

**Studiengang**

**Angewandte Informatik (FPO2020)**

**Audio Reactive LED-Strip in Python**

Schriftliche Ausarbeitung im Modul

Skriptsprachen

Tim Halle

Matrikelnummer: 30159345

Dozent: Prof. Dr. Heiner Giefers Abgabedatum: 19.02.2023

Inhalt

[Abbildungsverzeichnis 3](#_Toc127699113)

[1 Einleitung 4](#_Toc127699114)

[1.1 Hardware 4](#_Toc127699115)

[1.2 Funktion 5](#_Toc127699116)

[1.3 Verwendete Technologien 5](#_Toc127699117)

[1.4 Struktur der Anwendung 5](#_Toc127699118)

[2 Programmablauf 6](#_Toc127699119)

[2.1 Abfrage der Sonos Anlage 6](#_Toc127699120)

[2.2 Musikmodus 7](#_Toc127699121)

[2.2.1 Musikmodus starten und beenden 7](#_Toc127699122)

[2.2.2 Audioaufnahme 8](#_Toc127699123)

[2.2.3 Farbberechnung anhand von Wellenlänge 9](#_Toc127699124)

[2.2.4 Farbberechnung und Anzeige anhand von Dezibel Werten 12](#_Toc127699125)

[2.3 Probleme der Farbanzeige auf dem LED-Streifen 12](#_Toc127699126)

[3 Fernsehmodus 13](#_Toc127699127)

[4 Fazit und Potential 13](#_Toc127699128)

[Quellenangaben 14](#_Toc127699129)

# Abbildungsverzeichnis

[Abbildung 1- Funktion der Anwendung 4](#_Toc127699104)

[Abbildung 2 - Codeausschnitt Abfrage der Sonos Anlage 7](#_Toc127699105)

[Abbildung 3 - Starten und beenden Musik Thread 8](#_Toc127699106)

[Abbildung 4 - Darstellung Fourier Transformation (https://de.wikipedia.org/wiki/Schnelle\_Fourier-Transformation) 9](file:///C:\Users\timga\Documents\Uni\5.%20Semester\Skriptsprachen\Skriptsprachen\Ausarbeitung\AudioReactiveLEDStrip.docx#_Toc127699107)

[Abbildung 5 - Frequenzen von Farben (https://www.lernhelfer.de/schuelerlexikon 10](#_Toc127699108)

[Abbildung 6 - Funktion Berechnung Farben 11](#_Toc127699109)

[Abbildung 7 - Ergebnis Zeitmessung 13](file:///C:\Users\timga\Documents\Uni\5.%20Semester\Skriptsprachen\Skriptsprachen\Ausarbeitung\AudioReactiveLEDStrip.docx#_Toc127699110)

[Abbildung 8 - Codeauschnitt Messung Ausführdauer 13](file:///C:\Users\timga\Documents\Uni\5.%20Semester\Skriptsprachen\Skriptsprachen\Ausarbeitung\AudioReactiveLEDStrip.docx#_Toc127699111)

# Einleitung

Die vorliegende Ausarbeitung beschreibt den Aufbau, die Funktion sowie technische Hintergründe des „audio reactive LED-Strips“. Der im Rahmen des Projekts verfasste vollständige Sourcecode ist im verlinkten [GitHub Repository](https://github.com/HalleTim/Skriptsprachen) zu finden. Für eine bessere Lesbarkeit sind Bezeichnungen, welche aus dem Code übernommen wurden, wie das Wort *<Beispiel>* in diesem Satz, kursiv geschrieben und in eckige Klammern gefasst.

## Hardware

Für die Anwendung werden ein Raspberry Pi, ein LED-Streifen sowie ein Mikrofon und eine Musikanlage der Marke Sonos 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.



Abbildung - Funktion der Anwendung

## 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 Sonos Anlage wiedergegeben wird. In diesem Modus 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 Abfragen von Parametern der Sonos Anlage.

## 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 [2.1 Abfrage der Sonos Anlage](#_Abfrage_der_Sonos) 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.

## 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 weiterhin die Funktionalität der Farbberechnung zu Audiosignalen. In der *<main.py>* Datei sowie in allen anderen Dateien werden wesentliche Parameter aus der zentralen Konfigurationsdatei bezogen. Dies erleichtert Entwicklern und Anwendern Änderungen vorzunehmen und konsistent zu testen. Im Folgenden soll der Ablauf der Ausführung und die technischen Hintergründe erläutert werden.

