



## Projektarbeit

Dipl. Techniker HF Informatik Fachrichtung Applikation  
TEKO Luzern

Dozent: Bruno Hammer

---

Vorgelegt am 23. März 2022

**Adrian Birrer**  
L-TIN-19-Mi-a  
[adrian.birrer@edu.teko.ch](mailto:adrian.birrer@edu.teko.ch)

**Janik Schilter**  
L-TIN-19-Mi-a  
[janik.schilter@edu.teko.ch](mailto:janik.schilter@edu.teko.ch)

## Management Summary

Bei der auditiven Wahrnehmung geht es nicht nur um das Hören von laut und leise, sondern viel mehr um das differenzierte Hören von akustischen Signalen. Die auditive Wahrnehmung ist, wie das Hören, eine Grundvoraussetzung für die Sprachentwicklung von Kindern.

Diese Projektarbeit befasst sich mit der Wichtigkeit des auditiven Lernens und stellt eine praktische Lösung zur Verfügung, diese Fähigkeiten spielerisch zu fördern.

Das Spielkonzept von **KLANGMEMORY** ist vergleichbar mit einem gewöhnlichen Memory-Spiel, jedoch wird anstelle von Bildern mit Klängen gespielt. Die Memorykarten beinhalten einen integrierten NFC-Tag. Wird eine Memorykarte auf den RFID-Leser gelegt, so erfolgt mittels integrierten Lautsprechern ein Klang, welcher je nach ausgewählter Kategorie für diese Karte bestimmt ist.

Das entwickelte System überzeugt durch intuitive und kinderleichte Handhabung, geringem Kostenaufwand und grossen Erweiterungsmöglichkeiten, da die Memorykarten beliebig bespielt werden können. Die Wünsche und Erwartungen der Projektbeteiligten wurden erfüllt.

Nach Fertigstellung der Projektarbeit waren alle Musskriterien vollumfänglich erfüllt. Die Wunschkriterien wurden aus Kostengründen nicht berücksichtigt. Die Projektbeteiligten behalten sich vor, die Wunschkriterien zu gegebener Zeit, ausserhalb der Projektarbeit zu erfüllen.



Abbildung 1: Klangmemory



# Inhaltsverzeichnis

<b>1 Vorwort</b>	2
1.1 Einleitung . . . . .	2
<b>2 Richtlinien und Vorgaben</b>	3
<b>3 Initialisierung</b>	4
3.1 Projektorganisation . . . . .	4
3.1.1 Ablage . . . . .	4
3.1.2 Kommunikation . . . . .	4
3.1.3 Entwicklungsumgebung . . . . .	5
3.1.4 Programmiersprache . . . . .	5
3.1.5 Grobe Zeitplanung . . . . .	5
3.1.6 Meilensteine . . . . .	6
3.1.7 Die Projektbeteiligten . . . . .	7
<b>4 Analyse/Projektplan</b>	8
4.1 IST-Analyse . . . . .	8
4.2 Zeitplan . . . . .	9
4.3 Flussdiagramm Funktionsablauf . . . . .	10
4.4 Lastenheft . . . . .	11
4.4.1 Funktionale Ziele . . . . .	11
4.4.2 Nicht Funktionale Ziele . . . . .	12
4.4.3 Abgrenzungskriterien . . . . .	12
4.4.4 Produkteinsatz . . . . .	12
4.4.5 Produktfunktionen . . . . .	13
4.4.6 Qualitätsanforderungen . . . . .	13
4.4.7 Technische Produktumgebung . . . . .	13
4.5 Risikoanalyse . . . . .	14
<b>5 Entwurfsphase</b>	16
5.1 Spielkonzept . . . . .	16
5.2 Aufbau . . . . .	16
5.3 Zielplattform . . . . .	17



<b>6 Implementierungsphase</b>	18
6.1 Iterative Vorgehensweise . . . . .	18
6.2 Einrichten Raspberry . . . . .	18
6.2.1 Power ON/OFF Button für Raspberry . . . . .	18
6.2.1.1 Verdrahtung am Raspberry . . . . .	19
6.2.1.2 Software – GPIO Überwachung aktivieren . . . . .	20
6.2.1.3 Raspberry Bootloader – update . . . . .	20
6.2.2 RFID Card Reader anschliessen . . . . .	21
6.2.3 Anschluss der Lautsprecher . . . . .	22
6.2.4 Aktivierung des SPI-Interface . . . . .	23
6.2.5 Setup Python-Library für RC522 . . . . .	24
6.3 Implementierung der Datenstrukturen . . . . .	25
6.3.1 Implementierung Audio Ausgabe - Umsetzung . . . . .	25
6.3.1.1 Typ CardMatcher . . . . .	26
6.3.1.2 Typ Category . . . . .	26
6.3.1.3 MaryTTS . . . . .	26
6.3.2 Datenstruktur Audiodateien . . . . .	27
6.4 Implementierung der Benutzeroberfläche . . . . .	27
<b>7 Abnahmephase</b>	28
7.1 Testszenarien . . . . .	28
7.1.1 1. Black-Box-Test . . . . .	28
7.1.2 2. Black-Box-Test . . . . .	29
<b>8 Einführungsphase</b>	30
<b>9 Abschluss</b>	31
9.1 Sachergebnis . . . . .	31
9.1.1 Überprüfung Funktionale Ziele . . . . .	31
9.1.2 Überprüfung Nicht Funktionale Ziele . . . . .	32
9.1.3 Überprüfung Abgrenzungskriterien . . . . .	32
9.1.4 Überprüfung Produkteinsatz . . . . .	33
9.1.5 Überprüfung Produktfunktionen . . . . .	33
9.1.6 Überprüfung Qualitätsanforderungen . . . . .	34
9.1.7 Überprüfung technische Produktumgebung . . . . .	34



9.2 Projektverlauf . . . . .	35
9.2.1 Zeitmanagement . . . . .	35
9.2.2 Kostenmanagement . . . . .	37
9.2.3 Planungsqualität . . . . .	37
9.2.4 Abweichungen . . . . .	38
9.2.4.1 Meilensteine . . . . .	38
9.2.4.2 Remote-Verbindung Visual Studio Code . . . . .	39
9.3 Ausblick in die Zukunft . . . . .	42
9.3.1 Restaktivitäten . . . . .	42
9.3.2 Zukünftige Ergänzungen und Erweiterungen . . . . .	42
9.3.3 Folgeprojekte . . . . .	42
<b>10 Schlussteil/Reflexion</b>	43
10.1 Reflexion Adrian Birrer . . . . .	43
10.2 Reflexion Janik Schilter . . . . .	44
10.1 Projektunterlagen . . . . .	45



## Abkürzungsverzeichnis

- Black-Box-Test:** Bei Black-Box-Testverfahren werden die Testfälle ausschließlich aus der Spezifikation des zu testenden Objekts abgeleitet, ohne den Code zu berücksichtigen.
- GPIO:** General Purpose Input Output - Bezeichnet programmierbare Ein- und Ausgänge, die man für unterschiedliche Zwecke nutzen kann.
- NFC:** Near Field Communication - Daten kontaktlos über eine kurze Distanz austauschen. Der wesentliche Unterschied zu RFID besteht darin, dass ein NFC Gerät nicht nur als Lesegerät, sondern auch als Tag agieren kann.
- Raspberry Pi:** Der Raspberry Pi ist ein Minicomputer auf einer nur Scheckkarten-großen Platine, der sich vielfältig erweitern lässt. Er wurde ursprünglich zu Lern- und Demonstrationszwecken entwickelt, eignet sich aber auch für die Realisierung verschiedener privater oder professioneller Anwendungen.
- RFID:** Radio Frequency Identification - Daten kontaktlos über eine kurze Distanz austauschen.
- TTS:** Text-To-Speech-System - wandelt einen geschriebenen Text in eine Sprachausgabe um.
- URL:** Mit der URL wird eine Adresse bezeichnet, die eine Datei auf einem Server angibt. Bekannter sind die sprachgebräuchlichen Begriffe „Internetadresse“ oder „Webadresse“.



# 1 Vorwort

Während der Weiterbildung zum Dipl. Techniker/in HF Informatik werden mehrere Semesterarbeiten durchgeführt. Wir befinden uns zur aktuellen Zeit im 5. Semester unserer Weiterbildung und dürfen hiermit unsere vierte Semesterarbeit durchführen.

Ziel dieser Arbeit ist, ein Projektthema aus dem Themenbereich der Informatik zu wählen und bei der Durchführung unsere Kenntnisse in verschiedenen Bereichen der Weiterbildung zu festigen und erweitern.

## 1.1 Einleitung

Wir haben uns dazu entschieden eine Software-Applikation in Verbindung mit elektronischen Komponenten zu entwickeln, um unsere Kenntnisse der Programmiersprache Java und in der Elektronik zu erweitern und vertiefen.

Nach reichlichem Überlegen sind wir zum Schluss gekommen, dass wir ein Spiel entwickeln möchten und entschieden uns dazu, ein sogenanntes Klangmemory zu entwickeln.



## 2 Richtlinien und Vorgaben

Die Richtlinien bzw. die Aufgabenstellung wurden folgendermassen definiert:

**Umfang:** ca. 100 Stunden für jedes Projektmitglied

**Thema:** Das Thema kann selbst bestimmt werden, sollte jedoch aus dem Gebiet Software-Engineering, Datenbank, Systemtechnik oder Netzwerktechnik/Telekommunikation stammen.

**Aufgabenstellung:** Die Arbeit wird vorzugsweise in einer zweier Gruppe erarbeitet und soll in erster Linie dazu dienen, die neu erworbenen Kenntnissen sowie Fähigkeiten aus dem Schulunterricht oder auch aus der Praxis zu festigen und zu erweitern. Alle Erkenntnisse sollen in einer schriftlichen Dokumentation festgehalten werden und das Niveau der Arbeit muss einem HF Student im 5. Semester würdig sein. Wert bei der Korrektur wird auf ein systematisches Vorgehen, eine dem Niveau entsprechende Arbeitsweise und ein sauber dokumentierter Bericht gelegt.

**Start Semesterarbeit:** 22. November 2021

**1. Projektsitzung:** 15. Dezember 2021

**2. Projektsitzung:** 16. Februar 2022

**Abgabe Semesterarbeit:** 16. März 2022

**Präsentation:** 23. März 2022 und 30. März 2022



## 3 Initialisierung

### 3.1 Projektorganisation

Die Projektorganisation ist mitentscheidend darüber, ob ein Projekt erfolgreich sein wird oder ob mit strukturellen Problemen zu kämpfen sein wird. Die Projektorganisation beinhaltet die Struktur, Gestaltung, Regeln und Hilfsmittel für die systematische Durchführung eines Projektes.

