CSUF

COLLEGE OF

Engineering and Computer Science

AudioLink

Computer Engineering Program

Electrical & Computer Engineering (ECE) Department

Team: Corey Hoang, Rolando Aguirre, Jonathan Ditloff

Team Number: 3

Faculty Advisor: Dr. Faller

Spring 2025

TABLE OF CONTENTS

I. Abstract

In collaborative music settings, managing multiple electronic instruments such as keyboards, guitars, and beat pads presents challenges in volume control, sound effect processing, and spatial sound placement. Traditional solutions such as external mixers and Digital Audio Workstations (DAWs) require complex setups, external power, and often the assistance of a sound engineer. These systems are not optimized for portability or intuitive real-time use.

This project presents AudioLink, a portable audio interface designed to enable up to four musicians to collaboratively perform in real time. The system supports two USB inputs for MIDI instruments and two mono jacks for electric guitars, with each input routed to its own channel featuring independent volume, gain, and audio effects control. A horizontally mounted touchscreen interface allows intuitive real-time adjustment, and all audio signals are routed to a single 3.5mm output for use with any speaker or amplifier.

AudioLink aims to lower the barrier to live musical collaboration by consolidating key audio features into a compact and affordable device. The system prioritizes low-latency audio performance, safe output levels, and intuitive user interaction, all while adhering to ethical design standards and promoting inclusive access to live sound mixing technology.

II. Introduction

In today's world, it is almost impossible to escape from the ever-present sensation of music as we know it. It has been an integral part of human culture since the dawn of time, and has been known to bring people together: musicians and listeners alike. Musicians, in particular, however, have the innate need to create through collaboration and experimentation. Whether it's in a band setting or a simple session between musicians, the drive for cooperative creativity is ever so apparent.

However, live collaborative music settings also come with the challenges of volume control, sound effect mixing, sound placement, and overall variability of sound. This is especially true when it comes to electronic instruments such as keyboards, guitars, and beat pads. As Gibson explains in *The Art of Mixing,* live audio involves not only sound production, but also spatial design, where each instrument must be carefully positioned in a 3D auditory field through volume, EQ, and effects [1]. This spatial balance is usually achieved through traditional setups that require external mixers or Digital Audio Workstations (DAWS), which are software-based environments for audio recording and mixing. However, these methods are not specifically designed for portability, real-time user control, or operation without a dedicated sound engineer.

AudioLink integrates these functions into a single device, enabling real-time collaborative mixing in a portable form factor. It also simplifies a key aspect of the 3D auditory design process that's usually difficult to achieve in a live collaborative music setting: outputting all instruments' audio to a single, unified source. This device offers two inputs for keyboards or beat pads, and two inputs for electric or bass guitars, allowing up to four musicians to interface simultaneously. Each input is assigned to an independent channel with control over volume, gain, and multiple sound effects and samples for each instrument, and is real-time adjustable via a horizontally mounted touchscreen on the top of the device. All

channels are routed to a single 3.5mm audio jack, where the audio output is combined for use with any speaker or amplifier of the user's choice. This system allows for a convenient way for multiple musicians to express themselves collaboratively and creatively in real time, and removes the need for extra guitar pedals, guitar amplifiers, or separate instrument speakers.

This system focuses specifically on hardware integration, low-latency digital signal processing, and user interface design for real-time use. It does not include multitrack recording, wireless connectivity, Bluetooth capability, or advanced DAW-style editing capabilities. Audiolink is designed for co-located live collaboration, such as rehearsals, jam sessions, or small performances. It also assumes that users will provide their own instruments and speakers and have access to a power outlet. Limitations include the finite number of inputs, dependence on compatible USB/MIDI devices, and the absence of battery power (requiring an external power supply). These constraints were intentionally chosen to maintain portability and ensure the system remains intuitive, affordable, and accessible to musicians in casual or mobile settings.

