operational amplifier, in the non-inverting configuration, followed by a bandpass filter. The signal, which initially has a voltage level between 100 mV and 2 V, receives a gain (adjustable between 1 and 20 by a 500 kΩ potentiometer) so that it can be optimized for the analog-to-digital converter (ADC). In addition, this part of the prototype is responsible for shifting the offset of the signal, so that the SPI converter and consequently the Raspberry Pi do not receive negative voltage values, which would damage the device. The bandpass filter is composed by two first-order low-pass filters and two first-order high-pass filters. The two first-order low-pass filters function is to attenuate higher frequencies, which can generate aliasing. The two first-order high-pass filters are responsible for removing the mains voltage noise and the DC component of the signal.

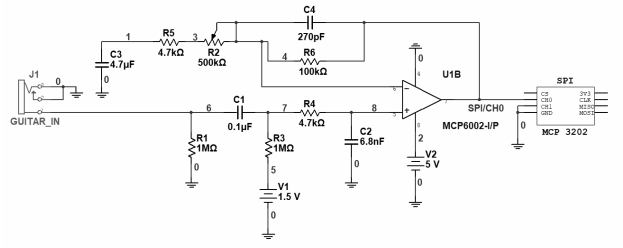

Figure 7: Signal acquisition circuit

To simulate the frequency response (Figure 8[12]) and the amplitude graph (Figure 9[12]) of this circuit, the input signal was modeled as a sinusoid, with a voltage of 2 Vpp (0.707 VRMS) and a frequency of 440 Hz. Potentiometer R2 was set so that the voltage gain of the operational amplifier had its minimum value. This resulted in a gain of approximately 1.12, or approximately 1 dB. This causes the curve corresponding to the system output to be shifted upwards by 1 dB. Therefore, one can consider as cutoff frequencies of the bandpass filter those whose attenuation is at -2 dB. These frequencies, ω_1 and ω_2 are approximately 1.5 Hz and 4.65 kHz. According to the amplitude plot, it can be seen that the signal at the output of the amp-op has a voltage of 2.25 Vpp an offset of approximately 1.6 V.

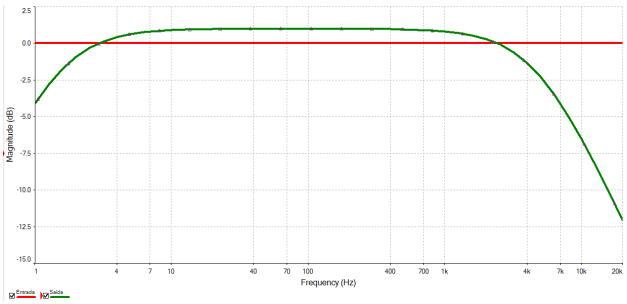

Figure 8: Signal acquisition circuit - Frequency response

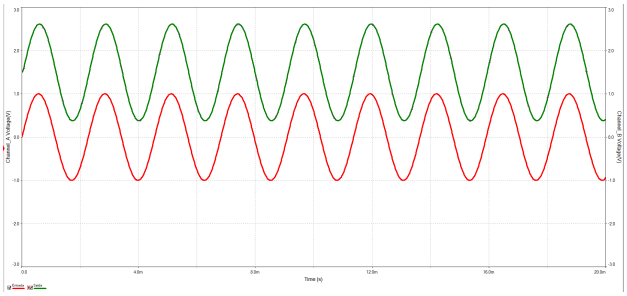

Figure 9: Signal acquisition circuit - Amplitude graph

After simulating this stage, the circuit was mounted on a breadboard and connected to an oscilloscope. A sinusoid generated by the oscilloscope was used as the input signal, with a voltage of 2 Vpp and frequency of 440 Hz. By adjusting the potentiometer to obtain the minimum gain value, an output voltage of approximately 2.3 Vpp was obtained. As it was observed in the simulation, the output signal of this stage has an offset of approximately 1.6 V. With the potentiometer set to obtain the maximum gain value, the voltage of the input signal reduced to 160 mVpp. The measured voltage at the output of the acquisition stage was 3.2 Vpp. Since the Raspberry Pi Zero W does not read analog quantities, after the conditioning stage the signal goes through an external analog-to-digital converter, to then be received by the board. For this step, the MCP3202 was used, an ADC with 12-bit resolution and SPI interface, from Microchip. The reference voltage adopted in this element was 3.3 V, as this is the maximum voltage supported by the Raspberry Pi in its I/O pins. As the MCP3202 has limiting diodes in its internal circuitry, the use of diodes for microcontroller protection

is not necessary.

