# 1 Einleitung

Die vorliegende Ausarbeitung beschreibt den Aufbau, die Funktion sowie technische Hintergründe des „Audio reactive LED-Strips". Der vollständige Sourcecode ist im verlinkten GitHub Repository zu finden.

## 1.1 Hardware

Für die Anwendung werden ein Raspberry Pi, ein LED-Streifen sowie ein Mikrofon und eine Sonos Musikanlage benötigt. Auf dem Raspberry Pi wurde als Betriebssystem Raspberry-OS installiert. Der in Python geschriebenen Sourcecode wird ebenfalls auf dem Raspberry ausgeführt. Um die LEDs des LED-Streifens anzusteuern wird ein Kabel an einen GPIO-Pin des Raspberry PI angeschlossen. Die Stromversorgung des Streifens erfolgt über ein separates Netzteil, um bei längerem Betrieb den Raspberry Pi nicht zu überlasten. Zusätzlich wird via USB eine Soundkarte an dem Pi angeschlossen, welche das eingehenden Audiosignal von dem Mikrofon an den Pi weiterleitet.

## 1.2 Funktion

Die Funktionalität der Anwendung lässt sich in zwei Modi unterteilen. Der erste Modus ist der Musikmodus. Dieser dient dazu, visuelle Effekte anhand von Audioaufnahmen zu berechnen und auf den LED-Streifen darzustellen während auf der Sonos Anlage Musik abgespielt wird.

Der zweite Modus ist der TV-Modus. Dieser ist aktiv so bald über den Fernseher Filme oder Serien geschaut werden und der Ton dieser über die bereits erwähnte Sonos Anlage wiedergegeben wird. In diesem Modi soll der LED-Streifen konstant in einer Farbe leuchten. Wird weder ferngesehen noch Musik gehört wird der LED-Streifen abgeschaltet. Die Umschaltung zwischen den zwei Darstellungsmodi sowie dem ein bzw. ausschalten des LED-Streifens erfolgt automatisch durch Abfrage von Parametern der Sonos Anlage.

## 1.2 Verwendete Technologien

Die Anwendung wurde in der Skriptsprache Python in der Version 3.9.7 erstellt. Eine Ausführung ist jedoch auch mit niedrigeren Versionsnummern bis Version 3.6 möglich. Für die Ausführung werden verschiedene standardmäßig nicht installierte Pakete benötigt. Als relevanteste Abhängigkeiten sind hier pyaudio, Sonos Controller (kurz SOCO) und neopixel zu erwähnen. Pyaudio stellt verschiedene Funktionen zur Aufnahme, Verarbeitung und Speicherung von Audiosignalen zur Verfügung. Innerhalb dieser Anwendung wird es verwendet um die im Raum gespielte Musik aufzunehmen. Die SOCO-Bibliothek ermöglicht Lautsprecher der Marke Sonos zu kontrollieren und deren Zustand (bspw. aktuell gespieltes Lied) auszulesen. Hierbei ist zu erwähnen das SOCO über keine vollständige Implementierung für die Wiedergabe von TV-Inhalten verfügt. Wie mit dieser Einschränkung umgegangen wurde, wird im Kapitel Realisierung erläutert. Die Bibliothek neopixel dient zur Ansteuerung der LEDs. Sie bietet zum einen Funktionen, um den gesamten Streifen in einer Farbe leuchten zu lassen oder auch einzelne LEDs ansteuern.

## 1.3 Struktur der Anwendung

Um für dritte und spätere eventuelle Weiterentwicklungen die Übersichtlichkeit zu erleichtern wurde der Quellcode auf verschiedene Python Dateien aufgeteilt. Folgende Dateien sind Bestandteil der Anwendung:

• config.py - Konfigurationsdatei

• led.py - LED-Funktionen

• microInput.py - Aufnahme von Audiosignalen

• TVStateEnum.py - Enumeration für Status der Sonos Anlage

• wave2RGB.py - Funktionen zur Farbberechnung

• main.py - Hauptdatei zur Ausführung

Die Ausführung der Anwendung erfolgt durch die main.py Datei. Diese ruft einzelne Funktionen und Klassen der anderen Dateien auf und beinhaltet unteranderem auch Funktionalität der Farbberechnung zu Audiosignalen. In der main.py Datei und auch in allen anderen Dateien werden wesentliche Konfigurationsparameter aus der zentralen Konfigurationsdatei bezogen. Dies erleichtert Entwicklern und Anwendern Änderungen vorzunehmen und zu konsistent zu testen. Im Folgenden soll der Ablauf der Ausführung und die technischen Hintergründe erläutert werden.