As the engineers of Audiolink, it was important to consider and analyze ethical and professional responsibilities throughout the design process. Special attention was given to minimizing latency and audio distortion to ensure accurate real-time feedback for musicians, which is a critical requirement in live performance environments where timing and clarity directly affect user experience. The system includes a user-friendly touchscreen interface to support accessibility for users with different technical backgrounds. By consolidating multiple audio functions into a single device, AudioLink reduces the need for expensive, bulky external equipment, which can otherwise pose financial or logistical challenges. All input and output levels are limited to safe operating ranges to help prevent hearing damage. In addition, the design process followed best engineering practices, including signal integrity testing, modular design, and responsible component selection, to ensure that the device performs reliably and safely in real-world use cases.

The impact of this device lies in its potential to bring people and musicians closer together by facilitating accessible, real-time musical collaboration. In a world where music often serves as a universal language, AudioLink lowers the barrier to live group performance by eliminating the need for costly, complex, and bulky equipment. This makes collaborative music-making more attainable for schools, community centers, independent artists, and performers in resource-limited environments. By simplifying sound management and spatial mixing, the device promotes creativity and shared expression in both professional and casual settings. On a broader scale, AudioLink supports the cultural and emotional value of music as a unifying force across different communities, while also encouraging hands-on engagement with music technology.

III. Methodology

Designing a real-time audio interface with mobile capability requires a clear understanding of the functional needs of musicians in live collaborative settings. To be effective, the system must prioritize low latency, signal clarity, and a compact, portable form factor, in that order. These design priorities informed both the hardware selection and the software architecture of the system described in the following sections.

AudioLink utilizes four Teensy 4.1s, one per each instrument channel, to handle the main workload that a real-time audio processing system requires. This is a microcontroller that's equipped with a 600MHz ARM Cortex-M7 processor, Digital Signal Processing (DSP) capabilities, Audio Libraries, Direct Memory Access (DMA) support, USB host capability, 1MB RAM, 8MB flash memory, and an add-on Teensy audio shield.

The 600 MHz ARM Cortex-M7 processor delivers the computational performance necessary for executing tasks such as filtering, mixing, equalization, and applying time-domain effects with minimal latency. The processor includes dedicated Digital Signal Processing (DSP) extensions, enabling fast and efficient execution of operations like multiply-accumulate and fixed-point arithmetic. These are critical for real-time filter calculations and modulation effects. In addition, the Teensy Audio Library provides a modular audio framework that simplifies the implementation of streaming audio systems. It supports CD-quality (44.1 kHz) audio, handles buffer management, signal routing, and real-time scheduling, allowing developers to construct low-latency audio pipelines with ease [2]. The system also leverages Direct Memory Access (DMA), which allows audio data to move between peripherals and RAM without CPU intervention. This reduces system overhead and ensures continuous audio streaming without glitches. Each Teensy 4.1 is also equipped with USB host capability, which is essential for supporting standard USB MIDI controllers such as keyboards and beat pads. USB host mode enables it to communicate directly with external

MIDI instruments by initiating and managing data transfers. This functionality allows each Teensy to receive real-time MIDI note and control messages without needing a software-based DAW. The Teensy's USB host support is compatible with the USBHost_t36 library, which enables seamless integration with a wide variety of class-compliant MIDI devices. This makes the system highly modular and flexible for real-world use, allowing musicians to plug in and begin performing immediately without reconfiguration or special drivers. The 1 MB of RAM enables temporary storage for audio buffers, intermediate processing, and polyphonic handling. Meanwhile, the 8 MB of onboard flash memory provides ample space for storing DSP code, waveform tables, presets, and MIDI drum sample data, all without encroaching on runtime performance. Additionally, a Teensy Audio Shield is soldered to the pins of the microcontroller, which integrates an SGTL5000 audio codec. This codec handles the conversion between analog and digital audio signals at 16-bit, 44.1 kHz resolution, ensuring high-fidelity I/O. It connects to the Teensy via the I²S protocol, allowing for synchronized, low-latency audio streaming. The shield also provides line-in and headphone/line-out jacks, simplifying integration with electric instruments and playback systems. Together, these features enable AudioLink to perform reliable, low-latency, high-quality audio processing suitable for live musical collaboration.

To handle real-time audio effect change requests and manage high-level system coordination, a Raspberry Pi 5 was implemented as the central controller for all input channels. This component was chosen for its ability to run a full Linux operating system, support high-resolution touchscreens, and handle multiple concurrent threads, making it ideal for managing user input, graphical control interfaces, and inter-device communication. While the Teensys handle time-critical audio processing, the Raspberry Pi oversees soft real-time control tasks, such as routing volume adjustments, toggling effects, and switching presets.

