Modular Desktop Audio Mixer Controller

Aarushi Sharma

Dylan Moon

ECE 445 Design Document - Spring 2026

TA: Yulei Shen

27 February 2026

Project No. 85

Abstract

This project defines a mixer control device in the form of a modular desktop panel. The product enables more physical interaction between a computer user and their computer's audio. A mixer panel generally consists of a base module and fader modules. The base module interfaces with the user's computer via USB. Fader modules can then be attached to the base module or to other fader modules. A computer application will serve as an interface between the hardware product and the computer's software mixer. The resulting mixer panel should be a hardware control interface of relatively low latency with one-to-one audio controls. Audio adjustments via the hardware product should be reflected in the computer's OS within a 5% margin of error- and vice versa. Audio control adjustments in both directions should meet 0.5 second latency. Our findings should reflect a precise and reliable product for effective audio control.

Contents

1. Introduction

1.1 Problem and Solution:

Modern desktop computers have generally revolved around a set paradigm for human-computer interaction: the keyboard and the mouse. However, analog control surfaces can be beneficial in interacting with the many analog-like controls present in a computer. With our project, we want users to be able to interact with a physical analog control for software volume mixers. These software volume mixers are prevalent in modern computing systems, shipping with the OS on Windows and Linux-based operating systems.

One of the main motivations of our project is the difficulty of accessing this mixer GUI. It is usually buried behind multiple menus. Power users might have music playing in the background, a call in the foreground, and application audio on top of that, which all need to be individually adjusted so that important details are heard. Gamers, who may frequently have full-screen applications occupying their screen, lose precious time when minimizing their game and searching for the volume mixer to turn down the loud voice call that they have in the background. The time spent doing so could be the difference between winning and losing their current match.

To address this shortcoming in user interface design, we will create a modular audio control panel that sits on a user's desktop, which can be physically interacted with to smoothly and easily change volumes of individual applications. Since the controls that we want to target are analog (volume controls for individual applications), the control surfaces that the user interacts with will be linear sliders. This allows for quick but granular control of the volume levels of various applications in the computer. To achieve this, the system will consist of at least two different types of modules. One type of module is a base station that connects to the computer. This module will control other modules, process the physical inputs and output signals from the other modules, and provide power to the other modules. The other necessary type of module are the fader modules. These modules contain a slider which the user can adjust. These modules connect to the base module, communicating slider position whenever the user changes the setting of a slider. We are designing the system so that the slider modules can be daisy-chained to give the user the freedom to choose the number of sliders to include in their setup. More details can be found in the Solution Components section.

1.2 Visual Aid

Our product targets users who utilize a PC at a desk. Ideally, the controls would sit on the desk near the user's peripherals

