HAW HAMBURG

BACHELORARBEIT

Echtzeitfähige Realisierung eines statischen Rotations-Lautsprecher-systems

vorgelegt am 19. September 2022
Daniel Friesen

Erstprüferin: Prof. Dr. Eva Wilk
Zweitprüfer: Prof. Dr. Jan Mietzner

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Department Medientechnik
Finkenau 35
20081 Hamburg

Zusammenfassung

Ziel dieser Arbeit ist die Konzeption eines statischen Lautsprechersystems, das den Klang eines Leslie-Rotationslautsprechers mithilfe digitaler Signalverarbeitung bestmöglich nachbildet. Dabei soll die Latenz des Systems nicht mehr als 10 ms betragen und somit den echtzeitfähigen Einsatz in Live-Situationen ermöglichen.

Der Rotationslautsprecher wird historisch eingeordnet, seine heutige Relevanz aufgezeigt und hinsichtlich seines Aufbaus und seiner Klangeigenschaften untersucht und die Besonderheiten herausgearbeitet. Die Simulation wird umgesetzt, indem das Nutzsignal entsprechend moduliert und auf stationäre Lautsprecher verteilt wird. Dafür werden existierende Modelle hinsichtlich der genannten Vorgaben untersucht und miteinander verglichen. Die Umsetzung erfolgt mithilfe eines passenden Audio-Entwicklungsboards und aktiven Lautsprechern, deren Auswahl im Rahmen dieser Arbeit evaluiert wird. Die Ergebnisse werden anschließend messtechnisch untersucht und durch Hörversuche bestätigt.

Abstract

This thesis discusses the conception of a loudspeaker array, that reproduces the sound of a Leslie rotary speaker using digital signal processing and stationary loudspeakers. The latency of the system is required to be less than 10 ms to be useful in real time live situations.

The rotary speaker is classified historically, and it is shown that it is still relevant today. The system is investigated regarding its physical structure and the sound it produces. The simulation is achieved by modulating the input signal accordingly and distributing it to appropriate loudspeakers. Existing models are being analysed and compared in regard of those requirements. The implementation is carried out on an audio development board and active loudspeakers the choice of which is evaluated in this thesis. The results are then analysed metrologically and confirmed with acoustic experiments.

Inhaltsverzeichnis

Abkürzungsverzeichnis

FIR	Finite Impulse Response
IIR	Infinite Impulse Response
LUT	Look-Up Table
DUT	Device under test
MLS	Maximum length sequence
rpm	rounds per minute

Abbildungsverzeichnis

Codeverzeichnis

Anhangsverzeichnis

1 Einleitung

Der Leslie-Lautsprecher ist ein ikonisches Effektgerät, welches auf zahlreichen, genreübergreifenden Musikaufnahmen – meistens zusammen mit einer elektronischen Orgel – zu hören ist. Ob bewusst oder unbewusst, viele Menschen haben den Klang des Rotationslautsprechers schon einmal wahrgenommen.

Dabei handelt es sich um einen Effekt, der durch mechanische Bewegung erzeugt wird. Deshalb setzt sich sein Klang aus mehreren akustischen Phänomenen zusammen. Durch diese Kombination entsteht eine auffällige Charakteristik, die bis heute als stilistisches Mittel in der Musikproduktion eingesetzt wird.

Aufgrund der aufwändigen Konstruktion ist die Nutzung solcher mechanischen Effektgeräte allerdings mit hohen Kosten verbunden. Das wirft die Frage auf, ob der heutige Stand der Technik eine Simulation ermöglicht, die den musikalischen Anforderungen des Originals gerecht wird und somit die Zugänglichkeit zu diesem Klang durch niedrigere Anschaffungskosten erhöht werden kann. Dabei soll der Aspekt des Echtzeitbetriebes im Vordergrund stehen, um den Einsatz in Live-Situationen zu ermöglichen.

Das System soll aus zwei wesentlichen Teilen bestehen. Zum einen sollen die akustischen Phänomene eines Leslie-Lautsprechers digital mithilfe geeigneter Hardware unter Echtzeitanforderungen simuliert werden. Zum anderen soll das simulierte Signal über eine Beschallungsanlage wiedergegeben werden, die die Authentizität des Effekts unterstützt. Dadurch soll das Konzept eines Systems entstehen, welches den Klang eines Leslie-Lautsprechers auf einer Bühne oder in einem Proberaum reproduzieren kann.

Zunächst wird der Leslie-Lautsprecher vorgestellt und ein Überblick über seine geschichtliche Entwicklung und die heutige Relevanz gegeben. Nach einer Beschreibung des Klanges und des Aufbaus folgt eine Analyse zu existierenden Ansätzen zur digitalen Modellierung des Effekts. Anschließend werden die Hard- und Softwareressourcen vorgestellt, die für die in dieser Arbeit vorgestellte Simulation verwendet werden. Für die Umsetzung wird ein Ansatz für die Modellierung gewählt und zunächst ohne Echtzeitanforderung mithilfe von Matlab implementiert und auf seine Machbarkeit überprüft. Anschließend folgen die wesentliche Implementierung auf dem Audioboard in der Programmiersprache C++ und die Beschreibung der Beschallungsanlage, die für die Klangwiedergabe genutzt wird. Zuletzt wird das System messtechnisch analysiert und die Ergebnisse in einem Hörversuch bestätigt.

2 Der Leslie-Lautsprecher

Unter einem Leslie-Lautsprecher – oder einem Rotationslautsprecher im Allgemeinen – versteht man ein akustisch-mechanisches Effektgerät. Er wurde ursprünglich für die elektromechanische Orgel konzipiert, wird jedoch mittlerweile für viele elektroakustischen Musikinstrumente verwendet. Durch den Einsatz rotierender Schallabstrahler im Inneren des Gerätes treten mehrere akustische Phänomene auf, die dem Lautsprecher seinen charakteristischen Klang verleihen.

Besonders bezeichnend für diesen Klang sind die Leslie-Modelle 147 und 122 (Henricksen, 1981). Da diese Modelle stellvertretend für den klassischen Klang eines Rotations-Lautsprechersystems betrachtet werden können und in der Populären Musik bis heute eine Rolle spielen (Vail, 2002, p. 10), werden im weiteren Verlauf diese Modelle genau betrachtet und analysiert.

2.1 Historische Einordnung

Laurens Hammond, geboren im Jahr 1895, war ein Ingenieur und Unternehmer aus den USA, der in seinem kleinen New Yorker Laboratorium 1921 einen Synchronmotor entwickelte, für den es seiner Meinung nach eine Vielzahl von Anwendungsmöglichkeiten geben würde (Faragher, 2011, p. 4).

Nach einigen erfolglosen Jahren, in denen Hammond nach einer lukrativen Anwendung suchte, brachte 1928 die Entwicklung eines elektrischen Uhrwerks – damals eine Neuheit – den finanziellen Durchbruch. Doch der Erfolg war nicht von Dauer: Die globale Wirtschaftslage verschlechterte sich und Hammond verlor das Patent an seinem Uhrwerk. Eine neue Idee musste her. Hammond wusste, dass er seinen Motor in Verbindung mit einem Generator dazu verwenden konnte, Töne zu erzeugen. Aus dieser Idee entstand die erste elektromechanische Orgel von Laurens Hammond, die er 1934 zum Patent anmeldete und 1935 in New York vor einem Fachpublikum der Öffentlichkeit präsentierte. Die Hammond-Orgel gewann schnell an Beliebtheit, weil sie eine vielfach günstigere und kompaktere Alternative zur konventionellen Pfeifenorgel mit einer vorzeigbaren Qualität darstellte (Faragher, 2011, p. 5f.).

Donald Leslie, geboren 1911, arbeitete als Radio- und Servicetechniker bei einem Einrichtungsgeschäft in Los Angeles, das selbst Hammonds Produkte vertrieb, als er sich 1937 eine eigene Orgel kaufte. Als passionierter Organist war seine Hoffnung, den Klang einer echten Orgel zu Hause reproduzieren zu können. Doch zu Hause angekommen, war der Klang im Vergleich zum großen Ausstellungsraum des Geschäfts mit hervorragender Akustik sehr enttäuschend (Vail, 2002, p. 129). Um Geld zu sparen, verzichtete er auf den Kauf einer Lautsprechereinheit des Herstellers und beschloss diese selbst zu bauen (Vail, 2002, p. 129).

Leslie stellte fest, dass die Klangquelle einer Pfeifenorgel sich im Raum bewegte, weil jeder gespielte Ton einer anderen Pfeife entsprang, die sich wiederum an einem anderen Ort befand. Diese wahrgenommene Bewegung wollte er auch mit seinem Lautsprecher für die Hammond-Orgel erzielen. Also experimentierte Leslie mit unterschiedlichen bewegten Lautsprechersystemen. Nach zahlreichen Iterationen

entstand nach wenigen Jahren das, was heute unter dem Namen Leslie-Lautsprecher bekannt ist (Vail, 2002, p. 129).

Als Leslie seine Erfindung 1940 Hammonds Unternehmen vorstellte, wurde er keineswegs mit offenen Armen empfangen. Zum einen wurde befürchtet, Leslies Lautsprecher würde sich negativ auf die Verkaufszahlen der hauseigenen Lautsprechereinheiten auswirken (Faragher, 2011, p. 101). Andererseits sah Hammond einen persönlichen Affront darin, dass jemand außerhalb seines Unternehmens seine eigene Entwicklung verbesserte (Vail, 2002, p. 11).

Leslie wurde von Hammond abgelehnt und gründete daraufhin sein eigenes Unternehmen, mithilfe dessen er seine Lautsprecher vertrieb. Hammond unternahm zahlreiche Versuche, seinen Konkurrenten zu sabotieren, zum Beispiel, indem er seinen Vertriebspartner*innen verbot, die Lautsprecher seines Konkurrenten zu verkaufen (Vail, 2002, p. 11). Oder indem er neuere Orgelmodelle mit Steckern versah, die den Betrieb mit Leslie-Lautsprechern verhindern sollten (Vail, 2002, p. 131).

Für Musiker*innen wurden Leslie und Hammond allerdings schnell unzertrennlich und die Kombination etablierte sich als fester Bestandteil von Jazz, R'n'B, Rock und Gospelmusik der 1950er und 60er Jahre (Vail, 2002, p. 15 f.).

Leslie verkaufte sein Unternehmen Electro Music 1965 an CBS, die es wiederum 1980 an Hammond verkauften. Hammond gab 1985 jedoch selbst das Geschäft auf und verkaufte es weiter. Nach einigen weiteren Übernahmen befinden sich die Rechte für die Produktion von Hammonds und Leslies seit 1991 im Besitz von Suzuki Musical Instruments und erfreuen sich bis heute großer Beliebtheit (Hammond Suzuki Europe B.V., 2022).

Der Höhepunkt der Hammond-Orgel und des damit einhergehenden Leslie-Lautsprechers ging in den 80er Jahren zwar zu Ende, als digitale Synthesizer in Mode kamen (Faragher, 2011, p. 12 f.), doch sie haben die Populärmusik so stark geprägt, dass ihr Klang bis heute relevant bleibt. Neben zahlreichen Nischenbands – wie zum Beispiel die Psychedelic Rock Band deWolff auf ihrem Album „Wolffpack" (DeWolff, 2021) –, deren Musik stilistisch an Rockmusik aus den 1960er und -70er Jahren angelegt ist, wird die elektronische Orgel auch in moderne Kontexte gesetzt – wie zum Beispiel bei dem Titel „2 Be Loved (Am I Ready)" von Lizzo (2021).

2.2 Aufbau

Durch den häufigen Weiterverkauf des Unternehmens ist es schwer zu sagen, wie viele Leslie-Lautsprecher und welche Modelle insgesamt verkauft wurden. Doch die beliebtesten Modelle mit dem längsten Produktionszeitraum sollen Modell 122 und Modell 147 sein (Vail, 2002, p. 138). Beide Modelle sind identisch in ihren Maßen und in einem Großteil ihrer Ausstattung. Den größten Unterschied stellen ihre Signaleingänge dar, die im 122er Modell symmetrisch und durch eine zusätzliche Leitung mit älteren Orgelmodellen kompatibel sind (Vail, 2002, p. 138). Das Modell 147 hat hingegen unsymmetrische

Eingänge, bietet jedoch aufgrund einer anderen Relais-Schaltung eine geringere Verzögerung beim Umschalten zwischen den zwei Geschwindigkeitsmodi schnell (*Tremolo*) und langsam (*Chorale*) bezeichnet (Faragher, 2011, p. 111).

Abbildung 1: Restaurierter Leslie-Lautsprecher Modell 122
Quelle: (Rathjens, 2019)

Die klassischen Modelle des originalen Leslie-Lautsprechers sind ungefähr 104 cm hoch, 74 cm breit und 53 cm tief (Faragher, 2011, p. 109). Das rechteckige Gehäuse aus Holz ist im Inneren in drei Bereiche unterteilt: Im oberen Teil befindet sich der Schallabstrahler für die Hochtöne. Im mittleren Teil befinden sich die Schallwandler und die Motoren. Im unteren Bereich befindet sich der Schallabstrahler für die tiefen Töne sowie der Verstärker und die Frequenzweiche (Henricksen, 1981).

Der Röhrenverstärker hat eine Leistung von 40 Watt und ist mit Röhren des Typs 6550 bestückt. Er speist eine passive Frequenzweiche mit einer Trennfrequenz bei 800 Hz und einer Dämpfung von 12 dB/Oktave. Sowohl die Frequenzweiche als auch die Schallwandler haben eine Impedanz von 16 Ohm (Henricksen, 1981).

An dieser Stelle soll zwischen den Begriffen Schallwandler und Schallabstrahler unterschieden werden. Die Erzeugung des Schalls läuft bei einem Leslie-Lautsprecher in zwei Stufen ab. Die Wandlung der elektrischen Energie in Schallenergie erfolgt über stationäre Lautsprecher. Einem Kalottenlautsprecher für die hohen und einem Konuslautsprecher für die tiefen Töne (Henricksen, 1981). Diese sind hier mit dem Begriff Schallwandler gemeint. Der erzeugten Schallenergie wird über Schallabstrahler eine Richtwirkung verliehen, im oberen Frequenzbereich durch ein rotierendes Horn und im unteren Frequenzbereich durch einen schaufelförmigen Rotor.

Die Einheit für die hohen Töne besteht aus einem ¾“ Kalottenlautsprecher des Herstellers Jensen, der an einem Rohr befestigt ist, das den mittleren und den oberen Teil des Gehäuses verbindet. Dieses Rohr fungiert auch als Lager für das rotierende Horn, welches im oberen Teil mit der Röhre verbunden ist. Das Horn ist um 90° gekrümmt und über einen Riemen mit dem Motor verbunden. Auf den ersten Blick

sieht die Hochtoneinheit so aus, als würde der Schall von zwei Hörnern abgestrahlt werden. Bei einem von beiden handelt es sich jedoch lediglich um ein Gegengewicht, damit bei der Rotation keine Unwucht erzeugt wird. An der Öffnung des Horns befindet sich ein kegelförmiger Diffusor, der dafür sorgt, dass die Richtwirkung des Horns reduziert wird. Laut von Henricksen durchgeführten Messungen ist dieser Diffusor dafür verantwortlich, dass der Schall an der Hornöffnung eine nahezu kugelförmige Abstrahlcharakteristik besitzt (Henricksen, 1981).

Die Tieftoneinheit befindet sich im unteren Teil des Leslies. Der Konuslautsprecher mit einem Durchmesser von 15“ befindet sich im mittleren Teil und ist nach unten auf eine rotierende Holztrommel, den Bass-Rotor, gerichtet. In deren Mitte befindet sich eine schaufelförmige Rampe, die den Schall um 90° umlenkt und in horizontale Richtung projiziert. Der Bass-Rotor ist seitlich von einem Stoff bedeckt, um den Luftwiderstand bei hohen Rotationsgeschwindigkeiten zu reduzieren.

Die Motoren, von denen die Rotoren angetrieben werden, verfügen bei den Leslie-Modellen 122 und 147 über zwei Geschwindigkeitseinstellungen, zwischen denen während des Betriebes umgeschaltet werden kann.

2.3 Klang

Der Klang eines Leslie-Lautsprechers kommt primär durch zwei Effekte zustande, die maßgeblich der Bewegung der Schallquellen zuzuschreiben sind: die Amplitudenmodulation und die Frequenzmodulation. Bei der Amplitudenmodulation ist der Schallpegel aufgrund der Richtwirkung der rotierenden Schallquellen dann am höchsten, wenn sich das Horn bzw. die Öffnung der Basstrommel in Richtung der Hörposition befindet. Beim Horn spielt für diesen Effekt auch die Entfernung zwischen Schallquelle und Hörposition eine Rolle. Umgekehrt verhält es sich, wenn die Schallquelle in die entgegengesetzte Richtung zeigt. Dadurch entsteht eine Modulation des Schallpegels an einer gegebenen Hörposition (Henricksen, 1981).

Zur Frequenzmodulation kommt es vor allem durch die Bewegung des Horns. Aufgrund des Doppler-Effekts werden in der Zeit, in der sich das Horn auf die Hörposition zubewegt, die abgestrahlten Frequenzen erhöht, und umgekehrt werden sie vermindert, wenn sich das Horn von der Hörposition wegbewegt (Henricksen, 1981).

Es wäre allerdings eine Vereinfachung anzunehmen, dass diese beiden Phänomene die einzigen sind, die bei der Nutzung eines Leslie-Lautsprechers auftreten. Donald Leslie sagte dazu selbst: *„Everybody describes it as the Doppler effect, but they really don’t understand what it’s doing. There’s a complex situation going on with the moving speaker. When the speaker moves toward you, the pitch will be going up. At the same time, there’s radiation going over to a wall and reflecting back to you, and that sound is going down in pitch. The Leslie creates a broad sound made up of an almost infinite number of frequencies from all these changing reflections that are bouncing off the walls.”* (Vail, 2002, p. 134) Seiner

Meinung nach spielen also die Reflektionen innerhalb des Lautsprechers ebenfalls eine wichtige Rolle für die Entwicklung des Klanges.

Insgesamt kann der Klang eines Leslies als füllig und weiträumig bezeichnet werden. Durch die Modulationen entsteht ein abwechslungsreicher Klang, auch wenn eine Note oder ein Akkord für längere Zeit gehalten wird.

Ein etabliertes Element in der Verwendung eines Leslie-Lautsprechers ist das Umschalten zwischen den beiden Rotationsgeschwindigkeiten während des Betriebes. Die langsame Einstellung (*Chorale*) lässt die Rotoren mit einer Frequenz von ca. 40 rpm, bzw. 0,67 Hz rotieren (Herrera, et al., 2009). Die schnelle Rotationseinstellung (*Tremolo*) hat eine Frequenz von etwa 415 rpm, oder 6,92 Hz (Herrera, et al., 2009). Da die beiden Schallabstrahler von separaten Motoren betrieben werden, drehen sich Horn- und Bass-Rotor nicht mit derselben Geschwindigkeit. Der Motor der Basseinheit tendiert dazu, sich langsamer zu drehen und braucht aufgrund der höheren Masse des Bass-Rotors und der daraus resultierenden Trägheit eine längere Zeit, um nach dem Umschalten zwischen *Chorale* und *Tremolo* die Zielgeschwindigkeit zu erreichen. Diese Eigenschaft ist fester Bestandteil des Leslie-Klanges und wird oft als stilistisches Mittel eingesetzt, auch wenn sie von Donald Leslie stets als Imperfektion gesehen wurde. (Faragher, 2011, p. 103).

Diese Eigenschaften verleihen dem Leslie einen unverwechselbaren Klang, der in Verbindung mit der Hammond-Orgel nicht wegzudenken und ein Teil ihrer Charakteristik ist. Auch andere Instrumente werden mithilfe des Leslie-Lautsprechers modifiziert. Ein berühmtes Beispiel dafür ist das Stück „While My Guitar Gently Weeps“ von den Beatles (Harrison, 1968), auf dem Eric Clapton sein Gitarrensolo durch einen Leslie spielte und aufnahm (Vail, 2002, p. 132).

2.4 Bestehende Ansätze zur Modellierung

Bereits Leslie hat sich mit Möglichkeiten der Simulation seiner Erfindung mittels elektrischer Schaltungen beschäftigt, allerdings kam er zu keinem zufriedenstellenden Ergebnis (Vail 2002: S. 139). Im Rahmen dieser Arbeit werden Ansätze zur Modellierung des Leslie-Lautsprechers betrachtet, die auf digitaler Signalverarbeitung beruhen und in wissenschaftlichen Fachzeitschriften erschienen sind.

Im 2011 erschienenen Artikel „Computationally efficient Hammond Organ synthesis“ (Pekonen, et al., 2011) beschreiben die Autor*innen ein Modell, das den Effekt des Leslie-Lautsprechers mithilfe von in Reihe geschalteten Allpass-Filtern nachbildet. Da Allpass-Filter einen konstanten Frequenzgang und einen nichtlinearen Phasengang aufweisen, eignen sie sich gut dafür, die Gruppenlaufzeit eines Signals gezielt zu beeinflussen und so die gewünschte Frequenzmodulation zu erzeugen. Dazu wird der Filterkoeffizient der Kaskade durch ein zeitabhängiges Modulationssignal ersetzt. Dasselbe Signal kann auch zur Amplitudenmodulation verwendet werden, wenn es entsprechend skaliert wird. Die Autor*innen des Artikels verwenden eine Sinusfunktion als Modulationssignal, weil das die Rotationsbewegung der Schallabstrahler am besten beschreibt. Sie nutzen zur Erzeugung des Horn-, bzw. Bass-Rotor-Signals

jeweils eine separate Filterkaskade, um die unterschiedlichen Geschwindigkeiten der Motoren wie beim Original abbilden zu können. Dieses Modell ist denen ähnlich, die in mehreren anderen Veröffentlichungen vorgeschlagen werden (Smith, et al., 2002) (Zölzer, 2011, p. 93) und bietet den Vorteil eines geringen Rechenaufwandes aufgrund der geringen Menge an Rechenoperationen. Die Autor*innen betonen jedoch selbst, dass das Modell unvollständig sei. So würden darin Einflüsse der Lautsprecher, des Verstärkers oder des Gehäuses außer Acht gelassen. Dadurch werden auch die Reflektionen innerhalb des Gehäuses nicht mit modelliert, obwohl sie für den Klang von entscheidender Bedeutung sind.

Im Artikel "Real-time perceptual simulation of moving sources" (Kronland-Martinet & Voinier, 2008) wird ein Modell zur Simulation bewegter Schallquellen vorgestellt. Darin werden vier Wahrnehmungshinweise ausgemacht, die dafür sorgen, dass eine bewegte Schallquelle lokalisiert wird. Diese bestehen aus

- Schalldruck: Je näher sich eine Schallquelle befindet, desto höher ist auch der Schalldruck und damit auch die Schallintensität – der Klang wird als lauter wahrgenommen.
- Timbre: Durch frequenzabhängige Schallabsorption der Luft werden hohe Frequenzen mit zunehmendem Abstand zwischen Schallquelle und Hörposition gedämpft.
- Dopplereffekt: Die relative Geschwindigkeit zwischen Schallquelle und Hörposition bewirkt eine Frequenzverschiebung.
- Raumklang: Je nach Raumbeschaffenheit und Abstand zur Schallquelle ergeben sich unterschiedliche Verhältnisse zwischen Direkt- und Diffusschall.

Da dieses Modell durch die Koordinaten von Hör- und Schallquellenposition gesteuert wird, kann es auf viele Situationen angewendet werden. Nach einer allgemeingültigen Implementierung in der Entwicklungsumgebung Max/MSP, wird es von den Autoren anschließend auf zwei Beispiele angewendet. Eines der beiden Beispiele ist der Leslie-Lautsprecher. Es wird angenommen, in Referenz an Henricksen (1981), dass das Horn an seiner Öffnung eine kugelförmige Abstrahlcharakteristik besitzt und die wahrnehmbare Frequenzmodulation des Bass-Rotors vernachlässigbar ist. Zudem werden die Reflektionen im Inneren des Gehäuses berücksichtigt und in Form von Spiegelschallquellen in das Modell mit aufgenommen. Um das Modell mit dem Original vergleichen zu können, wird ein Testsignal in Form einer Sinusschwingung bei 800 Hz über einen Leslie 122A wiedergegeben und die Aufzeichnung mit dem gleichen Testsignal aus dem Modell verglichen und hinsichtlich Frequenz- und Amplitudenmodulation mittels Hilbert-Transformation untersucht. Hier fällt auf, dass die einfache Anwendung des Modells ohne Spiegelschallquellen lediglich eine sinusförmige Modulation des Testsignals bewirkt, sowohl hinsichtlich der Amplitude als auch der Frequenz. Werden jedoch die Spiegelschallquellen dazugeschaltet, entwickelt sich ein weitaus komplexeres Modulationsmuster, das der Messung am Original durchaus ähnlich sieht. Dies kann als starker Hinweis darauf gedeutet werden, dass die Reflektionen innerhalb des Gehäuses einen wesentlichen Beitrag zur Entstehung des Klanges eines Leslie- Effekts beitragen und nicht vernachlässigt werden sollten. Neben der einfachen Implementierung solcher Spiegelschall-

quellen bietet dieses Modell den Vorteil, auf geometrischen Daten zu beruhen. So können Klangeigenschaften, wie zum Beispiel die Hörposition oder die räumlichen Dimensionen, direkt im Modell beeinflusst werden, ohne dafür aufwändige Messungen durchführen zu müssen.

