Bernburg
Dessau
Köthen

Hochschule Anhalt
Anhalt University of Applied Sciences

emw
Fachbereich
Elektrotechnik, Maschinenbau
und Wirtschaftsingenieurwesen

Projektarbeit

als Wahlpflichtmodul des Bachelor-Studiengangs
Medientechnik

Max Pinkert

Name

Medientechnik, 5014556

Studiengang, Matrikelnummer

Thema:

„Arduino basierte Unity3D Steuerung“

Herr Prof. Tümler

Hochschulmentor(in)

Wintersemester 2023

Bearbeitungszeitraum

01.03.2023

Abgabe am

Inhaltsverzeichnis

1 Einleitung

Im Printworks London, einer der international anerkanntesten Veranstaltungsorte für elektronische Musik und Performances, werden regelmäßig beeindruckende Animationen geboten die in Kombination mit Audio hervorragend harmonieren. Licht, Ton und Controller für Videotechnik bilden Herzstücke, die diese Darbietung von visuellen Effekten ermöglicht. Einzelne Elemente der Animation werden durch das Betätigen von Fadern, Reglern und Buttons angesteuert, die sich in Form und Farbe modulieren lassen. In dieser Projektarbeit wird eine Arduino basierte Steuerung vorgestellt, die verschiedene Elemente einer Unity3D Animation modifizieren soll.

1.1 Motivation

Das Printworks London hat sich zum Ziel gesetzt Licht, Ton und Video perfekt aufeinander abzustimmen, um Publikum und Künstlern eine erstuanliche Darbietung visueller Animationen bieten zu können. Dabei werden Controller verschiedener Hersteller für die Änderung von Parametern eingesetzt. Ein wichtiger Aspekt dieses Ziels ist die Steuerung von Controllern, die es ermöglicht, audiovisuelle Effekte in Echtzeit zuerzeugen und anzupassen. Die derzeit verfügbaren Controller sind zwar leistungsstark aber komplex, teuer und schwierig zu bedienen. Die Vielzahl der Controller und damit verbundene unterschiedliche Steuerungsmöglichkeiten , erschwert den Einstieg vieler unerfahrener Anwender. Die Kombination von Arduino und Unity3D bietet eine Möglichkeit, Einsteigern die Steuerung von Controllern und Signalverarbeitung näher zu bringen und ein besseres Verständis über diese Technik zu vermitteln.

1.2 Zielsetzung

Das Ziel dieser Arbeit ist es, eine neue Methode zur Steuerung von Controllern die einen Arduino verbaut haben und der Entwicklungsumgebung Unity3D zu entwickeln. Hierbei soll insbesondere die Integration von Arduino zu Unity3D untersucht werden. Die Arbeit wird sich mit den grundlegenden Komponenten der Steuerungsmöglichkeiten des Arduino in Unity3D befassen. Außerdem stellt diese Arbeit einen detaillierten Schaltplan und eine Anleitung zur Montage und Programmierung des Controllers bereit. Die entwickelte Methode soll anschließend anhand der Funktionalität getestet und evaluiert werden.

1.3 Aufbau der Arbeit

Die Arbeit gliedert sich in sechs Kapitel. Im ersten Kapitel wird eine Einführung in das Thema gegeben und die Motivation und Zielsetzung erläutert. Das zweite Kapitel beschäftigt sich mit den Grundlagen des Controllers und den verwendeten Technologien, insbesondere Arduino und Unity3D. Im dritten Kapitel wird die verwendete Methodik erläutert, einschließlich der Komponenten und Werkzeuge, die für die Entwicklung der Steuerung verwendet werden. Das vierte Kapitel beschreibt den Prozess der Integration von Arduino und Unity3D zur Steuerung des Controllers. Im fünften Kapitel werden verschiedene Anwendungsbeispiele für die Steuerung von Controllern präsentiert und diskutiert. Abschließend wird im sechsten Kapitel eine Zusammenfassung der Ergebnisse und ein Ausblick auf mögliche zukünftige Entwicklungen gegeben. Insgesamt soll diese Arbeit einen Beitrag zur Verbesserung der audiovisuellen Erfahrung in Veranstaltungsorten wie dem Printworks London leisten, indem sie eine einfachere und zugänglichere Methode zur Steuerung von Controllern aufzeigt.

2 Stand der Technik

Bevor man sich mit der Arduino basierten Unity3D Steuerung für Controller befassen kann, ist es wichtig, die zugrunde liegenden Technologien und Komponenten zu verstehen. In diesem Kapitel werden die grundlegenden Konzepte und Komponenten vorgestellt, die für das Verständnis und die Entwicklung der Steuerung erforderlich sind. Hierbei werden auch aktuelle Entwicklungen und Stand der Technik berücksichtigt, um eine dazu gehörende Einordnung der Technologien zu ermöglichen. In dieser Arbeit bilden Arduino, Unity3D, Potentiometer und der Multiplexer IC4051 die Grundlagen für die zu entwickelnde Arduino-basierte Unity3D-Steuerung. In den folgenden Kapiteln wird untersucht, wie diese Technologien verwendet werden können, um eine interaktive Steuerung zu entwickeln.

2.1 Arduino

Arduino ist eine Open-Source-Plattform, die es Entwicklern ermöglicht, einfach und kostengünstig Mikrocontroller-basierte Projekte zu entwickeln. Die Arduino-Plattform ist besonders für Anfänger geeignet, da sie eine benutzerfreundliche Entwicklungsumgebung und eine einfache Einführung der Programmiersprache C bietet. Arduino, welches in Abbildung 1 gezeigt wird, umfasst eine Hardware-Plattform, die aus einem Mikrocontroller und verschiedenen Anschlüssen für Sensoren und Aktoren besteht, sowie eine Software-Plattform, die es ermöglicht, den Mikrocontroller mithilfe der C Programmiersprache zu programmieren. Die Hardware-Plattform wurde so entwickelt, dass sie einfach mit grundlegenden Kenntnissen der Programmiersprache C zu verwenden ist, ohne dass tiefgreifende Elektronikkenntnisse erforderlich sind. Grundkenntnisse im Bereich Elektrotechnik sind jedoch Vorraussetzung für den weiteren Verlauf dieser Arbeit. Die Software-Plattform besteht aus einer Entwicklungsumgebung, der Arduino IDE, in der Code geschrieben und auf den Mikrocontroller übertragen werden kann (vgl. [1]).