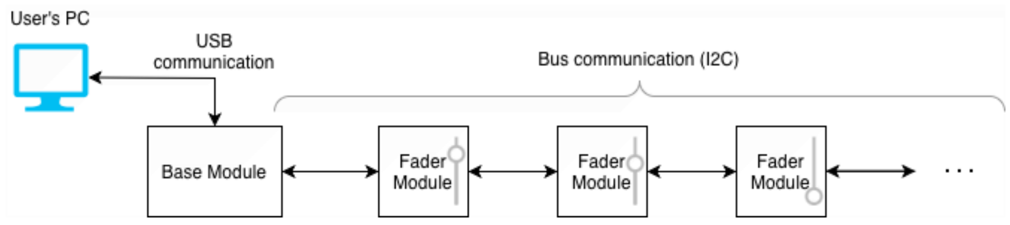

Figure 1: Visual aid, physical layout of system

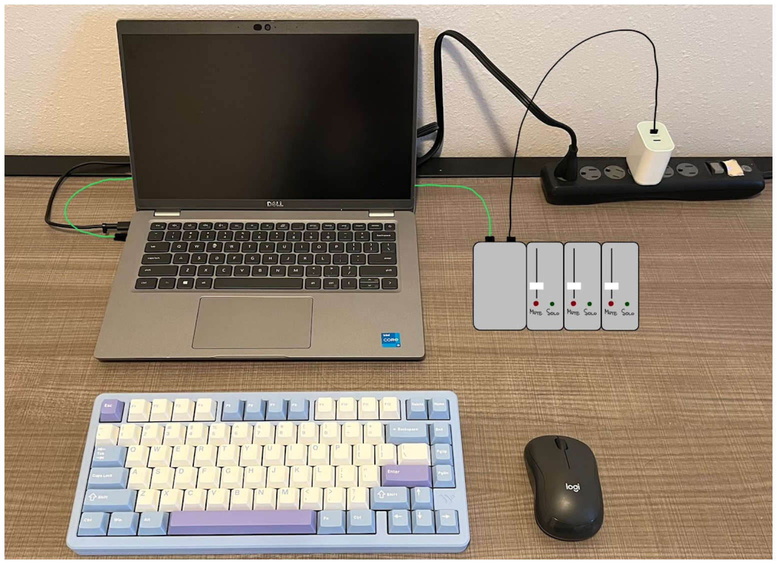

Figure 2: Visual aid, example setup on a user's desk. Final product may look different.

1.2 High-Level Requirements

A successful Audio Mixer Control project must fulfill the following:

1. Audio adjustments via the hardware's fader modules should be reflected in the computer's OS settings within 0.5 second of latency & within a 5% margin of error.
2. Audio adjustments via the computer's built-in OS settings should adjust the physical position of that setting's designated fader within 0.5 second of latency and within a 5% margin of error. These adjustments should be differentiable and treated separately to Fader position updates via Firmware.
3. Module additions and removals to an Audio Mixer Panel must be seen by the hardware within 1 second of latency, observable by the accompanying desktop GUI.
4. Reassignments of audio settings to fader modules via the accompanying desktop GUI must be reflected by the Audio Mixer Panel hardware within 1 second.

2. Design

2.1 Block Diagram

Figure 3 depicts the general block diagram of our solution. The Base module contains the control subsystem, which sends and receives messages from the PC and connected Fader modules to set volumes and fader positions. Each Fader module contains an MCU for communication with the Base module and motor faders to be able to read and write data from the physical interface. The motor will be driven by each fader's individual MCU running a control loop. Finally, the PC will be running software that communicates with the Base module over USB.

