Technische Dokumentation

Audiolighter

Von

Sebastian Schultz, 2292895
Tobias Pleß, 2069863
Trutz Gebhardt, 2357854
Abdulkerim Ekmekci, 1982235

19. Juli 2019

Mobile Systeme & IT-Systeme

Studiengänge Medientechnik (B.Sc.) und Media Systems (B.Sc.)
Sommersemester 2019

Hochschule für angewandte Wissenschaften Hamburg /
Hamburg University of Applied Sciences

Department Medientechnik
Fakultät Design, Medien und Information

Inhaltsverzeichnis

Vorwort

Im Rahmen des Kurses Mobile Systeme & IT-Systeme bestand die Semesterleistung darin ein Projekt zu verwirklichen, bei dem ein DMX-fähiges Gerät angesprochen werden soll. Die DMX-Werte sollten dabei durch einen Arduino erzeugt werden. Die Informationen zur Steuerung des Arduinos sollen von einer selbst entwickelten Android-App kommen. Der Arduino und das Smartphone sollen dabei über Bluetooth verbunden werden.
Im Rahmen der Ideen- & Gruppenfindung am 02.04.2019 entschieden wir uns dazu eine Anwendung zu kreieren, die es dem Anwender ermöglicht Audiosignale in Form eines Liedes visuell über eine Lichtinstallation darzustellen. Dabei sollten die LEDs der ausgewählten LED-Bar frequenzabhängig leuchten.

Hardwarekomponenten

Im folgenden werden die Hardwarekomponenten aufgelistet und eine kurze Einführung in deren Spezifikationen und wie sie im Projekt angewendet wurden.

Samsung Galaxy S8

Das Samsung Galaxy S8 ist ein Smartphone, dass die benötigten Funktionen bietet und über ausreichend Rechenleistung verfügt um eine FFT durchzuführen. Das gerät läuft unter Android 8.0 und ist bluetoothfähig, was die Realisierung unserer Idee möglich macht. Im Projekt wurde auf diesem Mobiltelefon unsere App installiert.

Arduino MEGA 2560

Der Arduino MEGA 2560 verfügt über 54 digitale I / O-Pins und 16 analoge Eingänge. Außerdem sind noch 4 UARTs (serielle Hardware-Ports), ein 16-MHz-Quarzoszillator, eine USB-Verbindung, eine Stromversorgungsbuchse, einen ICSP-Header und eine Reset-Taste vorhanden. In unserem Projekt hat er per Bluetooth ankommende Informationen verarbeitet und in DMX Signale umgewandelt.

HM-10 Modul

Das HM-10 Modul ist ein Bluetooth Transmitter/Receiver. Es arbeitet mit Bluetooth 4.0 Low Energy und wurde in unserem Projekt als Empfänger am Arduino angeschlossen.

MAX485 Modul

Das MAX485 Modul erzeugt aus einem einfachen Signal ein differentielles Signal. Wir nutzen es um aus dem einfachen DMX-Signal ein differentielles DMX-Signal zu erzeugen wie es in der Praxis verwendet wird. Dadurch wird der Signaltransport über DMX-Leitungen deutlich resistenter gegenüber Störungen.

Litecraft Powerbar x.15

Die Litecraft Powerbar x.15 ist eine LED Bar und verfügt über 15 Optiken die mit jeweils einer roten, grünen, blauen und weißen LED befeuert werden. Sie kann mit bis zu 65 DMX-Kanälen gesteuert werden. Wir verwenden sie im 60-Kanal-Modus mit dem wir jede LED einzeln ansteuern können. Dabei sind die DMX-Kanäle 1 / Rot, 2 /Grüne, 3 /Blau und 4 /Weiß.

Peripherie

Zusätzlich kommen noch ein Grove Mega Shield sowie DMX-Kabel, DMX-Adapter und USB-Kabel zum Einsatz.

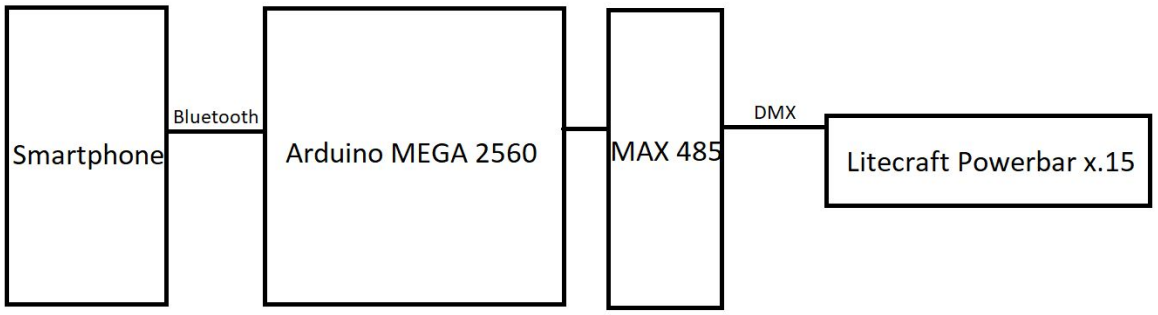

Abb. 1: schematische Projektübersicht

Software Arduino

Funktionen

Setup

Im Void Setup wird zunächst das Lesen der seriellen Schnittstellen mit einer Baudrate von 115200 Bps gestartet und die Verwendung der DMX Bibliothek initialisiert. Zudem werden mit einer For-Schleife die DMX-Werte der DMX-Kanäle 1-63 auf 0 gesetzt.