# Programmablauf

## 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 zu diesem Zeitpunkt 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 in einen anderen gewechselt werden sollte. Wie genau der Wechsel in einen anderen Modus erfolgt, wird im 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 ermittelt werden welcher Zustand von der *<TVStateEnum>* zurückgegeben werden muss. Anhand des zurückgegebenen Wertes kann in der Schleife dann die Entscheidung über den eventuell zu aktivierenden Modus getroffen werden.

Ein Sonderfall ist die Wiedergabe von Audioinhalten eines Fernsehers. Wie bereits in der Einleitung erwähnt, implementiert SOCO diesen Fall nicht vollständig. Wird das Audiosignal eines Fernsehers über die Anlage wiedergeben, so kann SOCO keine Information über den aktuell wiedergegeben Titel ermitteln. Jedoch wird erkannt das die Anlage aktuell Audio abspielt. Aus diesem Grund ist die Abfrage in der zweiten Fallunterscheidung wie in der Abbildung 2 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 und der Ressourcenverbrauch reduziert werden.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung - Codeausschnitt Abfrage der Sonos Anlage

## Musikmodus

### Musikmodus starten und beenden

Der Musikmodus 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. Dass der Musikmodus weiterhin aktiv ist, wird anhand des Status des entsprechenden Threads überprüft. Wäre beispielsweise der Musik Thread inaktiv, kann der Musikmodus nicht in Verwendung sein und müsste gestartet werden. Wird im Main Thread ein Wechsel in einen anderen Modus erkannt, sendet er über eine Queue eine Task an den Musik Thread, welche die Terminierung über einen entsprechenden Handler einleitet. Der Handler sorgt im Anschluss auch dafür, dass die Task wieder aus der Queue entnommen wird. Ein Ausschnitt dieses Ablaufs ist in der Abbildung 3 zu sehen.



Abbildung - Starten und beenden Musik Thread

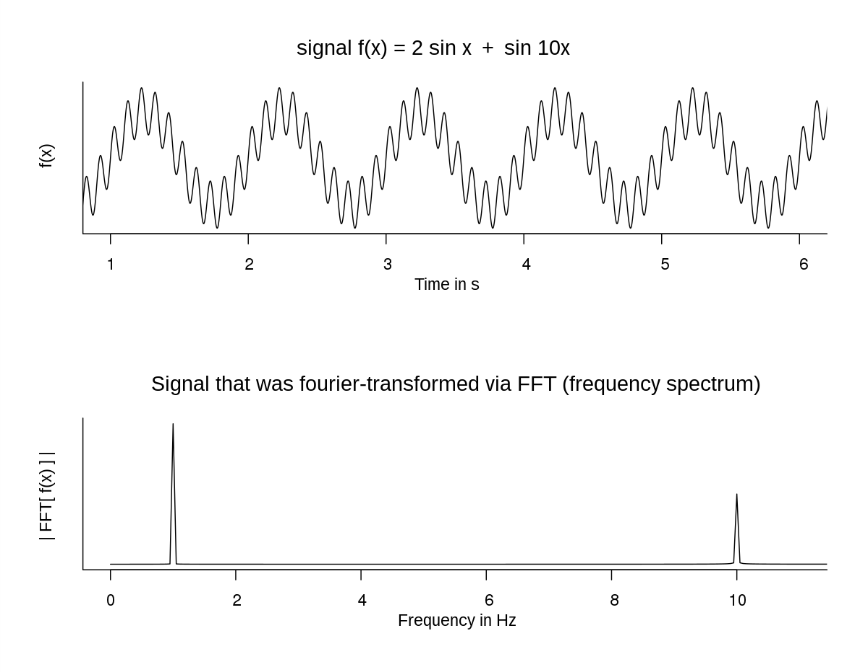
Zu der Realisierung dieser Funktion sollte erwähnt werden, dass Python kein echtes Multithreading unterstützt. Daher laufen die Threads nicht parallel auf unterschiedlichen Kernen. Für den gewünschten Zweck ist die vorhandene Implementierung jedoch hinreichend, da der Main Thread die meiste Zeit wartend verbringt und daher ohnehin selten Rechenzeit in Anspruch nimmt. Bei einer späteren Weiterentwicklung könnte statt eines zweiten Threads ein separater Prozess erzeugt werden, welcher die Ausführung des Musikmodus übernimmt. Dies würde eine parallele Ausführung ermöglichen und eventuell die Performance der Anwendung steigern.