Im Paper „Discrete time emulation of the Leslie speaker" (Herrera, et al., 2009) wird eine Leslie-Simulation mithilfe von zeitabhängigen FIR-Filtern vorgestellt. Dazu werden zunächst Impulsantworten des Horn- und Bass-Rotors mittels Sinussweep im statischen Zustand bei unterschiedlichen Rotationswinkeln gemessen. Durch den Einsatz von zwei Messmikrofonen können im Anschluss an die Messung die genauen Positionen des Horns und des Rotors ermittelt werden, indem die Laufzeitdifferenz trianguliert wird. Die Impulsantworten werden dann nach Position, bzw. Rotationswinkel sortiert und in einer Matrix angelegt. Für die Filterung des Eingangssignals werden drei Methoden untersucht und miteinander verglichen. Die ersten beiden Methoden basieren auf der Faltung des Eingangssignals mit den gemessenen Impulsantworten und unterscheiden sich lediglich in der Implementierung. Bei der ersten Methode wird ein FIR-Filter genutzt, dessen Filterkoeffizienten zu jedem Abtastzeitpunkt der Impulsantwort der entsprechenden Position des Rotors entsprechen. Liegt zur jeweiligen Position keine exakte Messung vor, wird sie aus den benachbarten Impulsantworten interpoliert. Bei der zweiten Methode wird das Eingangssignal mit der Impulsantwort gefaltet, die an der entsprechenden Position der Schallquelle gemessen wurde, und zum Ergebnis hinzuaddiert. Bei der dritten Methode werden die Impulsantworten auf eine geringe Anzahl FIR-Filter mit festgesetzten Koeffizienten reduziert. Das zeitabhängige Filtern wird dann durch eine Gewichtung der Filter erzielt, die der Rotation der Schallquelle entspricht. Die letztgenannte Methode soll laut den Autoren besonders recheneffizient sein und einen angenehmen Klang produzieren. Der Nachteil dieser Methode besteht darin, dass sie vollständig auf Messungen basiert und es dadurch nicht möglich ist, einzelne Parameter im Nachhinein zu verändern.

3 Modellierung

Das Ziel der Modellierung ist ein echtzeitfähiges System, das den Klang eines Leslie-Lautsprechers bestmöglich nachbildet. Dafür soll ein beliebiges Nutzsignal mithilfe digitaler Signalverarbeitung dementsprechend moduliert und über eine geeignete Beschallungsanlage wiedergegeben werden. Neben einer funktionierenden Modellierung des Klanges steht dabei die Möglichkeit des Live-Einsatzes im Vordergrund. Das heißt, dass das System in einem Proberaum oder auf einer Bühne eingesetzt werden kann, ohne dafür zusätzliche Technik wie Workstations oder spezielle Beschallungsanlagen zu benötigen.

3.1 Technische Anforderungen

Klang

Das Ziel ist eine authentische Nachbildung des Klanges eines klassischen Rotationslautsprechers. Als Vorbild dafür dienen die Leslie Modelle 122 und 147. Diese Modelle verdanken einen Teil ihrer Klangeigenschaften dem in ihnen verbauten Röhrenverstärker. Eine vollwertige Röhrensimulation würde

allerdings den Rahmen dieser Arbeit übersteigen und wird in der Simulation daher nicht berücksichtigt. Bei Bedarf kann dem Signalweg jedoch eine nichtlineare Verzerrung in Form eines externen, digitalen oder analogen Effektgerätes hinzugefügt werden, um die gewünschte Wirkung zu erzielen.

Bedienbarkeit

Wie bereits in Kapitel 2.3 beschrieben, ist ein wichtiges stilistisches Element in der Bedienung eines Leslie-Lautsprechers die Steuerung der Rotationsgeschwindigkeit. Daher soll auch in der Simulation eine Funktion implementiert werden, die es erlaubt, zwischen den beiden Geschwindigkeitsmodi *Chorale* und *Tremolo* umzuschalten. Idealerweise geschieht das über eine physische Komponente wie einen Fußschalter oder Pedal.

Echtzeitfähigkeit

Eine wichtige Anforderung des Systems ist die Echtzeitfähigkeit. Das bedeutet, dass die Latenz des Systems nicht so hoch sein darf, dass es im Anwendungsfall eine*n Musiker*in dadurch irritieren könnte. Der genaue Wert für diese Obergrenze ist individuell und lässt sich nicht genau ermitteln. Laut Dickreiter kann „eine Gesamtverzögerung bis ca. 10ms akzeptiert werden“ (Dickreiter, et al., 2014, p. 202). Dieser Wert soll auch als Obergrenze für das System, das dem Modell in dieser Arbeit zugrunde liegt, betrachtet werden.

Formfaktor

Um den Einsatz des Systems in Live-Situationen zu ermöglichen, soll die Räumlichkeit des Klanges durch Lautsprecherstereofonie erreicht werden. Andere Systeme wie die binaurale Reproduktion der Ohrsignale oder die Wellenfeldsynthese (Dickreiter, et al., 2014, p. 218 f.) sind in diesem Fall nur bedingt geeignet, weil sie entweder das Tragen von Kopfhörern oder aufwändige, fest installierte Lautsprecheranordnungen benötigen. Stattdessen soll eine Anordnung von Lautsprechern gewählt werden, die die Räumlichkeit der Schallquelle nachbildet, ohne dabei den Formfaktor eines elektromechanischen Rotationslautsprechers um ein Vielfaches zu übersteigen. So ist auch eine gewisse räumliche Flexibilität gewährleistet, was in Live-Situationen ebenfalls eine wichtige Rolle spielt.

Statische Lautsprecher

Im Gegensatz zum elektromechanischen Vorbild soll das hier beschriebene System mit statischen Schallabstrahlern realisiert werden. Dazu soll die Rotation mithilfe digitaler Signalverarbeitung simuliert und der Klang auf einem geeigneten Beschallungssystem wiedergegeben werden.

3.2 Ressourcen

3.2.1 Audio-Entwicklungsboard

Um das in das System eingehende Signal zu verarbeiten, wird das Audio-Entwicklungsboard „Daisy Seed“ von der Firma Electrosmith verwendet. Dabei handelt es sich um ein eingebettetes System für

echtzeitfähige Audioanwendungen, welches mit einer breiten Auswahl an Programmiersprachen kompatibel ist und dabei relativ günstig in der Anschaffung ist. Es enthält einen ARM Cortex-M7 Prozessor mit einer Taktgeschwindigkeit von 480 MHz und verarbeitet Audiosignale mit 24 Bit Tiefe bei einer Abtastrate von bis zu 96 kHz (Electrosmith, 2021). Es verfügt außerdem über 32 GPIO-Pins, über die externe Sensoren und Aktoren angesteuert werden können, sowie über zusätzliche serielle Schnittstellen für die Einbindung von weiteren Geräten oder Speichermedien. Als Open Source-Plattform besteht einfacher Zugang zu Dokumentationen und viele elementare Funktionen sind bereits über Libraries integriert.

Ein Entwicklungsboard, welches ähnliche Eigenschaften aufweist, ist das „Teensy USB development board“. Es besitzt den gleichen Prozessor und hat mit 600 MHz eine höhere Taktgeschwindigkeit. Allerdings besitzt dieses Entwicklungsboard keinen integrierten Digital/Analog-Wandler, weshalb im Rahmen dieser Arbeit das System von Electrosmith verwendet wird.

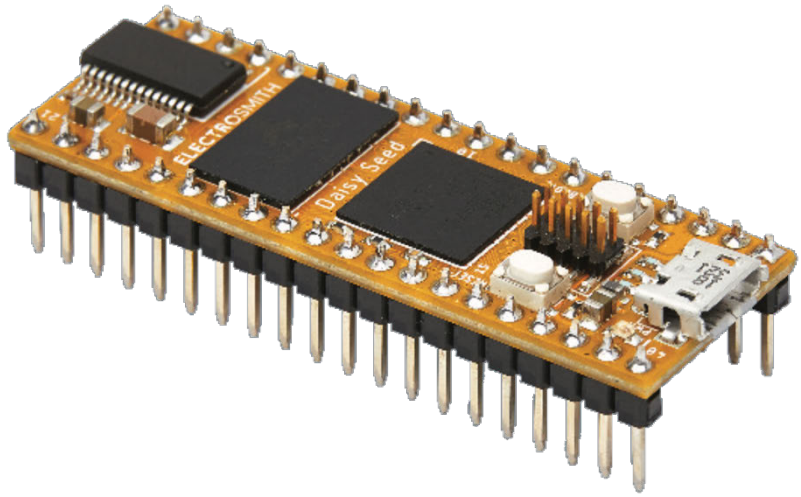

Abbildung 2: Audio Entwicklungsboard „Daisy Seed“ von Electrosmith
Quelle: (Electrosmith, 2021)

3.2.2 Lautsprecher

Es werden zwei verschiedene Lautsprecher-Typen verwendet. Zwei Breitbandlautsprecher sollen die akustischen Eigenschaften des Horn-Rotors simulieren. Ein Lautsprecher mit möglichst großer Membranfläche wird für die Simulation des Bass-Rotors eingesetzt.

Um die hohen Frequenzen wiederzugeben, werden Yamaha Hs-5 Aktivlautsprecher verwendet. Dabei handelt es sich um 2-Wege-Breitbandlautsprecher, in denen ein 5“ Konuslautsprecher für Tieftöne und ein 1“ Kalottenlautsprecher für Hochtöne verbaut sind. Die Grenzfrequenz der internen Frequenzweiche liegt bei 2 kHz. Die Leistung des Hochtöners beträgt 25 Watt und die des Tieftöners 45 Watt (Yamaha, 2017). Da die Lautsprecher für den professionellen Studiogebrauch konzipiert sind, kann davon ausgegangen werden, dass sie dafür geeignet sind, den erforderlichen Frequenzbereich von 800 Hz – 20 kHz

hinreichend unverfälscht wiederzugeben. Sollen jedoch größere Räume oder Flächen mit dem System beschallt werden, sollte ein Paar Lautsprecher mit größerer Leistung in Betracht gezogen werden.

General Specifications

System Type		2-way bi-amp powered speaker, Bass-reflex type
Components	LF	5" cone
	HF	1" dome
Frequency Range	(-10dB)	54Hz-30kHz
Crossover Frequency		2kHz
Output Power*1	Dynamic	70W (LF: 45W, HF: 25W)
Controls		LEVEL control (+4 dB, center click)
		EQ: HIGH TRIM switch (+/- 2 dB at HF), ROOM CONTROL switch (0/-2/-4 dB under 500Hz)
1/8 Power Consumption		45W
Rigging Points		HS5I: Four surfaces x 2 x M5 (60mm pitch)
Material, Finish, Color		MDF, Black/White
Dimensions (W x H x D)		170mm x 285mm x 222mm (6.7" x 11.2" x 8.7")
Net Weight		5.3kg (11.7lbs)
Packaging		Single
Optional Accessories		HS5I: BWS20-190, BWS20-120, BCS20-210, BCS20-150, BWS20-190

*1 Power rating (120V, 25°C). This is total value of individual output power.

Abbildung 3: Technische Daten des Yamaha HS-5 Breitbandlautsprechers
Quelle: (Yamaha, 2017)

Für die tiefen Töne wird ein 150 Watt Bassverstärker mit integriertem 15“ Konuslautsprecher verwendet. Der Durchmesser des Lautsprechers entspricht damit auch dem des original Leslie-Schallwandlers und eignet sich deshalb gut für die Wiedergabe der tiefen Frequenzen. Um eine möglichst unverfälschte Klangwiedergabe zu erreichen, werden die integrierten Filter des Verstärkers auf Nullposition gestellt und die Lautstärke durch die Verstärkung an die der Breitbandlautsprecher angepasst.

3.2.3 Software

Matlab

Matlab ist eine Software für „Programmierung und numerische Berechnungen, die [...] zur Analyse von Daten, Entwicklung von Algorithmen und Erstellung von Modellen verwendet wird.“ (Mathworks, 2022) Es bietet eine breite Palette an Werkzeugen, die zur Signalverarbeitung und -analyse eingesetzt werden können. Zum Beispiel gibt es integrierte Funktionen für „die Analyse und Vorverarbeitung von Zeitreihendaten und Algorithmen für das Design, die Analyse und Implementierung digitaler Filter“ oder „die Visualisierung und Vorverarbeitung von Signalen im Zeit-, Frequenz- und Zeit-Frequenz-Bereich“ (Mathworks, 2022). Die Software eignet sich daher gut, das Modell des Rotationslautsprechersystems auf seine Machbarkeit zu überprüfen und erste Daten und Messungen darüber zu erhalten. Es wird ebenfalls dafür eingesetzt, Datensätze wie Filterkoeffizienten und Delay-Zeiten für die Implementierung auf dem Audioboard vorauszuberechnen.

Visual Studio Code

Für die Implementierung des C++ Codes auf dem Audioboard wird der Quelltexteditor Visual Studio Code verwendet. Als eine der meistgenutzten Entwicklungsumgebungen (stackoverflow, 2021) bietet

es komfortable Merkmale wie automatische Codevervollständigung und eine große Auswahl an verfügbaren Erweiterungen. Die Hilfsprogramme und Libraries des Audioboards sind ebenfalls auf Visual Studio Code ausgerichtet, sodass es eine ideale Schnittstelle zwischen Hardware und Software bietet.

3.2.4 Messtechnik

Für Messungen der Audiosignalverarbeitung am Audioboard wird der APx-515 Audio Analyzer verwendet. Er ist in der Lage alle relevanten Audiomessungen, wie Frequenzgänge, Latenzen und Impulsantworten, in kurzer Zeit zu liefern und ist durch die mitgelieferte Software einfach zu bedienen (Audioprecision, 2019, p. 1 f.).

Für die Analyse von digitalen Signalen und aufgezeichneten Testsignalen wird ebenfalls Matlab verwendet.

3.3 Software-Implementierung

Als Grundlage für das Modell wird die Arbeit von Kronland-Martinet und Voinier (2008) herangezogen. Da deren Ansatz auf physikalischen Größen beruht, ist es unkompliziert, einzelne Parameter zu verändern und so das System zu optimieren, ohne dafür neue Messungen durchführen zu müssen. Es können zum Beispiel die simulierten Hörpositionen so gewählt werden, dass sie zur Lautsprecheranordnung passen. Außerdem lässt sich durch die Wahl der Anzahl an Spiegelschallquellen die beanspruchte Rechenleistung so steuern, dass die Anforderung an die maximale Latenzzeit nicht überschritten wird und eine ideale Balance aus Effizienz und Klangqualität erreicht wird.

Die wichtigste Größe, die dafür zu betrachten ist, ist die Distanz zwischen Schallquelle und Hörposition. Aus dieser Größe ergeben sich unmittelbar die nötigen Parameter für Frequenz- und Amplitudenmodulation. Um den Zusammenhang zwischen Frequenzmodulation und Distanz zu erläutern, soll zunächst der Dopplereffekt und dessen digitale Implementierung betrachtet werden. Der Dopplereffekt beschreibt eine Frequenzverschiebung, die entsteht, wenn Schallquelle und Hörposition relativ zueinander in Bewegung sind. Bewegen sich Schallquelle und Hörposition aufeinander zu, wird das Signal mit einer höheren Frequenz wahrgenommen. Bewegen sie sich voneinander weg, wird das Signal mit einer tieferen Frequenz wahrgenommen (Friesecke, 2014, p. 22). Um diesen Effekt im digitalen Bereich zu implementieren, wird die Änderung im Abstand zwischen Schallquelle und Hörposition als Änderung der Zeit betrachtet, die der Schall benötigt, um jene Strecke zu durchqueren. Dafür werden variable Delaylines eingesetzt, also zeitabhängige Verzögerungselemente. Der Wert für die Verzögerung kann zum Beispiel von einem niedrigfrequenten Sinussignal angesteuert werden. Solch ein digitaler Audioeffekt wird *Vibrato* genannt (Zölzer, 2011, p. 75). Es sind Verzögerungen nötig, bei denen es sich nicht um ganzzahlige Vielfache eines Sample-Intervalls handelt. Dafür werden Fractional Delay-Lines eingesetzt, bei denen Abtastwerte aus beliebigen Zeitintervallen aus den benachbarten Abtastwerten interpoliert werden (Zölzer, 2011, p. 73 f.).

Zwischen der Distanz von Hörposition zu Schallquelle und der Amplitude des Signals besteht ebenfalls ein direkter Zusammenhang. Für Punktschallquellen verhält sich „die Schallintensität bei konstanter Schallleistung der Quelle umgekehrt proportional zu Zunahme der Oberfläche“ (Weinzierl, 2008, p. 27). Aufgrund der Messungen und den dazugehörigen Ausführungen Henricksens (1981) wird beim Horn von einer Punktschallquelle mit kugelförmiger Ausbreitungscharakteristik ausgegangen. Befindet sich das Horn während der Rotation jedoch in Bewegung, entspricht die Ausbreitung auf horizontaler Ebene einer Linienschallquelle, besonders bei kurzen Entfernungen zwischen Schallquelle und Hörposition, relativ zum Rotationsradius. In diesem Fall verringert sich laut Weinzierl (2008) die Schallintensität mit

$$I \sim \frac{1}{r} \tag{1}$$

Durch die Annahme einer kugelförmigen Schallausbreitung an der Öffnung des Horns, lässt sich die momentane Entfernung zwischen Schallquelle und Hörposition berechnen. Abbildung 1 zeigt eine schematische Darstellung des Horn-Rotors (P) als Schallquelle von oben und seinen dazugehörigen Spiegelschallquellen und deren Rotationsrichtung (P1 – P3) in dieser Ebene. Die Entfernung zwischen einer Schallquelle und der Hörposition (Plist) kann berechnet werden, indem die Positionsdaten in ein kartesisches Koordinatensystem eingefügt werden und zwischen zwei Punkten der Betrag gebildet wird. Dies lässt sich beschreiben durch die Formel

$$dd(t) = \sqrt{(x_{Plist} - x_P(t))^2 + (y_{Plist} - y_P(t))^2} \tag{2}$$

Wobei *dd* die Strecke zwischen Direktschallquelle und Hörposition zum Zeitpunkt t darstellt und $x_P(t)$ bzw. $y_P(t)$ die Position der Schallquelle auf der x-, bzw. y-Achse. Für die Spiegelschallquellen kann genauso vorgegangen werden. Eine Ausnahme bilden die Rotationspositionen, bei denen der Punkt, an dem Schall reflektiert wird, außerhalb des Gehäuses liegt. Während sich der Rotor in diesem Bereich befindet, sollten die entsprechenden Spiegelschallquellen inaktiv sein. Allerdings handelt es sich hierbei um eine Abstraktion des akustischen Verhaltens innerhalb des Gehäuses. Im Normalfall sind alle Seiten eines Leslie-Lautsprechers verschlossen und der Schall entweicht durch lammelenartige Öffnungen, die sich in der Holzverkleidung befinden. Der Schall wird also permanent an allen Innenwänden des Gehäuses reflektiert, während ein Teil des Schalls durch die Lamellen entweicht und in den Raum abgestrahlt wird. Somit stellen die drei simulierten Spiegelschallquellen eine Vereinfachung dieses akustischen Verhaltens dar. Ein interessanter Ansatz für zukünftige Implementierungen wäre eine komplexere Modellierung des Verhaltens mit dem Ansatz der Schallausbreitung als Schallstrahl, zum Beispiel mithilfe eines Raytracing Algorithmus (Weinzierl, 2008, p. 282).

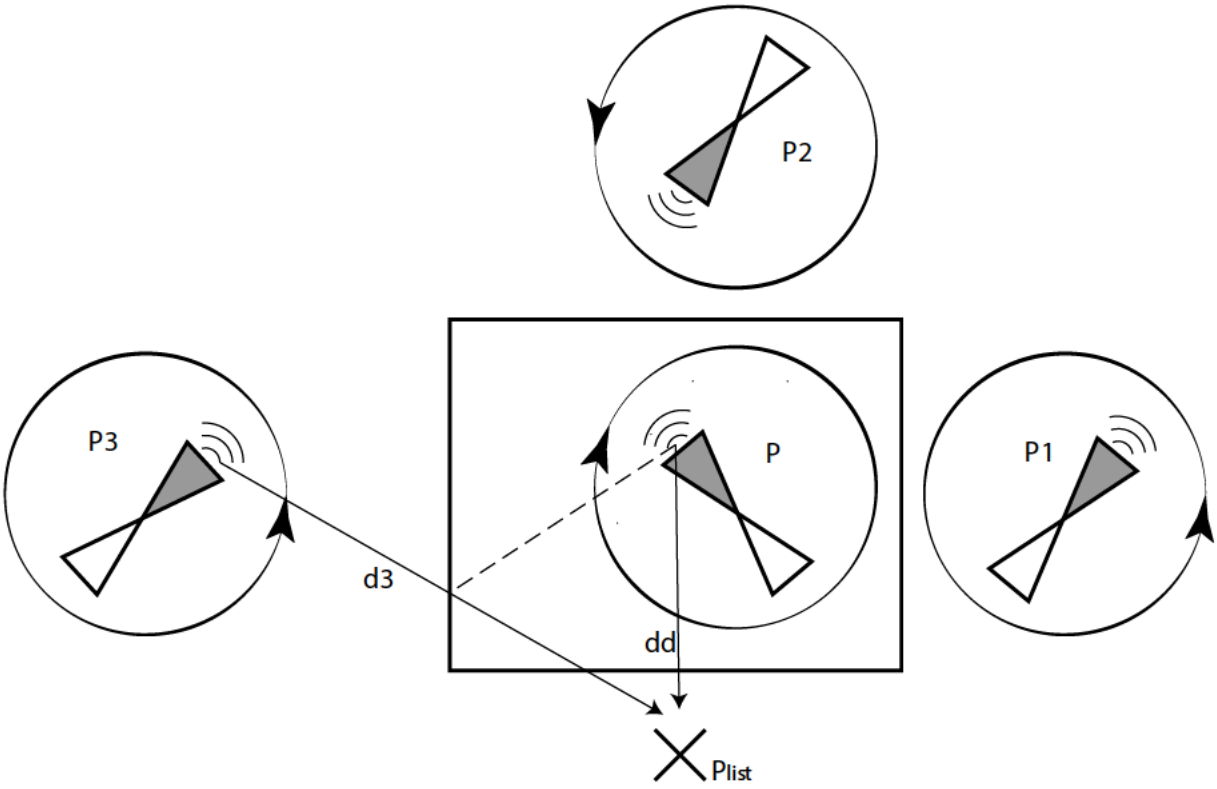

Abbildung 4: Darstellung des Schallweges des Horns und seiner Spiegelschallquellen zur Hörposition
Quelle: Eigene Darstellung

Das Eingangssignal wird mittels IIR-Filter in zwei Frequenzbereiche aufgeteilt, deren Trennfrequenz bei 800 Hz liegt. Für den Bass-Rotor werden keine Spiegelschallquellen simuliert, weil die Wellenlänge bei der Trennfrequenz bereits bei

$$\lambda = \frac{c}{f} = \frac{343\frac{m}{s}}{800\,Hz} \approx 0{,}43m \qquad (3)$$

und damit bereits knapp unter der Seitenlänge des Gehäuses liegt. Damit eignet sich die Betrachtung der Schallausbreitung als Schallstrahl nicht mehr, weil dafür „der Durchmesser der reflektierenden Fläche mindestens mehrere Wellenlängen der reflektierenden Schallwelle [messen sollte]“ (Dickreiter, et al., 2014, p. 18). Eine andere Überlegung wäre, das Gehäuse des Bass-Rotors hinsichtlich seiner Eigenschaften zu stehenden Wellen zu untersuchen. Laut Henricksen (1981) ist beim Tieftonsignal, welches vom Bass-Rotor abgestrahlt wird, lediglich Amplitudenmodulation wahrnehmbar, aber keine Frequenzmodulation. Da die Implementierung des Doppler-Effekts nur einen geringen Aufwand mit sich bringt, soll in dieser Arbeit an gegebener Stelle evaluiert werden, ob die Implementierung dem Klang zuträglich ist.

