

## **HTBLuVA Salzburg**

**Höhere Lehranstalt für  
Elektronik und Technische Informatik**

# **DIPLOMARBEIT**

Gesamtprojekt

## **Gitcon**

### **Entwicklung einer MIDI-Schnittstelle für E-Gitarren**

Daniel Bräumann	5AHEL	Betreuer:
Simon Grundner	5AHEL	Prof. Dipl.-Ing. Siegbert Schrempf
Laurenz Hözl	5AHEL	

ausgeführt im Schuljahr 2022/23

---

Abgabevermerk:

Datum: 31.03.2023

übernommen von:

## Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Salzburg, am 31.03.2023

Verfasserinnen / Verfasser:

---

Daniel Bräumann

---

Simon Grundner

---

Laurenz Hözl

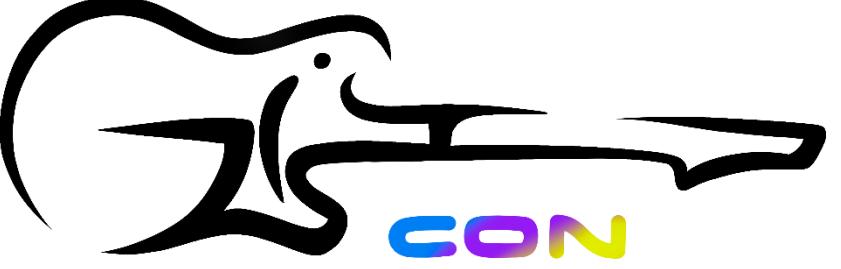
**DIPLOMARBEIT****DOKUMENTATION**

Namen der Verfasserinnen / Verfasser	Daniel Bräumann Simon Grundner Laurenz Hölzl
Jahrgang Schuljahr	5AHEL 2022/23
Thema der Diplomarbeit	Gitcon – Entwicklung einer MIDI-Schnittstelle für E-Gitarren

Aufgabenstellung	Virtuelle Instrumente sind in der modernen Musikproduktion aufgrund ihrer Vielseitigkeit weit verbreitet. Das Projekt macht es möglich, diese virtuellen Instrumente auch mit einer E-Gitarre zu spielen. Hierfür wird das analoge Audiosignal direkt von der Ausgangsbuchse der Gitarre abgegriffen, in MIDI-Noten umgewandelt und via USB an eine Digital Audio Workstation (DAW) übertragen.
------------------	---

Realisierung	Das Gitarrensignal muss zur weiteren Verarbeitung vorbereitet werden. Um die Funktion des FFT-Algorithmus zu prüfen, wurden diverse Testungen durchgeführt.
--------------	---

Ergebnisse	Das Projekt erkennt die gespielten Noten und überträgt sie ohne spürbare Latenz an die DAW.
------------	---

Typische Grafik, Foto etc. (mit Erläuterung)	 <i>Abbildung 1: Gitcon Logo</i>
---	---

Möglichkeiten der Einsichtnahme in die Arbeit	Schulbibliothek der HTBLuVA Salzburg
---	--------------------------------------

Approbation (Datum / Unterschrift)	Prüferin / Prüfer	Direktorin / Direktor Abteilungsvorständin / Abteilungs- vorstand
---------------------------------------	-------------------	---

# DIPLOMA THESIS

## Documentation

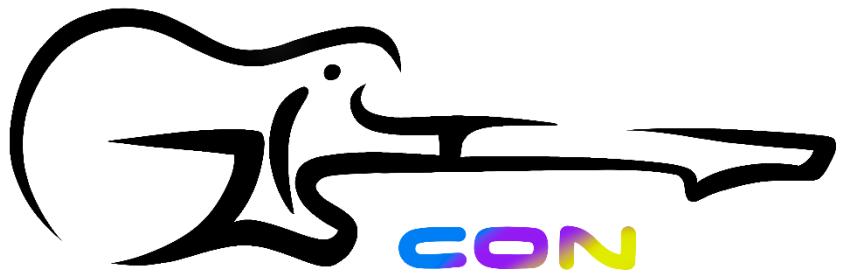
Author(s)	Daniel Bräumann Simon Grundner Laurenz Hözl
Form Academic year	2022/23
Topic	Gitcon - Development of a MIDI gateway for electric guitars

Assignment of Tasks	The present project enables the use of an electric guitar as a MIDI device. The device should reliably convert individual notes and chords into MIDI format with the lowest possible latency.
---------------------	---

Realisation	The guitar signal needs to be prepared for further processing. Various tests were carried out to verify the functionality of the FFT algorithm.
-------------	---

Results	The project detects the played notes and transfers them to the DAW without noticeable latency.
---------	--

Illustrative Graph, Photo  
(incl. explanation)



Accessibility of  
Diploma Thesis

Accessible in the library of the HTBLUvA Salzburg

Approval  
(Date / Sign)

Examiner

Head of College  
Head of Department

## Vorwort

Die vorliegende Diplomarbeit ermöglicht die Verwendung einer E-Gitarre als MIDI-Controller. Das Gerät soll einzelne Noten zuverlässig mit möglichst geringer Latenz in das MIDI-Format umwandeln. Die MIDI-Signale werden anschließend an die USB-Schnittstelle eines PCs übertragen.

Der Name „Gitcon“ ist eine Abbreviatur von „Guitar Converter“, der Aufgrund der Funktion der Platine, nämlich Gitarrensignale in MIDI-Noten zu konvertieren, gewählt wurde.

Die Projektidee kam von Simon Grundner und Laurenz Hözl. Da sich beide in ihrer Freizeit viel mit Musik befassen. Durch ihr Interesse an digitaler Audio-Verarbeitung, war die Projektidee geboren.

Moderne Musikstücke werden üblicherweise in einer digitale Produktionsumgebungen aus mehreren Tonspuren zusammen gemischt. Um mit diesen Programmen zu interagieren, kommen MIDI-Controller zum Einsatz. Diese verwenden das MIDI-Protokoll, um Noten- und Parameter-eingaben an den Computer zu übertragen. Die Idee war es nun, eine elektrische Gitarre mit diesem Protokoll kompatibel zu machen, sämtliche gespielte Noten zu erkennen und in MIDI-Signale umzuwandeln.

Die individuellen Aufgabenstellungen wurden anhand der Spezialgebiete jedes Teammitglieds gewählt. Simon beschäftigte sich mit der Entwicklung der Hardware Frontend-Platine für den Microcontroller, Daniel widmete sich dem Entwurf der analogen Signalverarbeitungskette und Laurenz arbeitete an der Implementierung der digitalen Signalverarbeitung in die Firmware.

Durch unser Projekt haben wir als Team gelernt, wie wichtig gute Kommunikation ist, um gemeinsam Ziele zu erreichen. Im Laufe der Projektarbeit traten immer wieder Schwierigkeiten auf, die wir mit Erfolg überwinden konnten. Dank der exzellenten Zusammenarbeit im Team und der großartigen Unterstützung unseres Projektbetreuers konnten wir alle Schwierigkeiten meistern und das Projekt umsetzen. Die Realisierung von Gitcon hat

nicht nur unsere Fähigkeiten zur technischen Problemlösung erweitert, sondern auch unsere Team- und Kooperationsfähigkeiten gefördert.

An dieser Stelle möchten wir uns bei allen bedanken, die diese Diplomarbeit unterstützt haben. Zunächst bei unserem Projektbetreuer Prof. Dipl.-Ing. Siegbert Schrempf, der uns nicht nur mit seiner Expertise unterstützt, sondern auch immer wieder motiviert hat. Außerdem bedanken wir uns bei Prof. Mag. Paul Schwaiger, der mit seiner Kritik geholfen hat, die Projektpräsentation weiter zu verbessern.

## Inhaltsverzeichnis

1	Überblick.....	13
1.1	Was ist ein MIDI-Controller? .....	13
1.2	MIDI-Controller vs. Synthesizer	<b>Fehler!</b> <b>Textmarke</b> <b>nicht definiert.</b>
1.3	Gitcon als MIDI-Controller .....	13
2	Systemspezifikationen .....	15
2.1	Zielbestimmungen .....	15
2.1.1	Musskriterien.....	15
2.1.2	Wunschkriterien.....	15
2.1.3	Abgrenzungskriterien .....	15
2.2	Produkteinsatz .....	15
2.2.1	Anwendungsbereiche.....	15
2.2.2	Zielgruppen.....	15
2.2.3	Betriebsbedingungen .....	15
2.3	Produktumgebung .....	16
2.3.1	Software .....	16
2.3.2	Hardware .....	16
2.4	Produktfunktionen .....	16
2.5	Produktleistungen.....	17
2.6	Benutzungsoberfläche .....	18
2.7	Entwicklungsumgebung .....	19
2.7.1	Software .....	19
2.7.2	Hardware .....	19
2.7.3	Orgware .....	19
2.8	Qualitätsziel Bestimmungen .....	20
2.9	Globale Testszenarien und Testfälle .....	20

3 Projektmanagement .....	24
3.1 Arbeitsaufteilung .....	24
3.2 GANTT-Diagramme .....	25
3.3 Projektverwaltung.....	28
3.3.1 Versionskontrolle .....	28
3.3.2 Integrierte Tools .....	29
4 Grundlagen und Methoden.....	30
4.1 Grundlagen Filter.....	30
4.1.1 Warum werden Filter benötigt? .....	30
4.1.2 Analoge Filter .....	30
4.1.3 Aktive Filter.....	31
4.1.4 Unterschied zwischen aktiven und passiven Filtern .....	32
4.1.5 Operationsverstärker.....	33
4.1.6 Unterschied zwischen analoge und digitale Filter .....	37
4.2 Aktiver Filter .....	39
4.2.1 Sallen-Key-Filter .....	39
4.2.2 Butterworth.....	42
4.2.3 Mathematische Beschreibung .....	44
4.3 Grundlagen der digitalen Signal-Verarbeitung (DSV) .....	49
4.3.1 Allgemein.....	49
4.3.2 Einleitung Fourier Transformation .....	50
4.3.3 Ursprung der Fourier Transformation .....	51
4.3.4 Herleitung der Diskreten Fourier Transformation .....	53
4.3.5 Fast Fourier Transform .....	55
4.3.6 Firmware .....	60
4.3.7 Ausgabe .....	69

4.4 C-Programmiersprache.....	72
4.4.1 Agile Softwareentwicklung mit C .....	72
4.5 ESP32 .....	73
4.5.1 Bootloader Brennen mit dem ESP-Tool.....	73
4.5.2 Einrichten in PlatformIO (PIO) .....	76
4.5.3 ESP IoT Development Framework (ESP IDF).....	81
4.6 Realtime Operating-System (RTOS) .....	81
4.7 Universal Asynchronous Receive and Transmit (UART) .....	81
4.8 Musical Instrument Digital Interface (MIDI) .....	83
4.8.1 Kommandos .....	<b>Fehler! Textmarke nicht definiert.</b>
5 Ergebnisse .....	87
5.1 Blockschaltbild .....	87
5.2 Hardware .....	89
5.2.1 Versorgung .....	89
5.2.2 Analog-Frontend (AFE) .....	93
5.2.3 Digital Frontend .....	99
5.2.4 Layout der Platine .....	102
5.3 Firmware.....	108
5.3.1 Toolchain .....	<b>Fehler! Textmarke nicht definiert.</b>
5.3.2 Treiber Firmware Diagramm.....	108
5.3.3 Doxygen .....	109
5.4 Software .....	111
5.4.1 Virtueller MIDI-Port .....	111
5.4.2 MIDI Serial Bridge.....	112
5.4.3 Ableton Live Setup .....	113
5.5 CAD-Modelle .....	114

6 Fehlererfassung .....	115
6.1 Platine.....	115
6.2 Bestückung.....	115
6.3 ADC Channel 2 auf ADC Channel 1 überbrücken .....	116
6.4 Fehlende Features .....	116
7 Glossar .....	<b>Fehler! Textmarke nicht definiert.</b>
8 Abbildungsverzeichnis.....	117
9 Literaturverzeichnis .....	<b>Fehler! Textmarke nicht definiert.</b>
10 Anhang .....	<b>Fehler! Textmarke nicht definiert.</b>
10.1 PCB-Fertigungsunterlagen .....	130
10.2 Firmware Referenz Handbuch .....	130

## 1 Überblick

Das Projekt Gitcon beschäftigt sich rundum mit digitaler Musikproduktion und mit den Werkzeugen, Audio elektronisch zu manipulieren. Das Gitcon-Device ist ein MIDI-Controller, der das auf einer E-Gitarre Gespielte analysiert und in digitale Noten, sogenannte MIDI-Noten umwandelt.

### 1.1 Was ist ein MIDI-Controller?

Wie bereits erwähnt werden in der modernen Musikproduktion digitale Produktionsumgebungen, sogenannte Digital Audio Workstations (DAW), genutzt. Diese lassen sich über das MIDI-Protokoll mithilfe eines MIDI-Controllers bedienen. Die am weitesten verbreitete Form dieses Eingabegeräts ist das Keyboard. Dieses verfügt über Klaviertasten, welche für die Noteneingabe genutzt werden und einige Regler und Knöpfe mithilfe derer Parameter in der DAW eingestellt werden können.



Abbildung 2: MIDI-Klavier (Novation<sup>1</sup> Launchkey)

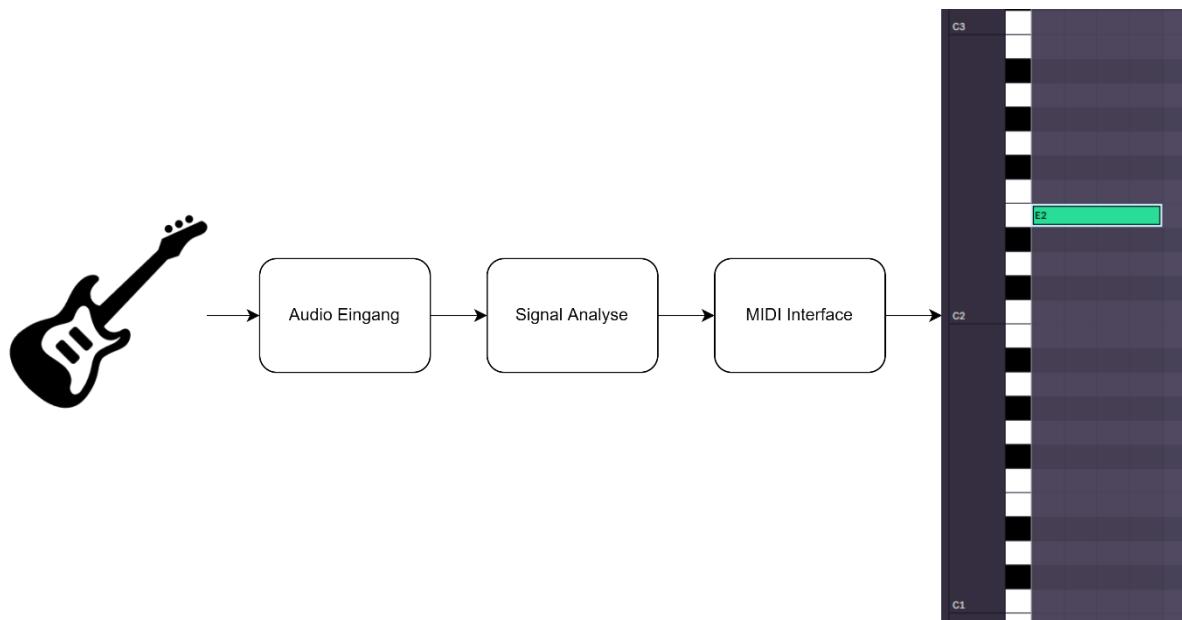
### 1.2 Gitcon als MIDI-Controller

Sowie ein MIDI-Klavier beim Tastenanschlag die Note vermittelt, soll unser Projekt eine MIDI-Note mit dem Saitenanschlag einer Gitarre senden. Wird

---

<sup>1</sup> <https://novationmusic.com/de/keys/launchkey>

beispielsweise eine E-Saite angeschlagen, so soll in der virtuellen Klaviatur (Piano Roll) der DAW die Taste „E“ aufleuchten.



## 2 Systemspezifikationen

### 2.1 Zielbestimmungen

#### 2.1.1 Musskriterien

Einzelne Noten müssen zuverlässig erkannt und umgewandelt werden.

#### 2.1.2 Wunschkriterien

Noten sollen mit möglichst geringer Latenz übertragen werden.

#### 2.1.3 Abgrenzungskriterien

Projekt soll nicht auf verschiedene E-Gitarren getestet und optimiert werden.

Das Produkt soll nicht auf Basis anderer Musikinstrumente funktionieren, welche ähnliche elektrische Ausgänge haben.

### 2.2 Produkteinsatz

#### 2.2.1 Anwendungsbereiche

Der Anwendungsbereich findet sich in der Musikproduktion als innovatives Notationstool und im Lehrbereich um Anfängern das Erlernen des Notenlesens zu erleichtern.

#### 2.2.2 Zielgruppen

Zielgruppen sind sowohl Musikproduktions-Neueinsteiger, welche Gitarre spielen als auch bereits erfahrene Produzenten, welche auf der Suche nach einzigartigen und inspirierenden Eingabemethoden sind.

#### 2.2.3 Betriebsbedingungen

Die Versorgung sowie die Datenübertragung erfolgen über USB. Hierzu wird eine E-Gitarre via einer 6,3mm Buchse an die Platine angeschlossen. Da kein Überspannschutz vorliegt darf der Eingang nur mit einer geringen Leistung beschalten werden.

## 2.3 Produktumgebung

### 2.3.1 Software

- Silicon Labs VCP Driver
- Hairless MIDI (v0.4)
- LoopMIDI (v1.0.16)
- Ableton Live Suite (v11.2.7)
- Gitcon Firmware

### 2.3.2 Hardware

- Selbst entwickelte ESP32 32-Bit Mikroprozessor Platine
- 6,3mm Mono Audio Klinkenstecker
- Micro USB (AB) Male zu USB (A) Male
- Gitarre mit elektrischem Ausgang

## 2.4 Produktfunktionen

### /F0010/ Transienten Erkennung:

Es wird erkannt, ob eine neue Saite angeschlagen wird.

### /F0020/ Noten Erkennung:

Die Note der angeschlagenen Saite wird erkannt und anschließend in das MIDI-Format konvertiert.

### /F0030/ USB Kommunikation:

Das Gerät wird als MIDI Device in der DAW erkannt, uns sendet die mit der Gitarre Gespielten Noten an einen Kanal des virtuellen MIDI-Ports am PC.

### /F0050/ Ausgabe:

Eingelesene Noten werden bei aktivierter Aufzeichnung in der DAW auf Pianorolls angezeigt und gespeichert.

## 2.5 Produktleistungen

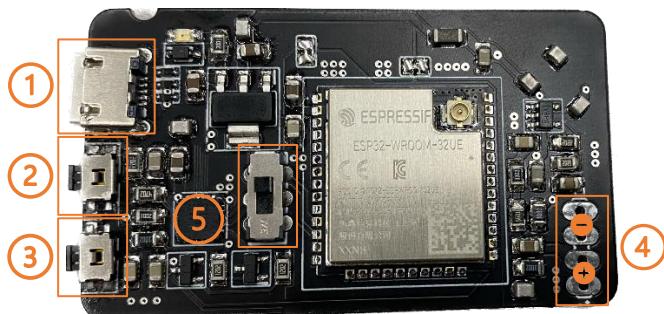
### L010/ Latenz:

Die Noten sollen ohne große Verzögerungen ankommen und so einen Liveeinsatz ermöglichen.

### L020/ Genauigkeit:

Das Signal soll zuverlässig in die richtigen Frequenzen aufgespalten werden.

## 2.6 Benutzungsoberfläche



### /B001/ USB-Buchse

An der Micro-USB Buchse werden die an den PC zu übertragenden Daten bereitgestellt. Die USB-Schnittstelle wird zum Programmieren, sowie zum Übertragen von MIDI-Daten benutzt.

### /B002/ Boot-Mode Taster

Wird der ESP32, während der Boot-Mode Taster gedrückt ist, zurückgesetzt, so wechselt er in den Download-Modus. Im Download-Modus kann dann eine neue Firmware auf den ESP32 gespielt werden.

### /B003/ Reset Taster

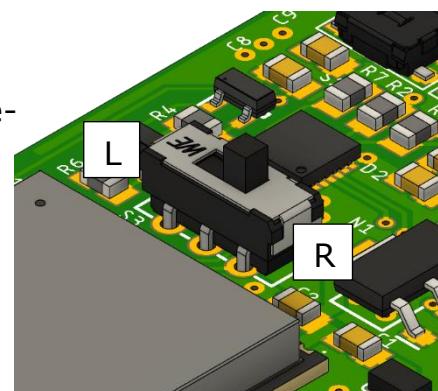
Der Reset-Taster setzt den ESP32 zurück.

### /B004/ Analoger Eingang

Am analogen Eingang werden die beiden 6,3mm Mono-Audio-Buchsen, welche am Gehäuse befestigt sind, angeschlossen.

### /B005/ Slider-Switch

Mit dem Slider Switch, wird der UART-Kanal ausgewählt. Ist der Slider-Switch in der Links-Stellung (L), so kann der ESP32 über die USB-Schnittstelle programmiert bzw. gedebuggt werden. Hat man die Rechts-Stellung (R) selektiert, ist man im MIDI-Übertragungsmodus. Auf die Schnittstelle werden nun sämtliche erkannte Noten Übertragen.



## 2.7 Entwicklungsumgebung

### 2.7.1 Software

- PlatformIO (Core v6.1.6, Home v3.4.3)
- ESP IoT Development Framework (v5.3.0)
- Autodesk EAGLE (v9.6.2)
- Autodesk Fusion 360 (v2.0.15509)
- ESP Flash Download Tool (v3.9.4)
- LTSpice (XVII)
- Saturn PCB Toolkit (v8.23)
- Ableton Live Suite (v11.2.7)
- Audacity

### 2.7.2 Hardware

- Prototypen
  - Firmware Test-board
  - Filter Prototyp

### 2.7.3 Orgware

- GitHub Desktop (v3.2.0)
  - (<https://github.com/s-grundner/MTAP-MIDI-Guitar-Converter>)
- DrawIO/diagrams.net
- Obsidian (v1.1.9)
- Pro Create (v5.3.1)

## 2.8 Qualitätsziel Bestimmungen

	sehr wichtig	wichtig	weniger wichtig	unwichtig
Robustheit			x	
Zuverlässigkeit	x			
Korrektheit	x			
Benutzungsfreundlichkeit			x	
Effizienz	x			
Portierbarkeit		x		
Kompatibilität			x	

## 2.9 Globale Testszenarien und Testfälle

### T0020/ Noten Erkennung: FFT-Unitest

Die Note der angeschlagenen Saite wird erkannt und anschließend in das MIDI-Format konvertiert. Siehe: 4.3.7 Ausgabe p. 69

### T0030/ USB Kommunikation: MIDI Unitest

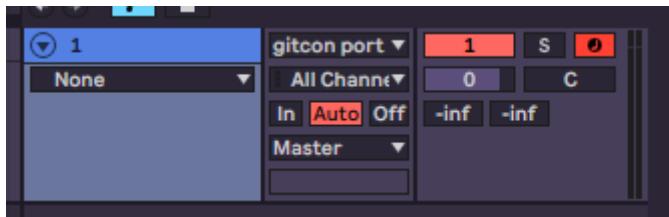
Das Gerät wird als MIDI Device in der DAW erkannt und sendet die mit der Gitarre Gespielten Noten an einen Kanal des virtuellen MIDI-Ports am PC. Die USB-Kommunikation und die Übertragungskapazität wurde mittels eines Unitest am ESP32 geprüft. Der Unitest sendet Unterschiedliche MIDI-Signale auf den MIDI-Kanal 1:

Status	Parameter 1	Parameter 2
MIDI Note On	Note: C4	Velocity: 127
MIDI Note Off	Note: C4	Velocity: 0
Pitch Bend		0 (Min)
Pitch Bend		16383 (Max)
Pitch Bend		8192 (Mitte)
MIDI Note On	Note: C4	Velocity: 127

Pitch Bend	Aufwärtsrampe (8192 (Mitte) bis 16383 (Max))	
Pitch Bend	Abwärtsrampe ()	
Pitch Bend	Aufwärtsrampe (0 (Mitte) bis 8192 (Mitte))	
MIDI Note Off	Note: C4	Velocity: 0

*Tabelle 1: MIDI-Unitest*

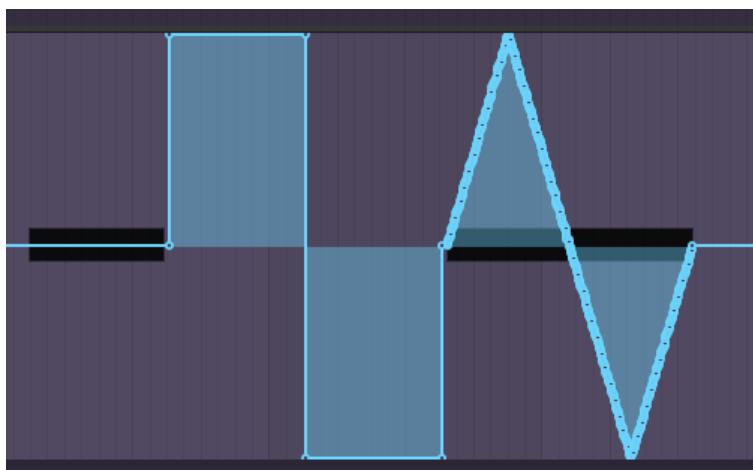
Eine MIDI-Spur wird mit folgenden Einstellungen zur Aufnahme scharfge stellt:

*Abbildung 3: Konfiguration einer MIDI-Spur beim MIDI-Unitest*

Anschließend wird die Aufnahme gestartet:



*Abbildung 4: Starten der Aufnahme des MIDI-Unitests* Das in Abbildung 5 Ergebnis wird erwartet:

*Abbildung 5: Ergebnis einer Aufnahme des MIDI-Unitests*

### /T0050/Ausgabe: Live-Konversion

Eingelesene Noten werden bei aktiver Aufzeichnung in der DAW auf Panorolls angezeigt und gespeichert.

Eine MIDI-Spur wird mit folgenden Einstellungen zur Aufnahme scharfgestellt:

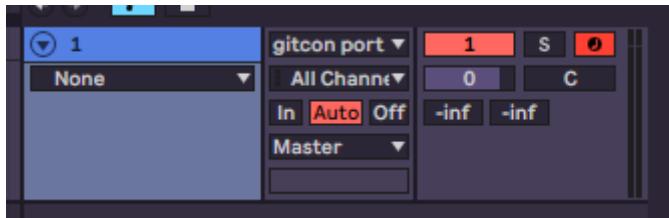


Abbildung 6: Konfiguration einer MIDI-Spur beim Live-Konversion Test

Anschließend wird die Aufnahme gestartet:



Abbildung 7: Starten der Aufnahme des Live-Konversionstest

Beim Anschlagen der tiefen E-Saite wurde folgendes Ergebnis aufgezeichnet

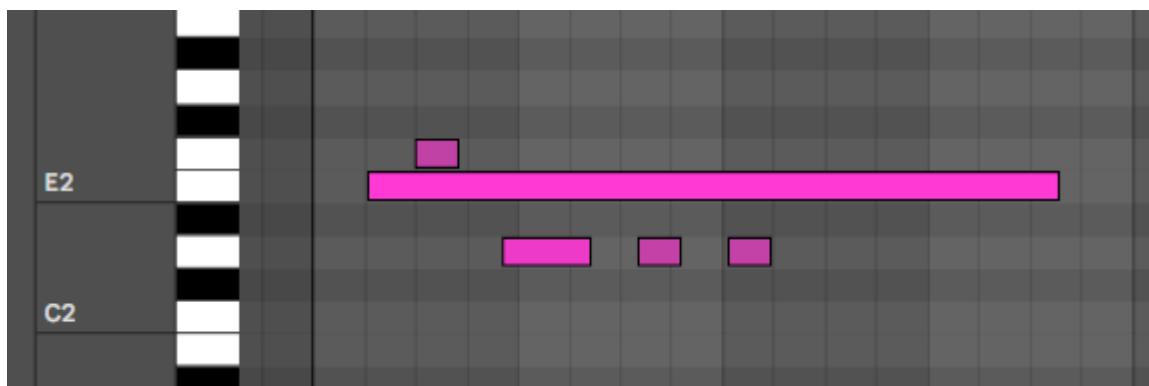


Abbildung 8: Ergebnis der Live-Konversion

### 3 Projektmanagement

#### 3.1 Arbeitsaufteilung

Daniel Bräumann

- Implementierung der analogen Signalverarbeitungskette
- Entwurf des Tiefpassfilters und Verstärkers
- Vermessung des analogen Frontend

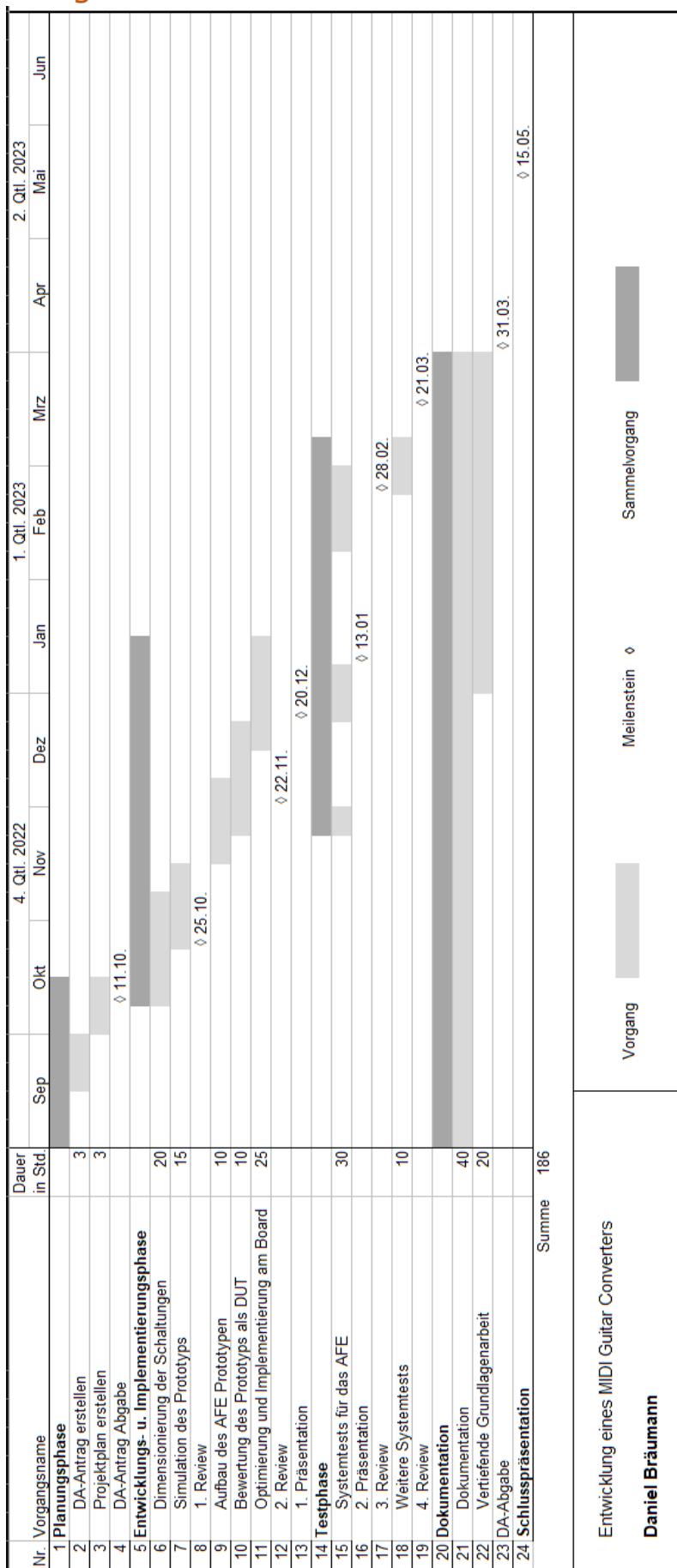
Simon Grundner

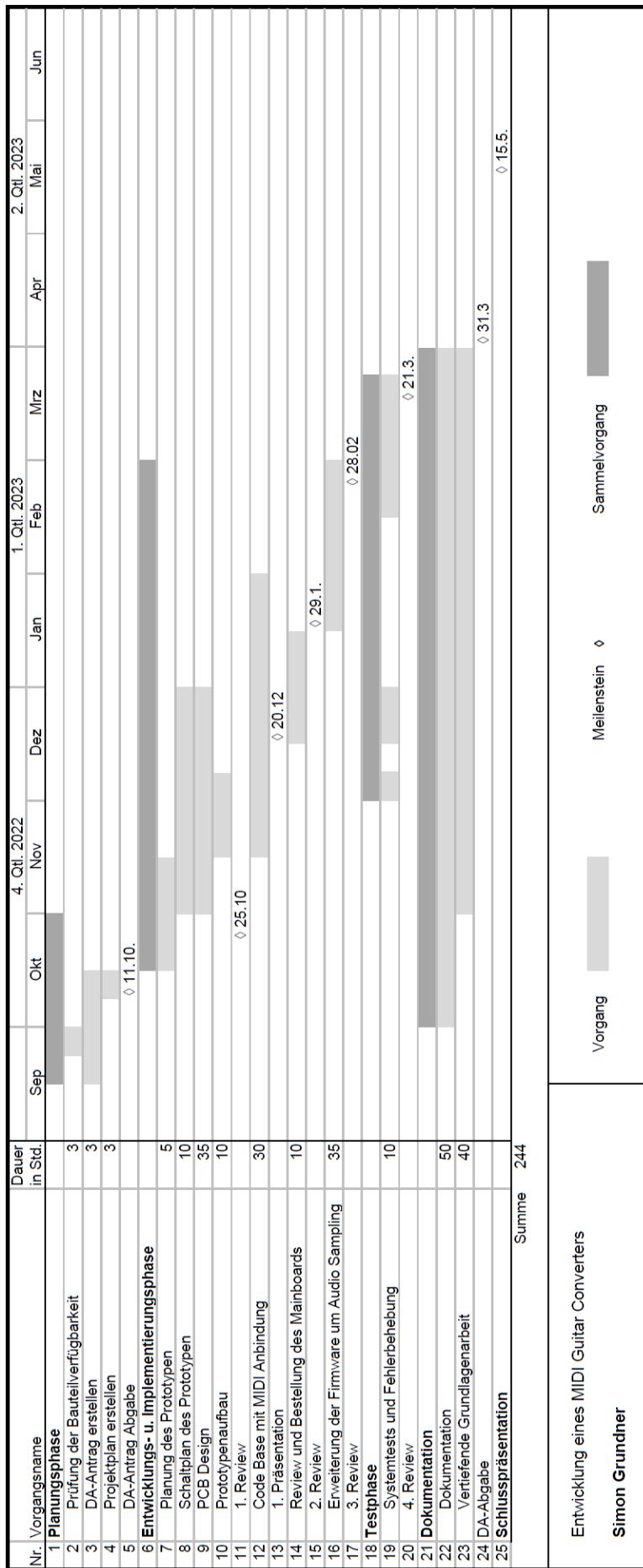
- Schaltungsentwurf
- Prototypenbau
- Platinen Entwurf
- Code-Grundgerüst und Kernel
- Audiosampling
- MIDI-Interfacing

Laurenz Hözl

- Implementierung der FFT
- Fertigung der Testdaten
- Finetuning der Sample- und FFT-Parameter

### 3.2 GANTT-Diagramme





Nº	Vorlagenname	Dauer in Std.	4. Qtr 2022			1. Qtr 2023			2. Qtr 2023		
			Sep	Okt	Nov	Dez	Jan	Feb	Mar	Apr	Mai
1	Planungsphase										
2	Recherche zur Umsetzbarkeit	3									
3	DA-Antrag erstellen	3									
4	Projektplan erstellen	3									
5	DA-Antrag Abgabe										
6	Entwicklungs- u. Implementierungsphase										
7	Recherche zur FFT	5									
8	Aufsetzen der Entwicklungsumgebung	10									
9	1. Review										
10	FFT Algorithmus	30									
11	Testdatenfertigung	10									
12	Überprüfung mit Matlab	3									
13	2. Review										
14	1. Präsentation										
15	Noten Umwandlung & Übertragung	25									
16	2. Präsentation										
17	3. Review										
18	Testphase										
19	Test und Feinjustierung der Firmware	15									
20	4. Review										
21	Dokumentation										
22	Dokumentation	60									
23	Vertiefende Grundlagenarbeit	30									
24	DA-Abgabe										
25	Schlusspräsentation										
	Summe	197									
Entwicklung eines MIDI Guitar Converters			Vorgang								
Laurenz Hözl											
Mallenstein											
Sammelvorgang											

### 3.3 Projektverwaltung

Um das Projekt während der Entwicklungsphase zu verwalten, wurde GitHub verwendet. GitHub ist eine weitverbreitete, webbasierte Plattform, mit welcher man das Projekt, in einer sogenannten „**Repository**“ ablegen und von dort aus verwalten kann. Besonders bei Softwareprojekten erleichtert GitHub den Arbeitsfluss immens, da die Plattform einige ausschlaggebende Benefits mit sich bringt.

