

Digital Art WiSe 2024/2025

Mixtape

In dieser interaktiven Kunstinstallation nehmen wir dich in 270 Grad mit auf eine Reise durch die Geburtstagserinnerungen eines Lebens - geführt durch Narration und der eigenen Bedienung eines Radios.

Martin Lauterbach

martin.lauterbach@student.reutlingen-university.de

Marco Holzäpfel

marco.holzaepfel@student.reutlingen-university.de

Rosalie Wieland

rosalie.wieland@student.reutlingen-university.de

Simon Knoblich

simon.knoblich@student.reutlingen-university.de

Jonathan Psenicka

jonathan.psenicka@student.reutlingen-university.de

Sissi Wu

sissi.wu@student.reutlingen-university.de

Alle Trailer, Abschnitte in DNxHQ und ein Durchlauf finden Sie in folgender Playlist:

https://www.youtube.com/playlist?list=PLJR0d_Vv370_5z8SbQHiUYpTRRZ9-fPv3

Erstellungsdatum: 23. Juni 2025

Abgabedatum: 24. Januar 2025



Hochschule Reutlingen
Reutlingen University

Inhaltsverzeichnis

1 LogLine: Thema und Grundaussage	4
1.1 Logline	4
1.2 POV	4
2 Ablauf der Story, POV Betrachter	5
2.1 Einleitung	5
2.2 Hauptteil	5
2.3 Ende	6
3 Expose	7
4 Produktion	13
4.1 Grundriss des Aufbaus	13
4.2 Technische Umsetzung	14
4.2.1 Benötigte Technik und Material:	14
4.2.2 Beschreibung der Medien	16
4.2.3 Arduino-Code	17
4.2.4 Touch-Designer: Seriell Schnittstelle (COM)	19
4.2.5 TouchDesigner-Code (onFrameStart)	20
5 Teamarbeit	36
5.1 Martin Lauterbach	37
5.2 Marco Holzäpfel	39
5.3 Rosalie Wieland	40
5.4 Simon Knoblich	41
5.5 Jonathan Psenicka	42
5.6 Sissi Wu	43
6 Installationsausführung	43

Tabellenverzeichnis

1	Benötigte Komponenten für das Radio	14
2	Raumtechnik (bereits verbaut)	14
3	Raumgegenstände	15
4	Technik zum Filmen	15
5	Software	15
6	Beschreibung der Medien	16

Abbildungsverzeichnis

1	Moodboard	7
2	Bild1: Book of Love	8
3	Bild2: Person dreht am Knopf	8
4	Bild3: Vogelperspektive Installation	9
5	Bild4: Seitenblick auf Radio	9
6	Bild5: Top-Over View des Radios	10
7	Bild6: Frontansicht Radio	10
8	Bild7: Sicht von Eingangstür	11
9	Bild8: Abschnitt Kind Geburtstagstorte	11
10	Bild9: Abschnitt Studium full-wide	12
11	Bild10: Behind-The-Scenes Jugend	12
12	Skizze Rauminstallation	13
13	Rotary Encoder to Arduino-Klon Steckplan	18
14	Teambild	36
15	Marty mit seinem Radio	37
16	Marty am Debuggen	37
17	Schnitt Kind	37
18	Schnitt Jugend	37
19	Schnitt Love	38
20	Schnitt Study	38
21	Schnitt BookOfLove	38
22	Schnitt Trailer	38
23	Love Behind-The-Scenes	39
24	Marco Protagonist	39
25	Rosie Protagonist	40
26	Simon und Marty	41
27	Simon Code	41
28	Jonathan	42
29	Sissi	43

1 LogLine: Thema und Grundaussage

1.1 Logline

Eine 270°-Installation, in der der Besucher durch die Interaktion mit einem Radio und die Nutzung von Licht und Sound die Erinnerungen eines Lebens von der Kindheit bis ins hohe Alter erleben kann. Ziel ist es, die Komplexität und Gefühlstiefe jedes Menschen zu vermitteln und Empathie zu fördern. Die Installation richtet sich an eine breite Besuchermenge, die sich für interaktive und emotionale Kunstprojekte interessiert.

1.2 POV

Wenn Sie den Raum betreten, fällt ein Lichtstrahl eines Scheinwerfers auf eine Couch und ein Radio. Das Radio beginnt zu leuchten und zu sprechen -> es lädt Sie ein, es zu benutzen, um wie in einem Tagebuch die Erinnerungen wiederzuerleben.

Der Drehknopf des Radios kann genutzt werden, um mit dem Mix aus Rauschen sowie der Musik und dem Licht, die durch das Rauschen hin durchbrechen und die Nähe anhand der Lautstärke signalisieren, einen Sender zu finden. Zudem verändert sich das Rauschen auf den Wänden.

Sobald dieser gefunden wurde, erwacht der Raum durch 270°-Projektionen und Surround-Sound zum Leben. Die Musik geht in den Raum über.

Die Projektionen erzählen, unterlegt mit den Worten des Radios und überlegt mit der Musik, die Erinnerung an ein Geburtstag.

Je nach Sender, wird hier ein anderer Lebensabschnitt beleuchtet. Sie können durch fünf verschiedene Kanäle navigieren. Die Erinnerungen reichen von einem einsamen Geburtstag mit dem Vater und seinen Fantasiehelden über die Teenagerjahre mit Freunden bis hin zum Erwachsenenleben mit der ersten Liebe, der langen Arbeit und dem ersten Kind.

Jeder Kanal hat eine eigene Musik und eine eigene Farbpalette. Sie helfen beim Finden des Senders, da jeder Abschnitt sein eigenes Lied, aus dem Radio kommend, besitzt.

Der letzte Sender führt Sie dabei auf eine bewegende Reise in das, was wie die Zukunft scheint, und endet damit, dass die Person alt ist und eine Familie hat.

Es wird ein tiefgründiges Zitat gesprochen:

Verstehen kann man das Leben nur rückwärts. Leben kann man es nur vorwärts.

Sören Kierkegaard <3

Es ist ein beeindruckendes Erlebnis in einem 270°-Raum, mit der Interaktivität des Radios, den belebenden Lichtern und den breit gefächerten emotionalen Einblicken.

All dies, in der Hoffnung, dass man die Komplexität und die Gefühlstiefe jedes anderen Menschen vor die Augen geführt bekommt.

2 Ablauf der Story, POV Betrachter

2.1 Einleitung

1. Ankunft und Eintritt: Der Besucher betritt die Installation durch einen schmalen, abgedunkelten Eingang, der ihn langsam in die Atmosphäre der Installation eintauchen lässt.
2. Erster Eindruck: Im Raum fällt bei Eintritt ein Lichtstrahl eines Scheinwerfers auf eine Couch und ein Radio. Der Raum ist ansonsten in sehr gedämpftem Licht gehalten, was eine geheimnisvolle und einladende Stimmung erzeugt, den Fokus aber auf die Kombi Couch-Radio leitet.
3. Interaktion mit dem Radio: Das Radio beginnt zu leuchten und spricht den Besucher direkt an, lädt ihn ein, es zu benutzen, um wie in einem Tagebuch die Erinnerungen wiederzuerleben. Dies weckt die Neugier des Besuchers und ermutigt ihn, den Drehknopf des Radios zu benutzen.

2.2 Hauptteil

1. Erkundung der Sender: Der Besucher dreht den Knopf des Radios und erlebt einen Mix aus einem Radio-Rauschen per Audio und Video sowie dem Gegenspieler: Musik und Licht, welches durch das Rauschen hin durchbrechen. Dies führt ihn dazu, verschiedene Sender zu erkunden, welche durch die steigende Lautstärke der Musik und ein heller werdendes Licht gefunden werden kann.
2. Finden eines Senders: Sobald ein Sender gefunden wird, erwacht der Raum durch 270°-Projektionen und Surround-Sound zum Leben. Die Musik geht in den Raum über und die Projektionen beginnen, die Erinnerung an einen Geburtstag zu erzählen.
3. Erinnerungen erleben: Der Besucher navigiert durch fünf verschiedene Kanäle, die jeweils einen anderen Lebensabschnitt beleuchten. Jeder Kanal hat eine eigene Musik und Farbpalette, die durch die Lichter im Raum dargestellt werden.
4. Die Installation startet am Wert 0, und der Drehmechanismus des Radios erlaubt eine Erhöhung dieses Werts. Dabei sind die Geburtstagserinnerungen linear nach Alter angeordnet, können aber "über-dreht" werden.
5. Emotionale Reise: Die Erinnerungen reichen von einem Geburtstag mit dem Vater und Fantasiehelden über die Teenagerjahre mit Freunden bis hin zum Erwachsenenleben mit der ersten Liebe, der langen Arbeit und dem ersten Kind.

-
6. Zukunftsvision: Der letzte Sender führt den Besucher auf eine bewegende Reise in das, was wie die Zukunft scheint, und endet damit, dass die Person alt ist und eine Familie hat.