3.3.1 Matlab

In der Implementierung mittels Matlab wird zunächst die grundlegende Machbarkeit des Modells ohne Echtzeitanforderung untersucht. Das Ausgangssignal verfügt über einen Kanal und die räumlichen Dimensionen werden geschätzt und als gleich für beide Rotoren betrachtet. Es geht hierbei hauptsächlich

darum, die Auswirkung der Spiegelschallquellen auf das Ausgangssignal zu testen und erste messtechnische Daten zu sammeln. Das vollständige Modell wird später direkt auf dem Entwicklungsboard in C++ implementiert.

Codeblock 1: Leslie Simulation in Matlab (I)
Quelle: Eigene Darstellung

```
% LESLIE DIMENSIONS
WIDTH = 0.74;
DEPTH = 0.53;
R = 0.2; %RADIUS HORN%CREATING 800 HZ SINE WAVE
FS= 44100; %SAMPLERATE
SECS = 8; %LENGHT IN SECONDS
SMPL = SECS*FS; %LENGHT IN SAMPLES
PITCH = 800; %PITCH FREQ IN HZ
PS = PITCH/FS;

X = ZEROS(SMPL,1);

FOR P=1:(SMPL-1)
   X(P)=0.5*SIN(PS*2*PI*P);
END

LEN=LENGTH(X);

TIME = 0:1/FS:LEN/FS-1/FS;

%LISTENER POSITION
PLIST = [3*WIDTH/2; -0.5];

HORNRPM = 390;
BASSRPM = 335;

ROTFREQHORN = HORNRPM/60; %ROTATION FREQUENCY
ROTFREQBASS = BASSRPM/60;
```

In den ersten Zeilen werden die Variablen für die Maße des Leslie-Gehäuses und den Radius des Horns definiert. Da genauere Daten nicht aufzufinden waren und diese Implementierung zur Überprüfung der Machbarkeit dient, werden die in der Literatur genannten (Faragher, 2011, p. 109) Außenmaße als Innenmaße verwendet und der Radius des Horn-Rotors geschätzt. In den anschließenden Zeilen wird ein Sinussignal erzeugt, welches als Testsignal fungiert. Die Frequenz von 800 Hz wird gewählt, weil

dadurch sowohl der Horn- als auch der Bass-Rotorsimulator gleichermaßen angesteuert werden. Außerdem wurde so ein Signal auch von Kronland-Martinet und Voinier (2008) für ihre Messungen verwendet. So können die erzielten Ergebnisse später miteinander verglichen werden. Die Variable „LEN“ beschreibt die Länge des Gesamtsignals in Samples und dient der späteren Speicherzuweisung. Bei der Speicherzuweisung wird ein Array mit der benötigten Anzahl an Elementen erstellt und mit Nullen gefüllt. Dadurch muss die Größe des Arrays beim Schreiben der Daten nicht angepasst werden, was zu einer Reduktion der benötigten Rechenzeit führt. Die Variable „TIME“ ist ein eindimensionales Array, welches für die Umrechnung von Samples in Sekunden bei der graphischen Darstellung des Signals benötigt wird. In den darauffolgenden Zeilen wird die Hörposition und die Rotationsgeschwindigkeit der Rotoren definiert. Die Werte für die Samplerate, die Rotationsgeschwindigkeit und die Hörposition werden anschließend an die Funktionen für die Simulation des Horn- bzw. Bass-Rotors weitergegeben.

Das Ziel dieser Funktion ist es, alle Parameter zu berechnen, die nicht von dem Eingangssignal abhängig sind. So kann die erforderliche Rechenleistung bei der Signalverarbeitung auf ein Minimum beschränkt werden.

Codeblock 2: Funktion für die Simulation des Horn-Rotors (I)
Quelle: Eigene Darstellung

```
function [ID,FRACD,I1,FRAC1,I2,FRAC2,I3,FRAC3, LD, L1, L2, L3,NSAMGD,G1,G2,G3] =
LESLIEHORN(FS,ROTFREQHORN, PLIST)
%LESLIE TABULATED DELAY TIMES FOR IMAGE SOUND SOURCES INSIDE LESLIE CABINET

%SPEED OF SOUND
CSOUND = 343;

% LESLIE DIMENSIONS
HEIGHT = 0.83;
WIDTH = 0.73;
DEPTH = 0.53;
R = 0.2; %RADIUS HORN

NSAM = ROUND(FS/ROTFREQHORN); %NUMBER OF SAMPLES PER ROTATION
```

Nach der erneuten Definition einiger Variablen wird mit „Nsam“ die Anzahl der Samples innerhalb einer Umdrehung des Horns berechnet. Anschließend werden die Koordinaten der kreisförmigen Bahn des Horns und dessen Spiegelschallquellen erstellt. Bei dieser Implementierung wird die Anzahl der Spiegelschallquellen auf die drei beschränkt, die sich auf einer horizontalen Ebene befinden. Auf diese Art können alle Positionen in einem zweidimensionalen Koordinatensystem beschrieben werden und der Code bleibt überschaubar.

Codeblock 3: Funktion für die Simulation des Horn-Rotors (II)
Quelle: Eigene Darstellung

```
%CREATING CIRCLE
CENTER = [3*WIDTH/2 DEPTH/2]; %CENTER COORDINATES
ANGLE = 0:((2*PI)/NSAM):2*PI; %DIVDING CIRCLE INTO SAMPLES PER ROTATION

%CIRCLE STARTING CENTER RIGHT GOING CLOCKWISE
CXD = R*COS(ANGLE)+CENTER(1);
CYD = R*(-SIN(ANGLE))+CENTER(2);

%CIRCLE STARTING CENTER LEFT GOING
CX13 = R*(-COS(ANGLE))+CENTER(1);
CY13 = R*(-SIN(ANGLE))+CENTER(2);

%CIRCLE STARTING CENTER RIGHT GOING COUNTER CLOCKWISE
CX2 = R*COS(ANGLE)+CENTER(1);
CY2 = R*SIN(ANGLE)+CENTER(2);

P = [CXD;CYD]; %DIRECT CIRCLE COORDINATES
P1 = [4*WIDTH-CX13;CY13]; %1ST IMAGE SOURCE COORDINATES
P2 = [CX2;2*DEPTH-CY2]; %2ND IMAGE SOURCE COORDINATES
P3 = [2*WIDTH-CX13;CY13]; %3RD IMAGE SOURCE COORDINATES
```

Um die Kreiskoordinaten zu erhalten, werden die Koordinaten für den Mittelpunkt der Kreisbahn ermittelt, der zur Direktschallquelle gehört („Center“). Eine Längeneinheit entspricht dabei einem Meter. Da später nur der Betrag der Strecke zwischen den Punkten im Koordinatensystem relevant ist, spielt die absolute Lage der Kreisbahnen keine Rolle. Abbildung 5 zeigt die gewählte Positionierung.

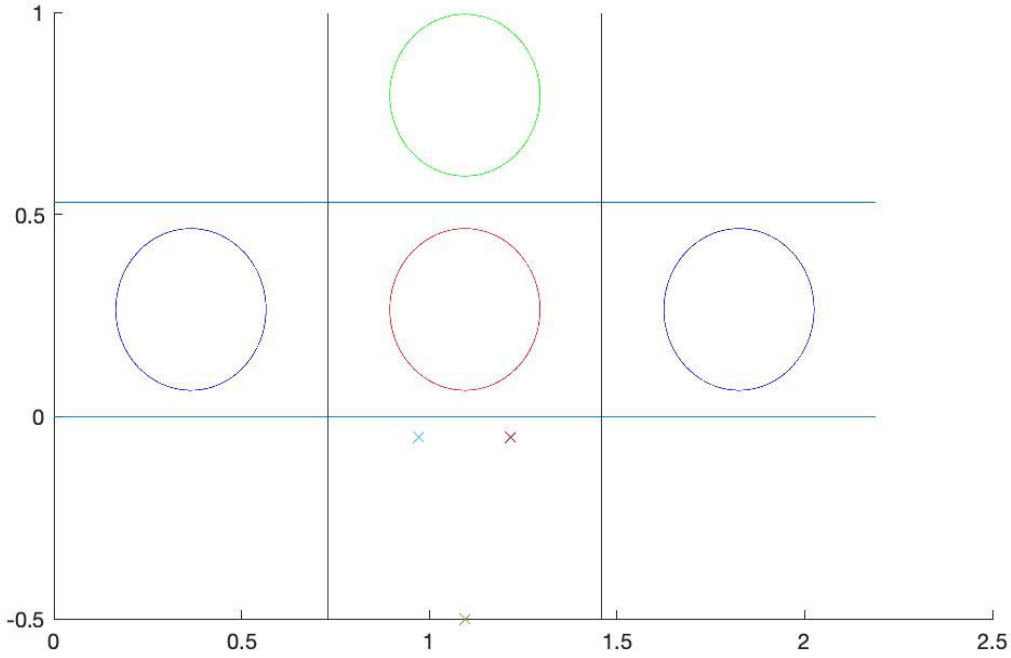

Abbildung 5: Direkt- und Spiegelschallquellen im Koordinatensystem
Quelle: Eigene Darstellung

Die Variable „Angle“ beinhaltet ein eindimensionales Array, welches den Kreis in N_{sam} Winkel aufteilt. Die Funktion

$$Angle(n) = \frac{2\pi}{N_{sam}} * n \tag{4}$$

beschreibt damit den Winkel des Horns *n* Samples nach Beginn der Rotation. Mithilfe der Formel

$$P_{x,y}(n) = \begin{cases} x(n) = r * \cos\big(Angle(n)\big) + Center_x \\ y(n) = r * \sin\big(Angle(n)\big) + Center_y \end{cases} \tag{5}$$

werden die Kreiskoordinaten berechnet. Dabei ist besonders auf die richtige Phase und Rotationsrichtung (siehe Abbildung 4) der Kreisfunktion zu achten. Anschließend werden die Arrays P1 bis P3, die die Koordinaten der Spiegelschallquellen beinhalten, auf die richtige Position verschoben.

Codeblock 4: Funktion für die Simulation des Horn-Rotors (III)
Quelle: Eigene Darstellung

```
%DELAY TIME MEMORY ALLOCATION
TD = ZEROS(NSAM,1);
T1 = ZEROS(NSAM,1);
T2 = ZEROS(NSAM,1);
T3 = ZEROS(NSAM,1);
```

```
%DINSTANCE SOURCE LISTENER
FOR N=1:NSAM
DD = NORM(P(:,N)-PLIST); %DIRECT DISTANCE
D1 = NORM(P1(:,N)-PLIST);
D2 = NORM(P2(:,N)-PLIST);
D3 = NORM(P3(:,N)-PLIST);

%GAIN BASED ON DISTANCE
GD(N) = 1/DD;
G1(N) = (1/SQRT(D1));
G2(N) = (1/SQRT(D2));
G3(N) = (1/SQRT(D3));

MAXDISTANCED=MAX(GD);
MAXDISTANCE1=MAX(G1);
MAXDISTANCE2=MAX(G2);
MAXDISTANCE3=MAX(G3);

MAXDISTANCE = [MAXDISTANCE3 MAXDISTANCE2 MAXDISTANCE1 MAXDISTANCED];

MAXDISTANCETOTAL=MAX(MAXDISTANCE);

GD=TRANSPOSE(GD)/MAXDISTANCETOTAL;
G1=TRANSPOSE(G1)/MAXDISTANCETOTAL;
G2=TRANSPOSE(G2)/MAXDISTANCETOTAL;
G3=TRANSPOSE(G3)/MAXDISTANCETOTAL;

%DELAY IN SAMPLES
TD(N) = (DD/CSOUND)*FS;
T1(N) = (D1/CSOUND)*FS;
T2(N) = (D2/CSOUND)*FS;
T3(N) = (D3/CSOUND)*FS;
END
```

Die Variable „T“ beinhaltet die gesuchte Zeit in Samples, die der Schall benötigt, um die Strecke zwischen Schallquelle und Hörposition zu durchqueren. Die Länge dieser Strecke kann über den Matlab Befehl `norm`, welcher die euklidische Distanz eines Vektors ausgibt, berechnet werden. Die zugehörige Formel lautet

$$d(n) = \sqrt{(P_x(n) - Plist_x)^2 + (P_y(n) - Plist_y)^2} \quad (6)$$

Wobei d(n) die Strecke zwischen Schallquelle und Hörposition und der Punkt $P_{x,y}(n)$ die Position der Schallquelle n Samples nach Beginn der Rotation darstellt. Um die Zeit zu erhalten, wird die Formel

$$T(n) = \frac{d(n)}{c_{sound}} * Fs \tag{7}$$

angewandt. Die erhaltene Delay-Zeit in Samples wird unter den Variablen Td, für die Direktschallquelle, und T1, T2 und T3, für die Spiegelschallquellen, als Array gespeichert. Mit der Variable g wird der Verstärkungsfaktor der jeweiligen Schallquellen berechnet. Für das Direktsignal wird der entfernungsabhängige Faktor für Linienschallquellen aus folgender Formel verwendet.

$$gd(n) = \frac{1}{dd(n)} \tag{8}$$

Um die Direktschallquelle von den Spiegelschallquellen abzuheben und dadurch die Lokalisation der Rotation zu verbessern und den Klang besser zu definieren, werden die Verstärkungsfaktoren der Spiegelschallquellen mit der Formel

$$g_{SSQ}(n) = \frac{1}{\sqrt{d_{SSQ}(n)}} \tag{9}$$

berechnet. Dadurch wird gewährleistet, dass die Schallintensität der Direktschallquelle, im Vergleich zu den Spiegelschallquellen mit größerer Entfernung der Hörposition zur Schallquelle, geringer wird. Da diese Entfernung in der endgültigen Simulation geringer als 1 m ist, weist die Direktschallquelle eine höhere Intensität als die Spiegelschallquellen auf. Die Verstärkungsfaktoren werden nach deren Berechnung auf den größten auftretenden Verstärkungsfaktor normiert. Diese Vorgehensweise basiert nicht auf streng physikalischen Prinzipien, sondern wird zugunsten einer effektiven Implementierung an diesen angelehnt. Abbildung 6 zeigt den zeitlichen Verlauf der Verstärkung im Zeitraum einer Umdrehung. Die rote Kennlinie zeigt den Verlauf der Direktschallquelle. Auf der linken Seite der Abbildung beträgt die Entfernung zwischen dem Gehäuse und der Hörposition 1,2 m und auf der rechten Seite 0,2 m.

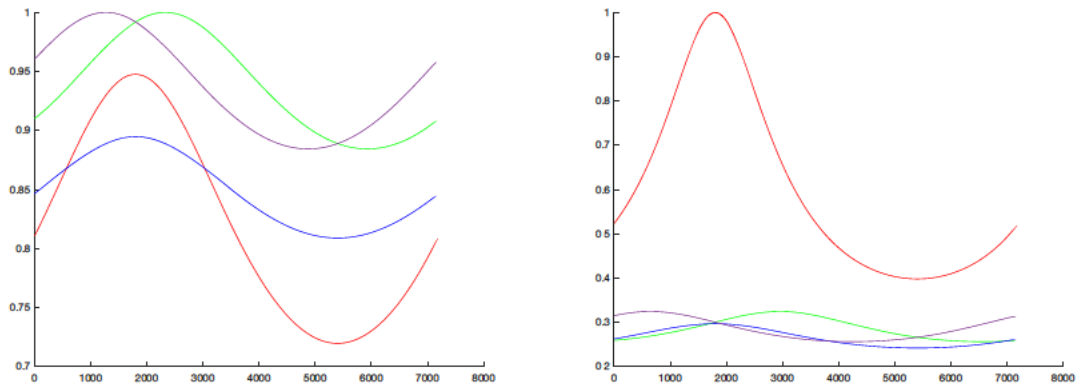

Abbildung 6: Verstärkungsfaktoren im Verlauf einer Umdrehung mit Abstand von 1,2 m (links) und 0,2 m (rechts)
Quelle: Eigene Darstellung

Codeblock 5: Funktion für die Simulation des Horn-Rotors (IV)
Quelle: Eigene Darstellung

```
LD = CEIL(MAX(TD));
L1 = CEIL(MAX(T1));
L2 = CEIL(MAX(T2));
L3 = CEIL(MAX(T3));

%MEMORY ALLOCATION FOR DELAYLINE PARAMETERS
ID=ZEROS(NSAM,1);
I1=ZEROS(NSAM,1);
I2=ZEROS(NSAM,1);
I3=ZEROS(NSAM,1);

FRACD=ZEROS(NSAM,1);
FRAC1=ZEROS(NSAM,1);
FRAC2=ZEROS(NSAM,1);
FRAC3=ZEROS(NSAM,1);

FOR N=1:(NSAM-1)

      ID(N)=FLOOR(TD(N));
   FRACD(N)=TD(N)-ID(N);

      I1(N)=FLOOR(T1(N));
   FRAC1(N)=T1(N)-I1(N);

      I2(N)=FLOOR(T2(N));
   FRAC2(N)=T2(N)-I2(N);

      I3(N)=FLOOR(T3(N));
   FRAC3(N)=T3(N)-I3(N);
END

END
```

Die Variable $L_{d,1,2,3}$ enthält die maximale Delay-Zeit jeder Schallquelle, die für die spätere Speicherzuweisung der Delay-Lines von Bedeutung ist. Anschließend folgt die Speicherzuweisung der Arrays, die für die lineare Interpolation der Fractional Delays benötigt werden. Diese bestehen aus den Variablen $i_{d,1,2,3}$, welche die abgerundeten, ganzzahligen Delay-Zeiten in Samples beinhalten, und den Variablen

$frac_{d,1,2,3}$ mit den dazugehörigen Dezimalwerten. Für die Simulation des Bass-Rotors wird eine Funktion mit demselben Aufbau verwendet, mit dem Unterschied, dass keine Spiegelschallquellen simuliert werden (siehe Anhang 3)

Codeblock 6: Leslie-Simulation in Matlab (II)
Quelle: Eigene Darstellung

```
%Calling back Horn and Bass rotor functions
[ihd,frachd,ih1,frach1,ih2,frach2,ih3,frach3,Lhd,Lh1,Lh2,Lh3,Nsamh,ghd,gh1,
gh2,gh3] = LeslieHorn(Fs,RotfreqHorn, Plist);


[ibd,fracbd,Lbd,Nsamb,gbd] = LeslieBass(Fs,RotfreqBass, Plist);


%memory allocation Delay lines
Delaylinehd=zeros(Lhd,1);
Delaylineh1=zeros(Lh1,1);
Delaylineh2=zeros(Lh2,1);
Delaylineh3=zeros(Lh3,1);


Delaylinebd=zeros(Lhd,1);


%memory allocation output signal
yhd = zeros(LEN,2);
yh1 = zeros(LEN,2);
yh2 = zeros(LEN,2);
yh3 = zeros(LEN,2);
yh = zeros(LEN,2);


ybd = zeros(LEN,2);
yb = zeros(LEN,2);
y = zeros(LEN,2);


%declaring rotation index
h=1;
b=1;
%crossover filtering of input signal
crossFilt           =           crossoverFilter('NumCrossovers',1,'Crosso-
verFrequencies',800,'CrossoverSlopes',12,'SampleRate', Fs);
[xlow, xhigh] =  (crossFilt(x));
```

Die Werte, die nicht vom Eingangssignal abhängig sind, werden aus den Funktionen für die Bass- und die Hornsimulation abgerufen. Es erfolgt erneut eine Speicherzuweisung für die Delay-Lines und deren

Ausgangssignale. Anschließend wird das Eingangssignal durch eine Frequenzweiche in zwei Frequenzbereiche gefiltert. In Matlab kann dafür die Funktion `crossoverFilter` genutzt werden. Dazu werden die Attribute der Funktion dem Objekt `crossFilt` zugewiesen und anschließend auf das Signal angewandt. Die Trennfrequenz wird auf 800 Hz angegeben und mit einer Dämpfung von 12 dB/Okatave. Daraus resultieren die Arrays `xlow`, welches das Signal mit den tiefen Frequenzen enthält, und `xhigh`, welches das Signal mit den hohen Frequenzen enthält.

Codeblock 7: Leslie-Simulation in Matlab (III)
Quelle: Eigene Darstellung

```
%Delay line callback loop
tic
for n=1:(LEN-1)


   %---Linear Interpolation Horn-----------------------------
   Delaylinehd=[xhigh(n);Delaylinehd(1:Lhd-1)];
   yhd(n,:)=Delaylinehd(ihd(h)+1)*frachd(h)+Delaylinehd(ihd(h))*(1-
frachd(h));


   Delaylineh1=[xhigh(n);Delaylineh1(1:Lh1-1)];
   yh1(n,:)=Delaylineh1(ih1(h)+1)*frach1(h)+Delaylineh1(ih1(h))*(1-
frach1(h));


   Delaylineh2=[xhigh(n);Delaylineh2(1:Lh2-1)];
   yh2(n,:)=Delaylineh2(ih2(h)+1)*frach2(h)+Delaylineh2(ih2(h))*(1-
frach2(h));


   Delaylineh3=[xhigh(n);Delaylineh3(1:Lh3-1)];
   yh3(n,:)=Delaylineh3(ih3(h)+1)*frach3(h)+Delaylineh3(ih3(h))*(1-
frach3(h));


   %horn gain calculation
   yh(n,:)=ghd(h)*yhd(n,:)+gh1(h)*yh1(n,:)+gh2(h)*yh2(n,:)+gh3(h)*yh3(n,:);


      %---Linear Interpolation Bass-----------------------------
   Delaylinebd=[xlow(n);Delaylinebd(1:Lbd-1)];
   ybd(n,:)=Delaylinebd(ibd(b)+1)*fracbd(b)+Delaylinebd(ibd(b))*(1-
fracbd(b));
```

In der folgenden `for`-Schleife werden die Signale durch Fractional Delay-Lines verarbeitet. Eine Delay-Line besteht aus zwei Komponenten. Zunächst wird das Sample des Eingangssignals x(n) zum Zeitpunkt n in die Delay-Line geschrieben. Im zweiten Schritt wird das Sample des Ausgangssignals y(n) zum Zeitpunkt n aus der Delay-Line gelesen. Dieses Sample befindet sich in der Stelle Td(h), bzw.

Td(b) der Delay-Line. Dabei steht h, bzw. b, für den Rotationsindex, also die Position des Horns, bzw. des Bass-Rotors zum Zeitpunkt n. Da Td die Zeit darstellt, welche der Schall benötigt, um die Strecke von Schallquelle zu Hörposition zu durchqueren, wird jenes Sample ausgelesen, welches sich zum Zeitpunkt n-Td(h;b) am Eingang des Systems befand.

$$y(n) = x(n - Td(h; b)) \tag{7}$$

Da die Werte von Td aus nicht-ganzen Zahlen bestehen, muss das Sample an dieser Stelle aus den benachbarten Samples interpoliert werden (siehe Abbildung 3).

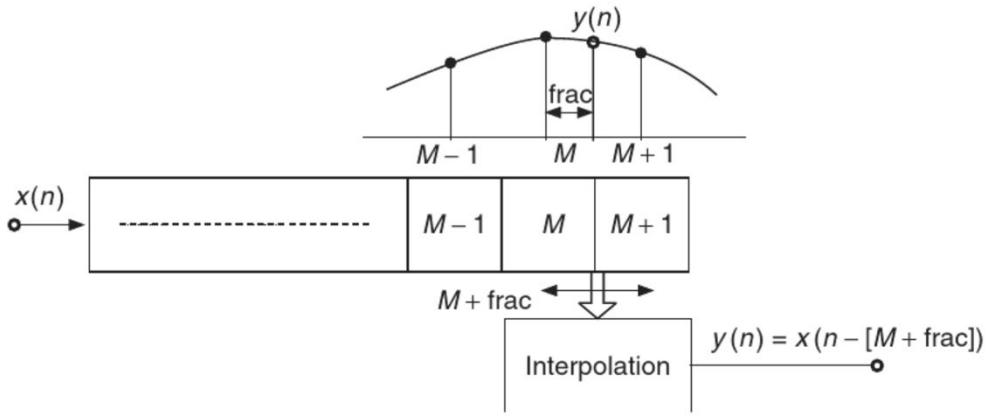

Abbildung 7: Fractional Delay-Line mit Interpolation
Quelle: (Zölzer, 2011, p. 75)

Es gibt verschiedene Methoden, um die Interpolation vorzunehmen. Hier wird die lineare Interpolation verwendet, da sie wenig Rechenleistung beansprucht und somit zur Realisierung eines echtzeitfähigen Systems zuträglich ist. Mit der Formel (Zölzer, 2011, p. 75)

$$y(n) = x(n - [M + 1]frac + x(n - M)(1 - frac) \tag{8}$$

wird eine Gerade zwischen den benachbarten, ganzzahligen Samples gebildet und der Wert der Geraden an der Stelle M+frac bestimmt. Die Größe M in Abbildung 7 wird in diesem Code als Variable „i“ dargestellt.