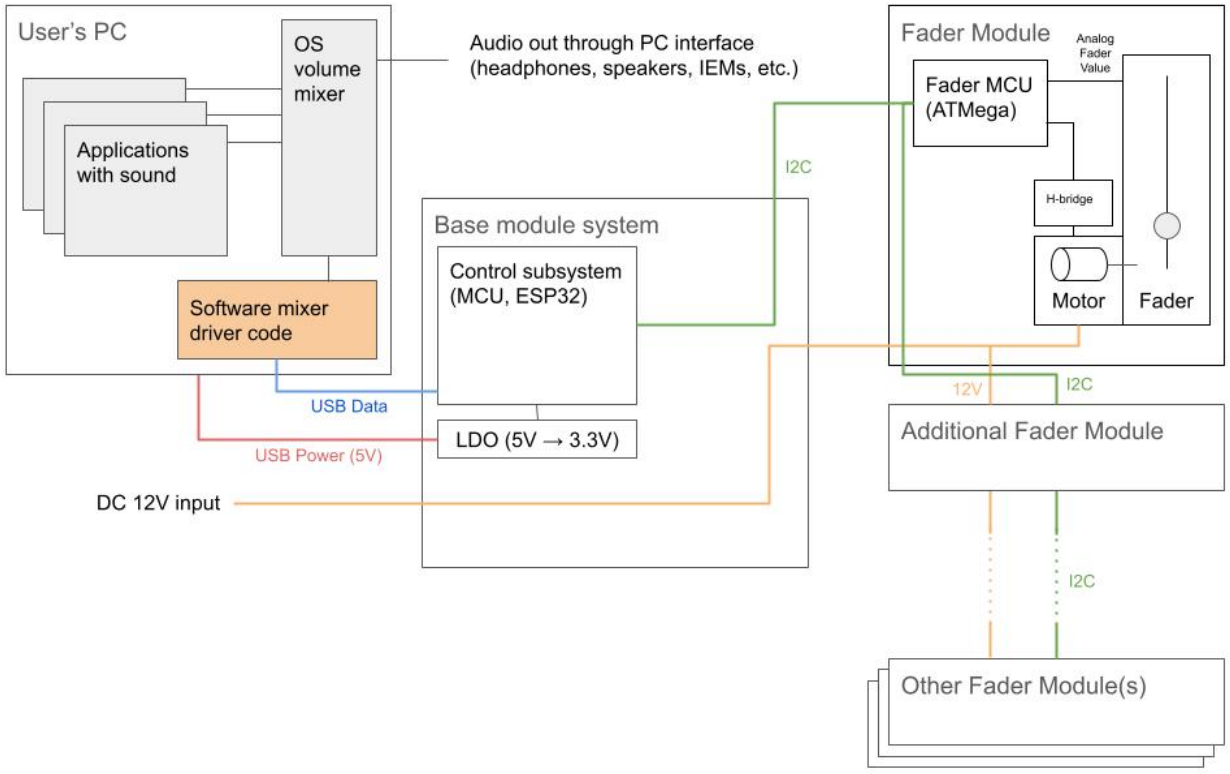

Figure 3: Block Diagram of Product with Subsystems

2.2 Base Module

The Base Module forms the foundation of an Audio Mixer Control panel. The Base Module connects to the user's computer via USB and also accepts a 12V DC power input to drive the motor faders. The USB connection provides a data connection and 5V DC power for the module, which is stepped down via a regulator to the logic voltage of 3.3V, which can supply the ESP32 microcontroller.

The ESP32 microcontroller interfaces between the computer's software audio mixer configuration and connected Fader Modules. The Base Module has a series of Pogo pin connectors, through which its microcontroller transmits and receives data via I2C. A Fader Module can be connected via the Pogo pin with a magnetic closure. This enables quick but stable connections with Fader Modules for modular mixer setups. Modules can thus be chained to the base module and reconfigured with ease.

The Base Module additionally routes 12V power from an external source to attached Fader Modules over a Pogo pin in the series of pins, which will be used by the Fader Modules to drive the motorized fader. This is required because the motors could create a lot of noise if powered off of the same rail as the logic circuitry, which could cause microcontrollers to restart or display other erratic behavior. Additionally, this allows us to disregard the 500mA limit of the 5V USB connection. This 12V source must be able to handle a peak current of 3A to be able to drive multiple of the fader modules.

Table 1: Base Module Subsystem – Requirements & Verification

Requirements	**Verification**
● The Base Module should receive 5V from the USB connection and 12V from the external DC source (USB PD) and be able to pass it on through the Pogo pins	● Connect USB and external power source, probe 5V and 12V Pogo pins with multimeter. Measurement should be within 4.5 to 5.5V for the 5V pin and 11.5 and 12.5V for the 12V pin.
● The Base Module must be able to detect the addition or removal of any Fader Modules from the panel.	● Start with no Fader modules connected. Attach a Fader module. Verify that the software shows the connected Fader module in the configuration screen. Then detach a Fader module. Verify that no Fader modules are present in the configuration screen.
● The Base Module must be able to dynamically manage I2C addresses for connected Fader Modules & inform Fader Modules of their assigned address.	● Implicitly required to detect multiple Fader modules. Start with one Fader Module connected. Attach another fader module. Verify that the Base Module shows two connected Fader modules.

2.2 Fader Modules

Each Fader Module features a fader, a motorized linear slider that can be used to configure an audio setting. Fader Modules can be chained to a Base Module through Pogo connections, creating shared buses for data and power across the chain. These connections deliver 12V power necessary to operate the motor for each fader, along with I2C data lines for communication between the Base and Fader modules. As long as the modules are magnetically attached, the Pogo connections must maintain an electrical connection between each connected module; if there are small blips in the connection quality, the fader module could randomly restart, causing user confusion.

Each Fader Module has an on-board microcontroller, which uses the I2C bus for bidirectional communications. When a fader is manually adjusted, the microcontroller should forward these adjustments to update software mixer values on the computer. When the microcontroller hears that its Fader Module's assigned audio setting has been adjusted in computer software, it should update the fader's position via its motor. Thus, the microcontroller must distinguish between and respectively handle updates originating from computer software and fader movement to ensure synchronization between the computer and the Audio Mixer device. The microcontroller must be able to quickly send fader position information to the Base Module and also receive and execute fader positioning instructions to maintain low latencies that are crucial for a good user experience when adjusting analog interfaces.