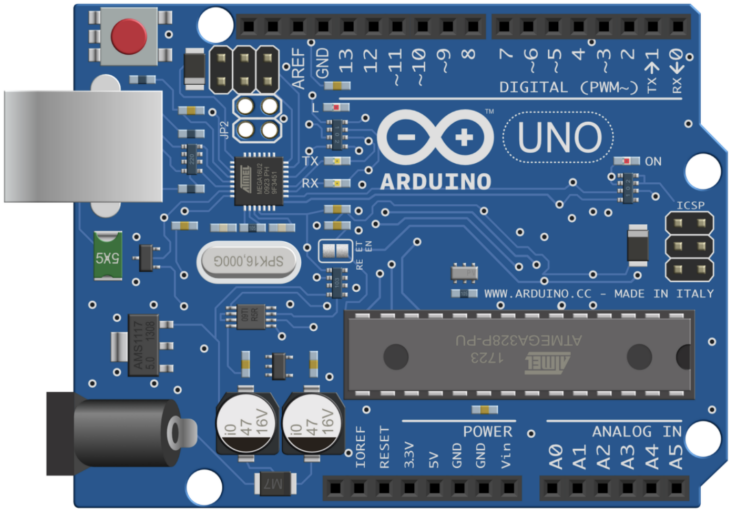

Abbildung 1: Arduino UNO Mikrocontroller [Quelle: Abb1]

[1]

Die Verwendung eines Arduino-Boards ist eine beliebte Methode, um verschiedene Projekte im Bereich der Elektronik und Programmierung umzusetzen. Das Arduino-Board ist ein Mikrocontroller-Board, das auf einem ATmega328-Mikrocontroller basiert und von der Arduino-Software unterstützt wird. Es bietet eine einfache Möglichkeit, Sensoren, Aktoren und andere elektronische Bauteile zu steuern und Daten zu sammeln. Das Arduino verfügt über eine Anzahl von drei verschiedene Arten von Pins, die in der Abbildung 2 vorgestellt werden. Die Power Pins werden für die Stromversorgung der anzuschließenden Sensoren und Aktoren bereitgestellt. Digitale Pins können entweder als Eingänge oder Ausgänge verwendet werden, während analoge Pins für angeschlossene analoge Sensoren und Aktoren genutzt werden (vgl. [2]).

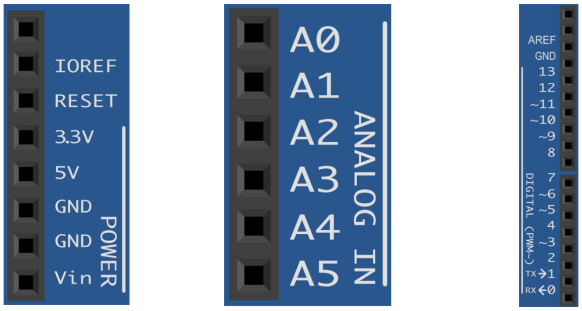

Abbildung 2: Diese Abbildung zeigt Power Pins (linke Abbildung), Analog Pins (mittlere Abbildung) und Digital Pins (rechte Abbildung) [Quelle: Abb2]

2.2 Unity3D

Unity3D ist eine der bekanntesten Game-Engines, die es derzeit auf dem Markt gibt. Diese Engine wird von Entwicklern und Studios eingesetzt, um 2D- und 3D-Visualisierungen, Animationen und Spiele für eine Vielzahl von unterschiedlichen Plattformen zu entwickeln. Die Engine bietet eine benutzerfreundliche Entwicklungsumgebung, viele Tools und eine umfangreiche Dokumentation, um die Entwicklung von Anwendungen und Spielen zu erleichtern. Unity3D unterstützt mehrere Plattformen, einschließlich PC, Mac, Android und iOS. Die Engine verfügt über eine große Community und eine reiche Dokumentation, die es Entwicklern ermöglicht, Projekte einfach und effizient umzusetzen. Außerdem gibt es eine Vielzahl von Bibliotheken und Assets, die es ermöglichen, die Entwicklung von Anwendungen und Spielen zu beschleunigen (vgl. [3]).

„Es gibt verschiedene Arten von Assets. Sie können Texturen, Animationen, Modelle, Charaktere, Projektbeispiele, Tutorials oder Editor-Erweiterungen sein.“ [4].

2.3 Multiplexer-Chip IC4051

Der Multiplexer IC4051 ist ein elektronischer Schaltkreis, der es ermöglicht, mehrere analoge Signale auf eine einzige Leitung digital zu übertragen. Ein Multiplexer verfügt über mehrere analoge und digitale Eingänge und einen Ausgang und kann so konfiguriert werden, dass nur eines der

[1] Quelle Abb1, Abb2 : https://edistechlab.com/arduino_uno/?v=3a52f3c22ed6

Eingangssignale zum Ausgang weitergeleitet wird. Der IC4051 ist ein 8-Kanal-Multiplexer, was bedeutet, dass er bis zu 8 analoge Eingangssignale verarbeiten und auf eine einzige Leitung übertragen kann. Dies macht ihn zu einer praktischen Lösung für Anwendungen, bei denen es erforderlich ist, mehrere Signale zu verwalten und zu übertragen, ohne dass eine Vielzahl von Leitungen benötigt werden.

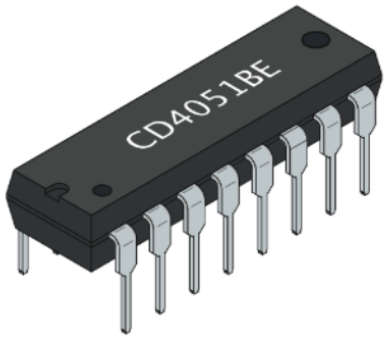

Abbildung 3: Multiplexerchip der Reihe IC4051 [Quelle: Abb3]

[2]

Die Verwendung eines Multiplexers wie dem IC4051 kann dazu beitragen, die Anzahl der erforderlichen Anschlüsse zu reduzieren und den elektronischen Schaltkreis einfacher und kompakter zu gestalten. Außerdem kann es die Übertragung von Signalen vereinfachen, indem es ermöglicht, mehrere Signale verarbeiten zu können (vgl. [5]).

2.4 Potentiometer

Ein Potentiometer besteht aus einem Widerstand, der in der Regel als linear oder logarithmisch ausgelegt ist, und einem Schleifer, der sich entlang des Widerstands bewegen kann. Wenn der Schleifer sich entlang des Widerstands bewegt, ändert sich der Widerstandswert und somit auch die Spannung, die an diesem Punkt gemessen wird. Die Verwendung von Potentiometern in verschiedener Controller Steuerungen ermöglicht beispielsweise eine präzise und intuitive Steuerung von Parametern wie Helligkeit, Farbtemperatur, Effektintensität und vielen anderen.

[2]Quelle Abb3 : https://edistechlab.com/mulitplexer-cd4051be-einfach-erklaert/?v=3a52f3c22ed6

3 Konzeption

Die Konzeption beinhaltetet die Anforderungsanalyse für die Realisierung einer Arduino basierten Steuerung in Unity3D. In diesem Abschnitt wird auf spezielle Rahmenbedingungen und Anwendungsorientierte Nutzung eingegangen. Außerdem werden einzelne Komponenten nochmals genauer betrachtet, damit im Rahmen dieser Arbeit, die Umsetzung und Anwendung eines Controller entwickeln werden kann. Die Aufgabe des Controller beinhaltet Potentiometerwerte auszulesen und an Unity3D zusenden. Diese Potentiometerwerte werden auf die Größe einzelner Objektgruppen Einfluss haben. Das Ziel der Arduino-basierten Unity3D-Steuerung ist es, eine interaktive Steuerung für eine Anwendung zu entwickeln, die es dem Benutzer ermöglicht, die Anwendung durch Bewegungen oder andere Interaktionen verschiedene Objekte in Unity3D zu steuern (vgl. [1]).