2.3 Ende

1. Reflexion und Ausgang: Nach dem Durchlaufen der Sender und dem Erleben der verschiedenen Erinnerungen soll der Betrachter fühlen, als ob er gerade die rohen Emotionen des Erzählers*in geschenkt bekommen hat.
2. Die POV erlaubt das Einfühlen in die Sicht/den Verstand/die Aufnahme des Erzählers. So soll der Besucher in seinem tiefem Erleben verstärkt werden.
3. Gefühl und Message: Der Besucher verlässt die Installation mit einem Gefühl der Verbundenheit und Empathie, beeindruckt von der emotionalen Tiefe und Vielfalt der menschlichen Erfahrungen. Die Message, dass jeder Mensch eine einzigartige und wertvolle Geschichte hat, bleibt nachhaltig im Gedächtnis.

3 Expose



Abbildung 1: Moodboard



Abbildung 2: Drei Personen schauen den letzten Abschnitt Book of Love



Abbildung 3: Eine Person nutzt das Radio über den Drehknopf, die Feedback-LED-Reihe leuchtet



Abbildung 4: Vogelperspektive der laufenden Installation, in der rechten unteren Ecke ist das Radio und ein Besucher

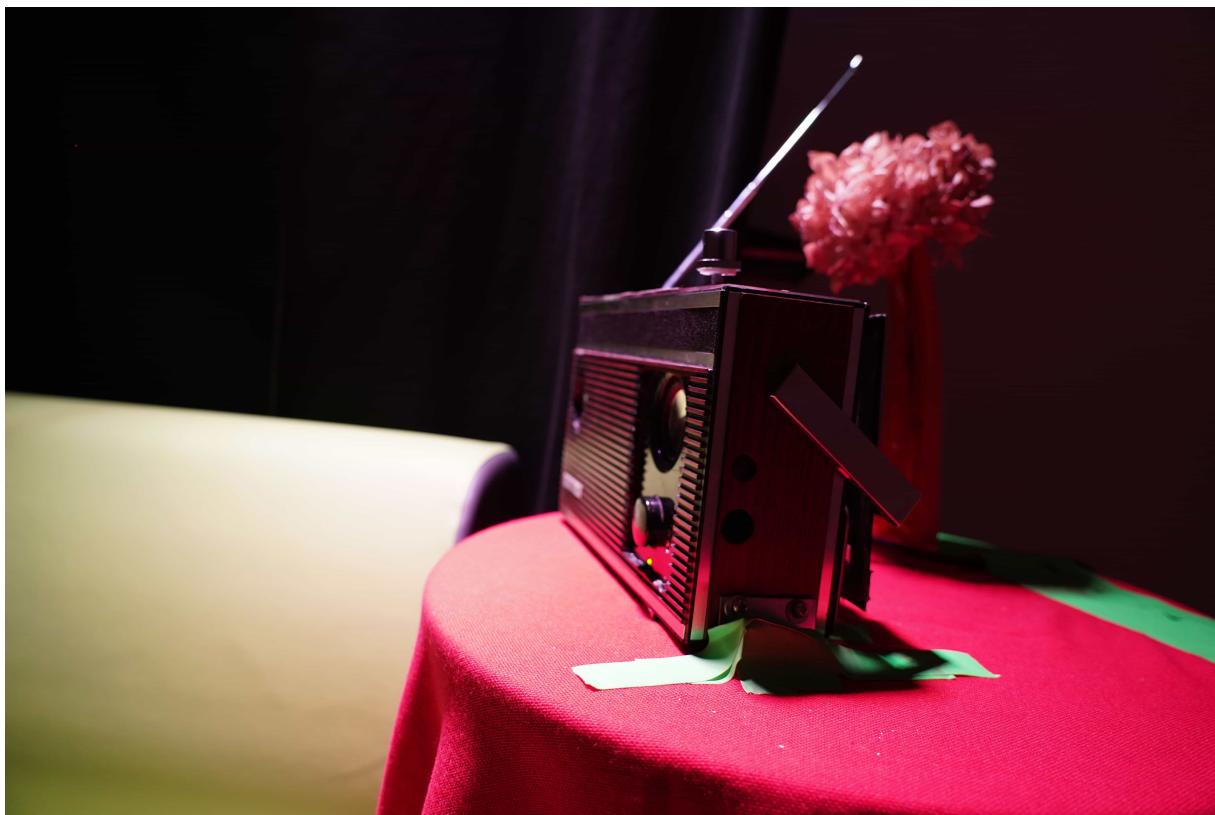


Abbildung 5: Seitenblick auf das Radio, gehüllt mit Licht



Abbildung 6: Top-Over View des Radios bei laufender Installation. Das Drehrad und die Zustandsanzeige-LED-Leiste sind sichtbar



Abbildung 7: Frontansicht auf Radio mit laufendem Jugend-Abschnitt im Hintergrund



Abbildung 8: Sicht von der Eingangstür. Vor dem Besucher ist das Sofa, dahinter das Radio auf einem Stehtisch. Alles wird in einem Lichtkegel gebadet.



Abbildung 9: Standbild aus Abschnitt Kind. Der Vater zündet die Kerzen der Geburtstagstorte an, während das Kind gespannt wartet. Die Teller am Tisch sind leer, niemand außer der Vater und das Kind sind anwesend.

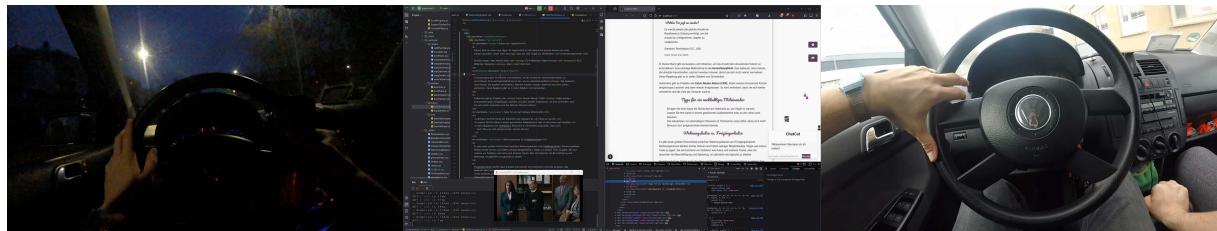


Abbildung 10: Standbild aus Abschnitt Studium. Auf der linken Seite ist eine Fahrt zur Universität, in der Mitte eine Programmierumgebung, auf der rechten ein Bild der Fahrt von der Universität nach Hause.



Abbildung 11: Behind-The-Scenes-Bild von Abschnitt Jugend. Die Kamerafrau, die die Perspektive darstellt, trägt eine GoPro Hero 3+ auf dem Kopf

4 Produktion

4.1 Grundriss des Aufbaus

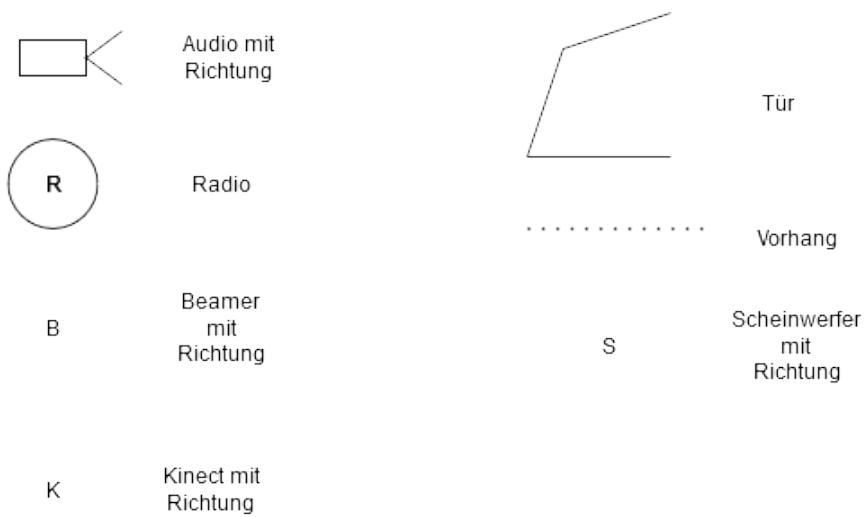
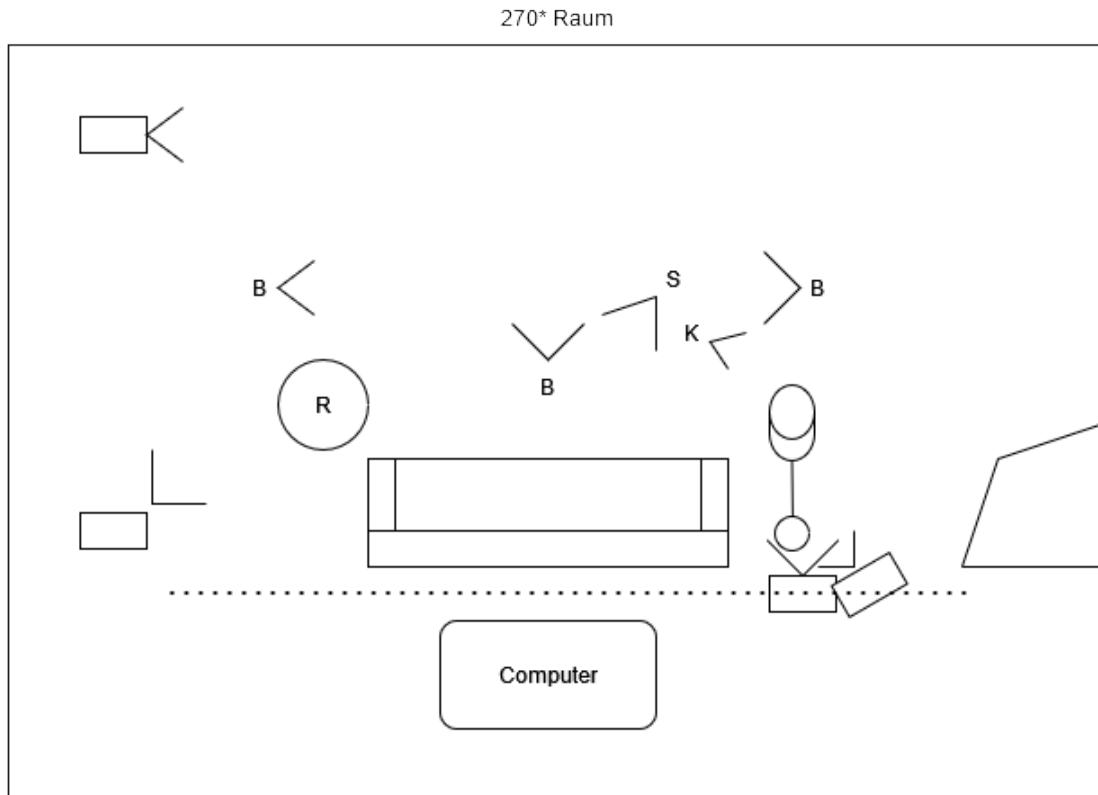


