Electric Violin Audio Processor

Design Document

ECE 445: Spring 2022

Wei Gao (weigao4)

Alex Seong (aseong2)

Scott Foster (scottbf2)

Date Written: March 6, 2022

TA: Jeff Chang

Team: 22

Contents

1 Introduction

1.1 Problem Statement

Current electric violin pickups tend to fall in one of two categories. Inexpensive pickups are readily available for either acoustic or solid-body violins, but produce a sound quality which is sometimes described as "tinny" or "nasal", and whose harmonic content is too limited for a significant amount of sound design to be carried out. These typically have one piezoelectric sensor for the entire bridge. High-quality pickups produce a "rich" sound with well-balanced harmonic content which is well-suited for use with effects pedals and other sound design tools, but are expensive and often hard to obtain due to low production volume. These typically have at least one sensor for each string.

(The sound quality of pickups is highly subjective. An example of the "nasal" sound of an acoustic violin piezo pickup is demonstrated in [1].)

We have done some prior work with 3D printing electric violin bridges having one sensor for each string. It is difficult to ensure that each string has a similar sound quality or volume by changing the mechanical design of the bridge alone, except by trial and error. Furthermore, the type of strings used can drastically affect the sound of the instrument; for example, steel strings are characteristically "bright" and tinny, while Thomastik Dominant synthetic strings are known for having a "thin"-sounding E string whose timbre contrasts with that of the other three strings. (This is likewise a very subjective assessment, but [2] shows an example of discussion on this topic.)

The sound of the electric violin can also be affected by the properties of the effects chain or sound system, such as the size of the amplifier speaker.

2 Proposed Solution

We will design an audio processor which boosts/attenuates and filters each string of a four-string electric violin individually, then mixes the four string signals to the instrument output jack. Furthermore, the user interface will allow the user to save and recall user-defined "presets" of audio parameters, to account for use with different effects chains or sound systems.

Figure 1 shows the relevant parts of the Mina electric violin. The design is publicly available

online [3]. Figure 2 demonstrates the intended integration of the solution into the violin body, with the processor placed in one of the decorative bouts of the violin.

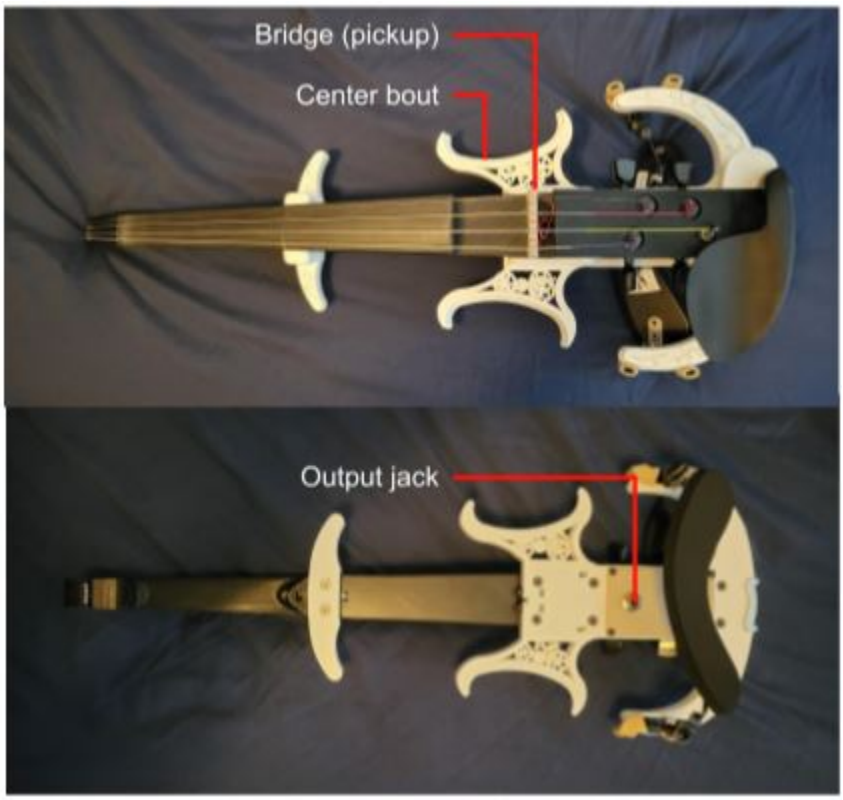

Figure 1: The Mina electric violin

3 High Level Requirements

1. The volume of each string should be adjustable with gain between negative infinity (mute) and +3 dB, and the string signal should be filtered using a bandpass filter with variable bandwidth and center frequency between $100 \pm 2\%$ and $8000 \pm 2\%$ Hz.

2. Two sets of audio parameters (i.e. gain, filter center frequency and bandwidth, and volume) must be able to be saved and recalled as presets using the user interface.

3. The processor must fit in a space of 150x100x50mm (± 1 mm in each dimension), which is roughly the size of the decorative center bout on the Mina electric violin. The PCB should be housed in an enclosure attached to this part of the violin.

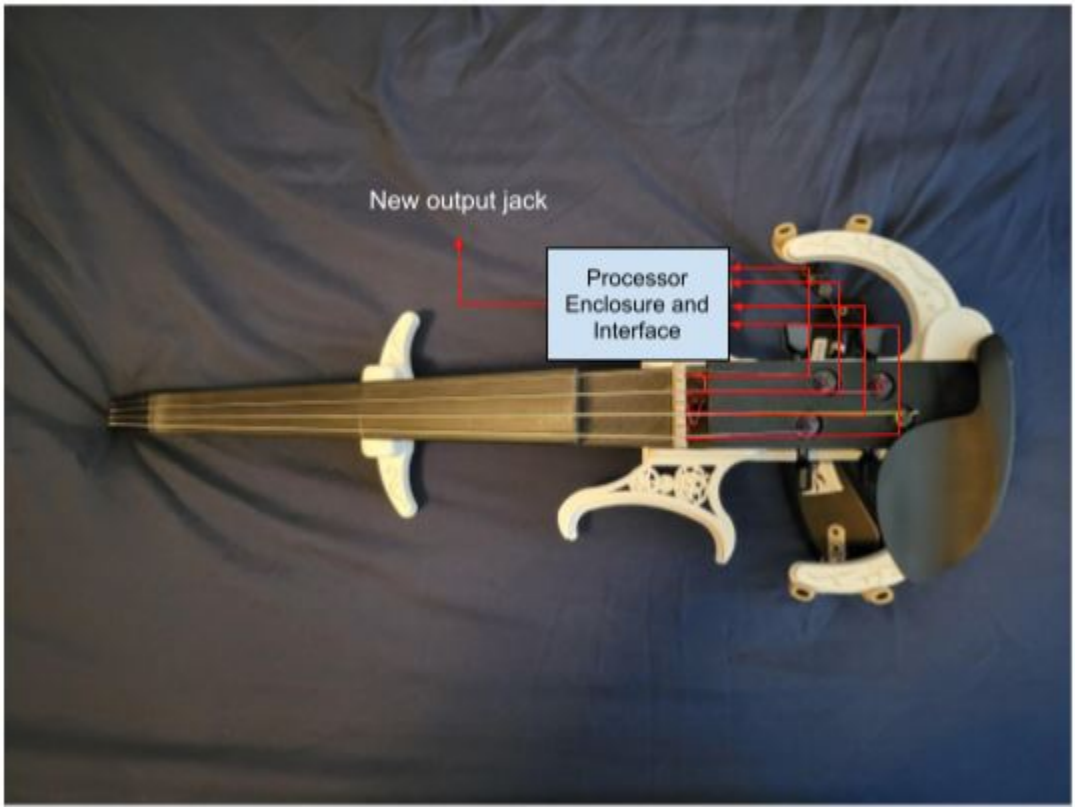

Figure 2: Sketch of proposed solution

4 Design

Figure 3 shows the block diagram for the proposed solution. One digital signal processor handles audio processing for all four strings. Each LED lightbar has eight LEDs. For the sake of brevity, the rotary encoders, pushbuttons, LED indicators, and LED lightbars are only drawn once. However, one rotary encoder and lightbar is to be included for each of the four audio parameters (gain, center frequency, bandwidth, and volume). Likewise, one pushbutton is to be used for string and preset selection. Four LED indicators will be included to indicate the active string (only one will be lit at a time), and the same scheme will be used to indicate the active preset (for a total of two preset indicators). There is also one LED serving as a power indicator. Together, there are 39 LEDs, four rotary encoders, and two pushbuttons in the interface.