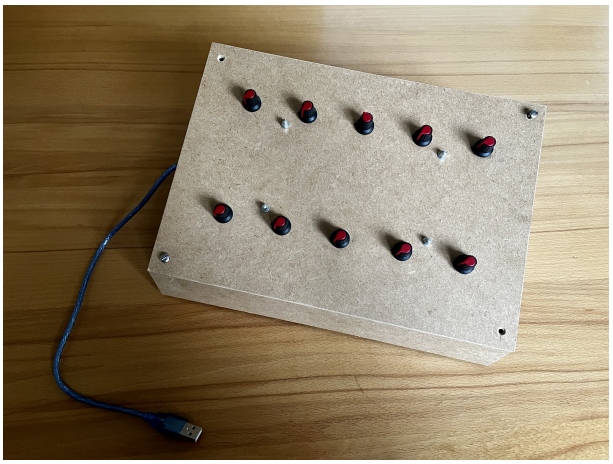

Abbildung 4: Systementwurf

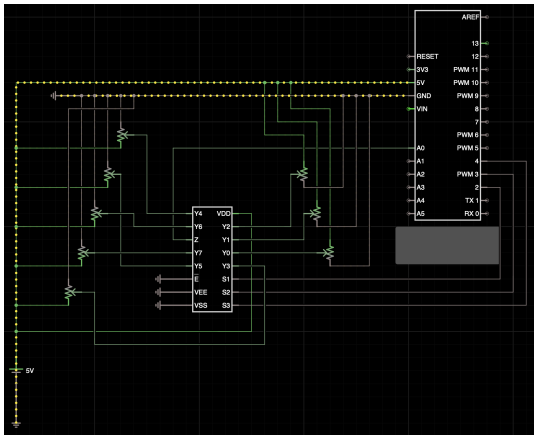

Abbildung 5: Entwurf des integrierten Schaltkreis

3.1 Controller Architektur

Die Architektur der Arduino-basierten Unity3D-Steuerung basiert auf der Verwendung von Poteniometern, diese sind über den Multiplexer IC4051 an den Arduino angeschlossen und übertragen ihre Signale an Unity3D. Wie die Komponenten miteinander verbunden werden, zeigt die Abbildung 5, die einen detailierten Schaltungsentwurf des Controllers darstellt. Die Auswahl der Potentiometern als Sensoren erfüllen die mindest Anforderungen der Arbeit, die Einfluss auf die Größe von Objektgruppen haben soll. Die 8 Objektgruppen stellen 8 Frequenzbänder dar. Die mittleren Pins des Potentiometer, die als Signalleitungen für die ausgegebenen Werte dienen, werden über den Multiplexer IC4051 an den Arduino angeschlossen. Der Multiplexer IC4051 ermöglicht es, mehrere Sensoren, in diesem Fall die Potentiometer, auf eine einzige Leitung an den Pin A0 zu übertragen.

Die Auswahl der aktuell betätigten Potentiometer wird durch die Select-Pins A, B und C am IC4051 gesteuert, diese sind mit den digitalen Ausgängen 2, 3 und 4 am Arduino verbunden. Die Steuereingänge dienen dazu, den Ausgang des Multiplexers auf einen der Eingänge zu schalten. Durch die Kombination der Steuersignale an den drei Eingängen der Select-Pins können insgesamt 8 verschiedene Eingänge ausgewählt werden. Dies ermöglicht es, mehrere Potentiometer an nur einem Analogeingang (Pin A0) des Mikrocontrollers anzuschließen. Abbildung 6 zeigt auf wie die Verteilung der Pins am Multiplexer ist und wie diese, an die verschiedenen Komponenten angeschlossen werden.

Nr.	Name	Funktion
1	Kanal 4 - In/Out	Kanal 4
2	Kanal 6 - In/Out	Kanal 6
3	COM - In/Out	Gemeinsamer In/Out Kanal
4	Kanal 7 - In/Out	Kanal 7
5	Kanal 5 - In/Out	Kanal 5
6	Inhibit	Alle Kanäle an/aus
7	VEE	Negative Spannungsversorgung
8	VSS	GND
9	C	Kanal Auswahl - C
10	B	Kanal Auswahl - B
11	A	Kanal Auswahl - A
12	Kanal 3 - In/Out	Kanal 3
13	Kanal 0 - In/Out	Kanal 0
14	Kanal 1 - In/Out	Kanal 1
15	Kanal 2 - In/Out	Kanal 2
16	VDD	Spannungsversorung 3-20 Volt

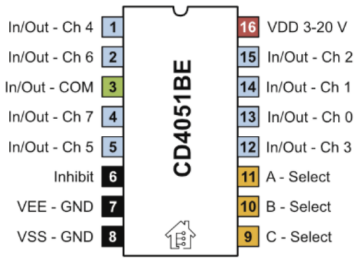

Abbildung 6: Funktionsweise des IC4051 Multiplexer [Quelle: Abb6]

[3]

Die Ansteuerung der Steuereingänge erfolgt über die digitalen Ausgänge des Mikrocontrollers erfolgen. Um beispielsweise den Eingang 3 des Multiplexers zu wählen, muss der binäre Wert 001 an den Steuereingängen A, B und C angelegt werden. Um den Eingang 5 zu wählen, muss hingegen der binäre Wert 100 anliegen.

3.2 Kommunikations Schnittstellen mit Unity3D

Um die Steuerung des Controllers mit dem Arduino zu ermöglichen, muss eine Kommunikationsverbindung zwischen Unity3D und Arduino hergestellt werden. In diesem Kapitel werden drei Möglichkeiten gezeigt, um die Kommunikation zwischen Unity3D und Arduino zu ermöglichen. Diese 3 Varianten umfassen MIDI, Ardity und SerialPort in der C# Umgebung. Der Arduino überträgt die verarbeiteten Signale an Unity3D, wo sie zur Steuerung der Anwendung verwendet werden. Die Steuerung wird in Unity3D programmiert, um die empfangenen Signale zu verarbeiten und die Visualisierungsanwendung entsprechend zu steuern. In dieser Projektarbeit wird sich auf die Variante des SerialPort bezogen um dem Benutzer die Programmiersprache C# und dem seriellen Datentransfer zwischen Unity3D und Arduino näher zu bringen und einen tieferen Einblick zwischen den Komponenten zu bekommen.

[3] Quelle Abb6 : https://edistechlab.com/mulitplexer-cd4051be-einfach-erklaert/?v=3a52f3c22ed6

3.2.1 MIDI

MIDI (Musical Instrument Digital Interface) ist ein Industriestandard, der verwendet wird, um elektronische Musikinstrumente und andere Geräte miteinander zu verbinden. MIDI wird häufig zur Steuerung von Musiksoftware und digitalen Audio-Workstations verwendet. Es ist auch ein gängiges Protokoll zur Steuerung von Beleuchtung und visuellen Effekten, was es zu einer Wahl für die Steuerung eines Controller macht, der visuelle Effekte steuern soll. MIDI ist ein serieller Datenstrom, der aus einer Reihe von Nachrichten besteht. Es gibt drei Arten von Nachrichten die das MIDI Protokoll umfasst, diese sind MIDI-Noten, Controller-Nachrichten und System-Nachrichten. Controller-Nachrichten können verwendet werden, um den Wert eines Potentiometers zu übertragen, während Noten-Nachrichten zur Übertragung von binären Werten verwendet werden können. Um die MIDI-Verbindung zwischen dem Arduino und Unity3D herzustellen, müssen spezielle MIDI-Shields oder USB-MIDI-Interfaces verwendet werden (vgl. [6, 7]).

