Modular Desktop Audio Mixer Controller

Aarushi Sharma

Dylan Moon

ECE 445 Final Report - Spring 2026

TA: Yulei Shen

06 May 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 is a hardware control interface of relatively low latency with one-to-one audio controls. Audio adjustments via the hardware product are reflected in the computer's OS within a 5% margin of error- and vice versa. Audio control adjustments in both directions meet 0.5 second latency. Our findings reflect a precise and reliable product for effective audio control.

Contents

1. Introduction **4**
1.1 Problem and Solution: 4
1.2 Visual Aid 5
1.2 High-Level Requirements 5
2. Design **7**
2.1 Block Diagram 7
2.2 Base Module 8
2.2 Fader Modules 10
2.3 OS Integration: Accompanying GUI 13
3. Verification **15**
4. Cost & Schedule **17**
4.1 Cost Analysis 17
5. Conclusion **18**
5.1 Ethical Considerations 18
5.2 Societal Impact 19
6. References **20**
Appendix A Base Module Schematics **21**
Appendix B Fader Module Schematics **25**

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. For example, one ships with the OS on the Windows operating system.

One of the main motivations of our project is the difficulty of accessing this mixer GUI. For Windows version 25H2, testing showed that it takes a minimum of 4 mouse actions to access this menu. For previous versions of Windows, it took up to 6. Needing to perform all of these actions can significantly disturb someone's workflow. For example, 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. Needing to perform so many actions before being able to access the volume mixer might result in them missing an important detail in a call. 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 have created 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 are linear sliders. This allows for quick but granular control of the volume levels of various applications in the computer. To achieve this, the system consists of 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 type of module is the fader module. These types of modules contain a slider which the user can adjust. These modules connect to the base module, communicating slider position whenever the user moves a slider. We designed 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 other computer peripherals so that the user has quick and easy access to the control sliders of the product. Figure 1 and 2 show the physical layout of the system and a possible user's setup, respectively.

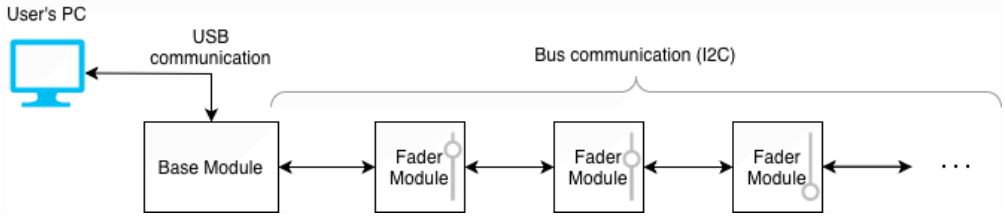

Figure 1: Visual aid, physical layout of system

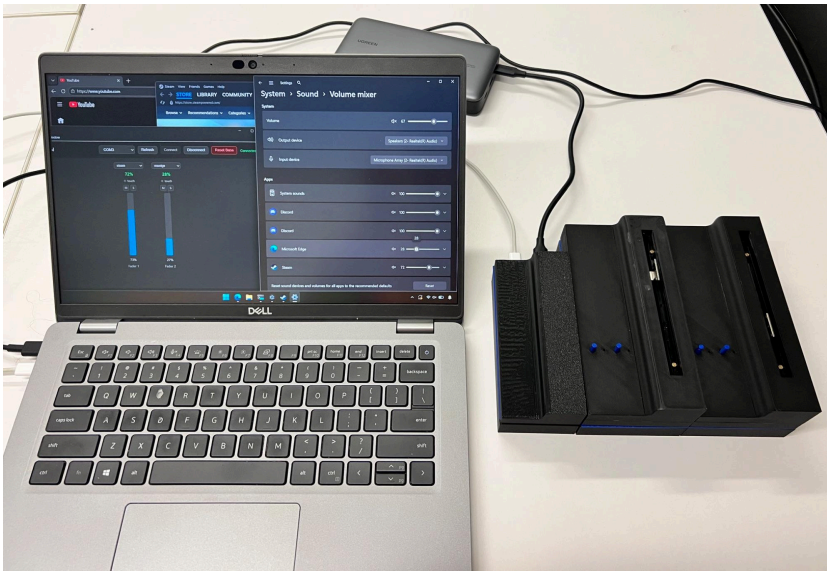

Figure 2: Visual aid, example setup on a user's desk

1.2 High-Level Requirements

Our successfully functioning Audio Mixer Control project fulfills the following high-level requirements:

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 application'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.

We made one block-level change over the course of this project's design and implementation. Instead of using 12V DC input to the base module to drive all fader modules as originally planned, we used 9V. Additionally, this input is specifically delivered through USB Power Delivery (PD).