Abbildung 12: Skizze des 270* Raum

4.2 Technische Umsetzung

4.2.1 Benötigte Technik und Material:

Tabelle 1: Benötigte Komponenten für das **Radio**

Komponente	Menge
Altes Radio	1
Arduino	1
Breadboard Jumper Cables	2-4 Stück, 20cm
Arduino Breadboard	1
Rotary Potentiometer	1
USB 2.0 Cable Type A/B mit Active Extender	1, 5m
line-adressierbare LEDs	10
AUX Lautsprecher	1

Tabelle 2: **Raumtechnik** (bereits verbaut)

Komponente	Menge
Beamer	3
Lautsprecher	4
Kinect	1
Steuernder Computer	1
Subwoofer	1

Tabelle 3: Raumgegenstände

Komponente	Menge
Couch	1
Couchbeistelltisch	1
Stehlampe	1

Tabelle 4: Technik zum Filmen

Komponente	Menge
Sony Alpha 7	1
GoPro Hero 3+ Silver	1
Kamerastativ	1
Gimbal	1

Tabelle 5: Software

Software	Nutzen
DaVinci Resolve Studio 19	Schneiden der Abschnitte, Trailer, Colour Correcting, Audio-Mix
TouchDesigner 2022.32660	Erstellung der Installations-Programmierung
PyCharm	Erstellung der Installations-Programmierung

4.2.2 Beschreibung der Medien

Tabelle 6: Beschreibung der Medien für die Installation *Mixtape des Lebens*

Medienart	Inhalt	Begründung	Gestaltung	Umsetzung
Video	Aufnahme von verschiedenen Geburtstagen, alles aus der POV-Sicht	Wir sehen durch unseren Protagonisten, das ganze Leben. POV lässt den Zuschauer selber in diesen Momenten sehen und fühlen		siehe Tabelle 4, Tabelle 5
Audio	Pro Abschnitt ist ein Lied hinterlegt, dass zum Lebensabschnitt passt. Zusätzliche Synchronisation, um die Szene zu beleben. Das weiße Rauschen zwischen den Szenen mixt sich mit dem Lied des nahsten Abschnittes und verändert seine Lautstärke anhand der Entfernung.	Die Musik vermittelt mit dem Video eine einzigartige Symbiose. Der Bruch mit den echten Stimmen, hebt die Situation hervor und erlaubt einen genaueren Einblick. Das Laut und leiser hilft der Zuschauer*in den Sender zu finden		
Visualisierung	Fortschrittsbalken sowie Icons, die helfen den richtigen Sender beim Drehen des Radioknopfes zu finden	Unterstützt den Besucher*in die Abschnitte zu finden		Pythoncode in Abschnitt 4.2.5
Licht	DMX-Lampe strahlt mit einem Lichtkegel auf den Platz, Radio an dem die Besucher*in platz nehmen sollen.	Die Besucher*innen werden eingeladen und motiviert sich dem Platz zu nähern und an dieser Kunstinstitution teilzunehmen		
Animation	Rauschen des Bildschirmes während der Sendersuche	Der abgespielte Abschnitt leitet nach dem Ende auf das Rauschen, damit soll ausgedrückt werden, dass dieser Abschnitt vorbei ist und ein Sender angedreht werden muss	Perlin (GPU rendered), Seed wird in jedem Frame neu gesetzt durch Python Skript	2D

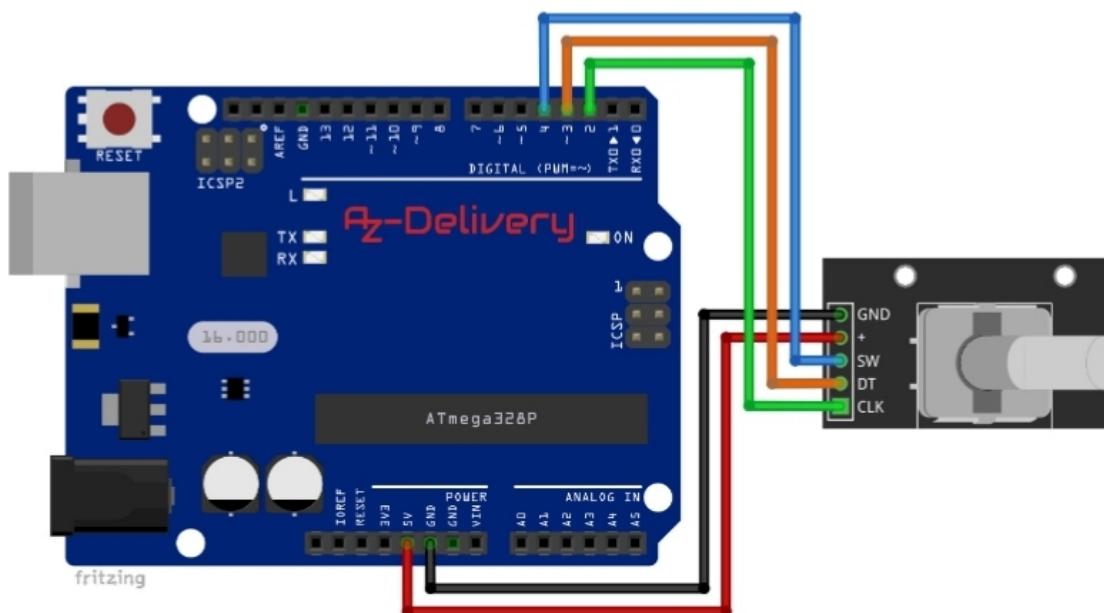
4.2.3 Arduino-Code

```
1 #define CLK_PIN 2
2 #define DT_PIN 3
3 #define SW_PIN 4
4
5 int last_position = 0;
6 int n = 0;
7 bool taster = LOW;
8 bool last_taster = LOW;
9
10 void setup() {
11     pinMode(CLK_PIN, INPUT_PULLUP);
12     pinMode(DT_PIN, INPUT_PULLUP);
13     pinMode(SW_PIN, INPUT_PULLUP);
14     Serial.begin(9600);
15 }
16
17 void loop() {
18     n = digitalRead(CLK_PIN);
19     taster = !digitalRead(SW_PIN);
20     if (taster != last_taster) {
21         if (taster) {
22             // Send click command
23             Serial.println("CLICK");
24         }
25         delay(10);
26         last_taster = taster;
27     }
28     // Check for rotary encoder rotation
29     if ((last_position == 0) && (n == HIGH)) {
30         if (digitalRead(DT_PIN) == LOW) {
31             // Send up command
32             Serial.println("UP");
33         } else {
34             // Send down command
35             Serial.println("DOWN");
36         }
37     }
38     last_position = n;
39 }
```

Listing 1: Arduino Code

Connecting the module with Atmega328p

Connect the KY-039 module with the Atmega328p as shown on the following connection diagram:



KY-040 pin > Mc pin

+ (VCC)	>	5V
GND	>	GND
CLK	>	D2
DT	>	D3
SW	>	D4

Red wire

Black wire

Green wire

Orange wire

Blue wire

Abbildung 13: Rotary Encoder to Arduino-Klon Steckplan

4.2.4 Touch-Designer: Seriell Schnittstelle (COM)

1. Serial DAT erstellen

- Verbinden mit dem richtigen COM-Port und der Baudrate (9600)

2. COMP BASH

- Parameters: Frequency | INT, 0 bis 100
- Parameters: Click | Toggle

```

1 def onReceive(dat, rowIndex, message, bytes):
2     command = message.strip()
3     if command == "UP":
4         new_frequency = op('radio_component').par.Frequency +
5             1
6         if new_frequency > 100:
7             new_frequency = 0 # Reset to 0 if it exceeds 100
8             op('radio_component').par.Frequency = new_frequency
9     elif command == "DOWN":
10        new_frequency = op('radio_component').par.Frequency -
11            1
12        if new_frequency < 0:
13            new_frequency = 100 # Reset to 100 if it goes
14                below 0
15            op('radio_component').par.Frequency = new_frequency
16    elif command == "CLICK":
17        op('radio_component').par.Click = not op(
18            'radio_component').par.Click
19
20 return