#### 3.1.1 Ablage

Die schriftliche Dokumentation sowie alle dazu benötigten Unterlagen werden auf einem gemeinsam genutzten OneDrive Konto gespeichert. So haben immer alle Projektmitglieder die gleichen Voraussetzungen und auch den gleichen Stand. Die Daten für die Programmierung werden auf dem Gitblit von Bruno Hammer abgelegt:

<https://birrera@elad.ch/gitblit/r/~{}birrera/>  
Projektarbeit\_Klangmemory.git

Sofern Änderungen an der Programmierung vorgenommen werden, werden diese täglich hochgeladen. So kann das Backup sichergestellt werden und es ist auch möglich, jederzeit auf einen alten Stand zurückzukehren.

#### 3.1.2 Kommunikation

Aufgrund der Corona Pandemie findet die Kommunikation bevorzugt über das «Teams» von Microsoft statt. Ein Account dazu wird von der TEKO zur Verfügung gestellt. Eine weitere Methode um kleinere Sachen zu besprechen ist der Kurznachrichtendienst Whatsapp. Mindestens einmal in der Woche wird über den aktuellen Stand gesprochen und das weitere Vorgehen definiert.



### 3.1.3 Entwicklungsumgebung

Die Entwicklungsumgebung Netbeans, in der Version 8.2, bietet alle benötigten Funktionen, um das Klangmemory umzusetzen. Netbeans ist des weiteren Open Source und auch kostenfrei für die kommerzielle Nutzung.

### 3.1.4 Programmiersprache

Auf Grund der wissensbedingten Kriterien bei der Auswahl der Programmiersprachen, haben sich die Autoren auf die Sprache Java in der Version 8 festgelegt, die auch Open Source und Lizenzfrei für die Kommerzielle Nutzung freigegeben ist.

### 3.1.5 Grobe Zeitplanung

Phase	Zeitaufwand geschätzt
Initialisierungsaufwand	7h
Analyse- und Entwurfsphase	23h
Implementierungs- / Abhahme- / Einführungsphase	136h
Abschluss- / Dokumentationsphase	36 h
<b>TOTAL</b>	<b>202 h</b>

Tabelle 1: Zeitplanung

### 3.1.6 Meilensteine

Folgende Tabelle ist ein Auszug aus dem erstellten Zeitplan. Der komplette Zeitplan ist im Anhang ersichtlich.

<u>Meilensteine / Termine</u>					
1. Projektsitzung		15.12.2021	15.12.2021	14:30	18:00
2. Projektsitzung		16.02.2022	16.02.2022	14:30	18:00
Abgabe Projektarbeit		16.03.2022	16.03.2022	17:00	
Präsentation Gruppe 1		23.03.2022	23.03.2022	12:30	18:00
Präsentation Gruppe 2		30.03.2022	30.03.2022	12:30	18:00
Ferienabwesenheit	JS	24.12.2021	26.12.2021		
Ferienabwesenheit (ZügeIn)	AB	25.12.2021	31.12.2021		

Abbildung 2: Meilensteine

### 3.1.7 Die Projektbeteiligten



- Projektleiter
- Janik Schilter
- Tel: 079'454'65'14
- E-Mail: janik.schilter@edu.teko.ch

Abbildung 3: Janik Schilter

---



- Projektmitglied
- Adrian Birrer
- Tel.: 078'717'40'34
- E-Mail: adrian.birrer@edu.teko.ch

Abbildung 4: Adrian Birrer

---

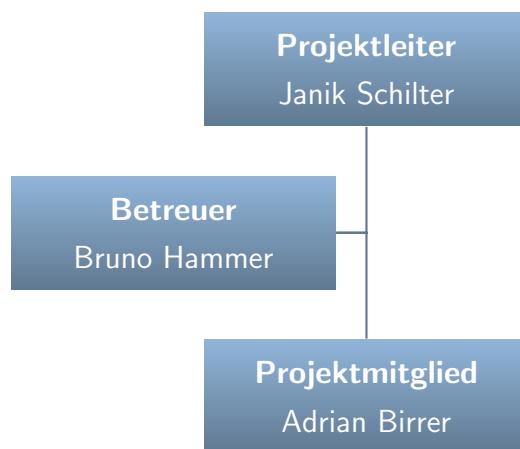


Abbildung 5: Organigramm



## 4 Analyse/Projektplan

### 4.1 IST-Analyse

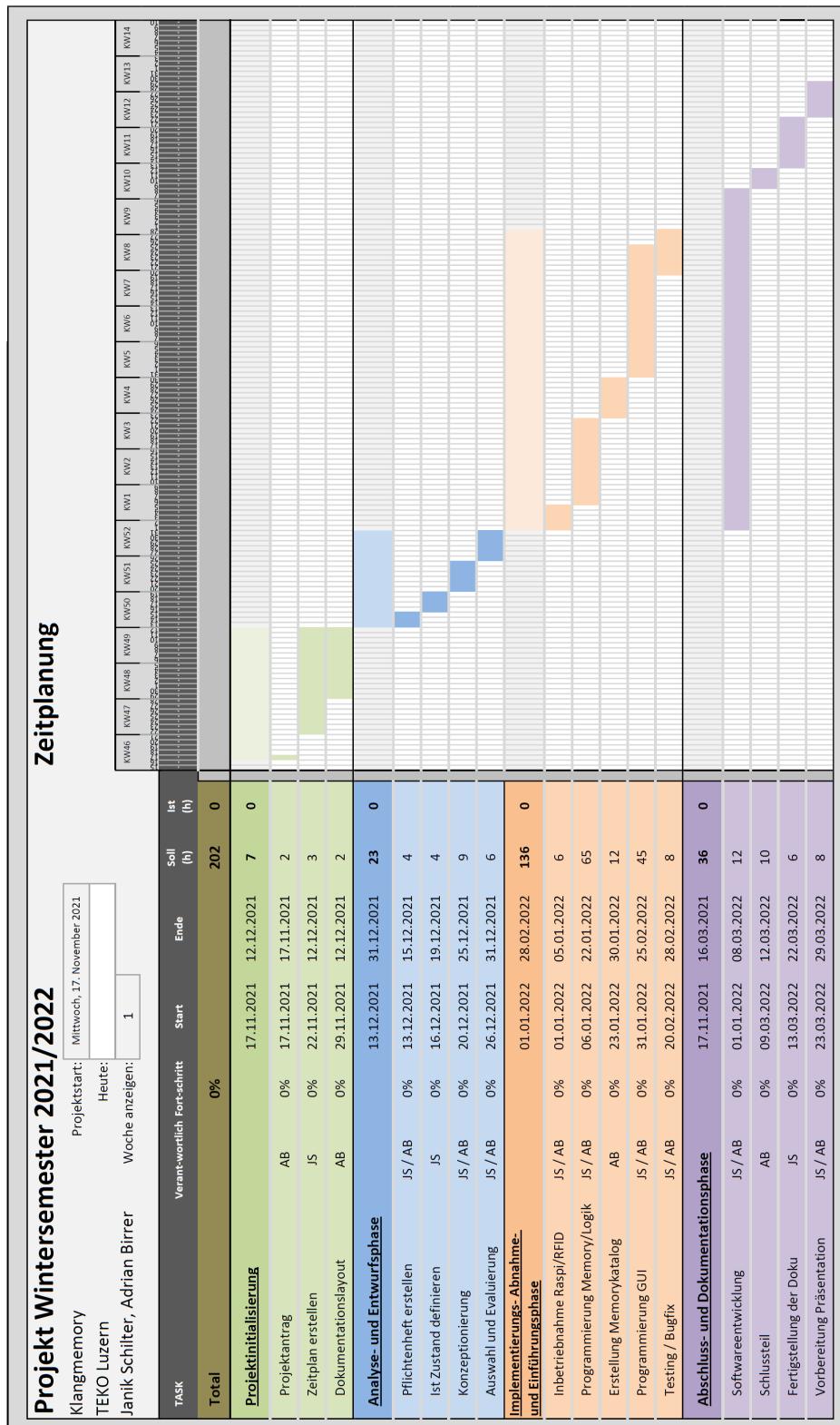
Zur Zeit des Projektbeginnes starten die Autoren mit vier Semestern an Informatikwissen. Diese beinhalten Grundlagen in Java, Datenbanken/SQL sowie unter anderem Betriebssysteme und Netzwerktechnik.

Da der Projektbeteiligte Adrian Birrer im Frühling stolzer Vater wurde, kam der Wunsch auf, ein Spiel für seinen kleinen Junior zu entwickeln. Somit entschieden sich die Projektbeteiligten ein sogenanntes Klangmemory zu erstellen. Dieses funktioniert wie ein gewöhnliches Memory, aber statt mit Bildern wird mit Klängen gespielt. Ausserdem sind Klangmemorys eine tolle Möglichkeit für Kinder zum Erkennen von Geräuschen und zur spielerischen Förderung der kognitiven Fähigkeiten.

Klangmemorys als Produkt sind bereits auf dem Markt erhältlich. Jedoch möchten die Projektbeteiligten ein Klangmemory entwickeln, welches verschiedene Klangkategorien besitzt und mit verhältnismässig geringem Aufwand mit neuen Kategorien ergänzt werden kann. Dadurch kann der Spielspass länger erhalten bleiben. Ausserdem wird die Möglichkeit vorgesehen, das Spiel auch in französischer oder englischer Sprache zu spielen.



## 4.2 Zeitplan



### 4.3 Flussdiagramm Funktionsablauf

Das Flussdiagramm ist eines der am häufigsten verwendeten Diagramme, um einen Prozess oder Arbeitsablauf darzustellen. Flussdiagramme beschreiben einen Prozess mit einer festen Reihenfolge der Arbeitsprozesse. Schritt für Schritt werden so komplexe Abläufe verständlich dargestellt.