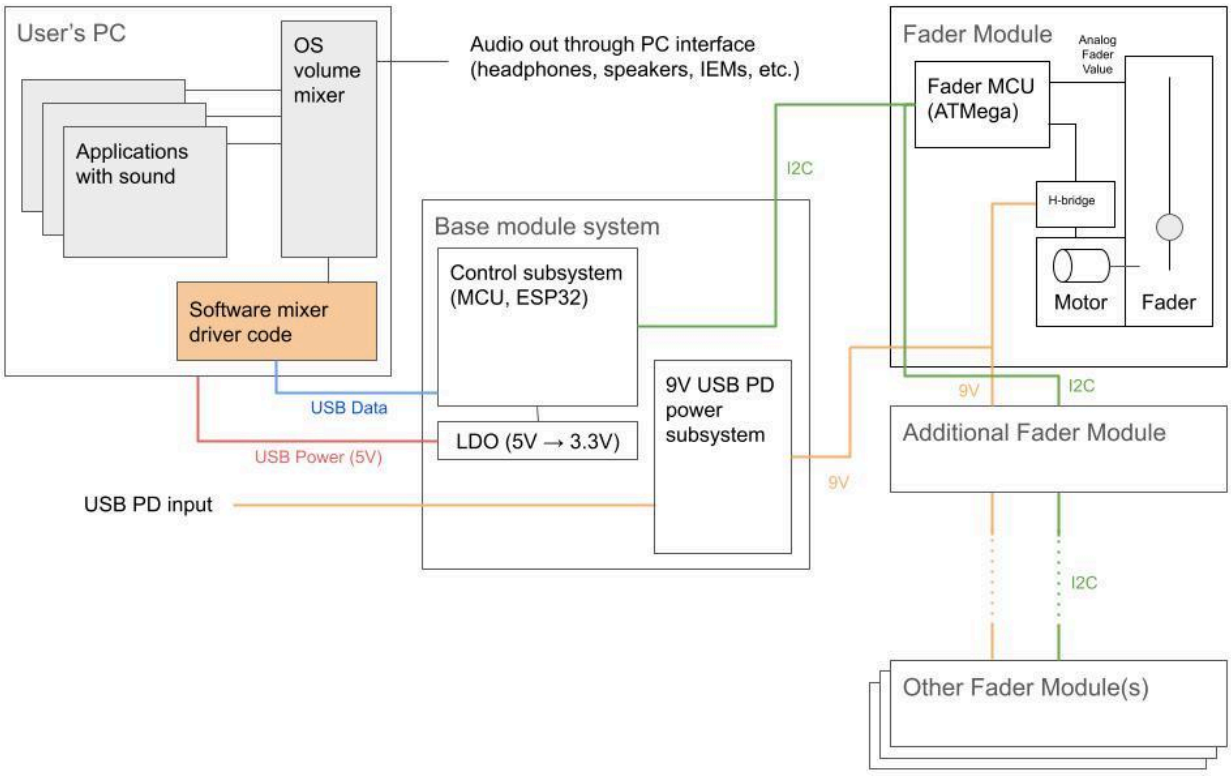

Figure 3: Block Diagram of Product with Subsystems

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 USB PD 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 USB PD connection is negotiated to a power level of 9V 3A, which is used to drive the motors.

One alternative to the design that we chose not to move forward with was the 12V DC input via a barrel jack. We initially considered this design due to its simplicity, as we would only need to route the DC voltage lines from the jack to the H-bridges on the fader modules. However, we chose to switch to the USB PD DC input due to experiences of frustrations with devices that operated on barrel jacks, most importantly a non-universal connector and variations in provided voltage and current with each DC barrel jack adapter. We also knew that most modern phone and laptop chargers support USB PD specifications, so a user might be more likely to possess a PD-compatible charger rather than a 12V DC adapter that terminates with a barrel jack of a specific size. Because we decided to switch to using USB PD, we also decided to change the voltage to 9V. This is because the initial 12V is not a widely supported voltage level of PD (some adapters optionally implement it in fixed-voltage mode and PPS adapters can supply it). On the other hand, 9V is a required voltage level for all USB PD compatible devices and is still within the operating voltage range of our motorized slide potentiometer.

The data side of the Base module includes the ESP32 microcontroller, which interfaces between the computer's software audio mixer configuration via USB and connected Fader Modules via I2C. The ESP32 has dedicated USB D+ and D- pins which, when connected to a PC, allow communication over the USB cable. Additionally, the ESP32 has assigned I2C pins which are connected to 2 of the pins on the 5-pin pogo connector on the side of the Base module. The pins transmit 9V power, 3.3V power, and carry the I2C bus. A pinout of the connector can be seen in Figure 4. A Fader Module can be connected via the Pogo pin with a magnetic closure. This enables quick and stable connections with Fader Modules for modular setup. Modules can thus be chained to the base module and reconfigured with ease. The schematics of the data side of the Base module can be found in Figure A.1 and A.3.

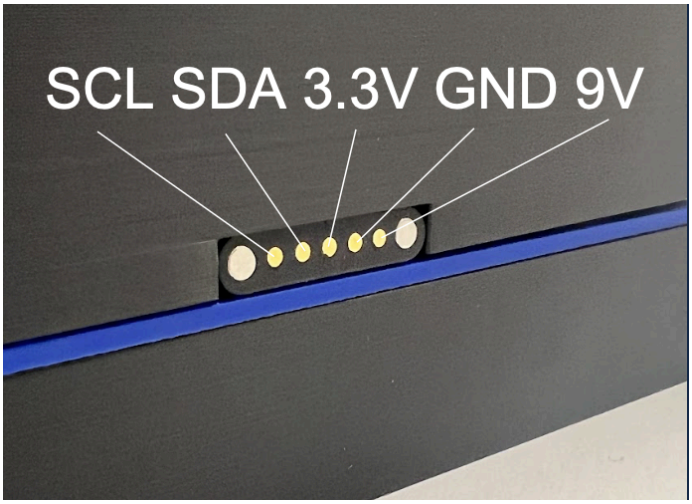

Figure 4: Guide for Base module connector pinout

The Base Module also routes 9V power from an external source to attached Fader Modules over a Pogo pin in the series of pins, which is used by the Fader Modules to drive the motorized fader. This is required because the motors create a lot of noise if powered off of the same rail as the logic circuitry, which causes microcontrollers to restart or display other erratic behavior. Additionally, this allows us to disregard the 500mA limit of the 5V USB connection. We achieve the USB PD negotiation using the CYPD3177 EZ-PD BCR, which is configured to a standard PD voltage level of 9V and 3A via resistor dividers on the VBUS_MIN/MAX and ISNK_COARSE/FINE configuration pins. Additionally, two 2SJ652 P-channel MOSFETs were used to create a two-way gate to protect against misconfigured voltage and backflow. The schematic for the USB PD negotiation circuit is shown in Figure A.2.

One of the parts that we considered while designing the power subsystem in the Base module was the amount of current that the motors would draw. According to the data sheet, the motors in the motorized slide potentiometers can draw up to 800 mA of starting current. Additionally, since our system is modular, we needed to account for multiple motors starting simultaneously. We decided to allow up to 3 motors to start simultaneously. Any more than that is limited by software to start with a small time offset, when the other motors would not be drawing 800 mA anymore. Therefore, the power circuit needs to support up to 2.4 A. The required trace width for 2.4 A was calculated with the KiCAD PCB calculator. For a trace to support 2.4 A with ΔT of max 10°C, the trace width needs to be 40 mils. To be additionally safe, we also applied the 80% rule and used trace widths of 50 mils in our PCB design.

