VIRTUAL SYNTHESIZER WITH MIDI KEYBOARD

By

Connor Barker

Dylan Pokorny

Patrick Ptasznik

Final Report for ECE 445, Senior Design, Spring 2025

TA: Eric Tang

Project No. 59

7 May 2025

Abstract

This paper presents the design and development of a low-cost virtual synthesizer built around an ESP32-S3 microcontroller and controlled via a MIDI keyboard. Aimed at beginners in music production, the project addresses common barriers such as high software costs, limited hardware access, and complex mixing techniques. The synthesizer supports multiple waveforms and real-world instrument presets, enabling interactive sound design through onboard controls and visual feedback via an LCD. Housed in a portable, “boom-case” inspired briefcase, the system includes subsystems for audio output, power regulation, user input, and display. The device processes MIDI input in real-time with low latency and supports polyphony of eight notes. By integrating essential music creation features into an affordable, user-friendly format, the design makes music production more accessible for new creators.

Contents

1 Introduction

1.1 Problem

Music production has a high barrier to entry due to expensive software, lack of accessible tools, and the challenge of mixing and arranging sounds effectively. Industry-standard VSTs (Virtual Studio Technology) are costly, with popular options like Nexus ($120), Omnisphere ($499), and ElectraX ($150), which require a significant investment just to access high-quality sound libraries. Such an expense discourages beginners who may not be ready to make such a financial commitment while still learning the basics of music creation.

Outside of the costs of software, beginners struggle to experiment with their musical ideas without proper hardware. MIDI controllers and synthesizers are great for refining compositions, but without them, creating structured music often leads to offbeat, unbalanced, or disorganized results. Trial and error becomes the main method of learning, making music creation inefficient and much harder to smoothly develop.

Finally, mixing presents another challenge, as beginners often misjudge volume levels, layering, and effects. Poorly balanced mixes can make instruments overpower one another or disappear, while excessive effects like reverb can “blur” the sound. Without guidance, achieving a clean, professional mix becomes frustrating, slowing progress and limiting a musical artist’s creative potential.

1.2 Solution

Our virtual synthesizer lowers the barrier to entry for beginners by combining essential hardware and software features found in modern VSTs while remaining affordable and beginner-friendly. It includes a MIDI keyboard, allowing users to play notes and experiment with melodies smoothly. Customizable sounds and effects let users shape their own tones and explore sound design interactively, making learning more engaging and entertaining.

The synthesizer offers multiple instrument options, such as synths, flutes, and pianos, enabling users to experiment across different musical styles. A user interface consisting of an LCD screen provides real-time feedback on selected instruments and effects, helping beginners understand how different settings work together and influence sound. By offering a cost-effective, all-in-one solution, this virtual synthesizer makes music production more accessible without the need for expensive software or complex setups.

1.3 High-Level Requirements

Instrument Simulation and Sound Generation

The virtual synthesizer will be capable of playing a selection of sound waveforms, which will be heard with no observable latency between playing on the keyboard and auditory feedback.

Polyphony and Chord Support

The virtual synthesizer will be capable of sounding multiple notes simultaneously in order to support playing chords.

Octave Range and Note Accuracy

The virtual synthesizer will be capable of playing multiple octave ranges of accurately pitched musical notes.

2 Design

As shown in Figure 1, the entire virtual synthesizer is housed in a customized briefcase made to resemble a retro-styled 'Boombox'. The synthesizer will receive power from a North-American wall outlet using a consumer wall adapter. The user will also have to plug in the MIDI keyboard into the input hole on the side of the briefcase. The MIDI keyboard and AC adapter will be outside the briefcase, and will connect to the PCB itself through ports into the side of the briefcase.

The inside of the briefcase contains our ESP32-S3 Microcontroller, MAX98357 I2S DAC and Amplifier, and all backsides of the speakers, LED's, and user controls along with their necessary wiring. Realistically, the user will not have to open the briefcase in order to properly use the synthesizer, adding an extra layer of tidiness and creativity.

Finally, when the virtual synthesizer is in use (assuming the briefcase is closed), the user will have the speakers, LED's, LCD Screen, and user controls visible to them. Here, they can interact with the user controls subsystem, which consist of potentiometers and buttons to adjust volume, wave type, and special effects. The LED's are also displayed here, which are separated into two sets of five. The first set is a frequency visualizer which lights up the corresponding LED when playing frequencies within a specified range. The second set of LEDs is a counter for how many keys are currently being pressed.

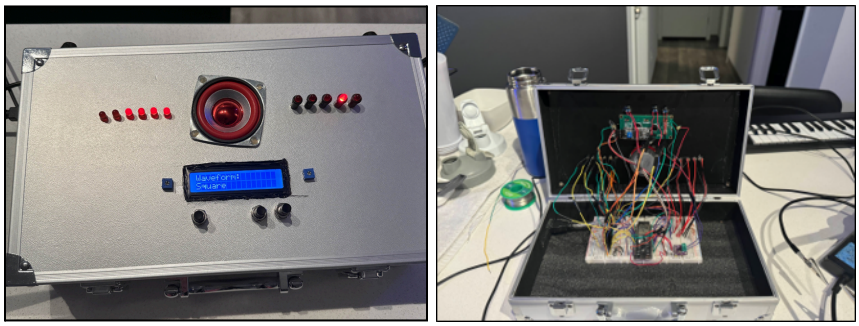

Figure 1. The project is implemented in a briefcase housing with ports for external devices.