Zur vereinfachten Darstellung sind im erstellten Diagramm nur vier mögliche Memorykarten abgebildet:

Karten mit Katzengeräusch: *Card-Cat1 + Card-Cat2*

Karten mit Hundegeräusch: *Card-Dog1 + Card-Dog2*

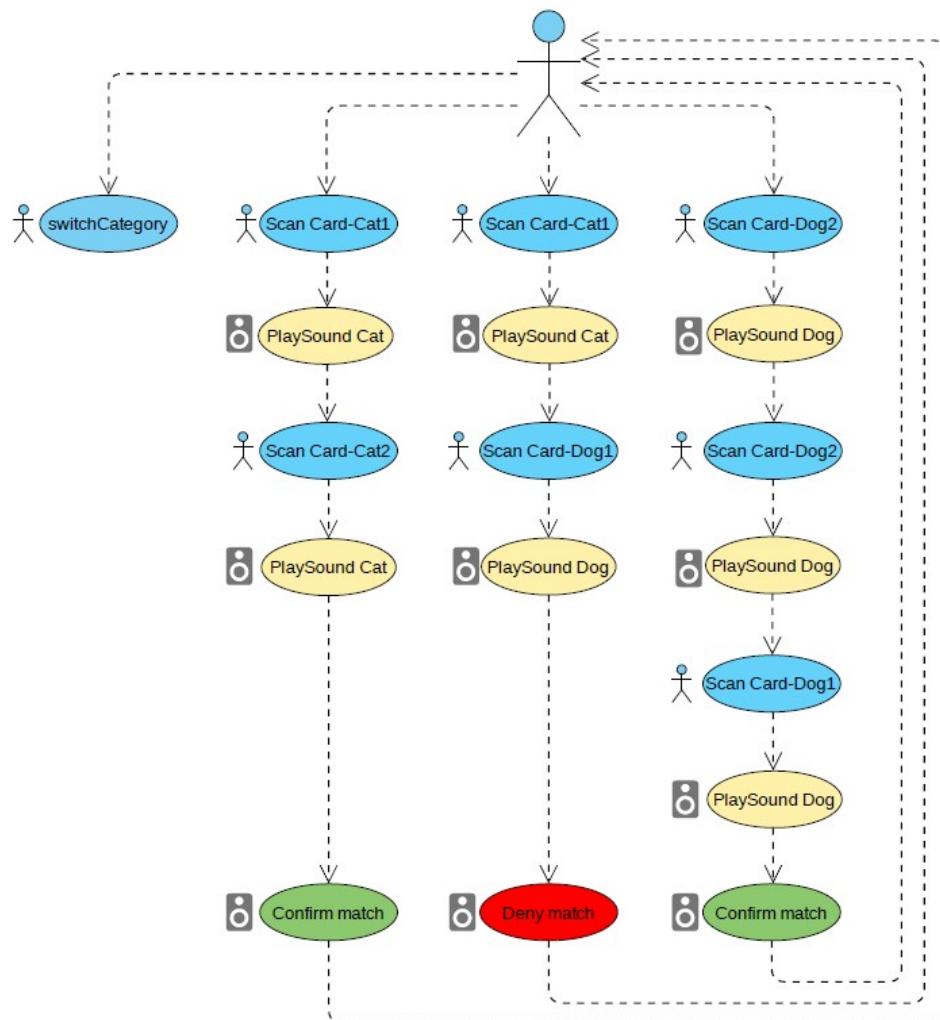


Abbildung 6: Use-Case-Diagramm



## 4.4 Lastenheft

Nachfolgend werden die Ziele nach Funktionalen- und nicht Funktionalen Zielen unterschieden. Funktionale Ziele sind meist von Produkt zu Produkt unterschiedlich und meist nur für ein Produkt anwendbar. Hingegen können die nicht Funktionalen Ziele auf mehrere Produkte, ganze Designs oder gesamte Architekturen angewendet werden. Alle Ziele werden klassifiziert nach Kann oder Muss Kriterien. Muss Kriterien müssen zwingend erfüllt werden, ansonsten kann das Projekt abgebrochen werden. Hingegen die Kann Ziele sind nicht zwingend nötig, sollten aber ein Bestandteil des Projektergebnis sein. Gründe für ein nicht enthalten sind Kosten oder Zeitgründen.

### 4.4.1 Funktionale Ziele

#	Beschrieb	Kann/Muss
1.0	Das Klangmemory soll von Kindern ab 3 Jahren bedient werden können.	Muss
2.0	Das Klangmemory muss ohne Natel, Tablet oder PC gespielt werden können.	Muss
3.0	Das Memory soll aus mindestens 20 Karten (10 Paare) bestehen.	Muss
4.0	Jede Karte sollte mindestens 3 verschiedene Geräusche abspielen können. Nicht willkürlich ein Geräusch, sondern es können dementsprechende Kategorien vor dem Spiel gewählt werden wie z.B. Tiergeräusche oder Fahrzeuge.	Muss
5.0	Jedes abgespielte Geräusch sollte zwischen 3 und 10 Sekunden dauern.	Muss
6.0	Über eine Benutzeroberfläche können die Karten mit neuen Geräuschen bespielt werden. Dabei können die Kategorien erweitert oder gelöscht werden.	Kann
6.1	Über die Benutzeroberfläche soll es auch möglich sein, neue zusätzliche Karten zu erstellen oder alte defekte zu ersetzen.	Kann

Tabelle 2: Funktionale Ziele



#### 4.4.2 Nicht Funktionale Ziele

#	Beschrieb	Kann/Muss
1.0	Das Memory soll keine jährlichen Gebühren verursachen.	Muss
2.0	Als Programmiersprache soll hauptsächlich Java verwendet werden. Sollte dies nicht möglich sein oder der Aufwand zu hoch, dann kann auf eine andere Sprache ausgewichen werden.	Kann
3.0	Das Memory muss transportiert werden können.	Muss

Tabelle 3: Nicht Funktionale Anforderungen

#### 4.4.3 Abgrenzungskriterien

Zum Projektergebnis gehört nur die Softwareentwicklung und die eingesetzte Hardware. Die Herstellung der Memory Karten sowie auch der Box ist nicht im Projekt enthalten und wird separat hergestellt.

#### 4.4.4 Produkteinsatz

Anwendungsbereiche:

Das Produkt wird im privaten Bereich genutzt werden. Somit ist zum Zeitpunkt der Projektdurchführung keine Markteinführung geplant.

Zielgruppe:

Kinder ab 3 Jahren



#### 4.4.5 Produktfunktionen

Das Klangmemory soll über mindestens drei Kategorien verfügen.

Mögliche Kategorien:

- Tiergeräusche
- Fahrzeugklänge
- Stimmen von bekannten Persönlichkeiten

Die Lautstärke der Geräusche soll über einen Drehregler eingestellt werden können.

#### 4.4.6 Qualitätsanforderungen

Kartenleser:

Bei Auflegen der Memorykarte soll innerhalb einer Sekunde ein Geräusch abgespielt werden.

Lautsprecher:

Die Qualität der abgespielten Klänge soll für den Spieler angenehm und verständlich sein.

#### 4.4.7 Technische Produktumgebung

Hardware:

- Raspberry Pi 4 Model B
- RFID-Leser: RFID-RC522
- NFC-Tags: Protokoll 13,56 MHz ISO14443A Universal Label RFID Tags
- Lautsprecher: manhattan 2600 Series Speaker System

Entwicklungsumgebung:

- Visual Studio Code

Programmiersprache:

- Java Version 8

## 4.5 Risikoanalyse

Das Ziel einer Risikoanalyse ist, sich mit möglichen Risiken eines Projektes auseinander zu setzen und deren bewusst zu werden. Risiken können, im Fall des Eintritts, grossen Schaden zufügen, daher gilt es, wenn möglich diese zu minimieren. In einem ersten Verfahren werden alle möglichen Risiken eruiert, welche dann in einer zweiten Phase nach der Eintrittswahrscheinlichkeit und dem Schadenspotential in einer Risikomatrix eingeteilt werden. Sind das Schadenspotential und die Wahrscheinlichkeit besonders hoch, ist es unausweichlich, dass Massnahmen ergriffen werden müssen, um den Eintritt zu verhindern. Ist im Gegensatz das Potential sowie auch die Wahrscheinlichkeit auf einen Eintritt tief, ist es gut, wenn das Risiko erkannt wird, aber es müssen keine Massnahmen ergriffen werden.

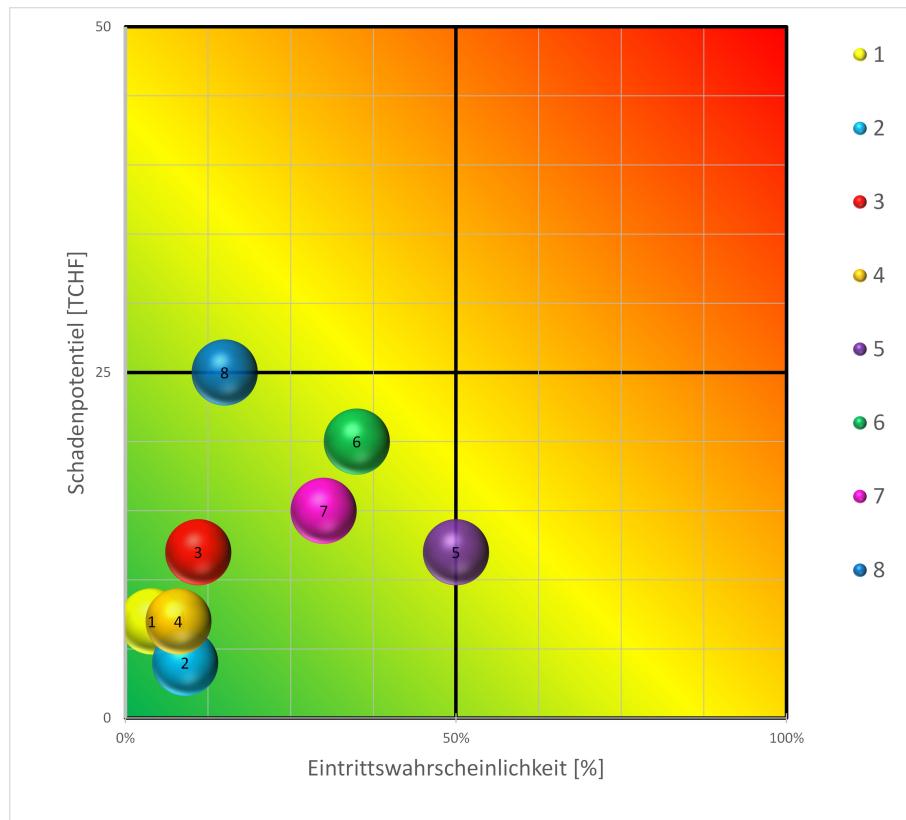


Abbildung 7: Risikoanalyse

Wie die Risikoanalyse nun klar aufzeigt, gibt es in dieser Analyse vier Kategorien, welche unter höherer Beobachtung stehen müssen aber ein Eingreifen nicht zwingend notwendig ist.



Nr.	Kategorie (Liste)	Eintritts- wahrschein- lichkeit [%]	Schadenspotential [TCHF]	Risiko [TCHF]
1	Anforderungen sind unvollständig	4%	7	0.28
2	Abweichungen vom Qualitätsziel	9%	4	0.36
3	Ausfall Projektleiter/-Mitarbeiter	11%	12	1.32
4	Konflikte zwischen Mitarbeitern	8%	7	0.56
5	Verfügbarkeit elektronische Komponenten	50%	12	6
6	Hardware- und Softwareprobleme	35%	20	7
7	Reichweite RFID-Leser	30%	15	4.5
8	Datenverlust (Backup)	15%	25	3.75

Abbildung 8: Kategorien Risikoanalyse

## #5 Verfügbarkeit elektronischer Komponenten

Ein großes Problem ist die Halbleiterknappheit, in den Medien oft auch „Chipknappheit“ genannt. Sie wurde primär durch die Pandemie verursacht. Produktionswerke mussten herunterfahren, während die Nachfrage deutlich stieg. Damit das Projekt nicht an der Verfügbarkeit von Komponenten scheitert, sind diese frühzeitig zu bestellen und allenfalls Alternativen zu überlegen.