Table 1: Base Module Subsystem – Requirements & Verification

Requirements	Verification
• The Base Module should receive 5V from the USB connection and regulate it to 3.3V and 9V 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 3.3V and 9V Pogo pins with multimeter. Measurement should be within 2.8V to 3.8V for the 3.3V pin and 8.5 and 9.5V for the 9V 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 9V 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, an ATMega328P-AUR, which uses the I2C bus for bidirectional communications. When a fader is manually adjusted, the microcontroller forwards 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 updates the fader's position via its motor through a simple PD control loop, with feedback provided by the SERVO potentiometer track. Thus, the microcontroller distinguishes between and respectively handles updates originating from computer software and fader movement to ensure synchronization between the computer and the Audio Mixer device. The microcontroller quickly sends fader position information to the Base Module and also

receives and executes fader positioning instructions to maintain low latencies that are crucial for a good user experience when adjusting analog interfaces.

Essential to the Fader module's functionality is capacitive touch sensing, a key feature in the Bourns PSM01 motorized potentiometer fader components that we have selected for our Fader modules. Touch sensing allows our Fader Modules to accept adjustments to the faders when they are being touched by the user. When touch is sensed, the Module can correctly differentiate these adjustments from unexpected deviations of the fader position from the computer's settings.

The PSM01 maintains two sets of potentiometer tracks and a capacitive touch sensing pin (alongside its motor inputs) (*Fig. 5*). Each Fader PCB connects to its PSM01 via a 6-pin IDC connection. Our original design called for this connector to carry 3.3V and GND connections alongside the two potentiometer track and the touch sense output. Touch sense was connected to PB2 Pin 14 on the ATMega, which by itself was a general-purpose digital input/output (GPIO) (*Fig. B.4)*. We had intended on implementing capacitive touch sensing using the QTouch Library as suggested by the ATMega's datasheet. After PCB assembly, we found that QTouch was outdated and its current successor was incompatible with PC0. QTouch installation required a proprietary Windows application, which we decided was too much implementation overhead.

The firmware for this project had been implemented in Arduino IDE, which already had an ADCTouch library. Our motor controls only used the SERVO potentiometer track. This meant that the LINE track, connected to Pin 23 ADC0, was unused. We thus swapped the TOUCH connection in for the LINE connection on the PSM01's connector (*Fig. B.2*). Pin 5, on the PSM01 connector which previously connected to TOUCH on the PCB remained unconnected (*Fig. B.2*). We finally updated the filtering components for the previous LINE connection: R4 was swapped from 10k to 100k Ω while C7 on Pin 23 was removed (*Fig. B.4*). We were thus able to implement capacitive touch sensing without using a different IDE and without the implementation overhead of using ATMega tools.

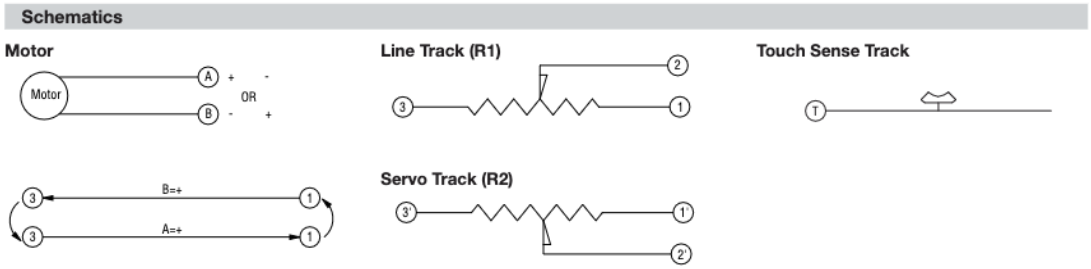

Figure 5: Connection Schematics for the Bourns PSM01 Motorized Potentiometer Fader [4]

The overall PCB incorporates decoupling for power on the 5-pin bus (*Fig. B.1*), the ATMega (*Fig. B.4*), and the DRV8871DDAR motor H-bridge (*Fig. B.2*), debounce for button

peripherals (*Fig. B.3*), and filtering for the motor potentiometer inputs (*Fig. B.4*). Pin 4 on the DRV limits the current output to approximately 2.8 A (*Fig. B.2*) via pull-down resistors for R_{ILIM} = 22.5k Ω according to the equation given in its datasheet (*Eq. 1*). This limit was chosen according to max power draw from the base module.

$$I_{TRIP}(A) = \frac{V_{ILIM}\,(kV)}{R_{ILIM}\,(k\Omega)} = \frac{64\,(kV)}{R_{ILIM}\,(k\Omega)} \qquad (1)\ [2]$$

Table 2: Fader Module Subsystem – Requirements & Verification

Requirements	**Verification**
• **Requirements**	• **Verification**
• Each Fader module must be able to receive messages designated for itself over I2C.	• Adjust the volume in the OS mixer of an application connected to a fader and observe the fader's physical position move to match it.
• Physical adjustment of Faders must be broadcasted to the device over I2C.	• Verify that movement of a Fader is detected by the Base module via debugger or by the OS application
• Computer-side adjustments to audio settings must move the corresponding Fader.	• Adjust an audio setting to 50% on the computer. • Verify that the assigned Fader module for the setting moved halfway between both ends of the fader.

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 can configure the settings controlled by each Fader Module through a graphical user interface (GUI). This GUI displays each Fader Module actively connected to the device and allows the user to assign an audio setting for each Fader (*Fig. 6*). Addition or removal of Fader Modules from the system will be instantaneously reflected in the GUI. The fader-setting configuration is transmitted to the Audio Mixer hardware.

