

ulrich.audio

ConcreteJungle

a simplified machine to capture the embedded world



SICHERHEITSHINWEISE

Hinweise zur Vermeidung von Feuer, elektrischen Schlägen oder Verletzung von Personen

Über die Warnung- und Vorsicht-Hinweise

	Diese Warnungen sollen den Anwender auf die Gefahren hinweisen, die bei unsachgemäßem Gebrauch des Gerätes bestehen.
	Dieses Zeichen wird verwendet, um den Anwender auf das Risiko von Verletzungen oder Materialschäden hinzuweisen, die bei unsachgemäßem Gebrauch des Gerätes entstehen können. - Die oben genannten Faktoren beziehen sich sowohl auf häusliches Inventar als auch auf Haustiere.

Über die Symbole

	Das Symbol macht den Anwender auf wichtige Hinweise und Warnungen aufmerksam. Das im Dreieck befindliche Zeichen gibt eine genaue Definition der Bedeutung (Beispiel: Das Symbol weist auf allgemeine Gefahren hin).
	Das Symbol weist auf Dinge hin, die zu unterlassen sind. Das Symbol im Kreis definiert dieses Verbot näher (Beispiel: das Zeichen links besagt, dass das Gerät nicht geöffnet bzw. auseinandergenommen werden darf).
	Das Symbol weist auf Dinge hin, die zu tun sind. Das Symbol links definiert diese Aktion näher (Beispiel: Das Zeichen links besagt, dass der Netzstecker des Gerätes aus der Steckdose zu entfernen ist).

BEACHTEN SIE AUCH DIESE HINWEISE

WARNING

- Lesen sie sorgfältig die Bedienungsanleitung, bevor sie das Gerät benutzen.
- Das Gerät und das Netzteil dürfen nicht geöffnet oder in irgendeiner Weise verändert werden.
- Nehmen sie keine Reparaturversuche vor. Überlassen sie dieses einem qualifizierten Techniker.
- Vermeiden Sie Umgebungen mit:
- Extremen Temperaturen (z.b. direkte Sonneneinstrahlung, direkte Nähe eines Heizkörpers etc.)
- Stellen sie das Gerät immer auf eine feste Unterlage die nicht wackelt.
- Verwenden sie nur den für das Gerät vorgesehenen Netzadapter. Die Benutzung eines anderen Netzadapters kann einen Kurzschluss, einen elektrischen Schlag bzw. die Beschädigung oder Zerstörung des Gerätes zur Folge haben.
- Vermeiden Sie Beschädigungen des Netzkabels. Knicken Sie es nicht, treten sie dicht darauf und stellen sie keine schweren Gegenstände auf das Kabel. Ein beschädigtes Kabel birgt nicht nur die Gefahr eines elektrischen Schläges, sondern kann auch einen Brand auslösen. Verwenden sie deshalb niemals ein beschädigtes Netzkabel.

WARNING

- Achten sie auf eine moderate Lautstärke, auch wenn sie einen Kopfhörer benutzen. Wenn sie meinen, dass ihr Gehör beeinträchtigt ist, suchen sie einen Gehörspezialisten auf.
- Achten sie darauf, dass keine Flüssigkeiten und Gegenstände in das Gehäuse gelangen.
- Schalten sie das Gerät sofort aus, trennen sie es von der Stromversorgung, wenn:
 - Der Netzadapter beschädigt ist.
 - Ein Gegenstand oder Flüssigkeit in das Gehäuse gelangt ist.
 - Das Gerät nass geworden ist(z.B. durch Regenwasser)
 - Das Gerät nicht normal funktioniert

Benachrichtigen sie umgehend ihren Vertragspartner über den Zustand des Gerätes.

WARNUNG

- In Haushalten mit Kindern sollte ein Erwachsener solange für Aufsicht sorgen, bis das betreffende Kind das Gerät unter Beachtung aller Sicherheitsvorschriften zu bedienen weiß.
- Bewahren sie das Gerät vor heftigen Stößen und lassen sie es nicht fallen.
- Vermeiden sie es, das Gerät mit vielen anderen Geräten zusammen an der selben Steckdose zu betreiben. Ganz besonders Vorsichtig sollten sie bei der Verwendung von Verlängerungen mit Mehrfachsteckdosen sein: Der Gesamtverbrauch aller angeschlossenen Geräte darf niemals die in Watt oder Ampère angebende zulässige Höchstbelastung überschreiten. Eine übermäßige Belastung durch zu hohen Stromfluss kann das Kabel bis zum Schmelzen erhitzen.
- Bevor Sie das Gerät im Ausland benutzen befragen sie ihren Vertragspartner.
- Die Batterien dürfen nicht neu aufgeladen, erhitzt oder auseinandergerissen werden. Werfen sie Batterien nie ins Feuer bzw. ins Wasser.

VORSICHT

- Das Gerät und der Netzadapter müssen immer ausreichend belüftet sein.
- Ziehen sie nie am Kabel, sondern fassen sie es beim aus- und einstecken des Netzkabels immer am Stecker an.
- Wenn das Gerät längere Zeit nicht benutzt wird, trennen sie den Netzadapter vom Stromnetz.
- Achten sie darauf, dass die Kabel nicht durcheinandergeraten. Verlegen sie die Kabel ausserdem so, dass Kinder nicht an sie herankommen können.

VORSICHT

- Berühren sie das Netzkabel bzw. das Netzteil niemals mit nassen Händen.
- Bevor sie das Gerät bewegen, um es an einem anderen Platz zu aufzustellen, sollten sie die Verbindung Netzkabel>>Gerät, sowie alle anderen Kabelverbindungen trennen.
- Wenn sie das Gerät reinigen wollen, schalten sie es vorher aus und trennen sie es vom Netz.
- Bei Gewitter sollten sie das Gerät vom Stromnetz nehmen.
- Bei falscher Handhabung können Batterien Explodieren bzw. auslaufen und dadurch Schäden oder Verletzungen verursachen. Beachten sie bitte folgende Sicherheitshinweise:
 - Installieren sie die Batterien vorschriftsmäßig und achten sie auf die Polarität.
 - Mischen sie keine alten mit neuen Batterien und verwenden sie für ein Batterie-Set immer nur identische Batterietypen.
 - Wenn das Gerät längere Zeit nicht benutzt wird, nehmen sie die Batterien heraus.
- Wir bedanken uns bei Ihnen, dass Sie dafür Sorge tragen werden, alte Batterien umweltgerecht dem gesetzlich vorgeschriebenen Sondermüll zuzuführen.

Inhalt

Sicherheitshinweise

Inhalt

über ConcreteJungle

Vorder- und Rückseite

Bedienelemente

 Vorderseite

 Hinterseite

Schnellstart

 Vorbereitung

 Erstes Sample

Technische Dokumentation

 Anschlüsse Verbindungen

 Blockdiagramme

 Stromlaufpläne

 Programmcode

 Processing

 Arduino

 Python

 PureData

Eigenschaften

Wir bedanken uns für Ihre Entscheidung zum ConcreteJungle Sampler

Um alle Funktionen dieses Gerätes genau kennenzulernen, lesen Sie die Anleitung bitte ganz durch und bewahren Sie diese an einem sicheren Ort auf.

FREESOUND.ORG Application Programming Interface

Der ConcreteJungle verfügt über eine Schnittstelle zum Freesound.org Archiv. Hiermit eröffnet sich Ihnen die gesamte Welt der Freesound.org Datenbanken. Lassen Sie sich von der unendlichen Auswahl an hochwertigen Sounds verführen.

Alle Sounds für GEO-TECHNO, GEO-GOA, GEO-WAVE, GEO-HOUSE

Ganz egal welches Genre Sie anstreben, der ConcreteJungle wird sie zwingen sich von ihrer kulturellen Engstirnigkeit zu distanzieren und mit neuen Geräuschen zu experimentieren.

Kontroll-Möglichkeiten für Live Performances

Die Geräusche können in Echtzeit über verschiedene Bedienregler während des Playback verändert werden. Die Beschränkung auf wenige Parameter wird Sie an eine völlig neue Art von Arrangement heranführen.

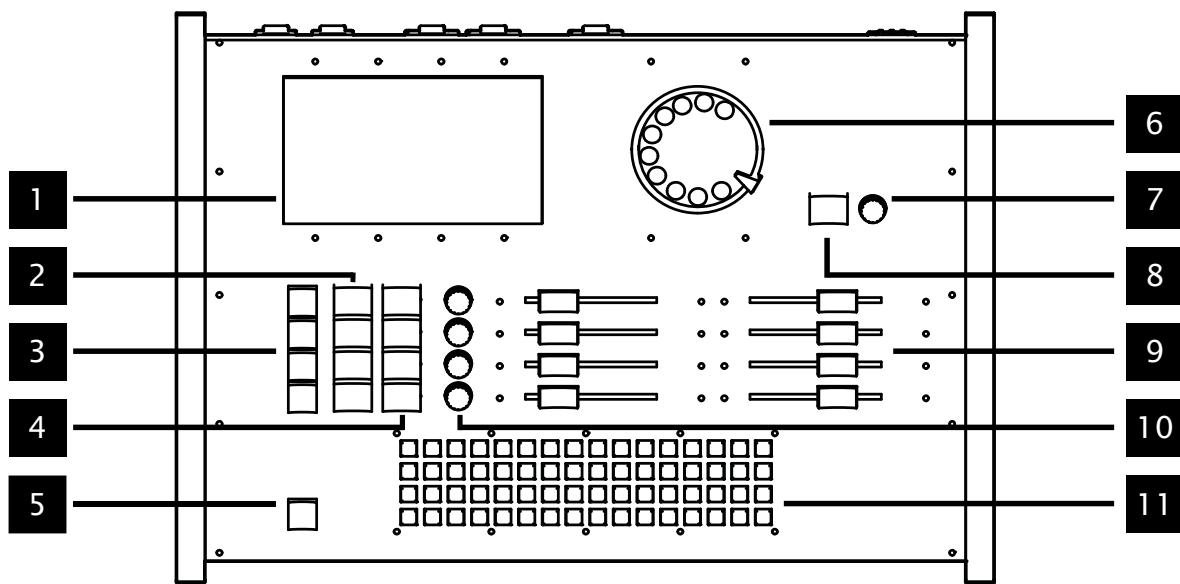
Interagieren sie mit der Außenwelt

Der ConcreteJungle verfügt über einen CLOCK Ausgang der es Ihnen spielend leicht ermöglicht andere analoge Geräte mit Ihrem Gerät zu synchronisieren.

Unbegrenzte klangliche Möglichkeiten

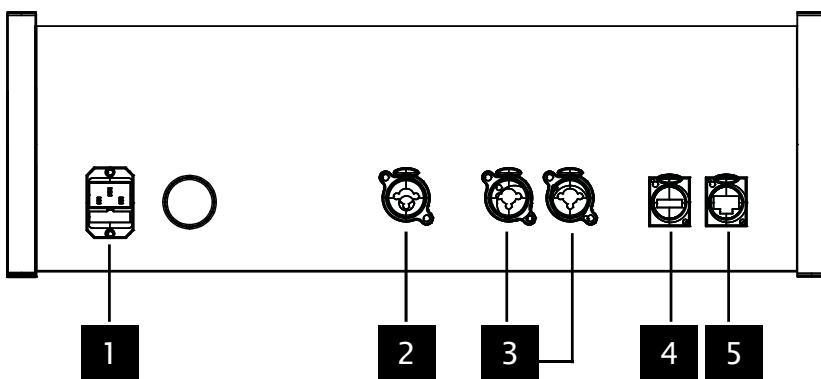
Dank der durchdachten Bearbeitungsmöglichkeiten lassen sich perkussive Elemente, sowie Melodien und Drones erzeugen. Mit dem Tempo Regler steuern Sie stufenlos den Schweiß-Fluss Ihrer Audienz.

2. Vorder- und Rückseite



1	Sample Selection Display	s.11	6	Sample Selection Dial	s.16
2	Load Sample Buttons	s.12	7	Tempo Knob	s.17
3	Prelisten Buttons	s.13	8	START/STOP Button	s.18
4	LOOP Buttons	s.14	9	In- and Out-Point Fader	s.19
5	Listen Button	s.15	10	Sample Tempo Knobs	s.20
			11	Sequencer	s.21

Front Panel



1	Power Connector & Switch	s.11	2	CLOCK Output	s.16
3	Audio Output L+R	s.12	4	USB Connector	s.17
5	RJ45 Network Connector	s.12			

2.1 Bedienelemente

2.1.1 Sample-Auswahl Display

Das Sample-Auswahl Display dient zur Eingabe der Koordinaten für die Sample Auswahl. Um einem Punkt auf der dargestellten Karte wird bei anhaltender Berührung ein Radius um den Ausgangspunkt erstellt, der mit der Länge der Berührung größer wird. Der Radius definiert den Umkreis um die Ausgangskoordinate in dem nach Samples gesucht wird in Kilometern.

