HAW HAMBURG

BACHELORARBEIT

Entwicklung eines haptischen Controllers zur drahtlosen Steuerung von Ableton Live

vorgelegt am 22. Juni 2026

Oliver Parvini, [redacted]

Erstprüferin: Prof. Dr. Eva Wilk
Zweitprüfer: B. Sc. Matthias Fehling

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Department Medientechnik
Finkenau 35
20081 Hamburg

Zusammenfassung

In dieser Bachelorarbeit wird die Entwicklung eines drahtlosen Controllers beschrieben, der ausgewählte Funktionen von Ableton Live über eine Kombination aus haptischer Steuerung und visueller Rückmeldung zugänglich macht.

Das System verbindet ein Stream Deck Module zur diskreten Bedienung und visuellen Anzeige mit zwei 360°-Potentiometern zur kontinuierlichen Steuerung. Beide werden über einen Raspberry Pi drahtlos an Ableton Live angebunden. Eine technische Messung und eine Nutzungsevaluation zeigen, dass der Potentiometer-Pfad mit geringer Latenz reagiert und die Bedienung nach einer kurzen Einführung erlernbar ist.

Abstract

This bachelor's thesis describes the development of a wireless controller that makes selected functions of Ableton Live accessible through a combination of haptic control and visual feedback.

The system combines a Stream Deck Module for discrete operation and visual display with two 360-degree potentiometers for continuous control. Both are connected wirelessly to Ableton Live through a Raspberry Pi. A technical measurement and a usability evaluation show that the potentiometer path responds with low latency and that the operation is learnable after a short introduction.

Inhaltsverzeichnis

Abkürzungsverzeichnis

ADC	Analog-to-Digital Converter
API	Application Programming Interface
BLE	Bluetooth Low Energy
CAD	Computer-Aided Design
CC	Control Change
HAT	Hardware Attached on Top
HCI	Human-Computer Interaction
HID	Human Interface Device
HTTP	Hypertext Transfer Protocol
I2C	Inter-Integrated Circuit
ICMP	Internet Control Message Protocol
IP	Internet Protocol
JSON	JavaScript Object Notation
LOM	Live Object Model
LSB	Least Significant Bit
OSC	Open Sound Control
PLA	Polylactic Acid
RTT	Round Trip Time
SDK	Software Development Kit
SVG	Scalable Vector Graphics
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UPS	Uninterruptible Power Supply
UUID	Universally Unique Identifier

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Die Arbeit mit digitalen Audio Workstations (DAWs) erfolgt überwiegend am Bildschirm. Aufnahme, Arrangement, Mischung und Performance werden in der Regel mit der Maus und der Tastatur bedient. Diese Steuerung ist präzise, bindet den Arbeitsablauf jedoch an einen festen Platz vor dem Monitor.

Hardware-Controller ergänzen diese Arbeitsweise um physische Bedienelemente. Sie verlagern bestimmte Bedienhandlungen vom Bildschirm auf das Gerät, verbleiben dabei aber meist an dem Arbeitsplatz.

Diese Arbeit untersucht, wie sich der Nutzen physischer Bedienelemente erhalten lässt, mit dem Ziel, einen portablen, drahtlosen Controller als Prototyp zu entwickeln, der haptische Steuerung mit visueller Rückmeldung verbindet und im praktischen Einsatz erprobt wird.

2 Problemstellung

Die Grenzen der bisherigen Bedienung lassen sich präzisieren, wenn man die Arbeit mit einer DAW als Abfolge unterschiedlicher Handlungen begreift. Schwierigkeiten entstehen dabei sowohl in der Art der Eingabe als auch in deren Zugänglichkeit. Entlang dieser Ebenen lassen sich die Problemfelder genauer benennen.

Diskrete und kontinuierliche Bedienhandlungen

Ein Tastendruck löst eine diskrete Eingabe aus, etwa die Auswahl eines Tracks oder den Start einer Aufnahme. Anders verläuft eine kontinuierliche Eingabe, die einen Parameter über einen Wertebereich hinweg verändert, beispielsweise die Lautstärke eines Tracks. Welches Eingabegerät optimal ist, hängt von der Art der Aufgabe ab (Rogers et al., 2005). Maus und Tastatur bedienen diskrete Aktionen direkt, fallen für kontinuierliche Wertänderungen jedoch auf eine indirekte Drag-Geste des Cursors zurück. Eine einzige Eingabeart deckt daher diskrete und kontinuierliche Handlungen nicht gleichermaßen ab.

Haptik

Physische Drehregler oder Fader erzeugen eine direkte mechanische Rückmeldung, da der Anwender spürt, wie weit er gedreht oder geschoben hat (De Pra et al., 2020). Der Aufmerksamkeitsbedarf eines Eingabegeräts hängt davon ab, wie gut es zur jeweiligen Eingabe passt (McLaughlin et al., 2009). Ein Bedienelement, das seine Bewegung selbst führt, bindet bei kontinuierlichen Eingaben nur wenig Aufmerksamkeit, sodass der Blick bei der musikalischen Tätigkeit bleiben kann. Wessel und Wright (2001) sehen die fehlende körperliche Verbindung zwischen Geste und Klang als Hindernis dafür, dass Computer als Musikinstrumente überzeugen. Für kontinuierliche musikalische Eingaben sind haptische Bedienelemente damit besser geeignet als bildschirmgebundene Eingabeformen.

Räumliche Bindung

Ein haptisches Bedienelement ist nur dort nutzbar, wo der Anwender es erreichen kann. Hardware-Controller sind meist kabelgebunden und verbleiben daher am Arbeitsplatz. Bei Tätigkeiten, die ohnehin am Arbeitsplatz stattfinden, etwa beim Mischen an der Abhörposition (ITU-R, 2015), entsteht daraus keine Einschränkung. Findet die Handlung jedoch im Raum statt, beispielsweise beim Einspielen externer Instrumente oder bei Gesangsaufnahmen, dann erzwingt eine DAW-Bedienung einen Wechsel zur Workstation und unterbricht den Arbeitsprozess. Solche Unterbrechungen führen zu einer Verzögerung der Wiederaufnahme sowie zu einem erhöhten Stressempfinden und beeinträchtigen damit die ursprüngliche Tätigkeit (Hausen et al., 2013). Bei musikalischen Eingaben dürfte dieser Effekt besonders relevant sein, da sich eine begonnene Spielbewegung nicht ohne Verlust unterbrechen lässt. Solche Unterbrechungen könnten durch einen drahtlos einsetzbaren Controller vermieden werden.

Verständliche Belegung

Eine DAW bietet mehr Funktionen und Parameter als ein kompakter Controller physisch abbilden kann (Gohlke et al., 2010). Werden Bedienelemente deshalb mehrfach belegt, muss die aktive Belegung am jeweiligen Bedienelement erkennbar bleiben. Die Usability-Forschung bezeichnet dieses Prinzip als „Wiedererkennen statt Erinnern" (Nielsen, 1994). Eine Bedienoberfläche soll sichtbar bleiben, damit sich Nutzende Informationen nicht von einem Bedienschritt zum nächsten merken müssen.

Die Problematik lässt sich nicht in ihrer Gesamtheit lösen, wenn die Problemfelder isoliert betrachtet werden. Ein Controller, der nur eines dieser Felder adressiert, lässt die übrigen Schwächen der bisherigen Bedienform bestehen.

3 Bestehende Ansätze

Es werden drei am Markt verfügbare Systeme betrachtet. Die Einordnung erfolgt jeweils entlang der beschriebenen Problemfelder.

3.1 Stream Deck +

Das Stream Deck + ist ein USB-Controller mit acht LCD-Tasten, vier Encodern mit Druckfunktion und einem LCD-Touchstreifen. Jede Taste ist gleichzeitig ein eigenes Display, wodurch Bedienelement und Anzeigefläche am selben Ort liegen. Der Touchstreifen kann Werte und Beschriftungen der vier Encoder darstellen sowie Seitenwechsel auslösen (Elgato, o. D.-a).

Über die zugehörige Software lassen sich Tasten und Encoder mit Aktionen, Profilen und kombinierten Abläufen belegen. Ein Software Development Kit (SDK) ermöglicht die Entwicklung eigener Plugins (Elgato, o. D.-b).

Abbildung 1: Stream Deck +.
Quelle: Elgato (o. D.-a).

In Bezug auf die vier Problemfelder bietet das Stream Deck + vor allem eine visuelle Belegung, die die aktuelle Zuordnung unmittelbar erkennbar macht. Diskrete Eingaben erfolgen über die physischen LCD-Tasten. Mit den vier Encodern lassen sich Werte kontinuierlich ändern, und die Rastung gibt eine mechanische Rückmeldung. Sie sind allerdings vergleichsweise klein, was präzise Bewegungen über größere Wertebereiche erschwert. Die räumliche Bindung zum Arbeitsplatz bleibt durch die USB-Anbindung bestehen.

3.2 nOb Control

nOb Control ist ein USB-Controller mit einem einzigen, großformatigen Endlosdrehregler. Das Gerät fungiert als HID-Eingabegerät (Human Interface Device) und steuert den Parameter, über dem sich der Mauszeiger befindet, durch simulierte Klick- und Ziehbewegungen. Über das zugehörige Assignment Center können zusätzlich Tastatur-, MIDI- oder OSC-Zuweisungen hinterlegt werden (NOB Control Solutions, 2023).

Abbildung 2: nOb Control.
Quelle: NOB Control Solutions (2023).

nOb Control adressiert die kontinuierliche Steuerung und die haptische Eingabe. Mit dem großen, freilaufenden Drehregler lässt sich der Wert geführt und ohne Blick auf den Bildschirm ändern. Diskrete Eingaben bietet das Gerät dagegen nicht, da die Auswahl des Zielparameters über die Mausposition am Bildschirm erfolgt. Da der Bildschirm auch die einzige Rückmeldung zur aktuellen Zuordnung liefert, ist am Bedienelement keine sichtbare Belegung vorhanden. Ohne Sichtkontakt zum Bildschirm lässt sich das Gerät nicht sinnvoll bedienen, sodass es an den Arbeitsplatz gebunden bleibt.

3.3 Knobbler

Bei Knobbler handelt es sich um eine iPad-App zur Steuerung von Ableton Live. Eine erste Version kombinierte ein TouchOSC-Layout mit einem Max for Live-Device, um Parameterzuordnungen auf dem Tablet sichtbar zu machen (Steinkamp, 2022). Die aktuelle Version Knobbler4 setzt diesen Ansatz als eigenständige App fort. Parameter-, Track- und Gerätenamen sowie Trackfarben werden mit dem Live-Set synchronisiert und Mappings lassen sich innerhalb des Sets speichern (Steinkamp, o. D.).

Abbildung 3: Knobbler.
Quelle: Steinkamp (2022).

Die sichtbare Belegung entsteht durch die direkte Anbindung an das Live-Set, und sowohl diskrete als auch kontinuierliche Eingaben sind über das Touch-Display möglich. Die räumliche Bindung an die Workstation entfällt durch den drahtlosen Betrieb des iPads. Offen bleibt die haptische Eingabe, da die Bedienung über eine glatte Touchfläche ohne mechanische Führung erfolgt und so keine blickunabhängige Drehbewegung erlaubt.

Keines der drei Systeme deckt alle vier Problemfelder ab. Das Stream Deck + und nOb Control bleiben kabelgebunden, und Knobbler verzichtet auf eine geführte mechanische Bewegung. Ein Gerät, das alle vier Problemfelder als zusammengehörige Eigenschaften eines Bedienkonzepts behandelt, existiert bislang nicht.

4 Systemanforderungen

Aus den beschriebenen Problemfeldern und den Befunden zu bestehenden Ansätzen ergeben sich die folgenden Systemanforderungen. Funktionale Anforderungen beschreiben, welche Bedienhandlungen der Controller ermöglichen soll. Nichtfunktionale Anforderungen beschreiben qualitative Eigenschaften der Umsetzung.

Tabelle 1: Funktionale Anforderungen

	Funktionale Anforderung
F1	Das System soll Transportfunktionen der DAW steuern können.
F2	Tracks sollen ausgewählt und aufnahmebereit geschaltet werden können.
F3	Devices eines Tracks sollen zugänglich gemacht werden, d. h. anzeigbar, auswählbar und aktivierbar oder deaktivierbar sein.
F4	Parameter von Devices sollen ausgewählt und kontinuierlich verändert werden können.
F5	Mix-relevante Werte wie Volume, Pan und Sends sollen verändert werden können.
F6	Das System soll die Navigation im Arrangement unterstützen.

Tabelle 2: Nichtfunktionale Anforderungen

	Nichtfunktionale Anforderung
N1	Die Bedienlogik soll sich Nutzenden mit DAW-Erfahrung nach einer kurzen Einführung in die Grundbedienung erschließen.
N2	Die visuelle Rückmeldung soll die aktuelle Funktion eines Bedienelements und die DAW-Zustände eindeutig erkennbar machen.
N3	Der Wechsel eines Mapping-Ziels soll nicht zu unbeabsichtigten Veränderungen des Zielwerts führen.
N4	Änderungen in Ableton Live sollen auf der Bedienoberfläche ohne störend wahrnehmbare Verzögerung sichtbar werden.
N5	Die Verzögerung zwischen einer Bedienhandlung am Controller und ihrer Wirkung in Ableton Live soll so gering sein, dass sie nicht als störend wahrgenommen wird.
N6	Das System soll im Betrieb ohne Kabelverbindung zum Hostrechner auskommen. Das umfasst sowohl die Datenübertragung als auch die Stromversorgung.

5 Entwicklungskonzept

Aus den Anforderungen werden im Folgenden die Bauteile und die grundlegende Systemarchitektur abgeleitet.

5.1 Wahl der kontinuierlichen Steuerung

Für die kontinuierliche Steuerung kommen verschiedene Bauformen in Betracht.

Fader besitzen eine gut ablesbare absolute Position, benötigen aber vergleichsweise viel Platz. Bei dynamisch belegten Controllern entsteht außerdem das Problem, dass die physische Faderposition nicht zwangsläufig dem aktuell ausgewählten Softwarewert entspricht. Motorisierte Fader könnten diesen Unterschied ausgleichen, würden jedoch die Baugröße, den Energiebedarf und den technischen Aufwand erhöhen.

Rotatorische Bedienelemente hingegen sind kompakter und entsprechen der üblichen Darstellungsform auf der Oberfläche von Audio-Plugins. Encoder können endlos drehen und relative Steuerdaten erzeugen, besitzen allerdings häufig eine gerasterte Haptik. Klassische Potentiometer bieten eine glatte, kontinuierliche Drehbewegung, sind aber in der Regel absolute Bedienelemente mit mechanischen Endpunkten.

Ein 360°-Potentiometer besitzt keinen mechanischen Endanschlag, bietet jedoch den gleichmäßigen Drehwiderstand eines klassischen Potentiometers. Da die Schleiferspannung analog vorliegt, lässt sich die Drehposition mit hoher Auflösung erfassen. Die Auswertung für 360°-Potentiometer ist in mehreren offenen Implementierungen dokumentiert und wird über einen Mikrocontroller realisiert (Brandal, 2020; Little Devil, o. D.; Shafran, o. D.). Damit würde die Anforderung an die kontinuierliche Parametersteuerung (F4) erfüllt werden.

Eine musikalische Steuerung betrifft häufig miteinander verknüpfte Parameter. Orio et al. (2001) und Wessel und Wright (2001) heben für musikalische Interfaces den Wert der simultanen Kontrolle über mehrere Parameter hervor. Daher sind für den Prototyp zwei Potentiometer vorgesehen, sodass sich Parameterpaare wie Input- und Output-Gain gleichzeitig steuern lassen, ohne bei jeder Änderung zwischen zwei Zielen umschalten zu müssen.

5.2 Visuelle Rückmeldung und Bedienoberfläche

Die Bedienoberfläche soll diskrete Aktionen auslösen (F1, F2, F3, F6) und zugleich anzeigen, welche Funktion ein Bedienelement im aktuellen Kontext hat (N2). Da sich Funktionen und Werte je nach Track oder Device ändern, wäre eine feste Beschriftung am Gehäuse ungeeignet. Eine reine Bildschirmoberfläche, wie sie Ableton Live bereits bietet, löst diese Aufgabe visuell, verlangt jedoch ständige Aufmerksamkeit auf den Monitor.

Mit einer Kombination aus physischen Tasten und einem separaten Display könnten Funktionen angezeigt und über die Tasten ausgelöst werden. Bei einem kompakten Controller entsteht dabei ein zusätzliches Zuordnungsproblem, da die Anzeige und das zugehörige Bedienelement räumlich getrennt sind und der Nutzer erkennen muss, welche Information zu welchem Button gehört. Außerdem erhöht eine solche Lösung die Anzahl der Bauteile und den Platzbedarf.

Eine Touchscreen-Oberfläche kann Seiten, Beschriftungen und Zustände flexibel darstellen, wie es bei Knobbler der Fall ist. Touchscreens fehlt jedoch eine taktile Abgrenzung einzelner Bedienelemente. Ein Stream Deck verbindet beim einzelnen Bedienelement Taste und Display, sodass eine diskrete Eingabe und deren Anzeige zusammenfallen. Damit entfällt das Zuordnungsproblem zwischen Anzeige und Bedienelement, und je nach Seite oder Betriebsmodus lassen sich unterschiedliche Funktionen auf derselben Taste darstellen (N2).

5.3 Ableton Live Zugriffsschicht

Eine feste MIDI-Zuweisung genügt für das Bedienkonzept nicht. Das in Ableton Live integrierte MIDI-Mapping (Map Mode) erstellt eine reine Zuweisungstabelle zwischen MIDI-Nachrichten und Live-Parametern (Ableton, o. D.-a). Es kann daher weder den aktuellen Kontext eines Live-Sets auswerten noch Zustandsänderungen an den Controller zurückgeben.

Ableton stellt als tief integrierte Schnittstelle das Live Object Model (LOM) zur Verfügung, eine hierarchische Anordnung von Objekten, Eigenschaften und Funktionen, die externe Prozesse über eine Skriptschnittstelle adressieren können (Cycling '74, o. D.-a).

Ein Device-Parameter lässt sich beispielsweise über einen Canonical Path im LOM adressieren (s. Abb. 4). Seine Eigenschaften umfassen unter anderem Name, Wert, Wertebereich sowie einen formatierten Anzeigewert mit Einheit (Cycling '74, o. D.-b). Damit lässt sich ein Parameter verändern und zugleich für die Anzeige am Stream Deck aufbereiten.

Den Zugriff auf das LOM realisiert ein Ableton Remote Script, ein in Python implementierbares Modul, das in Ableton Live als Control Surface eingebunden wird und über die Live-API die laufende Steuerlogik für externe Controller bereitstellt (Structure Void, o. D.). Benutzerdefinierte Remote Scripts werden in der User Library abgelegt und in den MIDI-Einstellungen als Control Surface ausgewählt (Ableton, o. D.-b).

Über diese bidirektionale Struktur lassen sich die funktionalen Anforderungen sowie die visuelle Rückmeldung (N2) und die Zustandsänderungen (N4) umsetzen.

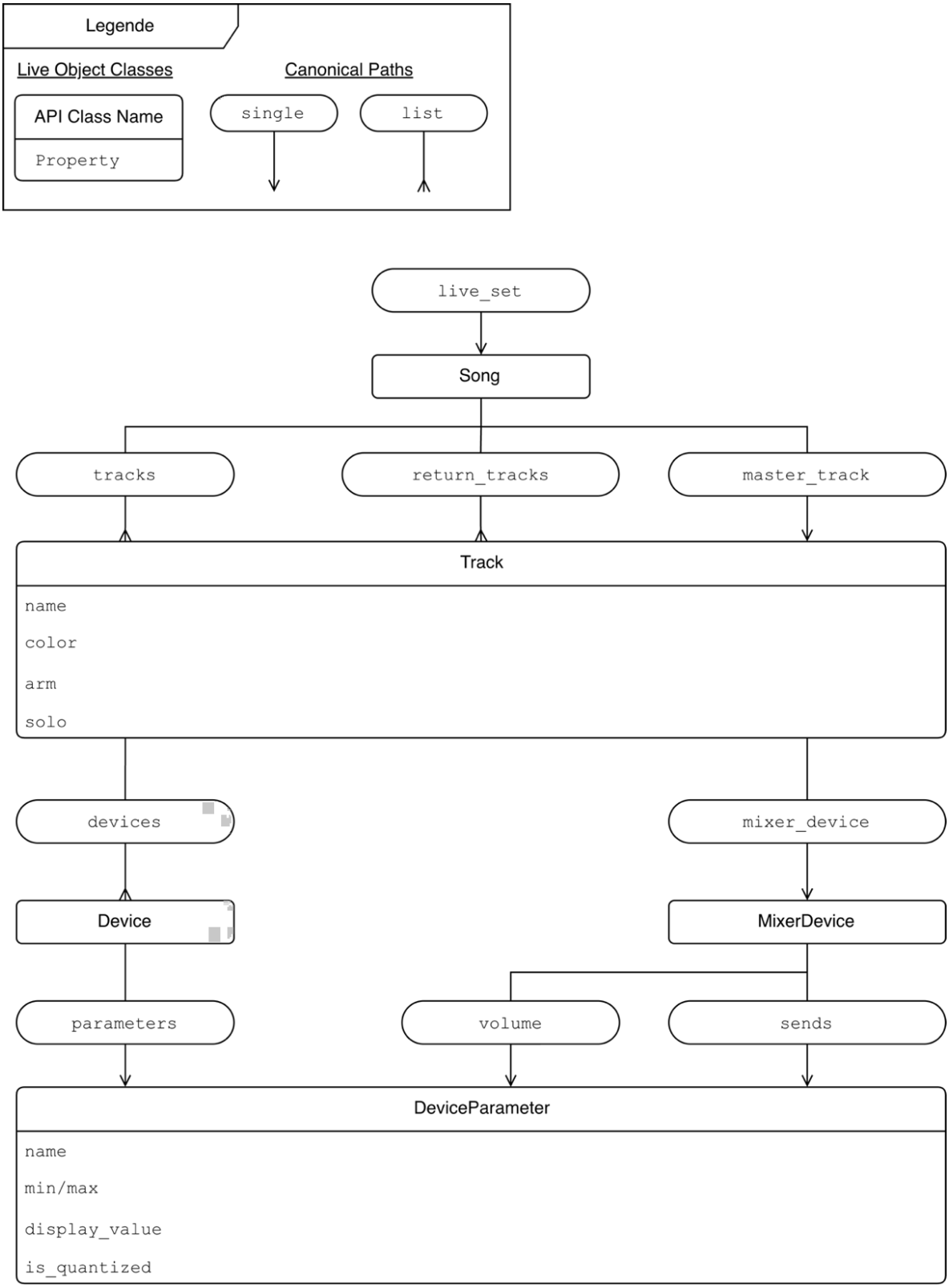


Abbildung 4: Vereinfachte Darstellung des LOM.
Quelle: Eigene Darstellung in Anlehnung an Cycling '74 (o. D.-a).

5.4 Drahtlose Anbindung

Für die drahtlose Übertragung kommen BLE-MIDI (Bluetooth Low Energy MIDI), OSC sowie USB-over-IP in Frage.

