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.)

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; the processor is to be housed in an enclosure attached to the right center bout.)
Figure 1: The Mina electric violin
2 High Level Requirements

1. Be able to adjust the volume of each string using a gain between negative infinity (mute) and +3 dB, and filter the string signal using a bandpass filter with variable bandwidth and center frequency between 100 and 6000 Hz.

2. Be able to save, overwrite, and recall two sets of audio parameters (gain, filter center frequency and bandwidth, and volume) using the user interface.

3. Fit in a space of 150x100x50mm, 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.
3 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.
4 Subsystems

4.1 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.

4.1.1 Subsystem requirements and verification

Supply voltages. The power supply for the audio processor should regulate 6V nominal battery power to 5V (±0.25) analog power and 3.3V (±0.17) digital power.

Verification steps. To verify this, the battery should be connected to the power supply a digital multimeter should be connected to the output of the 5V analog voltage regulator, and the reading be checked to be 5 ± 0.25V. The multimeter probe may then be moved to the output of the 3.3V digital regulator, and the reading be checked to be 3.3 ± 0.17V.

Supply current. The power supply should provide enough current for all components, with some overhead. In particular, we have chosen to use a total of 39 LEDs in the user interface, each rated to draw 5 mA when on. In the worst case when all LEDs are on, they will require 195 mA. (The interface microcontroller and audio DSP will also require their share of current, but this will likely vary with computational load.) Therefore, the digital portion of the power supply should be able to supply 300mA (±15) of current in total to the other subsystems, and so the analog portion should supply at 500mA (±25).

Verification steps. To verify this, the interface processor should be programmed to switch on all 39 LEDs in the user interface, then perform a busy-wait routine to add computational load. The audio DSP should be similarly commanded to busy-wait. A digital multimeter should be connected between the battery positive terminal and the positive input of the power supply, and the current be measured. The current should be checked to be less than 315 mA.

Protective features. The power supply should include reverse-polarity protection, and also shut down the audio processor if an overcurrent event, defined as 1 ± 0.1 A, is detected flowing from the
battery.

Verification steps. This should be verified by “shorting” the battery voltage rail through a $6\Omega \pm 1\%$ power resistor. The power light at the end of the power supply stages should shut off when the resistor is connected, indicating that the overcurrent protection has activated.

4.2 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 should have a bandpass filter with variable center frequency (between 100 and 2000 Hz) and bandwidth. Also, the processor should have a variable gain control both before (“gain”) and after (“volume”) the filter stage.

The audio processing microcontrollers should have SPI connectivity with the interface microcontroller as a bus responder.

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 I²S, as well as SPI connectivity with the interface microcontroller as a bus responder.

4.2.1 Subsystem requirements and verification

The requirements and verification steps for the audio processing subsystem are given in Table 1.
The audio processor must have a variable center frequency between 100 and 8000 Hz.

The verification steps for this requirement are summarized in Fig 4.

While playing a white noise signal into the audio processor, configure the filter to have unity gain/volume, a center frequency of 440 Hz, and bandwidth of 1 octave using the interface knobs. Using an oscilloscope to probe the output signal of the processor, verify the setting is correct by checking that the spectrum of the output has a peak at 440 ± 1 Hz and bandwidth of 1 ± 0.1 octaves.

Then, use a signal generator to play a sine wave whose frequency sweeps linearly from 100 to 8000 Hz over the course of 5 seconds. Using an oscilloscope to monitor the time-domain output signal of the audio processor, verify that the output amplitude is highest at time $\frac{7900}{5} f_0$, where $f_0 = 440 \pm 1$ Hz is the center frequency.

The audio processing must introduce latency of no more than 100 milliseconds between the input and output audio streams.

Connect a signal generator to one of the processor inputs, then play a click (impulse) into the processor. Monitor the time-domain signal at both the input and the output of the processor using an oscilloscope. Verify that the impulse response measured at the output begins no later than 100 ± 10 ms from the beginning of the impulse signal at the input.

The beginning of the impulse response is defined as the first time at which the output signal from the processor exceeds 20% of the maximum amplitude of the impulse signal.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>The audio processor must have a variable center frequency between 100 and 8000 Hz.</td>
<td>The verification steps for this requirement are summarized in Fig 4. While playing a white noise signal into the audio processor, configure the filter to have unity gain/volume, a center frequency of 440 Hz, and bandwidth of 1 octave using the interface knobs. Using an oscilloscope to probe the output signal of the processor, verify the setting is correct by checking that the spectrum of the output has a peak at 440 ± 1 Hz and bandwidth of 1 ± 0.1 octaves. Then, use a signal generator to play a sine wave whose frequency sweeps linearly from 100 to 8000 Hz over the course of 5 seconds. Using an oscilloscope to monitor the time-domain output signal of the audio processor, verify that the output amplitude is highest at time $\frac{7900}{5} f_0$, where $f_0 = 440 \pm 1$ Hz is the center frequency.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>The audio processing must introduce latency of no more than 100 milliseconds between the input and output audio streams.</td>
<td>Connect a signal generator to one of the processor inputs, then play a click (impulse) into the processor. Monitor the time-domain signal at both the input and the output of the processor using an oscilloscope. Verify that the impulse response measured at the output begins no later than 100 ± 10 ms from the beginning of the impulse signal at the input. The beginning of the impulse response is defined as the first time at which the output signal from the processor exceeds 20% of the maximum amplitude of the impulse signal.</td>
</tr>
</tbody>
</table>

Table 1: Requirements and verification for audio processing.
Figure 4: Verification of filter center frequency, bandwidth, and frequency range.
4.3 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 5.