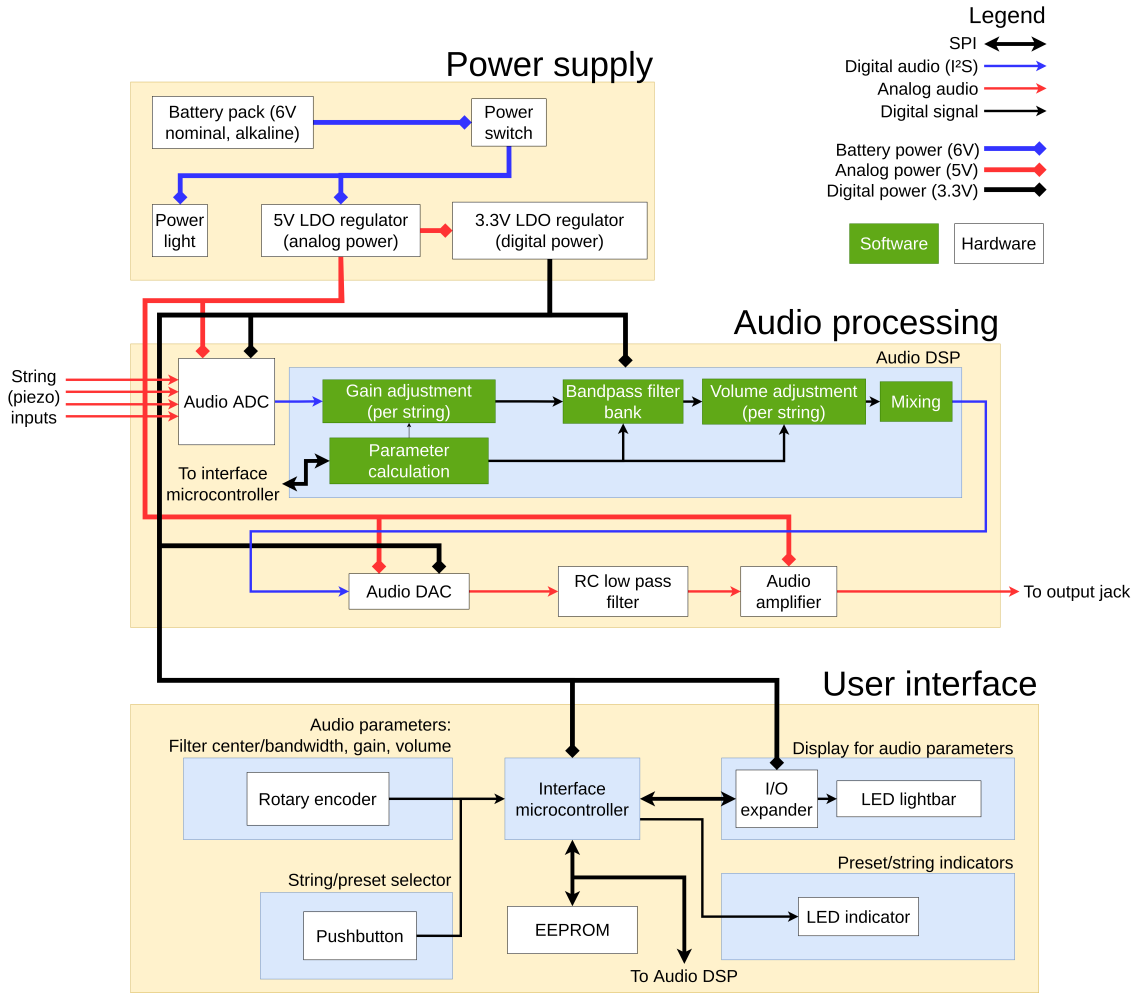

Figure 3: Block diagram of proposed solution

5 Subsystems

5.1 Power Supply

The power supply of the audio processor produces analog and digital supply voltages, to be used in the rest of the processor, from a battery pack mounted on the instrument. The analog power supply is 5V nominal, and the digital power supply is 3.3V nominal. The choice of separate voltages for the analog and digital sections of the design is motivated by the need to isolate the analog circuitry from noise produced, as well as design flexibility for the analog circuitry.

Table 1 shows the requirements and verification steps for this subsystem.

Requirement	Verification steps
The power supply for the audio processor should regulate the 6V nominal battery voltage to a 5 ± 0.25V analog and 3.3 ± 0.17 digital power supply.	1. Connect a 6±0.3V power supply to the battery voltage input (this range simulates fully charged and almost-dead batteries). 2. Verify that the 5V regulator outputs 5 ± 0.25 V on its output voltage pin. 3. Verify that the 3.3V regulator outputs 3.3 ± 0.17 v on its output voltage pin.
The power supply should be able to provide 500±25 mA to all components (assuming 300 ± 15 mA allotted to digital components and 200 ± 10 mA to analog).	1. Turn on all 39 LEDs in the user interface, and program the microcontrollers to do something computationally intensive. 2. Verify that the current drawn from the battery input by the 5V regulator is 300 ± 15 mA.
The power supply should protect from reverse-polarity and overcurrent events. Overcurrent is defined as drawing 1±0.1 A or more from the battery input.	1. Connect a 6±0.3 V power supply to the battery input. 2. Connect a $6 \pm 1\%$ power resistor between the input to the 5V regulator and ground. 3. Verify that the power light turns off, indicating that power is lost to the rest of the audio processor.

Table 1: Requirements and verification for power supply.

5.2 Audio Processing

This subsystem applies gain/volume adjustments and filtering to the input signals from each piezo pickup, then mixes the four string signals to the final instrument output.

For each string, the processor has a gain control prior to filtering which varies between $-\infty$ and $+3(\pm 0.5)$ dB, and the same type of gain control (called "volume") after filtering. The separation of

these gain controls allows more flexibility in the sound design of each string. Piezoelectric sensors generally output a voltage signal around 200 mV_{p-p}. To boost that signal to a level such that it is usable by our DSP chip we make use of a preamp such as the one shown in figure 4. Piezoelectric sensors are able to be represented in a circuit by a voltage source in series with a capacitor, allowing for a preamp circuit that can easily be configured for the signals received from the pickups in the form of an op-amp charge amplifier.

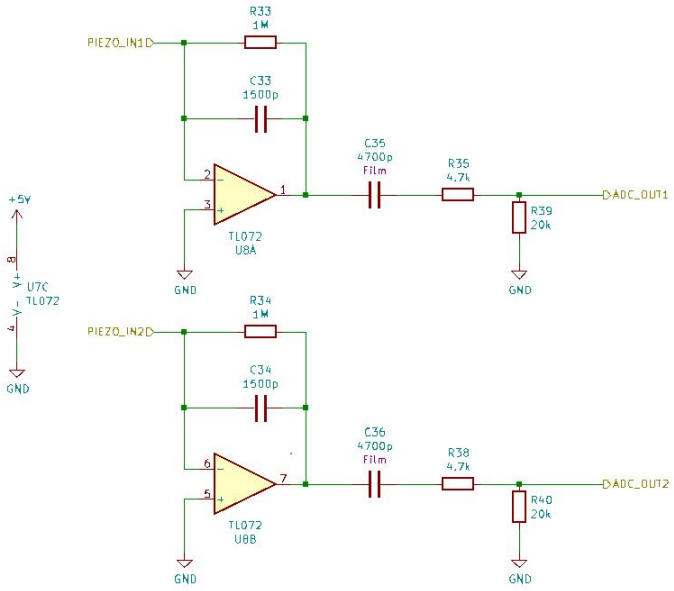

Figure 4: Op-amp Charge Amplifier [4]

For each string, the processor has a bandpass filter with variable center frequency (nominally between 100 and 8000 Hz) and bandwidth. The bounds of the center frequency are based on the frequencies of the open strings as stated in [5]. The open G string has a frequency of 196 Hz and is the lowest note on the string, while the open E string has a frequency 659.3 Hz. The violin can play a few octaves above open E, and so choosing 8000 Hz as an upper bound gives flexibility in filtering these higher notes (and harmonics).

