Hochschule Karlsruhe
University of
Applied Sciences

Fakultät für
Elektro- und
Informationstechnik

Electrical- and Information Engineering-
Information Engineering

Project

A digital Guitar Effects Pedal or what the modern Hendrix would use

Author: Markus Velm
Semester: Two
Professor: Prof. Dr. Manfred Litzenburger

List of Abbreviations

ADC Analog-Digital Converter
CPU Central Processing Unit
DAC Digital-Analog Converter
DMA Direct Memory Access
EDA Electronic Design Automation
FFT Fast Fourier Transform
FIR Finite Impulse Response
IC Integrated Circuit
IDE Integrated Development Environment
IIR Infinte Impulse Response
I^2C Inter-Integrated Circuit
I^2S Inter-Integrated Sound
MSB Most Significant Bit
PCB Printed Circuit Board
SPI Serial Peripheral Interface

Contents

1. Introduction **1**

2. Goal **2**

3. Basics **3**
3.1. Digital Filters . . . 3
3.1.1. Finite Impulse Response Filter . . . 3
3.1.2. Infinite Impulse Response Filter . . . 4
3.1.3. Filter Design and comparison . . . 6
3.2. Microcontroller-Peripherals . . . 8
3.2.1. Inter-Integrated Sound . . . 8
3.2.2. Direct Memory Access . . . 9

4. Requirements and concept **11**
4.1. Challenges and requirements in Audio Signal Processing . . . 11
4.1.1. Latency . . . 11
4.1.2. Packet losses and cracking noise . . . 11
4.2. Concept . . . 11
4.2.1. Signalflow . . . 12
4.2.2. Double Buffering and DMA . . . 12
4.2.3. Design criteria . . . 13

5. Implementation **15**
5.1. Introduction of the systems used . . . 15
5.2. Development of the PCB . . . 15
5.3. Implementation of the filters . . . 16
5.3.1. Implementation using Finite Impulse Response (FIR)-filters 16
5.3.2. Implementation using biquad-filters . . . 19

6. Evaluation **21**
6.1. Evaluation of the hardware . . . 21
6.2. Evaluation of the filters . . . 21
6.2.1. Measurement of the FIR- and Biquad-filters . . . 22
6.2.2. Hearing-tests of the implemented filters . . . 23

7. Conclusion and outlook **25**

A. Schematic **26**

B. Printed Circuit Board (PCB) **30**

Bibliography **31**

List of Figures **32**

List of Tables **33**

1. Introduction

Guitar pedals are widely used. Classic ones are analog but there is an increasing number of digital effects due to processing power of PCs or microcontrollers. Both concepts have reasonable advantages. For the analog concept the unbeatable advantage is that there is minimal latency. The response of an analog system is the fastest you can get. The downside of analog processing is the lack of flexibility.

This project is about an implementation of a guitar effect on a microcontroller based platform. The effect which should be realized is a variable bandpass-filter. Due to its acoustic characteristics it is also called »Wah-Wah«-effect. Guitar legends such as Jimi Hendrix, especially on the song Voodoo Child (Slight Return), and also Eric Clapton used this effect a lot, often in combination with a heavy distortion, also called *Fuzz* [1] [2].

2. Goal

The goal of this project is to implement a digital bandpass filter, which is variable in its center frequency. The core of this is a STM32 microcontroller.

Additionally, an own PCB should be developed to integrate all components onto one single board. There should also be the opportunity to change the software of the microcontroller in order to extend the functionality. Other effects like reverb, delay or other filters should be possible to implement.

3. Basics

3.1. Digital Filters

3.1.1. Finite Impulse Response Filter

Digital Filters are a huge field of digital signal processing. The most common filters are FIR-filters. These are also called »non-recursive filters«. The basic structure is a weighted shift register without feedback. The missing feedback is what makes the FIR-filter stable per construction.

In Figure 3.1 the structure of a FIR-filter is shown.

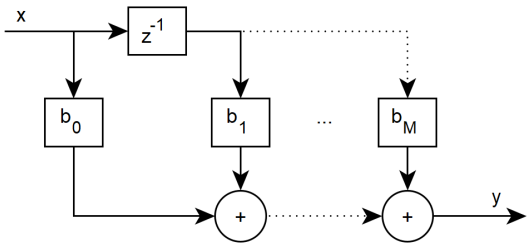


Figure 3.1: Basic structure of a M-th order FIR-filter in the first canonical form [3]

It can be seen that there are multiple components in this filter. The first component is the delay block, which is described as z^{-1}. This block is responsible for adding a delay of one clock cycle to the input data. Secondly, there are multiplication blocks, noted as b_M. These blocks weight the input data and give the result to an adder, which adds $M + 1$ weighted and delayed signal samples. This sum is the output of an FIR-filter. It also means, that the output is only dependend on the present and the M last input samples and not of the past output.

The input signal is x, whereas the output is y. The order of this Filter is M which means the amount of delay blocks.

A special case of the FIR-filter is the *Moving Average Filter*, at which all coefficients are $\frac{1}{M+1}$.

The whole system can be described with the difference equation in the time-domain (Equation 3.1)

$$y[n] = b_0 \cdot x[n] + b_1 \cdot x[n-1] + ... + b_M \cdot x[n-M] \tag{3.1}$$

,

or with the transfer function in the z-domain (Equation 3.2)

$$H(z) = b_0 + b_1 \cdot z^{-1} + ... + b_M \cdot z^{-M} \tag{3.2}$$

.

The output can be calculated with the convolution (noted as $*$) of the impulse response $h[n]$, which is simply the sequence of the filter coefficients, and the input signal $x[n]$ (Equation 3.3).

$$y[n] = x[n] * h[n] \tag{3.3}$$

Another way to describe the filtering result is in the frequency domain. If the spectrum of the input is given with $X(z)$ and the frequency response of the filter is $H(z)$ than the output is $Y(z)$, which can be calculated like shown in Equation 3.4.

$$Y(z) = X(z) \cdot H(z) \tag{3.4}$$

The stability of the FIR-filter is characteristic, this can be seen in the pole-zero plane. All poles are in the center of the unit circle. This makes this filter always stable, as it is necessary that the poles of a digital filter are in the unit circle.

If the value of the z-plane is evaluated on the unit circle, the result is the frequency response of the filter (Figure 3.2).

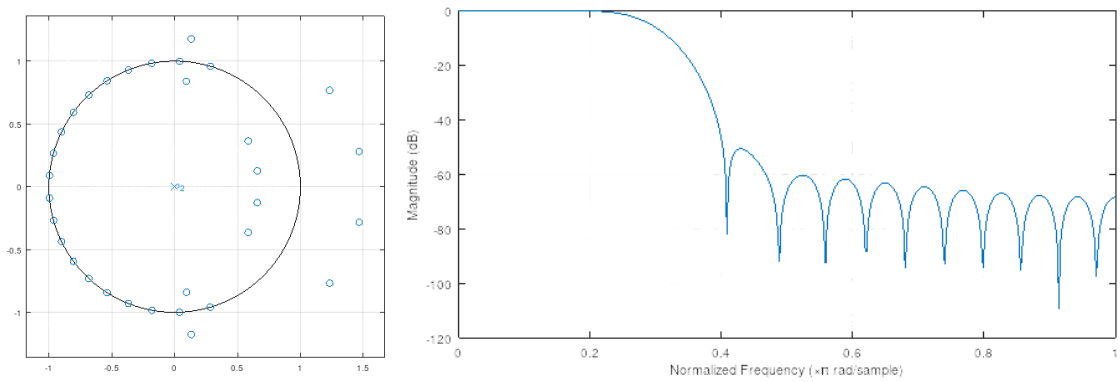


Figure 3.2: z-plane plot (left) and frequency response (right) of a FIR-lowpass-filter

3.1.2. Infinite Impulse Response Filter

Another filter type is the Infinte Impulse Response (IIR) filter. The structure is recursive as it is shown in Figure 3.3.

Due to this recursive structure, this type of filter is no longer necessarilly stable. This means that it can oscillate and possibly the output can grow exponentially. To fulfill stability, all the poles of the IIR-filter have to be in the unit-circle [3].