4.2. Output Stage Development

After the signal passes through the acquisition stage and the microcontroller applies the desired distortion effect, the signal generated by the Raspberry Pi's PWM outputs passes through the output stage. This part of the prototype consists of a third-order active filter (of Sallen-Key topology, shown in Figure 10[12]), a tone control circuit and an operational amplifier in non-inverting configuration with variable gain. To illustrate the frequency response of the Sallen-Key filter, magnitude and phase plots were generated in MATLAB®software. According to these, shown in Figure 11[12], the cutoff frequency is approximately 4.9 kHz. This results in the removal of high-order harmonics, which can generate noise in the desired signal.

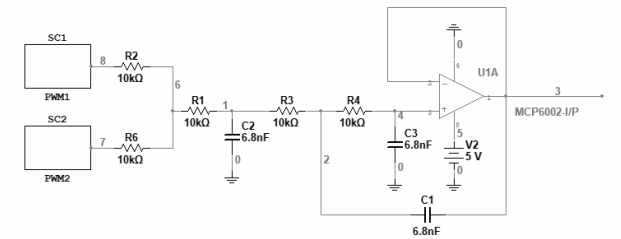

Figure 10: Sallen-Key filter circuit

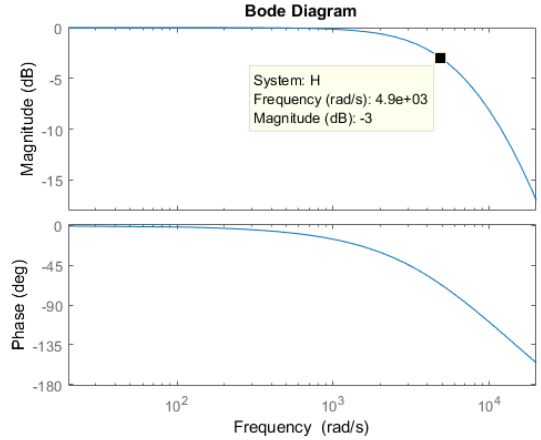

Figure 11: Sallen-Key filter - frequency response

The tone adjustment circuit is composed of the parallel association between a low-pass filter and a high-pass filter, where potentiometers selects the bandwidth of the two filters. One feature that was observed during the simulation of this part of the circuit is that there is an attenuation of the signal throughout the selected bandwidth (from 20 to 20 kHz), especially in the region of mid frequencies (around 1 kHz), regardless of the setting of the potentiometers. This is due to the sum of the responses of the two filters used in the tone adjustment. To correct this effect, an operational amplifier was inserted in series with the filters, in the non-inverting configuration, with a 10 kΩ potentiometer for gain adjustment. The schematic of this circuit is shown in Figure 12[12]. The frequency response of the tone control circuit, shown in Figure 13[12], was generated in the software Multisim®, with the adjustment potentiometers of the two filters adjusted to expand the signal passband as much as possible. The potentiometer connected to the operational amplifier was adjusted so as to obtain a gain of 12 dB.The curve in red represents the signal at the input of the operational amplifier, while the curve in blue represents the signal at the output of the amp-op. With this stage simulation completed, the circuit was mounted on a breadboard.