Codeblock 8: Leslie-Simulation in Matlab (IV)
Quelle: Eigene Darstellung

```
%bass gain calculation
yb(n,:)=gbd(b)*ybd(n,:);

%final output signal
y(n,:)= yh(n,:) + yb(n,:);

%resetting rotation index after full rotation
if h == Nsamh-1
    h = 1;
```

```
    else
        h = h+1;
    end

    if b == Nsamb-1
        b = 1;
    else
        b = b+1;
    end

end
toc

%normalise output signal
y = y/max(y);

%playback sound
sound(y,Fs);
```

Nachdem die Ausgänge der einzelnen Delay-Lines berechnet wurden, werden diese mit ihren respektiven Gain-Faktoren multipliziert und anschließend zusammenaddiert. In den folgenden beiden if-Statements wird über den Rotationsindex abgefragt, ob das Horn bzw. der Bass-Rotor bereits eine volle Rotation vollendet hat. Die Bedingung erfüllt ist, wird der Index auf 1 zurückgesetzt.

Nach der `for`-Schleife ist das Ausgangssignal vollständig berechnet und wird über den Befehl `normalise` normalisiert und anschließend ausgegeben. Die Ausgabe der Simulation ist außerdem als Audiodatei auf dem beiliegenden Datenträger unter dem Dateinamen „Hörbeispiel_2“ in der Rotationsgeschwindigkeit *Tremolo* und unter „Hörbeispiel_3“ in der Geschwindigkeitseinstellung *Chorale* zu finden. „Hörbeispiel_1“ enthält die unbearbeitete Tonaufnahme die dem Paper von Kronland-Martinet und Voinier (2008) entnommen wurde.

3.3.2 Audioboard

Um den Code aus Matlab auf dem „Daisy Seed“ zu implementieren, muss dieser in C++ vorliegen. Die Syntax von der Matlab Programmiersprache, basiert zwar ebenfalls auf C, ist jedoch eine höhere Programmiersprache bei der viele Funktionen und Befehle bereits eingebunden und vorformuliert sind.

Um redundante Rechenoperationen zu vermeiden, werden die Werte der Delay-Lines, die unabhängig vom Eingangssignal sind, über ein Matlab Skript (siehe Anhang 4) vorberechnet und als Listen auf das Audioboard importiert. Diese Listen, oder LUTs (Look Up Tables), enthalten die Werte für Td, also die Delay-Zeiten in Samples, welche in C++ Arrays als Fließkommazahlen gespeichert sind.

Dieser Schritt ließe sich in zukünftigen Implementierungen umgehen, indem die Parameter direkt über das Audioboard eingegeben und die Listen mithilfe eines helper-scripts berechnet werden.

Diese Arrays beinhalten bis zu 9000 Einträge, sodass bereits ein LUT, das Werte für vier Delay-Lines enthält, auf eine Dateigröße von ca. 144 kB kommt. Der für die Programmierung vorgesehene Flash-Speicher des „Daisy Seed“ beträgt allerdings nur 128 kB. Für die Implementierung der Simulation werden neun Delay-Lines benötigt. Das Problem kann umgangen werden, indem das Programm nicht auf dem Flash-Speicher, sondern auf dem SRAM-Speicher gespeichert und von dort aus ausgeführt wird. Der Speicher weist eine ähnliche Lesegeschwindigkeit wie der Flash-Speicher auf und hat eine Größe von 480 kB (Electrosmith, 2021). Um den Speicher zu nutzen, muss eine entsprechende Anweisung in das Makefile eingefügt werden.

Codeblock 4: Anweisung zur Speicherung und Ausführung des Programms über SRAM
Quelle: Getting started – Daisy Bootloader

```
APP_TYPE = BOOT_SRAM
```

Abbildung 8 zeigt ein schemenhaftes Signalflussdiagramm der Implementierung auf dem Audioboard. Der Frequenzweichenbaustein besteht aus zwei IIR-Filtern zweiter Ordnung, einem Hochpass- und einem Tiefpassfilter. Die entsprechenden Filterkoeffizienten wurden mithilfe von Matlab erstellt, sodass bei einer Dämpfung von 12 dB/Oktave die Grenzfrequenz jeweils bei 800 Hz liegt. So soll die Frequenzweiche des originalen Leslies mit möglichst geringem Rechenaufwand nachgebildet werden.

Um das Absorptionsverhalten des Holzes wiederzugeben, wird ein FIR-Filter in Erwägung gezogen, der diese Eigenschaften nachbildet und auf den Signalweg der Spiegelschallquellen angewendet werden kann. Ein Blick auf den frequenzabhängigen Absorptionsgrad von Holz zeigt allerdings, dass Holz vor Hohlraum bei Frequenzen ab 1000 Hz bereits über 93 % der Schallenergie reflektiert (schweizer-fn). Das entspricht einer Dämpfung von weniger als 1 dB und deshalb ist davon auszugehen, dass die Auswirkungen dieses Filters in diesem Frequenzbereich nicht hörbar wären.

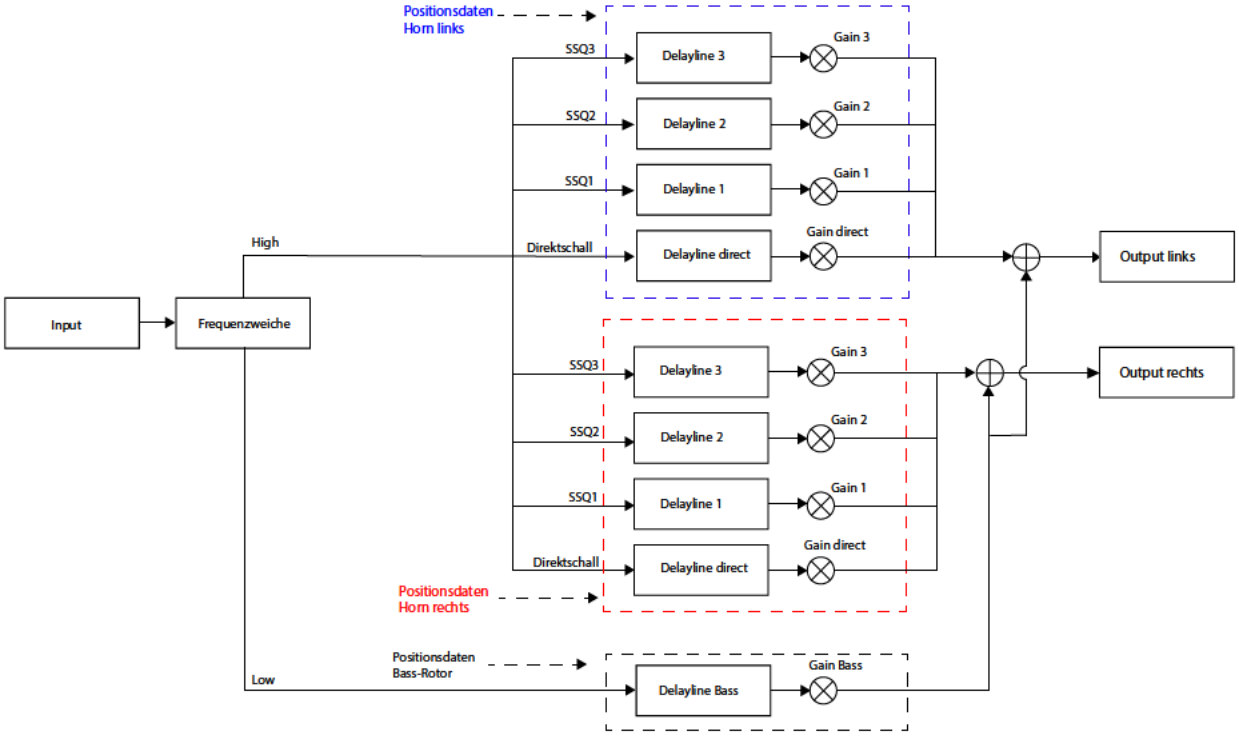

Abbildung 8: Signalflussdiagramm der Simulation auf Audioboard
Quelle: Eigene Darstellung

Die Signale werden durch variable Delay-Lines verarbeitet und mit einem zeitabhängigen Verstärkungsfaktor multipliziert. Die Positionsdaten des Horns, bzw. des Bass-Rotors werden jeweils durch einen Rotationsindex angegeben. Dieser gibt an, an welcher Position die Werte für Delay-Zeiten und Verstärkung aus dem LUT ausgelesen werden sollen, und gibt somit auch indirekt an, auf welcher Position sich das virtuelle Horn bzw. der Bass-Rotor befindet. Über den Rotationsindex kann auch die Rotationsgeschwindigkeit gesteuert werden, ohne dafür zusätzliche LUTs oder Rechenoperationen zu benötigen. Wird zum Beispiel nur jeder vierte Wert des LUTs ausgelesen, wird nur ein Viertel der Zeit für eine ganze Rotation benötigt und die Rotationsgeschwindigkeit erhöht sich um den Faktor Vier. Auf die gleiche Art kann ein Wert mehrmals ausgelesen werden und die Geschwindigkeit so verringert werden. Werden die Werte mit einer Abtastfrequenz von 48 kHz berechnet, hat dieses Vorgehen erst beim Überspringen bzw. Wiederholen von ca. 50-60 Werten hörbare Auswirkungen. Die Delay-Zeiten werden mit dem zuvor erwähnten Matlab Skript „`Leslie_LUT_Generator`“ (siehe Anhang 4) für eine Rotationsfrequenz von 6,92 Hz für den Horn-Rotor für jeweils den linken und rechten Kanal und 5,9 Hz für den Bass-Rotor berechnet (Herrera, et al., 2009).

Bei einem originalen Leslie-Lautsprecher ist die Rotationsgeschwindigkeit in der Einstellung *Tremolo* etwa um den Faktor Zehn größer als in der Einstellung *Chorale* (siehe Kapitel 2.3). Um Speicherplatz auf dem Audioboard einzusparen, werden die Delay-Zeiten im `Leslie_LUT_Generator` mit der

Rotationsgeschwindigkeit der Einstellung *Tremolo* berechnet. Wird dann im Betrieb der Simulation jeder Wert der Liste zehnmal ausgelesen, resultiert daraus eine Rotationsgeschwindigkeit, die der Einstellung *Chorale* entspricht.

Der Hauptteil des C++ Codes besteht aus vier Teilen. Der erste Teil besteht aus obligatorischen Deklarationen von Variablen und Verweisen zur Einbindung externer Libraries und Datensätze. Aus Gründen der Übersichtlichkeit ist dieser Teil im Anhang 5 dieser Arbeit zu finden. Der zweite Teil besteht aus dem „Audio Callback Loop“. In diesem wird durch eine `for`−Schleife eine bestimmte Anzahl Samples verarbeitet. Dort findet die hauptsächliche Signalverarbeitung statt. Der dritte Teil besteht aus der `main`-Funktion. In ihr werden Funktionen und Hardware-Schnittstellen initialisiert. Auch der Audio Callback Loop wird von dort abgerufen. Der letzte Teil besteht aus den Funktionen für die IIR-Filter, auf die aus dem Audio Callback Loop aus zugegriffen wird, um das Eingangssignal in tiefe und hohe Frequenzen zu trennen.

Codeblock 9: Leslie-Simulation (I): Main-Funktion
Quelle: Eigene Darstellung

```
int main(void)
{
    // initialize seed hardware and daisysp modules
    float sample_rate;

    hw.Configure();
    hw.Init();
    hw.SetAudioBlockSize(128);

    sample_rate = hw.AudioSampleRate();

    //Initialize the button on pin 28
    button1.Init(hw.GetPin(28), sample_rate / 48.f);

    // Initialize delaylines
    deld_L.Init();
    del1_L.Init();
    del2_L.Init();
    del3_L.Init();

    deld_R.Init();
    del1_R.Init();
    del2_R.Init();
    del3_R.Init();
    deldbass.Init();
```

```
    // start callback
    hw.StartAudio(AudioCallback);

    while(1) {}
}
```

In Codeblock 9 wird die Hardware initialisiert und über den Befehl `SetAudioBlockSize` die Anzahl an Samples festgelegt, die vom Audio Callback Loop verarbeitet werden, bevor sie über den Digital/Analog-Wandler ausgegeben werden. Je höher die Blockgröße ist, desto höher ist auch die Latenz des Gesamtsystems. Bei aufwändiger Signalverarbeitung kann es allerdings passieren, dass nicht alle Samples des Blocks rechtzeitig verarbeitet werden, bevor sie ausgegeben werden sollen, sodass es zu Fehlern bei der Wiedergabe kommt. Die Firmware des Audioboards erlaubt die Verarbeitung von Blockgrößen von 2 bis 128 Samples. Die voreingestellte Samplerate des Audioboards beträgt 48 kHz. Bei einer Blockgröße von 128 Samples beträgt die dadurch verursachte Latenz also

$$t = \frac{B_{size}}{f_s} = \frac{128}{48000\ Hz} \approx 0{,}0027\ s \approx 2{,}7\ ms \tag{10}$$

Die Initialisierung von Pin 28 dient der Implementierung eines Tasters, der dafür verwendet wird, die Rotationsgeschwindigkeit zwischen *Chorale* und *Tremolo* umzuschalten. Daraufhin erfolgt die Initialisierung der Delay-Lines. Dabei handelt es sich um eine Klasse aus der Daisy Library, in der die nötige Funktionsweise der Delay-Lines, inklusive Sample-Interpolation bei Fractional Delay-Lines, bereits implementiert ist. Der Befehl `StartAudio` gibt die verarbeiteten Samples aus dem Audio Callback Loop wieder.

Codeblock 10: Leslie-Simulation (II): Filterfunktionen
Quelle: Eigene Darstellung

```
double filterH (double a0, double a1, double a2, double b1, double b2,
float input)
{
    double outd;
    outd = input * a0 + z1H;
    z1H = input * a1 + z2H - b1 * outd;
    z2H = input * a2 - b2 * outd;

    return outd;
}

double filterL (double a0, double a1, double a2, double b1, double b2,
float input)
{
```

```
    double outd;
    outd = input * a0 + z1L;
    z1L = input * a1 + z2L - b1 * outd;
    z2L = input * a2 - b2 * outd;

    return outd;
}
```

Codeblock 10 zeigt die Funktionen für die beiden IIR-Filter der Frequenzweiche. Dafür wird die Darstellung der transponierten Direktform 2 mit zwei Speichergliedern gewählt. Diese Form weist bessere Eigenschaften beim Umgang mit Fließkommazahlen auf als die Direktform 1 und bietet durch die geringe Zahl an Speichergliedern eine gute Recheneffizienz (Redmon, 2012). Um die Filterkoeffizienten zu ermitteln, wird die integrierte Matlab Funktion `FilterBuilder` verwendet (Mathworks, 2022). Die Frequenzweiche besteht aus zwei IIR-Filtern zweiter Ordnung, einem Hoch- und einem Tiefpassfilter. Dadurch wird eine Dämpfung von 12 dB/Oktave erreicht, was auch der Dämpfung der Frequenzweiche eines originalen Leslie-Lautsprechers entspricht. Die 3 dB Grenzfrequenz wird mit 800 Hz angegeben und mit der Butterworth Methode gestaltet.

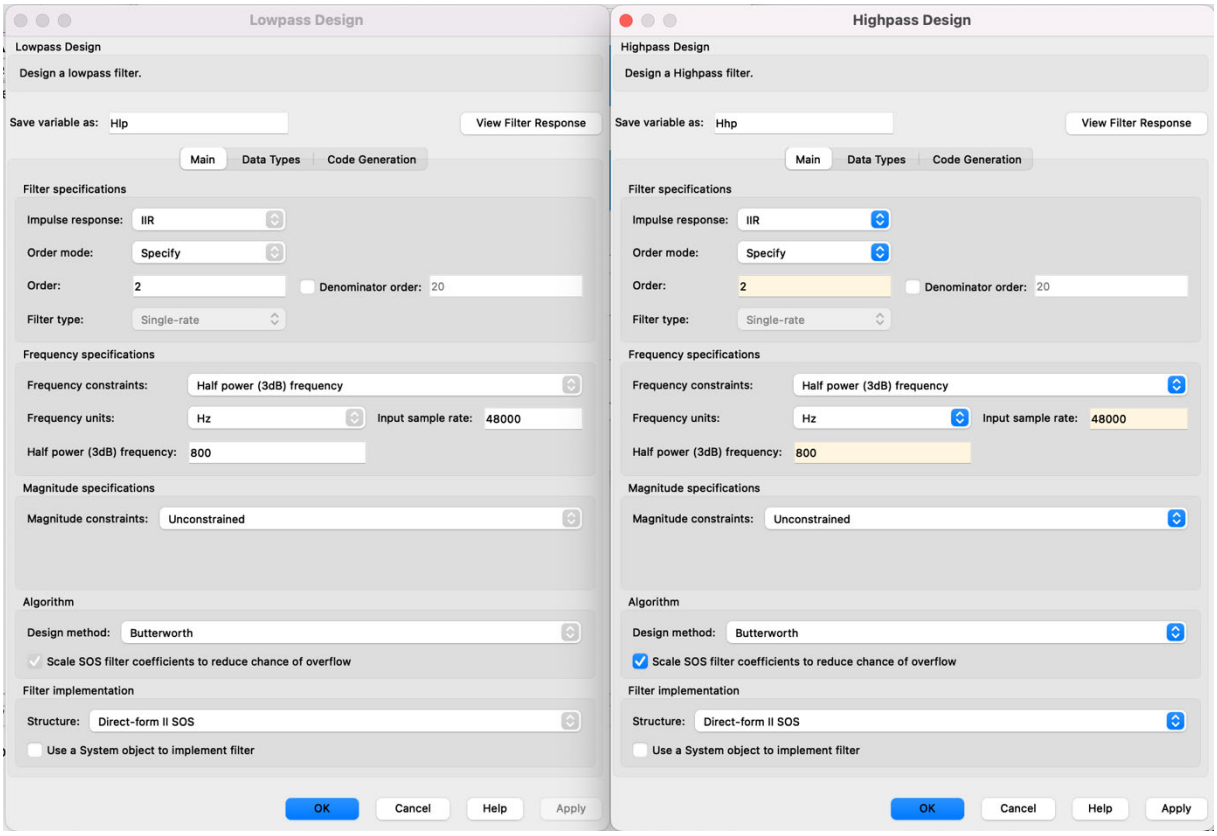

Abbildung 9: Gestaltungsparameter für IIR-Filter der digitalen Fequenzweiche
Quelle: Eigene Darstellung

Mithilfe der Matlab Funktion *fvtool* wird der Amplitudenfrequenzgang der beiden Filter in einem Plot dargestellt.

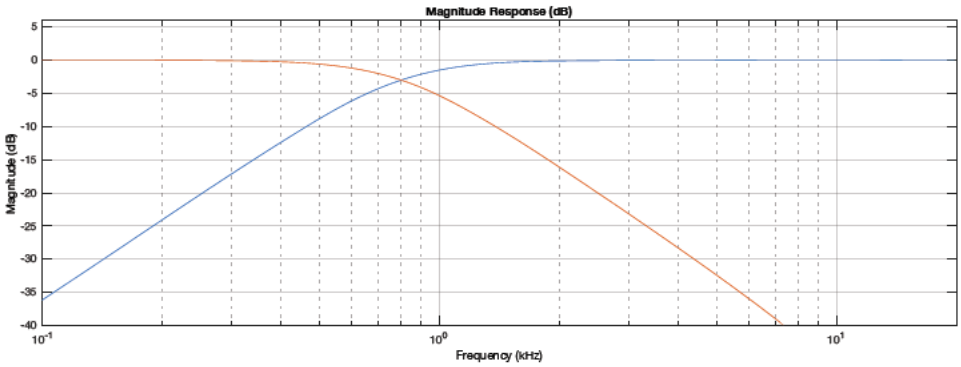

Abbildung 10: Amplitudenfrequenzganz für IIR-Filter der digitalen Frequenzweiche
Quelle: Eigene Darstellung

Die beiden Filter zeigen das gewünschte Verhalten für Dämpfung und Grenzfrequenz. Die Filterkoeffizienten werden im ersten Teil des Codes als `double` Variablen deklariert, um darauf bei der Ausführung der Filterfunktionen zugreifen zu können. Der Datentyp `double` ist in diesem Fall besser geeignet als `float`, weil dieser die doppelte Anzahl an Bits enthält und eine Genauigkeit von maximal 15 Nachkommastellen besitzt, statt sechs wie im Fall von `float` (vgl. Neumann 2020: S. 40). Die höhere Genauigkeit verbessert die Leistung der Filter, weil Rundungsfehler dadurch reduziert werden.

Codeblock 11: Leslie-Simulation (III): Audio Callback Loop (I)
Quelle: Eigene Darstellung

```
static void AudioCallback(AudioHandle::InterleavingInputBuffer  in,
                          AudioHandle::InterleavingOutputBuffer out,
                          size_t                                size)
{
    //loadMeter.OnBlockStart();
    for(size_t i = 0; i < size; i += 2)
    {

        //input variable
        in1 = in[i+1];

        float outputLow = filterL(a0L, a1L, a2L, b1L, b2L, in1);
        float outputHigh = -filterH(a0H, a1H, a2H, b1H, b2H, in1);

        //set current delaytime based on position index
        currentDelayd_horn_L = delaytime_direct_horn_L[rotation];

        // Write to the delay
```

```
        deld_L.Write(outputHigh);

        deld_L.SetDelay(currentDelayd_horn_L);

        //calculate Gain
        gain_direct_L = (48000/(currentDelayd_horn_L*csound))/5.8812f;

        // Read from delay line
        deld_out_horn_L = deld_L.Read()*gain_direct_L;
```

Die Attribute der `AudioCallback` Funktion `InterleavingOutputBuffer` und `InterleavingInputBuffer` geben an, dass die Audiosamples am Eingang und Ausgang eindimensional vorliegen. Alle geraden Indizes der Sampleblocks beinhalten also den ersten Kanal, während alle ungeraden Indizes den zweiten Audiokanal beinhalten. Durch die `for`-Schleife werden die Samples nacheinander verarbeitet.

Um eine bessere Übersichtlichkeit zu gewähren, zeigt Codeblock 11 exemplarisch die Verarbeitung einer Delay-Line. Im vollständigen Code gibt es davon insgesamt neun. Vier Delay-Lines pro Kanal, jeweils eine simulierte Direktschallquelle und drei Spiegelschallquellen. Zusätzlich gibt es eine separate Delay-Line, um den Bass-Rotor zu simulieren. So können Horn- und Bass-Rotor mit unterschiedlichen Rotationsgeschwindigkeiten simuliert werden. Der vollständige Code ist als Text im Anhang 5 und als .cpp Datei auf dem beiliegenden Datenträger im „Audioboard"-Verzeichnis zu finden.

Zunächst wird das Signal durch die Frequenzweiche in zwei Frequenzbereiche aufgeteilt. Der Grund für das negative Vorzeichen des Hochpassfilters wird in Kapitel 4.1 erläutert. Anschließend wird die aktuelle Delay-Zeit aus dem LUT-Array vom SRAM-Speicher ausgelesen und in der Hilfsvariablen `currentDelay_horn` gespeichert. Die Sample-Werte werden dann mithilfe des Befehls `delayline.write` in die Delay-Lines geschrieben und über `delayline.SetDelay` bestimmt, um wie viele Abtastzeiten diese verzögert werden sollen. Um die Gain-Faktoren zu bestimmen, wird aus der Variablen T, die für die Zeit steht, die der Schall braucht, um die Entfernung zwischen Schallquelle und Hörposition zurückzulegen, die Entfernung d berechnet. Dazu wird die Formel für T nach d umgestellt, um daraus die Verstärkungsfaktoren mit $\frac{1}{d(n)}$ für die Direktschallquellen und $\frac{1}{\sqrt{d(n)}}$ für die Spiegelschallquellen zu berechnen. Die Verstärkungsfaktoren werden mit dem größten auftretenden Wert normiert.

$$d(n) = \frac{T(n) * c_{sound}}{f_s} \tag{11}$$

$$g_{direct} = \frac{1}{d(n)} * \frac{1}{d(n)_{max}} = \frac{f_s}{T(n) * c_{sound}} * \frac{1}{d(n)_{max}} \tag{12}$$

$$g_{SSQ} = \frac{1}{\sqrt{d(n)}} * \frac{1}{d(n)_{max}} = \frac{1}{\sqrt{\frac{T(n)*c_{sound}}{f_s}}} * \frac{1}{d(n)_{max}} \quad (13)$$

Mit dem Befehl `delay.read` wird das aktuelle Ausgangssample aus der Delay-Line gelesen und nach Multiplikation mit dem Gain-Faktor in einer neuen Variablen `del_out_horn` gespeichert.