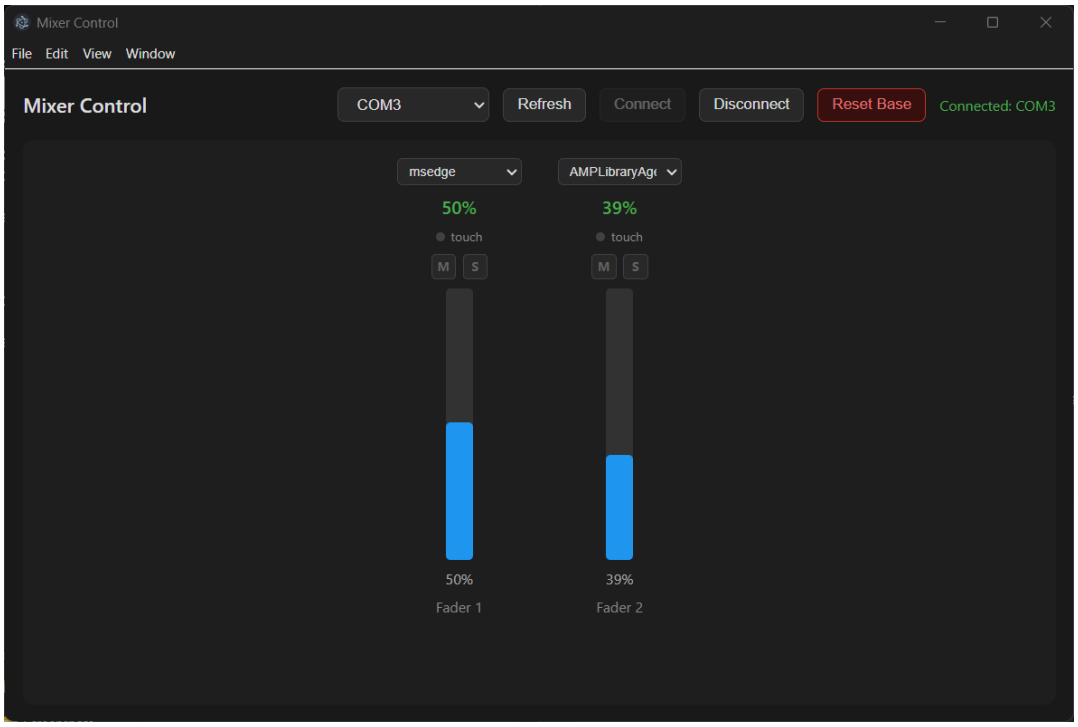

Figure 6: Application GUI with two faders attached

Table 3: OS Integration Subsystem – Requirements & Verification

Requirements	Verification
• The OS application should broadcast audio setting updates via the computer to the Base Module	• Adjustments to an audio setting must be received by the audio panel • Confirm via debugger on the base module • Ensure that adjustments are reflected in Fader position
• The OS application should receive Fader position updates from the Base Module and update the corresponding computer audio setting accordingly	• Confirm that application receives updates from the panel (programmed into the Base module or via movement of a Fader) • Confirm that corresponding computer audio settings are updated
• GUI should be able to configure applications to certain faders	• Connect one Fader module. Assign some audio application to the fader in the GUI. • The App volume should change with Fader movement
• 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

3. Verification

Testing of the system consisted of some individual module testing for the Base module, but most of the testing involved the entire system. We conducted testing this way because the Fader modules were not able to operate in standalone mode and we lacked debug console connections for the Fader module. However, we were able to verify our high-level requirements and each module's individual requirements with the entire system configuration. Additionally, for all of the per-module requirements and verifications, we did simple testing with the entire system and were able to verify functionality of each individual module's features.

To verify the high-level requirements, we filmed a video with a known frame-rate of 59.94 frames per second to measure the amounts of time required. The first high-level requirement was that adjustments on the hardware side should be reflected in the Windows volume mixer within 0.5 seconds and within a 5% margin of error. We counted 8 frames between the hand movement on the fader and the value updating in the volume mixer, and since each frame is 1/59.94=0.016 seconds, the latency is 0.133 seconds, which is well within the requirement. Since the fader is an analog control, it is difficult to determine if an intermediate value is within the 5% margin of error, but we did verify that at the end-stops the volume mixer setting was at 0% and 100%.

The next high-level requirement was the opposite direction: adjustments to the software mixer should be reflected on the hardware within 0.5 seconds and within a 5% margin of error. We started with a Windows volume mixer setting of 100%, then clicked on the mixer to bring it down to 52%. We counted 18 frames between the Windows volume mixer GUI updating and the physical fader coming to rest at the target. With this value, we can find that the latency is $0.016 \times 18 = 0.300$ seconds, which is within the requirement of 0.5 seconds. For the 5% margin of error, we relied on the physical fader position readout of our system, which read the same percentage (52%) as the value that we set with the volume mixer.

The next high-level requirement relates to module connection and disconnection latency. We counted 35 frames between complete module connection and GUI updating to show the fader. This corresponds to a latency of $0.016 \times 35 = 0.583$ seconds. Furthermore, disconnection showed a 39 frame latency. This corresponds to $0.016 \times 39 = 0.650$ seconds. Both values were within the range, fulfilling the requirement

We were able to validate that reassignments of audio settings to fader modules was reflected by the Audio Mixer Panel within 1 second when setting up our fader panel. We started with the panel connected and uninitialized with no applications assigned to any of the two modules. We assigned an application at an arbitrary volume setting of 58% to a Fader. Upon assignment, a video recording showed the assigned Fader Module moving its fader to 58%. This movement was observed to have completed 17 frames after the assignment, or $0.016 \times 17 = 0.272$

seconds. Thus, the hardware received and reflected the GUI assignment in 0.272 second, well within the 1 second threshold.

4. Cost & Schedule

4.1 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. We worked roughly 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**
ESP32 RF TXRX MOD BT WIFI PCB TH SMD	Espressif Systems	1	$5.92	Link
ATMEGA328P-AUR IC MCU 8BIT 32KB FLASH 32TQFP	Microchip Technology	2	$5.48	Link
PSM01-081A-103B2 SLIDE POT 10K OHM 0.5W TOP 100MM	Bourns Inc.	2	$42.48	Link
IC REG LINEAR 3.3V 1A TO252-3	Diodes Incorporated	1	$0.28	Link
5 Pin Magnetic Connector	SHENZHEN YIWEI TECHNOLOGY CO.,LTD	3	$20.85	Link
DRV8871DDAR IC MOTOR DRVR UNIPLR 8SO PWRPAD	Texas Instruments	2	$5.62	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)	N/A	3	~ $10	N/A
PCB	N/A	3	~ $25	N/A
Misc components from ECE shop (Resistors, caps, etc.)	Varies	N/A	$30 (approx)	N/A

5. Conclusion

Our final Modular Desktop Audio Mixer Controller successfully realized our high-level goal of providing an accurate, effective, and more natural physical control system. Through module and system-level verifications, we were able to accomplish a functional bridge between our computer and our hardware through the Base Module and the OS Application, with the Base Module successfully providing necessary power inputs to the Audio Panel. The fader modules achieved accurate and near-instant audio controls with bidirectional synchronization. Our GUI was successful for configuration and visualization of the system.

