DYNAMIC VIOLIN FINGERBOARD

By

Adrian Ignaci

Kamil Waz

Sophia Wilhelm

Final Report for ECE 445, Senior Design, Spring 2026

TA: Manvi Jha

6 May 2026

Project No. 93

Abstract

The dynamic violin fingerboard aids self-taught violinists in learning correct finger placements and rhythms. Users upload a MIDI file for a piece they would like to learn and configure the piece settings. A dynamic LED display laid on top of the fingerboard then illuminates the locations where the user should place their fingers at the appropriate times. Potentiometers measure the position of a finger placed along any of the 4 paths (strings) on the fingerboard. These measurements are turned into an accuracy metric that is provided to the user via an LCD display. This allows us to collect information on how accurate the placement is, rather than a simple yes or no as to whether they play the right note.

Contents

1. Introduction

1.1 Problem

Most people would like to learn an instrument; however, not only are the instruments expensive, but the lessons are just as (if not more) costly. This also assumes lessons are even available where they live. For this reason, many people try to teach themselves how to play, either through experimentation or online resources. However, this path has a distinct lack of feedback that would help correct poor habits or otherwise incorrect playing.

1.2 Solution

Our project seeks to give those self-learning a violin an extra source of feedback with respect to finger placement (creating the notes) as well as the rhythm played. A dynamic LED display laid on top of the fingerboard would allow learners to better understand proper finger placement in addition to its relation to the specific note's duration.

Furthermore, by using linear/membrane potentiometers we measure the position of a finger placed along any of the 4 paths (strings) on the fingerboard. This allows us to collect information on how accurate the placement is, rather than a simple yes or no as to whether they play the right note.

To encourage building good habits and continuous practice, we would like to allow users to upload pieces they would like to learn. Thus, users will be allowed to upload files (MIDI) that can then be used on the fingerboard along with an adjustable tempo. This, paired with individual settings for full piece playthroughs and learning (only playing the next note after the user plays it) will help encourage good, accurate playing whilst making it fun.

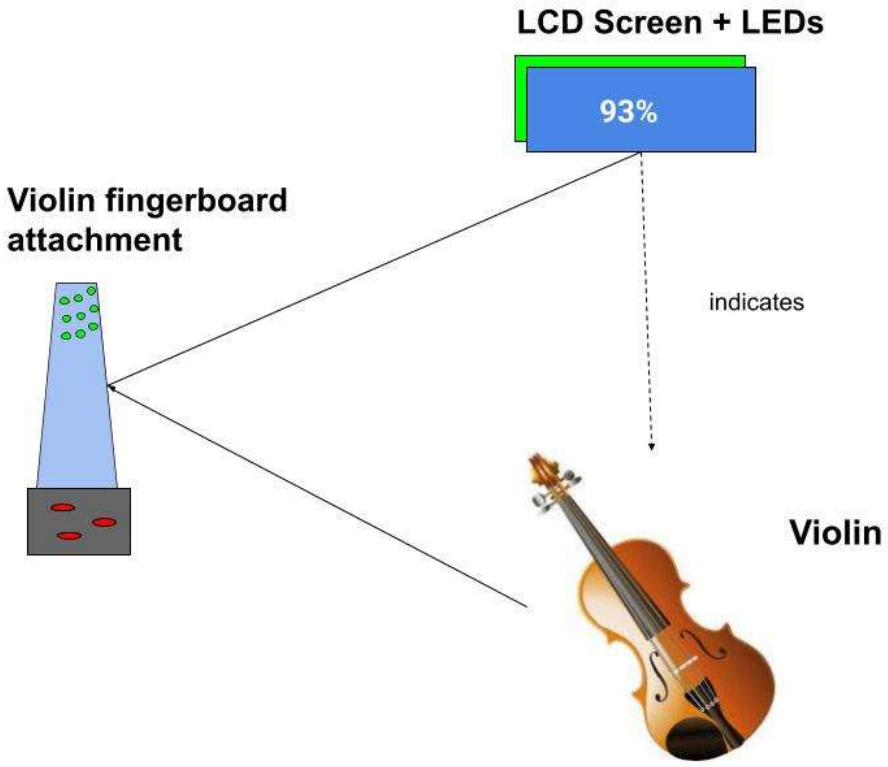

Figure 1: Basic Product Interaction

1.3 Requirements

This project will be considered successful if:

- The LEDs on the fingerboard turn on independently from each other in response to a signal from the control unit.
- A MIDI file less than 1 MB can be uploaded to the control unit and sets the control sequence for the fingerboard LEDs.
- Potentiometer membrane can detect finger placement within 2 mm and measurements are converted to an accuracy score that is provided to the user at the end of the piece.

