**ECE 445**

SENIOR DESIGN LABORATORY

DESIGN DOCUMENT

---

# Virtual Synthesizer

Using MIDI Keyboard

---

**Team No. 59**

Dylan Pokorny
[dylangp2@illinois.edu]

Connor Barker
[cbarker4@illinois.edu]

Patrick Ptasznik
[pptas2@illinois.edu]

**TA:** Eric Tang

**Professor:** Yang Zhao

March 1, 2025

# Table of 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 Visual Aid

To add more creativity and uniqueness to our virtual synthesizer, we decided to house it in a small briefcase, which is then customized to give a ‘boombox’ feel. The image shows the main idea of how our synthesizer will be laid out. Bear in mind that the speakers, LED’s, and user controls shown in the visual aid are from the backside, meaning that their functional sides will be facing outwards(on the outer side of the briefcase) for the user to see when using the virtual synthesizer.

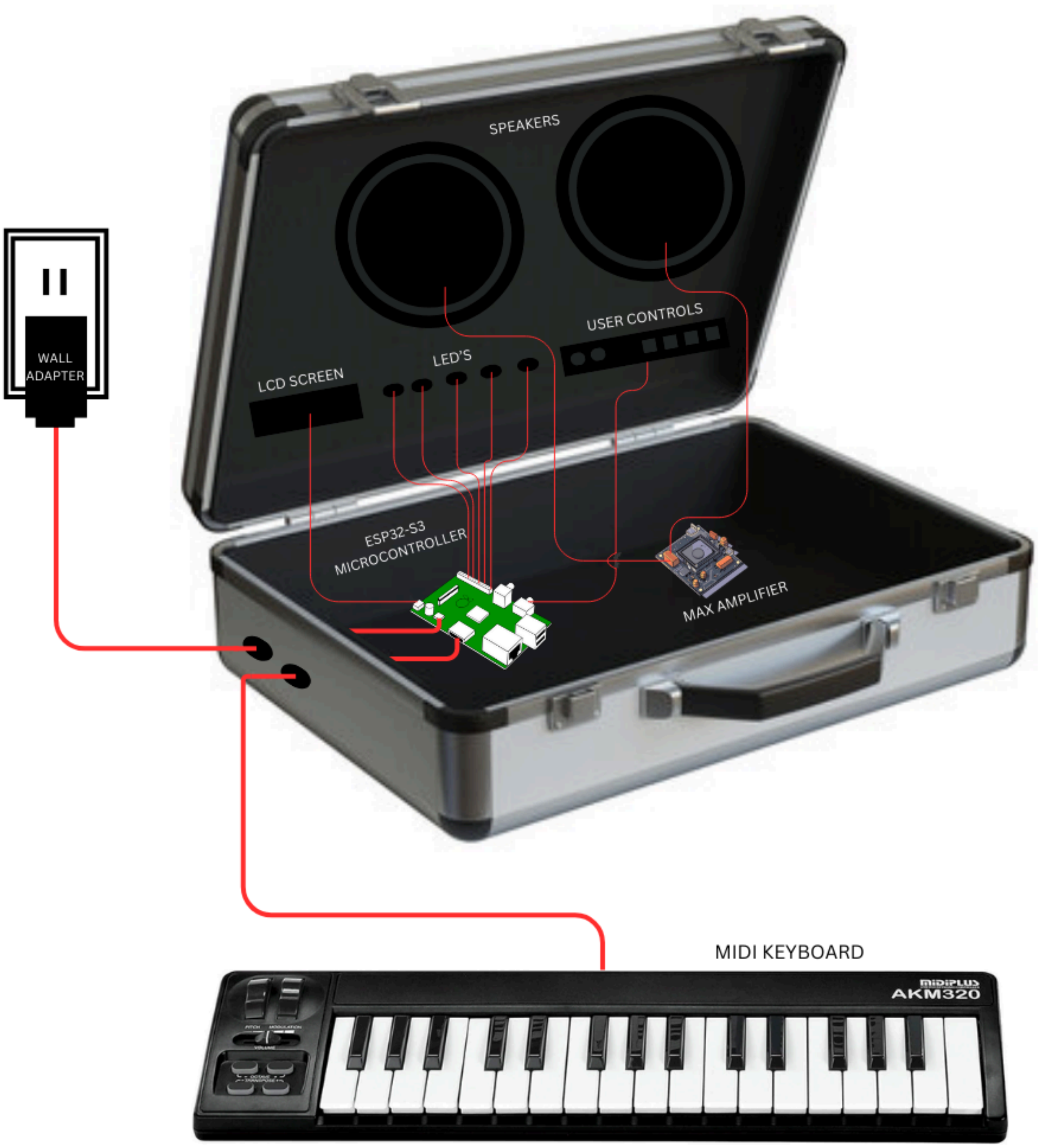

**Figure 1: Visual Aid of the Virtual Synthesizer**

## 1.4 High-Level Requirements

### Instrument Simulation and Sound Generation

The main functionality of our synthesizer is to accurately simulate a variety of instruments while maintaining high audio quality and low latency. It will support fundamental waveforms such as sine, saw, square, and triangle. This will provide a broad spectrum of synthetic tones for the user to work and experiment with. On top of these fundamentals, real-world instrument sounds such as piano and flute will be integrated, expanding the synthesizer's versatility for different musical styles and tastes. Each instrument preset will be designed to offer clear and distinguishable sound profiles, ensuring a diverse and enjoyable music creation experience. The synthesizer will process MIDI keyboard inputs with no humanly noticeable latency and a high signal-to-noise ratio, closely matching the audio quality of professional virtual synthesizers on today's market.