Codeblock 12: Leslie-Simulation (III): Audio Callback Loop (II)
Quelle: Eigene Darstellung

```
        //  position of speakers rotation and speed control
        counter ++; //helper variable to get rotation speed from sample
progression


        if (counter > skip) {        //if helper variable is larger than
speed variable advance on delaytime by one sample
            rotation += 2;
            rotation_bass += 2;
            counter = 0;
        }
        fonepole(skip, skip_target, 0.000020833f); //low pass filter to
simulate rotation dynamics


        if (rotation >= 7200) {          // reset LUT counter variable af
ter full rotation
            rotation = 0;
            rotation_count ++;
        }


        if (rotation_bass >= 9001) {
            rotation_bass = 0;
        }


        //button push
        button1.Debounce();


        if(button1.RisingEdge()) {
            skip_target = 10;
        }


        if(button1.FallingEdge()) {
            skip_target = 1;
        }
```

Die Variable `counter` dient dazu, die Durchgänge der `for`-Schleife zu zählen und so einen Bezug zur Rotationsgeschwindigkeit herzustellen. Die Variable `skip` gibt an, nach wie vielen Durchgängen der Schleife zum nächsten Wert im LUT übergegangen werden soll. Ein `skip`-Wert von 10 ergibt eine Rotationsgeschwindigkeit, die um den Faktor Zehn geringer ist als die Geschwindigkeit, mit der die Werte berechnet wurden. Über die Betätigung eines angeschlossenen Tasters kann zwischen den Werten 1 und 10 hin- und hergeschaltet werden, was in etwa dem Geschwindigkeitsunterschied zwischen *Tremolo* und *Chorale* eines originalen Leslies entspricht. Weil in der Realität die Geschwindigkeitsänderung der Motoren nicht sofort eintritt, sondern eine gewisse Zeit vergeht, bis die Rotoren ihre Zielgeschwindigkeit erreichen, wird beim Umschalten zwischen *Chorale* und *Tremolo* der `skip`-Wert mithilfe eines FIR-Tiefpassfilters 1. Ordnung geändert. Mit `fonepole` ist ein solches Filter in die Daisy Library integriert. Der Filterkoeffizient wird mithilfe der Formel

$$b = \frac{1}{t * f_s} \tag{14}$$

berechnet, wobei t für die Dauer des Übergangs zwischen den beiden Geschwindigkeitsmodi steht. Die Formel wurde aus der Dokumentation der Daisy Library entnommen (Electrosmith, 2021). Für die Übergangsdauer wurde die Messung aus dem Paper von Herrera et al. (2009) verwendet und beträgt 1 s. Durch `if`-Statements werden Rotationsindizes nach Vollendung einer ganzen Umdrehung wieder auf 1 zurückgesetzt.

Codeblock 13: Leslie-Simulation (III): Audio Callback Loop (III)
Quelle: Eigene Darstellung

```
        // Calculate output gain of horn rotor
        sig_out_horn_L  = (deld_out_horn_L + del1_out_horn_L+ del2_out_horn_L
+ del3_out_horn_L);
        sig_out_horn_R      =    (deld_out_horn_R    +    del1_out_horn_R    +
del2_out_horn_R + del3_out_horn_R);


        sig_out_L = sig_out_horn_L + deld_out_bass;
        sig_out_R = sig_out_horn_R + deld_out_bass;


        // Output
        out[LEFT]  = sig_out_L;
        out[RIGHT] = sig_out_R;
```

Die Ergebnisse der Delay-Lines der einzelnen Kanäle werden addiert und mit den Variablen sig_out_L und sig_out_R zusammengefasst. Der Ausgang des Bass-Rotors wird auf beide Kanäle addiert. Die zusammengefassten Signale werden an die entsprechenden Ausgangskanäle weitergegeben.

3.4 Hardware-Implementierung

Das vorgestellte System wird für den Live-Gebrauch in Live- oder Probesituationen konzipiert und erfordert deshalb eine hinreichend einfache und mobile Hardware-Konfiguration, um praktikabel zu sein.

Das Ziel besteht darin, die digitale Simulation durch eine Hardware-Konfiguration zu ergänzen, die dem Effekt eine bessere Wirkung verleiht und dafür sorgt, dass die räumliche Wirkung eines elektromechanischen Rotationslautsprechers trotz stationärer Schallwandler zur Geltung kommt. Im Gegensatz zu den in Kapitel 2.4 erwähnten Ansätzen zur Modellierung soll das Modell in diesem Fall nicht für die Simulation des Effekts auf einer konventionellen Beschallungsanlage, wie Kopfhörern oder Stereolautsprechern, konzipiert werden. Es soll ein eigenständiges System darstellen, welches unabhängig von zusätzlicher Hardware operieren kann, wie sein analoges Vorbild. Durch die Modularität der einzelnen Komponenten kann auf individuelle Anforderungen reagiert werden. So können für die Beschallung größerer Räume leistungsstärkere Verstärker und Lautsprecher verwendet werden, während die grundlegende Signalverarbeitung gleichbleibt.

Dabei sollen, bis auf die rotierenden Schallabstrahler, charakteristische Eigenschaften des originalen Leslie-Lautsprechers auf das Modell übertragen werden. Das verarbeitete Stereosignal wird in zwei Frequenzbereiche mit einer Grenzfrequenz von 800 Hz aufgeteilt und an unterschiedliche Lautsprecher verteilt. Um die tiefen Frequenzen wiederzugeben, wird, wie beim Original, ein 15“ Lautsprecher verwendet.

3.4.1 Signalfluss

Als Signalquelle wurde in diesem Aufbau ein Audio-Player verwendet. Es ist jedoch möglich, jede Art von Signal mit Line-Pegel zu verwenden. Da das Audioboard nur über zwei Ausgangskanäle verfügt, muss das Signal danach in zwei Frequenzbereiche aufgetrennt werden, um es an die entsprechenden Lautsprecher verteilen zu können. Diese Frequenztrennung ließe sich auch mit passiven Bauelementen umsetzen, was Formfaktor und Komplexität der Konfiguration erheblich verringern würde. Aus zeitlichen Gründen wird dafür jedoch das digitale Audiomischpult SI Expression II verwendet, da sich Grenzfrequenz und Güte der Filter damit unkompliziert und schnell einstellen lassen. Nachdem die Einstellungen am Mischpult vorgenommen sind, wird deren Funktion mithilfe des APx-515 Messgerätes überprüft (siehe Abbildung 11).

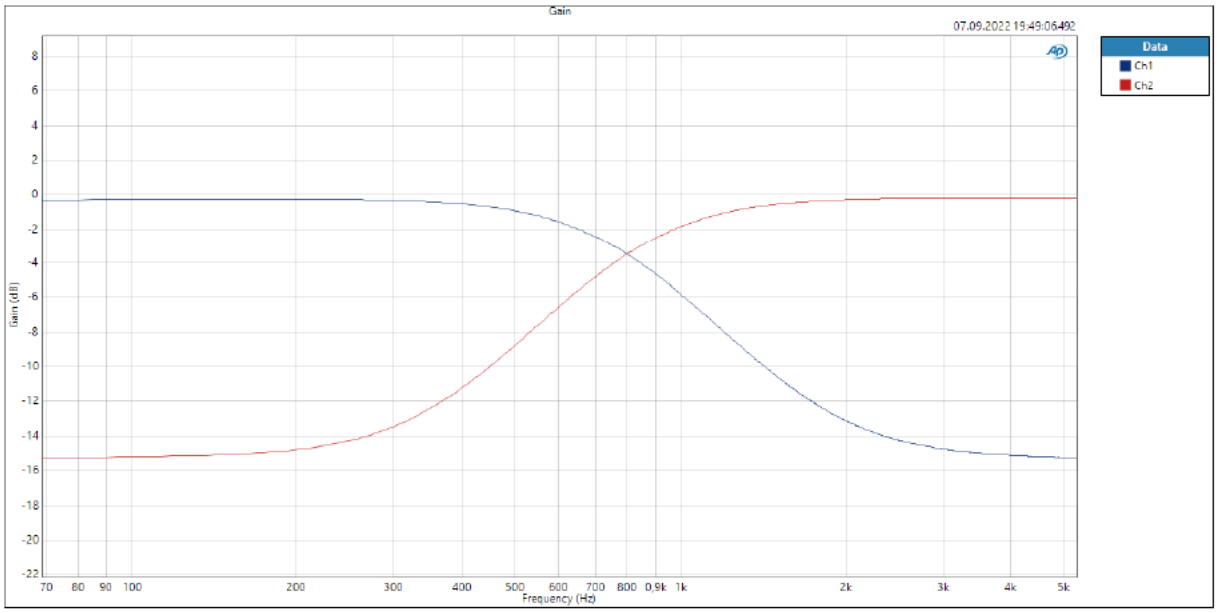

Abbildung 11: Frequenztrennung vom Ausgangssignal des Audioboards
Quelle: Eigene Darstellung

Das Hochtonsignal wird an zwei aktive Breitbandlautsprecher gesendet. Das Tieftonsignal wird auf einen Kanal zusammengemischt und in einen Combo-Bassverstärker gesendet, der über einen 15“ Lautsprecher verfügt.

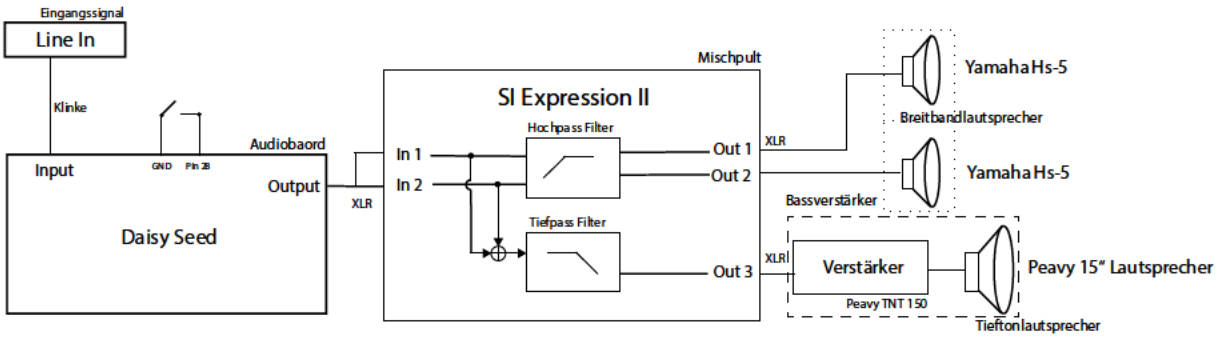

Abbildung 12: Hardware-Implementierung der Leslie-Simulation als Blockdiagramm
Quelle: Eigene Darstellung

Eine Stereo-Audioaufnahme des Ausgangssignals des Audioboards ist auf dem beiliegenden Datenträger unter dem Dateinamen „Hörbeispiel_6“ zu finden. Während der Aufnahme wird mehrmals zwischen den Geschwindigkeitsmodi *Chorale* und *Tremolo* gewechselt.

3.4.2 Lautsprecheranordnung

Um einen Bezug zwischen der digitalen Simulation und der physischen Lautsprecheranordnung herzustellen, wird der Abstand zwischen den stationären Lautsprechern so gewählt, dass er mit der Breite eines Leslie-Lautsprechers übereinstimmt. Die Hörpositionen, mit denen die Simulation betrieben wird,

liegen dann genau auf dem Rand des Gehäuses, was in der realen Lautsprecheranordnung der Position der Lautsprechermembran entspricht.

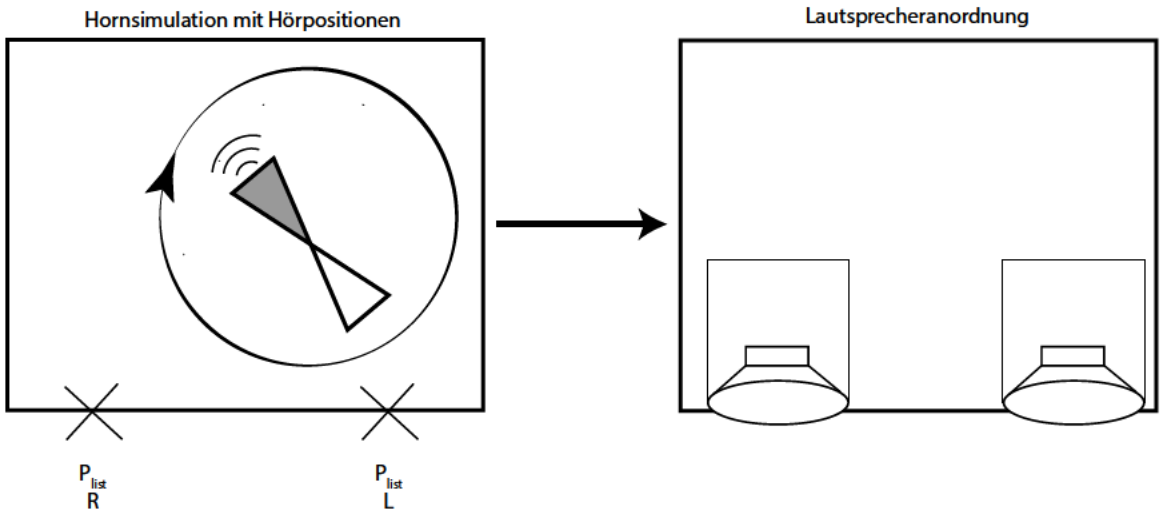

Abbildung 13: Bezug zwischen simulierten Hörpositionen und Lautsprecheranordnung
Quelle: Eigene Darstellung

Die Delay-Zeiten im Leslie_LUT_Generator werden jeweils für die Koordinaten der Hörposition $P_{list}R$ für den rechten Kanal und den Koordinaten der Hörposition $P_{list}L$ für den linken Kanal berechnet. Die notwendigen Maße des Gehäuses werden anhand von geeigneten Fotos des Lautsprechers ermittelt.

Das gleiche geschieht mit dem Bass-Rotor, wobei in dem Fall nur eine Hörposition simuliert wird. Die Hörposition befindet sich dann in der Mitte, was ebenfalls der tatsächlichen Lautsprecheranordnung entspricht.

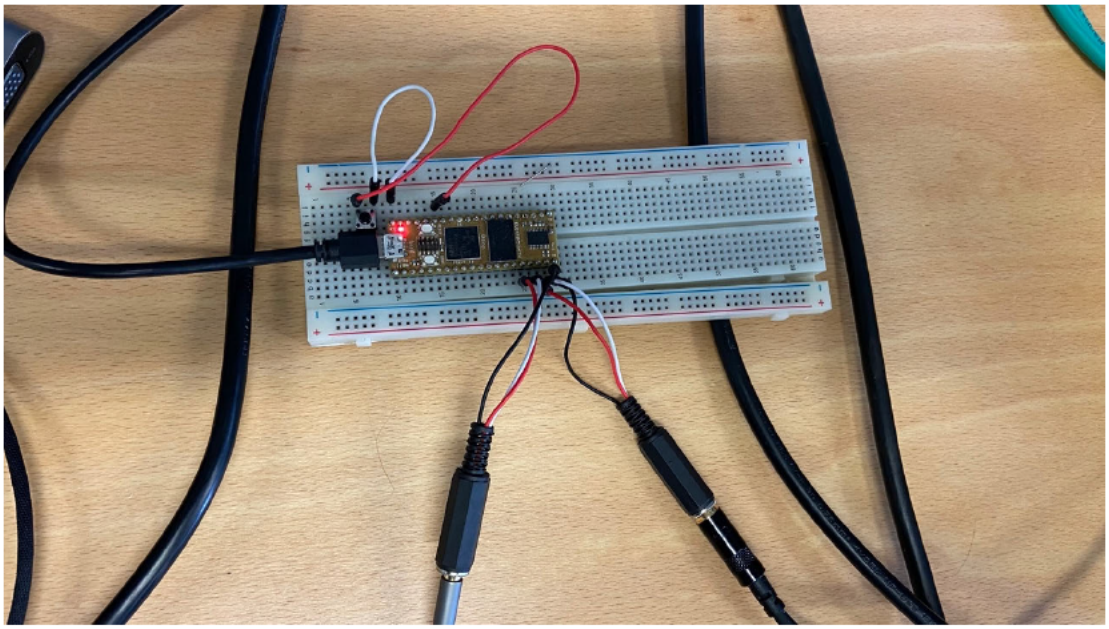

Abbildung 14: Angeschlossenes Audioboard
Quelle: Eigene Darstellung

Abbildung 15: Aufbau des gesamten Systems
Quelle: Eigene Darstellung

Eine Stereo-Audioaufnahme des Audiosignals in ca. 1 m Entfernung vor der Lautsprecheranordnung wird mit dem Zoom H4N Pro Recorder aufgezeichnet und ist auf dem beiliegenden Datenträger unter dem Dateinamen „Hörbeispiel_7“ zu finden.

4 Evaluierung

4.1 Messungen

Um Messungen am Audioboard durchzuführen, wird es – wie in Abbildung 16 zu sehen ist – mit dem APx-515 Audio Analyzer verbunden. Dafür werden die 3,5 mm Stereobuchsen, die an das Audioboard angeschlossen sind, auf separate XLR-Stecker für beide Kanäle adaptiert.

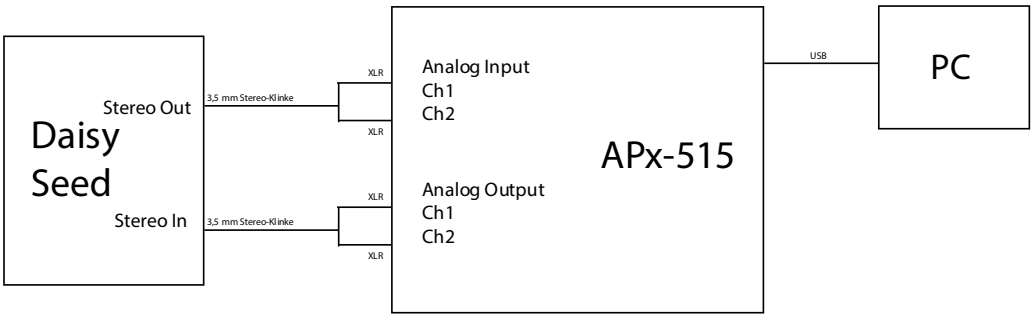

Abbildung 16: Blockdiagramm zum Messaufbau des Audioboards
Quelle: Eigne Darstellung

Um die Echtzeitfähigkeit des Systems sicherzustellen, wird die Latenz zwischen Ein- und Ausgang des Audioboards gemessen. Dafür wird über die APx-515 Software eine DUT-Delay Messung durchgeführt. Dabei wird die Verzögerung einer Maximallängensequenz (Dickreiter, et al., 2014, p. 641) zwischen Ein- und Ausgang des Messgerätes bestimmt (Audioprecision, 2019, p. 315 f.). Da es sich bei dem Modell um ein zeitabhängiges System handelt, würde eine Messung bei laufender Simulation zu ungenauen Ergebnissen führen oder eine Korrelation des Testsignals wäre gar nicht erst möglich. Deshalb werden die Maximalwerte der Delay-Zeiten in den Delay-LUTs ermittelt und alle Delay-Lines auf diese statischen Werte gesetzt. Dadurch wird aus der Simulation ein lineares, zeitinvariantes System, an dem die DUT Delay-Messung durchgeführt werden kann. Die Sample-Blockgröße wird ebenfalls auf den größtmöglichen Wert von 128 Samples pro Block gesetzt, um die größtmögliche Verzögerung zu messen, die durch das System verursacht werden kann.

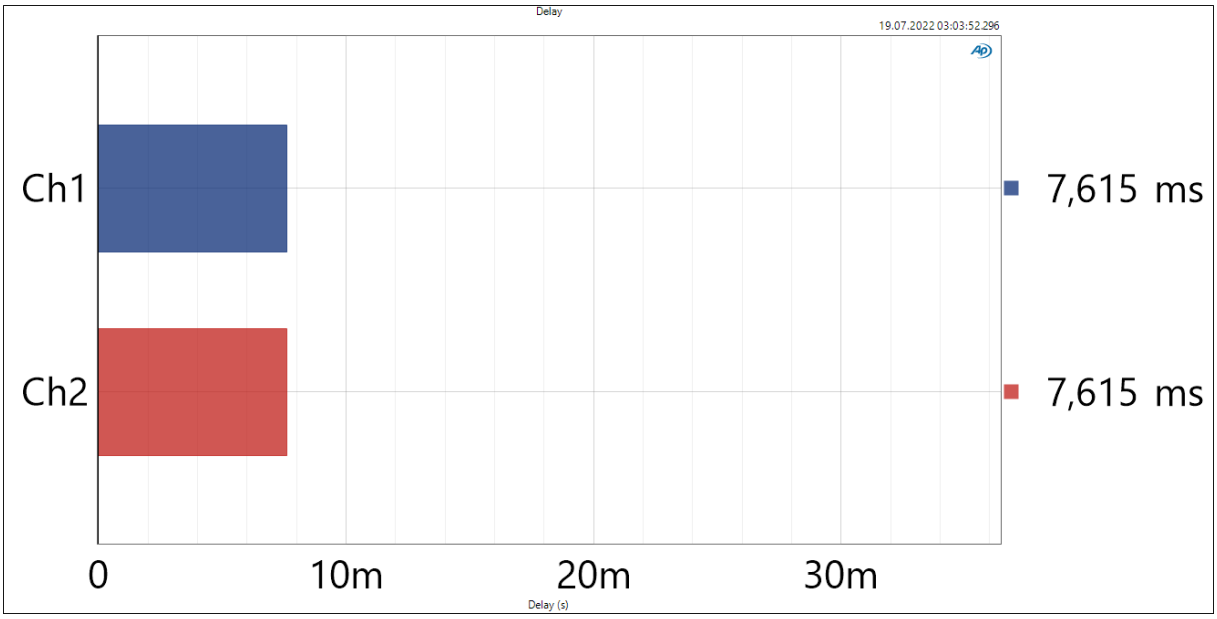

Abbildung 17: DUT-Delay Messung am Audioboard
Quelle: Eigene Darstellung

Wie auf Abbildung 17 zu sehen ist, beträgt die Latenz des Audioboards unter den genannten Umständen 7,615 ms. Die Latenz des verwendeten Mischpultes zur Frequenztrennung beträgt laut Datenblatt weniger als 0,8 ms (Soundcraft, 2020). Damit beträgt die Gesamtlatenz des Systems maximal 8,415 ms und die Anforderung der Echtzeitfähigkeit wird erfüllt. Ein denkbarer Entwicklungsschritt in einer zukünftigen Version des Systems wäre die Umsetzung der Frequenztrennung mit passiven Bauelementen, wodurch sich sowohl die Latenzzeit als auch der Formfaktor verringern würden.

Darüber hinaus ist es möglich, die Prozessorauslastung des Audioboards auszulesen und anzuzeigen. Dafür gibt es in der Daisy Library eine vorgefertigte Helper-Klasse namens `loadMeter`. Damit lässt sich aus der Zeit, die das Audioboard benötigt, um die `for`-Schleife zu bearbeiten, eine Auslastung des Prozessors messen und über einen seriellen Monitor ausgeben (Electrosmith, 2021). Die Messung wird bei laufender Simulation ausgeführt und mithilfe des seriellen Monitors der Arduino Software angezeigt.

```
Processing Load :
Max:  12.292
Avg:  12.209
Min:  12.047
```

Abbildung 18: Prozessorauslastung des Audioboards bei laufender Simulation
Quelle: Eigene Darstellung

Wie in Abbildung 18 zu sehen ist, beträgt die Auslastung des Prozessors 12,2 % im Durchschnitt. Die Messung wurde bei einer Blockgröße von 128 ausgeführt. Das bedeutet, dass es möglich ist, weitere Rechenoperationen der Signalverarbeitung durchzuführen, ohne dass es zu Fehlern in der Audiowiedergabe kommt. Eine Möglichkeit wäre, Spiegelschallquellen höherer Ordnung zu implementieren, allerdings müsste die Größe der Delay-LUTs dann weiter reduziert oder komprimiert werden. Die Speicherauslastung der vorgestellten Implementierung beträgt ca. 68 %.

Memory region	Used Size	Region Size	%age Used
FLASH:	0 GB	128 KB	0.00%
DTCMRAM:	50720 B	128 KB	38.70%
SRAM:	333568 B	480 KB	67.86%
RAM_D2_DMA:	16 KB	32 KB	50.00%
RAM_D2:	0 GB	256 KB	0.00%
RAM_D3:	0 GB	64 KB	0.00%
ITCMRAM:	0 GB	64 KB	0.00%
SDRAM:	0 GB	64 MB	0.00%
QSPIFLASH:	0 GB	7936 KB	0.00%

Abbildung 19: Speicherauslastung des Audioboards
Quelle: Eigene Darstellung

Um ein genaues Bild der Funktion der digitalen Filter auf dem Audioboard zu bekommen, wird die betreffende Messung isoliert vom restlichen Code ausgeführt.