We intend to use the biquadratic bandpass filter implementation from [6]: given the sampling frequency F_s, and the center frequency f_0 and quality factor Q of the filter, we calculate parameters

$$\omega_0 = \frac{2\pi f_0}{F_s}$$

$$\alpha = \frac{\sin(\omega_0)}{2Q}$$

Note that relation of the bandwidth BW with Q is given by

$$\frac{1}{Q} = 2\sinh\left(\frac{\omega_0 \mathrm{BW}\ln(2)}{2\sin(\omega_0)}\right)$$

Then the transfer function of the filter is

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{a_0 + a_1 z^{-1} + a_2 z^{-2}}$$

where $b_0 = -b_2 = \sin(\omega_0)/2$, $b_1 = 0$, $a_0 = 1 + \alpha$, $a_1 = -2\cos(\omega_0)$, and $a_2 = 1 - \alpha$.

The DSP used is the AK7738VQ chip. This may end up being overkill in terms of processing power and peripherals, but due to the low cost and on-board computing power, there is no reason to not use it. It comes loaded with 2 24-bit stereo ADCs, as well as a 24-bit mono ADC. We will also be ablt to utilize the 32-bit DAC. The chip itself is capable of 28-bit floating point calculations at frequencies of up to 48 kHz, leaving more than enough overhead for sampling at a 44.8 kHz frequency. Both of these ICs have more than enough capability to be able to provide the precision and fidelity of signal that we are striving for.

The audio processing microcontroller connects as a SPI bus responder to the interface microcontroller. Due to its many peripherals and GPIO pins, the DSP will simply be able to take in the encoder outputs, as well as the pre-amplified signal(s), and will be able to convert then, filter, analyze, and manipulate the input signals, before converting the resulting waveform back into an analog signal for output.

The mixer should combine the four processed audio signals to be sent to the instrument output jack. The mixer microcontroller should have four audio inputs and one audio output over I^2S, as well as SPI connectivity with the interface microcontroller as a bus responder.

Table 2 shows the requirements and verification steps for this subsystem.