2.1.2 LOAD-Taster

Mit den LOAD-Tastern wird ein Sample auf die entsprechende Spur geladen. Nachdem Sie eine SOURCE ausgewählt haben und die Samples auf das Gerät geladen sind, können Sie ein Sample aus der SOURCE einem Track zuweisen. Halten sie hierfür den LOAD-Taster der entsprechenden Spur gedrückt und wählen Sie auf der Sample-Auswahl-Wählscheibe die Nummer des Samples, dass auf die Spur geladen werden soll. Das geladene Sample ist sofort verwendbar.

2.1.3 Vorhör-Taster (TRACK)

Auf jede Spur kann ein Sample geladen werden. Dieses kann vom Anwender verändert werden (siehe 2.1.9,2.1.10). Um die Auswirkungen der eingegebenen Veränderungen zu überprüfen, verfügt jede Spur über einen eigenen Vorhör-Taster. Hier kann jederzeit das aktuell eingestellte Sample mit den eingebrachten Veränderungen überprüft werden. Drücken Sie hierfür den Vorhör-Taster der Spur, dessen Sample Sie überprüfen möchten. (Diese Funktion steht im LOOP-Modus nicht zur Verfügung).

2.1.4 LOOP-Schalter

Die LOOP-Schalter schalten für ein bereits auf eine Spur geladenes Sample zwischen den Funktionen Einmal Abspielen (OneShot) oder Endlosschleife (Loop) um. Ist der LOOP-Schalter einer Spur inaktiv, wird die Wiedergabe des Samples über den Sequenzer gesteuert (Siehe 2.1.11,2.1.9). Ist der LOOP-Schalter aktiv geschaltet (Kontrollleuchte EIN), wird das Sample in einer Endlosschleife abgespielt. Im LOOP-Modus spielt die entsprechende Spur unabhängig von der eingestellten Sequenz oder ihrer Geschwindigkeit.

2.1.5 Vorhör-Taster

Der Vorhör-Taster dient zur Überprüfung der SOURCE und zur Suche nach Samples für die aktuelle Komposition. Nachdem Sie eine SOURCE erstellt (Siehe SOURCE erstellen, siehe 2.1.2) haben und diese auf das Gerät geladen ist, können Sie die verfügbaren Samples vorhören. Halten Sie hierfür die Vorhör-Taste gedrückt und wählen Sie auf der Sample-Auswahl-Wählscheibe die Nummer des Samples, welches Sie vorhören möchten.

2.1.6 Sample-Auswahl-Wählscheibe

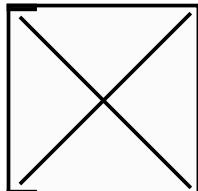
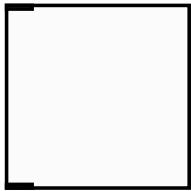
Die Sample-Auswahl-Wählscheibe ist das zentrale Auswahl-Element auf Ihrer Bedienoberfläche. Es regelt die Übergabe eines Samples aus der SOURCE in den STOCK und ermöglicht dem Anwender das Vorhören der Samples aus der SOURCE sowie das Überprüfen der Samples auf einer Spur (siehe 2.1.3,2.1.5,2.1.1).

2.1.7 TEMPO-Drehregler

Der Tempo-Drehregler bestimmt die Abspiel-Geschwindigkeit einer eingestellten Sequenz (siehe 2.1.11, 2.1.8).

2.1.8 START/STOP Schalter

Der START/STOP Schalter schaltet das Abspielen einer Sequenz ein- oder aus. Der Status der Sequenz wird über die Kontrollleuchte im Schalter angezeigt.



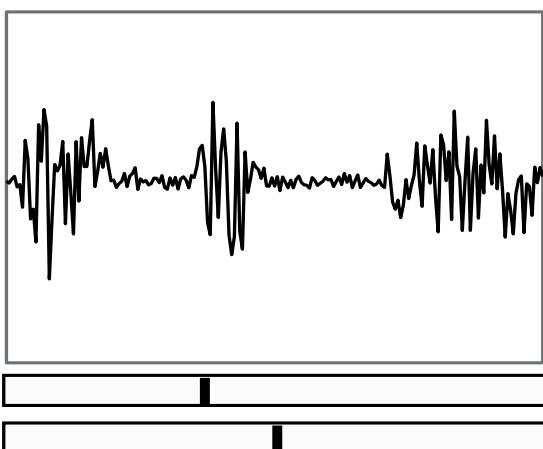
2.1.9 In- und Out-Punkt Schieberegler

Jede Spur verfügt über je einen In- und einen Out-Punkt Schieberegler.

Sie bestimmen die Abspielpositionen innerhalb des Samples, dass in die Spur geladen wurde. Der Einstellbereich wird in Relation zur Sample-Länge bestimmt und ermöglicht bei beliebiger Sample-Länge immer die Einstellung der Abspielpositionen über die gesamte Länge bis zum einzelnen Bit.

Werden In-Punkt- (links) und Out-Punkt-Regler (rechts) gegeneinander über den jeweiligen Mittelpunkt verstellt, wird das geladene Sample rückwärts abgespielt.

Wird ein neues Sample auf eine Spur geladen, werden die Schieberegler mittels Motorsteuerung auf ihre jeweiligen Nullpunkte zurückgestellt. Hierdurch wird ein ungewolltes Vermischen von eingestellten Parametern vermieden.



2.1.10 Sample-Tempo-Drehregler

Die Sample-Tempo-Drehregler stehen für jede Spur zur Verfügung. Mit ihnen hat der Anwender die Möglichkeit, die Abspielgeschwindigkeit des Samples einer Spur zu kontrollieren. Hierbei wird die Tonhöhe stets linear zur Geschwindigkeitsänderung mit verändert. Je schneller ein Sample abgespielt wird, desto höher wird seine Tonhöhe (pitch).

Durch die Änderung der Tonhöhe eines Samples, kann seine Charakteristik grundlegend verändert werden.

2.1.11 Sequenzer

Das zentrale Steuerelement ihres Gerätes, stellt der Sequenzer dar.

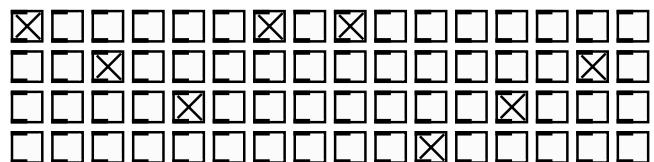
Mit ihm werden Komposition und Rhythmus ihrer musikalischen Zusammenstellung definiert.

Das verbaute Sequenzer-Modul bietet 16 Schritte (steps) für jede der vier verfügbaren Spuren. Ist ein Sample auf eine Spur geladen, kann mittels Tasteneingabe auf dem Sequenzer seine Abspielposition im gegebenen Takt bestimmt werden.

Das Tastenfeld des Sequenzers besteht aus (4x16) hintergrundbeleuchteten Elastomer-Gummi-Tastern.

Ein aktiver STEP gibt seinen Status über die eingebauten Kontrollleuchten wieder.

Die Taktabfolge der Sequenz kann jederzeit verändert werden; unabhängig davon ob die Sequenz spielt oder ruht.



2.2 Bedienelemente

2.2.1 Netzanschlussstecker & Ein/Aus Schalter

Der Netzanschlussstecker versorgt das Gerät mit elektrischem Strom. Fügen Sie den mitgelieferten Steckverbinder in die dafür vorgesehene Kupplungsdoose ein.

Benutzen sie nur Steckverbinder die der Norm IEC 60320 C13 gerecht werden und die sich technisch sowie mechanisch in einwandfreiem Zustand befinden.
(Achtung Lebensgefahr!).

Mit dem EIN/AUS-Taster wird das Gerät Ein-beziehungsweise Ausgeschaltet. Halten Sie hierfür den EIN/AUS-Taster >2 Sekunden gedrückt. Nach einer kurzen Initialisierungsphase ist der Sampler einsatzbereit.

2.2.2 CLOCK-Ausgang

Der CLOCK Ausgang ermöglicht die Synchronisation mit weiterer externer Hardware. Das CLOCK-Signal gibt einen elektrischen Puls in der eingestellten Taktrate des Sequenzers aus. Sofern sie über andere Hardware verfügen die mittels CLOCK-Signal gesteuert werden kann (Synthesizer, Drum-Computer, Keyboards o.ä.), können Sie diese im selben Takt laufen lassen wie das Basisgerät.

2.2.3 Audio Ausgang L+R

Der Audioausgang gibt das erzeugte Audiosignal an Ihren Verstärker oder Lautsprecher aus. Verbinden Sie die 6,35mm-Klinken/XLR-Kombibuchsen mit Ihrem Ausgabegerät oder Verstärker. Das Signal wird in Stereo ausgegeben. Für jede Seite des Signals steht eine Buchse zur Verfügung. Schließen Sie ihr Klinken- oder XLR-Kabel für die linke Seite an den Eingang für die linke Seite an Ihrem Ausgabegerät an. Verfahren Sie auf gleiche Weise für die rechte Seite. Die Ausgangsimpedanz liegt bei 400Ω bei einem maximalen Ausgangspegel von 2dBV.

2.2.4 USB/Netzwerk Verbinder

Die beiden Anschlussbuchsen USB/A und RJ45 dienen der Wartung und der Konfiguration des Gerätes.

Vor der ersten Benutzung muss der integrierte WLAN Adapter mit dem lokalen Drahtlosnetzwerk verbunden werden (Lesen Sie hierzu den Abschnitt „Erste Benutzung“).

Benutzen Sie die Verbinder nur für den vorgesehenen Zweck. Unsachgemäße Bedienung kann zur Beschädigung oder Zerstörung des Gerätes führen.

3. Schnellstart

3.1.1 Vorbereitungen

Verbinden Sie das mitgelieferte Netzkabel mit der Anschlussbuchse auf der Rückseite des Gerätes (siehe 2.2.1).

Stecken Sie das dafür vorgesehene Ende des Netzkabels in eine Steckdose die den Anforderungen für den Betrieb des Gerätes entspricht.

3.1.2 Netzwerkverbindung einrichten

Um den ConcreteJungle Sampler verwenden zu können, muss das Gerät über einen Internetzugang verfügen.

Im Folgenden wird die Einrichtung der Netzwerkverbindung erklärt:

1. Nachdem Schritt 3.1.1 abgeschlossen ist, warten Sie, bis das Gerät einsatzbereit ist.
2. Schließen Sie eine externe USB-Tastatur (nicht im Lieferumfang enthalten) an die USB-Anschlussbuchse auf der Rückseite des Gerätes an (siehe 2.2.4).
3. Drücken Sie die ESC-Taste.
4. Drücken Sie auf der Tastatur Strg+Alt+N
5. Im Dialogfeld die lokale Netzwerkverbindung auswählen und die persönlichen Zugangsdaten eingeben.
6. Schließen Sie das Dialogfeld „Netzwerk-einstellungen“ und starten Sie das Gerät neu.

SSID :

KEY :

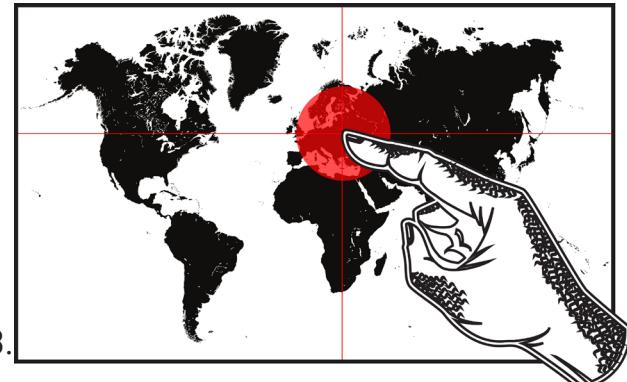
3.1.3 Starten Sie den ConcreteJungle

Starten Sie das Gerät in dem Sie den Ein/Aus-Taster >2 Sekunden gedrückt halten. (siehe 2.2.1)

3.2.1 Eine Sample suche initiieren

Nachdem Sie die Schritte aus Kapitel 3.1 ausgeführt haben, ist das Gerät betriebsbereit. Beginnen Sie nun mit Ihrer ersten Sample-Suche. Berühren Sie dafür das Sample-Auswahl Display mit Ihrem Zeigefinger und halten Sie den Finger so lange auf dem Display, bis der sich einstellende Radius Ihren Vorstellungen entspricht (siehe 2.1.1).

Sobald Sie den Finger vom Display nehmen, beginnt das Gerät damit die Samples aus der ausgewählten Region auf das Gerät zu laden. Sobald der Ladevorgang abgeschlossen ist, wird auf dem Display eine Benachrichtigung angezeigt.



Im nächsten Schritt können Sie Ihr erstes Sample auf eine der vier verfügbaren Spuren laden. Wählen Sie hierfür eine Spur aus, halten Sie den LOAD-Taster der entsprechenden Spur gedrückt und wählen Sie auf der Sample-Auswahl-Wählscheibe die Nummer des Samples, dass Sie auf die Spur laden möchten (siehe 2.1.6).