Codeblock 14: Code für die Durchführung der Messung an IIR-Filter
Quelle: Eigene Darstellung

```
        //input variable
        in1 = in[i+1];

        float outputLow = filterL(a0L, a1L, a2L, b1L, b2L, in1);
        float outputHigh = filterH(a0H, a1H, a2H, b1H, b2H, in1);

        sig_out = outputLow + outputHigh;

        // Output
        out[LEFT]  = sig_out;
        out[RIGHT] = sig_out;
    }
```

Da das System für einkanalige Signale ausgelegt ist, wird nur der rechte Kanal durch die Filter verarbeitet. Die Messung wird mithilfe eines MLS-Signals durchgeführt, welches am APx-515 Audio Analyzer erzeugt und in den Eingang des Audioboards gespeist wird. Das Signal am Ausgang des Audioboards wird von der APx Software mit dem Eingangssignal korreliert und daraus eine komplexe Übertragungsfunktion erstellt.

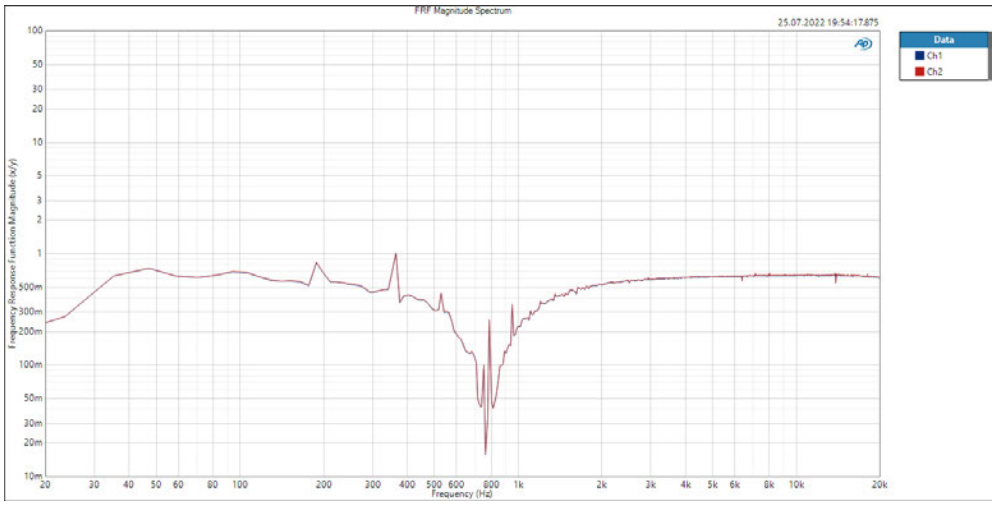

Abbildung 20: Amplitudenfrequenzgang der digitalen Frequenzweiche
Quelle: Eigene Darstellung

Wird der Amplitudenfrequenzgang des Filters betrachtet, ist ein deutlicher Einbruch bei der Grenzfrequenz von 800 Hz zu sehen. Das Signal bricht an dieser Stelle von ca. 500 mV auf bis zu ca. 5 mV ein, was einer Dämpfung von 40 dB entspricht.

$$L = 20log_{10}\frac{U_1}{U_2} = -40\ dB \tag{15}$$

Da die 3 dB Grenzfrequenz beider Filter bei 800 Hz liegen sollte, ist bei dieser Frequenz eine Dämpfung von 6 dB zu erwarten. Aufgrund der hohen Abweichung zwischen vorhergesagten und gemessenen Werten wird die Funktion der Filter genauer untersucht.

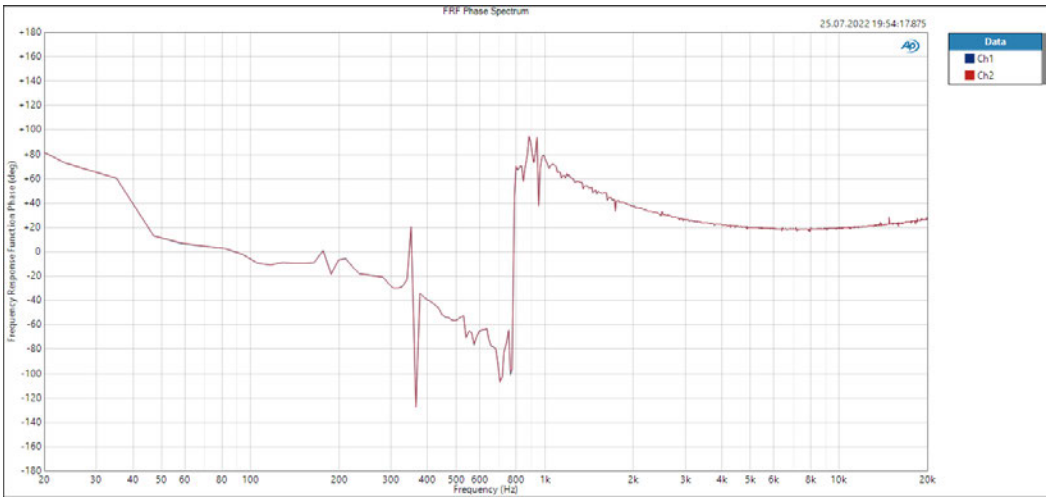

Abbildung 21: Phasenfrequenzganz der digitalen Frequenzweiche
Quelle: Eigene Darstellung

Auf dem Phasenfrequenzgang derselben Messung ist zu sehen, dass die Phase der Frequenzweiche bei der Grenzfrequenz sprunghaft um 180 Grad ansteigt. Diese Phasenverschiebung ist eine mögliche Ursache für die hohe Dämpfung in diesem Frequenzbereich, weil es dadurch zu einer Signalauslöschung im Zeitbereich kommt. (Friesecke, 2014, p. 30). Um diesem unerwünschten Verhalten entgegenzuwirken, wird die Polarität des Ausgangssignals am Hochpassfilter invertiert. Dadurch soll die Auslöschung im Zeitbereich verhindert und die Dämpfung im Grenzbereich verringert werden. Die Implementierung erfolgt über einen Vorzeichenwechsel am Ausgang des Hochpassfilters.

Codeblock 15: Invertierung des Ausgangssignals aus dem Hochpassfilter
Quelle: Eigene Darstellung

```
        float outputLow = filterL(a0L, a1L, a2L, b1L, b2L, in1);
        float outputHigh = -filterH(a0H, a1H, a2H, b1H, b2H, in1);
```

Wenn die Messung mit geändertem Vorzeichen wiederholt wird, ist der sprunghafte Anstieg des Phasengangs nicht mehr zu sehen (Siehe Abbildung 22). Der Blick auf den Amplitudenfrequenzgang zeigt nun ebenfalls einen weitestgehend konstanten Verlauf, wobei dieses Mal eine leichte Erhöhung von knapp 6 dB bei der Grenzfrequenz zu sehen ist. Da durch die Phasenverschiebung der beiden Filtersignale zuvor eine Auslöschung im Zeitbereich verursacht wurde, bewirkt die Invertierung eines der Signale eine Verstärkung an dieser Stelle.

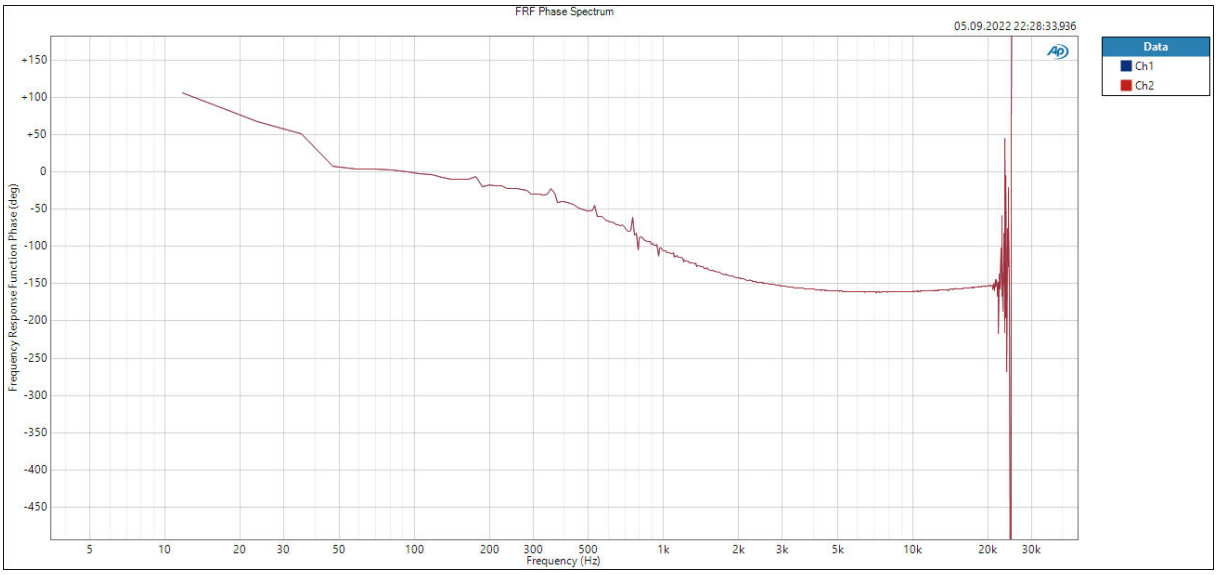

Abbildung 22: Phasenfrequenzgang der digitalen Frequenzweiche mit invertiertem Hochpasssignal
Quelle: Eigene Darstellung

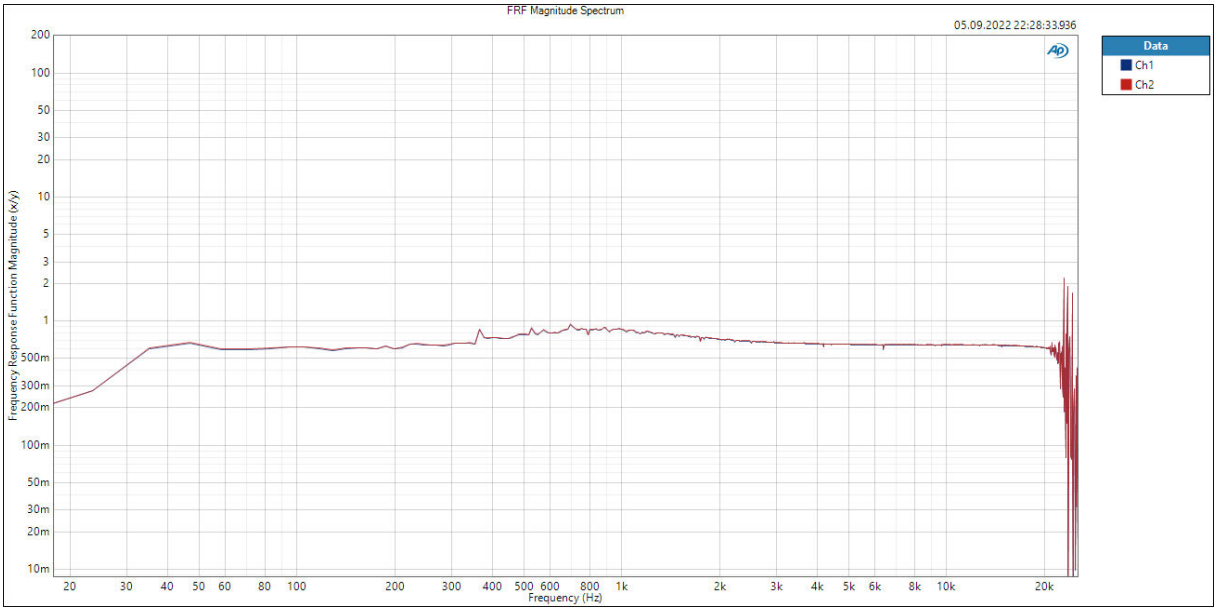

Abbildung 23: Phasenfrequenzgang der digitalen Frequenzweiche mit invertiertem Hochpasssignal
Quelle: Eigene Darstellung

Kronland-Martinet und Voinier (2008) beschreiben in ihrem Artikel eine Messung, die mithilfe eines Testsignals in Form von einem Sinus mit einer Frequenz von 800 Hz an einem Leslie-Lautsprecher durchgeführt wird. Das Signal wird mit einem Mikrofon aufgezeichnet und mithilfe eines Hilbert-Transformators wird ein analytisches Signal erzeugt (Grünigen 2014: S. 37). Das analytische Signal ist gegeben durch

$$Z(t) = s(t) + iH[s](t) = A(t)e^{i\phi(t)} \tag{16}$$

Wobei s(t) das aufgezeichnete Signal und H den Hilbert-Transformator darstellt (Kronland-Martinet & Voinier, 2008). Der Betrag A(t) des Signals Z(t) bildet die Amplitude der Einhüllenden von s(t) zum Zeitpunkt t. Durch die Ableitung der Phase $\frac{d\phi}{dt}$ lässt sich die momentane Frequenz zum Zeitpunkt t ermitteln.

Abbildung 24 zeigt das Ergebnis dieser Messung von Kronland-Martinet und Voinier und es ist zu beachten, dass das Signal ein aperiodisches und komplexes Verhalten aufweist, wobei die Maxima der momentanen Frequenz mit den Minima des Betrages der Amplitude zusammenfallen. Um auszuschließen, dass dieses Verhalten durch Interferenzen zwischen Horn- und Bass-Rotor verursacht wird, führen die Autoren weitere Messungen mit ausgeschaltetem Bass-Rotor durch und kommen auf ähnliche Ergebnisse. Sie schlussfolgern daraus, dass dieses Verhalten durch raumakustische Phänomene innerhalb des Gehäuses entstehen muss (Kronland-Martinet & Voinier, 2008).

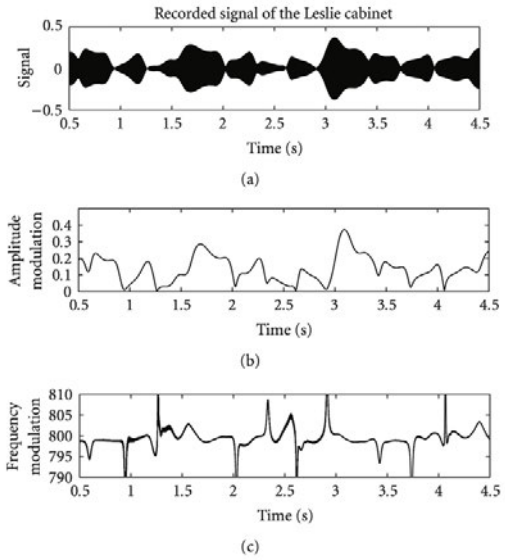

Abbildung 24: Analyse eines sinusförmigen 800 Hz Testsignals durch einen Leslie-Lautpsprecher
(a) zeigt das aufgezeichnete Signal im Zeitbereich, (b) zeigt die Amplitudenmodulation und (c) die Frequenzmodulation
Quelle: (Kronland-Martinet & Voinier, 2008)

Bei der Modellierung des Horn-Rotors stellen Kronland-Martinet und Voinier fest, dass bei einer Simulation, bei der ausschließlich die Direktschallquelle ohne Reflektionen berücksichtigt wird, sowohl die Frequenz- als auch die Amplitudenmodulation als einfache, sinusförmige Modulationen auftreten. Werden die Spiegelschallquellen jedoch hinzugeschaltet, entsteht ein Modulationsmuster, welches eine ähnliche Charakteristik aufweist wie die Messung am Original (siehe Abbildung 25).

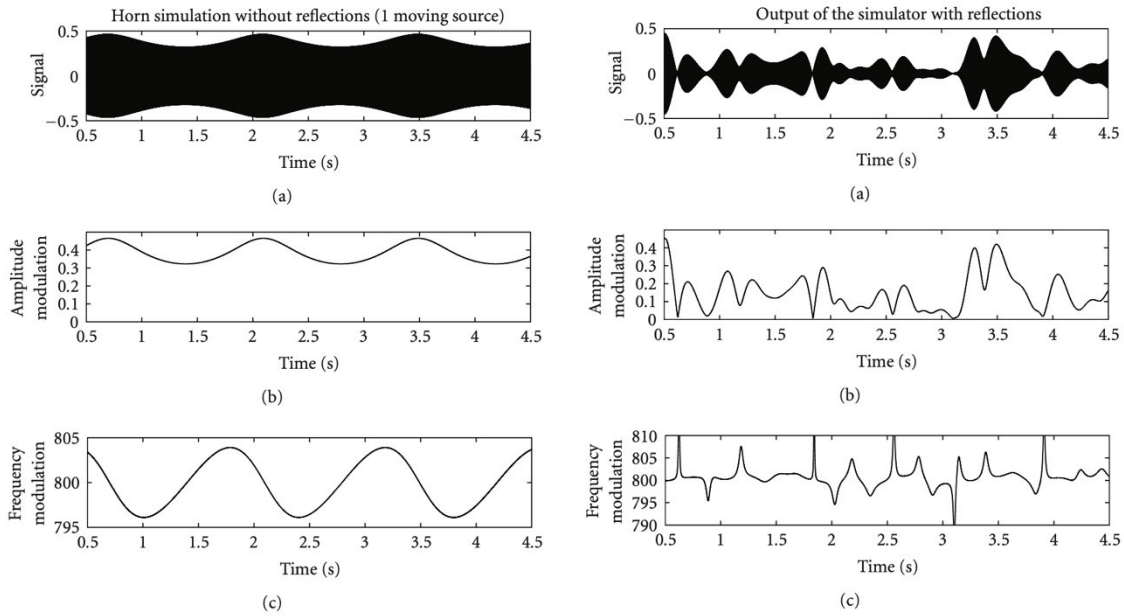

Abbildung 25: Analyse eines sinusförmigen 800 Hz Testsignals, erzeugt von einem Horn-Simulator
(a) zeigt das aufgezeichnete Signal im Zeitbereich, (b) zeigt die Amplitudenmodulation und (c) die Frequenzmodulation
Quelle: (Kronland-Martinet & Voinier, 2008)

Da im Rahmen dieser Arbeit kein elektromechanischer Rotationslautsprecher für Messungen zur Verfügung stand, wird die Messung Kronland-Martinets und Voiniers als Grundlage für die Überprüfung des hier beschriebenen Modells auf seine Funktionsweise betrachtet. Das Ausgangssignal des Matlab Modells wird der gleichen Analyse unterzogen. Dafür wird das analytische Signal mithilfe des Befehls `hilbert(y)` gebildet und dessen Betrag und die Ableitung des Phasenwinkels ermittelt.

Codeblock 16: Signalanalyse mittels Hilbert-Transformation in Matlab
Quelle: (Mathworks, 2022)

```
% hilbert transform for signal analysis
zfreq = hilbert(y(:,1));
instfrq = Fs/(2*pi)*diff(unwrap(angle(zfreq)));

env = abs(zfreq);
```

Der Versuchsaufbau aus dem Paper von Kronland-Martinets und Voinier wird wiederholt und das Ergebnis analysiert. Die Charakteristik des Ausgangssignals (siehe Abbildung 26 und Abbildung 27) entspricht dem der referenzierten Autoren. Auch hier entwickelt das Signal der Horn-Simulation mit eingeschalteten Spiegelschallquellen ein kompliziertes Modulationsmuster mit nicht-periodischem Verhalten.

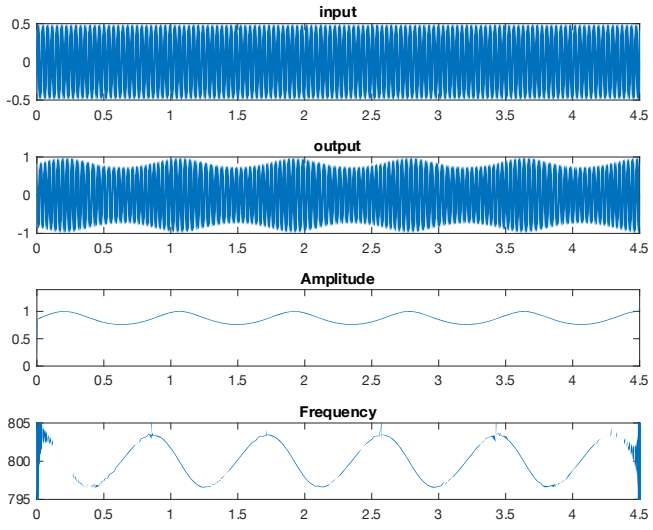

Abbildung 26: Eigens durchgeführte Analyse eines sinusförmigen 800 Hz Testsignals, erzeugt von einem Horn-Simulator in Matlab – ausschließlich Direktschallquelle aktiv
Quelle: Eigene Darstellung

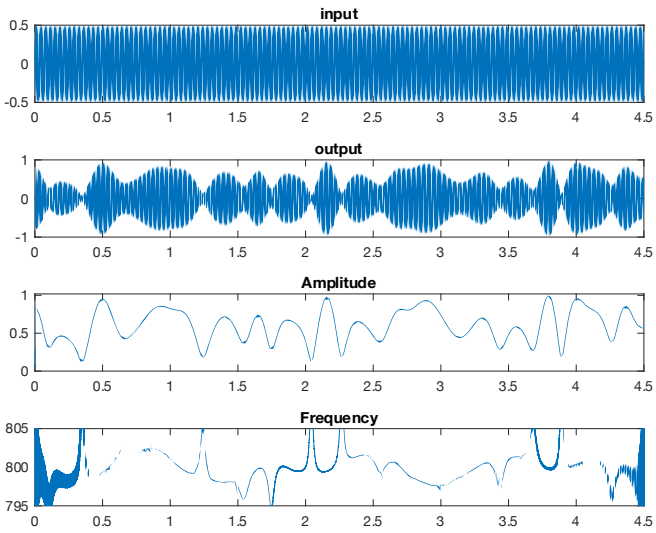

Abbildung 27: Eigens durchgeführte Analyse eines sinusförmigen 800 Hz Testsignals, erzeugt von einem Horn-Simulator in Matlab – Direktschallquelle und Spiegelschallquellen aktiv
Quelle: Eigene Darstellung

Außerdem wird diese Messung am Audioboard durchgeführt. Dafür wird ein sinusförmiges Testsignal an den Eingang des Systems gegeben und das Ausgangssignal mit einem Audiorecorder aufgezeichnet. Das aufgezeichnete Signal wird ebenfalls mit Matlab auf die gleiche Art und Weise wie die Messung zuvor analysiert. Das Ergebnis ist in Abbildung 28 zu sehen. Um den Unterschied zwischen reiner Direktschallquelle und eingeschalteten Spiegelschallquellen hörbar zu machen, werden beide Varianten am Signalausgang des Audioboards aufgezeichnet. Unter dem Dateinamen „Hörbeispiel_4“ ist die Aufnahme zu finden, auf der lediglich die Direktschallquelle des Horns und der Bass-Rotor aktiv sind. In „Hörbeispiel_5“ sind bei der Aufzeichnung auch die Spiegelschallquellen der Hornsimulation aktiv.

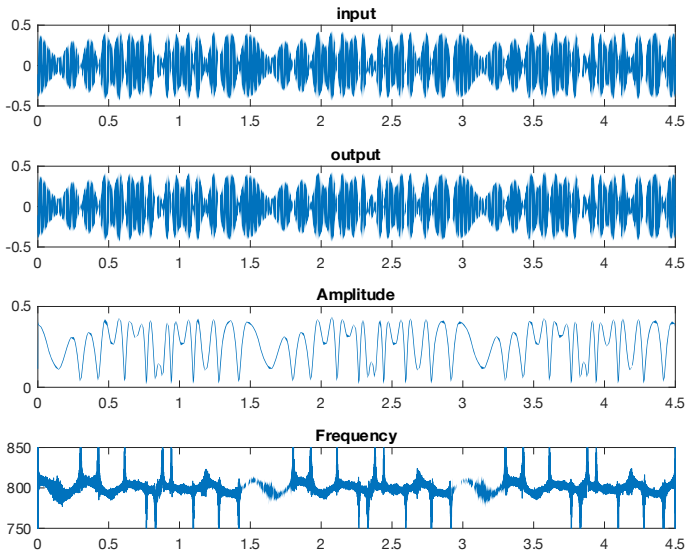

Abbildung 28: Analyse eines sinusförmigen 800 Hz Testsignals am Ausgang des Audioboards

Quelle: Eigene Darstellung

Auch hier zeigt sich das charakteristische Modulationsverhalten, welches bereits bei der Messung am Original beobachtet werden konnte. Das unterstützt Donald Leslies eigene Aussage aus Kapitel 2.3, nämlich dass die Überlagerung des reflektierten Schalls mit dem Direktschall zu einem komplizierten Modulationsverhalten führt. Bewegt sich die Direktschallquelle auf die Hörposition zu, so entfernt sich die Spiegelschallquelle, die an der gegenüberliegenden Gehäusewand reflektiert wird, von der Hörposition. Durch den Dopplereffekt wird die Frequenz der Direktschallquelle angehoben und die Frequenz der Spiegelschallquelle vermindert. Zusätzlich entsteht durch die Entfernungsdifferenz zur Hörposition eine Phasenverschiebung. So entstehen aus einem reellen Eingangssignal ein komplexes Ausgangssignal. Das führt zu zusätzlichen Modulationen von Frequenz und Amplitude.