3.2.2 Ardity

Ardity ist eine Open-Source-Bibliothek, die speziell für die Verbindung zwischen Arduino und Unity3D entwickelt wurde. Mit Ardity können Daten zwischen Unity3D und Arduino senden und empfangen werden. Ardity verwendet die Standard-Firmware, die auf dem Arduino ausgeführt wird, um eine einfache und zuverlässige Verbindung herzustellen. Die Kommunikation zwischen Unity3D und Ardity erfolgt über die Serial-Kommunikation. Unity3D sendet Befehle an Ardity, die auf dem Arduino interpretiert werden. Mit Ardity können sowohl digitale als auch analoge Daten übertragen werden. Es können auch andere Ereignisse ausgelöst werden, wenn bestimmte Aktionen auf dem Arduino ausgeführt werden (vgl. [8]).

3.2.3 SerialPort in C#

Eine weitere Möglichkeit, die Verbindung zwischen Unity3D und Arduino herzustellen, besteht darin, die serielle Schnittstelle des Arduino direkt in der C#-Umgebung von Unity3D zu verwenden. Dazu kann die SerialPort-Klasse in der .NET Framework-Bibliothek verwendet werden, die vorher im User-Interface eingestellet werden muss. Die SerialPort-Klasse ermöglicht eine Verbindung mit seriellen Geräten, einschließlich des Arduino. Durch Verwendung der SerialPort-Klasse können Daten von und zu dem Arduino übertragen werden. Es ist auch möglich, den Wert von Potentiometern und anderen analogen Eingaben zu übertragen, indem die AnalogRead-Funktion auf dem Arduino angewandt um Daten an Unity3D zu senden (vgl. [9]).

3.3 Unity3D Anwendung

In dieser Arbeit werden die Potentiometerwerte in einer Visualisierungsanwendung in Echtzeit dargestellt. Dabei werden die Potentiometerwerte über einen Multiplexer ausgelesen und als Visualisierung von 8 Frequenzbändern in einer Unity3D-Szene dargestellt und entsprechend der ankommenden Daten aktualisiert. Diese Visualisierung, der Unity3D Animation ist in Abbildung 7 dargestellt. In der Arduino-Programmierung wird der Multiplexer 4051 verwendet, um die Potentiometerwerte von insgesamt 8 Potentiometern auszulesen.Die ausgelesenen Potentiometerwerte werden dann an Unity3D übertragen, wo sie von einem C#-Skript, ein Assests, verarbeitet werden. In diesem Skript werden die Potentiometerwerte auf die Größe der Objekte, der 8 Frequenzbänder, Einfluss haben.

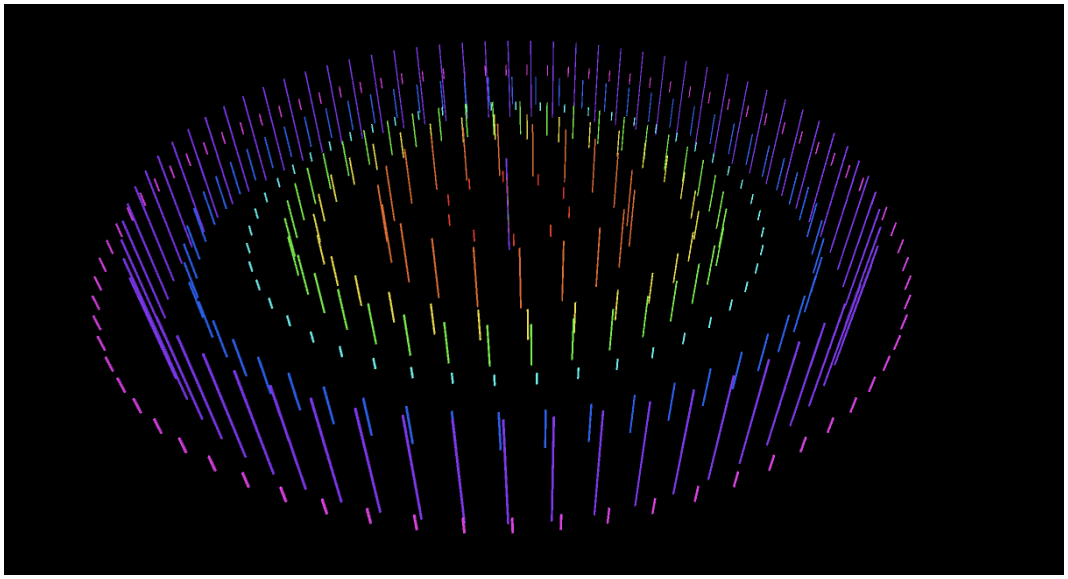

Abbildung 7: Visualisierungs Animation in Unity3D

Die Umsetzung dieser Anwendung erfordert eine genaue Abstimmung zwischen der Arduino-Programmierung und der Unity3D-Programmierung. Diese Synchronisation wird mit der richtigen Einstellung von Eingangs-Port der verbunden Hardware, in diesem Fall ein Mac, und der zusammenhängenden Baudrate hergestellt. Die Baudrate beschreibt die Datenübertragungsrate. Anbei müssen die Multiplexer-Steuerung und die serielle Kommunikation zwischen Arduino und Unity3D korrekt implementiert werden, um eine reibungslose Übertragung der Potentiometerwerte zu gewährleisten.

4 Implementierung

Die Integration des Arduino in Unity3D ist ein wichtiger Schritt bei der Entwicklung einer Arduino-basierten Steuerung für ein 3D-Spiel oder eine Anwendung. In diesem Kapitel werden wir uns mit der Integration des Arduino in Unity3D befassen um eine Visualisierung von Audiomaterial steuern zu können.

4.1 User Interface

Bevor Arduino in Unity3D integriert werden kann, müssen bestimmte Komponenten miteinander abgestimmt werden. Bestimmte Einstellungen werden vorgenommen um eine Interaktion zwischen Objektgruppen und Arduino zu erreichen. Dabei müssen Einstellungen im User-Interface vorgenommen werden. In der Toolbar unter Project Settings wird im Abschnitt Player das richtige Framework gewählt, damit eine Kommunikation zwischen Ardunio und Unity3D besteht. Der Pfad zur Einstellung des richtigen Framework wird in Abbildung 8 und 9 gezeigt.