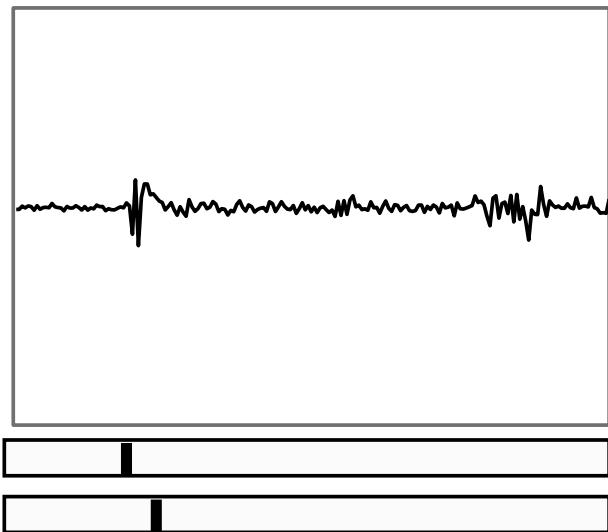
Selbstverständlich können Sie vorher die zur Verfügung stehenden Samples vorhören indem Sie die Schritte aus Kapitel 1.1.5 ausführen.

3.2.3 Ein Sample bearbeiten

Als Nächsten Schritt können Sie Ihr Sample trimmen oder die Geschwindigkeit und Tonhöhe verändern.

Stellen Sie hierfür nach Belieben die entsprechenden Regler ein
(siehe 2.1.9, 2.1.10).

Sie können ihr Ergebnis jederzeit über die Vorhörtaster (TRACK) überprüfen
(siehe 2.1.3).

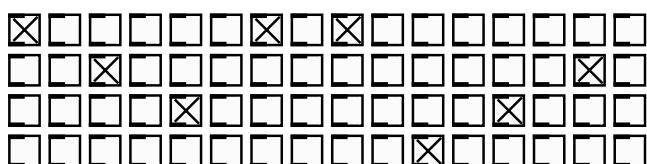


3.2.4 Eine Sequenz eingeben

Um die, im vorangegangen Schritt, erzielten Ergebnisse rhythmisch anzuordnen, können Sie nun eine Sequenz in das Tastenfeld des 4-Spur Sequenzers eingeben (siehe 2.1.11).

Nach der Eingabe der Sequenz, können sie diese mit dem Start/Stop Schalter der Sequenzersteuerung starten (siehe 2.1.8).

Die Geschwindigkeit der abgespielten Sequenz können Sie nach belieben mit dem Tempo-Drehregler variieren (siehe 2.1.7).



4. Appendix

4.1 Begriffserklärungen

Nachfolgend werden einige Begriffe erklärt, denen Sie bei der Arbeit mit dem ConcreteJungle und in dieser Anleitung begegnen werden.

Lesen Sie diesen Abschnitt sorgfältig und prägen Sie sich das gebrauchte Vokabular sorgsam ein. Es wird Ihnen helfen Ihr neues Gerät besser zu verstehen und sich unter gegebenen Umständen mit unserem fachlich geschultem Service-personal zu verständigen.

4.1.1 Sample (Musik)

Ein Sample ist ein digitalisiertes analoges Audiosignal.

Hierbei werden dem analogen Audiosignal über einen A/D-Wandler Ausschnitte (Samples) entnommen und gespeichert ([https://de.wikipedia.org/wiki/Sampling_\(Musik\)#Audiosample](https://de.wikipedia.org/wiki/Sampling_(Musik)#Audiosample), Stand: 2.4.2020)

4.1.2 Trimmen

Trimmen bezeichnet das Beschneiden eines Samples (siehe 4.1.1).

Dabei wird der Beginn oder das Ende des Abspielbereichs eines Samples verändert. Hierdurch können einzelne Bereiche eines Samples isoliert werden.

4.1.3 In- Out-Point

In- und Out-Point eines Samples beschreiben die, im Trimmprozess (siehe 4.1.2) definierten Abspielpunkte eines Samples.

Wird der Out-Point in Relation zur Sample-Länge vor dem In-Point gesetzt, spielt der Klang erzeuger das Sample rückwärts ab.

4.1.4 Pitch

Der Pitch beschreibt die Tonhöhe einer Audioaufnahme (Sample). Durch ein Verändern der Abspielgeschwindigkeit wird die Tönhöhe einer Aufnahme proportional zur Geschwindigkeit höher oder tiefer.

4.1.5 SOURCE

Als SOURCE wird im Betrieb des ConcreteJungle die Auswahl von Samples bezeichnet, die bei der Sample-Suche um eine eingegebene Koordinate gefunden und gespeichert wird.

4.1.6 STOCK

Der STOCK beschreibt die Auswahl von Samples, die der Anwender aus der SOURCE in die bearbeitungsebene seiner Komposition übernommen hat.

4.2.1 Sample (Signalverarbeitung)

Unter Abtastung (englisch sampling) wird in der Signalverarbeitung die Registrierung von Messwerten zu diskreten, meist äquidistanten Zeitpunkten verstanden. Aus einem zeitkontinuierlichen Signal wird so ein zeitdiskretes Signal gewonnen.

Bei mehrkanaligen Signalen ergibt jede Abtastung ein „Sample“ aus mehreren Abtastwerten. Die Zahl der Samples pro Sekunde wird Abtastrate genannt. ([https://de.wikipedia.org/wiki/Abtastung_\(Signalverarbeitung\)](https://de.wikipedia.org/wiki/Abtastung_(Signalverarbeitung)), Stand: 2.4.2020)

4.3.1 Sequenz/Sequenzer

Eine Sequenz ist eine Abfolge von Abspielbefehlen die sowohl rhythmische als auch zeitliche Informationen beinhaltet.

Der Sequenzer gibt die gespeicherten Befehlspunkte (Steps) an den Klangerzeuger weiter, der die gespeicherten Samples mit den eingestellten Parametern abspielt.

4.3.2 Track

Ein Track ist ein Bestandteil des Sequenzers. Der im ConcreteJungle verbaute Sequenzer beinhaltet vier Tracks mit je 16 Schritten.

Jedem Track ist ein spezifisches Sample zugewiesen.

4.3.3 Komposition

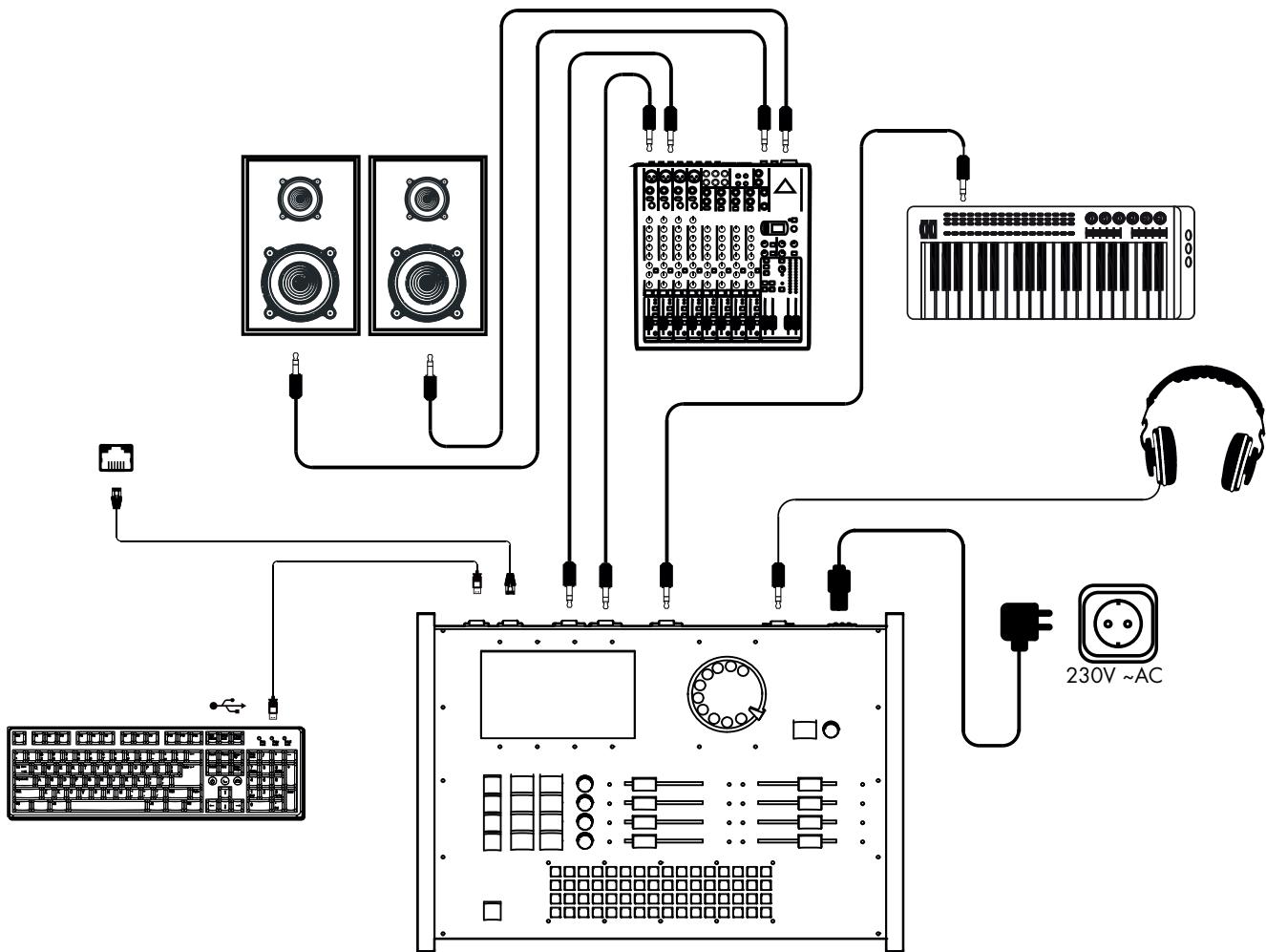
Eine Komposition setzt sich aus den vier verschiedenen Spuren des Sequenzers zusammen und beinhaltet die Summe aller erzeugten Klanginformationen.

4.4.1 Geo-Tag

Ein Geo-Tag ist ein Meta-Element, das über die geographische Position zahlreicher Medien, wie Fotos, Videos oder Webseiten informiert. Dabei werden üblicherweise Längen- und Breitengrad gespeichert oder zusätzliche Informationen wie Land, Höhe oder Ortsnamen sind häufig enthalten. (<https://de.wikipedia.org/wiki/Geo-Tag>, Stand 2.4.2020)

