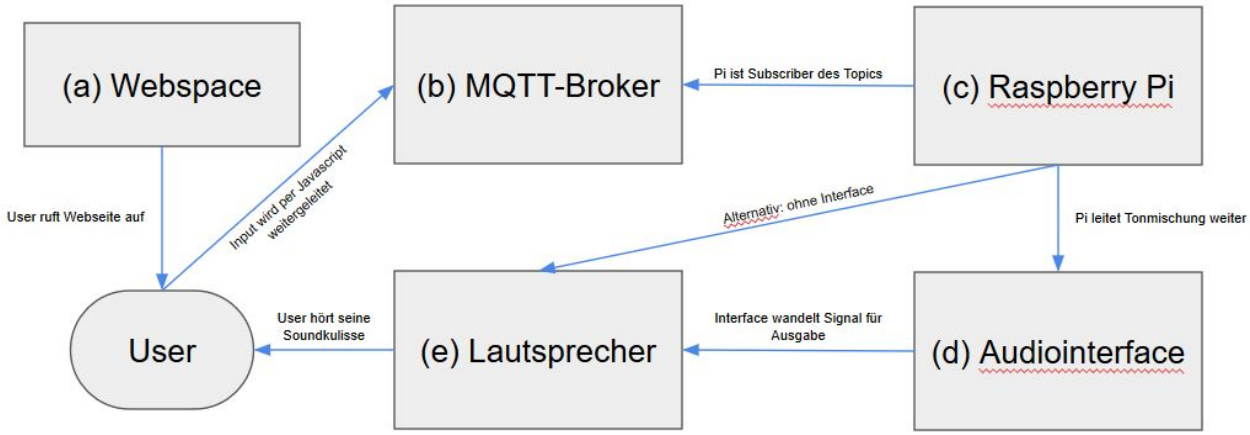
Technische Dokumentation

HAW Hamburg | Medientechnik | MOSY/ITS | Prof. Plaß | Prof. Edeler

A M B I E N T B O X

DISCOVER THE SOUNDTRACK OF YOUR LIFE

1. Hardware-Komponenten



a. **Webspace**
   Auf dem Webspace wird das Frontend der Anwendung gehostet. Es enthält die HTML-Files für das Bedienungsmenü, jedoch auch die Javascriptdateien, die die Inputs des Users an den MQTT-Broker senden.
b. **MQTT-Broker**
   Die zentrale Schnittstelle des Projektes. Während das Script des Frontends die Inputs des Users hier auf einem Topic veröffentlicht, ist das Backend (auf dem Pi) ein Subscriber dieses Topics und führt entsprechend geforderte Befehle aus
c. **Raspberry Pi**
   Der Hauptcomputer. Hier findet die Echtzeitabmischung der Tonspuren, wie auch das Umsetzen neuer Befehle (Lautstärkenänderung, abspielen stoppen, starten, ...) statt.
d. **Audiointerface (optional)**
   Da der Analogwandler für den Klinkenausgang des Pis nicht besonders hochwertig ist, bietet es sich an, ein USB-Audiointerface zu verwenden, das für den Audiooutput zuständig ist. Diese Komponente ist jedoch optional.
e. **Lautsprecher**
   Schließlich wird der Ton über die Lautsprecher ausgegeben, die entweder am Pi oder am Audiointerface angeschlossen sind.

1. Frontend (HTML/CSS/Javascript)

Das Interface wurde in HTML, CSS und JavaScript erstellt. Mit dem Aufruf der Website wird eine Verbindung zwischen User und MQTT-Broker hergestellt. Durch die Interaktion auf der Seite werden Strings an den Broker gesendet, die im Anschluss vom Backend aufgenommen und interpretiert werden.

Die Sounds sind in fünf Kategorien eingeteilt, welche durch namenstypische Bilder untereinander auf der Website angeordnet sind. Bei Auswahl einer Kategorie öffnen sich die enthaltenen Komponenten. Neben den vorgefertigten „Presets“ und eines „Block-Resets“ erscheint der Kern des Projektes, die Lautstärkeregler der enthaltenen Soundelemente.