If time permits, the following features will be added:

- Real-time accuracy feedback.
- Multiple songs can be uploaded to the controller at once and the user can choose which to play.

2 Design

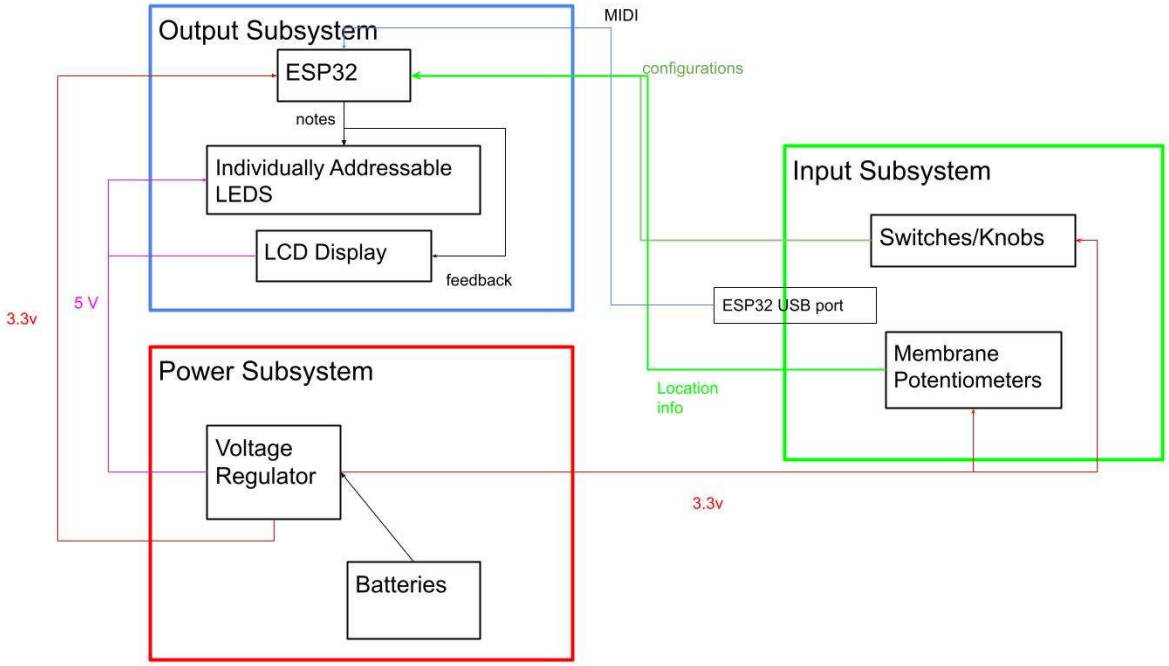

Figure 2: Block diagram of project's three major subsystems

Figure 3: Device placement on a full size violin using an elastic strap. Highlighting indicates the sensor strips. The box shown on the back is the main module enclosure (approximately 6" x 6" x 3"). Original photo from Amro Music [7]

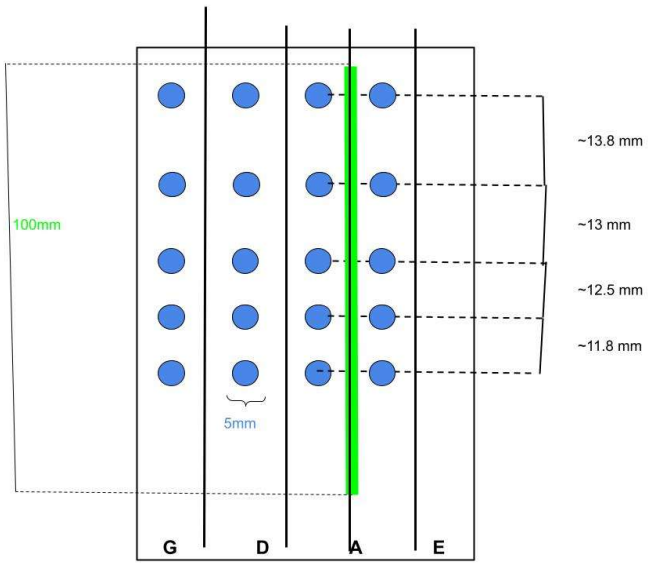

Figure 4: Sensor and LED placement on the fingerboard. Each string will have a sensor (green) and accompanying LEDs (blue). Note the logarithmic scaling of relative note distances.