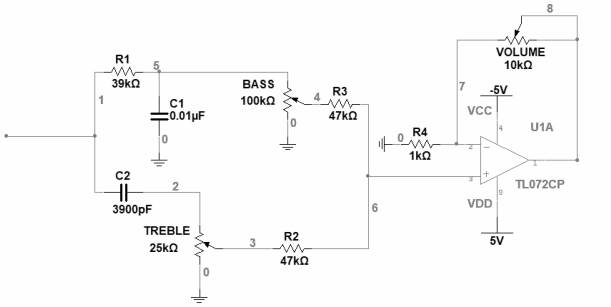

Figure 12: Tone and volume adjustment circuit

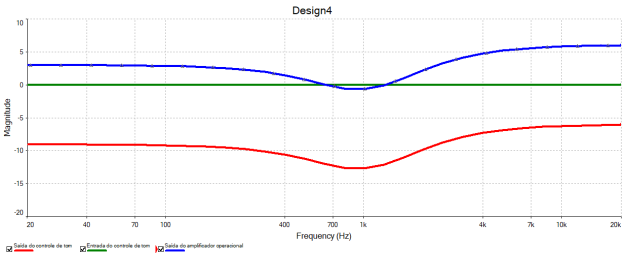

Figure 13: Tone adjustment circuit - Frequency response

5. Raspberry Pi Implementation

Parallel to the study of each system, its characteristics and its implementation computational feasibility, different prototyping platforms were analyzed (Arduino®DUE Raspberry Pi, BeagleBoard, etc.) and it was defined that the hardware used would be a Raspberry Pi, Zero W model. The choice of this hardware was motivated by its low cost, easy access, processing capacity and memory (it has a single-core ARM BCM2835 processor with 1GHz clock and 512 MB RAM), considered sufficient for this study. The Raspberry Pi Zero W has 40 programmable input and output pins, called General Purpose Input/Output (GPIO). The connections of this equipment to the input and output circuits are made through these pins. The operating system used on the board was Raspbian Stretch Lite, a free software based on the Debian Stretch OS that has only the command line interface (or Terminal). After the simulation and breadboard assembly of the analog input and output circuits, the algorithms implemented on the Raspberry Pi relative to each of the five distortion models were developed. For this step, it was necessary to install the bcm2835 library, a C programming language library that provides access to the GPIO pins, allowing the reading and configuration of digital inputs and outputs, serial communications via SPI and I2C interfaces, and access to the system timers. As the programming language adopted in this stage of the

project was the Python language, it was the Python 3 programming environment had to be installed, as well as the PyBCM2835 (Python extension of the bcm2835 library) and NumPy libraries. In addition, the Git version control system was added to the operating system. It is worth noting the lack of documentation for the PyBCM2835 library, which made this task difficult. The algorithm that served as the basis for the elaboration of the codes for each distortion effect, present in the Pedal Pi project, was developed in the C programming language. That said the strategy adopted was to transcribe the base code to the Python language and modify the transcribed code so that it could implement each effect in an optimized way. The compatibility of the data types from the C language to the Python language was accomplished by means of the ctypes library. The sequence of instructions of the codes implemented in the microcontroller can be seen in the flowchart shown in Figure 14.

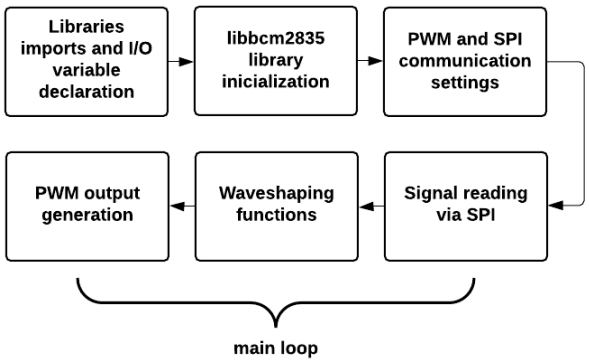

Figure 14: Flow chart of distortion algorithms

In this project, the reconstruction of the signal after applying the mathematical models is done by means of two PWM signals in parallel. One of the PWM outputs is responsible for the D/A conversion of the 6 most significant bits, while the other output converts the 6 less significant bits. The sampling rate, bounded by the supply voltage of the MCP 3202, is approximately 50 kHz [14]. The clock used for SPI communication was 2 MHz and the switching frequency of the PWM signals used was 150 kHz. Subsequently, the five distortion algorithms were hosted via Git in a private repository on the GitHub platform, so that the author could make later modifications remotely.

6. Results

The distortion effects unit was tested through the following experiment: the prototype was assembled according to the block diagram shown in Figure 6. Initially, the input signal was modeled as a sinusoid, generated by an oscilloscope, with a voltage of 2Vpp (0.707 VRMS) and a frequency of 440 Hz. It was observed that the interference caused by the AC mains hum was very significant, which made it impossible to visualize the harmonics added by the distortion algorithms. Different filtering techniques were adopted in an attempt to attenuate that interference, but these did not have the desired effect. The solution adopted to obtain a better visualization was to change the input signal frequency to 120 Hz.

It can be noticed that, among the symmetric distortion models, the presence of the odd harmonics is more evident in the hard clipping function (Figure 19[12] and Figure 20[12]), in which it is possible to clearly see the third, fifth and seventh harmonics. In contrast to this function, the soft clipper (Figure 21[12] and Figure 22[12]) was the function in which the upper harmonics were least evident. In the other two models (hyperbolic tangent - Figure 15[12] and Figure 16[12] and arc-tangent - Figure 17[12] and Figure 18[12]), it is possible to see the third harmonic more clearly.