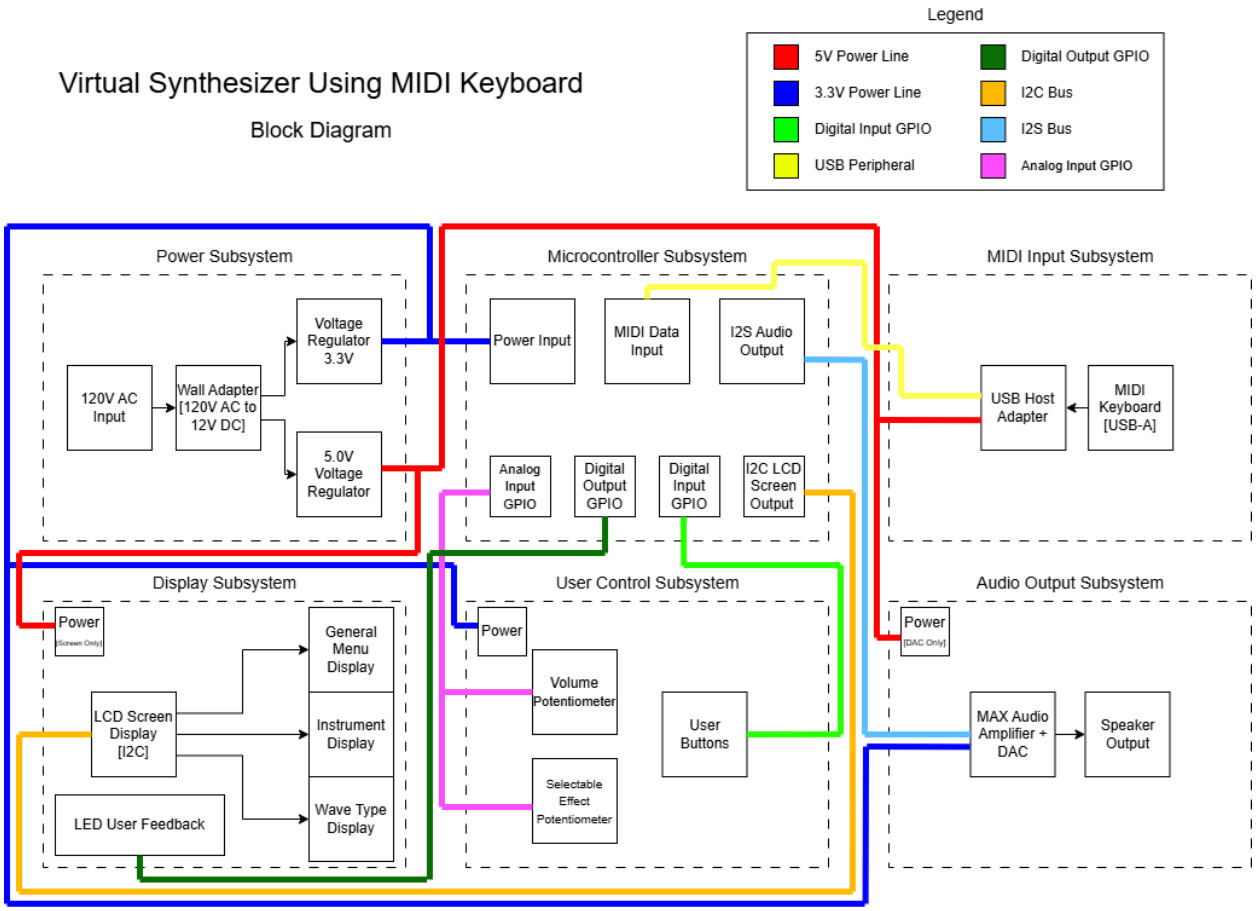

Figure 2: Block Diagram for Virtual Synthesizer with MIDI keyboard.

Our block diagram is shown in Figure 2. The Power subsystem provides 3.3 V to the Microcontroller, User Controls, and Audio Output subsystems. A 5.0 V regulator provides power to the Display, MIDI Input, and Audio Output subsystems. The Microcontroller receives digital and analog inputs from the User Controls, as well as digital MIDI information from the MIDI Input via a USB controller peripheral on the microcontroller. The Microcontroller performs the signal processing required to output digital on/off signals to the Display LEDs, digital display information to the Display LCD screen via I2C bus, and also outputs a digital audio waveform to the Audio Output subsystem using I2S bus via the on-chip I2S peripheral. The user will operate the User Controls and receive visual and auditory feedback from the Display and Audio Output subsystems.

2.1 Microcontroller Subsystem

This subsystem handles the signal processing from the MIDI keyboard and user controls and outputs digital I2S signals to the I2S Digital-to-Analog-Converter (DAC) and Amplifier in the audio output subsystem. An ESP32-S3 receives MIDI audio through its USB Host Controller. The analog inputs (potentiometers) from the user controls subsystem are received by the EPS32-S3 internal ADCs. The

digital inputs from user controls are taken as digital inputs to the ESP32-S3. The ESP32-S3 will produce a waveform and perform digital signal processing on it, then outputs the waveform as an I2S audio signal through its I2S0 peripheral (See Section 2.7 for software details).

The I2S bus is usually used to interface between chips on a PCB. However, we are using it here as a means of outputting an audio waveform. This was chosen as an alternative to using a microcontroller with a built-in Digital-to-Analog-Converter (DAC). These built-in DACs have low bit depths (typically 12 bits), rendering them unsuitable for outputting more detailed audio waveforms. The I2S peripherals on the ESP32-S3 offer bit depths of up to 32 bits, allowing far more detail in the audio waveforms. The I2S0 peripheral on the ESP32-S3 further offers two output modes: Pulse-Code Modulation (PCM, typical of the I2S bus) and Pulse-Density Modulation (PDM). While we have opted for the more typical PCM mode and will run the signal through a specialized converter, the PDM mode offers an alternative design which does not require an external DAC, as PDM is extremely simple to convert to analog: a low-pass filter is all that is required. We chose to use the external DAC instead of a simple low-pass filter because the conversion would be of higher quality, and the chosen DAC also includes an amplifier, which would have otherwise been separately required since we are using a passive speaker. The I2S bus will interface with our MAX98375A, which is an I2S DAC and Amplifier, in the Audio Output subsystem.

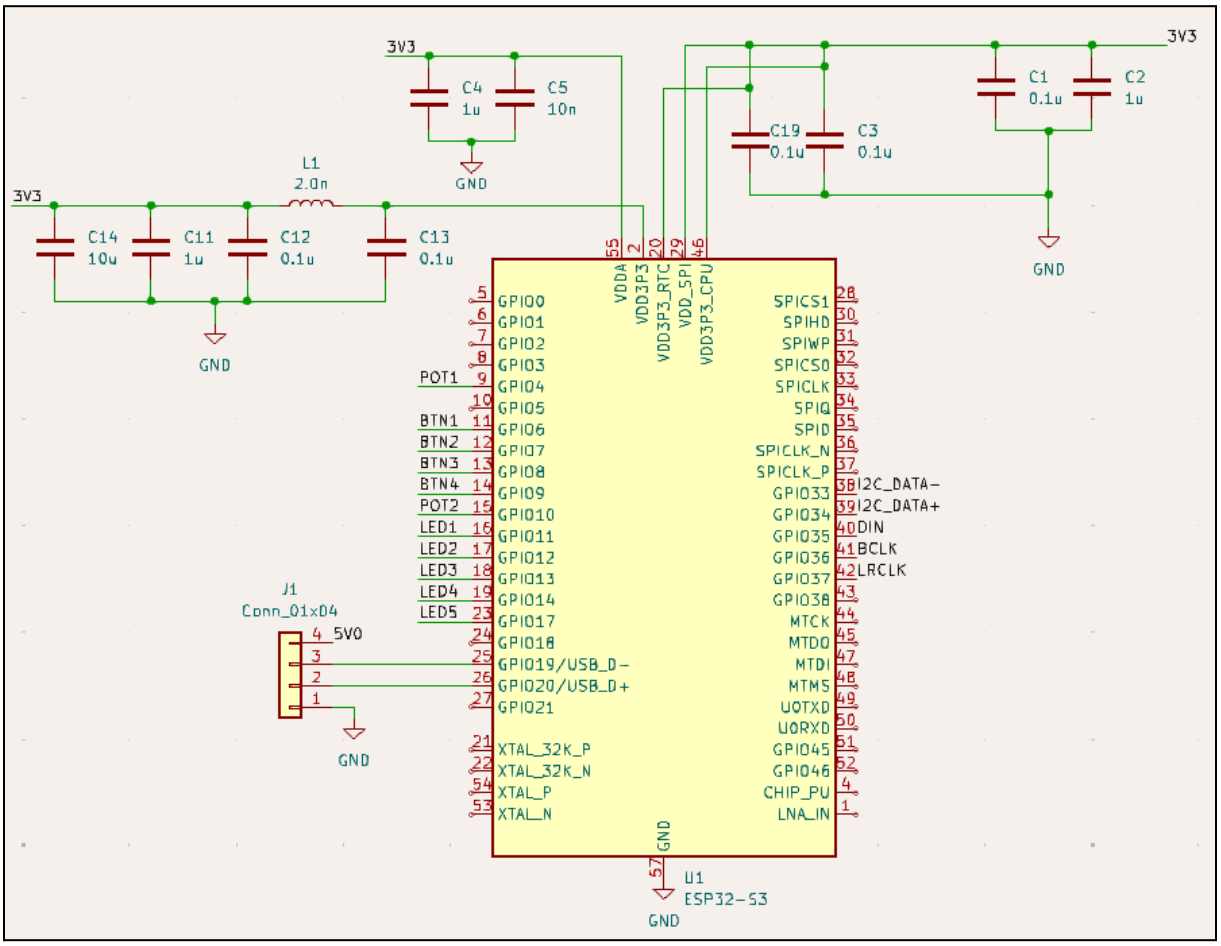

Figure 3. The PCB schematic of the Microcontroller Subsystem.