Table 2: Fader Module Subsystem – Requirements & Verification

Requirements	**Verification**
● Each Fader module must be able to receive messages designated for itself over I2C.	● We can create a test program on the Base Module to send an update to a Fader Module's Mute status via I2C. (i.e. wait(10s) -> send Mute update -> wait) ● With the test program, we should see the Fader Module's Mute indicator turn on. ● Turn on the Mute status of an app connected to a certain fader via the OS Integration Software. The Mute indicator light should turn on.
● Physical adjustment of each fader must be broadcasted by its Module over I2C.	● Using a debugger on the Base Module, we can create a test program that prints updates received by the Base Module via I2C. ● With the test program, moving a fader should result in the Base Module printing the update from the Fader Module. ● With a complete project, moving a fader should result in its setting being updated on the computer.
● Computer-side adjustments to audio settings must result in the actuation of settings' assigned faders to the correct corresponding position.	● Adjust the volume of a specific application that is associated with the Fader module in the OS volume mixer using the mouse to 50%. The motor should move the fader to a position that is halfway between both ends of the fader, within a 10% tolerance.
● All Fader Modules must be able to maintain bidirectional continuity over the I2C bus.	● Create a test program on the Base Module that sends an I2C update for the Mute setting on the very last Fader Module in the chain connected to the Base. (i.e. wait(10s) -> send Mute update -> wait) ● With the test program, we should see the last Fader Module's Mute indicator turn on. ● Turn on the Mute status of an app connected to the last fader via the OS Integration Software. The Mute indicator light on that Fader Module should turn on.

2.3 OS Integration: Accompanying GUI

A dedicated application running on the computer forms the OS-side of the Audio Mixer interface. Through the USB connection with the Base Module, this program will maintain synchronization between the software settings and the physical fader positions.

When a Fader is moved, the application will receive the Fader-side update and utilize Windows APIs to update the OS mixer values. Likewise, when a user updates an audio setting on the computer's UI, the application will broadcast the update to the setting's corresponding fader (if assigned).

The program should also be used to configure the settings controlled by each Fader Module through a graphical user interface (GUI). This GUI will display each Fader Module actively connected to the device and allow the user to assign an audio setting for each Fader. Addition or removal of Fader Modules from the system will be instantaneously reflected in the GUI. The fader-setting configuration should be transmitted to and maintained by the Audio Mixer hardware.

Table 3: OS Integration Subsystem – Requirements & Verification

Requirements	**Verification**
● The OS application should broadcast audio setting updates via the computer to the Base Module	● Create a test program on the OS application to send audio setting update(s) to the Base Module. A test program on the Base Module (via debugger) should print that the Module has received the respective updates. ● Altogether, adjusting computer audio settings should result in actuation of corresponding Faders to the appropriate positions.
● The OS application should receive Fader position updates from the Base Module and update the corresponding computer audio setting accordingly	● Create a test program on the Base Module to send a Fader position update over USB. The OS application test program should update an audio setting upon receiving the message. ● Create a test program on the Base Module to send updates for two separate Faders. The OS application test program should update the two respective audio settings correctly. ● Altogether, moving Fader positions on Fader Modules should update the corresponding computer audio settings.
● GUI should be able to configure applications to certain faders	● Connect Base with 1 Fader module to the computer. Play some audio with an application on the computer. Utilize the GUI and assign the application to the fader. Then physically move the fader. The volume of the App should change.
● GUI should dynamically display all available Fader Modules	● Connecting and removing a Fader Module to the Base Module should result in the GUI reflecting the Panel Layout within 1 second

