Hochschule der Medien

Fakultät Electronic Media

Bachelorarbeit zum Thema:

Entwicklung eines konfigurierbaren Audio Effektgeräts

Zur Erlangung des Grades:

Bachelor of Engineering

Vorgelegt von: Salih Cekici
E-Mail: sc061@hdm-stuttgart.de

Matrikelnummer: 38332
Studiengang: Audiovisuelle Medien

Abgabedatum: 18.10.2023

Erstbetreuer: Prof. Dr. Frank Melchior
Zweitbetreuer: Prof. Oliver Curdt

Ehrenwörtliche Erklärung

Hiermit versichere ich, Salih Cekici, ehrenwörtlich, dass ich die vorliegende Bachelorarbeit mit dem Titel: „Entwicklung eines konfigurierbaren Audio Effektgeräts“ selbstständig und ohne fremde Hilfe verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen wurden, sind in jedem Fall unter Angabe der Quelle kenntlich gemacht. Ebenso sind alle Stellen, die mit Hilfe eines KI-basierten Schreibwerkzeugs erstellt oder überarbeitet wurden, kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden.

Ich habe die Bedeutung der ehrenwörtlichen Versicherung und die prüfungsrechtlichen Folgen (§ 24 Abs. 2 Bachelor-SPO, § 23 Abs. 2 Master-SPO (Vollzeit)) einer unrichtigen oderunvollständigen ehrenwörtlichen Versicherung zur Kenntnis genommen.

Stuttgart, 18.10.2023 _____

Ort, Datum Salih Cekici

Kurzfassung

Die vorliegende Arbeit befasst sich mit der Entwicklung eines Hardware Audio Effektgeräts. Aufgabe dieses Gerätes ist es, selbst entwickelte Audio Effekte bei geringstem Aufwand bei einem haptischen Gerät auszuprobieren. Basierend auf einem Raspberry Pi Einplatinencomputer und selbst gestalteten Platinen wurde ein funktionaler Prototyp erstellt. Zusätzlich wurden bei dieser Arbeit Recherchen über gängige Effekt-Algorithmen, wie Echos und Filter, gemacht. Anhand dieser Algorithmen wurde aus einer Kombination von Echo und Filter bestehendes Effekt entwickelt. Dieser Effekt wurde mit dem entstandenen Gerät ausprobiert. Schließlich wurden Tests über die Funktionalitäten des Geräts durchgeführt. Verwirklicht wurde das ganze Projekt größtenteils der Verwendung von der Software Max und dessen Software Erweiterung RNBO.

Abstract

This thesis deals with the development of a hardware audio effect device. The goal of this device is to try out self-developed audio effects with the least amount of effort using a haptic device. Based on a Raspberry Pi single board computer and self-designed printed circuit boards, a functional prototype was created. In addition, research on common effect algorithms, such as echoes and filters, was done in this work. Based on these algorithms, a custom effect was built and this was tested with the created device. Finally, tests were performed on the functionalities of the device. The whole project was realized mostly using the software Max and its software extension RNBO.

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

A

ADC	Analog-to-Digital Converter
AUI	Audiointerface
ASDT	Audio System Development Tool

C

CPU	Central Processing Unit

D

DAC	Digital-to-Analog Converter
DAW	Digital Audio Workstation
DSP	Digital Signal Processing

E

EDA	Electronic Design Automation
EF	Envelope Follower
EG	Effektgerät

F

FDM	Fused Deposition Modeling
FIR	Finite Impulse Response

G

GPIO	General Purpose Input/Output
GUI	Graphical User Interface

H

HAT	Hardware Attached on Top

I

IC	Integrated Circuit
IIR	Infinite Impulse Response

K

kΩ	kilo Ohm

L

LFO	Low Frequency Oscillator

M

Max	MAX/MSP

O

OSC	Open Sound Control

P

PCB	Printed Circuit Board
Pd	pure data

R

RPI	Raspberry Pi

S

SSH	Secure Shell
SPI	Serial Peripheral Interface

T

THT	Through Hole Technology
TPF	Tiefpass Filter

U

UI	User Interface

Formelverzeichnis

Quelltextverzeichnis

1. Einführung

1.1. Motivation

Eine Vielzahl von Effektgeräten (kurz EGs) wird seit den 1960ern von kommerziellen Herstellern zum Kauf angeboten. Einige typische Beispiele für solche EGs sind Delays, Reverbs, Distortion Geräte uvm.. Diese EGs dienen meistens für einen bestimmten Zweck und sind in der Funktion nicht veränderbar.

Das Ziel der vorliegenden Arbeit ist es, ein vielseitig programmierbares EG zu entwerfen und dieses prototypisch umzusetzen. Um dies zu verwirklichen, werden die Forschungsfragen gestellt: Wie kann ein EG in 2023 entwickelt werden?

Das Ziel war es, am Ende dieser Arbeit ein funktionierendes EG zu erzeugen. Dies wird mit der vorliegenden Arbeit dokumentiert. Dieses Gerät soll sich je nach verschiedenen Anwendungsanforderungen umprogrammieren lassen. Zudem soll es möglich sein, dieses Gerät über das Heimnetzwerk kabellos zu programmieren.

1.2. Übersicht

Im zweiten Kapitel werden die benötigten theoretischen Grundlagen für die Verständlichkeit der Arbeit ermittelt. Es wird erklärt, wie ein digitales Audiosignal aufgebaut ist. Im weiteren Kapitels wird zwischen verschiedenen Arten der Audiosignalverarbeitung unterschieden. Eine gängige Methode, die Prozessabläufe zu erklären, sind Flussdiagramme. Wie Flussdiagramme in der digitalen Signalverarbeitung für Audioeffekte angewendet werden, wird ebenfalls im zweiten Kapitel behandelt. Schließlich wird über Artefakte, die im Bereich der digitalen Audiotechnik auftreten können, gewarnt und deren Ursachen verdeutlicht.

Die softwareseitige Entwicklung wurde in Max durchgeführt. Der haptische Teil des Endproduktes basiert auf einem Raspberry Pi (RPI) Einplatinencomputer. Die Kommunikation zwischen dem RPI und der Programmierumgebung Max wird durch die Softwareerweiterung RNBO ermöglicht. Das dritte Kapitel stellt die verwendeten Entwicklungsumgebungen vor. Dabei werden die Unterschiede zwischen Max/MSP und RNBO im Detail behandelt und die neuen Möglichkeiten der Softwareerweiterung RNBO

beschrieben. Die Informationen stammen größtenteils aus der umfangreichen Web-Dokumentation des Software Entwicklers Cycling ’74 (o.D.).

Um die Möglichkeiten des entstandenen EGs zu demonstrieren, wird ein Audioeffekt in RNBO entwickelt. Für die Entwicklung des eigenen Effekts wurde eine Recherche über bereits existierende und häufig verwendete Audioeffekt-Algorithmen gemacht. In Kapitel vier wurden einige dieser Effekt-Algorithmen erklärt. Diese Algorithmen basieren auf Büchern von Zölzer (2011) und Cipriani & Giri (2020).

Die in der Theorie beschriebenen Effekte sollen auf dem EG programmiert werden. Die Umsetzung der Algorithmen findet in der grafischen Programmierumgebung Max und seiner Software Erweiterung RNBO statt. Im ersten Teil des fünften Kapitels wird jedes Teil des entwickelten RNBO Patches Schritt für Schritt unter die Lupe genommen. Zudem wird im gleichen Abschnitt ein Überblick über die Zusammenarbeit zwischen dem RPI und RNBO gegeben.

Eines der Hauptziele dieser Arbeit besteht darin, ein funktionsfähiges haptisches Audio-EG zu entwickeln. Im zweiten Teil des fünften Kapitels wird detailliert erklärt, wie ein EG basierend auf einem RPI entworfen und entwickelt wurde. Dabei werden Themen wie die Herausforderungen bei der Entwicklung und Designentscheidungen behandelt. Das Endprodukt sollte zudem haptische Bedienelemente besitzen. Um diese Elemente beim Anbringen an das Gehäuse des Geräts zu vereinfachen, wurden zwei Erweiterungsplatinen für den RPI konstruiert. Für diese Erweiterungen wurden zwei Platinen (auch Printed Circuit Boards, PCB) entworfen und angefertigt. Im ersten Teil des fünften Kapitels wird über die Entwicklung dieser Platinen und des allgemeinen haptischen Teil des EG detailliert berichtet.

Schließlich werden in Kapitel sechs die entwickelten Effekte getestet. Hierbei werden die Effekte auf die Latenz und Funktionsfähigkeiten sowie auf die Abweichungen der Werte getestet. Basierend auf den Ergebnissen werden in dem letzten Kapitel die Schlussfolgerungen aus der gesamten Arbeit gezogen. Folglich werden die Erfahrungen zusammengefasst, die im Laufe dieser Arbeit über die relativ neue Entwicklungsumgebung RNBO gesammelt wurden.

2. Theoretische Grundlagen von Digital Audio

2.1. Digitales Audiosignal

Für das Bearbeiten von Audios im digitalen Umfeld muss zunächst die Funktionsweise und der Aufbau des digitalen Audiosignals verstanden werden. Um akustische Signale speichern und bearbeiten zu können, wird das analoge Audiosignal in digitale Zahlenwerte umgewandelt. Dies erfolgt durch einen ADC (Analog-to-digital Converter). Das analoge Audiosignal wird in bestimmten Zeitintervallen ausgelesen und der Amplitudenwert des Signals wird auf einen Zahlenwert quantisiert. Dieses kontinuierliche Auslesen und Umwandeln wird als Abtasten bezeichnet (Zölzer, 2011).

Der Bereich der Zahlenwerte wird durch die Bittiefe der Abtastung bestimmt. Bei einer Abtastung mit 16-Bit-Bittiefe betragen die Zahlenwerte zwischen 0 und 65536. Die Abtastrate bestimmt, wie oft die Werte ausgelesen werden (Steppat, 2014). Es ist möglich, diese Zahlenwerte zu speichern, zu verarbeiten oder unverändert wiederzugeben. Die digitalen Zahlenwerte werden wiederum durch einen DAC (Digital-to-analog Converter) in ein analoges Signal umgewandelt. Dadurch ist, je nach Kontext, das Abhören der veränderten/unveränderten Signale möglich.

Für die Analog-Digital-Umwandlung und die Digital-Analog-Umwandlung wird in den meisten Fällen ein Audiointerface (AUI) genutzt. AUIs sind ein wesentlicher Bestandteil der digitalen Audioverarbeitung, denn Sie übernehmen die Aufgabe für die A/D und die D/A Wandlung. Der Central Processing Unit (CPU) spart dadurch wichtige Ressourcen, die für die digitale Signalverarbeitung wichtig sind. Falls in einem der erwähnten Medien in dieser Arbeit kein externes AUI angeschlossen wird, ist davon auszugehen, dass das Medium intern bereits einen A/D- und D/A-Wandler besitzt. Die technische Erklärung und Funktionsweise eines AUI ist kein Teil dieser Arbeit.

Des Weiteren wird in dieser Arbeit davon ausgegangen, dass es sich bei den Signal-Flussdiagrammen um ein Signal handelt, das mit einem ADC erzeugt wurde und mit einem DAC wiedergegeben wird. Im Falle des in dieser Arbeit entwickelten EG wird eine USB-Soundkarte mit einer Bittiefe von 16-Bit und einer Abtastrate von 48kHz angewendet.

2.2. Digitale Audiosignalverarbeitung

Digital Signalverarbeitung ist das Verfahren, ein gespeichertes Audio oder eine Echtzeit-Klangquelle zu verändern. Signalverarbeitung könnte in drei verschiedenen Szenarien vorkommen (Cipriani & Giri, 2019):

Bereits existierende Datei, die bearbeitet und als eine veränderte Datei gespeichert wird
Der erste Fall ist, dass ein Audio schon existiert und gespeichert ist. Diese gespeicherte Datei kann dann innerhalb des gleichen Mediums mit einem Programm bearbeitet und als eine veränderte Datei gespeichert werden. Beispielprogramme wären eine DAW (Digital Audio Workstation) oder ein Media Player.

Bereits existierende Datei, die in Echtzeit verändert und wiedergegeben wird
Ein weiterer Fall ist, dass eine Audiodatei durch ein Programm verändert und in Echtzeit wiedergegeben wird. Die Veränderung ist beim Abhören wahrzunehmen und könnte als eine separate Datei wieder abgespeichert werden.

Echtzeit Signalverarbeitung
Schließlich gibt es die Möglichkeit, das Audiosignal während des Geschehens mit einem Programm zu verändern und mit einer übersehbaren Verzögerung wiederzugeben. Der Prozess soll dabei so schnell passieren, dass es dem Zuhörer so vorkommt, als wäre das veränderte Signal die eigentliche Schallquelle. Während dieses Prozesses kann das veränderte Audiosignal auch abgespeichert werden.

Ziel dieser Arbeit ist es, ein EG zu entwickeln, das ein eingehendes Line-Pegel-Signal verändert und mit möglichst niedriger Verzögerung wiedergibt. Aus diesem Grund fällt dieses EG in die dritte der oben aufgelisteten Kategorien.

Die Programmierung und Umsetzung eines Effektes kann in kürzester Zeit sehr kompliziert werden, insbesondere wenn es um komplexere oder kaskadierte Anwendungen handelt. Eine sehr häufig verwendete Methode, um die Verständlichkeit der ausgeführten Programmabläufe zu vereinfachen, sind Flussdiagramme. In dem nächsten Abschnitt werden wir dieses Thema anhand eines Beispiels genauer behandeln.

2.3. Flussdiagramme

Die Erzeugung der digitalen Effekte erfolgt durch mehrere Bearbeitungsschritte und viele hintereinander ausgeführte mathematische Funktionen. Um den Signalfluss besser nachvollziehen zu können, werden im Bereich der Audiosignalverarbeitung meistens Flussdiagramme benutzt. Dabei ist der Signalfluss von links nach rechts und von oben nach unten in Pfeilrichtung abzulesen. Neben dem Flussdiagramm werden zudem die mathematischen Operationen von Samples angezeigt. Im Fall einer einfachen Verstärkerschaltung sehen die zwei gängigsten Methoden, diese Operation zu beschreiben, sieht so aus (**Abbildung 2.1**).

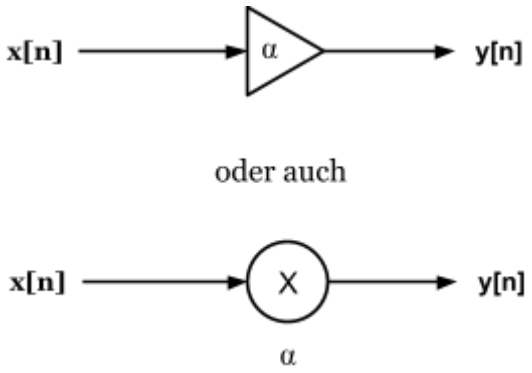

Abbildung 2.1 Verstärkung um α-Fache

In der Abbildung 2.1 wird das einkommende Signal x[n] (bzw. einzelne Samples) um das α-Fache vergrößert. Dabei repräsentiert x[n] das eingehende zeitdiskrete Signal, y[n] das Ausgangssignal und n die einzelnen Samples. Die Multiplikation ist mit einem Dreieck (oder auch, wie im unteren Diagramm, mit einem Kreis und einem Multiplikationszeichen) in der Mitte dargestellt (Abbildung 2.1). Der Multiplikationsfaktor α befindet sich in dem oberen Diagramm in der Mitte des Dreiecks. Hingegen ist im unteren Diagramm der Faktor unter dem Operationszeichen abgebildet. Beide Zeichnungen beschreiben die gleiche mathematische Operation. Beide Darstellungsweisen werden in dieser Arbeit vorkommen.

Die mathematische Operation, die hier ausgeführt wird, wäre dann die in der Gleichung (2.1) beschriebene Funktion.

$$y[n] = \alpha \cdot x[n] \tag{2.1}$$

3. Technologische Grundlagen

3.1. Max und Max Patches

Anhand der Flussdiagramme ist es möglich, in der gewünschten Programmierumgebung Ideen für Signalbearbeitung umzusetzen. Eine der gängigsten Audio Entwicklungsplattformen ist Max des Unternehmens Cycling ‘74.

Max ist eine grafische Programmierumgebung für die Entwicklung von digitalen Audio- und Video Anwendungen. Es basiert auf vorprogrammierten “Kästen", die bestimmte kleinere Aufgaben eigenständig ausführen. Man kann diese Kästen mit den sog. Patch-Cords miteinander verbinden und somit komplexe Anwendungen erzeugen. Es gibt drei verschiedene Arten von Kästen: Object Boxes, Message Boxes und Comment Boxes. Object Boxes sind Elemente, die in dem Patch kleinere Aufgaben übernehmen und ausführen. Message Boxes können Nachrichten erzeugen und sie an verschiedene Objekte senden. Die Comment Boxes haben keinen Einfluss auf die Funktion des Programms, verbessern jedoch dessen Verständlichkeit. Alle diese Informationen und viele weitere ausführliche Erklärungen der Funktionen jeweiliger Objekte sind auf der Dokumentationsseite von Max zu finden (Cycling ‘74, o.D.).

Die Max-Version des Verstärker Beispiels aus dem vorherigen Kapitel wird in der folgenden **Abbildung 3.1** dargestellt.

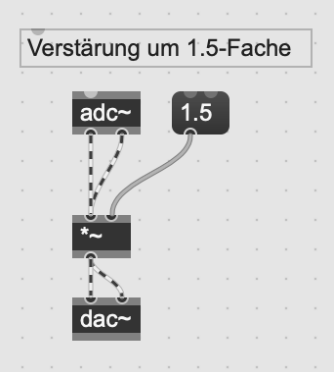

Abbildung 3.1 Verstärkung um 1.5-Fache in einem Max-Patch[1]

[1] Ein Patch ist ein in Max erstellte Datei. Es enthält Elemente wie Boxes und Patchcords.