Moving forward, our primary uncertainty concerns the robustness of the 5-pin magnetic connections between our base modules (*Fig. 5*). During our demo, a short circuit from sliding a Fader Module to disconnect it from the Base damaged the ESP32's I2C pin on the Base Module, requiring replacement. To address this future design choices such as I2C buffers and keying on the 3D-printed enclosures (*Fig. 7*) should be considered.

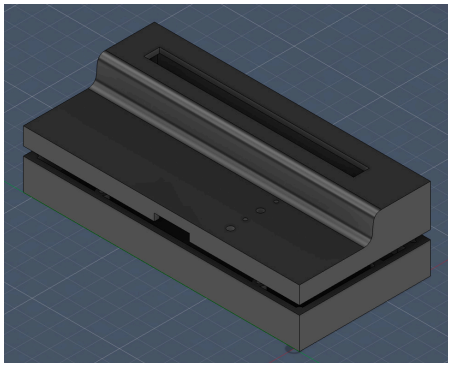

Figure 7: Fusion CAD Model of 3D Printed Fader Enclosure

5.1 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 ensure that the product solely interacts with audio controls with the user's knowledge, and maintain that the application does not collect or store any user data.

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" [3]. 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. Distribution of this product to the public would require postage clear notice of these risks for the product's users, i.e. via warning labels.

By IEEE I Part 5, we must also be "honest and realistic in stating claims or estimates based on available data" [3]. Our successful audio mixer upholds 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" [3].

5.2 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.

6. 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] “DRV8871 3.6-A Brushed DC Motor Driver With Internal Current Sense (PWM Control),” Ti.com, Jul. 2016. https://www.ti.com/general/docs/suppproductinfo.tsp?distId=10&gotoUrl=https%3A%2F%2Fwww.ti.com%2Flit%2Fgpn%2Fdrv8871 (accessed May 06, 2026).

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

[4] “PSM Series Motorized Slide Potentiometer,” Bourns, Mar. 2020. https://www.bourns.com/docs/Product-Datasheets/psm.pdf (accessed May 06, 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).

Appendix A Base Module Schematics

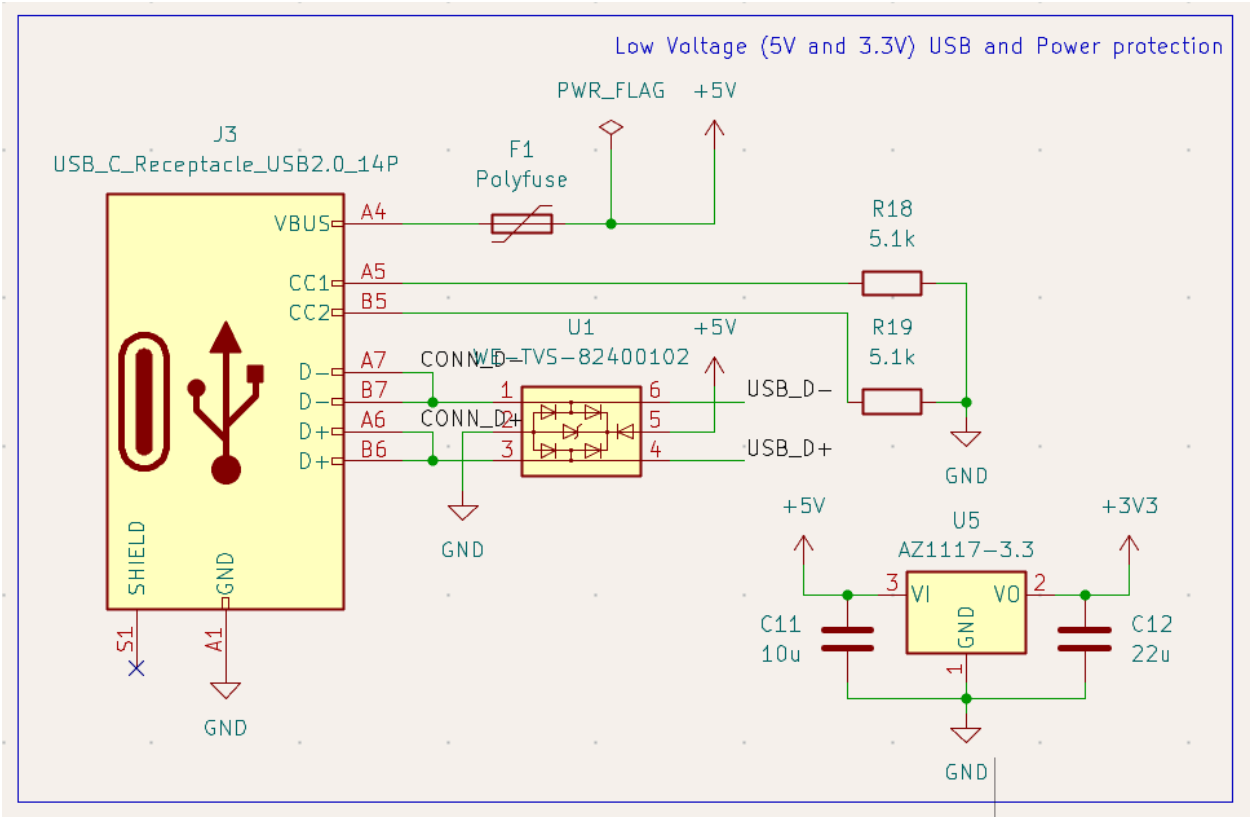

Figure A.1: Data (PC) connection USB C. Also note the LDO Regulator for 5V to 3.3V in the bottom right.