Shown in Figure 3 is the Microcontroller subsystem schematic of our PCB, which was mostly non-functional. Note also that the number of LED outputs are only half that of our final design, and the number of buttons is one greater than our final implementation. These are because, after our PCB was determined non-functional, our design developed to include more Display subsystem functionality (increasing the LED count) and only required three buttons to effectively navigate the user interface.

Also in Figure 3 is a power circuit which receives the 3.3 V line from the Power subsystem and runs it past some reactive components into the power pins of the ESP32-S3. These components and their values are recommended from the manufacturer of the ESP32-S3. On the PCB, the capacitors are placed close to the ESP32-S3 to minimize non-ideal effects of tracing.

One reason for the PCB being non-functional is attributed to an absent part of the microcontroller schematic. In order to program and enable or reset the ESp32-S3, the CHIP_PU pin must have a controllable voltage, whereas in our design we left it floating. GPIO0 must have a controllable voltage as well, as it must be high for operation but brought low for programming the ESP32-S3 using the USB peripheral. Without these changes, our PCB design will not function.

2.2 Power Subsystem

The power subsystem draws power from a 120V 60Hz AC wall outlet using a consumer 12V adapter. The 12V supply will be regulated to 3.3 V and 5.0 V using voltage regulators. The regulated 5.0 V line will power the USB interface to the MIDI keyboard, the LCD screen for user interface, and the I2S DAC and Amplifier in the audio output subsystem. The 3.3 V line will power the ESP32-S3 and the user controls (buttons and potentiometers) that send signals to the ESP32-S3, as well as controlling the mode select pin on the I2S DAC and Amplifier (MAX98357A). The 3.3 V and 5.0 V regulators will be supplemented with capacitors and inductors as listed on their datasheets under their typical application.

There are current draw requirements imposed on the 3.3 V and 5.0 V regulators. The ESP32-S3 has a typical draw of 500 mA, and another 100 mA can be allocated for the user controls and DAC mode select, which don't take much current. The 3.3 V regulator must be capable of drawing at least 600 mA of current. The MIDI keyboard is powered through USB 2.0, which has a rated maximum current draw of 500 mA. The LCD screen uses very little power when the backlight is off, but a typical current draw for an LCD1602 with a backlight on is 200 mA on the higher end (our selected LCD screen, chosen for its efficient pricing, does not have an available datasheet). The I2S DAC and Amplifier has a maximum current output of 1.6 A. Then, our 5.5V regulator requires a current rating for at least 2.3 A.

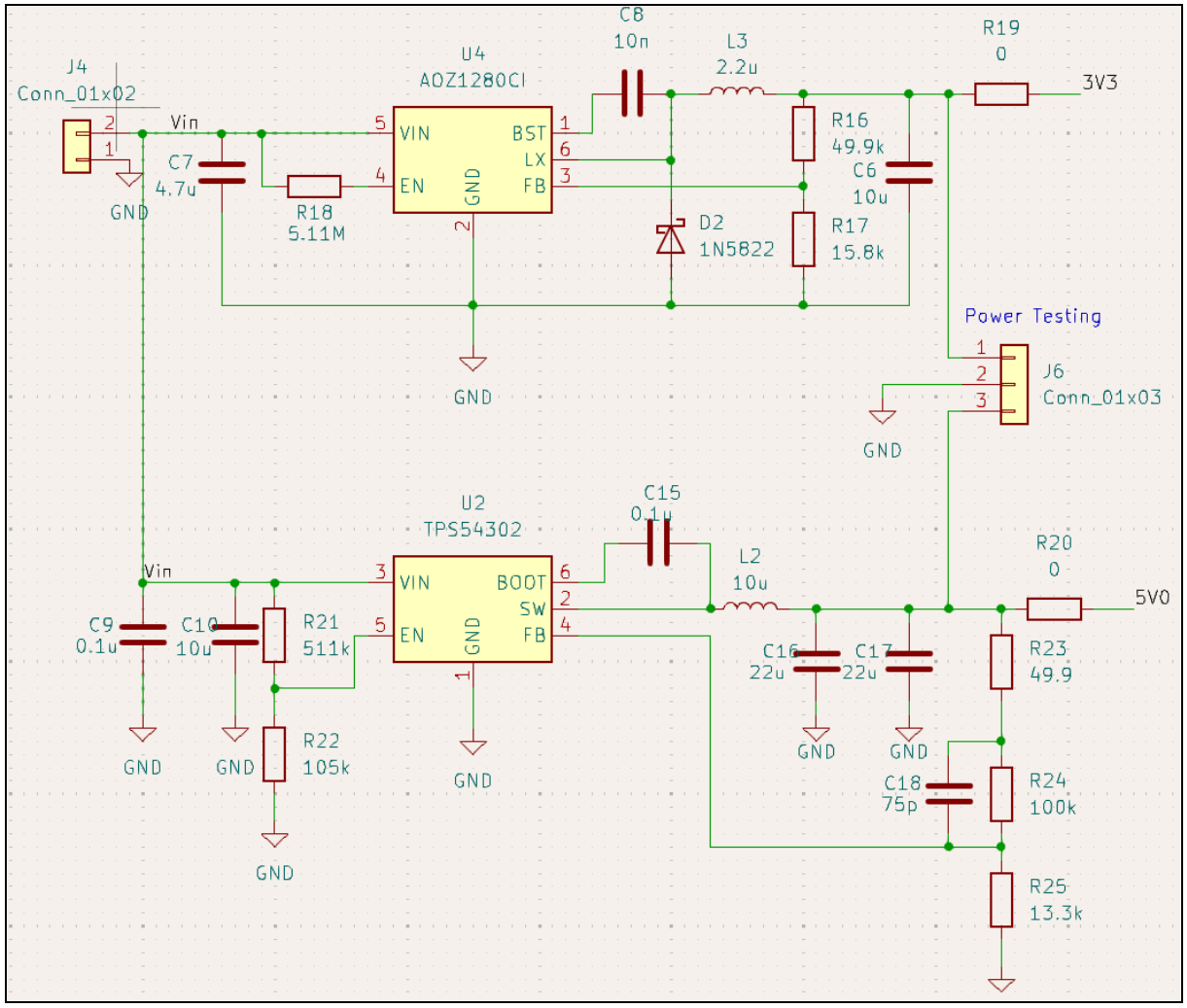

Figure 4. The PCB schematic of the Power subsystem. Vin represents the 12 V input from the wall adapter.

Shown in Figure 4 is the PCB schematic of our Power subsystem. Although our PCB did not function due to design flaws in the Microcontroller subsystem, the inclusion of a testing connector (J6) and the relative independence of the power circuits allowed us to perform our verification procedures on the Power subsystem. The verification revealed mistakes during soldering of the 5.0 V regulator circuit, having swapped the positions of inductor L2 and resistor R23. During this incorrect operation, the feedback resistors R24 and R25 were damaged, reading incorrect resistances, such that the 5.0 V circuit was still non-functional even after the misplaced parts were corrected.

2.3 User Controls Subsystem

The user controls receive 3.3 V from the power subsystem to operate. The synthesizer will use two potentiometers for user control, one dedicated for volume control and the other dedicated for post-effects. The volume potentiometer allows for a nearly continuous volume control range (nearly continuous, because the potentiometer signal will be converted through a 12-bit ADC and the volume will be adjusted digitally). Additionally, three buttons will be used to navigate through menus related to instrument selection and post-effects such as distortion. Each button will be connected to GPIO pins on the ESP32-S3, which will poll for activation. The buttons and potentiometers will be powered directly from the 3.3 V power line.