BLE-MIDI überträgt ausschließlich MIDI-Nachrichten (Wang et al., 2019), während OSC frei definierte Steuernachrichten über das Netzwerk überträgt und etwa von AbletonOSC zur Ansteuerung des LOM genutzt wird (Jones, 2023). Beide adressieren jedoch nur die musikalische Steuerebene, während das Stream Deck als USB-Gerät weder über MIDI noch über OSC kommuniziert und daher eine zusätzliche Übertragungs- und Protokollschicht erfordern würde.

USB-over-IP umgeht diese Trennung, indem es die USB-Endpunkte der Geräte direkt über das Netzwerk weiterleitet. Aus Sicht des Hostrechners erscheinen diese weiterhin als lokal angeschlossene USB-Geräte (VirtualHere, o. D.). Für die Übertragung wird im Prototyp ein Raspberry Pi verwendet, der den Host, das Stream Deck und den Mikrocontroller miteinander verbindet.

Diese Entscheidung gilt für den konkreten Aufbau und ist nicht als generelle Bewertung von USB-over-IP zu verstehen.

5.5 Stromversorgung

Das Stream Deck und der Mikrocontroller hängen am Raspberry Pi und werden über dessen USB-Anschlüsse mitgespeist, sodass der Pi die Versorgung der gesamten Eingabeebene übernimmt. Fällt die Spannung unter Last ab, drohen ein Neustart oder die Trennung der USB-Geräte. Für die Anforderung N6 braucht der Prototyp deshalb eine mobile Stromquelle, die den Pi samt angeschlossenen Geräten versorgt und beim Wechsel zwischen Netz- und Akkubetrieb nicht unterbricht. Der Ladezustand soll zugleich für die Anzeige auf der Bedienoberfläche verfügbar sein.

Eine USB-Powerbank ließe sich einfach anschließen, bietet jedoch meist keinen unterbrechungsfreien Pass-Through-Betrieb. Beim Umschalten auf den Akku fällt die Ausgangsspannung kurzzeitig ab. Ein eigens aufgebautes Akkupack mit Lade- und Schutzelektronik wäre technisch möglich, würde jedoch den Aufbau- und Sicherheitsaufwand erhöhen, ohne einer Fertiglösung gegenüber Vorteile zu bieten.

Der Prototyp setzt stattdessen auf einen UPS HAT, ein Aufsatzmodul für den Raspberry Pi mit einer Schaltung für Akkus und einem I2C-Bus, über den Werte zur Spannung und zum Ladezustand ausgelesen werden können (Waveshare, o. D.).

Abbildung 5 fasst alle Komponenten zusammen. Die Eingabeebene bilden das Stream Deck und die Potentiometer an einem Mikrocontroller. Der Raspberry Pi mit UPS HAT versorgt das System mit Strom und überträgt die Eingaben drahtlos, die auf dem Hostrechner das Stream Deck-Plugin und das Remote Script mit dem LOM verbinden.

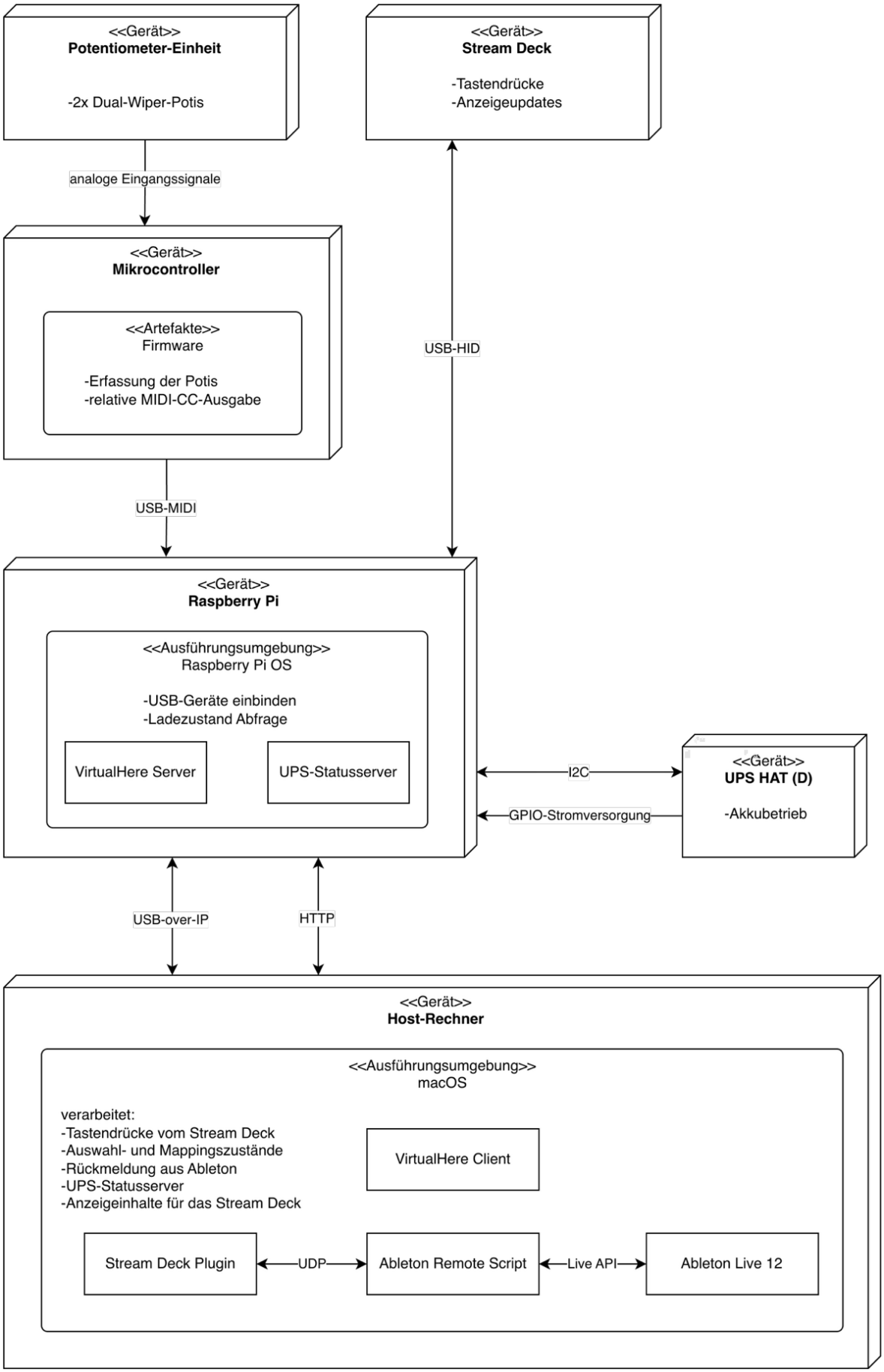


Abbildung 5: Gesamtarchitektur des Controllers.

6 Realisierung

Dieses Kapitel beschreibt die technische Umsetzung des Prototyps. Die Darstellung orientiert sich an der Systemstruktur. Die Hardware, Firmware und Software entstanden iterativ und in mehreren parallelen Schritten. Die gezeigten Code-Artefakte, also Firmware, Remote Script und Stream Deck-Plugin, entstanden unter Einsatz von KI-gestützten Werkzeugen. Konzeption, Architekturentscheidungen, Integration und Prüfung des Systems liegen beim Autor. Der vollständige Quellcode ist im USB-Anhang (01_Quellcode) hinterlegt.

6.1 Hardware

Die Hardware des Prototyps umfasst ein Stream Deck Module, zwei 360°-Dual-Wiper-Potentiometer an einem Pro Micro, einen Raspberry Pi 3 Model B+ und ein UPS HAT der Modellreihe D.

6.1.1 Stream Deck Module

Als Bedieneinheit wird ein Stream Deck Module mit 15 Tasten verwendet. Das Modul entspricht funktional einem Stream Deck, wird aber ohne Gehäuse für die Integration in eigene Systeme ausgeliefert (Elgato, 2025). Der Anschluss erfolgt über USB-C, und auf dem Hostsystem wird das Modul von der Software des Stream Decks erkannt.

Abbildung 6: Stream Deck Module mit 15 Tasten.
Quelle: Elgato (2025).

6.1.2 360°-Potentiometer und Pro Micro

Für die kontinuierliche Steuerung werden zwei 360°-Dual-Wiper-Potentiometer der Serie Alpha RV112FF mit jeweils 2,5 kΩ eingesetzt (s. Anhang A) und mit einem Arduino-kompatiblen Pro Micro kombiniert (SparkFun, o. D.).

Ein Dual-Wiper-Potentiometer besitzt zwei Schleiferausgänge, die als W1 und W2 bezeichnet werden (s. Abb. 8). Beide Ausgänge liefern Signale, die um etwa 90° phasenversetzt sind. Aus diesem Wertepaar rekonstruiert die Firmware die Drehposition.

Abbildung 7: Alpha RV112FF
Quelle: Shafran (o. D.).

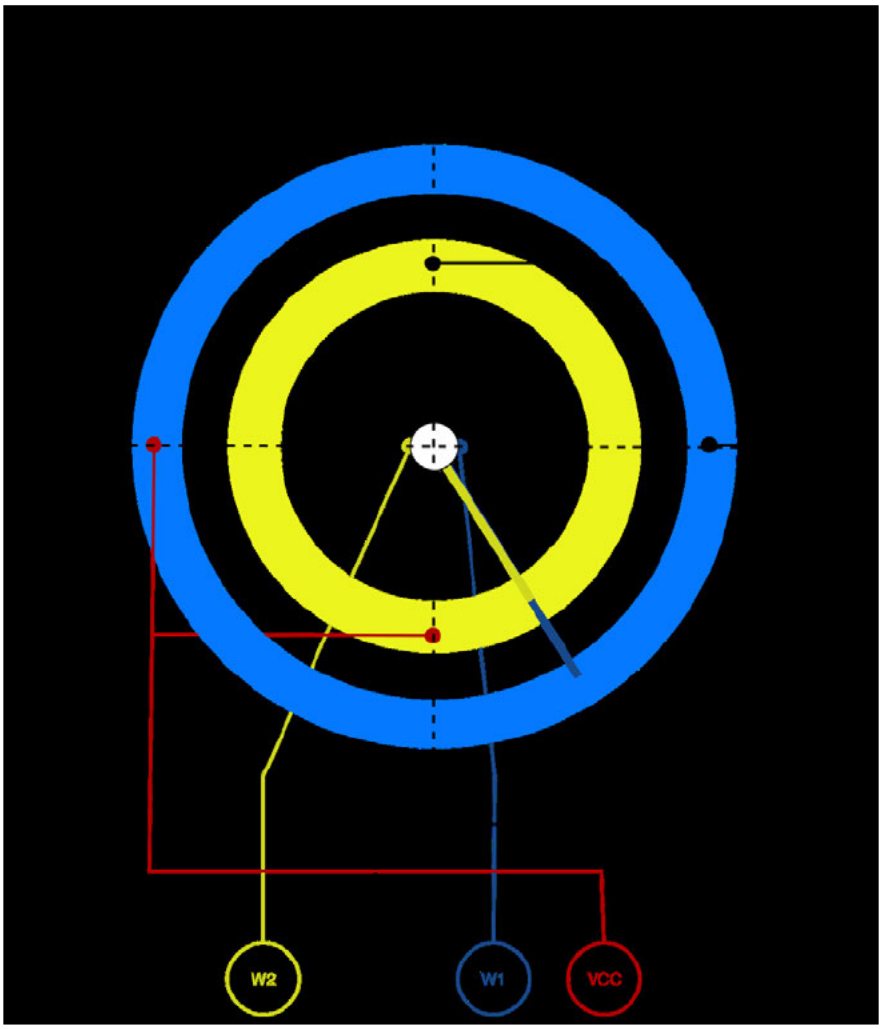


Abbildung 8: Schematisches Modell eines 360°-Potentiometers.

Beide Potentiometer werden gemeinsam über einen VCC-BUS mit der Versorgungsspannung und über einen GND-BUS mit der Masse verbunden. Die vier Schleifersignale werden jeweils an einen analogen Eingang des Pro Micro angeschlossen. Im konkreten Aufbau befinden sich die beiden Schleifer des ersten Potentiometers an A0 und A1 und die beiden Schleifer des zweiten Potentiometers an A2 und A3 (s. Abb. 9).

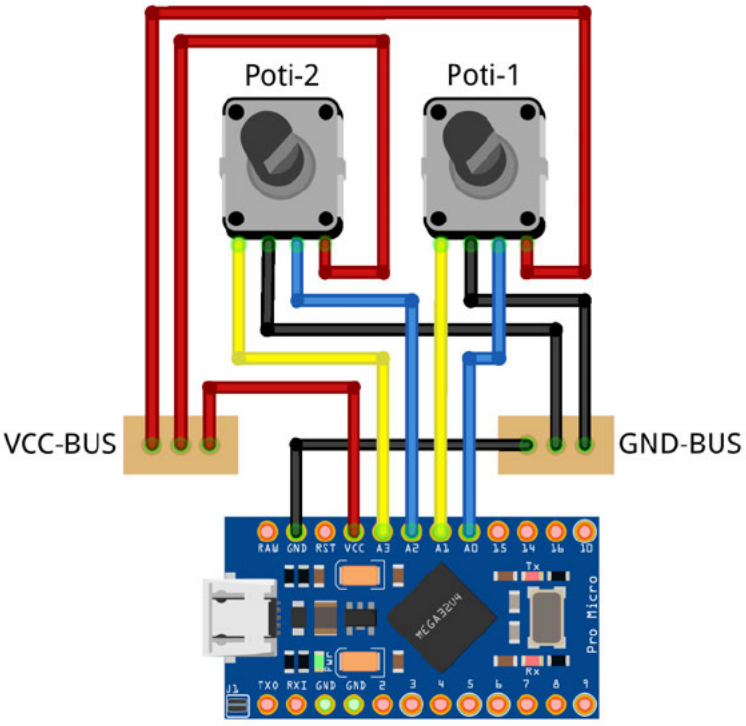


Abbildung 9: Verschaltung der 360°-Potentiometer.

6.1.3 Raspberry Pi und UPS HAT

Die USB-Geräte des Controllers werden an einem Raspberry Pi 3 Model B+ zusammengeführt. Seine vier USB-2.0-Anschlüsse nehmen das Stream Deck und den Mikrocontroller auf und lassen zwei Anschlüsse für weitere Belegungen frei. Das integrierte WLAN übernimmt die Anbindung an den Hostrechner und eine I2C-Schnittstelle dient der Abfrage des UPS HAT (Raspberry Pi Ltd., o. D.). Da der Pi nur USB-Weiterleitung und I2C-Polling leistet, erweist sich seine Rechenleistung im Betrieb als ausreichend. Sein Stromverbrauch liegt zudem unter dem des neueren Pi 4 (Tom's Hardware, 2019). Das ist im Akkubetrieb von Vorteil, da auf eine aktive Kühlung im geschlossenen Gehäuse verzichtet werden kann.

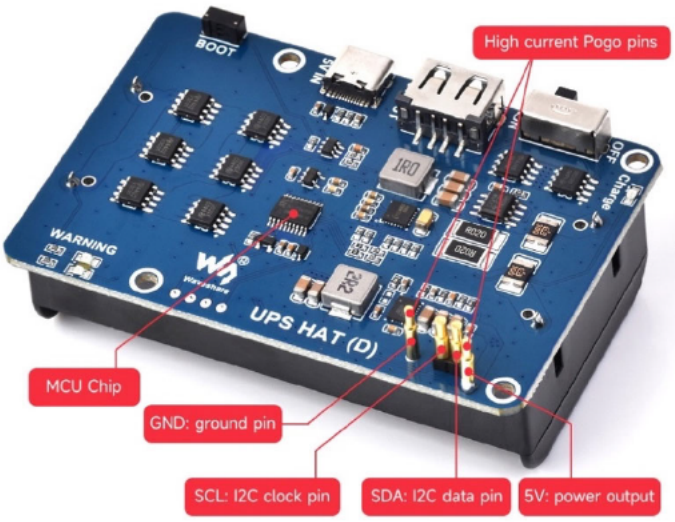


Abbildung 10: Waveshare UPS HAT (D).
Quelle: Waveshare (o. D.).

Als Stromversorgung dient ein Waveshare UPS HAT (D). Das Modul integriert Ladeelektronik, Spannungswandlung und Statusmessung auf einer Platine und nimmt zwei 21700-Lithiumzellen in Parallelschaltung auf (Waveshare, o. D.). Der HAT wird unterhalb des Pi montiert, sodass die Baugruppe kompakt bleibt und die Oberseite des Pi für Anschlüsse und die Gehäuseintegration frei bleibt. Die Laufzeit hängt vom Gesamtverbrauch ab und wird in dieser Arbeit nicht festgelegt.

6.2 Firmware

Der Pro Micro liest die beiden Potentiometer aus und gibt deren Drehbewegung an den Raspberry Pi weiter. Die Signalverarbeitung folgt einer online verfügbaren Implementierung für 360°-Potentiometer (Shafran, o. D.). Die übernommene Umsetzung liefert einen absoluten Reglerwert von 0 bis 127. Die vorliegende Firmware gibt die Drehbewegung als relative Schritte aus.

6.2.1 Einlesen der Schleifersignale

Jedes Potentiometer wird im Code als `DualWiperPot`-Objekt eingeführt, das die beiden Wiper-Pins, die zugehörige CC-Nummer, zwei Glättungsfilter und den Bewegungszustand für die spätere Schrittberechnung bündelt. Als CC-Nummern dienen 14 und 15, da sie im MIDI-Standard nicht fest belegt sind und daher nicht mit bestehenden Zuweisungen kollidieren (MIDI Association, o. D.).

Die Funktion `readAnalogStable` liest jeden Wiper-Pin zweimal und verwendet nur den zweiten Wert. Die erste Wandlung nach einem Wechsel des ADC-Kanals ist auf dem Pro Micro-Chip

(ATmega32U4) ungenau und wird laut Datenblatt verworfen (Atmel, 2010). Anschließend glättet ein adaptiver Filter die Rohwerte abhängig von der Größe der Drehbewegung (Clarke, 2019).

Codeblock 1: Aufbau des DualWiperPot-Objekts und Einlesen der Schleifersignale
Quelle: Eigener Code, Glättung mit ResponsiveAnalogRead (Clarke, 2019), dual_pot_midi_firmware.ino

```
85    static inline int readAnalogStable(int pin) {
87    analogRead(pin);
88    return analogRead(pin);
89    }
126   struct DualWiperPot {
127   int w1Pin;
128   int w2Pin;
130   uint8_t cc;
132   ResponsiveAnalogRead r1;
133   ResponsiveAnalogRead r2;
136   int pendingDelta;
137   bool movementActive;
138   int activeDirection;
162   int readDialPosition() {
164   int raw1 = readAnalogStable(w1Pin);
165   int raw2 = readAnalogStable(w2Pin);
167   r1.update(raw1);
168   r2.update(raw2);
170   int w1 = r1.getValue();
171   int w2 = r2.getValue();
319   DualWiperPot pot1(POT1_WIPER1, POT1_WIPER2, POT1_CC, 0.02);
320   DualWiperPot pot2(POT2_WIPER1, POT2_WIPER2, POT2_CC, 0.02);
```

Der `snap`-Parameter steuert ihre Stärke und ist hier auf 0,02 eingestellt. Im Stillstand dämpft die Bibliothek stark. Bei schneller Drehung lässt die Dämpfung nach, sodass der Filter dem Schleifer eng folgt. Diese Dämpfung unterdrückt das LSB-Rauschen der ADC-Werte (Clarke, 2016), das sonst ungewollte Richtungsumkehrungen in der Automationskurve auslösen würde.

6.2.2 Positionsberechnung mit *atan2*

Die Rohwerte des 10-Bit-ADC liegen im Bereich von 0 bis 1023, auf den der Glättungsfilter eingestellt ist. Beide Achsen werden umgerechnet, sodass der Vektor *(fx, fy)* symmetrisch zum Ursprung liegt und die Werte im Bereich von −1 bis +1 liegen.

Codeblock 2: Normalisierung der Schleifersignale
Quelle: In Anlehnung an Shafran (o. D.), dual_pot_midi_firmware.ino

```
157   void begin() {
158   r1.setAnalogResolution(1023);
159   r2.setAnalogResolution(1023);
160     }


173   double fx = ((double)w1 / 511.5) - 1.0;
174   double fy = ((double)w2 / 511.5) - 1.0;
```

$$x = \frac{w_1}{511{,}5} - 1$$

$$y = \frac{w_2}{511{,}5} - 1$$

Codeblock 3: Winkelberechnung und Abbildung auf 0 bis 2047
Quelle: In Anlehnung an Shafran (o. D.), dual_pot_midi_firmware.ino

```
175   double angle = atan2(fy, fx);


177  int dialPosition = (int)lround((angle + PI) * ((double)DIAL_MAX_POSITION
/ (2.0 * PI)));
178   return clampDialPosition(dialPosition);
```

Die Funktion *atan2* berücksichtigt die Vorzeichen beider Koordinaten und bestimmt den Quadranten eindeutig. Sie liefert den Winkel im Bereich von $-\pi$ bis $+\pi$.

$$\theta = atan2(fx, fy)$$

Die Addition von π verschiebt den Winkel zunächst in den positiven Bereich von 0 bis 2π. Anschließend wird er auf den Bereich 0 bis 2047 abgebildet und auf die nächste ganze Stufe gerundet, da die Position als ganzzahliger Wert geführt wird. Die Auflösung ist feiner als die 1024 Werte eines einzelnen 10-Bit-Kanals, weil die Position aus beiden Schleiferkanälen berechnet wird.

$$p = round\left(\frac{(\theta + \pi) \cdot 2047}{2\pi}\right)$$

6.2.3 Relative Bewegungsberechnung

Eine absolute Position würde einen Wertsprung auslösen, sobald die physische Reglerstellung und der Parameterwert auseinanderlaufen. Da der Positionswert zyklisch ist, muss der Übergang zwischen 2047 und 0 korrigiert werden. Dazu vergleicht `computeDelta` die Differenz mit einer Schwellenwertgrenze von 1024 Schritten. Übersteigt ihr Betrag diese Grenze, hat sich der Schleifer über die Übergangsstelle hinwegbewegt.

Codeblock 4: Differenzbildung zwischen zwei Positionswerten
Quelle: In Anlehnung an Shafran (o. D.), dual_pot_midi_firmware.ino

```
92    static int computeDelta(int prevPos, int currentPos) {
93    int delta = currentPos - prevPos;
```

$$\Delta p = p_{\text{aktuell}} - p_{\text{vorher}}$$

Die Firmware addiert oder subtrahiert die Stufenzahl einer Umdrehung und erhält so einen kurzen Drehweg statt eines Sprungs über fast den gesamten Kreis. Das Vorzeichen des Ergebnisses gibt dabei die Drehrichtung an.

Codeblock 5: Korrektur des Übergangs
Quelle: In Anlehnung an Shafran (o. D.), dual_pot_midi_firmware.ino

```
94    if (delta > DIAL_WRAP_HALF) delta -= DIAL_RESOLUTION_STEPS;
95    else if (delta < -DIAL_WRAP_HALF) delta += DIAL_RESOLUTION_STEPS;
96    return delta;
97      }
```

$$\Delta p_{\text{korrigiert}} = \begin{cases} \Delta p - 2048 & \text{wenn } \Delta p > 1024 \\ \Delta p + 2048 & \text{wenn } \Delta p < -1024 \\ \Delta p & \text{sonst} \end{cases}$$

6.2.4 Filterung

Die Filterstufe `addPendingDelta` entfernt das Schwanken der ADC-Werte im Stillstand sowie Sprünge des *atan2*-Ergebnisses bei bestimmten Schleiferwinkeln. Beide würden ungewollte Wertänderungen in Ableton auslösen. Der Akkumulator und der Bewegungszustand befinden sich im Objekt des Potentiometers, sodass jedes Potentiometer einzeln gefiltert wird.