Its transfer function can be described as follows:

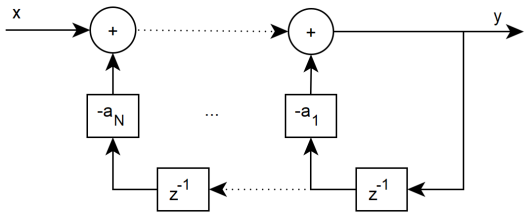


Figure 3.3: Basic structure of a M order IIR-filter in the first canonical form [3]

$$H(z) = \frac{1}{a_1 \cdot z^{-1} + ... + a_M \cdot z^{-M}} \tag{3.5}$$

In the time domain, the output is equal to

$$y[n] = x[n] - a_1 \cdot y[n-1] - a_2 \cdot y[n-2] - ... - a_M \cdot y[n-M] \tag{3.6}$$

.

The output is dependend on the current signal and all the M outputs before. In literature and other documentation, the a_0-coefficient is also referenced. This is the amplification of the output signal. Due to filter-stability-reasons, this coefficient is typically set to o [4] [5].

If the two types of filters are getting merged, the result is also an IIR-filter, plus a few additional feed-forward-terms. A widely used example of this is the *Biquad-filter* [4].

These filters use three filter-taps in the forward- and two taps in the feedback-path. The structure is shown in Figure 3.4.

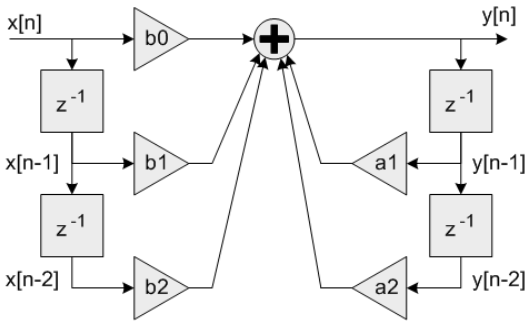


Figure 3.4: Structure of a biquad filter [4]

The transfer function of this filtertype is:

$$H(z) = \frac{b_0 + b_1 \cdot z^{-1} + b_2 \cdot z^{-2}}{a_1 \cdot z^{-1} + a_2 \cdot z^{-2}} \tag{3.7}$$

Here it can be seen, why this filter is called »biquad«: it is the short form for »biquadratic«, which relates to the powers of the numerator- and denominator-terms. Also in this structure, the a_0-coefficient is set to 0.

This transfer function leeds to an output of:

$$y[n] = b_0 \cdot x[n] + b_1 \cdot x[n-1] + b_2 \cdot x[n-2] - a_1 \cdot y[n-1] - a_2 \cdot y[n-2] \quad (3.8)$$

It is worth noting, that the denominator-coefficients have to be negated to match the structure in Figure 3.4.

3.1.3. Filter Design and comparison

This section deals with designing filters. There are a few ways to calculate the numerator and denominator-coefficients. But this shouldn't be part of this section, the goal is more like looking at how to design a filter in a high-level perspective.

First of all, the number of filtertaps will be discussed. As stated earlier, FIR-filters have the convenient property that they are always stable. The downside of these filters are, that for example a narrow-band bandpass filter needs more filtertaps, than a comparable IIR-filter [3]. This should not be a huge problem in the most cases, but if the goal is to variate the filter-coefficients, this may lead to problems. Additionally, a longer filter could lead to a bigger latency.

Secondly, the shape of the frequency responses of the FIR and IIR-filters differ. The recursive filter is more often used to model an actual analog filter [3].

In order to compare these two types of filters, two bandpass-prototypes are being simulated. Given is a 3 dB-bandwidth at a certain frequency which should be achieved by both filters. The number of necessary coefficients for this frequency response is a metric which can be used to compare the FIR- with the IIR-filter.

The bandwidth is defined as

$$BW = f_{3dB} = f_{high} - f_{low} \quad (3.9)$$

and it can be used to calculate the quality-factor of the filter:

$$Q = \frac{f_{res}}{BW} \quad (3.10)$$

Given is a bandwidth of 1 kHz at the mid-frequency of 5 kHz. This leads to a quality-factor of 5.

For the generation of these filters, the python package »scipy« with the module »signal« was used [6].

For the FIR-Filter 85 coefficients were necessary to achieve these criteria, while the IIR-Filter just needs 6 coefficients. This is due to the fact, that IIR-filters have poles, which can produce sharper edges, but it can also lead to instability. This makes the IIR-Filter more suitable, if the number of coefficients is the only criterion.

If the frequency response of the filter is evaluated, it can be observed that the 3 dB bandwidth is not the only difference. Also the shape of these filters differ. This may be important for some applications, for this specific problem, is has to be evaluated.

In the following graphics, there is the FIR- on the left and the IIR-bandpass-filter frequency response on the right.

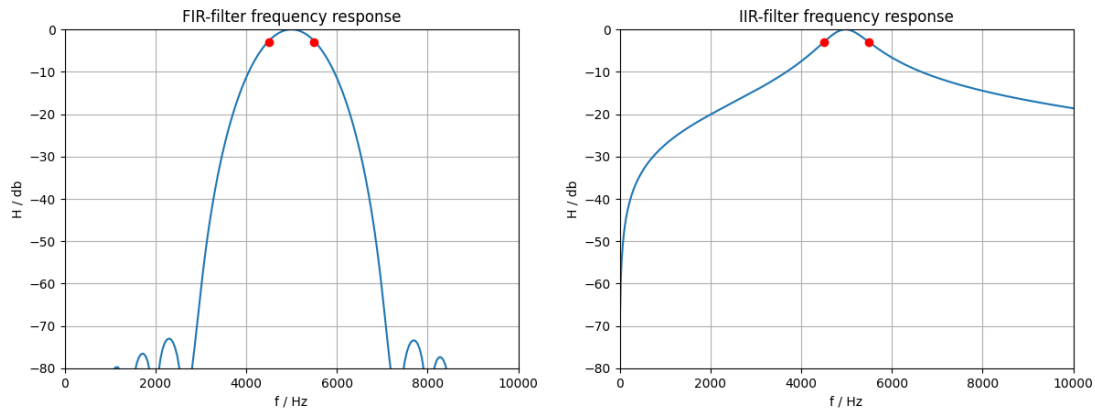


Figure 3.5: Comparison of the frequency response of a FIR- and IIR-bandpass-filter

The red dots mark the desired bandwidth.

It can be seen, that the frequency response of the FIR-filter has a convex form, whereas the frequency response of the IIR-filter is concave in shape.

It is worth noting that by changing the mid-frequency and holding the quality-factor constant, the bandwidth also changes proportionally (see Equation 3.10). By decreasing the frequency the FIR-filter is not capable of achieving this bandwidth if the number of coefficients do not increase. This is not true for the IIR-filter, the bandwidth can always be achieved. This is shown in Figure 3.6.

As a reference, an analog bandpass-filter is shown in Figure 3.7, designed with the Texas Instruments filterdesign-tool [7]. The parameters the tool gets are also a bandwidth of 1 kHz at a mid-frequency of 5 kHz.

The tool calculated a 6-th order filter with the given filtertype being »Bessel«. This type was also used to calculate the IIR-filter coefficients.