#### 3.3.1 Versionskontrolle

GitHub ermöglicht einen strukturierten Arbeitsfluss. Ein sogenannter „**Commit**“ wird ausgeführt, wenn ein Kollaborator mit einem Arbeitsschritt fertig ist. Dieser Commit wird mit einer Überschrift, welche den Arbeitsschritt kurz beschreibt, einer detaillierten Beschreibung und einer Prüfsumme versehen. Mithilfe der Prüfsumme kann zu jederzeit auf diesen Commit referiert werden. Dies stellt sicher, dass Änderungen am Projekt fortlaufend dokumentiert sind.

Die Plattform ermöglicht das Erstellen von Abzweigungen vom Hauptprojekt, im folgenden „**Branches**“ genannt, mit denen Änderungen vorgenommen werden können, ohne dass das beständige Projekt verändert wird. Wenn nun während der Entwicklung kritische Änderungen erfolgen, zum Beispiel ein Refactor<sup>2</sup> oder ein neues Feature, ist man gegen mögliche fatale Fehler abgesichert.

Sobald ein Kollaborator seine Arbeit an einem Branch erledigt hat, wird eine **Pull-Request** erzeugt bei dem der Kollaborator eine Anfrage an den Administrator oder Projektleiter der Repository schickt, um den Code in das Hauptprogramm zu integrieren. Nach der Genehmigung kommt der Merge-Algorithmus von GitHub zum Einsatz, welcher neue Änderungen in die Code-Base einfügt. Kann der Algorithmus den Code nicht automatisch integrieren, kann auch manuell entschieden werden, ob ein Code-Abschnitt in den Main-Branch vereinigt wird oder nicht.

---

<sup>2</sup> Variablen und Funktionsnamen anpassen, um der Funktionalität zu entsprechen

### 3.3.2 Integrierte Tools

#### 3.3.2.1 Trello

Trello ist eine webbasierte Projektmanagement-Software, welche es dem Team ermöglicht hat, die einzelnen Aufgaben visuell zu organisieren. Mittels einer flexiblen Board-Struktur können Karten mit den Arbeitsaufgaben in Listen organisiert werden.

Die Karten können mit Checklisten, Fälligkeitsdaten und Benutzerzuweisungen versehen werden, um den Überblick über die Aufgaben und Fortschritt zu behalten.

Auf der Website der Repository sind unter dem Reiter „Projects“ die Trello-Boards für das Projekt vorhanden.



Abbildung 9: Trello Liste mit zwei Karten

#### 3.3.2.2 GitHub Copilot

GitHub Copilot ist ein auf künstlicher Intelligenz (KI)-basiertes Tool von GitHub und kann als Erweiterung für den Visual Studio Code Editor installiert werden. Copilot ist darauf optimiert, Code anhand der bereits eingegebenen Informationen in der Datei zu generieren oder vervollständigen. Dieses Werkzeug war beim Entwickeln eines Code-Prototypen sehr hilfreich. Es ist jedoch nötig, die generierten Codeblöcke immer zu validieren.



Abbildung 10: Copilot Logo

## 4 Grundlagen und Methoden

### 4.1 Grundlagen Filter

#### 4.1.1 Warum werden Filter benötigt?

Filter werden in der Elektronik verwendet, um unerwünschte Frequenzen aus einem Signal zu entfernen oder zu reduzieren. Ein Filter kann in der Lage sein, ein bestimmtes Frequenzband durchzulassen und alles außerhalb dieses Bereichs zu unterdrücken, oder bestimmte Frequenzen zu unterdrücken und den Rest durchzulassen.

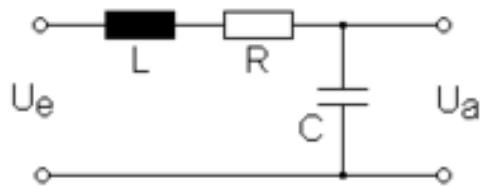
Filter werden in vielen Bereichen der Elektronik eingesetzt, wie zum Beispiel in der Tontechnik, wo sie zur Entfernung von Rauschen und anderen Störungen verwendet werden. Ein anderer Anwendungsfall ist in der Datenkommunikation, wo sie zur Verarbeitung von Signalen und zur Entfernung von Interferenzen eingesetzt werden.

Insgesamt sind Filter ein wichtiges Werkzeug in der Elektronik, um die Qualität von Signalen zu verbessern und unerwünschte Störungen zu entfernen.

#### 4.1.2 Analoge Filter

##### 4.1.2.1 *Passive Filter*

Ein passiver Filter besteht ausschließlich aus passiven Bauelementen, wie Widerstand, Spule und Kondensator. Mithilfe dieser Bauteile kann man keine Leistungsverstärkung erzielen. Weiters benötigt dieser Filter keine externe Stromquelle für den Betrieb. Da die Eingangsimpedanz niedrig und die Ausgangsimpedanz hoch ist, wird eine Selbstregulierung der Spannungen ermöglicht, die die Lasten antreibt.



*Abbildung 11: passiver LCR-Tiefpass [35]*

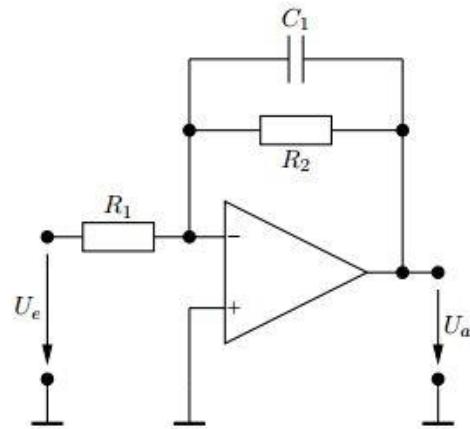
Der Lastwiderstand ist normalerweise nicht vom restlichen Netzwerk isoliert, dadurch kann die Charakteristik der Schaltung und der Filterprozess bei Veränderung der Last beeinflusst werden. Da ein passiver Filter keine Bandbreitenbeschränkungen aufweist, ermöglicht dies einen zufrieden-

stellenden Betrieb bei sehr hohen Frequenzen. Allerdings tendiert der in der Schaltung verwendete Induktor bei niedrigeren Frequenzen größer zu sein, wodurch die gesamte Schaltung komplexer wird. Außerdem steigen die Kosten, wenn eine höhere Qualität und eine kleine Größe erwünscht ist. Weiters erzeugen passive Filter aufgrund des thermischen Rauschens in den Elementen ebenfalls ein hörbares Rauschen. Jedoch kann dies bei richtiger Auslegung der Bauteile minimiert werden.

Weil keine Verstärkung vorhanden ist, muss diese zu einem späteren Zeitpunkt durchgeführt werden. Dazu werden oft Pufferverstärker verwendet, um die Differenzen in der Ausgangsschaltung zu kompensieren.

#### 4.1.3 Aktive Filter

Im Gegensatz zu passiven Filtern, die nur aus passiven Bauelementen bestehen, kommen bei aktiven Filtern Transistoren oder Operationsverstärker zum Einsatz, außerdem werden keine Induktoren verwendet. Anders als bei passiven Filtern benötigen aktive Filter aufgrund der energieverbrauchenden, aktiven Elemente eine externe Stromquelle. [1]



*Abbildung 12: aktiver Tiefpass [36]*

Da keine Induktoren zum Einsatz kommen, wird die Schaltung kompakter und weniger schwer. Die Eingangsimpedanz ist hoch und die Ausgangsimpedanz ist niedrig, so können niedrige Lasten am Ausgang angesteuert werden. Weiters ist die Last von der internen Schaltung isoliert, daher hat die Veränderung der Last keinen Einfluss auf die Charakteristik des Filters.

Das Ausgangssignal hat eine Leistungsverstärkung, auch können die Parameter wie Verstärkung und Grenzfrequenz beliebig angepasst werden. Probleme bei aktiven Filtern sind, dass eine Änderung in der Stromversorgung eine Änderung der Ausgangssignalgröße verursachen kann, weiters

werden die Hochfrequenzbereiche durch die Eigenschaften der aktiven Elemente begrenzt. Außerdem können Rückkopplungsschleifen, die zur Regelung der aktiven Komponenten verwendet werden, zu Schwingungen und Rauschen beitragen.

#### 4.1.4 Unterschied zwischen aktiven und passiven Filtern

- Bei passiven Filtern wird die Energie des Signals verbraucht, es ist jedoch keine Leistungsverstärkung verfügbar, während bei aktiven Filtern eine Leistungsverstärkung zur Verfügung steht.
- Aktive Filter benötigen eine externe Stromquelle. Passive Filter arbeiten nur am Signaleingang.
- Ausschließlich passive Filter verwenden Induktivitäten.
- Aktive Filter verwenden Transistoren und Operationsverstärker, die aktive Bauelemente sind.
- Passive Filter haben theoretisch keine Frequenzbegrenzungen, während aktive Filter von den aktiven Elementen eine Einschränkung aufweisen.
- Außerdem sind passive Filter etwas stabiler und können auch großen Strömen standhalten.
- Passive Filter sind preiswerter als aktive Filter, jedoch sind aktive Filter meist kompakter.

#### 4.1.5 Operationsverstärker

Operationsverstärker kurz OPV sind eine der mächtigsten Bauteilen in der Elektronik. Dieses Thema ist sehr komplex und man könnte allein darüber eine ganze wissenschaftliche Arbeit verfassen, weshalb an dieser Stelle nur das Grundkonzept im Kontext der Filterthematik vorgestellt werden soll. OPVs sind elektronische Verstärker, deren Name sich vom mathematischen Begriff „Operator“ ableiten lässt. Sie werden vielfältig in der Audiotechnik verwendet und ersetzen häufig diskret aufgebaute Schaltungen mit nur einen integrierten Schaltkreis (IC) . Operationsverstärker haben meist zwei Eingänge und einen Ausgang, worüber Signale verstärkt, addiert, subtrahiert, integriert, differenziert und geschaltet werden. Der OPV kommt oft als Differenzenverstärker zum Einsatz, sodass ein Signal an einem Eingang anliegt, welches über den Ausgang zum Eingang zweier rückgekoppelt wurde. Damit lassen sich schnell und unkompliziert mit Hilfe von Widerständen und Kondensatoren Verstärkerschaltungen realisieren. Weiters kommen zu diesen Ein- und Ausgängen noch Anschlüsse für die Spannungsversorgung hinzu. Man spricht hier von aktiven Bauteilen, welche eine Verstärkung bereitstellen können, also wird auch hier eine Versorgungsspannung benötigt.

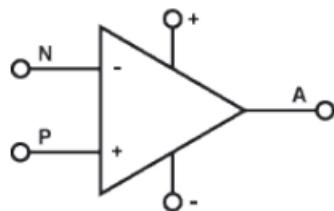


Abbildung 14: Operationsverstärker  
schaltbild

Die obige Grafik zeigt das Schaltbild eines Operationsverstärkers. P ist der nicht-invertierende, N ist der invertierende Eingang. A ist der Ausgang, plus (+) und minus (-) sind die Eingänge für die Versorgungsspannung. [2]

#### 4.1.5.1 Eigenschaften des OPVs

Operationsverstärker sind besonders vielseitig einsetzbar. Ihre Funktionen hängen hauptsächlich von der externen Beschaltung des OPVs ab [2]. Im Vergleich zu einem herkömmlichen Verstärker ist der Unterschied gar nicht so groß. Während ein Transistorverstärker festgelegte Eigenschaften aufgrund seiner Schaltung aufweist, wird die Funktionsweise eines Operationsverstärkers erst durch seine externe Beschaltung definiert [2]. Die Eingänge N und P sowie der Ausgang A beziehen sich auf ein gemeinsames Massepotenzial. Der OPV kann eine Spannungsdifferenz zwischen den Eingängen N und P verstärken, wobei der Verstärkungsfaktor  $v$  durch die äußere Beschaltung bestimmt wird. Die Ausgangsspannung ist in Phase mit der Eingangsspannung am nicht-invertierenden Eingang.

OPVs haben ähnlich wie herkömmliche Verstärker Kennlinien, die die Verstärkungscharakteristik beschreiben. Diese Kennlinien werden als "Ausgangssteuierbarkeit" bezeichnet. Da es keine perfekten OPVs mit unendlicher Verstärkung gibt, ist es wichtig, diese Kennzahlen zu berücksichtigen, um die Schaltung in einem vernünftigen Arbeitspunkt zu betreiben. Dies hilft, Verzerrungen, Sättigungs- und Clipping-Effekte zu vermeiden. [2]

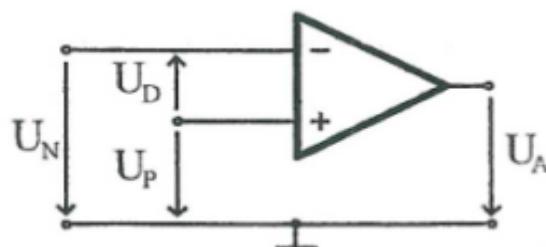


Abbildung 15: Schaltbild OPV

$$\text{Für die Differenzverstärkung gilt: } U_D = U_P - U_N \quad (1)$$

$$\text{Für die Ausgangsspannung gilt: } U_a = v * U_D = v * (U_P - U_N) \quad (2)$$

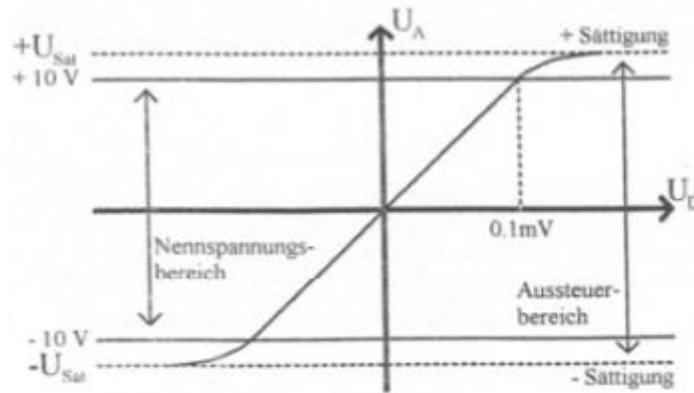


Abbildung 16: Differenzenverstärkung OPV

In dieser Grafik wird die Differenzverstärkung eines Operationsverstärkers, die auch als Leerlaufverstärkung bekannt ist, dargestellt. Diese Verstärkung hängt von der Versorgungsspannung des OPVs ab und hat ihre Grenzen im Sättigungsbereich, der sowohl im Negativen als auch im positiven Bereich auftritt. Im Idealfall ist die Leerlaufverstärkung linear und frequenz-unabhängig, aber bei einem realen Operationsverstärker gibt es aufgrund seiner Bandbreite eine obere Grenzfrequenz. Obwohl die Bandbreite bei 0 Hz beginnt, gibt es bestimmte Grenzen, die zu beachten sind. [2]

#### 4.1.5.2 Invertierender und nicht-invertierender OPV

Damit ein Operationsverstärker als Differenzverstärker verwendet werden kann, muss ein Teil des Ausgangssignals an einen der Eingänge zurückgeführt werden. [3] Dies wird als Gegenkopplung oder Feedback bezeichnet. Zunächst wird die einfachste und am häufigsten verwendete Operationsverstärkerschaltung betrachtet, der Impedanzwandler, der auch als Spannungsfolger bezeichnet wird.

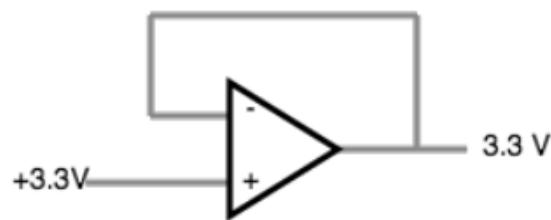


Abbildung 17: OPV als Impedanzwandler

Der Spannungsfolger weist eine Verstärkung von eins auf und somit der Eingangsspannung „folgt“. Wenn die Spannung am positiven Eingang erhöht wird, erkennt der OPV das unterschiedliche Potential und verstärkt die Ausgangsspannung, bis beide Potentiale wieder ident sind. Da die Eingangsimpedanz sehr hoch und die Ausgangsimpedanz sehr gering ist, ist das Ziel einen Verbraucher mit kleiner Impedanz an eine Quelle anschließen zu können, ohne die Quellschaltung zu sehr zu belasten. [3]

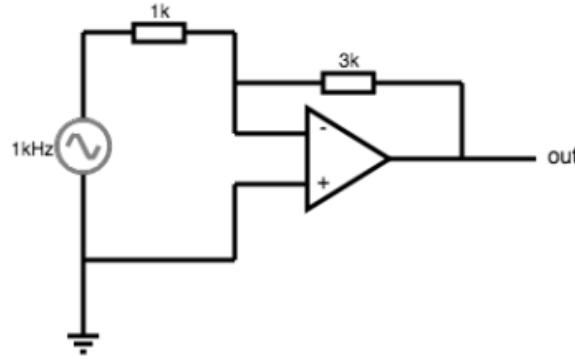


Abbildung 18: Invertierender OPV

Diese Abbildung zeigt die Konfiguration als invertierenden Verstärker. Der Strom fließt von der Wechselstromquelle durch  $R_1$  und  $R_2$ . Da die Eingangsimpedanz sehr hoch ist, wird der Strom durch den OPV vernachlässigbar klein. Das Ausgangssignal ist zum Eingangssignal invertiert, das Verhältnis aus  $R_1$  zu  $R_2$  ergibt den Verstärkungsfaktor. [3]

$$\text{Dadurch gilt: } U_a = -U_e * \frac{R_2}{R_1} \quad (3)$$

Der OPV versucht den invertierenden Eingang auf dasselbe Potential, wie des positiven Eingangs, also Masse zu bringen. Das Ausgangssignal muss also so verstärkt werden, um den Spannungsabfall an  $R_2$  zu kompensieren. Die Spannung würde in diesen Fall verdreifacht werden, was einen Verstärkungsfaktor  $v = 3$  ergeben würde.

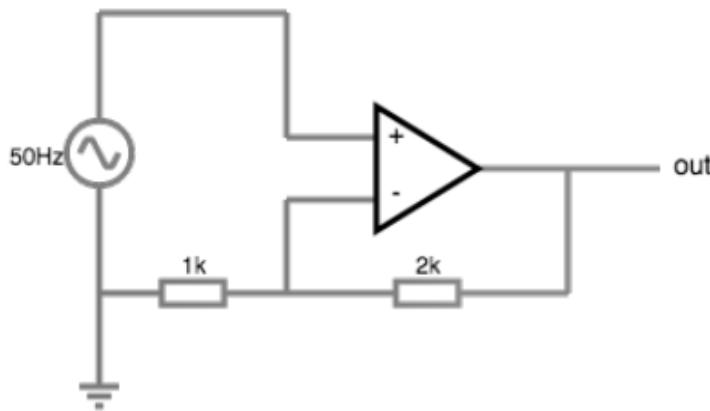


Abbildung 19: Nicht-invertierender OPV

Diese Abbildung zeigt jetzt die gegenteilige Schaltung. Der nicht-invertierende Verstärker arbeitet ebenso mit Gegenkopplung, die Widerstände  $R_1$  und  $R_2$  bilden einen Spannungsteiler, der als unbelastet angenommen wird, da der OPV eine hohe Eingangsimpedanz besitzt. [3]

$$\text{Für den Verstärkungsfaktor gilt: } U_a = U_e \left( 1 + \frac{R_2}{R_1} \right) \quad (4)$$

Auch hier wird eine Verstärkung von drei erreicht. Es kommt bei Schaltungen dieser Art weniger auf die Widerstandswerte, sondern mehr auf deren Verhältnis an. Jedoch fließt bei zu niedrigem Wert ein hoher Strom, den der Operationsverstärker im Feedbackloop leisten muss, dies kann zu Verzerrungen oder Überhitzungen des OPVs führen. Andernfalls sind zu hohe Werte ebenfalls kritisch, da sie zu erhöhtem Rauschen oder Oszillation führen können. [3]

#### 4.1.6 Unterschied zwischen analoge und digitale Filter

Analoge Filter sind leicht zu implementieren, da man das zu filternde kontinuierliche Signal am Eingang einspeisen kann. Digitale Filter hingegen, arbeiten zeitdiskret. Anstatt eine analoge Schwingung, nehmen sie einen Datenstrom als Eingang, welcher aus **Samples** (Stichproben) des Signals besteht. Um das Signal in die Samples zu zerlegen, liest ein Analog-Digital-Umsetzer in jeder Abtastperiode den Momentanwert des Signals ein. Die

Abtastfrequenz wird dabei so hoch gewählt, dass das Signal aus den Samples möglichst genau rekonstruiert werden kann. (siehe 4.1.6.1)

Digitale Filter bieten einige Vorteile. Darunter zählt, dass ein einziger Filter mehrere Eingangssignale filtern kann, ohne etwas an der Hardware geändert zu haben. Weiters variiert die Leistung nicht mit den Umgebungsbedingungen, wonach immer eine konstante Leistung herrscht.

Die Hauptanwendungen für digitale Filter seien zum Beispiel die Trennung von kombinierten Signalen und die Wiederherstellung von verzerrten Signalen. Digitale Filter erzielen hier meist bessere Ergebnisse, aber diese Probleme sind auch mittels Analogfilter problemlos zu lösen.

Weiters unterscheiden sie sich bei der Bandbreite. Während bei Digitalfiltern die Bandbreite durch den Start der Aufnahme eingestellt wird, gilt diese bei Analogfiltern als unbegrenzt, wodurch die Auflösung nicht als statisch, sondern variabel gilt und jederzeit erhöht oder gesenkt werden kann, ohne Qualität einbüßen zu müssen.

Nachteile von digitalen Filtern sind, dass sie wesentlich teurer als Analogfilter sind und dazu eine höhere Latenz aufweisen. Dazu kommen noch eine geringere Bandbreite und ein störendes Quantisierungsrauschen.

Analoge Filter werden in einer Vielzahl von Anwendungen eingesetzt, z.B. in der Tontechnik, wie in Verstärker und Equalizern, um unerwünschte Rausch- und Störgeräusche herauszufiltern und die Klangqualität zu verbessern. Auch in der Telekommunikation sind sie von wichtiger Bedeutung. Hier werden Filter verwendet, um das Signal-Rausch-Verhältnis zu verbessern und unerwünschte Signale zu unterdrücken. Zum Beispiel in Mobilfunkgeräten, um Interferenzen von benachbarten Frequenzen zu reduzieren. Sie können auch Rauschen entfernen, um klare Messergebnisse zu erhalten, ein Anwendungsbereich hierfür ist ein Elektrokardiogramm (EKG). In der Regel werden analoge Filter dort eingesetzt, wo die Signale in kontinuierlicher Form vorliegen und es wichtig ist, spezifische Frequenzen herauszufiltern oder zu unterdrücken.

#### 4.1.6.1 *Shannon-Nyquist Theorem*

Das Shannon-Nyquist Theorem ist ein Abtasttheorem, welches besagt, dass die Abtastfrequenz eines analogen Signals mindestens doppelt so hoch wie die höchste Frequenzkomponente des Signals sein muss. Dies ist aufgrund der sogenannten Nyquist-Frequenz möglich, die die höchste Frequenz angibt, die durch die Abtastung erfasst werden kann.

$$f_{\text{Abtast}} > 2 * f_{\text{max}} \quad (5)$$

Dieses Abtasttheorem ist vor allem in der digitalen Signalverarbeitung und digitalen Kommunikation von Bedeutung, da es als Grundlage für die Abtastung, Kodierung und Quantisierung gilt.

## 4.2 Aktiver Filter

### 4.2.1 Sallen-Key-Filter

Der Sallen-Key Filter ist einer der meistgenutzten Filter in der Signalverarbeitung, um Frequenzen in einem Signal zu verstärken oder abzuschwächen. Der Filter besteht lediglich aus einem Operationsverstärker und einigen passiven Bauelementen, weswegen er als aktiver Filter bezeichnet wird. Es ist wichtig zu beachten, dass der Sallen-Key eine Filtertopologie und keine Filtercharakteristik, wie Butterworth, Bessel usw. Jedoch können verschiedene Charakteristiken in verschiedene Topologien implementiert werden, je nach Änderung der Komponentenwerte verändert sich die Filtercharakteristik.

Sallen-Key werden meist als Tief- oder Hochpass verwendet, wobei bei Erweiterung auch ein Bandpass ermöglicht wird. Weiters weist der Sallen-Key eine gute Linearität und eine geringe Verzerrung auf, was es zu einer effektiven Lösung für die Signalverarbeitung macht. Dazu ist diese Topologie sehr leicht zu realisieren und auch in der Lage eine hohe Güte zu erreichen, was dazu führt, dass unerwünschte Frequenzen sich sehr stark unterdrücken lassen oder spezifische Frequenzen sehr genau verstärken lassen.

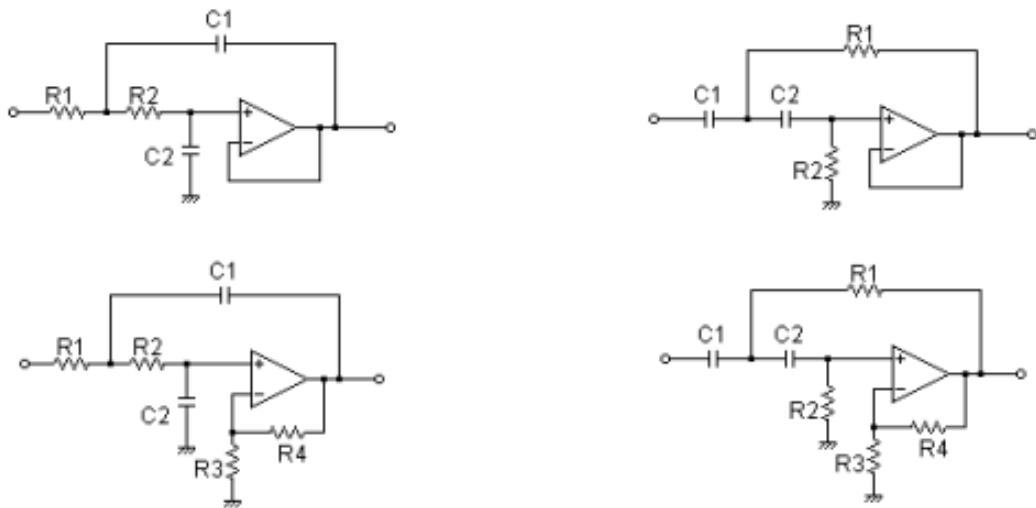


Abbildung 20: Sallen-Key Tiefpass (links) & Hochpass (rechts)

Jedoch gibt es auch Nachteile des Sallen-Key Filters. Dazu zählen eine begrenzte Bandbreite und hohe Empfindlichkeit bei Veränderung der passiven Komponenten, die zu einer Verschiebung der Filtercharakteristik führen können. Daher sollte die Überwachung der Filtercharakteristik an oberster Stelle stehen, um eine unerwünschte Veränderung zu bemerken. [2]

#### 4.2.1.1 Unterschiede in den Ordnungen

Filter gibt es in diversen Ordnungen, mit jeweils anderen Eigenschaften. So ist es auch beim Sallen-Key der Fall. Eine Änderung, die sofort auffällt, ist der Aufbau der Schaltung. Während beim Sallen-Key erster Ordnung nur jeweils ein Widerstand und Kondensator verwendet werden, benötigt ein Filter zweiter Ordnung vier passive Bauteile.

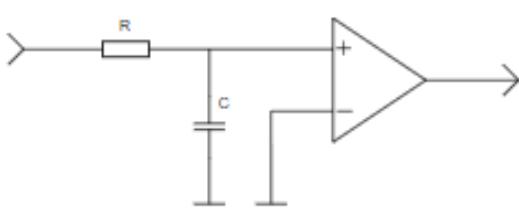


Abbildung 21: Filter erster Ordnung

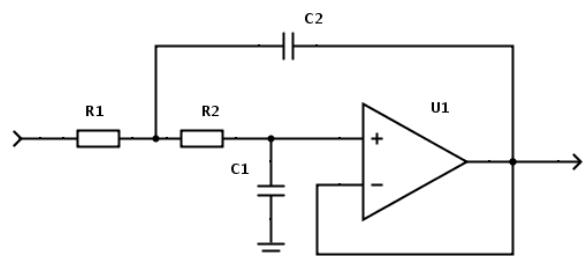


Abbildung 22: Filter zweiter Ordnung

Weiters unterscheiden sich die Ordnungen auch in der Flankensteilheit. In erster Ordnung fällt die Kurve um 20dB/Dekade (6dB/Oktave), bei einem Sallen-Key zweiter Ordnung beträgt die Flankensteilheit 40dB/Dekade (12db/Oktave).

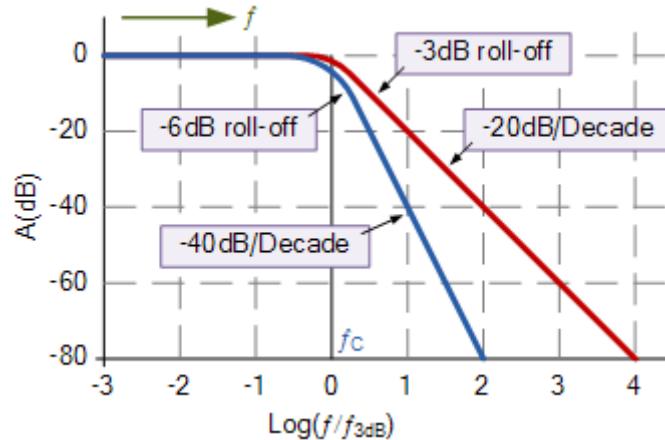


Abbildung 23: Flankensteilheit der Ordnungen [24]

Während der Sallen-Key erster Ordnung nur eine Resonanzfrequenz hat, hat der Sallen-Key zweiter Ordnung zwei Resonanzfrequenzen. Diese beeinflussen die Filtercharakteristik maßgeblich in deren Umgebung. Auch die Dämpfung verändert sich je nach Ordnung. Das bedeutet, dass ein Filter zweiter Ordnung eine höhere Dämpfungsfähigkeit aufweist und somit unerwünschte Frequenzen besser unterdrücken, kann als ein Filter erster Ordnung. [4]

#### 4.2.2 Butterworth

Die Butterworth Filtercharakteristik ist eine der beliebtesten Charakteristiken, da sie eine auf einer flachen Übertragungsfunktion basiert, was bedeutet, dass die Verstärkung im Frequenzbereich möglichst gleichmäßig ist. Weiters weist sie eine maximale lineare Phasenreaktion auf, was bedeutet, dass alle Frequenzen innerhalb des Durchlassbereichs des Filters ohne Phasenverschiebung passieren. Dies ist ein wichtiger Vorteil bei der Verarbeitung von Signalen, bei denen eine gleichmäßige Phasenreaktion erforderlich ist, wie zum Beispiel bei der Signalübertragung oder in der Audio- und Musikproduktion.