## #6 Hardware- und Softwareprobleme

Softwareentwicklung ist sehr dynamisch und komplex. Auch verschiedene Hardware Komponenten die zusammenspielen müssen, können zu einem komplexen System heranwachsen. Wichtig ist hierbei, dass laufend getestet und nötigenfalls eingegriffen wird.

## #7 Reichweite RFID-Leser

Die Reichweite des RFID-Lesers ist abhängig von dem Transponder und dem Lesegerät und beläuft sich nur auf wenige Zentimeter. Es ist daher notwendig, dass bereits beim Kauf darauf geachtet wird, dass die Reichweite den Ansprüchen genügt und dass dies auch getestet wird unter den effektiven Bedingungen. Sollte die Reichweite nicht ausreichen, gibt es noch eine Möglichkeit von der Reichweiten Erhöhung. Sollte auch dies nicht ausreichen, ist eine zwingende Anpassung notwendig bevor die Karten und auch die Box erstellt werden.

## #8 Datenverlust (Backup)

Ein Datenverlust kann jederzeit passieren. Daher ist es wichtig, dass die Daten nicht nur lokal, sondern auch noch anderweitig gespeichert werden. Die Autoren haben sich, wie üblich in der Softwareentwicklung, sich für ein regelmässiges Backup via Versionsverwaltung entschieden.



## 5 Entwurfsphase

### 5.1 Spielkonzept

Die Memorykarten beinhalten einen integrierten NFC-Tag. Wird eine Memorykarte auf den RFID-Leser gelegt, so erfolgt ein Klang welcher je nach ausgewählter Kategorie für diese Karte bestimmt ist. Beim konventionellen Memory gibt es jeweils zwei optisch identische Karten, jedoch ist beim Klangmemory jeweils der Klang identisch. Werden zwei akustisch identische Karten nacheinander auf den RFID-Leser gehalten, so erfolgt eine akustische Bestätigung für das korrekte Auflegen.

Die Auswahl der verschiedenen Kategorien erfolgt über separate Karten. Auch diese Karten werden über den RFID-Leser gehalten und dadurch wird die Kategorie im System geändert.

Falls die Projektressourcen genügen, wird zusätzlich ein simples Webinterface entwickelt. Der Benutzer hat in diesem Webinterface die Möglichkeit, zusätzliche Kategorien zu erstellen oder bestehende Kategorien anzupassen oder auch die Sprache zu ändern.

### 5.2 Aufbau

Als Basis für das Klangmemory wird ein Raspberry Pi verwendet, auf welchem das Spiel ausgeführt wird. Da Adrian gelernter Schreiner ist, wird das Raspberry Pi in einer Holzkiste "versteckt". Die Holztruhe beinhaltet Lautsprecher, einen ON/OFF-Button und einen RFID-Leser. Die Memorykarten bestehen jeweils aus zwei zusammengeklebten Holzfunieren mit einem NFC-Tag dazwischen. Die Schreinerarbeiten sind jedoch nicht Bestandteil des Projektes und der Projektplanung.



## 5.3 Zielplattform

Als Betriebssystem für den Raspberry wurde das Raspberry PI OS Lite (32-Bit) installiert. Dieses Betriebssystem ist die offizielle Standarddistribution von der Raspberry Pi Foundation und wurde zudem auf die eingebaute Hardware optimiert. Auch haben die Autoren bereits gute Erfahrungen mit diesem Produkt gemacht, so dass kein anderes Betriebssystem in Frage kam.



## 6 Implementierungsphase

### 6.1 Iterative Vorgehensweise

Es gibt verschiedene Vorgehensmodelle um eine Software zu entwickeln. Jedes Modell hat seine Berechtigung und wird in der Softwareentwicklung unterschiedlich eingesetzt. Für die Entwicklung des Klangmemorys setzten die Entwickler auf die iterative Vorgehensweise.

Bei der iterativen Entwicklung werden einzelne Phasen mehrmals durchlaufen. Innerhalb jeder Iteration wird die Software geändert, weiterentwickelt und erweitert. Da die Software schrittweise, also iterativ verbessert wird, entsteht keine Notwendigkeit die vollständige Spezifikation zu Beginn zu erstellen. Es werden nur die wichtigsten Anforderungen definiert. Der iterative Ansatz bietet Möglichkeiten, im Rahmen des Projektablaufes auf Änderungen der Anforderungen flexibel zu reagieren. So bietet es den Entwicklern auch die Chance, laufend über mögliche Ergebnisse zu entscheiden.

### 6.2 Einrichten Raspberry

Nachfolgend werden alle nötigen Schritte und Installationen erläutert, damit das Raspberry auch funktionsfähig wird, für die einzelnen Audiodateien abspielen zu können.

#### 6.2.1 Power ON/OFF Button für Raspberry

Damit der Raspberry Pi eingeschaltet werden kann, ohne dass das USB-Kabel gezogen werden muss, wird ein ON/OFF-Button installiert. Über die GPIOs verfügt der Raspberry nun die Möglichkeit, ein einfaches Ein- und Ausschalten über einen On/OFF-Button zu realisieren.

### 6.2.1.1 Verdrahtung am Raspberry

Der Taster wird am GPIO3 (gelbe Leitung) und an einem beliebigen GND-Pin (schwarze Leitung) verbunden.

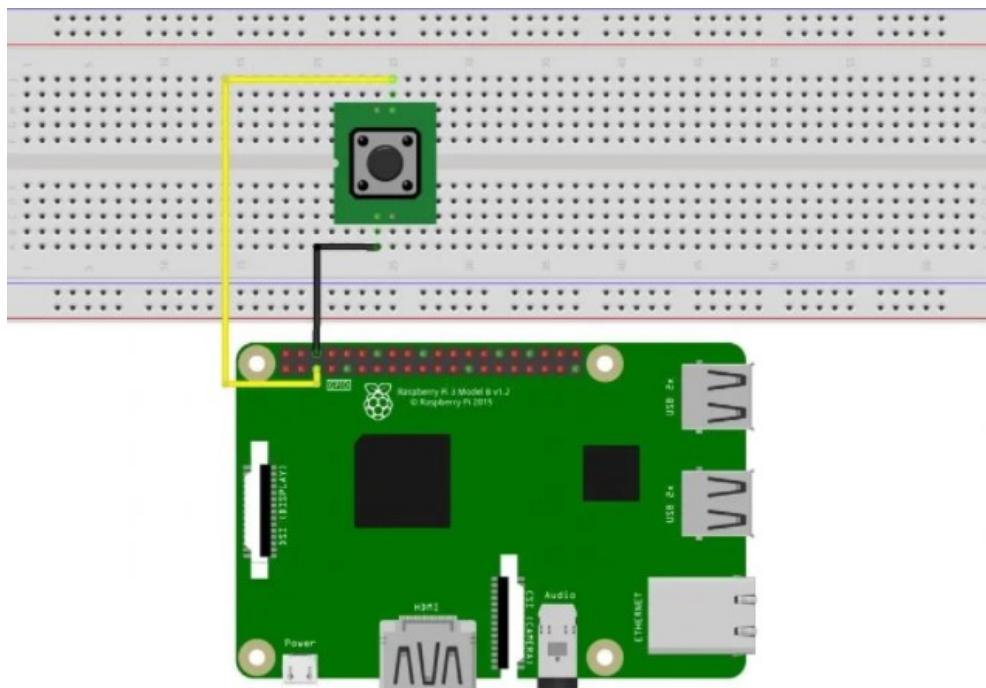


Abbildung 9: ON/OFF-Button / Verdrahtung an Raspberry

GPIO Pin Out-Header:

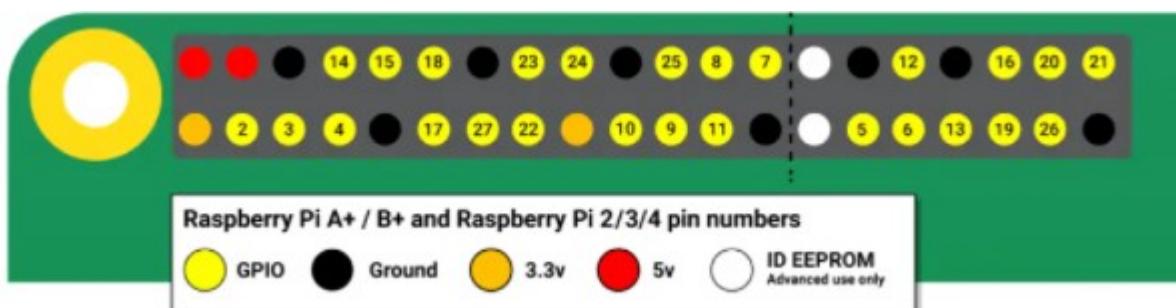


Abbildung 10:  $\text{GPIO}_{PinOut} - \text{Header}$



### 6.2.1.2 Software – GPIO Überwachung aktivieren

Die Konfiguration des ON/OFF-Button wird wie folgt ausgeführt:

```
sudo nano /boot/config.txt
```

Folgender Inhalt wird an das Ende der Datei hinzugefügt:

```
# Ein- und Ausschalten  
dtoverlay= gpio-shutdown, gpio-pin=3, active_low=1, gpio_pull=up
```

### 6.2.1.3 Raspberry Bootloader – update

Für das erfolgreiche Bootloader-Update wird eine Internetverbindung benötigt:

```
sudo apt update  
sudo apt full-upgrade  
sudo apt install rpi-eeprom  
sudo rpi-eeprom-update
```

Im Anschluss wird ein reboot durchgeführt:

```
sudo reboot
```

### 6.2.2 RFID Card Reader anschliessen

Als erstes wird der RFID-Leser gemäss untenstehendem Schema mit dem Raspberry Pi verdrahtet:

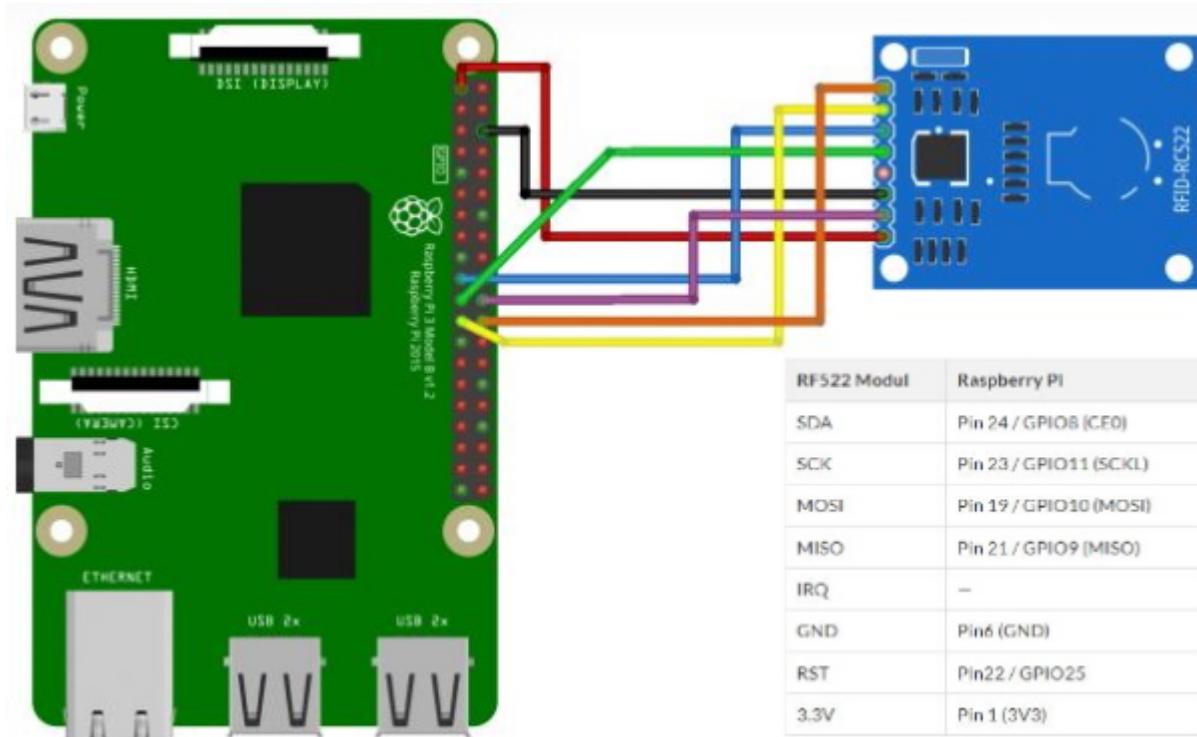


Abbildung 11: RFID Card Reader Anschluss an Raspberry



### 6.2.3 Anschluss der Lautsprecher

Die gewählten Lautsprecher *manhattan 2600 Series Speaker System* werden über USB 3.0 und den 3.5mm Audio Jack mit dem Raspberry Pi verbunden.

Damit der Audio-Output über den im Raspberry Pi verbauten Audio-Jack funktioniert führen wir folgenden Befehl aus:

```
amixer cset numid=3 2
```

Zudem wird in der Boot-Config folgende Anpassung erstellt:

```
sudo nano /boot/config.txt
```

```
# Disabling HDMI-Sound
hdmi_drive=2
hdmi_ignore_hotplug=1
enable_tvout=1
```



#### 6.2.4 Aktivierung des SPI-Interface

Um das RFID RC522 Shield verwenden zu können brauchen wir den SPI Bus. Damit der Kernel beim Starten geladen wird, gehen wir wie folgt vor:

```
sudo raspi-config
```

Im geöffneten Configuration Tool wählen wir *Interface-Options* und bestätigen im Anschluss das Aktivieren des SPI-Interface durch Bestätigung mit *Yes*.

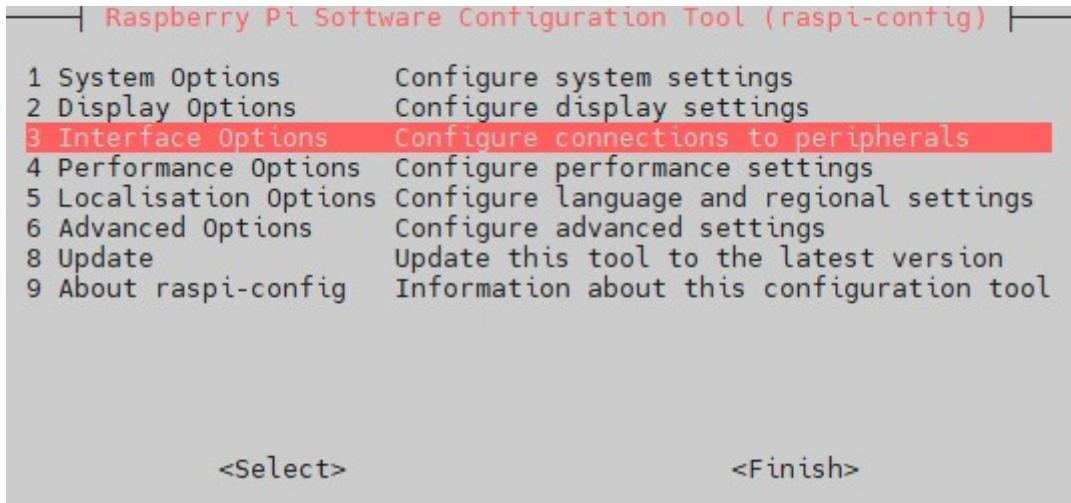


Abbildung 12: Raspberry Pi-config

Nun starten wir den Raspberry Pi neu, damit die angepasste Boot-Config geladen wird. Mit folgendem Befehl wird kontrolliert ob die Aktivierung des SPI-Interface erfolgreich war.

```
lsmod | grep spi
```

Falls im Anschluss *spi\_bcm2835* gelistet wird, ist das Interface aktiv:

```
spidev 20480 0  
spi_bcm2835 24576 0
```



### 6.2.5 Setup Python-Library für RC522

Damit mit dem RFID-Leser interagiert werden kann, installieren wir gemäss folgenden Schritten eine Python-Library:

Im ersten Schritt wird der Raspberry Pi aktualisiert damit sicher gestellt werden kann, dass die aktuellste Version verwendet wird:

```
sudo apt-get update  
sudo apt-get upgrade
```

Im Anschluss werden die `python3-dev` und `python-pip` packages installiert:

```
sudo apt-get install python3-dev python3-pip
```

Nun wird mithilfe von `python pip` die Python Library `spidev` installiert. Die Library `spidev` wird für die Interaktion mit dem RFID-Leser benötigt:

```
sudo pip3 install spidev
```

Als Nächstes wird mit `python pip` die Library des MFRC522 installiert.

```
sudo pip3 install mfrc522
```

## 6.3 Implementierung der Datenstrukturen

Nachfolgend wird ein Klassendiagramm dargestellt. Klassendiagramme sind Strukturdiagramme innerhalb der Unified Modeling Language (UML) und sie veranschaulichen Systeme der objektorientierten Programmierung. Sie dienen dazu, die Beziehung einer Klasse zu einer anderen darzustellen und repräsentieren auch die Attribute des Systems.

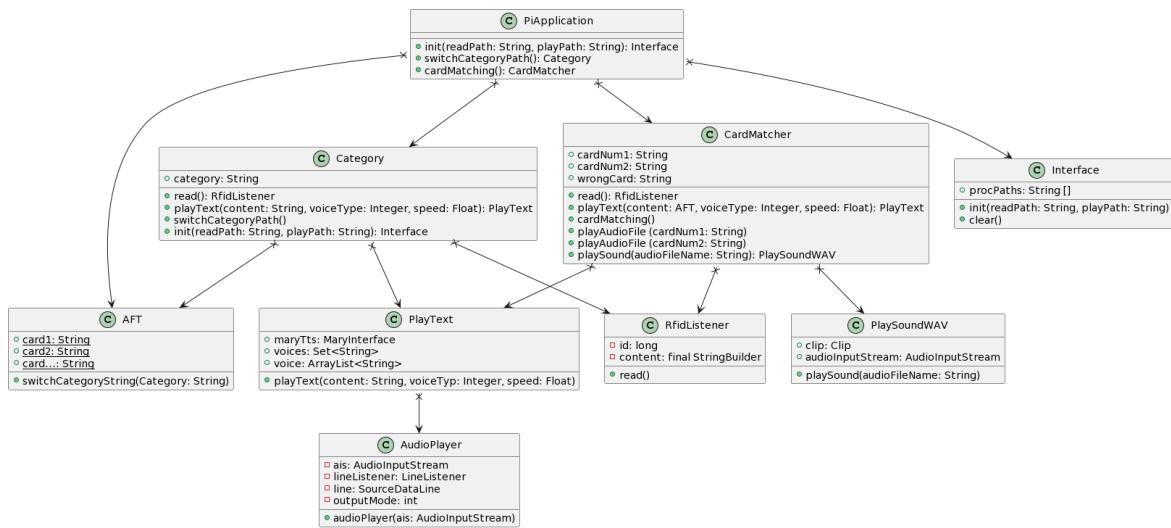


Abbildung 13: UML

### 6.3.1 Implementierung Audio Ausgabe - Umsetzung

Jeder RFID-Tag wird vorgängig mit einer eindeutigen Nummer bzw. Text beschrieben (z.B. 1.1 = Karte 1 von Audio 1 wenn es sich um eine Audiokarte und mit z.B. Kat.1 wenn es sich um eine Kategorie handeln soll).

Mit dem Start des Programms wird die Main Klasse PiApplication aufgerufen. Diese Klasse wurde ziemlich schlank gehalten um eine gute Übersicht zu behalten. In dieser wird nur der Pfad der Audiodateien hinterlegt und zwei Konstruktoren um ein neues Objekt vom Typ CardMatcher oder vom Typ Category zu erzeugen. Ein solches Objekt wird erzeugt, in dem ein RFID-Tag am Leser gescannt und ausgewertet wird. Damit sich das Programm nicht beendet, wird die Klasse PiApplication in einer Endlosschleife betrieben.



### 6.3.1.1 Typ CardMatcher

Wenn der RFID-Tag mit einer Nummer von z.B. 1.1 beschrieben ist, wird als erstes der String cardNum1 beschrieben und das dazugehörige AudioFile über den PlaySoundWAV abgespielt. Der zweite RFID-Tag wird dann als cardNum2 gespeichert. Wie schon beim cardNum1 wird auch hier das dazugehörige AudioFile wiedergegeben. In einem nächsten Verfahren wird nun die cardNum2 mit der cardNum1 verglichen. Handelt es sich um das gleiche Audio aber nicht um die gleiche Karte, wird ein entsprechender Text abgespielt aus der AFT Klasse, welcher dem Nutzer gratuliert, dass er das korrekte Paar gefunden hat. Ist dies nicht der Fall, kommt der Text: „Sorry, das war wohl nichts“. Diese Texte werden mit Hilfe einer Sprachsynthese von MaryTTS (Text-To-Speech-System) kreiert sowie wiedergegeben und könnten jederzeit angepasst werden. Nachdem die zweite Karte abgespielt wurde, wird dieses Objekt wieder gelöscht.