Shown in Figure 5 is the PCB schematic for our User Controls subsystem. Note that some changes were made in the final implementation, due to the PCB being non-functional and our project ultimately implementing a development board in its place. While all Espressif documentation indicates that the ADC peripherals on the ESP32-S3 measure between 0-1.1 V, the development board proved capable of measuring the full digital range of 0-3.3 V. This eliminated the need for a Zener diode and its current limiting resistor. Additionally, while our PCB included four buttons, during our implementation only three buttons were required for effective user control.

2.4 MIDI Input Subsystem

The user's note input will be taken through our MIDI Input subsystem, consisting of a MIDI keyboard. In order to connect the MIDI keyboard to the ESP32-S3 microcontroller, we use a USB female connector that allows the keyboard's USB cable to interface with the microcontroller. The USB connector is responsible for splitting the MIDI signal into two data lines, D+ and D-. The USB's VCC and GND pins are connected to the 5.0 V and GND pins from the Power subsystem. The two data outputs will then be connected to the corresponding USB-capable pins on the ESP32-S3, enabling it to read and process data inputted through the user themselves through the MIDI keyboard. When in USB host mode, the ESP32-S3 can read the key presses from the keyboard and relay this data to be processed.

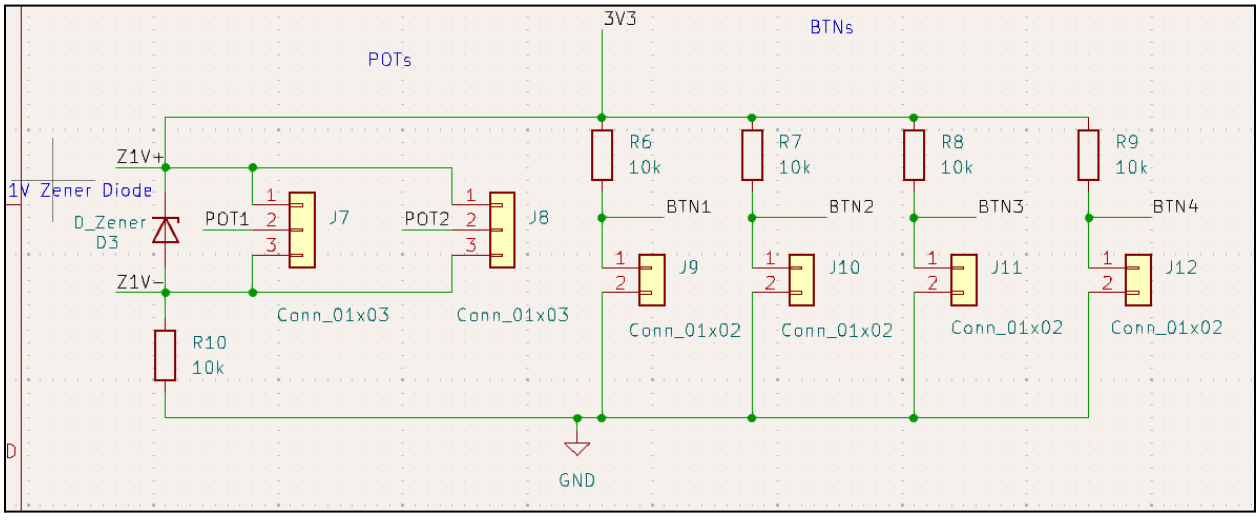

Figure 5. The PCB schematic for our User Controls subsystem.

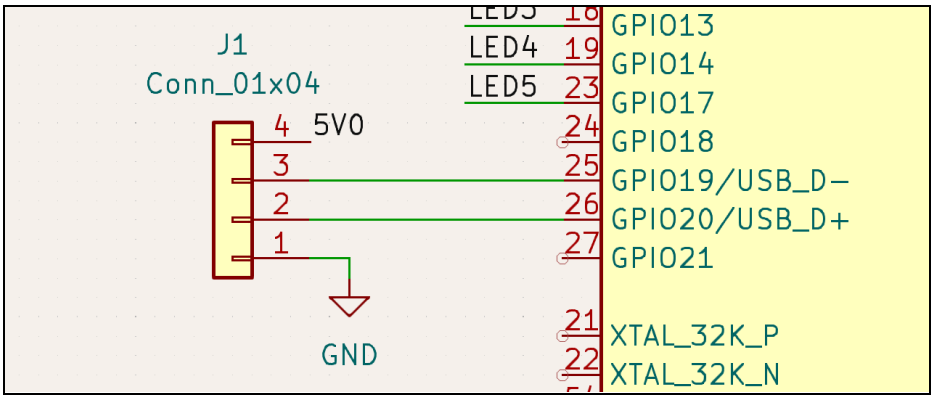

Figure 6. The traces between the MIDI keyboard's USB and the Microcontroller and Power subsystems.

Shown in Figure 6 is the PCB schematic tracing between the USB connector from the MIDI keyboard and the 5.0 V line from the power subsystem and the USB-capable pins of the ESP32-S3. Note that this USB connector is the same connector intended to be used to program the ESP32-S3.

2.5 Display Subsystem

The LCD screen will be used to display information about the currently selected instrument and volume level on the synthesizer. When the user switches instruments, the LCD screen will update to show the instrument name, helping beginners easily understand what sound they are working with. Additionally, when the volume is adjusted, the screen will display the current volume level on a scale from 1 to 10, providing clear feedback to the user. The display module being used—the GeeekPi 1602—communicates via I2C, which requires two data pin connections to the ESP32-S3, and 5 V power from the power subsystem. To enhance user feedback, we added five LEDs that will be responsible for counting how many keys are being played, as well as five LEDs that act as frequency visualizers, lighting up when a frequency corresponding to the LED's range is played. Each LED channel includes a current-limiting resistor to prevent damage to components and ensure safe operation.

The requirements of the current limiting resistor are calculated as

$$R_{max} = \frac{V_{max} - V_{FWD}}{I_{min}} = \frac{3.6 - 1.9}{0.005} = 540\ \Omega \quad (1)$$

$$R_{min} = \frac{V_{min} - V_{FWD}}{I_{max}} = \frac{3.0 - 1.9}{0.020} = 55\ \Omega \quad (2)$$

where V_{max} and V_{min} are the ESP32-S3's maximum and minimum digital high voltages, I_{max} and I_{min} are the maximum and minimum currents for safe active operation of the LEDs, and V_{FWD} is the forward voltage of the LEDs.

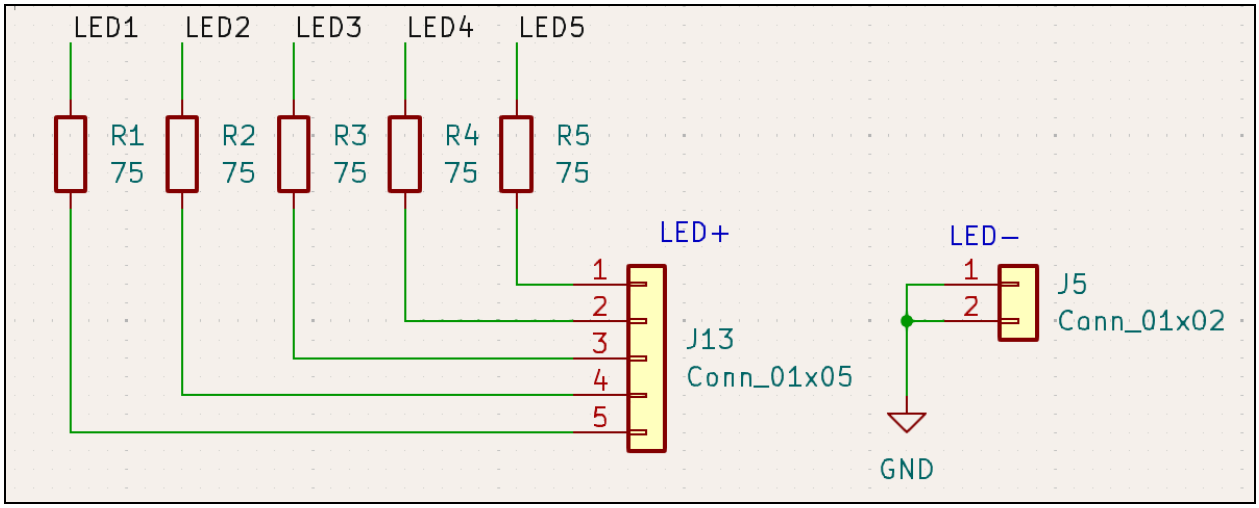

Figure 7. The PCB schematic of our original LEDs design.