The high level design consists of three major systems. The power system, as the name implies, is responsible for delivering and regulating the flow of power to the components in each subsystem. This is complicated by the existence of multiple separate, but interdependent, sub-circuits with different power needs. The output system encompasses the microprocessor and is how we provide feedback to the end user. Similarly, the input subsystem allows the user to interact meaningfully with the device. Without it, there would be no way to understand what the user is doing relative to our expectations or, as with the case of file uploads, what those expectations should even be.

The total weight of the entire project must not exceed 1lb to avoid disrupting the balance of the instrument or physically straining the user.

2.1 Power subsystem

The power subsystem will provide necessary power to all other subsystems and components.

It will provide the microcontroller with up to 250mA at 3V, and the LEDs and LCD with 500mA at 5V. A voltage regulator will constrain the provided voltage to within the safety tolerances of the individual parts as needed. Due to the differing voltage requirements for each subcircuit, they must remain relatively isolated. So, an increase in power consumption from one subcircuit should not cause the voltage supplied to the microcontroller to fall below 3 volts at any point (Espressif Systems, 2019). Similarly, an increased consumption should never allow the voltage of the LEDs to fall below 3 volts (Feztek).

The batteries provide the overall power; the voltage regulator helps isolate the two separate subcircuits (as they have different voltage requirements) while allowing for consistent current flow.

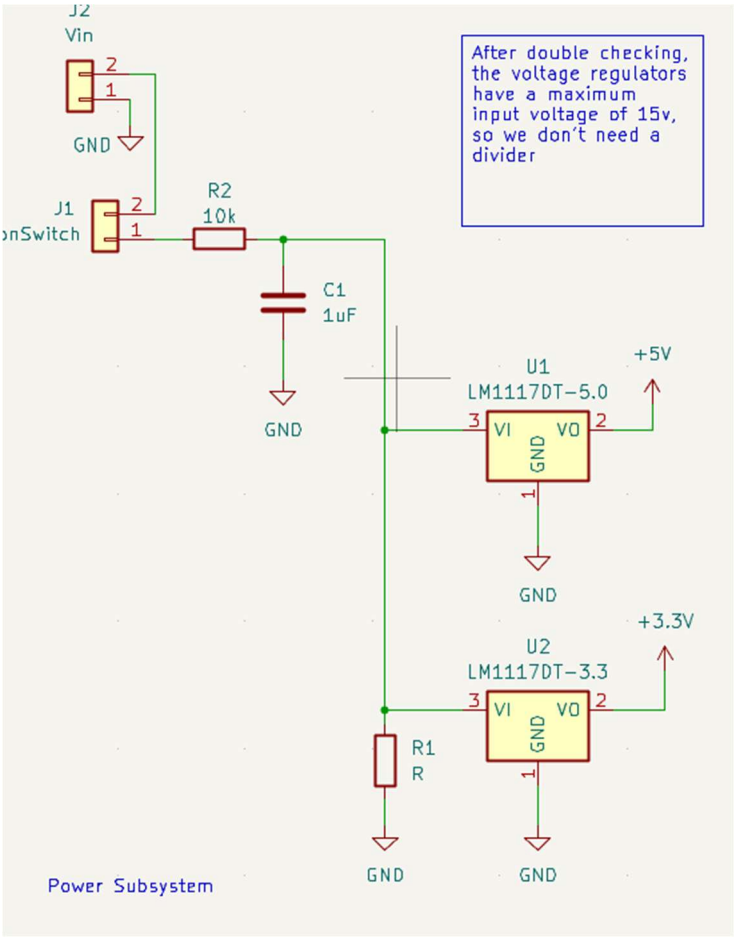

Figure 5: Power Subsystem Schematic

Table 1: Requirements and Verification for Power Subsystem

Requirements	Verification
Must supply a consistent voltage to the ESP32 microcontroller that is within 3.3V +- 5% and a current of 250mA	• Connect voltage supply to another subsystem • Connect positive lead of multimeter to the microcontroller and the negative lead of the multimeter to ground • Check the multimeter to make sure that the voltage that goes through the microcontroller deviates at most 5% from 3.3V • Increase the voltage supply to the other subsystem and make sure that the voltage that goes through the microcontroller still says within 5% of 3.3V
Must supply a consistent voltage to the LEDs and LCD display that is within 5V +- 5% and a current of 500mA	• Connect voltage supply to another subsystem • Connect positive lead of multimeter to the LCD display and LEDs and the negative lead of the multimeter to ground • Check the multimeter to make sure that the voltage that goes through the the LCD display and LEDs deviates at most 5% from 5V • Increase the voltage supply to the other subsystem and make sure that the voltage that goes through the LCD display and LEDs still says within 5% of 5V