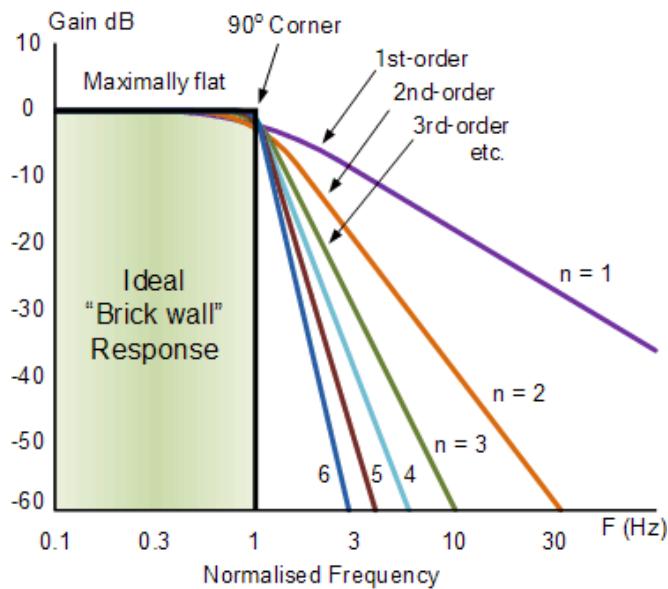


Abbildung 24: Butterworth Frequenzverhalten [33]

Die Übertragungsfunktion ist charakterisiert durch eine Glättungskurve, die zu einem sanften Roll-Off führt. Dies führt zu einem Kompromiss zwischen der Dämpfung von unerwünschten Frequenzen und der Aufrechterhaltung einer möglichst linearen Phasenreaktion innerhalb des Durchlassbereichs.

Außerdem ist die Butterworth Charakteristik für ihre hohe Stabilität bekannt. Anders als bei anderen Filtertypen, wie Chebyshev- oder Elliptischen-Filter, die ein Rippelverhalten in der Übertragungsfunktion aufweisen, weist ein Butterworth-Filter keine unerwünschten Spitzen oder Wellen auf. Dadurch wird die Filterleistung und Genauigkeit der Signalverarbeitung verbessert.

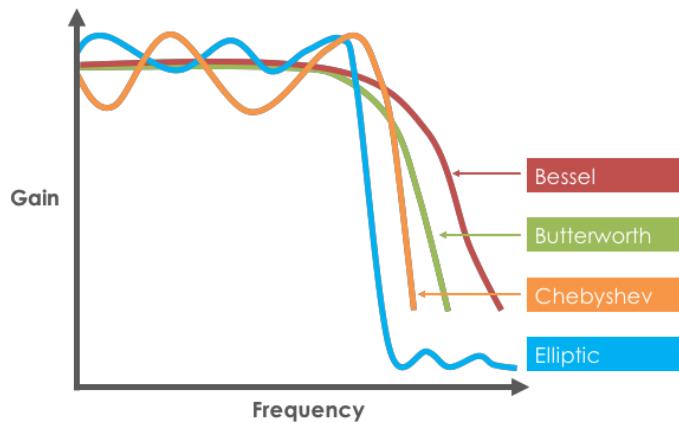


Abbildung 25: Frequenzverhalten der Filtercharakteristiken [32]

#### 4.2.2.1 Toleranzschema

Um einen Filter erfolgreich zu dimensionieren, müssen zuallererst die Anforderungen des zu entwerfenden Filters festgelegt werden, das sogenannte Toleranzschema. Dies beschreibt die zulässigen Bereiche des Amplitudengangs, dazu wird der Frequenzgang in drei Arbeitsbereiche unterteilt.

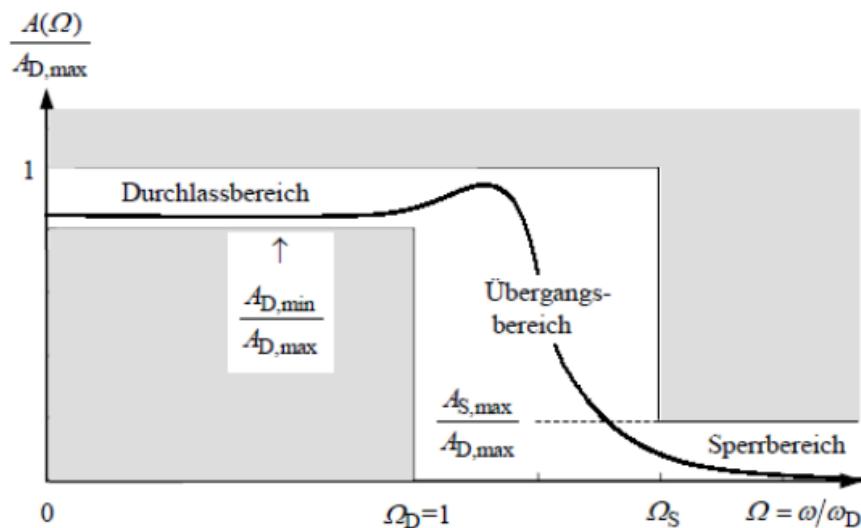


Abbildung 26: Toleranzschema [31]

- Im Durchlassbereich wird die maximal erlaubte Welligkeit des Amplitudengangs angegeben. Das Nutzsignal soll außerdem möglichst nicht beeinträchtigt werden. Der Toleranzbereich wird durch das Verhältnis  $\frac{A_{D,min}}{A_{D,max}}$  festgelegt, wobei  $A_D$  die Grundverstärkung des Filters ist.

- Der Übergangsbereich ist jener Bereich, wo die Dämpfung des Filters bis auf die vom Sperrbereich festgelegte Mindestdämpfung anwachsen soll. Der Bereich beginnt bei  $\Omega = 1$  und endet bei  $\Omega = \Omega_S$ .
- Die Mindestdämpfung wird zu Beginn des Sperrbereichs bei  $\Omega_S$  spezifiziert. Diese wird mit dem Quotienten  $\frac{A_{S,max}}{A_{D,max}}$  angegeben.

## 4.2.3 Mathematische Beschreibung

### 4.2.3.1 Allgemeine Herleitung der Übertragungsfunktion

Bei der allgemeinen Herleitung der Übertragungsfunktion werden die Bauteile durch Impedanzen ersetzt. Außerdem wird der Operations-verstärker als ideal angenommen, deshalb hat diese Schaltung bei A nur einen Knotenpunkt. Es wird eine komplexe Frequenzvariable  $p = s = j\omega$  verwendet, um die Herleitung übersichtlicher zu gestalten. Diese Herleitung kann zur Weiteren Berechnung eines Tiefpasses oder Hochpasses verwendet werden.

[5]

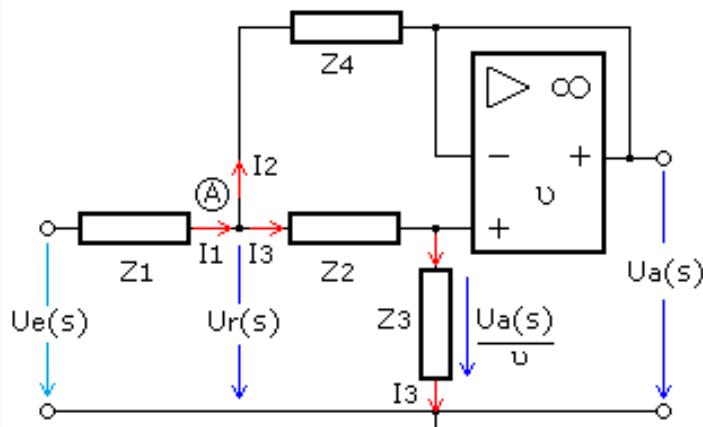


Abbildung 27: Schaltbild Filter zweiter Ordnung

$$I_1(s) = I_2(s) + I_3(s)$$

$$I_1(s) = \frac{U_e(s) - U_r(s)}{Z_1(s)}$$

$$I_2(s) = \frac{U_r(s) - U_a(s)}{Z_4(s)}$$

$$I_3(s) = \frac{U_r(s)}{Z_2(s) + Z_3(s)}$$

$$I_3(s) = \frac{U_a(s)}{\nu * Z_3(s)}$$

$$I_1 = I_2 + I_3$$

durch Einsetzen den Teilgleichungen:

$$\frac{U_e - U_r}{Z_1} = \frac{U_r - U_a}{Z_4} + \frac{U_a}{\nu * Z_3}$$

$$\text{mit: } U_r = I_3 * (Z_2 + Z_3)$$

$$\frac{U_e - I_3 * (Z_2 + Z_3)}{Z_1} = \frac{I_3 * (Z_2 + Z_3) - U_a}{Z_4} + \frac{U_a}{\nu * Z_3}$$

$$\frac{U_e}{Z_1} - \frac{U_a * (Z_2 + Z_3)}{\nu * Z_1 * Z_3} = \frac{U_a * Z_2 + U_a * Z_3 - U_a * \nu * Z_3 + U_a * Z_4}{\nu * Z_3 * Z_4}$$

$$\frac{U_e}{Z_1} - \frac{U_a * (Z_2 + Z_3)}{\nu * Z_1 * Z_3} = \frac{U_a[(Z_2 + Z_4) + Z_3(1 - \nu)]}{\nu * Z_3 * Z_4}$$

$$\frac{U_e}{Z_1} = \frac{U_a[(Z_2 + Z_4) + Z_3(1 - \nu)]}{\nu * Z_3 * Z_4} + \frac{U_a(Z_2 + Z_3)}{\nu * Z_1 * Z_3}$$

$$\frac{U_e}{Z_1} = \frac{U_a\{Z_1[(Z_2 + Z_4) + Z_3(1 - \nu)] + Z_4(Z_2 + Z_3)\}}{\nu * Z_1 * Z_3 * Z_4}$$

Allgemeine Übertragungsfunktion:

$$\frac{U_a}{U_e} = \frac{\nu * Z_3 * Z_4}{Z_1 * Z_2 + Z_1 * Z_4 + Z_2 * Z_4 + Z_3 * Z_4 + Z_1 * Z_3(1 - \nu)} \quad (6)$$

#### 4.2.3.2 Sallen-Key-Tiefpass 2. Ordnung

Für die obige Schaltung wird bei einem Tiefpass 2. Ordnung die Impedanzen Z1 und Z2 ohmsche Widerstände eingesetzt. Für Z3 und Z4 werden kapazitive Blindwiderstände verwendet. [5]

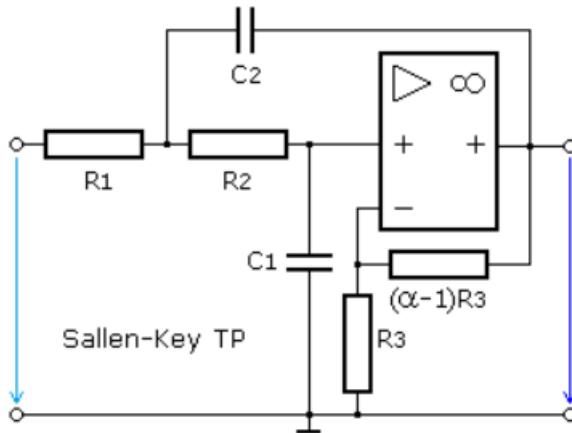


Abbildung 28: Schaltbild Sallen-Key zweiter Ordnung

Sallen-Key TP mit Verstärkung:

$$X_c = \frac{1}{j\omega c}$$

$$Z_3 = \frac{1}{s * C_1} ; Z_4 = \frac{1}{s * C_2}$$

$$\underline{G}_{TP}(s) = \frac{\underline{U}_a(s)}{\underline{U}_e(s)} = \frac{v * \frac{1}{s * C_1} * \frac{1}{s * C_2}}{R_1 R_2 + R_1 \frac{1}{s * C_2} + R_2 \frac{1}{s * C_2} + \frac{1}{s * C_1} * \frac{1}{s * C_2} + R_1 \frac{1}{s * C_1} (1 - v)}$$

$$\underline{G}_{TP} = \frac{v}{R_1 R_2 C_1 C_2 s^2 + C_1 s (R_1 + R_2) + 1 + R_1 C_2 s (1 - v)}$$

$$\underline{G}_{TP} = \frac{v}{1 + R_1 R_2 C_1 C_2 s^2 + s [C_1 (R_1 + R_2) + R_1 C_2 (1 - v)]} \quad (7)$$

allgemein normierter TP 2. Ordnung:

$$\underline{G}_{TP} = \left( \frac{1}{1 + j \frac{\omega}{\omega_g}} \right)^2 = \frac{1}{1 - \left( \frac{\omega}{\omega_g} \right)^2 + ja \frac{\omega}{\omega_g}}$$

Mit  $s = j\omega$  folgt:

$$\underline{G}_{TP} = \frac{\nu}{1 - R_1 R_2 C_1 C_2 \omega^2 + j\omega [C_1(R_1 + R_2) + R_1 C_2(1 - \nu)]} \quad (8)$$

Durch einen Vergleich sich entsprechender Komponenten im normierten allgemeinen TP-Filter zweiter Ordnung können Grenzfrequenzen und Dämpfungswerte (a) für unterschiedliche Dimensionierungen von R und C ermittelt werden. Die Simulationsergebnisse für unterschiedliche Verstärkungen sind auf den Ausgangswert 0dB gesetzt. Die Amplitudenerhöhung nimmt bei Verstärkung größer 1,5 deutlich zu.

$$\left(\frac{\omega}{\omega_g}\right)^2 = R_1 R_2 C_1 C_2 \omega^2$$

$$\omega_g^2 = \frac{1}{R_1 R_2 C_1 C_2} ; f_g = \frac{1}{2\pi\sqrt{R_1 R_2 C_1 C_2}}$$

$$a \frac{\omega}{\omega_g} = \omega [C_1(R_1 + R_2) + R_1 C_2(1 - \nu)]$$

$$a = \omega_g [C_1(R_1 + R_2) + R_1 C_2(1 - \nu)] \quad (9)$$

Nach Einsetzen von  $\omega_g$ :

$$a = \frac{C_1(R_1 + R_2) + R_1 C_2(1 - \nu)}{\sqrt{R_1 R_2 C_1 C_2}}$$

Mit  $R_1 = R_2 = R$  und  $C_1 = C_2 = C$

$$f_g = \frac{1}{2\pi R C} ; a = 3 - \nu$$

Wenn Widerstände und Kondensatoren unterschiedliche Werte haben:

$$f_g = \frac{1}{2\pi\sqrt{R_1 R_2 C_1 C_2}} \quad (10)$$

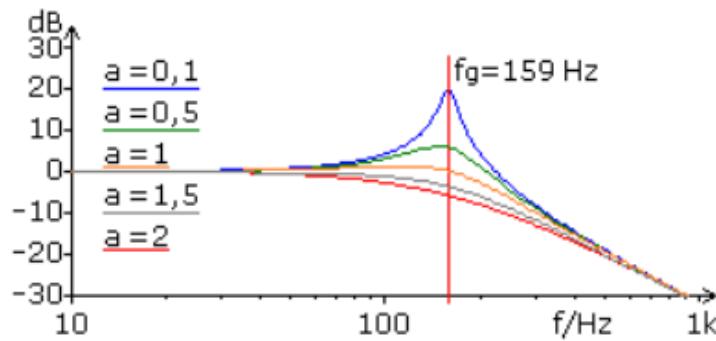


Abbildung 29: Dämpfungskurven

In Abbildung 24 ist zu sehen, wie sich die Dämpfungskurve mit diversen Faktoren verändert. Der Dämpfungsfaktor beschreibt die Fähigkeit eines Systems, unerwünschte Schwingungen abzuschwächen oder zu unterdrücken. Man sieht, dass bei einer niedrigen Dämpfung das System mehr Zeit benötigt, um abzuklingen bzw. hält die Schwingung länger an. Anders als bei einem hohen Dämpfungsfaktor, wo sich das System schneller beruhigt.  
[5]

## 4.3 Grundlagen der digitalen Signal-Verarbeitung (DSV)

### 4.3.1 Allgemein

Schlägt man auf einer Gitarre eine Saite an, so schwingt diese mit einer gewissen Frequenz. Am ESP32 soll nun mithilfe eines Algorithmus die Frequenz und daraus folgend die gespielte Note erkannt werden. Hierfür wird der Fast Fourier Transform (FFT) Algorithmus verwendet. Mithilfe der FFT kann das Frequenzspektrum des Signals ermittelt werden.

Aufgrund des Aufbaus einer E-Gitarre gibt es neben der Grundfrequenz aber noch eine Vielzahl anderer Schwingungen, welche beispielsweise durch das Kabel, Holz oder Pickup (Tonabnehmer) der Gitarre auftreten können. Diese „Zusatzschwingungen“ treten bei allen Instrumenten in einer individuellen Kombination auf und sind der Grund, warum sich die Gleiche Note auf zum Beispiel Klavier und Gitarre unterschiedlich anhört.

Spielt man einen Ton auf einem Instrument so treten neben dem Grundton noch sogenannte Obertöne oder Teiltöne auf. Diese sind höher als der tatsächlich gespielte Grundton und stellen bei dessen Erkennung eine maßgebliche Herausforderung dar.

Um ein möglichst obertonfreies Signal zu gewährleisten, ist es ratsam, den Pickup, welcher am Hals sitzt zu wählen. Die Brücke reflektiert einige Schwingungen und beeinflusst dadurch das Signal.

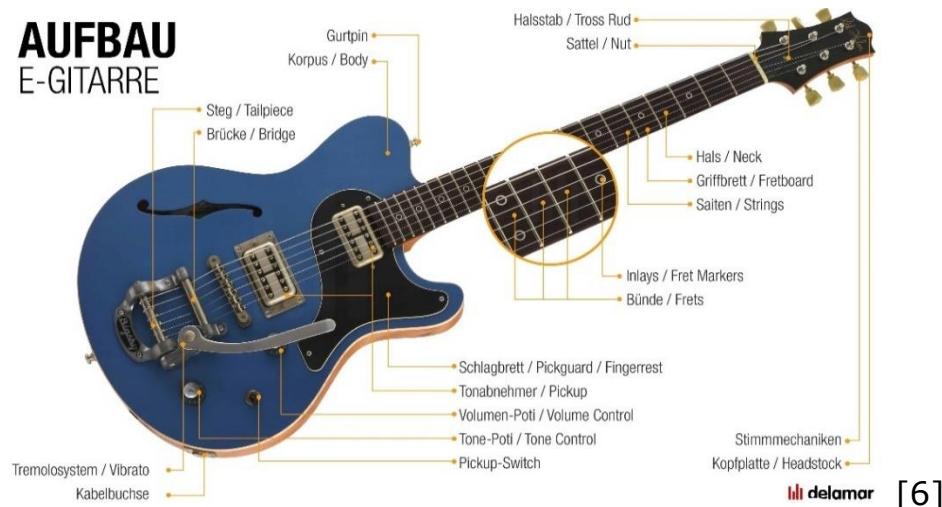


Abbildung 30: Aufbau einer E-Gitarre

### 4.3.2 Einleitung Fourier Transformation

Die Fourier Transformation ist eine Integraltransformation, welche 1822 von Jean Baptiste Fourier eingeführt wurde und genutzt wird, um das diskrete Frequenzspektrum aperiodischer Signale zu ermitteln. Ihr gegenüber steht die Fourier Reihe, mit welcher sich das Spektrum periodischer Funktionen berechnen lässt. Sie ist in vielen Bereichen von Wissenschaft und Technik unerlässlich. Vor allem bei der Datenkomprimierung spielt sie eine große Rolle. Um sie von einem Computer ausführen zu lassen gibt es die Diskrete Fourier Transformation (DFT) und die Schnelle Fourier Transformation (FFT).

Eigentlich handelt es sich beim FFT-Algorithmus nur um eine Möglichkeit die DFT mit hoher Geschwindigkeit durchzuführen. James Cooley und John W. Tukey wiederentdeckten ihn im Jahr 1965. Wiederentdeckung deshalb, weil Carl Friedrich Gauß den Algorithmus in seiner ersten Form bereits 1805 entdeckte und verwendete, diese aber nie publizierte. Erst die durch diesen Algorithmus erreichte hohe Durchführungsgeschwindigkeit ermöglicht eine so breit gefächerte Anwendung.

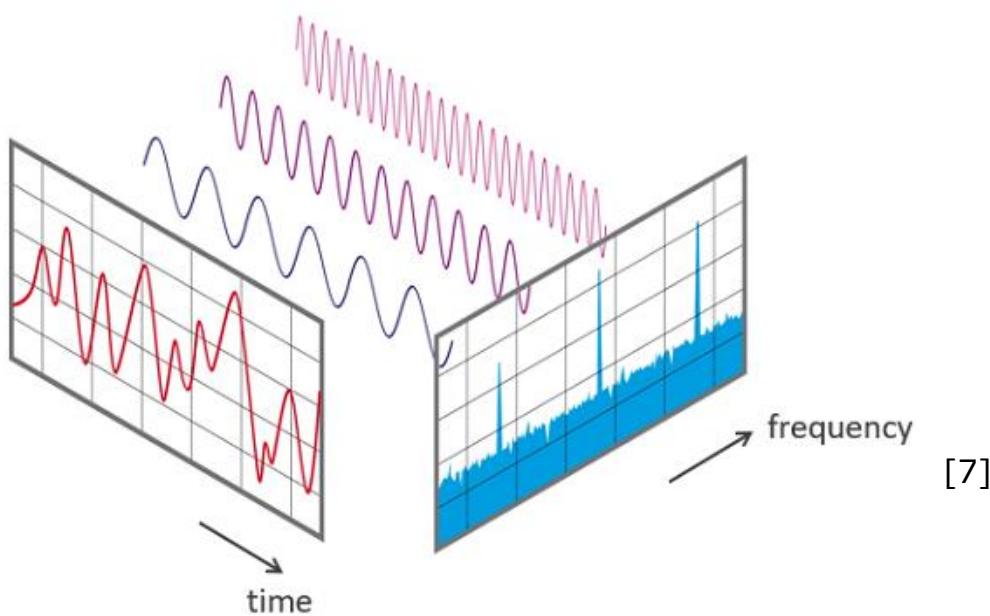


Abbildung 31: Veranschaulichung Fourier Transformation

### 4.3.3 Ursprung der Fourier Transformation

#### 4.3.3.1 Fourier Reihe

1807 fand Jean Baptiste Fourier heraus, dass sich eine periodische Funktion als eine Linearkombination von Sinus- und Cosinus-Schwingungen, eine sogenannte Fourier-Reihe, ausdrücken lässt:

$$x_p(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k * \cos(2\pi k f_0 t) + b_k * \sin(2\pi k f_0 t)) \quad (11)$$

Hierbei sind  $a_k$  und  $b_k$  sogenannte Fourier-Koeffizienten, welche den Amplituden der entsprechenden (also kten) Schwingungsanteile gleichkommen. Für den Fall „ $k = 0$ “ existiert das, den arithmetischen Mittelwert darstellende, zeitunabhängige Glied  $\frac{a_0}{2}$ . Die Grundfrequenz der Fourier-Reihe ist über  $f_0$  dargestellt.

Wird nun die Cosinus-Funktion durch  $\cos(j\omega t) = \frac{1}{2} * (e^{-j\omega t} + e^{j\omega t})$  und Sinus-Funktion durch  $\sin(\omega t) = \frac{1}{2j} * (e^{j\omega t} - e^{-j\omega t})$  ersetzt so erhält man die komplexe Fourier-Reihe.

$$x_p(t) = \sum_{k=-\infty}^{\infty} c_k * e^{j\omega t} \quad (12)$$

Der Koeffizient  $c_k$  ist der komplexe Fourier Koeffizient, aus welchem sich Amplituden und Phasen der Harmonischen berechnen lassen. Multipliziert man hier beide Seiten mit  $e^{-j\omega t}$  und integriert diese anschließend von  $-\frac{T_0}{2}$  bis  $\frac{T_0}{2}$  erhält man die als Analysegleichung bezeichnete Bestimmungsgleichung:

$$c_k = \frac{1}{T_0} * \int_{-\frac{T_0}{2}}^{\frac{T_0}{2}} x_p(t) * e^{-j\omega t} dt \quad (13)$$

Das Bestimmen von  $c_k$  über eine Integration gestaltet sich in der Praxis sehr mühsam. Mithilfe der FFT lassen sich diese Koeffizienten viel einfacher und effizienter berechnen. [8, p. 25 ff.]

### 4.3.3.2 Von der Fourier Reihe zur Fourier Transformation

Da die Fourier Reihe in ihrer Anwendung auf periodische Signale beschränkt ist, wurde die Fourier Transformation eingeführt. Wie bereits erwähnt wird diese genutzt, um das diskrete Frequenzspektrum aperiodischer Signale zu ermitteln. Damit dies möglich wird, wird das Eingangssignal als periodisches Signal angenommen, wobei seine Periode unendlich groß ist. Sie lässt sich herleiten, indem man in der Gleichung für die komplexe Fourier Reihe  $c_k$  durch das Integral aus der Analysegleichung und  $x_p(t)$  durch  $x(t)$  ersetzt.

$$x(t) = \sum_{k=-\infty}^{\infty} \left( \int_{-\frac{T_0}{2}}^{\frac{T_0}{2}} x(t) * e^{-j\omega t} dt \right) * e^{j\omega t} * \frac{1}{T_0} \quad (14)$$

Hierbei entspricht  $kf_0$  einem Punkt auf der Frequenzachse und kann daher als  $f$  geschrieben werden. Da die Periode eines aperiodischen Signals  $T_0 = \infty$  ist die Grundfrequenz  $f_0 = \frac{1}{T_0}$  demzufolge unendlich klein und wird als deshalb mit  $df$  ersetzt.

$$x(t) = \int_{-\infty}^{\infty} \left( \int_{-\infty}^{\infty} x(t) * e^{-j\omega t} dt \right) * e^{j\omega t} dt \quad (15)$$

Das eingeklammerte Integral heißt Fourier Transformierte  $X(f)$  und ist eine Funktion von  $f$ . Somit erhält man als Formel für die Fourier Transformation:

$$X(f) = \int_{-\infty}^{\infty} x(t) * e^{-j\omega t} dt \quad (16)$$

Dies kann nun auf  $x(t)$  umgeformt werden, womit sich die Inverse Fourier Transformation ergibt:

$$x(t) = \int_{-\infty}^{\infty} X(f) * e^{j\omega t} df \quad (17)$$

Zusammenfassend kann gesagt werden, dass  $X(f)$  die Fourier-Transformierte von  $x(t)$  und  $x(t)$  die inverse Fourier-Transformierte von  $X(f)$  ist. Dabei gibt  $X(f)$  die Stärke und Phase an mit welcher eine komplexe Sinusschwingung mit der Frequenz  $f$  im Signal  $x(t)$  vorhanden ist. [8, p. 28 ff.]

#### 4.3.4 Herleitung der Diskreten Fourier Transformation

Tatsächlich handelt es sich bei der Diskreten Fourier Transformation (DFT) um eine Annäherung der Fourier Transformation, welche es ermöglicht sie effizient von einem digitalen Rechner berechnen zu lassen.

Hierbei wird die Formel für die Fourier Transformierte (16) als Ausgangspunkt genutzt. Das zeitkontinuierliche Signal wird durch seinen Abtastwert  $x(nT)$  und das Differential durch das Abtastintervall  $T$  ersetzt. Zur Annäherung des Integrals wird die Summe verwendet:

$$X_s(f) = \sum_{n=-\infty}^{+\infty} x(nT) * e^{-j\omega nT} * T \quad (18)$$

Da eine unendliche Anzahl an Abtastwerten unmöglich zu berechnen ist, werden eine endliche Anzahl  $N$  dieser herausgeschnitten/„gefenstert“ (engl: windowing). Außerdem kann der Faktor  $T$  aus „Bequemlichkeit“ weggelassen werden.

$$X_{sw}(f) = \sum_{n=0}^{N-1} x(nT) * e^{-j\omega n \frac{f}{f_s}} \quad (19)$$

Dies Funktion ist  $f_s$ -periodisch und hat nur an  $N$ -Stellen linear unabhängige Funktionswerte. Ausgewertet wird sie an  $N$  gleichentfernten Frequenzstellen  $f = 0, \frac{f_s}{N}, 2 * \frac{f_s}{N}, \dots, (N - 1) * \frac{f_s}{N}$ . Werden der Einfachheit halber wieder einige Faktoren  $(\frac{f_s}{N}, T)$  und die Kennzeichnung  $sw$  weggelassen ergibt sich die Definition (Analysegleichung) der DFT:

$$X[k] = \sum_{n=0}^{N-1} x[n] * e^{-jkn \frac{2\pi}{N}} \quad (20)$$

Die inverse DFT (IDFT) (Synthesegleichung) ist definiert als:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] * e^{jkn \frac{2\pi}{N}} \quad (21)$$

[8, p. 163 ff.]

#### 4.3.4.1 Matrix-Interpretation der DFT

Führt man den Drehfaktor (engl: twiddle factor)  $W_N = e^{-j2\pi/N}$  ein, so kann die DFT und IDFT auch folgenderweise geschrieben werden:

$$X[k] = \sum_{n=0}^{N-1} x[n] * W_N^{kn} \quad (22)$$

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] * W_N^{-kn} \quad (23)$$

Diese beiden Sequenzen lassen sich in Vektorform darstellen:

$$\mathbf{x}_N = \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix}, \quad \mathbf{X}_N = \begin{bmatrix} X[0] \\ X[1] \\ \vdots \\ X[N-1] \end{bmatrix} \quad (24)$$

Definiert man nun noch die DFT-Matrix:

$$\mathbf{W}_N = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & W_N^1 & \cdots & W_N^{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{N-1} & \cdots & W_N^{(N-1)(N-1)} \end{bmatrix} \quad (25)$$

...so lässt sich die N-Punkte-DFT als Matrizenmultiplikation schreiben:

$$\mathbf{X}_N = \mathbf{W}_N \mathbf{x}_N \quad (26)$$

Multipliziert man die DFT mit der inversen DFT-Matrix (28) so wird die IDFT erhalten:

$$\mathbf{x}_N = \mathbf{W}_N^{-1} \mathbf{X}_N \quad (27)$$

Definiert ist die inverse DFT-Matrix als:

$$\mathbf{W}_N^{-1} = \frac{1}{N} \mathbf{W}_N^* \quad (28)$$

[8, p. 165 f.]

Somit lässt sich folgendes aussagen:

- „Die diskrete Fourier Transformation ist eine Multiplikation, nämlich des Signalvektors mit der DFT-Matrix. Das Resultat ist der DFT-Koeffizienten-vektor.“ [8, p. 166]

#### 4.3.4.2 Eigenschaften der DFT

- „Die DFT einer Linearkombination von Signalen ist gleich der Linearkombination ihrer DFTs.“ [8, p. 169]
- „Die DFT und die IDFT sind  $N$ -periodisch.“ [8, p. 169]
- „Die Energie des Signals im Zeitbereich ist gleich der Energie des Signals im Frequenzbereich geteilt durch  $N$ .“ (Parseval-Theorem) [8, p. 170]
- „Die DFT eines reellen Signals ist bezüglich dem Punkt  $k = N/2$  symmetrisch.“ [8, p. 170]

#### 4.3.4.3 DFT als Annäherung der Fourier Transformation

Vorausgesetzt ein zeitkontinuierliches Signal  $x(t)$  ist beschränkt auf ein Intervall der Dauer  $T_0$ , dann lässt sich für dieses an den diskreten Frequenzpunkten  $f_k = \frac{f_s}{N}$  durch die DFT folgendermaßen annähern:

$$X(f)|_{f=k\frac{f_s}{N}} \approx TX[k], \quad k = \begin{cases} -\frac{N}{2}, \dots, -1, 0, 1, \dots, \frac{N}{2}-1 & : N \text{ gerade} \\ -\frac{N-1}{2}, \dots, -1, 0, 1, \dots, \frac{N-1}{2} & : N \text{ ungerade} \end{cases} \quad (29)$$

Die Messdauer  $NT$  muss hierbei größer oder gleich der Signaldauer  $T_0$  sein. Sollte das Messfenster länger sein als die Signaldauer, so wird der Signalvektor mit Nullen ergänzt, bis er die Länge  $N$  hat. Dadurch wird ebenfalls eine bessere graphische Auflösung erzielt. Durch Verkleinerung des Abtastintervalls  $T = \frac{1}{f_s}$  wird die physikalische Auflösung des Spektrums verbessert. Proportional zur Verkleinerung der Abtastintervalls wird auch der Approximationsfehler, welcher durch die Bandüberlappung entsteht.

[8, p. 171]

#### 4.3.5 Fast Fourier Transform

Gegensätzlich zur populären Meinung ist die FFT selbst keine Transformedierte, sondern eine Methode zur effizienten Berechnung der DFT. Betrachtet man die Definition der DFT:

$$X[k] = \sum_{n=0}^{N-1} x[n] * W_N^{kn} \quad (30)$$

...so stellt man fest, dass der Rechenaufwand dieser, bei  $N * N$  Multiplikationen und  $N * (N - 1)$  Additionen, etwa  $N^2$  beträgt. Die Verwendung der FFT vermindert den Rechenaufwand, setzt aber voraus, dass  $N$  eine Zweierpotenz ist.

Zuerst wird die Folge  $x[n]$  in zwei Teile geteilt, die eine mit den geraden, die andere mit den ungeraden Abtastwerten. Die DFT-Summe kann nun ebenfalls in zwei Teilsummen aufgespalten werden:

$$X[k] = \sum_{n=0}^{\frac{N}{2}-1} x_1[n] * W_N^{2nk} + W_N^k * \sum_{n=0}^{\frac{N}{2}-1} x_2[n] * W_N^{2nk} \quad (31)$$

Wobei  $W_N^{2nk}$  auch geschrieben werden kann als:

$$W_N^{2nk} = e^{-j\frac{2\pi}{N}2nk} = e^{-j\frac{2\pi}{N/2}nk} = W_{N/2}^{nk} \quad (32)$$

Somit ergibt sich:

$$X[k] = \sum_{n=0}^{\frac{N}{2}-1} x_1[n] * W_{\frac{N}{2}}^{nk} + W_N^k * \sum_{n=0}^{\frac{N}{2}-1} x_2[n] * W_{\frac{N}{2}}^{nk} = X_1[k] + W_N^k * X_2[k] \quad (33)$$

Nun fällt auf, dass  $X_1[k]$  und  $X_2[k]$  die DFTs von  $x_1[n]$  und  $x_2[n]$  repräsentieren.