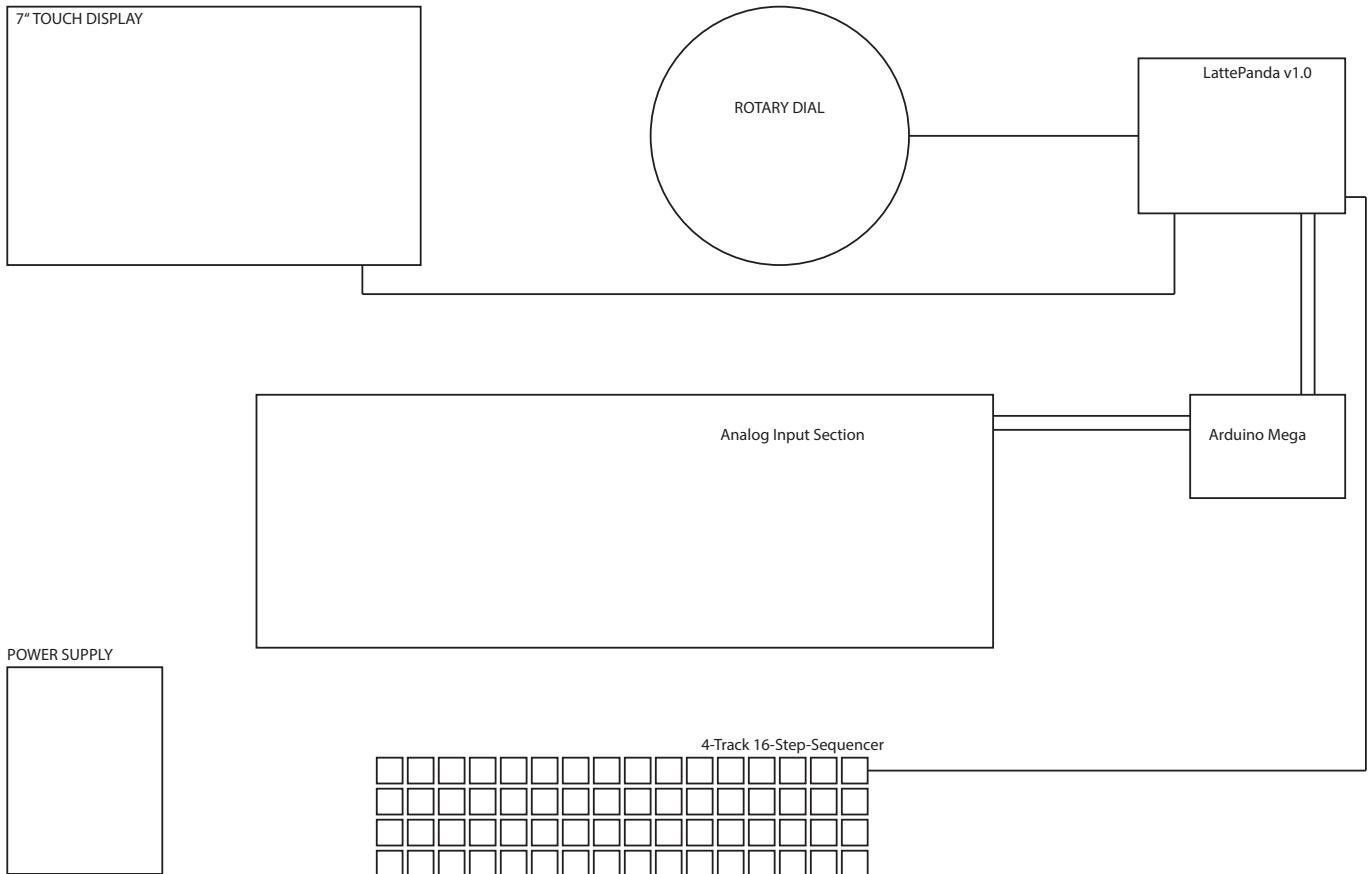
5. Technische Dokumentation

4.1.1 Anschlüsse / Verbindungen

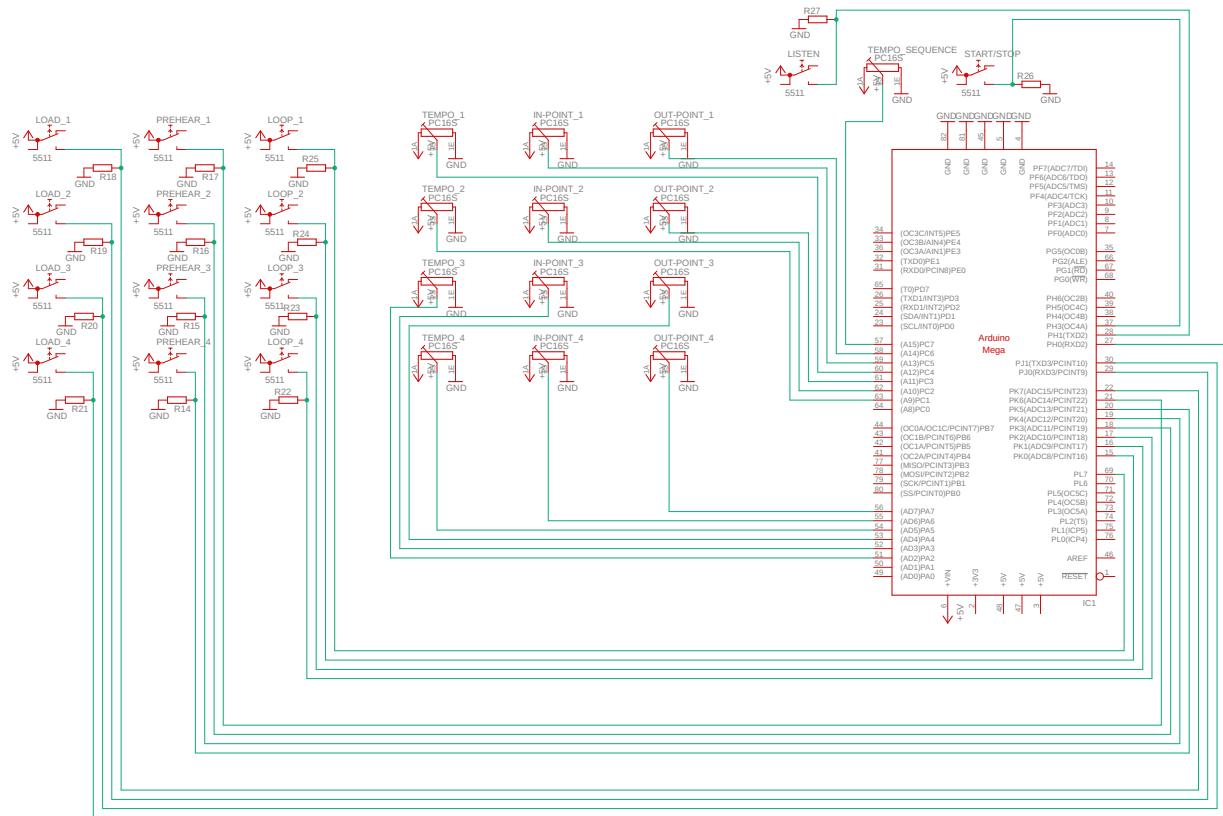


Folgen sie der Übersicht um Ihren Sampler mit Peripheriegeräten und dem Stromnetz zu verbinden.
Für eine genaue Beschreibung der erforderlichen Verbindungen, lesen Sie den Abschnitt Anschlüsse.

4.1.2 Blockdiagramm



4.1.3 Stromlaufplan „Analog Input Section“



4.1.4 Programmcode

Im Folgenden finden Sie eine Übersicht der Programme, die auf den einzelnen Modulen des ConcreteJungle-Samplers laufen. Sie dienen lediglich der Information.

Verändern Sie kein Programm auf Ihrem Gerät. Bei Problemen im Zusammenhang mit den Programmcodes wenden Sie sich an Ihren Vertragspartner.

4.1.4.1 Sample-Auswahl-Display

Processing 3.5.3

```
1 //import hypermedia.net.*;
2 import oscP5.*;
3 import netP5.*;
4
5 OscP5 oscP5;
6 NetAddress myRemoteLocation;
7
8 PShape world;
9 int sizeX=1800;
10 int sizeY= 1000;
11
12 int lange;
13 int breite;
14 int radius;
15
16 int val;
17 int y;
18 int radiusC=1;
19 int m=255;
20
21 void setup() {
22   size(1024, 608);
23
24   myRemoteLocation = new NetAddress("127.0.0.1", 12000);
25
26   stroke(255, 0, 0, m--);
27   fill(255, 0, 0, 200);
28
29   world = loadShape("worldmap.svg");
30 }
31 void draw() {
32   background(102);
33   shape(world, 0, 0, width, height); // Draw at coordinate (110, 90) at size 100 x 100
34
35   //fadenkreuz zeichnen
36   line(0, mouseY, width, mouseY);
37   line(mouseX, 0, mouseX, height);
38   //aiming point
39   circle();
40 }
41
42 void circle() {
43   //aiming point wächst bei gedrückter maustaste
44   if (mousePressed == true) {
45     ellipse(mouseX, mouseY, radius++, radius++);
46     fill(255, 0, 0, m--);
47   } else {
48     radius=1;
49     m=255;
50   }
51 }
52
53 void mouseReleased()
```

```
53 void mouseReleased()
54 {
55     float breite = mouseX;
56     float lange = mouseY;
57     breite=(map(breite, 0, 1024, -180, 180)+8);
58     lange=(map(lange, 0, 600, 90, -90)+25);
59     radius=(radius * 40);
60     println("Länge: ", breite);
61     println("Breite: ", lange);
62     println("DIAMETER: ", radius);
63
64     String CoordinateWidth = str(breite);
65     String CoordinateLength = str(lange);
66     String content = str(radius);
67     String message = ( CoordinateWidth +" "+ CoordinateLength +" "+ content ); // the mes-
       to send
68
69     OscMessage myOscMessage = new OscMessage("/incommingWLR");
70     myOscMessage.add(breite);
71     myOscMessage.add(lange);
72     myOscMessage.add(radius);
73     OscP5.flush(myOscMessage, myRemoteLocation);
74
75     println (message);
76 }
```

4.2.2 16-Step Sequencer Bedienelement // Arduino 1.8.5

```
1  ****
2  This is a test example for the Adafruit Trellis w/HT16K33
3
4  Designed specifically to work with the Adafruit Trellis
5  ----> https://www.adafruit.com/products/1616
6  ----> https://www.adafruit.com/products/1611
7
8  These displays use I2C to communicate, 2 pins are required to
9  interface
10 Adafruit invests time and resources providing this open source code,
11 please support Adafruit and open-source hardware by purchasing
12 products from Adafruit!
13
14 Written by Limor Fried/Ladyada for Adafruit Industries.
15 MIT license, all text above must be included in any redistribution
16 ****
17
18 #include <Wire.h>
19 #include "Adafruit_Trellis.h"
20
21 ****
22 This example shows reading buttons and setting/clearing buttons in a loop
23 "momentary" mode has the LED light up only when a button is pressed
24 "latching" mode lets you turn the LED on/off when pressed
25
26 Up to 8 matrices can be used but this example will show 4 or 1
27 ****
28
29 #define MOMENTARY 0
30 #define LATCHING 1
31 // set the mode here
32 #define MODE LATCHING
33
34
35 Adafruit_Trellis matrix0 = Adafruit_Trellis();
36 Adafruit_Trellis matrix1 = Adafruit_Trellis();
37 Adafruit_Trellis matrix2 = Adafruit_Trellis();
38 Adafruit_Trellis matrix3 = Adafruit_Trellis();
39 // you can add another 4, up to 8
40
41 // Just one
42 //Adafruit_TrellisSet trellis = Adafruit_TrellisSet(&matrix0);
43 // or use the below to select 4, up to 8 can be passed in
44 Adafruit_TrellisSet trellis = Adafruit_TrellisSet(&matrix0, &matrix1, &matrix2,
&matrix3);
45
46 // set to however many you're working with here, up to 8
47 #define NUMTRELLIS 4
48
49 #define numKeys (NUMTRELLIS * 16)
50
51
52 // Connect Trellis Vin to 5V and Ground to ground.
53 // Connect the INT wire to pin #A2 (can change later!)
54 #define INTPIN A2
55 // Connect I2C SDA pin to your Arduino SDA line
56 // Connect I2C SCL pin to your Arduino SCL line
57 // All Trellises share the SDA, SCL and INT pin!
58 // Even 8 tiles use only 3 wires max
59
60
61 void setup() {
62   Serial.begin(9600);
63   // Serial.println("Trellis Demo");
64
65   // INT pin requires a pullup
66   pinMode(INTPIN, INPUT);
67   digitalWrite(INTPIN, HIGH);
68
```

```

69 // begin() with the addresses of each panel in order
70 // I find it easiest if the addresses are in order
71 trellis.begin(0x71,0x72,0x73,0x74); // only one
72 // trellis.begin(0x70, 0x71, 0x72, 0x73); // or four!
73 trellis.setBrightness(0);
74 // light up all the LEDs in order
75 for (uint8_t i=0; i<numKeys; i++) {
76     trellis.setLED(i);
77     trellis.writeDisplay();
78     delay(50);
79 }
80 // then turn them off
81 for (uint8_t i=0; i<numKeys; i++) {
82     trellis.clrLED(i);
83     trellis.writeDisplay();
84     delay(50);
85 }
86 }
87
88
89 void loop() {
90     delay(30); // 30ms delay is required, dont remove me!
91
92     if (MODE == MOMENTARY) {
93         // If a button was just pressed or released...
94         if (trellis.readSwitches()) {
95             // go through every button
96             for (uint8_t i=0; i<numKeys; i++) {
97                 // if it was pressed, turn it on
98                 if (trellis.justPressed(i)) {
99                     Serial.print("v"); Serial.println(i);
100                    trellis.setLED(i);
101                }
102                // if it was released, turn it off
103                if (trellis.justReleased(i)) {
104                    Serial.print("^"); Serial.println(i);
105                    trellis.clrLED(i);
106                }
107            }
108            // tell the trellis to set the LEDs we requested
109            trellis.writeDisplay();
110        }
111    }
112
113    if (MODE == LATCHING) {
114        // If a button was just pressed or released...
115        if (trellis.readSwitches()) {
116            // go through every button
117            for (uint8_t i=0; i<numKeys; i++) {
118                // if it was pressed...
119                if (trellis.justPressed(i)) {
120                    /*Serial.print("v");*/ Serial.write(i);
121                    // Alternate the LED
122                    if (trellis.isLED(i))
123                        trellis.clrLED(i);
124                    else
125                        trellis.setLED(i);
126                }
127            }
128            // tell the trellis to set the LEDs we requested
129            trellis.writeDisplay();
130        }
131    }
132 }

```

4.2.3 Waehlscheibe // Arduino 1.8.5

```
1 int needToPrint = 0;
2 int count;
3 int in = 2;
4 int lastState = LOW;
5 int trueState = LOW;
6 long lastStateChangeTime = 0;
7 int cleared = 0;
8
9 // constants
10
11 int dialHasFinishedRotatingAfterMs = 100;
12 int debounceDelay = 10;
13
14 void setup()
15 {
16 Serial.begin(9600);
17 pinMode(in, INPUT);
18 }
19
20 void loop()
21 {
22 int reading = digitalRead(in);
23
24 if ((millis() - lastStateChangeTime) > dialHasFinishedRotatingAfterMs) {
25 // the dial isn't dialed, or has just finished dialing
26 if (needToPrint) {
27 // if it's just finished being dialed, we need to send the number down the serial
28 // line and reset the count. We mod the count by 10 because '0' will send 10 pulses.
29 Serial.println(count % 10, DEC);
30 needToPrint = 0;
31 count = 0;
32 cleared = 0;
33 }
34 }
35
36 if (reading != lastState) {
37 lastStateChangeTime = millis();
38 }
39 if ((millis() - lastStateChangeTime) > debounceDelay) {
40 // debounce - this happens once it's stabilized
41 if (reading != trueState) {
42 // this means that the switch has either just gone from closed->open or vice versa.
43 trueState = reading;
44 if (trueState == HIGH) {
45 // increment the count of pulses if it's gone high.
46 count++;
47 needToPrint = 1; // we'll need to print this number (once the dial has finished rotating)
48 }
49 }
50 }
51 lastState = reading;
52 }
```