2.4 Tolerance Analysis: Motor Fader Voltage Drop Across Modules

We want to analyze the voltage drop across multiple modules to make sure that we can support multiple motors actuating at once. According to the data sheet, the motor fader RSA0N11M9 draws a max current of 800mA at 10V DC [3]. Since motors draw the most power when starting, we can convert this to an effective resistance of 12.5 Ohms. A standard Pogo pin can transmit 2A continuous, 3A burst current, while the contact resistance is 50 mΩ [4]. From this, if we assume that 3 faders are trying to start moving at the exact same time, each fader would draw 12 / 12.5 = 0.96 amps of current, or a combined 2.88A. This is under the 3A burst current that the Pogo pin can carry, and if required, the design can have multiple Pogo pins to carry the 12V line to increase the current capacity. To calculate the drop from the contact resistance, we can do as follows: The Pogo pin between the base and the first fader module is carrying 2.88A, so the voltage drop over the Pogo pin is 0.144V. Between the first and second fader module, the current is 1.92A, therefore the voltage drop is 0.096V. Between the second and third fader, the current is 0.96A, therefore the voltage drop is 0.048V. Combined, at the third fader, the voltage drop is 0.288V, which should not be an issue. Also, if such a situation arises where many faders need to start at exactly the same time, we can program the microcontroller to stagger the start times of the potentiometers by milliseconds, which should prevent multiple motors from drawing startup current at the exact same times, at the expense of some unperceivable latency.

3. Cost & Schedule

3.1 Cost Analysis

The total cost of parts according to the table below would be $209.81. For the labor cost, we assume that a full-time year is (40 hrs/wk) * (52 wks/yr) = 2080 hours/yr. Since UIUC Computer Engineering majors make $103,222 on average starting [5], we extrapolate an hourly rate of ($103,222/yr) / (2080 hrs/yr) = $49.62 per hour. Assuming 10 hours per week over the 12 weeks of this project, labor costs would be around (10 hrs/wk) * (12 wks) * ($49.62/hr) = $5954.40 in labor per teammate. In total, the project would cost around $12,118.61.

Table 4: Itemized list of Components and Costs

Description	**Manufacturer**	**Quantity**	**Extended Price**	**Link**
RF TXRX MOD BT WIFI PCB TH SMD	Espressif Systems	1	$5.92	Link
IC MCU 8BIT 32KB FLASH 32TQFP	Microchip Technology	2	$5.48	Link
SLIDE POT 10K OHM 0.5W TOP 100MM	Bourns Inc.	2	$42.48	Link
IC REG LINEAR 5V 1A TO252	Rohm Semiconductor	3	$2.94	Link
5 Pin Magnetic Connector	SHENZHEN YIWEI TECHNOLOGY CO.,LTD	3	$20.85	Link
IC MTR DRV BIPLR 12-55V TO220-15	Texas Instruments	2	$65.02	Link
USB TYPE C, 3.2GEN2, 24P, RECEPT	GCT	2	$1.60	Link
SWITCH TACTILE SPST-NO 0.05A 24V	TE Connectivity ALCOSWITCH Switches	4	$0.52	Link
Enclosures (3D printed) (approx)	N/A	3	$10 (approx)	N/A
PCB	N/A	3	$25 (approx)	N/A
Misc components from ECE shop (Wires, resistors, capacitors, etc.)	Varies	N/A	$30 (approx)	N/A

3.2 Schedule

Table 5: Schedule for Project Progression

Week	**Task**	**Person**
03/02 - 03/09	Research I2C Addressing	Aarushi
	Research Windows API	Dylan
	Design 3D Print Enclosures	Aarushi
	Organize Parts List & Order Parts	Dylan
	Finalize PCB Design for Base & Fader Modules	Everyone
	03/05 PCBWay Order Round 2	Everyone
03/09 - 03/16	Enclosure Prototypes	Aarushi
	OS Application Skeleton Development (no GUI)	Dylan
	Breadboard Fader Circuit & Verify Actuation	Aarushi
	03/12 PCBWay Order Round 3	Everyone
03/16 - 03/23	Begin OS Components (w/ Windows API)	Dylan
	Develop I2C Firmware (Fader/Base)	Aarushi
	Develop USB Firmware (Base)	Dylan
	Develop Fader Actuation Firmware	Aarushi
03/23 - 03/30	PCB Assembly	Dylan
	Pogo Connection Stability Tests	Dylan
	Implement Bidirectional Audio Adjust Sync (Base)	Aarushi
	Test & Finalize Enclosures (Magnetic Connections)	Aarushi
	Finalize Fader Actuation Firmware	Aarushi
	Module-Level Fader Actuation Tests	Aarushi
	03/26 PCBWay Order Round 4 (Final)	Everyone
03/30 - 04/06	Finalize I2C Firmware	Aarushi
	Preliminary (Module-Level, I2C) Integration Tests	Everyone
	Finalize USB Firmware	Aarushi
	Finalize Core OS Components	Dylan
	System-Level Tests	Everyone
	PCB Tweaks & Bugfix	Everyone
04/06 - 04/13	Develop GUI for OS Component	Dylan
	Iterate on Firmware for Stable Operation	Aarushi
	Latency Optimizations (0.5s)	Everyone