2.2 Input subsystem

The input subsystem is responsible for managing user input. This includes the buttons and switches located on the main-body unit that the user uses to control configuration settings. Also included are the membrane potentiometers mounted on the fingerboard that measure user accuracy. The transfer of information will be routed through a microcontroller.

The input subsystem must be able to detect the finger placement of the user within 2mm and convert this as part of an accuracy score. The ESP32, while part of the output subsystem, does take input via the USB port. Thus, it should download a provided file within 5.5 seconds.

The switches and knobs will provide the output subsystem, specifically the ESP32 controller with the proper context under which to operate. It will allow the user to specify not only the tempo, but also what manner of play (progressive or run-through) to proceed with. Meanwhile, the membrane potentiometers will also interface with the ESP32 to allow data tracking and, in the case of progressive play, continue the piece when sufficient accuracy is achieved.

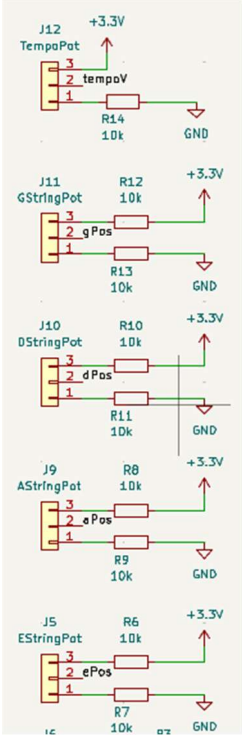

Figure 6: Schematic of external connections to sensors for each string and tempo control. Note that the ribbon sensors (membrane potentiometers) must be placed on the fingerboard and thus cannot be soldered on the main PCB

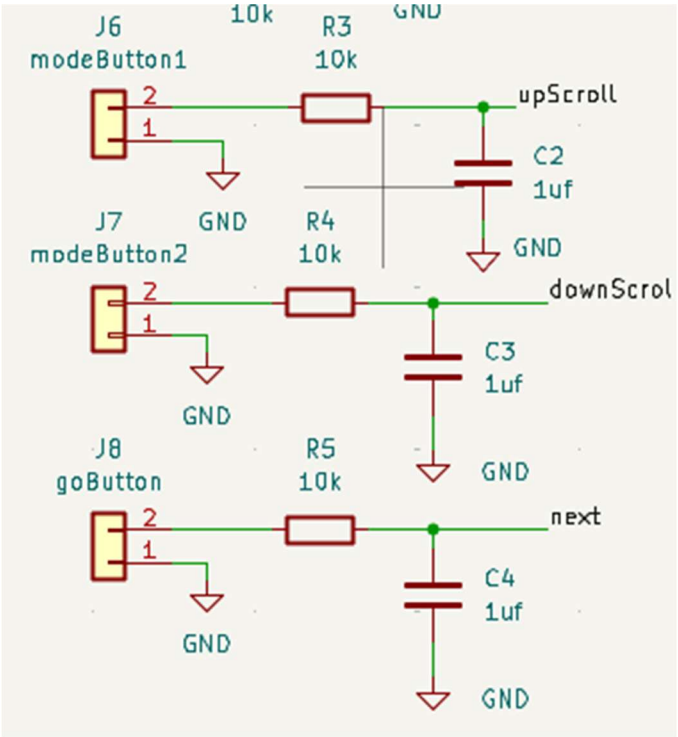

Figure 7: External connections to buttons used for configuring the device before use. Note the 10ms debounce circuit.