Setzen der Lampenfarbe (SetLampColor)

In Zeile 26 wird die Funktion gestartet die den einzelnen LEDs die DMX-Werte zuweist. Da wir nur die rote, grüne und blaue LEDs benutzen, nicht aber die weiße, werden hier immer nur 3 Werte geschrieben. Die Lampen ID (“lampid”) bezeichnet dabei die position der Lampe auf der Powerbar.

IDLE-Funktion

Ab Zeile 41 startet die IDLE-Funktion. Sie dient dazu ein nettes Licht zu erzeugen, wenn keine Daten zu einem Lied per Bluetooth übermittelt werden. Zunächst wird mittels For-Schleife der DMX-Kanal um einen erhöht und dann mit einer weiteren For-Schleife der DMX-Wert hochgezählt. Das führt dazu, dass die LEDs “langsam” nacheinander hoch gefadet werden. Wenn nun der höchste DMX-Wert erreicht ist, werden die einzelnen Optiken nacheinander ausgeschaltet. Mit zwei If-Anwendungen wird sichergestellt, dass sobald ein Bluetooth-Signal anliegt, die IDLE-Funktion unterbrochen wird und die Bluetooth-Daten verarbeitet werden. Die IDLE-Funktion wird gestartet, wenn der “idletimer” die in Zeile 151 festgelegt Anzahl erreicht hat. In unserem Projekt arbeiten wir mit ca. 10 Sekunden.

Hauptfunktion

Im loop findet dann die eigentliche Hauptfunktion statt. Es werden die Daten der Bluetooth-Übertragung gelesen und auf Kommas untersucht. Wenn ein Komma gefunden wird, wird es in den Buffer geschrieben, mit Nullen aufgefüllt und in eine Zahl umgewandelt. Zudem wird bei jedem Durchlauf der Counter hochgezählt. Sobald ein “!” im String vorhanden ist, wird dieser ausgewertet. Die Position der Lampe wird durch den Counter vorgegeben. Dabei entsprechen die in einem Datensatz zuerst ankommenden Zahlen dem Bassbereich und die letzten Zahlen dem Hochtonbereich. Die Höhe des Value-Wertes bestimmt nun die LED die leuchtet. Ist der Wert kleiner gleich 33 leuchtet die Rote LED. Ist der Wert zwischen 33 und 66 leuchtet die grüne LED und wenn der Wert größer als 100 ist, leuchtet die

Blaue LED. Damit das funktioniert, werden die neuen Informationen in die SetLampColor-Funktion geschrieben, die den Werten dann die DMX-Wert zuweist.

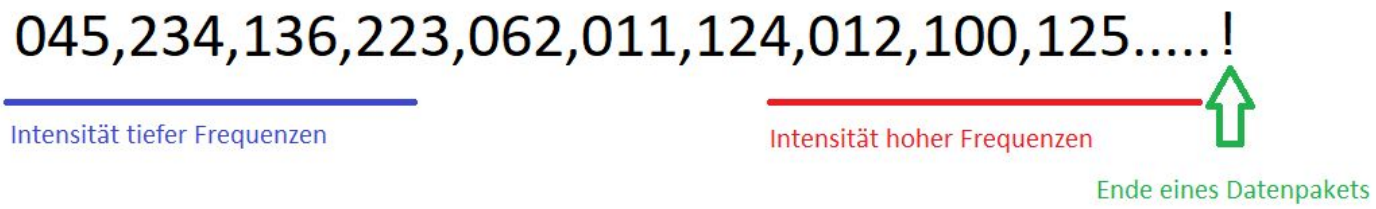

Abb. 2: Beispielprotokoll

Software Android

Audiolighter

Audiolighter ist eine Android-Applikation zur visuellen Wiedergabe von Audio-Elementen. Auf dem Android-Gerät abgespielte Audio-Elemente werden in Echtzeit analysiert und verarbeitet. Über eine Bluetooth-LE Verbindung werden die gewonnenen Daten an einen Arduino übertragen. Dieser wertet die erhaltenen Daten aus und erzeugt je nach erhaltenem Wert ein Licht in einem anderen Spektrum. Die einzelnen Elemente und deren technische Umsetzung werden nachfolgend beschrieben.