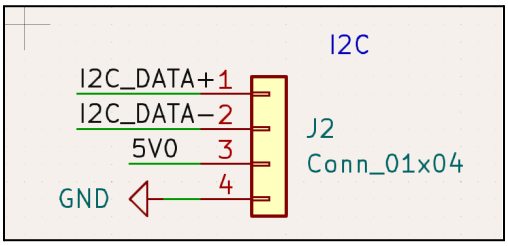

Figure 8. The PCB schematic of our I2C connector which connects to the LCD screen.

Shown in Figure 7 is the PCB schematic of our original LEDs circuit. Note that the schematic only has 5 LEDs where our final implementation has 10. The additional five LEDs were implemented after the PCB was determined to be non-functional. Note also that the GND connector for the LEDs has only two terminals, because the currents are not high and thus more than one GND wire could be placed in a single terminal. Shown in Figure 8 is the PCB schematic for our connector which interfaces with the LCD screen. The I2C bus only requires two data lines, and the screen requires 5.0 V power.

2.6 Audio Output Subsystem

The Audio Output subsystem is responsible for receiving a digital audio waveform via the I2S bus from the Microcontroller subsystem and converting it to an amplified analog audio waveform which will drive a 4 Ω speaker rated for 5 W, ensuring sufficient volume while maintaining a high level of sound clarity with no unnecessary noise.

The I2S digital-to-analog conversion is performed by an external I2S DAC and Amplifier, the MAX98357A. The MAX98357A offers several options to separate the left and right channels from I2S, but because our project does not benefit from stereo audio, we will opt for the mono-channel audio setting of (left/2 + right/2). The MAX98357A also offers fixed amplifier gains: 15, 12, 9, 6, and 3 dB. As the MAX only provides a power rating for the gain of 12 dB, we must select an amplifier gain of 12dB.

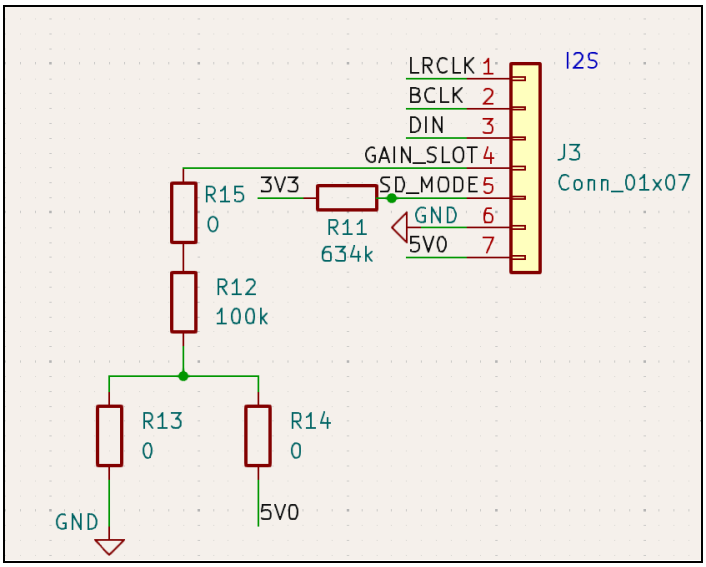

Figure 9. The PCB schematic of the I2S bus between the ESP32-S3 and the MAX98357A.

Shown in Figure 9 is the PCB schematic of the I2S bus between the ESP32-S3 and the MAX98357A. Following the MAX98357A datasheet, to select the mono-channel audio setting of (left/2 + right/2) we pulled up SD_MODE with a 634 kΩ resistor. To select an amplifier gain of 12 dB, the GAIN_SLOT pin was left floating by disconnecting the 0 Ω resistor R15. The resistor configuration connected to GAIN_SLOT is designed to be adjustable via soldering and capable of selecting any of the gains offered by the MAX98357A.

2.7 Software Design

System Architecture & Core Components

The software design centers around three main functional modules: USB MIDI input, digital audio synthesis, and effect processing. The core idea of the virtual synthesizer is to take incoming MIDI messages and process them accordingly. These messages include note on/off commands, note velocity, and note numbers, which are read and routed to the audio synthesis system. To handle MIDI communication over USB, we referenced an open-source GitHub repository that includes the necessary drivers for the ESP32-S3 MIDI Input [11].

The synthesis system dynamically generates waveforms based on incoming note data from the MIDI keyboard. Each waveform, such as sine or square, is produced using a mathematical equation that calculates sample values according to the current note's frequency. This also applies if multiple notes are being played simultaneously, superposing the waveforms. Audio effects are applied directly to the generated waveform using digital signal processing. These effects include reverb, delay, and chorus– reverb uses an echo buffer with feedback, delay is done through sample buffering in the time domain, and chorus uses a modulated delay line with a low-frequency oscillator (LFO) to create an effect of auditory depth. The final audio signal is streamed via I2S to our implemented DAC.

User Interaction & Embedded Integration

Another main function of the software design was to manage user interaction, which is handled through a combination of physical controls and visual feedback elements to benefit the user as they are using the virtual synthesizer. An LCD screen, driven by I2C, provides real-time feedback on what the currently selected waveform is, the volume level on a scale of 0-10, and what audio effect is being modified at the moment. Using a set of digital input buttons, the user can toggle between selecting waveforms and effects, while the potentiometers adjust parameters such as volume and effect intensity.

To enhance user feedback, an LED-based visual feedback system provides life performance data. This system is made of two separate sets of LEDs– the left-hand LED bar consists of six LEDs and shows the number of active notes currently being played, while the right-hand LED bar includes five LEDs that act as frequency bins, showing the frequencies of the notes currently being played.

The embedded software leverages the ESP32-S3's capabilities of multitasking and peripheral interfacing, where we have several different types of protocols working together to make the virtual synthesizer work as intended. As stated before, the LCD screen is integrated via I2C, the potentiometers use analog input from the user, and digital I/O for the buttons and LEDs. The USB MIDI handling is managed through an interrupt-driven approach, where a callback function is triggered only when a user plays a note, reducing CPU usage. Finally, the audio data itself is streamed using Direct Memory Access (DMA) versus CPU-driven audio transfers, as streaming directly from memory to the I2S peripheral reduces CPU load and allows for low-latency output, offering the user the best experience possible.

3 Design Verification

Refer to Table 3 in Appendix A for the requirements and verifications. This section will be dedicated to explaining the purpose and reasoning behind each requirement.

3.1 Microcontroller Verification

The MAX98357A datasheet provides an equation for the voltage output in dBV given by Equation (3).

$$Output\ signal\ level\ (dBV) = input\ signal\ level\ (dBFS) + 2.1\ dB + selected\ amplifier\ gain\ (dB) \quad (3)$$

Observing the output waveform from the MAX98357A will verify that the waveform is generating properly, but in order to test the volume control, an expected voltage amplitude will have to be calculated. From Equation (3), the output voltage can be calculated with a selected amplifier gain of 12 dB and a volume setting of five, which halves the input signal level

$$-3\ dBFS + 2.1\ dB + 12\ dB = 11.1\ dBV$$

$$Vpp = 10^{\frac{11.1\ dBV}{20\ dBV}}\ V = 3.589\ V$$

where V_{pp} is the peak-to-peak voltage of the output of the MAX98357A (i.e. the analog audio waveform).