Nachdem festgestellt ist, dass sich das Modell ähnlich verhält wie das analoge Original, können weitere Messungen vorgenommen werden. Es wurde bereits erwähnt, dass es sich bei einem Leslie-Effekt nicht um ein zeitinvariantes System handelt. Durch die Rotation der Schallabstrahler ist die Bedingung

$$x(t-\tau) \Rightarrow y(t-\tau) \tag{17}$$

nicht erfüllt (Gründigen, 2014, p. 66). Durch die Definition des Rotationsindex auf einen festen Wert lässt sich die Simulation gewissermaßen anhalten und als LTI-System untersuchen.

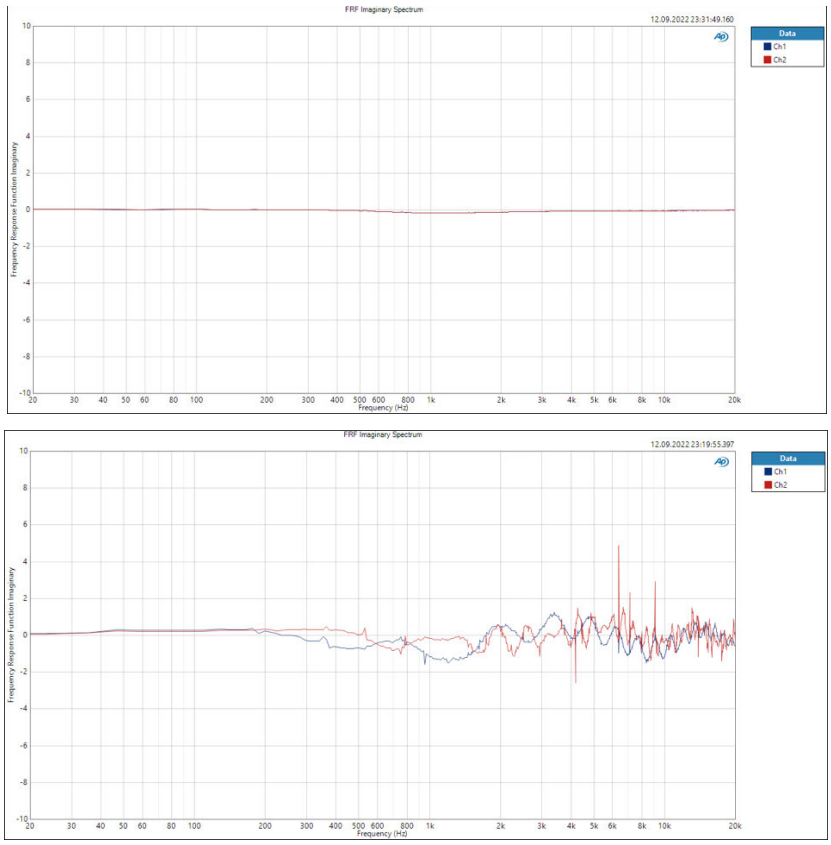

Abbildung 29: Imaginärteil der Übertragungsfunktion mit aktiven Spiegelschallquellen (unten) und inaktiven Spiegelschallquellen (oben)
Quelle: Eigene Darstellung

Die Imaginärteile der komplexen Übertragungsfunktion für aktive und inaktive Spiegelschallquellen werden miteinander verglichen. Dies bestätigt die vorangegangene Aussage, dass es bei Simulation des reflektierten Schalls innerhalb des Gehäuses zur Entstehung eines komplexen Signals kommt. Wird lediglich die Direktschallquelle im Freifeld simuliert (Abbildung 29 oben), liegt hingegen ein reelles LTI-System vor. Das bedeutet, dass „ein sinusförmiges Eingangssignal zu einem sinusförmigen Ausgangssignal derselben Frequenz [verarbeitet wird]“ (Gründigen, 2014, p. 131).

Als nächstes werden die Amplitudenfrequenzgänge des Systems an zwei Hornpositionen bestimmt. Eine Messung wird in der virtuellen Startposition des Horns durchgeführt, eine zweite nach einer halben Rotation des Horns. Die Ergebnisse sind in Abbildung 30 zu sehen.

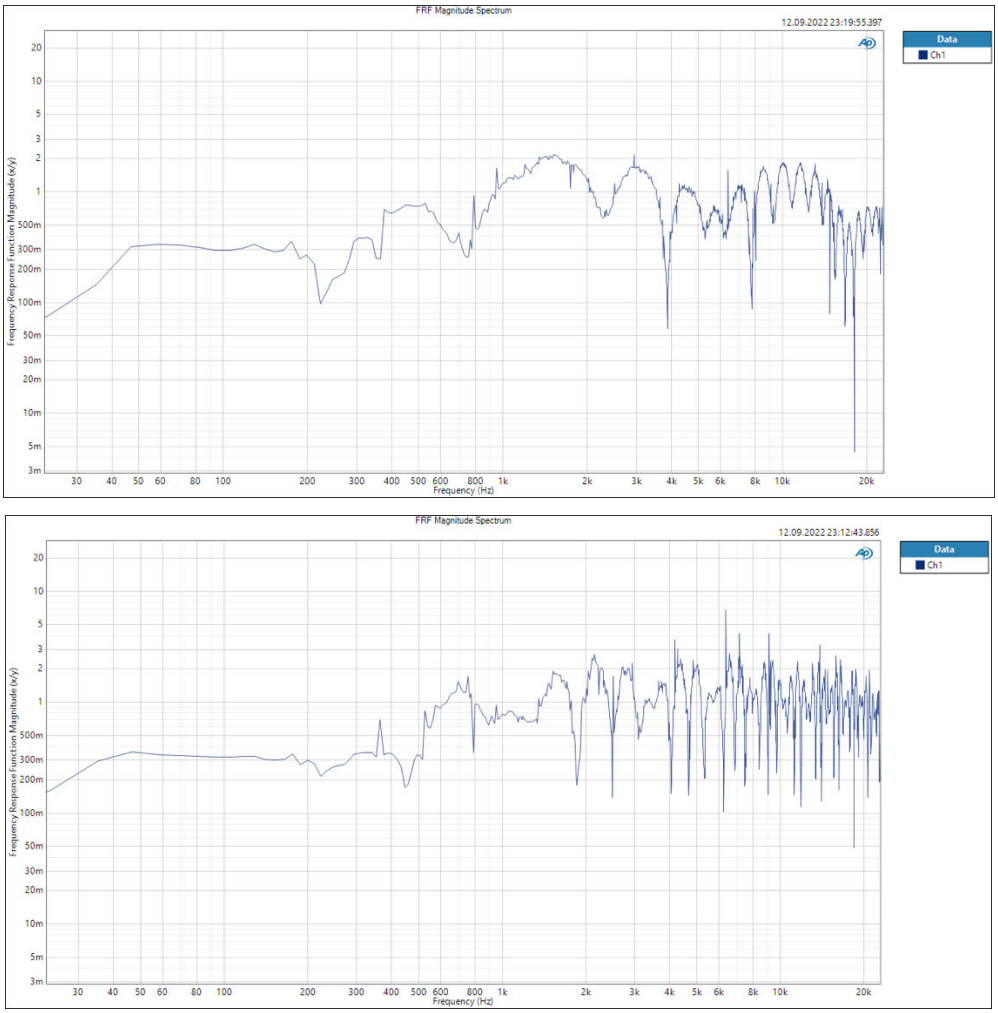

Abbildung 30: Amplitudenfrequenzgang bei Startposition des Horns (oben) und nach halber Umdrehung (unten)
Quelle: Eigene Darstellung

Es ist an beiden Positionen ein ausgeprägter Kammfiltereffekt zu sehen. Da sich Frequenzminima und -maxima in Anzahl und Position auf der Frequenzachse zwischen den beiden Messungen unterscheiden, kann davon ausgegangen werden, dass diese sich ebenfalls über die Zeit ändern. Die Übertragungsfunktionen weisen dadurch eine Ähnlichkeit zu Chorus-, Flanger- und Phaser-Effekten auf, welche durch Überlagerung des Originalsignals mit dem verzögerten Signal erstellt werden und ebenfalls eine Kamm-

filterübertragungsfunktion aufweisen (Dickreiter, et al., 2014, p. 409). Die Erzeugung des Schalls innerhalb des relativ kleinen und geschlossenen Gehäuses des Leslie-Lautsprechers begünstigt die Kammfiltereffekte und ist wahrscheinlich ein beabsichtigter Grund für die Klangeigenschaften des Effektgerätes. Im Rahmen der Recherche der Arbeit wurde jedoch keine explizite Erwähnung über dessen Einfluss auf den Gesamtklang gefunden.

4.2 Hörversuch

Ein Teil der Planung dieser Arbeit war die Durchführung eines Hörversuchs mit Freiwilligen, um den Klang des konzipierten Systems unabhängig bewerten zu können. Ein idealer Versuchsaufbau würde in einem Blindtest bestehen, bei dem den Proband*innen ein Signal abwechselnd von einem elektromechanischen Rotationslautsprecher und dem hier vorgestellten Modell vorgespielt wird, ohne dass sie den Klang visuell zuordnen können, zum Beispiel durch einen Sichtschutz oder verbundene Augen. Anschließend würde man die Probanden die Klänge nach unterschiedlichen Kriterien bewerten und miteinander vergleichen lassen.

Es stellte sich jedoch früh heraus, dass die Beschaffung eines Rotationslautsprechers aufgrund ihrer Seltenheit kein einfaches Unterfangen ist. Gut erhaltene Exemplare von originalen Leslie-Lautsprechern stehen meist in professionellen Tonstudios und werden nicht verliehen. Dies hängt damit zusammen, dass elektromechanische Orgeln, die früher der Hauptgrund für die Anschaffung solcher Lautsprecher waren, zum großen Teil von digitalen Keyboards abgelöst wurden, die bereits eine große Auswahl an Klängen und Effekten integriert haben. Diese Geräte bieten heutzutage vor allem im Amateurbereich eine einfachere und günstigere Alternative. Daher werden analoge Geräte nun als Antiquitäten gehandelt und finden ihren Einsatz vor allem bei Liebhaber*innen und aufwändigen Musikproduktionen.

Der Effekt selbst ist jedoch nicht in Vergessenheit geraten und es gibt zahlreiche Simulationen für verschiedene Instrumente (Thomann, 2019). Diese Simulationen beschränken sich allerdings auf die Verarbeitung des Eingangssignals ohne Einbeziehung der Beschallungsanlage, wie es in dem System der Fall ist, welches im Rahmen dieser Arbeit konzipiert wurde. Deshalb ist ein Vergleich zwischen einer auf dem Markt erhältlichen Simulation und diesem System nur bedingt sinnvoll, zumal festgestellt werden müsste, wie nah die diesbezügliche Simulation dem Klang des Originals ist.

Eine weitere Möglichkeit bestünde darin, eine Aufnahme zu finden, auf der sowohl das direkte Signal eines Instrumentes – im besten Fall einer Orgel – aufgezeichnet als auch die Aufzeichnung eines Leslie-Lautsprechers desselben Signals vorhanden ist. Mithilfe des Direktsignals könnte durch die in dieser Arbeit vorgestellte Simulation eine Aufnahme angefertigt werden, die direkt mit dem aufgezeichneten Leslie-Signal verglichen werden könnte. Leider konnte im Rahmen dieser Arbeit trotz Sichtung zahlreicher Mehrspuraufnahmen keine Aufzeichnung gefunden werden, die diese Kriterien erfüllt.

Um den Klang des Systems dennoch unabhängig einschätzen zu können, wurde ein im Umfang reduzierter Hörversuch mit einer kleinen Versuchsgruppe von drei Personen mit medientechnischem Hintergrund durchgeführt.

Der Hörversuch wurde mit den Teilnehmenden einzeln umgesetzt. Zunächst wurde ein unbearbeitetes Signal einer Orgel über die Beschallungsanlange des Simulationssystems abgespielt. Dieses Signal stammt aus dem bereitgestellten Archiv zum Paper von Kronland-Martinet und Voinier (2008). Anschließend wurde dasselbe Signal mit eingeschalteter digitaler Simulation abgespielt und die Teilnehmenden wurden nach wahrgenommenen Unterschieden zwischen den beiden Hörbeispielen gefragt. Danach wurde das verarbeitete Hörbeispiel aus dem Paper von Kronland-Martinet und Voinier (2008) vorgespielt und es wurde erneut nach Eindrücken und Unterschieden gefragt. Zum Schluss wurden den Teilnehmenden mehrere Aufnahmen authentischer Leslie-Aufnahmen vorgespielt, um einen Kontext herzustellen und den Klang in Bezug darauf einzuordnen.

4.3 Klangliche Beurteilung

Die wichtigsten charakteristischen Merkmale eines Rotary-Effekts sind Frequenz- und Amplitudenmodulation (Vail, 2002, p. 134). Diese Eigenschaften sind auch bei dem hier konzipierten System vorhanden und klar erkennbar und wurden von allen Teilnehmenden des Hörversuchs wahrgenommen. Wie sich aus den Messungen ergeben hat, dürfen für eine authentische Rotationslautsprechersimulation diese Modulationen nicht isoliert voneinander betrachtet werden. Sie müssen in einen akustischen Kontext miteinander gesetzt werden, um den raumakustischen Eigenschaften des physischen Effektgerätes gerecht zu werden. Dass die Simulation von Spiegelschallquellen dafür ein geeigneter Ansatz ist, zeigt sich auch bei den Hörtests. Wird die Simulation ausschließlich mit Direktschallquelle betrieben, ist der musikalische Klang des Ausgangssignals dünn und weist nur bedingte Räumlichkeit auf. Durch die periodischen Modulationsmuster wirkt der Klang vorhersehbar und monoton. Mit eingeschalteten Spiegelschallquellen entwickelt sich ein weitaus komplexeres Klangbild, das eine breitere räumliche Wahrnehmung bietet und einem Instrument eine Art von Mehrstimmigkeit verleiht, was auch von den Teilnehmenden des Hörversuchs bestätigt wurde.

Die räumlich wahrgenommene Rotation kommt besonders bei langsamer Rotationsgeschwindigkeit zur Geltung und wenn zwischen den beiden Geschwindigkeitsmodi *Tremolo* und *Chorale* umgeschaltet wird. Dann entsteht der Eindruck, der Klang würde zwischen den beiden Breitbandlautsprechern in einer kreisenden Bewegung hin- und herwandern. Laut der Aussage eines Teilnehmers des Hörversuchs war diese Bewegung breiter als der Abstand zwischen den Lautsprechern. Dieser Effekt kann sogar verstärkt werden, wenn die Lautsprecher um etwa 10°-15° nach außen gedreht werden.

Die Simulation des Bass-Rotors wurde sowohl mit als auch ohne den Einsatz von Frequenzmodulation getestet. Die Frequenzmodulation erweist sich hierbei als deutlich weniger ausgeprägt als bei der Horn-

Simulation. Die erzeugte Modulation ist dennoch wahrnehmbar und bewirkt eine subtile Verbesserung des Gesamtklanges und wird deswegen auch in der finalen Simulation verwendet.

Ein großer Nachteil dieses Systems ist die Richtungsabhängigkeit. Die Abbildung des Effekts funktioniert vor der Beschallungsanlage sehr gut, nimmt jedoch stark ab, sobald sich der*die Zuhörer*in aus der Achse bewegt und verschwindet fast vollständig, sobald er*sie sich hinter der Lautsprecheranordnung befindet. Das stellt einen großen Unterschied zum analogen Vorbild dar, dessen Abstrahlverhalten durch die stattfindende Rotation auf horizontaler Ebene in alle Richtungen gleich ist.

4.4 Fazit

Die vorgestellte Methode ist dafür geeignet, den Klang eines Rotationslautsprechers nachzubilden. Durch die angestellten Messungen hat sich herausgestellt, dass die Berücksichtigung der Schallreflektionen innerhalb des Gehäuses für eine authentische Simulation unentbehrlich ist. Durch die resultierende Signalüberlagerung entstehen zusätzliche Modulationen in Form von zeitabhängigen Kammfiltereffekten, die zu einem vielschichtigen Klangbild beitragen. Diese Effekte sind in Beschreibungen, die in der Literatur zu finden sind, oft unterrepräsentiert. Eine präzisere Simulation der Klangreflektionen, zum Beispiel durch Spiegelschallquellen höherer Ordnung oder eines Raytracing Algorithmus, könnte zu einer Verbesserung des Modells beisteuern.

Eine interessante Ergänzung für die Simulation des Bass-Rotors wäre die Untersuchung der Raummoden innerhalb des Gehäuses. Eine kurze Untersuchung mit geschätzten Werten hat gezeigt, dass sich bei räumlichen Dimensionen dieser Größenordnung stehende Wellen bei mehreren Frequenzen bis 800 Hz ausbilden. Für eine akkurate Simulation wäre allerdings der Zugriff zu einem Originalgerät von Vorteil, um die notwendigen Größen präzise zu bestimmen und diese messtechnisch zu überprüfen.

Die Berechnung der Delay-Zeiten über Matlab gestaltet sich in der Praxis als ineffizient. In zukünftigen Implementierungen sollte sie direkt auf dem Audioboard stattfinden. Dadurch könnten Parameter wie virtuelle Hörpositionen oder räumliche Dimensionen während des Betriebes eingestellt und ein vielfältigeres Klangbild ermöglicht werden.

Die gewählte Lautsprecheranordnung ist dafür geeignet, den Effekt unidirektional wiederzugeben. Für eine omnidirektionale Abstrahlcharakteristik wäre eine Anordnung mit vier Lautsprechern und entsprechend vier virtuellen Hörpositionen denkbar. Das verwendete Audioboard unterstützt voreingestellt zwar nur zwei Ausgangskanäle, es gibt jedoch zusätzliche serielle Audioschnittstellen, auf die mithilfe weiterer Hardware zugegriffen werden kann (Electrosmith, 2021). Das Problem des begrenzten Speicherplatzes der vorausberechneten Delay-Zeiten würde jedoch weiter bestehen bleiben. Eine Möglichkeit, dieses Problem zu beheben, wäre eine Speicherung auf dem QSPI-Speicher des Audioboards mit einer Größe von 64 MB. Der Zugriff auf diesen Speicher erfolgt jedoch zu langsam für Echtzeitanwendungen, sodass eine Zwischenspeicherung der Daten nötig wäre (Electrosmith, 2021). Diese Zwischenspeicherung könnte zum Beispiel mit einem Ringpuffer umgesetzt werden.

Um das Modell in der Praxis einsetzen zu können, sollte außerdem eine passive Frequenzweiche eingesetzt und zusammen mit dem Audioboard in einem Gehäuse verbaut werden. Der verwendete Taster ließe sich mit einem Fußschalter ersetzen und würde eine einfache Steuerung der Rotationsgeschwindigkeit während des Betriebes ermöglichen.

5 Zusammenfassung

In dieser Arbeit werden historische Hintergründe und Relevanz des Leslie-Rotationslautsprechers erläutert und seine Besonderheiten in Bezug auf Klang und Aufbau untersucht. Durch seine hohe Popularität in Verbindung mit der Hammond-Orgel ist der Leslie auf vielen wegweisenden Veröffentlichungen der Pop-Musik zu hören. Auch heute ist der Klang aufgrund seiner einzigartigen Zusammensetzung unterschiedlicher akustischer Effekte relevant.

Es werden existierende Modelle aus wissenschaftlichen Veröffentlichungen zur Simulation eines Rotationslautsprechers analysiert und miteinander verglichen. Es wird ein Ansatz für die Simulation gewählt, welcher das Modell der Schallausbreitung als Schallstrahl nutzt, um die akustischen Eigenschaften des Leslie-Lautsprechers zu simulieren. Anhand des Abstandes zwischen Schallquelle und Hörposition werden die Verzögerung und die Verstärkung des Direktschalls und der Schallreflektion an den Gehäusewänden berechnet und auf das Eingangssignal angewendet. Für den Bass-Rotor wird aufgrund der größeren Wellenlänge nur die Direktschallquelle simuliert.

Mithilfe eines Matlab Modells wird die Simulation ohne Echtzeitanforderung überprüft und erste Erkenntnisse gesammelt. Die Implementierung findet in der Programmiersprache C++ statt und wird auf einem Audio Entwicklungsboard ausgeführt. Dafür werden Werte, die vom Eingangssignal unabhängig sind, in Matlab vorberechnet und auf dem Entwicklungsboard als Listen gespeichert, um eine bessere Recheneffizienz zu gewährleisten und zu garantieren, dass die Echtzeitanforderung erfüllt ist. Über die Implementierung eines Tasters kann während des Betriebes zwischen zwei Rotationsgeschwindigkeiten umgeschaltet werden.

Für die Klangwiedergabe wird ein Beschallungssystem zusammengestellt, welches die Eigenschaften des Originals nachbilden kann. Dafür werden aktive Breitbandlautsprecher für die Hornsimulation und ein Bassgitarrenverstärker mit 15“ Lautsprecher für die Bass-Rotorsimulation verwendet.

Durch Messungen wird die korrekte Funktion der Frequenzweiche überprüft und die Gesamtsimulation mit Messungen aus der Literatur verglichen. Durch die Analyse eines sinusförmigen Testsignals wird festgestellt, dass die Simulation der Schallreflektion innerhalb des Gehäuses einen wesentlichen Beitrag für die Bildung des Klangbildes leistet. Eine nähere Untersuchung der Simulation bei stillstehenden Rotoren zeigt einen zeitabhängigen Kammfiltereffekt im Spektrum der Hornsimulation. Dieser Phasing- oder Flanger-Effekt entsteht durch Schallreflektionen im Gehäuse und wird bei der Beschreibung des Originaleffekts oft unterrepräsentiert.

Abschließend werden mögliche Verbesserungen für zukünftige Implementierungen aufgezeigt. Durch die Erweiterung der Hornlautsprecher auf eine Gesamtzahl von vier kann eine omnidirektionale Abstrahlcharakteristik erreicht werden. Und durch eine alternative Möglichkeit zur Speicherung von Delay-Zeiten könnten mehr Schallreflektionen simuliert werden, um ein authentischeres Klangbild zu erzeugen.

Literaturverzeichnis

Audioprecision, 2019. *ap.com.* [Online]
Available at: https://www.ap.com/download/apx500-user-manual-2/?wpdmdl=5866&ind=MTU3Mzg1NDA1OXdwZG1fQVB4NTAwX1VzZXJzX01hbnVhbC5wZGY
[Accessed 06 09 2022].

DeWolff, 2021. *Wolffpack.* [Sound Recording] (Mascot Records).

Dickreiter, M., Dittel, V., Hoeg, W. & Wöhr, M., 2014. *Handbuch der Tonstudiotechnik.* 8., überarbeitete und erweiterte Auflage ed. Berlin/Boston: De Gruyter.

Electrosmith, 2021. *Daisy Seed Datasheet.* [Online]
Available at: https://www.electro-smith.com/daisy/daisy
[Accessed 09 09 2022].

Electrosmith, 2021. *Github.* [Online]
Available at: https://electro-smith.github.io/libDaisy/md_doc_md__a7__getting__started__daisy__bootloader.html
[Accessed 28 07 2022].

Electrosmith, 2021. *github.com.* [Online]
Available at: https://github.com/electro-smith/DaisySP/blob/master/Source/Utility/dsp.h
[Accessed 15 09 2022].

Electrosmith, 2021. *github.com.* [Online]
Available at: https://electro-smith.github.io/libDaisy/md_doc_md__a3__getting__started__audio.html
[Accessed 25 08 2022].

Electrosmith, 2021. *github.com.* [Online]
Available at: https://github.com/electro-smith/Hardware/blob/master/reference/daisy_patch/ES_Daisy_Patch_Rev4.pdf
[Accessed 09 09 2022].

Electrosmith, 2021. *github.com.* [Online]
Available at: https://electro-smith.github.io/libDaisy/md_doc_md__a7__getting__started__daisy__bootloader.html
[Accessed 08 09 2022].

Faragher, S., 2011. *The Hammond organ: an intoduction to the instrument and the players who made it famous.* Milwaukee: Hal Leonard Books.

Friesecke, A., 2014. *Die Audio-Enzyklopädie.* 2. Auflage ed. Berlin/Boston: De Gruyter.

