Hochschule der Medien Stuttgart
Studiengang Audiovisuelle Medien
Bachelor of Engineering

Bau und Funktionsweise eines MIDI-Mergers auf Arduino-Basis

Schriftliche Ausarbeitung
im Rahmen der Lehrveranstaltung
„Ton-Seminar“
Wintersemester 2025/26

Vorgelegt von
Richard Jacob
Matrikel-Nr.: 45966
rj023@hdm-stuttgart.de

Betreuer: Prof. Oliver Curdt

Tübingen, 28. Februar 2026

Inhaltsverzeichnis

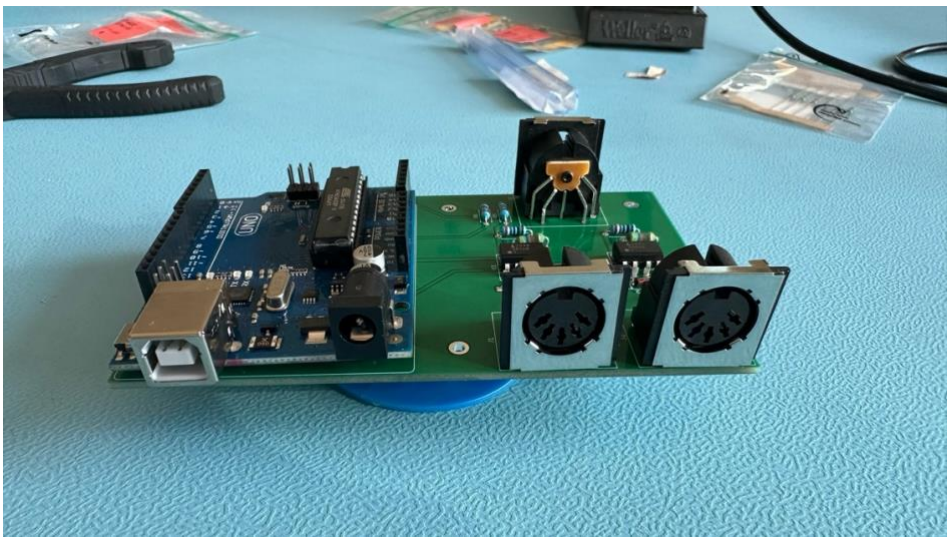

Abbildung 1: 2-IN-1-OUT MIDI-Merger

1. Theoretischer Hintergrund

1.1. MIDI – Historische Einordnung

MIDI (Musical Instrument Digital Interface) wurde Anfang der 1980er Jahre entwickelt, um ein damals zentrales Problem elektronischer Musikinstrumente zu lösen: Geräte verschiedener Hersteller waren technisch nicht miteinander kompatibel. Synthesizer, Drumcomputer oder Sampler nutzten jeweils proprietäre Steuerprotokolle, sodass ein Instrument von Yamaha nicht ohne Weiteres mit einem Gerät von Roland oder Sequential Circuits kommunizieren konnte.
Im Jahre 1982/83 wurde schließlich MIDI 1.0 von Dave Smith und Ikutaro Kakehashi vorgestellt [1]. Der Standard war offen, herstellerübergreifend und kostengünstig umsetzbar – drei Eigenschaften, die maßgeblich zu seiner raschen Verbreitung beitrugen. Innerhalb weniger Jahre entwickelte sich MIDI zum globalen Standard für musikalische Steuerdaten, Sequencing und Synchronisation. Auch heute, über vier Jahrzehnte später, basiert ein Großteil der Musikproduktion weiterhin auf MIDI 1.0. Erst 2020 wurde mit MIDI 2.0 eine Erweiterung veröffentlicht, die eine höhere Auflösung und bidirektionale Kommunikation ermöglicht, jedoch vollständig abwärtskompatibel bleibt [2]. Damit bleibt MIDI 1.0 weiterhin relevant und unverzichtbar.