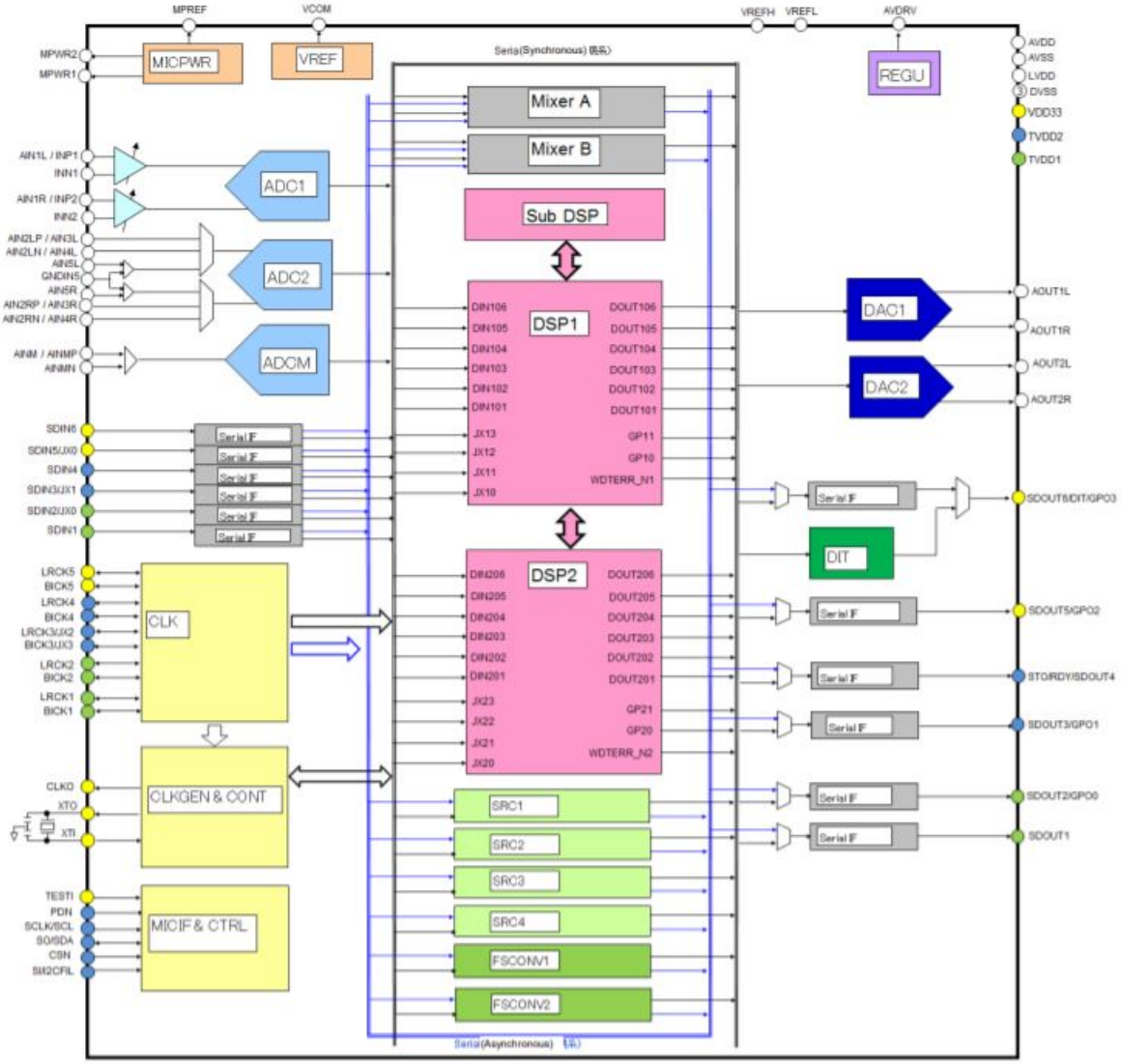

Figure 5: AK7738VQ Block Diagram

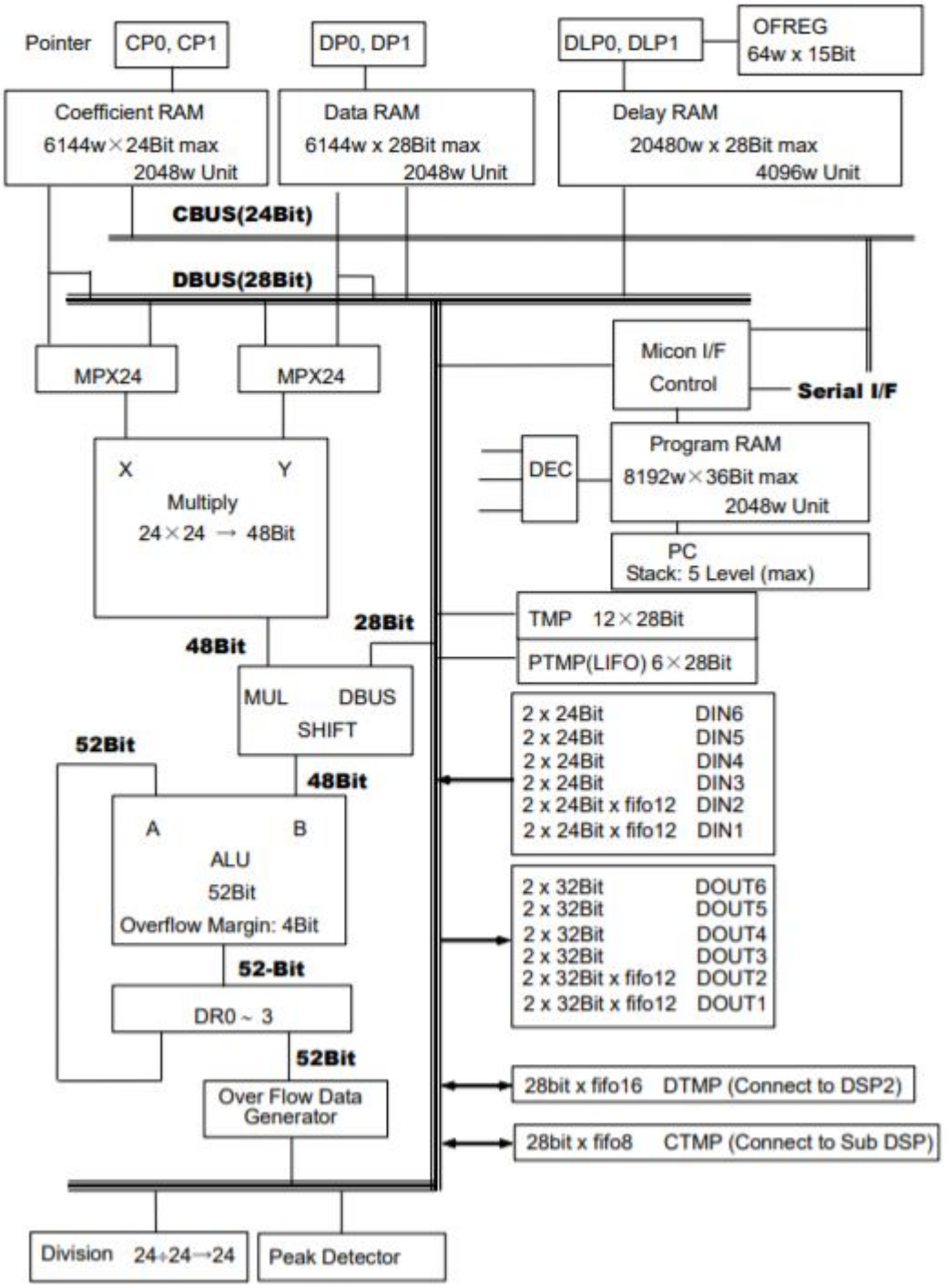

Figure 6: AK7738VQ DSP Blick Diagram

Requirement	Verification steps
The audio processor must have a variable center frequency between $100 \pm 2\%$ and $8000 \pm 2\%$ Hz.	The verification steps for this requirement are summarized in Fig 7. 1. Play a white noise signal into one channel of the audio processor. 2. Configure the channel gain and volume to unity, and the filter to center frequency 100 ± 2 Hz and bandwidth of 1 ± 0.1 octaves. 3. Verify with oscilloscope FFT that the spectral peak of the output signal occurs at 440 ± 2 Hz and the half-power points are 1 ± 0.1 octaves apart. 4. Play a sinusoid signal which sweeps linearly from 100 to 8000 Hz over 5 seconds into the processor. 5. Verify in the time domain that the output amplitude is highest at time $\frac{7900 f_0}{5} \pm 10\%$, where $f_0 = 100 \pm 2$ is the center frequency. 6. Repeat these steps for 440 ± 9 and 8000 ± 160 Hz center frequency.
The audio processing must introduce latency of no more than 100 ± 10 milliseconds between the input and output audio streams.	1. Play an impulse (click) into a channel of the audio processor. 2. Verify on an oscilloscope that the impulse response at the output of the processor begins no later than 100 ± 10 ms after the impulse input begins. The beginning of a signal is defined as the first time when the signal amplitude reaches 5% of its peak value.

Table 2: Requirements and verification for audio processing.

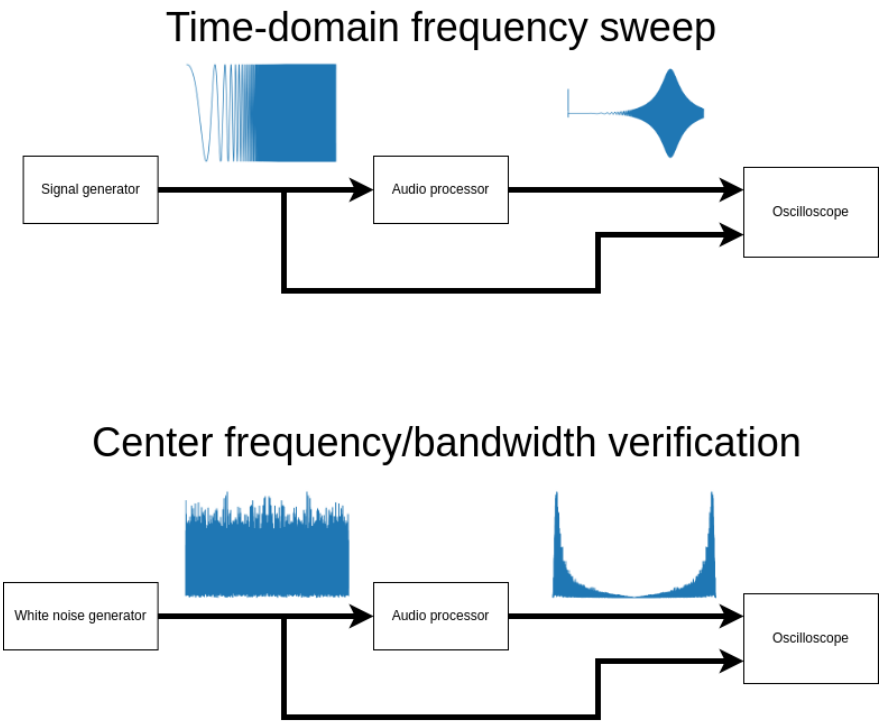

Figure 7: Verification of filter center frequency, bandwidth, and frequency range.

5.3 User Interface

This subsystem provides controls for the user to change the gain, filter center frequency, and volume for each string, and to save combinations of these parameters as presets. To do this, the interface should have rotary encoders to control the gain, filter center frequency, filter bandwidth, and volume of the active string. A sketch of how the user interface will be designed is displayed in Fig 8.

The interface will also have a button to rotate between the active string, and four LED indicators to show which string is active. Similarly, a button is included to rotate between the active preset, and two preset indicators. To enter the preset mode, the user will turn on a toggle switch for the presets.A set of 4 light bars with 8 lights each should be included to indicate the current intensity of each parameter.

The interface should have an external SPI memory for saving presets, and be able to communicate with the audio DSP over SPI as a bus controller. The interface firmware should save the current audio parameters to memory when the active string or preset is changed, and one second after the user changes any of the audio parameters by rotating the encoders.

Table 3 shows the requirements and verification steps for this subsystem.

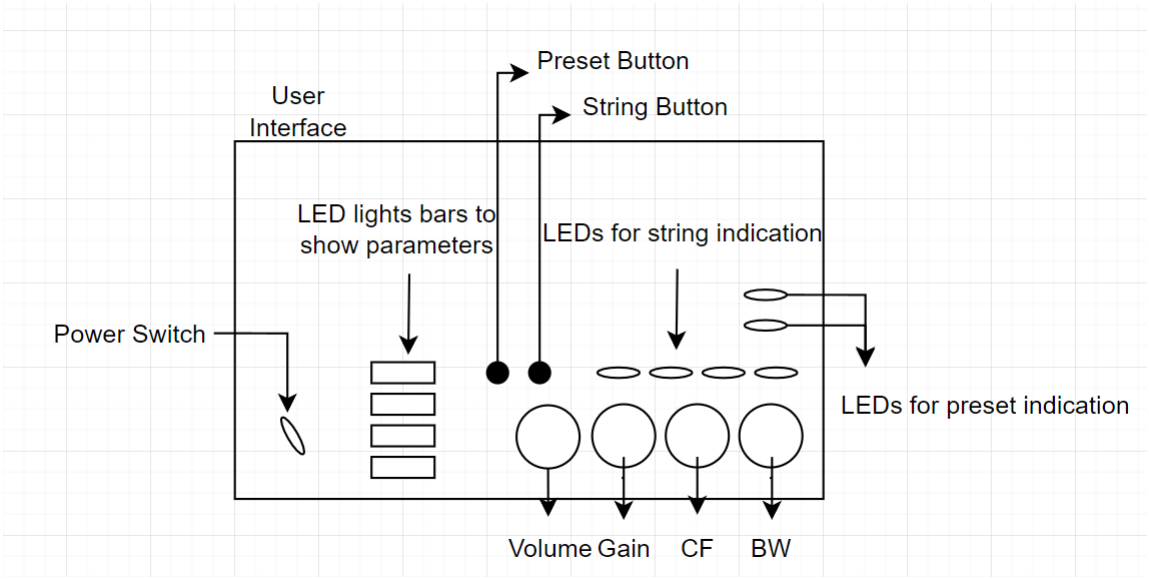

Figure 8: Sketch of proposed user interface

Requirement	Verification steps
The rotary encoders must be able to control the gain, filter center frequency, and the volume of each string.	1. Slowly turn each knob while monitoring the output signal to verify that the output signal actually matches with the implemented range. 2. Verify with the status LED bars that will light up more on the light bars as each control parameters increase with the turning of each rotary encoder.
The button to switch between strings must output correct signal to activate the right string.	1. Connect the button to LED indicators and verify that the each LED light up in correct order on each push of the button
The button to switch between presets must load the correct preset that is saved in the SPI external memory.	1. Save the two presets with extreme parameters. For example, a full gauge of each parameters for the first preset and zero gauge for the second preset. 2. Connect the button to LED light bar indicators and verify that the light bars fully lights up and turns off on each push of the button. 3. Verify that the two LEDs for the preset notification lights up accordingly on each push of the button.