### Audioaufnahme

Die Aufnahme und Verarbeitung des Audiosignals beginnen mit dem Starten des Musik Threads, sobald das entsprechende Ergebnis der [Sonos Abfrage](#_Abfrage_der_Sonos) erhalten wurde. 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 Pi 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 *<recordAudio>* 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. Diese Ausschläge werden im Folgenden für die Berechnung der RGB-Farbwerte verwendet. Die Anwendung bietet die Möglichkeit zwischen zwei Darstellungsmöglichkeiten zu wählen. Die Berechnung der Darstellung kann auf Basis der Wellenlänge oder eines berechneten Dezibel Wertes erfolgen. Welche der Möglichkeiten verwendet wird, kann in der Konfigurationsdatei festgelegt werden.

### Farbberechnung anhand von Wellenlänge

Für die Berechnung der Farben aus den Ausschlägen des Audiosignals wurden Grundlagen und Inhalte des Psynesthesia Repositories genutzt. Verwendete Codeausschnitte aus diesem Repository sind in der Funktion zur Umwandlung und Berechnung der Farbwerte zu finden (*<freqToRGB>* und *<wavelen2RGB>*) und sind im Sourcecode 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

Abbildung - Darstellung Fourier Transformation (https://de.wikipedia.org/wiki/Schnelle\_Fourier-Transformation)

Fourier-Transformation in ihre Frequenzbestandteile zerlegt werden. Wie in dem Beispiel aus Abbildung 4 zu sehen, werden aus der oben dargestellten Schwingung Frequenzen und deren Anteile im Bereich von 1 Hz und 10 Hz berechnet.

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 die Frequenz mit dem größten Anteil bestimmt wird. Die so bestimmte Frequenz wird im Anschluss für die Berechnung der Farbe verwendet.

Ton und Licht bestehen aus einer ähnlichen Art von Welle. Ein Ton wird eher als eine 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 eher 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 5 zu sehen. Alle Frequenzen darunter zählen zum Bereich des Infraroten Lichts und sind für Menschen als solches nicht erkennbar.

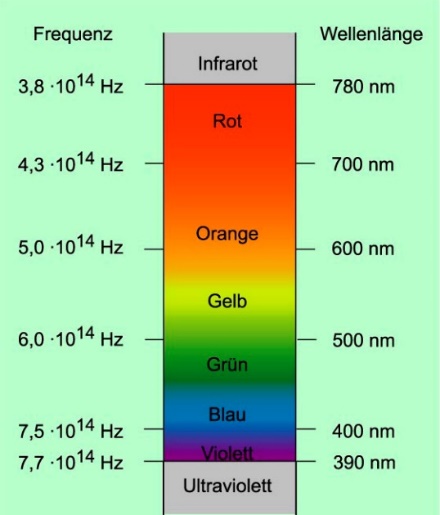


Abbildung - Frequenzen von Farben (<https://www.lernhelfer.de/schuelerlexikon>

/physik-abitur/artikel/spektrum-elektromagnetischer-wellen)

An diesem Punkt wird bereits deutlich das ein physikalischer Zusammenhang von Tonfrequenzen und Lichtfrequenzen besteht, 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 6 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. Nach Abschluss der Berechnung des RGB-Farbwertes, kann mithilfe der Klasse LED die Farbe entsprechend angezeigt werden.

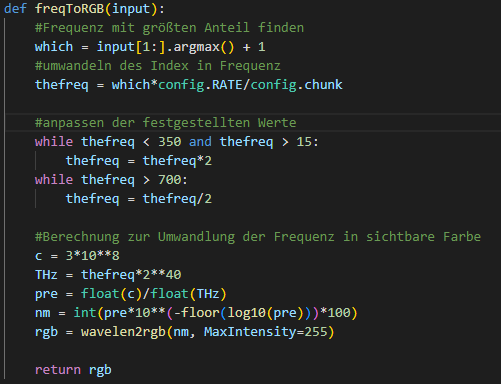


Abbildung - Funktion Berechnung Farben

#### Anzeigen der Farben auf den LED-Streifen

Nach dem 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 nutzt zu Adressierung der angeschlossenen LEDs 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-Farbwert dar. Die berechneten Farben von den Frequenzen, 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. So wird verfahren bis irgendwann der Musikmodus durch einen Moduswechsel der Sonos Anlage beendet wird. Bei Beendigung des Musikmodus, werden alle LEDs ausgeschaltet (siehe oben Abbildung 2).