In der Abbildung 3.1 ist ganz oben eine Comment Box zu sehen, die die Funktion des darunterliegenden Patches beschreibt. Ein “adc~” Objekt liest die Zahlenwerte, die die eingebaute Soundkarte des Rechners erzeugt, und wandelt sie in ein Format um, das in Max bearbeitbar ist. Das Multiplikationsobjekt multipliziert die ankommenden Werte mit einem Multiplikator. Der Multiplikator wird mit der Message Box bestimmt. Durch die Änderung der Zahlenwerte in der Message Box kann man den Verstärkungsfaktor ändern. Das “dac~” Objekt wandelt wiederum die Zahlenwerte in ein analoges Audiosignal und gibt es durch das eingestellte Ausgangsgerät (z.B. Kopfhörer, eingebauter Lautsprecher usw.) wieder.

3.2. RNBO

RNBO ist zurzeit (Oktober 2023) eine neue Erweiterung für Max, die Anfang November 2022 mit der 8.5 Version veröffentlicht wurde (Bracken, 2022). Anders als herkömmliche Max Patches ermöglicht diese Erweiterung Max-Benutzern, ihre Patches für verschiedene Plattformen zu exportieren. Die Benutzeroberfläche ähnelt sehr denen von konventionellen Max-Patches. RNBO besitzt zudem die meisten Objekte, die bei Max zu finden sind. Um RNBO nutzen zu können, müssen die Max-Benutzer eine zusätzliche Lizenz erwerben.

Man startet ein RNBO-Patch, indem man ein “rnbo~” Objekt in einem Max-Patch erzeugt. Schon kann man per Doppelklick auf das Objekt den RNBO Patch öffnen und die gewünschten Effekte programmieren. Viele RNBO Objekte sind mit dem gleichen Namen wie die äquivalenten Max Objekte zu finden. Dadurch lassen sich bereits existierende Patches von Max auf RNBO leicht übertragen. In der **Abbildung 3.2** ist wieder das Verstärker Beispiel in Form eines RNBO-Patches zu sehen.

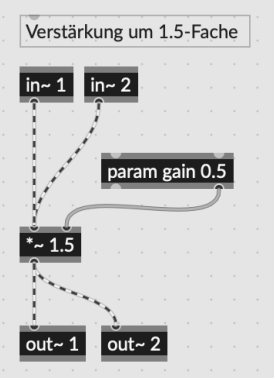

Abbildung 3.2 Verstärkung um 1.5-Fache in einem RNBO-Patch

Anders als bei Max-Patches existieren bei RNBO-Patches “param”-Objekte. Bei der Erzeugung eines param Objekts wird im Hintergrund ein Parameter generiert. Im Fall des in der Abbildung 3.2 dargestellten Patches heißt der erstellte Parameter “gain” und wird mit dem Wert 0.5 initialisiert. “param”-Objekte ermöglichen die externe Ansteuerung der Patches. Durch diese Parameter kann beispielsweise der Patch durch einen Slider in einer Webseite, ein analoges Potentiometer an einem RPI oder durch ein Stück C++ Code verändert werden. Viele weitere Möglichkeiten stehen den Benutzern zur Verfügung. Das Thema Ansteuerung von Objekten per Parameter wird im Kapitel 5.1 genauer behandelt.

Nach der Programmierung stehen den Nutzern verschiedene Export Optionen zur Verfügung:

Exportieren als C++ Code

Die erste Option ist es, den programmierten Patch als kompakten Code zu exportieren. Das exportierte Stück Code kann in das existierende C++ Projekt integriert werden, um die Stärke und Einfachheit von Max nutzen zu können. Man könnte sich die Frage stellen, warum jemand die Anwendung Max benutzen soll, wenn er schon die Programmiersprache C++ beherrscht. Die Antwort auf die Frage ist, dass diese Option es ermöglicht, rapide Prototypen in kürzester Zeit mit wenig Aufwand für laufende Desktop- und Mobileanwendungsprojekte zu erstellen.

Exportieren als VST3/AU

Zudem ermöglicht RNBO die lange erwartete Funktion, RNBO Patches in gängigen Plug-in Formaten für Digital Audio Workstations (DAWs) zu exportieren. Dies ermöglicht den Benutzern, mit RNBO generierte Patches mit anderen zu teilen. Natürlich könnte man seine Plug-ins auch zum Kauf anbieten.

Exportieren als Web Anwendung

Eine weitere Möglichkeit ist, Patches im Web Format zu exportieren und die Anwendungen in eine Webseite einzubauen. Damit könnte man innerhalb einer Webseite interessante audiovisuelle Effekte erzeugen.

Exportieren als Max-Objekt

Mit RNBO kann man seine eigenen Max Objekte (wird auch Max Externals genannt) erzeugen. Dies war bisher auch möglich. Allerdings waren dabei weitere Programmierungsschritte mit C++ notwendig. RNBO ermöglicht das Exportieren von Max

Externals, ohne programmieren zu müssen. Max-Externals können gespeichert, wiederverwendet und mit anderen Max-Benutzern geteilt werden.

Exportieren an Raspberry Pi

Letztendlich kann man einen RNBO Patch auf einen Raspberry Pi exportieren. Das ist besonders interessant, da ein RPI über General Purpose Input/Output (GPIO) Pins verfügt. Dies erlaubt das Anbringen von verschiedenen Sensoren an den RPI. Durch das Anbringen von typischen Hardware Bedienelementen, wie ein Potentiometer oder Drucktaster, kann man einfach ein digitales Gitarren-Effektpedal oder einen Vintage Synthesizer nachbauen. Somit kann man einen Raspberry Pi als Synthesizer, Instrument oder EG nutzen.

3.3. Alternativen zur Max & RNBO

Um die Frage "Wie kann ein Effektgerät in 2023 entwickelt werden?" im weiteren zu beantworten werden weitere Möglichkeiten zur Audiosysteme entwicklung angeschaut:

3.3.1 Pd (pure data)

Wem die kommerzielle Software von Cycling ‘74 nicht preiswert genug erscheint, dem steht eine andere Open Source Option für die Entwicklung von digitalen Audioverarbeitungssystemen zur Verfügung. Einer der Namen, die einem als erstes in diesem Bereich einfällt, ist pure data (Pd).

Pd wird seit 1996 von Miller Puckette entwickelt und kann über die Webseite heruntergeladen werden (Pure Data, o.D.). Bei der Verwendung dieser Software müssen sich die Nutzer weder für die Webseite anmelden noch müssen sie irgendwelche Abonnements abschließen. Die Benutzeroberfläche der Software ist im Vergleich zu Max und RNBO auf den ersten Blick viel einfacher gehalten und könnte neue Benutzer am Anfang verwirren. Doch die grundlegenden Elemente sind denen von Max und RNBO größtenteils sehr ähnlich. Man muss sich nur mit den Benennungen der Objekte auseinandersetzen. Dies ist dank der integrierten Dokumentation unschwer zu bewältigen. Ein Nachteil von Pd-vanilla[2] ist, dass es nicht so viele Anpassungen der Oberfläche erlaubt, wie bei Max. Ein Vorteil ist hingegen, dass Pd auf allen gängigen Betriebssystemen (Windows, Mac und Linux) installiert werden kann. Damit ist auch der Betrieb von Pd wie RNBO auf ein RPI möglich.

[2] Da Pure Data eine Open Source Software ist, existieren viele weitere Vererbungen dieser Software. Die herkömmliche Version der Software heißt Pd-vanilla. Eine andere Version dieser Software, die für die Integration in DAWs optimiert ist, heißt plug data. Solche und viele andere Versionen von pure data für andere Spezialisierungen sind auf der pure data Webseite zu finden (Pure Data, o.D.).

In der **Abbildung 3.3** ist die Pd-vanilla Version des Verstärkerbeispiels zu sehen. Zur Abwechslung wurde in diesem Beispiel ein "Slider"[3] für die Änderung des Verstärkungsfaktors ausgewählt (Abbildung 3.3). Solche und viel weitere Graphical User Interface (GUI) Elemente sind auch selbstverständlich bei Max und RNBO verfügbar.

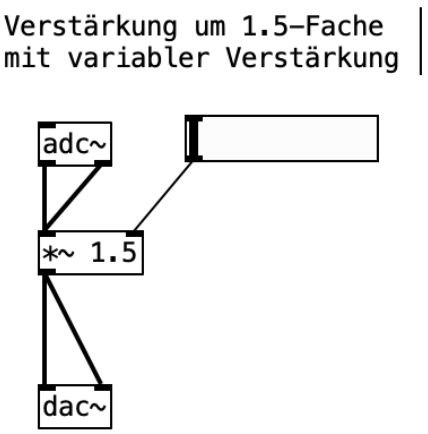

Abbildung 3.3 Verstärkung um 1,5-Fache in einem Pd-vanilla Patch

3.3.2 Teensy und das Audio System Design Tool für die Teensy Audio Library

Eine weitere Möglichkeit zur Entwicklung eines eigenständig funktionierenden Audiosystems ist eine Serie von Entwicklungsboards namens Teensy. Teensy wird von einer kleinen 4-köpfigen Firma in den USA entwickelt (Stoffregen, o.D.). Dabei handelt es sich um USB-basierte Mikrokontrollerentwicklungssysteme (PJRC, o.D.). Sie ähneln sich sehr an die von Arduino Boards, sind aber leistungsfähiger. Solche kleinen Entwicklungsboards richten sich insbesondere an die Hardwareentwicklung. Teensys sind Entwicklungsboard und können nach der Programmierung eigenständig betrieben werden. Sie sollten also nicht direkt mit Max oder Pd verglichen werden. Trotzdem sind sie für die Entwicklung von Effekten sowie anderen interaktiven Anwendungen gut zu gebrauchen und sollten in dieser Auflistung nicht fehlen.

Programmiert werden diese Boards mit einer spezifisch für Teensy geschriebenen Bibliothek für die Arduino IDE[4]. Besonders interessant sind die kleinen Boards für die Projekte, in denen Audioverarbeitung eine große Rolle spielt. Dank des leistungsstarken Prozessors kann der Teensy Audiosignale in nutzbarer Qualität verarbeiten.[5] Zudem bietet die Firma PJRC

[3] Ein Slider ist ein GUI-Element, das bei vielen grafischen Entwicklungsumgebungen benutzt wird.
[4] Integrierte Entwicklungsumgebung (engl. Integrated Development Environment)
[5] Dies gilt für den Fall bei der Benutzung des Teensys zusammen mit einem Audio Adapter Board von PJRC.

eine Audioentwicklungsbibliothek und ein Audio Adapter Board für die Aufrüstung der Teensyboards für audiospezifische Anwendungen. Darüber hinaus hat PJRC ein Online-Tool für das Erstellen von Code für die Teensy Audio Library, um die Entwicklung im Bereich der Audioverarbeitung zu vereinfachen.

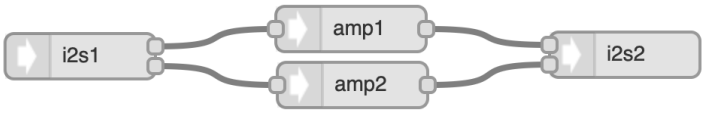

Abbildung 3.4 Verstärker Beispiel im ASDT für Teensy

In der **Abbildung 3.4** ist das Verstärkerbeispiel dieses Mal im Audio System Design Tool von Teensy (ASDT) dargestellt. Die als "i2s1" und "i2s2" bezeichneten Objekte repräsentieren die Ein- und Ausgänge des Audio Adapter Boards. Die beiden "amp" Objekte bezeichnen die Verstärkung einer Leitung, ähnlich wie ein multiplikator Objekt (*~) bei Max oder Pd.

```
#include <Audio.h>
#include <Wire.h>
#include <SPI.h>
#include <SD.h>
#include <SerialFlash.h>

// GUItool: begin automatically generated code
AudioInputI2S            i2s1;           //xy=349,274
AudioAmplifier           amp1;           //xy=529,256
AudioAmplifier           amp2;           //xy=530,293
AudioOutputI2S           i2s2;           //xy=711,273
AudioConnection          patchCord1(i2s1, 0, amp1, 0);
AudioConnection          patchCord2(i2s1, 1, amp2, 0);
AudioConnection          patchCord3(amp1, 0, i2s2, 0);
AudioConnection          patchCord4(amp2, 0, i2s2, 1);
// GUItool: end automatically generated code

// Hinzugefügtes Code startet ab hier:
float verstaerkung;

void setup() {
 verstaerkung = 0.5;
}

void loop() {
 amp1.gain(verstaerkung);
 amp2.gain(verstaerkung);
}
```

Quelltext 3.1 Code zum Betreiben von Teensy als Verstärker um 0,5-Fache

Im **Quelltext 3.1** ist oben mit normaler Schriftart das von dem ASDT für Teensy erstellte Stück Code zu sehen. Wichtig zu erwähnen ist dabei, dass von dem ASDR Code in der Arduino IDE verändert werden muss, damit er den gewünschten Effekt hat. Für den

gewünschten Effekt (für die Verstärkung um das 0,5-Fache) muss der Code ähnlich wie im unteren Teil im Quelltext 3.1 erweitert werden. Die Änderungen wurden mit fetter Schriftart deutlich gemacht (Quelltext 3.1). Möchte man die Verstärkung mit einem angeschlossenen Potentiometer ansteuern, ist dies natürlich auch möglich. Dies erfordert jedoch umfangreichen Code, der in der Regel nicht besonders kompliziert ist.

3.3.3 Daisy Seed von Electro-Smith

Eine weitere eingebettete Plattform für die Entwicklung von Audios ist Daisy Seed von Electro-Smith. Dieser Firma gibt an, dass man dieses Board mit den gängigsten Audio Entwicklungsplattformen (Arduino, C++, Pd oder Max/MSP Gen~) programmieren kann (Electro-Smith, o.D.). Dabei kann man mit diesem Entwicklungsboard eine Audioqualität von 24 Bit bei 96 kHz Abtastfrequenz erreichen (Electro-Smith, o.D.). Diese Eigenschaften sind sehr vielversprechend. Wegen der hohen Anschaffungskosten wurde dieses Entwicklungsboard jedoch in dieser Arbeit noch nicht im Detail getestet worden.

3.4 Vergleich von Entwicklungsumgebungen

Hier werden verschiedenen Entwicklungsumgebungen gegenübergestellt (**Abbildung 3.5**). Der Vergleich wurde mit Blick auf die Audioverarbeitung erstellt, die mindestens einen Stereo-Eingang (oder zwei Mono-Eingänge) für eingehende Signale und einen Stereo-Ausgang (oder zwei Mono-Ausgänge) erfordert. Diese Plattformen wurden hinsichtlich ihrer Fähigkeiten verglichen, mit dem Ziel, hochauflösende Audiosignale (mindestens 16 Bit 44,1 kHz) zu verarbeiten. Man darf nicht vergessen zu erwähnen, dass für jede dieser Plattformen ein separater Rechner für die Programmierung notwendig ist. Möchte man zusätzlich haptische Bedienelemente wie Potentiometer und Knöpfe an die Geräte anbringen, entstehen weitere Kosten.

Für diese Arbeit wurde die Entwicklungsumgebung Max zusammen mit der RNBO Erweiterung ausgewählt. Der Grund für diese Entscheidung ist die Stabilität und das breite Spektrum von Support für diese Programme. Außerdem wurde für die Anwendungen eine kostenfreie Studentenlizenz während der Dauer dieser Arbeit zur Verfügung gestellt.

Name des Plattforms	Kann betrieben werden unter	Zusätzliche Hardware notwendig	Erwerb einer Lizenz notwendig	Preis
Max	Windows, Mac	kein Standalone-Betrieb	ja	9.99$/Monat 99$/Jahr 399$ einmalig (Cycling '74/Max, o.D.)
RNBO	Windows, Mac, Linux (RPI), Web, C++	ja[6]	ja	10$/Monat 100$/Jahr 299$ einmalig (Cycling '74/RNBO, o.D.)
Pd-vanilla	Windows, Mac, Linux	ja[7]	nein	Kostenfrei
Teensy	Standalone	ja[8]	nein	47-58 € (Berry Base, o.D.)
Daisy Seed	Standalone	nein	nein	38€ (Schneidersladen, o.D.)

Abbildung 3.5 Vergleich gängiger Audio Entwicklungsumgebungen

[6] Bei dieser Option ist für die Standalone Inbetriebnahme mindestens ein Einplatinencomputer und ein passendes Audio Interface notwendig.

[7] Bei dieser Option ist für die Standalone Inbetriebnahme mindestens ein Einplatinencomputer und ein passendes Audio Interface notwendig.

[8] Bei dieser Option ist für die Standalone Inbetriebnahme ein passendes Teensy Audio Adapter Board notwendig.

4. Verschiedene Audioeffekte

4.1. Delay-basierte Audioeffekte

4.1.1. Echo

Der größte Vorteil bei digitaler Audiobearbeitung ist die Möglichkeit, das Audiosignal verzögern zu können (Cipriani & Giri, 2020). Diese Verzögerungen können entweder für die Verbesserung des Audiosignals oder für kreative Zwecke verwendet werden. Die meisten gängigen Audioeffekte, von Chorus Effekte bis hin zu Raumsimulationen, basieren auf Delays.

In der **Abbildung 4.1** beschreibt Maempel et al. (2008) die Erzeugung gängiger Delay-basierte Effekte. Indem man bestimmte Kombinationen von Parametern aus der Tabelle (Abbildung 4.1) auswählt, können diverse Delay- oder Modulationseffekte produziert werden (Maempel et al., 2008).