### Polyphony and Chord Support

Our synthesizer will support at least four simultaneous notes, allowing users to play chords and layered harmonies, which are essential for advanced music production. This requires efficient handling of multiple MIDI inputs without interference between one another, ensuring that pressing multiple keys at once does not cause audio dropouts or glitches. Regardless of note order, speed, or combination, the system will maintain smooth and uninterrupted sound output. Just as with instrument simulation, there will be no noticeable latency, even at maximum polyphony, ensuring a responsive playing experience similar to what a professional-grade synthesizer would offer.

## Octave Range and Note Accuracy

To provide an extra level of flexibility, the synthesizer will support a minimum range of three octaves, giving users access to both melodic and harmonic structures. Each note within this octave range will be precisely mapped to its correct MIDI frequency, ensuring the user will receive their desired pitch. Transitions between octaves will be seamless, with no unexpected jumps or distortions in sound. The polyphony system will be optimized in such a way that playing speed and multiple-note combinations do not affect the clarity and/or accuracy of the output, ensuring a natural and high-quality playing experience.

# 2 Design

## 2.1 Physical Design

As shown in the visual aid, the entire virtual synthesizer is housed in a customized briefcase made to resemble a retro-styled ‘Boombox’. As shown in the Visual Aid section, the synthesizer will receive power from a regular wall outlet using a 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 holes cut into the side of the briefcase.

Moving onto the inside of the briefcase– this is the ‘backend’ of the synthesizer– the inside contains our ESP32-S3 Microcontroller, MAX98357 I2S DAC 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, etc. Last but not least, the LED’s are displayed here, which count how many keys are being played, making the synthesizer more sophisticated and aesthetically pleasing.