Zum schnellen Erhalt harmonisch abgemischter Sounds wurden Presets eingefügt. Diese sind eine Voreinstellung unterschiedlicher Kanäle aus verschiedenen Kategorien. Mit dem Klick oder Tap auf einen der „Preset-Buttons“ werden indirekt drei unterschiedliche Methoden aufgerufen. Die Funktion „gimmePS()“ wählt je nach ID des Buttons, das dazugehörige „Preset“ aus. Zusätzlich wird vorab mit der Funktion „pahoMessage()“ eine Nachricht mit dem Inhalt „reset“ an den MQTT-Broker gesendet um alle aktuellen Sounds zurückzusetzen. Die dritte Methode „hilfe()“ ruft für alle übergebenen Soundnamen und Lautstärkewerte ebenfalls „pahoMessage()“ auf. Für das Feedback an den User werden die angesprochenen Input Regler auf der Seite ebenfalls verschoben.

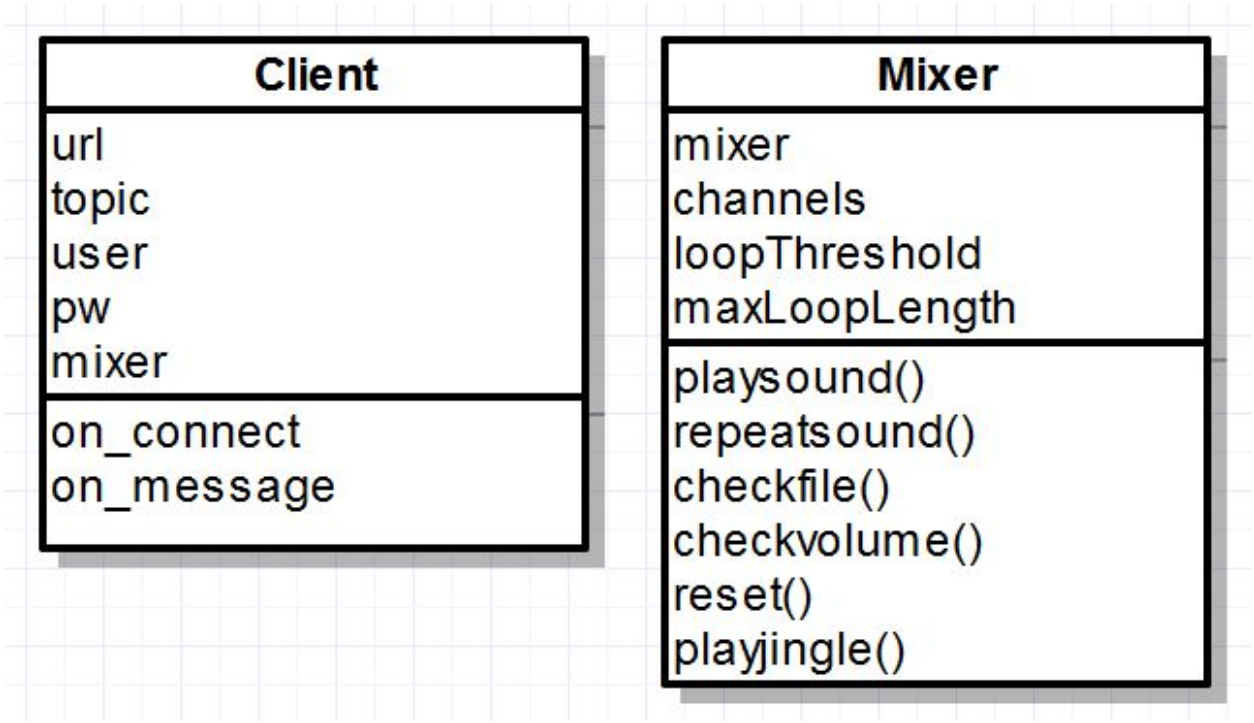
Der „Block-Reset“ ist für ein schnelles Löschen aller Sounds einer Kategorie zuständig. Er führt die Methode „resetBlock()“ aus. Je nach Kategorie wird ein Start- und ein Endwert übergeben, die der Position der Sounds im Soundarray entsprechen. Diese werden durch eine Paho-Message auf null gesetzt und ebenfalls lokal auf der Webseite geändert.