Table 3: Requirements and verification for user interface.

6 Cost and Schedule

6.1 Cost

Table 4 shows the estimated cost of components for the current power supply and user interface design (not including the cost of the enclosure and circuit board). The audio processor design is still tentative, so the parts for it are not included.

The prices are from Digikey and are current as of Feb. 23, 2022. Some of the unit prices are at higher price breaks, reflecting our intent to purchase more than is needed for a single prototype. This is to prevent us from becoming bottlenecked from losing or damaging components during assembly and testing.

The estimated cost of labor for us is given in Table 5, on the assumption that the total price of labor is 2.5 times the hourly wage times the number of hours worked to design and assemble the prototype. As the final layout of the user interface is not yet determined, we are currently unable to estimate the cost of labor for the machine shop.

Item	Part no.	Unit cost ($)	Qty	Extended cost ($)
0.1 uF capacitors	06035C104KAT2A	0.0239	21	0.502
10 uF capacitors	106BPS050M	0.293	2	0.586
2-pin keyed header (pins)	LHA-02-TS	0.074	3	0.222
2-pin MTA connector (sockets)	3-641535-2	0.212	3	0.636
10-pin JTAG header	3221-10-0100-00	0.615	1	0.615
270 Ω LED resistors	ERJ-3GEYJ271V	0.0162	39	0.632
10 kΩ resistors	RMCF0603JG10K0	0.0061	14	0.085
1 A resettable polyfuse	0ZCJ0050AF2E	0.177	1	0.177
P-channel MOSFET	DMP2066LSN-7	0.433	1	0.433
3.3V regulator	XC6227C331PR-G	0.833	2	1.666
5V regulator	NCP1117IDT50T4G	0.593	1	0.593
EEPROM	BR25H010FVT-2CE2	0.482	1	4.82
Microcontroller	MKL43Z128VLH4	7.11	1	7.11
I/O expander	MCP23S08-E/SO	1.40	4	5.60
Amber LEDs	IN-S42BT5A	0.115	39	4.49
Panel-mount buttons	PS1023ABLK	2.01	2	4.02
Tactile pushbutton	PTS636 SM50 SMTR LFS	0.118	1	0.118
Rotary encoders	EN12-HN22AF25	0.938	4	3.75
Multi DSP with ADC and DAC	AK7738VQ	12.40	1	12.40
Op-amp	TL072HIDDFR	0.50	2	1.00
Total	-	-	-	49.46

Table 4: Preliminary component list with costs.

Contributor	Hourly wage ($)	Labor hours (estimated)	Total labor cost ($)
Wei	40	210	8400
Alex	40	220	8800
Scott	40	200	8000
Total	120	630	25,200

Table 5: Estimated labor cost for the three designers.

6.2 Schedule

A schedule for the design of our prototype audio processor is given in Table 6.

Task	Assignee	Due date
Complete Audio DSP schematic	Scott	2022-02-24
Complete PCB layout	Scott, Wei	2022-02-28
Create interface mechanical layout	Wei	2022-03-05
Design user interface firmware	Alex	2022-03-12
Design audio DSP firmware	Scott	2022-03-12
Board assembly	Wei	2022-03-23
Validate power supply	Wei	2022-03-25
Validate user interface	Alex	2022-04-01
Validate audio processing	Scott	2022-04-01
Integrate prototype	Everyone	2022-04-09
Test integrated prototype	Wei	2022-04-16

Table 6: Schedule for completing the project.

7 Tolerance Analysis

Latency Latency is a huge problem when it comes to real-time DSP, especially if the signals being processed require heavy computation. The latency for the ADC and DAC components are the most meaningful when finding tolerances for the delay. The latencies for both the ADCs and DACs are relative to the sampling frequency. From the AK7738VQ datasheet, the latency value for the ADC is determined by Latency $= \frac{5}{F_s}s$ where $F_s = 48$ kHz, therefore

$$Latency = 0.104ms$$

While the same process for the DAC results in

$$Latency = \frac{6.6667}{48\,kHz} = .139ms$$

Due to the publicly stated value of 10 ms being the threshold at which latency becomes noticeable, utilizing the maximum number of ADCs and DACs available on our chip;

$$Latency_{max} = \frac{10\,ms}{3_{ADC} + 1_{DAC}} = \frac{10}{4} = 2.5\,ms$$

Proving that even with the most processing done on the signals through the multiple ADCs and DACs, the latency is still unnoticable.

Sampling The sampling rate we require for our signal fidelity and integrity to meet our expectations is that of 44.8 kHz. Our usable sampling frequencies are much higher, but the common sampling frequency of 48 kHz would allow for tolerances of

$$\frac{48 - 44.8}{48} = 0.06667$$

allowing for a tolerance range of $\pm 6.667\%$ in terms of relative frequencies, much greater than our needed tolerance range of $\pm 2\%$.

8 Ethics and Safety

This project presents few safety concerns. All voltages used are 6V or less, using alkaline battery chemistry. Once placed in an enclosure, there would be no meaningful contact between the user and the circuits inside. We have included a reverse polarity protection MOSFET in the power supply design to guard against the possibility of a user installing batteries backward. All capacitors are used are either electrolytic or ceramic and will create an open circuit in the event of a failure.

This project presents few significant ethical concerns, although following section 1.5 of the ACM Code of Ethics [7], we must acknowledge that similar prior work exists. Notably, the Strados electric violin, created by ZETA Violins [8], uses an internal active preamplifier system which allows the volume of each string and the overall gain to be adjusted manually [9]. ZETA calls this pickup system "patented" on their website, but does not list a patent number; additionally, we could not find a relevant patent upon searching for "ZETA violin" in Google Patents. Therefore, we cannot immediately confirm whether this patent exists, let alone if the product is still under patent protection. This may pose an obstacle if we choose to commercialize the project in the future.

Our pickup design is inspired by Richard Barbera's design, "Resonant pick-up system" [10]. The patent expired over a decade ago, and since our bridge is not carved of wood, it is unlikely that our design would be infringing on this patent anyway.

References

[1] H. Reich, *I Had to Build a Custom Mute Switch for my Violin.* YouTube, Jun 2019. [Online]. Available: https://www.youtube.com/watch?v=oYsp7OIMFAs

[2] S. Tsuchiya, "What's wrong with dominant e?" Jun 2008. [Online]. Available: https://www.violinist.com/discussion/archive/14090/

[3] J. Axelsson, "The mina violin (electric) by jaxelsson," Feb 2017. [Online]. Available: https://www.thingiverse.com/thing:2122773

[4] "endolith", "Electric violin." [Online]. Available: http://www.endolith.com/wordpress/2007/10/13/electric-violin/

[5] C. R. Nave, "The violin." [Online]. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/Music/violin.html

[6] R. Bristow-Johnson, "Rbj audio-eq-cookbook," May 2005. [Online]. Available: https://www.musicdsp.org/en/latest/Filters/197-rbj-audio-eq-cookbook.html

[7] "Acm code of ethics and professional conduct," Jun 2018. [Online]. Available: https://www.acm.org/code-of-ethics

[8] "Strados modern - zeta violins: Electric violins cello bass: Zeta mandolins: Pickups repairs," Feb 2016. [Online]. Available: https://zetaviolins.com/strados-modern

[9] "Emg mxrp-5 internal preamp for zeta violins," Jan 2021. [Online]. Available: https://www.electricviolinshop.com/emg-mxrp-5-preamp

[10] R. Barbera, "Resonant pick-up system," Sep 1989, uS4867027A.