# 2 Programmablauf

## 2.1 Abfrage der Sonos Anlage

Die Ausführung beginnt in der main Methode innerhalb der gleichnamigen Datei. Hier werden

einige notwendige globale Variablen initialisiert und alle wesentlichen Methoden aufgerufen.

Das zentrale Element der Methode ist eine endlose Schleife. Innerhalb dieser Schleife wird in

jedem Durchlauf der aktuelle Zustand der Sonos Anlage erfasst und ausgewertet um zu ermitteln welcher Modus im Moment gewünscht ist. Der Zustand wird in Form einer Variable des Typs TVStateEnum gespeichert. Diese beinhaltet Zustände für die Wiedergabe von Musik und TV-Audio sowie einen Pause Zustand. Diese Zustände werden im weiteren Programmablauf verwendet, um festzustellen welcher Modus aktuell ausgeführt wird und ob der Modus gewechselt werden sollte. Wie genau der Wechsel in einen anderen Modus erfolgt, wird in dem folgenden Kapitel beschrieben.

Die Abfrage der Sonos Anlage erfolgt über einen Aufruf der updateCurrentState Funktion. Diese fragt mithilfe der SOCO Bibliothek den Wiedergabestatus und das aktuell gespielte Lied ab. Durch diese Abfrage kann ermittle welcher Zustand der TVStateEnum zurückgegeben werden muss. Ein Sonderfall ist die Wiedergabe von Audioinhalten eines Fernsehers. Wie bereits in der Einleitung erwähnt, implementiert SOCO diesen Fall nicht vollständig. Wird ein solches Audiosignal über die Anlage wiedergeben wird keine Information über den aktuell wiedergegeben Titel zurückgegeben, jedoch wird erkannt das die Anlage aktuell Ton abspielt. Aus diesem Grund ist die Abfrage in der zweiten Fallunterscheidung wie in der Abbildung 1 zu sehen entsprechend angepasst. Wurde nach der Status Abfrage festgestellt das der aktuelle Modus nicht gewechselt werden muss, wird der main Thread einige Sekunden pausiert. Durch dieses Vorgehen können Abfragen ohne Mehrwert vermieden werden und der Ressourcenverbrauch reduziert werden.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung 1 - Codeausschnitt Abfrage der Sonos Anlage

## 2.2 Musikmodus

### 2.2.1 Musik Modus starten und beenden

Der Musik Modus dient dazu passende Farben anhand des Audiosignals zu berechnen und auf dem LED-Streifen darzustellen. Jedoch ist es erforderlich, parallel zu der Verarbeitung der Audiosignale, weiterhin den Status der Sonos Anlage zu beobachten, um einen späteren erneuten Wechsel des Modus zu ermöglichen. Die Parallelität dieser Abfrage und der Audioverarbeitung wurde durch Multi Threading ermöglicht. Wird in den Musik Modus gewechselt wird ein neuer Thread (im folgenden Musik Thread genannt) gestartet, welcher die Ausführung der nötigen Funktionen übernimmt.

Der Main Thread prüft im Anschluss lediglich den Zustand der Sonos Anlage in regelmäßigen

Intervallen. Wird im Main Thread ein Wechsel in einen anderen erkannt, sendet er über eine Queue an den Musik Modus Thread eine Task, welche die Terminierung über einen entsprechenden Handler einleitet und die Task aus der Queue wieder entnimmt. Dieser Ablauf ist in der Abbildung 2 zusehen.



Abbildung - Starten und beenden Musik Thread

Zu der Realisierung dieser Funktion sollte erwähnt werden, dass Python kein

Echtes Multithreading unterstützt. Daher laufen die Threads nicht echt parallel auf unterschiedlichen Kernen. Für den gewünschten Zweck ist die vorhandene Implementierung jedoch hinreichend, da der Main Thread die meiste Zeit wartend verbringt und daher ohnehin selten Rechenzeit in Anspruch nimmt. Bei einer späteren Weiterentwicklung könnten statt eines zweiten Threads ein separater Prozess erzeugt werden, welcher die Ausführung des Musik Modus übernimmt. Dies würde eine parallele Ausführung ermöglichen und möglicherweise die Performance der Anwendung steigern.