4.2.3 Analog Input Elements // Arduino 1.8.5

```
2 #include
3
4
5 Adafruit_MotorShield AFMSbot(0x61); // Rightmost jumper closed
6 Adafruit_MotorShield AFMStop(0x60); // Default address, no jumpers
7
8
9 Adafruit_DCMotor *mFader6 = AFMSbot.getMotor(1);
10 Adafruit_DCMotor *mFader8 = AFMSbot.getMotor(2);
11 Adafruit_DCMotor *mFader1 = AFMSbot.getMotor(3);
12 Adafruit_DCMotor *mFader3 = AFMSbot.getMotor(4);
13
14 Adafruit_DCMotor *mFader2 = AFMStop.getMotor(1);
15 Adafruit_DCMotor *mFader4 = AFMStop.getMotor(2);
16 Adafruit_DCMotor *mFader5 = AFMStop.getMotor(3);
17 Adafruit_DCMotor *mFader7 = AFMStop.getMotor(4);
18
19
20 int motorSpeed = 250;
21 int Dword;
22
23 //BUTTON INITIALIZE
24 int play = 27;
25 int prehear = 29;
26
27 int listen1 = 45;
28 int listen2 = 43;
29 int listen3 = 41;
30 int listen4 = 39;
31
32 int load1 = 53;
33 int load2 = 51;
34 int load3 = 49;
35 int load4 = 47;
36
37 int loop1 = 37;
38 int loop2 = 35;
39 int loop3 = 33;
40 int loop4 = 31;
41
42 //POT INITIALIZE
43 int tempo = A7;
44 int speed1 = A0;
45 int speed2 = A1;
46 int speed3 = A2;
47 int speed4 = A3;
48
49
50
51 void setup() {
52
53     Serial.begin(9600);           // set up Serial library at 9600 bps
54
55     AFMSbot.begin(); // create with the default frequency 1.6KHz
56     AFMStop.begin();
57
58     // Set the speed to start, from 0 (off) to 255 (max speed)
59     mFader1->setSpeed(motorSpeed);
60     mFader2->setSpeed(motorSpeed);
61     mFader3->setSpeed(motorSpeed);
62     mFader4->setSpeed(motorSpeed);
63     mFader5->setSpeed(motorSpeed);
64     mFader6->setSpeed(motorSpeed);
65     mFader7->setSpeed(motorSpeed);
66     mFader8->setSpeed(motorSpeed);
67
68     //Define Buttons as Inputs..
69     pinMode(play, INPUT);
70     pinMode(prehear, INPUT);
71
72     pinMode(listen1, INPUT);
73     pinMode(listen2, INPUT);
74     pinMode(listen3, INPUT);
75     pinMode(listen4, INPUT);
76
77     pinMode(load1, INPUT);
```

```

79  pinMode(load2, INPUT);
80  pinMode(load3, INPUT);
81  pinMode(load4, INPUT);
82
83  pinMode(loop1, INPUT);
84  pinMode(loop2, INPUT);
85  pinMode(loop3, INPUT);
86  pinMode(loop4, INPUT);
87
88 }
89
90 void loop() {
91   serialListen();
92
93   doAnythingToDo();
94
95 }
96
97 void resetTrack1() {
98   int fader1 = analogRead(A15);
99   int fader2 = analogRead(A11);
100  if (fader1 >= 1) {
101    mFader1->run(BACKWARD);
102  }
103  delay(200);
104  mFader1->run(RELEASE);
105
106  if (fader2 >= 1) {
107    mFader2->run(BACKWARD);
108  }
109  delay(200);  mFader2->run(RELEASE);
110 }
111
112 void resetTrack2() {
113   int fader3 = analogRead(A14);
114   int fader4 = analogRead(A10);
115
116   if (fader3 >= 1) {
117     mFader3->run(FORWARD);
118   }
119   delay(220);
120   mFader3->run(RELEASE);
121
122   if (fader4 >= 1) {
123     mFader4->run(FORWARD);
124   }
125   delay(220);  mFader4->run(RELEASE);
126 }
127
128 void resetTrack3() {
129   int fader5 = analogRead(A13);
130   int fader6 = analogRead(A9);
131
132   if (fader5 >= 1) {
133     mFader5->run(BACKWARD);
134   }
135   delay(250);
136   mFader5->run(RELEASE);
137
138   if (fader6 >= 1) {
139     mFader6->run(BACKWARD);
140   }
141   delay(250);
142   mFader6->run(RELEASE);
143 }
144
145 void resetTrack4() {
146   int fader7 = analogRead(A12);
147   int fader8 = analogRead(A8);
148   if (fader7 >= 1) {
149     mFader7->run(FORWARD);
150   }
151   delay(250);
152   mFader7->run(RELEASE);
153
154   if (fader8 >= 1) {
155     mFader8->run(FORWARD);
156   }
157   delay(250);
158   mFader8->run(RELEASE);
159 }
160
161 void serialListen() {
162   Dword = Serial.read();
163
164

```

```

165  if (Dword == '1') {
166    resetTrack1();
167  }
168
169  if (Dword == '2') {
170    resetTrack2();
171  }
172
173  if (Dword == '3') {
174    resetTrack3();
175  }
176
177  if (Dword == '4') {
178    resetTrack4();
179  }
180
181 }
182
183
184 void doAnythingToDo() {
185   int BSplay = digitalRead(play);
186   int BSprehear = digitalRead(prehear);
187
188   int BSlisten1 = digitalRead(listen1);
189   int BSlisten2 = digitalRead(listen2);
190   int BSlisten3 = digitalRead(listen3);
191   int BSlisten4 = digitalRead(listen4);
192
193   int BSload1 = digitalRead(load1);
194   int BSload2 = digitalRead(load2);
195   int BSload3 = digitalRead(load3);
196   int BSload4 = digitalRead(load4);
197
198   int BSloop1 = digitalRead(loop1);
199   int BSloop2 = digitalRead(loop2);
200   int BSloop3 = digitalRead(loop3);
201   int BSloop4 = digitalRead(loop4);
202
203   int PVtempo = analogRead(tempo);
204   int PVspeed1 = analogRead(speed1);
205   int PVspeed2 = analogRead(speed2);
206   int PVspeed3 = analogRead(speed3);
207   int PVspeed4 = analogRead(speed4);
208
209 //TRACK1
210   int fader1 = analogRead(A15);
211   int fader2 = analogRead(A11);
212 //TRACK2
213   int fader3 = analogRead(A14);
214   int fader4 = analogRead(A10);
215 //TRACK3
216   int fader5 = analogRead(A13);
217   int fader6 = analogRead(A9);
218 //TRACK4
219   int fader7 = analogRead(A12);
220   int fader8 = analogRead(A8);
221
222   Serial.print(BSplay);
223   Serial.print(" ");
224   Serial.print(BSprehear);
225   Serial.print(" ");
226
227   Serial.print(BSlisten1);
228   Serial.print(" ");
229   Serial.print(BSlisten2);
230   Serial.print(" ");
231   Serial.print(BSlisten3);
232   Serial.print(" ");
233   Serial.print(BSlisten4);
234   Serial.print(" ");
235
236   Serial.print(BSload1);
237   Serial.print(" ");
238   Serial.print(BSload2);
239   Serial.print(" ");
240   Serial.print(BSload3);
241   Serial.print(" ");
242   Serial.print(BSload4);
243   Serial.print(" ");
244
245   Serial.print(BSloop1);
246   Serial.print(" ");
247   Serial.print(BSloop2);
248   Serial.print(" ");
249   Serial.print(BSloop3);
250   Serial.print(" ");

```

```
251     Serial.print(BSloop4);
252     Serial.print(" ");
253
254     Serial.print(PVtempo);
255     Serial.print(" ");
256     Serial.print(PVspeed1);
257     Serial.print(" ");
258     Serial.print(PVsspeed2);
259     Serial.print(" ");
260     Serial.print(PVsspeed3);
261     Serial.print(" ");
262     Serial.print(PVsspeed4);
263     Serial.print(" ");
264     Serial.print(fader1);
265     Serial.print(" ");
266     Serial.print(fader2);
267     Serial.print(" ");
268     Serial.print(fader3);
269     Serial.print(" ");
270     Serial.print(fader4);
271     Serial.print(" ");
272     Serial.print(fader5);
273     Serial.print(" ");
274     Serial.print(fader6);
275     Serial.print(" ");
276     Serial.print(fader7);
277     Serial.print(" ");
278     Serial.println(fader8);
279
280 }
```

4.2.3 Freesound SoundFetcher // Python 3.6

```
1. # fs sampler
2. # version: 0.1
3. # Dieses Programm dient dazu anhand von spezifischen Parametern aus der online Library Free Sound Dateien
4. # zu selektieren und diese in ein Verzeichnis lokal runter zu laden
5.
6. from __future__ import print_function
7. import freesound
8. import os
9. import sys
10. from oauthlib.oauth2 import BackendApplicationClient
11. from requests_oauthlib import OAuth2Session
12. import hashlib
13. import random
14. import time
15. import subprocess
16.
17. # OSC
18. from osc4py3.as_eventloop import *
19. from osc4py3 import oscmethod as osm
20.
21. class config():
22.     """
23.         Start here to edit your config of the program
24.     """
25.     APIKEY="Ld15pPetuu7VMVOGEzgkrvTp23Iaip10UmuszK44"
26.     OAUTHTOKEN="VegqtgKdHqbR0KwcElFha5FvC2vhQ"
27.     FETCHDIRNAME = "fetchedSounds"
28.     COUNT = 10 #Maximale Anzahl der Sounds (-1 fÃ¼r unbegrenzt)
29.     PAGESIZE = 300 #Ergebnisse pro Abruf von Freesounds INT (Seitenbasiert, Seite n kann spezifiziert
30.     werden)
31.     DEBUG = True
32.     MINSOUNDDURATION = 1
33.     MAXSOUNDDURATION = 60
34.
35. class fsFetcher():
36.     def __init__(self):
37.         self.fsClient = freesound.FreesoundClient()
38.         self.fsClient.set_token(config.APIKEY)
39.         #programm status (True = weiter arbeiten, False = Fehler)
40.         self.state = False
41.
42.     def createDirs(self):
43.         self.path_name = os.path.join(os.getcwd(), config.FETCHDIRNAME)
44.         if config.DEBUG:
45.             print ("directory path:")
46.             print(self.path_name)
47.         try:
48.             if config.DEBUG:
49.                 print("creating dir for previews...")
50.             os.mkdir(self.path_name)
51.         except (FileExistsError):
52.             if config.DEBUG:
53.                 print ("dir already created, skipping")
54.         except:
55.             if config.DEBUG:
56.                 print ("Cannot create folder: "+self.path_name+" ! Cannot continue")
57.             return False
58.         try:
59.             if config.DEBUG:
60.                 print ("creating dir for wave files...")
61.             os.mkdir(self.path_name+'/wav')
62.         except (FileExistsError):
63.             if config.DEBUG:
64.                 print ("dir already created, skipping")
65.         except:
66.             if config.DEBUG:
67.                 print ("cannot create folder: "+self.path_name+"/wav ! Cannot continue")
68.             return False
69.         self.state = True
70.         return True
```

```

70.     def md5(self, fname):
71.         hash_md5 = hashlib.md5()
72.         with open(fname, "rb") as f:
73.             for chunk in iter(lambda: f.read(4096), b ""):
74.                 hash_md5.update(chunk)
75.         return hash_md5.hexdigest()
76.
77.     def selectSounds(self, minDuration=config.MINSOUNDDURATION, maxDuration=config.MAXSOUNDDURATION, geo=[-10,52,4000]):
78.         """
79.             Diese Methode wÃ¤hlt Sounds aufgrund folgender Parameter aus der Freesound Library
80.             Parameter: minimale Dauer, maximale Dauer, Geotags: Breitengrad, LÃ¤ngengrad, Entfernung (Radius)
81.             als INT
82.             Text Search Request:
83.             >>> sounds = c.text_search(
84.             >>>     query="dubstep", filter="tag:loop", fields="id,name,url"
85.             >>> )
86.             >>> for snd in sounds: print snd.name
87.
88.             Geotag Filter:
89.                 #filter={!geofilt sfield=geotag pt=<LATITUDE>,<LONGITUDE> d=<MAX_DISTANCE_IN_KM>}
90.             """
91.             if not (self.state):
92.                 return False
93.             if config.DEBUG:
94.                 print ("fetching sound data from freesounds")
95.                 print ("geotags:")
96.                 print (geo)
97.             soundGeoTagging = geo
98.             start = time.time()
99.             queryFilter = "{{!geofilt sfield=geotag pt={0},{1} d={2}}}".format(geo[0],geo[1],geo[2])
100.            #queryFilter = "type:wav {!geofilt sfield=geotag pt=13,52 d=2000}"
101.            queryFields ="id,name,duration,md5,type,previews"
102.            sounds = self.fsClient.text_search(filter=queryFilter,fields=queryFields,
103.                                         page_size=config.PAGESIZE)
104.            stop = time.time()
105.            if config.DEBUG:
106.                print ("dauer fÃ¼r freesounds abruf: ")
107.                print (stop-start)
108.            return self.filterByDuration(sounds, minDuration, maxDuration)
109.
110.        def downloadSounds(self, soundsObject):
111.            if not (self.state):
112.                return False
113.            #self.fsClient.set_token(config.OAUTHTOKEN, "oauth")
114.
115.            Download Sound Files
116.            Erwartet ein Sound Objekt mit einer Liste von Sounds
117.            """
118.            i = 0
119.            for sound in soundsObject:
120.                if (i >= 0) & (i < config.COUNT):
121.                    self.nameFileByIndex(sound, i)
122.                    #filename = "sound_"+str(i)+".wav"
123.                    #sound.retrieve_preview(self.path_name, name=filename)
124.                    i += 1
125.                else:
126.                    return
127.            return
128.        def nameFileByIndex(self, soundObject, i):
129.            if not (self.state):
130.                return False
131.            filename = "sound_"+str(i)
132.            if config.DEBUG:
133.                print("\t\tDownloading:", soundObject.name)
134.                print("as: "+filename)
135.            soundObject.retrieve_preview(self.path_name, name=filename)
136.
137.            Loglevel ffmpeg
138.                -loglevel [repeat+]loglevel | -v [repeat+]loglevel