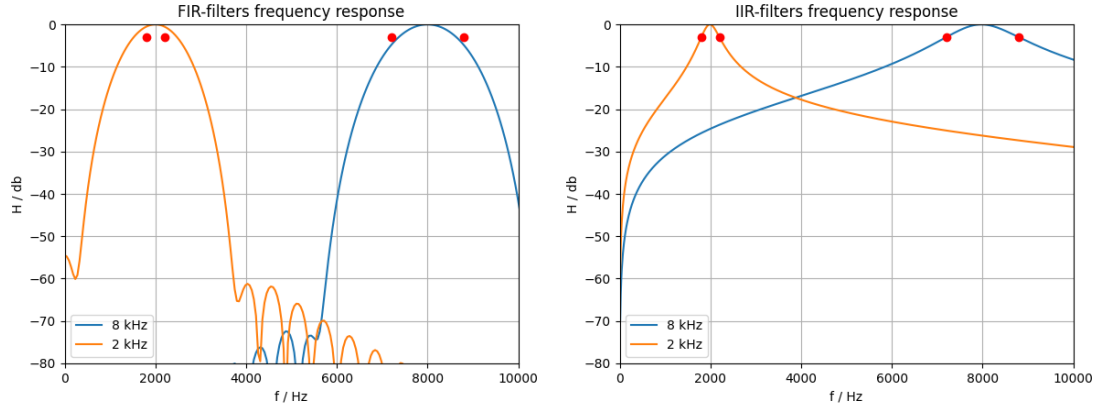


Figure 3.6: Comparison of the frequency response of a FIR- and IIR-bandpass-filter at different mid-frequencies

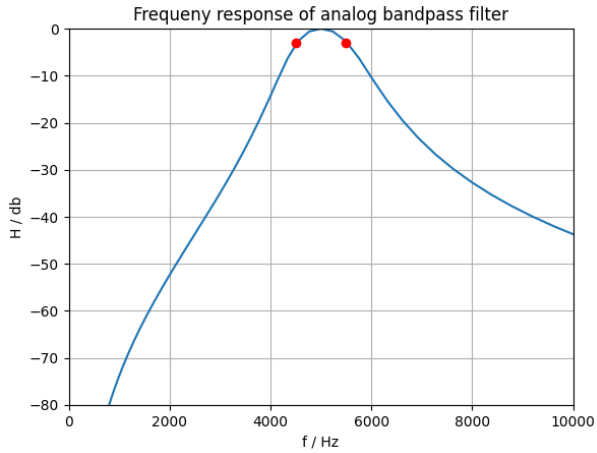


Figure 3.7: Analog bandpass filter frequency response

It can be seen, that the basic shape of the IIR-filter is much closer to the analog equivalent than the FIR-filter.

3.2. Microcontroller-Peripherals

3.2.1. Inter-Integrated Sound

The Inter-Integrated Sound (I^2S) protocol was developed by *Philips* to share audio between Integrated Circuits (ICs). A similarity to the Serial Peripheral Interface (SPI) protocol can be recognized.

There are three signal lines:

- SCK: Clock signal
- SD: Data signal

- WS: Word select, for distinction between left and right channel

So the data is transmitted over the same signal line by time division multiplexing. To illsutrate the timing of the protocol the corresponding diagrams are shown in Figure 3.8 [8].

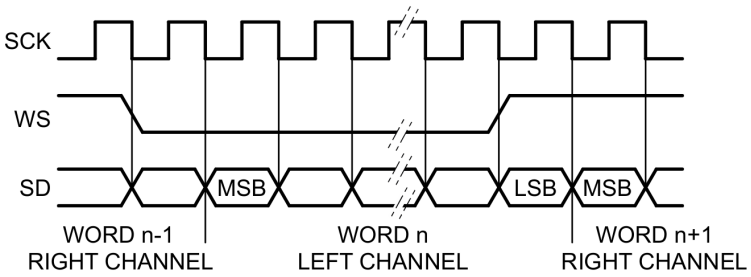


Figure 3.8: Timing diagramm of the I^{2}S protocol [8]

The word length can vary between transmitter and receiver. That is why the Most Significant Bit (MSB) is sent first in the standard configuration. Additionally the participants do not need to know the word length of the counterpart in advance [8].

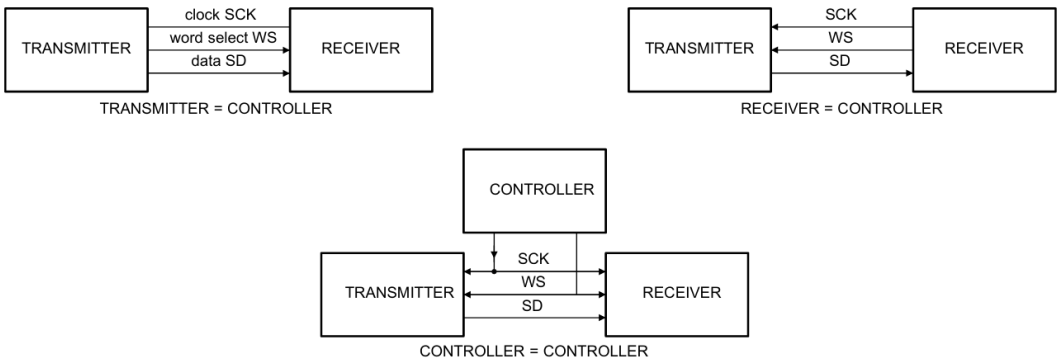


Figure 3.9: The three basic configurations of the I^{2}S protocol [8]

The wiring of this serial bus is possible in three basic configurations (shown in Figure 3.9). Thereby the controller is always responsible for the SCK and the WS line.

3.2.2. Direct Memory Access

Direct Memory Access (DMA) is a feature in computer systems that allows certain hardware components or peripheral devices to directly access the system's main memory independently of the Central Processing Unit (CPU). This enables data to be transferred between memory and peripherals, such as Analog-Digital Converters (ADCs) or I^{2}S, without requiring constant CPU intervention. As a result, the

CPU is freed up to perform other tasks while the data transfer occurs in parallel, improving overall system efficiency and performance [9].

The implementation of DMA hardware varies depending on the microcontroller model. In the STM32H7 series, multiple DMA channels are available, allowing multiple data streams to be handled simultaneously by a single DMA controller. There are also two operational odes: normal mode and circular mode. In normal mode, each data transfer must be manually restarted after the previous transfer completes. Conversely, in circular mode, the transfer requires only a single initialization, as it continuously cycles through the buffer, automatically reloading data as needed [9].

This setup enhances efficiency for continuous data operations, such as audio processing or real-time sensor data acquisition, as it minimizes CPU intervention and supports uninterrupted data flow [9].

For this task it is advantageous, because the signal processing can be done while the data transfer is managed by the DMA.

4. Requirements and concept

4.1. Challenges and requirements in Audio Signal Processing

4.1.1. Latency

As we have to sample a block of data in the digital domain, there is a latency between the input and output depending on some parameters. Firstly the buffer size has a large impact on the latency. The greater the buffer, the greater the latency. It is application depending what latency is acceptable, this is why that parameter limits the system and should be evaluated carefully.

Secondly other components, which are application depending, adds latency to the whole signal path. For example filtering or other modifications take some time to be computed. Also the signal has to be converted from analog to digital and transmitted over an interface. All these components latencies add up to an overall system latency.

In this application, where a guitar signal is processed, the overall latency of the system should be less than 5 ms [10].

4.1.2. Packet losses and cracking noise

In some audio applications cracking noise can occur. This is due to some impulses on the audio signal. This can be caused by an error of the input sample, for example particles on a vinyl record, or by slow implementations of digital signal processing algorithms.

If the processing of the input data takes longer than the actual transfer, packets of audio data will then be dropped. An impulse or step on the output data will be seen, this can be heard as a cracking noise [11].

4.2. Concept

The basic concept is shown in Figure 4.1.

The input data is coming from the Line In, is sampled by a ADC and transmitted over I^2S to the microcontroller. There the data will be processed and the output will be transmitted again over I^2S to a Digital-Analog Converter (DAC). The output is again routed to a Line Jack.

Additionally a potentiometer is connected to set the midfrequency of the filter.

To ensure an adequate sampling rate for audio quality, the frequency is selected to be 48 kHz.

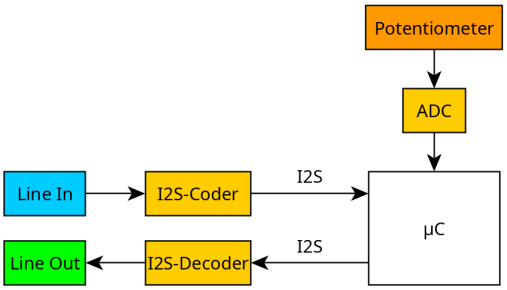


Figure 4.1: Basic concept

4.2.1. Signalflow

The basic idea is to filter the input signal with a bandpass filter. The mid frequency should be derived from a potentiometer which is simply connected to an ADC inside the microcontroller. It has to be evaluated how exaclty the filter is adjusted.

The first attempt will be a lookup-table and everytime the position of the potentiometer changes, the corresponding filter coefficients will be loaded.