## 2.2 Block Design

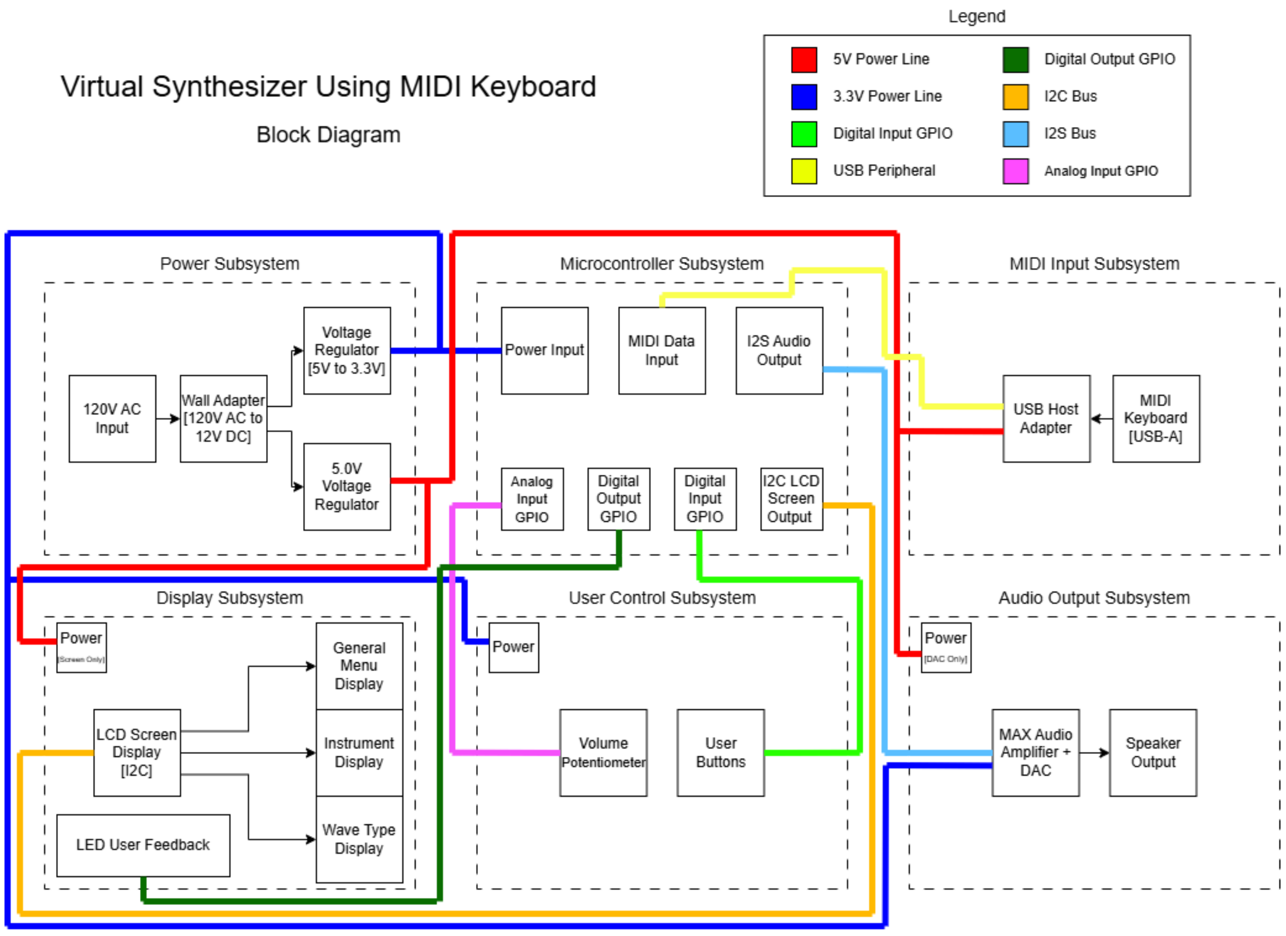

**Figure 2: Block Diagram for Virtual Synthesizer**

## 2.3 Subsystems

### 2.3.1 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.

**I2S Peripheral on ESP32-S3 Microcontroller**

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 if problems arise in our implementation of the converter, as PDM is extremely simple to convert to analog: a low-pass filter is all that is required.

**I2S DAC & Amplifier**

The I2S bus will interface with our MAX98375A, which is an I2S DAC and Amplifier. The MAX will convert the digital I2S signal into an analog audio waveform capable of being output onto speakers. The MAX 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 MAX also offers fixed amplifier gains: 15, 12, 9, 6, and 3 dB. As the MAX has an output power rating of 3.2 Watts, we must select an amplifier gain of 9dB. See the Risk Analysis section for calculations.

Following the MAX98357A datasheet, to run the DAC in (left/2 + right/2) mode, we must connect the SD_MODE pin to the logic voltage level (3.3V) through a large pullup resistor (634 kΩ). To select the amplifier gain of 9dB, we must leave the GAIN_SLOT pin unconnected.