Sobald eine Drehrichtung festgelegt ist, entfernt der Filter einzelne Gegenschritte. Eine tatsächliche Richtungsänderung bestünde aus mehreren aufeinanderfolgenden Gegenschritten. Sprünge von mehr als 512 Schritten gelten als ADC-Spitze und setzen den Akkumulator zurück. Sprünge werden maximal auf 63 begrenzt.

Codeblock 6: Filterung der Einzeländerungen
Quelle: Eigener Code, dual_pot_midi_firmware.ino

```
181   void addPendingDelta(int delta) {
182   if (delta == 0) return;
183   unsigned long now = millis();


187   if (abs(delta) > REL_GLITCH_REJECT_STEPS) {
188   pendingDelta = 0;
189   movementActive = false;
190   activeDirection = 0;
191   lastDeltaMs = now;
```

```
192   return;
193     }

195   if (delta > REL_SINGLE_SAMPLE_LIMIT_STEPS) {
196   delta = REL_SINGLE_SAMPLE_LIMIT_STEPS;
197     } else if (delta < -REL_SINGLE_SAMPLE_LIMIT_STEPS) {
198   delta = -REL_SINGLE_SAMPLE_LIMIT_STEPS;
199     }

206   if (movementActive && activeDirection != 0
207   && delta != 0 && ((delta > 0) != (activeDirection > 0))
208   && abs(delta) <= 1) {
209   return;
210     }

212   delta *= REL_SENSITIVITY;
213   pendingDelta += delta;
214   lastDeltaMs = now;

216       if (pendingDelta > REL_ACCUMULATOR_LIMIT) pendingDelta =
REL_ACCUMULATOR_LIMIT;
217     if (pendingDelta < -REL_ACCUMULATOR_LIMIT) pendingDelta = -
REL_ACCUMULATOR_LIMIT;

219   if (abs(pendingDelta) >= REL_START_THRESHOLD_STEPS) {
220   movementActive = true;
221   activeUntilMs = now + REL_ACTIVE_WINDOW_MS;
222   activeDirection = (pendingDelta > 0) ? 1 : -1;
223     } else if (movementActive) {
224   activeUntilMs = now + REL_ACTIVE_WINDOW_MS;
225     }
```

Der gefilterte Schritt wird dem Akkumulator hinzugefügt, der auf 2047 begrenzt ist und bei schnellen Drehungen nicht überschritten wird. Erreicht der Akkumulator die Startschwelle von 2 Schritten, gilt die Bewegung für 200 ms als aktiv. In diesem Fenster müssen einzelne Folgeschritte die Startschwelle nicht erneut überwinden.

6.2.5 Ausgabe und Schrittweite

Ob aus dem Akkumulator eine MIDI-Nachricht erzeugt wird, entscheidet `flushPendingDelta`. Zuerst prüft die Funktion, ob das Aktivfenster abgelaufen ist, und setzt in diesem Fall den Bewegungszustand zurück (Zeilen 230 bis 233).

Codeblock 7: Verwerfen alter Restwerte
Quelle: Eigener Code, dual_pot_midi_firmware.ino

```
228   void flushPendingDelta() {
229   unsigned long now = millis();
230   if (movementActive && (long)(now - activeUntilMs) >= 0) {
231   movementActive = false;
232   activeDirection = 0;
233     }


235   if (pendingDelta == 0) return;


237   if (!movementActive && abs(pendingDelta) < REL_START_THRESHOLD_STEPS)
{
238   if (now - lastDeltaMs >= REL_IDLE_RESET_MS) {
239   pendingDelta = 0;
240     }
241   return;
242     }


244   if (now - lastSendMs < REL_SEND_INTERVAL_MS) return;
```

Liegt der angesammelte Wert unter der Startschwelle von 2 Schritten und ist keine Drehung aktiv, wird nichts gesendet. Der Akkumulator wird erst nach 280 ms Ruhe gelöscht, sodass eine neue Drehung mit leerem Akkumulator beginnt und frühere Restwerte nicht einfließen. Zusätzlich gibt die Funktion frühestens alle 2 ms eine Nachricht aus. Die genannten Zeitfenster und Schwellen wurden durch Erprobung festgelegt.

Dadurch bleibt die MIDI-Datenrate bei etwa 500 Nachrichten pro Sekunde. Bei moderatem Drehtempo ist jedes Delta nur ein bis zwei Schritte groß, sodass die Automationskurve in Ableton fein ausfällt. Schnelle Drehungen erzeugen größere relative Schrittwerte, die in Codeblock 8 begrenzt werden.

Codeblock 8: Begrenzung der relativen Schrittweite
Quelle: Eigener Code, dual_pot_midi_firmware.ino

```
247   int sendDelta = pendingDelta / REL_DELTA_DIVISOR;
248   if (sendDelta == 0) return;


250   if (sendDelta >  REL_MAX_STEP) sendDelta =  REL_MAX_STEP;
251   if (sendDelta < -REL_MAX_STEP) sendDelta = -REL_MAX_STEP;


253   pendingDelta -= sendDelta * REL_DELTA_DIVISOR;


255   if (!sendRelativeDelta(sendDelta, now)) return;
```

Der zu sendende Schritt ist auf ±63 begrenzt, dem maximalen Wert für eine relative MIDI-Nachricht. Alles, was darüber hinausgeht, bleibt im Akkumulator und wird beim nächsten Aufruf gesendet, wodurch eine schnelle Drehung auf mehrere Nachrichten verteilt wird. Die Funktion `sendRelativeDelta` gibt den begrenzten Schritt zusammen mit einem aktuellen Zeitstempel aus. Zur Einstellung der Schrittgröße ist `REL_DELTA_DIVISOR` auf 1 gesetzt.

6.2.6 Relative MIDI-Codierung

Die relative Bewegung wird als MIDI-CC-Nachricht mit einem Datenwert zwischen 0 und 127 übertragen (MIDI Association, o. D.). Dieser Wert gibt sowohl die Drehrichtung als auch die Schrittgröße an. Positive Bewegungen haben Werte von 1 bis 63, während negative Bewegungen als Werte von 65 bis 127 codiert werden. Der Wert 64 markiert die Grenze zwischen Vorwärts- und Rückwärtsbewegung.

Codeblock 9: Relative MIDI-Codierung im Zweierkomplement
Quelle: Eigener Code, dual_pot_midi_firmware.ino

```
119   static inline uint8_t encodeRelativeTwosComplement(int d) {
120   if (d > 0) return (uint8_t)d;
121   if (d < 0) return (uint8_t)(128 - (-d));
122   return 0;
123     }
```

$$m(d) = \begin{cases} d & \text{wenn } 1 \le d \le 63. \\ 128 - |d| & \text{wenn } -63 \le d \le -1 \\ 0 & \text{wenn } d = 0 \end{cases}$$

Die Sendefunktion ignoriert eine Schrittweite von 0, sodass keine MIDI-Nachrichten ausgehen, solange das Poti nicht gedreht wird. Andernfalls gibt sie den codierten Wert über die Funktion `controlChange` als CC-Nachricht per USB-MIDI aus (Arduino, 2020).

6.3 Raspberry-Pi-Integration

Ein VirtualHere Linux USB Server auf dem Pi erkennt die angeschlossenen Geräte und macht sie über das Netzwerk für den Client-Prozess (s. Abb. 11) auf dem Hostrechner verfügbar (VirtualHere, o. D.). Ein USB-over-IP-Server leitet den USB-Verkehr weiter, statt die Geräte selbst zu übertragen.

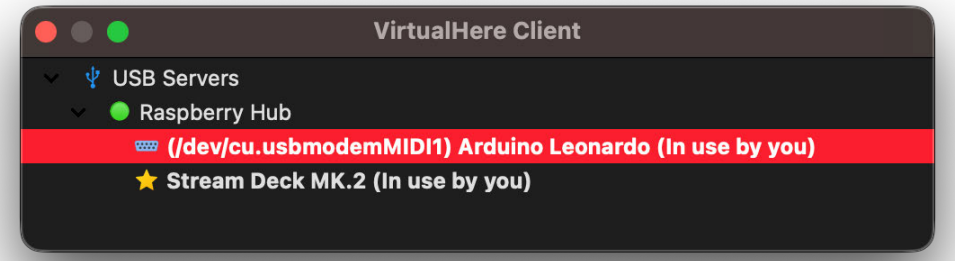


Abbildung 11: VirtualHere-Client.

Hierfür wird der Server als `systemd`-Dienst auf dem Raspberry Pi betrieben, sodass die Weiterleitung beim Hochfahren des Pi automatisch startet. Der Pro Micro erscheint im Audio-MIDI-Setup von macOS als USB-MIDI-Gerät und ist in Ableton Live als MIDI-Eingang verfügbar.

Der USB-over-IP-Verkehr läuft über TCP im lokalen Netzwerk. Ein Powersave-Modus führte an der WLAN-Schnittstelle zu Latenzspitzen, weshalb er auf dem Pi dauerhaft deaktiviert wurde. Dies erfolgt erneut über einen `systemd`-Dienst, der den Befehl `iw dev wlan0 set power_save off` nach jedem Neustart ausführt (Raspberry Pi Stack Exchange, 2019).

Ein separater Prozess liest den Ladezustand des UPS HAT (D) aus und stellt ihn über einen HTTP-Endpunkt im Netzwerk bereit. Er läuft getrennt von VirtualHere, damit der Ladezustand nicht über den latenzkritischen USB-over-IP-Pfad geht.

Die Statuswerte liefert ein INA219-Strom- und Spannungsmessschip auf dem UPS HAT, der vom Server über I2C abgefragt wird. Der Chip muss einmalig konfiguriert werden, wobei die erforderlichen Werte aus dem Waveshare-Beispielcode für das Modul stammen (Waveshare, o. D.). Pro Anfrage schreibt der Server den Kalibrierwert neu und liest anschließend die Messwerte aus.

Codeblock 10: Auslesen des UPS-HAT-Ladezustands
Quelle: Eigener Code, INA219-Auswertung in Anlehnung an Waveshare (o. D.), ableton-live-controller/raspberry-pi/ups_status_server.py

```
87    def read_status(self) -> dict[str, Any]:
90    bus_voltage_v = (self._read_register(REG_BUS_VOLTAGE) >> 3) * 0.004
95       current_a  =  -(self._signed_16(self._read_register(REG_CURRENT))  *
self._current_lsb) / 1000.0
99    percent = clamp((bus_voltage_v - 3.0) / 1.2 * 100.0, 0.0, 100.0)
```

```
100    charging = current_a > 0.05
104    return {
107    "percent": round(percent, 1),
114    "charging": charging,
119      }

125    def get_reader() -> Ina219UpsReader:
126    global reader
127    if reader is None:
128    reader = Ina219UpsReader()
129    return reader
```

Der Server gibt die Werte als JSON aus. Das Stream Deck-Plugin fragt sie in regelmäßigen Abständen über http://streamdeck.local:8765/status ab und stellt auf der Oberfläche dar.

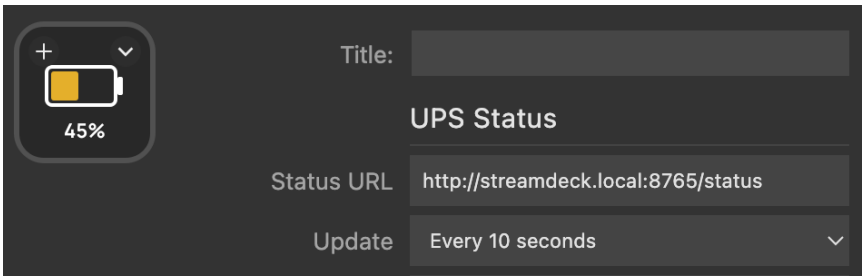


Abbildung 12: Property Inspector der UPS-Statusanzeige.

6.4 Stream Deck-Plugin

Das Stream Deck-Plugin läuft außerhalb von Ableton Live und hat daher keinen direkten Zugriff auf das LOM. Plugin und Remote Script tauschen ihre Steuernachrichten über UDP auf localhost aus, ein verbindungsloses Protokoll, das die Pakete ohne vorherigen Verbindungsaufbau direkt zustellt. Jede Funktion des Stream Decks steckt in einer eigenen Action-Klasse, die `plugin.ts` beim Start registriert.

Codeblock 11: Registrierung der Stream Deck-Actions
Quelle: Eigener Code, src/plugin.ts

```
161    streamDeck.actions.registerAction(new PluginSlotAction());
163    streamDeck.actions.registerAction(new PluginParameterSlot1Action());
176    streamDeck.actions.registerAction(new MidiMapPot1Action());
189    streamDeck.actions.registerAction(new UpsStatusAction());


193    void streamDeck.connect();
```

Tasten und Klassen hängen über eine gemeinsame Action-UUID zusammen, die in der Klasse, in der `manifest.json` und im Profil identisch sind.

Die Action-Dateien in `src/actions/` beschreiben das Verhalten der einzelnen Tasten und die `ableton*.ts`-Dateien bilden die Schnittstelle zum Remote Script. Gemeinsame Zustandsdateien speichern, welche Spuren, Geräte, Parameter und Mapping-Ziele derzeit ausgewählt sind. Daraus erzeugt das Plugin die Tastenbilder.

6.4.1 Kommunikation mit dem Remote Script

Die UDP-Kommunikation mit dem Remote-Script befindet sich in `remoteScriptClient.ts`. Das Plugin und Ableton Live laufen auf demselben Rechner, weswegen `HOST` auf die Loopback-Adresse 127.0.0.1 zeigt.

Codeblock 12: UDP-Ziel des Remote Scripts
Quelle: Eigener Code, src/remoteScriptClient.ts

```
19    const HOST = "127.0.0.1";
20    const PORT = 34344;
21    const EVENT_PORT = 34345;
```

Anfragen schickt das Plugin über den Command-Port 34344 und die Push-Events laufen über den getrennten Event-Port 34345 zurück. Die Trennung erfolgt, damit die Events nicht auf gedrosselte Anfragen warten müssen.

Codeblock 13: Senden eines UDP-Kommandos mit Antwort
Quelle: Eigener Code, src/remoteScriptClient.ts

```
562   function sendUdpRequest(
563    message: string,
564    timeoutMs = 2000,
565   ): Promise<UdpResult> {
568    return new Promise((resolve) => {
569    const client = dgram.createSocket("udp4");

581    client.on("message", (msg) => {
583    finish({ ok: true, response: msg.toString("utf8") });
584    });

590    client.on("listening", () => {
591    const payload = new Uint8Array(Buffer.from(message, "utf8"));
592    client.send(payload, 0, payload.length, PORT, HOST, (err) => {
596    });

599    client.bind();
604    });
605   }
```

Die Funktion `sendUdpRequest` sendet eine Anfrage über einen eigenen Socket und gibt dessen Antwort zurück. Bleibt eine Antwort aus, beendet ein Timeout nach 2000 ms den Versuch, damit das Plugin nicht hängen bleibt. Damit viele gleichzeitige Abfragen den Remote Script nicht überlasten, lässt ein Filter höchstens drei davon zur selben Zeit zu.

Codeblock 14: Senden eines No-Reply-Kommandos
Quelle: Eigener Code, src/remoteScriptClient.ts

```
680     export function sendRemoteScriptCommandNoReply(message: string):
Promise<boolean> {
681     const coalesceKey = noReplyCoalesceKey(message);
685     if (coalesceKey) {
686     pendingNoReplyByKey.set(coalesceKey, message);
687     if (!pendingNoReplyOrder.includes(coalesceKey)) {
688     pendingNoReplyOrder.push(coalesceKey);
689     }
690     scheduleNoReplyQueue();
691     return Promise.resolve(true);
692     }
693     return sendNoReplyNow(message);
694   }
```

Eine schnelle Poti-Bewegung erzeugt in kurzer Zeit viele Werte. Diese Steuerwerte sendet das Plugin ohne Antwort und am Filter vorbei, damit sie nicht verzögern. Bei sehr schneller Bewegung ersetzt ein neuer Wert den noch nicht gesendeten vorherigen Wert, sodass das Plugin keine bereits überholten Zwischenwerte nachschickt.

Ein zentraler Dispatcher nimmt die Anfragen am Command-Port entgegen und leitet jede Anfrage an die zuständige Funktion weiter. Die Antwort geht dabei an die Absenderadresse zurück.

Beim Start meldet das Plugin dem Remote Script mit dem Kommando `eventPort:34345`, wohin die Push-Events gehen sollen. Das Remote Script übernimmt die gemeldete Adresse und sendet dorthin den aktuellen Transport- und Trackstatus. Das Plugin wiederholt diese Meldung alle 15 Sekunden, sodass das Remote Script seine Zieladresse auch nach einem Ableton-Neustart erfasst, ohne dass man es neu laden muss.

Ein Push-Event hält das Tastenbild aktuell, sobald sich ein Wert in Ableton per Maus oder über eine Automation ändert.

Codeblock 15: Empfang von Ereignissen über den Event-Kanal
Quelle: Eigener Code, src/remoteScriptClient.ts

```
350   export function startRemoteScriptEventListener(): void {
351     if (eventSocket) return;
352     const socket = dgram.createSocket("udp4");
355     socket.on("message", (msg) => {
```

```
358    const parsed = JSON.parse(msg.toString("utf8"));
360    event = parsed as Record<string, unknown>;
365    for (const listener of remoteScriptEventListeners) {
366    void Promise.resolve(listener(event)).catch(() => {});
367    }
368    });
380     socket.bind(EVENT_PORT, HOST, () => {
}


390            export    function    subscribeRemoteScriptEvents(listener:
RemoteScriptEventListener): () => void {
391    startRemoteScriptEventListener();
392    remoteScriptEventListeners.add(listener);
393   return () => {
394    remoteScriptEventListeners.delete(listener);
395   };
396   }
```

Da sich ein UDP-Port nicht mehrfach exklusiv binden lässt, gibt es auf Port 34345 nur einen Socket. Dieser Socket wertet jede Nachricht (JSON) aus und verteilt sie an die Action-Klassen, die sich dafür angemeldet haben. So lesen mehrere Tasten dieselben Ereignisse, ohne jeweils einen eigenen Socket zu öffnen.

6.4.2 Action-Klassen und Button-Ereignisse

Der Parameterbutton legt beim Druck sein Ziel als Mapping-Ziel im gemeinsamen Zustand ab und speichert darin den Track, den Device-Pfad und den Parameter. Ein anschließender Druck auf „Map Pot 1“ oder „Map Pot 2“ liest dieses Ziel erneut aus.

Codeblock 16: Parameterbutton speichert ein Mapping-Ziel
Quelle: Eigener Code, src/actions/plugin_parameter.ts

```
930   await setSelectedMidiMappingTarget({
931    kind: "parameter",
932    targetKey: target.targetKey,
933    trackIndex: target.trackIndex,
935    deviceIndexPath: target.deviceIndexPath,
936    parameterIndex: target.parameterIndex0,
942   });
```

Eine Action-Klasse definiert das Verhalten einer Taste. Das SDK ruft sie über eine Methoden auf, sobald die Taste erscheint, gedrückt oder losgelassen wird (Elgato, o. D.-b). Aus der Zeit zwischen `onKeyDown()` und `onKeyUp()` leitet die Klasse ab, ob der Anwender kurz drückt oder lange hält, sodass eine Taste zwei Funktionen auslösen kann.

Codeblock 17: Aufbau einer Action-Klasse
Quelle: Eigener Code, src/actions/track_fx_overview.ts

```
42                  @action({     UUID:     "com.oliver-parvini.ableton-live-
controller.trackfxoverview" })
43                      export     class     TrackFxOverviewAction     extends
SingletonAction<Record<string, never>> {
47          override  async  onWillAppear(ev:  WillAppearEvent<Record<string,
never>>): Promise<void> {
48      const action = ev.action as KeyAction;
50      await setCachedQueuedActionSvg(action, buildIcon(), this.lastImage);
51      }


53       override async onKeyDown(ev: KeyDownEvent<Record<string, never>>):
Promise<void> {
54      const action = ev.action as KeyAction;
64     let trackIndex = getTrackBrowserSourceTrackIndex(action.device.id);
65      if (typeof trackIndex !== "number") {
75      await action.showAlert();
76      return;
77      }
78      setFocusedTrack(trackIndex);
80              await   streamDeck.profiles.switchToProfile(action.device.id,
TRACK_FX_OVERVIEW_PROFILE);
85      }
86     }
```

Die Action-Klasse erbt von `SingletonAction`, weshalb das SDK nur ein einziges Objekt anlegt. Dieses bedient alle Tasten desselben Typs gemeinsam, statt für jede Taste eine eigene Instanz zu erzeugen. Beim Tastendruck ermittelt es, für welche Spur die gedrückte Taste steht. Diese Zuordnung hat das Plugin beim Öffnen des Track-Browsers hinterlegt. Fehlt sie, bleibt der Anwender auf der Seite und erhält eine Warnung, andernfalls wechselt das Plugin zur FX-Übersicht der Spur.

Dasselbe Prinzip nutzen auch die Mapping-Buttons, bei denen ein kurzer Druck das ausgewählte Ziel auf ein Poti mappt und ein langer Druck den Fine-Mode des Potis umschaltet. So erfüllt eine Taste zwei Funktionen, ohne dass die begrenzte Tastenzahl ein eigenes Bedienelement für den Fine-Mode kosten würde.

Codeblock 18: Kurz- und Langdruck beim Mapping-Button
Quelle: Eigener Code, src/actions/midi_map_buttons.ts

```
325       override  async  onKeyUp(ev:  KeyUpEvent<Record<string,  never>>):
Promise<void> {
330       const  startedAt  =  this.keyDownStartedAt.get(action.id)  ??
Date.now();
```

```
332     if (Date.now() - startedAt >= LONG_PRESS_MS) {
333     await this.toggleFineMode(action);
334     return;
335     }

452     const selection = getSelectedMidiMappingTarget();
457     const ok = await this.mapSelection(selection);
}
```

Die Druckdauer ergibt sich aus der Differenz der Zeitstempel von Drücken und dem Loslassen. Die in der Klasse hinterlegte Schwelle liegt bei 500 ms. Ab dieser Druckdauer gilt der Druck als lang und schaltet den Fine-Mode im Remote Script um.

6.4.3 Visualisierung

Das Plugin erzeugt die Tastenbilder im Code als Scalable Vector Graphic (SVG) und macht damit die aktuelle Funktion einer Taste erkennbar (N2). Fertige Bilddateien sind dafür nicht nötig.

Eine gemeinsame Darstellungsschicht setzt jedes Tastenbild aus Panel, Statusbalken, Kopfzeile, Name, Wert und Poti-Indikatoren zusammen (s. Abb. 13). Das Panel übernimmt die Farbe des Ableton-Tracks. Die Textfarbe richtet sich nach der Helligkeit der Trackfarbe, damit der Text auf jedem Button gut lesbar bleibt.