1.2. MIDI – Technischer Überblick

MIDI überträgt keine Audiosignale, sondern rein digitale Steuersignale. Die Übertragung erfolgt seriell mit einer Baudrate von 31.250 Bit pro Sekunde nach dem sogenannten 8N1-Schema, also mit einem Startbit, acht Datenbits und einem Stoppbit [3]. Diese Struktur macht MIDI technisch einfach, gleichzeitig aber robust und kompatibel für eine enorme Bandbreite an Geräten.
Musikalische Informationen werden in Form von Befehlen (Messages) übertragen. Eine typische Note-On-Nachricht besteht aus drei Bytes: einem Statusbyte, das Nachrichtentyp und MIDI-Kanal definiert, dem Notenwert (0–127) und der Anschlagsstärke [3]. Neben Notenbefehlen existieren weitere Nachrichtentypen wie Control Change, Program Change oder Pitchbend sowie zeitkritische System-Real-Time-Nachrichten wie Clock und Sequenz-Start/ Sequenz-Stop.
Physikalisch erfolgt die Verbindung über eine fünfpolige DIN-Buchse, von der jedoch nur zwei Pins aktiv zur Datenübertragung genutzt werden [3]. Das erleichtert die Implementierung und trägt zur hohen Störsicherheit des Standards bei.

1.3. Optokoppler und galvanische Trennung

Eine Besonderheit des MIDI-Standards ist die vorgeschriebene galvanische Trennung am MIDI-Eingang. Diese wird durch einen Optokoppler realisiert, der das elektrische Signal optisch überträgt und dadurch sämtliche Masseverbindungen zwischen zwei Geräten verhindert [4]. Das schützt die angeschlossenen Geräte vor Brummschleifen und Spannungsschwankungen [4] und sorgt gleichzeitig dafür, dass jedes MIDI-Gerät unabhängig von der elektrischen Beschaffenheit des anderen funktioniert.
Der Optokoppler wandelt das einkommende elektrische MIDI-Signal zunächst in Licht um; ein Fototransistor übersetzt dieses Licht anschließend wieder zurück in elektrische Spannung. Diese Entkopplung ist ein wesentlicher Grund dafür, dass MIDI-Setups auch mit vielen Geräten ausgesprochen zuverlässig funktionieren.

1.4. Was ist ein MIDI–Merger?

Ein MIDI-Merger ist ein Gerät, das zwei oder mehr MIDI-Eingänge zu einem einzigen Ausgang zusammenführt. Dies klingt trivial, ist aber technisch anspruchsvoll, da MIDI ein serielles Protokoll ist: Nachrichten dürfen nicht durcheinandergeraten, Statusbytes müssen korrekt zugeordnet

bleiben, und zeitkritische Signale wie MIDI-Clock müssen ohne Verzögerung weitergegeben werden.
Ein Merger erlaubt es, mehrere Steuerquellen gleichzeitig zu nutzen – etwa einen Sequencer und ein Keyboard, obwohl das angeschlossene Gerät nur einen einzigen MIDI-Eingang besitzt. In komplexen Hardware-Setups oder bei Geräten mit eingeschränkten MIDI-Funktionen ist ein Merger daher essenziell, um flexible Arbeitsabläufe zu ermöglichen und redundantes Umstecken zu vermeiden.

2. Problemstellung, Motivation und Ziel

In vielen Hardware-Setups übernimmt nur eines der Geräte die Rolle des Sequencers, während andere Sampler oder Klangerzeuger ausschließlich über MIDI gesteuert werden können. So auch im zugrunde liegenden Szenario: Ein erster Sampler verfügt über Sequencer und Pads und kann selbstständig Noten senden, während ein zweiter Sampler keine eigene Möglichkeit zur Notenauslösung besitzt und vollständig auf MIDI-Eingaben angewiesen ist.
Unter normalen Bedingungen läuft die Verbindung zwischen beiden Geräten problemlos. Der erste Sampler sendet sowohl die MIDI-Clock als auch die Notenbefehle an den zweiten, der dadurch synchron betrieben werden kann. Kritisch wird die Situation jedoch dann, wenn der erste Sampler in den Aufnahmemodus versetzt wird. Viele ältere Geräte deaktivieren während des Aufnehmens ihren MIDI-Ausgang vollständig. In diesem Moment empfängt der zweite Sampler kein Clock-Signal und keine Noten mehr und kann nicht weiterspielen, obwohl er für den Produktionsprozess aktiv bleiben müsste.
Da der zweite Sampler nur einen einzigen MIDI-Eingang besitzt, sind einfache Alternativen wie das Umstecken des Kabels oder das Zwischenschalten einer Digital Audio Workstation (DAW) zwar denkbar, aber im Arbeitsalltag unpraktisch und störend. Besonders beim Resampling, bei dem Audio zwischen den Samplern hin- und hergeführt wird, führen solche Workarounds zu unnötigen Unterbrechungen und erschweren einen flüssigen Workflow.
Daraus ergibt sich die Notwendigkeit eines Geräts, das zwei unabhängige MIDI-Quellen gleichzeitig an einen einzigen MIDI-Eingang weitergeben kann, ohne manuelle Eingriffe am Setup zu erfordern. Ein solcher MIDI-Merger ermöglicht es, neben dem Hauptsequencer ein zweites Gerät – etwa ein MIDI-Keyboard – als zusätzliche Steuereinheit zu nutzen, während zeitkritische Informationen wie MIDI-Clock weiterhin korrekt übertragen werden.
Ziel des Projekts war daher die Entwicklung eines kompakten, USB-betriebenen 2-IN-1-OUT MIDI-Mergers, der die eingehenden Datenströme ohne Verzögerung zusammenführt und so den beschriebenen Workflow-Konflikt löst. Die Umsetzung auf Arduino-Basis bietet eine transparente, kostengünstige und erweiterbare Lösung, die sich besonders gut für Lern- und Experimentierzwecke eignet.