**Microcontroller Requirements**

| **Requirement** | **Verification** |
|---|---|
| When the MAX98357A output pins are connected across a passive 4Ω speaker, and the microcontroller is set to output a digital sine wave of 440Hz with a PCM amplitude of the maximum word value ($2^{31}$) through the 32-bit I2S bus, the voltage across the output should show a 3.578 ± 5% V 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. A sine wave should be visible on the oscilloscope. Using the cursors on the oscilloscope, find the voltage difference peak to peak and divide by 2 to find the amplitude. 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. |

### 2.3.2 Power Subsystem

The power subsystem draws power from a 120V 60Hz AC wall outlet using a 12V adapter. The 12V supply will be regulated to 3.3V and 5.0V using voltage regulators. The 5.0V regulator is necessary because of the adapter's unspecified tolerance. The regulated 5.0V line will power the USB interface to the MIDI keyboard, the LCD screen for user interface, and the I2S DAC in the audio output subsystem. The 3.3V 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 3.3V and 5.0V 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 3.3V and 5.0V regulators. The ESP32-S3 has a typical draw of 500mA, and another 100mA can be allocated for the user controls and DAC mode select, which don't take much current.  The 3.3V regulator must be capable of drawing at least 600mA of current.

The MIDI keyboard is powered through USB 2.0, which has a rated maximum current draw of 500mA. 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 200mA 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 1.6A. Then, our 5.5V regulator requires a current rating for at least 2.3A.

**Power Requirements**

| Requirement | Verification |
|---|---|
| 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). | As this is a DC voltage, either an oscilloscope or voltmeter will suffice 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. |
| The 3.3V 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. |
| The 3.3V 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. |
| The 3.3V regulated line must be capable of | After verifying the output voltage of the |

| | |
|---|---|
| outputting at least 600mA. | regulator, put a resistor in series with the output such that the current draw should be 600mA. The resistor value is calculated as $R \geq \frac{V}{600\, mA}$. Use a multimeter to measure the current through the resistor. |
| The 5.0V 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. |
| The 5.0V 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. |
| The 5.0V regulated line must be capable of outputting at least 2300mA. | 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 $R \geq \frac{V}{2300\, mA}$. Use a multimeter to measure the current through the resistor. |
| The 3.3V and 5.0V regulated lines must be capable of outputting their required currents | After verifying the current draws of each individual voltage regulator as listed above, |

| | |
|---|---|
| listed above (600 mA and 2300mA) simultaneously. | 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. |

### 2.3.3 User Controls Subsystem

The user controls receive 3.3V from the power subsystem to operate. The synthesizer will use a potentiometer for user control, allowing for a nearly* continuous volume control range. Because the potentiometer sends a signal to the built-in 12 bit ADC on the microcontroller, which operates within 0-1.1V, it requires an additional step-down. A 1.1V diode will fulfill this purpose, with a large resistor to limit current to ground. Additionally, multiple buttons will be used to navigate through menus related to instrument selection and post-effects such as distortion. Each button will be connected to pins on the ESP32-S3, which will poll for activation. The buttons will be powered directly from the 3.3V power line.

*nearly continuous, because the potentiometer signal will be converted through a 12-bit ADC and the volume will be adjusted digitally.

**User Controls Requirements**

| Requirement | Verification |
|---|---|
| The potentiometer should output a voltage between 1.0-1.1V at its maximum value, and a voltage value of 0-0.1V 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. |
| The voltage across each button is 0-0.9V when the button is not pressed, and greater than 2.4 V when the button is 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. |

### 2.3.4 MIDI Keyboard Subsystem

The user's core input will be taken through our MIDI keyboard subsystem. 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 $V_{CC}$ and GND pins are connected to the 5.0V 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.

**MIDI Keyboard Requirements**

| Requirement | Verification |
|---|---|
| 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. |

### 2.3.5 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, 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 5V power from the power subsystem. Finally, to enhance user feedback, we are adding five LEDs that will be responsible for counting how many keys are being played. Each LED channel includes a current-limiting resistor to prevent damage to components and ensure safe operation. This resistor should be large, to reduce the power drawn from the ESP32-S3, as power consumed follows $P = I^2R$.

**Display Requirements**

| Requirement | Verification |
|---|---|
| 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. |
| The LEDs should light up for voltages within 3.0 and 3.6 V. | 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. |
| The current allowed through the LED must be between 10-20 mA with a voltage of 3.6V 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. |

### 2.3.6 Audio Output Subsystem

The audio output subsystem is responsible for receiving the previously amplified electrical signal which will drive a 4-Ohm speaker rated between 3.2W and 5W, ensuring sufficient volume while maintaining a high level of sound clarity with no unnecessary noise.