Realisierung (Android-Applikation)

Die Web-Applikation Audiolighter, die im Rahmen des Kurses „Mobile Systeme“ konzipiert wurde, besteht hauptsächlich aus Java-Elementen, deren Inhalte und Funktionen durch XML-Elemente aufgerufen und gesteuert werden.
Das Grundgerüst der App wurde mithilfe eines unter der Apache Lizenz stehenden Audio-Visualizer Projektes realisiert. Diesem Projekt liegt eine Android Bibliothek zugrunde, deren Code zur Bearbeitung des Projektes erweitert wurde. Bestehende Elemente des User-Interfaces wurden durch selbsterstellte Shapes ersetzt.
Der Code zur Durchführung einer FFT wurde aus einer freien Sammlung übernommen und integriert.

Funktionen

User Interface

Das User Interface besteht aus einer Hauptseite mit mehreren Buttons. Über diese können verschiedene voreingestellte Audio-Elemente abgerufen werden. In den neu aufgerufenen Activities werden die Audio-Elemente visuell dargestellt. Über die Android eigene Back-Funktion kann der User zur Hauptseite zurückkehren.

Datenverarbeitung der Audio-Elemente

Wird eine der Visualizer-Activities aufgerufen wird automatisch ein zuvor eingestelltes Audio-Element abgespielt. Die aus dem Audio-Stream entnommenen Bit-Werte werden als double-Werte interpretiert und zu einem Array aus komplexen Zahlen gewandelt. Diese Werte werden anschließend mithilfe einer FFT einem neuen Array mit den ausgerechneten Frequenzanteile hinzugefügt.
Die Werte des Arrays werden in 68er-Schritten zu 15 Mittelwerten zusammengefasst und in einer maximalen Nachrichtengröße von 22-Bit, die Nachricht wird bereits hier in kleinere Abschnitte aufgeteilt, zunächst über eine Broadcast Funktion innerhalb der Applikation weitergeleitet. Der Broadcast wird in der MainActivity ausgelesen und die Nachrichten werden schlussendlich über den Aufruf der write()- Methode der BluetoothLeService Klasse an den Arduino übermittelt.

Bluetooth Verbindung

Die Bluetooth Verbindung wird über die BluetoothLeService.kt-Datei geregelt. Diese wird beim Start der Applikation aufgerufen. Die MAC-Adresse des Arduino ist voreingestellt und die Android- Applikation verbindet sich nur zu diesem. Informationen können bis zu einer maximalen Größe von 22 Bit übertragen werden.

Fazit

Leider werden im Verlauf der Verarbeitung Daten verschluckt. Das heißt, dass einige Intensitätswerte einfach nicht mitgeschickt wurden. So kommen wir pro Übertragung nicht auf die “richtigen” 15 Werte, sondern erhalten weniger Werte. Dann verschieben sich für die nächste Übertragung die Werte, sodass der Audiolighter nicht korrekt funktioniert. Wir haben hier natürlich als erstes versucht herauszufinden, welcher Schritt des Projektes die Probleme verursacht. Zuerst bauten wir eine separate Ausgabe der geschickten Bluetooth Werte auf der Android-Seite ein und kontrollierten, ob überhaupt die korrekten Intensitätswerte rausgeschickt wurden. Dies bestätigte sich, die Ausgabe zeigt an, dass alle Werte übertragen werden. Da wir auf der Arduino Seite aber nicht alle Werte empfangen

haben, haben wir vermutet, dass es einfach eine zu große Menge an Daten ist, die wir verschicken wollen. Wir haben dies mit einem weiteren Test aber widerlegen können. Wir haben in Android Studio eine Test-App geschrieben, die zufällige Intensitätswerte zwischen 0 und 255 an den Arduino schickt. Hier war es möglich, wesentlich mehr Werte pro Sekunde als in der richtigen Übertragung zu senden. Die Werte kamen weiterhin alle korrekt an, das heißt es wurden zwischendurch keine Werte "verschluckt". Trotz intensiver Hilfe durch die beratenden Professoren und Tutoren konnten wir bis zum Schluss leider nicht herausfinden, an welcher Stelle die Daten verschwinden. Unsere Vermutung ist, dass die Berechnung der FFT Werte auf dem Smartphone die Bluetooth Übertragung in einer Weise stört. Durch das Auftreten dieses Fehlers kommen die Daten nicht in der richtigen Reihenfolge an, bzw. fehlen ganze Datenstränge, sodass die Zuweisung der Frequenzen auf die einzelnen Segmente der LED-Bar nicht funktioniert. Nichtsdestotrotz ist die visuelle Wirkung des Audiolighters angenehm anzuschauen und die Frequenzanalyse auf der Android Seite erkennbar.

Quellen

Gaurav Kumar: „Audio Visualizer", unter:
https://github.com/gauravk95/audio-visualizer-android(abgerufen am 28.5.2019)

Autor unbekannt: „Fast Fourier Transform",
unter:https://rosettacode.org/wiki/Fast_Fourier_transform#Java (Abgerufen am 28.5.2019)

Allgemein:
https://www.arduino.cc/reference/en/