3. Hardware-Konzept und -Aufbau

3.1. Komponentenübersicht

Für den MIDI-Merger wurde ein überschaubarer und leicht nachvollziehbarer Hardwareaufbau gewählt, der ausschließlich auf gängigen Elektronikkomponenten basiert. Im Zentrum steht ein Arduino Uno. Durch seine Zuverlässigkeit, einfache Programmierbarkeit und umfangreiche Dokumentation ist er für Projekte dieser Art sehr gut geeignet.
Die beiden MIDI-Eingänge werden gemäß der offiziellen Spezifikation umgesetzt und jeweils über einen Optokoppler des Typs 6N138 [4] galvanisch getrennt. Diese Bauteile stellen sicher, dass der Arduino elektrisch nicht mit externen Geräten verbunden ist und schützen die Schaltung vor

Brummschleifen oder Spannungsschwankungen. Ergänzt werden sie durch 1-kΩ- und 220 -Ω-Widerstände, die die Stromstärken im LED- und Transistorzweig des Optokopplers begrenzen und so einen zuverlässigen Betrieb gewährleisten.
Als physische Anschlüsse dienen drei fünfpolige DIN-Buchsen, zwei für die MIDI-Eingänge und eine für den zusammengeführten MIDI-Ausgang. Die gesamte Schaltung wurde zunächst auf einem Steckbrett aufgebaut, um während der Entwicklung flexibel experimentieren und eventuelle Änderungen unkompliziert vornehmen zu können. Die Spannungsversorgung erfolgt über die 5-Volt-USB-Versorgung des Arduinos, wodurch keine zusätzliche Stromquelle notwendig ist. Nachdem der Aufbau erfolgreich getestet war, wurde auf Basis der Schaltpläne eine Leiterplatte entworfen, auf der alle Bauteile fest verlötet werden können.
Der verwendete Aufbau entspricht vollständig der MIDI-Norm und gewährleistet, dass zwei voneinander unabhängige Datenströme sauber eingelesen und später softwareseitig zusammengeführt werden können.