```

```

138.         Set the logging level used by the library.
139.         â€˜quiet, -8â€™
140.         Show nothing at all; be silent.
141.         â€˜panic, 0â€™
142.         Only show fatal errors which could lead the process to crash, such as an assertion failure.
This is not currently used for anything.
143.         â€˜fatal, 8â€™
144.         Only show fatal errors. These are errors after which the process absolutely cannot continue.
145.         â€˜error, 16â€™
146.         Show all errors, including ones which can be recovered from.
147.         â€˜warning, 24â€™
148.         Show all warnings and errors. Any message related to possibly incorrect or unexpected events
will be shown.
149.         â€˜info, 32â€™
150.         Show informative messages during processing. This is in addition to warnings and errors. This
is the default value.
151.         â€˜verbose, 40â€™
152.         Same as info, except more verbose.
153.         â€˜debug, 48â€™
154.         Show everything, including debugging information.
155.         â€˜trace, 56â€™
156.         """
157.         subprocess.call(['ffmpeg', '-v', 'warning', '-y', '-i',
self.path_name+'/'+filename, self.path_name+'/wav/'+filename+'.wav'])
158.         return
159.
160.     def nameFileByName(self, soundObject):
161.         if not (self.state):
162.             return False
163.         fullfilepath = self.path_name+"/"+soundObject.name
164.         if (os.path.isfile(fullfilepath)):
165.             if config.DEBUG:
166.                 print ("dateiname vorhanden: "+soundObject.name)
167.             else:
168.                 if config.DEBUG:
169.                     print ("datei muss geladen werden:")
170.                     print("\t\tDownloading:", soundObject.name)
171. #if sound.name.endswith(sound.type):
172.         filename = soundObject.name
173.         soundObject.retrieve_preview(self.path_name, name=filename)
174.     #else:
175.     #    filename = "%s.%s" % (sound.name, sound.type)
176.     #    sound.retrieve_preview(self.path_name, name=filename)
177.         return
178.
179.     def filterByDuration(self, soundsObject, minDuration, maxDuration):
180.         if not (self.state):
181.             return False
182.         sounds = soundsObject
183.         soundList = []
184.         tmp = time.time()
185.         for sound in sounds:
186.             soundList += [sound]
187.         random.shuffle(soundList)
188.         filteredObjects = []
189.         i = 0
190.         if config.DEBUG:
191.             print ("dauer fÃ¼r shuffle: ")
192.             print (time.time()-tmp)
193.             print ("Preselected Sounds:")
194.             print (soundList)
195.         for sound in soundList:
196.             if (i >= 0) & (i < config.COUNT):
197.                 if (int(sound.duration) >= minDuration) & (int(sound.duration) <= maxDuration):
198.                     filteredObjects += [sound]
199.                     i += 1
200.         if config.DEBUG:
201.             print ("selected sounds after filtering:")
202.             print (filteredObjects)
203.         return filteredObjects
204.

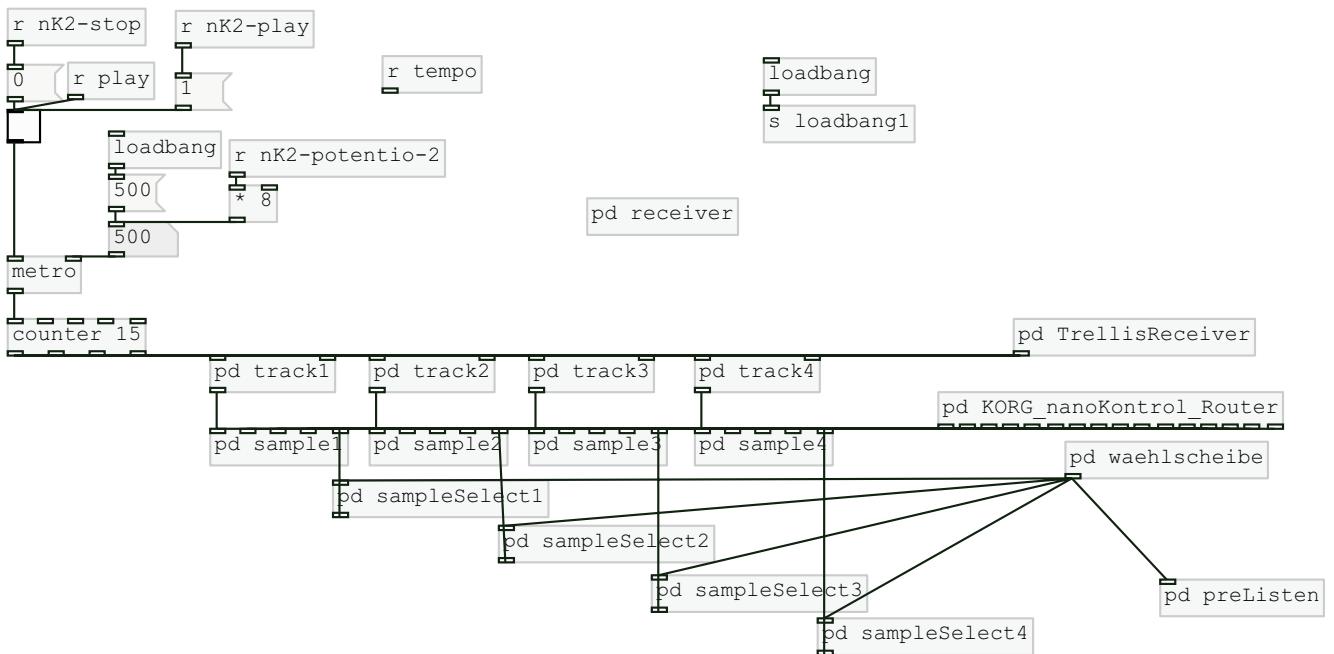
```

```

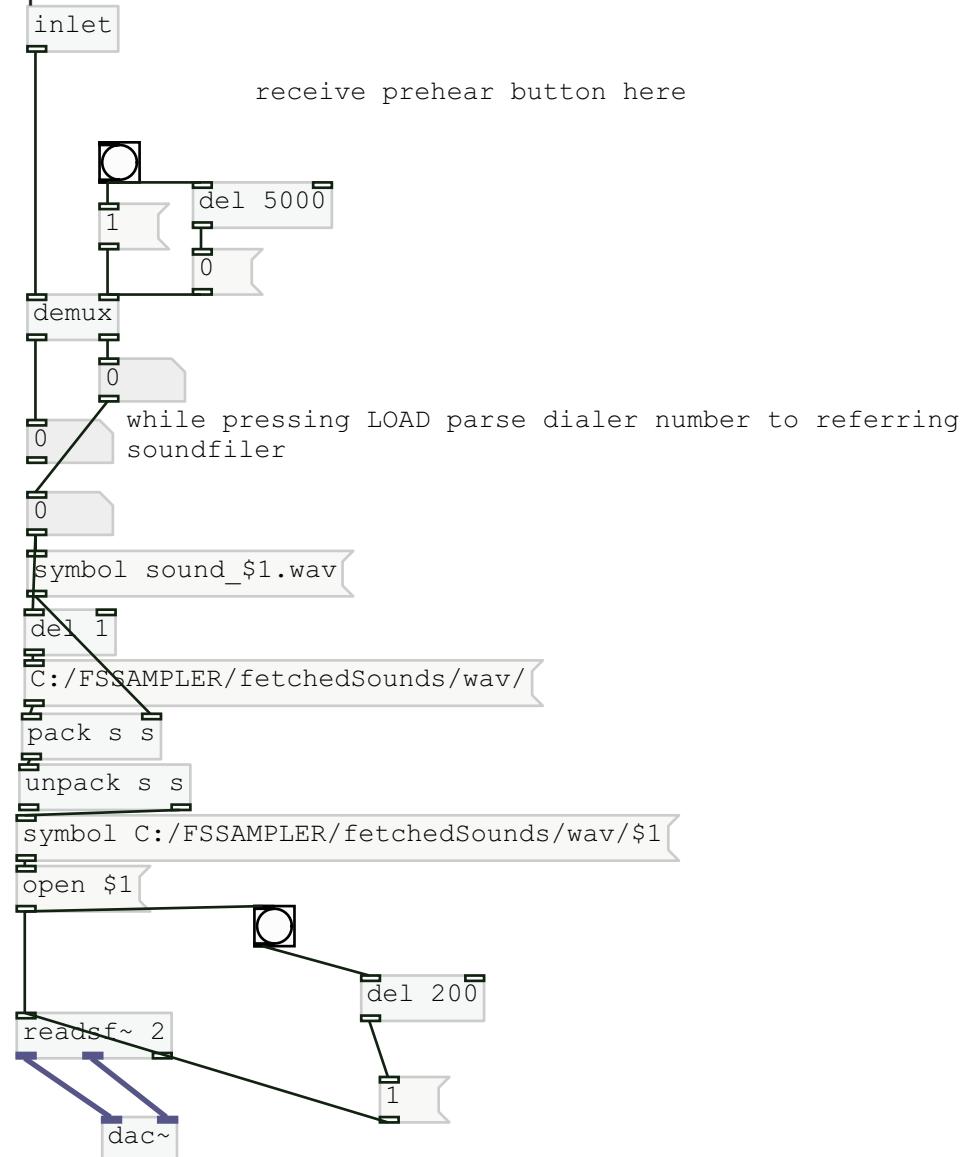
205. class OSCListener():
206.     def __init__(self):
207.         self.serverip = "127.0.0.1"
208.         self.port = 12000
209.         #Programmaufruf
210.         self.appStart = time.time()
211.         self.soundFetcher = fsFetcher()
212.         self.soundFetcher.createDirs()
213.     def handlerForWLR(self, x, y, z):
214.         # Will receive message data unpacked in x,y,z
215.         # Koordignatenaufruf, Download
216.         sounds = self.soundFetcher.selectSounds(geo=[x,y,z])
217.         download = self.soundFetcher.downloadSounds(sounds)
218.
219.
220.     def startup(self):
221.         # Start the system.
222.         if config.DEBUG:
223.             print("starting up Server...")
224.             osc_startup()
225.
226.         # Make server channels to receive packets.
227.         osc_udp_server(self.serverip, self.port, "aservername")
228.
229.         # Associate Python functions with message address patterns, using default
230.         # argument scheme OSCARG_DATAUNPACK.
231.         osc_method("/incommingWLR*", self.handlerForWLR)
232.         if config.DEBUG:
233.             print("listening...")
234.
235.     def shutdown(self):
236.         osc_terminate()
237.
238.     def listenLoop(self):
239.         # Periodically call osc4py3 processing method in your event Loop.
240.         finished = False
241.         while not finished:
242.             try:
243.                 osc_process()
244.             except KeyboardInterrupt:
245.                 finished = True
246.                 osc_terminate()
247.                 raise
248.
249.         # Properly close the system.
250.         self.shutdown()
251.
252.
253.     server = OSCListener()
254.     server.startup()
255.     server.listenLoop()
256.
257.     appStop = time.time()
258.     print ("Programmdauer: ")
259.     print (appStop - appStart)