3.2 Power Verification

The power requirements test voltage level, voltage ripple, and current draw. The voltage must be of accepted ranges for our components, especially the microcontroller, with minimal ripple for stable operation. The current draws are found by summing the expected current draws for each subsystem as listed in Section 2.1.

The requirements related to the 5.0 V power line all failed verification. This was due to mistakes during soldering of the 5.0 V regulator circuit, having swapped the positions of inductor L2 and resistor R23. During this incorrect operation, the feedback resistors R24 and R25 were damaged, reading incorrect resistances, such that the 5.0 V circuit was still non-functional even after the misplaced parts were corrected. This was due to imprecise silkscreen labelling, as seen in Figure 10. The swapped components have overlapping silkscreen labels, making it difficult to determine which component belongs on which footprint.

3.3 User Controls Verification

The potentiometer wipers must provide a voltage range spanning the measurable range of the internal ADCs on the ESP32-S3. Originally, this range was expected to be 0-1.1 V, but after testing with the development board, it was found to be 0-3.3 V. The buttons must provide a digital-low voltage for the ESP32-S3 when in either the on or off state and a digital-high voltage for the other state. It is not critical to specify which state offers which digital voltage, since it is a simple matter to negate the inputs in the program run by the ESP32-S3.

3.4 Display Verification

The LEDs must be distinctly and visibly lit when a digital-high voltage from the ESP32-S3 is supplied. The current through the LEDs have to be within operating ranges such that the LED will turn on but not burn out. The LCD screen must be capable of displaying information transmitted through the I2C bus from the ESP32-S3.

3.5 Audio Output Verification

The speakers must be capable of playing audible sounds according to a driving voltage, supplied by the MAX98357A as determined by the I2S output of the ESP32-S3.

3.6 MIDI Input Verification

The MIDI keyboard must be capable of transmitting MIDI note information through a USB 2.0 cable. A DAW is an available method of testing that capability without requiring a functioning microcontroller to test that capability.

4 Costs

4.1 Parts

Table 1: Bill Of Materials

Part Description	Manufacturer	Price
ESP32-S3 DevKit	Espressif	$15.00
Gikfun 4-ohm 5W stereo speaker	Gikfun	$11.69
6in USB 2.0 A female to USB 4 pin adapter	StarTech	$7.79
Aluminum Briefcase Housing	Toyvian	$25.89
Max98357 I2S DAC Amplifier 5pcs	Teyleten Robot	$13.88
Wall Adapter [120V AC to 12V DC]	GuanTing	$6.99
10k-ohm Potentiometer 2pcs	ECE supply shop	Already owned – $0
220-ohm resistor 6pcs	ECE supply shop	Already owned – $0
Red 1V LED 6x	ECE supply shop	Already owned – $0
SPST button 5x	E-Switch	$8.70

POT Knobs 1/8" diameter ~3/8" hole length	Vishay	$4.70
1/8" shaft diameter 7/8" shaft length	ECE supply shop	$15.56
Voltage Regulator [12V to 3.3V] 10pcs	ANMBEST	$4.99
2V 2mA LED 5pcs	Lumex Opto	$8.85
AOZ1280CI Voltage Regulator	Alpha and Omega Semiconductor	$1.86
TPS54302 Voltage Regulator	Texas instruments	$1.06
ESP32-S3 QFN Chip	Espressif	$3.35
16x2 I2C LCD display 2pcs	Geeekpi	$9.99
MIDI Keyboard	Midiplus	Already owned – $0
C_0805_2012Metric_Pad1.18x1.45mm_HandSolder	Samsung	$0.1
C_0805_2012Metric_Pad1.18x1.45mm_HandSolder	Kemet	$0.32
C_0805_2012Metric_Pad1.18x1.45mm_HandSolder	Murata	$0.45
C_0805_2012Metric_Pad1.18x1.45mm_HandSolder	Samsung	$0.24
C_0805_2012Metric_Pad1.18x1.45mm_HandSolder	Samsung	$0.24
C_0805_2012Metric_Pad1.18x1.45mm_HandSolder	Yageo	0.13
C_0805_2012Metric_Pad1.18x1.45mm_HandSolder	Samsung	$0.56
D_DO-201AD_P15.24mm_Horizontal	STMicro	$0.32
D_DO-201AD_P15.24mm_Horizontal	Diotec Semiconductor	$0.59
MountingHole_3.2mm_M3	N/A	$0
PinHeader_1x04_P2.54mm_Verti	N/A	$0

cal		
PinHeader_1x07_P2.54mm_Vertical	N/a	$0
PhoenixContact_GMSTBVA_2,5_2-G_1x02_P7.50mm_Vertical	On Shore Technology	$0.69
PhoenixContact_GMSTBVA_2,5_3-G_1x03_P7.50mm_Vertical	On Shore Technology	$1.92
PhoenixContact_MCV_1,5_3-G-3.5_1x03_P3.50mm_Vertical	Wurth Elektronik	$2.12
PhoenixContact_MCV_1,5_2-G-3.5_1x02_P3.50mm_Vertical	Phoenix Contact	$2.25
PhoenixContact_MCV_1,5_5-G-3.5_1x05_P3.50mm_Vertical	Phoenix Contact	$2.36
L_0805_2012Metric_Pad1.05x1.20mm_HandSolder	Abracon	$0.1
L_0805_2012Metric_Pad1.05x1.20mm_HandSolder	TDK	$0.1
L_0805_2012Metric_Pad1.05x1.20mm_HandSolder	TDK	$0.1
R_0805_2012Metric_Pad1.20x1.40mm_HandSolder	Yageo	$0.5
R_0805_2012Metric_Pad1.20x1.40mm_HandSolder	ROHM Semiconductor	$1.25
R_0805_2012Metric_Pad1.20x1.40mm_HandSolder	Vishay	$0.1
R_0805_2012Metric_Pad1.20x1.40mm_HandSolder	Vishay	$0.24
R_0805_2012Metric_Pad1.20x1.40mm_HandSolder	Panasonic	$0.52
R_0805_2012Metric_Pad1.20x1.40mm_HandSolder	Panasonic	$0.36
R_0805_2012Metric_Pad1.20x1.40mm_HandSolder	Vishay	$0.2
R_0805_2012Metric_Pad1.20x1.40mm_HandSolder	Vishay	$0.55

R_0805_2012Metric_Pad1.20x1.40mm_HandSolder	Panasonic	$0.1
R_0805_2012Metric_Pad1.20x1.40mm_HandSolder	Stackpole Electronics	$0.1
R_0805_2012Metric_Pad1.20x1.40mm_HandSolder	Vishay	$0.16
R_0805_2012Metric_Pad1.20x1.40mm_HandSolder	KOA Speer	$0.18

4.2 Labor

To calculate our labor cost, we are taking an estimate of 200 labor hours at an hourly pay rate of $25 an hour. Using the necessary equation, the labor costs come out to:

$$\frac{\$25}{hr} * \ 2.5 \ * 200 \, hours \ = \ \$12,500$$

Below is a general layout of our schedule with the time frame, specific tasks, and division of labor specified. We are only mentioning the time we have left in project creation– from the creation of this design document until the end of the semester.

Table 2. Weekly Schedule of Team 59 members.