### 6.3.1.2 Typ Category

Wenn der RFID-Tag mit einem Text wie z.B. Kat.2 beschrieben ist, wird die Kategorie entsprechend dem Tag gewechselt. Bei jedem Systemstart ist standardmäßig die Kategorie 1 ausgewählt. Ein Kategoriewechsel wird akustisch mit der Sprachsynthese von MaryTTS angekündigt. Nachdem die Kategorie gewechselt ist, wird dieses Objekt wieder gelöscht.

### 6.3.1.3 MaryTTS

MaryTTS ist open source und eine in Java geschriebene Syntheseplattform. Sie wurde ursprünglich als Gemeinschaftsprojekt entwickelt und ist in der aktuellsten Version 5.2 erhältlich. Dabei werden die Sprachen Deutsch, Britisch- und Amerikanisches Englisch, Französisch, Italienisch, Luxemburgisch, Russisch, Schwedisch, Telugu und Türkisch angeboten.

Bei der Sprachsynthese wird die menschliche Stimme durch den Computer erzeugt. Ein Text-To-Speech-System (TTS) wandelt dabei geschriebenen Text in eine Sprachausgabe um. Die maschinelle Erzeugung der menschlichen Sprache ist kompliziert, aber es wurden in den letzten Jahren grosse Fortschritte erzielt. Gegenüber



der Wiedergabe von vorgefertigten Sprachaufnahmen hat eine TTS den Vorteil, sehr flexibel beliebige Texte zu sprechen. Das System verwendet grosse Bibliotheksdateien, die heruntergeladen und in einem eigenen Projekt verwendet werden können.

### 6.3.2 Datenstruktur Audiodateien

Die Audiodateien werden auf dem Raspberry Pi in einem eigenen Ordnersystem hinterlegt. In welchem spielt keine Rolle, diese muss einfach korrekt in der Klasse PiApplication definiert werden. Für jede Kategorie muss ein Ordner bestehen und diese Ordner wiederum mit je 12 Audiodateien (Audio 1 bis Audio 12). So kann auf einfache Weise die Kategorie gewechselt werden und es spielt eine andere Audiodatei ab, sofern in den anderen Kategorien auch ein anderes File gespeichert wird.

.. / Kat.1	.. / Kat.2	.. / Kat.3
.. / Kat.1 / Audio1	.. / Kat.2 / Audio1	.. / Kat.3 / Audio1
.. / Kat.1 / Audio2	.. / Kat.2 / Audio2	.. / Kat.3 / Audio2
.. / Kat.1 / Audio3	.. / Kat.2 / Audio3	.. / Kat.3 / Audio3
.. / Kat.1 / Audio4	.. / Kat.2 / Audio4	.. / Kat.3 / Audio4
.. / Kat.1 / Audio5	.. / Kat.2 / Audio5	.. / Kat.3 / Audio5
.. / Kat.1 / Audio6	.. / Kat.2 / Audio6	.. / Kat.3 / Audio6
.. / Kat.1 / Audio7	.. / Kat.2 / Audio7	.. / Kat.3 / Audio7
.. / Kat.1 / Audio8	.. / Kat.2 / Audio8	.. / Kat.3 / Audio8
.. / Kat.1 / Audio9	.. / Kat.2 / Audio9	.. / Kat.3 / Audio9
.. / Kat.1 / Audio10	.. / Kat.2 / Audio10	.. / Kat.3 / Audio10
.. / Kat.1 / Audio11	.. / Kat.2 / Audio11	.. / Kat.3 / Audio11
.. / Kat.1 / Audio12	.. / Kat.2 / Audio12	.. / Kat.3 / Audio12

Abbildung 14: Struktur Audiodateien

### 6.4 Implementierung der Benutzeroberfläche

Die Erstellung einer Benutzeroberfläche ist ein Kann Ziel. Da die Programmierung erheblich mehr Zeit in Beanspruchung nahm, haben sich die Autoren dazu entschieden, auf eine Benutzeroberfläche zu verzichten.



## 7 Abnahmephase

In dieser Phase wird die Anwendung überprüft und mit den Anforderungen verglichen, bevor man sie an die Endnutzer freigibt. Mit Tests können Fehler nachgewiesen und verbessert werden, bevor es im Live-Betrieb verwendet wird. Daher ist diese Phase sehr wichtig und soll frühzeitig begonnen werden.

### 7.1 Testszenarien

Je zwei RFID Tags spielen die gleiche Audiodatei ab. Da dies alles akustisch abläuft, wurde darauf verzichtet, Unit Tests zu schreiben. Stattdessen wurden laufend manuelle Tests durchgeführt. Wicht das Ergebnis vom erhofften Ergebnis ab, wurde sofort interveniert und nachgebessert. Mit Beendigung der Programmierung, wurden dann Systemtests durchgeführt.

#### 7.1.1 1. Black-Box-Test

Der erste Black-Box-Test verlief nicht zufriedenstellend. Nach mehreren korrekten Abspielungen von den Audiodateien, funktionierte plötzlich nichts mehr. Das System warf nur noch Fehlermeldungen.

```
Mar 19, 2022 8:41:15 AM AudioPlayerText.AudioPlayer run
WARNING: null
java.lang.IllegalArgumentException: No line matching interface SourceDataLine supporting format PCM_SIGNED 16
000.0 Hz, 16 bit, mono, 2 bytes/frame, little-endian is supported.
    at javax.sound.sampled.AudioSystem.getLine(Unknown Source)
    at AudioPlayerText.AudioPlayer.run(AudioPlayer.java:275)

gestartet
cancel durchlaufen
```

Abbildung 15: Fehlermeldung Black-Box-Test



### 7.1.2 2. Black-Box-Test

Nach langem Suchen und Ausprobieren wurde festgestellt, das nach jedem Abspielen einer Audiodatei ein neuer Dienst auf dem Raspberry geöffnet wurde. Die alten Dienste wurden nicht entfernt und dies führte schlussendlich zu den oben genannten Fehlermeldungen.

Da die Projektressourcen zu diesem Zeitpunkt bereits aufgebraucht waren, konnte diese Problematik im Rahmen dieser Projektarbeit nicht gelöst werden. Die Projektbeteiligten haben sich jedoch dazu entschieden, diese Problematik zu gegebener Zeit, ausserhalb der Projektarbeit, zu beheben.



## 8 Einführungsphase

Um dieses Projekt in einer eigenen Umgebung zu starten, kann man sich die Anwendung mittels folgender URL auf den eigenen Rechner laden und mit der Anleitung im Kapitel 6 *Implementierungsphase* alle notwendigen Schritte für das korrekte Setup eines eigenen Klangmemories unternehmen.

[https://schiltej@elad.ch/gitblit/r/~birrera/  
Projektarbeit\\_Klangmemory.git](https://schiltej@elad.ch/gitblit/r/~birrera/Projektarbeit_Klangmemory.git)



## 9 Abschluss

### 9.1 Sachergebnis

#### 9.1.1 Überprüfung Funktionale Ziele

#	Beschrieb	Kann/Muss	Ziel erreicht
1.0	Das Klangmemory soll von Kindern ab 3 Jahren bedient werden können.	Muss	Ja
2.0	Das Klangmemory muss ohne Natel, Tablet oder PC gespielt werden können.	Muss	Ja
3.0	Das Memory soll aus mindestens 20 Karten (10 Paare) bestehen.	Muss	Ja
4.0	Jede Karte sollte mindestens 3 verschiedene Geräusche abspielen können. Nicht willkürlich ein Geräusch, sondern es können dementsprechende Kategorien vor dem Spiel gewählt werden wie z.B. Tiergeräusche oder Fahrzeuge.	Muss	Ja
5.0	Jedes abgespielte Geräusch sollte zwischen 3 und 10 Sekunden dauern.	Muss	Ja
6.0	Über eine Benutzeroberfläche können die Karten mit neuen Geräuschen bespielt werden. Dabei können die Kategorien erweitert oder gelöscht werden.	Kann	Nein
6.1	Über die Benutzeroberfläche soll es auch möglich sein, neue zusätzliche Karten zu erstellen oder alte defekte zu ersetzen.	Kann	Nein

Tabelle 4: Funktionale Ziele

#### Funktionalen Ziele #1.0 - #5.0:

Diese Ziele konnten vollumfänglich erreicht werden.

#### Funktionalen Ziele #6.0 - #6.1:

Da die Abweichungen im Projektverlauf (siehe Punkt 9.2.1) mit erheblichen Mehraufwänden einhergingen, entschieden sich die Projektbeteiligten auf die Umsetzung der *Kann-Ziele* zu verzichten.



### 9.1.2 Überprüfung Nicht Funktionale Ziele

#	Beschrieb	Kann/Muss	Ziel erreicht
1.0	Das Memory soll keine jährlichen Gebühren verursachen.	Muss	Ja
2.0	Als Programmiersprache soll hauptsächlich Java verwendet werden. Sollte dies nicht möglich sein oder der Aufwand zu hoch, dann kann auf eine andere Sprache ausgewichen werden.	Kann	Ja
3.0	Das Memory muss transportiert werden können.	Muss	Ja

Tabelle 5: Nicht Funktionale Anforderungen

#### Nicht Funktionalen Ziele #1.0 - #3.0:

Diese Ziele konnten vollumfänglich erreicht werden.

### 9.1.3 Überprüfung Abgrenzungskriterien

#### Definierte Abgrenzungskriterien

*Zum Projektergebnis gehört nur die Softwareentwicklung und die eingesetzte Hardware. Die Herstellung der Memory Karten sowie auch der Box ist nicht im Projekt enthalten und wird separat hergestellt.*

#### Überprüfung

Die Abgrenzungskriterien wurden eingehalten.



### 9.1.4 Überprüfung Produkteinsatz

#### Definierter Produkteinsatz

Anwendungsbereiche:

*Das Produkt wird im privaten Bereich genutzt werden. Somit ist zum Zeitpunkt der Projektdurchführung keine Markteinführung geplant.*

Zielgruppe:

*Kinder ab 3 Jahren*

#### Überprüfung

Der Produkteinsatz wurde wie geplant umgesetzt.

### 9.1.5 Überprüfung Produktfunktionen

#### Definierter Produkteinsatz

*Das Klangmemory soll über mindestens drei Kategorien verfügen.*

Mögliche Kategorien:

- Tiergeräusche
- Fahrzeugklänge
- Stimmen von bekannten Persönlichkeiten

*Die Lautstärke der Geräusche soll über einen Drehregler eingestellt werden können.*

#### Überprüfung

Die Produktfunktionen wurden wie geplant umgesetzt.