3.2. Funktionsweise

Die Schaltung folgt einem klaren und gut nachvollziehbaren Signalfluss, der sich in wenige aufeinander abgestimmte Verarbeitungsschritte gliedert. Die beiden MIDI-Signale gelangen zunächst über einen 220-Ω-Widerstand in die LED des jeweiligen Optokopplers [4]. Dieses Bauteil begrenzt den Strom und sorgt dafür, dass die LED stets innerhalb der spezifikationsgerechten Werte betrieben wird. Gleichzeitig stellt er sicher, dass nur normgerechte MIDI-Signale verarbeitet werden.
Im Inneren des Optokopplers wird das elektrische Eingangssignal in Licht umgewandelt, das wiederum einen Fototransistor ansteuert. Dieser erzeugt ein neues digitales Signal, das elektrisch vollständig vom ursprünglichen Gerät isoliert ist. Durch diese galvanische Trennung werden Masseprobleme und Störungen verhindert, wie sie in MIDI-Setups häufig entstehen können. Der Arduino erhält somit ein sauberes, störungsfreies Datensignal, ohne direkt mit dem Sender verbunden zu sein.
Das digital galvanisch getrennte Signal wird anschließend an die Eingänge des Arduinos weitergegeben. Verwendet werden hierbei die Pins D2 und D3, die hardwareseitig Interrupt-fähig sind. Diese Eigenschaft ist entscheidend, da sie dem Mikrocontroller ermöglicht, auf eingehende Bits unmittelbar zu reagieren und den seriellen Datenstrom mit der MIDI-typischen Geschwindigkeit von 31.250 Baud zuverlässig einzulesen.
Im Arduino werden die beiden Signale parallel verarbeitet. Das bedeutet, dass fortlaufend überprüft wird, ob an einem der beiden Eingänge ein neues Byte eingetroffen ist. Sobald dies geschieht, wird es ohne Verzögerung an den Ausgang weitergeleitet. Der MIDI-Merger arbeitet damit nach einem transparenten Durchschleifprinzip: Er analysiert die Nachrichten nicht inhaltlich, sondern sorgt dafür, dass alle eingehenden Bytes unverändert in den zusammengeführten Ausgangsstrom gelangen. Dieses Vorgehen ist besonders vorteilhaft für zeitkritische MIDI-Daten wie Clock-Signale, da sie ohne zusätzliche Verarbeitungsschritte sofort weitergegeben werden.
Die Ausgabe des zusammengeführten Datenstroms erfolgt über die UART-Schnittstelle des Arduinos, die auf die für MIDI vorgeschriebene Baudrate eingestellt ist [5]. Bevor das Signal die MIDI-Ausgangs-Buchse erreicht, durchläuft es erneut einen 220-Ω-Widerstand, der den Ausgang gemäß Spezifikation schützt [4]. Auf diese Weise wird sichergestellt, dass das ausgegebene Signal vollständig kompatibel zu allen MIDI-Geräten bleibt und eine Strombegrenzung im Falle eines Kurzschlusses vorliegt.
Insgesamt entsteht ein stabiler und störungsfreier Signalfluss: Die eingehenden Daten werden empfangen, optisch isoliert, im Arduino zusammengeführt und schließlich normgerecht wieder ausgegeben. Trotz des überschaubaren Aufwands entsteht so ein funktionaler und stabiler MIDI-Merger, der sich problemlos in bestehende musikalische Arbeitsumgebungen integrieren lässt.
Der Hardwareaufbau lässt sich somit als eine klare Folge von Schritten beschreiben:

Empfangen → galvanisch Trennen → Einlesen → Zusammenführen → Ausgeben

4. KiCad

Zur Planung und Umsetzung des Hardware-Aufbaus wurde die Open-Source-Software KiCad [1] verwendet. Sie bietet Werkzeuge zum Entwerfen elektronischer Schaltpläne, zum Erstellen von Leiterplattenlayouts sowie zur Generierung realistischer 3D-Visualisierungen. Auf diese Weise lässt sich der Übergang vom experimentellen Steckbrett-Aufbau zu einer dauerhaft nutzbaren und sauber gefertigten Platine präzise planen.
Im Folgenden werden der Schaltplan, das Leiterplattenlayout und das daraus generierte 3D-Modell des MIDI-Mergers vorgestellt.

4.1. Schaltplan

In Abbildung 2 ist der in KiCad erstellte Schaltplan des MIDI-Merges dargestellt. Er zeigt die beiden MIDI-Eingänge, die über die Optokoppler galvanisch getrennt werden, die zugehörigen Widerstände, die Verschaltung des Arduinos sowie die MIDI-Ausgangsstufe. Die gesamte Schaltung wurde gemäß den Spezifikationen des MIDI-Standards ausgelegt.