The second approach is, to calculate the filter coefficients for a biquad-filter if the value of the potentiometer changes. This is reasonable to evaluate as the number of coefficients is small in comparison to a FIR-filter.

4.2.2. Double Buffering and DMA

To solve the problem with packet losses it is recommended to use the double buffering method. It is basically a buffer which is cut in half. If the first half is filled with data then this half will be processed while the other half is filled with the second half of the input data. Is the second half filled then this half will be processed and so on.

The advantage is that it is not necessary to wait for the whole input data block to process. This will reduce the time between processing and sampling data [12].

To take advantage of this, it is useful to combine this technique with a DMA, which is already discussed in section 3.2.2.

The whole procedure is shown in Figure 4.2.

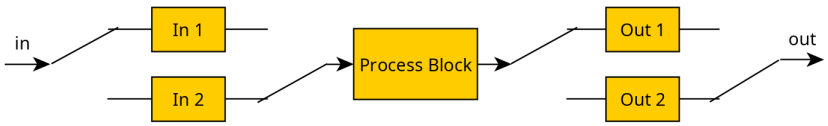


Figure 4.2: Double buffering concept [12]

4.2.3. Design criteria

As the goal is to develop a variable bandpass filter, not only a set of coefficients for one filter has to be stored, but many versions of this filter with different mid-frequencies.

As a reference, the frequency response of the *Cry-Baby*-Pedal has been measured, which is shown in Figure 4.3.

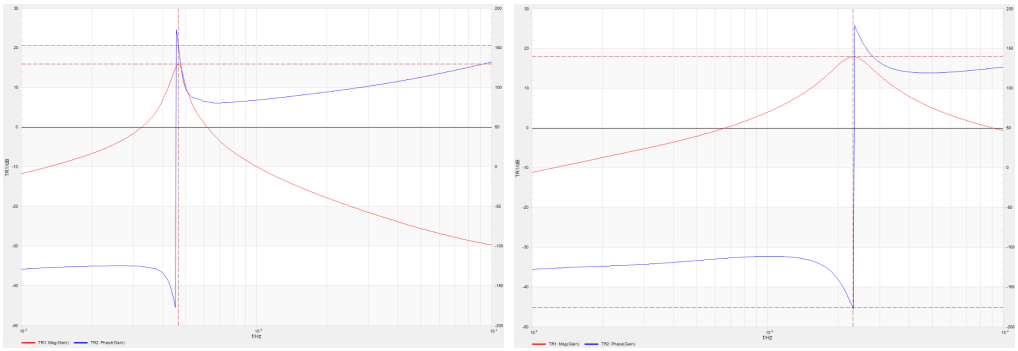


Figure 4.3: Measurement of the frequency response of a Wah-Wah-pedal at the lowest and highest frequencies

The resonance frequency of this filter varies between 466 Hz and 2,3 kHz.

It is also to be seen, that as the frequency changes, the bandwidth also seem to change. Firstly, it has to be mentioned that this is a logarithmic scale. Nevertheless, by measuring the bandwidths of this filter, it turns out, that the quality factor also changes. The bandwidth at 466 Hz is measured to be 54 Hz, which leads to a Q-factor of:

$$Q = \frac{466\,Hz}{54\,Hz} = 8,63 \tag{4.1}$$

At 2,3 kHz the bandwidth is 600 Hz resulting in a quality factor of:

$$Q = \frac{2,3\,kHz}{600\,Hz} = 3,833 \tag{4.2}$$

To keep the Q-factor constant, with increasing the resonance frequency, the bandwidth also has to increase (Equation 3.10) [13].

For the design of the digital filters, it is assumed, that a constant quality-factor is sufficient to model the analog reference.

To conclude, there are two design criteria for the filters which can be derived.

- mid-frequency varies from 460 Hz to 2,3 kHz
- it is assumed, that a comparable filter is achieved by using a constant quality-factor of 5

Another factor is the buffer size, as it has a direct effect on the overall latency. Because it is not necessary to use a big buffer size, the choice fell on 128 samples, which means 64 samples per processing step, as double buffering is used. This is also evaluated in later steps.

5. Implementation

5.1. Introduction of the systems used

For the first implementation of basic filtering operations, the evaluation board *Nucleo-H743ZI* of STMicroelectronics is used. The breakout board *PmodI2S Stereo IN/OUT* is used in order to sample audio data from a 3,5 mm jack input.

The choice for the Integrated Development Environment (IDE) fell on the *STM32CubeIDE* as it provides an easy to use installation and development process.

Digital filters need filter coefficients. For the first approach these coefficients were computed beforehand and then stored in the memory of the microcontroller. For these computations, own Python scripts were developed.

In order to design the PCB, the Electronic Design Automation (EDA)-software *KiCAD* was used.

5.2. Development of the PCB

In order to get a handy pedal, the whole hardware should be integrated into one single PCB. The dimensions of this should not exceed the standard size of effect pedals. As a reference the *Boss DS-1* distortion pedal were used to derive a reasonable size for the PCB. The dimensions of the distortion pedal are 73 x 129 mm, which should not be exceeded. The final PCB now has dimensions of 56,39 x 84,33 mm.

Furthermore, there should be enough inputs, to provide the ability to extend the functionality of this board. The idea is, to develop a platform, where any effect can be implemented. In order to fulfill this, three analog and four digital inputs are connected to solderpads for later use. Additionally, an Inter-Integrated Circuit (I^2C) connector is also provided, if for example a little display should be used.

The main components that were chosen are:

- STM32H725 microcontroller
- CS5343-CZZ Audio DAC
- CS4344-CZZR Audio ADC

The reason for the choice of these components is that these are also present on the Nucleo- and the breakout-board, except for the microcontroller. Therefore the smallest package with 68 pins were chosen.

A blockdiagram of the PCB is shown in the following figure.

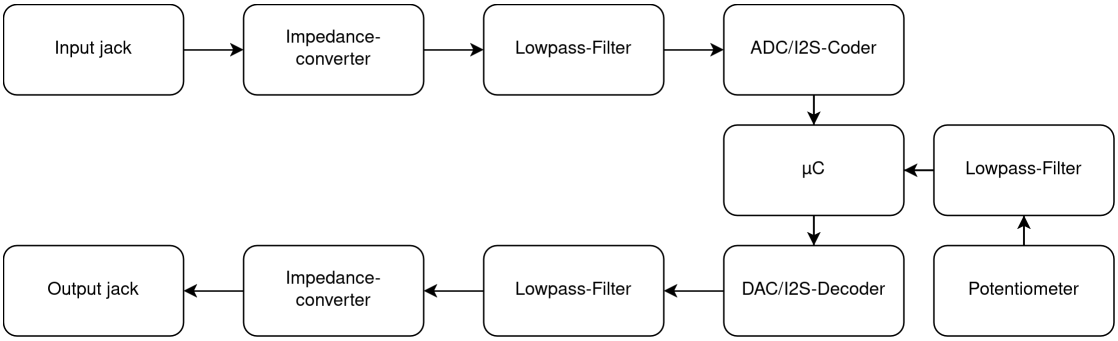


Figure 5.1: Blockdiagram of the PCB

5.3. Implementation of the filters

5.3.1. Implementation using FIR-filters

For the first tries, FIR-filters were used, because they always guarantee stability [3]. The first prototype of the coefficients combines a set of 32 coefficient sets, which were calculated beforehand, using the python scripts, with the following parameters:

Table 5.1: Parameters for FIR-filterdesign

quality factor	number of coefficients
5	130

Figure 5.2 shows a FIR-model of the analog reference at the lowest and (460 Hz) and the highest frequency (2242 Hz). These filters are generated by using the window-method with the Hamming-window. Also the Blackman- and the Bartlett window are tested. These simulation results are found in Figure 5.3 and Figure 5.4.