Effekt / Artefakt	delay time	modulation	feedback
Verzögerung single delay	> 40 ms	nein	nein
Verdopplung double tracking	20-40 ms	nein	nein
Echo	> 100 ms	nein	ja
multitap delay	mehrere diskret justierbare Delayzeiten	beliebig	beliebig
Chorus	15-30 ms	ja	nein
Flanger	1-10 ms	ja	ja

Abbildung 4.1 Typische Delayeffekte (Maempel et al., 2008, S. 749)

Die Herstellung eines Echos und Modulationseffekte sind sehr ähnlich. Unterscheidend ist dabei der Betrag der Delayzeit im Vergleich zum Originalsignal. Wenn die Delayzeit kürzer

als 25~35 ms beträgt, werden zwei Signale als eins wahrgenommen (Cipriani & Giri, 2020). Beträgt die Delayzeit länger als 35-40 ms, wird das verzögerte Signal als unabhängiges Signal wahrgenommen. Auf diese Weise gelingt es mit einer Delayzeit zwischen 100 und 1000 ms, einen typischen Echo Effekt zu erzeugen (Maempel et al., 2008).

Es ist also möglich, durch die Änderung der Delayzeit verschiedene Effekte, wie Chorus/Flanger, zu generieren. Alleine die Delay Zeit zu variieren genügt jedoch nicht für die Herstellung dieser Effekte. Zusätzlich ist eine Modulation der Delayzeit für Effekte, wie Chorus und Flanger, notwendig. Diese Modulation kann beispielsweise mit einem niederfrequenten Oszillator (engl. Low Frequency Oscillator, LFO) erreicht werden.

In der **Abbildung 4.2** ist der Aufbau eines typischen Delay Effekts mit Feedback beschrieben.

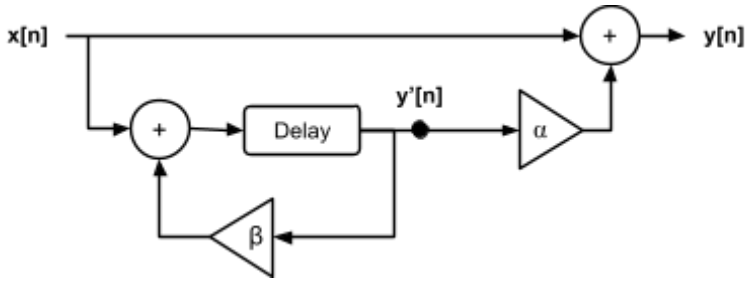

Abbildung 4.2 Delay mit Feedback (Cipriani & Giri, 2020, S. 227)

Diese Flussdiagramm beschreibt folgende Gleichung (4.1) in der Zeitdomäne:

$$y[n] = x[n] + \alpha \cdot (x[n-k] + \beta \cdot y'[n-k]) \; wobei \; k > 0, \; |\beta| < 1 \qquad (4.1)$$

Bei der Formel (4.1) repräsentiert y'[n] das verzögerte Audiosignal, wobei die Verzögerung durch k Samples beschrieben wird. Der Parameter β beschreibt den Verstärkungsfaktor für das Feedback. Der Faktor α beschreibt das Verhalten zwischen dem unveränderten Signal x[n] und dem veränderten Signal y'[n]. Es ist also möglich, ihn als Parameter zur Einstellung des Dry-Wet-Verhaltens zu benutzen. In dem nächsten Kapitel wird erklärt, wie diese Delay Lines für die Filterung bestimmter Frequenzen angewendet werden.

4.1.2. FIR Filter

Bei Audiofiltern geht es um die Filterung bestimmter Frequenzen eines Audiosignals. Diesen Effekt kann man für die Glättung eines rauschenden Signals, Isolierung eines Frequenzbandes oder für gestalterische Zwecke verwenden. Das einfachste Vorgehen bei der Implementierung eines Filters wäre die Mittelwertbildung der einzelnen Samples. Dies verursacht jedoch große Verzögerungen und Verluste in der Auflösung des Signals. Daher wird dies im Bereich Audio eher selten verwendet.

Ein gängiges Verfahren, das im Bereich der digitalen Audio verwendet wird, sind Finite Impulse Response (FIR) Filter. Die Idee dabei ist, dass für jeden gegebenen Samplewert die daneben liegenden Filterwerte um bestimmte Koeffizienten multipliziert werden. Diese Koeffizienten werden mit der erzielten Grenzfrequenz berechnet. Diese Art von Filter wird auch für die Glättung eines rauschenden Signals verwendet. Den Aufbau eines FIR Filters zweiter Ordnung beschreibt Cipriani und Giri (2019) in der **Abbildung 4.3.**

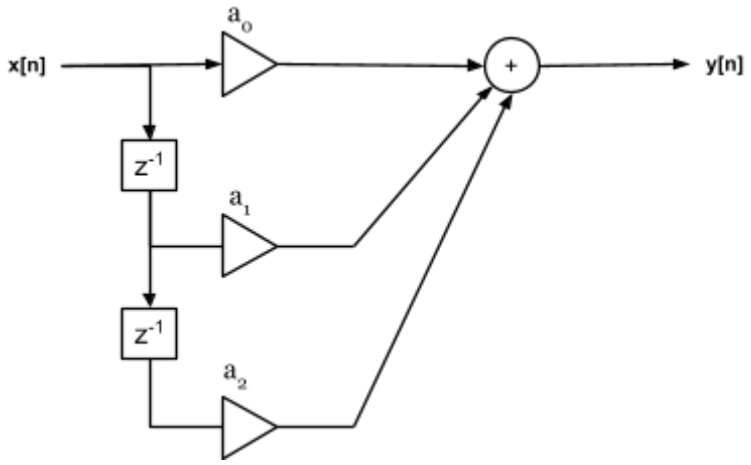

Abbildung 4.3 Darstellung eines FIR Filters zweiter Ordnung (Cipriani & Giri, 2019, S. 392)

Diese Darstellung beschreibt die folgende mathematische Funktion in der Zeitdomäne:

$$y[n] = a_o x[n] + a_1 x[n - 1] + a_2 x\,[n - 2] \tag{4.2}$$

Dabei beschreibt n die Nummer des Samples, wobei $x[n]$ den aktuellen, $x[n - 1]$ den nächsten und $x[n - 2]$ den danach kommenden Samplewert bezeichnet. Durch die bestimmung der a_o, a_1 und a_2 kann eine Gewichtung zwischen den Samplewerten gemacht

werden. Bei einem FIR-Filter werden nur die ankommenden Samplewerte für Gewichtung mit eingerechnet (Cipriani & Giri, 2019). Sie meinen zudem, dass dementsprechend keine Rekursion stattfindet. Deswegen besitzen diese Filter auch keine Möglichkeit für die Rückkopplung des Signals. Ein fortgeschrittener Typ von Filter, die diese ermöglichen, sind Infinite Impulse Response (IIR) Filter.

4.1.3. IIR Filter

Der Aufbau dieser Art von Filter ähnelt sehr an die von IIR-Filtern. Der unterscheidende Aspekt ist, dass bei IIR Filter einige Samples am Ausgang gespeichert und nach der Berechnung erneut in die Gewichtung einbezogen werden (Cipriani & Giri, 2019). Zölzer (2011) beschreibt einen kanonischen Filter zweiter Ordnung wie in **Abbildung 4.3**.

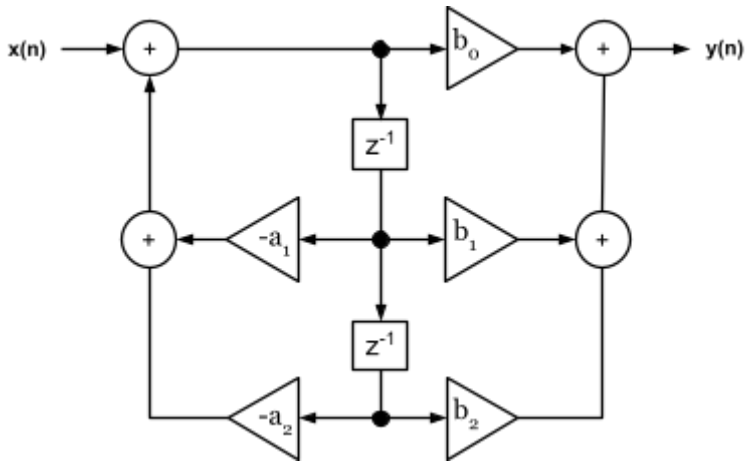

Abbildung 4.3 Digitaler kanonischer Filter zweiter Ordnung (Zölzer, 2011, S. 49)

Hier kommen zwei neue Komponenten in den Aufbau, nämlich die zwei Samples, die gespeichert und wieder in den Filter gegeben werden. Diese Samples bezeichnet Zölzer (2011) in seinem Aufbau als a_1 und a_2.

Durch die Abbildung 4.3 berechnet Zölzer (2011) die Gleichungen (4.3) und (4.4).

$$x_h(n) = x(n) - a_1 x_h(n-1) - a_2 x_h(n-2) \tag{4.3}$$

$$y(n) = b_0 x_h(n) + b_1 x_h(n-1) + b_2 x_h(n-2) \tag{4.4}$$

Die Herleitung der Gleichungen (4.3) und (4.4) ergibt die Übertragungsfunktion (4.5) (Zölzer, 2011).

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{4.5}$$

Für die Manipulation der Grenzfrequenz und Güte des Filters werden die Koeffizienten K und Q benutzt. Die Berechnung von K und Q für Tiefpassfilter beschreibt Zölzer (2011) durch die Gleichung (4.6).

$$K = tan(\pi \cdot \frac{f_c}{f_s}),\ Q = \frac{1}{\sqrt{2}} \tag{4.6}$$

Schließlich ist es möglich, mit den Koeffizienten der **Abbildung 4.4** einen Tiefpassfilter zweiter Ordnung zu konstruieren.

Filter Typ	b_0	b_1	b_2	a_1	a_2
Tiefpass	$\frac{K^2Q}{K^2Q+K+Q}$	$\frac{2K^2Q}{K^2Q+K+Q}$	$\frac{K^2Q}{K^2Q+K+Q}$	$\frac{2Q(K^2-1)}{K^2Q+K+Q}$	$\frac{K^2Q-K+Q}{K^2Q+K+Q}$
Hochpass	$\frac{Q}{K^2Q+K+Q}$	$-\frac{2Q}{K^2Q+K+Q}$	$\frac{Q}{K^2Q+K+Q}$	$\frac{2Q(K^2-1)}{K^2Q+K+Q}$	$\frac{K^2Q-K+Q}{K^2Q+K+Q}$
Bandpass	$\frac{K}{K^2Q+K+Q}$	0	$-\frac{K}{K^2Q+K+Q}$	$\frac{2Q(K^2-1)}{K^2Q+K+Q}$	$\frac{K^2Q-K+Q}{K^2Q+K+Q}$

Abbildung 4.4 Koeffizienten für Tiefpass-, Hochpass- und Bandpassfilter zweiter Ordnung (Zölzer, 2011, S. 50)

Diese Berechnungen mögen im ersten Blick kompliziert erscheinen. Doch die meisten Programme für Audiosignalverarbeitung verfügen über vorinstallierte Werkzeuge, um die gewünschten Filterkoeffizienten zu berechnen.

4.2. Nicht lineare Audioeffekte

4.2.1 Clipping und Klicks bei Audio

Mit den Verstärker-Beispielen aus den vorherigen Abschnitten ist unschwer zu erkennen, dass manche Samples womöglich den maximalen Wertebereich der Bittiefe überschreiten können. Auch Cipriani und Giri (2019) deuten darauf hin, dass der überschrittene Bereich weggeschnitten und der maximal mögliche Signalwert wiedergegeben wird. Das verursacht jedoch eine Verzerrung im Signal, das vom ursprünglichen Signal abweicht (Cipriani & Giri, 2019). Dieses Ereignis wird als “Clipping” bezeichnet und ist im Bereich der Digital Audio unerwünscht. Durch Kompression oder Limitierung des Signals ist es möglich, solchen Effekte vorzubeugen.

In der **Abbildung 4.5** wird links ein Sinussignal dargestellt. Dieses Signal wird um einen bestimmten Faktor α verstärkt. Die Verstärkung ist so hoch, dass der maximale Wert, den das System wiedergeben kann, überschritten wird. Folglich werden die überschrittenen Bereiche abgeschnitten und so verzerrt wiedergegeben. Die Abbildung 4.5 wurde zur Veranschaulichung mit der Software Pure Data erstellt und gibt dabei keine Messergebnisse wieder.

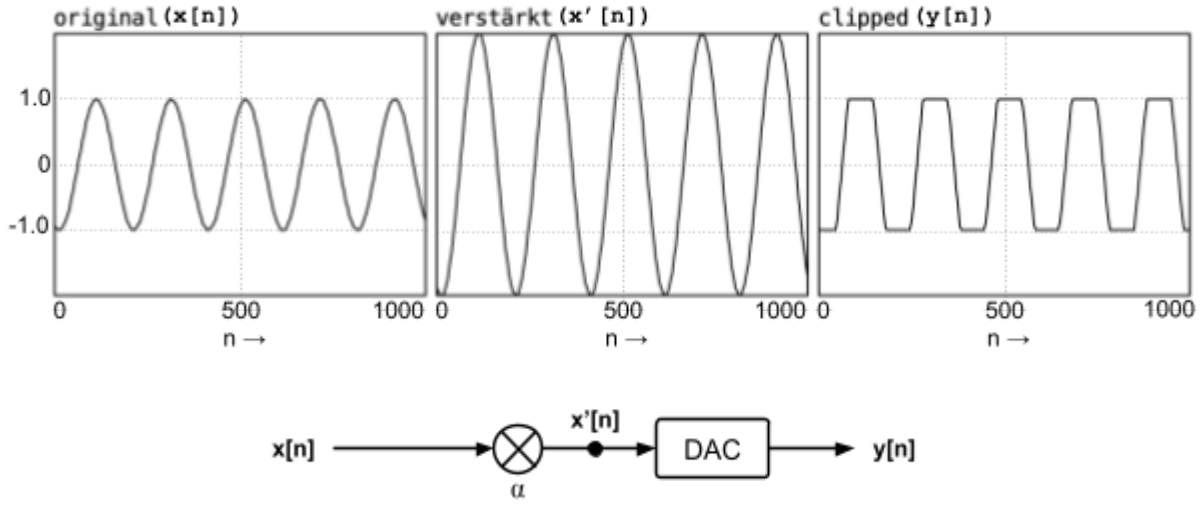

Abbildung 4.5 Clipping

Eine weitere Ursache für unerwünschte Effekte sind große Veränderungen der Amplitudenwerte, die in kurzen Zeitabschnitten geschehen. Rapide Veränderungen erzeugen im Bereich der Digital Audio Klickgeräusche (Cipriani & Giri, 2019). Eine Ursache für solche Artefakte ist, dass der CPU zu hoch beansprucht wird. Wenn die CPU-Belastung zu hoch ist

und Audio nicht in der ausgewählten Buffer-Rate bearbeitet werden kann, kommt es zu Knistern im Audiosignal (Ableton, o.D.).

Bei der Programmierung von Audioeffekten muss darauf besonders Acht gegeben werden, damit keine unerwünschten Ergebnisse entstehen. Durch Techniken, die wir im Weiteren betrachten werden, ist es möglich, solchen Fehler vorzubeugen und die Dynamik des Signals zu verändern.

4.2.2. Kompressoren und Envelope Following

Nichtlineare Verarbeitung beinhaltet Effekte, wie Distortion, Overdrive, Kompression uvm.. Man spricht von einer nichtlinearen Verarbeitung, wenn in den relevanten Signalverarbeitungsalgorithmen oder im Signalverarbeitungsgerät das zu bearbeitende Signal die Voraussetzungen der Linearität nicht erfüllt (Zölzer, 2011, S. 101).

Ein typisches Beispiel für eine solche Verarbeitung ist ein Kompressor. Das Hauptziel eines Kompressors besteht darin, die Dynamik eines Audiosignals zu verändern. In anderen Worten bedeutet das, dass er die Unterschiede der niedrigsten und höchsten Amplitudenwerte des Signals verringert. Hier kommt der Envelope Follower (EF) zum Einsatz. Das durch den EF erzeugte Steuersignal wird verwendet, um den Kompressor so zu steuern, dass er die Lautstärke des verarbeiteten Signals an den Stellen reduziert, an denen sie am höchsten sind. Ein Envelope Follower erzeugt dabei basierend auf die Amplitudenwerte des Signals Kontrollwerte, die zum Ansteuern verschiedener Parameter dienen.

Die Erstellung eines EFs (oder auch manchmal Amplituden- /Pegelerkennungsschema genannt) beschreibt Zölzer (2011) im folgenden Blockdiagramm (**Abbildung 4.6**).

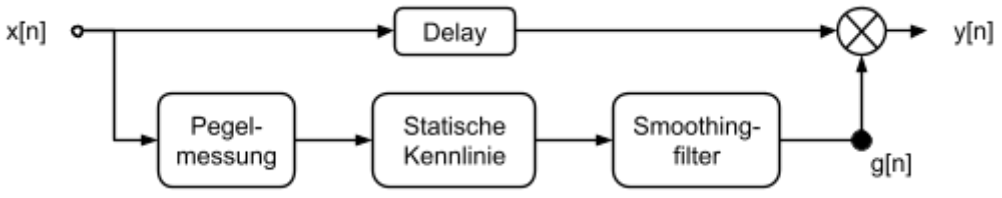

Abbildung 4.6 Blockdiagramm einer Dynamikbeeinflussung (Zölzer, 2011, S. 106)

Wie es im Schema zu sehen ist, besteht der wesentliche Teil der dynamischen Verarbeitung aus vier Schritten (Abbildung 4.6). Im ersten Schritt werden aus den Amplituden des ankommenden Signals die absoluten Werte ausgerechnet. Aus der Hüllkurveninformation wird mit der Statischen Kennlinie der Verstärkungsfaktor g[n] abgeleitet. Anschließend wird ein Filter für die Glättung des Signals angewendet, um sprunghafte Verstärkungsänderungen zu vermeiden. Um den Effekt des Signals zu verstärken, kann schließlich der Steuerwert um einen bestimmten Faktor multipliziert werden. Zölzer (2011) meint außerdem, dass das ankommende Signal optional für die Verarbeitung um eine bestimmte Zeit verzögert werden kann. Diese Verzögerung ermöglicht die voreilende Steuerung der Dynamik.