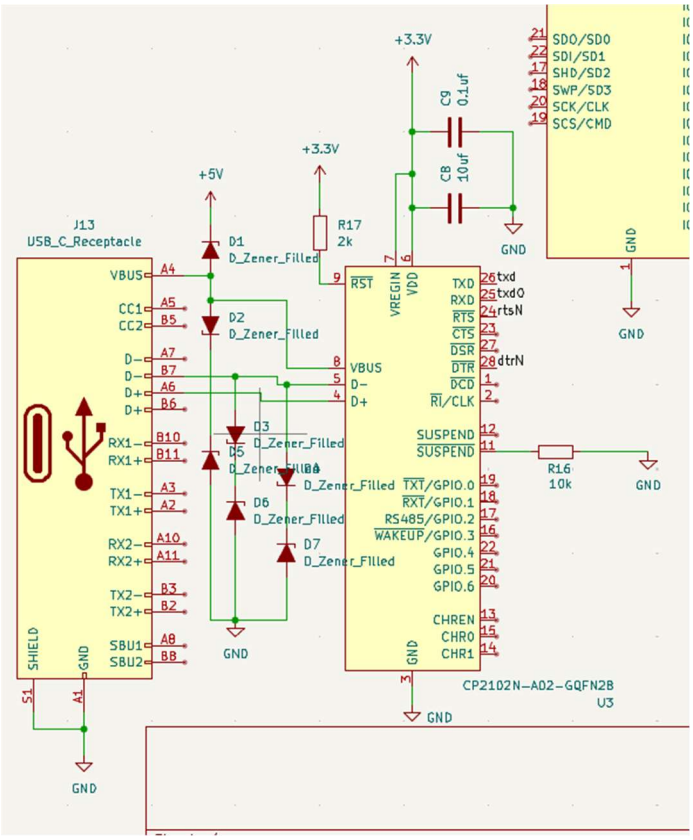

Figure 8: Custom USB-C compatible programming and file upload interface utilizing a UART-USB master bridge

Table 2: Requirements and Verification for Input Subsystem

Requirements	Verification
The input subsystem should be able to detect finger placement of the user under 2mm +- 0.1mm	• Use a ruler to measure the distance between a finger and a certain location on the membrane potentiometer • Slowly inch finger towards desired location and stop moving finger once the potentiometer detects you're in the "correct" place • Measure distance between finger and ideal location and see if it is within 2mm. Detecting correct finger placement when the finger is up to 2.1mm away from the desired location is tolerable
ESP32 should take input via a USB port and download desired MIDI file within 5 seconds +- 0.5 seconds	• Use a stopwatch and start it when the file is transmitted and stop it once the programmer receives the file and provides the notification that the file has been downloaded. This should take under 5 seconds, but taking as long as 5.5 seconds is acceptable

2.3 Output subsystem

The output system is responsible for providing visual responses to the user. Individually addressable LEDs on the fingerboard guide the user through learning: 20 LEDs will be divided between the four strings to allow guidance for 5 notes per string (see figure 4). A Liquid Crystal Display makes setting configurations more user-friendly and shows statistical information taken from a playing analysis computed by the microcontroller.

In the context of musical performance, it is important to have a consistent tempo. Therefore, we want the latency between the microcontroller and the LEDs to be self-consistent, within a variation tolerance of 7ms. Changes to the user-input settings will be reflected on the LCD display in under 1s. Final stats will be shown on the display within 10s of the piece finishing.

The ESP32 is the brain of the project and will analyze all the data to be communicated back to the user. This includes reading MIDI files and telling the user what input to provide. For this reason, it will control the LEDs via custom mapping to show finger positioning and give feedback on the LCD when a piece is over.

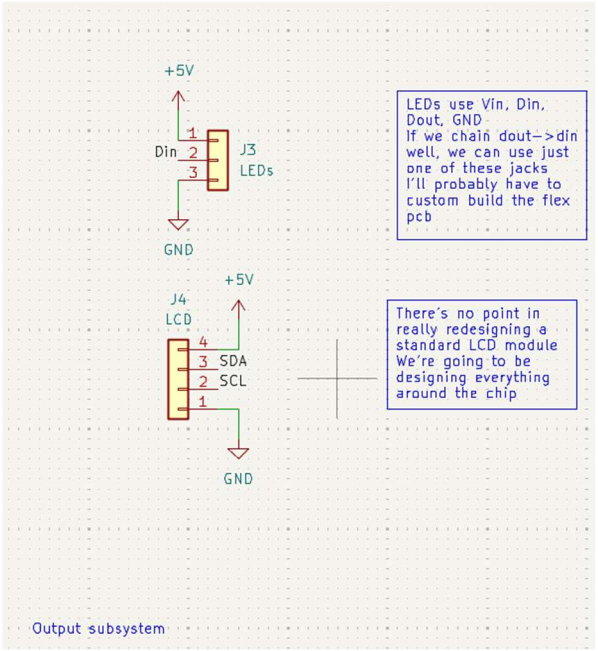

Figure 9: External pin connections to allow for LED and LCD control. Note that the LEDs will be on the fingerboard and thus cannot be mounted directly on the main PCB