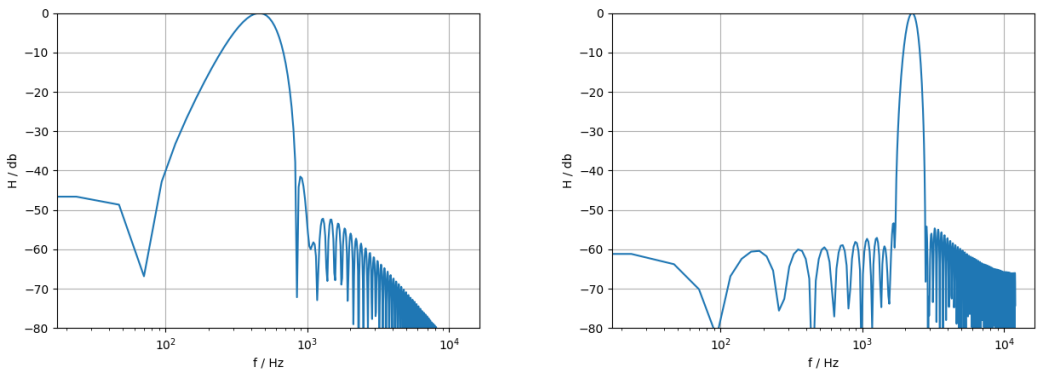


Figure 5.2: Simulation of the FIR-model frequency response at 460 Hz and 2242 Hz using the Hamming-window

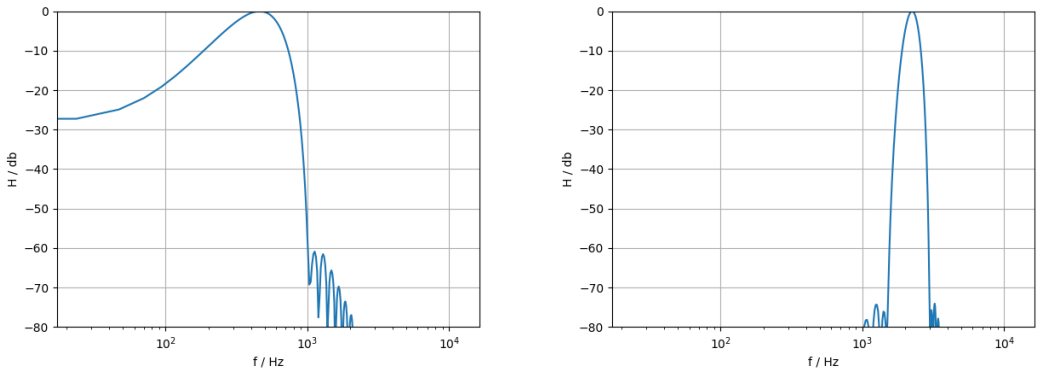


Figure 5.3: Simulation of the FIR-model frequency response at 460 Hz and 2242 Hz using the Blackman-window

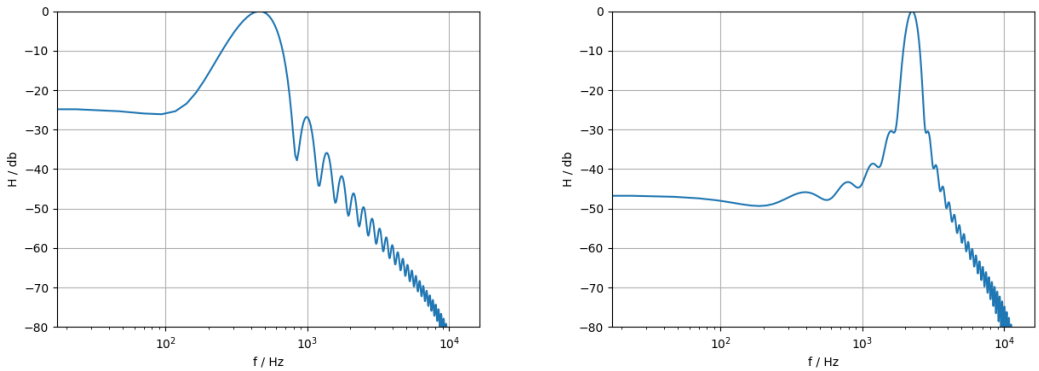


Figure 5.4: Simulation of the FIR-model frequency response at 460 Hz and 2242 Hz using the Bartlett-window

It can be clearly observed, that the shape of the filters are not close to the analog reference. The filter edges, especially near DC, are not as sharp as the actual measured reference filter.

The flowchart of the software using the FIR-filters is shown in Figure 5.5.

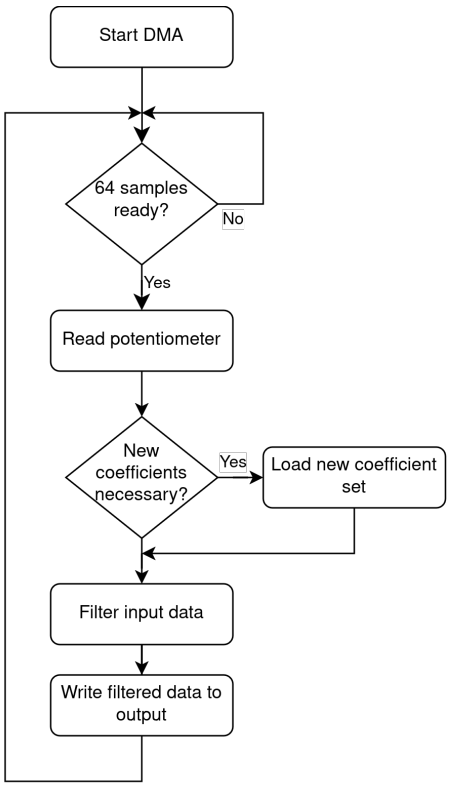


Figure 5.5: Flowchart of the signal processing using the FIR-filter

5.3.2. Implementation using biquad-filters

The second approach to model the filter is by using a biquad-filter [4]. As already mentioned, the filter coefficients are being calculated at runtime.

Therefore following equations are used [5]:

$$\omega_0 = 2\pi \cdot \frac{f_c}{f_s} \tag{5.1}$$

$$\alpha = \frac{\sin \omega_0}{2 \cdot Q} \tag{5.2}$$

$$b_0 = \alpha \tag{5.3}$$

$$b_1 = 0 \tag{5.4}$$

$$b_2 = -\alpha \tag{5.5}$$

$$a_0 = 1 + \alpha \tag{5.6}$$

$$a_1 = -2 \cdot \cos(\omega_0) \tag{5.7}$$

$$a_2 = 1 - \alpha \tag{5.8}$$

where f_c is the wanted center frequency, f_s the sampling frequency and Q the quality factor.

Designing a bandpass-filter in biquad-structure results in the frequency responses shown in Figure 5.6.

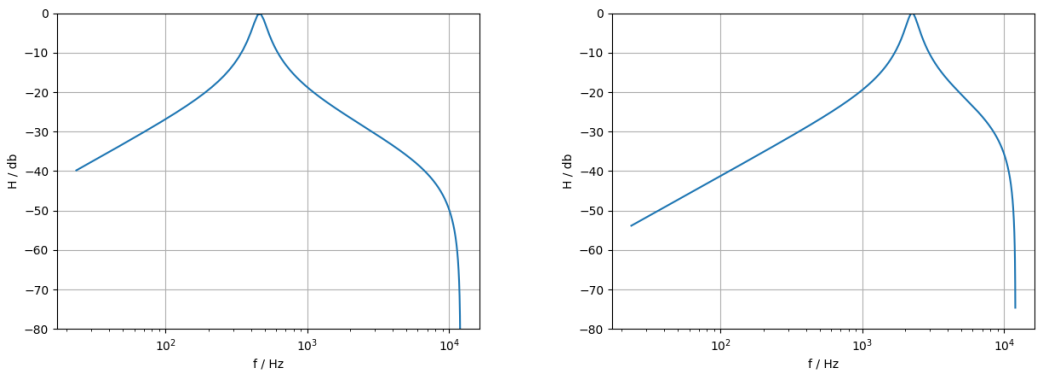


Figure 5.6: Simulation of the IIR-model frequency response at 460 Hz and 2300 Hz

It shows that the edges of a biquad-filter with five coefficients are sharper than a FIR-filter with 130 coefficients. Also it can be seen that there is a lower attenuation in the stopband in comparison to the FIR-filter implementation.

In general, the overall shape is closer to the analog reference in comparison to the non-recursive filters.

The following flowchart shows the signal processing by using the Biquad-filters.