Abbildung 4.7 Aufbau eines Envelope Followers in Pd

Nach der Idee von Zölzer (2011) wurde ein Aufbau zum Erzeugen eines Envelope Followers in Pd erstellt (**Abbildung 4.7**). Das ankommende Signal (in diesem Fall eine Sinuswelle mit einer Frequenz von 440 Hz) wird in das "abs~" Objekt weitergeleitet. Dieses Objekt berechnet die absoluten Werte des Signals. Die dadurch entstehende Funktion kann als |x[n]| beschrieben werden. Schließlich wird diese Funktion mit dem "lop~"[9] Filter geglättet. Um den Pegelunterschied nach der Filterung auszugleichen, wird das Steuersignal um das 1,5-Fache verstärkt. In jedem Schritt des Signalflusses wurden die Werte zum Betrachten in einem Array gespeichert. Dadurch sind die Graphen in der **Abbildung 4.8** entstanden. Diese Graphen sollen die einzelnen Schritte der Signalgenerierung für die Steuerung der Dynamik veranschaulichen.

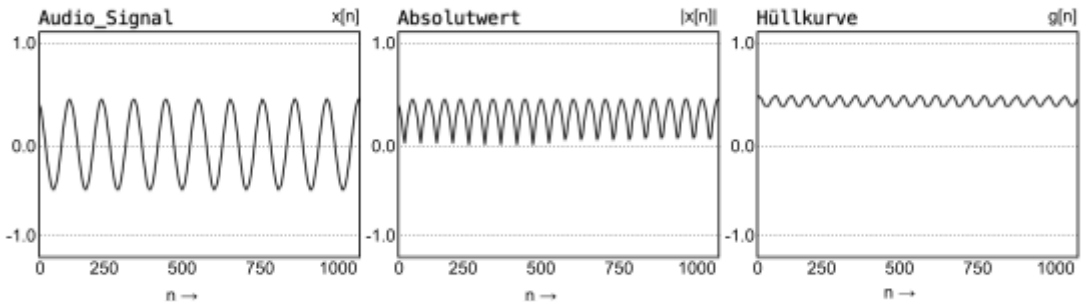

Abbildung 4.8 Audio Signal x[n], sein Absolutwert und die gefilterte Hüllkurve g[n]

[9] Das lop~ Objekt ist die Implementierung eines Einpoligen TPF in Pd

Im Falle eines Kompressors wird der EF für die Verbesserung des Audiosignals benutzt. Der Faktor g[n] wird verwendet, um den Signalpegel zu verändern und die Dynamik des Signals zu reduzieren. Es ist aber auch möglich, diesen Wert für andere kreative Zwecke anzuwenden. Auch Cipriani & Giri (2020) weisen auf die Möglichkeiten zur Steuerung der Effekte mittels der Hüllkurveninformation eines Signals hin. Ausgehend von diesem Gedanken wird in dem entwickelten Audio Effekt die Hüllkurve für das Ansteuern der Grenzfrequenz eines Tiefpassfilters (TPF) benutzt. Im nächsten Kapitel wird dieses Effekt im Detail erklärt.

5. Entwicklungsprozess des Effektgeräts

5.1. Software Implementierung

5.1.1. Übersicht des implementierten Effekts

Mit dem aufgebauten Effekt wurde erzielt, den Klang des ankommenden Signals mit einem Filter und Delay Effekt zu manipulieren. Die Grenzfrequenz des Filters soll mit der Hüllkurve des ankommenden Signals gesteuert werden. Bei dem Filter handelt es sich um einen Tiefpassfilter. Die Hüllkurve des ankommenden Signals steuert die Bewegung der Grenzfrequenz des Tiefpassfilters in die Richtung der Tiefen Frequenzen. Zusätzlich kann der Benutzer die Geschwindigkeit und Verlauf dieser Bewegung beeinflussen. Der theoretische Aufbau wird in der **Abbildung 5.1** dargestellt.

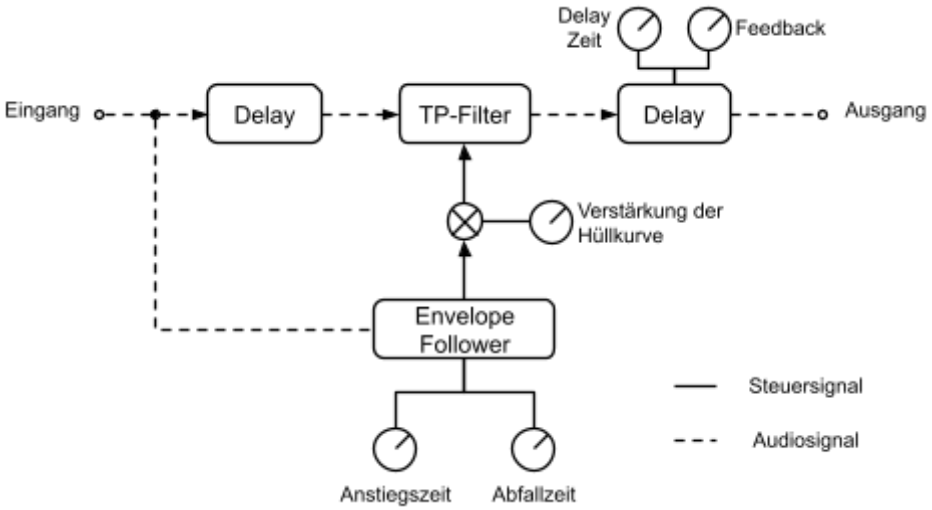

Abbildung 5.1 Aufbau des implementieren Effekts

Im Vergleich zu bisherigen Abbildungen existieren in der Abbildung 5.1 zwei verschiedene Arten von Leitungen. Die gestrichelte Leitung beschreibt den Pfad, den das Audiosignal zurücklegt. Die durchgängige Leitung hingegen bezeichnet die Zahlenwerte, die zur Steuerung verschiedener Parameter dienen.

Das Audiosignal soll zuerst an den Envelope Follower übertragen werden, um die Hüllkurve zu berechnen (Abbildung 5.1). Gleichzeitig wird es durch den oberen Pfad verzögert. Diese

Verzögerung dient dazu, dem RPI genügend Zeit zu verschaffen, um die Hüllkurve zu bestimmen, diese Werte für den Filter anzupassen und die Filterkoeffizienten zu berechnen. Die vom Hüllkurven-Folger erzeugten Werte werden zunächst in einen Wert umgewandelt, der innerhalb des Arbeitsbereichs des Filters liegt. Dieser Wert wird bei der Berechnung durch zwei Parameter geglättet. Schließlich wird er um einen einstellbaren Faktor verstärkt. Das gefilterte Signal wird dann in einem Echoeffekt weitergegeben. Der Echo-Effekt soll dazu beitragen, die Dauer des Signals bei perkussiven Signalen mit kurzer Hüllkurve zu verlängern.

Idee für diesen Effekt entstand aus der Annahme, dass ein Instrument seine Hüllkurve nicht oder sehr schwer manipulieren lässt. Beispiel für so ein Instrument wäre ein Keyboard mit vorinstallierten Stimmen, Kalimba oder Schlagzeug. Bei der Umsetzung dieser Idee wurde auf die gesammelten Informationen aus Kapitel vier zurückgegriffen.

5.1.2. Max-Patch

Zu Beginn der Implementierung des Effekts muss ein Max Patch erstellt werden. Dieser Patch hat die Aufgabe, den RNBO Patch zu generieren und zu benennen. Es ist möglich, Audio- und Steuersignale an den erstellten RNBO Patch zu senden. Dadurch ist man in der Lage, die RNBO Umgebung innerhalb von Max zu testen und für die Fehlersuche zu verwenden. Anstatt also nach jeder Änderung das Programm neu zu kompilieren und an den RPI zu senden, können die Änderungen sofort innerhalb von Max getestet werden. Eine weitere Erleichterung des Max Patches ist, dass der Benutzer den Effekt mit seinen eigenen Audiodateien ausprobieren kann, ohne ein weiteres Hardwaregerät zu benötigen.

Im Fall des entwickelten Effekts wurde für das User Interface (UI) ein Foto von dem entwickelten haptischen Gerät verwendet. Grund für diese Entscheidung ist es, den Benutzern ein möglichst realistisches Gefühl von dem haptischen Gerät zu ermitteln. In der folgenden Abbildung ist eine Bildschirmaufnahme von dem fertigen Max Patch zu sehen **(Abbildung 5.2)**.

Links oben im Patch unter dem Titel ist ein “playlist~” Objekt von Max zu sehen (Abbildung 5.2). Dieses Objekt erlaubt es, eigene Audiodateien in den Patch zu laden und diese abzuspielen. Die Ausgänge des playlist~ Objekts werden im Hintergrund direkt ohne Verstärkung an die Eingänge des rnbo~ Patches geleitet. Somit ist es möglich, gewünschte Audiodateien per Drag-and-drop mit dem Effekt innerhalb von Max auszuprobieren.

Abbildung 5.2 Der MeinEffekt.maxpat Patch

Die anderen in der Abbildung 5.2 dargestellten Elemente dienen größtenteils zur Veränderung der Parameter des rnbo~ Patches. Bei den Bedienelementen handelt es sich um "pictctrl" Objekte. Dies sind im Grunde genommen herkömmliche Interface Objekte, deren Erscheinung durch Bilder verändert werden können. In diesem Falle wurden die 3D Dateien der Potikappen (Anhang A: Effektgeraetknopf_1.stl, Effektgeraetknopf_2.stl) in einem 3D Grafikprogramm aus dem gleichen Winkel in 64 Positionen gerendert. Die Positionen ergaben sich durch die Rotation der Modelle von 0° bis 270° in regelmäßigen Abständen. Die gerenderten einzelnen Frames wurden mit Hilfe von FFmpeg[10] in eine einzige Bilddatei umgewandelt. Diese Bilddateien (Anhang A: Potsheet1.png, Potsheet2.png, Potsheet3.png, Potsheet4.png, Potsheet5.png) kann Max für die Darstellung der pictctrl Objekte interpretieren. In der **Abbildung 5.3** wird das Vorgehen der Umwandlung durch FFmpeg veranschaulicht.

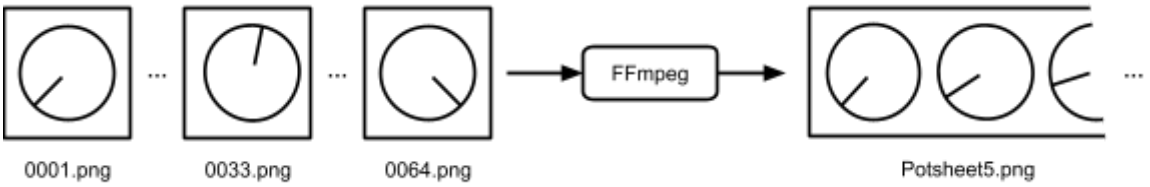

Abbildung 5.3 Vorgehen mit FFmpeg

Um diese Wandlung durchzuführen, wurde für den Fall des fünften Potis dieser Befehl über die Console in dem entsprechenden Ordner ausgeführt (**Quelltext 5.1**).

[10] FFmpeg ist ein führendes Multimedia-Framework, das gängige AV-Dateien verarbeiten kann (FFmpeg, o.D.).

```
ffmpeg -i %04d.png -filter_complex tile=64x1 Potsheet5.png
```

Quelltext 5.1 FFmpeg Befehl (InterPixel, 2021)

Die pictctrl Objekte liefern je nach Zeigerposition einen Integerwert von 0 bis 127, wobei diese Minimum und Maximum Werte auch verändert werden können. Diese Werte werden passend zu den im RNBO generierten Parametern skaliert. Für die Skalierung werden Objekte namens “scale” benutzt. Die mathematische Operation, die das scale Objekt im Hintergrund ausführt, kann mit Gleichungen (5.1) beschrieben werden. Zusätzlich kann das Objekt im gegebenen Wertebereich exponentielle Skalierungen ausrechnen. Für diesen Fall wurde jedoch nur die lineare Skalierung (5.1) verwendet.

$$y(n) = a_{unten} + \left(a_{oben} - a_{unten}\right) \cdot \left|\frac{n - e_{unten}}{e_{oben} - e_{unten}}\right| \tag{5.1}$$

Dabei beschreibt n den Eingangswert in das Objekt, der skaliert werden soll. $y(n)$ ist der Ausgangswert. e_{unten} und e_{oben} sind die unteren und oberen Schranken des Eingangswerts. Gleicherweise beschreiben a_{oben} und a_{unten} den Maximal- und Minimalwert des Ausgangs.

5.1.3. Audio Eingang

Für den Audioeingang von außerhalb der RNBO Umgebung wird das RNBO Objekt “in~” verwendet. Dieses Objekt verbindet den Audioeingang mit einer verfügbaren Audioquelle in der exportierten Entwicklungsumgebung. Im Falle des Hardware Geräts verbindet RNBO den Audioeingang mit dem in Jack eingestellten Audio Device. Eine Bildschirmaufnahme von dem relevanten Teil des rnbo Patches ist in der **Abbildung 5.4** zu sehen.

Nachdem die linke und rechte Eingänge empfangen werden, werden die zwei Signale mit einem “*~” Objekt um 0.5 Fache verstärkt (Abbildung 5.4). Dies hat den Grund, Übersteuerungen bei dem summierten Signal zu vermeiden.
Aus dem Ausgang des summierten Signals werden zwei verschiedene Pfaden generiert. Der erste Pfad geht unverändert an die Envelope Following. Der zweite Pfad wird um 30 ms verzögert, um genügend Zeit für die Kalkulationen zu haben. Wie dieser Betrag ausgerechnet wurde, wird im Kapitel sechs detaillierter erklärt.

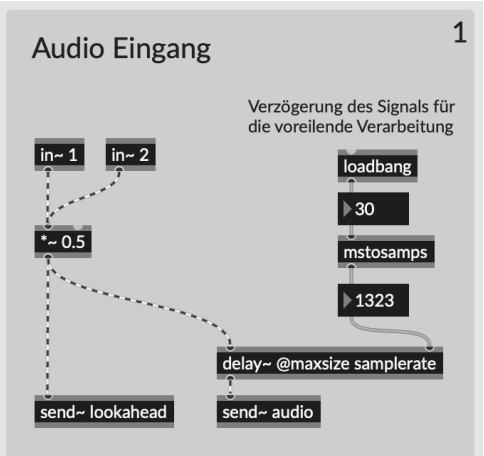

Abbildung 5.4 Audio Eingang des rnbo~ Patches

5.1.4. Envelope Follower

Für das Envelope Following wurde die Theorie aus dem Kapitel 4.2.2. angewendet, um den Aufbau in RNBO nachzubauen. Darauf basierend hat sich der folgende Patch im rnbo~ ergeben (**Abbildung 5.5**).

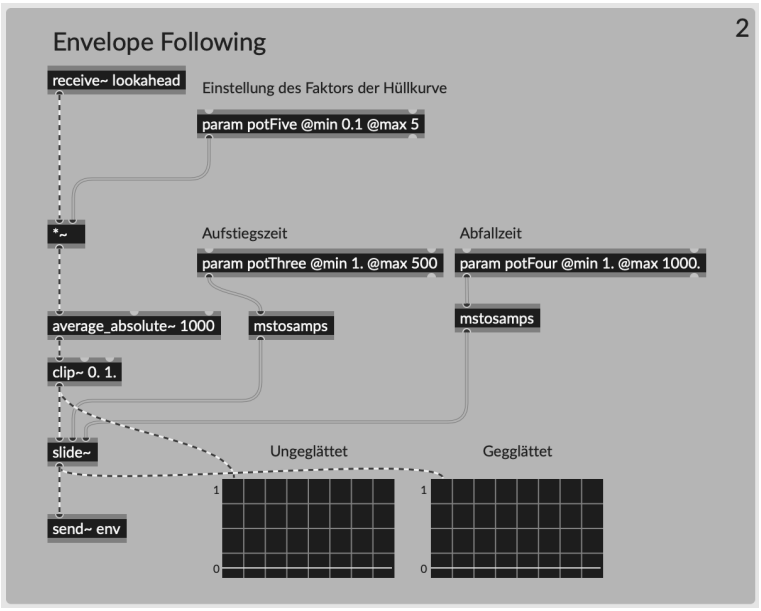

Abbildung 5.5 Envelop Following im rnbo~ Patch

*~

Das einkommende Signal vom Audioeingang wird um einen Faktor verstärkt (Abbildung 5.5). Dieser Faktor wird mit dem fünften Poti auf dem EG bestimmt und beträgt zwischen 0.1

und 5. Dies ermöglicht genügend Verstärkung, um auch von leisen Signalen den Hüllkurvenverlauf auszurechnen. Außerdem hilft dieses Parameter, den Effekt der Envelope Following zu betonen. Um die Verstärkungsfaktor einzustellen, wurde der Parameter "potFive" erstellt und an den rechten Eingang des *~ Objekts angeschlossen.

average_absolute~

Nach der Verstärkung geht das Audiosignal in das "average_absolute~" Objekt rein. Dieses Objekt rechnet, wie der Name schon verrät, den absoluten Durchschnittswert für die eingestellte Samplemenge. In diesem Falle sind es 1000 Samples. Es ist wichtig zu erwähnen, dass die gemessenen Werte innerhalb des average_absolute~ Objects durch einen simplen FIR Filter geglättet werden (Cycling'74, o.D.). Aus diesem Grund wurde ein weiterer Schritt zum Filtern des Signals weggelassen.

clip~

Durch die Verstärkung ist es möglich, dass einige Samples den Wert 1 überstreiten. Demnächst werden solche Werte, die außerhalb von 0 und 1 liegen, durch das "clip" Objekt eliminiert und auf den nächstmöglichen Wert gesetzt.

slide~

Wie in der Abbildung 5.1 beschrieben, sollte das Signal der Hüllkurve in der Anstiegszeit und Abfallzeit durch den Benutzer angepasst werden können. Um dieses Ziel zu erreichen, wurde das Steuersignal als nächstes an das "slide~" Objekt angebunden. Mit diesem Objekt kann ein steigendes Signal entsprechend der angegebenen Zeit schrittweise erhöht (bzw. ein fallendes Signal schrittweise verringert) werden. Zur Veränderung dieser Zeiten wurden die Parameter "potThree" und "potFour" erstellt. Durch einige Proben hat sich herausgestellt, dass eine Anstiegszeit von 0 bis 500 ms und eine Abfallzeit zwischen 1 und 1000 ms am vielfältigsten ist.

send~

Das "send~" Objekt ermöglicht eine Signalleitung ohne Patch Cords zu verbinden. Das von dem Envelope kommende Signal wird durch das send~ Objekt als "env" benannt. Mit einem gleichnamigen "receive~" Objekt kann dieses Signal an einen Signaleingang eines beliebigen Objektes innerhalb des gleichen Patches gebunden werden.