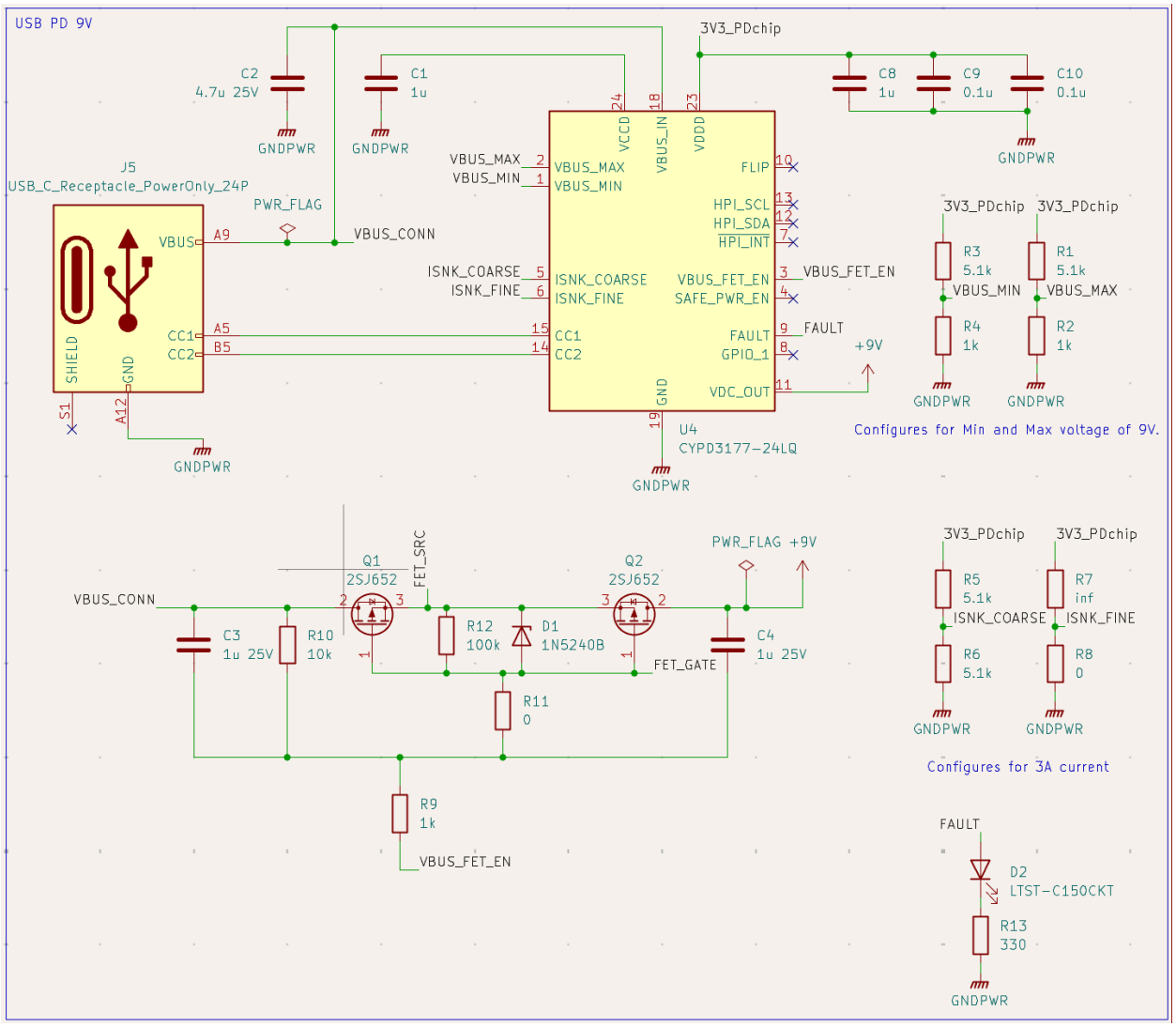

Figure A.2: USB PD circuitry.