Das heißt, dass eine  $N$ -Punkte-DFT in zwei  $N/2$ -Punkte-DFT zerlegt wurde, welche jeweils nur mehr  $(\frac{N}{2})^2$  Rechenoperationen erfordern. Die Multiplikation mit  $W_N^k$  und anschließende Addition benötigen nochmals  $N$  Operationen. Der endgültige Rechenaufwand beträgt somit  $2\left(\frac{N}{2}\right)^2 + N$ .

Das folgende Signalflussdiagramm beschreibt die Aufspaltung einer 8-Punkte-DFT graphisch:

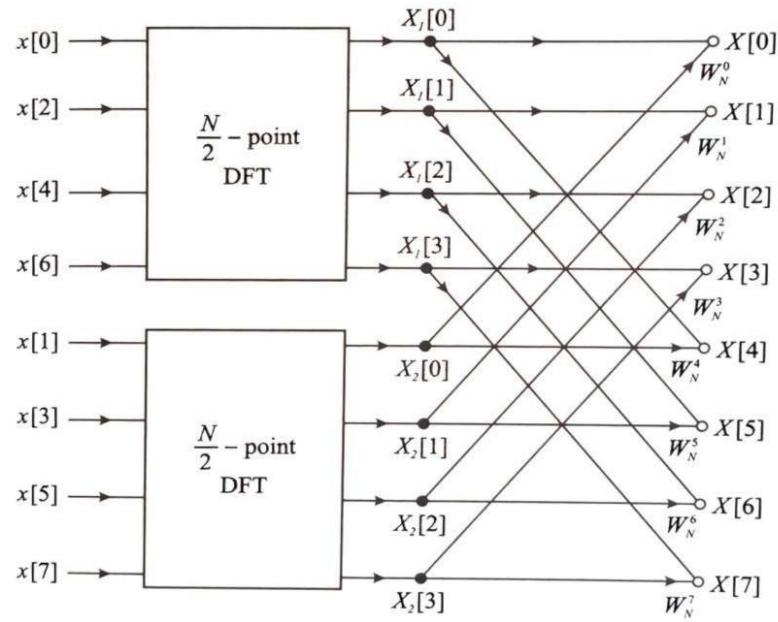


Abbildung 32: Aufspaltung 8-Punkte-DFT

Die Vorgehensweise der FFT ist somit eklatant: Die beiden  $N/2$ -Punkte-DFTs werden nun fortlaufend immer weiter zerlegt (in diesem Fall in je zwei  $N/4$ -Punkte-DFTs usw.) bis nur noch 2-Punkte-DFTs übrig sind. Das setzt jedoch voraus, dass es sich bei  $N$  um eine Zweierpotenz handelt. Die Anzahl der benötigten Zerlegungsschritte beträgt somit  $q = \log_2(N)$ . Im folgenden Bild wird die weitere Zerlegung der 8-Punkte-DFT dargestellt:

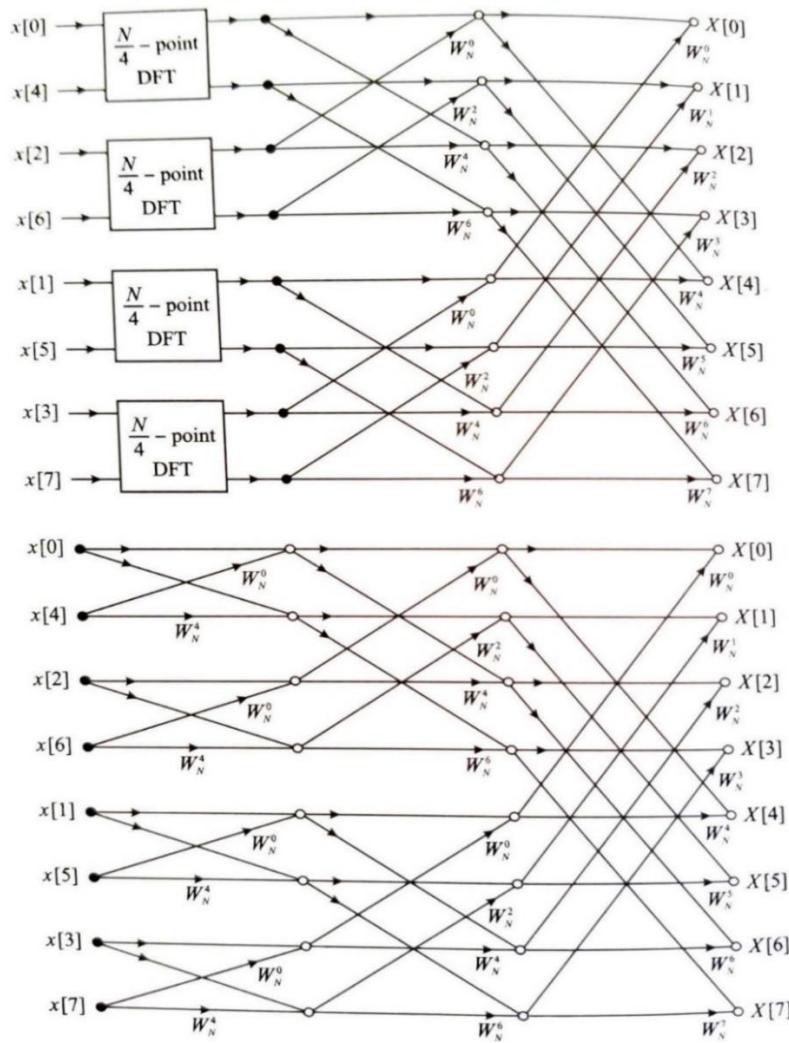


Abbildung 33: weitere Zerlegung 8-Punkte-DFT

Wird der Untere Teil des Bildes betrachtet, so lässt sich feststellen, dass dieser aus  $N/2$  sogenannten Butterfly oder Schmetterlings-Graphen besteht, welche folgendermaßen aussehen:

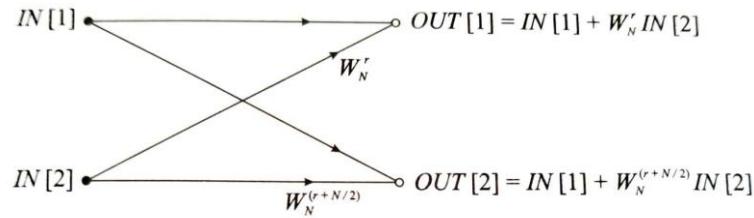


Abbildung 34: Butterfly-Graph

Zur Berechnung eines solchen Graphen werden je zwei komplexe Multiplikationen und Additionen benötigt. Dies resultiert in einem Rechenaufwand von  $N$  Operationen pro Zerlegungsebene.

Mit  $q$  Zerlegungsebenen ergibt sich ein Rechenaufwand von

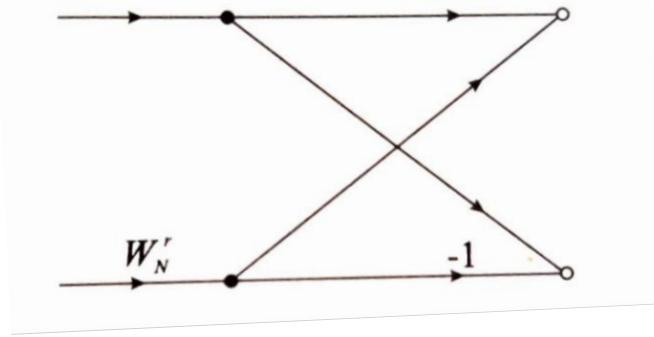
$$N_q = N * \log_2(N). \quad (34)$$

Bedenkt man hierbei, dass zur Berechnung der regulären DFT  $N^2$  Operationen benötigt werden, so stellt dies eine massive Aufwandssenkung dar.

Berücksichtigt man noch die Beziehung

$$W_N^{r+N/2} = W_N^{N/2} * W_N^r = -W_N^r, \quad (35)$$

lässt sich der Graph noch weiter vereinfachen, so, dass nur noch **eine** Multiplikation pro Butterfly erforderlich ist:



*Abbildung 35: vereinfachter Butterfly*

Alle oben beschriebenen Maßnahmen machen die FFT dermaßen effizient, dass bei ihrer Verwendung, bei beispielsweise einer 1024-Punkte-DFT, über 99% der Rechenoperationen eingespart werden können. [8, p. 174 ff.]

#### 4.3.6 Firmware

Das DSP-Programm, welches am ESP32 läuft, führt eine FFT mit dem Eingangssignal durch. Um ein kontinuierliches Einlesen der Gitarrensignale zu ermöglichen wird der DMA-Controller genutzt.

Wegen ihrer einfachen Anwendung, guten Dokumentation und ihrer hohen Performance wurde die FFT-Bibliothek von Robin Scheibler verwendet. Bei dieser handelt es sich um eine Implementierung von radix-2, split-radix und den Basen vier und acht. Diese Basen limitieren die maximal mögliche Größe der FFT.

Bei der Erstellung des Programms wurde zuerst die richtige Anwendung FFT-Library und ihrer Funktionen getestet. Danach wurde die Weiterverarbeitung der Ergebnisse realisiert.

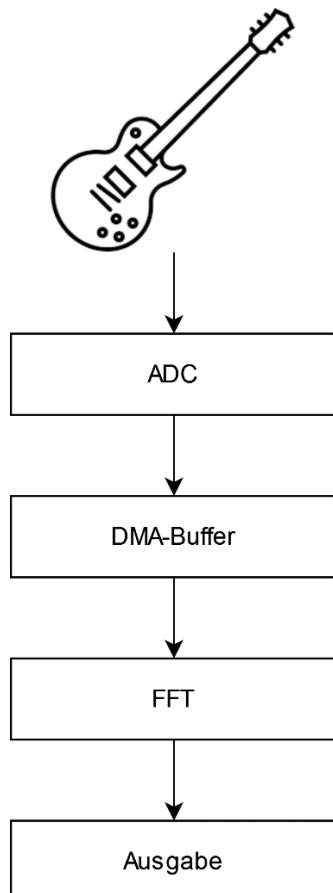


Abbildung 36: Dataflow Firmware

#### 4.3.6.1 Erklärung der FFT anhand eines Testprogramms

```
#define NFFT 8192
#define F_ABТ 44100
```

*Codesegment 1: Macros*

Zuerst werden für NFFT die Größe des FFT-Buffers und für F\_ABТ die in der Audiotechnik übliche Abtastrate von 44,1 kHz festgelegt.

```
#include <stdio.h>
#include <stdlib.h>
#include "esp_log.h"
#include "driver/adc.h"
#include "esp_adc_cal.h"
#include <math.h>
#include "fft.h"
#include "processed-data.h" //enthält test_buffer
```

*Codesegment 2: Includes*

Codesegment 2 importiert sämtliche benötigte Bibliotheken. Der Header „processed-Data.h“ enthält die vorbereiteten Testdaten. (siehe Datenauszug 2)

```
float fft_buffer[NFFT];
float magnitude[NFFT / 2];
float frequency[NFFT / 2];
float keyNR[NFFT / 2];
float ratio = (float)F_ABТ / (float)NFFT;
```

*Codesegment 3: Array Initialisierung*

Alle benötigten Buffer und Variablen werden in Codesegment 3 initialisiert. Das Array „fft\_buffer“ enthält nach dem Ausführen der FFT die Ergebnisse. Die drei darauffolgenden Arrays beinhalten jeweils die Magnitude, die Frequenz und die am Piano korrespondierende Tastennummer. Die Variable „ratio“ beinhaltet die Auflösung des Amplitudenspektrums in Hertz.

```
float getMaxMag();
float max;
```

*Codesegment 4: Deklaration von max*

Um unwesentliche Frequenzen relativ zur Lautstärke auszuschließen, wird die Funktion „getMaxMag()“ eingeführt welche die höchste auftretende Magnitude ermittelt.

```

float getMaxMag()
{
    float max = 0;
    for (int i = 0; i < NFFT / 2; i++)
    {
        if (magnitude[i] > max)
        {
            max = magnitude[i];
        }
    }
    return max;
}

```

*Codesegment 5: Ausprogrammierung „getMaxMag()“*

```

void app_main(void)
{
    fft_config_t *real_fft_plan = fft_init(
        NFFT,
        FFT_REAL,
        FFT_FORWARD,
        test_buffer,
        fft_buffer
    );
    ...
}

```

*Codesegment 6: Start "main-loop"*

Codesegment 6 startet den „main-loop“. Die FFT wird mit den Argumenten: Größe des Eingangspuffers, Art der FFT, Eingangspuffer und Buffer für die Ergebnisse konfiguriert.

```
fft_execute(real_fft_plan);
```

*Codesegment 7: FFT-Ausführung*

Die Ergebnisse werden anschließend verarbeitet.

```

for (int k = 1; k < NFFT / 2; k++)
{
    magnitude[k] = 2 * sqrt(
        pow(fft_buffer[2 * k], 2) +
        pow(fft_buffer[2 * k + 1], 2)
    ) / NFFT;
    frequency[k] = k * ratio;
    keyNR[k] = log2(frequency[k] / 440) * 12 + 49 + 20;
}
max = getMaxMag();

```

*Codesegment 8: Verarbeitung der FFT-Daten*

Der abgebildete For-Loop ist für die Berechnung der Einzelnen Signifikanten Werte wie etwa Magnitude, Frequenz und die am Piano korrespondierende Tastennummer zuständig.

Zur Berechnung der Tastennummer wird die Formel

$$keyNR = \log\left(\frac{f}{440Hz}\right) * 12 + 49 \quad (36)$$

verwendet. Die 440Hz sind der Kammerton A. Hierbei ist zu beachten, dass, um die Midi-Notennummer zu erhalten noch 20 dazu addiert werden müssen, das Midi-Format besitzt 128 mögliche Notennummern, von denen die ersten 20 nicht belegt sind. [9]

```
for (int k = 1; k < NFFT / 2; k++)
{
    // printf("%d-th freq : %f+(%f)j\n", k, fft_buffer[2*k],
    fft_buffer[2*k+1]);

    // printf("%f\t(%f)j\n", fft_buffer[2*k],
    fft_buffer[2*k+1]);

    if (magnitude[k] > max*0.5)
    {
        printf(" %d-th magnitude: %f => corresponds to %f
               Hz\n", k, magnitude[k], frequency[k]);
        printf("keyNR: %d\n", (int)round(keyNR[k]));
    }
    // printf("%f\n", magnitude[k]);
}
// printf("Middle component : %f\n", fft_buffer[1]);
// N/2 is real and stored at [1]
```

*Codesegment 9: Ausgabe*

Mit der If-Verzweigung wird überprüft, ob eine Magnitude einen Schwellenwert übersteigt, was bedeuten würde, dass diese Frequenz im Eingangssignal vorkommt. Die vereinzelten, auskommentierten „printf“ Funktionen dienen, um die einzelnen Zwischenwerte auszugeben um diese mit Matlab zu Überprüfen. Im endgültigen Programm sind diese nicht mehr vorhanden da sie rein zum Debuggen dienen.

```
fft_destroy(real_fft_plan);
```

*Codesegment 10: FFT-Zerstörung*

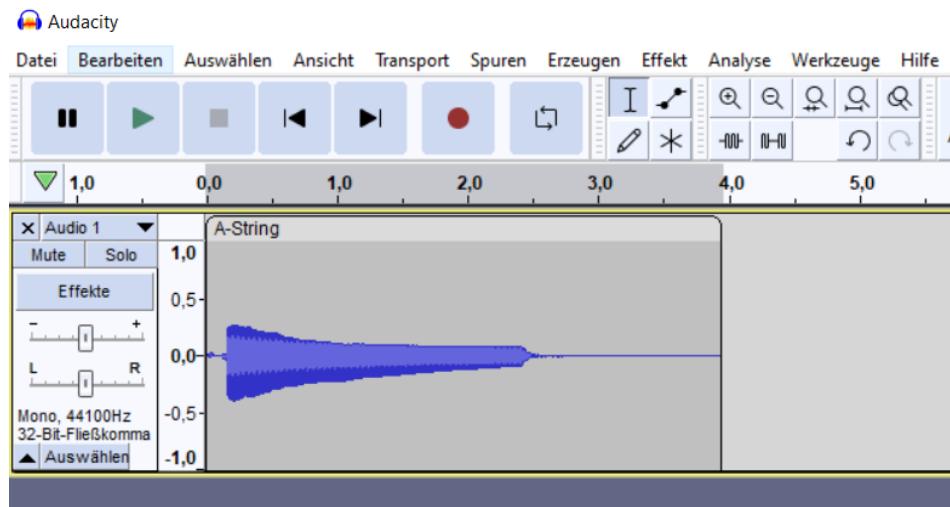
In Codesegment 10 wird die FFT-Konfiguration zerstört. Dies ist ein wichtiger Schritt da für jeden Durchlauf des Algorithmus eine neue Konfiguration erstellt werden muss.

#### 4.3.6.2 Testdatenfertigung

Zur Aufbereitung der Testdaten wurde das Audiobearbeitungsprogramm Audacity und ein eigens entwickeltes Python-Tool genutzt.

Für den Testdurchlauf wurde das Anschlagen einer A-Saite gewählt. Die Aufnahme erfolgt mit einer Samplerate von 44,1 kHz.

Das aufgenommene Signal ist in Abbildung 37 in Audacity zu sehen. Die Abnahme der Signalstärke beziehungsweise das Ausklingen der Seite kann an der anfänglichen Amplitude von circa 0,25 beobachtet werden.



*Abbildung 37: Audacity Interface*

Um nun die einzelnen Punkte aus dem Signal zu extrahieren, muss ein Bereich im Signal ausgewählt werden.

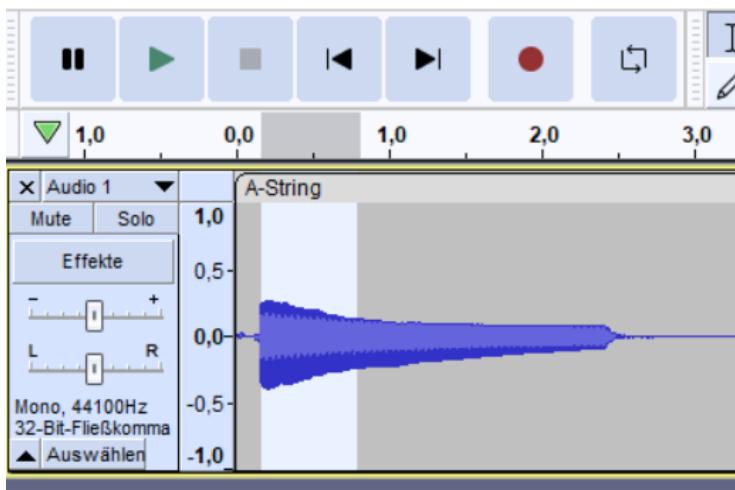


Abbildung 38: Auswahl eines Signalabschnitts

Anschließend muss in der Menüleiste unter Werkzeuge die Option Sample-Datenexport gewählt werden.

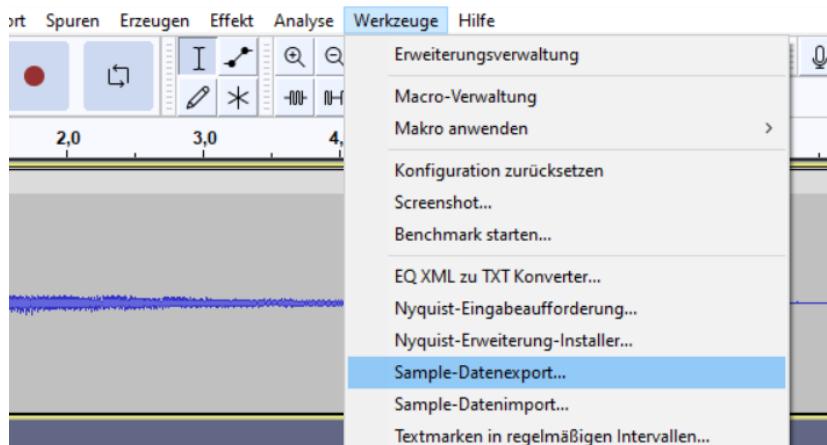


Abbildung 39: Audacity Tools

Daraufhin öffnet sich das folgende Fenster.

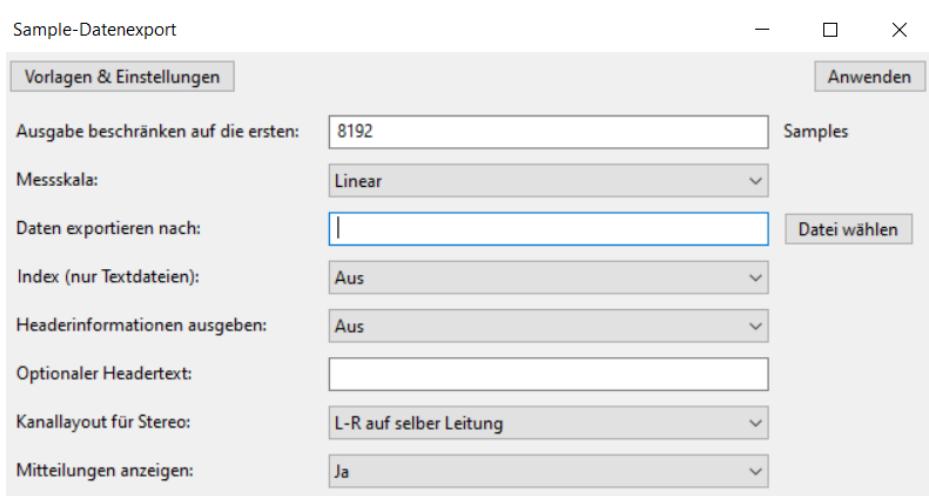


Abbildung 40: Sample-Datenexport Fenster

Hier wird als erste Option die Anzahl der Samplepunkte gewählt. Da im Testprogramm für die maximale Buffer Größe NFFT 8192 definiert wurde, wird dies hier ebenfalls gewählt. Die Option Messskala muss auf linear gestellt werden. Als Zielort wird der Pfad zum Python Tool angegeben.

Durch einen Klick auf den Button „Anwenden“ werden die Daten exportiert und das Fenster schließt sich.

0.00256  
0.00223  
0.00204  
...  
*Datenauszug 1: Ausgabe Audacity*

Der obenstehende zeigt die ersten drei Datenpunkte der ausgegebenen Datei.

#### **4.3.6.3 Das Python Tool**

Um die von Audacity ausgegebene Datei für den C-Compiler lesbar zu machen, müssen die Daten in ein Array gespeichert werden. Um die zu automatisieren, wurde mit Python das im Folgenden beschrieben Tool entwickelt. Die Sprache eignet sich aufgrund ihrer Einfachheit gut für solche Werkzeuge.

Am in Audacity festgelegten Zielort befindet sich nun die Datei „sample-data.txt“. Diese enthält einen Datenpunkt pro Zeile. Damit das Testprogramm damit arbeiten kann muss nun ein Array mit den Werten befüllt werden, welches in der Datei „processed-data.h“ erstellt wird.

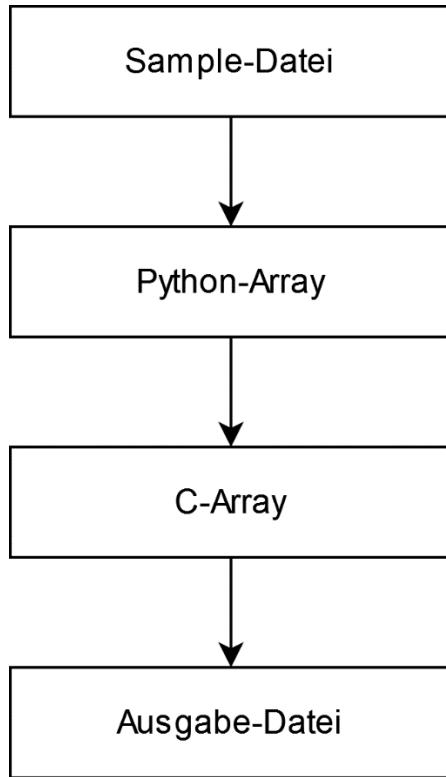


Abbildung 41: Dataflow Python Tool

Die Funktionsweise des Tools ist im Folgenden beschrieben.

```
import shutil  
Codesegment 11: Import des "shutil" Moduls
```

In Codesegment 11 wird Pythons „shutil“-Modul importiert, welches benötigt wird, um mit externen Dateien zu arbeiten.

```
arr=[]  
arr = [0 for i in range(8192)]  
i = 0  
Codesegment 12: Variablen Initialisierung
```

Ein Array mit der Größe 8192 wird in Codesegment 12 erstellt. In dieses werden die Werte aus der „.txt“ Datei gespeichert. Die Variable *i* dient hier als Zähler.

```
with open('sample-data.txt') as f:
    while True:
        line = f.readline()
        if not line:
            break
        line = line.strip()
        data = float(line) * 25
        #print(data)
        arr[i] = data
        i += 1
f.close()
```

*Codesegment 13: Datei auswerten*

Codesegment 13 öffnet die Datei „sample-data.txt“. In der While-Schleife wird jede Zeile an die korrespondierende Stelle im Array gespeichert. Die If-Verzweigung beendet die Schleife, sobald sie am Ende der Datei angelangt ist.

```
file_to_delete = open("processed-data.h", 'w')
file_to_delete.close()
```

*Codesegment 14: Löschung vorhandener Ausgabedatei*

Das Array soll in den Header „processed-data.h“ gespeichert werden, um es einfach in das Testprogramm integrieren zu können. Um sicherzustellen, dass keine Überschneidungen mit einer veralteten Version des Headers auftreten, wird diese gelöscht.

```
with open('processed-data.h', 'w') as d:
    d.write('float test_buffer[] = {')
    for data in arr:
        d.write(str(data) + ',')
    d.write('};')
d.close()
```

*Codesegment 15: Ausgabe der Datei*

Codesegment 15 erstellt den Header als leere Datei. Dieser wird befüllt mit einem nach C-Syntax deklarierten Array. Nach der geöffneten, geschwungenen Klammer wird der Inhalt des Python-Arrays in für die C-Syntax gültiger Form geschrieben und die geschlossene Klammer angehängt. Daraufhin wird die Datei geschlossen.

```
shutil.copyfile('processed-data.h',
'/home/laurenz/Dokumente/GitHub/MTAP-MIDI-Guitar-
Converter/firmware/ESP_DSP/src/processed-data.h')
```

*Codesegment 16: Kopieren der Datei*

In Codesegment 16 wird die Datei noch in das Verzeichnis des Testprogramms kopiert. Nach dem Ausführen des Tools ist der Test Buffer bereit und das Testprogramms kann ausgeführt werden

#### 4.3.7 Ausgabe

Der Output nach einem Ausführen sieht folgendermaßen aus:

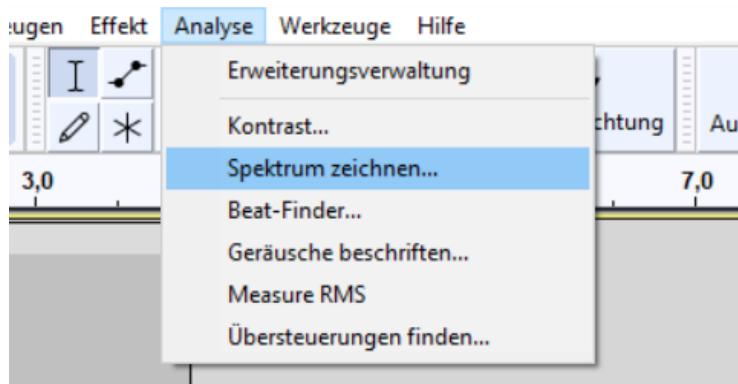
```
20-th magnitude: 2.373904 => corresponds to 107.666016 Hz
keyNR: 25
21-th magnitude: 2.073598 => corresponds to 113.049316 Hz
keyNR: 25
41-th magnitude: 3.630623 => corresponds to 220.715332 Hz
keyNR: 37
```

*Datenauszug 2: Output des Testprogramms*

Die Richtigkeit des Outputs wird mit Audacity und Matlab überprüft.

##### 4.3.7.1 Audacity

In Audacity muss in der Menüleiste unter Analyse, Spektrum zeichnen gewählt werden.



*Abbildung 42: Audacity Analyse Tools*

Im dadurch geöffneten Fenster kann nun das Frequenzspektrum untersucht werden.

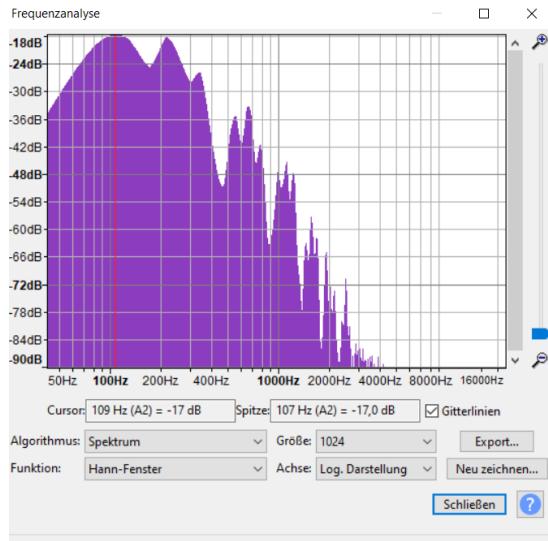


Abbildung 43: Frequenzanalyse Audacity

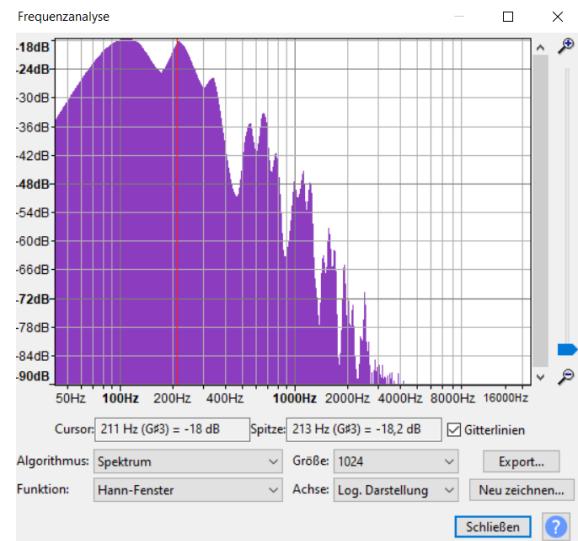


Abbildung 44: Frequenzanalyse Audacity

In Abbildung 43 ist zu erkennen, dass neben der Grundfrequenz von 107 Hz noch eine zweite sehr prominent heraussticht. Dabei handelt es um eine Harmonische, deren Frequenz ein Vielfaches der Grundfrequenz ist. Diese haben nicht immer eine niedrigere Amplitude als der Grundton, weshalb sie bei der Notenerkennung eine besondere Herausforderung sind.

Um etwaige Erkennungsprobleme, durch Harmonische, zu umgehen, wird im Hauptprogramm die tiefste Frequenzspitze als Grundton angenommen.

#### 4.3.7.2 Matlab

Audacity liefert mit seiner Frequenzanalyse das Frequenzspektrum, welches jedoch interpoliert wird, um einen Graphen zu zeichnen. Ob die Ausgabe des Programms und somit der FFT stimmt, wird deshalb mit Matlab geprüft.

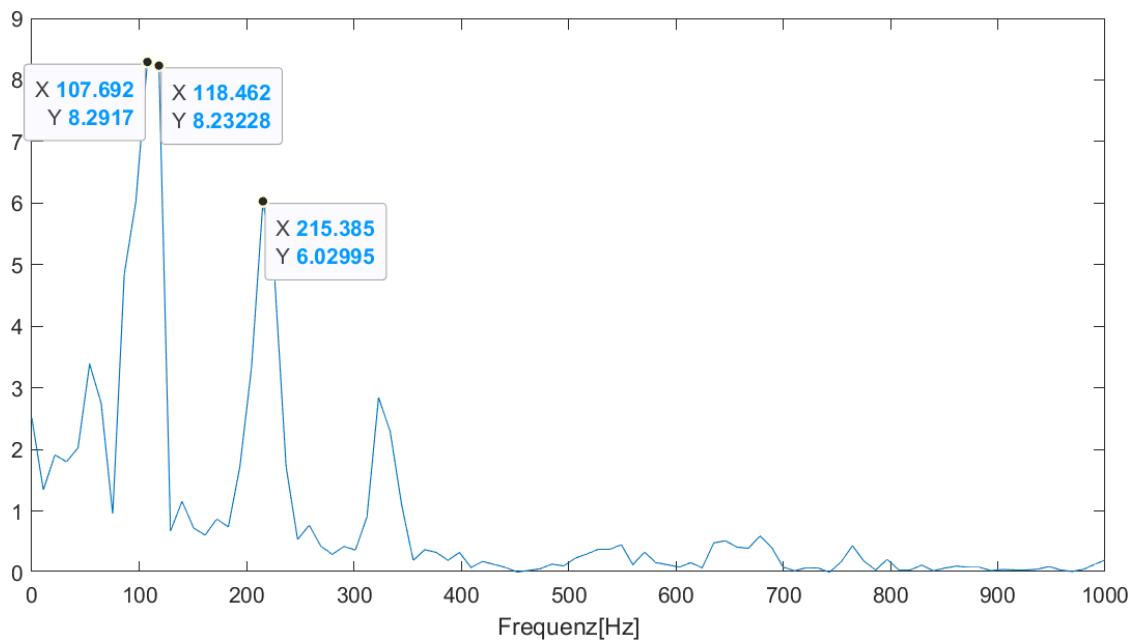
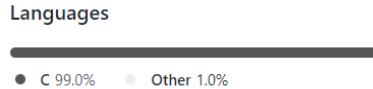


Abbildung 45: Matlab FFT

Abbildung 34 zeigt den Plot der Matlab FFT. In diesem wurden die 3 Frequenzen, welche das Testprogramm ausgibt, markiert. Es fällt auf, dass bei den zwei rechteren Punkten eine Abweichung von circa 5 Hz zum Testprogramm besteht. Das liegt daran, dass das Testprogramm nur einen kleinen Teil (8192 Datenpunkte) analysiert, Matlab aber die komplette „.wav“ Datei. Außerdem treten in C durch die verschiedenen Datentypen immer wieder Rundungsfehler auf. Da die Frequenzen trotz der Abweichung zu den gleichen Noten korrespondieren, kann dieser Fehler vernachlässigt werden.