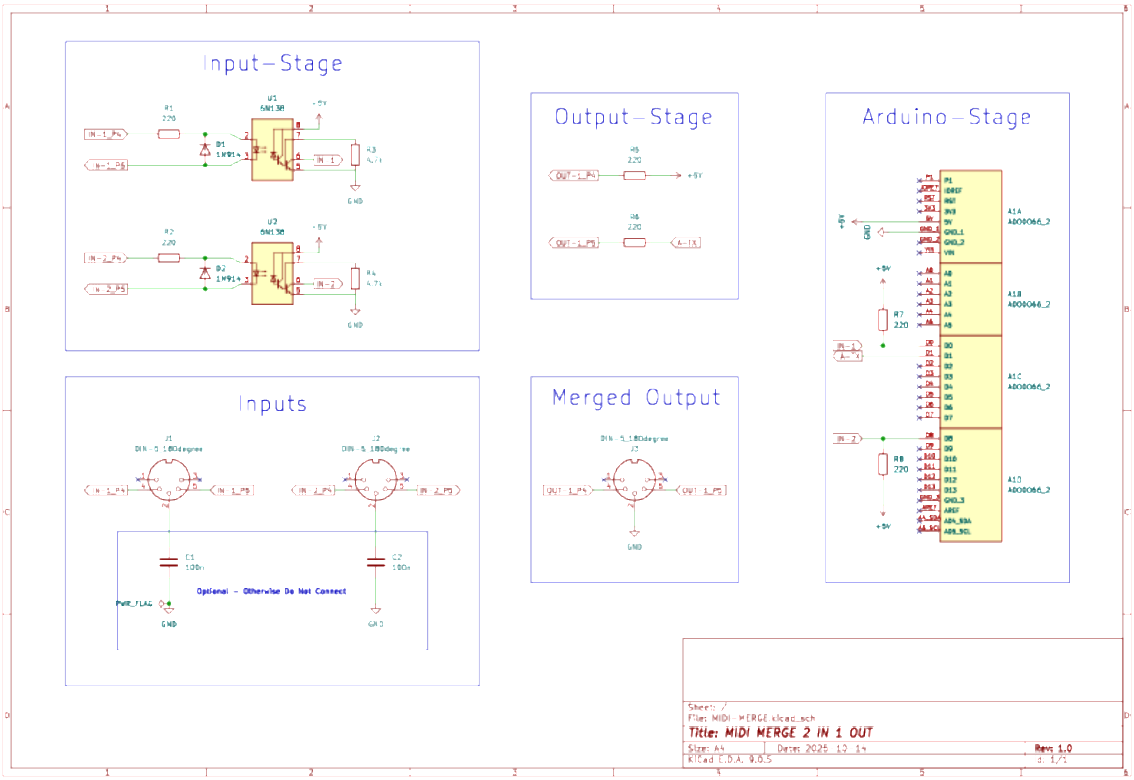

Abbildung 2: Schaltplan des MIDI-Mergers

4.2. Leiterplattenlayout

In Abbildung 3 ist das Leiterplattenlayout abgebildet. Erkennbar ist die Anordnung der Bauteile sowie die Platzierung der Leiterbahnen zur physikalischen Verbindung einzelner Pins und Komponenten.

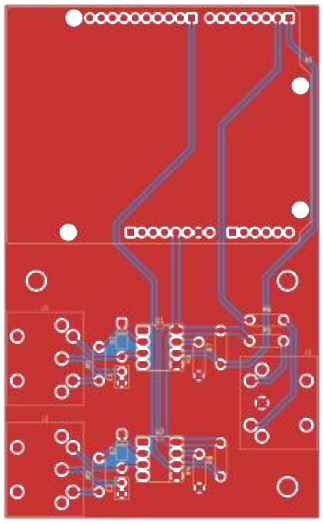

Abbildung 3: Leiterplattenlayout

4.3. 3D–Modell

In Abbildung 4 ist ein 3D-Modell der Platine einschließlich der elektrischen Bauteile dargestellt. Es veranschaulicht die räumliche Anordnung des Arduino-Boards, der DIN-5-Buchsen, der Optokoppler und weiterer elektronischer Komponenten. Zudem ermöglicht das 3D-Modell die Überprüfung des Leiterplattendesigns auf fehlerhafte Anordnung der Komponenten.