Die Input-Regler, wichtig für die Manuelle Bedienung der App, erstellen eine Paho-Message an den MQTT-Broker. Durch die Methode “pahoMessage()” wird ihre ID, und ihr Value als String an das Pie weitergeleitet. Hierbei ist wichtig, dass ihre ID dem Names des jeweiligen Soundfiles auf dem Pie entspricht. Der Value wird als die Lautstärke des Sounds umgewandelt.

Der letzte Bestandteil des Layouts ist der “Zurücksetzen”-Button. Dieser löscht jegliche Aktionen des Users für einen schnellen Reset.

2. Backend (Python)

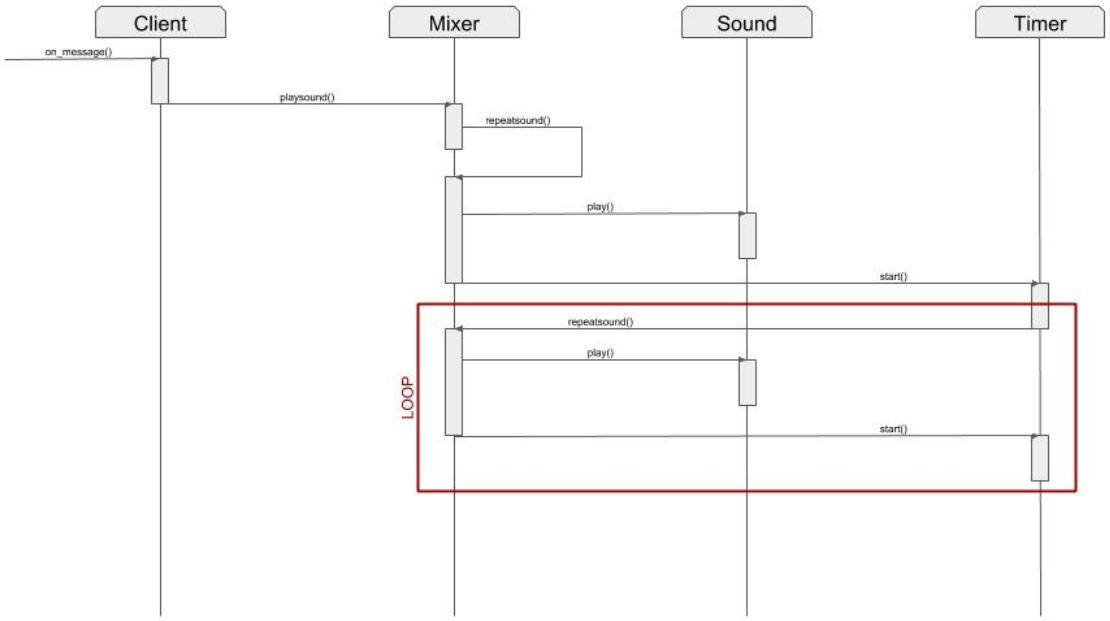
Das Backend ist in Python geschrieben und wird auf dem Raspberry Pi ausgeführt. Es kümmert sich sowohl um den Empfang neuer Befehle über den MQTT-Broker, als auch deren Ausführung.
Zur Verbindung zum MQTT-Broker und dem Empfang von Nachrichten ist die Klasse “Client” zuständig. Der “Mixer” wiederum ist für das Abspielen und Abmischen der Sounds zuständig. Das Pi greift direkt auf die Soundfiles zu (vorzugsweise über einen USB3-Stick für schnellere Zugriffszeiten), und spielt diese mittels eines erweiterten Pygame-Mixer-Modules ab.



Da die MQTT-QoS im Frontend auf 1 gesetzt wurde, kann davon ausgegangen werden, dass jede Nachricht übertragen wird. Eine mehrfache Übertragung eines Befehls stellt kein Problem dar, da über diese Befehle nur Sound abgespielt oder gestoppt werden kann; beides Aktionen, die bei mehreren Ausführungen keinerlei Konflikte verursachen können.

Dadurch, dass sowohl Pi als auch Website durch dasselbe MQTT-Topic miteinander kommunizieren, kann eine Ambientbox durch mehrere User verwendet werden, genauso jedoch auch kann ein User mehrere Ambientboxen kontrollieren.

Die Verarbeitung eines Befehls könnte folgendermaßen ablaufen:



In diesem Beispiel soll ein Sound mit einer Länge kleiner als der *maxLoopLength (siehe Code)* abgespielt werden.