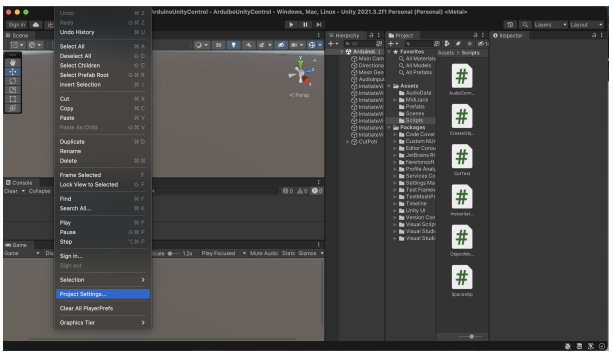

Abbildung 8: Einstellungspfad in Unity3D

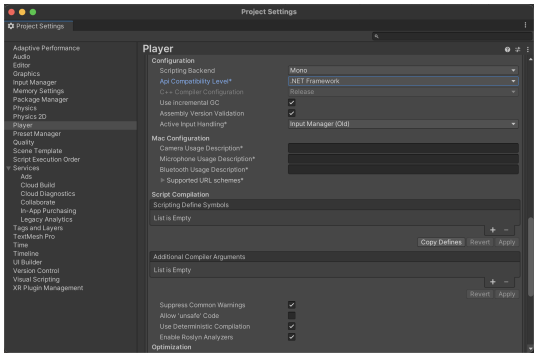

Abbildung 9: Einstellung des Framework

4.2 Arduino Code

Im Folgenden wird erklärt wie die Signale im Arduino verarbeitet werden, um eine interaktive Steuerung zu entwickeln. Der Arduino Code beschreibt eine Schleife, die in jedem Durchlauf die analogen Werte eines Potentiometers ausliest und zusammen mit einer binären Bitzuweisung die 3 Select-Pins angesteuert werden. Zunächst werden Variablen für den Potentiometerwert und Binärzuweisung der Select-Pins in den Zeilen 1 bis 6 des Arduino Codes deklariert. In der Setup-Funktion werden die Pin-Modi und die Serielle Schnittstelle initialisiert. Mit dem Befehl pinMode werden Pin und die Funktion des Pins beschrieben. Das erste Argument im Befehl pinMode gibt den Pin an, welcher verwendet werden soll und das zweite Argument weist dem Pin zu, ob sich der Pin als Aus- oder Eingang verhalten soll. Die Variabe i steht als Zählervariable zur Verfügung. In der Setup-Funktion, ab Zeile 9, werden für die digitalen Ausgänge 2, 3 und 4 des Arduino als Output definiert und mit dem Befehl Serial.begin(9800) wird die Baudrate auf 9800 Bit pro Sekunde gesetzt.

```
1   int potiWert = 0;
2   int i = 0;
3
4   int bit1 = 0;
5   int bit2 = 0;
6   int bit3 = 0;
7
8
9   void setup() {
10    // put your setup code here, to run once:
11    pinMode(2, OUTPUT);
12    pinMode(3, OUTPUT);
13    pinMode(4, OUTPUT);
14    Serial.begin(9600);
15
16  }
```

Abbildung 10: Arduino Code Teil 1

In der loop-Funktion, der Abbildung 11, wird in einer For-Schleife von 0 bis 7 durchgezählt. Dabei wird in jedem Durchlauf die Binärzahl des jeweiligen Zählwerts auf die drei digitalen Ausgänge des Arduinos gesendet. Danach wird der analoge Wert des an den Pin A0 angeschlossenen Potentiometers ausgelesen und über die serielle Schnittstelle an den Computer übertragen. Die Werte werden dabei durch ein Komma getrennt und in einem einzigen Datenstring augegeben. Die richtige Schreibweise der Ausgabe und die richtige Anordnung des Datenstrings ist für das kommende Einlesen in Unity3D entscheidend.

```
18  void loop() {
19    // put your main code here, to run repeatedly:
20
21    for (i = 0; i <= 7; i++) {
22
23      bit1 = bitRead(i, 0);
24      bit2 = bitRead(i, 1);
25      bit3 = bitRead(i, 2);
26
27      digitalWrite(2, bit1);
28      digitalWrite(3, bit2);
29      digitalWrite(4, bit3 );
30
31      potiWert = analogRead(A0);
32      Serial.print(potiWert);
33
34      if (i < 7) {
35        Serial.print(",");
36      }
37      //Serial.print(" , ");
38
39    }
40    Serial.println("");
41
42  }
```

Abbildung 11: Arduino Code Teil 2

In der Abbildung des zweiten Teils des Arduino Codes (Abbildung 11) sind die Variablen bit1, bit2 und bit3 jeweils mit einem bitRead-Befehl verknüpft und der Zählervariable i verbunden. Dieser bitRead-Befehl dient zur ansteuerung der Select-Pins um den Multiplexer 4051 anzusteuern. Der Multiplexer 4051 ist ein analoger Multiplexer, der es ermöglicht, ein analoges Eingangssignal über einen digitalen Ausgang auszuwählen. In diesem Fall werden die binären Bitstellen 0 bis 2 des i-Counters verwendet, um den Multiplexer 4051 anzusteuern. Die Bitstelle 0, stellt dabei das niederwertigste Bit dar. Jedes Bit wird über seinen jeweiligen AUsgang an die digitalen Pins des Multiplexer übertragen. Der Multiplexer wählt dann den entsprechenden analogen Eingang aus und gibt ihn über einen digitalen Ausgang aus. Der Wert des Potentiometers wird mit analogRead() eingelsen. Der Multiplexer wird durch die Änderung der Bits 0 bis 2 gesteuert, um den Wert des Potentiometers an jedem der acht Kanäle des Multiplexers zu übergeben. Die Abbildung 12 stellt die Zuweisung der Bits in Bezug auf deren Ausgänge des Multiplexer dar.

	A	B	C
0	0	0	0
1	1	0	0
2	0	1	0
3	1	1	0
4	0	0	1
5	1	0	1
6	0	1	1
7	1	1	1

Abbildung 12: Wahrheitstabelle des Multiplexer 4051 [Quelle: Abb12]

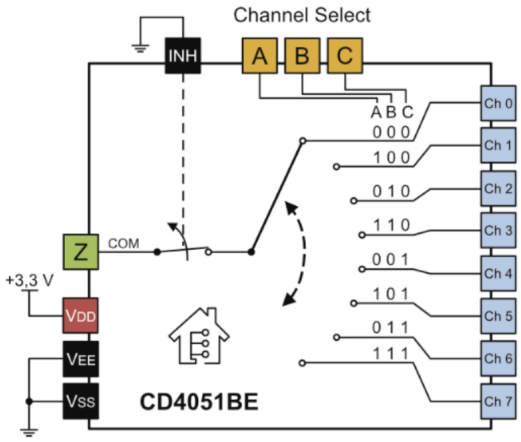

Abbildung 13: Interne Schaltung des Multiplexer 4051 [Quelle: Abb13]

4

[4]Abb12, Abb13: https://edistechlab.com/mulitplexer-cd4051be-einfach-erklaert/?v=3a52f3c22ed6