```

4.2.5 TouchDesigner-Code (**onFrameStart**)

Der folgende Python-Code läuft dauerhaft in einem Python Script Node, und wird bei jedem Frame Start ausgeführt. Die Installation ist dabei in drei simulierte Zustände unterteilt:

looking for start: Es befindet sich keine Person im Raum, und die Kinect triggert, falls jemand hereinkommt

starting: Die DMX leuchtet auf, Das Radio narratiert an, nach 15 sekunden wird auf ongoing gewechselt

ongoing: Während ongoing wird dauerhaft der Input des USers durch das Radio durch die Überprüfung eines INT-Werts in einem Bash überprüft. Dieser wertet durch ein zuvor erwähntest python skript die in UTF-8 codierten COM Signale aus und verändert den INT Wert. Von diesem Wert abhängig, in einer Range von 0 bis 200, werden in bestimmten Intervallen entweder ein Bündel an Static-Noise visuell und auditiv ausgegeben, oder ein Abschnitt mit seinem Film den verschiedenen Ausgängen zugewiesen. Die Abschnitte werden im Raum auditiv über die 4 Lautsprecher dargestellt. In den Static-Buffern zwischen den Abschnitten wird der Ton als Mix eines statischen weißen Rauschens und einem Lied des nächsten Abschnittes gemixt. Nach erfolgreichen durchlaufen eines Abschnittes, überprüft durch sein infochop lastframe attribut, wird eine Tonspur der Erzählerin aus dem Radio gespielt und die Installation wird automatisch über den INT-Wert in einen Buffer gesetzt.

```

1 # me - this DAT
2 #
3 # frame - the current frame
4 # state - True if the timeline is paused
5 #
6 # Make sure the corresponding toggle is enabled in the
7 # Execute DAT.
8 import random
9
10 # Definition of the Section class
11 class Section:
12     def __init__(self, video_file, audio_file, info_chop,
13                  audio_movie, narrator_voice):
14         self.video_file = video_file # Video file input
15         component
16         self.audio_file = audio_file # Audio file input
17         component

```

```
14     self.info_chop = info_chop # info_chop used to check
15         if last frame reached for stopping the video
16     self.audio_movie = audio_movie # audio movie used to
17         connect video audio to output room
18     self.narrator_voice = narrator_voice # used to play
19         one time when a last frame has been reached
20
21
22
23 # boolean state variables of the installation
24 global in_looking_start
25 global in_start
26 global in_ongoing
27
28
29 # for counting stars
30 global elapsedDelta
31 global lastFrequency
32
33 # runs once so we don't have to use methods
34 global booleanOnce
35 booleanOnce = False
36
37
38 global connectMultiple
39
40
41 global startPlay
42
43
44 def onStart():
45     return
46
47 def onCreate():
48     return
49
50 def onExit():
51     return
52
53
54 # Looping code goes here (runs every frame)
55 def onFrameStart(frame):
56     print(f"New Frame: {frame}")
57
58     global startPlay
```

```
52
53     global sections
54     global booleanOnce
55
56     global in_looking_start
57     global in_start
58     global in_ongoing
59
60     global elapsedDelta
61     global lastFrequency
62
63     global connectMultiple
64     in_static = False
65
66     # Get the current frequency value from the
       radio_component
67     frequency = op('radio_component').par.Frequency.eval()
68     print("NF")
69     print(frequency)
70
71     def show_icons():
72         add_from_icons = op('add4')
73         add_icons_screen = op('add5')
74
75         # connect add4 to add5
76         add_from_icons.outputConnectors[0].connect(
77             add_icons_screen.inputConnectors[0])
78
79     def hide_icons():
80         add_from_icons = op('add4')
81         add_icons_screen = op('add5')
82
83         # disconnect add4 from add5 if there is a connection
84         if add_from_icons.outputConnectors[0].connections:
85             add_from_icons.outputConnectors[0].disconnect()
86
87     def initialize_sections():
88
89         KIND = Section("kindmoviefilein", "kindAudio", "
90             last_frame_kind",
91                 "audioMovieKind", "narratorKind")
92         JUGEND = Section("jugendmoviefilein", "jugendAudio",
```

```

91         "last_frame_jugend",
92                     "audioMovieJugend", "narratorJugend"
93             )
94     STUDY = Section("studymoviefilein", "studyAudio", "last_frame_study",
95                     "audioMovieStudy", "narratorStudy")
96     LOVE = Section("lovemoviefilein", "loveAudio", "last_frame_love",
97                     "audioMovieLove", "narratorLove")
98     BOOK_OF_LOVE = Section("bookOfLovemoviefilein", "bookOfLoveAudio",
99                     "last_frame_bookoflove", "audioMovieBookOfLove",
100                    "narratorBookOfLove")
101
102     return KIND, JUGEND, STUDY, LOVE, BOOK_OF_LOVE

103
104     def reloadPulse(node):
105         node.par.reloadpulse.pulse()

106
107     def start_LookingForStart():
108         global in_looking_start, in_start, in_ongoing
109         hide_icons()
110         op('progressSlider').par.value0 = 0
111         in_looking_start = True
112         in_start = False
113         in_ongoing = False

114
115     def start_inStart():
116         global in_looking_start, in_start, in_ongoing
117         in_looking_start = False
118         in_start = True
119         in_ongoing = False

120
121     def start_inOngoing():
122         global in_looking_start, in_start, in_ongoing
123         in_looking_start = False
124         in_start = False
125         in_ongoing = True

126
127     # inner function to cut everything away from videofileout
128     def cut_off_input_to_windows():
129         for input connector in op('videoFileHere').

```

```
    inputConnectors:  
        input_connector.disconnect()  
  
128  
129     # inner function to cut off all audio connections to room  
     audio out  
130     def cut_all_audio_connections():  
131         for input_connector in op('audiofileoutROOM').  
     inputConnectors:  
        input_connector.disconnect()  
  
133  
134     def cut_off_comp_conns():  
135         for input_connector in op('compBuffer').  
     inputConnectors:  
        input_connector.disconnect()  
  
137  
138     if booleanOnce is False:  
  
139         hide_iicons()  
  
141  
142         # set value of progressSlider to 0  
op('progressSlider').par.value0 = 0  
  
144  
145         lastFrequency = 0  
146         elapsedDelta = 0  
  
147  
148         startPlay = False  
  
149  
150         connectMultiple = False  
  
151  
152         sections = initialize_sections()  
  
153  
154         # starting with kinect pooling state  
start_LookingForStart()  
  
156  
157         # turn dmx off  
op("lampeAn").par.value0 = 0  
  
159  
160         cut_all_audio_connections() # audio room  
161         cut_off_input_to_windows() # window  
162         cut_off_comp_conns() # comp  
  
163  
164         booleanOnce = True
```

```
165
166     # check which state the system is in
167     if in_looking_start:
168         print("Currently looking for start.")
169
170     # kinect polling if math is 1
171     # if math is 1 -> dmx on -> radio output rosi
172     if op('kinectMath')[0] == 1:
173
174         cut_all_audio_connections()    # audio room
175         cut_off_input_to_windows()   # window
176         cut_off_comp_connss()       # comp
177
178         # dmx an
179         op('lampeAn').par.value0 = 1
180         # output of rosi audio file to radioaudioout
181         op('hallo').outputConnectors[0].connect(op(
182             'audiofileoutRADIO').inputConnectors[0])
183         # Set the audiofilein to play once
184         op('hallo').par.repeat = False
185         reloadPulse(op('hallo'))
186         # Start playing the audiofilein
187         op('hallo').par.play = True
188
189         # state to start
190         start_inStart()
191
192         elapsedDelta = 0
193
194     elif in_start:
195         print("Starting.")
196         op('progressSlider').par.value0 = 0
197
198         # measure time, 10 seconds (assuming 60 fps)
199         elapsedDelta += 1
200
201         # if elapsed -> end audio rosi, dim down lamp
202         if elapsedDelta > 900:
203
204             # AUDIO
205             # stopping
206             op('hallo').par.play = False
```

```
206
207     # Disconnect the audiofilein from audiofileout
208     op('hallo').outputConnectors[0].disconnect()
209
210     #DMX off
211     op("lampeAn").par.value0 = 0
212
213     # set starting value to buffer before kind, after
214     # book of love
215     op('radio_component').par.Frequency = 25
216     lastFrequency = 25
217
218     # state to ongoing
219     connectMultiple = True
220     startPlay = True
221     start_inOngoing()
222
223     show_icons()
224
225     elapsedDelta = 0
226
227 elif in_ongoing:
228     video_crop = op('videoFileHere')
229
230     # between 0 and 1, 200/frequency
231     op('progressSlider').par.value0 = frequency / 200
232
233     # randomly set a number between 0 and 9000 for the
234     # seed in noise1
235     op('noise1').par.seed = random.randint(0, 9000)
236
237     def connectToComp(audioFile):
238         global connectMultiple
239
240         audio_in = op(audioFile)
241         audio_out = op('compBuffer')
242
243         if connectMultiple:
244             reloadPulse(audio_in)
245             audio_in.par.play = True
246
247             # Schleife durch die Eingangsverbindungen des
```

```
246             Composite Buffers
247             for i in range(len(audio_out.inputConnectors)):
248                 if not audio_out.inputConnectors[i].connections:
249                     audio_out.inputConnectors[i].connect(
250                         audio_in)
251                     print(f"Connected {audioFile} to
252                         input {i} of compBuffer")
253                     return
254             print("No empty connection found in
255                 compBuffer")
256
257 # function to play the narrator voice once when the
258 # last frame has been reached
259 def play_narrator_voice(narratorvoice):
260     narrator = op(narratorvoice)
261     connectToComp(narratorvoice)
262     narrator.par.repeat = False
263     narrator.par.play = True
264     reloadPulse(narrator)
265     print(f"Playing narrator voice: {narrator}")
266
267 # Function to assign video file if different
268 def assign_video_file(audioMovie, videoFile, infochop
269 ) :
270
271     global connectMultiple
272     global startPlay
273
274     connectMultiple = True
275
276     cut_off_comp_conns()
277
278     audioMovie = op(audioMovie)
279     videoFile = op(videoFile)
280     info_chop = op(infochop)
281
282     # input connection is p
283     # Check if there are any connections
284
285     if video_crop.inputConnectors[0].connections[0].
```

```

owner != videoFile:
    if startPlay:
        #AUDIO
        # cut off all audio connections to room
        #audio out
        cut_all_audio_connections()
        # assign the audioMovie file to audio out
        op("audiofileoutROOM").inputConnectors
            [0].connect(audioMovie)

    # VIDEO
    # cut of all connections of last video
    # files to video out
    cut_off_input_to_windows()
    # assigning video to output
    video_crop.inputConnectors[0].connect(
        videoFile)
    # Start playing the audiofilein
    videoFile.par.play = True
    reloadPulse(videoFile)

    startPlay = False
    print(f"Assigned video file: {videoFile}"
          )

# Check if the video has elapsed using Info CHOP
if info_chop['last_frame'] == 1:
    # Change frequency to the middle of the next
    # buffer
    if 35 <= frequency <= 45:
        op('radio_component').par.Frequency = 60
            # Middle of buffer between KIND and
            JUGEND
        play_narrator_voice("narratorKind")
    elif 75 <= frequency <= 85:
        op('radio_component').par.Frequency = 100
            # Middle of buffer between JUGEND
            and STUDY
        play_narrator_voice("narratorJugend")
    elif 115 <= frequency <= 125:
        op('radio_component').par.Frequency = 140
            # Middle of buffer between STUDY and