In the asymmetric distortion function (Figure 23[12] and Figure 24[12]), it is possible to notice the presence of both odd and even harmonics. This demonstrates that the simulation results and the results obtained in the tests with the finalized prototype are convergent.

Finally, with the finished prototype, tests were performed with an electric guitar in order to evaluate the sound created by each effect, as well as its characteristics in relation to the frequencies of notes and chords of the instrument. It was possible to notice the presence of AC mains hum, also influenced by the single-coil pickup present in the guitar used. The audio sample of each effect are available on [15].

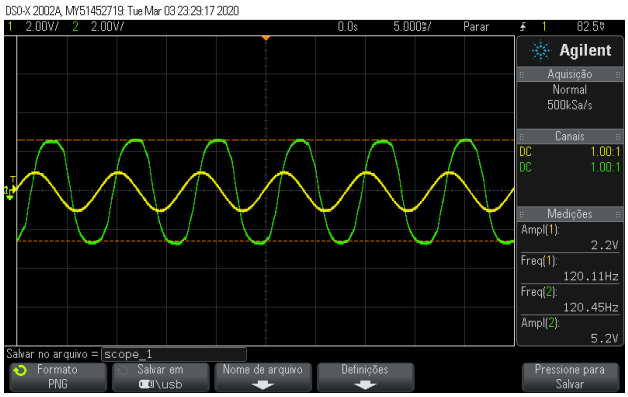

Figure 15: Hyperbolic tangent - Amplitude graph ($k = 2$)