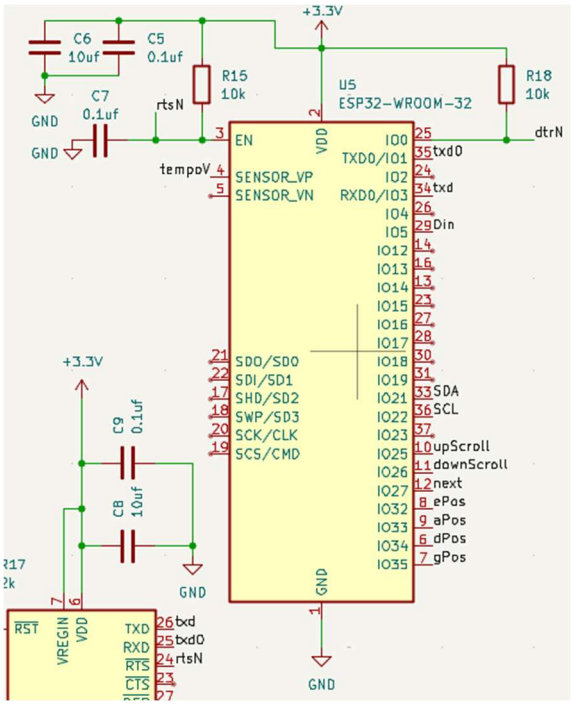

Figure 10: Connections to the ESP32 chip. Note the capacitors used to filter noise from the enable signal.

Table 3: Requirements and Verification for Output Subsystem

Requirements	Verification
Latency between microcontroller and LEDs must be self-consistent within a variation tolerance of 7ms +- 1ms	• This will be very difficult to measure with a stopwatch so we cannot use completely quantitative measurements • Humans recognize latency between notes when it is above 7ms so as long as there is a consistent tempo between successive notes and we cannot notice any tempo changes then it meets the requirement
Sheet music to addressable LED mapping should be 98% +- 1% accurate	• We provide the ESP32 sheet music for five different violin pieces, so 5 tests • Between all those tests, at most 1-2 incorrect LEDs should light up.
Changes to user-input settings should be reflected on the LCD display within 1 second +- 0.5 seconds	• Use a stopwatch, start it when user confirms a change in input settings and stop it when the changes are reflected on the LCD display • The changes should be reflected within 1 second although a delay of up to 1.5 seconds is acceptable

2.4 Tolerance analysis

2.4.1 Fingerboard unit resilience

The individually addressable LEDs and the membrane potentiometer are located on the fingerboard underneath the strings—where they will be consistently pressed down on by the strings and the user's fingers. This puts them at risk of advanced degradation. The potentiometer is designed for frequent contact, so it will likely be ok, but the LEDs are not. If this becomes an issue, the proposed solution is to offset the LEDs from the strings; far enough to not come in contact with the strings but close enough that it is still obvious which finger position each LED corresponds to.

2.4.2 Variation in power requirements

The LEDs require 5V. The LCD will require either 5V or 3V depending on the specific display used. The microcontroller runs on 3.3V and will break if it receives 5V. We are currently planning on having separate batteries to handle the different voltage requirements. If that doesn't work out, a voltage regulator will step down the voltage to the level needed by the microcontroller. Another possibility we have considered was using two resistors as a voltage divider to split the voltage going into the LEDs and the microcontroller to reduce cost. We want to use common enough resistance values where the ratio of the two resistances is 2:1. We intend on using a 9V battery for testing. Given this information, we can calculate the voltages going through the LED and the Microcontroller. Let's assume R1 is twice the resistance of R2.