4.3 Integration des Arduino in Unity3D

Das C# Skript DatenStream enthält die Klassen System.Collections, System.Collections.Generic und UnityEngine, die für jede Unity3D Anwendung als Vorrausetzungen gelten. DatenStream wird in Unity3D als Assets verwendet, um eine serielle Verbindung zum Arduino-Board herzustellen. Außerdem werden die Klassen System.IO und System.IO.Ports importiert. Diese zusätzlichen Klassen definieren eine Reihe benötigte Variablen, die von Unity3D während der Laufzeit einer seriellen Kommunikation gebraucht werden (vgl. [10]).

```
5   using System.IO;
6   using System.IO.Ports;
7
8   public class DatenStream : MonoBehaviour
9   {
10      public Transform Band1, Band2, Band3, Band4, Band5, Band6, Band7, Band8;
11      public float speed = 6f;
12
13      public string SP;
14      public static string SerialPort;
15
16      public int Baud;
17      public static int Baudrate;
18
19      public string receivestring;
20      SerialPort data_stream ;
21
22      public string[] datas = {"0", "0", "0", "0", "0", "0", "0", "0", "0"};
23
24      void Start()
25      {
26          SerialPort = SP;
27          Baudrate  = Baud;
28          data_stream = new SerialPort(SerialPort, Baudrate);
29          data_stream.Open();
30      }
31      void Update()
32      {
33          receivestring = data_stream.ReadLine();
34          datas = receivestring.Split(",");
35
36          if (datas.Length != 8)
37              datas = "0,0,0,0,0,0,0,0".Split(",");
38      }
39  }
```

Abbildung 14: DatenStream Code zur Auswertung der Potentiometerwerte

Die Variable data_stream des Datentyps SerialPort enthält die Baudrate von 9800 Bit pro Sekunde und den Port des angeschlossenen Arduino. Die 8 Transformationsobjekte stellt die Variable Band 1 bis 8 dar, die als visuelle Objekte für die verschiedenen Frequenzbänder verwendet werden. Der Datentyp SerialPort der Variable data_stream ermöglicht eine serielle Kommunikation mit dem Arduino. In die Variable recievestring werden die Potentiometerwerte gespeichert. Die Start-Methode initialisiert die serielle Verbindung mit den entsprechenden Parametern (Port und Baudrate). Mit dem Befehl data_stream.Open() wird eine serielle Verbindung zwischen Unity3D und dem Arduino hergestellt. In der Update-Methode werden die Potentiometerwerte eingelesen um eine Liste des Datenstrings zu erhalten. Dieser Code wird an eine übergeordnete Objektgruppe angehangen, die die 8 Objektgruppen der Frequenzbänder enthält.

4.4 Visualisierung in Unity3D

Bei dieser 3D Visualisierung interagieren die 8 Frequenzbänder mit einer Audiodatei. Diese Skripte CreateObjects und ObjectMove erzeugen 3D Objekte, die je nach Audiosignal skaliert und verschoben werden. In diesem Abschnitt wird auf die Interaktion zwischen den C# Skipten eingegangen und wie diese miteinander zusammenwirken.

4.4.1 CreateObjects

Dieser Code wird in Unity3D verwendet, um Objekte zu erstellen und in einer Reihe anzuordnen. Die Klasse CreateObjects enthält die Variablen radius und amountToSpawn, die den Radius der angeordneten Objekte angibt und wie viele Objekte erzeugt werden. Weiterhin wird ein Array von Objekten, der Variable _sampleCube des Datentyps Gameobject, mit einer Länge von 512 erzeugt.

```
5   public class CreateObjects : MonoBehaviour
6   {
7       public float radius = 1f;
8       public int amountToSpawn = 0;
9
10      public float _startScale, _scaleMultiplier;
11
12      public GameObject _sampleCubePrefab;
13      GameObject[] _sampleCube = new GameObject[512];
14
15      void Start()
16      {
17
18              for (int i = 0; i < amountToSpawn; i++)
19              {
20                  float angle = i * Mathf.PI*2f / amountToSpawn;
21                  Vector3 newPos = new Vector3(Mathf.Cos(angle)*radius, 0 , Mathf.Sin(angle)*radius);
22                  GameObject go = Instantiate(_sampleCubePrefab, newPos, Quaternion.identity);
23              }
24      }
25
26
27      void Update()
28      {
29
30      }
31  }
```

Abbildung 15: Objekte in Kreistruktur anordnen

Die Start-Methode wird zu Beginn aufgerufen und enthält eine Schleife, die eine bestimmte Anzahl von Objekten mit dem Namen _sampleCubePrefab erzeugt. Jedes Objekt wird mit einem bestimmten Abstand zum Mittelpunkt platziert. Insgesamt wird dieser Code verwendet um Objekte in Unity3D kreisförmig anzuordnen.

4.4.2 AudioConnect

In Unity3D können die Audiodaten mithilfe der AudioSource-Komponente und der AudioListener-Komponente verarbeitet werden. Die AudioSource-Komponente dient als Quelle für die einzubindende Audiodatei, während die AudioListener-Komponente die Audiodatei empfängt. Mithilfe der AudioListener-Komponente können Frequenzspektrumdaten abgerufen werden, um die Übertragung der Audiodatei weiter auf die ansteuerbaren Objekte zu übertragen.

```
4
5   [RequireComponent (typeof(AudioSource))]
6   public class AudioConnect : MonoBehaviour
7   {
8       AudioSource _audioSource;
9
10      public static float[] _samples = new float[512];
11      public static float[] _freqBand = new float[8];
12
13      void Start()
14      {
15          _audioSource = GetComponent<AudioSource>();
16      }
17
18      void Update()
19      {
20          GetSpectrumAudioSource();
21          FrequencyBands();
22      }
23
24      void GetSpectrumAudioSource()
25      {
26          _audioSource.GetSpectrumData(_samples, 0, FFTWindow.Blackman);
27      }
28
```

Abbildung 16: Integration einer Audiodatei in Unity3D

Dieser Code in Abbildung 16 wird in Unity verwendet, um Audiodaten aus einer Audioquelle zu bekommen. Die Zeile [RequireComponent(typeof(AudioSource))] gibt an, dass das Objekt, das dieses Skript verwendet, eine AudioSource-Komponente haben kann. Das Skript enthält die Methode Start(), die die AudioSource-Komponente dem Objekt verleiht. Die Methode Update() wird in jedem Frame aufgerufen und ruft zwei andere Methoden auf. Die Methode GetSpectrumAudioSource() extrahiert die Audiodaten aus der AudioSource-Komponente mit Hilfe der Methode GetSpectrumData(). Es speichert die Daten in einem Array von 512 Einheiten mit dem Namen _samples. Die Methode FrequencyBands(), die in Abbildung 17 näher betrachtet wird, berechnet die Frequenzbänder der Audiodatei, indem sie das Spektrum des Audio-Samples in 8 Frequenzbänder

aufteilt. Es verwendet den _samples-Array und berechnet den Durchschnitt des Signals jedes Frequenzbandes. Die Ergebnisse werden im _freqBand-Array gespeichert (vgl. [11]).

```
29      void FrequencyBands()
30      {
31          int count = 0;
32
33          for (int i = 0; i < 8; i++){
34
35              float average = 0;
36              int sampleCount = (int)Mathf.Pow(2,i) * 2;
37
38              if (i == 7){
39                  sampleCount += 2;
40              }
41              for (int j = 0; j < sampleCount; j++){
42
43                  average += _samples[count] * (count + 1);
44                      count++;
45              }
46              average /= count;
47              _freqBand[i] = average * 10;
48          }
49      }
50  }
```

Abbildung 17: Zerlegung der Audiodatei in Samples

FFTWindow.Blackman ist Teil eines Arguments, das an eine Fast-Fourier-Transform (FFT)-Methode in Unity3D übergeben wird, um eine Blackman-Fensterfunktion auf das Audiosignal anzuwenden, bevor es analysiert wird. Die Blackman-Fensterfunktion ist eine von mehreren Methoden, die verwendet werden, um die Amplituden von Frequenzen in einem Signal zu analysieren. Die Anwendung dieser Funktion auf das Signal verringert die Auswirkungen von Signalverzerrungen an den Rändern des Signals und hilft, ein genaueres Frequenzspektrum zu erzeugen (vgl. [12, 13]).