The interface should also have a button to rotate between the active string, and four LED indicators to show which string is active. Similarly, a button should be included to rotate between the active preset. To enter the preset mode, the user will turn on a toggle switch for the presets. A set of 4 light bars with 32 bars on each should be included to indicate the current intensity of each parameter.

The interface should have an external SPI EEPROM 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.

4.3.1 Subsystem requirements and verification

The requirements and verification steps for the user interface subsystem are given in Table 2.
The rotary encoders must be able to control the gain, filter center frequency, and the volume of each string.

Verification steps: Slowly turn each knob while monitoring the output signal to verify that the output signal actually matches with the implemented range. Additionally, 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.

Verification steps: Connect the button to LED indicators and verify that 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 EEPROM.

Verification steps: 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. Then, 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. Additionally, verify that the two LEDs for the preset notification lights up accordingly on each push of the button.

Table 2: Requirements and verification for user interface.

![Figure 5: Sketch of proposed user interface](image-url)
5 Tolerance Analysis

The project as a whole is very complicated, similar subjects have been written about to earn the author their PHD. When it comes to DSP, to complete the process accurately and efficiently can be very complicated. Many equations will have to be taken into account for our design to be able to correctly implement our desired features. Our parameters and algorithmic functions will have to be carefully calculated while still allowing for tolerances due to the nature of the project and the requirements imposed. The initial sensors as well as the ADC will both be required to be heavily tested and fine tuned to allow for the signals that are being processed to be as true to the original sound as possible. There is the inherent risk of lots of signal noise or crosstalk between the strings, resulting in convoluted signals. The real time DSP process is also quite complex, especially when utilizing an approach where there are more avenues for hardware failure due to the multiple different ICs that will be used. The algorithms to complete such a process have been heavily researched [4–7], but as our implementation aims to progress electric violin design and sound creation/emulation, there are the possibility of pitfalls. While the entire project has difficult problems throughout each subsection, the high-tier, efficient implementation of a modular approach to DSP of each string individually will be of utmost importance.

The feasibility and functionality of such a design can be proven, no matter the outcome of the overall design process, through mathematical analysis and simulation of the algorithms and methods we are going to implement.

Continuous convolution where \( x(t) \) is the response of the violin string and \( h(t) \) is a linear behavior to produce \( y(t) \)

\[
y(t) = \int_{0}^{\infty} h(\tau)x(t - \tau) \, d\tau
\]

Rewritten in discrete space as

\[
y[n] = \sum_{k=0}^{N-1} h[k]x[n - k]
\]

Shown crucially in the frequency domain as

\[
y[n] = x[n] \ast h[n] = F^{-1}[X[k]H[k]]
\]
Which will allow the preservation of only frequencies that are present in bowed strings. Discrete Fourier and inverse discrete Fourier transforms are also of utmost importance:

\[
X[k] = \frac{1}{N} \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N} \tag{4}
\]

\[
x[n] = \sum_{k=0}^{N-1} X[k] e^{j2\pi kn/N} \tag{5}
\]

Deconvolution requires an offset \( s \) to reduce errors due to transfer function components being equal to zero, offset \( s \) incorporated by

\[
\hat{X}^{-1}(\omega) = \frac{1}{X(\omega) + s} \tag{6}
\]

With inverse filter \( \hat{X}^{-1}(\omega) \) multiplied by \( Y(\omega) \) to produce true impulse response \( \hat{h}(t) \)

\[
\hat{h}(t) = F^{-1}[Y(\omega)\hat{X}^{-1}(\omega)] \tag{7}
\]

Achieved similarly in the time domain as

\[
\hat{x}^{-1}(t) = F^{-1}\left[\frac{1}{X(\omega) + s}\right] \tag{8}
\]

\[
\hat{h}(t) = y(t) * \hat{x}^{-1}(t) \tag{9}
\]

6 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 [8], we must acknowledge that similar prior work exists. Notably, the Strados electric
violin, created by ZETA Violins [9], uses an internal active preamplifier system which allows the volume of each string and the overall gain to be adjusted manually [10]. 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” [11]. 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


A Circuit Schematics

Figure 6: User interface overview.
Figure 7: User interface microcontroller.

Figure 8: Rotary encoders and quadrature decoder.
Figure 9: Lightbar indicator, used for audio parameters.

Figure 10: Power supply.