## 4.4 C-Programmiersprache

Die Funktionen in der Firmware des Projektes werden beinahe ausschließlich mit der Programmiersprache C realisiert. C ist eine weitverbreitete imperative, prozedurale Programmiersprache, welche bekannt und beliebt für die Programmierung in Hardwarenähe ist. Das im Folgenden beschriebene ESP IoT Development Framework basiert auf der Programmiersprache C.



### 4.4.1 Agile Softwareentwicklung mit C

In C lassen sich die Methoden der agilen Softwareentwicklung anwenden, auf welche beim Firmware-Development geachtet wurden. Drei bekannte Denkweisen im Extreme-Programming sind die KISS-, DRY- und YAGNI-Prinzipien.

KISS was so viel heißt wie „Keep it Simple, Stupid!“ [10], ist eine im Extreme Programming angewandte Methode, welche besagt, dass die kürzeste Lösung oft die Einfachste ist. Es sollen daher keine überflüssigen Methoden implementiert werden und Klammer Neste so seicht wie möglich gehalten werden.

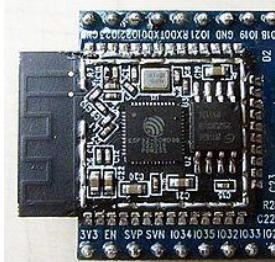
DRY steht für „Don't Repeat Yourself“ [10]. Mehrfach vorkommender Code soll in Funktionen gefasst werden, sodass bei Änderungen nur an einer Stelle gearbeitet werden muss. Daher soll eine Methode nur eine Funktion haben.

YAGNI ist die Abbreviatur der Phrase “You Ain’t Gonna Need It” [10]. Aus dem Englischen übersetzt, bedeutet sie so viel wie „Du wirst es nicht brauchen“. Oftmals hat man während der Programmierung neue Einfälle, welche über die bisherigen Planungen hinausragen und erst viel später, wenn überhaupt, benötigt werden. Anstatt die Zeit in Arbeit zu investieren, welche möglicherweise überflüssig ist, sollte man sich auf die wesentlichen Problemstellungen fokussieren.

## 4.5 ESP32

Der ESP32 ist ein Leistungsstarkes System on a Chip (SoC) mit einem Dual-Core 32-bit Xtensa LX6 Prozessor des Herstellers **Espressif**, welcher oft aufgrund seiner Vielzahl an Peripherien, Protokollen und Sensorschnittstellen, im Internet of Things (IoT) Verwendung findet.

Integrierte Schnittstellen sind beispielsweise UART, SPI, CAN, I2C, I2S, WLAN und Bluetooth. [11] [12]



*Abbildung 46: ESP32 Prozessor (Mitte), PIF WLAN-Antenne (links)*



*Abbildung 47: ESP-WROOM mit IPX-Connector*

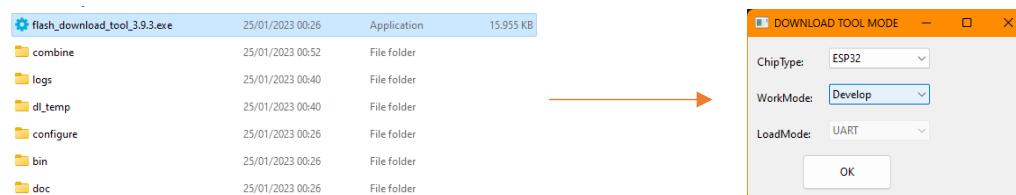
Für Prototypen sind ESP32-WROOM Module, halbfertige PCB-Module mit gekerbten Löchern als Pins, üblich. Module unterscheiden sich grundlegend im Footprint, welcher durch die Art der Antenne, PIF-Antenne<sup>3</sup> oder IPX<sup>4</sup> Connector, bestimmt ist. Eine vom Footprint unabhängige Kenngröße, ist die Größe des Flash-Memory.

### 4.5.1 Bootloader Brennen mit dem ESP-Tool

Um den ESP32 programmieren zu können, muss zuerst die Firmware (ESP-AT) auf den Chip heruntergeladen werden.

*Abbildung 48: Download des Flash Tools*

Dafür kommt das ESP Flash Download Tool<sup>5</sup> von Espressif<sup>6</sup> zum Einsatz. Dieses Tool wird dafür benötigt, die Binary-Files der Firmware über USB auf den SPI-Flash zu spielen. Im Nächsten muss zuerst das Tool heruntergeladen, entpackt und die .exe ausgeführt werden. [13]



*Abbildung 49: File Folder des heruntergeladenen Ordner*

<sup>3</sup> Planar Inverted F-Shaped Antenna

<sup>4</sup> Koaxial-Steckverbinder

<sup>5</sup> <https://www.espressif.com/en/support/download/other-tools>

<sup>6</sup> <https://www.espressif.com/en>

Mit den gezeigten Einstellungen, kann nun das Aufsetzen der Firmware fortgesetzt werden.

Natürlich werden auch die ESP-AT Binaries<sup>7</sup> selbst benötigt, welche man in den SDK-Downloads auf der Espressif Website findet. Von dem entsprechenden Modul (hier der ESP32-WROOM) muss nun die empfohlene Version der Firmware heruntergeladen werden. ESP-AT beinhaltet die Binärdateien der Partitions-Tabelle, des Bootloaders, sowie Encrytion-Keys und Certificate-Authorities (CA) von Netzwerkprotokollen wie MQTT.

Hat man den Firmware Folder entpackt, befindet sich in dem Ordner ein `flasher_args.json` File, in welchem man die Pfade zu den Binärdateien findet, sowie den Offset, welcher bestimmt an welcher Stelle im SPI-Flash die Binary installiert werden soll. [13]

```
"flash_files" : {
    "0x8000" : "partition_table/partition-table.bin",
    "0x10000" : "ota_data_initial.bin",
    "0xf000" : "phy_multiple_init_data.bin",
    "0x1000" : "bootloader/bootloader.bin",
    "0x100000" : "esp-at.bin",
    "0x20000" : "at_customize.bin",
    "0x24000" : "customized_partitions/server_cert.bin",
    "0x39000" : "customized_partitions/mqtt_key.bin",
    "0x26000" : "customized_partitions/server_key.bin",
    "0x28000" : "customized_partitions/server_ca.bin",
    "0x2e000" : "customized_partitions/client_ca.bin",
    "0x30000" : "customized_partitions/factory_param.bin",
    "0x21000" : "customized_partitions/ble_data.bin",
    "0x3B000" : "customized_partitions/mqtt_ca.bin",
    "0x37000" : "customized_partitions/mqtt_cert.bin",
    "0x2a000" : "customized_partitions/client_cert.bin",
    "0x2c000" : "customized_partitions/client_key.bin"
},
```

*Abbildung 51: flasher\_args.json - flash\_files*

Den Flasher Arguments entsprechend, müssen die Pfade sowie deren Offset in das Flash-Download-Tool eingetragen werden [13]:

#### ESP32-WROOM-32 Series

- v2.4.0.0 [ESP32-WROOM-32\\_AT\\_Bin\\_V2.4.0.0.zip](#) (Recommended)
- v2.2.0.0 [ESP32-WROOM-32\\_AT\\_Bin\\_V2.2.0.0.zip](#)
- v2.1.0.0 [ESP32-WROOM-32\\_AT\\_Bin\\_V2.1.0.0.zip](#)
- v2.0.0.0 [ESP32-WROOM-32\\_AT\\_Bin\\_V2.0.0.0.zip](#)
- v1.1.2.0 [ESP32-WROOM-32\\_AT\\_Bin\\_V1.1.2.0.zip](#)
- v1.1.1.0 [ESP32-WROOM-32\\_AT\\_Bin\\_V1.1.1.0.zip](#)
- v1.1.0.0 [ESP32-WROOM-32\\_AT\\_Bin\\_V1.1.0.0.zip](#)
- v1.0.0.0 [ESP32-WROOM-32\\_AT\\_Bin\\_V1.0.0.0.zip](#)
- v0.10.0.0 [ESP32-WROOM-32\\_AT\\_Bin\\_V0.10.0.0.zip](#)

*Abbildung 50: ESP-AT Versionen*

<sup>7</sup>[https://docs.espressif.com/projects/esp-at/en/latest/esp32/AT\\_Binary\\_Lists/ESP32\\_AT\\_binaries.html#](https://docs.espressif.com/projects/esp-at/en/latest/esp32/AT_Binary_Lists/ESP32_AT_binaries.html#)



Abbildung 52: Tool Interface mit eingetragenen Binaries

Auch die SPI Flash Konfigurationen werden aus den Flasher Arguments entsprechend übernommen: [13]

```
"flash_settings" : {
    "flash_mode": "dio",
    "flash_size": "detect",
    "flash_freq": "40m"
},
```

Abbildung 53: *flasher\_args.json - flash\_settings*

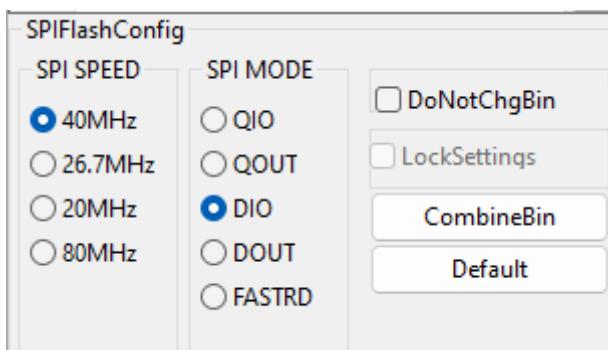
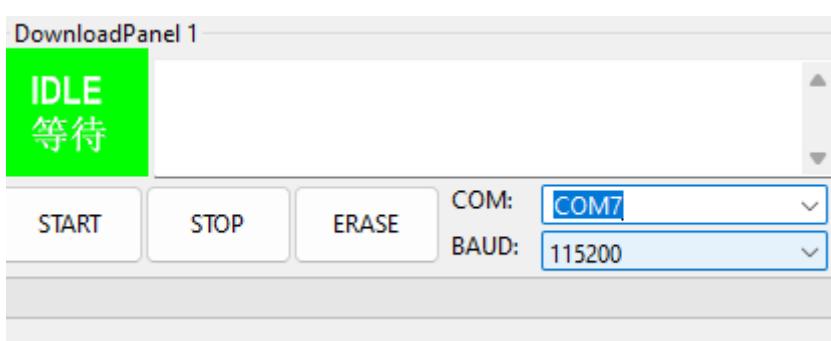


Abbildung 54: SPI Flash Config in dem GUI

Nun muss nur noch der richtige COM-Port selektiert werden und „Start“ ausgeführt werden.



Der Bootloader sollte nun erfolgreich auf den ESP32 gebrannt worden sein. Der Erfolg kann in PlatformIO, welches im folgenden Kapitel installiert wird validiert werden.

#### 4.5.2 Einrichten in PlatformIO (PIO)

Der für das Projekt verwendete Texteditor ist Visual Studio Code. Dieser kann mit diversen Plug-Ins um Features erweitert werden. Um SoCs zu programmieren ist die Erweiterung „PlatformIO“ erforderlich. Im Extensions-Tab lässt sich das Plug-In mit einem Klick auf den Button „Install“ installieren.

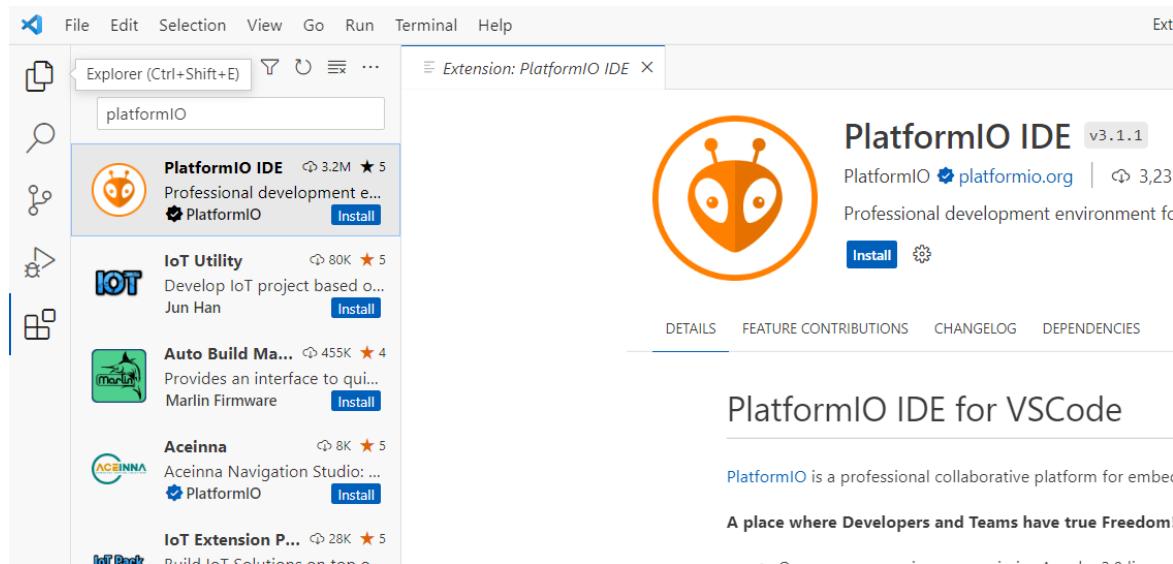


Abbildung 55: PlatformIO Installation

Nachdem der Installationsvorgang abgeschlossen ist, erscheint ein neuer Tab unter dem Extensions-Tab.

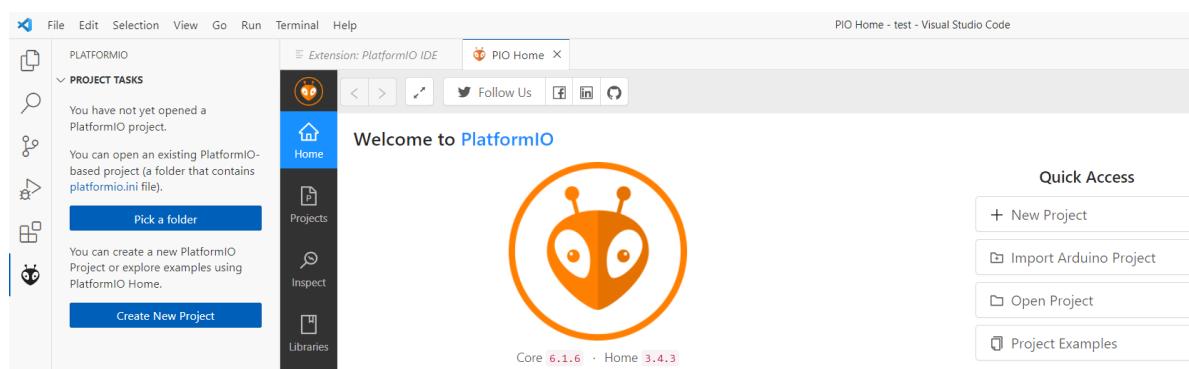


Abbildung 56: PlatformIO Home

Mit einem Klick auf die Schaltfläche „New Project“ öffnet sich ein neues Fenster, in welchem ein neues Projekt erstellt werden kann.

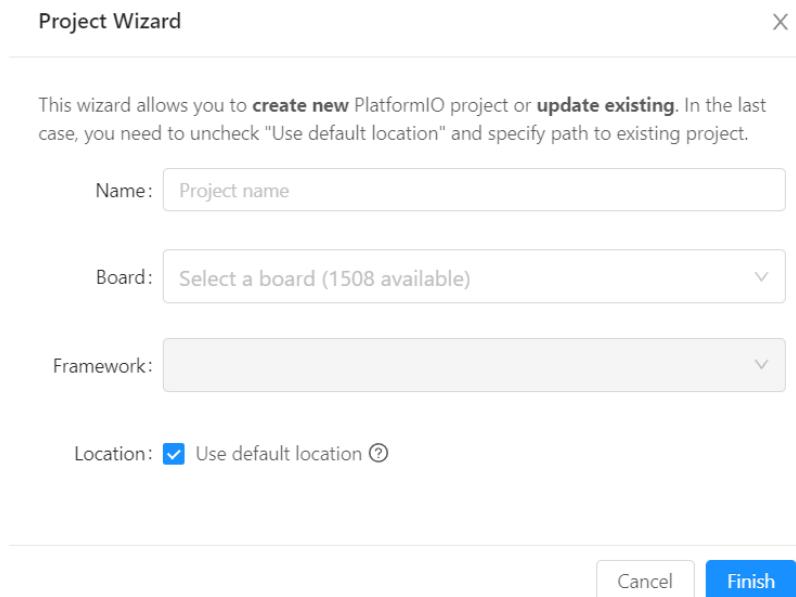


Abbildung 57: PlatformIO Project Wizard

Das Projekt wurde folgendermaßen konfiguriert:

---

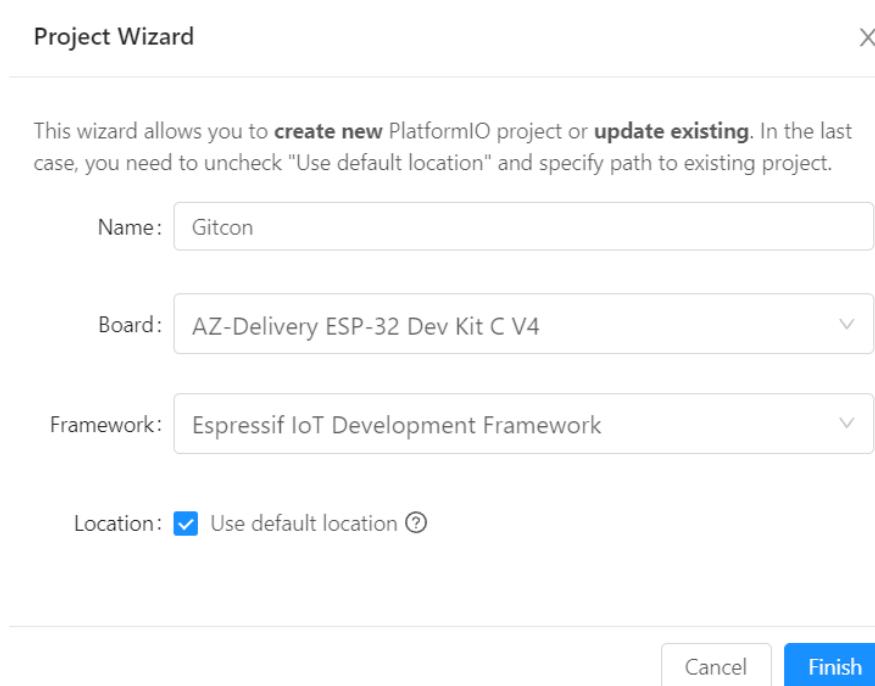


Abbildung 58: Konfiguriertes Projekt

Wird die Option „Use default location“ abgewählt, so kann ein individueller Dateipfad für das Projekt festgelegt werden.

Nach einem Klick auf „Finish“ wird das Projekt erstellt. Ist der Vorgang abgeschlossen, erscheint im Filebrowser das von PlatformIO strukturierte Projekt.

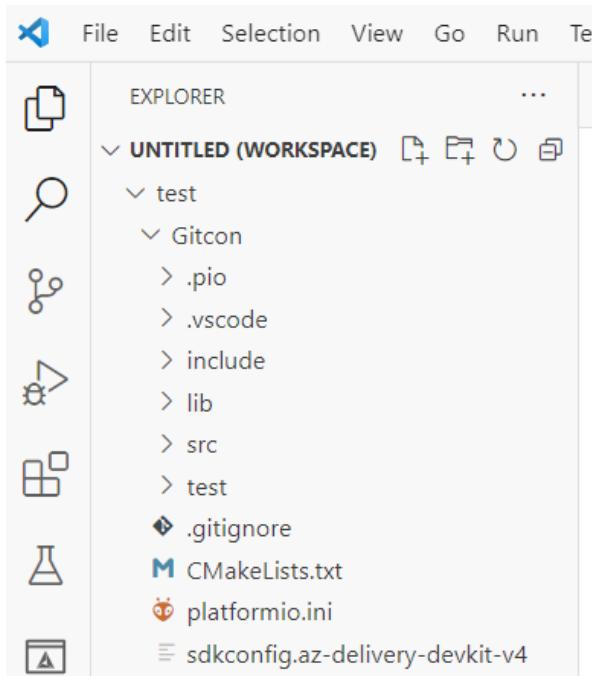


Abbildung 59: PlatformIO Projekt Struktur

Die Datei „platformio.ini“ dient als Konfigurationsdatei

Nun müssen einige Libraries importiert werden. Dafür muss im PlatformIO-Home Fenster der Reiter „Libraries“ ausgewählt werden. Im Suchfeld kann dann nach der gewünschten Bibliothek gesucht werden.

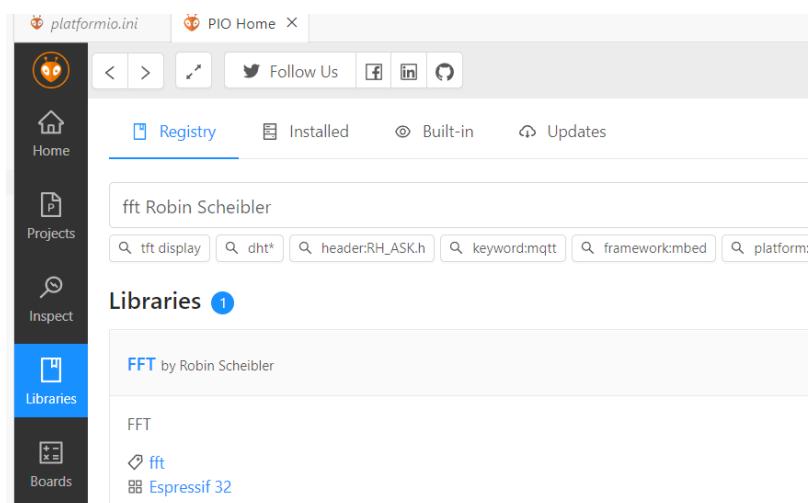


Abbildung 60: PlatformIO Library Reiter

Wählt man die Bibliothek „FFT“ aus, so öffnet sich ein Untermenü, in welchem sie dem Projekt hinzugefügt werden kann.

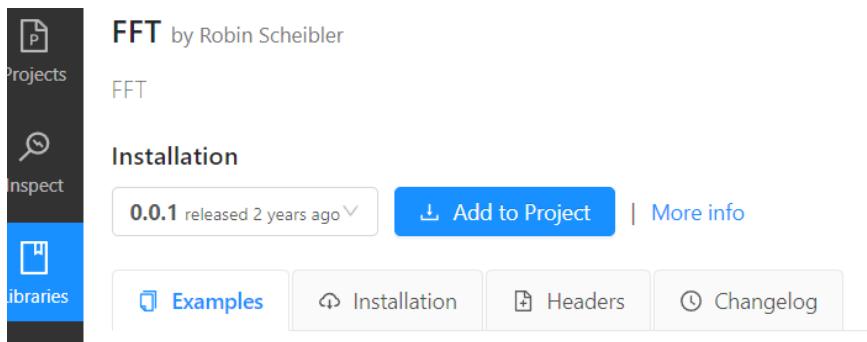


Abbildung 61: Bibliothek Untermenü

Der Knopf „Add to Project“ öffnet den Importdialog.

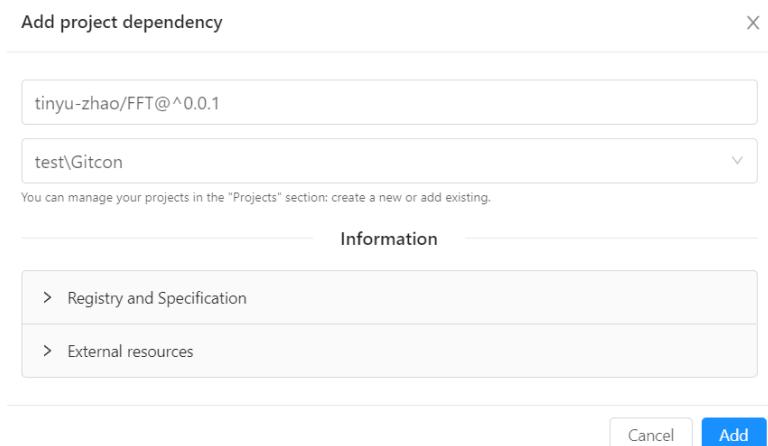


Abbildung 62: Einbinden von Dependencies

Nach dem die Schaltfläche „Add“ betätigt wurde, findet sich in der Datei „platformio.ini“ ein neuer Eintrag.

```

10
11 [env:az-delivery-devkit-v4]
12 platform = espressif32
13 board = az-delivery-devkit-v4
14 framework = espidf
15 lib_deps = tinyu-zhao/FFT@^0.0.1
16

```

Abbildung 63: platformio.ini

Eine Änderung der Konfigurationsdatei wird erst mit dem nächsten „Build“ übernommen, bei welchem die Library in den „Include“ Ordner geladen wird.

Befehle wie „Build“ oder „Upload“ können im Project-Tasks Reiter ausgeführt werden.

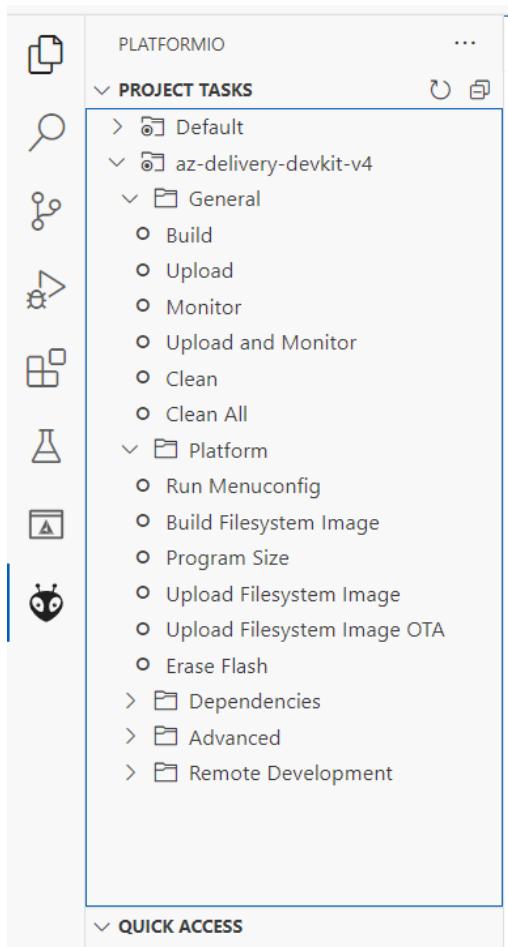


Abbildung 64: Project-Tasks Panel

Wird der Befehl „Upload“ ausgeführt, sollte der Richtige Port automatisch vom Programm gewählt werden. Dieser kann aber auch manuell in der Statusleiste am unteren Rand des Fensters geändert werden. Dazu muss auf das Steckersymbol geklickt werden. Ist das Herunterladen des Bootloaders im vorherigen Kapitel 4.5.1 misslungen, so wird beim Hochladen des Codes im Terminal ein Fehler ausgegeben.

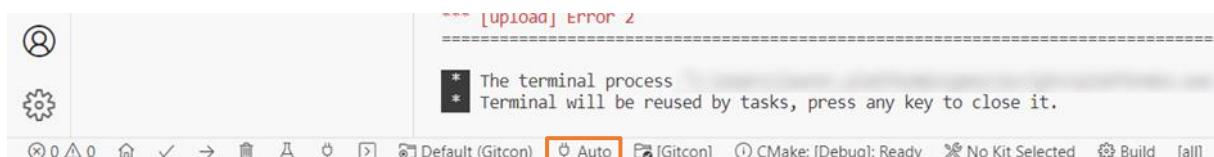


Abbildung 65: Statusleiste

### 4.5.3 ESP IoT Development Framework (ESP IDF)

Das Internet of Things Development Framework (IDF) ist ein von Espressif Systems bereitgestellte Firmware, zur Entwicklung von Software auf ESP-SoCs. Das Framework ist in der Programmiersprache C verfasst und Open Source auf der ESP-IDF Projektseite dokumentiert. Mit dem IDF lassen sich mittels bereitgestellter Funktionen die Peripherien des Systems ansteuern.

## 4.6 Realtime Operating-System (RTOS)

Oft haben Programme mehrere dedizierte Prozesse mit striktem Ablauf. Wenn nun Echtzeitdaten vermessen werden, wie zum Beispiel Audio, müssen die Laufzeiten dieser Prozesse geregelt werden, um sich gegenseitig nicht zu blockieren, da sie sich meistens einen Kern teilen. Dafür kommt ein Echtzeitbetriebssystem zum Einsatz. Der im Gitcon implementierte FreeRTOS Kernel übernimmt die besagte Aufgabe, Tasks zu priorisieren und ermöglicht dadurch einen zeitlich reibungslosen Ablauf zu ermöglichen.

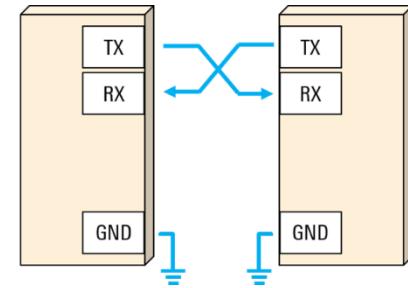
### 4.6.1 FreeRTOS

FreeRTOS ist ein marktführendes Echtzeitbetriebssystem, das für eingebettete Systeme und IoT-Geräte entwickelt wurde. Es bietet einen Multitasking-Kernel, der die gleichzeitige Ausführung mehrerer Tasks mit jeweils eigenen Prioritätsstufen und Laufzeitbeschränkungen ermöglicht. FreeRTOS ist Open Source unter der MIT Lizenz und kann privat, als auch kommerziell genutzt werden. [14]

FreeRTOS ist hochgradig konfigurierbar und seine Struktur betont geringen Overhead und hohe Leistung, wodurch es für Systeme mit begrenzten Ressourcen geeignet ist. Es enthält Funktionen wie Task-Scheduling, Kommunikation zwischen Tasks und Speicherverwaltung. FreeRTOS ist in C geschrieben und enthält Ports für eine Vielzahl von Mikrocontrollern und Prozessoren unter anderem für ESP32 basierte Systeme.

## 4.7 Universal Asynchronous Receive and Transmit (UART)

UART steht für Universal Asynchronous Receive and Transmit und ist, wie der Name bereits sagt, ein serielles, asynchrones<sup>8</sup> Übertragungsprotokoll. Mit UART, welche Datenübertragungen im Vollduplex-Betrieb unterstützt, werden zwei Peripherien miteinander verbunden. Die Bus-Teilnehmer sind hierbei gleichgestellt, was heißt, dass es keinen Mastercontroller gibt, welcher den Bus steuert. Die Daten werden einfach gesendet. Bei der Verdrahtung ist zu beachten, dass die Übertragungsleitungen überkreuzt angeschlossen werden müssen (siehe **Fehler! Verweisquelle konnte nicht gefunden werden.**). [15]



*Abbildung 66: UART-Verbindung*

UART spielt eine wichtige Rolle bei der Arbeit mit SoCs, da deren Firmware oft über eine USB-Bridge den Flashspeicher beschreiben. Die USB-Bridge spricht das System dabei mit UART an. Auch das Debuggen erfolgt meistens über diese serielle Schnittstelle.

Die übertragenen Datenpakete haben folgendes Format:

Start Bit ( 1 bit )	Data Frame ( 5 to 9 Data Bits )	Parity Bits ( 0 to 1 bit )	Stop Bits ( 1 to 2 bits )
------------------------	------------------------------------	-------------------------------	------------------------------

*Abbildung 67: UART-Datenpaket*

Ein Startbit signalisiert dem Empfänger, dass eine Übertragung beginnt. Das Potential auf der Übertragungsleitung ist *Normally-High* und wird durch das Startbit auf *Low* gezogen. Anschließend folgt das Datenframe, welches je nach Konfiguration fünf bis neun Bit lang ist. Eine Paritätsbit dient zur Validierung der Übertragung. Dieses Bit kann aber auch im Controller ausgeschalten werden. Am Ende der Übertragung setzt ein Stop Bit den Bus wieder auf den Idle-Zustand. [16]

<sup>8</sup> Asynchron: Ohne Takteleitung

## 4.8 Musical Instrument Digital Interface (MIDI)

Das MIDI-Protokoll wurde in den frühen 80ern entwickelt und standardisiert die Kommunikation zwischen Computern und Musik-Hardware, sogenannten MIDI-Controllern. Jedes Mal, wenn eine Taste auf einem Controller gedrückt wird, erstellt dieser eine MIDI-Nachricht und sendet sie an den Computer. Diese Tasten sind nicht nur auf die Klaviatur eines MIDI-Controllers beschränkt, es können ebenfalls MIDI-Wörter gesendet werden, welche andere Parameter in einer digitalen Musikproduktionsumgebung steuern, wie zum Beispiel die Intensität eines Audioeffekts.

Beispielsweise kann die Grenzfrequenz des in Abbildung 68 gezeigten digitalen Filters mittels eines encodierten Potentiometers am MIDI-Controller gesteuert werden.

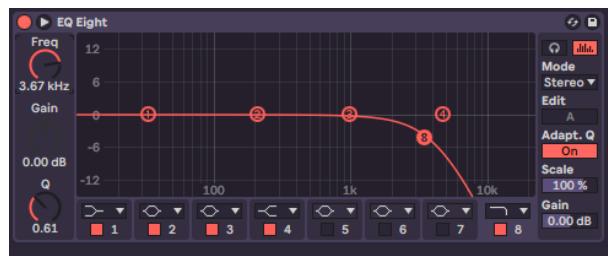


Abbildung 68: Ableton Live Effekt EQ Eight (Filter)

Es ist wichtig zu wissen, dass MIDI-Signale nichts mit niederfrequenten Audiosignalen zu tun haben. Weder analoges- noch digital aufgefasstes Audio kommt in einer MIDI-Kommunikation vor. [17]

Das MIDI-Protokoll beschreibt unter anderem die zu übertragenden Noten mit digitalen Wörtern, welche in einer genormten Tabelle festgehalten sind. Jede Note hat daher eine Adresse, durch welche dann ein anderes digitales Instrument weiß, mit welcher Tonhöhe es diese Note spielen muss.