Bei eingehender Nachricht wird im Client die Methode “on_message()” aufgerufen. Diese Methode untersucht, ob es sich bei der eingehenden Nachricht um 1-2 Worte (die hier die Argumente darstellen sollen) handelt. Ist dies der Fall, wird mit diesen Argumenten die Methode “playsound()” im Mixer aufgerufen. Ist nur ein Argument angegeben, muss dieses “reset” heißen, und führt zum sofortigen Reset der Ambientbox. Dies ist zum Beispiel dann nützlich, wenn sehr viele Sounds abgespielt werden und die Box schnell zur Ruhe gebracht werden soll.

Sind zwei Argumente angegeben worden, stellt das erste Argument den Namen des Sounds dar, der abgespielt werden soll, das zweite die Lautstärke. Es wird untersucht, ob es sich beim ersten Argument um ein gültiges wav-File handelt, das im Unterordner “audio” platziert ist, und ob es sich bei dem zweiten Argument um eine ganze Zahl zwischen 0 und 100 handelt.

Ist beides der Fall, wird der Sound in Schleife abgespielt. Falls der Sound kürzer als die in der Variable “loopThreshold” angegebene Länge in Sekunden, wird ein Timer in einem Thread gestartet, der, abhängig von der Lautstärke des Sounds, den Sound nach einigen Sekunden erneut abspielt.

Wird die Lautstärke eines Sounds auf “0” gesetzt, wird dessen Playback sofort gestoppt und der Sound aus dem directory der momentan abgespielten Sounds gelöscht.

3. Generierung der Sounds

Für die Ambientbox musste eine Datenbank an Sounds erstellt werden, die es erlaubte eine akustische Fläche zu malen, welche stets kombinierbar mit Wettereffekten und Instrumentalspuren ist und mit Detailspuren ein stimmungsvolles Gesamtbild kreiert.

Teilweise sind die Sounds in Eigenregie erstellte Field- und Detailaufnahmen, zum Großteil aber auch adaptierte und nachbearbeitete Sounds aus diversen Datenbanken, sowie eigenen Archiven. Aufnahmeorte umfassen unter anderem den Wildpark schwarze Berge, den Hamburger Flughafen und das Dach der Karstadtfiliale in der Mönckebergstraße.

Atmosphärische Stereo-Sounds sind größtenteils in der jeweiligen Umgebung mit einem ZOOM H6 Sechsspurgerät (fest auf Stativ) in X-Y-Mikrofonie entstanden.
Detailaufnahmen wurden mit demselben Gerät generiert, hierbei war allerdings die Mikrofonierung häufig ein Richtrohr (Superniere) mit entsprechender Schwingungsentkopplung (Korb) und Windschutz (ähnlich Filmton).
Diese Anordnung wurde dann, je nach Aufnahmeobjekt, per Hand (bewegliche Objekte) oder ebenfalls stativiert benutzt.

Mit den in unserer Datenbank enthaltenen Instrumentals sollen zum einen Akzente gesetzt werden können, die sich nahtlos in die bereits geschaffene Atmosphäre einfügen lassen, andererseits sollen aber auch bereits geschaffene Flächen untermalt und Emotionen angeheftet werden können.

Hierfür haben wir kurze Passagen als Partitur komponiert, die als MIDI-Dateien herausgerechnet und instrumentiert wurden.
So hatten wir eine Auswahl an kurzen angedeuteten Melodien und Rhythmen, welche wir nun in Audacity mit Effekten belegten.
Überwiegend wurden diverse Formen des Faltungshall und diverse Delays benutzt.

Durch die technische Begrenzung de Raspberry-Pi war eine Abtastfrequenz von 44.100Hz mit einer Bittiefe von 16 vorgegeben (.wav), die es galt einzuhalten. Die auditive Bearbeitung der Geräusche geschah hauptsächlich mit Audacity.
Besonders war darauf zu achten, die Pegel der einzelnen Sounds anzugleichen und dabei die Sinnhaftigkeit der entsprechend wahrgenommenen „Lautheit“ untereinander zu gewährleisten.

4. Gruppe

2218011 - Eric Goofers
2185142 - Timo Gundlach
2206404 - Malte Gruntzdorff
2138141 - Felix Stackfleth