	Adjustment Accuracy Optimizations (<5%)	Everyone
	Optimize Modularity Components (Fader Detection & GUI)	Dylan
	Progress Demo	Everyone
04/13 - 04/20	Final System Integration Tests	Everyone
	Polish Hardware & Firmware	Everyone
	Prepare Supporting Documents	Everyone
04/27 - 05/04	**Demo**	Everyone

4. Conclusion

4.1 Ethical Considerations

This product, though an interface for enhancing existing audio controls, must uphold several ethical considerations. Privacy is particularly paramount in the computer application component of the product, with ACM principle 1.6 underlining "informed consent for automatic data collection, and to review, obtain, correct inaccuracies in, and delete their personal data" [1]. When designing the OS component of this product, we must ensure that the product solely interacts with audio controls with the user's knowledge, and maintain that the design will forgo any collection or storage of any data as intended.

We must additionally take measures to mitigate potential health risks for this product in accordance with IEEE's Code of Ethics I Part 7- "to hold paramount the safety, health, and welfare of the public" [2]. Potential hazards of this product include pinch points on the faders and the magnetic module interconnects, as well as mild magnetic interference to medical devices (like pacemakers) from these magnetic interconnections. The design of the device's body should prevent any extrusions that increase pinch risk or guard against accidental access to pinch points. We must moreover post clear notice of these risks for the product's users, i.e. via warning labels. Through the design process of the product, we must take careful notice of any warnings attached to hardware components and ensure that the product informs users of these warnings.

By IEEE I Part 5, we must also be "honest and realistic in stating claims or estimates based on available data" [2]. A successful audio mixer product must uphold our success criteria, and thus the performance metric claims within these criteria. We are responsible for ensuring transparency in the latency and accuracy of audio control features through measurements collected by replicable means. Under this code, we will also remain receptive and responsive to feedback for this device and "seek, accept, and offer honest criticism of technical work" [2].

4.2 Societal Impact

Considering economic, social, and global factors, we want this solution to make it easier for people to interact with their personal computers. We also believe that our solution will increase people's productivity with their computers, especially heavy multitaskers. We also hope that with such a product, accessibility for computer usage will increase, even by a little bit. Furthermore, as a heavily modular project, we want to keep the ecosystem open, so that even with the conclusion of our project, others can consult our work and extend upon it, creating new types of modules and software to improve the workflow of computer users.

5. References

[1] Association for Computing Machinery, “ACM Code of Ethics and Professional Conduct,” *Association for Computing Machinery*, Jun. 22, 2018. https://www.acm.org/code-of-ethics

[2] IEEE, “IEEE Code of Ethics | IEEE,” *Ieee.org*, 2020. https://www.ieee.org/about/corporate/governance/p7-8

[3] “RSA0N11M9A0K Product information | RS**N1*M Series | Slide Potentiometer (Master type) | Potentiometers | Products Search | Products & Technologies | Alps Alpine,” *Alpsalpine.com*, 2026. https://tech.alpsalpine.com/e/products/detail/RSA0N11M9A0K/ (accessed Feb. 14, 2026).

[4] “1.00 Pogo Pin Connector P08824SH1,” *Ccpcontactprobes.com*, 2026. https://www.ccpcontactprobes.com/en/product/794-mm-pogo-pin-694-mm-working-height-p08824sh1 (accessed Feb. 14, 2026).

[5] G. E. O. of M. and Communications, “Salary Averages,” ece.illinois.edu. https://ece.illinois.edu/admissions/why-ece/salary-averages (accessed Feb. 27, 2026).