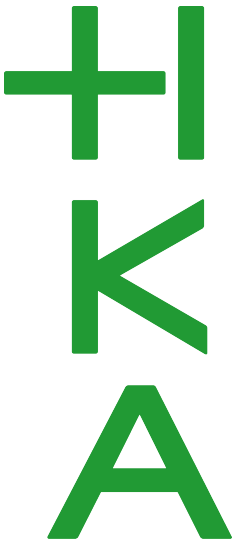
**Hochschule Karlsruhe**
University of
Applied Sciences

Fakultät für
**Elektro- und**
**Informationstechnik**



Electrical- and Information Engineering-
Information Engineering

# Project

## Implementation of a Guitar Effects Pedal on a Mikrocontroller Basis

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

## List of Abbreviations

**ADC** Analog-Digital-Converter
**CLI** Command Line Interface
**CPU** Central Processing Unit
**DMA** Direct Memory Access
**EDA** Electronic Design Automation
**FIR** Finite Impulse Response
**IC** Integrated Circuit
**IDE** Integrated Development Environment
**IIR** Infinte Impulse Response
**$I^2S$** Inter-Integrated Sound
**LTI** Linear Time Invariant
**LTV** Linear Time Variant
**MSB** Most Significant Bit
**PCB** Printed Circuit Board
**RAM** Random Access Memory
**SPI** Serial Peripheral Interface

# Contents

# 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 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

# 3 Basics

## 3.1 Digital Signal Processing

### 3.1.1 Sampling and quantization

If data from the real world should be processed in a digital system, it has to be sampled. This process has significant effects on the original signal, so they have to be considered if digital signal processing is taken seriously.

### 3.1.2 Digital Filters

#### 3.1.2.1 Finite Impulse Response Filter

Digital Filters are a huge field of digital signal processing. The most common filters are Finite Impulse Response (FIR)-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.

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

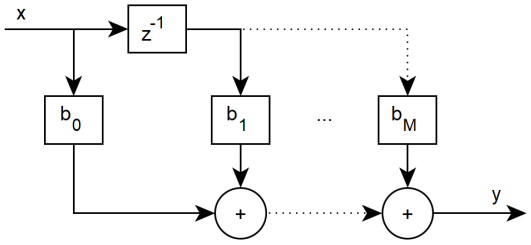



Figure 3.1: Basic structure of a $M$ order FIR-filter in the first canonical form [3]

adder, which adds $M + 1$ weighted and delayed signal samples. This sum is the output of an FIR-filter.

The input signal is $x$, whereas the output is $y$. The order of this Filter is $M$ which means the amount of filter taps or 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 union 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 filterung result is in the frequency domain. If the frequency of the input is given with $X(z)$ and the frequency response of the filter is $H(z)$ than the ouput 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 middle of the unit circle, which is necessary for the stability of a system.

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).

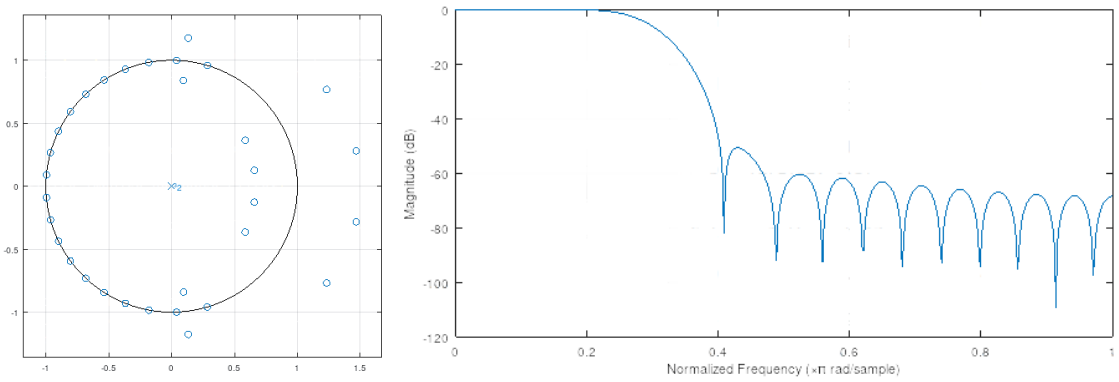



Figure 3.2: z-plane plot (left) and frequency response (right) of a FIR-filter

#### 3.1.2.2 Infinite Impulse Response Filter

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

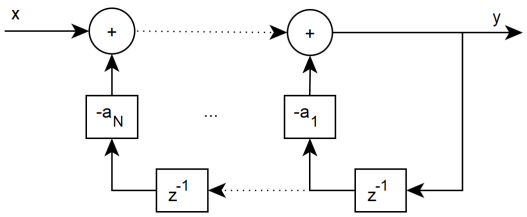



Figure 3.3: Basic structure of a $M$ order IIR-filter in the first canonical form [3]

#### 3.1.2.3 Filter Design

### 3.1.3 LTI and LTV-Systems

The filters above are all Linear Time Invariant (LTI)-systems. That means, they firstly have a linear behaviour, so filters can be combined in any way without changing the output. Secondly, the filters never change in time. This is not the case for a variable bandpass, as it changes its coefficients. Therefore it is linear, but no longer time-invariant. Such systems are then called Linear Time Variant (LTV). This may have an effect to the output signal, so this has to be evaluated.