Communication between the Raspberry Pi and each Teensy is handled over a Controller Area Network (CAN bus). This protocol was selected for its low latency, high reliability, and noise-resistant differential signaling, making it well-suited for real-time embedded control environments. Each Teensy node is equipped with a module that listens for CAN messages corresponding to its assigned channel, ensuring a timely and deterministic response to user interactions initiated from the Raspberry Pi's touchscreen interface [3].

The graphical user interface (GUI) for AudioLink was developed using Python and the PyQt5 framework, optimized for launch on an official Raspberry Pi 7-inch touchscreen. The GUI acts as the centralized control platform for four audio channels, two inputs for keyboards or beat pads, and two inputs for electric or bass guitars, each handled by a Teensy 4.1 microcontroller. A tab-based layout allows users to select between MIDI1, MIDI2, Guitar1, and Guitar2 using a navigation bar implemented at the top of the screen with QStackedWidget and custom-styled buttons.

Within the MIDI1 and MIDI2 tabs, users have access to rotary knobs for adjusting reverb, delay, and color parameters, along with master volume sliders and waveform selection buttons (**Figure 1).** These controls are methodically grouped and styled for high visibility and touchscreen interaction. The layout emphasizes clean sectioning and real-time feedback for each parameter. MIDI instrument channels support audio effects tailored to synthesized input and allow users to shape the character of their keyboard or pad sound with precision.

Figure 1. GUI layout of the MIDI tab. Reverb, delay, volume, and waveform options are presented in intuitive, touch-friendly sections.

The Guitar1 and Guitar2 tabs are optimized for electric instruments and provide additional DSP control beyond basic reverb and delay. Each guitar channel features toggled on, these sections reveal multiple rotary dials for adjusting parameters such as BPM, delay feedback, reverb time, modulation rate, bitcrusher depth, and more. Effects are visually separated into labeled boxes, with color-coded toggle states for quick identification during live use. A sample layout of the Guitar1 tab is shown in **Figure 2**.