5.1.5. Mapping der Frequenz

Die durch die Hüllkurve des Signals erstellten Werte betragen eine Zahl von 0 bis 1. Dieser Wertebereich ergibt für die Grenzfrequenz eines Filters keinen Sinn und muss entsprechend der Hörschwelle angepasst werden. Für Skalierung dieses Signals wurde vorgenommen, das scale~ Objekt zusammen mit dem "mtof~" Objekt zu benutzen (**Abbildung 5.6**).

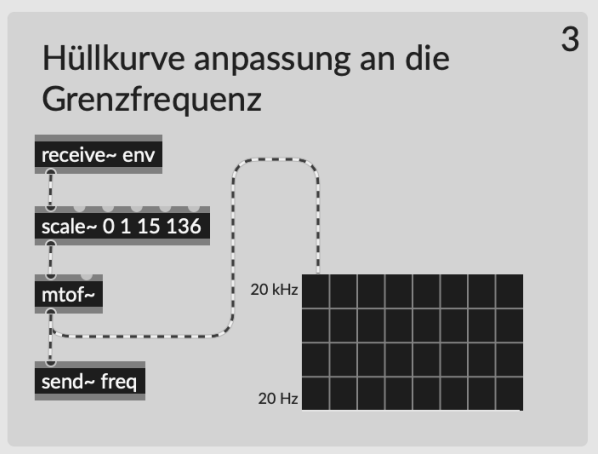

Abbildung 5.6 Hüllkurve Anpassung an die Grenzfrequenz im rnbo~ Patch

Das mtof~ Objekt wurde eigentlich dafür entwickelt worden MIDI Noten von 0 bis 127 auf eine Frequenz in musikalischen Intervallen zu skalieren. Den Zusammenhang dahinter erklären Cipriani und Giri (2019) sehr gut in ihrem Kapitel über den Zusammenhang zwischen Frequenz und musikalischen Intervallen. Cipriani und Giri (2019) meinen, dass durch die Halbierung der Frequenz einer Note die gleiche Note in einer Oktave tiefer erreicht wird. Sie meinen zudem, dass moderne Musik die gleichstufig temperierte Stimmung verwendet (Cipriani & Giri, 2019). Nach diesem Stimmungssystem wird eine Oktave in 12 gleich große Intervallen aufgeteilt, wobei der Abstand zwischen jedem Ton einen Halbtonschritt beträgt. Dementsprechend soll zwischen zwei Noten ein Frequenzunterschied um das $\sqrt[12]{2}$-Fache liegen.

rnbo~ nimmt als Basisnote für die gleichstufig temperierte Stimmung außergewöhnlicherweise die Note A4, was in der MIDI Skala die Note 69 entspricht. Als Basisfrequenz für die gleichmäßige Skalierung wird aber wie gewohnt die Frequent 440Hz angewendet. Aus diesen Informationen kann die mathematische Funktion, die das mtof~ Objekt ausführt, in der Gleichung (5.2) beschrieben werden.

$$f(x) = 440 \cdot \left(\sqrt[12]{2}\right)^{x-69},\ x \in \Re \tag{5.2}$$

Dabei beschreibt x den Eingangswert und kann auch einen Float-Wert annehmen.

Für die Skalierung der zwischen 0 und 1 liegenden Hüllkurven Werte wurde wieder das scale~ Objekt angewendet. Durch die Rechnung für die unteren Hörschwelle Wert von $f(x) = 20$ Hz wird der minimal Wert $x_{min} \approx 15.487$ ausgerechnet (5.3). Dieser Wert wurde auf 16 aufgerundet und als untere Grenze für den Ausgang des scale~ Objekts eingestellt (Abbildung 5.6). Gleicherweise wurde für die Frequenz $f(x) = 20$ kHz der Wert $x_{max} \approx 135.076$ ausgerechnet, der Wert auf 135 aufgerundet und als obere Grenze angenommen. Die Leitung, die die Frequenz Information überträgt, wird mit dem send~ Objekt "freq" bezeichnet.

5.1.6. Echo Effekt

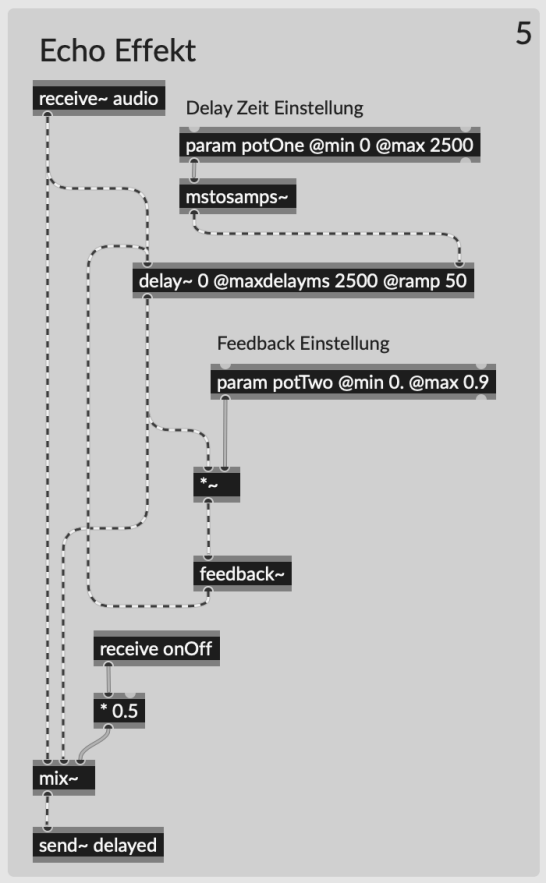

Abbildung 5.7 Echo Effekt im rnbo~ Patch

Während im Hintergrund die Hüllkurve ausgerechnet und an die Grenzfrequenz angepasst wird, wird im eigentlich hörbaren Bereich des Patches auf dem Audiosignal ein Echo Effekt angewendet. In der **Abbildung 5.7** ist zu sehen, dass von dem Audio Eingang verzögert gesendete Audio Signal durch das receive~ Objekt empfangen wird. Der restliche Aufbau ähnelt sehr dem Aufbau in der Abbildung 4.2 in Kapitel 4.1.1.. Üblicherweise werden für Delay Lines in Max die "tapin~" und "tapout~" Objekte verwendet. Dies ist in RNBO nicht der Fall. Die Bearbeitung der Delays ist in RNBO etwas anders:

Das Signal, das zu bearbeiten ist, wird wieder mit dem receive~ Objekt empfangen. Anschließend wird das Signal, das verzögert werden möchte, in einem delay~ Objekt weitergeleitet. In RNBO ist es wichtig, dass im delay~ Objekt die maximal benötigte Zeit eingetragen wird. Dies wird durch den "@maxdelayms" Attribute angegeben. Auch in unserem Beispiel wurde eine maximale Delayzeit von 2500 ms und eine "ramp" Zeit von 50 ms angegeben. Das "@ramp" Attribute bestimmt die Veränderung der Delayzeit pro Zeiteinheit. Dies kann wie ein eingebautes "slide~" Objekt mit gleicher Anstieg- und Abfallzeit für Nummerwerte angenommen werden.

Nach dem Delay sollte für die Feedbackleitung üblicherweise das Signal mit einem Dämpfungsfaktor wieder in den Eingang des Delays geleitet werden. RNBO erlaubt jedoch solche Feedbackleitungen nicht. Stattdessen muss diese Leitung durch ein zusätzliches "feedback~" Objekt kenntlich gemacht werden (Abbildung 5.7). Die Menge der Feedback wird durch den Parameter potTwo bestimmt. Der Parameter wird durch die Attributes "@min" und "@max" zwischen 0 und 0.9 eingegrenzt. Dies soll die Übersteuerung des Signals vermeiden.

Zum Schluss wird das verzögerte Signal mit dem unveränderten Signal zusammengemischt. Das Mischverhältnis beträgt 50:50 und wird durch den "mix~" Objekt bestimmt. Im Idealfall sollte der Benutzer dieses Mischverhältnis selber auswählen können. Aufgrund der niedrigen Anzahl von Potis wurde aber die Änderungsmöglichkeit dieses Parameters weggelassen. Stattdessen wurde der Standard Wert auf 0.5 gestellt, was sich durch die Probe des Effekts am angenehmsten angehört hat.

5.1.7. Filterung

Die Filterung, die in dem Effekt ausgeführt wird, basiert auf die in Kapitel 4.1.3. gesammelten Informationen. Am Anfang wird das durch den Echo Effekt gesendete Signal,

wie wir es schon kennen, mit dem receive~ Objekt empfangen (siehe **Abbildung 5.8**). Für die Verständlichkeit können wir dieses Signal "delayed" Signal nennen.

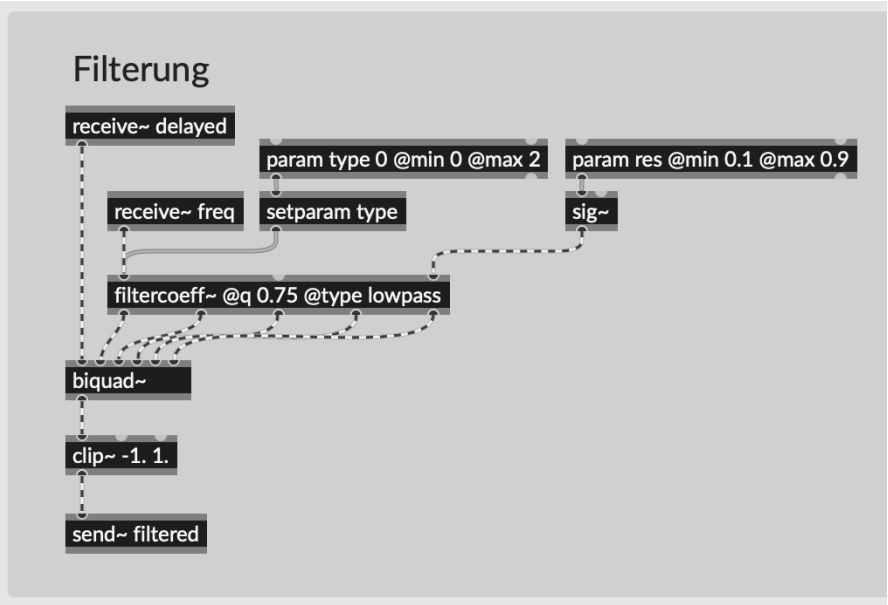

Abbildung 5.8 Filterung im rnbo~ Patch

Gleicherweise wird das auf die Frequenz umgewandelte Steuersignal von der Hüllkurve mit dem receive~ Objekt empfangen. Dies wird für die Verständlichkeit, wie in der Abbildung 5.8 "freq" benannt. Das delayed Signal wird als nächstes in ein "biquad~" Objekt eingeleitet. Das biquad~ Objekt ist die Implementierung eines IIR-Filters zweiter Ordnung in RNBO. Es soll dabei einkommende Signale durch die gegebenen Filterkoeffizienten filtern. Die Filterkoeffizienten, die aus dem Kapitel 4.1.3 schon bekannt sind, werden durch das "filtercoeff~" Objekt mit dem gegebenen freq Wert ausgerechnet. Wegen mangelnder Potis auf dem EG wurden auch hier manche Parameter schon voreingestellt. Der Typ des Filters kann nur im Max Patch durch den Parameter "type" eingestellt werden, wobei null für Tiefpass-, eins für Hochpass- und zwei die Bandpassfilter die Filterkoeffizienten ausrechnet. Für den Fall des haptischen Effektgeräts ist die Standardeinstellung null. Aus dem gleichen Grund wurde die Güte des Filters auf einen Wert von 0.75 voreingestellt. Dieser Wert kann wieder mit dem Parameter "res" durch das Max Objekt verändert werden. Das gefilterte Signal wird im nächsten Schritt durch ein clip~ Objekt zwischen die Werte -1 und 1 begrenzt, um mögliche Übersteuerungen durch die Kombination von Filter und Echo zu vermeiden. Schließlich wird das bearbeitete Signal an den letzten Schritt mit dem send~ Objekt gesendet.

5.1.8 Audio Ausgang

Das Audiosignal wird am Ende dieses Patches ausgegeben. Für die Ausgabe werden zwei “out~” Objekte verwendet, die den Audioausgang von RNBO mit einem verfügbaren Audiogang in der exportierten Entwicklungsumgebung verbinden. Bei dem Hardware Gerät wird der Stereo Ausgang des angeschlossenen USB Audio Interface verwendet. Ein wichtiger Punkt ist dabei, dass bei dem Ausgangssignal zwischen dem unveränderten und veränderten Signal hin und her geschaltet werden kann. Dies wird mit dem “mix~” Objekt und den “onOff” Parameter ermöglicht. Das mix~ Objekt gewichtet die Lautstärke der zwei Eingänge mit dem gegebenen Mischverhältnis Wert von 0 bis 1. Wobei 0 nur den linken Eingang des mix~ Objektes wiedergibt und 1 den rechten. In der Abbildung 5.9 ist der Aufbau des Audio Ausgangs im rnbo~ Patch zu sehen.

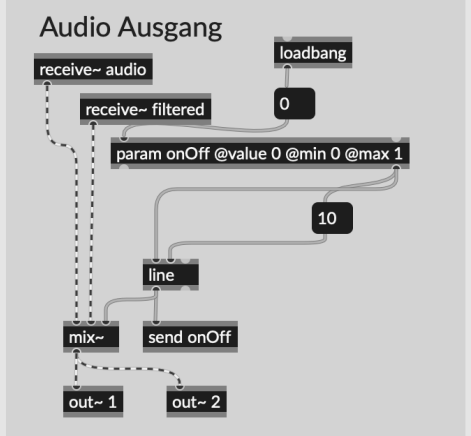

Abbildung 5.9 Audio Ausgang im rnbo~ Patch

5.1.9. Parameter und Open Sound Control (OSC) Nachrichten

In den rnbo~ Patches wurden viele param Objekte erstellt und darauf hingedeutet, dass diese Parameter durch die haptischen Bedienelemente verändert werden können. In diesem Abschnitt wird die Funktionsweise dieser Parameter und das Zusammenspiel mit dem RPI erklärt.

Max benutzt das OSCQuery Proposal[11] für die Kommunikation mit dem RPI, indem ein laufendes Code generiert wird (Cycling’74, o.D.). Somit kann man den auf RPI exportierten RNBO Patch mit Open Sound Control (OSC) Nachrichten ansteuern. Diese Steuerung kann auf verschiedene Weise stattfinden.

[11] OSCQuery-Proposal ist ein Framework. Es wurde dafür geschaffen, den Entwicklern die Erzeugung dynamischer Schnittstellen zu ermöglichen.

Die erste Möglichkeit ist es durch das automatisch erstellte Webinterface. Bei exportieren des Patches auf den RPI wird eine lokale Webseite (auch Webinterface genannt) generiert, auf den man in dem lokalen Netzwerk zugreifen kann. Auf diesem Webinterface erscheinen alle erstellten Parameter nacheinander. Für jeden Parameter wird zusätzlich ein Slider mit der Anzeige des aktuellen Wert des Parameters angezeigt. Die Parameter können durch diesen Slider verändert werden. Dabei werden die bei der Erstellung der Parameter angegebenen Minimal- und Maximalwerte in Betracht gezogen.

Eine weitere Möglichkeit ist es, über den Max Patch OSC Nachrichten zu senden. Dies kann mit einer Message Box und einem "udpsend" Objekt in Max erreicht werden. Bei der Verwendung des udpsend Objekts müssen die IP-Adresse und die Portnummer des Empfängers (in dem Fall der RPI) angegeben werden. Diese Informationen findet man in der Export-Leiste im rnbo~ Patch.

Die in dieser Arbeit verwendete Methode ist, die Parameter über den RPI zu senden. Dafür wird ein Python-Script auf dem RPI erstellt. In diesem Script werden mit der GPIO Zero Bibliothek[12] die an den ADC angeschlossene Potis ausgelesen. Die abgelesenen Werte werden mit Hilfe von einer anderen Bibliothek (python-osc[13]) per OSC an RNBO versendet. Im **Quelltext 5.2** ist dieses Python-Script zu sehen.

```
from gpiozero import MCP3008, Button, PWMLED
from pythonosc.udp_client import SimpleUDPClient
def pressed():
    global onoff
    onoff = 1 - onoff
    target.send_message("/rnbo/inst/0/params/onOff/normalized", onoff)
    led.value = 0.4*(onoff)
target = SimpleUDPClient("192.168.0.103", 1234)
onoff = 0
pot1 = MCP3008(0)
pot2 = MCP3008(1)
pot3 = MCP3008(2)
pot4 = MCP3008(3)
pot5 = MCP3008(4)
but = Button(6)
led = PWMLED(5)
led.value = 0.
but.when_pressed = pressed
while True:
    target.send_message("/rnbo/inst/0/params/potOne/normalized", pot1.value)
    target.send_message("/rnbo/inst/0/params/potTwo/normalized", pot2.value)
    target.send_message("/rnbo/inst/0/params/potThree/normalized", pot3.value)
    target.send_message("/rnbo/inst/0/params/potFour/normalized", pot4.value)
    target.send_message("/rnbo/inst/0/params/potFive/normalized", pot5.value)
```

Quelltext 5.2 Python Script für Potis und OSC Nachrichten

[12] Dies ist eine Bibliothek für die Steuerung der GPIO Pins des RPI.
[13] Eine simple Python-Bibliothek, um OSC Nachrichten über Python zu senden.