### 3.2 Audio-Interfaces

#### 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 3.4 [4].

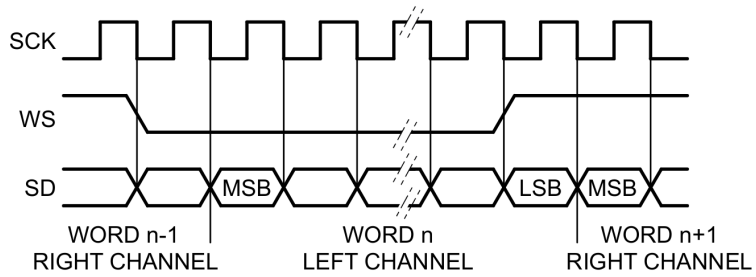



Figure 3.4: Timing diagramm of the $I^2S$ protocol [4]

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 [4].

The wiring of this serial bus is possible in three basic configurations (shown in Figure 3.5). Thereby the master is always responsible for the SCK and the WS line.

## 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

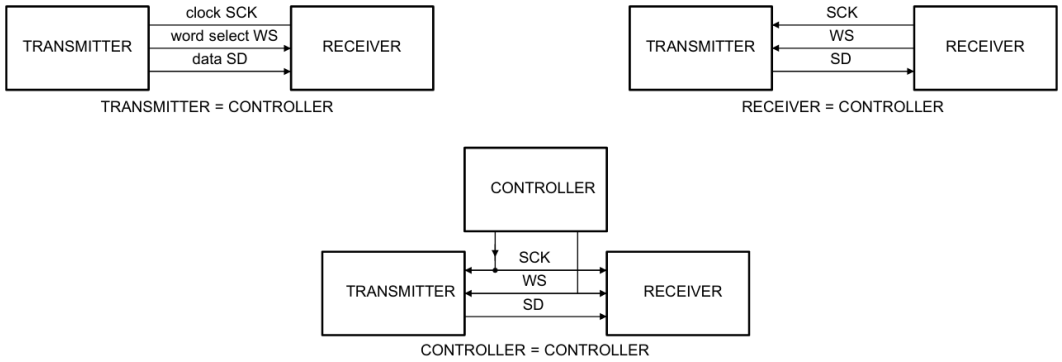



Figure 3.5: The three basic configurations of the I²S protocol [4]

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 [5].

### 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 [6].

## 4.2 Concept

The basic concept is shown in Figure 4.1.

It is to be seen, that the input data is comming from the Line In, is sampled by a codec and transmitted over I²S to the microcontroller. There the data will be processed and the output will be transmitted again over I²S to a codec. The output of the codec is again a 3,5 mm Line Jack.

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

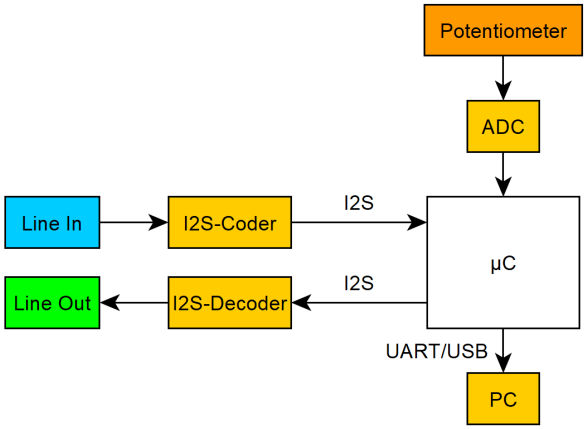



Figure 4.1: Basic concept

There is also a serial connection to a PC where a Command Line Interface (CLI) should be available to configure different parameters.

### 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 Analog-Digital-Converter (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.

### 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 [7].

An useful peripheral in the microcontroller is the Direct Memory Access (DMA). This peripheral allows a direct transfer of input data from the peripheral register (e.g. ADC-peripheral) to the Random Access Memory (RAM) without the need of the Central Processing Unit (CPU). This saves computing time, because the CPU can process the data simultaneously to the transfer done by the DMA.

The whole procedure is shown in Figure 4.2.

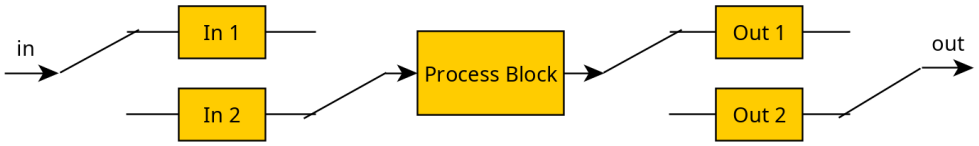



Figure 4.2: Double buffering concept [7]

# 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 3,5mm 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. As it would be too time consuming to calculate them during runtime, these coefficients where computed beforehand and then stored in the memory of the microcontroller. Therefore the software *Octave* was used.

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

## 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] *Um11732*, I²S bus specification, Rev. 3, XNP Semiconductors, 2022. [Online]. Available: `https://www.nxp.com/docs/en/user-manual/UM11732.pdf` (visited on 07/23/2023).

[5] 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).

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

[7] 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).

# List of Figures

# List of Tables