```

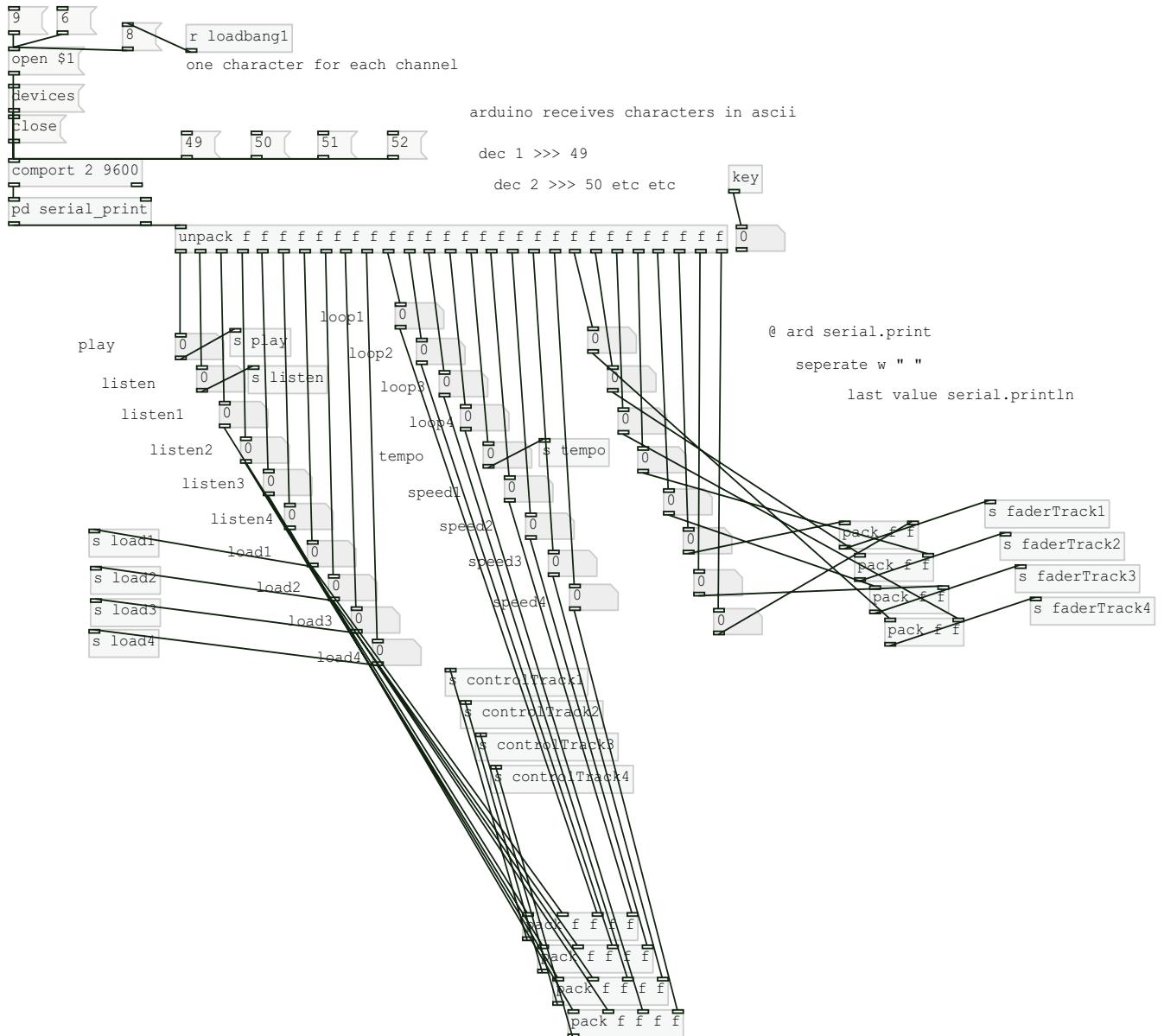
4.2.4 ConcreteJungleSampler_Main_01 // puredata-extended 0.43.4