**Audio Output Requirements**

| Requirement | Verification |
|---|---|
| Must be capable of playing a 5 V pure sine wave of 440 Hz. The frequency of the produced sound should be within ±1 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. |
| Must be capable of playing a 5 V square wave of 440 Hz and a duty ratio of 50%. The frequency of the produced sound should be within ±1 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. |

## 2.4 Risk Analysis

Our analog audio waveform is generated in the MAX98357A and sent to the speakers. The speakers have a power rating of 5W, but the MAX has a power rating of 3.2W, meaning that the MAX98357A is our limiting power rating. The MAX additionally amplifies the analog signal it generates, so we must select a gain that will not exceed the 3.2W rating, or else the MAX may incur damage. From the MAX98357A datasheet, the equation for output signal level is given:

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

where 0dBFS is referenced to 0dBV, and the output signal is across the OUT_P and OUT_N pins. We will be using the I2S bus in 32 bit mode. I2S is a PCM signal in two’s complement, meaning that our full scale is $2^{31}$ (imagining that every bit is a 1 for the entire waveform except the sign bit). The full scale represents a maximum amplitude that is constant for the entire waveform.

$$dBFS\ =\ 20log(\frac{DV_{RMS}}{2^{31}})$$

where $DV_{RMS}$ is the RMS value of the digital 32-bit data representing our audio waveform. Because we are outputting an audio waveform, we will be using RMS values in our calculations.

The MAX98357A is rated for 3.2W, and our speakers are rated as having an impedance of 4Ω. Calculating the maximum RMS voltage that the MAX can produce,

$$P\ =\ \frac{V^2}{R}\quad \Rightarrow \quad V_{max} = \sqrt{PR}\ =\ \sqrt{(3.2\ W)(4\ \Omega)}\ =\ 3.578\ V$$

Next we calculate the dBV level of the maximum RMS voltage across the output.

$$V_{max}(dBV)\ =\ 20log(\frac{V_{max}}{1})\ =\ 11.07\ dBV$$

Setting $V_{max}$(dBV) equal to the output level expression from the datasheet, we can solve for our maximum $DV_{RMS}$ in terms of our selected amplifier gain $A_V$.

$$DV_{RMS} = 2^{31} \times 10^{\frac{V_{max}(dBV)-2.1dB-A_V}{20}}$$

The MAX98357A supports five different gain settings. Evaluating $DV_{RMS}$ at a $A_V$ = 9dB reveals

$$A_V = 9\,dB \quad \Rightarrow \quad DV_{RMS} = 2^{31} \times 0.997$$

This shows that, when operating with a selected amplifier gain of $A_V$ = 9dB, the maximum RMS value of the 32-bit PCM signal representing our audio waveform is nearly a unity multiple of our full scale. This indicates that with a selected gain of 9dB, We couldn't exceed our output power rating unless every bit were a 1 for the entire waveform. Using a lower gain would prevent us from utilizing as much power in our output. Thus, we should select the 9dB amplifier gain in order to protect the MAX98357A.

## 2.5 Tolerance Analysis

The current limiting resistors in series with our output LEDs must limit current sufficiently such that the LEDs operate between 10-20 mA of current. The output voltage of the ESP32-S3 may vary between 3.0-3.6V. If the forward voltage of each LED is represented by $V_{FWD}$, then the maximum value of the resistor is given by

$$R = \frac{3.6 - V_{FWD}}{0.01} \text{ in } \Omega.$$

The minimum value then becomes

$$R = \frac{3.0 - V_{FWD}}{0.02} \text{ in } \Omega.$$

The MIDI keyboard doesn't have an available datasheet, but it is designed to be powered by USB 2.0. The power specifications for USB 2.0 are 5 ± 5% V and a maximum current of 500mA. Thus, the 5.0 V regulator in the power subsystem must regulate to at least this constraint, as it is the only guaranteed operating range of the MIDI keyboard.

The 3.3 V regulator in the power subsystem is in charge of the microcontroller. Because of this, any fluctuation in the power line may be observed in outputs from the ESP32-S3. This includes the 3.3 V line which uses a pullup resistor of 634 kΩ to select the appropriate mono-channel setting of the I2S DAC, as the datasheet for the MAX98357A prescribes to use the digital logic level. Thus the 3.3 V regulator is constrained by the operating range of the microcontroller, which is 3.0-3.6V.