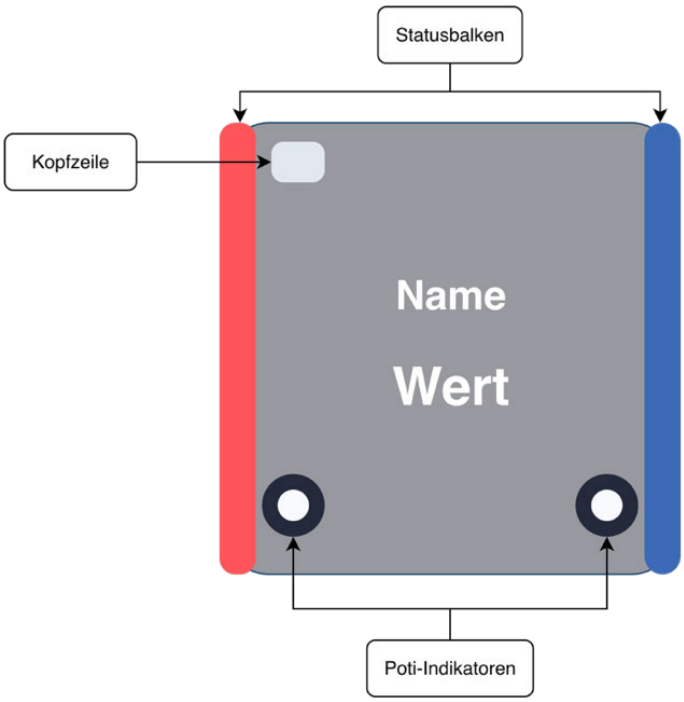


Abbildung 13: Aufbau eines Tastenbilds.

Im drahtlosen Betrieb überträgt VirtualHere die Bilder verzögert. Das Plugin sammelt sie deshalb, speichert die fertigen SVGs zwischen und sendet sie über eine Queue.

Codeblock 19: Queue für Stream Deck-Bildupdates
Quelle: Eigener Code, src/streamDeckImage.ts

```
122   export function setQueuedActionImage(
123    action: KeyAction<any>,
124    image: string,
125    priority = true,
126   ): Promise<void> {
141    pendingByAction.set(action.id, { action, image, priority });
142    queueImageAction(action.id, priority);
144    scheduleImageQueue();
147    resolve();
149   }
```

`setQueuedActionImage` speichert pro Taste nur das neueste Bild. Liegt für dieselbe Taste schon eines in der Queue, ersetzt das neue es, bevor das alte überhaupt gesendet wird.

Nach dem Einreihen meldet die Funktion das Bild sofort als erledigt, sodass der Buttonhandler nicht warten muss. Das eigentliche Senden erfolgt über einen Timer, weil die Bildübertragung über VirtualHere im WLAN spürbar lange dauert und ein Tastendruck sonst für diese Dauer blockieren würde.

Benutzeraktionen werden vorrangig mit einem schnellen Takt von rund 2 ms verarbeitet, sodass eine gedrückte Taste sofort reagiert. Hintergrundaktualisierungen laufen mit einem langsameren Takt von rund 22 ms. Innerhalb einer Prioritätsstufe bleibt die Reihenfolge des Eintreffens erhalten, sonst würden bei schnellem Tippen früher eingereihte Bildaktualisierungen nicht mehr gesendet werden.

6.4.4 Seitenstruktur

Das Skript `build-profiles.mjs` erzeugt die einzelnen Seiten bei jedem Build. Es bindet die insgesamt 29 Profile als `.sdProfile`-Bundles zusammen mit dem Manifest in das Plugin ein.

Codeblock 20: Erzeugung einer Profilseite
Quelle: Eigener Code, scripts/build-profiles.mjs

```
111   function action(uuid, name, settings = {}) {
 return { Name: name, Settings: settings, State: 0, UUID: uuid, States: [
/*... */ ] };
}


325   function buildMainOverviewActions() {
329    return {
331                    "1,0":     action("com.oliver-parvini.ableton-live-
controller.transportplaystop", "Play / Stop"),
332                    "2,0":     action("com.oliver-parvini.ableton-live-
controller.transportrecord", "Record"),
```

```
340                     "0,1":      action("com.oliver-parvini.ableton-live-
controller.pluginslot", "Track Plugin Slot", { trackIndex: 1, deviceIndex: 1
}),
356                     "4,1":      action("com.oliver-parvini.ableton-live-
controller.trackselect", "Track (Select)", { trackIndex: 1 }),
 //... weitere Tasten
 };
}


778          writeProfile("profiles/main-overview",    "Ableton    Overview",
buildMainOverviewActions());
```

Jede `build`-Funktion legt die Tasten einer Seite anhand ihrer Position im Tastenraster fest und übergibt Track- und Geräteindex an die Action. Viele Seiten gleichen sich, etwa die mehrseitigen Track-, Mixer- und Browser-Ansichten. Eine Schleife erzeugt diese Varianten automatisch, sodass sie untereinander konsistent bleiben und nicht einzeln im Stream Deck-Editor angelegt werden müssen.

Das Plugin gliedert sich in mehrere Seiten. Welche davon erscheint, hängt von der gewählten Funktion und der Dauer des Tastendrucks ab (s. Abb. 14).

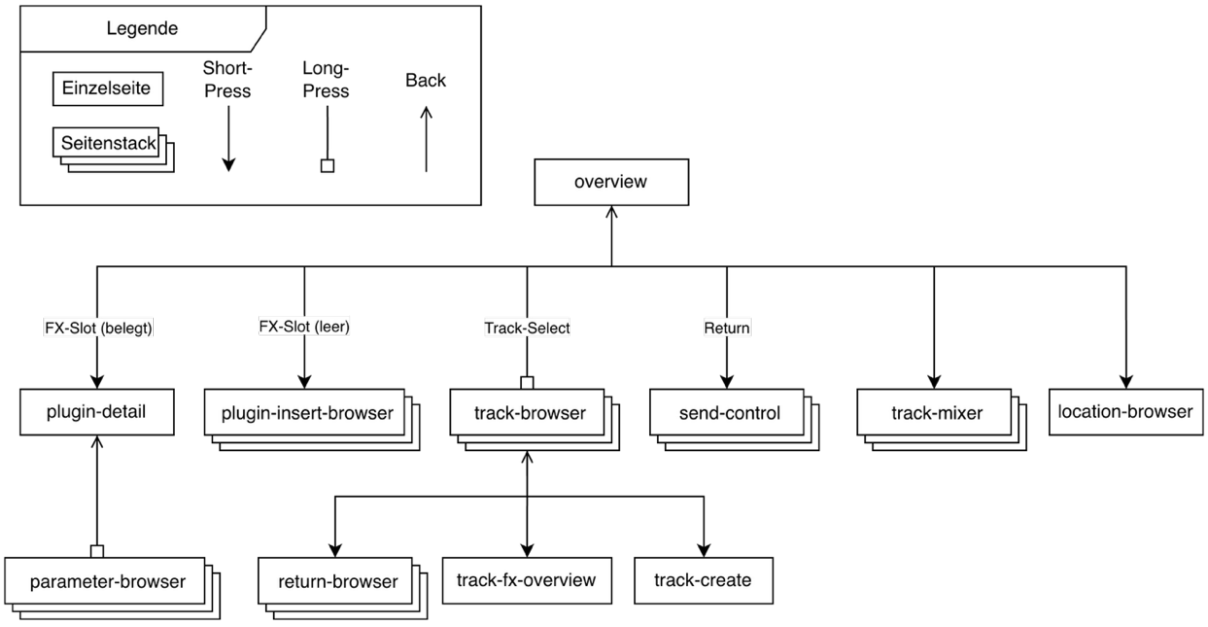


Abbildung 14: Seitenstruktur des Stream Deck-Plugins.

Die **Overview** ist die Einstiegsseite, deren obere Reihe den Ladestatus und die Transportfunktionen zeigt und die Navigation zu den anderen Seiten ermöglicht. Darunter zeigen zwei Reihen je eine ausgewählte Spur mit Name, Trackfarbe und ihren ersten vier Plugin-Slots.

Abbildung 15: Overview-Seite.

Die **Plugin-Detail-Seite** wird über einen belegten FX-Slot geöffnet und zeigt sechs konfigurierbare Parameter des ausgewählten Plugins, die über Map Pot 1 oder Map Pot 2 einem Poti zugewiesen werden. Toggle-Parameter (z. B. an/aus) werden anders dargestellt als kontinuierliche Werte. Die Seite umfasst zusätzlich Transportfunktionen, Bypass und Solo/Arm sowie einen Button zum Öffnen und Schließen des Plugin-Fensters.

Abbildung 16: Plugin-Detail-Seite.

Durch langes Halten eines Parameters öffnet man den **Parameter-Browser**. Dieser listet alle Parameter des ausgewählten Devices auf mehreren Seiten auf. Aus dieser Liste kann dann der Parameter ausgewählt werden, der den zuvor lange gehaltenen Parameter ersetzt.

Ein leerer Slot öffnet den **Plugin-Insert-Browser**, dessen Property Inspector die installierten Plugins in den jeweiligen Kategorien anzeigt. Ein Suchfeld grenzt die Liste ein. Der Tastendruck lädt das gewählte Device an die gemerkte Track- und Slot-Position. Der Profilwechsel erfolgt dabei sofort, während der Ladevorgang im Hintergrund weiterläuft.

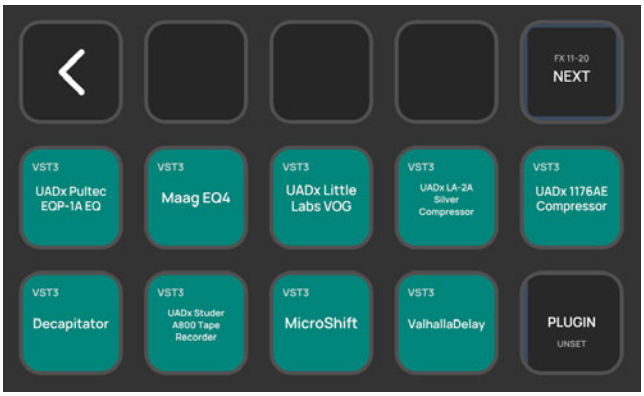


Abbildung 17: Plugin-Insert-Browser.

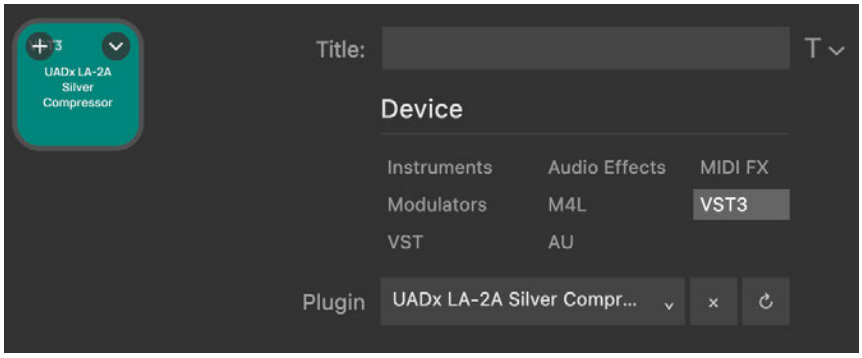


Abbildung 18: Property Inspector eines Plugin-Inserts.

Der **Track-Browser** listet die Tracks der geöffneten Session seitenweise auf, um sie mit den Tracks in der Overview auszutauschen. Von dort sind die anderen Seiten, wie Return-Tracks, das Anlegen neuer Spuren sowie die Track-FX-Overview erreichbar.

Abbildung 19: Track-Browser-Seite.

Auf der **Track-FX-Overview-Seite** werden für eine einzelne Spur bis zu 10 Plugin-Slots angezeigt, die als Erweiterung zu den nur vier auf der Overview dienen.

Über vorab konfigurierte Presets lassen sich über die **Track-Create-Seite** Audio-, MIDI- und Return-Spuren erstellen. Jedes Preset wird über den Property Inspector konfiguriert, in dem sich Spurtyp, Name, Eingang und Monitoring festlegen lassen.

Abbildung 20: Track-Create-Seite.

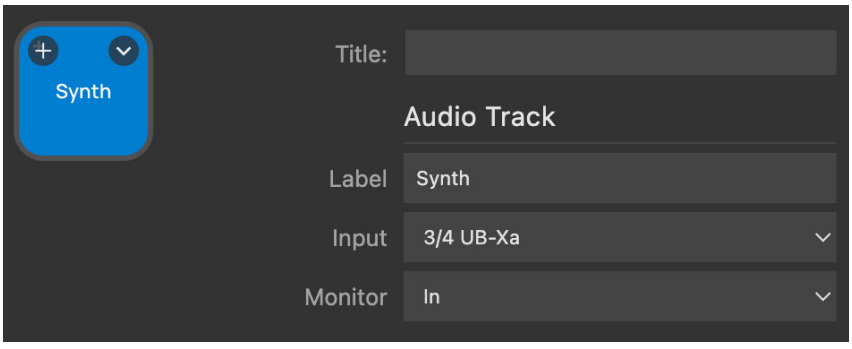


Abbildung 21: Property Inspector eines Track-Create-Presets.

Auf der **Send-Control-Seite** repräsentiert jede Taste eine Quellspur und zeigt deren aktuellen Send-Pegel an. Mehrere Send-Tasten können gemeinsam ausgewählt und als Gruppe einem Potentiometer zugewiesen werden.

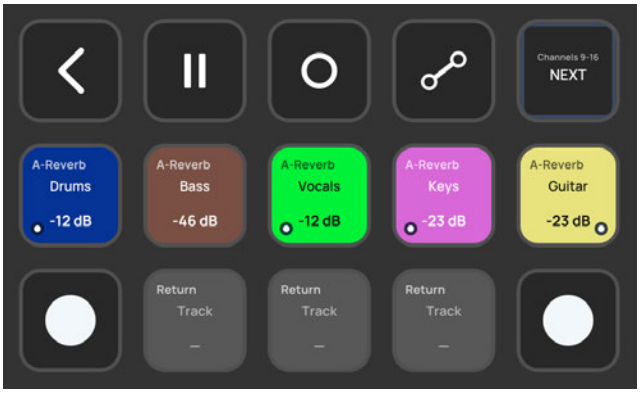


Abbildung 22: Send-Control-Seite.

Die **Track-Mixer-Seite** zeigt die Mixer-Parameter (Lautstärke, Pan) und ermöglicht deren Steuerung über die Potentiometer. Anders als sonst legt hier schon die Kanalauswahl das Mapping fest. Pot 1 steuert dauerhaft die Lautstärke und Pot 2 den Pan. Ausgewählt wird nur, welche Tracks betroffen sind. Die regulären Tracks sind dabei seitenweise blätterbar. Eine feste dritte Reihe enthält auf jeder Seite den Master-Track und bis zu vier Return-Tracks.

Abbildung 23: Track-Mixer-Seite.

Der **Location-Browser** verwaltet die Marker des Arrangements. Vorhandene Marker können angesprungen, neue gesetzt, vorhandene gelöscht und Loop-Bereiche zwischen zwei Positionen definiert werden. Die Einträge sind nach ihrer Position in der Timeline sortiert, sodass ein nachträglich gesetzter Marker unabhängig von der Erstellungsreihenfolge an der passenden Stelle erscheint. Die Wiedergabeposition kann über den Scrub-Button bewegt werden.

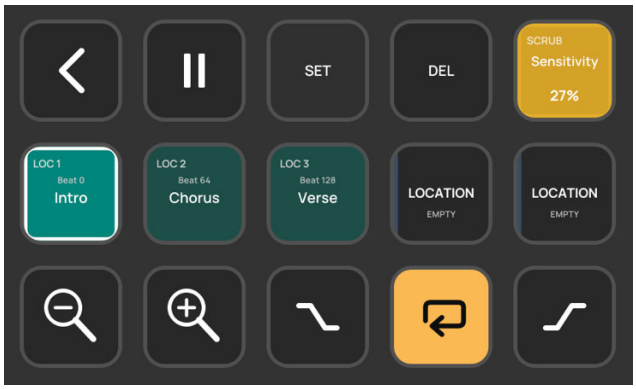


Abbildung 24: Location-Browser-Seite.

6.5 Ableton Remote Script

Das Remote Script ist in Python geschrieben (Ableton, o. D.-c) und besitzt eine eigene Laufzeitlogik. Welche Operation eine eingehende Nachricht auslöst, hängt vom aktuellen Zustand des Live-Sets ab. Einen vergleichbaren Ansatz verfolgt Jones (2023) mit AbletonOSC, das über eine API auf das LOM zugreift.

Ein Dispatcher liest den Kommandonamen am Anfang der Nachricht und leitet den Rest an die zuständige Handler-Funktion weiter. Welche das ist, bestimmt eine Tabelle, die jedem Kommandonamen seinen Handler zuordnet. Trifft eine unbekannte Nachricht ein, antwortet das Script mit einer Fehlermeldung und verwirft sie.

Codeblock 21: Dispatcher für UDP-Kommandos
Quelle: Eigener Code, AbletonControllerRemoteScript.py

```
5220  def _handle_udp_message(self, msg):
5232   command, _, payload = msg.partition(":")
5233   handler = self._command_handlers.get(command)
5234   if handler is None:
5235   self._send_udp(f"ERROR:Unknown command {msg}")
5236   return
5237   handler(payload)
```

Um ein Mapping zu setzen, codiert das Plugin Track, Device und Parameter in ein Kommando.

Codeblock 22: Mapping eines Parameters über Device-Pfad
Quelle: Eigener Code, src/abletonParameter.ts

```
200   const command = path.length > 1
947      ? `linkParamPath:${JSON.stringify({ pot: midiSlot, trackIndex: t, deviceIndexPath: path, parameterIndex: p })}`
948    : `linkParam:${midiSlot}:${t}:${d}:${p}`;
```

Für einzelne Devices genügt ein Kommando, dessen Werte durch Doppelpunkte getrennt sind. Verschachtelte Rack-Pfade lassen sich so nicht eindeutig abbilden, deshalb sendet das Plugin sie als JSON über das Kommando `linkParamPath`. Der Pfad enthält für jede Rack-Ebene einen Index und beginnt beim Device der obersten Ebene. Ein Device ohne Rack ergibt damit einen Pfad mit einem Eintrag, ein verschachteltes einen längeren, woran das Plugin die Kommandoform erkennt.

Codeblock 23: Verarbeitung des linkParamPath-Kommandos
Quelle: Eigener Code, AbletonControllerRemoteScript.py

```
6690  def _cmd_link_param_path(self, payload):
6692   data = json.loads(payload.strip())
6693   pot = int(data.get("pot"))
6694   track_idx = int(data.get("trackIndex"))
6695                                          device_index_path             = self._normalize_device_index_path(data.get("deviceIndexPath"))
6696   param_idx = int(data.get("parameterIndex"))


6703    if self._set_linked_target_path(pot, track_idx, device_index_path, param_idx):
6704   self._send_udp("OK")
```

Der Handler liest aus dem JSON-Objekt das Poti, den Track, den Device-Pfad und den Parameterindex. Den Pfad bringt das Script zunächst in eine geprüfte Folge von Indizes, die je einen Schritt durch die Rack-Struktur beschreiben, und löst sie auf einen vorhandenen Parameter auf. Nur wenn das gelingt, bestätigt es. Andernfalls geht eine Fehlermeldung zurück, dass das Ziel nicht gefunden wurde. So weiß die Plugin-Seite, ob eine Verknüpfung für Pot 1 oder Pot 2 steht. Ungültiges JSON oder ein leerer Pfad führen ebenfalls zu einer Fehlermeldung und verhindern eine halb gesetzte Verknüpfung.

Das Framework von Ableton Live ruft das Remote Script regelmäßig auf dem Haupt-Thread auf. Bei jedem dieser Aufrufe wendet es gepufferte Parameteränderungen an, sendet die gedrosselten Push-Events und prüft den UDP-Socket. Die Prüfung wartet nicht auf die Daten und blockiert den Tick daher nicht.

Codeblock 24: update_display-Tick als getakteter Arbeitsschritt
Quelle: Eigener Code, AbletonControllerRemoteScript.py

```
7072  def update_display(self):
7073   super().update_display()
7078   midi_active = (
7079           (time.perf_counter() - self._last_midi_activity_at) * 1000.0
7080           < MIDI_ACTIVITY_UDP_THROTTLE_MS
7081   )
7083    self._flush_pending_midi_changes(False)
7087      self._flush_pending_track_mixer_value_events()
7090      if not midi_active:
7091          self._push_transport_state_event_if_changed(False)
7092          self._push_tracks_event_if_changed(False)
        else UDP_MAX_PACKETS_PER_TICK
    #... eingehende UDP-Kommandos bis zum Per-Tick-Budget verarbeiten...
```

Das Flag `midi_active` markiert eine laufende Poti-Bewegung und bleibt aktiv, solange die letzte Bewegung weniger als 120 ms zurückliegt. In dieser Phase überspringt das Script die regelmäßige Abfrage von Transport- und Track-Zustand und verarbeitet vorrangig die eingehenden Steuerwerte. Nach Ablauf des Fensters wird die Zustandsabfrage wieder aufgenommen.

Damit nicht zu viele Poti-Daten auf einmal den Haupt-Thread blockieren, begrenzt das Script die Verarbeitung pro Tick auf höchstens 16 Pakete oder 1,5 ms. Während aktiver Poti-Bewegung senkt es dieses Budget auf 4 Pakete oder 1,0 ms, sodass Bedienaktionen weiterhin erfolgen können.

6.6 Mapping

Die Mapping-Logik trennt die Bewegung der Potentiometer von den Ableton-Zielen. Die Firmware kennt das gesteuerte Ableton-Ziel nicht und sendet nur relative Schritte. Die Zuordnung zu einem Parameter erfolgt erst auf der Host-Seite.

Codeblock 25: Weiterleitung der Poti-CCs an das Remote Script
Quelle: Eigener Code, AbletonControllerRemoteScript.py

```
271   def build_midi_map(self, midi_map_handle):
275       script_handle = self._c_instance.handle()
276       for channel in range(16):
277           Live.MidiMap.forward_midi_cc(script_handle, midi_map_handle,
channel, POT1_CC)
278           Live.MidiMap.forward_midi_cc(script_handle, midi_map_handle,
channel, POT2_CC)
```

Ableton ruft die Methode `build_midi_map` auf, wenn Live die MIDI-Zuordnung des Scripts aufbaut, etwa beim Start des Sets. Der übergebene Pfad `midi_map_handle` verweist auf diese Zuordnung. Das Script meldet die beiden Poti-CCs für alle 16 MIDI-Kanäle zur Weiterleitung an, damit sie unabhängig vom eingestellten Kanal des Controllers greifen. Ohne diese Anmeldung blieben die Poti-CCs im normalen MIDI-Pfad von Live, und das Remote Script könnte sie nicht für sein Mapping nutzen.

Dreht der Anwender ein Poti, ruft Ableton das Script mit der CC-Nachricht auf und übergibt ihm den relativen Schritt.

Umgekehrt muss das Plugin über externe Änderungen in Ableton informiert werden, etwa wenn der Anwender einen Track umbenennt oder ein Device hinzufügt. Dafür fragt das Remote Script den Zustand wiederholt ab. Da Live das Script regelmäßig aufruft, bildet es höchstens alle 150 ms eine kompakte Beschreibung der Tracks, die deren Anzahl, Name, Farbe, Arm-, Mute- und Solo-Status sowie Gruppen- und Fold-Zustand erfasst. Diese vergleicht es mit der vorherigen und meldet sie bei einer Abweichung als Push-Event an das Plugin.