$V_{LED} = V_{Battery} * (R1 / (R1+R2)) = 9V * (2/(2+1)) = 9V * (2/3) = 6V$

$V_{Microcontroller} = V_{Battery} * (R2 / (R1+R2)) = 9V * (1/(2+1)) = 9V * (1/3) = 3V$

These are not the typical voltage levels of each of the two components. According to the datasheets however, the addressable LEDs can handle voltages ranging from 3V to 7V so 6V falls within this range. Likewise, the ESP32 microcontroller can handle voltages between 3V and 3.6V so the voltage divider implementation can work despite the voltages not being quite the desired values.

2.4.3 Weight requirement

Violins typically weigh about 500g, so even a moderately heavy system will affect the balance of the instrument. We will thus strive to make the entire project as lightweight as possible. Specific steps we are taking to reduce weight are considering button batteries (smaller than an AA battery), finding the lightest possible LCD, and streamlining the casing and mount.

3. Cost and Schedule

3.1 Cost Analysis

3.1.1 Labor

We estimate our labor assuming three major roles are present: a PCB Design Engineer, Embedded Software Engineer, and a general Electrical Engineer (so that we account for safety testing, verifications, and any other miscellaneous work or assistance provided in the other areas).

A typical ECE graduate from UIUC makes between $90,000 (for EE) and $100,000 (for CompE) annually. Furthermore, a PCB designer makes on average $75,000 per year. An embedded software engineer on average makes approximately $111,000 per year. This is approximately equivalent to:

$$\frac{\$75000/year}{52weeks/year * 40hrs/week} \approx \$36/hr$$

$$\frac{\$111000/year}{52weeks/year * 40hrs/week} \approx \$53/hr$$

$$\frac{\$90000/year}{52weeks/year * 40hrs/week} \approx \$43/hr$$

Estimating the cost of labor thus is as follows (noting that once the product is designed and at sold at scale, this will not be the sale price):

$$\frac{\$36}{hr} * 80hr = \$1440$$

$$\frac{\$53}{hr} * 60hr = \$3180$$

$$\frac{\$43}{hr} * 60hr = \$2580$$

$$(\$1440 + \$3180 + \$2580) \times 2.5 = \$18000$$

Additionally, we will use an approximate 10 labor hours from the machine shop. Assuming a rate of $20 per hour, this adds another $200 of cost to the project. This yields:

$$Total\ Labor\ Cost = \$18200$$

3.1.2 Parts

Description	Manufacturer	Part #	Quantity	Cost/pc	Source
Membrane Potentiometer	Spectra Symbol	SP-L-0100-103-3-ST	4	$7.95	Adafruit
USB-C Gen 2 Receptacle	Amphenol ICC	10152803-00011LF	1	$1.08	Digikey
Voltage Regulator 3.3v	UMW	LD1117-3.3	1	$0.30	Digikey
Voltage Regulator 5v	Texas Instruments	LM1117DT-5.0/NOPB	1	$1.71	Digikey
16x2 LCD Display Module	Hosyond	B0BWTFN9WF	1 (comes in packs of 3)	$4.33	Amazon
Individually Addressable LEDs	Feztek	FZ2812-5050	20	$0.63	Digikey
ESP32 Microcontroller	Espressif Systems	ESP32-WROOM-32E-N4	1	$4.84	Digikey
USB-UART Bridge	Silicon Labs	CP2102N-A02-GQFN28	1	$4.32	Digikey
Knob/Rotary Potentiometer	Panasonic Electrical Components	EWV-YKXL16B14	1	$3.02	Digikey
On/Off Rocker Switch	E-Switch	RA1113112R	1	$0.64	Digikey
Buttons	Adafruit	1445 (Prod ID)	3	$0.95	Adafruit

3.1.3 Totals

Thus we can calculate the total cost of parts:

$$Total\ Parts\ Cost = (4 \times \$7.95) + (3 \times \$0.95) + (20 \times \$0.63)$$

$$+(\$4.84 + \$4.32 + \$3.02 + \$1.08 + \$0.30 + \$1.71 + \$0.64) = \$63.16$$

We end up with a grand total of:

$$Total\ Labor\ Cost + Total\ Parts\ Cost = \$18200 + \$63.16 = \$18263.16$$

3.2 Schedule

The following schedule shows our intended pace and work breakdown. It is subject to change and/or adaptation depending on the needs of the project and the individuals working on a particular task.