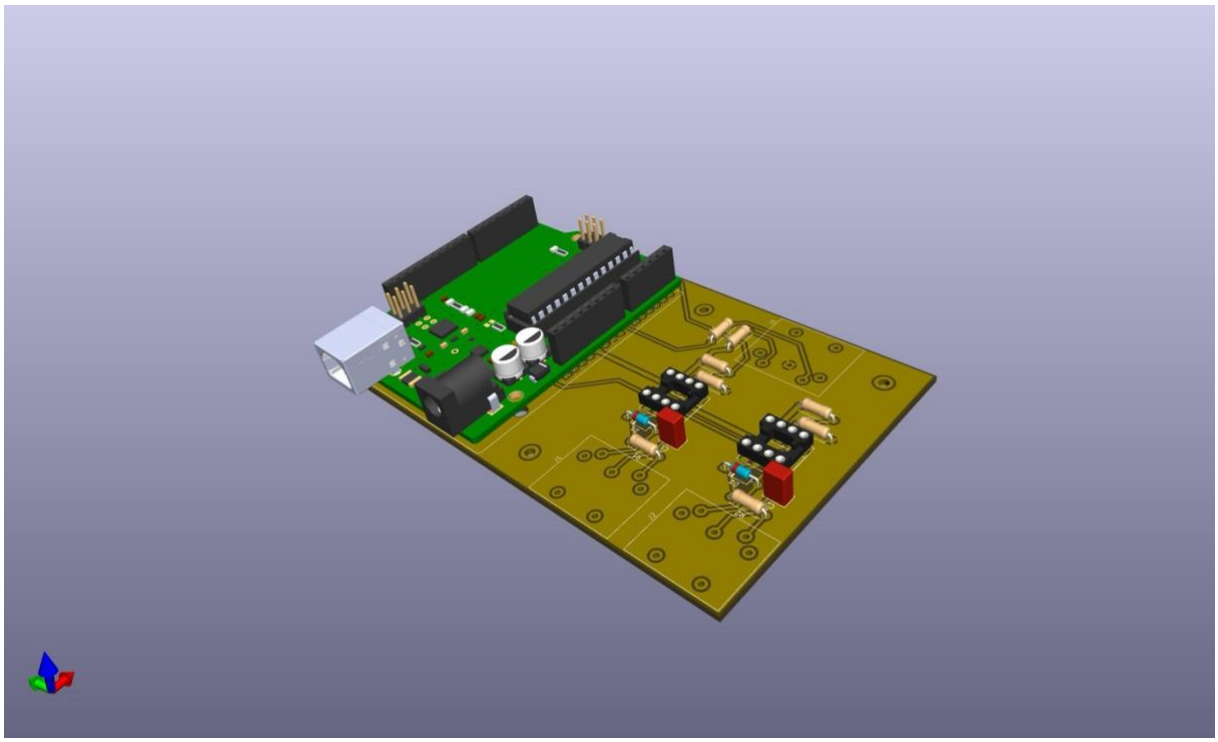

Abbildung 4: 3D-Modell der gesamten Platine inklusive Bauteile

5. Software und Funktionsprinzip

Die Software bildet das funktionale Zentrum des MIDI-Mergers. Während die Hardware den normgerechten Empfang und die elektrische Entkopplung übernimmt, sorgt die Software dafür,

dass beide MIDI-Datenströme zuverlässig ausgelesen und unmittelbar zu einem gemeinsamen Ausgangssignal zusammengeführt werden. Entwickelt wurde diese in der Arduino IDE [2], die eine direkte Anbindung an die Hardwarefunktionen des Mikrocontrollers ermöglicht.
Die Grundidee der Software besteht darin, möglichst wenig in das MIDI-Signal einzugreifen. Der Merger arbeitet vollständig nach dem Prinzip des transparenten Durchschleifens: Jedes eingehende Byte wird sofort erkannt und unverändert weitergegeben. Diese Strategie hält den Code übersichtlich und sorgt für eine niedrige Latenz.

5.1. Einlesen der MIDI-Datenströme

Der erste MIDI-Eingang wird über die integrierte Hardware-UART (Universal Asynchronous Receiver Transmitter) des Arduinos verarbeitet, die für kontinuierliche serielle Datenübertragung ausgelegt ist. Der zweite Eingang nutzt die Software-Serial-Bibliothek, die eine zusätzliche, softwarebasierte serielle Schnittstelle bereitstellt. Dadurch kann der Arduino Uno trotz nur einer physischen UART-Schnittstelle ein weiteres serielles Signal einlesen, wie zum Beispiel zwei unabhängige MIDI-Signale. Beide Schnittstellen arbeiten mit der MIDI-typischen Baudrate von 31.250 Bit/s.
In der Hauptschleife wird fortlaufend überprüft, ob neue Bytes eingetroffen sind. Sobald dies geschieht, werden sie direkt an den Ausgang weitergeleitet. Ein kurzer Ausschnitt des Codes verdeutlicht das Prinzip:

```
void loop() {
  // --- Eingang 1 lesen ---
  if (Serial.available()) {
    uint8_t data = Serial.read();
    Serial.write(data);        // an TX1 weiterleiten (MIDI-Out)
  }
```

Abbildung 5: Weiterleiten der Bytes von Eingang 1 zum Ausgang

Auf diese Weise reagiert der Merger schnell und behandelt beide Eingänge gleichberechtigt.

5.2. Zusammenführen der Daten

Das Zusammenführen erfolgt ohne zusätzliche Puffer oder interpretierende Routinen. Da MIDI ein serielles Protokoll ist, lassen sich die meisten Nachrichten unabhängig voneinander übertragen. Dadurch können eingehende Bytes direkt in den gemeinsamen Datenstrom übernommen werden. Der Merger verändert sie nicht und fügt keine Verzögerungen hinzu, was besonders in Hardware-Setups von Vorteil ist.