### 4.8.1 Status Bytes

Eine MIDI-Nachricht ist aus drei (manchmal zwei) Bytes aufgebaut:

- Status und Kanal
- Erstes Datenbyte
- Zweites Datenbyte

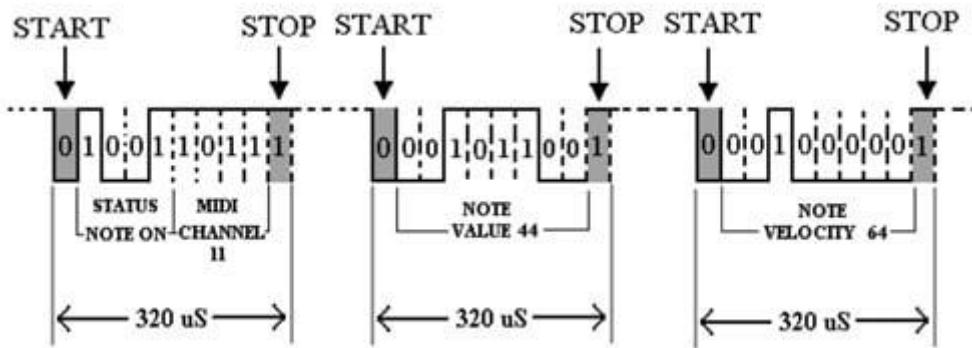


Abbildung 69: MIDI-Word

#### 4.8.1.1 MIDI-Status

Um das MIDI-Wort so kurz wie möglich zu halten, teilen sich das Status- und Kanal nibble ein Byte. Mit dem Status nibble wird dem Empfänger mitgeteilt, welche Funktion die darauffolgenden Datenbytes haben. Mit 16 verschiedenen Status, kann er mit vier Bit eindeutig identifiziert werden. Mögliche Status nibble sind:

Status	Hex-Wert	Funktion
Note Off	0x8	Beendet das Spielen einer Note
Note On	0x9	Startet das Spielen einer Note
Polyphon Pressure	0xA	Löst beim Nachdrücken einer Keyboard Taste aus (gilt für die jeweils gedrückte Note)
Control Change	0xB	Spricht Steuerelemente, wie z.B. Regler in der DAW an.
Program Change	0xC	Wechselt zwischen verschiedene Klänge und Instrumente
Channel Pressure	0xD	Löst beim Nachdrücken einer Keyboard Taste aus (gilt für alle Noten auf dem Kanal)

Pitch Bend	0xE	Ändert die Tonhöhe des Gesamten Kanals in feinen Schritten.
System Messages	0xF	MIDI Clock, Start, Stop, System Reset, Active Sensing

Tabelle 2: MIDI-Status

#### 4.8.1.2 MIDI-Kanal

Mit den Vier übrigen Bit wird der Kanal eingestellt, an den die Nachricht geschickt werden soll. Ein einzelner Controller kann also bis zu 16 verschiedenen MIDI-Kanäle in einer DAW ansprechen. Auf einem MIDI-Keyboard würden zum Beispiel die Klaviertasten auf einen anderen Kanal geschickt werden als die Drumpads.



Abbildung 70: Vier Verschieden MIDI-Spuren mit dem gleichen Controller als Input

Abbildung 71: Novation<sup>9</sup> Launchkey Mini

Drumpads (Channel 10)

Klaviertasten (Channel 1)

#### 4.8.1.3 Datenbytes

Für das Projekt sind nur die Funktionen *Note On* und *Note Off* relevant, weshalb die Dokumentation der Datenbytes auf diese beiden Status beschränkt ist.

Im Datenbyte eins ist bei beiden Status die Adresse der Note gespeichert, während in Byte zwei die Anschlagstärke übertragen wird. Ungewöhnlicherweise wird auch beim Ausschalten einer Note die Anschlagstärke (Loslass-

<sup>9</sup> <https://novationmusic.com/de/keys/launchkey-mini>

Geschwindigkeit) übertragen, welche je nach Empfänger beispielsweise als Nachklang interpretiert werden könnte. [17]

Die Adresse der Note lässt sich wie folgt herausfinden: Die Note A0 entspricht der Adresse 21 (dezimal). Von dort aus zählt man linear aufwärts bis 127, welches der Note G9 entspricht. Noten unter A0 können auch übertragen werden, jedoch sind die Frequenzen, welcher sie entsprechen, nicht mehr im hörbaren Bereich. Die Anschlagstärke einer Note bestimmt die initiale Lautstärke. Hat seine Note die Anschlagstärke 0, so wird sie automatisch als *Note Off* interpretiert. [18] [9] [17]

Soll beispielsweise die Note E2 mit einer Anschlagstärke von 64 auf Kanal 1 angeschaltet werden, sieht das übertragene Wort folgendermaßen aus:

	Status	Kanal	Byte 1 (Adresse)	Byte 2 (Anschlagstärke)
Dezimal	144		40	64
Hexadezimal	0x90		0x28	0x40
Binär	1001	0000	00101000	0100 0000

Tabelle 3: Beispiel eines MIDI Worts

Nach dieser Übertragung bleibt die Note so lange angeschaltet, bis ein *Note Off* Signal auf dieselbe Adresse und den selben Kanal gesendet wird.

## 5 Ergebnisse

Mit dem Wissen der Grundlagen, lassen sich nun die Entwicklungsergebnisse im Detail erklären. Hierbei wird darauf geachtet, dass jede Funktion hinterfragt und die Wahl der Komponenten genau begründet ist.

### 5.1 Blockschaltbild

Mit dem folgenden Funktionsblockdiagramm wird die Prozesskette des Guitar Converter beschrieben.

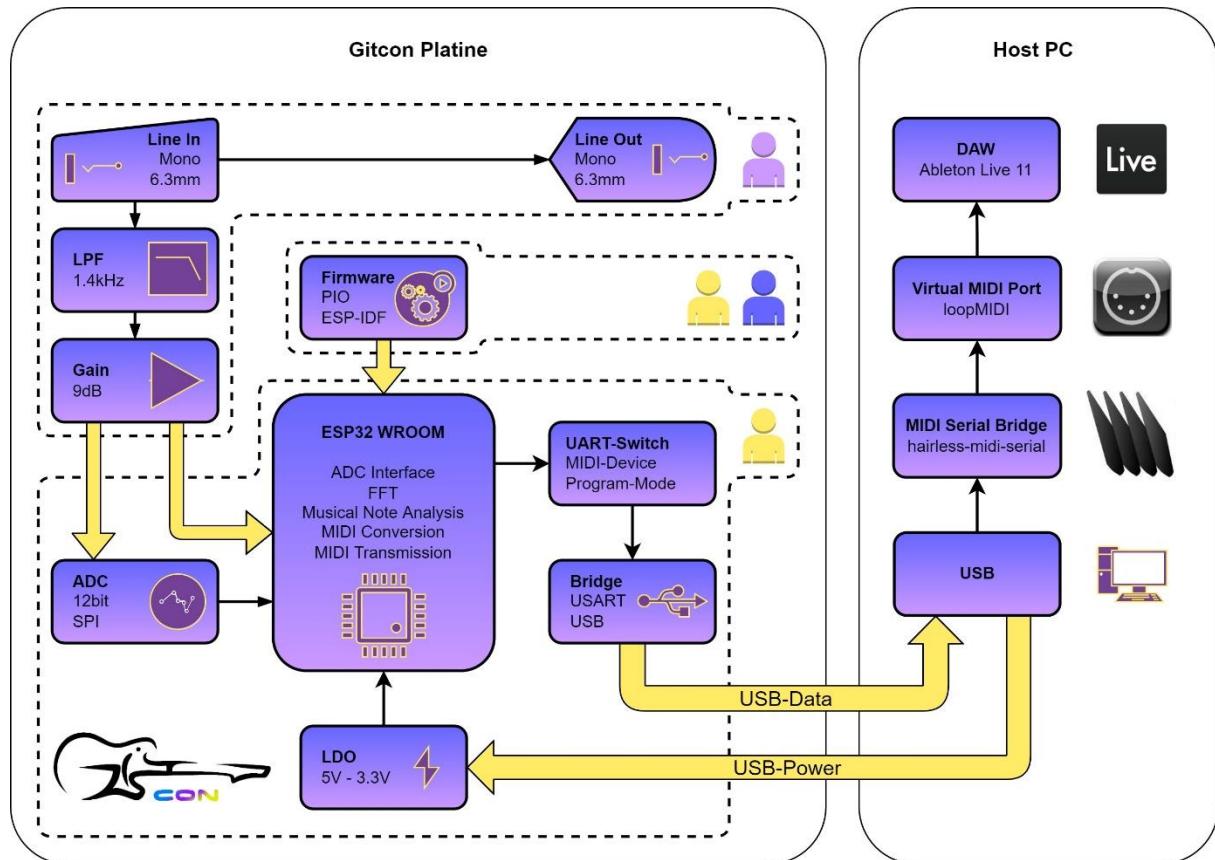


Abbildung 72: Funktionsblockdiagramm des Gitcon

Gestrichelte Linien visualisieren die drei primären Sektionen, in die die Prozesskette unterteilt ist:

- Analoges Frontend
- Digitalisierung des Signals
- Firmware, die für die DSV zuständig ist und die Daten auf den USB schickt.

Dicke Gelbe Pfeile stellen den Übergang einer Sektion in eine andere dar.

Mit den Figuren innerhalb der strichlierten Linien, wird noch einmal zusätzlich die Arbeitsunterteilung veranschaulicht. Mit der Farbe der jeweiligen Figur, wird auf ein Teammitglied referiert.



*Abbildung 73: Legende der Arbeitsunterteilung*

#### Sektion 1: Analoges Frontend

Das analoge Frontend besteht aus einem Tiefpassfilter und einem Verstärker. Da in der digitalen Signalverarbeitung nur die Grundschwingung erfasst werden soll, sind sämtliche höhere Frequenzen unbrauchbar und nur ein weiterer Störfaktor des Systems. Aufgrund dessen, werden die Frequenzen oberhalb des höchsten Tones, welchen man mit einer Gitarre im Standard-Tuning spielen kann( $\sim 1200\text{Hz}$ ), gedämpft. [19]

Die Spannungen, welche von den Pickups einer E-Gitarre induziert werden, haben eine sehr geringe Amplitude ( $\sim 100\text{mV}$ ) [19]. Werden mehrere Saiten gleichzeitig angeschlagen, kann sich die Spitzenspannung aufgrund der sechs Saiten auf circa das Sechsfache erhöhen. Der Verstärkungsfaktor im Pass-Band des Filters erhöht die Amplitude des Signals um 9dB (Faktor 2,82), um die Genauigkeit, mit welcher das Signal vom digitalen Hardware Frontend aufgefasst werden kann, zu erhöhen.

#### Sektion 2: Digital Hardware Frontend

Im digitalen Hardware Frontend findet die Digitalisierung des Signals statt. Ein Analog Digital Wandler (ADC) digitalisiert das Audio und stellt die Daten dem ESP32 bereit. Aufgrund von bekannten Ungenauigkeiten und Störeinflüssen des internen ADC des ESP32, wurde zur Absicherung ein externer 12-Bit ADC verbaut [20]. Dieser wurde aber wegen der zufriedenstellenden Funktionalität des internen AD-Wandlers nicht weiter benötigt und auch nicht auf der Hauptplatine verlötet.

### Sektion 3: Firmware

Sobald die Daten digitalisiert sind, analysiert die Firmware das aufbereitete Signal der E-Gitarre und erkennt die Frequenzanteile, aus welchen sich die gespielte Note ergibt. Erkannte Noten werden auf den USB-Port geschickt.

## 5.2 Hardware

Im folgenden Teil wird zunächst der Schaltplan des funktionalen Blockschaltbildes (Abbildung 72) dokumentiert, sowie die Wahl der Komponenten begründet. Darauffolgend werden die Layout Kriterien und Guidelines für das Leiterplatten-Design beschreiben.

### 5.2.1 Versorgung

Die Komponenten, die zur Realisierung des Blockdiagrammes gewählt wurden, müssen mit 3,3 Volt versorgt werden. Da uns der USB-Bus jedoch einen Pegel von 5 Volt zur Verfügung stellt, muss dieser zuerst heruntergeregt werden. Hierzu wird ein Low Drop-Out Regulator verwendet.

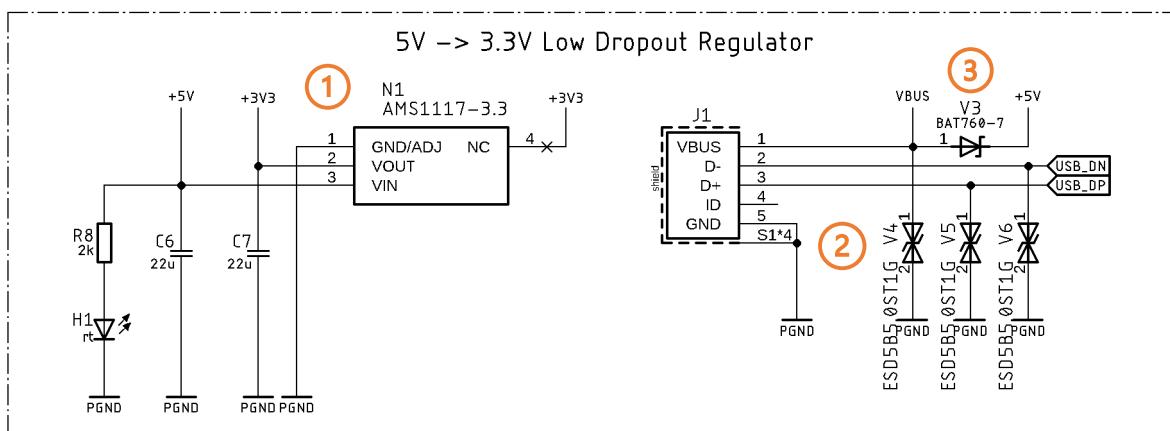


Abbildung 74: Spannungsregler und USB-Connector im Schaltplan

#### 5.2.1.1 LDO (AMS1117-3.3)

Ein Low-Dropout Regulator ist eine Art linearer Spannungsregler mit einem möglichst geringen Spannungsabfall über dem Ein- und Ausgangskontakt. LDOs werden verwendet, um das Rauschen in Versorgungsleitungen zu eliminieren aber auch, um eine (etwas) höhere Spannung in eine niedrigere Spannung zu regeln. In diesem Fall werden die 5 Volt der USB-Schnittstelle, in die von den Peripherien am Board benötigten, 3,3 Volt heruntergeregt.

Gegenüber Schaltreglern haben sie den Vorteil, dass sie kompakt sind und kein Schaltrauschen verursachen. Auch der Wirkungsgrad ist aufgrund der niedrigen Dropout-Spannung hoch genug, jedoch geringer als bei Schaltreglern. Mit einem maximalen Ausgangsstrom von 1 Ampere reicht er vollkommen aus, den Gitcon mit Strom zu versorgen. Jedoch ist zu beachten, dass, um die Stabilität zu gewährleisten, ein  $22\mu F$  (Tantal-Elektrolyt-) Kondensator am Ausgang angebracht

werden sollte. C6 und C7 halten hier die Ein- und Ausgangsspannung bei Spannungseinbrüchen möglichst konstant. Außerdem leuchtet eine rote LED leuchtet auf, wenn das Gerät eingesteckt ist.

### 5.2.1.2 ESD-Schutz

ESD ist die Abkürzung für *electrostatic discharge* und beschreibt die plötzliche Entladung eines Objekts auf ein anderes, wenn sie miteinander in Kontakt treten. Der Potentialunterschied zwischen den Objekten, kann je nach Material mehrerer **Kilovolt** erreichen.



Abbildung 75:  
ESD-Warnhinweis

Integrierte Schaltkreise, wie sie in diesem Projekt enthalten sind, sind typischerweise sehr empfindlich gegenüber statischer Entladung. Aufgrund des vielen Interagieren mit der Platine, zum Beispiel beim Ein- und Ausstecken, muss mittels TVS<sup>10</sup>-Dioden V4-6 eine Abschirmung gegen statische Entladung am USB-Eingang vorhanden sein.



Abbildung 76: ESD-Schutz in Schaltungen

<sup>10</sup> Transient Voltage Suppression

### 5.2.1.2.1 Kennwerte der TVS-Diode

#### Uni- oder Bidirektional

Je nachdem ob die zu schützende Leitung über oder unter dem Massepotential liegt, muss man bei unidirektionalen TVS-Dioden die Polung beachten. Bidirektionale arbeiten in beide Richtungen.



Abbildung 77: Symbol der Bipolaren TVS-Diode

#### Anzahl von Kanälen

Oft haben Konnektoren eine Vielzahl an Pole (z.B. HDMI) die geschützt werden müssen. Deshalb gibt es mehrere TVS-Dioden in einem einzigen Package.

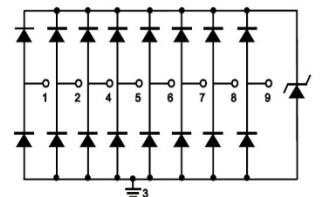


Abbildung 78: SC7538-08UTG Block Diagramm

Die Arbeitsspannung ist die maximale Spannung, welche im normalbetrieb auf der Leitung vorkommt. Hier 3,3 Volt.

#### Klemmspannung

Im Falle eines ESD-Spikes, wirkt die TVS-Diode niederohmig gegen Masse. Die Spannung, die dabei an der Diode abfällt, ist die Klemmspannung. Obwohl sie immer noch signifikant höher ist als die Betriebsspannung, ist sie aufgrund der kurzen Dauer nicht weiter gefährlich, trotzdem sollte sie in jedem Anwendungsfall möglichst gering gewählt werden. Einen genauen Wert dafür findet man nur sehr schwer. Man sollte im Datenblatt der TVS-Diode auf Verweise für typische Applikationen achten.

#### Intrinsische Kapazität

Wie bei jedem Bauteil, gibt es gewisse parasitäre Kenngrößen, die im inneren präsent sind. Bei der TVS-Diode ist die Kapazität eine störende, dennoch unvermeidbare, Größe. Bei TVS-Dioden mit hoher Kapazität besteht die Gefahr, dass sehr kurze ESD-Stöße nicht gefiltert werden können. Abgesehen von der intrinsischen Kapazität, ist zu berücksichtigen, dass auch die Leiterbahn von der Buchse zur Diode keine eigene signifikante Kapazität

aufweisen darf. Dies wird durch eine möglichst kurze Leiterbahnhöhung sowie eine dünne Leiterbahnbreite realisiert.

#### IEC61000-4-2 Rating

Die TVS-Diode verfügt über eine Rating-Stufe der IEC61000-4-2 Norm, welche Aussagen über die absoluten Höchstwerte des ESD-Spikes liefert.

#### 5.2.1.3 Verpolungsschutz

Obwohl es nur schwer möglich ist, einen USB-Stecker falschherum einzustecken, verfügt die Schaltung trotzdem über einen Verpolungsschutz. Sobald eine negative Spannung am Eingang anliegt, wirkt die Schottky-Diode V3 entgegen dem Stromfluss. Der Vorwärtsspannungsabfall an der Diode ist hierbei nicht problematisch, da die Spannung nach der Diode auf 3.3V heruntergeregt wird.

## 5.2.2 Analog-Frontend (AFE)

Warum wir einen Filter benötigen, lässt sich aus den Eigenschaften des Gitarren Pickups sowie das Schwingverhalten einer Saite erschließen.

### 5.2.2.1 Schaltung

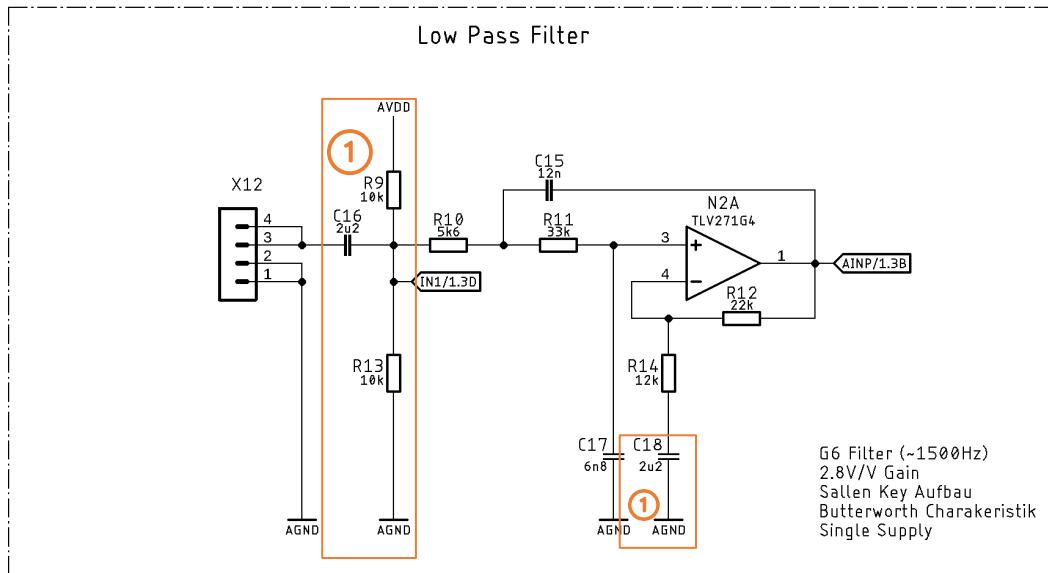


Abbildung 79: LPF-Schaltung

Als Operationsverstärker wird der OPV TLV271G4 von Texas Instrumentes verwendet, dieser ist Teil der TLV27x- Serie und wurde für Anwendungen mit niedrigem Stromverbrauch und dem Einsatz in batteriebetriebenen Geräten entwickelt. Der OPV hat eine breite Versorgungsspannungsbereich von 2,7V bis 12V und kann einen Ausgangsstrom von bis zu 60 mA liefern. Weiters verfügt dieser über einen hohen Eingangswiderstand, was zu einer erhöhten Empfindlichkeit gegenüber Eingangssignalen macht.

#### 5.2.2.1.1 Einkopplung des Audios

Da der Gitcon über keine negative Versorgung verfügt, muss das Audio zwischen Masse und +3.3 Volt eingekoppelt werden. Dafür sorgen die Widerstände  $R_9$ ,  $R_{13}$  und der Kondensator  $C_{16}$ . Wobei ein Kondensator nur Wechselspannungen durchlässt und Gleichspannungen herausgefiltert. Eine erfolgreiche Einkopplung wird erreicht, indem der Kondensator die Spannung auf das Potential zwischen  $R_9$  und  $R_{13}$  anhebt bzw. senkt.

### 5.2.2.2 Filterprototyp



Abbildung 80: Filter Prototyp mit TL074

### 5.2.2.3 Filterdimensionierung

Eine gute Möglichkeit einen Filter zu dimensionieren ist mithilfe von dem TI Filter Design Tool (<https://webench.ti.com/filter-design-tool/>). Durch Messen wurden die Bauteilwerte ermittelt und an eine optimale Übertragungskennline angepasst. Im folgenden Kapitel wird eine Dimensionierung eines Sallen-Keys durchgeführt.

#### 5.2.2.3.1 Berechnung eines idealen Sallen-Key:

Übertragungsfunktion für einen Sallen-Key zweiter Ordnung:

$$H(s) = \frac{\frac{1}{R_1 R_2 C_1 C_2}}{s^2 + s \left( \frac{1}{R_1 C_1} + \frac{1}{R_2 C_1} \right) + \frac{1}{R_1 R_2 C_1 C_2}} \quad (37)$$

durch vereinfachen:

$$H(s) = \frac{a_0}{s^2 + a_1 * s + a_0} \quad (38)$$

$$a_1 = \frac{1}{R_1 C_1} + \frac{1}{R_2 C_1} ; a_0 = \frac{1}{R_1 R_2 C_1 C_2}$$

Wenn man  $R_1$  und  $R_2$  (bzw. die normalisierten Werte  $R_{1n}$  und  $R_{2n}$ ) gleichsetzt und  $\omega_c = 1$  setzt, kann man sich die normalisierten Werte für  $C_1$  und  $C_2$  ( $C_{1n}$  und  $C_{2n}$ ) berechnen.

$$\omega_c = 1 \rightarrow a_0 = 1 ; a_1 = \sqrt{2}$$

$$R_{1n} = R_{2n} = 1 ; C_{1n} * C_{2n} = 1$$

$$C_{2n} = \frac{1}{C_{1n}} ; a_1 = \frac{2}{C_{1n}} = \sqrt{2}$$

(*∴ deshalb*)

$$\therefore C_{1n} = \sqrt{2} = 1,414 F$$

$$C_{2n} = \frac{1}{C_{1n}} = 0,707 F$$

Da die Widerstandswerte sehr klein und die Kapazitäten unrealistisch sind müssen diese skaliert werden. Wenn man m als Skalierungsfaktor bezeichnet, werden die Widerstände m-mal erhöht und die Kapazitäten um 1/m verringert, um dieselbe Grenzfrequenz von 1 zu behalten.

$$m = 10000$$

$$R_1 = R_{1n} * m = 10k\Omega$$

$$R_2 = R_{2n} * m = 10k\Omega$$

$$C_1 = \frac{C_{1n}}{m * \omega_0} = \frac{1,141}{10k * 2 * \pi * 10kHz} = 2,2nF$$

$$C_2 = \frac{C_{2n}}{m * \omega_0} = \frac{0,707}{10k * 2 * \pi * 10kHz} = 1,1nF$$

Dies ist eine ideale Dimensionierung, wobei beide Widerstände gleich sind.

### 5.2.2.4 Simulation

Der entworfene Filter wurde mit der Software LTspice simuliert. Die Gitarre liefert eine Spannung von 70-100mV. Diese Simulation wurde mit einem Eingangssignal von 100mV durchgeführt, daraus ergibt sich eine Ausgangsspannung von 280mV und eine Verstärkung von 9dB.

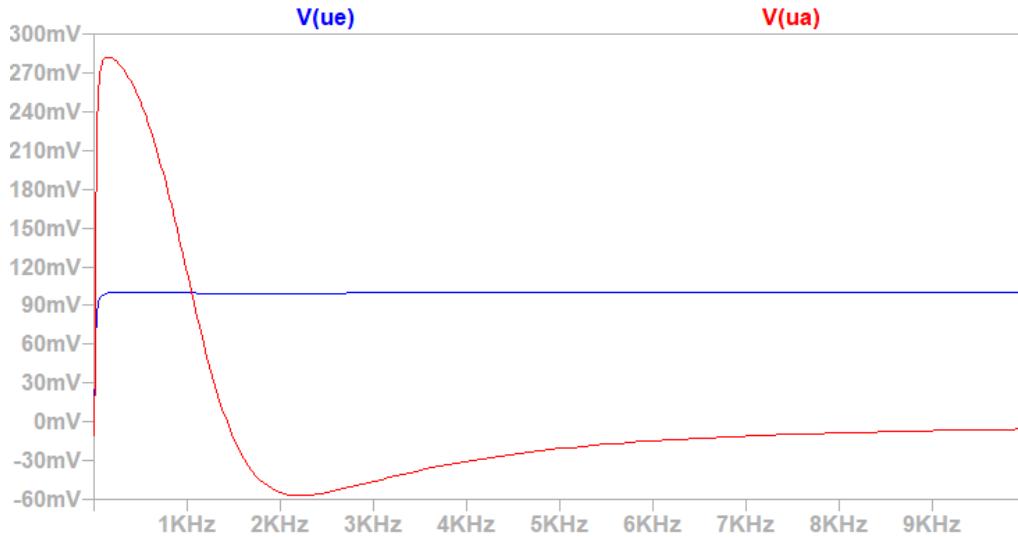


Abbildung 81: Spannungsverhältnis  $U_a/U_e$

Die negative Spannung ergibt sich aus der Phasendrehung in Abbildung 82  
**Abbildung 81 Fehler! Verweisquelle konnte nicht gefunden werden.**, da diese gegen  $-180^\circ$  dreht.

Die Verstärkung wird durch folgende Formel berechnet:

$$\nu = 20 * \log\left(\frac{U_a}{U_e}\right) = 20 * \log\left(\frac{280mV}{100mV}\right) = 9dB \quad (39)$$

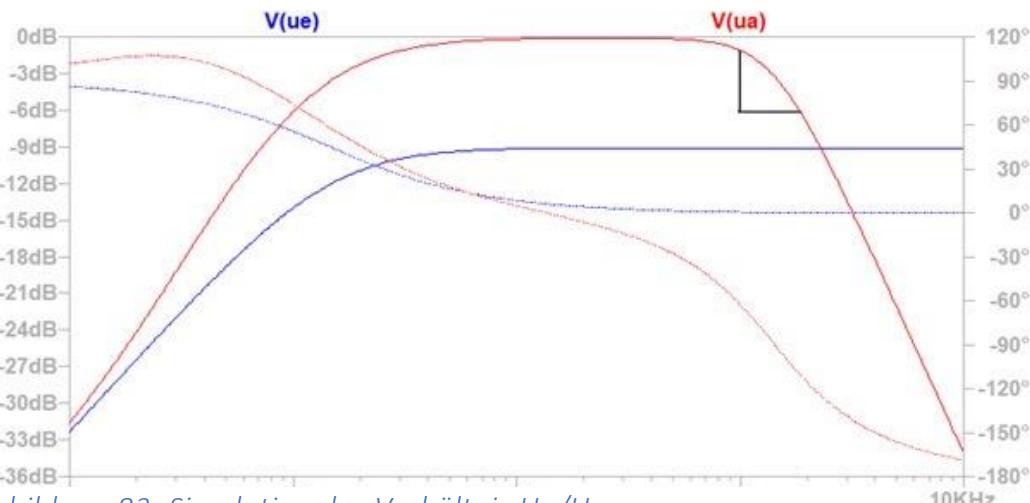


Abbildung 82: Simulation des Verhältnis  $U_a/U_e$

In der obigen Grafik ist die Verstärkung erkennbar, da  $U_a$  um 9dB höher als  $U_e$  ist. Weiters ist der Roll-off von -6dB gut erkennbar (siehe Abbildung 82), was einen Filter zweiter Ordnung charakterisiert.

#### 5.2.2.5 *Messtechnische Bestimmung des Frequenzgangs*

Die Bestimmung des Frequenzgangs wurde im Frequenzbereich von 207Hz bis 3140Hz durchgeführt, um den Roll-off Point und die abfallende Steigung gut zu erkennen.

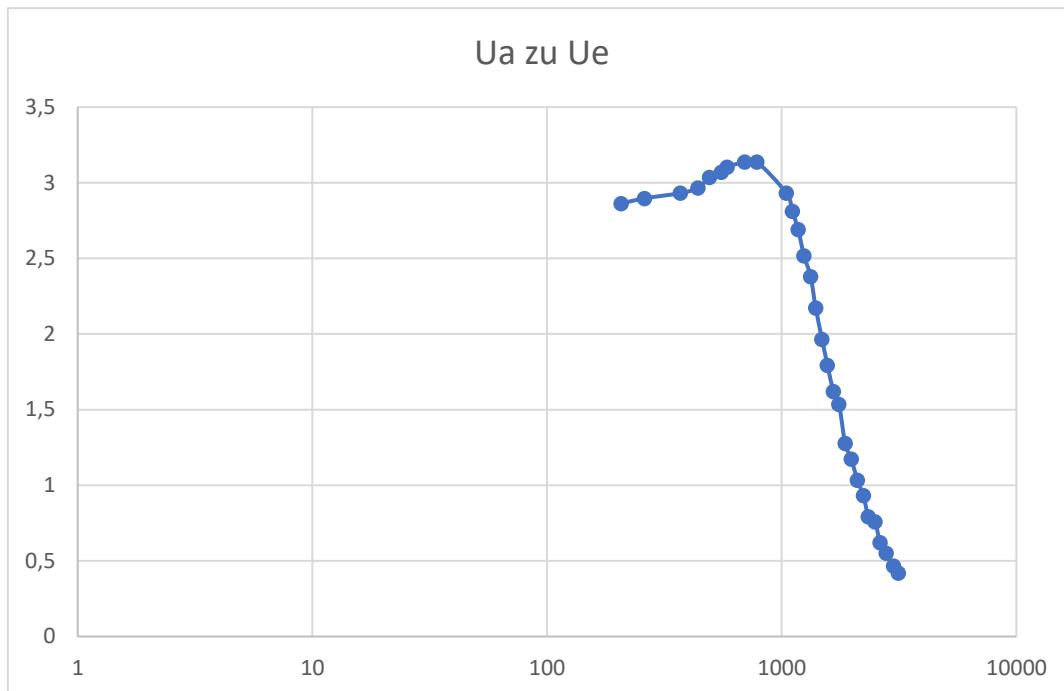


Abbildung 83: Messung des Filters

Abbildung 83 **Fehler! Verweisquelle konnte nicht gefunden werden.** zeigt die Messung des Verhältnisses  $U_a/U_e$ , hierbei ist der Roll-off bei der Grenzfrequenz, die etwas über 1000Hz liegt, ersichtlich. Auch zu erkennen ist die kontinuierlich fallende Steigung. Die Messwerte befinden sich im Toleranzbereich der Simulation und sind daher gültig.