Für das Ablesen des Tasters wurde eine ”pressed()” Funktion erstellt (Quelltext 5.2). Diese wird immer aufgerufen, wenn durch den angegebenen GPIO die Taste gedrückt wird. Diese Funktion schaltet den “but_state” Wert zwischen 0 und 1 hin- und her und schließlich sendet eine OSC Nachricht, die den onOff Parameter in RNBO aktualisiert. Ist der but_state 1, dann wird auch die angeschlossene LED auf einen Pulse With Modulation (PWM[14]) Wert von 0.4 gesetzt. Dies hat den Grund, dass die LED zu hell für den normalen Betrieb war.

Um den Script zur Steuerung der RNBO Parameter (Quelltext zu starten, muss eine Secure Shell (SSH[15]) Verbindung mit dem RPI erstellt werden. Ein automatisiertes Starten dieses Scripts ist in der Theorie nach verschiedenen Methoden möglich. Diese Methoden beinhalten entweder die Veränderung der “rc.local” Datei oder die “.bashrc” Datei. Eine weitere Methode, die versucht wurde, ist einen “cronjob” über “crontab” einzulegen. Für diese Arbeit wurde die dritte Methode angewendet. Es wurde ein Befehl zum Starten des Scripts am Ende der rc.local Datei hinzugefügt. Die Idee ist, dass unser Script zusammen mit anderen Initialisierung Schritten ausgeführt wird. Somit wird beim Neustart des RPI der Script automatisch starten.

[14] Eine Methode zur Steuerung der Helligkeit einer LED.
[15] Secure Shell ist eine Verbindungsmethode für nicht gesicherte Netzwerke (Secure Shell, 2023).

5.2. Hardwareentwicklung

Selbstgemachte EGs sind keine Neuheit in der Musikproduktion. Vor allem der Nachbau von berühmten Gitarren-Effektpedalen ist unter Gitarristen sehr beliebt. Im Internet werden vorgefertigte Schaltplatinen für gängige EGs zum selber zusammenlöten verkauft. Zudem existieren zahlreiche Online-Händler, die sich spezifisch auf den Bedarf von Hobby Elektronikern und Bauteilen elektronischer Geräte spezialisiert haben.

Trotz des umfangreichen Sortiments vieler Online-Händler, wurden bei diesem Projekt keine vorgefertigten Platinen benutzt, sondern es wurden die nötigen Platinen selbst entwickelt. Der Grund für diese Entscheidung ist, dass das entwickelte EG auf einem RPI basiert und dementsprechend besondere Anforderungen hat. In der **Abbildung 5.10** werden die Hauptkomponenten des EGs dargestellt.

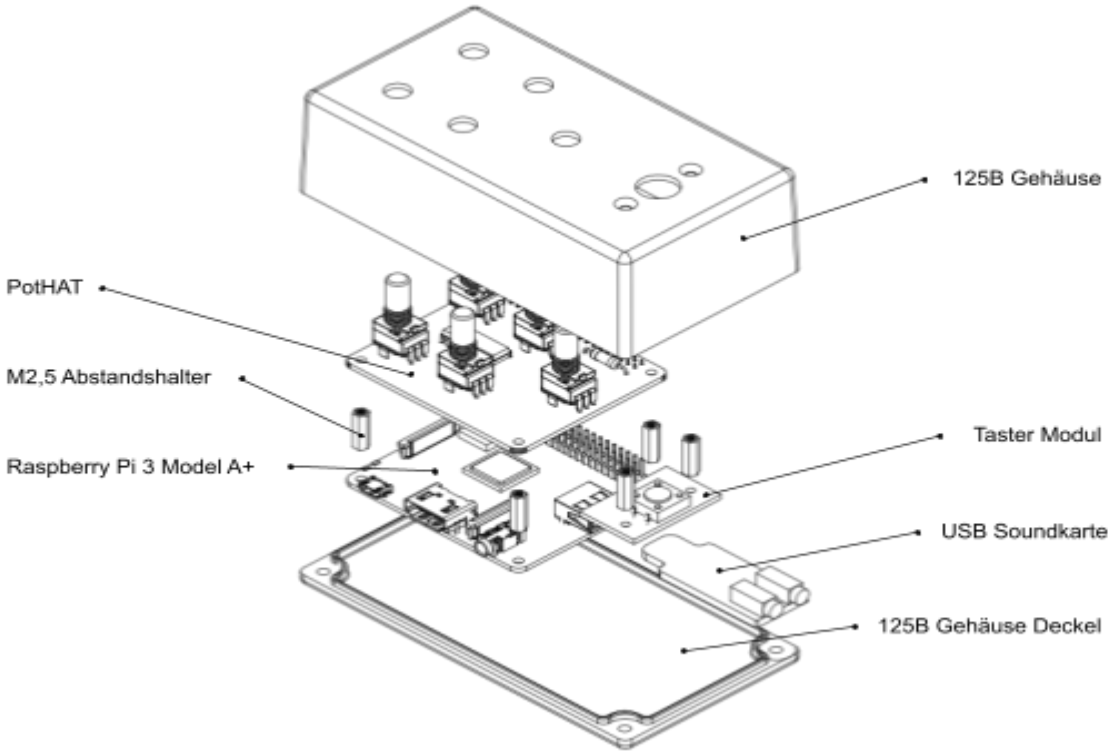

Abbildung 5.10 Hauptkomponenten des Effektgeräts[16]

[16] Bei der Erstellung dieser Abbildung wurde das 3D-Modell von einem RPI aus GrabCAD verwendet (GiantGerman, 2021). Zudem wurde von der gleichen Webseite ein 3D-Modell für die Potentiometer verwendet (Vasily Kashirin, 2021). Für die Darstellung des Tasters wurde ein Modell von dem Hersteller verwendet (Mouser, o.D.).

Der haptische Prototyp des Geräts besteht aus verschiedenen kleineren Komponenten. Jedes kleine Modul übernimmt eine Aufgabe. Die einzelnen Komponenten, die für diese Arbeit benutzt werden, beschrieben und erklärt.

5.2.1. Gehäuse

Ein wichtiger Schritt bei der Produktentwicklung ist das Gehäuse. Es vermitteltn nicht nur den ersten Eindruck über des Produkts, sondern es schützt auch die Elektronik, die sich im Gerät befindet. 3D-Druck, Spritzguss-Kunststoff, CNC gefrästes Aluminium oder sogar Kartonschachteln sind einige Möglichkeiten für das Erstellen eines Gehäuses für elektronische Geräte. Bei dieser Arbeit wurde ein Prototyp erstellt. Da Produktionskosten, Gewinn und Materialkosten eher unrelevant waren, standen viele Methoden zur Auswahl.

Im Bereich des Gitarreneffektbaus werden typischerweise Aluminium-Gussgehäuse verwendet. Diese Gussgehäuse gibt es in verschiedenen Größen und Formen und können nach Bedürfnis gebohrt/gefräst werden. Laut des Verkaufsrangs von musikding.de sind 4 der gängigsten Formen für Aluminium Gehäuse 1590A, 1590B, 1590BB und 125B (Brunner, o.D.). In der **Abbildung 5.11** sind die Formate der Gehäuse im Vergleich zueinander zu sehen. Dabei handelt es sich um ungefähre Skizzen, die nach Angaben von Online-Händlern erstellt wurden.

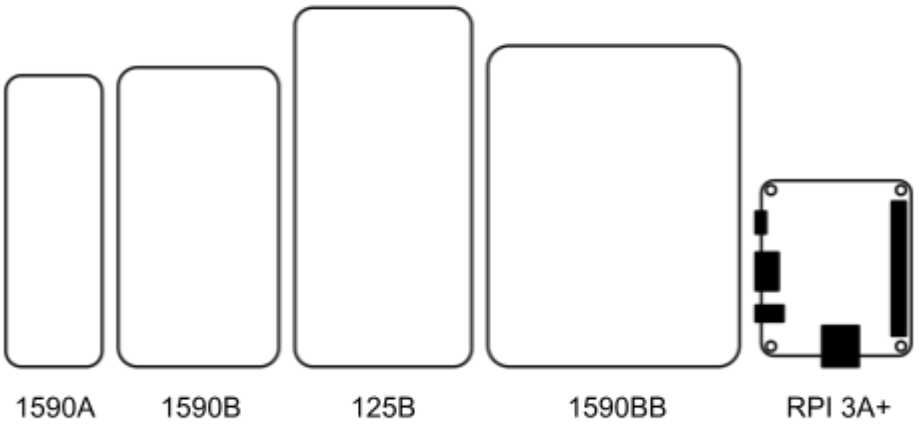

Abbildung 5.11 Gängige Aluminium Gehäusen im Vergleich zu einem RPI 3A+

Die ursprüngliche Idee war es, ein 3D-gedrucktes Gehäuse zu entwickeln. Nach einem Einblick in das Gehäusesortiment wurde klar, dass ein RPI und die benötigte restliche Elektronik für das EG in ein Gehäuse von Typ 125B reinpasst. Nach der Begutachtung verschiedener Aluminiumgehäusen wurde festgestellt, dass diese robuster als Kunststoff ist. Zudem lässt sich die Oberfläche von Aluminium besser verarbeiten, um es später lackieren

und beschriften zu können. Die in der Abbildung 5.1 dargestellten Komponenten sowie eine USB Soundkarte passen in ein 125B Gehäuse. Aus diesem Grund wurde ein Gehäuse vom Typ 125B ausgewählt. Dabei bleibt auch genug Spielraum für den Einbau und für die benötigte Verkabelung übrig. Die benötigten Löcher für den Einbau wurden größtenteils mit einer CNC-Fräse gebohrt. Seitliche Löcher für die DC-Buchse und für die Ein- und Ausgänge der Soundkarte mussten per Hand mit einem Standbohrer gebohrt werden.

Für das Fräsen mit einer CNC Maschine wurde der Mittelpunkt des Aluminiumgehäuse mit Hilfe eines Lineals und einer Schieblehre gemessen. Nach der Befestigung der 125B Gehäuse wurde die Werkzeugspitze des Fräsers an die auf das Gehäuse markierte Position gebracht. Mit einem Stück Papier wurde die Z Position des Werkzeugs fein eingestellt. Schließlich wurden die x-, y- und z-Achsen auf 0 gesetzt und die Fräsarbeit wurde gestartet. Beim Fräsen wurde ein 2mm Fräser mit zwei Flöten verwendet. Die Fräsgeschwindigkeit betrug dabei 60 mm/s. Gefräst wurde mit einer Z Schrittgröße von 0,5 mm. Der Fräser musste also 4 Durchgänge machen, um das 1,8 mm dicke Gehäuse vollständig bohren zu können.

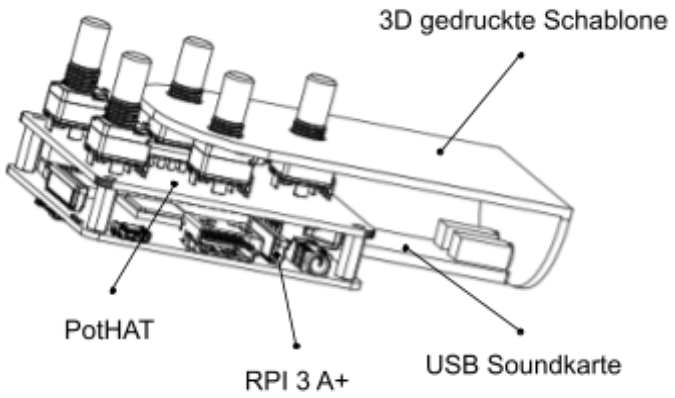

Abbildung 5.12 Schablone für die Bohrungen der Audio Ein- und Ausgänge[17]

Obwohl das Endgehäuse aus Aluminium besteht, wurden für die Bohrung mancher Löcher (Bohrungen für die Ein- und Ausgänge der USB Soundkarte) 3D gedruckte Schablonen benutzt. Grund dafür ist es, eine unnötige Verschwendung von Aluminiumgehäusen durch Fehlbohrungen zu vermeiden. Die Schablonen wurden dafür benutzt, die Koordinaten der Mittelpunkte von den zu bohrenden Löchern zu bestimmen und nicht als Schablone für den Bohrer. Es wurde also nur ein Teil des Gehäuses in einem 3D CAD Programm modelliert und

[17] Bei der Erstellung dieser Abbildung wurde das 3D-Modell von einem RPI aus GrabCAD verwendet (GiantGerman, 2021). Zudem wurde von der gleichen Webseite ein 3D-Modell für die Potentiometer verwendet (Vasily Kashirin, 2021).

additiv hergestellt. In der **Abbildung 5.12** ist die Zeichnung der Schablone zu sehen. Nach der Bestätigung der Koordinaten für die Löcher wurden mit Hilfe einer Schieblehre die Mittelpunkte der Löcher auf das Gehäuse markiert. Für die präzise Messung der Abstände wurde größtenteils die Tiefe-Messfunktion der Schieblehre benutzt.
Gebohrt wurden die Löcher mit einem Standbohrer und einer passenden 5,5 mm Bohrspitze. Das Loch für das Anbringen der DC-Buchse wurde auf die gleiche Weise hergestellt.

5.2.2. Raspberry Pi

Raspberry Pi ist eine Serie von Einplatinen Computern, die von der britischen Stiftung The Raspberry Pi Foundation entwickelt wurde. Sie dient hauptsächlich zur Förderung von Computerprogrammierung, Forschung und Bildung. RPI basiert auf einem SoC (System-on-Chip), der alle Hauptkomponenten des Computers auf einer einzigen Platine integriert. Zusätzlich verfügt der RPI über GPIO Pins, die das Anschließen verschiedener Sensoren und die Steuerung unterschiedlicher Module ermöglichen (Raspberry Pi Foundation, o.D.).

Aufgrund der mühelosen Integration mit der Programmierumgebung Max und RNBO wurde ein Modell der RPI Einplatinencomputer für dieses Projekt gewählt. Das im Umfang dieser Arbeit benutzte Modell ist ein Raspberry Pi 3 Modell A+. Der Grund für die Auswahl dieses Modells ist, dass es eines der kleinsten RPIs ist, das die RNBO-Integration unterstützt und dabei über USB-Typ-A Eingänge verfügt.

Die Hauptaufgabe des RPIs ist es, das von RNBO importierte Programm durchzuführen. Mit diesem Programm kann über das angeschlossene USB Audio Interface das ankommende Audiosignal verarbeitet und durch die DAC des Audio Interfaces wiedergegeben werden. Eine wichtige Herausforderung ist dabei, analoge Sensoren an den RPI anzuschließen und mit der analogen Spannung der Sensoren die Parameter des ausgeführten RNBO-Programms zu verändern. Ein typischer Sensor dieser Art wäre ein Potentiometer (kurz Poti). Um die Werte der Potis auslesen zu können, muss jedoch ein ADC benutzt werden. Der RPI ist aber mit keinem ADC ausgestattet. Man könnte auf die Idee kommen, das im angeschlossenen USB Audio Interface verfügbare ADC zu benutzen. Dieses ist jedoch eine andere Art von Wandler und ist spezifisch für Audios entwickelt worden. Für das Auslesen der analogen Spannung der Potis sind s.g. General-Purpose-ADCs zu benutzen. Dabei muss darauf geachtet werden, dass die benutzten ADC über genug Eingänge verfügen.

Somit kann man die gewünschte Anzahl an Potis an den RPI anschließen. Um die fehlende A/D-Wandler-Fähigkeiten des RPIs zu ersetzen, sind verschiedene Lösungen anwendbar:
Die erste Möglichkeit, analoge Signale in digitale Umzuwandeln, ist, einen Mikrocontroller, wie z.B. einen Arduino, zu benutzen. Gängige Mikrocontroller sind meistens mit ADC ausgestattet und besitzen dadurch mehrere analoge Eingänge für die A/D-Wandlung. Durch eine serielle Datenverbindung zwischen Mikrocontroller und RPI wäre es möglich, die Werte der Potis auszulesen und an den RPI per serielle Verbindung zu übertragen. Ein Nachteil dieser Lösung ist aber, dass man bei der Verwendung eines Mikrocontrollers zwei Geräte separat programmieren muss. Zum einen muss für den RPI ein Script geschrieben werden, zum anderen müsste der Mikrocontroller so programmiert werden, dass er dazu fähig ist, die ausgelesenen Werte an den RPI zu senden. Entscheidet man sich für die Entwicklung einer Erweiterungsplatine, die einen Mikrocontroller besitzt, ist ein weiterer Programmierungsschritt notwendig. Denn so ein Mikrocontroller wird werkseitig blank hergestellt. Er erfordert das Aufladen eines Bootloaders, um über gängige Entwicklungsumgebungen programmiert werden zu können. Angenommen man hat sich für diese Route bei der Entwicklung einer Erweiterungsplatine entschieden, dann müsste man zusätzlich für das Betreiben des Mikrocontrollers weitere Komponenten (wie z.B. Crystal-Oszillatoren, Kondensatoren, Stromversorgung Schaltungen usw.) verwenden.