4.4.3 ObjectMove

Der Code der Abbildung 18, definiert die Klasse ObjectMove. In diesem Skript gibt es eine Variable vom Typ GameObject mit dem Namen Poti, auf die später zugegriffen wird. Die Update() Methode ist der Hauptteil des Skripts und wird jeden Frame aufgerufen. In der Methode wird der Wert von yScale aktualisiert, indem er aus dem Array datas des GameObjects Poti entnommen wird, das in der DatenStream Klasse definiert wurde. Der Wert von yScale wird durch 100 geteilt, um ihn auf einen Wert zwischen 0 und 1 zu bringen. Anschließend wird die Skalierung des Objekts aktualisiert, indem der aktuelle Wert des Frequenzbands, das der Variable PotiN entspricht, dabei wird die Startskalierung _startScale mit 10 multipliziert. Die x- und z-Skalen werden durch xScale und zScale festgelegt.

```
4
5   public class ObjectMove : MonoBehaviour
6   {
7       public int _band;
8       public float xScale =0.1f;
9       public float zScale =0.1f;
10      public float _startScale, _scaleMultiplier;
11      public int PotiN;
12      public float yScale;
13      public GameObject Poti;
14
15      void Start()
16      {
17
18      }
19
20
21      void Update()
22      {
23          yScale = float.Parse(Poti.GetComponent<DatenStream>().datas[PotiN]);
24          yScale = yScale/100.0f;
25
26          transform.localScale =
27          new Vector3(transform.localScale.x, (AudioConnect._freqBand[PotiN]*10 + _startScale) * yScale, transform.localScale.z);
28      }
29  }
```

Abbildung 18: Skript für die Interaktion der Objekgruppen

In Verbindung mit den Skripten DatenStream, AudioConnect und ObjectMove können diese Daten verwendet werden, um visuelle Effekte zu erzeugen, die auf den Audioeingang reagieren. Das Skript CreateObjects erzeugt visuelle Objekte, während das Skript ObjectMove die Manipulation und Animation der Objekte steuert.

5 Validierung der Arduino-basierten Unity3D-Steuerung

Um die Funktionalität der Arduino-basierten Unity3D-Steuerung bewerten zu können, wird ein Test durchgeführt, der die Reaktionszeit der Übertragung misst und die übertragenden Daten gezeigt werden. Hierbei wird untersucht, wie genau die Übertragung ist, um die Anwendung effektiv steuern zu können.

5.1 Reaktionszeit und Datentransfer

Die Reaktionszeit ist ein wichtiger Faktor für eine Visualisierung in Echtzeit. Hierbei wird der zeitliche Aspekt der Steuerung in Bezug auf die Bewegung der Objekte untersucht und ob alle Daten korrekt an Unity3D übertragen werden. Die Daten sollten schnell genug übertragen werden, um eine verzögerungsfreie Interaktion zu ermöglichen.

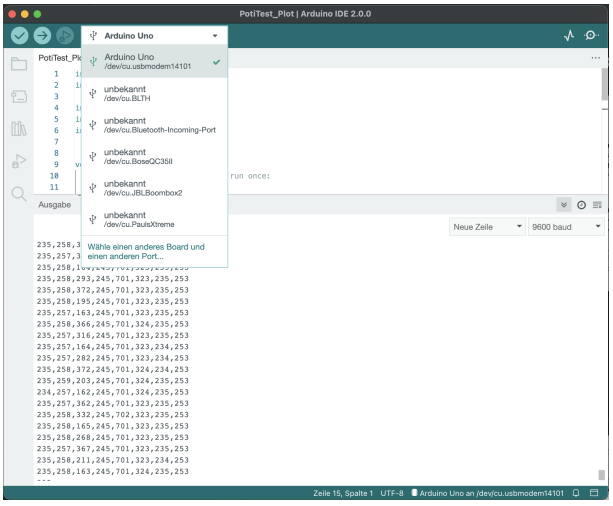

Abbildung 19: Potentiometerwerte im Serial Monitor der ArduinoIDE

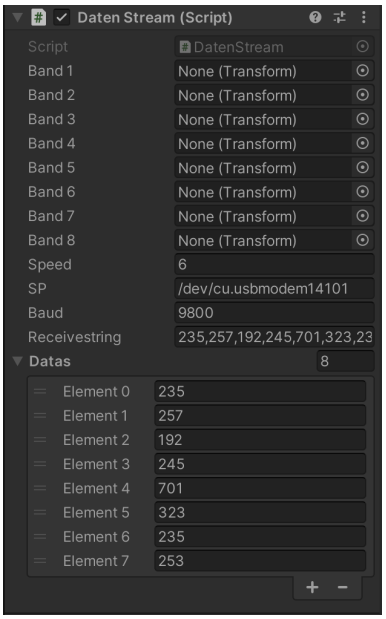

Abbildung 20: Datenstring in Unity3D

Um die Reaktionszeit zu testen, wird die Zeit zwischen Potentiometerabgriff und Interaktion der Objekte gemessen. Diese Reaktionsmessung wird durch starten und stoppen einer Uhr realisiert. Dabei wird der Start der Reaktionszeit mit der Betätigung eines Potentiometers festgelegt, dieser wird dabei von maximal Anschlag auf den minimal Anschlag zurückgedreht. Die Reaktionsmessung wird mit der vollständigen Eliminierung des visuellen Frequenzbandes beendet. Bei laufender Anwendung werden 5 Messungen im Abstand von 5 Sekunden durchgeführt. Dieser Test umfasst 3 Testdurchläufe die mit Beginn der Anwendung starten. In den Abbildungen 18 und 19 werden die Potentiometerwerte in der Arduino IDE und in Unity3D dargestellt, die zwischen den beiden Komponenten seriell übertragen werden.