Für die laufenden Parameterwerte nutzt das Remote Script dagegen die direkte Benachrichtigung von Live, die bei jeder Wertänderung sofort meldet. Es richtet sie gezielt nur für die gerade sichtbaren Parameter-, Send- und Mixer-Kacheln ein und entfernt sie beim Seitenwechsel wieder. So folgt deren Anzeige auch dann, wenn ein Wert in Ableton per Maus oder Automation verändert wird, ohne dass ein Potentiometer bewegt wurde.

Das Script liest das gespeicherte Ziel des jeweiligen Potis und entscheidet erst dann, ob die Bewegung einen Device-Parameter, einen Send, einen Mixerwert oder den Playhead steuert.

Codeblock 26: Empfang der CC-Nachricht
Quelle: Eigener Code, AbletonControllerRemoteScript.py

```
5194  def receive_midi(self, midi_bytes):
5202   status_byte = int(midi_bytes[0]) & 0xFF
5203   status = status_byte & 0xF0
5204      if status != 0xB0:
5205          return
```

```
5207        cc = int(midi_bytes[1]) & 0x7F
5208        value = int(midi_bytes[2]) & 0x7F
5209        pot = self._pot_for_cc(cc)
5210        if pot is None:
5211            return

5213        self._process_pot_midi(pot, value)
```

Der Empfang läuft im Haupt-Thread von Live und hält sich deshalb kurz. Das Script verwirft eingehende Bytes, deren Statusbyte keine CC-Nachricht anzeigt (`0xB0`), und behält von den übrigen nur die der beiden Poti-CCs (CC 14 und CC 15). Die so geprüfte Nachricht reicht es mit `pot` und `value` an die eigentliche Verarbeitung weiter.

Codeblock 27: Auflösen des Mapping-Ziels für den Poti
Quelle: Eigener Code, AbletonControllerRemoteScript.py

```
5036  target = self._normalize_link_target(self._linked_targets.get(pot))
5037  if target is None:
5038      return

5069  parameter = self._target_parameter_for_pot(pot, target)
```

Da aus der CC-Nachricht nur bekannt ist, dass sich Poti 1 oder 2 bewegt hat, schlägt das Script das für diesen Poti gespeicherte Ziel nach. Ist kein Ziel hinterlegt, verwirft es die Bewegung. Andernfalls löst es das Ziel in das konkrete Ableton-Objekt auf und speichert sich dieses je Poti, damit die Suche bei der nächsten Drehung entfällt. Beim Mapping verhindert ein Pickup-Mechanismus Wertsprünge (N3). Da das Poti nur relative Schritte sendet, bleibt der Ableton-Parameter beim Mapping zunächst auf seinem aktuellen Wert und wird erst durch die der nächste Drehbewegung weiter verändert.

Dazu führt das Remote Script je Potentiometer einen internen Wert im Bereich 0 bis 1023, den es beim Mapping auf den aktuellen Wert des Zielparameters setzt. Jeden eingehenden Schritt skaliert das Script mit dem Faktor (0,625 bzw. 0,25 im Fine Mode) und addiert ihn auf diesen Wert. Den so entstandenen Wert bildet es auf den tatsächlichen Wertebereich des Parameters ab und meldet ihn als Push-Event zurück. Der interne Wert sammelt die Drehschritte und liegt bewusst als Kommazahl vor, damit sich auch Bewegungen unterhalb eines ganzen MIDI-Schritts aufsummieren. Die in Ableton aufgezeichnete Automationskurve bewegt sich dadurch in Bruchteilen des Bereichs 0 bis 1023 statt in ganzen Stufen und zeigt weniger Quantisierungsstufen.

6.7 Gehäuse

Das Gehäuse wurde mittels 3D-Druck mit dem an der HAW verfügbaren CraftBot-3 gefertigt. Die Konstruktion wurde in Fusion 360 (Modelldatei in der digitalen Anlage, 02_CAD) erstellt und in OrcaSlicer in ein druckfähiges Format umgewandelt. Für das Filament wurde ein transluzentes weißes PLA gewählt, wodurch die Status-LEDs der Hardware sichtbar bleiben, ohne dass für jede LED ein eigenes Sichtfenster erforderlich ist.

Als Ausgangsbasis diente ein auf Printables veröffentlichtes Gehäusemodell für ein Stream Deck XL-Companion-Pi-Setup (Atelier Du Tech, 2024), das an die Abmessungen und Ausschnitte des Systems angepasst wurde.

Abbildung 25: Stream Deck XL-Companion-Pi-Setup.
Quelle: Atelier Du Tech (2024).

Für das Stream Deck Module, den Raspberry Pi, die Potentiometer, den Pro Micro und den UPS HAT (D) liegen CAD-Modelle der Hersteller und der GrabCAD-Community online vor. Sie erleichtern die Konstruktion, da sich die Bauteile im Gehäuseentwurf maßgenau platzieren lassen. Die verwendeten Modelle sind in der digitalen Anlage dokumentiert (02_CAD/3D-Modellquellen.md).

Der wesentliche limitierende Faktor für das Gehäuse war die Druckzeit, da der Drucker aus Brandschutzgründen nicht über Nacht betrieben werden durfte. Ein einteiliger Druck hätte zudem mehr Stützmaterial und ein höheres Risiko für Fehldrucke mit sich gebracht. Das Gehäuse wurde daher modularisiert.

Tabelle 3: Bauteiltabelle

Pos.	Bauteil	Baugruppe
1	Bodenplatte	Gehäuse
2	Seitenwand links	Gehäuse
3	Seitenwand rechts	Gehäuse
4	Frontblende	Gehäuse
5	Rückwand	Gehäuse
6	Gehäusedeckel	Gehäuse
7	Bedienblende	Gehäuse
8	Klemmrahmen	Gehäuse
9	Pro Micro ATmega32U4	Elektronik
10	Lochrasterplatine	Elektronik
11	RV112FF Potentiometer 1	Elektronik
12	RV112FF Potentiometer 2	Elektronik
13	UPS HAT (D)	Elektronik
14	Raspberry Pi 3 B+	Elektronik
15	Stream Deck Module (15)	Bedienelement
16	Drehknopf 1	Bedienelement
17	Drehknopf 2	Bedienelement

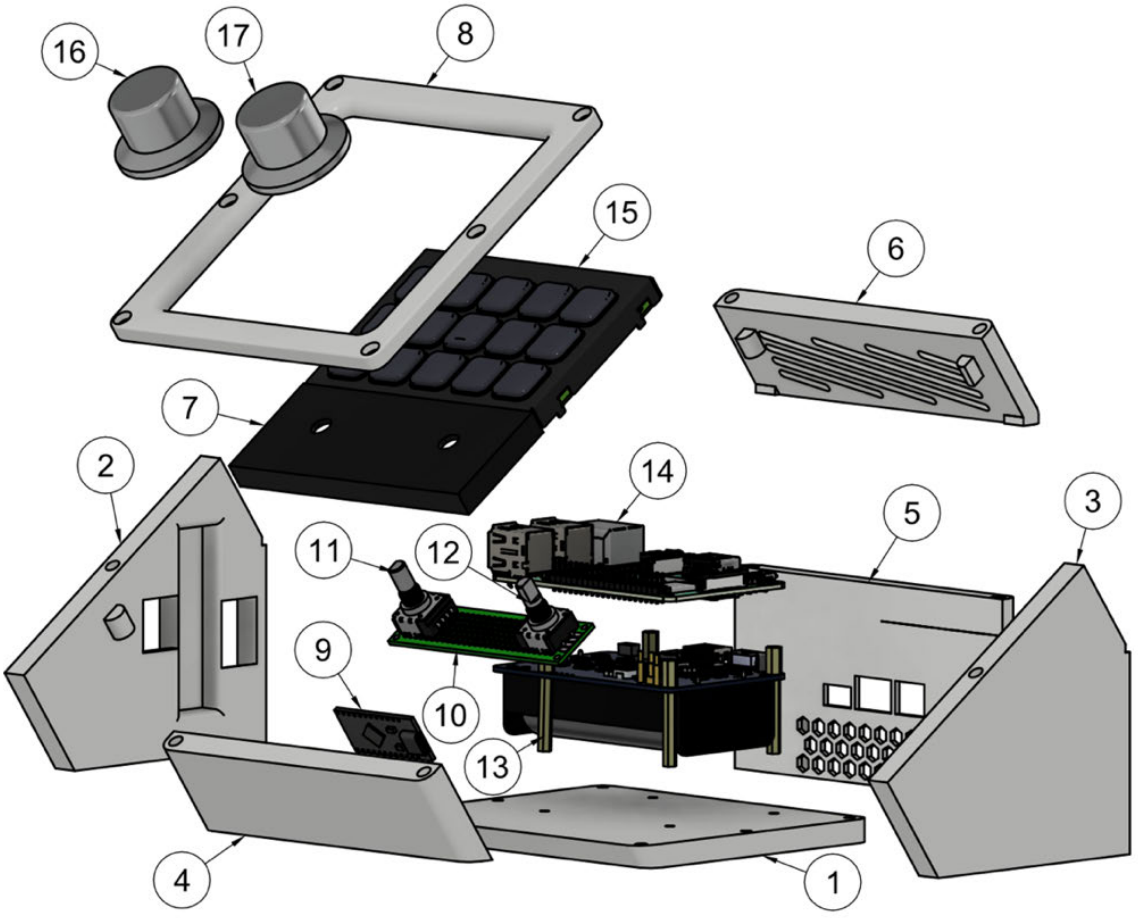


Abbildung 26: Explosionsansicht der Gehäusebauteile.

6.7.1 Anpassungen

Die Frontfläche wurde so angepasst, dass das Stream Deck Module bündig eingesetzt und mittels des Klemmrahmens (8) befestigt werden kann. Die Bedienblende (7) erhielt zwei Öffnungen für die Potentiometer. Ihr Abstand wurde so gewählt, dass sich die Daumen bei beidhändiger Bedienung nicht gegenseitig behindern.

Für den Pro Micro wurde ein passgenauer Einsteckblock an der Frontblende (4) konstruiert, sodass dieser ohne weitere Befestigung hält und sich für Anpassungen an der Verkabelung jederzeit entnehmen lässt. Er ist dabei bewusst nach außen gerichtet, damit seine Status-LED beim Hochladen der Firmware und beim Drehen der Potentiometer gut sichtbar bleibt.

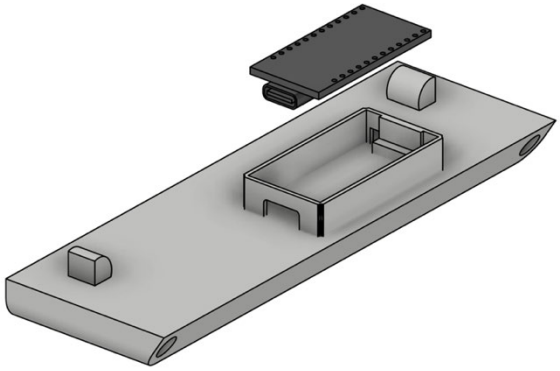


Abbildung 27: Einsteckblock für den Pro Micro.

Es wurden Aussparungen an der Rückwand (5) für die Ladeanschlüsse und den Ein-/Ausschalter des UPS HAT sowie an der linken Seitenwand (2) für die USB-Anschlüsse und den Ethernet-Port des Raspberry Pi vorgesehen.

Die Lüftungsschlitze im Gehäusedeckel (6) verbessern den Luftaustausch im Gehäuse, da PLA bereits ab seiner Glasübergangstemperatur von etwa 65 °C erweicht und an Steifigkeit verliert (Slavković et al., 2024).

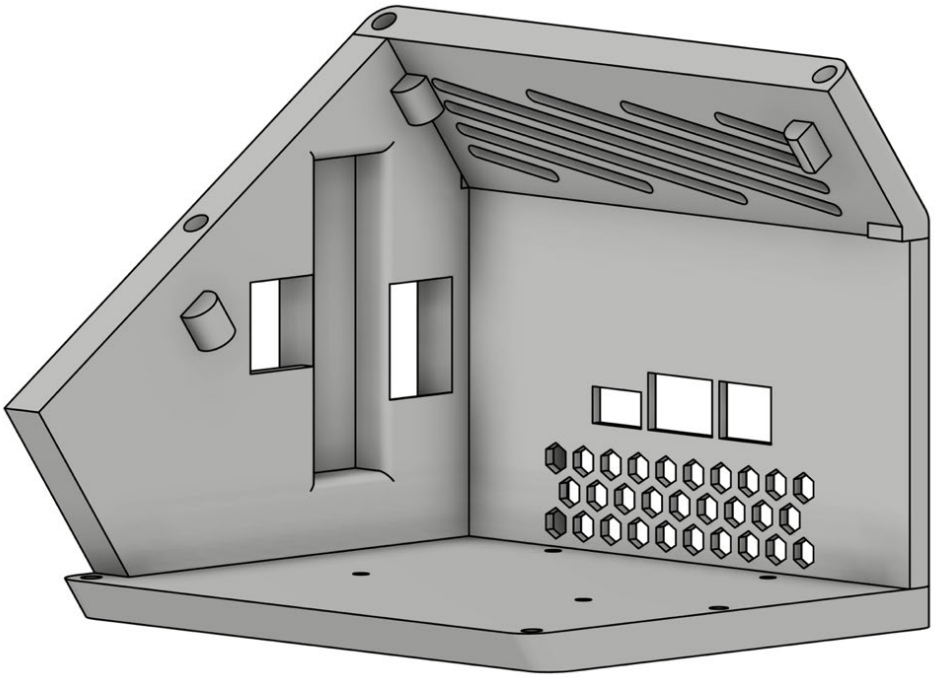


Abbildung 28: Innenansicht des Gehäuses.

6.7.2 Montage

Um die Gehäuseteile zu verbinden, kommen Gewindeeinsätze zum Einsatz, die durch Wärme in das gedruckte Material eingelassen werden. Solche Heat-Set-Inserts ermöglichen wiederholtes Verschrauben und halten besser als direkt in Kunststoff geschnittene Gewinde. Verwendet werden Einsätze für M2,5 mit einer Länge von 8 mm. Vor der Montage wurden Bohrungsdurchmesser, Einschmelztiefe und Sitz an einer Testplatte abgestimmt, da Bohrungen im 3D-Druck je nach Drucker, Material und Orientierung vom CAD-Maß abweichen können (CNC Kitchen, 2021).

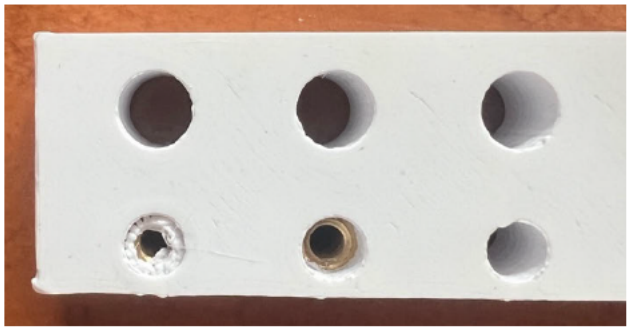


Abbildung 29: Testplatte der Heat-Set-Inserts.

Abbildung 29 zeigt diese Testplatte mit den erprobten Bohrungen. Bei zu kleinem Bohrungsdurchmesser staut sich das verdrängte Material am gegenüberliegenden Ende auf, während ein passender Durchmesser den Einsatz bündig aufnimmt. Verschraubt wird mit Linsenkopfschrauben M2,5×11 (Schlitz), für die Befestigung des Raspberry Pi an der Bodenplatte (1) mit M2×12 (Schlitz).

Auf die Achsen der Potentiometer sind zwei massive Aluminium-Drehknöpfe (16, 17) mit einem Durchmesser von 38 mm montiert (HiFi-Lab, o. D.). Wegen des großen Durchmessers legt der Daumen an der Knopfkante einen längeren Weg pro Drehung zurück, was feine, langsame Wertänderungen erleichtert, wie sie beim Schreiben von Automationen benötigt werden. Die Auswahl orientiert sich am Drehregler des nOb Control.

Die interne Verkabelung erfolgt über zwei Flachbandkabel. Beide Kabel haben auf der Geräteseite einen geraden USB-2.0-Typ-C-Stecker und auf der Pi-Seite einen gewinkelten USB-2.0-Typ-A-Stecker. Der gewinkelte Stecker hält die Kabel nah am Gehäuse und vermeidet Knickstellen.

Auf der Unterseite der Bodenplatte (1) sind zusätzlich selbstklebende Gummifüße angebracht, die ein Verrutschen des Controllers im Betrieb verhindern.

Abbildung 30: Der fertige Controller.

7 Evaluation

Die technische Evaluation prüft die Reaktionszeit des Systems (N5) und die Stabilität der drahtlosen Verbindung (N6). Bei der Nutzungsevaluation werden die Verständlichkeit und die Erlernbarkeit der Bedienung (N1, N2) betrachtet.

7.1 Technische Evaluation

Für eine musikalische Steuerung gilt als Richtwert eine Latenz von 10 ms mit einer Schwankung von unter 1 ms (Wessel & Wright, 2001). Für kontinuierliche Klangsteuerung werden höhere Werte toleriert, in der Regel 20 bis 30 ms (Lago & Kon, 2004; Mäki-Patola & Hämäläinen, 2004). Da die Potentiometer kontinuierlich bedient werden, ist die wahrnehmungsrelevante Schwelle eher am oberen Rand dieses Bereichs anzusetzen. Als Orientierungswert wird für sie dennoch ein Wert im niedrigen zweistelligen Millisekundenbereich (bis etwa 20 ms) gewählt. Der Stream Deck-Pfad ist vom Verarbeitungstakt von Ableton abhängig und wird separat betrachtet.

Die Schwierigkeit bei der Untersuchung des vorliegenden Systems bestand darin, den Eingabezeitpunkt verlässlich zu erfassen. Eine Bedienhandlung wird erst registriert, wenn ihre Verarbeitung den Hostrechner erreicht hat. Beim Stream Deck verläuft dieser Weg über den Raspberry Pi, VirtualHere, das Stream Deck-Plugin und das Remote Script. Bei den Potentiometern verläuft dieser zusätzlich über den Pro Micro. Ein im Code angelegter Zeitstempel liegt bereits innerhalb der Verarbeitungskette und markiert daher nicht den Beginn der Eingabe.

Als Vorbild für den hier genutzten Ansatz dient das Messwerkzeug OSRTT. Dabei handelt es sich um ein Latenztestgerät für Maus und Tastatur, bei dem eine leitende Folie auf die Taste geklebt und mit einer Spannungsquelle verbunden wird. Eine Berührung der Folie mit einem leitenden Kontakt löst zugleich den Tastendruck aus und startet über den geschlossenen Stromkreis die Zeitmessung. Der Endpunkt wird über einen externen Lichtsensor oder ein Mikrofon erfasst (OSRTT, o. D.). Start- und Endpunkt liegen außerhalb der zu messenden Kette, weshalb im Code kein Zeitstempel erforderlich ist und die gemessene Verzögerung den gesamten Weg von der Eingabe bis zur Reaktion umfasst. Dieses Prinzip wird für den vorliegenden Controller übernommen und an dessen Signalweg angepasst.

7.1.1 Messadapter

Für den Stream Deck-Pfad wurde ein externer Arduino Nano als Messadapter aufgebaut. Auf der getesteten Taste befindet sich ein kleines Stück leitfähiger Folie, das mit der Masse verbunden ist. Der Steckverbinder, der die Taste betätigt, ist mit einem Nano-Eingang verbunden. Sobald dieser beim Eindrücken die Folie berührt, erkennt der Nano den Kontakt und erzeugt einen Impuls. Dieser wird über einen Spannungsteiler an den Eingang eines Audio-Interfaces geführt.

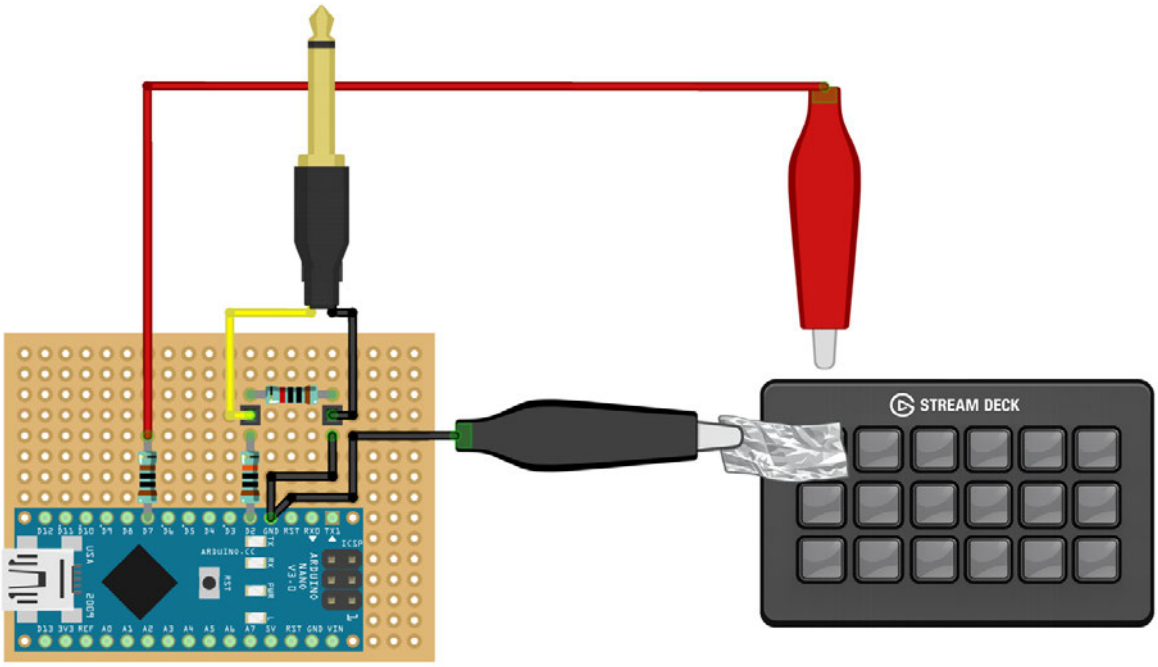


Abbildung 31: Messadapter für den Stream Deck-Pfad.

Mit demselben Tastendruck wird zugleich ein Clip in Ableton ausgelöst, der einen weiteren Impuls enthält. Die beiden Impulse werden links und rechts gepannt und auf einen Stereo-Aufnahmekanal geroutet.

Für den Poti-Pfad wurde der im Controller verbaute Pro Micro um einen digitalen Referenzausgang sowie um einen Testmodus erweitert. Der Testmodus erzeugt das Steuersignal unabhängig von einer realen Drehbewegung, um eine wiederholbare Folge identischer Messungen zu ermöglichen. In festen Abständen setzt die Firmware zunächst einen kurzen Referenzpuls an Pin D2 und sendet anschließend eine CC-Nachricht. Der Referenzpuls wird, wie beim ersten Adapter, über einen Spannungsteiler an einen Eingang des Audiointerfaces geführt und markiert den Sendezeitpunkt. Erreicht die CC-Nachricht das Remote Script in Ableton, löst dieses einen vorbereiteten Clip aus, dessen Impuls wie beim Stream Deck-Pfad die Reaktion markiert.

Beim Poti-Pfad wird nicht der Beginn der mechanischen Drehbewegung gemessen, da dieser Zeitpunkt nicht eindeutig definierbar ist. Die Eingangsverarbeitung der Potentiometer wird im Testmodus durch das synthetisch erzeugte Steuersignal umgangen, weshalb die Latenz erst ab dem Sendezeitpunkt der CC-Nachricht erfasst wird.