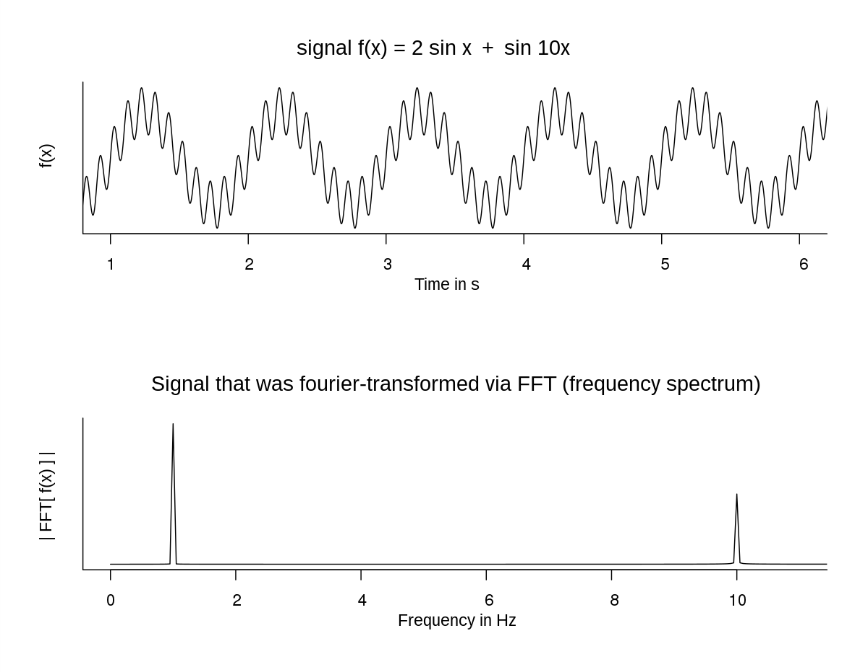
### 2.2.2 Audioaufnahme

Die Aufnahme und Verarbeitung des Audiosignals beginnen mit dem Starten des Musik Threads, sobald die Abfrage der Sonos Anlage das entsprechende Ergebnis produziert hat. Für die Aufnahme des Audiosignals wird in der Methode recordAudio zunächst ein Objekt der Klasse Rekorder erstellt, welche mithilfe der pyaudio Bibliothek Signale des Standardeingabegerätes des Raspberry erfasst. Im Konstruktor dieser Klasse werden verschiedene Parameter, wie die Abtastrate oder die Anzahl der Aufgenommenen Audiokanäle angegeben, um ein geeignetes Stream Objekt mit Pyaudio zu erzeugen. Wurde das Recorder Objekt erfolgreich erzeugt kann die zentrale permanente Schleife der Funktion

beginnen. Diese nimmt zunächst mithilfe des Recorder Objektes ein Audiosignal auf, welches im Anschluss für die Berechnung der darzustellenden Farben genutzt wird. Das Recorder Objekt gibt bei der Ausführung zunächst eine Liste an float Werten zurück, welche die Ausschläge des Mikrofons darstellen. Auf die Berechnung der Farbe soll im Folgenden Kapitel näher eingegangen werden.

### 2.2.3 Farbberechnung

Für die Berechnung der Farben aus den Ausschlägen des Audiosignals wurden Grundlagen und Inhalte des Psynesthesia Repositories verwendet. Verwendete Codeausschnitte aus diesem Repository sind in der Funktion zur Umwandlung und Berechnung der Farbwerte zu finden (freqToRGB und wavelen2RGB) und sind im Source Code entsprechend gekennzeichnet.

Für die Berechnung der dazustellenden Farbe, werden in der Funktion freqToRGB die Frequenzanteile der Musik näher betrachtet. Diese erhält als Übergabeparameter die Ausschläge des Mikrofons.

Musik, wie sie aus der Musikanlage ertönt, besteht aus einer Vielzahl an Frequenzen, welche zusammen eine entsprechende Schallwelle bilden. Diese Schallwelle kann nun durch die Fourier-Transformation in ihre Frequenzbestandteile zerlegt werden. Wie in dem Beispiel aus Abbildung 3 zu sehen, werden aus der oben dargestellten Schwingung Frequenzen und deren Anteile im Bereich von 1 Hz und 10 Hz berechnet.

Abbildung - 3 Darstellung Fourier Transformation

Die Fourier-Transformation wird in der vorliegenden Anwendung verwendet, um die Frequenz mit dem größten Anteil zu bestimmen. Hierfür wird zunächst die Transformation mithilfe von numpy durchgeführt, bevor im Anschluss das Maximum bestimmt wird. Die so bestimmte Frequenz wird im Anschluss für die Berechnung der Farbe verwendet.