5.2 Auswertung

Bei der Auswertung der 3 Messungen wurde zusätzlich noch der Durchschnitt aller Messungen berechnet und visualisiert. Dabei ist zu erkennen, dass mit zunehmender Zeit sich die Latenz der ankommenden Daten in Bezug auf die aktualisierte Animation erhöt. Die farbigen Graphen der Abbildung 20 zeigt die Verzögerung der Reaktion des betätigten Potentiometers. Anhand der Ergebnisse der Auswertung wird gezigt das die Anforderungen für den gängingen Betrieb im Eventbereich nicht ausreichen um eine Echtzeitübertragung von

Messung	Messphase1 in [s]	Messphase2 in [s]	Messphase3 in [s]	ΔT in [s]
1	1,12	1,60	0,91	1,21
2	1,38	1,23	1,33	1,31
3	1,30	2,45	1,41	1,72
4	2,86	3,70	2,93	3,613
5	6,24	4,55	4,97	5,253

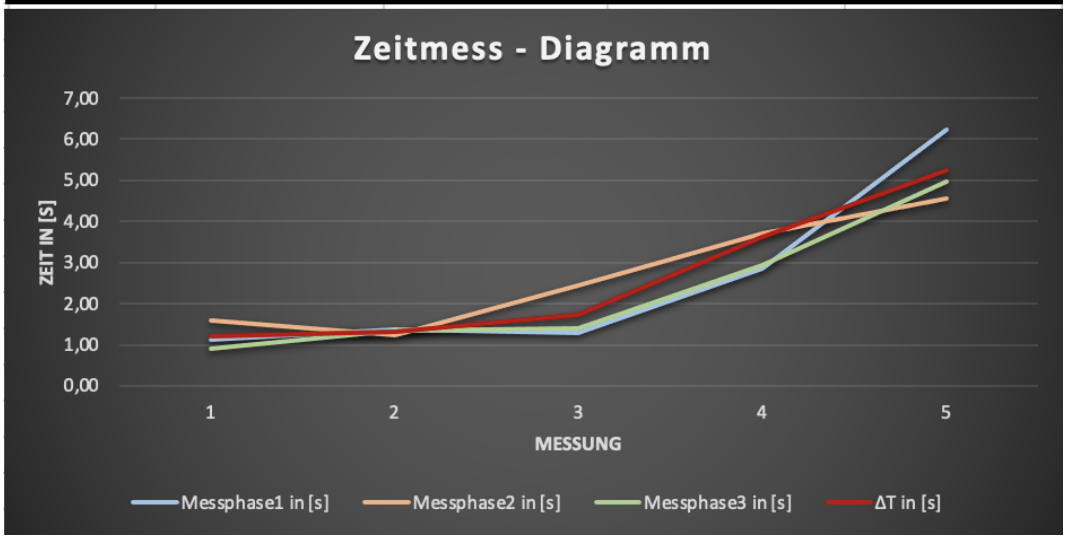

Abbildung 21: Zerlegung der Audiodatei in Samples

6 Diskussion und Ausblick

In diesem Kapitel wird die geschaffene Lösung des Controllers diskutiert und auf mögliche Einsatzgebiete dieser Technik eingegangen. Weiterhin werden mögliche Erweiterungen aufgezeigt um eine bessere Funktionalität zwischen technischer Umsetzung und visueller Anwendung zu ermöglichen.

6.1 Fazit der Umsetzung

Ziel der Arbeit war ein Controller zu entwerfen, der durch eine Arduino basierten Stuerung eine Unity3D Anwendung, in Form einer mit Audio interagierenden Visualisierung, modulieren soll. Durch die geschaffene Lösung mit Entwurf und Implementierung wurden Funktionalität des Controllers erfüllt. Die zeitliche Reaktion der ankommenden Potentiometerwerte auf verschiedene Objekte wurde mit einer Verzögerung der Zeit festgestellt und wäre für eine Anwendung im Echtzeitbetrieb für kommerzielle Veranstaltungen jedoch ungeeignet.

6.2 Einsatzgebiete und Erweiterung

Durch die Entwicklung eines einfachen Arduino basierten Controller wurde im Rahmen dieser Arbeit die Grundlage für eine modulierbare Anwendung geschaffen, die mithilfe der Programmiersprache C# und Unity3D realisiert werden konnte. Die Einsatzmöglichkeiten des Controllers kann mit einer besseren Echtzeitübertragung für kommerzielle Zwecke Anwendung finden. Im Rahmen dieser Arbeit wird Einsteigern die Möglichkeit geboten einen eigenen Controller zu entwickeln und weiter auszubauen. Darüber hinaus wird ein breites Spektrum ermöglicht, den Controller in verschiedenen Anwendungsgebiete einzusetzen. Die Verwendung in Virtual-Reality, Robotersetuerung und Unity3D Spielen wären Besipiele für weitere Umgebungen dieser Ausarbeitung.

7 Selbstständigkeitserklärung

Hiermit erklären wir, dass die Arbeit selbstständig verfasst, in gleicher oder ähnlicher Fassung noch nicht in einem anderen Studiengang als Prüfungsleistung vorgelegt wurde und keine anderen als die angegebenen Hilfsmittel [5] und Quellen, einschließlich der angegebenen oder beschriebenen Software, verwendet wurden.

Köthen den 1. März 2023
Max Pinkert

[5] Als Hilfsmittel wird die Website OpenAI für innovative Vorschläge der Struktur der Arbeit genutzt

Abbildungsverzeichnis

Literatur

[1] Philipp Jährling. Sensor- und bewegungsbasierte steuerung von multimedia-anwendung.

[2] Edis techlab: Arduino einfach erklärt. https://edistechlab.com/arduino_uno/?v=3a52f3c22ed6. letzter Aufruf: 09.08.2022.

[3] Unity: Get more with unity. https://unity.com/pages/more-than-an-engine. letzter Aufruf: 12.01.2023.

[4] Games und lyrik: Von spielen inspiriert. https://games-und-lyrik.de/was-sind-assets/. letzter Aufruf: 03.09.2022.

[5] Edis techlab: Multiplexer cd4051be einfach erklärt. https://edistechlab.com/mulitplexer-cd4051be-einfach-erklaert/?v=3a52f3c22ed6. letzter Aufruf: 10.08.2022.

[6] Midi: Musical instrument digital interface. https://de.wikipedia.org/wiki/MIDI. letzter Aufruf: 23.09.2022.

[7] Midi association: Official midi specifications. https://www.midi.org/specifications. letzter Aufruf: 23.09.2022.

[8] Ardity: Want to connect your arduino to unity over a com port? https://ardity.dwilches.com. letzter Aufruf: 18.12.2022.

[9] Microsoft: Serial port class. https://learn.microsoft.com/en-us/dotnet/api/system.io.ports.serialport?view=dotnet-plat-ext-7.0. letzter Aufruf: 03.11.2022.

[10] Unity insider forum: Objekte in unity per serial port über arduino steuern. https://forum.unity-community.de/topic/4943-objekte-in-unity-per-serial-port-ber-arduino-steuern/. letzter Aufruf: 09.11.2022.

[11] Audio visualization : Unity/c# tutorial [part 2 - getspectrumdata in unity]. https://www.youtube.com/watch?v=0KqwmcQqg0s&list=PL3POsQzaCw53p2tA6AWf7_AWgplskR0Vo&index=3. letzter Aufruf: 21.11.2022.

[12] How to choose fft window type. https://answers.unity.com/questions/1623827/how-to-choose-fft-window-type.html. letzter Aufruf: 21.11.2022).

[13] Audio visualization : Unity/c# tutorial [part 4 - eight frequency bands]. https://www.youtube.com/watch?v=mHk3ZiKNH48&list=PL3POsQzaCw53p2tA6AWf7_AWgplskR0Vo&index=6. letzter Aufruf: 21.11.2022.