7.1.2 Aufbau

Die Messungen finden in Regie 4 des Tonlabors der HAW statt. Für die drahtlose Anbindung wurde ein Router in das Hochschulnetz eingebunden. Ein zusätzlicher Switch ist dort nicht zulässig, da die Zugangs-Ports des Hochschulnetzes so konfiguriert sind, dass ein nachgeschalteter Switch automatisch erkannt und der Port deaktiviert wird. Ein Router ist dagegen erlaubt, sofern er über seinen WAN-Port

verbunden ist. Seine MAC-Adresse wurde dafür im Hochschulnetz freigeschaltet. Da das Tonlabor ein eigenes Netzsegment betreibt, wird dem Router eine feste IP-Adresse manuell zugewiesen.

Die Latenz wird in zwei Fällen gemessen, die sich durch die Host-Anbindung unterscheiden.

Im Fall C1 ist der Host kabelgebunden (Ethernet) mit dem Router verbunden und der Pi per WLAN. Im Signalweg liegt damit eine einzige Funkstrecke zwischen zwei Geräten, im Folgenden Funkhop. Diese Bedingung entspricht dem vorgesehenen Betrieb, bei dem nur der Controller drahtlos ist. Im Fall C2 sind Host und Pi beide per WLAN angebunden. Im Signalweg liegen zwei Funkhops, deren Schwankungen sich addieren. Beide Fälle wurden kurzzeitig nacheinander gemessen, sodass für C1 und C2 vergleichbare Bedingungen vorlagen.

Gemessen wurde mit einem RME-Babyface als Audiointerface (48 kHz, Eingangslatenz 1,4 ms). Der Router ist ein Netgear Nighthawk AC1900, der über das 5-GHz-Band (802.11ac) verbunden ist und sich in einem Abstand von etwa 4 m zum Controller befindet.

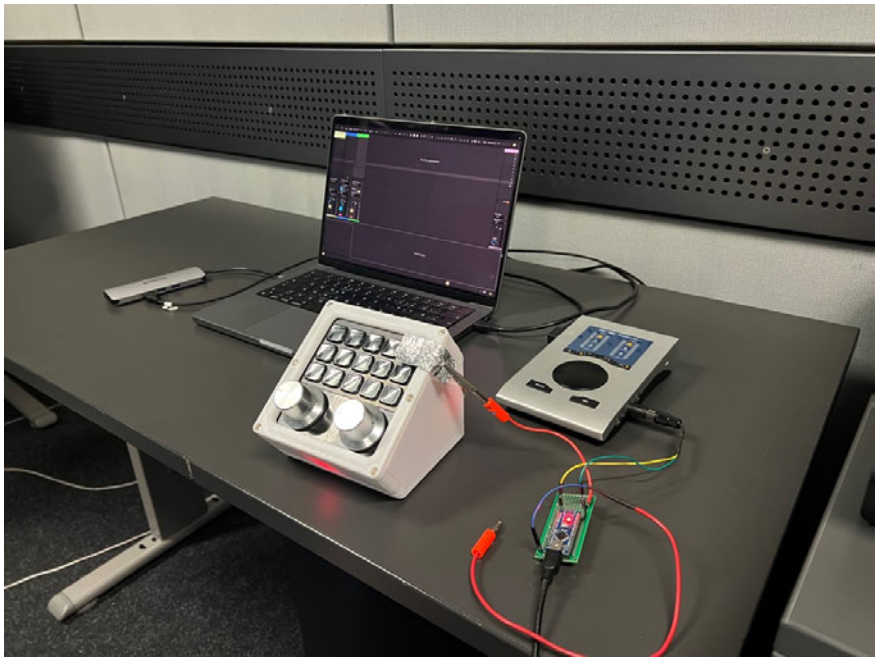


Abbildung 32: Messaufbau für den Stream Deck-Pfad.

Für beide Messpfade wird dasselbe Ableton-Testprojekt verwendet. Der Puls-Clip liegt auf Track 1, Scene 1 der Session-Ansicht. Damit ein ausgelöster Clip nicht erst am nächsten Rasterpunkt startet, werden Global Quantization und Clip Launch Quantization auf None gesetzt. Track Delays, Count-in und nicht benötigte Plugins werden deaktiviert. Auf dem Raspberry Pi wird vorab geprüft, ob eine Unterspannungs- oder Throttling-Warnung vorliegt, da Netzwerk- und Energiesparzustände die USB-over-IP-Verbindung beeinflussen können.

Abbildung 33: Messaufbau für den Potentiometer-Pfad.

Zur Absicherung der Messkette wurde ein Validierungstest durchgeführt, bei dem eine bekannte künstliche Verzögerung hinzugefügt und geprüft wurde, ob sie sich additiv in der gemessenen Latenz widerspiegelt. Stimmt die gemessene Differenz mit der eingestellten Verzögerung überein, erfasst der Aufbau Verzögerungen korrekt und linear.

7.1.3 Durchführung

Je Pfad und Bedingung wurden rund 60 Einzelmessungen aufgezeichnet. Im Poti-Pfad löst die Firmware im Testmodus in festen Abständen automatisch ein Referenzereignis aus (s. Abschnitt Messadapter). Der Tastenkontakt im Stream Deck-Pfad wird hingegen wiederholt betätigt. Die aufgenommenen Audio-Dateien werden durch ein Python-Skript ausgewertet, das Impulse in beiden Kanälen anhand ihrer Amplituden-Peaks (Peak-Detektion mit SciPy, Virtanen et al., 2020) erkennt, sie einander zuordnet und die Zeitdifferenzen ausgibt. Die Messdaten der Evaluation liegen in der digitalen Anlage (03_Evaluationsdaten).

Da einzelne verzögerte Pakete den Mittelwert verzerren können, wird nach Kleppmann (2017) der Median anstelle des Mittelwerts als zentrale Kennzahl verwendet und um die Standardabweichung und den Maximalwert ergänzt.

Die Stabilität der Übertragung wurde jeweils über etwa zwölf Minuten hinweg gemessen, während die Potentiometer gedreht, die Tasten gedrückt und die Seiten am Stream Deck gewechselt wurden. Erfasst wurden die Antwortzeit der VirtualHere-Verbindung, die ICMP-Antwortzeit (Internet Control Message Protocol) und der Paketverlust zwischen Host und Pi (`ping`), die USB-Disconnect-Meldungen des Pi-Kernels (`dmesg`) sowie der Unterspannungs- und Drosselungsstatus des Pi (`vcgencmd get_throttled`).

7.1.4 Ergebnisse

Ein Delay von 50 ms wurde in den Signalweg integriert, wobei sich im Poti-Pfad der Median um rund 50 ms und im Stream Deck-Pfad um rund 54 ms verschob, während die Verteilungsform jeweils nahezu unverändert blieb.

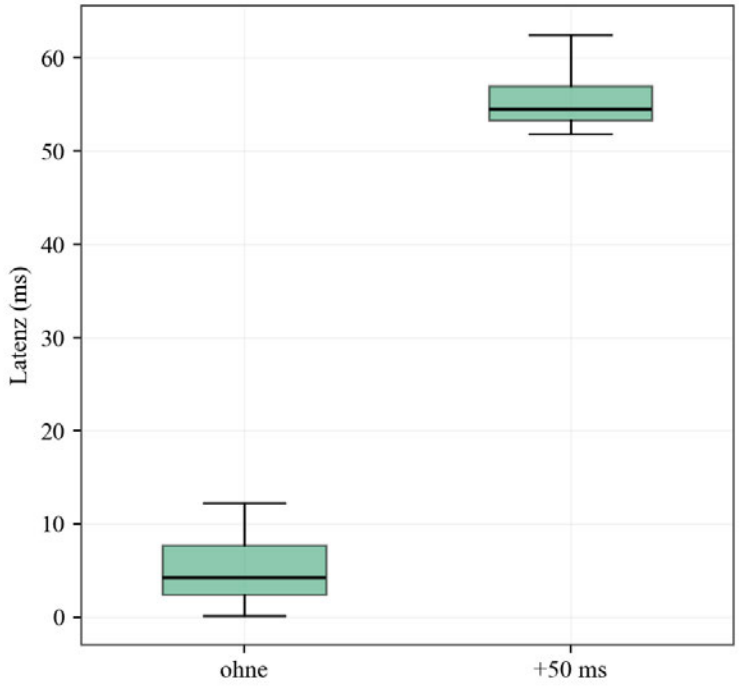


Abbildung 34: Validierung des Potentiometer-Pfads.

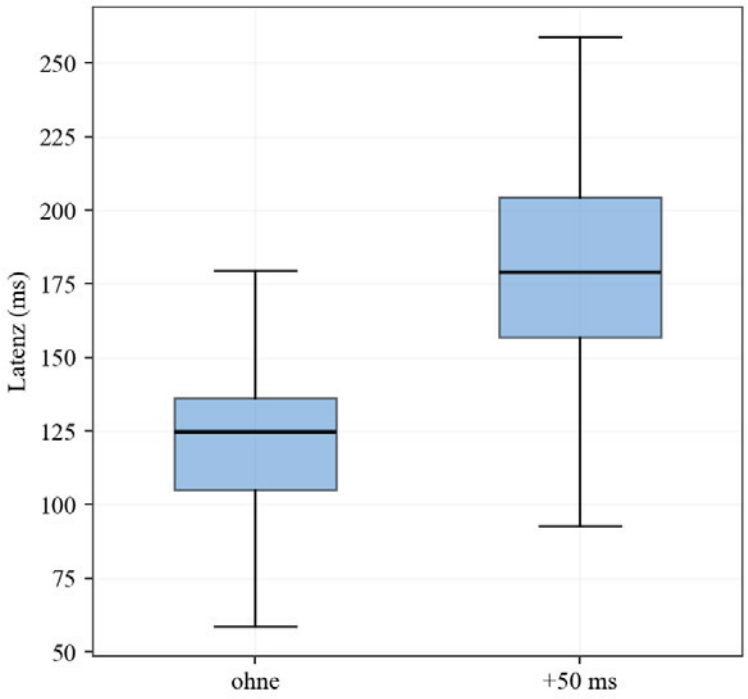


Abbildung 35: Validierung des Stream Deck-Pfads.

Tabelle 4: Latenz nach Messpfad und Bedingung.

Pfad	Median	Max	SD	n
Poti-C1	4 ms	12 ms	3 ms	62
Poti-C2	46 ms	110 ms	30 ms	61
Stream Deck-C1	125 ms	190 ms	29 ms	60
Stream Deck-C2	159 ms	271 ms	54 ms	60

Die Latenzkennwerte beider Pfade unterscheiden sich zwischen den Bedingungen C1 und C2 deutlich. Im Potentiometer-Pfad verarbeitet das Remote Script jede eingehende CC-Nachricht unmittelbar bei ihrem Eintreffen, ohne auf einen Verarbeitungstakt zu warten. In Bedingung C1 liegt der Median bei rund 4 ms und die Standardabweichung (SD) bei rund 3 ms.

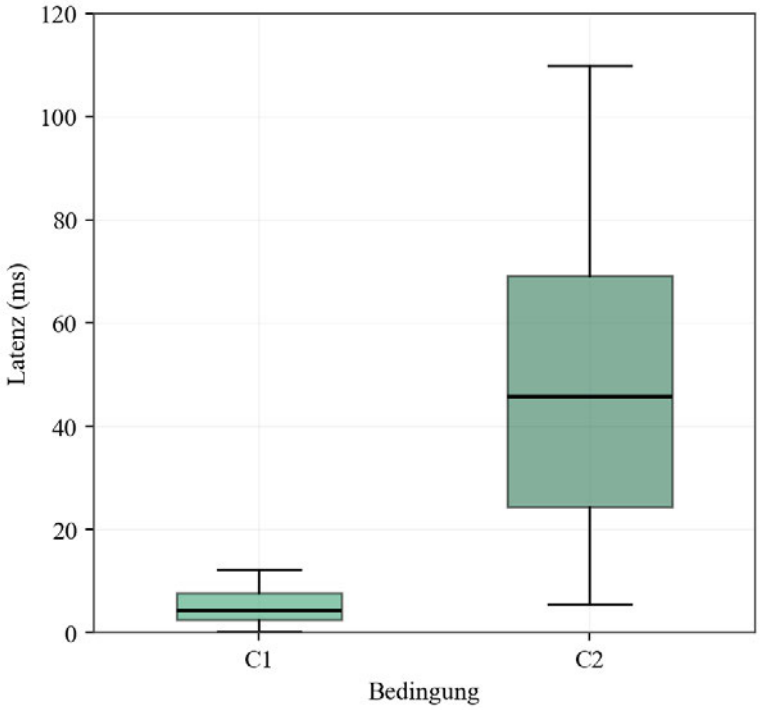


Abbildung 36: Latenzverteilung Potentiometer-Pfad.

Im Fall C2 kommt ein zweiter Funkhop hinzu, dessen Laufzeit und Schwankungen sich zu denen des ersten Hops addieren. Der Median steigt auf rund 46 ms, die Standardabweichung auf rund 30 ms, und einzelne Messungen erreichen Werte von bis zu etwa 110 ms.

Im Betrieb C1 bleibt die Latenz des Potentiometer-Pfads deutlich unter dem Zielwert von 20 ms, in C2 liegt sie dagegen klar darüber. Als ungünstigster Fall überschreitet C2 damit auch die strengere, für Musikinstrumente diskutierte Schwelle von rund 10 ms, die nach McPherson et al. (2016) viele gängige Controller nicht einhalten.

Der Stream Deck-Pfad liegt mit einem Median von rund 125 ms in C1 weit über dem Median des Potentiometer-Pfads. Das Remote Script verarbeitet die Stream Deck-Befehle erst in der Methode `update_display`, die Ableton im Betrieb nur etwa alle 100 ms zum Aktualisieren der Anzeige aufruft. Jeder Befehl wartet damit bis zum nächsten Aufruf, bevor er ausgewertet wird. Zur Wartezeit kommt ein Anteil für Verarbeitung und Übertragung hinzu, der den übrigen Medianteil erklärt.

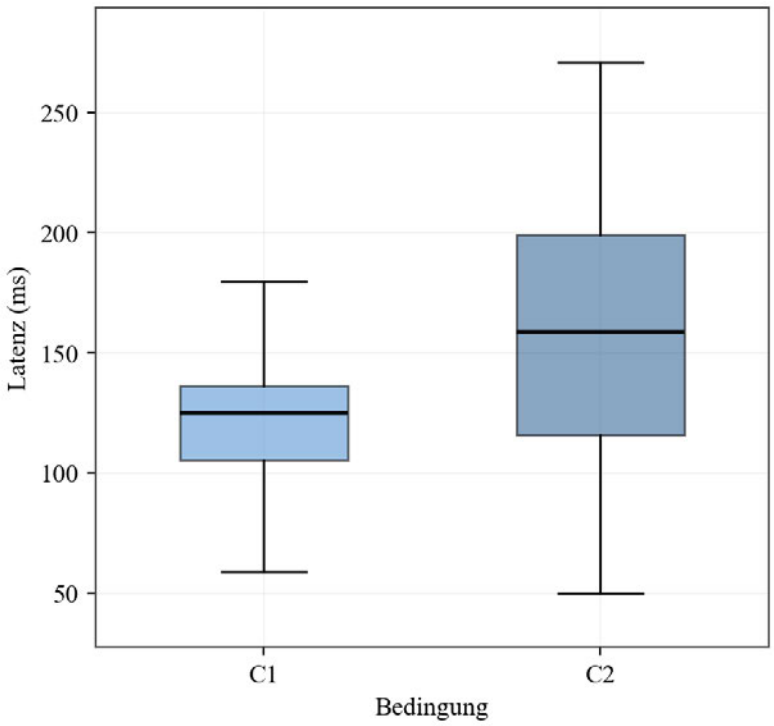


Abbildung 37: Latenzverteilung Stream Deck-Pfad.

Die Standardabweichung von rund 29 ms folgt aus der Verteilung dieser Wartezeit. Da die Befehle zu beliebigen Zeitpunkten eintreffen, ist die Wartezeit näherungsweise gleichverteilt über das rund 100 ms lange Fenster. Die Standardabweichung einer Gleichverteilung über ein Intervall der Breite b erfolgt über folgende Formel.

$$\sigma = \frac{b}{\sqrt{12}}$$

Bei einer Fensterbreite von rund 100 ms ergibt sich daraus:

$$\frac{100\ ms}{\sqrt{12}} \approx 28{,}9\ ms$$

Dieser Wert stimmt mit der gemessenen Standardabweichung von 28,8 ms nahezu exakt überein und bestätigt den Aufruftakt als mögliche Ursache der Streuung.

Unter C2 steigt der Median auf rund 159 ms, und die Streuung nimmt gegenüber C1 zu. Dieser Wert liegt deutlich über dem Zielwert, beeinträchtigt die Bedienung jedoch kaum. Die zeitkritische Steuerung

erfolgt über die Potentiometer, während das Stream Deck lediglich diskrete Auswahl- und Navigationsschritte übernimmt.

Tabelle 5: Stabilität im Betrieb (C1, C2).

Metrik	C1	C2
VirtualHere-RTT (Median / Max)	3,0 ms / 16,0 ms	6,3 ms / 49,1 ms
ping Host → Pi (Median / Max)	2,2 ms / 44,2 ms	4,5 ms / 47,0 ms

Die VirtualHere-Antwortzeiten bleiben über den gesamten Verlauf weitgehend konstant (Abbildung 38), während die `ping`-Messung bis auf einen einzelnen kurzen Ausschlag ebenfalls ruhig verläuft (Abbildung 39).

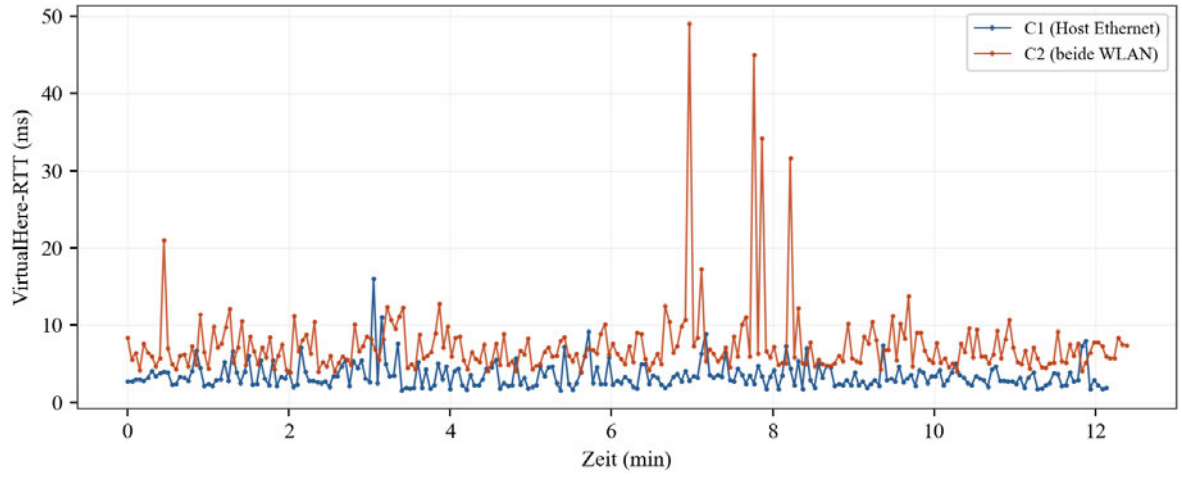


Abbildung 38: Antwortzeiten im Dauerbetrieb (VirtualHere-USB-RTT).

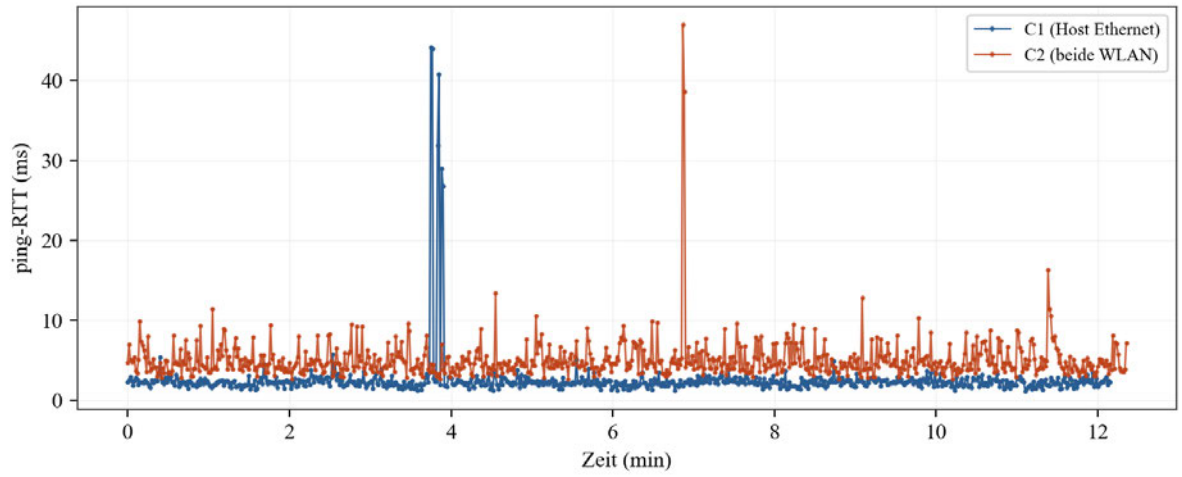


Abbildung 39: Antwortzeiten im Dauerbetrieb (ICMP-ping).

Der zweite Funkhop in C2 hebt den Median der VirtualHere-Antwortzeit von 3,0 auf 6,3 ms und den Maximalwert von 16,0 auf 49,1 ms. Der Ausschlag im `ping` erreichte unter C1 für rund zehn Sekunden bis zu 44 ms, während die VirtualHere-Antwortzeit zeitgleich ihren C1-Maximalwert erreichte. Da beide

Messungen denselben Funkhop nutzen, liegt eine kurzzeitige Störung dieses Hops nahe, die den `ping` stark und die Datenübertragung nur schwach traf.

In beiden Messungen trat kein Paketverlust auf (rund 730 Pakete in C1, rund 740 in C2), und auch unter zwei Funkhops kam es zu keinem USB-Reset oder Disconnect.

7.2 Nutzungsevaluation

Mit drei Testpersonen ist die Untersuchung bewusst klein gehalten und soll eventuelle Bedienhürden sichtbar machen sowie Hinweise liefern, ob die Bedienlogik nach einer kurzen Einführung trägt. Statistisch verallgemeinerbare Werte sind damit nicht angestrebt. Nielsen und Landauer (1993) zeigen, dass bereits kleine Stichproben einen erheblichen Teil der Usability-Probleme aufdecken können. Kiefer et al. (2008) setzen für Musik-Controller ein aufgabenbasiertes, beobachtungsgestütztes Vorgehen aus der HCI-Forschung ein. Leitend ist die Frage, ob eine Person mit DAW-Grundkenntnissen die zentralen Bedienhandlungen nach einer Einführung in die Steuerung größtenteils selbstständig ausführen kann.

Ausgewählt werden Personen mit DAW-Erfahrung, da die Bedienlogik des Controllers geprüft wird und nicht etwa das Verständnis von DAW-Konzepten. Alle Testpersonen arbeiten an derselben vorbereiteten Multitrack-Session, sodass musikalische Fähigkeiten nicht vorausgesetzt werden. Die Multitracks und deren Quelle sind im USB-Anhang hinterlegt (04_Multitrack).