Ton und Licht bestehen aus einer ähnlichen Form von Welle. Ein Ton wird ehr als Art Druckwelle verstanden, welche die Umgebungsluft in Schwingung versetzt. Jedem Ton kann dabei eine bestimmte Frequenz zugeordnet werden. Der Ton A hat beispielsweise eine Frequenz von 440Hz. Im Gegensatz dazu bestehen Farben auch aus Wellen, allerdings werden diese ehr als elektromagnetisch verstanden. Durch diesen Umstand lassen sich auch Farben unterschiedliche Frequenzen zuordnen. Im Unterschied zu akustischen Signalen sind Frequenzen von Farben allerdings um ein Vielfaches höher. So beginnt der für Menschen sichtbare Bereich erst bei einer Frequenz von mit der Farbe Rot wie in Abbildung 2 zu sehen. Alle Frequenzen darunter zählen zum Bereich des Infraroten Lichts und sind für Menschen als solches nicht erkennbar.

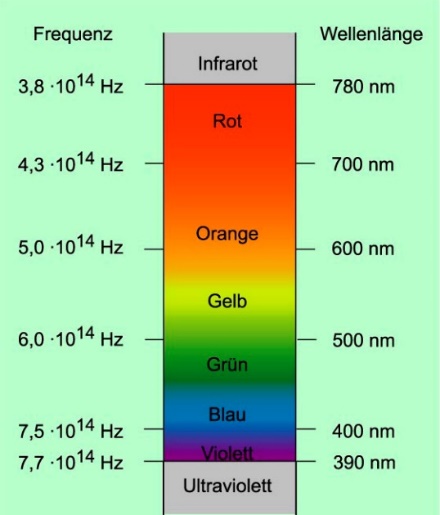


Abbildung Frequenzen von Farben

An diesem Punkt wird bereits deutlich das ein physikalischer Zusammenhang von Tonfrequenzen und Lichtfrequenzen möglich ist, allerdings jeder hörbare Ton lediglich Farben im Infraroten Farbbereich ergeben würde. Als mathematische Annäherung wurden in der Anwendung zur Darstellung von sichtbaren Farben die ermittelten Ton Frequenzen auf den Frequenzbereich von THz skaliert. So würde das Beispiel des Tons A mit einer Frequenz von 440Hz auf eine Frequenz von skaliert werden. Damit wäre dieser, wie in der Abbildung 2 zu sehen, dem Roten Farbbereich zu zuordnen. Zusätzlich zu der beschriebenen Skalierung werden noch leichte Anpassungen der Frequenz vorgenommen, um die berechneten Farben zu modifizieren.

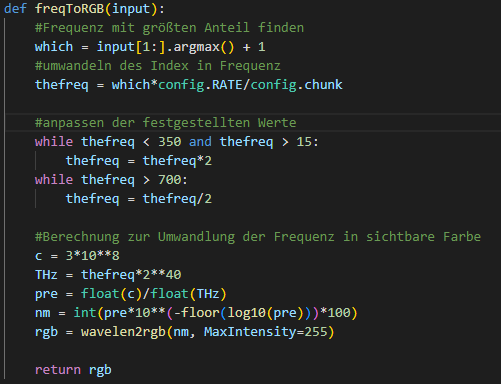


Abbildung - 5 Funktion Berechnung der Farben

## 2.2.4 Anzeigen der Farben auf den LED-Streife

Nach dem die Farben die Farben durch die vorangegangenen Funktionen berechnet wurden, werden mit Hilfe eines Objektes der Klasse ledStrip die Farben auf dem Streifen angezeigt. Für die Anzeige der LEDs wird die Bibliothek neopixel verwendet. Diese erstellt bei der Objekt Erstellung eine Liste, welche über die gleiche Anzahl an Elementen verfügt wie LEDs angeschlossen wurden. Der Wert jedes Elements stellt dabei die darzustellende Farbe als RGB Wert dar. Die berechneten Farben im Musikmodus, werden jeweils auf drei LEDs angezeigt. Wurde eine neue Farbe berechnet, werden die bisher angezeigten Farben um eine Position nach hinten geschoben, wodurch eine Farbe von dem Streifen verschwindet und am Anfang Platz entsteht, um eine neue Farbe anzuzeigen.