## 9.1.6 Überprüfung Qualitätsanforderungen

### Definierte Qualitätsanforderungen

*Kartenleser:*

*Bei Auflegen der Memorykarte soll innerhalb einer Sekunde ein Geräusch abgespielt werden.*

*Lautsprecher:*

*Die Qualität der abgespielten Klänge soll für den Spieler angenehm und verständlich sein.*

### Überprüfung

Die Qualitätsanforderungen wurden wie geplant umgesetzt.

## 9.1.7 Überprüfung technische Produktumgebung

### Definierte technische Produktumgebung

*Hardware:*

- *Raspberry Pi 4 Model B*
- *RFID-Leser: RFID-RC522*
- *NFC-Tags: Protokoll 13,56 MHz ISO14443A Universal Label RFID Tags*
- *Lautsprecher: manhattan 2600 Series Speaker System*

*Entwicklungsumgebung:*

- *Visual Studio Code*

*Programmiersprache:*

- *Java Version 8*

### Überprüfung

Als Entwicklungsumgebung wurde die IDE Netbeans verwendet. Die Beweggründe sind im Abschnitt 9.2.1.1 beschrieben. Ansonsten wurde die technische Produktumgebung wie geplant umgesetzt.



## 9.2 Projektverlauf

### 9.2.1 Zeitmanagement

Folgend ein Auszug aus unserem nachgeführten Zeitplan:

TASK	Verant-wortlich	Fort-schritt	Start	Ende	Soll (h)	Ist (h)
<b>Total</b>		<b>92%</b>			<b>202</b>	<b>202</b>
<b><u>Projektinitialisierung</u></b>			17.11.2021	12.12.2021	7	6
Projektantrag	AB	100%	17.11.2021	17.11.2021	2	1.5
Zeitplan erstellen	JS	100%	22.11.2021	12.12.2021	3	2.5
Dokumentationslayout	AB	100%	29.11.2021	12.12.2021	2	2
<b><u>Analyse- und Entwurfsphase</u></b>			13.12.2021	31.12.2021	<b>23</b>	<b>23</b>
Pflichtenheft erstellen	JS / AB	100%	13.12.2021	15.12.2021	4	3
Ist Zustand definieren	JS	100%	16.12.2021	19.12.2021	4	3
Konzeptionierung	JS / AB	100%	20.12.2021	25.12.2021	9	9
Auswahl und Evaluierung	JS / AB	100%	26.12.2021	31.12.2021	6	8
<b><u>Implementierungs- Abnahme- und Einführungsphase</u></b>			01.01.2022	28.02.2022	<b>136</b>	<b>141</b>
Inbetriebnahme Raspi/RFID	JS / AB	100%	01.01.2022	05.01.2022	6	55
Programmierung Memory/Logik	JS / AB	90%	06.01.2022	22.01.2022	65	67
Erstellung Memorykatalog	AB	100%	23.01.2022	30.01.2022	12	12
Programmierung GUI	JS / AB	0%	31.01.2022	25.02.2022	45	0
Testing / Bugfix	JS / AB	80%	20.02.2022	28.02.2022	8	7
<b><u>Abschluss- und Dokumentationsphase</u></b>			17.11.2021	16.03.2021	<b>36</b>	<b>32</b>
Softwareentwicklung	JS / AB	100%	01.01.2022	08.03.2022	12	10
Schlussteil	AB	100%	09.03.2022	12.03.2022	10	8
Fertigstellung der Doku	JS	100%	13.03.2022	22.03.2022	6	6
Vorbereitung Präsentation	JS / AB	100%	23.03.2022	29.03.2022	8	8

Abbildung 16: Zeitplan



## Projektaufwand

Wie im obenstehenden Zeitplan ersichtlich ist, haben wir unsere mit 202 Stunden budgetierte Zeit genau erreicht. Jedoch lag dies daran, dass das Projekt im budgetierten Rahmen nicht komplett finalisiert werden konnte.

## Projektinitialisierungsphase

In der Projektinitialisierungsphase benötigten wir 1h weniger als vorgesehen. Dies lag daran, dass wir aus unserem letzten Projekt bereits Vorlagen für die zu erstellenden Dokumente hatten.

## Analyse- und Entwurfsphase

In der Analyse- und Entwurfsphase wurden genau die budgetierten 23h Aufwand investiert.

## Implementierungs-, Abnahme- und Einführungsphase

Für die Softwareentwicklung wurden vorgängig 136h budgetiert. Der effektive Aufwand belief sich auf 141h. Die Implementierungsphase war um 49h zeitaufwändiger als von den Projektbeteiligten vorgesehen. Aus diesem Grund wurde auf das Kann-Ziel der Entwicklung eines GUI's verzichtet.

## Abschluss- und Dokumentationsphase

Für die Dokumentationsphase waren 36h Aufwand vorgesehen. Der Effektivaufwand hat 32h betragen. Auch hier konnten die Projektbeteiligten aus den Vorlagen aus vorgängigen Projekten profitieren und daher 4h Aufwand einsparen.

## Start- und Enddaten

Alle Termine konnten gemäss den Vorgaben aus den „Richtlinien für die Projektarbeit Techniker HF Informatik 2020/2021“ und gemäss Abweichungen in Absprache mit dem Projektbetreuenden eingehalten werden. Die Abweichungen sind im Punkt *9.2.21 Meilensteine* der Projektdokumentation ersichtlich.



### 9.2.2 Kostenmanagement

#### Kostenzusammenstellung Honorar

<b>Kostenpunkt</b>	<b>Budgetiert</b>	<b>Kosten</b>	<b>Eingehalten</b>
Honorar PL	CHF 36'360.00	CHF 36'360.00	<b>Ja</b>

Tabelle 6: Kostenzusammenstellung Honorar

#### Kostenzusammenstellung Material

<b>Kostenpunkt</b>	<b>Anzahl</b>	<b>Stückpreis</b>	<b>Preis</b>
RFID-Reader	2 Stück	CHF 10.60	<b>CHF 21.20</b>
Lautsprecher	2 Stück	CHF 14.30	<b>CHF 28.60</b>
RFID-Tags Ali	1 Stück à 100 Tags	CHF 10.50	<b>CHF 10.50</b>
RFID-Tags Amazon	1 Stück à 100 Tags	CHF 28.85	<b>CHF 28.85</b>
Raspberry Pi 4	2 Stück	bereits vorhanden	<b>CHF 00.00</b>
Breadboard	2 Stück	bereits vorhanden	<b>CHF 00.00</b>
<b>TOTAL</b>			<b>CHF 89.15</b>

Tabelle 7: Kostenzusammenstellung Material

### 9.2.3 Planungsqualität

Die Planung wurde zu Beginn des Projektes detailliert und mit genügend Freiraum erarbeitet, um für allfällige Abweichungen einen Puffer zu haben. Daher konnten wir uns sehr auf die geplanten Termine verlassen und diese auch ohne grössere Abweichungen einhalten. Da viele Dokumente von der bevorstehenden Projektarbeit übernommen werden konnten, hat das Erstellen der Dokumentation weniger Zeit beansprucht als noch im Vorprojekt.



## 9.2.4 Abweichungen

### 9.2.4.1 Meilensteine

Folgende Tabelle ist ein Auszug aus dem finalen Zeitplan. Der komplette Zeitplan ist im Anhang ersichtlich.

<u>Meilensteine / Termine</u>					
<b>1. Projektsitzung</b>		15.12.2021	15.12.2021	14:30	18:00
2. Projektsitzung		09.03.2022	09.03.2022	14:30	15:30
Abgabe Projektarbeit		23.03.2022	23.03.2022	17:00	
Präsentation		30.03.2022	30.03.2022	14:30	17:00
Ferienabwesenheit	JS	24.12.2021	26.12.2021		
Ferienabwesenheit (Zügeln)	AB	25.12.2021	31.12.2021		

Abbildung 17: Meilensteine Projektabschluss

#### 1. Projektsitzung

Aufgrund der aktuellen Pandemielage während der Projektierung wurde auf die *1. Projektsitzung* verzichtet.

#### 2. Projektsitzung

Die *2. Projektsitzung* wurde auf den Mittwoch 09.03.2022 verschoben, konnte dafür aber an der TEKO durchgeführt werden.

#### Abgabe der Projektarbeit

Die *Abgabe der Projektarbeit* wurde in Absprache mit dem Betreuer Bruno Hammer auf den Mittwoch 23.03.2022 verschoben.

#### Präsentation

Dadurch wird die *Präsentation* am 30.03.2022 durchgeführt werden.



### 9.2.4.2 Remote-Verbindung Visual Studio Code

#### Ursprüngliche Planung

Damit mit der IDE Visual Studio Code direkt auf dem Raspberry Pi gearbeitet werden kann, wird eine Remote-Verbindung eingerichtet. Dies erleichtert das Arbeiten auf dem Raspberry Pi erheblich. Somit soll via Desktop-PC oder Laptop direkt auf dem Raspberry Pi gearbeitet werden.

#### Problematik

- Probleme mit dem Java-Enviroment vom Raspberry Pi beim Kompilieren des Projektes. Dieses Problem konnte nicht gelöst werden.
- Probleme mit dem Einbinden der Python-Library und Python-Klassen auf dem Raspberry Pi. Auch dieses Problem konnte im Rahmen der Projektarbeit nicht gelöst werden.
- Zeitverluste durch Auftreten der vorgängig beschriebenen Problematiken.

#### Vorgehen zu Lösung der Problematiken:

Um eine effiziente Lösung zu eruieren, entschieden sich die Projektbeteiligten ein Projektmeeting mit dem Projekt-Betreuuenden Bruno Hammer zu organisieren.

#### Lösungsvorschlag vom Projekt-Betreuuenden

- Verwendung der IDE Netbeans
- Verwendung der Java-Remote-Platform von Netbeans

#### Java-Remote-Platform

Bei der Java Remote-Platform wird das Projekt auf dem Desktop-PC erstellt, bearbeitet, getestet und kompiliert. Beim Kompilieren deployed die Remote-Platform eigenständig ein JAR-File auf dem Raspberry Pi. Dadurch werden alle Sources und Libraries durch Netbeans im JAR-File eingebunden und sind auf dem Raspberry Pi verfügbar. Diese Lösung wurde von den Projektbeteiligten als ideal empfunden und dementsprechend umgesetzt.



## Visual Studio Code Remote-Verbindung

Die Remote-Verbindung vom Visual Studio Code konnte jedoch trotzdem in der frühen Projektentwicklungsphase verwendet werden zum Testen der Python-Klassen, des RFID-Readers und der NFC-Tags.

Nachfolgend ist eine Anleitung zu finden für die ursprünglich verwendete Remote-Verbindung vom Visual Studio Code.