7.2.1 Durchführung

Die Evaluation dauert rund 50 Minuten und findet in Regie 4 des Tonlabors statt, beginnend mit einer kurzen mündlichen Befragung (s. Anhang D). Vor dem Hintergrund der Vorerfahrung lässt sich beurteilen, ob die beobachtete Erlernbarkeit auch bei geringer Ableton-Erfahrung trägt oder stark vom Vorwissen abhängt.

Abbildung 40: Testaufbau in Regie 4.

Es folgt eine geführte Einführung, die rund zehn Minuten dauert und eine vollständige Tour durch alle in den Aufgaben vorkommenden Bedienhandlungen umfasst. Keine Aufgabe verlangt damit eine Funktion, die zuvor nicht gezeigt wurde. Die Testpersonen dürfen die gezeigten Handlungen selbst ausprobieren. Damit erhalten alle dieselbe Ausgangsbasis, und der Aufgabenteil prüft, ob die gezeigte Bedienlogik angewendet wird, und nicht, ob sie entdeckt wird.

Die Aufgaben werden mündlich nach dem Wortlaut des Testleitfadens (s. Anhang B) gestellt und benennen das Ziel sowie die zu verwendenden Tracks, Devices und Parameter, jedoch nicht den Bedienpfad. Die Aufgaben steigern sich in ihrer Komplexität, sodass einfache, früh geübte Handlungen in späteren Aufgaben bereits vertraut sein können.

Während der Bearbeitung werden für jede Aufgabe Beobachtungsnotizen (s. Anhang C) angelegt, die festhalten, wie viel Unterstützung erforderlich war (0 = keine Hilfe, 1 = Hinweis erforderlich, 2 = konkretes Erklären erforderlich, 3 = Aufgabe nicht gelöst). Die Hinweise sind für jede Aufgabe im Testleitfaden vordefiniert. Zusätzlich wird notiert, welcher Hinweis im Einzelfall nötig war, sodass nachvollziehbar bleibt, an welchem Schritt die Bedienung unklar war.

Nach den Aufgaben folgt ein Fragebogen (s. Anhang D) mit einer vierstufigen Zustimmungsskala ohne neutrale Mittelkategorie sowie mit offenen Fragen. Durch die fehlende Mitte muss sich für eine Richtung entscheiden werden. Zudem sind die Aussagen sowohl positiv als auch negativ formuliert, damit die Antworten nicht schematisch ausfallen. Die offenen Fragen sammeln konkrete Verbesserungsvorschläge und ergänzen Bereiche, die die Zustimmungsskala nicht abdeckt. Zum Abschluss wird dem Probanden Raum für mündliche Anmerkungen gegeben, um das Machtgefälle nach der Befragung abzumildern und mögliche Lücken im Feedback zu schließen.

7.2.2 Ergebnisse

Alle drei arbeiten regelmäßig mit einer DAW und steuern sie üblicherweise mit der Maus und der Tastatur. TP1 und TP2 nutzen Reaper und haben keine praktische Erfahrung mit Ableton, während TP3 mit Ableton Live arbeitet und daneben Hardware wie eine MPC One und ein Eurorack-System bedient, also den Umgang mit physischen Reglern gewohnt ist.

In der Beobachtung lösten alle drei Testpersonen sämtliche Aufgaben, die höchste vergebene Hinweisstufe lag bei 2. Die Grundnavigation aus Transportfunktionen und Track-Auswahl gelang ohne Hilfe. Erst mit zunehmender Aufgabenkomplexität nahmen die Hinweise zu, wobei der Hilfebedarf unterschiedlich ausfiel. TP2 brauchte über alle Aufgaben hinweg deutlich mehr Unterstützung als die anderen beiden, mit einer Summe der Hinweisstufen von 6 gegenüber 1 bei TP1 und 2 bei TP3. Die Ableton-Vorerfahrung erklärt diesen Abstand nicht, denn auch TP1 arbeitete nicht mit Ableton und kam fast ohne Hilfe aus. Worauf genau der Unterschied beruht, lässt sich nicht eindeutig benennen.

Die Steuerung der Potentiometer wurde von allen Testpersonen als präzise eingestuft, und sie verneinten, dass ein einzelnes Potentiometer für die Aufgaben ausgereicht hätte. In den Antworten auf

die offenen Fragen wurden die Potentiometer als größte Stärke genannt, von TP1 ausdrücklich als feiner bezeichnet als es mit der Maus möglich sei. Eine störend verzögerte Reaktion gab keine der Testpersonen an, was der gemessenen Latenz des Poti-Pfads entspricht. Diese durchweg gute Bewertung muss jedoch in Bezug auf die Testsituation betrachtet werden, denn die Testpersonen antworteten unmittelbar nach der angeleiteten Sitzung und in Gegenwart des Testleiters, was eventuell zu positiveren Antworten geführt haben könnte. Zudem ist die Frage zur Potentiometeranzahl unglücklich formuliert, da einige Aufgaben auf die simultane Steuerung ausgelegt waren.

Die größten Schwierigkeiten lagen in der Seitenstruktur, wobei TP1 und TP2 eher zustimmten, zeitweise verwirrt gewesen zu sein, während TP3 dies verneinte. In den Aufgaben zeigte sich das bei jeder Person an anderer Stelle. TP1 war die Mehrfachauswahl der Sends zunächst unklar, bei TP2 die Poti-Zuweisung und bei TP3 das Anlegen der Return-Tracks, weil die Returns im Mixer zwar sichtbar, jedoch nicht anlegbar sind. Auch einzelne Funktionen waren schwer auffindbar oder schwer zu erinnern, etwa der Fine-Mode bei TP2 und die Scrub-Taste bei TP3. Dass die Probleme bei jeder Person anders auftraten, spricht eher gegen einen einzelnen Strukturfehler und eher dafür, dass man sich bei der Bedienung einarbeiten muss.

Alle drei Testpersonen benötigten bei der Automationsaufgabe Hilfe, wobei TP1 und TP3 vergaßen, die Automationsaufnahme zu aktivieren. Das ist vor allem auf die Komplexität der Aufgabe zurückzuführen, bei der viele Bedienhandlungen erforderlich waren. Ein möglicher Grund für das Vergessen ist, dass man in manchen DAWs wie Reaper keine zusätzliche Aufnahme aktivieren muss, sobald die Automationsspur scharfgeschaltet ist. Insgesamt wirkt es eher wie das Übersehen eines einzelnen Schritts als ein grundsätzliches Verständnisproblem.

Beobachtung und Fragebogen widersprechen sich nicht grundlegend, doch fallen die Fragebogenangaben durchgehend positiver aus, als es das Verhalten nahelegt. Alle bewerteten die Einführung als ausreichend, kamen bei den schwierigeren Aufgaben jedoch nicht ohne Hilfe aus, und bei TP2 gingen der beobachtete Hilfebedarf und die eigene Bewertung am weitesten auseinander, denn der Fragebogen fiel fast genauso positiv aus wie bei den anderen Testpersonen. Diese positivere Selbsteinschätzung sollte im Hinblick auf die unmittelbaren Aufgabenerfolge betrachtet werden.

Bezogen auf die Anforderung N1 konnten die drei Testpersonen die Bedienung nach einer zehnminütigen Einführung dennoch weitgehend selbstständig anwenden. Diese Aussage besitzt jedoch Grenzen, denn die Hinweisstufen beruhen auf einer subjektiven Einschätzung meinerseits. Zudem greift die während der Aufgaben gegebene Hilfe in den Lernprozess ein und kann diesen zugunsten der Erlernbarkeit beeinflussen. Die Anforderung N2 ist teilweise erfüllt, denn das Erkennen der gerade gesteuerten Funktion gelang allen Testpersonen, während die visuelle Auffindbarkeit einzelner Funktionen ein Reibungspunkt blieb.

7.2.3 Abgeleitete Verbesserungen

TP2 wünschte sich eine Übersicht über die Tastenfunktionen in Form eines optional aktivierbaren „Infocharts“. TP3 wünschte sich den Wechsel zwischen Solo und Arm auch außerhalb des Hauptmenüs sowie eine einheitlichere Anzeige der Return-Kanäle im Mixer. Hinzu kamen mehr Feinheitsoptionen für das Potentiometer und per Tastendruck wählbare Quantisierungsstufen der Timeline. TP1 schlug vor, Locator-Marker direkt durch Antippen eines freien Slots zu setzen, da dies eher der Stringenz anderer Bedienhandlungen folgen würde.

Hinzu kommt, dass Maus und Tastatur die bevorzugten Eingabemethoden beim Laden und Suchen von Effekten (TP1), beim Setzen der Loop-Punkte und beim Wechsel zwischen Solo und Arm (TP3) blieben, sodass darauf zu schließen ist, dass diese Bedienkonzepte noch nicht vollständig ausgereift sind.

7.3 Grenzen der Evaluation

Die Evaluation beschreibt den aktuellen Stand eines Einzelprototyps. Die Ergebnisse hängen vom verwendeten Hostsystem, Netzwerk, Audiointerface, Ableton-Set und der konkreten Implementierung ab, wobei auch andere Netzwerke, Puffergrößen oder Plugins die Ergebnisse beeinflussen können.

Die Latenzmessung weist methodische Grenzen auf, da der Stream Deck-Test die Messung bereits beim Folienkontakt auf der Taste startet und nicht am internen Schaltpunkt des Stream Decks. Beim Poti-Test startet sie beim Senden eines MIDI-Ereignisses durch die Firmware und nicht beim Beginn einer Drehbewegung. Der Poti-Test erfasst zudem ein einzelnes Ereignis je Intervall und nicht den dichten Datenstrom einer länger anhaltenden Drehung. Der ausgegebene Impuls durchläuft außerdem den Audioausgabepuffer, der als Anteil im Messwert enthalten ist. Die WLAN-Kanalbelegung und mögliche koexistierende Netze wurden nicht erfasst, sodass die absoluten Streuungs- und Spitzenwerte, insbesondere in C2, nur eingeschränkt reproduzierbar sind. Auch die Stabilitätsmessung beruht je Bedingung auf einem einzelnen Dauerlauf ohne Wiederholung, sodass seltene, erst über längere Zeiträume auftretende Aussetzer nicht ausgeschlossen werden können.

Eine weitere Einschränkung betrifft die Blickunabhängigkeit. Die Möglichkeit einer blickunabhängigen Wertänderung gilt als Eigenschaft haptischer Bedienelemente. Zwar können die Potentiometer ohne direkten visuellen Bezug bedient werden, jedoch erfordern die Auswahl eines Mapping-Ziels und die Navigation durch die Seitenstruktur am Stream Deck Blickkontakt.

8 Zusammenfassung

Der Prototyp deckt die geforderten Bedienhandlungen ab, und die drei Testpersonen wendeten sie nach einer kurzen Einweisung selbstständig an. Am klarsten gelang die kontinuierliche Steuerung über die Potentiometer, die im vorgesehenen Betrieb mit einem Median von rund 4 ms reagiert und von den Testpersonen als präzise und als größte Stärke beschrieben wurde. Da die Steuerung relativ arbeitet, springt ein Wert beim Wechsel des Bedienziels nicht, und die Drehbewegung bleibt nach dem Zuweisen ohne Blick auf das Gerät bedienbar. Auch der drahtlose Betrieb hielt im Dauerlauf, da die Übertragung ohne Paketverlust und ohne Unterbrechung der Geräteverbindung verlief.

Die Grenzen des Systems betreffen vor allem die Reaktionszeit, die Sichtbarkeit der Belegung und den drahtlosen Betrieb. Die Latenz der Potentiometer hängt von der Netzanbindung ab und überschreitet den Zielwert, wenn zusätzlich zum Controller auch der Hostrechner drahtlos angebunden ist. Der Weg über das Stream Deck ist an den Anzeigetakt von Ableton gebunden und reagiert langsamer als der des Potentiometers, was für die diskrete Navigation jedoch genügt. In der Bedienung blieb das Auffinden einzelner Funktionen in der Seitenstruktur ein Reibungspunkt, und die autarke Stromversorgung über längere Sitzungen sowie eine durchgängig blickfreie Bedienung sind noch offene Bereiche.

Die Nutzungsevaluation mit drei Personen zeigt die Erlernbarkeit der Bedienung, erlaubt aber keine verallgemeinerbaren Aussagen, zumal der Fragebogen durchgehend positiver ausfiel als das beobachtete Verhalten. Für einige Aufgaben blieben Maus und Tastatur die bevorzugten Werkzeuge, sodass der Controller sie ergänzt, ohne sie zu ersetzen. Insgesamt trägt der Prototyp seine zentrale Aufgabe, die haptische und blickfreie Wertänderung im laufenden Betrieb, während die übrigen Anforderungen in abgestuftem Maß erreicht sind.

Literaturverzeichnis

Ableton. (o. D.-a). *MIDI and Key Remote Control*. Ableton Reference Manual Version 12. Abgerufen am 8. Mai 2026, von https://www.ableton.com/en/live-manual/12/midi-and-key-remote-control/

Ableton. (o. D.-b). *Installing third-party remote scripts*. Ableton Knowledge Base. Abgerufen am 8. Mai 2026, von https://help.ableton.com/hc/en-us/articles/209072009-Installing-third-party-remote-scripts

Ableton. (o. D.-c). *Creating your own Control Surface script*. Ableton Knowledge Base. Abgerufen am 31. Mai 2026, von https://help.ableton.com/hc/en-us/articles/206240184-Creating-your-own-Control-Surface-script

Alpha. (o. D.). *RV112FF Series Rotary Potentiometer* [Datenblatt]. Mouser Electronics. Abgerufen am 1. Juni 2026, von https://www.mouser.de/ProductDetail/Alpha-Taiwan/RV112FF-40-20A-B20K

Arduino. (2020). *MIDIUSB*. Arduino Libraries. Abgerufen am 31. Mai 2026, von https://www.arduinolibraries.info/libraries/midiusb

Atelier Du Tech. (2024). Stream Deck XL Companion Pi Case 02 [3D-Modell]. Printables. Abgerufen am 12. Juni 2026, von https://www.printables.com/model/865315-stream-deck-xl-companion-pi-case-02

Atmel. (2010). *ATmega16U4/32U4: 8-bit Microcontroller with 16/32K Bytes of ISP Flash and USB Controller* [Datenblatt]. Abgerufen am 14. Juni 2026, von https://cdn.sparkfun.com/datasheets/Dev/Arduino/Boards/ATMega32U4.pdf

Brandal, B. (2020, 22. Mai). *Driver/algorithm for 360 deg endless potentiometer*. Hackaday.io. https://hackaday.io/project/171841-driveralgorithm-for-360-deg-endless-potentiometer

Clarke, D. (2016). *Writing a Better Noise Reducing AnalogRead*. Abgerufen am 31. Mai 2026, von http://damienclarke.me/code/posts/writing-a-better-noise-reducing-analogread

Clarke, D. (2019). *ResponsiveAnalogRead*. Arduino Libraries. Abgerufen am 31. Mai 2026, von https://www.arduinolibraries.info/libraries/responsive-analog-read

CNC Kitchen. (2021). *Tipps & Tricks für Gewindeeinsätze im 3D-Druck*. Abgerufen am 12. Juni 2026, von https://www.cnckitchen.com/blog/tipps-amp-tricks-fr-gewindeeinstze-im-3d-druck-3awey

Cycling '74. (o. D.-a). *Live API overview*. Cycling '74 Documentation. Abgerufen am 8. Mai 2026, von https://docs.cycling74.com/userguide/m4l/live_api_overview/

Cycling '74. (o. D.-b). *Live Object Model*. Cycling '74 Documentation. Abgerufen am 8. Mai 2026, von https://docs.cycling74.com/apiref/lom/

De Pra, Y., Fontana, F., & Papetti, S. (2020). Interacting with digital audio effects through a haptic knob with programmable resistance. *Proceedings of the International Conference on Digital Audio Effects (DAFx)*.

Elgato. (o. D.-a). *Stream Deck +*. Abgerufen am 8. Mai 2026, von https://www.elgato.com/us/en/p/stream-deck-plus

Elgato. (o. D.-b). *Stream Deck SDK documentation*. Abgerufen am 31. Mai 2026, von https://docs.elgato.com/streamdeck/sdk/

Elgato. (2025, 18. Mai). *What are Stream Deck Modules? Build Stream Deck into your setup*. https://www.elgato.com/ww/en/explorer/products/stream-deck/stream-deck-modules-overview/

Gohlke, K., Hlatky, M., Heise, S., & Loviscach, J. (2010). FlexiKnobs: Bridging the gap between mouse interaction and hardware controllers. *Proceedings of the Fourth International Conference on Tangible, Embedded, and Embodied Interaction (TEI '10)*, 241–244. https://doi.org/10.1145/1709886.1709934

Hausen, D., Richter, H., Hemme, A., & Butz, A. (2013). Comparing input modalities for peripheral interaction: A case study on peripheral music control. *Human-Computer Interaction – INTERACT 2013, Lecture Notes in Computer Science, 8119*, 162–179. https://doi.org/10.1007/978-3-642-40477-1_10

HiFi-Lab. (o. D.). *Poti-Knopf Aluminium massiv 38 × 26 × 26*. Abgerufen am 12. Juni 2026, von https://www.hifi-lab.com/hifi-lab-poti-knopf-aluminium-massiv-38x26x26_1002090_1135/

International Telecommunication Union. (2015). *Methods for the subjective assessment of small impairments in audio systems* (Recommendation ITU-R BS.1116-3). https://www.itu.int/rec/R-REC-BS.1116-3-201502-I

Jones, D. (2023). AbletonOSC: A unified control API for Ableton Live. *Proceedings of the International Conference on New Interfaces for Musical Expression*, 436–440. https://doi.org/10.5281/zenodo.11189234

Kiefer, C., Collins, N., & Fitzpatrick, G. (2008). HCI methodology for evaluating musical controllers: A case study. *Proceedings of the International Conference on New Interfaces for Musical Expression*, 87–90. https://doi.org/10.5281/zenodo.1179577

Kleppmann, M. (2017). *Designing data-intensive applications: The big ideas behind reliable, scalable, and maintainable systems*. O'Reilly Media.

Lago, N. P., & Kon, F. (2004). The quest for low latency. *Proceedings of the International Computer Music Conference*, 33–36. https://www.ime.usp.br/~kon/papers/icmc04-latency.pdf

Little Devil, J. (o. D.). *EndlessPotentiometer*. GitHub. Abgerufen am 27. Mai 2026, von https://github.com/juanlittledevil/*EndlessPotentiometer*

Mäki-Patola, T., & Hämäläinen, P. (2004). Latency tolerance for gesture controlled continuous sound instrument without tactile feedback. *Proceedings of the International Computer Music Conference*, 409–416.

McLaughlin, A. C., Rogers, W. A., & Fisk, A. D. (2009). Using direct and indirect input devices: Attention demands and age-related differences. *ACM Transactions on Computer-Human Interaction, 16*(1), Article 2. https://doi.org/10.1145/1502800.1502802

McPherson, A. P., Jack, R. H., & Moro, G. (2016). Action-sound latency: Are our tools fast enough? *Proceedings of the International Conference on New Interfaces for Musical Expression*, 20–25. https://doi.org/10.5281/zenodo.3964611

MIDI Association. (o. D.). *MIDI 1.0 control change messages*. Abgerufen am 31. Mai 2026, von https://midi.org/midi-1-0-control-change-messages

Nielsen, J. (1994). Enhancing the explanatory power of usability heuristics. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 152–158. https://doi.org/10.1145/191666.191729

Nielsen, J., & Landauer, T. K. (1993). A mathematical model of the finding of usability problems. *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, 206–213. https://doi.org/10.1145/169059.169166

NOB Control Solutions. (2023). *nOb user guide: Version 1.0.3*. https://www.nobcontrol.com/sites/default/files/userguide.pdf

Orio, N., Schnell, N., & Wanderley, M. M. (2001). Input devices for musical expression: Borrowing tools from HCI. *Proceedings of the CHI Workshop on New Interfaces for Musical Expression (NIME)*.

OSRTT. (o. D.). *OSRTT: Open Source Response Time Tool* [Software]. GitHub. Abgerufen am 10. Juni 2026, von https://github.com/andymanic/OSRTT

Raspberry Pi Ltd. (o. D.). *Raspberry Pi 3 Model B+*. Abgerufen am 8. Mai 2026, von https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/

Raspberry Pi Stack Exchange. (2019). *Make „iw wlan0 set power_save off“ permanent*. Abgerufen am 2. Juni 2026, von https://raspberrypi.stackexchange.com/questions/96606/make-iw-wlan0-set-power-save-off-permanent

Rogers, W. A., Fisk, A. D., McLaughlin, A. C., & Pak, R. (2005). Touch a screen or turn a knob: Choosing the best device for the job. *Human Factors, 47*(2), 271–288. https://doi.org/10.1518/0018720054679452

Shafran, C. [justaboutdone]. (o. D.). *endless-pots*. GitHub. Abgerufen am 27. Mai 2026, von https://github.com/justaboutdone/*endless-pots*

Slavković, V., Hanželič, B., Plesec, V., Milenković, S., & Harih, G. (2024). Thermo-mechanical behavior and strain rate sensitivity of 3D-printed polylactic acid (PLA) below glass transition temperature (Tg). *Polymers, 16*(11), 1526. https://doi.org/10.3390/polym16111526

SparkFun. (o. D.). *SparkFun Qwiic Pro Micro - USB-C (ATmega32U4)*. Abgerufen am 8. Mai 2026, von https://www.sparkfun.com/sparkfun-qwiic-pro-micro-usb-c-atmega32u4.html

Steinkamp, Z. (o. D.). *Knobbler4*. Abgerufen am 12. Juni 2026, von https://plugins.steinkamp.us/m4l-*Knobbler4*

Steinkamp, Z. (2022, 16. März). *Knobbler: The pursuit of an auto-labeling control surface for Ableton Live*. https://steinkamp.us/posts/2022-03-16-knobbler

Structure Void. (o. D.). *Ableton Live Python MIDI remote scripts*. Abgerufen am 8. Mai 2026, von https://structure-void.com/ableton-live-midi-remote-scripts/

Tom's Hardware. (2019). *Raspberry Pi 4: Review, buying guide and how to use*. Abgerufen am 10. Juni 2026, von https://www.tomshardware.com/reviews/raspberry-pi-4

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods, 17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2

VirtualHere. (o. D.). *VirtualHere allows USB devices to be used remotely over a network.* Abgerufen am 8. Mai 2026, von https://www.virtualhere.com/

Wang, J., Mulder, A., & Wanderley, M. M. (2019). Practical considerations for MIDI over Bluetooth Low Energy as a wireless interface. *Proceedings of the International Conference on New Interfaces for Musical Expression*. https://www.nime.org/proceedings/2019/nime2019_paper006.pdf

Waveshare. (o. D.). *UPS HAT (D)*. Waveshare Wiki. Abgerufen am 8. Mai 2026, von https://www.waveshare.com/wiki/UPS_HAT_(D)

Wessel, D., & Wright, M. (2001). Problems and prospects for intimate musical control of computers. *Proceedings of the CHI Workshop on New Interfaces for Musical Expression (NIME)*.

Anhang

Anhang A: Datenblatt des 360°-Potentiometers (Alpha RV112FF)

Quelle: Alpha (o. D.).