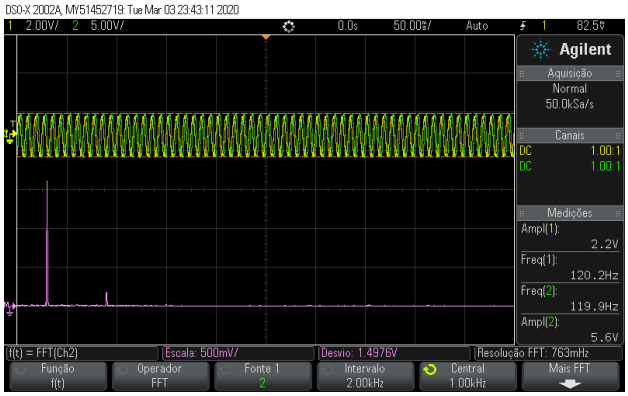

Figure 16: Hyperbolic tangent - Frequency response $(k = 2)$

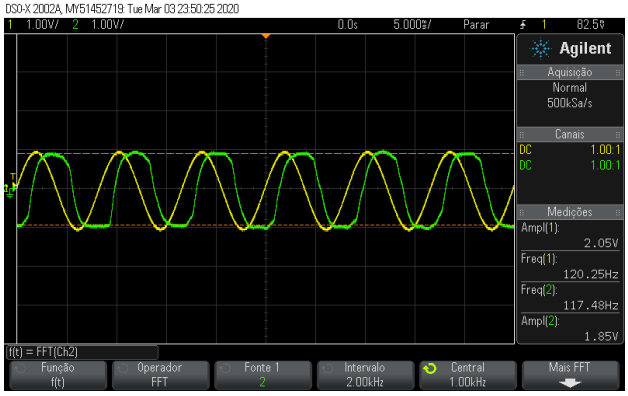

Figure 17: Arc-tangent - Amplitude graph $(k_d = 5)$

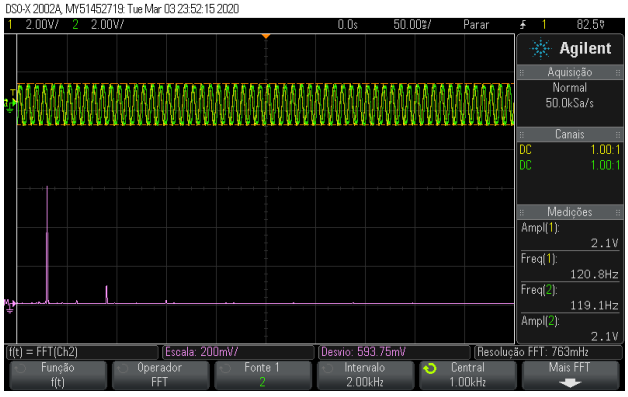

Figure 18: Arc-tangent - Frequency response $(k_d = 5)$

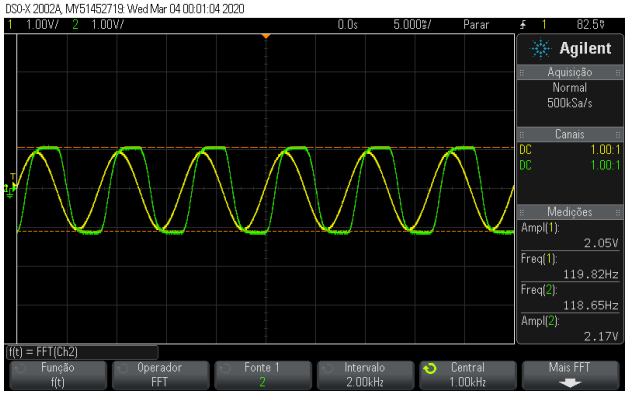

Figure 19: Hard clipping - Amplitude graph $(a = 500)$

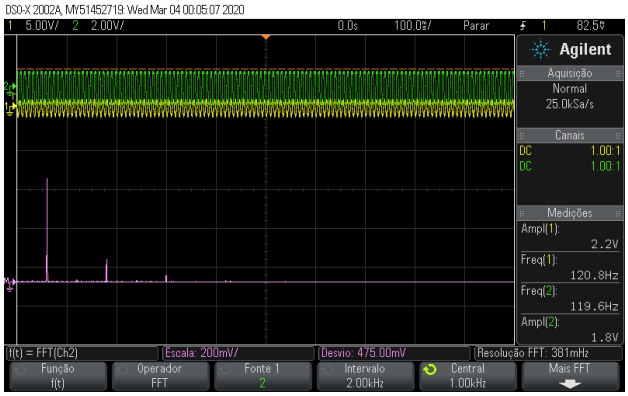

Figure 20: Hard clipping - Frequency response $(a = 500)$

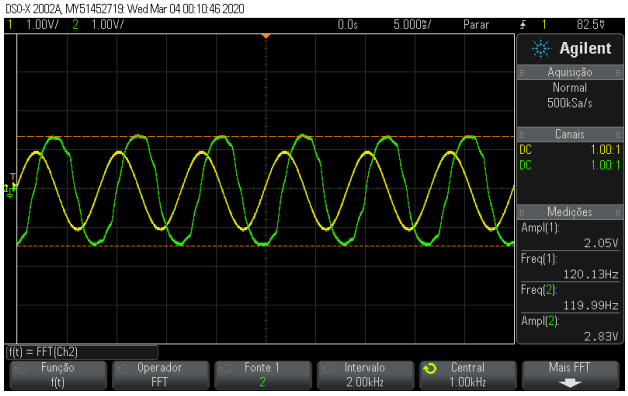

Figure 21: Soft clipping - Amplitude graph $(k = 2)$

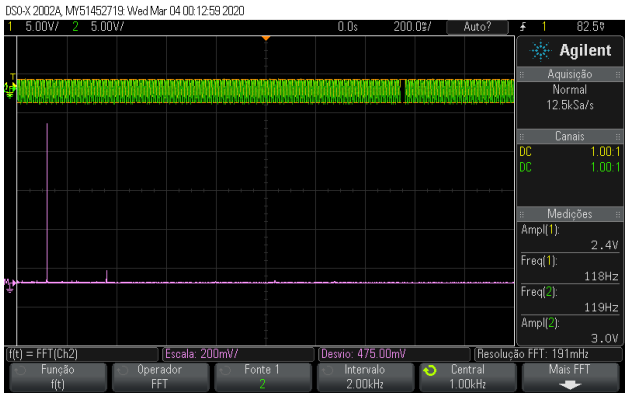

Figure 22: Soft clipping - Frequency response $(k = 2)$

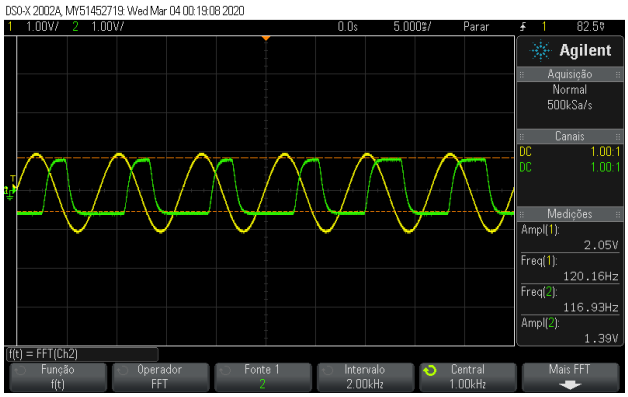

Figure 23: Asymmetric clipping - Amplitude graph $(k = 5)$

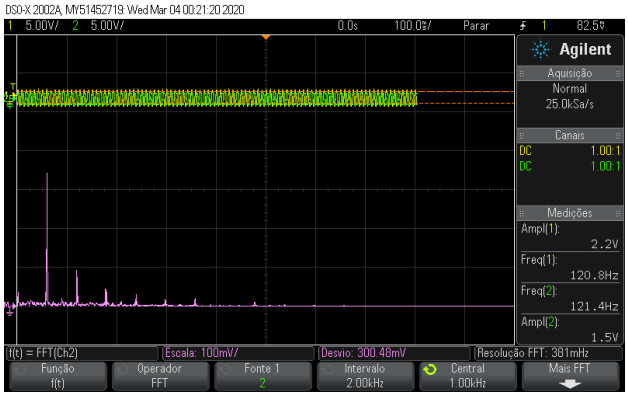

Figure 24: Asymmetric clipping - Frequency response $(k = 5)$

7. Conclusion

The purpose of this work was to design a microprocessor-based distortion effects unit for electric guitar. Through the realization of each step proposed in this study, the overall goal was completed, which can be seen by analyzing the results achieved at each stage of development. It was possible to see the convergence of the theory presented with the results obtained in practice. From the author's point of view, the sound results achieved by each of the algorithms were quite similar to each other. Even though the performance of the prototype was within expectations, it is possible to realize that the sound result obtained through the unit designed in this work falls short of the audio generated by analog equipment or digital equipment of higher value.

It is important to note that professional equipment with the same purpose of this prototype - sound mixers, pedals and digital amplifiers - use DSP's with their own specifications for such activity. In addition, the embedded software used in these units have complex techniques for simulating analog effects and noise elimination. Finally, such devices have electronic components of higher quality than those used in this project.

References

[1] Nicolau Leal Werneck. *Análise da distorção musical de guitarras elétricas*. Dissertação (mestrado), Universidade Estadual de Campinas - Faculdade de Engenharia Eletrica e de Computação, 2007.