Voraussetzung:

Das Raspberry Pi muss via Ethernet oder WiFi mit dem Internet verbunden sein.

Zudem muss auf dem Raspberry Pi SSH aktiviert sein.

Installation der Remote SSH extension:

Über den *Extensions-Tab* in VS Code, kann nach *Remote development* gesucht werden. Wähle *Remote Development extension* und installiere diese.

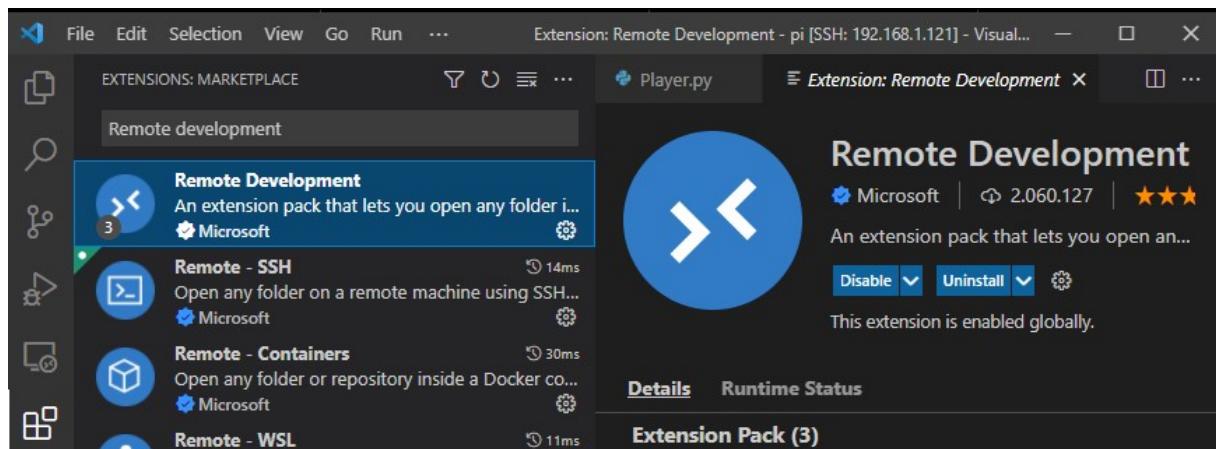


Abbildung 18: Remote-Extension

Aufbau der Verbindung zum Raspberry Pi:

1. Aufruf der command palette von VS Code mit *Ctrl+Shift+P*
2. Suche und wähle *Remote SSH: Connect current window to host*

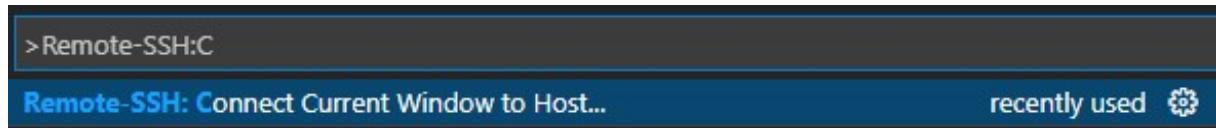


Abbildung 19: ConnectToHost



3. Eingabe der SSH Verbindungsdetails: *user@host*

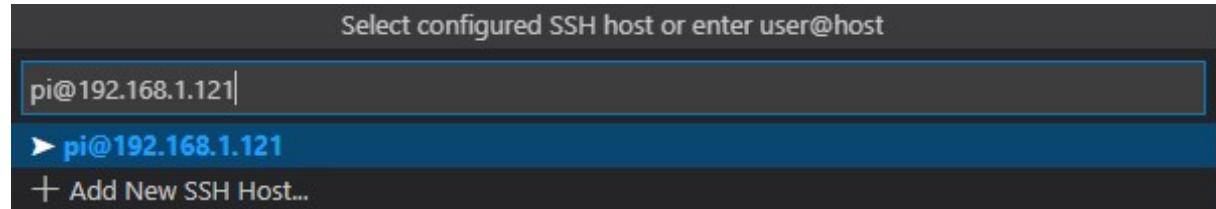


Abbildung 20: AddHost

4. Beim ersten Verbindungsversuch wird der Fingerprint validiert. Bestätige mit *Continue*.

5. Eingabe des Passwortes vom Raspberry Pi



## 9.3 Ausblick in die Zukunft

### 9.3.1 Restaktivitäten

Innerhalb dieser Projektarbeit sind ausser der Präsentation am Mittwoch 30. März 2022 keine Restaktivitäten eingeplant.

### 9.3.2 Zukünftige Ergänzungen und Erweiterungen

Zum jetzigen Zeitpunkt sind keine Ergänzungen und keine Erweiterungen am Raspberry oder an der Programmierung geplant. Denkbare und sinnvolle Ergänzungen wären z.B.:

- Raspberry mit Batterie betreiben um mehr Flexibilität zu erhalten
- Raspberry mit einer LCD Anzeige erweitern um Text nicht nur zu hören sondern auch anzuzeigen
- gewisse Komponenten ersetzen um die Box kleiner gestalten zu können

### 9.3.3 Folgeprojekte

In einem weiteren Projekt wird nun eine Box für den RFID Leser, den Raspberry und die Boxen erstellt, damit das Klangmemory auch sinnvoll sowie zweckmässig eingesetzt werden kann.

Softwaretechnisch sind momentan keine Folgeprojekte geplant, Möglichkeiten bestehen aber auch hier wie z.B.:

- eine Benutzeroberfläche implementieren, damit neue Audiodateien einfach hinzugefügt werden können
- Klangmemory umbauen damit auch Hörspiele abgespielt werden können



## 10 Schlussteil/Reflexion

In einer kurzen Stellungnahme der beiden Autoren wird nun ein Rückblick auf die geleistete Arbeit geworfen. Dabei wird reflektiert was evtl. noch verbessert werden kann, was gelernt wurde und wo Probleme aufgetaucht sind.

### 10.1 Reflexion Adrian Birrer

Mit dem Endergebnis bin ich sehr zufrieden und der Lernfaktor war bei dieser Arbeit sehr hoch. Wie in praktisch jedem Projekt verläuft nicht alles nach Plan und man muss Kompromisse eingehen oder Änderungen vornehmen. Unser Ziel von einem Klangmemory konnten wir aber vollumfänglich umsetzen. Wir konnten auch einen „Text-To-Speech“ implementieren, was zu Beginn nicht zu unseren Anforderungen zählte. Wie erwähnt läuft nicht immer alles nach Plan. Wir mussten beispielsweise die RFID-Tags ein zweites Mal bestellen, weil die ersten nicht funktionierten oder die IDE wechseln, weil eine Remote Verbindung mit Netbeans einfacher ist als mit dem Visual Studio. An dieser Stelle auch ein herzliches Danke schön an unseren Betreuer Bruno Hammer. Ohne das Gespräch welches wir einmal führen mussten, weil wir nicht mehr weiter wussten, wären wir wahrscheinlich falsch abgekommen und hätten die Entwicklung viel komplizierter durchgeführt. Ob wir dadurch auch ein funktionierendes Klangmemory mit der Programmiersprache Java hingekriegt hätten, bezweifle ich leicht.

Die Kommunikation mit Janik verlief einwandfrei. Wir haben uns viel über Teams oder WhatsApp ausgetauscht um den aktuellen Stand der Arbeit oder auch um das weitere Vorgehen zu besprechen. Des öfteren hatten wir jedoch Probleme damit, dass wenn ich etwas umsetzte, es bei ihm nicht lief oder umgekehrt. Diese Probleme dann zu lösen verursachte doch einigen Mehraufwand und müsste nicht sein. Oftmals lag das Problem darin, dass noch etwas zusätzliches implementiert werden musste, wir es aber nicht laufend notierten beim eigenen ausprobieren. So gingen dann wichtige Erkenntnisse vergessen und konnten dem Partner nicht korrekt mitgeteilt werden. Nichtsdestotrotz war es eine sehr gute Zusammenarbeit und ich würde ein Projekt mit Janik jederzeit wieder durchführen.



## 10.2 Reflexion Janik Schilter

Meine bisherigen Kenntnisse der Programmiersprache Java waren bisher eher theorieelastig. Daher war es mein Ziel, mit dieser Projektarbeit meine Java-Programmier-Kenntnisse zu verbessern. Die ersten Schritte in der Implementierungsphase dieses Projektes waren teilweise sehr frustrierend, da wir grössere Schwierigkeiten mit dem Aufsetzen des Remote Development hatten. Aber Mithilfe von diversen Websites, lehrreichen Gesprächen im Projektteam und nicht zuletzt, mit Hilfe vom Projektbetreuenden Bruno Hammer, waren wir in der Lage die anfänglichen Probleme zu überwinden. Die Zusammenarbeit im Projektteam empfand ich als sehr angenehm und unkompliziert. Durch Projektbesprechungen im Microsoft Teams und dem regelmässigen Austausch über Whatsapp, konnten wir trotz Social-Distanzierung eine einwandfreie Kommunikation und Zusammenarbeit gewährleisten.



## Anhang

### 10.1 Projektunterlagen

Alle Projektunterlagen sind im ELAD unter folgendem Link abgelegt:

<https://elad.ch/olat/auth/RepositoryEntry/94011417/CourseNode/105261196525671>

- Projektantrag
- Zeitplan
- Flussdiagramm Funktionsablauf
- UML



## Abbildungsverzeichnis

1	Klangmemory . . . . .	2
2	Meilensteine . . . . .	6
3	Janik Schilter . . . . .	7
4	Adrian Birrer . . . . .	7
5	Organigramm . . . . .	7
6	Use-Case-Diagramm . . . . .	10
7	Risikoanalyse . . . . .	14
8	Kategorien Risikoanalyse . . . . .	15
9	ON/OFF-Button / Verdrahtung an Raspberry . . . . .	19
10	GPIO <sub>PinOut</sub> – Header . . . . .	19
11	RFID Card Reader Anschluss an Raspberry . . . . .	21
12	Raspberry Pi-config . . . . .	23
13	UML . . . . .	25
14	Struktur Audiodateien . . . . .	27
15	Fehlermeldung Black-Box-Test . . . . .	28
16	Zeitplan . . . . .	35
17	Meilensteine Projektabschluss . . . . .	38
18	Remote-Extension . . . . .	40
19	ConnectToHost . . . . .	40
20	AddHost . . . . .	41

## Tabellenverzeichnis

1	Zeitplanung . . . . .	5
2	Funktionale Ziele . . . . .	11
3	Nicht Funktionale Anforderungen . . . . .	12
4	Funktionale Ziele . . . . .	31
5	Nicht Funktionale Anforderungen . . . . .	32
6	Kostenzusammenstellung Honorar . . . . .	37
7	Kostenzusammenstellung Material . . . . .	37