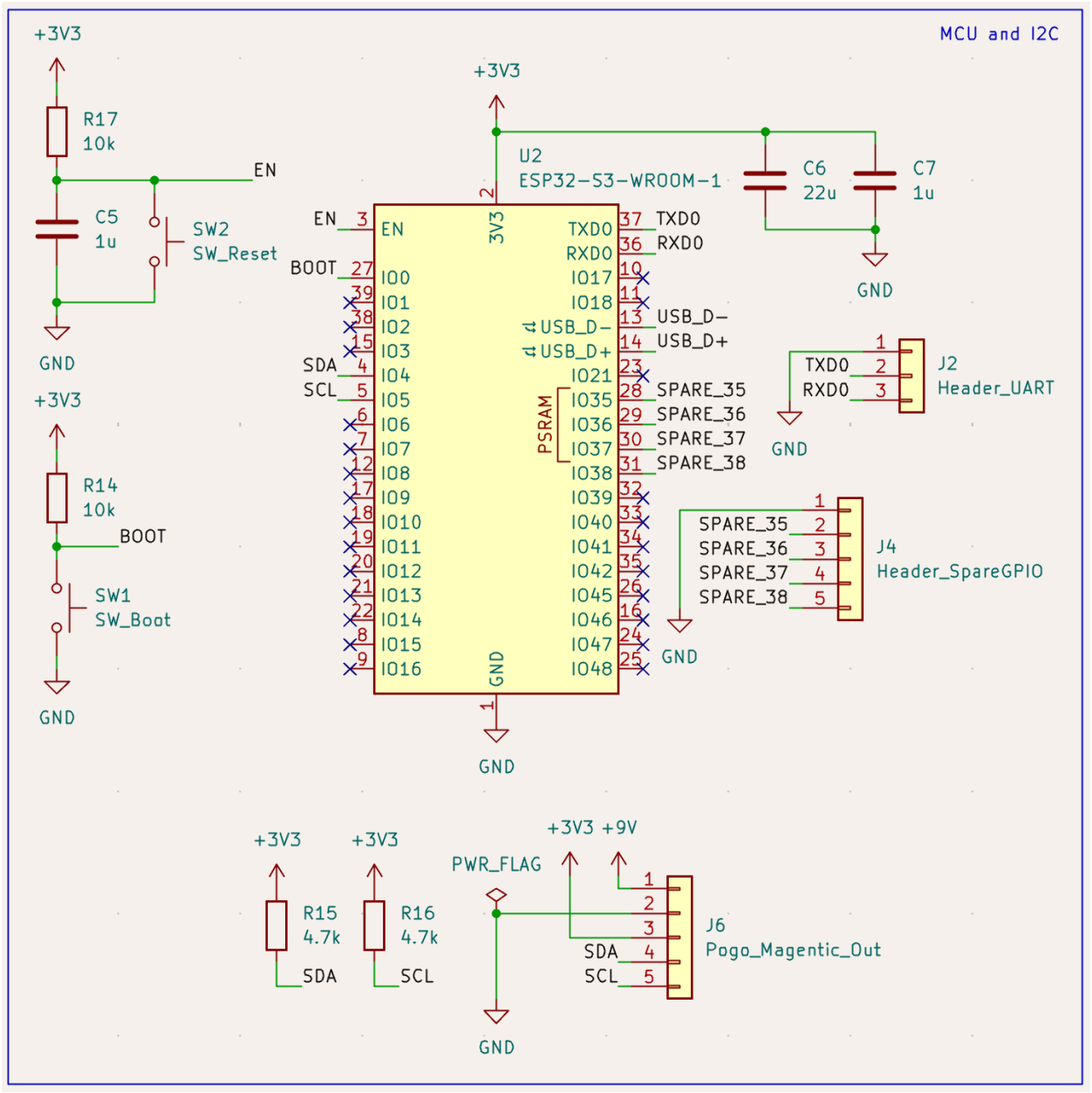

Figure A.3: MCU and connection headers.

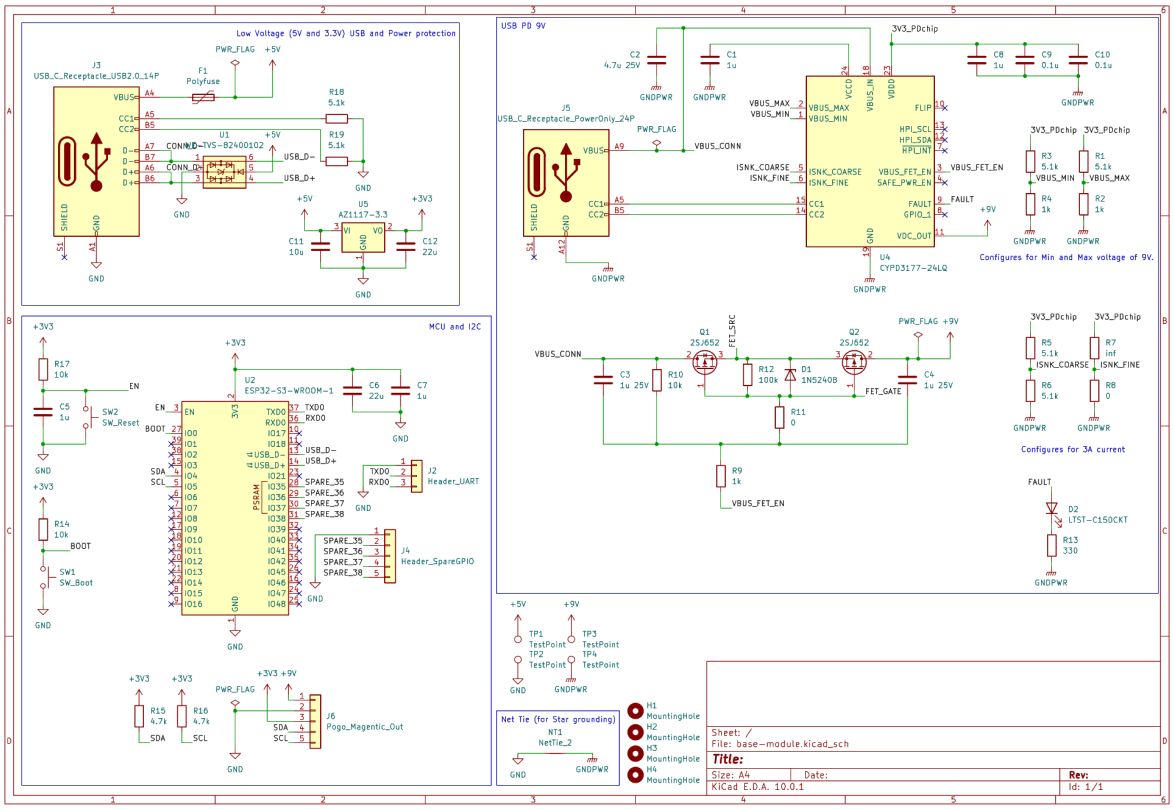

Figure A.4: Full Base module schematic.

Appendix B Fader Module Schematics

Close-up views of KiCad section components. These schematics have been updated for final hardware changes but do not represent the manufactured Fader PCBs themselves.