Freq	Ua	Ua/Ue	Ue
207	3,32	2,86206897	1,16
260	3,36	2,89655172	
370	3,4	2,93103448	
440	3,44	2,96551724	
493	3,52	3,03448276	
554	3,56	3,06896552	
586	3,6	3,10344828	
696	3,64	3,13793103	
784	3,64	3,13793103	
1047	3,4	2,93103448	
1111	3,26	2,81034483	
1175	3,12	2,68965517	
1245	2,92	2,51724138	
1328	2,76	2,37931034	
1396	2,52	2,17241379	
1485	2,28	1,96551724	
1567	2,08	1,79310345	
1661	1,88	1,62068966	
1754	1,78	1,53448276	
1865	1,48	1,27586207	
1980	1,36	1,17241379	
2100	1,2	1,03448276	
2230	1,08	0,93103448	
2341	0,92	0,79310345	
2500	0,88	0,75862069	
2624	0,72	0,62068966	
2793	0,64	0,55172414	
2997	0,54	0,46551724	
3140	0,488	0,42068966	

*Tabelle 4: Tabelle mit Messpunkten*

### 5.2.3 Digital Frontend

Die nötigen Peripherien des Gitcon werden mit einem **ESP-WROOM-32UE-N8** Modul gesteuert. Die Namensgebung des Moduls liefert Informationen über die Serie des Chips:

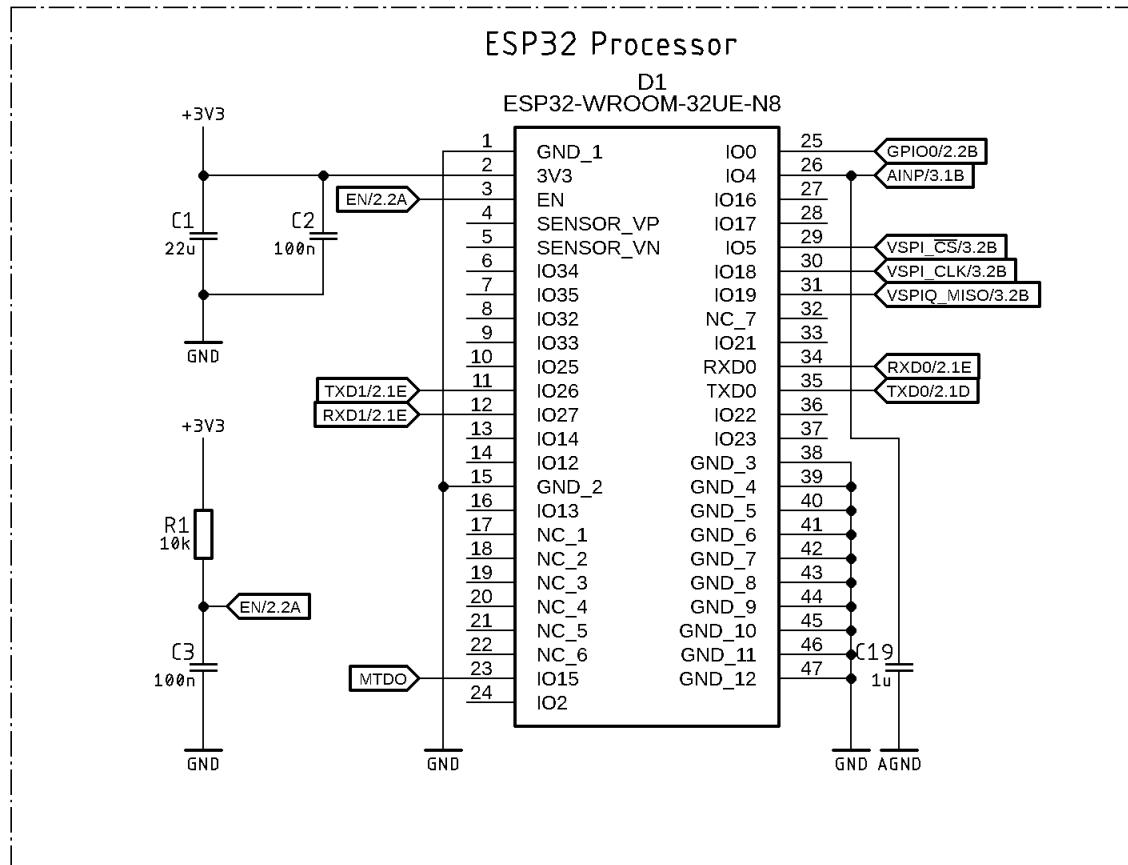
**ESP-WROOM-32** 32-Bit WROOM Modul

<b>U</b>	IPX Antennen Connector für externe Antennen
<b>E</b>	E-Serie
<b>N8</b>	8MB Flash Speicher
-	Kein Pseudostatisches RAM (PSRAM)

*Tabelle 5: ESP-WROOM Spezifikationen*

Da keine kabellosen Kommunikationssysteme wie WLAN oder Bluetooth benötigt werden, ist die Wahl auf ein kompaktes Modul mit IPX-Steckverbinde, anstelle einer großflächigen PIF-Antenne, gefallen.

#### 5.2.3.1 *Beschaltung des ESP-WROOM-32UE Chips*



*Abbildung 84: ESP32 Beschaltung*

Für den Betrieb des ESP32-WROOM Moduls, werden grundsätzlich nur zwei Entkopplungskondensatoren C1 und C2 benötigt. Sie müssen möglichst nah am Versorgungs-Kontakt platziert werden, wobei der HF-Entkopplungskondensator C2 näher als der Bulk-Kondensator C1 platziert werden sollte. Dies versichert, dass der ESP32 stabil versorgt wird, da ein Reset automatisch bei Spannungseinbrüchen ausgelöst werden würde.

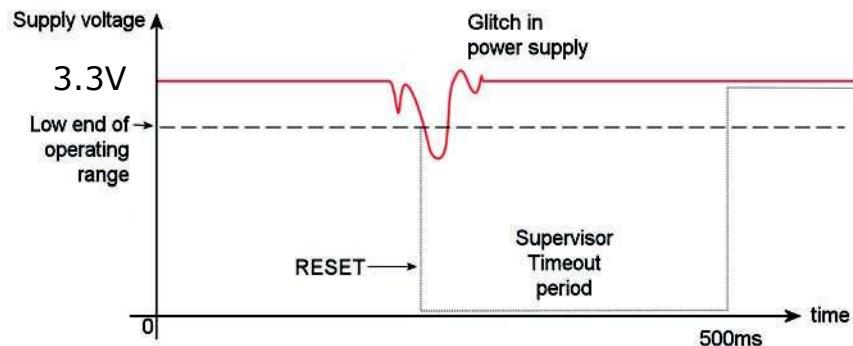


Abbildung 85: Reset bei Spannungseinbruch

Für den manuellen Reset Eingang wird ein Pull-Up-Widerstand (R1) in Kombination mit einem Entstör-Kondensator (C3) benötigt, welcher ungewollte Resets verhindert und zusätzlich die natürliche Prellung des Tasters unterdrückt. Die beiden Komponenten sollten möglichst nah am Reset-Pin platziert werden.

#### 5.2.3.1.1 Pin-Out

ESP32 GPIO	Label	Funktion
<b>GND_xx</b>	GND	Digitale Masse Pins, Heat-Sink vom Modul
<b>3V3</b>	+3V3	Versorgungsspannung
<b>EN</b>	EN	Enable / Reset Pin
<b>IO26</b>	TXD1	Transmit-Leitung des UART1 Treibers (MIDI TX)
<b>IO27</b>	RXD1	Receive-Leitung des UART1 Treibers (MIDI RX)
<b>IO0</b>	GPIO0	Boot-Mode Pin
<b>IO5</b>	VSPI_CS	Chipselect für den ADC
<b>IO18</b>	VSPI_CLK	Taktleitung des VSPI-Busses
<b>IO19</b>	VSPIQ_MISO	Master In Slave Out des VSPI-Busses
<b>RXD0</b>	RXD0	Receive-Leitung des UART0 Treibers (Debug/Programmierung)
<b>TXD0</b>	TXD0	Receive-Leitung des UART0 Treibers

## (Debug/Programmierung)

Tabelle 6: ESP32 Pin-Out

Um einen neuen kompilierten Code auf den ESP32 herunterzuladen, muss er sich während eines Resets im „Boot“ Modus befinden. Dieser Modus wird durch einen Taster, der, während dem Uploaden des Codes gedrückt werden muss, initialisiert.

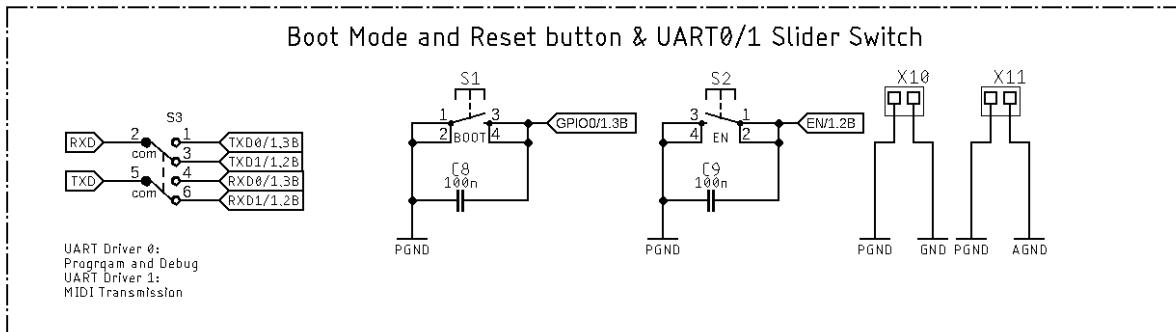


Abbildung 86: Taster und Schiebeschalter

## 5.2.3.2 Beschaltung der USB-Bridge

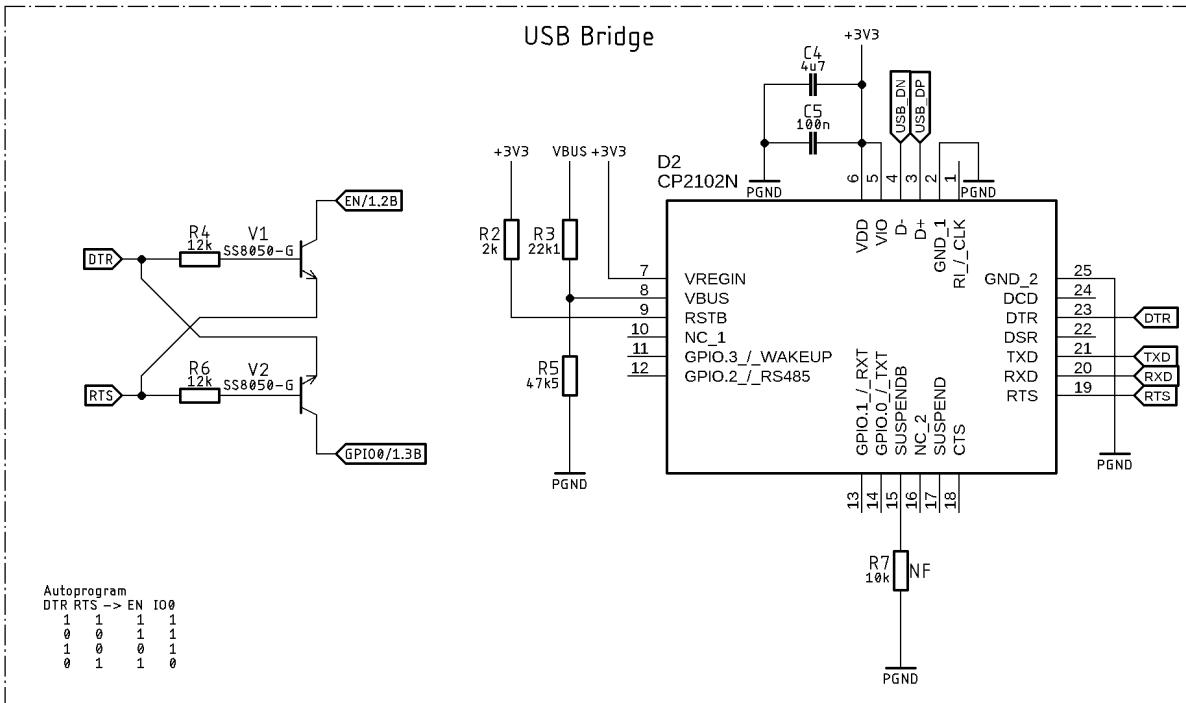


Abbildung 87: Beschaltung der USB-Bridge

Die Beschaltung der USB-Bridge wurde nach den Richtlinien im Datenblatt implementiert. Neben den Pull-Widerständen R2 und R7, wird sie mit einem Spannungsteiler, bestehend aus R3 und R5, beschalten. Dieser dient zur

Konfiguration eines integrierten Linearreglers, welcher aber im Bypass Modus ist. Das heißt, dass er nicht zur Versorgung externer Komponenten verwendet wird, da sein Versorgungsstrom maximal 100mA ist. Abzüglich des Stromes, den die Bridge selbst benötigt, ist der Ausgangsstrom des internen LVR zu niedrig, um als Hauptversorgung verwendet zu werden. Die besagte in Konfiguration wird im Datenblatt als „USB Self-Powered“-Konfiguration bezeichnet. Die beiden Transistoren dienen als elektrischer Boot- und Reset-Schalter, welcher von der USB-Bridge bei Bedarf angesteuert wird.

#### *5.2.3.3 Interner ADC mit DMA-Support*

Zur Erfassung der analogen Signale wird der im SoC integrierte Analog-Digital-Wandler verwendet. Die maximale Auflösung beträgt 12 Bit, wobei die Referenzspannung (1V1-3V3) in der Firmware konfiguriert werden kann. Der DMA-Controller schreibt die aus der AD-Konversion resultierenden Daten, in einen Direct Memory Access (DMA) Buffer. Ein DMA Buffer ermöglicht es der Firmware, die Daten kontinuierlich einzulesen. Diese Funktion ist wichtig, um die CPU zu entlasten, sodass trotz anderer Prozesse ein kontinuierlicher Datenstrom zur Verfügung steht und keine Lücken in den Audiodaten vorkommen.

#### *5.2.4 Layout der Platine*

Eine Zielsetzung bei dem Projekt ist es, sich von den Entwicklungsmodulen zu distanzieren und ein Gerät zu entwickeln, welches einem vermarktbaren Produkt ähnelt. Um dieses Ziel zu erfüllen, ist ein geeignetes Leiterplatten-design ausschlaggebend für die Funktionalität und einen kompakten Footprint des Gerätes. Bei dem Design der Leiterplatte wurden besonders auf die elektromagnetische Verträglichkeit geachtet.

##### *5.2.4.1 Versorgungsleitungen*

Die Versorgungsleiterbahnen sind leicht von den anderen Leiterbahnen zu unterscheiden, da sie im Vergleich viel dicker sind. Da über Versorgungsleitungen der meiste Strom fließt, ist es von Vorteil diese so breit wie möglich zu machen, um Verluste zu minimieren. Mit einer Breite von 32mil kann

eine solche Leiterbahn bis zu 3,8 Ampere (mit einem Temperaturanstieg um ca. 30°C) standhalten.

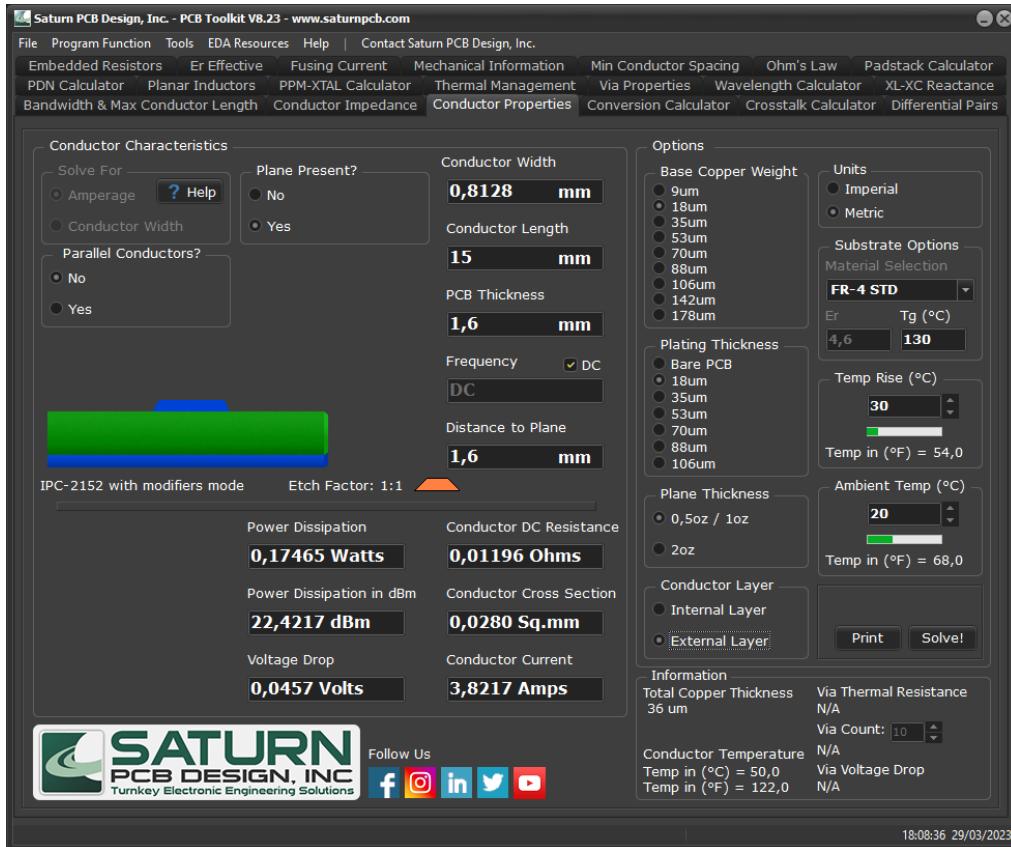


Abbildung 88: Berechnung des maximalen Stromes im PCB-Toolkit Rechner

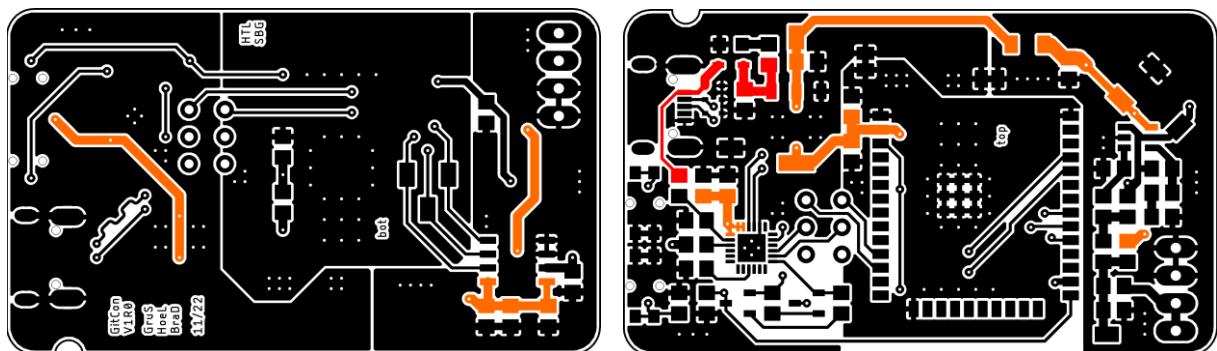


Abbildung 89: Versorgungsleitungen (5V rot, 3V3 orange)

Die Versorgung der USB-Bridge, des ESP32 und der Analog-Sektion sind Sternpunkt-förmig am LDO angeschlossen. Eine Besonderheit bei der Analog-Versorgung ist, dass sie zusätzlich mit einem passivem C-L-C Pi-Filter ausgestattet ist, um mittel- bis hochfrequentes Rauschen zu entfernen.



Abbildung 90: Komponenten des Pi-Filters

### 5.2.4.2 Massefläche

Die Masseanschlüsse der einzelnen Komponenten werden mit einer Massefläche verbunden. Die daraus resultierende induktionsarme Verbindung sorgt dafür, dass Stromrückflüsse einen nur sehr geringen Potentialunterschied verursachen.

#### 5.2.4.2.1 Sternpunktmasse

Treffen sich die einzelnen Masseflächen in nur einem Punkt, so spricht man von einer Sternpunktmasse. Eine Sternpunktmasse hat den Vorteil, dass Masseschleifen verhindert werden.

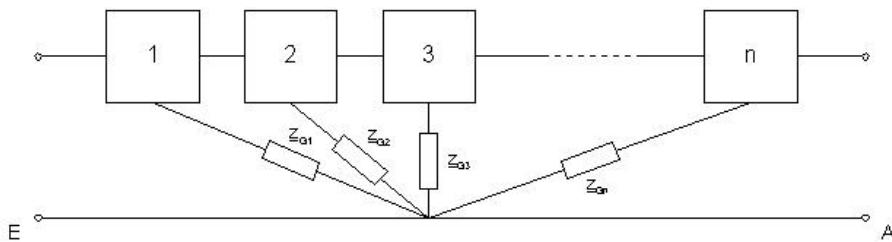


Abbildung 91: Visualisierung einer Sternpunktmasse

Eine Masseschleife entsteht, wenn sich mehrere Bauteile eine Impedanz (eine Leiterbahnenabschnitt) als Masse teilen. Durch den Stromrückfluss eines Bauteils auf diesem Abschnitt kann das Massepotential an einem anderen Bauteil angehoben werden. Man würde dann von einer galvanischen Kopplung sprechen.

In diesem Fall werden die Masse-Polygone möglichst nahe am LDO mit SMD-Überbrückungen verbunden. (siehe pink-markierte Felder in Abbildung 93)



Abbildung 92:  
SMD-Jumper

Auf der Platine werden die Komponenten auf drei verschiedene Arten klassifiziert: Digitale-, Analoge- und Leistungsbauteile.

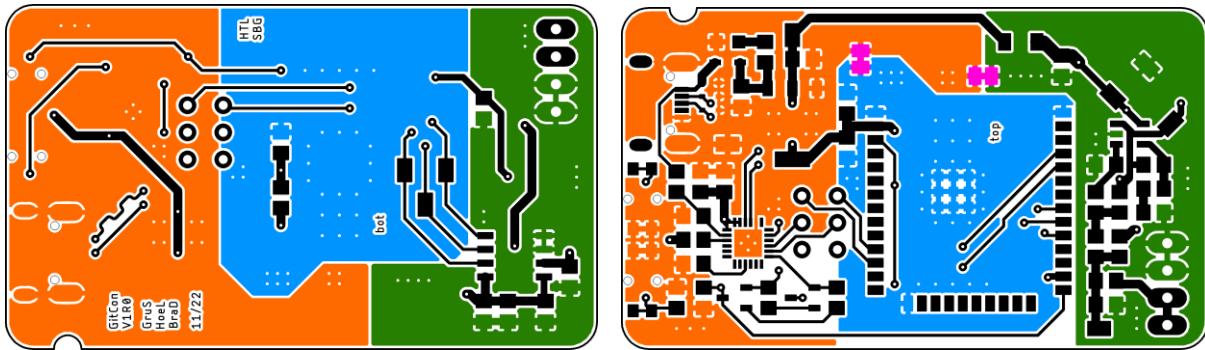


Abbildung 93: Bottom (links) und Top (rechts) Masse-Layer

#### P GND: Leistungsmasse (Orange)

Leistungsbauteile treiben oft hohe Ströme, welche ungewollte Differenzen auf dem Massepotential hervorrufen können. Isoliert man nun diese Masse von den signaltreibenden Bauteilen, ist nur das Leitungsbau teil von der galvanischen Kopplung betroffen.

#### GND: Digitalmasse (Blau)

Digitale Bauteile arbeiten oft mit hochfrequenten Signalen (hier z.B. der SPI-Bus). Wenn HF-Leitungen frei liegen, können sie auf nahen Leiterbahnen, durch das sich schnell ändernde elektrische Feld, Störungen verursachen (Crosstalk<sup>11</sup>). Diesen Effekt bezeichnet man als kapazitive Kopplung.

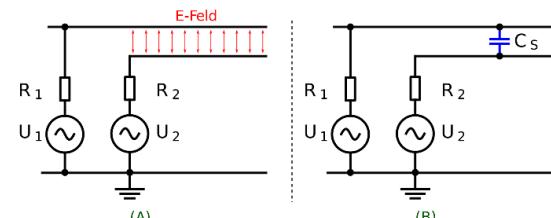


Abbildung 94: kapazitive Kopplung

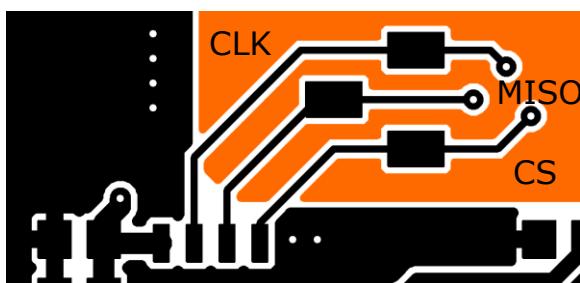


Abbildung 95: Massefläche zwischen HF-Leitungen

Um dieser Kopplung zwischen den Leiterbahnen entgegenzuwirken, muss zwischen besagten Leitungen eine Massefläche ausgeprägt sein. Da diese Störungen nun von der Masse absorbiert werden, muss die Fläche vom Rest isoliert werden. Dies wird

<sup>11</sup> Ungewollte Übertragung von Signalen auf eine andere Leitung durch Kopplungsarten

erzielt, indem man sie nahe an der Masse der Versorgungseinheit (LDO) anschließt.

#### AGND: Analogmasse (Grün)

Bei analogen Signalen ist oft das Rauschen ein kritischer Faktor (SNR). Beispielsweise haben die Audiosignale, welche von den Pick-Ups der E-Gitarre induziert werden, eine sehr geringe Amplitude. Benötigt nun eine andere Komponente für kurze Zeit einen hohen Strom oder generiert durch hohe Frequenzen ein Rauschen, so können Störungen auf der Masse auftreten, welches die Referenz für das Audiosignal verändert. Ein ADC kann dann das Signal nicht mehr korrekt erfassen und das Sample ist fehlerhaft. Eine störungsbefreite Masse unterstützt somit die Integrität des Signales und es kann in möglichst originaler Form weiterverarbeitet werden.

#### *5.2.4.2.2 Via-Stitching am Masse-Polygon*

Um den Stromrückflussweg von Komponenten so kurz wie möglich zu halten, wurden die Masseflächen an mehreren Stellen mit einer Vielzahl an Durch-kontaktierungen (Vias) „vernäht“. Das Massepotential wird dadurch über die gesamte Fläche konstant gehalten.



*Abbildung 96: Ground-Stitches auf der Platine*

#### *5.2.4.3 Platzierung der Entkopplungskondensatoren*

Entkopplungskondensatoren haben grundsätzlich zwei Aufgaben. Zum einen sollen sie bei Spannungseinbrüchen die Versorgung möglichst aufrecht erhalten, außerdem sollen hochfrequente Störsignale gefiltert werden.

Ein einziger Kondensator kann beide Aufgaben nicht ideal erfüllen, da mit niedrigeren Kapazitäten zwar die HF-Entkopplung besser ist, jedoch die Bulk-Kapazität fehlt, um einen Einbruch der Spannung zu kompensieren und umgekehrt. Deshalb ist es üblich, zwei Kondensatoren so nahe wie möglich an der Versorgung eines integrierten Schaltkreises (IC) auf der Platine zu montieren.

#### 5.2.4.4 *Audio*

Niederfrequente Audio-Signale, müssen von allen Stör- und Rauschquellen isoliert werden, sodass das Audio möglichst original aufgefasst werden kann. Auf der gegenüberliegenden Seite der Leiterplatte sind daher **keine** Leiterbahnen verlegt, sondern nur eine Massefläche.

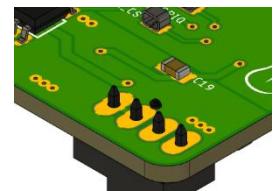


Abbildung 97:  
Masse unter Au-  
dio-Sektion

#### 5.2.4.5 *Differentielles Paar*

Die Datenleitungen eines USB sind ein differentielles Paar und müssen auf der Platine auch als solches geroutet werden. Datenübertragungen mit einem Differentiellen Paar unterscheiden sich von normalen Leitungen, da sie kein Bezugspotential (Masse) haben, sondern zwei Leitungen mit demselben Datensignal, wobei eines der beiden invertiert ist. Die Impedanz des Differentiellen Paars muss angepasst werden, da sonst die maximale Übertragungsrate eingeschränkt werden könnte. Die Leitungen sind angepasst, wenn beide Leitungen gleich lang geroutet sind und dieselbe Impedanz aufweisen. [21]

#### 5.2.4.6 *Abmessung und Kompakter Footprint*

Die Platine verfügt über eine Einkerbung, mit welcher sich die Platine mit einer M3 Schraube an ein Gehäuse befestigen lässt.

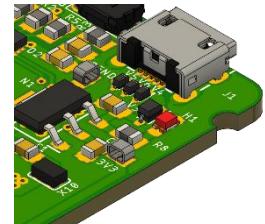


Abbildung 98: Einker-  
bung zur Befestigung

#### 5.2.4.7 *SMD-Testterminals*

Um Systemtests zu vereinfachen, wurden wichtige Signale mithilfe von Testterminals leicht für Tastköpfe von Oszilloskopen und Multimetern zugänglich gemacht. Insgesamt wurden auf der Platine

## 5.3 Firmware

### 5.3.1 Treiber Firmware Diagramm

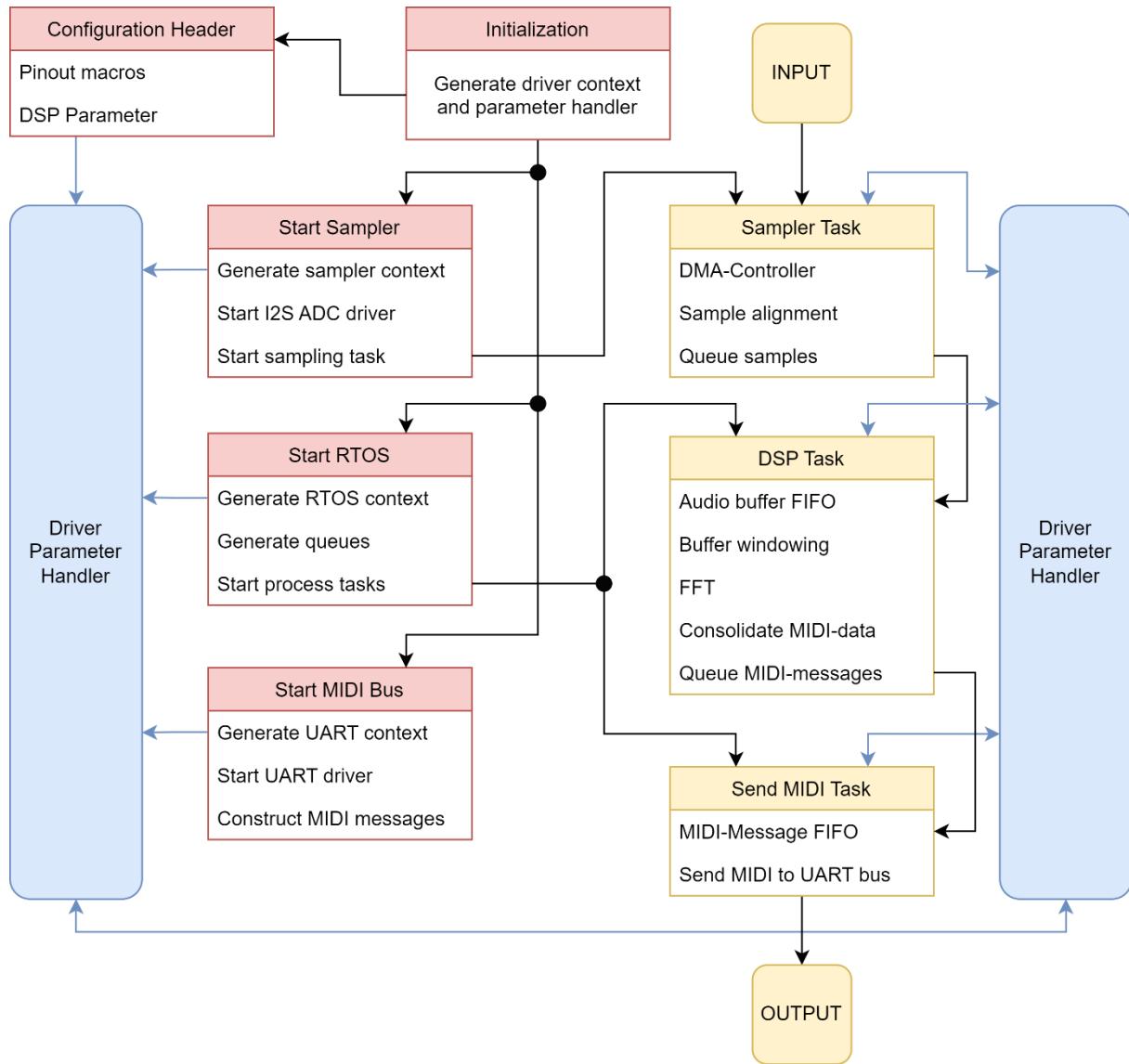


Abbildung 99: Firmware Struktogramm

Nach einem Firmware Reset erfolgt zunächst die Initialisierung der Datenstrukturen. Es werden die Treiber der Peripherien initialisiert und die Standardkonfiguration aus dem Config-Header geladen. Ein Sampler, der die Audiodaten vom Direct Memory Access Controller liest und den Audio Buffer bereitstellt, wird initialisiert und gestartet. Auch die Ressourcen für den Digital Signal Processing Algorithmus werden allokiert. Ebenfalls wird der Treiber für die MIDI-Schnittstelle gestartet und kann aus der DSP resultierende MIDI-Daten an die USB-Bridge senden.

Drei vom RTOS geregelte Tasks formen die Prozesskette. Mittels Queues (First In First Out, FIFO) werden die Daten in den nächsten Task übergeben. Von der zentralen Datenstruktur „Driver Parameter Handler“ kann von allen Tasks aus auf die Peripherien und FIFOs referenziert werden.

### 5.3.2 Doxygen

Der Code wurde mittels sogenannter ungarischer Notation versehen. Ungarische Notation kommt in der Firmware in Form von dokumentativen Kommentaren vor, welche anschließend von dem Programm **Doxygen** erkannt werden und daraus Referenzfiles generiert, welche hilfreich für die Dokumentation des Codes sind. Alle im Quellverzeichnis vorkommenden Dateien werden indiziert und es wird ein Inhaltsverzeichnis daraus erstellt. Weiters werden die verwendeten Datentypen aufgelistet und mit einer Beschreibung versehen, mit welchen der Code inline dokumentiert ist. In Summe ergibt sich daraus das Referenzhandbuch für die Firmware mit folgender Struktur:

1. Data Structure Index
  - Inhaltsverzeichnis für Typedefs und Datenstrukturen
2. File Index
  - Inhaltsverzeichnis für Header- und Quelldateien
3. Data Structure Documentation
  - Beschreibung der für Typedefs und Datenstrukturen
4. File Documentation
  - Beschreibung der Methoden und Algorithmen

Das Handbuch der Firmware dieses Projekts lässt sich im Anhang finden.