pd preListen

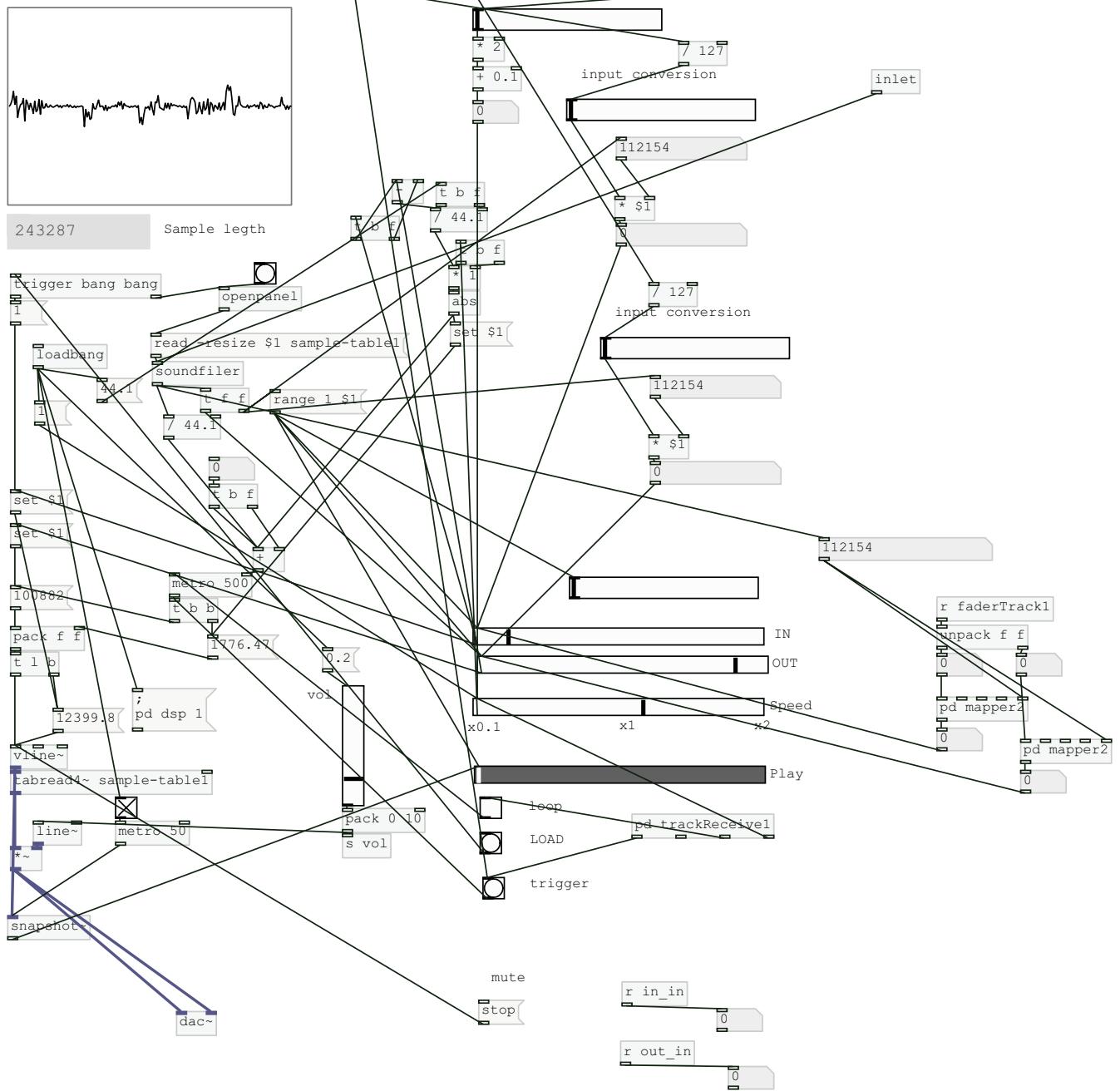


pd receiver

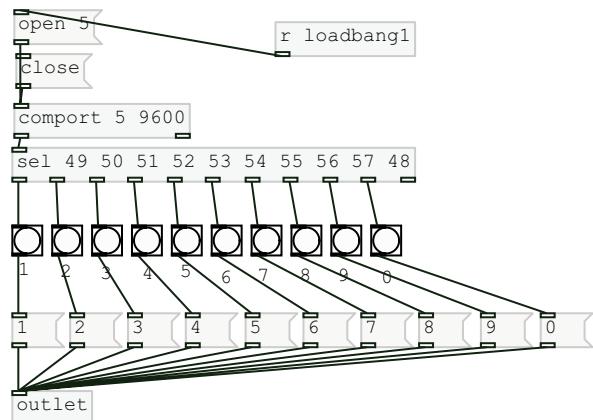


pd sample1

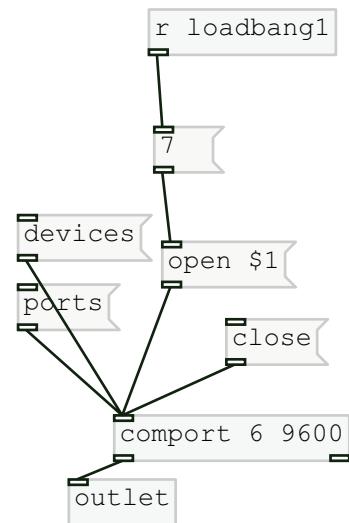
sample-table1



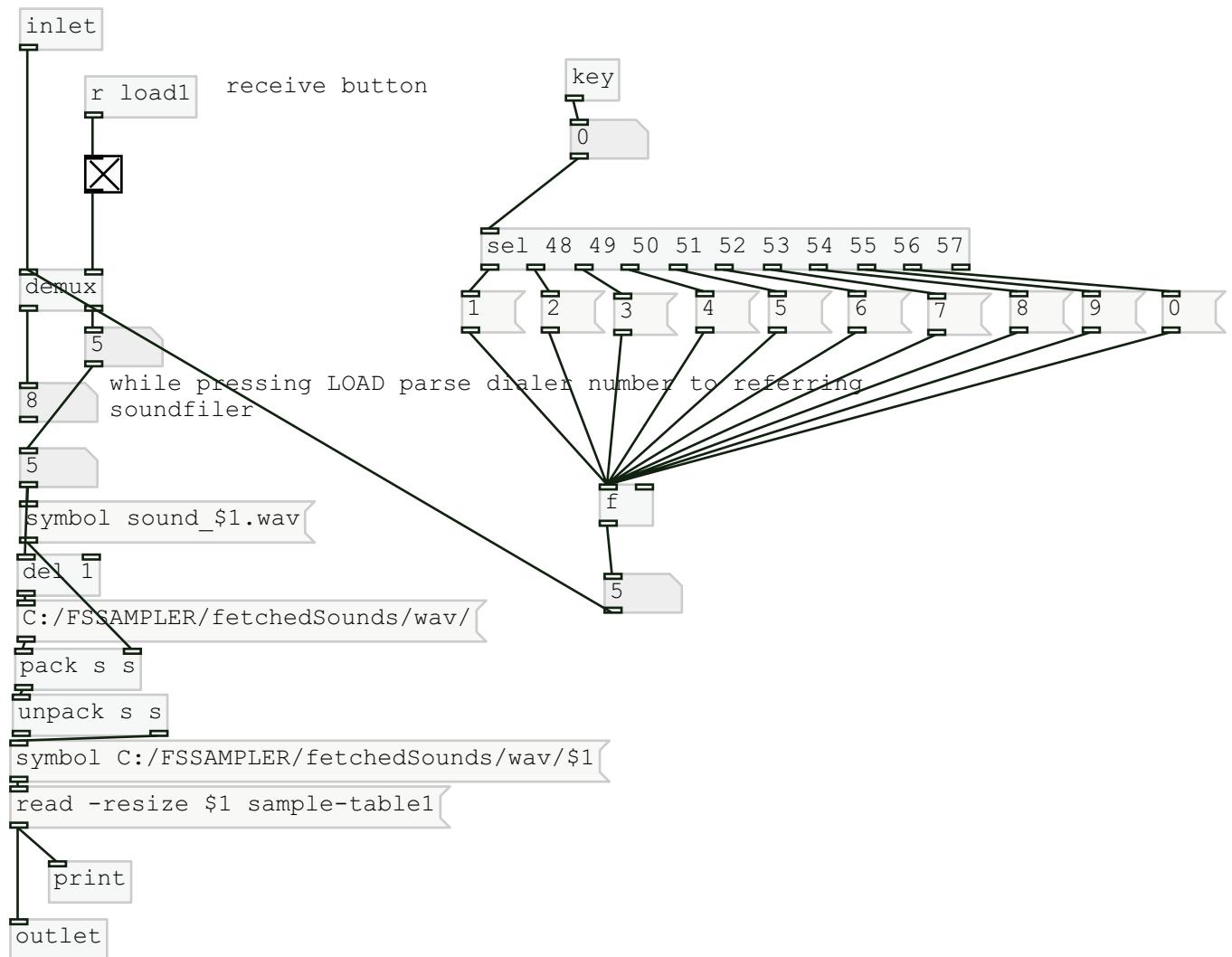
pd waehlscheibe



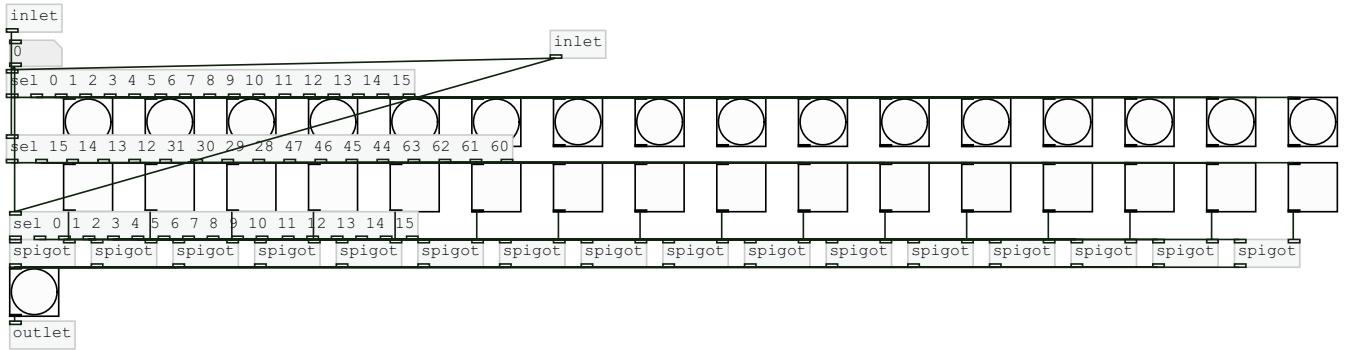
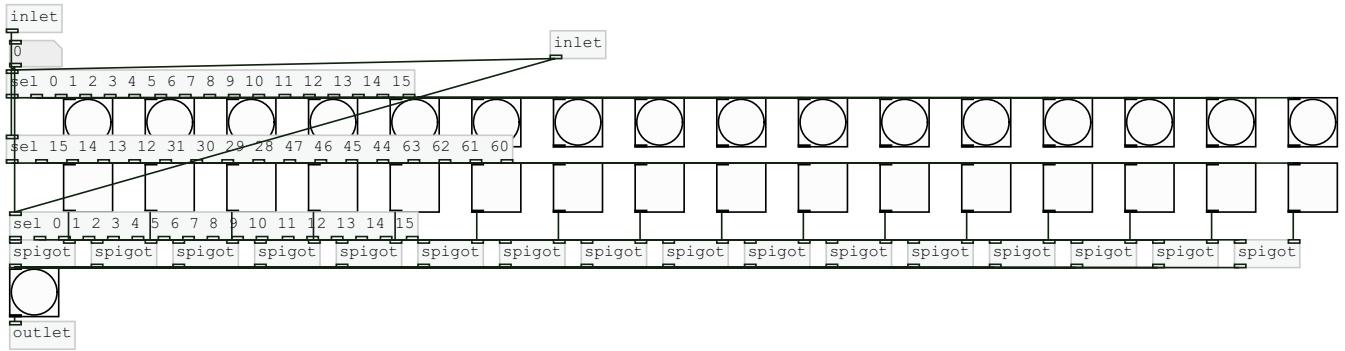
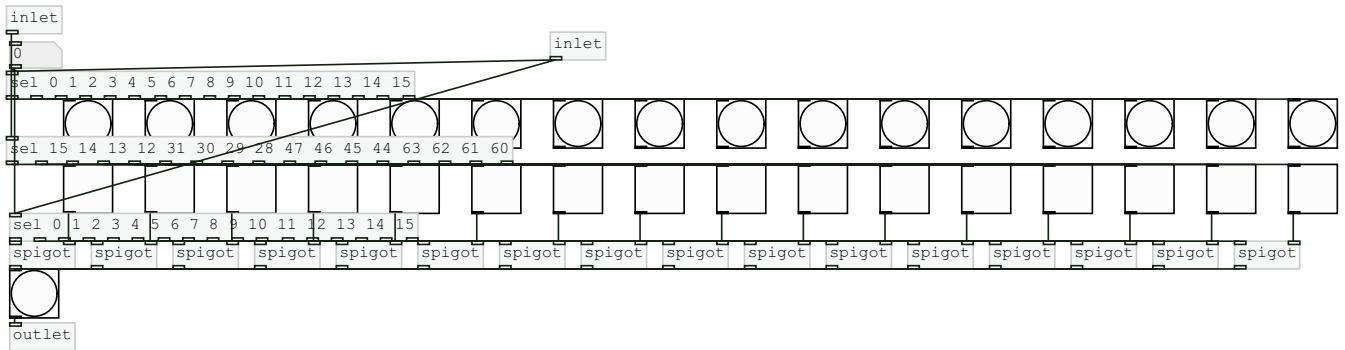
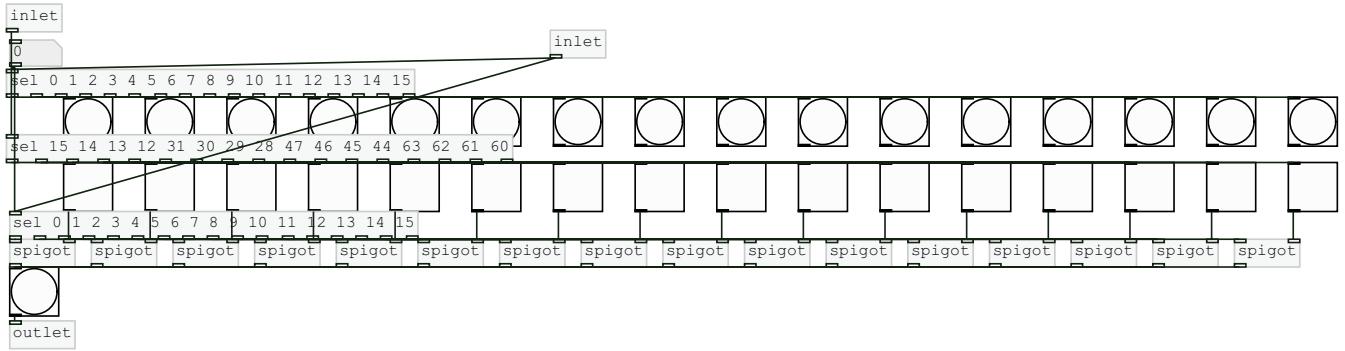
pd TrellisReceiver



pd sample1



pd track1-4



5. Hintergrund

5.1 Hintergrund

Sample heißt Probe. Dieser Begriff wird in ganz unterschiedlichen Zusammenhängen verwendet: In der Statistik ist das Sample eine Stichprobe, in der Audiotechnologie bedeutet es die digitale Abtastung eines analogen Klanges. Und dann gibt es natürlich das musikalische Sampling, wo ein Klang aus seinem ursprünglichen Kontext herausgelöst und damit verfügbar gemacht wird. Hier meint Sample ein (digital) gespeichertes Klangschnipsel, auf das beliebig zugegriffen werden kann.

In der vordigitalen Zeit beginnt das Sampling in den 40er Jahren nach dem Krieg in Frankreich, wo Pierre Schaeffer Naturklänge - sogenannte „konkrete“ Klänge - aufgenommen und auf Schallplatten mit Endlosrillen gepresst hat. Die Klänge können der Umwelt entstammen, aber auch aus Musikstücken. Dies ist ein beliebtes Verfahren im HipHop, wo aus kurzen Groove-Fragmenten älterer Musik (wie Soul, Funk oder Jazz) Loops gebaut werden, die das rhythmische Skelett eines Raps bilden.

Wichtig ist beim Sample auch seine Geschichte, die es transportiert, wodurch im Hörer bestimmte Emotionen hervorgerufen werden können.

Deshalb ist es so beliebt, Samples aus konnotierter - also mit Bedeutung aufgeladener - Musik zu verwenden, da sie vom Hörer wiedererkannt werden und dadurch bestimmte Stimmungen auslösen können.

Die freie Verfügbarkeit des Samples in der Musik beginnt eigentlich erst dort, wo es digital handhabbar wird.

„Immer den gleichen Stein den immer gleichen Berg hinauf wälzen.“

Das ist ein Sample, das von Heiner Müller stammt. Ich habe es deswegen ausgewählt, weil es selbst schon eine Endlos-Schleife darstellt. Der Stein des Sisyphos, der immer wieder hinaufgerollt wird und immer wieder hinunter fällt. Die einfachste Manipulation des Samples besteht darin, dass man es verkehrt herum abspielt. Ein anderes beliebtes Verfahren ist die Transposition des Samples auf unterschiedlichen Tonhöhen. Dabei fällt auf, dass beim Heruntertransponieren das Sample langsamer wird und bei Herauftransponieren schneller.

Nun gibt es aber auch interessante Methoden, Transposition und zeitliche Ausdehnung des Samples voneinander unabhängig zu machen. Durch Anwendung einer Technik, die sich Granularsynthese nennt, kann man das Sample stretchen - also dehnen - ohne dass sich seine Tonhöhe ändert.

In den 1980er wurden die ersten Sampler gebaut, eine Art Keyboard, mit dem man Klänge aufnehmen und wieder abspielen konnte. Über eine Tastatur konnte man diese Samples hinsichtlich ihrer Tonhöhe und Lautstärke verändern. Man nimmt beispielsweise den Klang einer Bongo auf und legt dieses Sample sozusagen auf die Tasten des Keyboards. So könnte man mit dem Klang dieser Bongo etwa die „Goldberg-Variationen“ spielen.

/// Karlheinz Essl / ORF Musiklexikon // 2.Dec 2017 // <http://www.esssl.at/bibliogr/musiklexikon-sample.html>

Sampling eröffnet dem Anwender eine praktisch grenzenlose Auswahl an Geräuschen, die es zu arrangieren und zu verarbeiten gilt. Allein in Deutschland werden rund 2,1 Millionen LPs pro Jahr verkauft. Bei einer durchschnittlichen Abspielzeit von 45 Minuten kommen hierbei 940,5 Millionen Minuten Sound zusammen. Macht 178,9 Jahre Sound pro Jahr. Daraus resultiert folgendes Problem - abzüglich der zu erwartenden Redundanz durch Kopien der selben Tonträger mag sich der ermittelte Wert vielleicht auf ein Zehntel reduzieren - immer noch stellt das den Anwender vor die Aufgabe 18 Jahre Musik in nur einem Jahr auf brauchbare Fragmente hin zu untersuchen.

Durch die Kommunikations- und Informations-Austauschmöglichkeiten, die der freien Welt gegenwärtig zur Verfügung stehen (Stand: 03.03.2020), addieren sich zu dem errechneten Wert noch die LP/CD/MC/Streaming Produktionen der restlichen Welt, alle nicht-kommerziellen Produktionen und alle außerhalb von Musikproduktion aufgenommenen zugänglichen Sounds. Darüber hinaus muss man sämtliche bis zum heutigen Zeitpunkt entstandenen Ton-Aufnahmen in die Auswahl mit einbeziehen.

Der hieraus resultierende Wert lässt sich nicht mehr statistisch ermitteln, dürfte die durchschnittliche Lebenserwartung eines Menschen aber um ein Vielfaches überschreiten.

„Irgendwo auf der „Masters of Dixieland vol. 1-6“ (Capitol Records Inc./ 1962 / California, USA/ LC 64077) muss eine passende Trompete zu finden sein.“

Wer nun zu Decidophobia und darüber hinaus vielleicht noch zum Prokrastinieren neigt, wird sich schwer tun die richtige Entscheidung (das richtige Sample) zu treffen.

Durch das Begrenzen der Bezugsquellen und der Menge an verfügbarem Material, wird der Prozess der musikalischen Verarbeitung des zugrundeliegenden Materials in den Vordergrund gestellt.

Wir möchten versuchen dem Anwender ein Werkzeug an die Hand zu geben, dass ihn bei der Suche und der Auswahl von Samples unterstützt und ihm die Zeit gibt, sich intensiver auf den Prozess der Produktion und des Arrangements zu konzentrieren.

Dazu reduziert Concrete Jungle die Auswahlmöglichkeiten auf zwei Parameter:

5.2 Basis

Concrete Jungle verfügt über eine AP-Schnittstelle zur Freesound.org Database. Die hier gespeicherten Sounds sind zum Teil mit Geo-Tags versehen und werden über die API auf die Maschine geladen.

Wir müssen uns nur noch für eine Region, ein Land oder einen Kontinent entscheiden, der die Quelle für unseren STOCK stellen soll. Für eine genaue Bereichs-Definition der Quelle geben wir den Radius, in Kilometern, um eine Koordinate mit in die Auswahl ein. Aus dem so erschlossenen Gebiet (SOURCE) werden alle verfügbaren Informationen gesammelt, gelistet und durch Filter auf eine Auswahl reduziert (STOCK).

Mit den so gewonnenen Informationen kann nun klanglich experimentiert werden. Hierzu lädt man Samples aus dem STOCK auf vier zur Verfügung stehende Spuren. Neben Anfangs- und Endpunkt der Sounds lassen sich die Abspielgeschwindigkeit- und Richtung kontrollieren. Zum Arrangieren steht ein 16-Step Sequenzer zur Verfügung. Mit den gegebenen lokal gebundenen Sounds, welche field recordings, Atmosphären, Stimmen und vieles mehr sein können, entstehen einzigartige Klanglandschaften, die einen unverkennbaren Bezug zu ihrem kulturellen, technischen oder geografischen Ursprung haben können.