Eine weitere und in diesem Projekt benutzte Lösung ist es, einen externen Analog-Digital-Wandler-Chip, kurz ADC-Chip, zu verwenden. Diese Lösungsmethode erfordert deutlich weniger Programmierung. ADC Chips werden mit verschiedenen Spezifikationen von einer breiten Menge von Herstellern angeboten und sind somit leichter in der Handhabung. Der bei der Entwicklung dieses Projekts benutzte Chip ist der MCP3008 von der Firma Microchip Technology. Dieser Chip wird häufig in Kombination mit einem RPI genutzt und benötigt keine weiteren Komponenten, wie z.B. Kondensatoren usw.. Dies ist ein großer Vorteil, denn dies erleichtert in einem späteren Zeitpunkt die Entwicklung einer Erweiterungsplatine. Der MCP3008 kann analoge Signale mit einer 10-Bit Auflösung umwandeln und verwendet das Serial Peripheral Interface (SPI[18]) Datenverarbeitungssystem. Der A/D Wandler wird an das RPI angeschlossen, indem er auf dem PotHAT gelötet wird.

[18] Das Serial Peripheral Interface ist eine Datenverarbeitungssystem zur Übertragen von Daten zwischen Mikrocontrollern und Integrated Circuits.

5.2.3. PotHAT

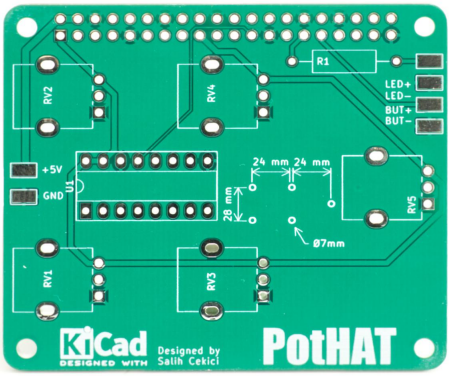

Abbildung 5.13 Eine leere PotHAT Platine

Eine von den zwei Platinen, die bei der Erstellung dieser Arbeit entwickelt wurde, ist der PotHAT. Der Name ist eine Abkürzung von Potentiometer HAT[19]. Der PotHAT soll die Verbindung verschiedener Komponenten an den RPI erleichtern. Diese Platine verfügt über verschiedene Through Hole Technology (THT) Lötstellen. Durch diese THT-Lötpunkte werden verschiedener Hardware-Bedienelemente und spezielle IC-Chips[20] (Integrated Circuit-Chips) mit dem RPI verbunden. In der **Abbildung 5.13** ist eine nicht bestückte PotHAT-Platine zu sehen.

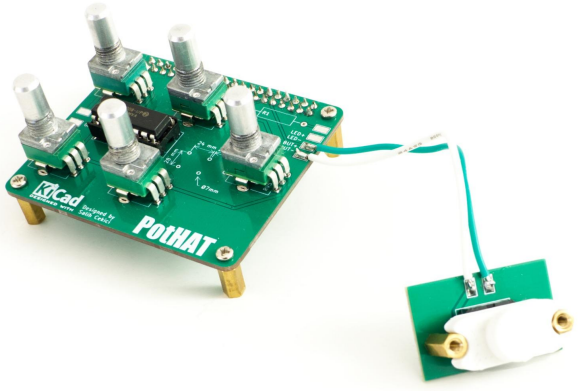

Abbildung 5.14 Fertig bestücktes PotHAT und daran angeschlossenes Taster Modul

[19] HAT steht für Hardware Attached on Top und bezeichnet die Erweiterungen, die für Raspberry Pis erstellt werden (Adams, 2014). HATs sollen den Hardware Entwicklern die Entwicklung von spezifischen Geräten erleichtern. Für viele spezialisierte Anwendungen werden sie auch von kommerziellen Herstellern zum Kauf angeboten.

[20] ICs sind millimeter große elektronische Schaltungen, die aus Halbleitermaterialien hergestellt werden (Integrierter Schaltkreis, 2023).

Wie im vorherigen Kapitel beschrieben, verfügt der RPi über keinen ADC. Dieses Problem sowie viele weitere Probleme bei der Entwicklung des Effektgeräts wird dank des PotHATs beseitigt. Zum einen dient er als Halter für fast die gesamte Elektronik im Gehäuse, zum anderen wird er mit dem ADC-Chip MCP3008 bestückt und ermöglicht dadurch das Ablesen der Potentiometerwerte mit dem RPI.

Ein weiterer Vorteil dieser Platine ist, dass es über die Lötpunkte, die er verfügt, die Verbindung zusätzlicher Komponenten (wie z.B. das Taster-Modul aus dem Kapitel 5.1.5 und einer Status LED aus dem Kapitel 5.1.6) an den RPI erleichtert. In der **Abbildung 5.14** ist ein Foto des fertig bestückten PotHAT Platine zu sehen. Darüber hinaus ist in der Abbildung 5.14 auch deutlich zu sehen, wie einer der erwähnten zusätzlichen Komponenten an das PotHAT mit einem Kabel angebunden wurde. Detaillierte Informationen über das angebundene Taster Modul sind im Kapitel 5.1.5 zu finden.

Konzipierung des PotHATs

Die Idee der Platine entstand bei der Probe der RNBO-Erweiterung mit dem RPI. Auf einem Breadboard wurden ein MCP3008 ADC-Chip und drei lineare 10kΩ Potentiometer platziert. Die Potis wurden an die drei von acht verfügbaren Eingängen des ADC-Chips mit Überbrückungskabel angebunden. Schließlich wurde der ADC mit einem selbst gecrimpten Dupont Überbrückungskabel direkt an die GPIO Pins des RPIs verbunden. Im Weiteren ist in der **Abbildung 5.15** ein Foto von diesem Aufbau zu sehen.

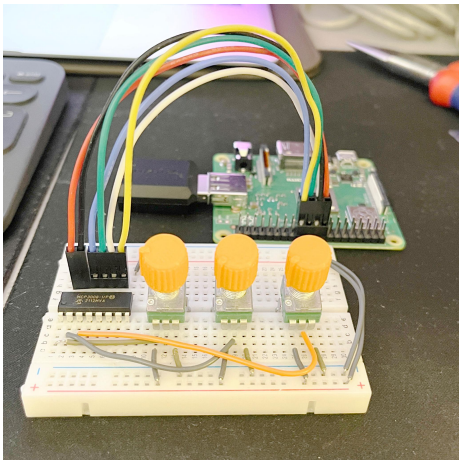

Abbildung 5.15 Breadboard Prototyp des PotHATs

Es wurde herausgefunden, dass der Prototyp die Erwartungen erfüllt und mit dem Probecode aus der RNBO Dokumentation (Cycling ‘74, o.D.) funktioniert. Daraufhin wurde angefangen, eine Platine zu konstruieren, die diese Komponenten durch Lötverbindungen sicher an einer Stelle hält. Die Platine soll dabei über so viel wie möglich Lötstellen für Potis verfügen. Damit wird erzielt, so viel wie möglich Potentiometer an den RPI zu verbinden. Denn die auf der Platine angebrachten Potis haben eine weitere Aufgabe:

Die Potis verfügen über ein M7 Gewinde an ihrem Gehäuse. Dadurch ist es möglich, sie mit den mitgelieferten M7 Unterlegscheiben und M7 Muttern an diversen Löchern zu befestigen. Die Potis sollten also den PotHAT festhalten und der PotHAT den RPI. Für eine gleichmäßige Gewichtsverteilung sollten mindestens vier Potis angebracht werden. Dies würde allerdings vermutlich für die Steuerung vieler Effekte nicht ausreichend sein. Deswegen wurde eine fünfte Poti Stelle mittig unter den vier Potis hinzugefügt. Dadurch hat sich das PCB-Layout wie in der **Abbildung 5.16** ergeben:

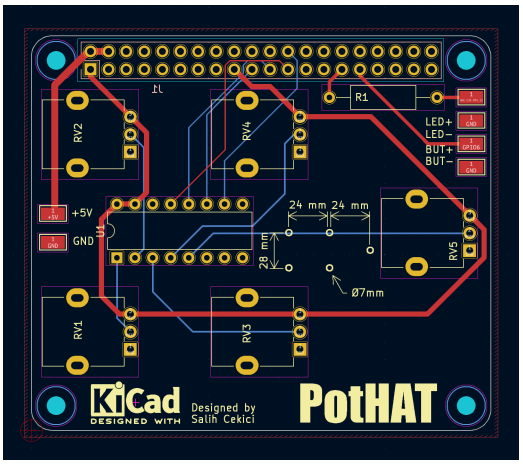

Abbildung 5.16 PCB Layout des PotHATs in Kicad

Die Platine wurde mit der Software Kicad entwickelt. Die Entwicklung des PCBs von PotHAT und des Tasten Moduls werden im nächsten Abschnitt im Detail behandelt.

5.2.4. Platinenherstellung

Konstruiert wurden die Platinen mit einer Electronic-Design-Automation Software (kurz EDA Software) Kicad. Im ersten Schritt wurde ein Schaltplan für die Platine erstellt. Bei der Erstellung des Schaltplans hat man sich an dem Breadboard Prototyp des Gerätes (wie in der Abbildung 5.15 dargestellt) orientiert. Darauf aufbauend wurden schon bei der Erstellung

des Prototyps Gedanken darüber gemacht, welche Komponenten später auf der Platine angebracht werden. In dem digitalen Anhang stehen die Fertigungsdateien sowie die Schaltpläne der Platine zum selber bestellen oder verbessern zur Verfügung. Sie sind unter dem Ordner PotHAT zu finden.

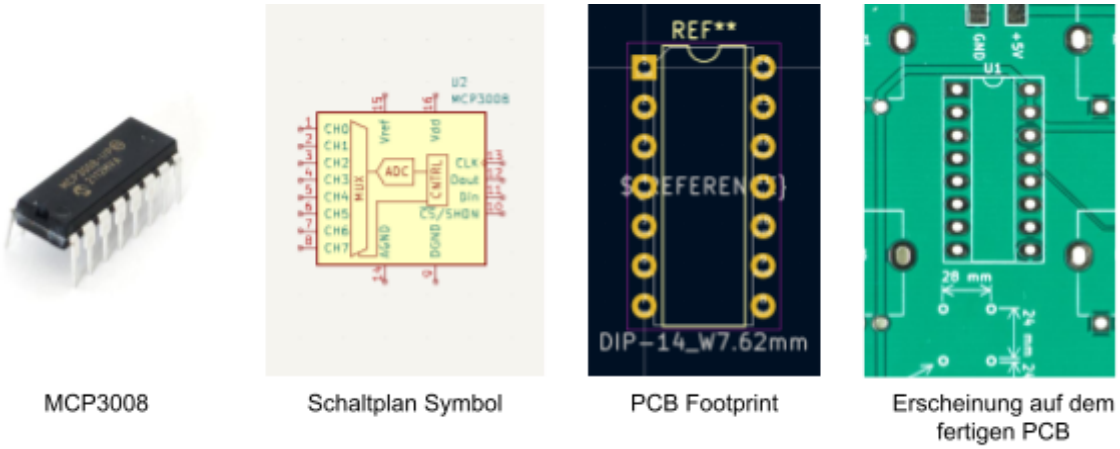

Abbildung 5.17 MCP3008 und seine Erscheinungen

EDAs kommen mit einer breiten Sammlung von vorgezeichneten PCB Footprints und Schaltsymbolen. Trotzdem haben EDAs nicht alle Footprints und Schaltsymbole von den elektronischen Komponenten, die auf dem Markt angeboten werden. In dem Fall müssen diese Zeichnungen selber erstellt werden. Namhafte Hersteller bieten diese Zeichnungen ihrer Produkte auf ihrer Webseite zum Herunterladen an. Um die benötigte Zeit für das Erstellen von neuen Schaltplan Symbolen oder das Herunterladen/Einbinden von Footprint Bibliotheken zu minimieren, wurde versucht, immer in Kicad verfügbare Komponente zu verwenden.

Nach dem Erstellen des Schaltplans müssen an alle Schaltsymbole Footprints zugewiesen werden. Diese Footprints bestimmen dann, wie der Platinenhersteller die Platine anfertigen muss. **Abbildung 5.17** zeigt für das Beispiel eines MCP3008 Chips wie seine eigentliche Erscheinung im Vergleich zu seinem Schaltplan Symbol und PCB Footprint aussehen.

5.2.5. Taster Modul

Die meisten EGs verfügen über mindestens einen Drucktaster. Mit dem Drucktaster kann man den Effekt um nützliche Funktionalitäten erweitern. Man kann das Gerät über den Taster ein- und ausschalten, für Tempo basierte Effekte das Tempo bestimmen oder im Fall eines Looper Effekts Parameter, wie den Start-Stop Moment, des Loops einstellen. Was auch

immer der Zweck des Tasters ist, ist er von EGs nicht wegzudenken und ist sicherlich eine nützliche Funktion. Auch bei diesem EG sollte also ein Drucktaster nicht fehlen.

Anstatt einen fertigen Fußschalter um die 2-3€ zu erwerben und dies mit benötigten Kabeln an das PotHAT zu löten, wurde eine eigene Platine für die Taster entwickelt. Diese Platine soll dabei einen generischen 12x12mm taktilen Taster festhalten. Zusätzlich befinden sich auf der Platine zwei Lötpads, um die Verbindung des Taster Moduls an einem der GPIO Pins des RPIs zu vereinfachen. Dementsprechend wurden auch auf dem PotHAT zwei Lötpads angebracht. Diese Pads sind mit dem GPIO Pin 6 des RPIs verbunden.

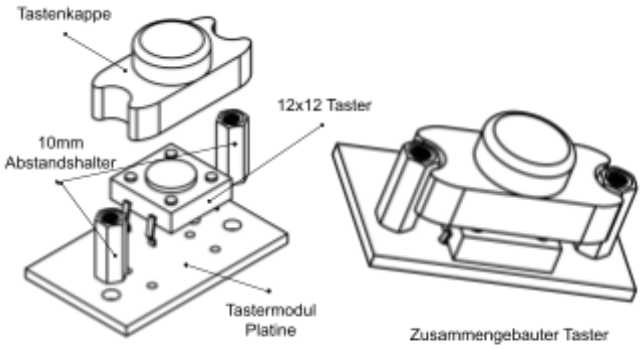

Abbildung 5.18 Aufbau des Tasters[21]

Das ganze Modul wird mit den angebrachten Löchern und einem M2,5x10mm Abstandshalter an das Gehäuse festgeschraubt. Der Benutzer soll aber den 12mm Taster nicht direkt betätigen, sondern über die 3D-gedruckte Tastenkappe. In der **Abbildung 5.18** ist der theoretische Aufbau des Tasters zu sehen. In einer weiteren Version wurde die Tastenkappe mit einer Hülle für eine LED neu konstruiert. Somit ist er dazu fähig geworden, eine gängige 5mm LED festzuhalten. Dadurch wurde die Status LED innerhalb der Tastenkappe versteckt. Die neue Konstruktion der Tastenkappe wurde jedoch mit einem anderen Material additiv hergestellt. Bei diesem Material handelte es sich um eine transparente PLA Filament Sorte. Dies ermöglicht am Ende des Zusammenbaus eine bessere Sichtbarkeit der Status LED. In der **Abbildung 5.19** ist der Zusammengebauter Taster mit der aktiven Status LED an dem Gehäuse des fertigen Geräts zu sehen.

[21] Bei der Erstellung dieser Abbildung wurde ein 3D Model von dem Hersteller verwendet (Mouser, o.D.).

Abbildung 5.19 Fertiges EG

5.2.6. Status LED

Bei der Arbeit mit einem RPI ist es möglich, dass die Fehlersuche schnell kompliziert wird. Zudem hat der RPI eine bemerkenswerte Startzeit. Die "Startzeit" bezieht sich auf die Zeitspanne, die erforderlich ist, damit der RPI nach dem Einschalten vollständig hochgefahren ist und das RNBO-Programm ausführen kann. Diese Dauer umfasst sämtliche Prozesse und Initialisierungen, die der RPI durchläuft, bevor er in der Lage ist, das RNBO-Programm auszuführen.

Um auf die Betriebsbereitschaft des EGs hinweisen zu können und gleichzeitig Informationen über den Status des Effekts bereitzustellen, wurde ein detailliertes Konzept entwickelt. In diesem Konzept war vorgesehen, das EG mit einer Status-LED auszustatten. Diese LED soll nicht nur den Bereitschaftsstatus des EG anzeigen, sondern auch darauf hinweisen, ob der Effekt aktiviert ist oder nicht. Dank dieser erweiterten Funktion können Benutzer auf einen Blick feststellen, in welchem Modus sich das EG befindet.

Die Verbindung dieser LED hat mit Hilfe der an das PotHAT angebrachten Lötstellen stattgefunden. Dadurch soll die elektrische Verbindung an das RPI gewährleistet werden. Befestigt wurde die LED an der Tastenkappe des Tasters aus dem Kapitel 5.1.5. Dies erforderte jedoch die Umkonstruktion der Tastenkappe. Zusätzlich musste man die Tastenkappe mit einem lichtdurchlässigen Material erneut erstellen, damit die LED sichtbar bleibt.

5.2.7. Potentiometer Kappen

An das PotHAT angelöteten Potis werden werkseitig ohne Kappe geliefert. Ohne einen passenden Knopf, ist die Bedienung der Potis in der Regel sehr schwer und erfordert einen hohen Kraftaufwand. Es ist also üblich, dass für diese Bedienelemente zusätzlich eine Kappe geschaffen wird.