5.3. Echtzeitfähigkeit

Zeitkritische MIDI-Nachrichten wie Clock- oder Start-/Stop-Signale bestehen aus einzelnen Bytes und müssen ohne Verzögerung weitergeleitet werden. Durch das unmittelbare Weiterleiten jedes empfangenen Bytes bleiben diese Nachrichten erhalten und gelangen in Echtzeit zum Ausgang. Die Synchronität zwischen den Geräten bleibt dadurch stabil.

5.4. Vorteile und Grenzen

Der minimalistische Ansatz bringt mehrere Vorteile mit sich: geringe Latenz, hohe Stabilität, niedrige Systemlast und universelle Kompatibilität. Gleichzeitig verzichtet die Software bewusst auf komplexere Funktionen wie Running-Status-Handling, Filterung oder Priorisierung. Für den praktischen Einsatz im beschriebenen Workflow – das parallele Auslösen und Resampling – sind diese zusätzlichen Funktionen nicht erforderlich. Der Merger erfüllt in dieser Form zuverlässig die gestellte Aufgabe.

5.5. Der vollständige Code in Arduino IDE

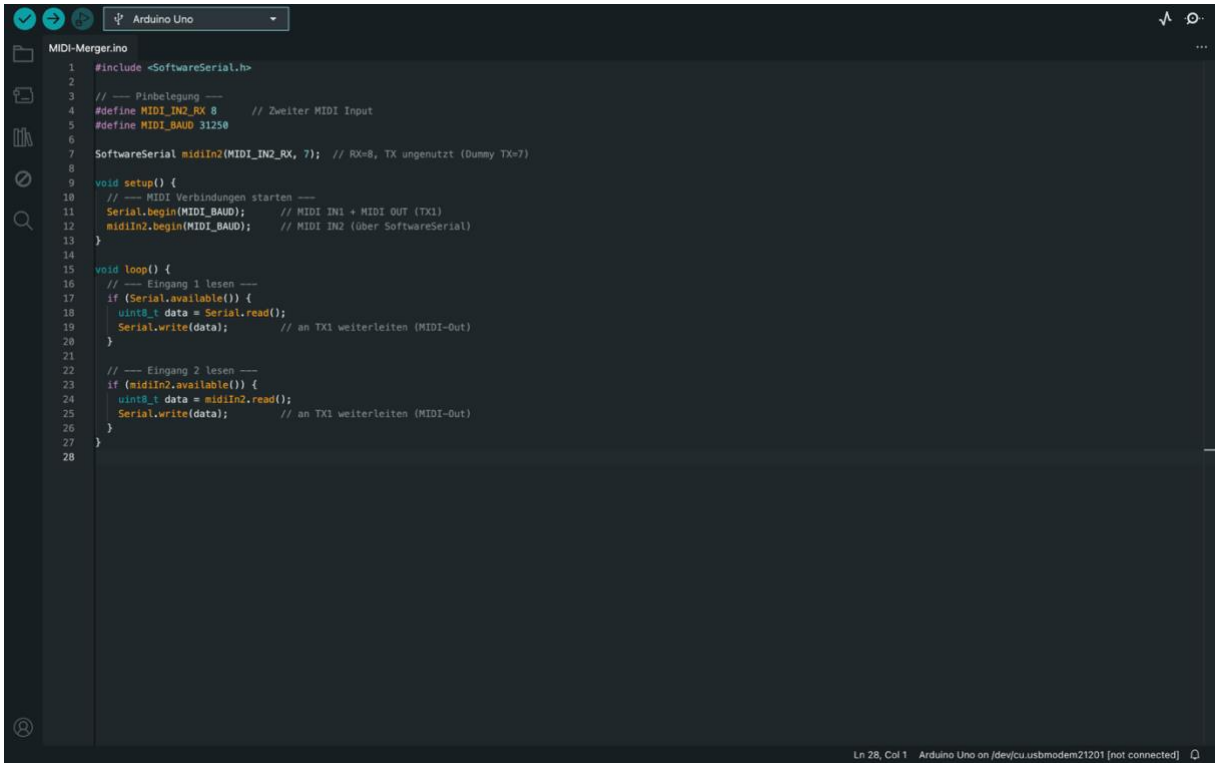

Abbildung 6: Der vollständige Code in Arduino IDE

6. Reflexion

6.1. Was gut funktioniert hat

Im Verlauf des Projekts hat sich gezeigt, dass die grundlegende technische Umsetzung zuverlässig arbeitet. Die MIDI-Eingänge mit den Optokopplern funktionieren stabil und erfüllen die Anforderungen der MIDI-Spezifikationen ohne Probleme. Auch der Arduino konnte beide Datenströme gleichzeitig einlesen und weiterleiten, selbst wenn viele Nachrichten in kurzer Zeit eintrafen.