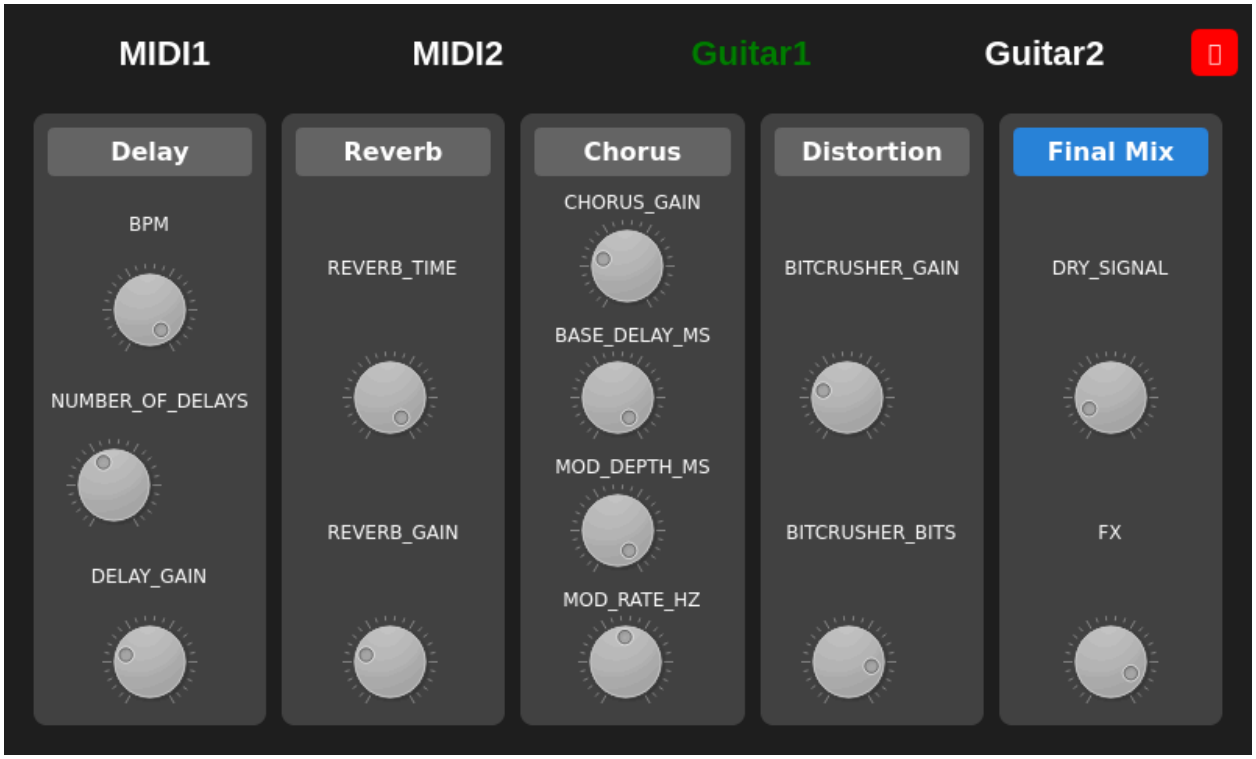

Figure 2. GUI layout of the Guitar1 tab. Includes toggled effect sections and dynamic control over advanced DSP parameters.

A core feature of the GUI is its MIDI pad mapping system, which allows users to assign specific drum sounds (e.g., Kick, Snare, Hat, Clap, Percussion) to individual MIDI notes. Each pad can be tapped to open an assignment dialog. When the user plays a note on their MIDI keyboard, the system detects the note via a background listener thread and displays it on screen. The user can then choose a drum type and select a specific sample from a scrollable list. This multi-step interaction ensures precise control over both mapping and sample selection. **Figures 3 and 4** illustrate the second and third windows involved in the MIDI pad assignment process.

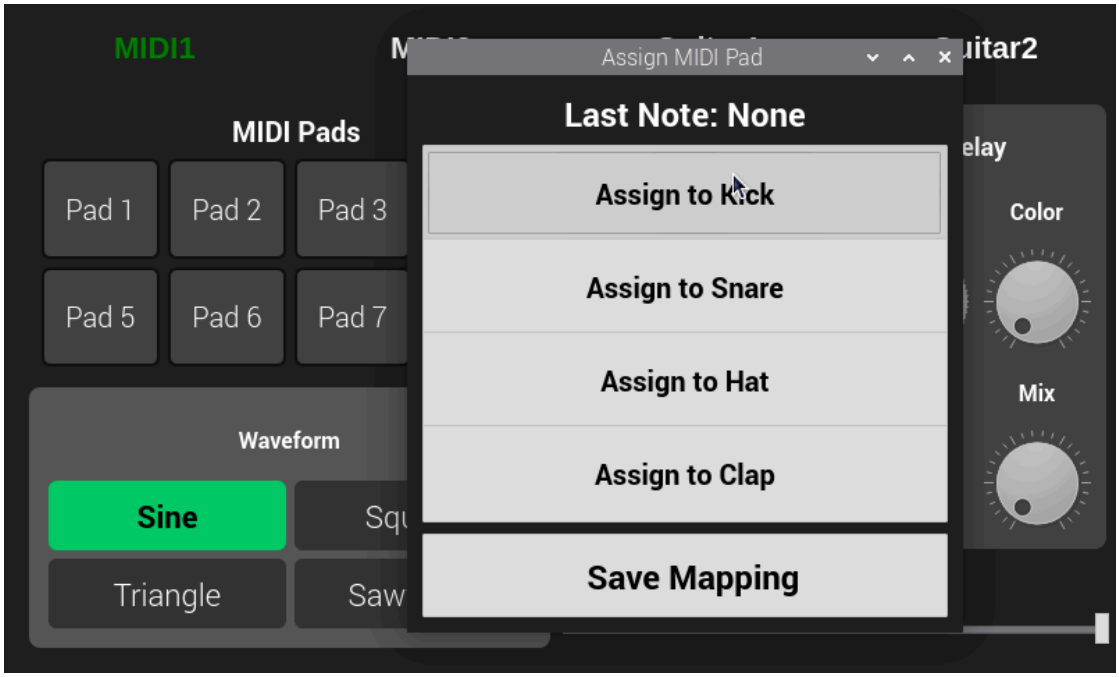

Figure 3. The pad mapping dialog displays the last received MIDI note and drum type selection options.