### Farbberechnung und Anzeige anhand von Dezibel Werten

Wurde dieser Effekt in der Konfigurationsdatei gewählt, wird in der Funktion *<AudioDB2Amplitude>* der mittlere Dezibel-Wert der Audioaufnahme ermittelt. Für diese Berechnung sind ebenfalls die Daten der Fourier Transformation notwendig, um im Anschluss die Berechnung der Dezibel-Werte der einzelnen der Frequenzbereiche zu ermöglichen. Für die darauffolgende Berechnung der Farbwerte, wird der mittlere Dezibel-Wert aller Frequenzen verwendet.

In der Funktion *<Impulse>* bestimmt die Klasse ledStrip eine Anzahl an LEDs welche kurz aufleuchten sollen. Die Funktion erhält als Übergabeparameter den zuvor ermittelten Dezibel-Wert. Dieser wird vor den weiteren Schritten modifiziert, um den Einfluss des Grundrauschens des Mikrofons auf die weitere Berechnung zu reduzieren. In den folgenden Schritten wird zunächst der prozentuale Anteil des Dezibel-Wertes vom festgelegten Maximum bestimmt. Dieser ermittelte Prozentwert legt fest wie viele der angeschlossenen LEDs aufleuchten sollen. Wurde die Anzahl der LEDs bestimmt, werden die Listen für die RGB-Farbwerte ausgehend von der Mitte mit den entsprechenden Farbwerten beschrieben.

## Probleme der Farbanzeige auf dem LED-Streifen

Bei der Ausführung der Anwendung fallen Probleme bei der Darstellung von Farben auf langen LED-Streifen auf. Werden beispielsweise 100 oder mehr LEDs angeschlossen ist eine zunehmende Verzögerung bei der Übertragung auf den Streifen zu sehen. Dieses Verhalten lässt darauf schließen, dass die Verzögerung bei der Aktualisierung des LED-Streifens entsteht.

Eine Analyse des Problems macht deutlich, dass die Zugriffe auf die Liste der LEDs von neopixel einen erheblichen Zeitaufwand darstellen. Wie in der Ausgabe aus Abbildungen 7 zu sehen, benötigt das Schreiben der RGB-Farbwerte in die Liste einen erheblichen Zeitaufwand. Wohingegen der Befehl zur Übertragung der Farben auf den LED-Streifen einen sehr geringen Zeitaufwand benötigt.

Ein Bild, das Text enthält.

Automatisch generierte BeschreibungEin Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung - Ergebnis Zeitmessung

Abbildung 8 - Codeauschnitt Messung Ausführdauer

Dieses Verhalten ist für die aktuelle Verwendung keine Einschränkung, da die Effekte nicht zeitkritisch sind. Allerdings schränkt dies zunehmend die Weiterentwicklung ein.

# Fernsehmodus

Wurde durch die Abfrage der Sonos Anlage ermittelt, dass im Moment ferngesehen wird, schaltet die Anwendung auf den entsprechenden Modus um. Hierfür werden durch das ledStrip Objekt alle LEDs in einer Farbe zum Leuchten gebracht. Die Farbe in der die LEDs leuchten kann in der Konfigurationsdatei entsprechend angepasst werden.

# Fazit und Potential

Zusammenfassend kann festgestellt werden, dass alle zu Beginn festgelegten Funktionen in die Anwendung implementiert werden konnten. Das Starten und Beenden der unterschiedlichen Modi wurde durch die Abfrage der Sonos Anlage automatisiert. Neben den Modi für Fernsehen und Musik, sind auch verschiedene Berechnungen von Effekten zu abgespielter Musik implementiert. Diese Effekte berücksichtigen unterschiedliche Aspekte des Audiosignals und sind in der Lage passende Farbewerte auf den LED-Streifen zu übertragen.

Für eine Weiterentwicklung bietet die Anwendung verschiedene Möglichkeiten, da beispielsweise Effekte hinzugefügt werden könnten. Hierfür wäre jedoch eine Steigerung der Performance von enormen Vorteil, da dies beispielsweise die Darstellung zeitkritischer Effekte, wie das Pulsieren im Rhythmus der Musik, ermöglichen würde. Hierfür müsste jedoch die thematisierte Verzögerung bei der Anzeige der RGB-Farbwerte verringert werden. Wie bereits erwähnt könnten andere Bibliotheken zur Steuerung des LED-Streifens oder auch die Verwendung eines schnelleren Speichermediums im Raspberry Pi getestet werden.

# Quellenangaben

Repository Psnesthesia, Stand 19.02.2023, <https://github.com/off-by-some/Psynesthesia>