POTENTIOMETERS

ALPHA

11mm Endless Rotary Type, Metal & Plastic Shaft

Application

✓ TV/Video, Audio, Musical instruments

Feature

✓ 360° degrees continuous potentiometer

■Specification

Total Rotational Angle	360° endless
Maximum Operating Voltage	5-15V DC
Insulation Resistance	100MΩ min at DC250V
Dielectric Strength	AC150V for 1 minute
Rotational Torque	10-200 gf.cm
Push/Pull Strength of the Shaft	6kgf for 3 sec
Rotation Life	15,000 cycles

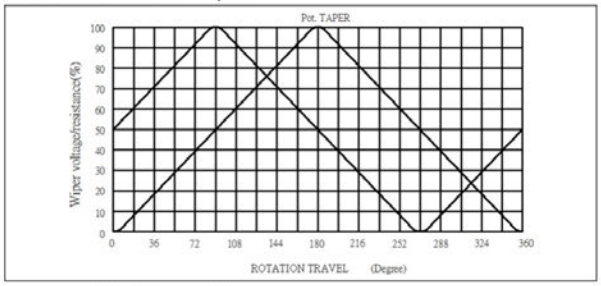
■How to order

RV112FF-40 - 15A - 0B10K

Model

Shaft Type

Standard Resistance Taper:



Taper and Resistance Value

Order Code	Taper	Resistance Value
0B1M	Linear (0B)	1MΩ

Taper: 0B taper

Resistance Value: 1KΩ to 500KΩ and 1MΩ

*Contact us for other requirements.

TAIWAN ALPHA ELECTRONIC CO., LTD.

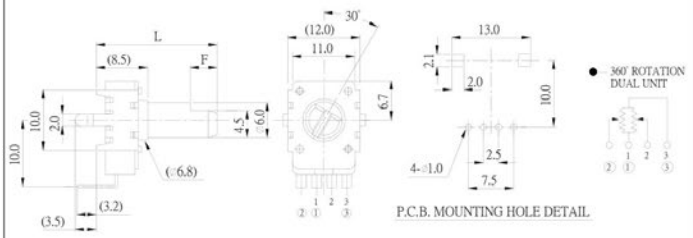
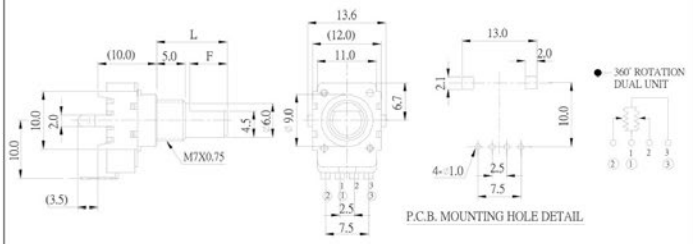
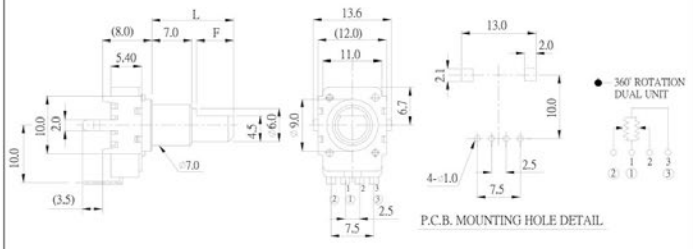
Contact: sales@taiwanalpha.com

ALPHA

11mm Endless Rotary Type, Metal & Plastic Shaft

■Model Description

Model	Number of Unit	Terminal Type	Bushing	Shaft	Rotational Angle
RV112FF-40	Dual unit	Vertical	N/A	Plastic	360°
RV112FF-40B1	Dual unit	Vertical	Metal	Metal	360°
RV112FF-40B3N	Dual unit	Vertical	Metal bushing without thread	Metal	360°

Order Code	Outline Drawing
RV112FF-40 	
RV112FF-40B1 	
RV112FF-40B3N 	

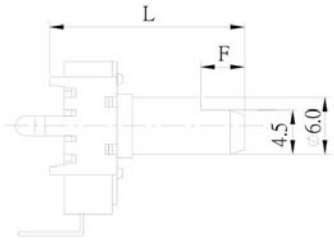
Back to top

ALPHA

11mm Endless Rotary Type, Metal & Plastic Shaft

■ A Type Plastic Shaft

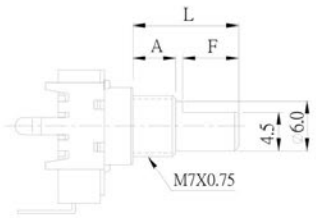
RV112FF-40-(L)A



Order Code	15A	20A	225A	25A		275A	30A	325A	375A
L	15	20	22.5	25	25	27.5	30	32.5	37.5
F	4.5	7	7	7	12	12	12	12	12

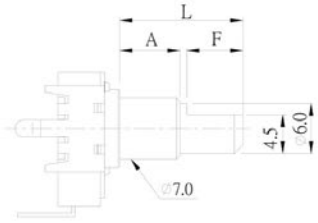
■ F Type Metal Shaft

RV112FF-40B1-(L)F



Order Code	15F	20F	25F
L	15	20	25
A	5	5	5

RV112FF-40B3N-(L)F



Order Code	15F	20F	25F
L	15	20	25
A	7	7	7

Design and specifications presented here are for the standard parts only. Please kindly contact us for your special requests and ask for the current technical specifications before purchase and/or use.

Back to top

Anhang B: Testleitfaden der Nutzerevaluation

Testleitfaden

1. Einführung

Grundprinzip vom Controller

Overview

- Einstiegsseite
- Transport: Play und Record
- Location-Browser, Mixer und den Track-Reihen

Location-Browser

- Locator-Marke setzen, anspringen, löschen
- Scrubbing
- Hinein- und herauszoomen
- Loop-Bereich aufspannen

Mixer

- Langer Druck im Overview: Wechsel zwischen Arm und Solo
- Reihe 1 Transportfunktionen, Arm Automation (Langer Druck zeigt Automationsspuren)
- Reihe 2 regulären Tracks (seitenweise blätterbar), Reihe 3 fest den Master und die Returns
- Linker Poti = Lautstärke
- Rechter Poti = Panning
- Langer Druck löst je nach global gewähltem Modus Arm oder Solo aus, bei Return-Tracks immer Solo, beim Master keine Aktion
- Mapping automatisch als Mehrfachauswahl
- Mapping bleibt nur bestehen, wenn man sich auf der Seite befindet

Track-Reihen

- Je nach Track andere Funktionen (Solo/Arm, Return Browser)
- Leeren Slot Effekt laden
- Langer Druck auf Effekt löschen
- Langer Druck Track öffnet die Liste zum Spurwechsel

Plugin-Detail und Parameter-Browser

- Plugin-Detail zeigt sechs Parameter des gewählten Devices
- FX-Open (einzeln öffnen und schließen, bei langem Druck alle schließen)
- Parameterbutton wählt das Ziel, dann Map Pot 1 oder Map Pot 2 zum Zuweisen, Fine Mode
- Toggle-Parameter sehen anders aus als kontinuierliche Werte
- Parameter-Browser listet alle Parameter des Devices auf, Austausch darüber möglich
- Bypass, Solo/Arm

Returns und Tracks hinzufügen

- Return-Send Seite (mehrere Zuordnungen möglich, Mapping bleibt bestehen, Mapping lösen hier wichtig)
- Neue Tracks über die Plus-Seite oder Return-Seite

2. Nutzertest

Die Aufgaben sind konkret vorgegeben. Es geht dabei nicht um die mischtechnische Qualität des Ergebnisses, sondern darum, ob die genannten Bedienhandlungen ausgeführt werden können.

Aufgabe 1 – Transport und Track-Auswahl

- Starte die Wiedergabe, hör dir den Track an und stoppe sie dann wieder.

  Funktion: Transport.

  Hinweis: obere Tastenreihe.

- Lade die Tracks Guitar DI und die Vocals in die Overview.

  Funktion: Track-Auswahl / Navigation.

  Hinweis: Langes Drücken öffnet die Trackliste.

Aufgabe 2 – Locator und Loop

- Setze einen Locator an den Anfang des Bereichs, in dem alle Instrumente zu hören sind, und einen zweiten an dessen Ende. Spanne zwischen beiden einen Loop auf.

  Funktion: Playhead-Scrub, Locations setzen, Loop.

  Hinweise: zum Finden der Stelle mit dem Poti durch die Session scrubben, Langes Drücken spannt Loop auf

Aufgabe 3 – Lautstärke und Panning

- Senke die Lautstärke der Guitar-DI- und der Djembe-Top-Spuren leicht ab. Panne anschließend die beiden Djembe-Spuren (Top, Bottom) gegensätzlich und die Guitar DI etwas nach rechts.

  Funktion: Track-Mixer.

  Hinweise: linker Poti Lautstärke, rechter Poti Panning, Djembe Bottom auf der nächsten Seite

Aufgabe 4 – Parameter steuern

- Vocals: Lade den Channel EQ und hebe den High-Gain leicht an, schalte zusätzlich den Highpass an.

  Funktion: Mapping eines einzelnen Parameters und Toggle

  Hinweise: Plugin auf einen leeren FX-Slot laden, Mapping über Map Poti-Button

- Vocals: Senke den Mid-Gain Parameter im Fine Mode fein ab.

  Funktion: Fine Mode.

  Hinweis: Langes Drücken auf den Map-Button schaltet in den Fine Mode um.

- Vocals: Lade einen LA2A-Kompressor und stelle gleichzeitig Gain und Peak Reduction ein
- Nutze die Solo- und Bypass-Funktion, um die Lautstärke der Vocals vor und nach dem Kompressor anzugleichen

  Funktion: zwei Parameter mappen, Solo und Bypass, öffnen des Plugin-Fensters

  Hinweise: Mapping über beide Map Poti-Buttons, Open-Plugin-Button

- Akkustikgitarre DI: Lade Microshift (Stereobreite) und stelle gleichzeitig Mix und Focus (Frequenzweiche) ein.

  Funktion: Track austauschen, zwei Parameter mappen, öffnen und schließen der Plugin-Fensters

  Hinweise: Langes Drücken auf den Open-Plugin-Button, schließt beide Fenster

Aufgabe 5 – Returns/Sends und Automation

- Lege einen Return-Track an und lade ValhallaPlate (Reverb) darauf.

  Funktion: Return anlegen, Device laden.

  Hinweis: über die Return-Seite.

- Lege den Reverb-Send für die Vocals auf ein Poti und die Reverb-Sends der beiden Gitarren auf den anderen Poti. Nimm eine Send-Automation auf, in der du den Sendanteil der beiden Tracks im Verlauf variierst.

  Funktion: Send-Steuerung und Automation.

  Hinweise: Zuerst die Potis zuweisen, dann die Aufnahme starten und sie während der Aufnahme bewegen.

Anhang C: Beobachtungsbögen des Nutzertests

Teilnehmernummer: 1

Aufgabe	Stufe (0–3)	Notiz
1 – Transport und Track-Auswahl	0	
2 – Locator und Loop	0	
3 – Mixer: Lautstärke, Panning, Solo	0	Versehentlich Vocal angefasst
4 – Parameter mit den Potentiometern	0	Microshift musste ich Funktion erklären (Focus)
5 – Returns/Sends und Automations	1	Arm Automation vergessen (record vergessen)

Teilnehmernummer: 2

Aufgabe	Stufe (0–3)	Notiz
1 – Transport und Track-Auswahl	0	
2 – Locator und Loop	2	Scrubbing Button vergessen Loop 2. Position halten vergessen
3 – Mixer: Lautstärke, Panning, Solo	1	Panning & nicht schnell gefunden Unselecten vergessen
4 – Parameter mit den Potentiometern	1	Fine Mode nicht gefunden
5 – Returns/Sends und Automations	2	Verwirrung hin- & hersuchen bei m Parameter Mapping und Sends

Teilnehmernummer: 3

Aufgabe	Stufe (0–3)	Notiz
1 – Transport und Track-Auswahl	0	
2 – Locator und Loop	0	Scrub button schon wieder nicht gefunden
3 – Mixer: Lautstärke, Panning, Solo	0	Ummappen vergessen
4 – Parameter mit den Potentiometern	1	Bypass vergessen
5 – Returns/Sends und Automations	1	Mix Ansicht gewählt anstatt Track Arm Automation, Record vergessen

Anhang D: Ausgefüllte Fragebögen der Nutzungsevaluation

Teilnehmernummer: 1

Mündliche Befragung

Dieser Leitfaden ordnet ein, wie regelmäßig und womit die Testperson arbeitet. Die genannten Antwortoptionen sind Anhaltspunkte. Abweichende Angaben werden notiert.

1. Mit welcher DAW oder welchen DAWs arbeitest du?

□ Ableton Live □ Logic □ FL Studio ☒ Reaper □ Nuendo □ Cubase □ Pro Tools
☒ andere: Audacity, Audition

2. Wie oft arbeitest du mit einer DAW?

□ täglich ☒ mehrmals pro Woche □ etwa einmal pro Woche

□ etwa einmal pro Monat □ seltener als monatlich □ aktuell gar nicht

3. Falls nicht Ableton, hast du schon einmal mit Ableton Live gearbeitet?

□ ja, gelegentlich □ ja, früher / nicht mehr □ nur kurz ausprobiert □ nur zugeschaut ☒ nein

4. Welche Tätigkeiten führst du in deiner DAW hauptsächlich aus?

□ Produktion □ Recording ☒ Sounddesign ☒ Mixing □ Mastering
□ anderes: _____

5. Wie steuerst du deine DAW üblicherweise?

☒ Maus und Tastatur □ MIDI-Controller □ DAW-Controller □ Ableton Push
□ Touchscreen/iPad
□ anderes: _____

1

Teilnehmernummer: 1

Fragebogen

Dieser Fragebogen erfasst deinen Eindruck nach der Nutzung des Controllers. Bitte kreuze die jeweils passende Antwort an. Die offenen Fragen am Ende beantwortest du in den dafür vorgesehenen Zeilen.

Die Angaben werden anonym ausgewertet.

	stimme gar nicht zu	stimme eher nicht zu	stimme eher zu	stimme voll zu
1. Ich empfand die Einführung als ausreichend, um die Aufgaben auszuführen.				X
2. Es fiel mir schwer zu erkennen, was ich gerade steuere.	X			
3. Ich zögerte selten, bevor ich eine Funktion auslöste.			X	
4. Die Seitenstruktur des Controllers verwirrte mich zeitweise.			X	
5. Die Potentiometer steuerten Effekte und Sends präzise.				X
6. Ein einzelnes Potentiometer hätte mir für die Aufgaben ausgereicht.		X		
7. Die Reaktion des Systems wirkte auf mich störend verzögert.	X			

2

Welche Bedienhandlung(en) hast du mit dem Controller verwendet?

Pegel d. Kanäle, Laden u. Bearbeiten v. Effekten, Anlagen Return (Hall), Panning
Ansicht, ~~Cursor~~ cursor-, Loop-Steuerung
Automatisierte Reverb-Sends

Welche Bedienhandlung(en) mit dem Controller hat/haben für dich am besten funktioniert?

schnelles Bearbeiten v. Effekten
alle Fein-Adjustments (mit Maus oft zu grob

Gab es eine oder mehrere Aufgabenstellungen, bei der/denen die Bedienung unklar und/oder umständlich war? Wenn ja, welche?

Mehrfachauswahl zunächst unklar

3

Gab es eine oder mehrere Situationen, in der/denen du lieber Maus und/oder Tastatur benutzt hättest? Warum?

Laden v. Effekten, Suchen mit Tastatur oft schneller

Für welche Arbeitsschritte würdest du den Controller am ehesten einsetzen (z.B. Recording, Sounddesign, Mixing)?

Sounddesign

Automationen

Was würdest du am Controller oder an der Anzeige auf dem Stream Deck verbessern?

Erstellen von Flags durch klicken auf Slot (sofern unbenutzt)

4

Teilnehmernummer: 2

Mündliche Befragung

Dieser Leitfaden ordnet ein, wie regelmäßig und womit die Testperson arbeitet. Die genannten Antwortoptionen sind Anhaltspunkte. Abweichende Angaben werden notiert.

1. Mit welcher DAW oder welchen DAWs arbeitest du?

☐ Ableton Live ☐ Logic ☐ FL Studio ☒ Reaper ☐ Nuendo ☐ Cubase ☐ Pro Tools
☐ andere: _____

2. Wie oft arbeitest du mit einer DAW?

☐ täglich ☒ mehrmals pro Woche ☐ etwa einmal pro Woche

☐ etwa einmal pro Monat ☐ seltener als monatlich ☐ aktuell gar nicht

3. Falls nicht Ableton, hast du schon einmal mit Ableton Live gearbeitet?

☐ ja, gelegentlich ☐ ja, früher / nicht mehr ☐ nur kurz ausprobiert ☒ nur zugeschaut ☐ nein

4. Welche Tätigkeiten führst du in deiner DAW hauptsächlich aus?

☒ Produktion ☒ Recording ☐ Sounddesign ☒ Mixing ☒ Mastering
☐ anderes: _____

5. Wie steuerst du deine DAW üblicherweise?

☒ Maus und Tastatur ☐ MIDI-Controller ☐ DAW-Controller ☐ Ableton Push
☐ Touchscreen/iPad
☐ anderes: _____

Teilnehmernummer: 2

Fragebogen

Dieser Fragebogen erfasst deinen Eindruck nach der Nutzung des Controllers. Bitte kreuze die jeweils passende Antwort an. Die offenen Fragen am Ende beantwortest du in den dafür vorgesehenen Zeilen.

Die Angaben werden anonym ausgewertet.

	stimme gar nicht zu	stimme eher nicht zu	stimme eher zu	stimme voll zu
1. Ich empfand die Einführung als ausreichend, um die Aufgaben auszuführen.				X
2. Es fiel mir schwer zu erkennen, was ich gerade steuere.		X		
3. Ich zögerte selten, bevor ich eine Funktion auslöste.	XX	X		
4. Die Seitenstruktur des Controllers verwirrte mich zeitweise.			X	
5. Die Potentiometer steuerten Effekte und Sends präzise.				X
6. Ein einzelnes Potentiometer hätte mir für die Aufgaben ausgereicht.	X			
7. Die Reaktion des Systems wirkte auf mich störend verzögert.	X			

Welche Bedienhandlung(en) hast du mit dem Controller verwendet?

Track Volumen, Return channel, Fx Steuerung, # Marker / Loop setzen, Panning, Poti Zuweisung,

Welche Bedienhandlung(en) mit dem Controller hat/haben für dich am besten funktioniert?

Track volum, Fx Steuerung, Poti Zuweisung

Gab es eine oder mehrere Aufgabenstellungen, bei der/denen die Bedienung unklar und/oder umständlich war? Wenn ja, welche?

Poti Zuweisung

3

Gab es eine oder mehrere Situationen, in der/denen du lieber Maus und/oder Tastatur benutzt hättest? Warum?

Nein

Für welche Arbeitsschritte würdest du den Controller am ehesten einsetzen (z.B. Recording, Sounddesign, Mixing)?

Sounddesign, Mixing, Recording (vielleicht, müsste ich mehr mit rumprobieren)

Was würdest du am Controller oder an der Anzeige auf dem Stream Deck verbessern?

Infochart (Knopfbelegung, doppeltdrücken o. löschen z.B)

4

Teilnehmernummer: 3

Mündliche Befragung

Dieser Leitfaden ordnet ein, wie regelmäßig und womit die Testperson arbeitet. Die genannten Antwortoptionen sind Anhaltspunkte. Abweichende Angaben werden notiert.

1. Mit welcher DAW oder welchen DAWs arbeitest du?

☒ Ableton Live ☐ Logic ☐ FL Studio ☐ Reaper ☐ Nuendo ☐ Cubase ☐ Pro Tools
☒ andere: MPC One, Euvorack

2. Wie oft arbeitest du mit einer DAW?

☐ täglich ☐ mehrmals pro Woche ☒ etwa einmal pro Woche

☐ etwa einmal pro Monat ☐ seltener als monatlich ☐ aktuell gar nicht

3. Falls nicht Ableton, hast du schon einmal mit Ableton Live gearbeitet?

☐ ja, gelegentlich ☐ ja, früher / nicht mehr ☐ nur kurz ausprobiert ☐ nur zugeschaut ☐ nein

4. Welche Tätigkeiten führst du in deiner DAW hauptsächlich aus?

☒ Produktion ☐ Recording ☒ Sounddesign ☒ Mixing ☐ Mastering
☐ anderes: _____

5. Wie steuerst du deine DAW üblicherweise?

☒ Maus und Tastatur ☐ MIDI-Controller ☐ DAW-Controller ☐ Ableton Push
☐ Touchscreen/iPad
☐ anderes: _____

Teilnehmernummer: 3

Fragebogen

Dieser Fragebogen erfasst deinen Eindruck nach der Nutzung des Controllers. Bitte kreuze die jeweils passende Antwort an. Die offenen Fragen am Ende beantwortest du in den dafür vorgesehenen Zeilen.

Die Angaben werden anonym ausgewertet.

	stimme gar nicht zu	stimme eher nicht zu	stimme eher zu	stimme voll zu
1. Ich empfand die Einführung als ausreichend, um die Aufgaben auszuführen.				X
2. Es fiel mir schwer zu erkennen, was ich gerade steuere.		X		
3. Ich zögerte selten, bevor ich eine Funktion auslöste.				X
4. Die Seitenstruktur des Controllers verwirrte mich zeitweise.		X		
5. Die Potentiometer steuerten Effekte und Sends präzise.			X	
6. Ein einzelnes Potentiometer hätte mir für die Aufgaben ausgereicht.	X			
7. Die Reaktion des Systems wirkte auf mich störend verzögert.	X			

2

Welche Bedienhandlung(en) hast du mit dem Controller verwendet?

Allgemeine Navigation in der Timeline und Bereiche markieren/loopen. Volume/Pan + Poti Mapping. Effekte auswählen/bedienen, Return Spuren für mehrere Tracks, Automationsfahrt aufgenommen (usw)

Welche Bedienhandlung(en) mit dem Controller hat/haben für dich am besten funktioniert?

Allgemeine Navigation u. Mapping sehr intuitiv, individuelles Poti Mapping sehr cool und durch 2 Potis sehr variabel nutzbar

Gab es eine oder mehrere Aufgabenstellungen, bei der/denen die Bedienung unklar und/oder umständlich war? Wenn ja, welche?

Return Track hinzufügen zunächst an falscher Stelle gesucht. Verwirrend, dass Returns im Mixer sichtbar sind aber nicht anlegbar.

3

Gab es eine oder mehrere Situationen, in der/denen du lieber Maus und/oder Tastatur benutzt hättest? Warum?

- Wechsel Solo/Arm (oft) da nur im Hauptmenü möglich)
- Loop setzen auf Tastaturen schneller, (Anfang/Endpunkt setzen cool aber dauert)

Für welche Arbeitsschritte würdest du den Controller am ehesten einsetzen (z.B. Recording, Sounddesign, Mixing)?

- Automationsfahrten
- Poti Mapping und Effektnutzung

Was würdest du am Controller oder an der Anzeige auf dem Stream Deck verbessern?

- Solo/arm im Hauptmenü wechsel.
- Anzeigenstringenz (Return Channel im Mixer)
- Feinheitsgrade beim Poti eventuell einstellbar?
- Quantisierungsstufen per Knopfdruck einstellen

4

Eigenständigkeitserklärung

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

Entwicklung eines haptischen Controllers zur drahtlosen Steuerung von Ableton Live

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