Figure 4. A scrollable sample selector is used to choose a specific sound within the selected drum type (e.g., kick).

All user interactions are linked to CAN bus communication with the Teensy 4.1 microcontrollers. Each GUI action, such as turning a dial or toggling an effect, triggers a signal that sends a CAN message using the Python-Can library. These messages include a target Teensy ID (e.g., 0x100 to 0x400), a parameter ID, and a float32 value representing the control change. Each Teensy listens for messages on its assigned ID and updates the relevant audio processing pipeline in real time. The CAN interface is initialized on GUI launch via Linux commands and shut down gracefully when the application exits.

To avoid blocking the main event loop, incoming MIDI messages are handled in a separate thread using PyQt5's QThread class. This multithreaded structure ensures that pad mapping and real-time adjustments remain responsive and do not interfere with GUI rendering or user input. Overall, the GUI integrates robust CAN communication, touchscreen-optimized layouts, and real-time feedback into a responsive system that empowers musicians to interactively control complex audio processing chains during performance.

The power architecture of AudioLink is based on a regulated 13.8V, 25A Powerwerx DC supply, which serves as the primary voltage source for both analog and digital subsystems. Power is first routed to a high-current bus bar, which distributes the 13.8V rail directly to the two preamps and the audio mixer, all of which require a higher voltage than the rest of the system in order to properly run. The same bus bar also feeds into a DROK DC buck converter, which steps the voltage down to a regulated 5V output at up to 20A. This 5V rail is then distributed via a secondary bus bar to power the system’s four Teensy 4.1 microcontrollers, each mounted to independent PCBs, as well as the Raspberry Pi 5 with an integrated touchscreen interface. By separating the 13.8V and 5V rails across two distinct distribution layers, the system ensures voltage isolation, minimizes ripple coupling between analog and digital domains, and maintains a stable current delivery to all processing components.