A Circuit Schematics

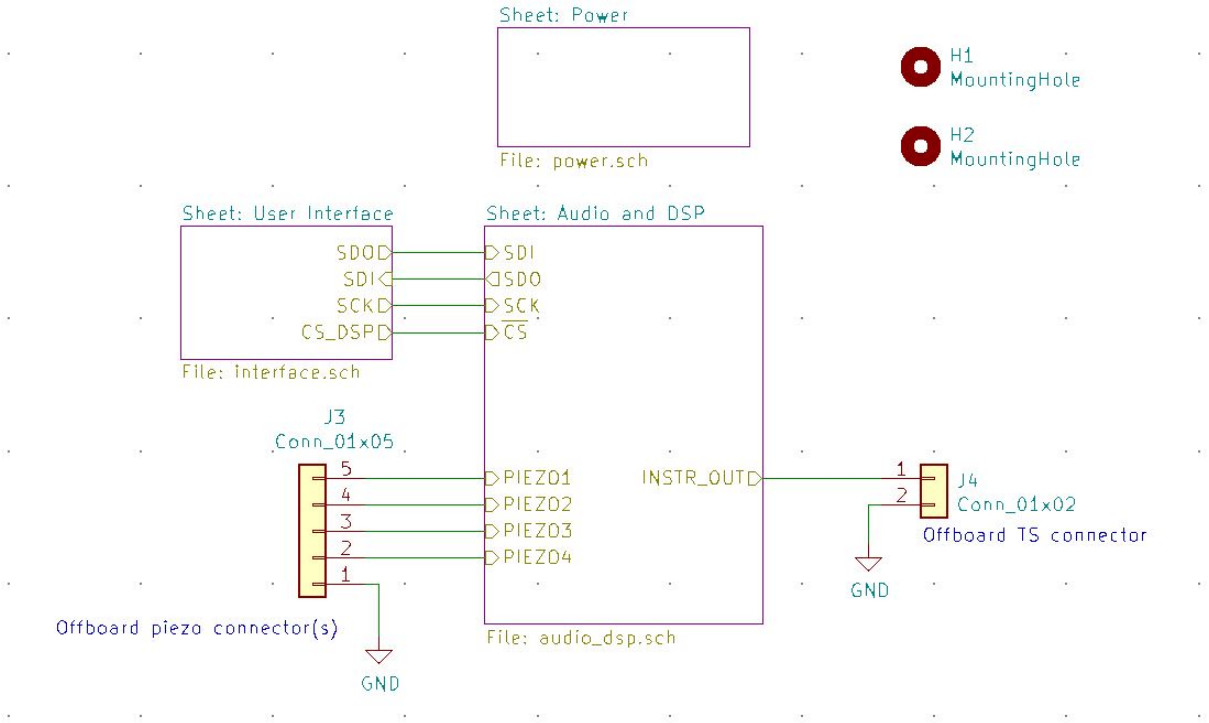

Figure 9: High-level subsystems.

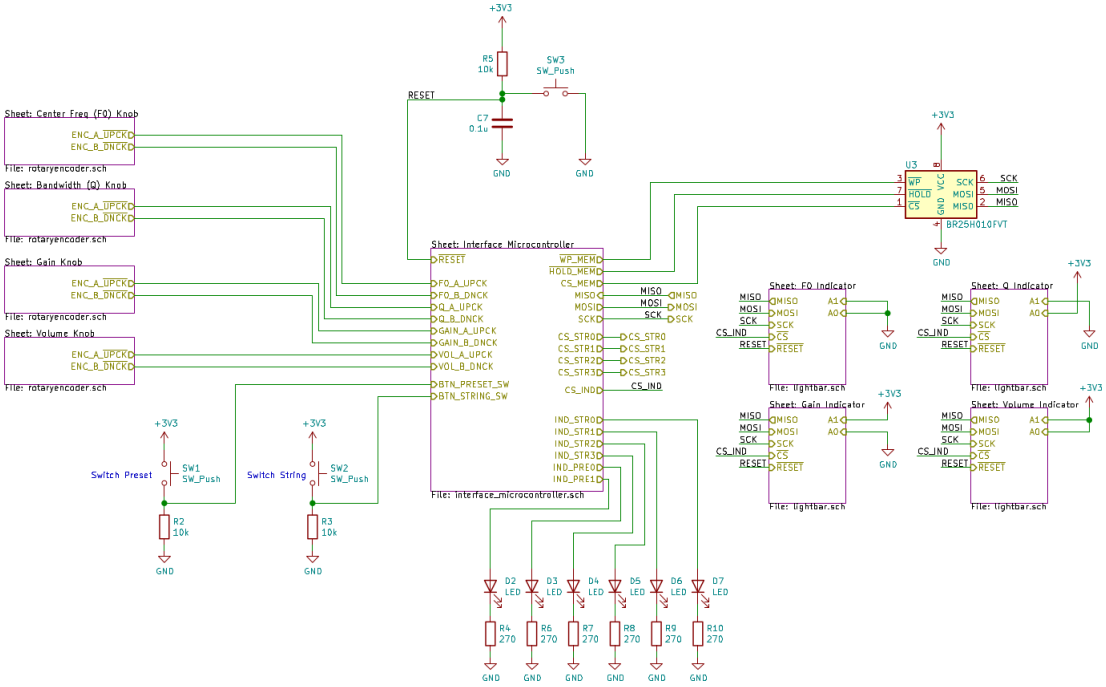

Figure 10: User interface overview.

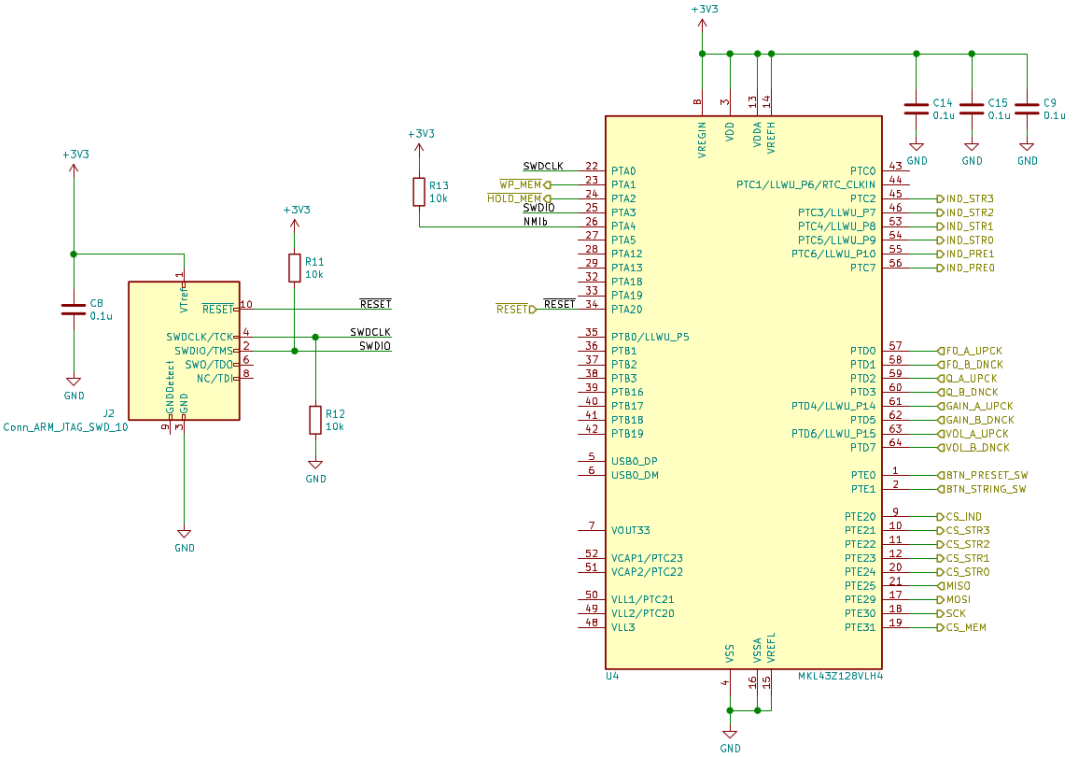

Figure 11: User interface microcontroller.