```

```

    LOVE
310     play_narrator_voice("narratorStudy")
311     elif 155 <= frequency <= 165:
312         op('radio_component').par.Frequency = 180
            # Middle of buffer between LOVE and
            BOOK_OF_LOVE
313         play_narrator_voice("narratorLove")
314         elif 195 <= frequency or 5 >= frequency :
315             op('radio_component').par.Frequency = 20
                # Middle of buffer between
                BOOK_OF_LOVE and KIND
316             play_narrator_voice("narratorBookOfLove")
317             print(f"Video elapsed. Changed frequency to
                  middle of next buffer.")
318             cut_all_audio_connections()

319
320     # inner method to make sure static is the video at
        video out
321     def static_to_video_out():
322         static_movie = op('staticmoviefilein')

323
324         # Check if there are any connections
325         if video_crop.inputConnectors[0].connections:
326             # Check if the static movie is already
                connected to the video output
327             if video_crop.inputConnectors[0].connections
                [0].owner != static_movie:
328                 cut_off_input_to_windows()

329
330         # Connect the static movie to the video output
331         video_crop.inputConnectors[0].connect(
332             static_movie)

333
334     # inner method to mix the audios of the closest
        sections together to audioOut
335     def mixing_closest_section_audios(leftAudioFile ,
336         linksBarriereFrequency , rightAudioFile ,
337         rechtsBarriereFrequency):
338
            global connectMultiple
            global startPlay

```

```
339         startPlay = True
340
341     def mixAudio(audioFile, lautstaerke):
342         audio_in = op(audioFile)
343         # set laustaecke of audio file
344         audio_in.par.volume = lautstaerke
345         static_audio = op("staticAudio")
346         # if lautstaerke is zero, skip setting the
347             # laustaecke of static
348         # if lautstaerke is not zero, invert value to
349             # 1 to assign to static
350         if not lautstaerke == 0:
351             staticVolume = 1 - lautstaerke
352             # set opaqueness of the static noise
353                 generator
354             op('level1').par.opacity = staticVolume
355             if staticVolume == 0:
356                 op('level1').par.opacity = 0.1
357                 staticVolume = 0.05 # atleast a
358                     little bit so the user knows he
359                     isnt in the section
360                     static_audio.par.volume = staticVolume
361
362         # 1 / 15 der laust rke je ganzzahl n her /
363             weiter
364
365         distanceLeft = frequency - linksBarriereFrequency
366         distanceRight = rechtsBarriereFrequency -
367             frequency
368
369         laustaekeLinks = 1/15 * distanceLeft
370         if laustaekeLinks > 1:
371             laustaekeLinks = 1
372             laustaekeLinks = 1 - laustaekeLinks
373
374         laustaekeRechts = 1/15 * distanceRight
375         if laustaekeRechts > 1:
376             laustaekeRechts = 1
377             laustaekeRechts = 1 - laustaekeRechts
378
379         connectToComp("staticAudio")
```

```
374     connectToComp(leftAudioFile)
375     # mix audio depending on lauststaerkeLinks
376     mixAudio(leftAudioFile, lauststaerkeLinks)
377
378     connectToComp(rightAudioFile)
379     # mix audio depending on lauststaerkeRechts
380     mixAudio(rightAudioFile, lauststaerkeRechts)
381
382     # assign compBuffer to audiofileoutAUDIO
383     audio_out = op('audiofileoutAUDIO')
384     audio_out.inputConnectors[0].connect(op("compBuffer"))
385
386     connectMultiple = False
387
388     def cutOutAudioMovies():
389         global sections
390         # Get the owner of the first connection
391         if op("audiofileoutROOM").inputConnectors[0].
392             connections:
393             owner = op("audiofileoutROOM").
394                 inputConnectors[0].connections[0].owner
395             if owner == op(sections[0].audio_movie):
396                 cut_all_audio_connections()
397             elif owner == op(sections[1].audio_movie):
398                 cut_all_audio_connections()
399             elif owner == op(sections[2].audio_movie):
400                 cut_all_audio_connections()
401             elif owner == op(sections[3].audio_movie):
402                 cut_all_audio_connections()
403             elif owner == op(sections[4].audio_movie):
404                 cut_all_audio_connections()
405
406     def rectangle_to_bright_blue():
407         op('rectangle1').par.fillcolorr = 0.4980392
408         op('rectangle1').par.fillcolorg = 0.8666667
409         op('rectangle1').par.fillcolorb = 1
410
411     def rectangle_to_nice_red():
412         op('rectangle1').par.fillcolorr = 0.5
413         op('rectangle1').par.fillcolorg = 0
414         op('rectangle1').par.fillcolorb = 0
```

```
413
414     def rectangle_to_nice_green():
415         op('rectangle1').par.fillcolorr = 0.682353
416         op('rectangle1').par.fillcolorg = 1
417         op('rectangle1').par.fillcolorb = 0.4980392
418
419     def rectangle_to_nice_yellow():
420         op('rectangle1').par.fillcolorr = 0.9137255
421         op('rectangle1').par.fillcolorg = 1
422         op('rectangle1').par.fillcolorb = 0
423
424     def rectangle_to_nice_purple():
425         op('rectangle1').par.fillcolorr = 0.3568628
426         op('rectangle1').par.fillcolorg = 0
427         op('rectangle1').par.fillcolorb = 1
428
429
430     # Switch case to check the current frequency value
431     if 35 <= frequency <= 45:
432         rectangle_to_nice_red()
433         in_static = True
434         assign_video_file(sections[0].audio_movie,
435                           sections[0].video_file, sections[0].info_chop)
436         print("Section: KIND")
437
438     elif 45 <= frequency <= 75:
439         rectangle_to_bright_blue()
440         cutOutAudioMovies()
441         static_to_video_out()
442         mixing_closest_section_audios(sections[0].
443                                         audio_file, 45, sections[1].audio_file, 75)
444         print("Buffer between KIND and JUGEND")
445
446     elif 75 <= frequency <= 85:
447         rectangle_to_nice_purple()
448         in_static = True
449         assign_video_file(sections[1].audio_movie,
450                           sections[1].video_file, sections[1].info_chop)
451         print("Section: JUGEND")
452
453     elif 85 <= frequency <= 115:
454         rectangle_to_bright_blue()
```

```
452         cutOutAudioMovies()
453         static_to_video_out()
454         mixing_closest_section_audios(sections[1].
455             audio_file, 85 , sections[2].audio_file, 115)
456         print("Buffer between JUGEND and STUDIEREN")
457
458     elif 115 <= frequency <= 125:
459         rectangle_to_nice_green()
460         in_static = True
461         assign_video_file(sections[2].audio_movie,
462             sections[2].video_file, sections[2].info_chop)
463         print("Section: STUDY")
464
465     elif 125 <= frequency <= 155:
466         rectangle_to_bright_blue()
467         cutOutAudioMovies()
468         static_to_video_out()
469         mixing_closest_section_audios(sections[2].
470             audio_file, 125 , sections[3].audio_file, 155)
471         print("Buffer between STUDY and LOVE")
472
473     elif 155 <= frequency <= 165:
474         rectangle_to_nice_yellow()
475         in_static = True
476         assign_video_file(sections[3].audio_movie,
477             sections[3].video_file, sections[3].info_chop)
478         print("Section: LOVE")
479
480     elif 165 <= frequency <= 195:
481         rectangle_to_bright_blue()
482         cutOutAudioMovies()
483         static_to_video_out()
484         mixing_closest_section_audios(sections[3].
485             audio_file, 165 , sections[4].audio_file, 195)
486         print("Buffer between LOVE and BOOK_OF_LOVE")
487
488     elif 5 <= frequency <= 35:
489         rectangle_to_bright_blue()
490         cutOutAudioMovies()
491         static_to_video_out()
492         mixing_closest_section_audios(sections[4].
493             audio_file, 5 , sections[0].audio_file, 35)
```

```
488         print("Buffer between BOOK_OF_LOVE and KIND")
489     else:
490         rectangle_to_nice_red()
491         in_static = True
492         assign_video_file(sections[4].audio_movie,
493                           sections[4].video_file, sections[4].info_chop)
494         print("Section: BOOK_OF_LOVE")
495
496
497     if in_static is False:
498         print("LF")
499         print(lastFrequency)
500         # checking if last frequency is the same as the
501         # current
502         if frequency == lastFrequency:
503             print("+1 lf")
504             elapsedDelta += 1
505             print(elapsedDelta)
506         else:
507             print("reset lf")
508             elapsedDelta = 0
509
510         lastFrequency = frequency
511
512         if elapsedDelta > 2100:
513             print("reached cutoff time!")
514             # cut of all connections of last video files
515             # to video out
516             cut_off_input_to_windows()
517             cut_all_audio_connections()
518             cut_off_comp_conns()
519             start_LookingForStart()
520
521     # END OF STATES THAT HAVE BEEN CONFIGURED
522     else:
523         print("Unknown state.")
524
525     return
526
527 def onFrameEnd(frame):
528     return
```

```
527  
528 def onPlayStateChange(state):  
529     return  
530  
531 def onDeviceChange():  
532     return  
533  
534 def onProjectPreSave():  
535     return  
536  
537 def onProjectPostSave():  
538     return
```