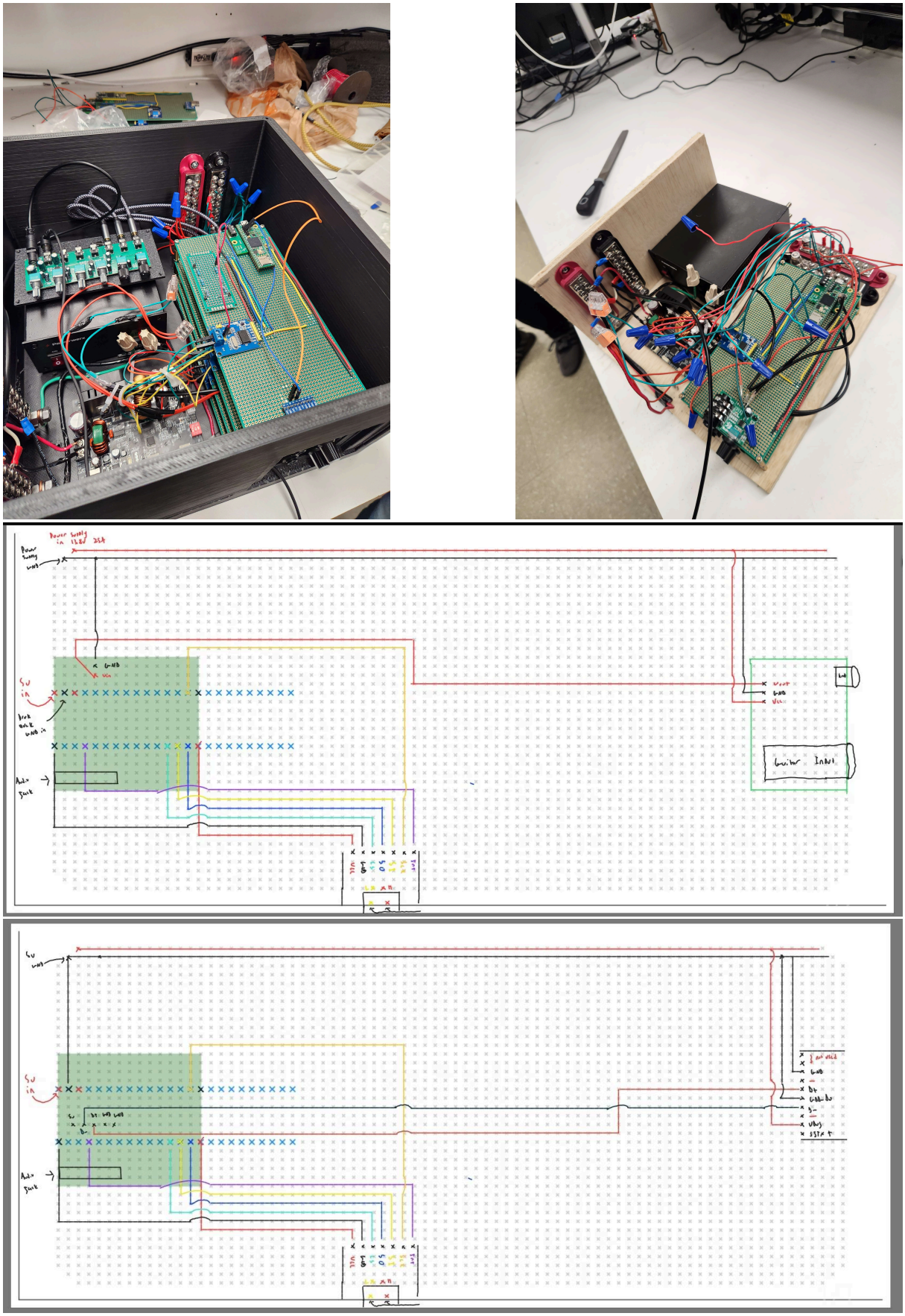

Figure 5a. and Figure 5b. PCB board schematics for Guitar (top) and MIDI (bottom) inputs.

IV. Results

To measure the latency of the system, two separate CAN log files were analyzed to evaluate the real-time communication and parameter control of the MIDI and guitar subsystems within the audio interface. Each Teensy 4.1 module received parameter updates over the MCP2515 CAN bus interface, which were logged and timestamped for validation.

The MIDI subsystem communicated using CAN ID 0x100. The primary parameter observed was the waveform type, which was updated multiple times between values 0.0 and 3.0, representing different waveform types (e.g., sine, square, sawtooth, triangle). These updates occurred at an average interval of less than one second, with a consistent CAN send latency between 0.113ms and 0.368 ms. The data confirms that waveform changes were successfully sent and received with minimal latency, ensuring seamless control during live performance or user interaction. These results demonstrate the system's ability to dynamically change oscillator types in response to user input, a critical feature for real-time MIDI synthesis.

Timestamp	Parameter	Value	Send Time (ms)
17:06:15	Waveform	0.0	0.368
17:07:01	Waveform	1.0	0.141
17:07:12	Waveform	2.0	0.113
17:07:13	Waveform	3.0	0.153
17:07:13 (later)	Waveform	0.0	0.298

Figure 6. MIDI CAN bus logs for latency measurements.

The guitar subsystem communicated using CAN ID 0x300. Multiple effects-related parameters were updated, including Delay Toggle, Division Mode, Number of Delays, and BPM. Delay effects were toggled on with Delay Toggle = 1.0, and further customization was achieved by updating the number of delays and beat division settings. The BPM value was set

to 5.0, which corresponds to an internal tempo range. The low latency, ranging from 0.109 ms to 0.204 ms, indicates effective synchronization between the user interface and the guitar effects processor. These updates confirm the reliability of real-time effect modulation via CAN bus, a key requirement for maintaining musical timing and expressiveness during performance.

Timestamp	Parameter	Value	Send Time (ms)
17:07:25	Delay Toggle	1.0	0.146
17:07:26	Division Mode	1.0	0.191
17:07:26	Number of Delays	1.0	0.204
17:07:27	Delay Toggle	1.0	0.109
17:07:30	BPM	5.0	0.122

Figure 6. Guitar CAN bus logs for latency measurements.