Gründigen, D. C. V., 2014. *Digitale Signalverarbeitung.* München: Carl Hanser Verlag.

Hammond Suzuki Europe B.V., 2022. *Hammond Europe.* [Online]
Available at: http://www.hammond.eu/Info/Story
[Accessed 09 09 2022].

Harrison, G., 1968. *While My Guitar Gently Weeps.* [Sound Recording] (Apple Records).

Henricksen, C. A., 1981. Unearthing the mysteries of the Leslie cabinet. *Recording Engineer/Producer magazine.*

Herrera, J., Hanson, C. & Abel, J. S., 2009. *Discrete Time Emulation of the Leslie Speaker.* New York, Audio Engineering Society.

Kronland-Martinet, R. & Voinier, T., 2008. Real-Time Perceptual Simulation of Moving Sound Sources. *EURASIP Journal on Audio Speech and Music Processing,* Volume 2008, p. 10.

Lizzo, 2021. *2 Be Loved (Am I Ready).* [Sound Recording] (Nice Life Recording Company).

Mathworks, 2022. *mathworkds.com.* [Online]
Available at: https://de.mathworks.com/help/signal/ug/hilbert-transform-and-instantaneous-frequency.html
[Accessed 05 09 2022].

Mathworks, 2022. *mathworks.com.* [Online]
Available at: https://de.mathworks.com/help/dsp/ref/filterbuilder.html?s_tid=doc_ta
[Accessed 12 07 2022].

Mathworks, 2022. *mathworks.com.* [Online]
Available at: https://de.mathworks.com/solutions/signal-processing.html
[Accessed 29 08 2022].

Neumann, M., 2020. *C Programmieren für Einsteiger.* Berlin: BMU Verlag.

Pekonen, J., Pihlajamäki, T. & Välimäki, V., 2011. *Computationally efficient Hammond organ synthesis.* Paris, Digital Audio Effects.

Rathjens, K., 2019. *amazona.de.* [Online]
Available at: https://www.amazona.de/velvet-box-die-leslie-model-122-145-147-und-251/
[Accessed 10 09 2022].

Redmon, N., 2012. *earlevel.com.* [Online]
Available at: https://www.earlevel.com/main/2012/11/26/biquad-c-source-code/
[Accessed 04 09 2022].

schweizer-fn, n.d. *schweizer-fn.de.* [Online]
Available at: https://www.schweizer-fn.de/stoff/akustik/absorptionsfaktoren.php
[Accessed 25 08 2022].

Smith, J., Serafin, S., Abel, J. & Berners, D., 2002. *Doppler simulation and the Leslie.* Hamburg, Digital Audio Effects.

Soundcraft, 2020. *soundcraft.com.* [Online]
Available at: https://www.soundcraft.com/zh/product_documents/si-expression-ug-de-02-05-13-pdf
[Accessed 10 09 2022].

stackoverflow, 2021. *stackoverflow.com.* [Online]
Available at: https://insights.stackoverflow.com/survey/2021#section-most-popular-technologies-integrated-development-environment
[Accessed 12 09 2022].

Thomann, 2019. *Thomann.de.* [Online]
Available at: https://www.thomann.de/blog/de/die-besten-rotary-effekte-fuer-jeden-geldbeutel/
[Accessed 11 09 2022].

Vail, M., 2002. *The Hammond Organ: Beauty in the B.* San Francisco: Backbeat Books.

Weinzierl, S., 2008. *Handbuch der Audiotechnik.* Berlin/Heidelberg: Springer-Verlag.

Yamaha, 2017. *markertek.com.* [Online]
Available at: https://www.markertek.com/Attachments/Specifications/Yamaha/HS5I-Specifications.pdf
[Accessed 13 09 2022].

Zölzer, U., 2011. *DAFX: Digital Audio Effects.* 2. Auflage ed. Vereintes Königreich: Wiley.

Anhang

Anhang 1: Matlab Skript Leslie Simulation Ausführung
Quelle: Eigene Darstellung

```
clear variables;
close all;

% Leslie Dimensions
Width = 0.74;
Depth = 0.53;
r = 0.2; %radius horn

infile = 'HammondDry.mp3';

% read in wav sample
[x, Fs] = audioread(infile);

LEN=length(x);

time = 0:1/Fs:LEN/Fs-1/Fs;

%Listener Position
Plist = [3*Width/2; -0.2];

HornRPM = 390;
BassRPM = 335;

RotfreqHorn = HornRPM/60; %Rotation Frequency
RotfreqBass = BassRPM/60;



%Calling back Horn and Bass rotor functions
[ihd,frachd,ih1,frach1,ih2,frach2,ih3,frach3,Lhd,Lh1,Lh2,Lh3,Nsamh,ghd,gh1,
gh2,gh3] = LeslieHorn(Fs,RotfreqHorn, Plist);

[ibd,fracbd,Lbd,Nsamb,gbd] = LeslieBass(Fs,RotfreqBass, Plist);

%memory allocation Delay lines
Delaylinehd=zeros(Lhd,1);
Delaylineh1=zeros(Lh1,1);
Delaylineh2=zeros(Lh2,1);
```

```
Delaylineh3=zeros(Lh3,1);


Delaylinebd=zeros(Lhd,1);


%memory allocation output signal
yhd = zeros(LEN,2);
yh1 = zeros(LEN,2);
yh2 = zeros(LEN,2);
yh3 = zeros(LEN,2);
yh = zeros(LEN,2);


ybd = zeros(LEN,2);
yb = zeros(LEN,2);
y = zeros(LEN,2);


%declaring rotation index
h=1;
b=1;


%crossover filtering of input signal
crossFilt = crossoverFilter('NumCrossovers',1,'Crosso-
verFrequencies',800,'CrossoverSlopes',12,'SampleRate', Fs);


[xlow, xhigh] =  (crossFilt(x));



%Delay line callback loop
tic
for n=1:(LEN-1)


   %---Linear Interpolation Horn-----------------------------
   Delaylinehd=[xhigh(n);Delaylinehd(1:Lhd-1)];
   yhd(n,:)=Delaylinehd(ihd(h)+1)*frachd(h)+Delaylinehd(ihd(h))*(1-
frachd(h));


   Delaylineh1=[xhigh(n);Delaylineh1(1:Lh1-1)];
   yh1(n,:)=Delaylineh1(ih1(h)+1)*frach1(h)+Delaylineh1(ih1(h))*(1-
frach1(h));


   Delaylineh2=[xhigh(n);Delaylineh2(1:Lh2-1)];
   yh2(n,:)=Delaylineh2(ih2(h)+1)*frach2(h)+Delaylineh2(ih2(h))*(1-
frach2(h));
```

```
   Delaylineh3=[xhigh(n);Delaylineh3(1:Lh3-1)];
   yh3(n,:)=Delaylineh3(ih3(h)+1)*frach3(h)+Delaylineh3(ih3(h))*(1-
frach3(h));

   %horn gain calculation
   yh(n,:)=ghd(h)*yhd(n,:)+gh1(h)*yh1(n,:)+gh2(h)*yh2(n,:)+gh3(h)*yh3(n,:);

      %---Linear Interpolation Bass-----------------------------
   Delaylinebd=[xlow(n);Delaylinebd(1:Lbd-1)];
   ybd(n,:)=Delaylinebd(ibd(b)+1)*fracbd(b)+Delaylinebd(ibd(b))*(1-
fracbd(b));

   %bass gain calculation
   yb(n,:)=gbd(b)*ybd(n,:);

   %final output signal
   y(n,:)= yh(n,:) + yb(n,:);

   %resetting rotation index after full rotation
   if h == Nsamh-1
       h = 1;
   else
       h = h+1;
   end

   if b == Nsamb-1
       b = 1;
   else
       b = b+1;
   end

end
toc

%normalise output signal
y = y/max(y);

% hilbert transform for signal analysis
zfreq = hilbert(y(:,1));
instfrq = Fs/(2*pi)*diff(unwrap(angle(zfreq)));
ifrq = instfreq(x,Fs);
```

```
env = abs(zfreq);

%plot signals
tiledlayout(5,1)

%plot input signal
nexttile
plot(time,x(:,1))
title('input')

%plot output signal
nexttile
plot(time,y(:,1))
title('output')

%plot amplitude modulation
nexttile
plot(time, env)
title('Amplitude')

%plot frequency modulation
nexttile
plot(time(2:end),instfrq)
ylim([750 850])
title('Frequency')

nexttile
plot(ifrq)
ylim([750 850])
title('Frequency')

%playback sound
sound(y,Fs);
```

Anhang 2: Matlab Funktion Hornsimulation
Quelle: Eigene Darstellung

```
function [id,fracd,i1,frac1,i2,frac2,i3,frac3, Ld, L1, L2, L3, Nsam,
gd,g1,g2,g3] = LeslieHorn(Fs,RotfreqHorn, Plist)
%LESLIE tabulated delay times for image sound sources inside Leslie Cabinet
```

```
%Speed of Sound
csound = 343;

% Leslie Dimensions
Width = 0.73;
Depth = 0.53;
r = 0.2; %radius horn

Nsam = round(Fs/RotfreqHorn); %Number of samples per rotation

%creating circle
Center = [3*Width/2 Depth/2]; %Center coordinates
angle = 0:((2*pi)/Nsam):2*pi; %divding circle into samples per rotation
cx = r*cos(angle)+Center(1);
cy = r*sin(angle)+Center(2);

P = [cx;cy]; %direct circle coordinates
P1 = [4*Width-cx;cy]; %1st image source coordinates
P2 = [cx;2*Depth-cy]; %2nd image source coordinates
P3 = [2*Width-cx;cy]; %3rd image source coordinates

%Delay time memory allocation
Td = zeros(Nsam,1);
T1 = zeros(Nsam,1);
T2 = zeros(Nsam,1);
T3 = zeros(Nsam,1);

%Dinstance Source Listener
for n=1:Nsam
dd = norm(P(:,n)-Plist); %direct distance
d1 = norm(P1(:,n)-Plist);
d2 = norm(P2(:,n)-Plist);
d3 = norm(P3(:,n)-Plist);

%gain based on distance

gd(n) = 1/dd;
g1(n) = 1/(sqrt(d1));
g2(n) = 1/(sqrt(d2));
g3(n) = 1/(sqrt(d3));
```

```
%Delay in samples
Td(n) = (dd/csound)*Fs;
T1(n) = (d1/csound)*Fs;
T2(n) = (d2/csound)*Fs;
T3(n) = (d3/csound)*Fs;
end

Ld = ceil(max(Td));
L1 = ceil(max(T1));
L2 = ceil(max(T2));
L3 = ceil(max(T3));

%memory allocation for Delayline parameters
id=zeros(Nsam,1);
i1=zeros(Nsam,1);
i2=zeros(Nsam,1);
i3=zeros(Nsam,1);

fracd=zeros(Nsam,1);
frac1=zeros(Nsam,1);
frac2=zeros(Nsam,1);
frac3=zeros(Nsam,1);

for n=1:(Nsam-1)

      id(n)=floor(Td(n));
   fracd(n)=Td(n)-id(n);

      i1(n)=floor(T1(n));
   frac1(n)=T1(n)-i1(n);

      i2(n)=floor(T2(n));
   frac2(n)=T2(n)-i2(n);

      i3(n)=floor(T3(n));
   frac3(n)=T3(n)-i3(n);
end

end
```

Anhang 3: Matlab Funktion Basssimulation
Quelle: Eigene Darstellung

```
function [id,fracd,Ld,Nsam,gd] = LeslieBass(Fs,RotfreqHorn, Plist)
%LESLIE tabulated delay times for image sound sources inside Leslie Cabinet

%Speed of Sound
csound = 343;

% Leslie Dimensions
Width = 0.73;
Depth = 0.53;
r = 0.2; %radius horn

Nsam = round(Fs/RotfreqHorn); %Number of samples per rotation

%creating circle
Center = [3*Width/2 Depth/2]; %Center coordinates
angle = 0:((2*pi)/Nsam):2*pi; %divding circle into samples per rotation
cx = r*cos(angle)+Center(1);
cy = r*sin(angle)+Center(2);

P = [cx;cy]; %direct circle coordinates

%Delay time memory allocation
Td = zeros(Nsam,1);

%Dinstance Source Listener
for n=1:Nsam
dd = norm(P(:,n)-Plist); %direct distance

%gain based on distance

gd(n) = 1/sqrt(dd);

%Delay in samples
Td(n) = (dd/csound)*Fs;
end

Ld = ceil(max(Td));

%memory allocation for Delayline parameters
```

```
id=zeros(Nsam,1);



fracd=zeros(Nsam,1);

for n=1:(Nsam-1)
      id(n)=floor(Td(n));
   fracd(n)=Td(n)-id(n);

end

end
```

Anhang 4: Matlab Code LUT-Generator
Quelle: Eigene Darstellung

```
clear variables;
close all;
Fs = 48000;
Fsdown = Fs/8;

%tremolo speeds
HornRPM = 390;
BassRPM = 335;

% chorale speeds
% HornRPM = 50;
% BassRPM = 40;

RotfreqHorn = HornRPM/60; %Rotation Frequency
RotfreqBass = BassRPM/60;

%Speed of Sound
csound = 343;

% Leslie Dimensions
Height = 0.83;
Width = 0.73;
Depth = 0.53;
r = 0.2; %radius horn

%Listener position
```

```
Plist = [5*Width/4; -0.2];

Nsam = round(Fsdown/RotfreqHorn); %Number of samples per rotation

%creating circle
Center = [3*Width/2 Depth/2]; %Center coordinates
angle = 0:((2*pi)/Nsam):2*pi; %divding circle into samples per rotation

%circle starting center right going clockwise
cxd = r*cos(angle)+Center(1);
cyd = r*(-sin(angle))+Center(2);

%circle starting center left going
cx13 = r*(-cos(angle))+Center(1);
cy13 = r*(-sin(angle))+Center(2);

%circle starting center right going counter clockwise
cx2 = r*cos(angle)+Center(1);
cy2 = r*sin(angle)+Center(2);

P = [cxd;cyd]; %direct circle coordinates
P1 = [4*Width-cx13;cy13]; %1st image source coordinates
P2 = [cx2;2*Depth-cy2]; %2nd image source coordinates
P3 = [2*Width-cx13;cy13]; %3rd image source coordinates

%Delay time memory allocation
Td = zeros(Nsam,1);
T1 = zeros(Nsam,1);
T2 = zeros(Nsam,1);
T3 = zeros(Nsam,1);

%Dinstance Source Listener
for n=1:Nsam
dd(n) = norm(P(:,n)-Plist); %direct distance
d1(n) = norm(P1(:,n)-Plist);
d2(n) = norm(P2(:,n)-Plist);
d3(n) = norm(P3(:,n)-Plist);

%gain based on distance
gd(n) = 1/dd(n);
g1(n) = 1/(sqrt(d1(n)));
```

```
g2(n) = 1/(sqrt(d2(n)));
g3(n) = 1/(sqrt(d3(n)));

%Delay in samples
Td(n) = (dd(n)/csound)*Fs;
T1(n) = (d1(n)/csound)*Fs;
T2(n) = (d2(n)/csound)*Fs;
T3(n) = (d3(n)/csound)*Fs;

%gdnew=1/(sqrt)
end

maxdistanced=max(gd);
maxdistance1=max(g1);
maxdistance2=max(g2);
maxdistance3=max(g3);

maxdistance = [maxdistance3 maxdistance2 maxdistance1 maxdistanced];

maxdistancetotal=max(maxdistance);

gd=transpose(gd)/maxdistancetotal;
g1=transpose(g1)/maxdistancetotal;
g2=transpose(g2)/maxdistancetotal;
g3=transpose(g3)/maxdistancetotal;

gd=transpose(gd);
g1=transpose(g1);
g2=transpose(g2);
g3=transpose(g3);
```

Anhang 5: C++ Code Leslie Simulation auf Audioboard
Quelle: Eigene Darstellung

```
#include "daisysp.h"
#include "daisy_seed.h"
#include "delaylut_bass.h"
#include "delaylut_hornL.h"
#include "delaylut_hornR.h"
```

```
// Interleaved audio definitions
#define LEFT (i)
#define RIGHT (i + 1)

// Set max delay time to 1000 samples
#define MAX_DELAY static_cast<size_t>(1000)

using namespace daisysp;
using namespace daisy;
static DaisySeed hw;
Switch button1;

// Filter function declaration
double filterH (double a0, double a1, double a2, double b1, double b2,
float input);
double filterL (double a0, double a1, double a2, double b1, double b2,
float input);

// Filter coefficients taken from MatLab filterBuilder
double a0H =  1;
double a1H = -2;
double a2H = 1;
double b0H = 1;
double b1H = -1.852146485395935515683163430367130786 18 ;
double b2H =  0.862348626030080889215412298653973266482;
double z1H = 0, z2H = 0;

double a0L = 0.002550535158536371138637832700624130666;
double a1L = 0.005101070317072742277275665401248261333;
double a2L = 0.002550535158536371138637832700624130666;
double b0L = 1;
double b1L = -1.852146485395935515683163430367130786 18;
double b2L = 0.862348626030080889215412298653973266482;
double z1L = 0, z2L = 0;

// global variable declaration
float currentDelayd_horn_L, currentDelay1_horn_L, currentDelay2_horn_L,
currentDelay3_horn_L, sig_out_horn_L,in1, feedback, deld_out_horn_L,
del1_out_horn_L, del2_out_horn_L, del3_out_horn_L;
float currentDelayd_horn_R, currentDelay1_horn_R, currentDelay2_horn_R,
currentDelay3_horn_R, sig_out_horn_R, deld_out_horn_R, del1_out_horn_R,
del2_out_horn_R, del3_out_horn_R;
```

```
float currentDelayd_bass, currentDelay1_bass, currentDelay2_bass, cur-
rentDelay3_bass, sig_out_bass, deld_out_bass, del1_out_bass, del2_out_bass,
del3_out_bass;
float sig_out, sig_out_L, sig_out_R, gain_direct_L, gain_1_L, gain_2_L,
gain_3_L, gain_direct_R, gain_1_R, gain_2_R, gain_3_R, gain_bass;
float csound = 343;


//index for position of rotating speaker and its image sources
int rotation = 0, counter = 0, rotation_bass = 0, rotation_count=0;
float skip = 1, skip_target =1;


// Declare a DelayLine of MAX_DELAY number of floats.
static DelayLine<float, MAX_DELAY> deld_L, del1_L, del2_L, del3_L,deld_R,
del1_R, del2_R, del3_R, deldbass;


static void AudioCallback(AudioHandle::InterleavingInputBuffer  in,
                          AudioHandle::InterleavingOutputBuffer out,
                          size_t                                size)
{
    for(size_t i = 0; i < size; i += 2)
    {

        //input variable
        in1 = in[i+1];

        float outputLow = filterL(a0L, a1L, a2L, b1L, b2L, in1);
        float outputHigh = -filterH(a0H, a1H, a2H, b1H, b2H, in1);

        //set current delaytime based on position index
        currentDelayd_horn_L = delaytime_direct_horn_L[rotation];
        currentDelay1_horn_L = delaytime_1_horn_L[rotation];
        currentDelay2_horn_L = delaytime_2_horn_L[rotation];
        currentDelay3_horn_L = delaytime_3_horn_L[rotation];

        currentDelayd_horn_R = delaytime_direct_horn_R[rotation];
        currentDelay1_horn_R = delaytime_1_horn_R[rotation];
        currentDelay2_horn_R = delaytime_2_horn_R[rotation];
        currentDelay3_horn_R = delaytime_3_horn_R[rotation];

        currentDelayd_bass = delaytime_direct_bass[rotation_bass];

        deld_L.SetDelay(currentDelayd_horn_L);
```

```
        del1_L.SetDelay(currentDelay1_horn_L);
        del2_L.SetDelay(currentDelay2_horn_L);
        del3_L.SetDelay(currentDelay3_horn_L);

        deld_R.SetDelay(currentDelayd_horn_R);
        del1_R.SetDelay(currentDelay1_horn_R);
        del2_R.SetDelay(currentDelay2_horn_R);
        del3_R.SetDelay(currentDelay3_horn_R);

        deldbass.SetDelay(currentDelayd_bass);

        //  position of speakers rotation and speed control
        counter ++; //helper variable to get rotation speed from sample
progression

        if (counter > skip) {        //if helper variable is larger than
speed variable advance on delaytime by one sample
            rotation += 2;
            rotation_bass += 2;
            counter = 0;
        }

        fonepole(skip, skip_target, 0.000020833f); //low pass filter to
simulate rotation dynamics

        if (rotation >= 7200) {                     // reset LUT counter
variable after full rotation
            rotation = 0;
            rotation_count ++;
        }

        if (rotation_bass >= 9001) {
            rotation_bass = 0;
        }

        //button push
        button1.Debounce();

        if(button1.RisingEdge()) {
            skip_target = 6;
        }
```

```
if(button1.FallingEdge()) {
    skip_target = 1;
}

// Write to the delay
deld_L.Write(outputHigh);
del1_L.Write(outputHigh);
del2_L.Write(outputHigh);
del3_L.Write(outputHigh);

deld_R.Write(outputHigh);
del1_R.Write(outputHigh);
del2_R.Write(outputHigh);
del3_R.Write(outputHigh);

// Write to the delay
deldbass.Write(outputLow);



//calculate Gain
gain_direct_L = (48000/(currentDelayd_horn_L*csound))/5.8812f;
gain_1_L = 1/((sqrt((currentDelay1_horn_L*csound)/48000))*5.8812);
gain_2_L = 1/((sqrt((currentDelay2_horn_L*csound)/48000))*5.8812);
gain_3_L = 1/((sqrt((currentDelay3_horn_L*csound)/48000))*5.8812);

gain_direct_R = (48000/(currentDelayd_horn_R*csound))/3.6347;
gain_1_R = 1/((sqrt((currentDelay1_horn_L*csound)/48000))*3.6347);
gain_2_R = 1/((sqrt((currentDelay2_horn_L*csound)/48000))*3.6347);
gain_3_R = 1/((sqrt((currentDelay3_horn_L*csound)/48000))*3.6347);

gain_bass = (48000/(currentDelayd_bass*csound))/6.0606;

// Read from delay line
deld_out_horn_L = deld_L.Read()*gain_direct_L;
del1_out_horn_L = del1_L.Read()*gain_1_L;
del2_out_horn_L = del2_L.Read()*gain_2_L;
del3_out_horn_L = del3_L.Read()*gain_3_L;

deld_out_horn_R = deld_R.Read()*gain_direct_R;
del1_out_horn_R = del1_R.Read()*gain_1_R;
```

```
        del2_out_horn_R = del2_R.Read()*gain_2_R;
        del3_out_horn_R = del3_R.Read()*gain_3_R;



        //deld_out_bass = deldbass.Read()*gain_bass;
        deld_out_bass = outputLow*gain_bass;



        // Calculate output gain of horn rotor
        sig_out_horn_L  = (deld_out_horn_L + del1_out_horn_L+
del2_out_horn_L + del3_out_horn_L);
        sig_out_horn_R  = (deld_out_horn_R + del1_out_horn_R +
del2_out_horn_R + del3_out_horn_R);

        sig_out_L = sig_out_horn_L + deld_out_bass;
        sig_out_R = sig_out_horn_R + deld_out_bass;

        // Output
        out[LEFT]  = sig_out_L;
        out[RIGHT] = sig_out_R;
    }
}

int main(void)
{
    // initialize seed hardware and daisysp modules
    float sample_rate;

    hw.Configure();
    hw.Init();
    hw.SetAudioBlockSize(128);



    sample_rate = hw.AudioSampleRate();

    //Initialize the button on pin 28
    button1.Init(hw.GetPin(28), sample_rate / 48.f);

    // Initialize delaylines
    deld_L.Init();
    del1_L.Init();
    del2_L.Init();
```

```
    del3_L.Init();

    deld_R.Init();
    del1_R.Init();
    del2_R.Init();
    del3_R.Init();

    deldbass.Init();

    // start callback
    hw.StartAudio(AudioCallback);

    while(1) {}
}

double filterH (double a0, double a1, double a2, double b1, double b2,
float input)
{
    double outd;

    outd = input * a0 + z1H;

    z1H = input * a1 + z2H - b1 * outd;
    z2H = input * a2 - b2 * outd;

    return outd;
}

double filterL (double a0, double a1, double a2, double b1, double b2,
float input)
{
    double outd;

    outd = input * a0 + z1L;

    z1L = input * a1 + z2L - b1 * outd;
    z2L = input * a2 - b2 * outd;

    return outd;
}
```

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit mit dem Titel:

_____

selbständig und nur mit den angegebenen Hilfsmitteln verfasst habe. Alle Passagen, die ich wörtlich aus der Literatur oder aus anderen Quellen wie z. B. Internetseiten übernommen habe, habe ich deutlich als Zitat mit Angabe der Quelle kenntlich gemacht.

Datum

Unterschrift