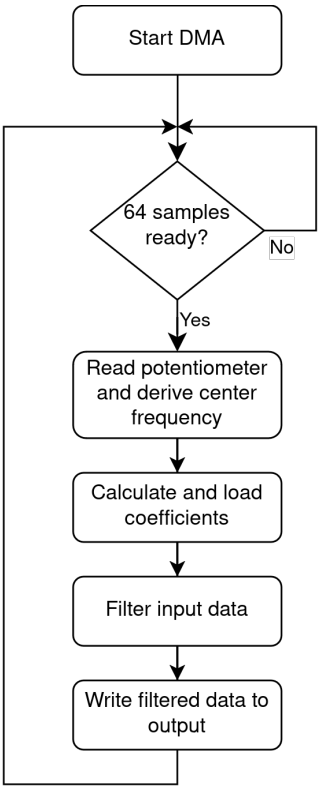


Figure 5.7: Flowchart of the signal processing using the Biquad-filter

6. Evaluation

6.1. Evaluation of the hardware

There are a few issues on the board, which have been found and fixed so far.

Issue 1:

The first issue is that no voltage was provided to the core of the microcontroller. This is, because the internal voltage regulator of this package is not connected to the core. This must be done externally. To fix this, a switched-mode power supply was added to the board, which outputs 1 V for the core.

Issue 2:

The output voltage of the extra added power supply is noisy, which led to connection errors with the debugger. In order to fix this, extra capacitors were added to the supply of the core.

Issue 3:

The input of the op-amp needs an DC-offset voltage as it has only single-supply. The offset voltage led to DC-coupling on the input port. To fix this, a decoupling capacitor was added between the input and the op-amp.

Current status:

It is possible to load firmware onto the microcontroller. Also, there is the ability to sample audio data, process it and playback the signal to the output.

A picture of the PCB with all modifications and a potentiometer connected is shown in the appendix in Figure B.1.

6.2. Evaluation of the filters

In order to validate that the implemented filters do in fact emulate a Wah-Wah-effect, the frequency response is measured. Additionally, the filters are subjected to a hearing test.

All these tests were done on the own build PCB.

6.2.1. Measurement of the FIR- and Biquad-filters

Firstly the FIR-implementation is evaluated.

To measure the frequency response, an oscilloscope with a signal generator is used. Therefore a sweep from 50 Hz to 10 kHz is applied to the input and a Fast Fourier Transform (FFT) of the output is calculated. The result of the filter at the center frequency of 460 Hz and 2242 Hz is shown in Figure 6.1. On channel 1 (yellow signal) is the output and on channel 2 (blue signal) the input signal. The FFT result is shown in purple.

Figure 6.1: Measurement of the FIR-implementation using Hamming window

The shape of the filter are as expected in comparison to the simulation (Figure 5.2). But it has to be mentioned that the center frequencies of the filters are not exactly on the desired frequencies. At the lowest center frequency there is a frequency shift of -20 Hz and at the highest -62 Hz.

Secondly, the biquad filters are measured. The results at 460 Hz and 2,3 kHz center frequency are show in Figure 6.2.

Figure 6.2: Measurement of the biquad-implementation

Also, this result is as expected except the frequency shift. There is also a frequency shift of -20 Hz at the lowest frequency and -75 Hz at the highest frequency.

Probably this is the result of a slight difference of the desired and the real sampling rate. This can be fixed by adjusting the sampling frequency when generating the coefficients.

In order to measure the latency of the system, a step was given to the input and the time difference to the step response is measured. This is shown in Figure 6.3.

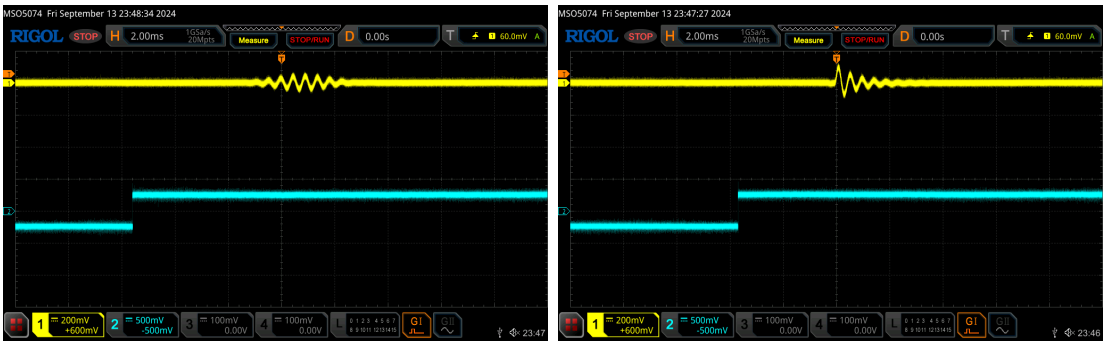


Figure 6.3: Measurement of the step response of the FIR- and Biquad filters

As it can be seen, the latency of the FIR-filter with 4,8 ms is higher than the Biquad-implementation with about 3,6 ms. Both filters fulfill the requirement of a maximum latency of 5 ms.

6.2.2. Hearing-tests of the implemented filters

The goal is to compare the FIR- with the Biquad implementation related to its audible properties. Because this is highly subjective, this is not an absolute or general result.

At the highest frequency (∼2,2 kHz) there is a noticeable difference. The FIR-implementation seems to be less broadband. This was tested by playing the same riff on the guitar, which includes notes on the low E-string and the G-, B- and high E-string. The notes on the low E-string seemed to be more attenuated than in the Biquad-implementation. This is reasonable as this is the behaviour the frequency response of the FIR-filters looks like.

If the filter at 440 Hz is applied, the difference is not as noticeable as at higher center frequencies.

The final test is the evaluation by adjusting the center frequency at runtime. Therefore a footpedal in which a potentiometer is built in, is connected to one of the analog input pins.

For the FIR-implementation there is some cracking noise on the output signal. This could be caused by loading the coefficients from a lookup table to the filter which take some time to be processed. In this time, some samples can get lost. This could be improved by for example loading the coefficients asynchronously.

The IIR-implementation works without cracking noise, as the amount of coefficients is much smaller, although they are being computed every process step.

In general the sound with the recursive filter sounds more »natural« than with the FIR-filter used.

Also the noticable latency is evaluated. A significant delay was recognized with a buffer size of 2048 samples.