Week	Task	Person
February 16th-20th	• High level designs of all subsystems • Research parts/components • Begin schematic and PCB design	• Sophia • Kamil • Kamil
February 23rd – 27th	• Finish schematic and order PCB • Order parts for prototype • Begin ordering final components • Begin software development (file storage)	• Kamil, Sophia • Adrian • Adrian • Kamil
March 2nd – 6th	• Debug PCB, order second round • Design custom LED configuration • Finish preliminary software (file reading, display, sensing) • Assemble prototype • Order final components • Send final design to machine shop	• Sophia, Adrian • Sophia • Kamil, Sophia • ALL • Adrian • Adrian
March 9th – 13th	• Breadboard demo • FINAL TWEAKS to machine shop • Debug PCB + redesign • Software tempo, playstyle control	• ALL • Adrian • ALL • Kamil, Sophia
March 16th – 20th	• SPRING BREAK • Software debugging	• Whoever wants to
March 23rd – 27th	• PCB Debugging • FINAL PCB ORDER • Software statistics and display • Refine prototype	• Sophia, Adrian • Sophia, Adrian • Kamil • ALL
March 30th – April 3rd	• Add wireless file communication (software) • Self order PCB (if applicable) and continue debugging	• Kamil • ALL (if the PCB is not functional yet this is the MAIN PRIORITY, not SW)

April 6th – 10th	• Progress demo • Assemble final system • Any applicable debugging	• ALL
April 13th – 17th	• Prepare mock up • Any applicable debugging	• ALL
April 20th – 24th	• Mock demo • Any applicable debugging	• ALL
April 27th – May 1st	• Last minute tweaks and debugging • FINAL DEMO	• ALL

4. Ethical considerations

Due to the nature of electrical systems, it can be inherently dangerous to keep a battery-operated device near people and on expensive equipment, as a fault in equipment could cause combustion leading to unjust harm to an individual and/or personal property, as outlined by the ACM (Association for Computing Machinery, 2026) 1.2 and IEEE (Institute of Electrical and Electronics Engineers, 2026) I.1. Thus, should there be a failure in the design or degradation over time, it is possible that the player or their instrument be harmed or severely injured.

As the designers, we are obligated to add redundancy both to prevent such accidents, as well as ways to mitigate accidents should they occur. This would include appropriately strong housing in case of any blown components and methods for heat dissipation to prevent overheating (which can cause warping in wood). The industry standard for electronics is generally plastic due to its electrical isolation and heat-resistant properties (AIP Precision Machining, 2025).

Furthermore, the ability for users to upload MIDI files without any sort of licensing or verification can enable, or otherwise encourage, the improper acquisition of creative works – specifically music. According to the US Copyright Office, derivative works made without permission from the original owner constitute copyright infringement (US Copyright Office, 2026). This can include MIDI files as they are generally treated as another medium expressing the same creative work. Sort of like uploading a digital copy of a physical book or translating it into a different language.

There seemingly isn't very much we can do to stop this without entirely removing the customizable uploads. The open-ended, almost open-source, aspect of this project rests on the proper conduct of the consumer when it comes to copyrighted works.

The self-sufficient nature of the product can potentially lead to decreased interest in receiving formal musical training. However, this product is not meant to be a replacement and treating it as would lead to overall worse musical performance across society.

References

[1] AIP Precision Machining. (2025, August 5). *Plastic for Electronic Enclosures*. Retrieved from AIP Precision Machining: https://aipprecision.com/plastic-for-electronic-enclosure-expert-guide-to-maximum-protection-standards/

[2] Association for Computing Machinery. (2026). *Code of Ethics*. Retrieved from Association for Computing Machinery: https://www.acm.org/code-of-ethics

[3] Institute of Electrical and Electronics Engineers. (2026). *IEEE Code of Ethics*. Retrieved from IEEE: https://www.ieee.org/about/corporate/governance/p7-8

[4] US Copyright Office. (2026). *Definitions*. Retrieved from U.S. Copyright Office: https://www.copyright.gov/help/faq/faq-definitions.html#:~:text=What%20is%20copyright%20infringement%3F,%2Dpeer%20(P2P)%20networking%3F

[5] Espressif Systems. (2019). *ESP32 Series Datasheet Version 5.2*. Retrieved from Espressif Systems: https://documentation.espressif.com/esp32_datasheet_en.pdf

[6] Feztek, *FZ2812-5050 Datasheet.* Retrieved from Feztek: https://www.feztek.com/uploads/1/2/6/3/126369845/fz2812-5050_datasheet.pdf

[7] Amro Music, *Violin Diagram*, Retrieved from Amro Music: https://www.amromusic.com/violin