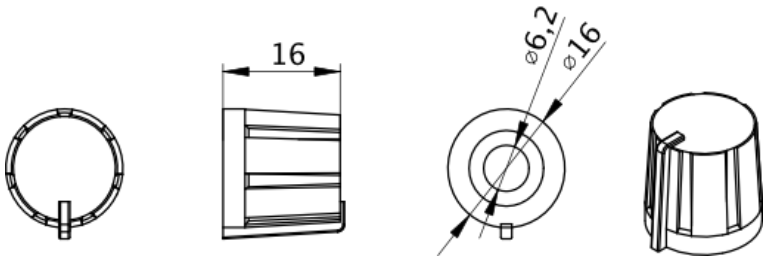

Abbildung 5.20 Potikappe 1

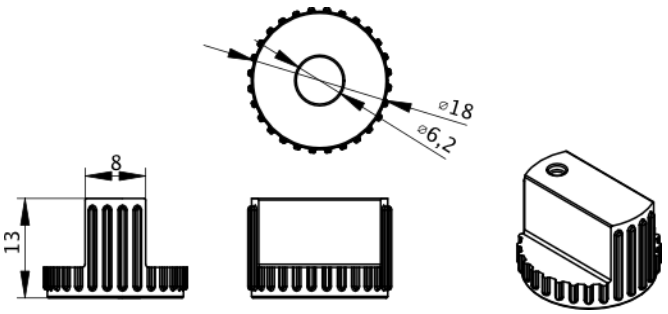

Abbildung 5.21 Potikappe 2

Für die Verbesserung der Bedienbarkeit und eine einzigartige Erscheinung wurden für dieses Gerät eigene Potikappen entworfen. Die in FreeCAD[22] konstruierte Kappen wurden durch Fused Deposition Modeling (FDM) Verfahren hergestellt. Für die Herstellung wurde das Material Filaflex 82A TPU von der Firma Recreus verwendet. Bei diesem Material handelt es sich um ein sehr flexibles Filament, das laut Recreus (o.D.) eine Bruchdehnung bis zu 650% erreichen kann. Der Grund für die Auswahl des Materials ist, dass es ein rutschfestes Greifen ermöglicht. Ein weiterer Grund ist, dass TPU dank seiner leichten Dehnbarkeit Presspassung mit der Hand ermöglicht. Durch die Konstruktion eines leicht kleineren Lochs (als der Durchmesser von dem Schaft des Potis) an der Kappe wird ein rutschfestes Halt an den Schaft des Potis gewährleistet. In der **Abbildung 5.20** und **Abbildung 5.21** sind die technischen Zeichnungen der Kappen zu sehen.

[22] FreeCAD ist eine Open-Source 3D CAD Software. Sie ermöglicht die parametrische Erstellung von 3D Modellen.

6. Test des Effektgeräts

6.1. Allgemeiner Test

Während des Zusammenbaus und der Entwicklung wurde das Gerät mehrmals auf Funktionen getestet. Getestet wurde dabei, ob die Patches auf den RPI übertragen werden, die Funktionen des PotHATs, die Status LED usw.. Diese sind in den meisten Fällen wie erwartet gelaufen. Man konnte die Parameter durch den PotHAT verändern und den Betriebsmodus mit der Status LED anzeigen. Diese Funktionen waren auch an dem ausgegebenen Audiosignal zu hören. In dieser Hinsicht war der PotHAT sowie der Tasten Modul ein Erfolg.

Nach dem Zusammenbau des Effektgeräts sowie des Effektalgorithmus sollte die Funktion des RNBO-Patches im Max mit dem haptischen Gerät systematisch verglichen werden. An diesem Punkt kam es zu Fehlern bei der Ausgabe und dem allgemeinen Verhalten des Geräts (siehe Kapitel 6.4 für weitere Informationen). Nach einer Fehlersuche wurde vermutet, dass die eingebaute USB-Soundkarte bei der Initialisierung von RNBO nicht richtig erkannt wird. Daraufhin wurde das Effektgerät aus seinem Aluminiumgehäuse ausgebaut. Weil die in dem Gehäuse eingebaute Soundkarte mit RNBO instabil läuft, wurde eine andere Soundkarte angeschlossen. Mit einer neuen Soundkarte und in dem ausgebauten Zustand konnte somit ein Test durchgeführt werden.

6.2. Aufbau des Tests

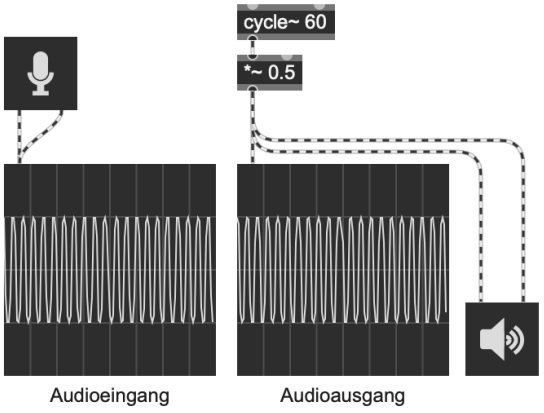

Abbildung 6.1 Pegelanpassung am AUI

Im ersten Schritt wurde in Max ein Sinus Signal vom Ausgang des AUI abgespielt. Der Ausgang des AUIs wurde direkt an den Eingang des AUIs angeschlossen und die Pegelwerte wurden angepasst. In der **Abbildung 6.1** ist eine Bildschirmaufnahme für den Moment des angepassten Pegels zu sehen.

Ohne die Werte des Eingang- und Ausgangspegel zu verändern, wurde das gleiche Sinus-Tonsignal durch den RPI geschickt. Da die Lautstärke des RPIs im Alsa-Mixer zu niedrig eingestellt war, wurde der Pegel in diesem Schritt angepasst. Somit ist der Pegelunterschied der zwei Systeme eliminiert worden.

6.3. Der Test

Nach der Pegelanpassung am AUI wurde der Testaufbau für die Audiobeispiele umgebaut (siehe Anhang A: /Max und RNBO Patches/TestPatch.maxpat). Im Test wurde eine Audiodatei (Anhang A: Soundbeispiele/kalimba_original.wav) mit bestimmten Parametern durch den rnbo~ Patch bearbeitet und aufgenommen. Anschließend wurde die gleiche Datei mit den gleichen Parametern über den RPI bearbeitet und aufgenommen. Bei diesem Test wurde herausgefunden, dass die Ergebnisse beider Aufnahmen sehr ähnlich sind. Aufgefallen ist dabei nur, dass die Audiosignale im haptischen Gerät etwas leiser erscheinen. Dies beeinflusst die Steuerwerte des Envelope-Followers und somit das Ergebnis des Filters. Vermutlich könnte es auch an der Mono-Verarbeitung des Signals liegen. Im Allgemeinen lässt sich jedoch sagen, dass das Effektgerät das Ziel, wofür er geschaffen wurde, erfüllt. Nämlich experimentieren und testen.

Die unveränderte Audiodatei zusammen mit den anderen Beispielen, die durch die Software und das Gerät verändert wurden, sind im Anhang A: /Soundbeispiele zu finden. Die Datei “kalimba_original.wav” ist dabei das unveränderte Signal. Die im Max durch den rnbo~ Patch bearbeitete Datei ist unter dem Namen “kalimba_rnbo.wav” im digitalen Anhang zu finden (Anhang A: /Soundbeispiele/kalimba_rnbo.wav). Die Datei, die durch den RPI bearbeitet und anschließend aus dem Ausgang des RPI aufgenommen wurde, ist “kalimba_rpi_hardware.wav” (Anhang A: /Soundbeispiele/kalimba_rpi_hardware.wav).

Um den Effekt auszuprobieren, kann das beigefügte Max Patch verwendet werden (Anhang A: /Max und RNBO Patches/MeinEffekt.maxpat). In diesem Patch befindet sich auch die RNBO Datei. Dies kann auf ein RPI exportiert werden.

6.5. Anmerkungen

Während der Tests nach üblichen Funktionen des Effektgeräts sind im haptischen Gerät von Zeit zu Zeit einige Instabilitäten aufgetreten.

Soundkarte:

Nach der Erstellung eines Patches in RNBO taucht der RPI in der Export-Leiste auf. Den erstellten Patch kann man nach essentiellen Einstellungen (Soundkarte auswählen, Abtastrate auswählen usw.) an den RPI senden. An diesem Moment läuft der Sketch wie erwartet. Wird der RPI jedoch ausgeschaltet und wieder ein, ist kein Effekt zu hören. Auch der Audioausgang scheint nicht zu funktionieren. Nach einer Suche von Fehlern wurde vermutet, dass der Jack-Server, der für den Audio-Ein- und Ausgang zuständig ist, nicht richtig gestartet wird. Verschiedene Versuche, den Server beim Hochfahren des RPI mit einem Script zu starten, sind gescheitert. Man musste immer zuerst eine SSH-Verbindung mit dem RPI herstellen, um den Jack-Server starten zu können. Es stellte sich heraus, dass die meisten Probleme an der verwendeten und in das Gehäuse eingebauten Soundkarte lag. Durch eine lose USB Verbindung und ein nicht fester Sitz am Gehäuse hat die USB Soundkarte Audioausfälle verursacht. Für künftige Entwicklungen sollen Soundkarten von unbekannten Herstellern vermieden werden.

Hochfahrzeit:

Es ist wichtig zu erwähnen, dass der RPI 3 Modell A+ ≈37 Sekunden brauchte, um hochzufahren und die benötigten Initialisierungen durchzuführen. Gemessen wurde diese Zeit, indem ein einfaches Patch auf einem RPI exportiert wurde. Dieser Patch soll nur ein Sinussignal an den Ausgängen ausgeben. Mit dem Anschließen des RPI an einer Stromquelle wird die Stoppuhr gestartet. Die Stoppuhr wird an dem ersten Moment der Erscheinung des Signals gestoppt. Nach drei Versuchen wurde der Mittelwert von allen Versuchen gebildet.

7. Fazit

Ziel dieser Arbeit war es, ein konfigurierbares Audio Effektgerät zu entwickeln und damit verbundene Fragen zu beantworten. In dieser Hinsicht wurde das Ziel erreicht. Es ist ein Audio Effektgerät entstanden, das in der Lage ist, in einer bekannten Audio-Entwicklungsumgebung erzeugte Patches abzuspielen. Zudem verfügt das Gerät über ausreichende Bedienelemente, um die erzeugten Audio Effekte bedienen zu können.

Bei der Entwicklung des Geräts wird die relativ neue Entwicklungsumgebung RNBO von Cycling ‘74 verwendet. Es ist eine vielfältige Entwicklungsumgebung, die den Benutzern unendliche Möglichkeiten gibt, ihre Kreativität zu entfalten. In Kombination mit dem entwickelten Effektgerät kann der Benutzer seinen Patch in wenigen Sekunden mit einem haptischen Gerät ausprobieren. Zudem ermöglicht das Effektgerät die Änderung der Audioeffektparameter durch die angebrachten Potentiometern und Taster.

Um das Effektgerät auszuprobieren, wurden Recherchen über Audioeffektalgorithmen durchgeführt. Nach dieser Recherche wurde ein Effekt bestehend aus einer Kombination von Echo und Filter entwickelt. Dieser Effekt wurde anschließend mit Audiodateien getestet und mit dem rnbo~ Patch verglichen, in dem er entwickelt wurde. Der Audioeffekt funktioniert wie erwartet. Bei dem Ausgang des haptischen Geräts sind jedoch einige Abweichungen zum rnbo~ Patch festgestellt worden. Die über dem haptischen Gerät laufenden Effekte sind denen von rnbo~ sehr ähnlich, sind aber nicht zu 100 % gleich. Außerdem benötigt das auf dem RPI basierende Gerät eine Hochfahrzeit von ≈35 Sekunden, die für manche Anwendungen eine Herausforderung darstellen könnte. Es darf auch nicht unerwähnt bleiben, dass einige Instabilitäten bei der Verwendung des haptischen EG erlebt wurden.

Nichtsdestotrotz lässt sich sagen, dass das entwickelte Audio-Effektgerät sich gut in den Musikproduktionsalltag integrieren lässt. Es wird vielen Audioenthusiasten erlauben, die Möglichkeiten der Audioentwicklung zu erkunden und durch Experimentieren neue Effekte zu entwickeln.

Literaturverzeichnis

Ableton. (o.D.). I*nklusion im Internet: Ableton Knowledge Base: How to avoid crackles and audio dropouts*. Abgerufen am 15.10.2023, von https://help.ableton.com/hc/en-us/articles/209070329-How-to-avoid-crackles-and-audio-dropouts

Adams, J. (2014, 31. Juli). *Inklusion im Internet: Introducing Raspberry Pi HATs*. raspberrypi. Abgerufen am 15.09.2023, von https://www.raspberrypi.com/news/introducing-raspberry-pi-hats/

Berry Base. (o.D.). *Inklusion im Internet: Dev Boards: Teensy*. Abgerufen am 10.10.2023, von https://www.berrybase.de/dev.-boards/teensy/

Bracken, B. (2022). *Inklusion im Internet: Max 8.5 Released With Support for RNBO*. cycling74. Abgerufen am 25.08.2023, von https://cycling74.com/forums/max-8-5-released-with-rnbo

Brunner, K. (o.D.). *Inklusion im Internet: Aluminium Gehäuse Natur*. musikding. Abgerufen am 06.09.2023, von https://www.musikding.de/Aluminium-Gehaeuse-Natur

Cipriani, A. & Giri, M. (2019). *Electronic Music and Sound Design: Theory and Practice with Max 8 - Volume 1*. [Apple Bücher]. Contemponet s.a.s.

Cipriani, A. & Giri, M. (2020). *Electronic Music and Sound Design: Theory and Practice with Max 8 - Volume 2*. [Apple Bücher]. Contemponet s.a.s.

Cycling ‘74. (o.D.). *Inklusion im Internet: Max 8: Documentation*. Abgerufen am 24.08.2023, von https://docs.cycling74.com/max8

Cycling ‘74/Max. (o.D.). *Inklusion im Internet: Max: Shop*. Abgerufen am 10.10.2023, von https://cycling74.com/shop/max

Cycling ‘74/RNBO. (o.D.). I*nklusion im Internet: RNBO: Shop*. Abgerufen am 10.10.2023, von https://cycling74.com/shop/rnbo

Electro-Smith. (o.D.). *Inklusion im Internet: Daisy: Daisy Seed*. Abgerufen am 10.10.2023, von https://www.electro-smith.com/daisy/daisy

FFmpeg. (o.D). *Inklusion im Internet: FFmpeg: About*. Abgerufen am 13.10.2023, von https://ffmpeg.org/about.html

GiantGerman. (2021, 25. Dezember). *Inklusion im Internet: Raspberry Pi Model 3 A+*. GrabCAD. Abgerufen am 15.09.2023, von https://grabcad.com/library/raspberry-pi-model-3-a-1

Integrierter Schaltkreis.(2023, 12. Juli). In *Wikipedia*. https://de.wikipedia.org/wiki/Integrierter_Schaltkreis

Interpixel. (2021, 18. Januar). *How to convert an Image Sequence into a Sprite sheet W/Blender & FFmpeg | InterPixel* [Video]. YouTube. https://youtu.be/JRjyN3FdseQ?si=HGT3UAz4MzmEtLlX

Maempel, HJ., Weinzierl, S., Kaminski, P. (2008). Audiobearbeitung. In S. Weinzierl (Hrsg.), *Handbuch der Audiotechnik* (S. 719-784). Springer. https://doi.org/10.1007/978-3-540-34301-1_13

Mouser. (o.D.). Inklusion im Internet: Omron B3F-4000. Abgerufen am 17.10.2023, von https://www.mouser.de/ProductDetail/Omron-Electronics/B3F-4000?qs=B3tblJ0Nlt8c5sbFRctxww%3D%3D

Pure Data. (o.D.). *Inklusion im Internet: Pure Data: Startseite*. Abgerufen am 01.09.2023, von https://puredata.info/

PJRC. (o.D.). I*nklusion im Internet: Teensy USB Development Board: Main Page*. Abgerufen am 10.10.2023, von https://www.pjrc.com/teensy/

Raspberry Pi Foundation. (o.D.). *Inklusion im Internet: Raspberry Pi Foundation: About us*. Abgerufen am 01.09.2023, von https://www.raspberrypi.org/about/

Recreus. (o.D.). *Inklusion im Internet: Filaflex 82A*. Abgerufen 13.10.2023, von https://recreus.com/de/filamente/9-684-filaflex-82a.html#/1-farbe-schwarz/2-durchmesser-175_mm/3-gewicht-500_gr

Secure Shell. (2023, 30. Juli). In *Wikipedia*. https://de.wikipedia.org/wiki/Secure_Shell
Schneidersladen. (o.D.). *Inklusion im Internet: Electrosmith - Daisy Seed*. Abgerufen am 10.10.2023, von https://schneidersladen.de/en/electrosmith-daisy-seed

Steppat, M. (2014). *Audioprogrammierung: Klangsynthese, Bearbeitung, Sounddesign*. Carl Hanser Verlag GmbH & Co. KG.

Stoffregen, P. (o.D.). *Inklusion im Internet: Paul Stoffregen*. Abgerufen am 10.10.2023, von https://www.linkedin.com/in/paulstoffregen/

Vasily Kashirin. (2021, 8. März). Inklusion im Internet: Alpha RD901F-40-15R1. GrabCAD. Abgerufen am 15.09.2023, von https://grabcad.com/library/alpha-rd901f-40-15r1-1

Zölzer, U. (2011). *DAFX: Digital Audio Effects* (2. Aufl.). Wiley.

Anhang

Anhang A:

Digitaler Anhang des gesamten Projekts im beigefügten USB-Stick

/3D Dateien/

Effektgeraetknopf_1.stl

Effektgeraetknopf_2.stl

EffektgeraetTaster.stl

EffektgeraetTasterLED.stl

/PotHAT PCB Dateien/

PotHAT.kicad_pro

PotHAT.kicad_sch

PotHAT.kicad_pcb

/Taster Modul PCB Dateien/

ButtonModulePanel_v1.kicad_pro

ButtonModulePanel_v1.kicad_sch

ButtonModulePanel_v1.kicad_pcb

/Max und RNBO Patches/

MeinEffekt.maxpat

TestPatch.maxpat

effekt_transparent.png

PotSheet1.png

PotSheet2.png

PotSheet3.png

PotSheet4.png

PotSheet5.png

button.png

/Soundbeispiele/

kalimba_original.wav

kalimba_rnbo_patch.wav

kalimba_rpi_hardware.wav