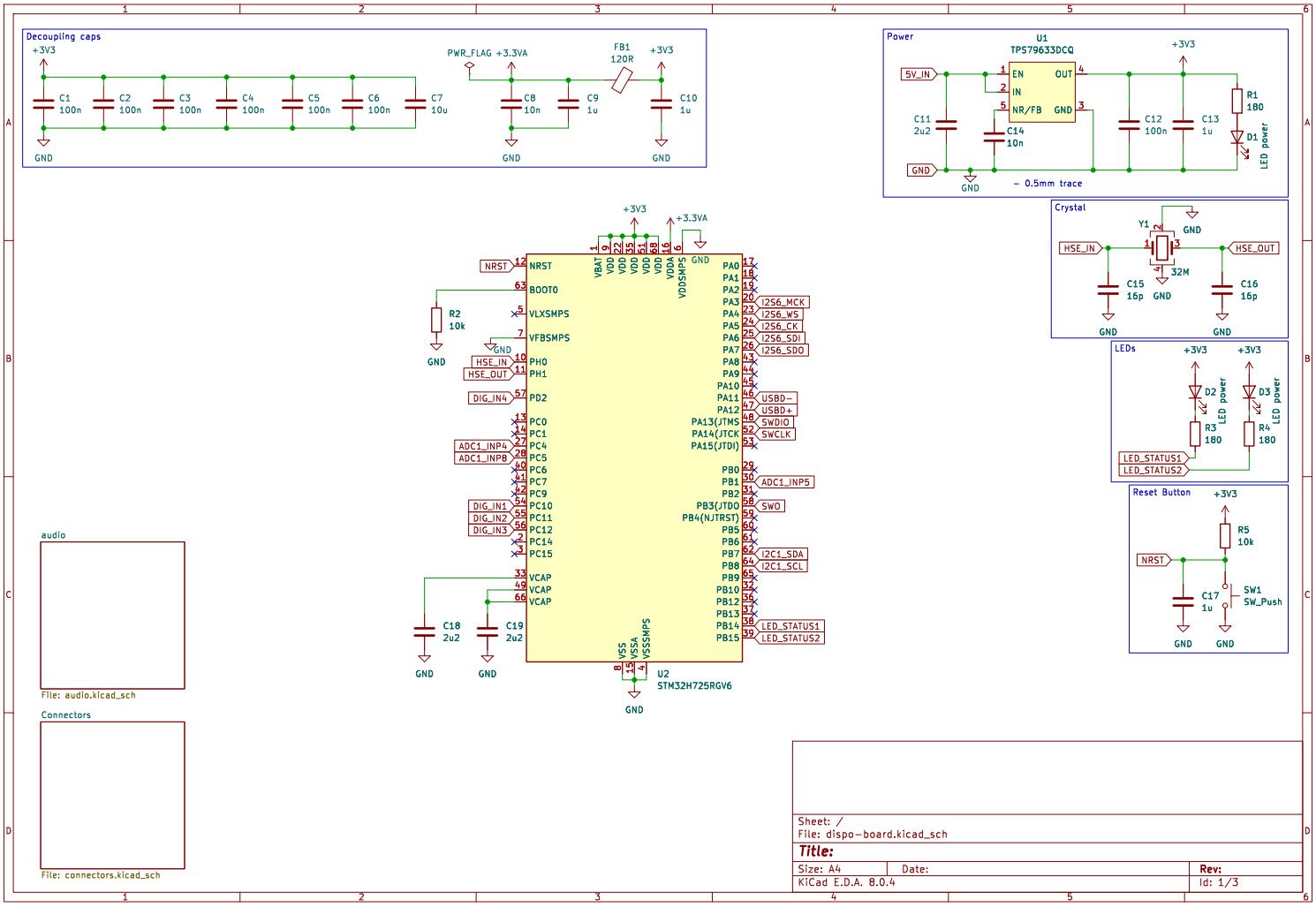
7. Conclusion and outlook

A PCB was designed, which works after fixing the main errors. There is also the possibility to further extend the functionality on this board. It is possible to implement new effects, like a delay or flanger. Also connecting a I^2C-display or adding buttons to vary between effects could be done.

To summarize, it can be said, that it is possible to build a digital guitar effects-board that emulates the filter characteristics of a Wah-Wah-effect using Biquad-filters. The adjustment works smootly, without any loss of samples. In contrast to the FIR-filter implementation, which produce cracking noise as the filter coefficients are being loaded. There is some way to fix this issue, but the filter characteristics does not model an actual analog filter, so there is no need to further investigate in this implementation.

A short description of how to use this board and all the source files can be found in the github repository `https://github.com/Velmi/guitar-dispo`.

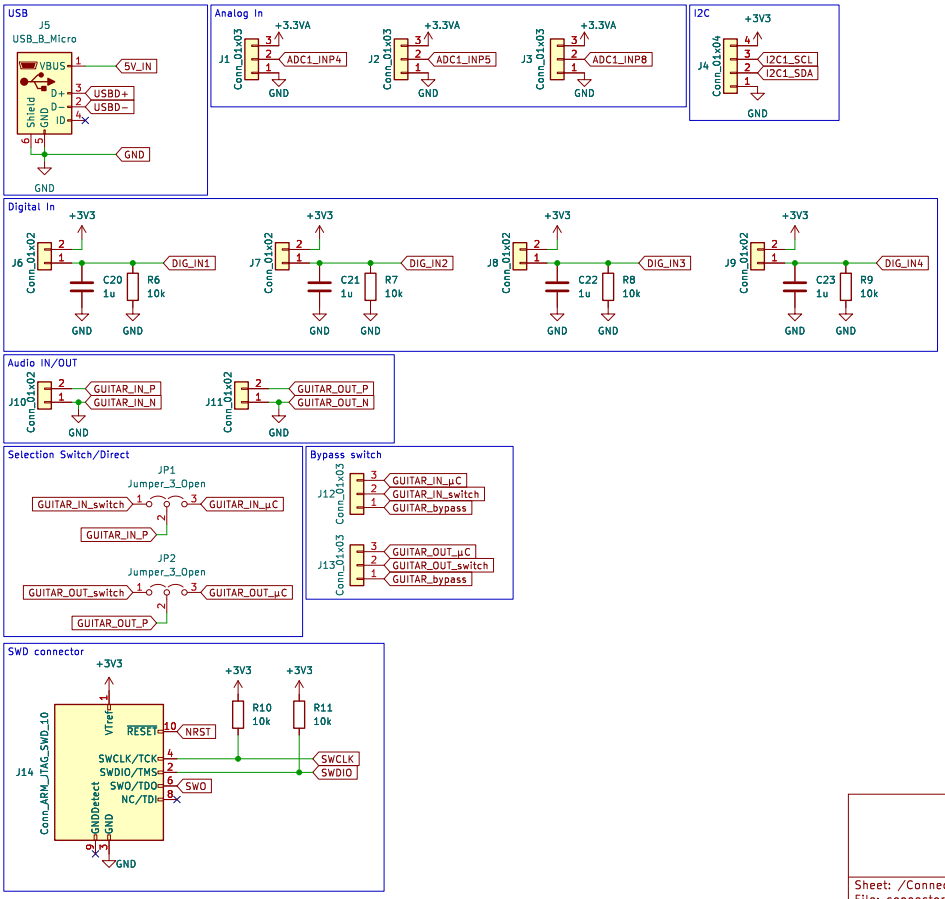
A. Schematic



Sheet: /
File: dispo-board.kicad_sch

Title:

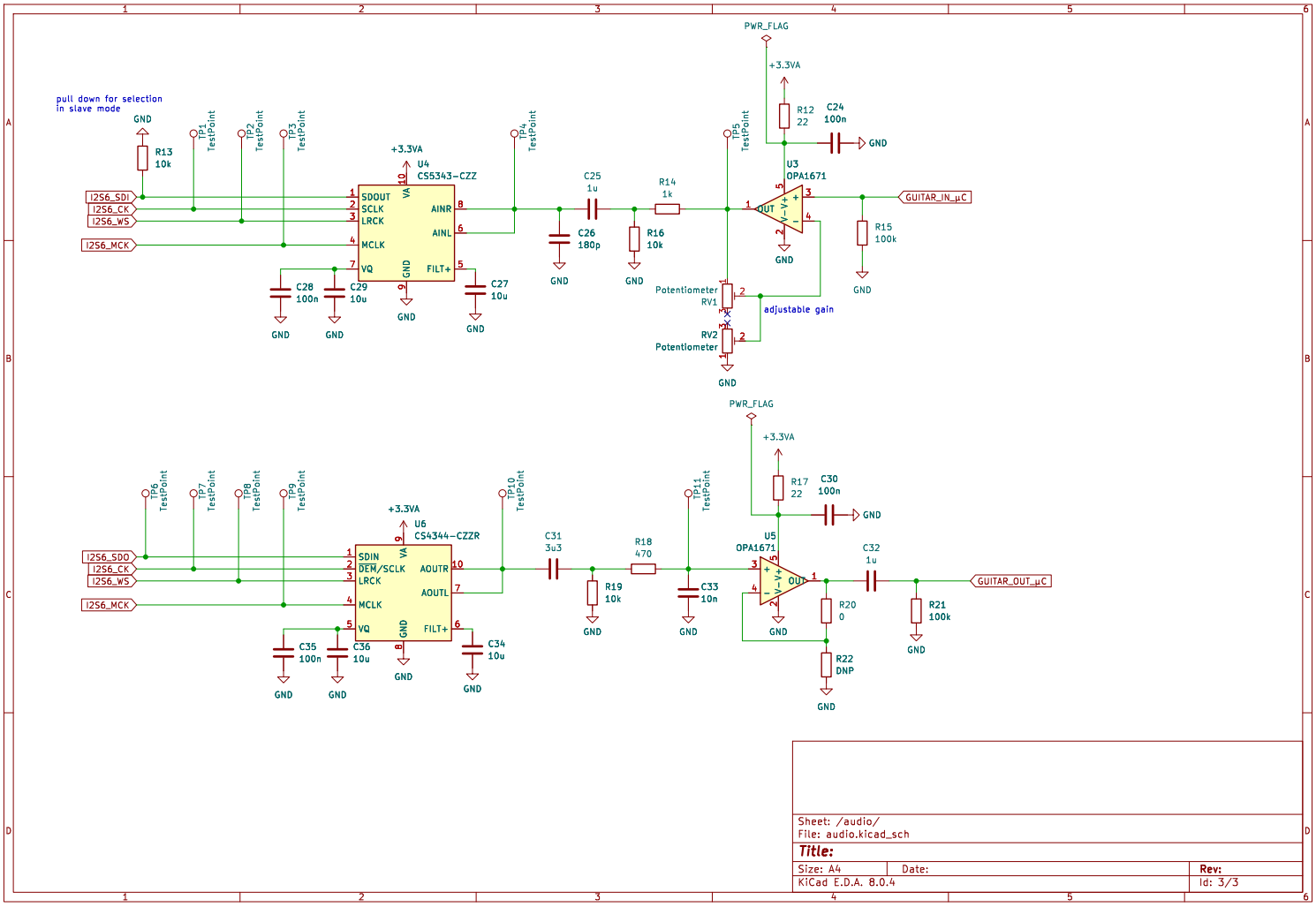
Size: A4	Date:	Rev:
KiCad E.D.A. 8.0.4		Id: 1/3