Der Aufbau auf dem Steckbrett erwies sich als besonders praktisch. Er machte es leicht, einzelne Bereiche der Schaltung zu testen, Fehler zu finden und Anpassungen vorzunehmen, ohne jedes Mal löten zu müssen. Dadurch blieb der Entwicklungsprozess flexibel und gut nachvollziehbar.
Insgesamt ließen sich alle zentralen Funktionen wie geplant realisieren, und das System arbeitete im Testbetrieb verlässlich.

6.2. Was weniger gut funktioniert hat

Trotz des funktionierenden Gesamtergebnisses traten im Laufe des Projekts einige Schwierigkeiten auf, die vor allem mit der praktischen Umsetzung zusammenhingen. Die Kontakte des Steckbretts waren zum Teil unzuverlässig, insbesondere die Masseverbindungen. Dadurch entstanden Fehler, die nur schwer nachzuvollziehen waren und teilweise erst nach längerem Suchen identifiziert werden konnten.

Hinzu kamen Probleme bei der Verbindung zwischen Computer und Arduino, da einige USB-Adapter oder Kabel nicht zuverlässig erkannt wurden, wodurch der Upload des Codes nicht möglich war oder das Debugging nicht genutzt werden konnte.
Ein weiterer Stolperstein war die Pinbelegung der Optokoppler. Da diese von Hersteller zu Hersteller variieren können. Dies führte zu anfänglicher Verwirrung, bis ein erneuter Blick in das Datenblatt für Aufklärung sorgte. Auch das MIDI-Protokoll selbst zeigte, wie sensibel dieses auf fehlerhafte Signale reagiert. Dadurch wurde deutlich, wie wichtig ein sauberer Aufbau und eine stabile Softwarestruktur sind.
Insgesamt lagen die meisten Herausforderungen nicht im Konzept, sondern in den kleinen aber typischen Problemen, die bei experimentellen Hardwareaufbauten auftreten können. Sie waren jedoch hilfreich, um ein besseres Gefühl für die Praxis elektronischer Schaltungen zu erlangen.

7. Nachwort

Das hier dokumentierte Projekt ist ursprünglich nicht im Rahmen des „Ton Seminar“ entstanden, sondern aus einer praktischen Fragestellung im Kontext eines musikalischen Setups: Wie lässt sich ein ergonomischer und effizienter Workflow zwischen zwei Hardwaresamplern erreichen, ohne wiederholt Kabel umstecken zu müssen oder Arbeitsabläufe zu unterbrechen?
Für das Seminar selbst erwies sich das Projekt als Glücksfall: Es verband technische, musikalische und praktische Aspekte und bot damit einen guten Rahmen, um sowohl den MIDI-Standard als auch den Entwicklungsprozess von Prototypen, Debugging und Leiterplattenentwicklung greifbar zu machen.

Falls Sie Fragen zu diesem Projekt, zu den Unterlagen oder zum Nachbau haben, stehe ich gerne zur Verfügung.
Sie erreichen mich unter meiner studentischen E-Mail-Adresse.

Abbildungsverzeichnis

Literaturverzeichnis

[1] The MIDI Association. (n.d.). The history of MIDI. Abgerufen am 28. Februar 2026, von https://midi.org/midi-history-chapter-6-midi-begins-1981-1983
[2] The MIDI Association. (2020). MIDI 2.0 overview. Abgerufen am 28. Februar 2026, von https://midi.org/midi-2-0
[3] The MIDI Association. (1996). MIDI 1.0 detailed specification (rev. 1996). Abgerufen am 28. Februar 2026, von https://midi.org/midi-1-0-detailed-specification
[4] The MIDI Association. (2014). 5-pin DIN electrical specifications. Abgerufen am 28. Februar 2026, von https://midi.org/5-pin-din-electrical-specs
[5] The MIDI Association. (n.d.). Arduino MIDI output basics. Abgerufen am 28. Februar 2026, von https://midi.org/arduino-midi-output-basics
[6] KiCAD (8.0) [Software]. (2024). Jean-Pierre Charras. [Stand 12. Dezember 2025] Verfügbar unter: https://www.kicad.org/
[7] Arduino IDE (2.3.8) [Software]. (2026). Jean-Pierre Charras. [Stand 27. Februar 2026] Verfügbar unter: https://www.arduino.cc/en/software/