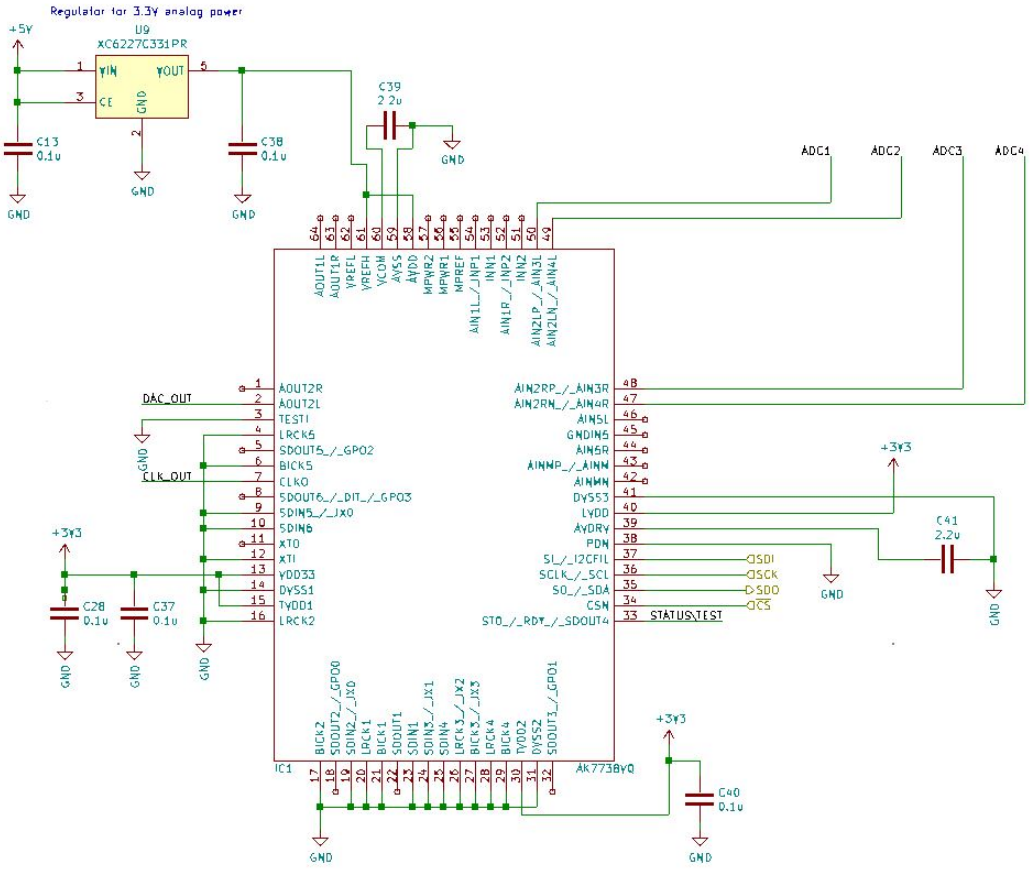

Figure 12: AK7738VQ DSP Circuit

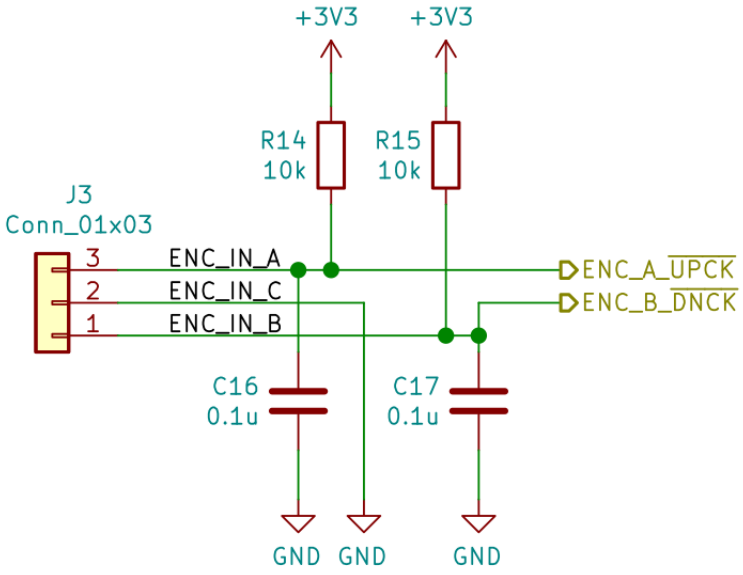

Figure 13: Rotary encoders and quadrature decoder.

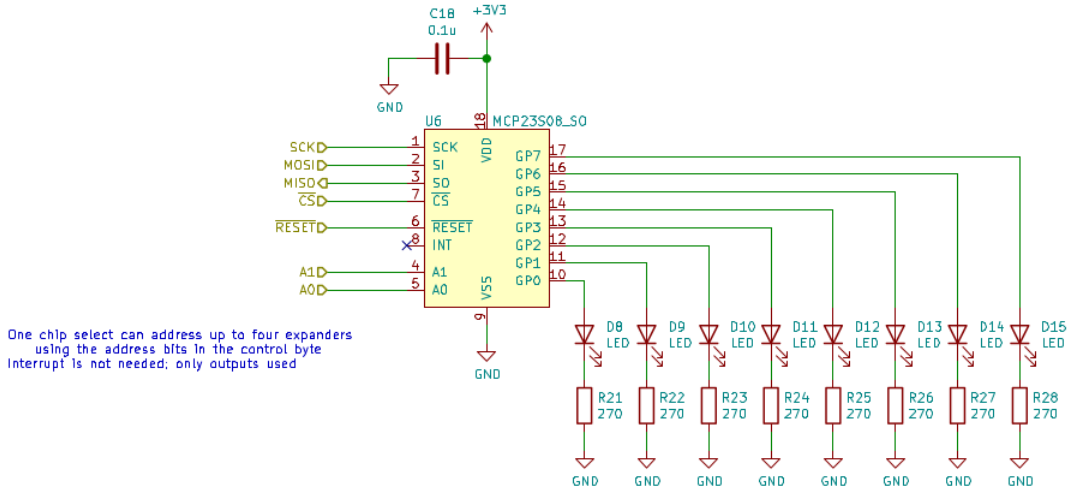

Figure 14: Lightbar indicator, used for audio parameters.