Kommentare ungarischer Notation unterscheiden sich durch ein zusätzliches Sonderzeichen von normalen Kommentaren:

	<b>Ungarische Notation</b>	<b>Standardkommentar</b>
<i>Multiline-Comment</i>	<code>/** ... */</code>	<code>/* ... */</code>
<i>Singleline-Comment</i>	<code>/// ...</code>	<code>// ...</code>
<i>Beispiel</i>	<pre>/**  * @file gitcon.h  * @author @s-grundner @Laurenz03  * @brief Gitcon Device Driver  * @version 0.1  * @date 2022-12-23  *  * @copyright Copyright (c) 2022  */</pre>	<pre>/*  * @file gitcon.h  * @author @s-grundner @Laurenz03  * @brief Gitcon Device Driver  * @version 0.1  * @date 2022-12-23  *  * @copyright Copyright (c) 2022  */</pre>

Tabelle 7: ungarische Notation im Vergleich zu herkömmlichen Kommentaren

## 5.4 Software

Um den Guitar-Converter am PC auslesen zu können, werden bestimmte Treiber benötigt. Im Folgenden werden die nötigen Konfigurationen der Software beschrieben, um den Gitcon betriebsbereit zu machen und ihn in der digital Audio Workstation Ableton Live verwenden zu können.

### 5.4.1 Virtueller MIDI-Port

Originale MIDI-Controller verfügen über zwei DIN 5-Pol Stecker, um die Parameter untereinander auszutauschen. Da am Mainboard eines PCs jedoch keine DIN-Stecker verbaut sind, werden die Daten über USB vermittelt. Damit die DAW nun MIDI-Geräte voneinander unterscheiden kann, gibt es softwarebasierte, virtuelle Ports.



Abbildung 100: MIDI-Port eines klassischen Controllers

Hierfür kommt die Software **loopMIDI** zum Einsatz. In ihr kann man Instanzen von virtuellen MIDI-Ports erzeugen, sodass die DAW weiß, welche MIDI-Signale tatsächlich vom dem Gitcon-Device stammen. Um einen Port zu erzeugen, gibt man den Namen in das Textfeld ein und klickt auf die Plus (+) Schaltfläche.

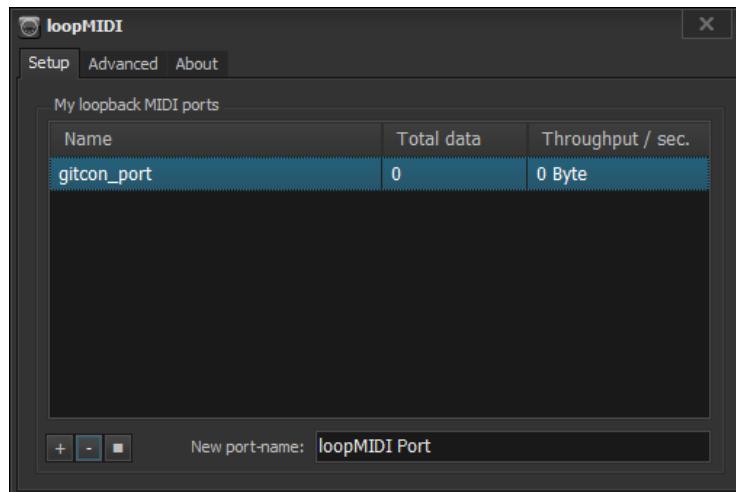


Abbildung 101: Loop MIDI GUI

## 5.4.2 MIDI Serial Bridge

Eine MIDI Serial Bridge wird benötigt, um erhaltene Bytes von dem COM-Port an einen MIDI-Port weiterzuleiten, welcher als Schnittstelle zur DAW dient.

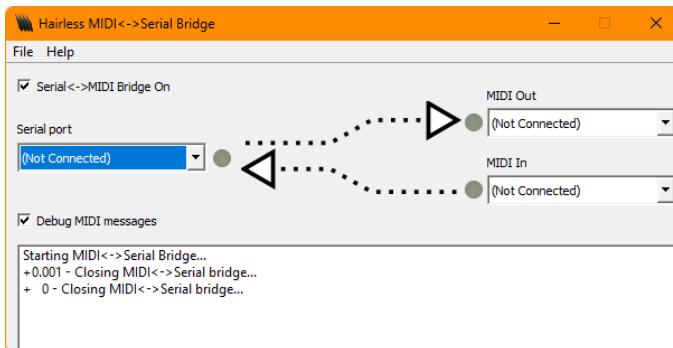


Abbildung 102: hairless-midiserial Benutzeroberfläche

Das hier verwendete Programm **hairless-midiserial**, ermöglicht es, einen Seriellen Port auszuwählen, von welchem die Daten, an ein MIDI-Port gesendet (MIDI-Out), beziehungsweise empfangen (MIDI In) werden können. Ein zuvor erzeugter virtueller MIDI-Port kann nun in der Benutzeroberfläche selektiert werden.

Das Programm ist Open-Source und der Code kann in der öffentlichen GitHub-Repository eingesehen werden.

- <https://GitHub.com/projectgus/hairless-midiserial>



Abbildung 103: MIDI-Port Dropdown-Menü

### 5.4.3 Ableton Live Setup

Mit dem Tastenkürzel **Ctrl + ,** werden die Voreinstellungen (Preferences) aufgerufen. Sofern zuvor ein virtueller MIDI-Port erzeugt worden ist, wird er von Ableton Live erkannt und kann im Reiter **MIDI** selektiert werden.

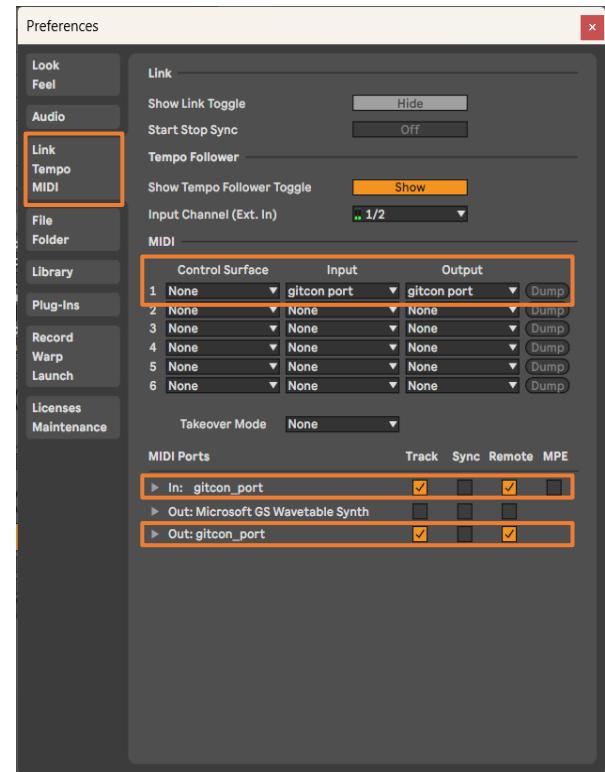


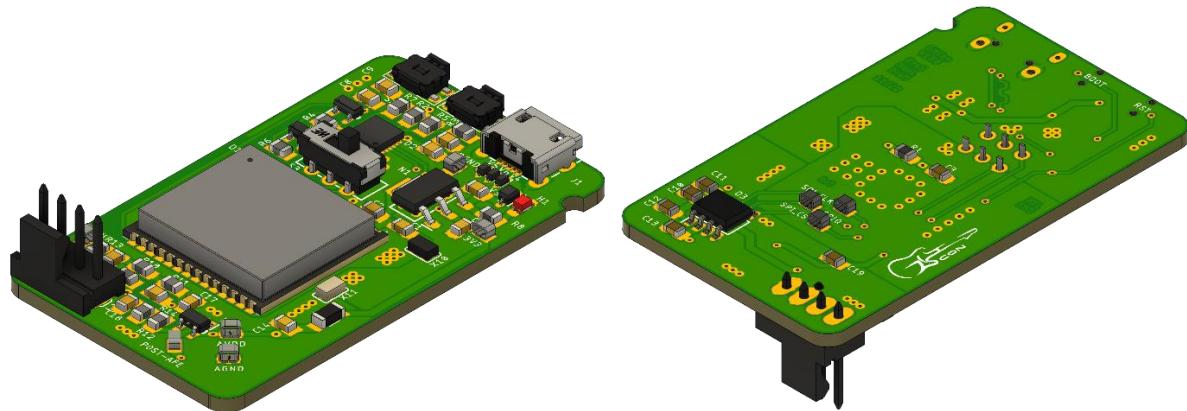
Abbildung 104: Live Preferences

Eingehende MIDI-Signale können nun von Ableton Live in einen MIDI-Kanal geroutet werden, in welchem dann die Signale zur Verfügung stehen. Das Gitcon Device schickt die MIDI-Noten auf den Kanal 1. In der MIDI-Spur muss also der entsprechende Kanal ausgewählt werden. Um nun die eintreffenden MIDI-Noten aufzunehmen, muss die MIDI-Spur aufnahmen bereit geschalten werden (Arming). Eine Spur ist für die Aufnahme scharfgestellt, wenn man die „Arm“-Schaltfläche anklickt, so dass sie rot aufleuchtet.



Abbildung 105: MIDI-Spur Konfiguration

## 5.5 CAD-Modelle



Um ein besseres Bild der Platine im Vorhinein zu haben, wurden für alle Komponenten ein 3D-Modell erzeugt und ein CAD-Modell der Platine in Fusion 360 kreiert. Das Modell war auch hilfreich festzustellen, ob jedes Bauteil im Layout genügend Spielraum hat, und es hardwaretechnisch keine Interferenzen gibt.

## 6 Fehlererfassung

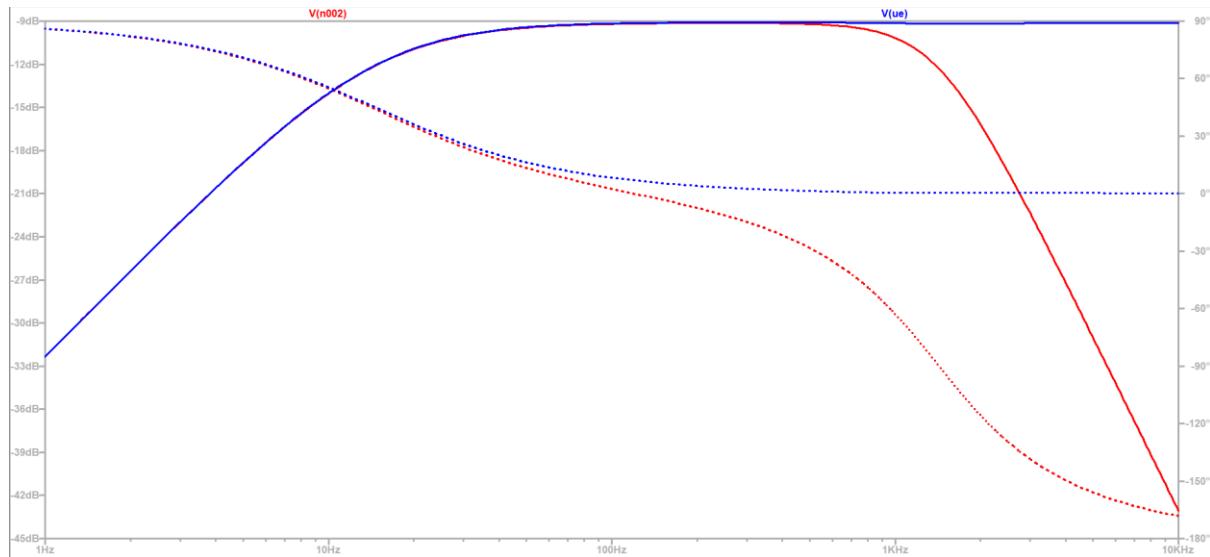
### 6.1 Platine

- Taster Footprint ist Falsch
- Teil des Doku-Layers wurde nicht auf den Silkscreen gedruckt.

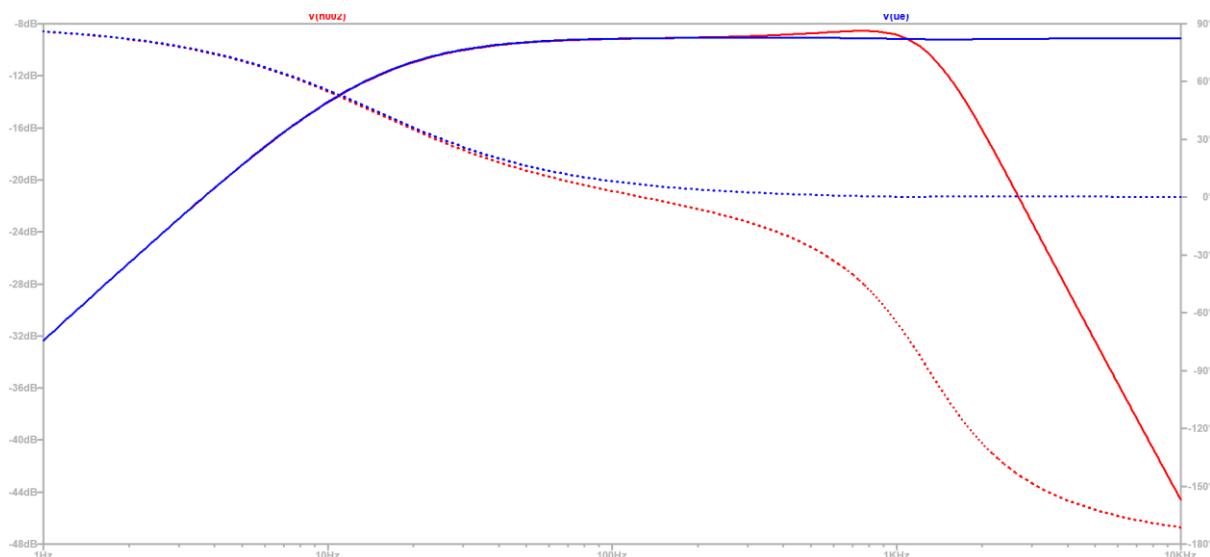
### 6.2 Bestückung

- 10nF statt 12nF bei Tiefpass verfügbar

Simulation mit 10nF (Ist)



Simulation mit 12nF (Soll)



Fazit: 10nF beeinflusst die Schaltung positiv, da die Resonanz an den Grenzfrequenzen (Güte- oder Q-Faktor) geringer ist.

### 6.3 ADC Channel 2 auf ADC Channel 1 überbrücken

Der I2S-Support der internen Analog-Digital-Wandler des ESP32 ist auf den ADC-Kanal 1 begrenzt. Da diese Beschränkung erst bei der Firmwareprogrammierung aufgefallen ist, wurde dieses Erratum auf der Platine mittels einer Überbrückung auf den korrekten Pin korrigiert.

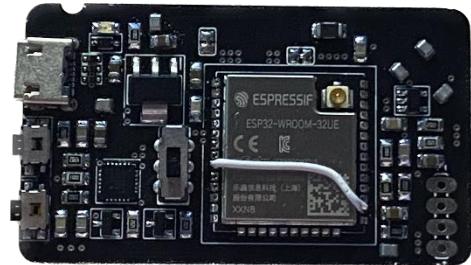


Abbildung 106: Weißes Kabel als Überbrückungsmedium der Pins

### 6.4 Fehlende Features

- Die Implementation einer Debug-LED wäre praktisch gewesen, um beliebige Parameter anzeigen zu lassen, zum Beispiel wann:
  - eine Transiente erkannt wurde;
  - eine Note übertragen wurde;
- Eine RX/TX-Package-LED an der UART-Bridge

## 7 Glossar

ADC .....	Analog-Digital-Converter
AFE .....	Analogue-Frontend
CAD .....	Computer Aided Design
CPU .....	Central Processing Unit
DAW .....	Digital Audio Workstation
DFT .....	Diskrete Fourier Transformation
DSP .....	Digital Signal Processing
DSV .....	Digitale Signalverarbeitung
EKG .....	Elektrokardiogramm
ESD .....	Electrostatic Discharge
ESP .....	Kürzel für Espressif Produkte
FFT .....	Fast Fourier Transform
FIFOs .....	First In First Out (Queue)
Gitcon .....	Guitar Converter
GND .....	Ground (Masse)
GPIO .....	General Purpose Input/Output
HF .....	Hoch Frequent
I2C .....	Inter Integrated Circuit
I2S .....	Inter Integrated Sound
IC .....	Integrated Circuit
IDF .....	IoT Development Framework
IoT .....	Internet of Things
LDO .....	Low Dropout Regulator
LED .....	Light Emitting Diode
MIDI .....	Musical Instrument Device Interface
MQTT .....	Message Queuing Telemetry Transport
OPV .....	Operationsverstärker
PC .....	Personal Computer
PCB .....	Printed Circuit Board
Pickup .....	Tonabnehmer
PIF .....	Planar Inverted F-Shaped
PIO .....	PlatformIO
PSRAM .....	Pseudostatisches RAM
RAM .....	Random Access Memory (Volatiler Speicher)
RTOS .....	Realtime Operating-System
SDK .....	Software Development Kit
SMD .....	Surface Mount Device
SNR .....	Signal to Noise Ratio
SoC .....	System on a Chip
TI .....	Texas Instruments (Firma)
TVS .....	Transient Voltage Suppression
UART .....	Universal Asynchronous Recieve and Transmit
USB .....	Universal Serial Bus

Via ..... Durchkontaktierung  
WLAN ..... Wireless Local Area Network

## 8 Abbildungsverzeichnis

Abbildung 1: Gitcon Logo .....	4
Abbildung 2: MIDI-Klavier (Novation Launchkey) .....	13
Abbildung 3: Konfiguration einer MIDI-Spur beim MIDI-Unitest .....	21
Abbildung 4: Starten der Aufnahme des MIDI-Unitests .....	21
Abbildung 5: Ergebnis einer Aufnahme des MIDI-Unitests .....	21
Abbildung 6: Konfiguration einer MIDI-Spur beim Live-Konversion Test.	22
Abbildung 7: Starten der Aufnahme des Live-Konversionstest .....	22
Abbildung 8: Ergebnis der Live-Konversion .....	23
Abbildung 9: Trello Liste mit zwei Karten .....	29
Abbildung 10: Copilot Logo .....	29
Abbildung 11: passiver LCR-Tiefpass [35] .....	30
Abbildung 12: aktiver Tiefpass [36] .....	31
Abbildung 13: OPV-Schaltsymbol .....	33
Abbildung 14: Operationsverstärker .....	33
Abbildung 15: Schaltbild OPV .....	34
Abbildung 16: Differenzenverstärkung OPV .....	35
Abbildung 17: OPV als Impedanzwandler .....	35
Abbildung 18: Invertierender OPV .....	36
Abbildung 19: Nicht-invertierender OPV .....	37
Abbildung 20: Sallen-Key Tiefpass (links) & Hochpass (rechts) .....	40
Abbildung 21: Filter erster Ordnung .....	40
Abbildung 22: Filter zweiter Ordnung .....	40
Abbildung 23: Flankensteilheit der Ordnungen [24] .....	41
Abbildung 24: Butterworth Frequenzverhalten [33] .....	42
Abbildung 25: Frequenzverhalten der Filtercharakteristiken [32] .....	43
Abbildung 26: Toleranzschema [31] .....	43
Abbildung 27: Schaltbild Filter zweiter Ordnung .....	44
Abbildung 28: Schaltbild Sallen-Key zweiter Ordnung .....	46
Abbildung 29: Dämpfungskurven .....	48
Abbildung 30: Aufbau einer E-Gitarre .....	49
Abbildung 31: Veranschaulichung Fourier Transformation .....	50

Abbildung 32: Aufspaltung 8-Punkte-DFT.....	57
Abbildung 33: weitere Zerlegung 8-Punkte-DFT .....	58
Abbildung 34: Butterfly-Graph.....	58
Abbildung 35: vereinfachter Butterfly .....	59
Abbildung 36: Dataflow Firmware .....	60
Abbildung 37: Audacity Interface .....	64
Abbildung 38: Auswahl eines Signalabschnitts .....	65
Abbildung 39: Audacity Tools .....	65
Abbildung 40: Sample-Datenexport Fenster .....	65
Abbildung 41: Dataflow Python Tool .....	67
Abbildung 42: Audacity Analyse Tools .....	69
Abbildung 43: Frequenzanalyse Audacity .....	70
Abbildung 44: Frequenzanalyse Audacity .....	70
Abbildung 45: Matlab FFT .....	71
Abbildung 46: ESP32 Prozessor (Mitte), PIF WLAN-Antenne (links) .....	73
Abbildung 47: ESP-WROOM mit IPX-Connector.....	73
Abbildung 48: Download des Flash Tools .....	73
Abbildung 49: File Folder des heruntergeladenen Ordner .....	73
Abbildung 50: ESP-AT Versionen.....	74
Abbildung 51: flasher_args.json - flash_files .....	74
Abbildung 52: Tool Interface mit eingetragenen Binaries .....	75
Abbildung 53: flasher_args.json - flash_settings .....	75
Abbildung 54: SPI Flash Config in dem GUI .....	75
Abbildung 55: PlatformIO Installation .....	76
Abbildung 56: PlatformIO Home .....	76
Abbildung 57: PlatformIO Project Wizard .....	77
Abbildung 58: Konfiguriertes Projekt .....	77
Abbildung 59: PlatformIO Projekt Struktur .....	78
Abbildung 60: PlatformIO Library Reiter .....	78
Abbildung 61: Bibliothek Untermenü .....	79
Abbildung 62: Einbinden von Dependencies .....	79
Abbildung 63: platformio.ini .....	79

Abbildung 64: Project-Tasks Panel .....	80
Abbildung 65: Statusleiste .....	80
Abbildung 66: UART-Verbindung.....	82
Abbildung 67: UART-Datenpaket.....	82
Abbildung 68: Ableton Live Effekt EQ Eight (Filter).....	83
Abbildung 69: MIDI-Word .....	84
Abbildung 70: Vier Verschieden MIDI-Spuren mit dem gleichen Controller als Input.....	85
Abbildung 71: Novation Launchkey Mini.....	85
Abbildung 72: Funktionsblockdiagramm des Gitcon .....	87
Abbildung 73: Legende der Arbeitsunterteilung.....	88
Abbildung 74: Spannungsregler und USB-Connector im Schaltplan .....	89
Abbildung 75: ESD-Warnhinweis.....	90
Abbildung 76: ESD-Schutz in Schaltungen .....	90
Abbildung 77: Symbol der Bipolaren TVS-Diode.....	91
Abbildung 78: SC7538-08UTG Block Diagramm .....	91
Abbildung 79: LPF-Schaltung.....	93
Abbildung 80: Filter Prototyp mit TL074.....	94
Abbildung 81: Spannungsverhältnis Ua/Ue.....	96
Abbildung 82: Simulation des Verhältnis Ua/Ue .....	96
Abbildung 83: Messung des Filters .....	97
Abbildung 84: ESP32 Beschaltung .....	99
Abbildung 85: Reset bei Spannungseinbruch .....	100
Abbildung 86: Taster und Schiebeschalter.....	101
Abbildung 87: Beschaltung der USB-Bridge .....	101
Abbildung 88: Berechnung des maximalen Stromes im PCB-Toolkit Rechner .....	103
Abbildung 89: Versorgungsleitungen (5V rot, 3V3 orange) .....	103
Abbildung 90: Komponenten des Pi-Filters .....	103
Abbildung 91: Visualisierung einer Sternpunktmasse .....	104
Abbildung 92: SMD-Jumper.....	104
Abbildung 93: Bottom (links) und Top (rechts) Masse-Layer.....	105

Abbildung 94: kapazitive Kopplung .....	105
Abbildung 95: Massefläche zwischen HF-Leitungen.....	105
Abbildung 96: Ground-Stitches auf der Platine.....	106
Abbildung 97: Masse unter Audio-Sektion .....	107
Abbildung 98: Einkerbung zur Befestigung .....	107
Abbildung 99: Firmware Struktogramm .....	108
Abbildung 100: MIDI-Port eines klassischen Controllers .....	111
Abbildung 101: Loop MIDI GUI .....	111
Abbildung 102: hairless-midiserial Benutzeroberfläche .....	112
Abbildung 103: MIDI-Port Dropdown-Menü .....	112
Abbildung 104: Live Preferences .....	113
Abbildung 105: MIDI-Spur Konfiguration .....	113
Abbildung 106: Weißes Kabel als Überbrückungsmedium der Pins.....	116
 Codesegment 1: Macros .....	61
Codesegment 2: Includes .....	61
Codesegment 3: Array Initialisierung .....	61
Codesegment 4: Deklaration von max .....	61
Codesegment 5: Ausprogrammierung „getMaxMag()“.....	62
Codesegment 6: Start "main-loop".....	62
Codesegment 7: FFT-Ausführung .....	62
Codesegment 8: Verarbeitung der FFT-Daten .....	62
Codesegment 9: Ausgabe .....	63
Codesegment 10: FFT-Zerstörung.....	64
Codesegment 11: Import des "shutil" Moduls .....	67
Codesegment 12: Variablen Initialisierung .....	67
Codesegment 13: Datei auswerten.....	68
Codesegment 14: Löschung vorhandener Ausgabedatei.....	68
Codesegment 15: Ausgabe der Datei.....	68
Codesegment 16: Kopieren der Datei .....	69

Datenauszug 1: Ausgabe Audacity ..... 66

Datenauszug 2: Output des Testprogramms..... 69

## 9 Literaturverzeichnis

- [ „ElectronicsTutorials,“ [Online]. Available: <https://www.electronics-tutorials.ws/de/filtern/aktiver-tiefpassfilter.html>.  
]
- [ S. Tietze, Halbleiter-Schaltungstechnik, Springer, 2002.  
2  
]  
  
[ H. Zumbahlen, „Basic Linear Design,“ Newens/Elsevier, 2008.  
3  
]  
  
[ „ElectronicsTutorials,“ [Online]. Available: <https://www.electronics-tutorials.ws/de/filtern/filter-zweiter-ordnung.html>.  
]  
  
[ „Elektroniktutor Analogverstärker,“ [Online]. Available:  
5 <https://www.elektroniktutor.de/analogverstaerker/aktivflt.html>.  
]  
  
[ P. Mahler, „Delamar,“ [Online]. Available:  
6 <https://www.delamar.de/gitarre/e-gitarre-aufbau-schaubild-39598/>.  
] [Zugriff am 12 03 2023].  
  
[ NTi Audio AG, „NTI Audio,“ [Online]. Available: <https://www.nti-audio.com/de/service/wissen/fast-fourier-transformation-fft>. [Zugriff am  
] 22 02 2023].  
  
[ D. C. v. Grünigen, Digitale Signalverarbeitung, Carl Hanser Verlag,  
8 2008.  
]

[ Inspired Acoustics, „MIDI note numbers and center frequencies,“ Inspired  
9 Acoustics, [Online]. Available:  
] [https://www.inspiredacoustics.com/en/MIDI\\_note\\_numbers\\_and\\_center\\_frequencies](https://www.inspiredacoustics.com/en/MIDI_note_numbers_and_center_frequencies).

[ H. S. Domareski, „DRY, KISS & YAGNI Principles,“ Medium, 14 06 2020.  
1 [Online]. Available: <https://henriquesd.medium.com/dry-kiss-yagni-principles-1ce09d9c601f>.

]

[ Espressif Systems (Shanghai) Co., Ltd,  
1 „<https://www.espressif.com/en/support/documents/technical-documents>,“ 13 02 2023. [Online]. Available:  
] [https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e\\_esp32-wroom-32ue\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf).

[ Wdwd, „ESP32,“ Wikimedia Foundation Inc, 12 03 2023. [Online].  
1 Available: <https://de.wikipedia.org/wiki/ESP32>.

2

]

[ Espressif Systems (Shanghai) Co., Ltd., „ESP-AT User Guide,“ Espressif  
1 Systems (Shanghai) Co., Ltd., 2023. [Online]. Available:  
3 [https://docs.espressif.com/projects/esp-at/en/latest/esp32/Get Started/Downloading\\_guide.html](https://docs.espressif.com/projects/esp-at/en/latest/esp32/Get Started/Downloading_guide.html).

[ „FreeRTOS,“ Amazon Web Services, Inc., [Online]. Available:  
1 <https://www.freertos.org/>.

4

]

[ „UART verstehen,“ Rohde & Schwarz Österreich GesmbH, 2023. [Online].  
1 Available: <https://www.rohde-schwarz.com/at/produkte/messtechnik/essentials-test-equipment/digital-oscilloscopes/uart->

5 verstehen\_254524.html#:~:text=UART%20steht%20f%C3%BCr%20U  
] niversal%20Asynchronous,zu%20senden%20und%20zu%20empfangen

..

[ E. Peňa, „A Hardware Communication Protocol Understanding Universal  
1 Asynchronous Receiver/Transmitter,“ Analog Devices, Inc., 04 12 2020.  
6 [Online]. Available: <https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html>.

[ ZeM College GbR, „ZeM College MIDI Kompendium,“ ZeM College GbR,  
1 2008. [Online]. Available: <https://www.zem-college.de/indexf.html>.

7

]

[ M. Association, „Expanded MIDI 1.0 Messages List (Status Bytes),“ MIDI  
1 Association, 2023. [Online]. Available:  
8 <https://www.midi.org/specifications-old/item/table-2-expanded-messages-list-status-bytes>.

[ „Pickup (music technology),“ Wikimedia Foundation, Inc., 02 03 2023.  
1 [Online]. Available:

9 [https://en.wikipedia.org/wiki/Pickup\\_\(music\\_technology\)](https://en.wikipedia.org/wiki/Pickup_(music_technology)).

]

[ Espressif Systems (Shanghai) Co., Ltd., „Analog to Digital Converter  
2 (ADC) Calibration Driver,“ Espressif Systems (Shanghai) Co., Ltd., 2023.  
0 [Online]. Available: [https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/adc\\_calibration.html](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/adc_calibration.html).

[ Z. Peterson, „What Are Differential Pairs and Differential Signals?,“ Altium  
2 Ltd., 21 03 2023. [Online]. Available:  
1 <https://resources.altium.com/p/what-are-differential-pairs-and-differential-signals>.

[ R. Shashikanth, „11 Best High-Speed PCB Routing Practices,” Sierra Circuits, 29 09 2020. [Online]. Available: <https://www.protoexpress.com/blog/best-high-speed-pcb-routing-practices/>.

[ The Sierra Circuits Team, „How to Handle Current Return Path for Better Signal Integrity,” Sierra Circuits, 08 02 2021. [Online]. Available: <https://www.protoexpress.com/blog/current-return-path-signal-integrity/>.

[ S. V, „How Via Stitching Facilitates High-Current PCB Designs,” Sierra Circuits, 23 03 2023. [Online]. Available: <https://www.protoexpress.com/blog/how-via-stitching-facilitates-high-current-pcb-designs/>.

[ Espressif Systems (Shanghai) Co., Ltd, „API Reference - ESP32,” Espressif Systems (Shanghai) Co., Ltd, 2016. [Online]. Available: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/index.html>.

[ „ElectronicsTutorials,” [Online]. Available: <https://www.electronics-tutorials.ws/filter/sallen-key-filter.html>.  
6  
]

[ „ScienceDirect,” [Online]. Available: <https://www.sciencedirect.com/topics/engineering/sallen-key>.  
7  
]

[ M. Hughes, „Ableton-Referenzhandbuch Version 11,” Ableton AG, 2021.  
2 [Online]. Available: <https://www.ableton.com/de/manual/credits/>.  
8  
]

[ M. Hughes, „Neue Clips aufnehmen,“ Ableton AG, 2021. [Online].  
2 Available: <https://www.ableton.com/de/manual/recording-new-clips/>.

9

]

[ Espressif Systems (Shanghai) Co., Ltd., „Get Started,“ Espressif Systems  
3 (Shanghai) Co., Ltd., 2023. [Online]. Available:  
0 <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get->  
] started/index.html.

[ „spsc.tugraz.at,“ [Online]. Available:  
3 data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAASQAAACtCAM  
1 AAAAu7/J6AAAB8IBMVEX///d3d3g4ODm5ubV1dUAAADb29v19fvqamrt  
] 7e20tLS6urrAwMDS0tJvb2/KysqZmZmurq4QEBCKioo8PDzw8PD6//9GRk  
b//r6+vorKyumpqaenp7///fp6emDg4N2dnZOTk4gICD//+xYWFj/+9+f  
m8Pt//38OmRkZH88N3iw5sAV.

[ „blog.bliley.com,“ [Online]. Available: [https://blog.bliley.com/fs/hubfs/filter\\_post/filter-response-comparison.png](https://blog.bliley.com/fs/hubfs/filter_post/filter-response-comparison.png).  
2 comparison.png?width=640&name=filter-response-comparison.png.  
]

[ „www.electronics-tutorials.ws,“ [Online]. Available:  
3 <https://www.electronics-tutorials.ws/wp-content/uploads/2018/05/filter-fil57.gif>.  
]

[ „www.electronics-tutorials.ws,“ [Online]. Available:  
3 <https://www.electronics-tutorials.ws/wp-content/uploads/2013/08/opamp78.gif?fit=333%2C224&fit=333,226>.  
]

[ „www.mikrocontroller.net,“ [Online]. Available:  
3 <https://www.mikrocontroller.net/attachment/305033/LCR-Tiefpass.png>.

5

]

[ „www.mikrocontroller.net,“ [Online]. Available:  
3 [https://www.mikrocontroller.net/attachment/305031/aktiver\\_tiefpass.p](https://www.mikrocontroller.net/attachment/305031/aktiver_tiefpass.p)  
6 ng.

]

[ A. Wegerle, „Landr Blog,“ [Online]. Available:  
3 <https://blog.landr.com/de/ist-midi-eine-einfuehrung-das->  
7 einflussreichste-tool-das-die-musik-je-gesehen-hat/. [Zugriff am 31 03  
] 2023].

## 10 Anhang

10.1 PCB-Fertigungsunterlagen

10.2 Firmware Referenz Handbuch