To assess the effectiveness of real-time spectral analysis in our system, we performed a Fast Fourier Transform (FFT) on incoming guitar signals using the Teensy Audio Library. FFT is a critical tool in digital signal processing that converts time-domain audio signals into the frequency domain, allowing us to analyze which frequencies are present and how strong they are over time. This type of analysis is essential in guitar signal processing for tasks like tone shaping, dynamic EQ, pitch detection, and audio-reactive effects. By monitoring the magnitude of a specific frequency bin, we gain insight into the energy distribution of the signal and verify that the system is correctly capturing and processing audio content.

The processed guitar signal was analyzed using a Fast Fourier Transform (FFT) to isolate frequency content over time, with the focus placed on a single frequency bin (bin 4). This bin corresponds to a low-frequency range relevant to the guitar input. The magnitude values extracted from this bin represent the strength or presence of that frequency component during each sample period. Over a sequence of 20 samples, the magnitudes ranged from a minimum

of 0.051 to a maximum of 0.108, with a mean amplitude of approximately 0.068. This suggests a relatively stable energy presence in that frequency bin with moderate variation. The standard deviation of 0.017 indicates a slight fluctuation in amplitude, which can be attributed to the natural variation in guitar plucking strength and tonal decay. These results confirm that the Teensy-based system is effectively detecting consistent energy from the guitar signal at a specific frequency band, which is useful for tasks like note detection, envelope tracking, or effects modulation based on spectral content. While the magnitude values in bin 4 suggest a consistent presence of low-frequency content from the guitar, the small but measurable fluctuations (standard deviation = 0.017) may also indicate the presence of low-level noise or harmonics in the signal. The fact that the bin never drops to zero and shows values like 0.051 even during presumed quieter moments suggests that some amount of background signal is being captured by the FFT process. Therefore, while the data supports reliable detection of intentional signal content (e.g., plucked strings), it also confirms the presence of persistent low-frequency signal components, which may include both musical sustain and system noise. This observation can inform future filtering strategies (e.g., gate thresholds or noise suppression) to enhance clarity and signal-to-noise ratio (SNR) in the audio processing pipeline.

	Sample	Bin	Magnitude
1	0	4.0	0.077
2	1	4.0	0.061
3	2	4.0	0.09
4	3	4.0	0.065
5	4	4.0	0.051
6	5	4.0	0.068
7	6	4.0	0.056
8	7	4.0	0.069
9	8	4.0	0.054
10	9	4.0	0.054

10	9	4.0	0.054
11	10	4.0	0.062
12	11	4.0	0.063
13	12	4.0	0.057
14	13	4.0	0.058
15	14	4.0	0.067
16	15	4.0	0.087
17	16	4.0	0.103
18	17	4.0	0.052
19	18	4.0	0.108
20	19	4.0	0.054

Figure 7a and Figure 7b. Guitar FFT isolated magnitude measurements.

V. Discussion

AudioLink demonstrates a fully functional real-time audio processing system designed for collaborative live performances. By distributing Digital Signal Processing (DSP) workloads across four Teensy 4.1 microcontrollers and using a Raspberry Pi 5 for touchscreen control and CAN bus communication, the system achieves modularity, low latency, and responsive input handling. This architecture enables independent frequency and effect control for two MIDI and two guitar inputs, with all settings adjusted in real time through a centralized touchscreen interface. Resulting in a streamline, musician-friendly platform replacing a traditional multi-device setup.

Key challenges during development included physical integration of components, wire management, and achieving clean, reliable communication between subsystems. Early testing revealed issues with the CAN bus modules, which failed to communicate properly due to receiving less than the required 5V input; resolving this required careful power distribution and consistent voltage regulation to each module. Additionally, we encountered audible noise in the analog signal path between the audio mixer and the Teensy Audio Shields. To address this, we replaced standard jumper wires with shielded coaxial cables, which significantly reduced electromagnetic interference and preserved audio fidelity. The use of perfboards for early prototyping further contributed to instability and wire clutter, reinforcing the need for custom PCB implementation in future iterations. Now that we've confirmed the audio mixer can handle input voltages up to 20V, we plan to increase the supply voltage slightly in future versions to further improve headroom and achieve even smoother analog performance.