Time Frame	Task	Division of Labor
Week of 3/3	Design document PCB design Breadboard Progress reports	Patrick/Dylan Everyone Connor Everyone
Week of 3/10	Breadboard Demo PCB design Finish power subsystem Working speakers	Everyone Everyone Dylan/Patrick Connor/Patrick
Week of 3/24	PCB design MIDI keyboard interfacing Synth development	Everyone Connor Everyone
Week of 3/31	PCB design Progress reports Finish other instruments	Everyone

Week of 4/7	Final PCB order if needed User interface (LED/LCD)	Everyone
Week of 4/14	Team Contract assessment Housing for PCB and peripherals	Everyone
Week of 4/21	Mock Demo	Everyone
Week of 4/28	Final Demo	Everyone
Week of 5/5	Final Presentation	Everyone

5 Conclusion

5.1 Accomplishments

Although our custom PCB ended up not working, we were still able to meet all of our high-level requirements using the ESP32-S3 DevKit. We successfully achieved polyphony with more than the originally promised 4 notes and extended that up to 8, allowing users to play chords without any noticeable lag or audio glitches. The synthesizer maintains smooth and accurate note playback across a full range of 128 notes, which is a much larger range than the original 3 octaves we proposed. All user controls, including volume adjustment and instrument switching, functioned reliably through the DevKit setup. The system also gave clear visual feedback using the LCD screen and LEDs, enhancing usability. Additionally, we achieved the reach goals of incorporating an extra set of LEDs for user feedback as well as accounting for note velocity in our feedback. Overall, even without the PCB, the core functionality of our design remained fully intact and user-friendly.

5.2 Uncertainties

Two key challenges we faced during the project were related to our PCB design and managing processing power for audio effects. Our PCB didn't function as intended due to a floating pull-up pin, which prevented the chip from communicating with the rest of our circuit. However, we identified the issue through testing and were able to pivot quickly by using the DevKit to keep the project on track. We also ran into performance limitations when applying multiple pitch-based effects at once, which pushed the processing limits of the microcontroller. This helped us better understand the constraints of real-time audio processing and led us to prioritize effects that could run smoothly without sacrificing audio quality. Both challenges ultimately gave us valuable insight and made the final system more reliable and efficient.

5.3 Ethical considerations

Intellectual Property and Open-Source Usage

Our virtual synthesizer relies on digital sound processing algorithms and MIDI communication protocols, some of which are covered under existing patents or open-source licenses. To ensure compliance with ethical standards, we carefully reviewed and adhered to any applicable open-source licensing agreements when using third-party code, libraries, or reference designs. Proper credit has been given where required, and we ensured that our project does not go against copyrighted assets/technologies.

Accessibility and Affordability

One of the main ethical thoughts behind this project is to make music production more accessible and affordable for individuals who may not have the financial support to purchase high-end virtual synthesizers or digital audio workstations (DAWs). By offering a low-cost, standalone synthesizer, we aimed to remove financial barriers for aspiring musicians while ensuring that the device remains user-friendly for those without advanced music creation experience.

Responsible Data Handling

Although our synthesizer does not collect personal user data, any future implementation of features such as user presets, saved settings, or DAW integration may require minimal data storage. In such a case, we will follow ethical guidelines for data privacy and user transparency, ensuring that any stored information remains local to the device and is not transferred or used in any other situations without the user's consent.

5.4 Future work

In the future, if this project were continued or expanded, we would work on getting a PCB mounted in the briefcase housing with standoffs and screws (drilled through the back panel). Additionally, more instrument waveforms and effects could be added to the current implemented roster. Possibly, this could include using an external memory to store wavetables for engineered sounds sourced from other works. An auxiliary connector could be included such that an external audio device (e.g. headphones or speakers) could be plugged in and play the audio output, while silencing the built-in speaker.

References

[1] Maxim Integrated Products, Inc., “Tiny, Low-Cost, PCM Class D Amplifier with Class AB Performance,” 2019. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/MAX98357A-MAX98357B.pdf

[2] Espressif Systems Co., “ESP32S3 Series Datasheet,” www.espressif.com, 2022. https://cdn-shop.adafruit.com/product-files/5477/esp32-s3_datasheet_en.pdf

[3] Shenzhen Anxinke Technology Co., “ESP-32S Datasheet,” Oct. 2016. Available: https://www.es.co.th/Schemetic/PDF/ESP32.PDF

[4] Espressif Systems Co., “ESP-IDF Programming Guide,” www.docs.espressif.com, 2016. https://docs.espressif.com/projects/esp-idf/en/v5.2.5/esp32s3/index.html

[5] Espressif Systems Co., “Schematic Checklist,” www.docs.espressif.com, 2023. https://docs.espressif.com/projects/esp-hardware-design-guidelines/en/latest/esp32s3/schematic-checklist.html

[6] Espressif Systems Co., “ESP32-S3 Technical Reference Manual Version 1.6,” 2024. Available: https://www.espressif.com/sites/default/files/documentation/esp32-s3_technical_reference_manual_en.pdf

[7] “Z-0234,” 52pi.com, 2022. https://wiki.52pi.com/index.php?title=Z-0234#1602_Serial_LCD_Module_Display

[8] Espressif Systems Co., “Inter-Integrated Circuit (I2C),” www.docs.espressif.com, 2016. https://docs.espressif.com/projects/esp-idf/en/v5.4/esp32s3/api-reference/peripherals/i2c.html

[9] Espressif Systems Co., “Inter-IC Sound (I2S),” www.docs.espressif.com, 2016. https://docs.espressif.com/projects/esp-idf/en/v5.4/esp32s3/api-reference/peripherals/i2s.html

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

[11] touchgadget, “GitHub - touchgadget/esp32-usb-host-demos: ESP32S2 Arduino USB host printer, MIDI, and keyboard demos,” GitHub, 2021. https://github.com/touchgadget/esp32-usb-host-demos/tree/main (accessed May 08, 2025).

Appendix A Requirement and Verification Table

Table 3: System Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
When the MAX98357A output pins are connected across a passive 4 Ω speaker, and the microcontroller is set to output a digital sine wave of 440 Hz with a MIDI signal of maximum velocity at volume setting 5 through the 32-bit I2S bus, the voltage across the output should show a 3.578 ± 5% Vpp amplitude sine wave of 440 ± 5 Hz.	Using an oscilloscope, attach the leads across the OUT_P and OUT_N pins (equivalently, attach to the speaker terminals). Set the ESP32-S3 to constantly emit the waveform specified in the requirement (this may be achieved through the other subsystems or by specially programming the ESP32-S3). A sine wave should be visible on the oscilloscope. Using the cursors on the oscilloscope, find the voltage difference peak to peak. To find the frequency, use the cursors to measure the time difference between two adjacent positive peaks. This represents the period of the waveform. Taking the inverse of this period will provide the frequency.	Y
The wall adapter must interface with a North American 120V 60Hz AC outlet and provide 12 ± 2 VDC (the tolerance is large because it only must satisfy the minimum input voltage to the voltage regulators).	Use a voltmeter to measure the output voltage relative to ground. A barrel plug connector is provided with the adapter, the multimeter terminals may contact the terminals of the connector.	Y
The 3.3 V regulated line must supply 3.3 ± 0.3 V, consistent with the tolerance of the ESP32-S3.	As this is a DC voltage, either an oscilloscope or voltmeter will suffice to measure the output voltage relative to ground.	Y
The 3.3 V regulated line must have a voltage ripple less than 50 mV.	Using an oscilloscope, connect the terminals to the circuit ground and the 3.3V line. Use cursors on the oscilloscope to find the difference of peaks in the voltage waveform.	Y
The 3.3 V regulated line must be capable of outputting at least 600mA.	After verifying the output voltage of the regulator, put a resistor in series with the output such that the current draw should be 600mA. The resistor value is calculated as $R \leq \frac{V}{600\ mA}$. Use a multimeter to measure the current through the resistor.	Y