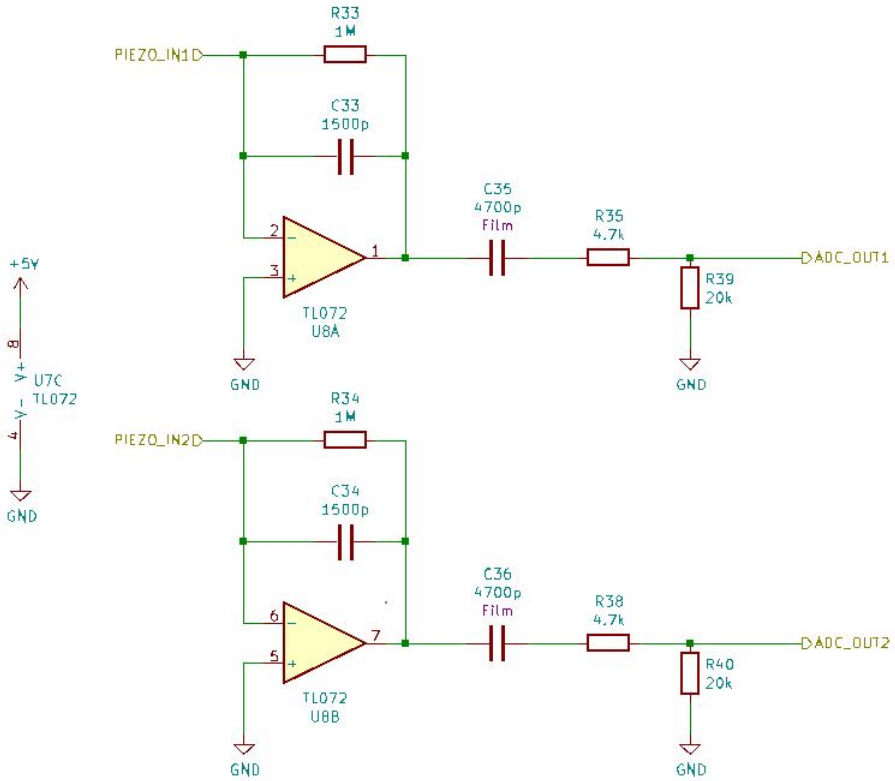

Figure 15: Preamp circuit schematic

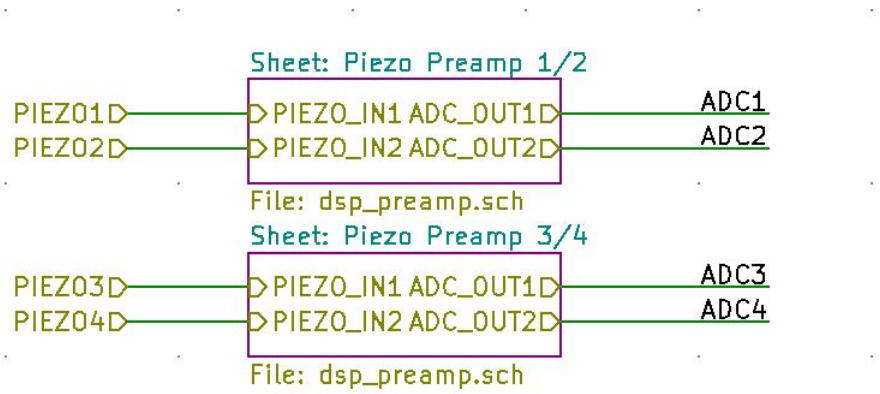

Figure 16: Modularized preamp circuits showing inputs and outputs

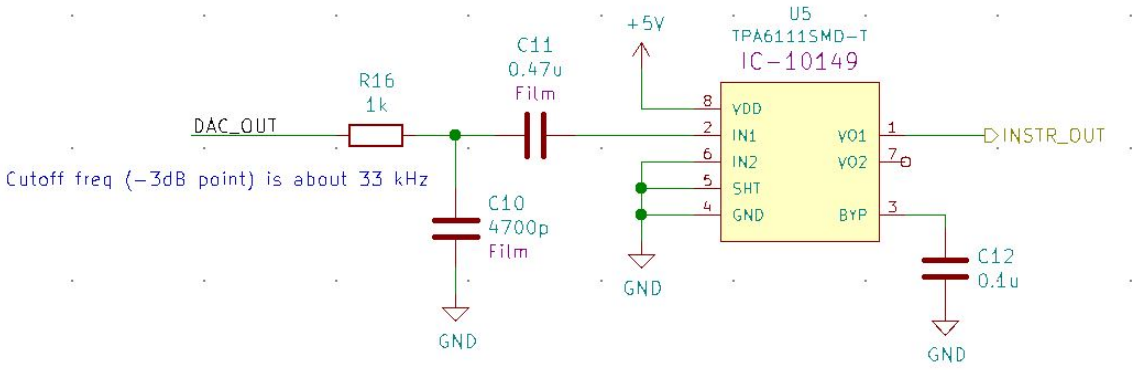

Figure 17: DAC output circuit

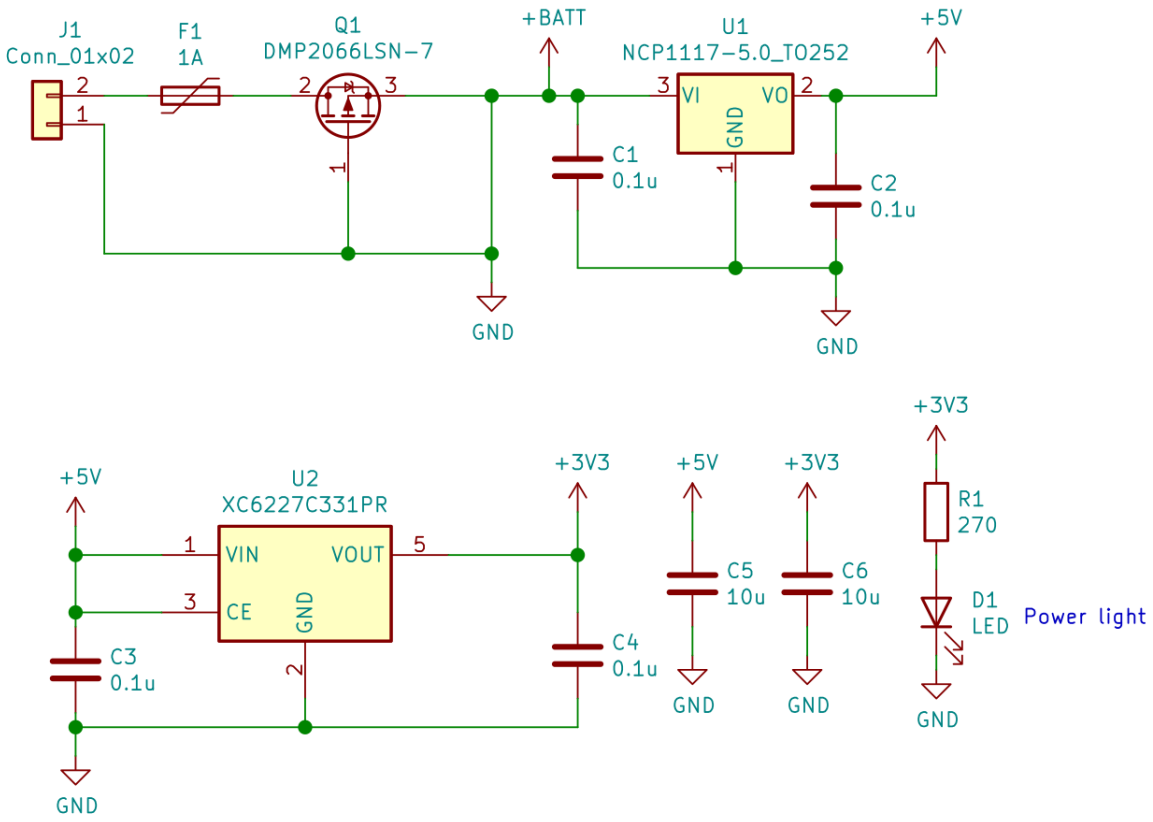

Figure 18: Power supply.

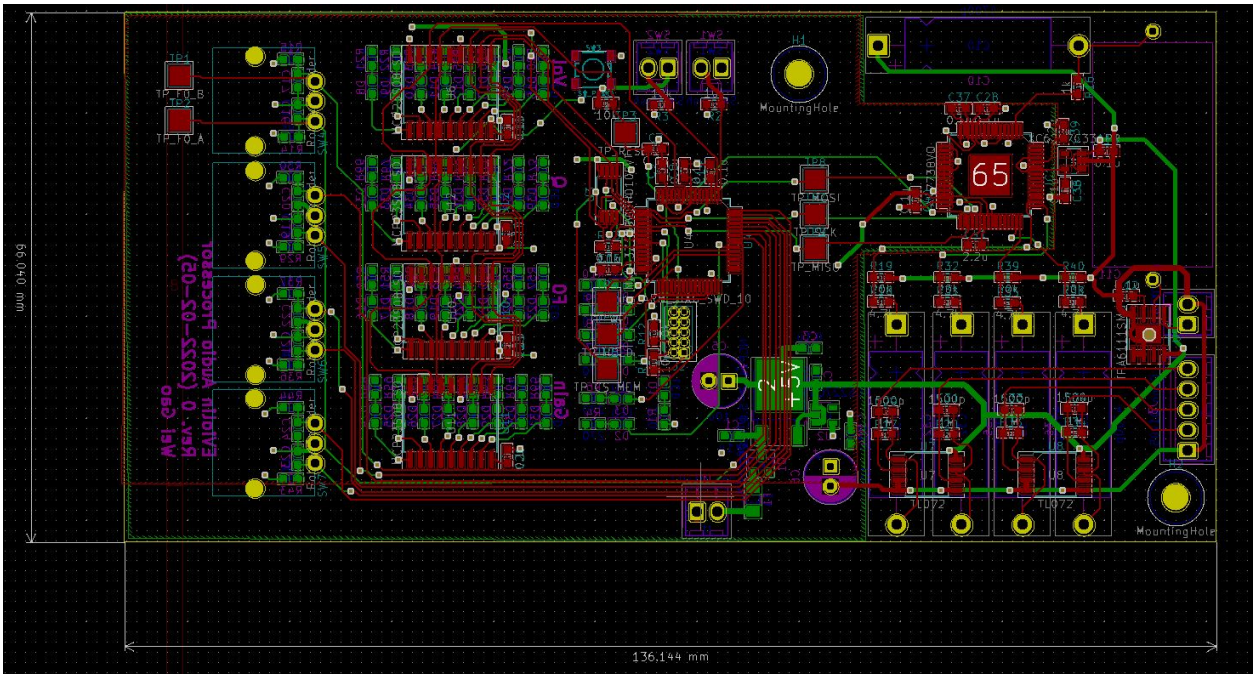

Figure 19: PCB including components and routing