Listing 2: Python Touch-Designer Code

5 Teamarbeit



Abbildung 14: Bild des Teams. Von links nach rechts: Jonathan, Marco, Rosie, Sissi, Simon, Marty (Martin)

Die sechs Künstler planten, drehten und dokumentierten gemeinsam. Da verschiedene Zeitpläne und technisches Know-How bestand, wurde das Schneiden und Synchronisieren der Footage auf einzelne Teammitglieder gelegt. Zu den meisten geplanten Terminen fanden sich in der Regel alle Mitglieder zusammen, um gemeinsam zu testen und zu knobeln.

Daher ist es richtig die Aussage zu treffen: Diese Installation ist eine Gruppeninstallation.

Im Folgendem soll dennoch aufgeführt sein, welche Aufgaben wie verteilt und ausgeführt wurden, um einen fairen Überblick zu gewährleisten.

5.1 Martin Lauterbach

- Radiobau



Abbildung 15: Bild wie Marty grinsend hinter dem von ihm ausgehöhltem und mit einem Arduino (Rotary Potentiometer, LED-Kabel) und einem Stereo-Lautsprecher ausgestattetem altem Radio steht

- Touchdesigner & Python Programmierung



Abbildung 16: Bild wie Marty am Installations-Computer die Installation on-the-fly editiert

- Schnitt Kind

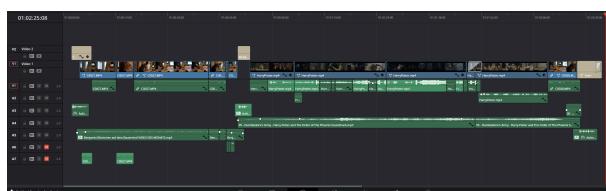


Abbildung 17: DaVinci Resolve 19 Schnitt des Abschnitts Kind

- Schnitt Jugend



Abbildung 18: DaVinci Resolve 19 Schnitt des Abschnitts Jugend

- Schnitt Love

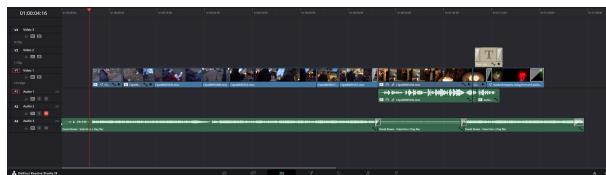


Abbildung 19: DaVinci Resolve 19 Schnitt des Abschnitts Love

- Schnitt Study



Abbildung 20: DaVinci Resolve 19 Schnitt des Abschnitts Study

- Schnitt BookOfLove

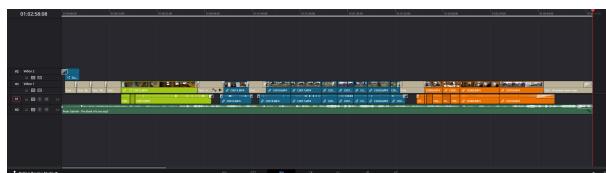


Abbildung 21: DaVinci Resolve 19 Schnitt des Abschnitts BookOfLove

- Schnitt Trailer

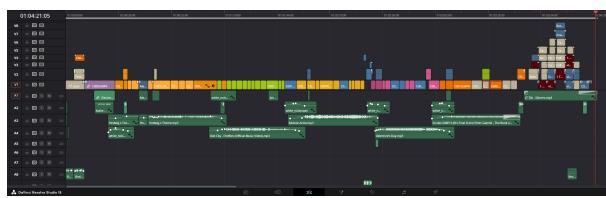


Abbildung 22: DaVinci Resolve 19 Schnitt des Abschnitts Trailers

- Dreh
- Drehbuch Kind
- Drehbuch Study
- Protagonist Trailer
- Dokumentation

5.2 Marco Holzäpfel

- Dreh
- Drehbuch Jugend



Abbildung 23: Bild des Partytisches, Behind-The-Scenes

- Protagonist als Vater



Abbildung 24: Marco als Vater

- Protagonist Trailer
- Dokumentation

5.3 Rosalie Wieland



Abbildung 25: Rosie als die POV

- Storyboard Trailer
- Drehbuch Love
- Dreh
- Stimme des Radios
- Protagonist der Videos
- Dokumentation
- Protagonist Trailer

5.4 Simon Knoblich



Abbildung 26: Simon übersieht die Installation

- Touchdesigner Python Programmierung

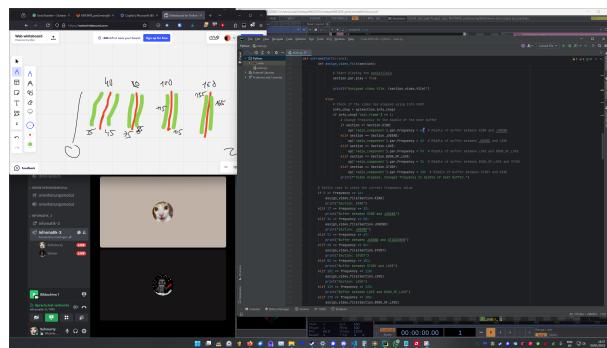


Abbildung 27: Simon und Marty coden zusammen das große Python Skript

- Dreh
- Drehbuch Study
- Coschnitt Trailer
- Protagonist Trailer
- Dokumentation