Sheet: /Connectors/
File: connectors.kicad_sch

Title:

Size: A4	Date:	Rev:
KiCad E.D.A. 8.0.4		Id: 2/3



Sheet: /audio/
File: audio.kicad_sch

Title:

Size: A4	Date:	Rev:
KiCad E.D.A. 8.0.4		Id: 3/3

B. PCB

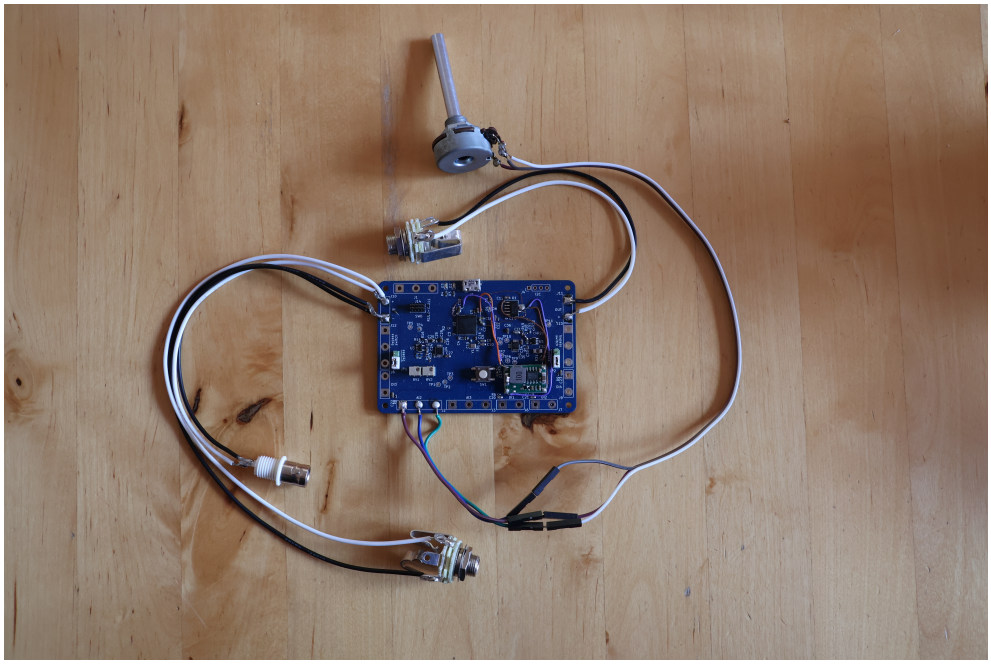


Figure B.1: The built PCB with all modifications and connectors

Bibliography

[1] D. J. Dailey, *Electronics for Guitarists*, 3rd ed. Springer, 2022.

[2] "Wah-wah." (), [Online]. Available: `https://de.wikipedia.org/wiki/Wah-Wah` (visited on 07/23/2023).

[3] M. Meyer, *Signalverarbeitung*, Analoge und digitale Signale, Systeme und Filter. Springer Vieweg, 2017.

[4] *Biquad cascade iir filters using direct form i structure*, Keil. [Online]. Available: `https://www.keil.com/pack/doc/CMSIS_Dev/DSP/html/group__BiquadCascadeDF1.html` (visited on 07/28/2024).

[5] R. Bristow-Johnson. "Cookbook formulae for audio equalizer biquad filter coefficients." (), [Online]. Available: `https://webaudio.github.io/Audio-EQ-Cookbook/audio-eq-cookbook.html` (visited on 09/12/2024).

[6] "Signal processing (scipy.signal)." (), [Online]. Available: `https://docs.scipy.org/doc/scipy/reference/signal.html` (visited on 10/20/2024).

[7] *Filter design tool*, Texas Instruments. [Online]. Available: `https://webench.ti.com/filter-design-tool/filter-response` (visited on 08/04/2024).

[8] *Um11732*, I^2S bus specification, Rev. 3, XNP Semiconductors, 2022. [Online]. Available: `https://www.nxp.com/docs/en/user-manual/UM11732.pdf` (visited on 07/23/2023).

[9] STMicroelectronics, *Rm0468 reference manual*, STMicroelectronics, 2021.

[10] P. Beckmann. "Real-time embedded audio signal processing." (2008), [Online]. Available: `https://dspconcepts.com/sites/default/files/2008-10-05_real-time_embedded_audio_signal_processing.pdf` (visited on 07/23/2023).

[11] D. Stotz, *Computergestützte Audio- und Videotechnik*, Multimediatechnik in der Anwendung, 3rd ed. Springer Vieweg, 2018.

[12] D. Katz, R. Gentile, and T. Lukasiak. "Fundamentals of embedded audio, part 3." (2007), [Online]. Available: `https://www.eetimes.com/fundamentals-of-embedded-audio-part-3/` (visited on 07/23/2023).

[13] R. Kories and H. Schmidt-Walter, *Taschenbuch der Elektrotechnik*. Europa Lehrmittel, 2017.

List of Figures

3.1. Basic structure of a M-th order FIR-filter in the first canonical form [3] . . . 3
3.2. z-plane plot (left) and frequency response (right) of a FIR-lowpass-filter . . . 4
3.3. Basic structure of a M order IIR-filter in the first canonical form [3] 5
3.4. Structure of a biquad filter [4] . . . 5
3.5. Comparison of the frequency response of a FIR- and IIR-bandpass-filter . . . 7
3.6. Comparison of the frequency response of a FIR- and IIR-bandpass-filter at different mid-frequencies . . . 8
3.7. Analog bandpass filter frequency response . . . 8
3.8. Timing diagramm of the I^2S protocol [8] . . . 9
3.9. The three basic configurations of the I^2S protocol [8] . . . 9
4.1. Basic concept . . . 12
4.2. Double buffering concept [12] . . . 12
4.3. Measurement of the frequency response of a Wah-Wah-pedal at the lowest and highest frequencies . . . 13
5.1. Blockdiagram of the PCB . . . 16
5.2. Simulation of the FIR-model frequency response at 460 Hz and 2242 Hz using the Hamming-window . . . 16
5.3. Simulation of the FIR-model frequency response at 460 Hz and 2242 Hz using the Blackman-window . . . 17
5.4. Simulation of the FIR-model frequency response at 460 Hz and 2242 Hz using the Bartlett-window . . . 17
5.5. Flowchart of the signal processing using the FIR-filter . . . 18
5.6. Simulation of the IIR-model frequency response at 460 Hz and 2300 Hz 19
5.7. Flowchart of the signal processing using the Biquad-filter . . . 20
6.1. Measurement of the FIR-implementation using Hamming window . 22
6.2. Measurement of the biquad-implementation . . . 22
6.3. Measurement of the step response of the FIR- and Biquad filters . . 23
B.1. The built PCB with all modifications and connectors . . . 30

List of Tables

5.1. Parameters for FIR-filterdesign 16