The 5.0 V regulated line must supply 5.0 ± 5% V, consistent with the ratings for USB 2.0.	As this is a DC voltage, either an oscilloscope or voltmeter will suffice to measure the output voltage relative to ground.	N
The 5.0 V regulated line must have a voltage ripple less than 50 mV.	Using an oscilloscope, connect the terminals to the circuit ground and the 5.0V line. Use cursors on the oscilloscope to find the difference of peaks in the voltage waveform.	N
The 5.0 V regulated line must be capable of outputting at least 2300 mA.	After verifying the output voltage of the regulator, put a resistor in series with the output such that the current draw should be 2300mA. The resistor value is calculated as $R \leq \frac{V}{2300\ mA}$. Use a multimeter to measure the current through the resistor.	N
The 3.3 V and 5.0 V regulated lines must be capable of outputting their required currents listed above (600 mA and 2300 mA) simultaneously.	After verifying the current draws of each individual voltage regulator as listed above, use the same setup as in the prior tests (same resistor values) with both resistors going to a common ground. Measuring the current through both resistors, separately.	N
The potentiometer should output a voltage of 3.3 ± 5% at its maximum value, and a voltage value of 0 ± 0.1 V at its minimum value.	Supply 3.3V to the user controls with a DC power supply. Using a voltmeter, attach one lead on the wiper and the other lead on the grounded terminal of the potentiometer. Turn the potentiometer to its maximum angle both ways, measuring the DC voltage across the wiper.	Y
The voltage across each button is 0-0.9V when the button is pressed, and greater than 2.4 V when the button is not pressed.	Supply 3.3V to the user controls with a DC power supply. Attach the leads of a voltmeter across a button. Record the voltage as the button is pressed and not pressed. Repeat for each button.	Y
Must be capable of transmitting MIDI information through a USB 2.0 interface.	Plug the USB cable into a computer running a MIDI-compatible DAW (digital audio workstation). Follow program instructions to connect the MIDI keyboard. The MIDI keyboard should be capable of playing sound through the DAW.	Y
Must be capable of displaying programmable characters to the LCD screen through the I2C bus.	When interfaced with a source of I2C data (e.g. the ESP32-S3) and provided power, the display should show the characters being written to it.	Y

The LEDs should light up for voltages within 3.0 and 3.6 V. (Applied across the LED and its current limiting resistor).	Using a DC power supply, power the diode in series with its current limiting resistor. At supply voltages of 3.0 and 3.6 V, the diode must light up.	Y
The current allowed through the LED must be between 5-20 mA with a voltage of ~3.3V (from the ESP32-S3) across the diode and its current limiting resistor.	Using a DC power supply, power the diode and its resistor with 3.6V. Use the multimeter to measure the voltage across the diode, using Ohm's Law to find the current through the diode.	Y
Must be capable of playing a pure sine wave of 440 Hz. The frequency of the produced sound should be within ±2 Hz of the driving voltage.	Using an ADALM2000 and Scopy, generate a 5 V pure sine waveform of 440 Hz. Connect the ADALM2000 P+ and P- terminals across the speaker terminals. When running the waveform, the speaker should play an audible sine wave. The frequency can be verified with an instrument tuning phone application.	Y
Must be capable of playing a square wave of 440 Hz and a duty ratio of 50%. The frequency of the produced sound should be within ±2 Hz of the driving voltage.	Using an ADALM2000 and Scopy, generate a 5 V square waveform of 440 Hz and a 50% duty ratio. Connect the ADALM2000 P+ and P- terminals across the speaker terminals. When running the waveform, the speaker should play an audible sine wave. The frequency can be verified with an instrument tuning phone application.	Y

Appendix B PCB Layout

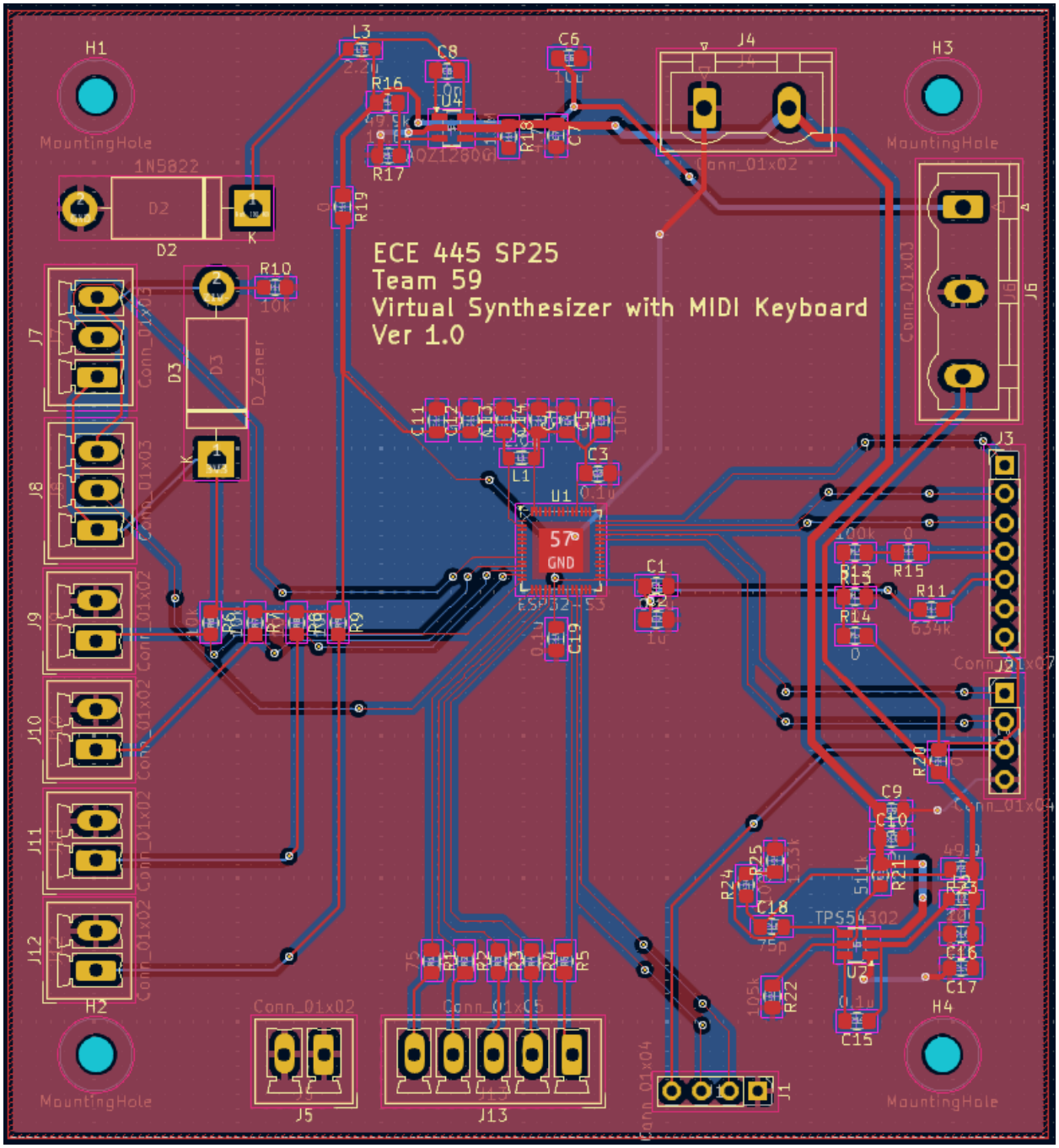

Figure 10. The KiCad PCB layout for the completed PCB (which was non-functional).