5.5 Jonathan Psenicka

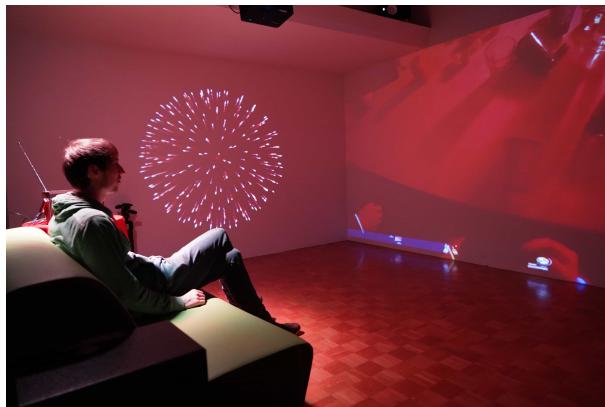


Abbildung 28: Jonathan schaut sich die Installation an

- Dreh
- Protagonist als Partner (jung)
- Protagonist Trailer

5.6 Sissi Wu



Abbildung 29: Sissi schaut sich die Installation an

- Dreh
- Verpflegung
- Protagonist Trailer

6 Installationsausführung

Die folgenden Seiten dokumentieren die Inbetriebnahme

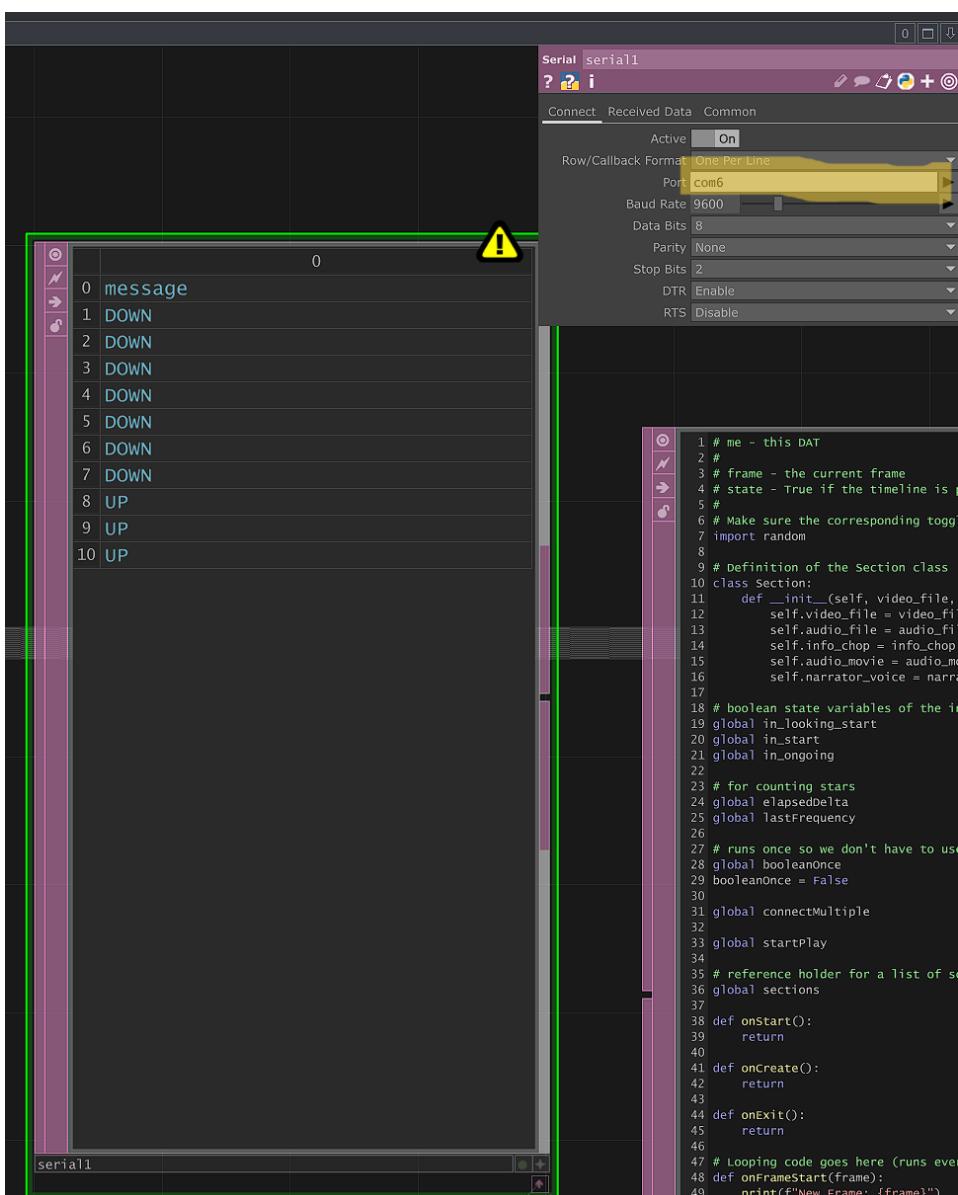
Anleitung Inbetriebnahme Projekt Mixtape

HARDWARE

1. Alles an Strom anschließen (Lautsprecher, Beamer, Radio und Windows-Rechner)
2. Beamer einschalten
3. LED-Licht einschalten
4. LED-Licht auf DMX-Modus stellen
5. Rechner einschalten
6. Nun muss der Radio an den PC über einen aktiv USB-Kabel an einem USB-Port verbunden werden.

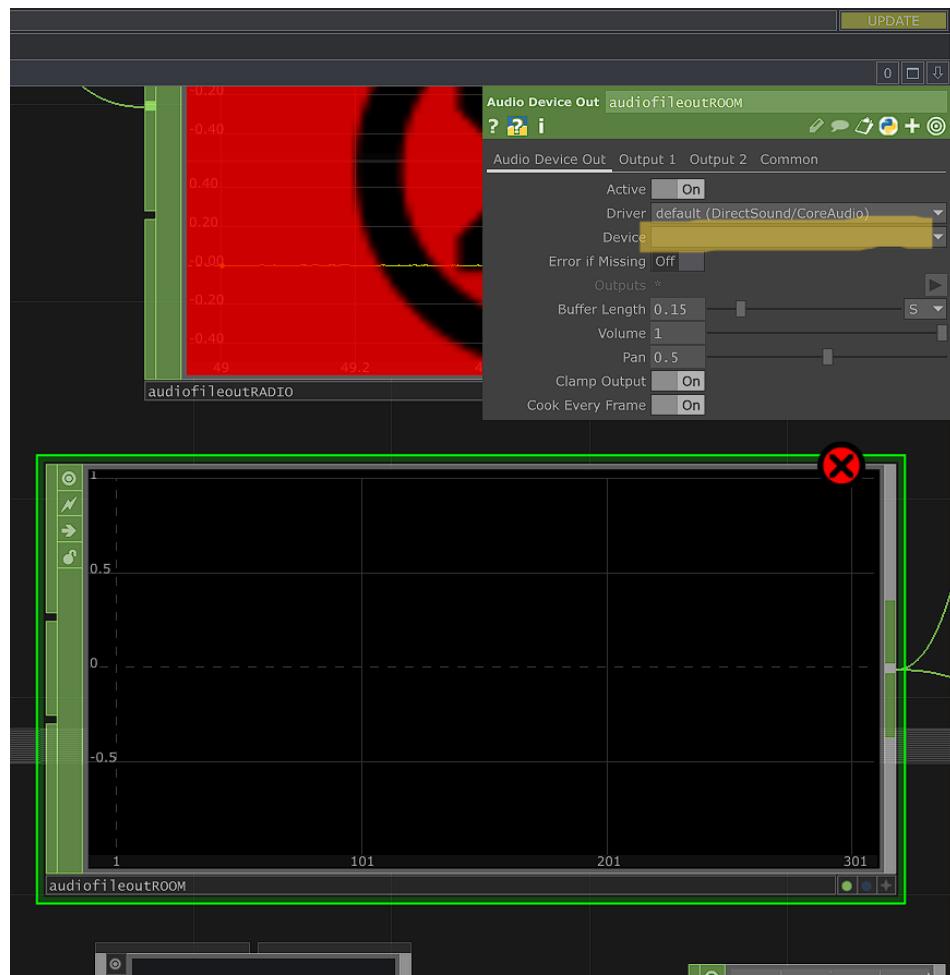
SOFTWARE

1. Wir haben eine Verknüpfung unserer Touchdesigner-File auf den Desktop gelegt, diese öffnen (Name: **MixTape.....**)
2. Jetzt serial1 auf den Richtigen COM-Port gesetzt werden, dass dieser Richtig erkannt wird. Dieser COM-Port ändert sich je nach USB-Port:

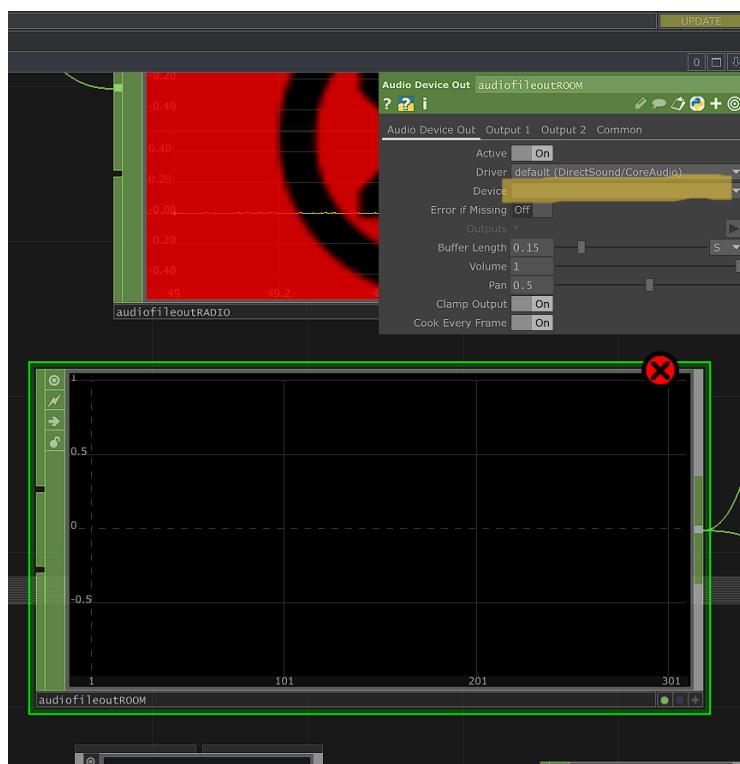


3. Nun muss das Audio für den Radio und den Raum eingestellt werden:

- Radio: Driver auf Realtek 2 stellen;



- Raum: Driver auf Focusrite USB Audio stellen;