[2] Neigmar de Souza. Guitarra elétrica: um ícone na cultura pop do Século XX. *Revista Vernáculo*, 2002.

[3] Thomaz Chaves A. Oliveira, Gilmar Barreto, and Alexander Mattioli Pasqual. Modelagem computacional de efeitos de distorções não lineares para guitarra elétrica. *Revista Brasileira de Computação Aplicada*, pages 69–84, 2013.

[4] John G Proakis. *Digital Signal Processing: Principles Algorithms and Applications*. Pearson Education India, 2007.

[5] André Wagner França. *Uso de processamento digital de áudio na implementação de efeitos em instrumentos musicais*. Monografia (Bacharelado em Ciência da Computação), Universidade de Brasília, 2015.

[6] Robert A Moog. Distortion sound effects circuit, December 25 1979. US Patent 4,180,707.

[7] Pedro Miguel Cruz and Nuno Borges Carvalho. A comprehensive analysis of the clipping effects on signals with different statistical patterns. In *2014 International Workshop on Integrated Nonlinear Microwave and Millimetre-wave Circuits (INMMiC)*, pages 1–3. IEEE, 2014.

[8] David Te-Mao Yeh. *Digital implementation of musical distortion circuits by analysis and simulation*. Ph.D thesis), Stanford University, 2009.

[9] Michel Doidic, Michael Mecca, Marcus Ryle, Curtis Senffner, et al. Tube modeling programmable digital guitar amplification system, August 4 1998. US Patent 5,789,689.

[10] Julius Orion Smith. *Physical audio signal processing: For virtual musical instruments and audio effects*. W3K publishing, 2006.

[11] Pierre Dutilleux and Udo Zölzer. Nonlinear processing. *DAFX: Digital Audio Effects*, 2004.

[12] Renato Santos Pereira. *Projeto de uma unidade de efeitos de distorção para guitarra elétrica*. TCC (graduação em Engenharia Elétrica), Instituto Federal de Educação, Ciência e Tecnologia do Espírito Santo, 2020.

[13] Andrew Gregory. Review: ElectroSmash Pedal Pi — HackSpace Magazine, 2018.

[14] MICROCHIP TECHNOLOGY INC. 2.7V Dual Channel 12-Bit A/D Converter with SPI Serial Interface, 2006.

[15] Renato Santos Pereira. Audio samples: Electric guitar distortion effects unit using a Raspberry Pi [Album]. `https://soundcloud.com/renatostosp/sets/prototype-audio-samples`, 2019.