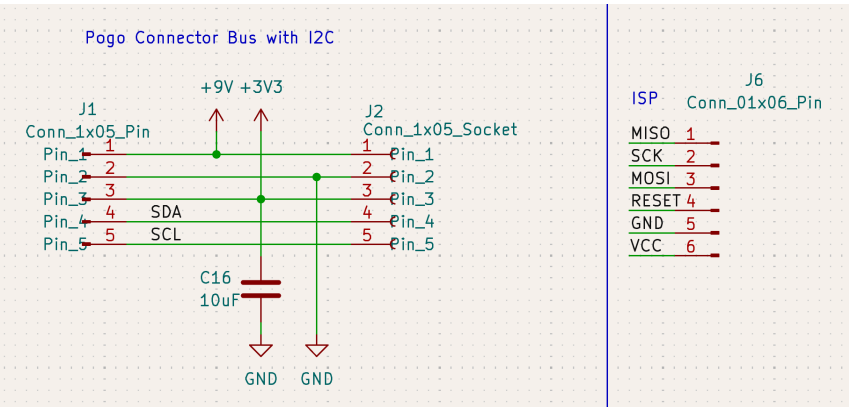

Figure B.1: 5-Pin Connector Bus and ISP Connector

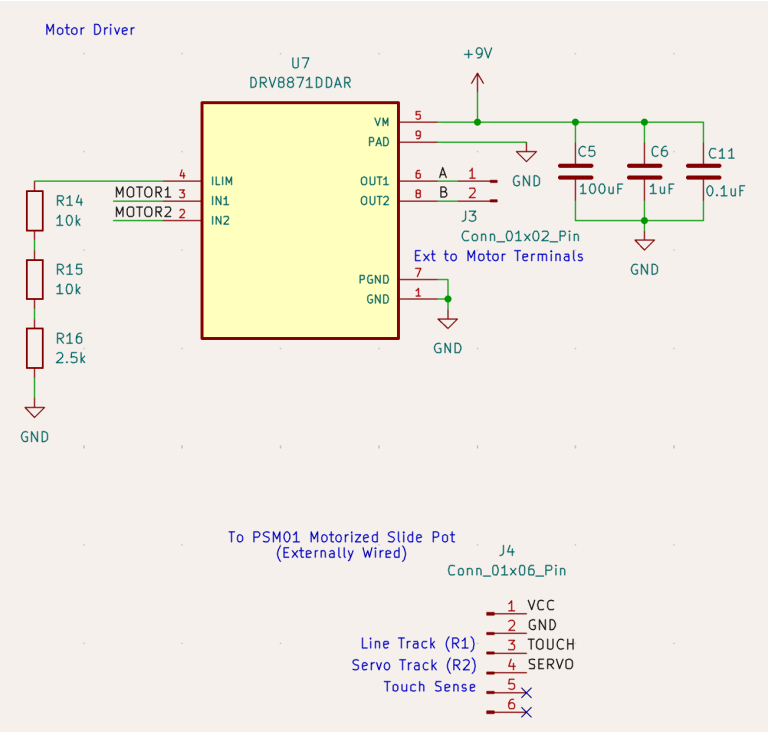

Figure B.2: Motor Driver with Connector to PSM01

TOUCH replaced preexisting Line potentiometer pin.
Previous TOUCH on Pin 5 is now unused.

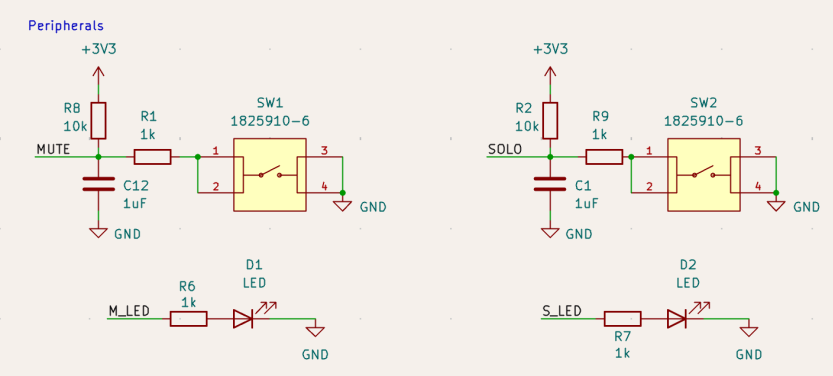

Figure B.3: Mute and Solo Button & LED Peripherals
R1 and R2 have been swapped from 10k to 1k Ω
For proper logic voltages.

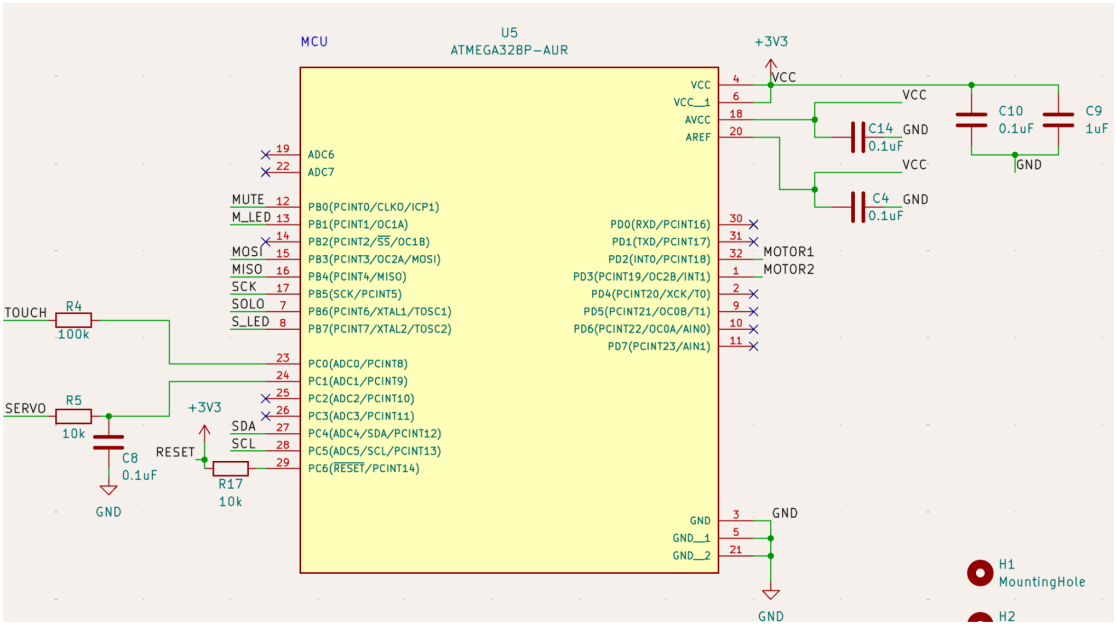

Figure B.4: ATMega328P Pin Connections
TOUCH was exchanged for unused LINE potentiometer connection at Pin 14.
R4 swapped from 10k to 100k Ω.
Filtering capacitor C7 removed at Pin 23,
Pin 14, previously used for TOUCH, disconnected.