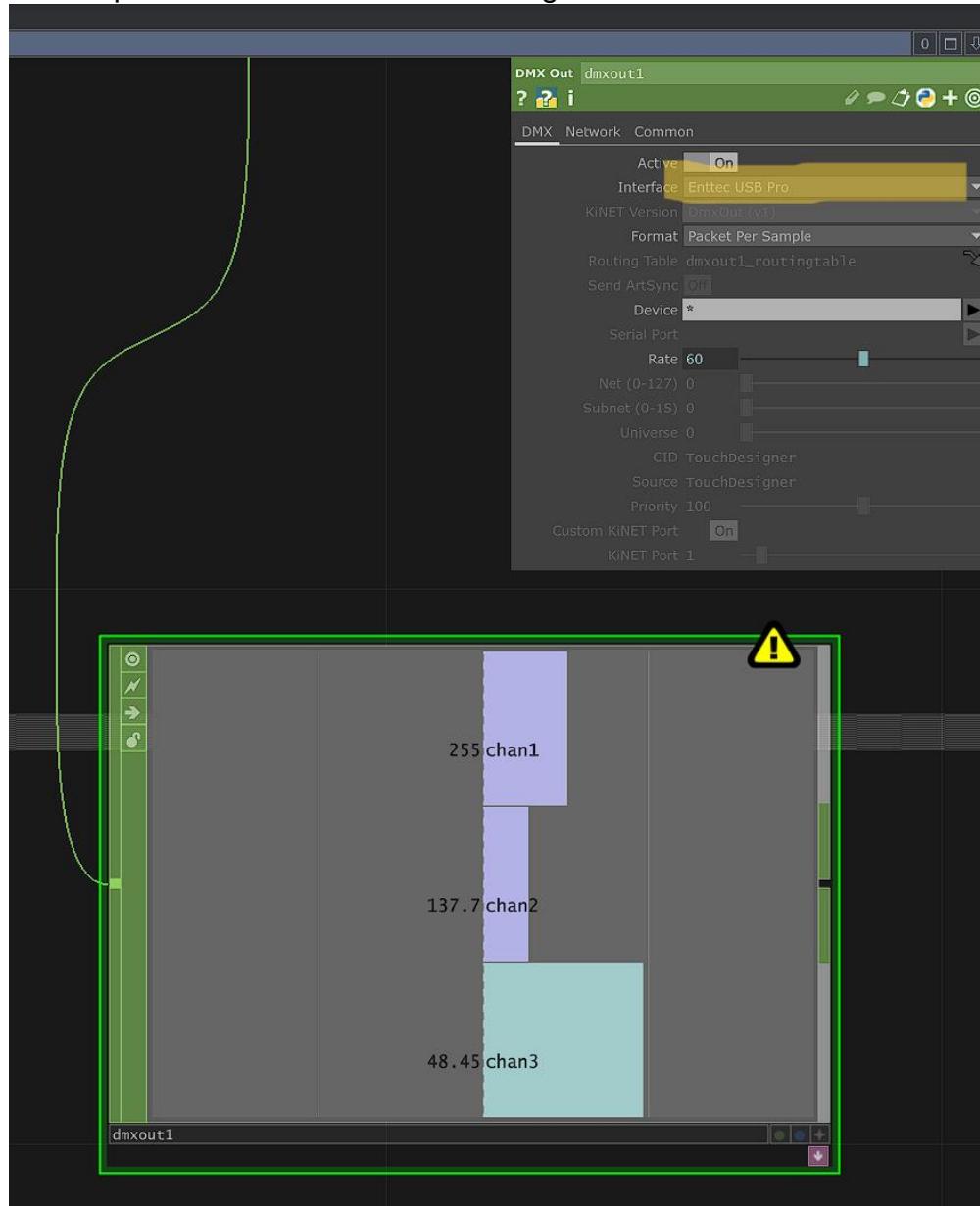


- Nun sollte der Installation Betriebsbereit sein.

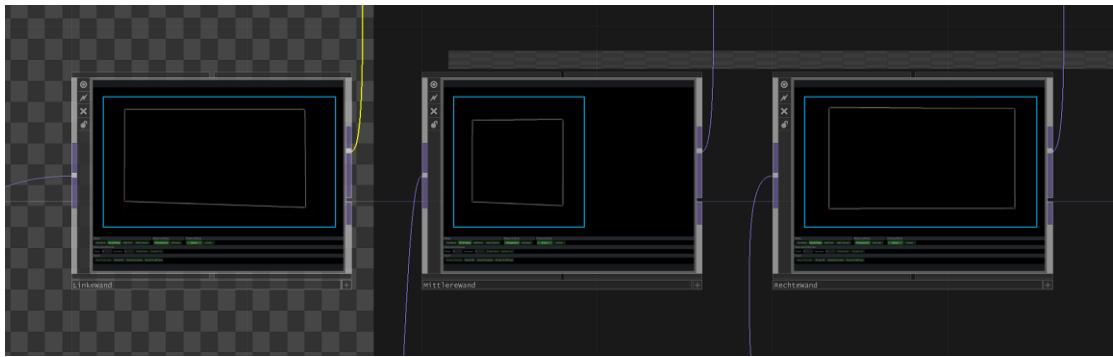
Informationen zu den einzelnen Komponenten:

Wir haben hier noch einige Informationen zu den einzelnen Komponenten, falls es zu Problemen der einzelnen Komponenten kommt:

- Licht: Adapter muss auf Enttec USB Pro gestellt werden



- Falls sich etwas mit den Beamer ändert, kann man über die Stoner das Bild wieder Positionieren:



- Die Struktur des Projekts ist großteilig mit Relativen Pfaden aufgebaut, deswegen haben wir hier nochmal die Struktur der Dateien:
 - o Original MixTape Datei (neuste Version benutzen)
 - Unter D:\MixTape\

Name	Date modified	Type	Size
📁 _RawCutConfigs	16/01/2025 21:51	File folder	
📁 AUDIO	16/01/2025 21:50	File folder	
📁 AUDIO_NarratorVoice	16/01/2025 21:50	File folder	
📁 Backup	16/01/2025 21:50	File folder	
📁 ICONS	16/01/2025 21:50	File folder	
📁 VIDEO	16/01/2025 21:51	File folder	
⌚ KnobelbachschesMIXTAPE.27.toe	16/01/2025 21:49	TouchDesigner	11.155 KB
⌚ KnobelbachschesMIXTAPE.30.toe	17/01/2025 14:24	TouchDesigner	22.289 KB
⌚ KnobelbachschesMIXTAPE.toe	16/01/2025 21:49	TouchDesigner	11.155 KB

- o Audio-Dateien:

FilmsAndSeries (D:) > MIXTAPE > AUDIO					Search A
Sort ▾ View ▾ ...					
Name	#	Title	Contributing artists	Album	
🎵 Fireflies.mp3		Fireflies	Owl City	Ocean Eyes	
⚠ Hallo.wav		Hallo	Rosie and Marty		
🎵 Hedwig's Theme.mp3		Hedwig's Theme	John Williams - To...	Hedwig's Theme	
🎵 Mixtap - Muerte En Hawaii.mp3		Calle 13 - Muerte En Haw...	Calle13VEVO	Calle 13 - Muerte En ...	
🎵 Molesk-Artist.mp3		Molesk	Atlas	Molesk	
🎵 Peter Gabriel - The Book of Love.mp3		Peter Gabriel - The Book o...	Peter Gabriel		
🎵 staticAUDIO.mp3		TV STATIC (4K 60FPS)	The Fowl Owl	TV STATIC (4K 60FPS)	
🎵 Valentine's Day.mp3	6	Valentine's Day	David Bowie	The Next Day (Deluxe...)	
⚠ white_noise.wav					

FilmsAndSeries (D:) > MIXTAPE > AUDIO_NarratorVoice

Name	#	Title	Contributing artists	Album
BookOfLove.wav	10	BookOfLove	Rosie as Voice and...	
DrehAnMeinemKnopf.wav	9	DrehAnMeinemKnopf	Rosie as Voice and...	
Jugend.wav	4	Jugend	Rosie as Voice and...	
LOve.wav	3	LOve	Rosie as Voice and...	
NarratorCommentary.aup3				
RosieReactionAudio.wav	1	RosieReactionAudio	Rosie as Voice and...	
RosiKommiKind.wav	6	RosiKommiKind	Rosie as Voice and...	
RosiKommiMusik.wav	7	RosiKommiMusik	Rosie as Voice and...	
RosiTrailerStart.wav	8	RosiTrailerStart	Rosie as Voice and...	
Study.wav	2	Study	Rosie as Voice and...	

- Video-Dateien (Abschnitte):

FilmsAndSeries (D:) > MIXTAPE > VIDEO

Name	Date	Type	Size	Length
Abschnitt_bookOfLove_HAP.mov	12/01/2025 02:15	MOV File	3.063.616 KB	00:02:58
Abschnitt_JUGEND_HAP.mov	12/01/2025 02:16	MOV File	6.338.713 KB	00:01:58
Abschnitt_Kind_HAP.mov	16/01/2025 21:49	MOV File	2.692.473 KB	00:02:24
Abschnitt_Love_HAP.mov	12/01/2025 02:17	MOV File	1.919.999 KB	00:01:27
Abschnitt_STUDY_HAP.mov	12/01/2025 02:16	MOV File	2.723.059 KB	00:00:47

- Icons (Fortschrittsleiste)