The user buttons must provide a digital high when pressed and a digital low when not pressed. While the ESP32-S3 datasheet doesn't seem to list digital terminal characteristics, other ESP32 datasheets do. From these, we find that logic high is below $0.25V_{DD}$, and a logic high is above $0.8V_{DD}$. Because $V_{DD}$ varies with the tolerance of the 3.3V regulator of being between 3.0-3.6 V, the button values may vary by the following equations.

$$V_{Pressed,min} = 0.8V_{DD,min} = 0.8 \times 3.0 = 2.4\,V$$

$$V_{Released,max} = 0.25V_{DD,max} = 0.25 \times 3.6 = 0.9\,V$$

# 3 Cost & Schedule

## 3.1 Cost Analysis

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:

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

On top of labor, we must take into account all the parts and components used to build our virtual synthesizer:

| Part Description | Manufacturer | Price | Link |
|---|---|---|---|
| ESP32-S3 DevKit | Espressif | $15.00 | Link |
| Gikfun 4-ohm 5W stereo speaker | Gikfun | $11.69 | Link |
| 6in USB 2.0 A female to USB 4 pin adapter | StarTech | $7.79 | Link |
| Max98357 I2S DAC Amplifier 5pcs | Teyleten Robot | $13.88 | Link |
| Wall Adapter [120V AC to 12V DC] | GuanTing | $6.99 | Link |
| Voltage Regulator [12V to 3.3V] 10pcs | ANMBEST | $4.99 | Link |
| 16x2 I2C LCD display 2pcs | Geeekpi | $9.99 | Link |
| MIDI Keyboard | Midiplus | Already owned – $0 | Link |
| | | Parts Total: **$70.33** | |

**Figure 3: Itemized List of Components**

Summing up the labor cost and component cost, our synthesizer has a grand total of $12,570.33.

## 3.2 Schedule

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.

| Time Frame | Task | Division of Labor |
|---|---|---|
| Week of 3/3 | Design document<br>PCB design<br>Breadboard<br>Progress reports | Patrick/Dylan<br>Everyone<br>Connor<br>Everyone |
| Week of 3/10 | Breadboard Demo<br>PCB design<br>Finish power subsystem<br>Working speakers | Everyone<br>Everyone<br>Dylan/Patrick<br>Connor/Patrick |
| Week of 3/24 | PCB design<br>MIDI keyboard interfacing<br>Synth development | Everyone<br>Connor<br>Everyone |
| Week of 3/31 | PCB design<br>Progress reports<br>Finish other instruments | Everyone |
| Week of 4/7 | Final PCB order if needed<br>User interface (LED/LCD) | Everyone |
| Week of 4/14 | Team Contract assessment<br>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 |

**Figure 4: Schedule for Duration of Project Development**

# 4 Ethics & Safety

## 4.1 Ethical Concerns

Intellectual Property and Open-Source Usage – Our virtual synthesizer will rely on digital sound processing algorithms and MIDI communication protocols, some of which may be covered under existing patents or open-source licenses. To ensure compliance with ethical standards, we will carefully review and adhere to any applicable open-source licensing agreements when using third-party code, libraries, or reference designs. Proper credit will be given where required, and we will ensure 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 aim 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 will 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.

### 4.2 Safety Concerns

Electrical Safety – Since the synthesizer is powered through a 120V AC wall adapter, proper voltage regulation, and insulation are going to be necessary to prevent electrical hazards. The buck converter and voltage regulators used in the system must be rated appropriately to ensure a stable power supply to the ESP32-S3, MIDI keyboard, and speakers. Additionally, all circuit components will be securely housed to prevent accidental short circuits or user exposure to high-voltage components.

Hearing Protection and Volume Control – Our synthesizer will output audio through a 3W-5W speaker, which, at maximum volume, could potentially reach unsafe listening levels, especially for long durations of time. To mitigate this risk, we will implement software-based volume limits to prevent excessive loudness. Additionally, users will have precise control over volume via a potentiometer to set the volume to a level they are comfortable with, which ultimately also reduces the risk of unintended loud sounds.

Compliance with University Lab Safety Policies – We will adhere to all of the University of Illinois' applicable electronics safety and laboratory safety guidelines. This includes proper use of PPE and safe handling of electrical components, and maintaining an organized workspace to minimize hazards and potential accidents.

## 5 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