VI. Conclusion

AudioLink successfully delivers a fully functional, real-time audio processing system designed for collaborative musical performances. By combining the high-speed digital signal processing capabilities of four Teensy 4.1 microcontrollers with the control and interface management of a Raspberry Pi 5, AudioLink enables independent frequency and effects manipulation for two MIDI instruments and two electric guitars. The integration of a responsive touchscreen interface, analog mixer, and clean power design ensures that the system remains user-friendly, and performance ready.

Through extensive testing and development, the team overcame numerous challenges such as those related to power distribution, communication stability, and analog noise. Ultimately refining the systems reliability and sound quality. While the system as it works extremely well moving forward, the addition of wireless connectivity and a battery-powered operation could further enhance the systems portability and flexibility. AudioLink is a strong foundation for accessible, compact, and high-quality audio collaboration with potential applications in rehearsals, educational environments, and live performances.

VII. Bill of Materials/IEEE Standards

Item Name	Vendor	Link to Purchase	Description / Justification	Price	Quantity	Total
Raspberry Pi 5	Amazon	link	Running the touchscreen for real time-audio control.	$152.99	1	$152.99
PCB Board	Amazon	link	Connecting Components and organizing wiring.	$17.59	4	$70.36
PCB Connector Pins	Amazon	link	Connecting components on PCB boards.	$8.99	1	$8.99
Teensy 4.1	Amazon	link	Handling audio effects and processing for each instrument.	$39.99	4	$159.96
CAN BUS Module	Amazon	link	Enables CAN communication between Teensy and Raspberry Pi.	$13.00	1	$13.00
Raspberry Ii Cooler	Amazon	link	Keeps Raspberry Pi at optimal temperature	$9.00	1	$9.00
Boat Bus Bar	Amazon	link	Distributes power efficiently to multiple components.	$19.99	1	$19.99
Raspberry Pi Touchscreen	Amazon	link	Allows users to interact with the GUI and adjust audio settings.	$99.98	1	$99.98
Coaxial Cable	Amazon	link	Used for clean and shielded audio signal transmission.	$12.99	1	$12.99
16 Gauge Solid Copper Wire	Amazon	link	Powers high-draw components with minimal resistance.	$22.29	1	$22.29
12 Gauge Wire	Amazon	link	Provides heavier gauge wiring for main power lines.	$24.79	1	$24.79
Drok Buck Converter	Amazon	link	Steps down voltage to appropriate levels for microcontrollers.	$34.99	1	$34.99
30DV 30Amp DC Power Suply	Amazon	link	Powers the entire system with stable, high current output.	$170.70	1	$170.70
Jumper Ribbon Cables	Amazon	link	For quick connections.	$6.98	1	$6.98
USB 3.0 Type A Female	Amazon	link	Allows USB connection for MIDI devices,	$19.00	1	$19.00
High-Impedance Preamp Board	Amazon	link	Boosts guitar signal to line level before processing.	$14.01	2	$28.02
6 Ways Stereo Mixer Board	Amazon	link	Mixes stereo signals from different Teensy output.	$16.70	1	$16.70
Wire Nuts	Amazon	link	Safely connects multiple wires together in power distribution.	$19.99	1	$19.99
Teensy Audio Shield	Amazon	link	Stereo output for Teensy 4.1.	$39.58	4	$158.32
USB C Wire Open End	Amazon	link	Power Raspberry Pi 5.	$8.99	1	$8.99
3.5mm Nylon Braided Aux Cab	Amazon	link	Connects instruments to Audio Mixer for single output.	$4.99	4	$19.96
3.5mm Female to Male Aux Ca	Amazon	link	Connects desirable speakers to Audio Mixer output	$9.99	1	$9.99
					Total Amount	$1,087.98

VIII. References

[1] D. Gibson, *The Art of Mixing: A Visual Guide to Recording, Engineering, and Production*, 2nd ed. Boston, MA: Cengage Learning, 2005.

[2] P. Stoffregen, “Teensy Audio Library Documentation,” PJRC, [Online]. Available: https://www.pjrc.com/teensy/td_libs_Audio.html. [Accessed: May 15, 2025].

[3] Robert Bosch GmbH, *CAN Specification Version 2.0*, Stuttgart, Germany: Bosch, 1991.