ECE 445

SENIOR DESIGN LABORATORY

PROJECT PROPOSAL

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

February 13, 2025

1 Introduction

1.1 Problem

Music production has a high barrier to entry for beginners due to expensive equipment, complex software, and the need for music theory knowledge. To begin, industry VSTs (Virtual Studio Technology) prices are out of reach for people beginning their journey into music production. Popular VSTs such as Nexus ($120), Omnisphere ($499), and ElectraX ($150) require the users to be willing to spend hundreds of dollars to get a quality wavetable. Additionally, learning music production without proper hardware is difficult without tools to practice musical ideas. Without practice instrumentation, forming mature musical pieces and modifying them intelligently is challenging. Songs might sound random, offbeat, or messy because of poor experimentation and modifications. Furthermore, beginners struggle with mixing because they don’t know how to balance sounds properly. Some sounds might be too loud, while others get lost, making the music unclear. Using effects like reverb too much can also make the mix sound messy instead of clean.

1.2 Solution

To lower the barrier for beginners in music production, our synthesizer combines 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, making the learning process more intuitive. Customizable sounds and effects let users shape their own tones and explore sound design in a hands-on way. The synthesizer offers multiple instrument options, such as synths, flutes, and pianos, enabling users to experiment with

different sounds for various musical styles. A built-in screen displays real-time information about the selected instrument and its effects, helping beginners understand how different elements shape sound. With its affordable price, this synthesizer makes music production accessible to more people, providing a hands-on way to learn without relying on expensive software or complicated setups.

1.3 High-Level Requirements

Instrument Simulation and Sound Generation – The primary goal of this project is to successfully simulate a variety of instruments, including fundamental waveforms such as sine, saw, square, and triangle. To enhance the complexity and versatility of our synthesizer, we will also integrate real-world instrument sounds, such as piano and flute. Each instrument preset will be distinct, ensuring clear differentiation between sound profiles. Our objective is to provide a seamless playing experience using a MIDI keyboard as input, with no humanly noticeable latency and a high signal-to-noise ratio, delivering audio quality comparable to professional virtual synthesizers available in today's market.

Polyphony and Chord Support – Another main goal of our virtual synthesizer is to be capable of playing at least four notes simultaneously to support advanced music creation. This requires efficient handling of multiple inputs from the MIDI keyboard at once, as we must ensure that the keys do not interfere with each other. Regardless of in what order or how quickly the specific note buttons are pressed, the system will maintain smooth and uninterrupted audio output through the speakers. As specified above for instrument sound and sound generation, there will be no noticeable latency on note presses, regardless of what amount of notes the user is pressing

consecutively(up to our synthesizer simultaneous note limit). This will ensure a natural and responsive playing experience, resembling professional virtual synthesizers.

Octave Range and Note Accuracy – Our last high-level requirement focuses on supporting a playable range of at least three octaves. This will provide the user with a resourceful palette to create music with both melodies and harmonic structures. Each note within this three(or more) octave range will be accurately mapped to its corresponding MIDI frequency, ensuring precise pitch reproduction. The synthesizer will also handle transitions between octaves in a way where there are no unintended distortions or jumps in the outputted audio. Playing speed and note combinations using our polyphony system will not have an effect on the notes, always ensuring a clean experience for the user.

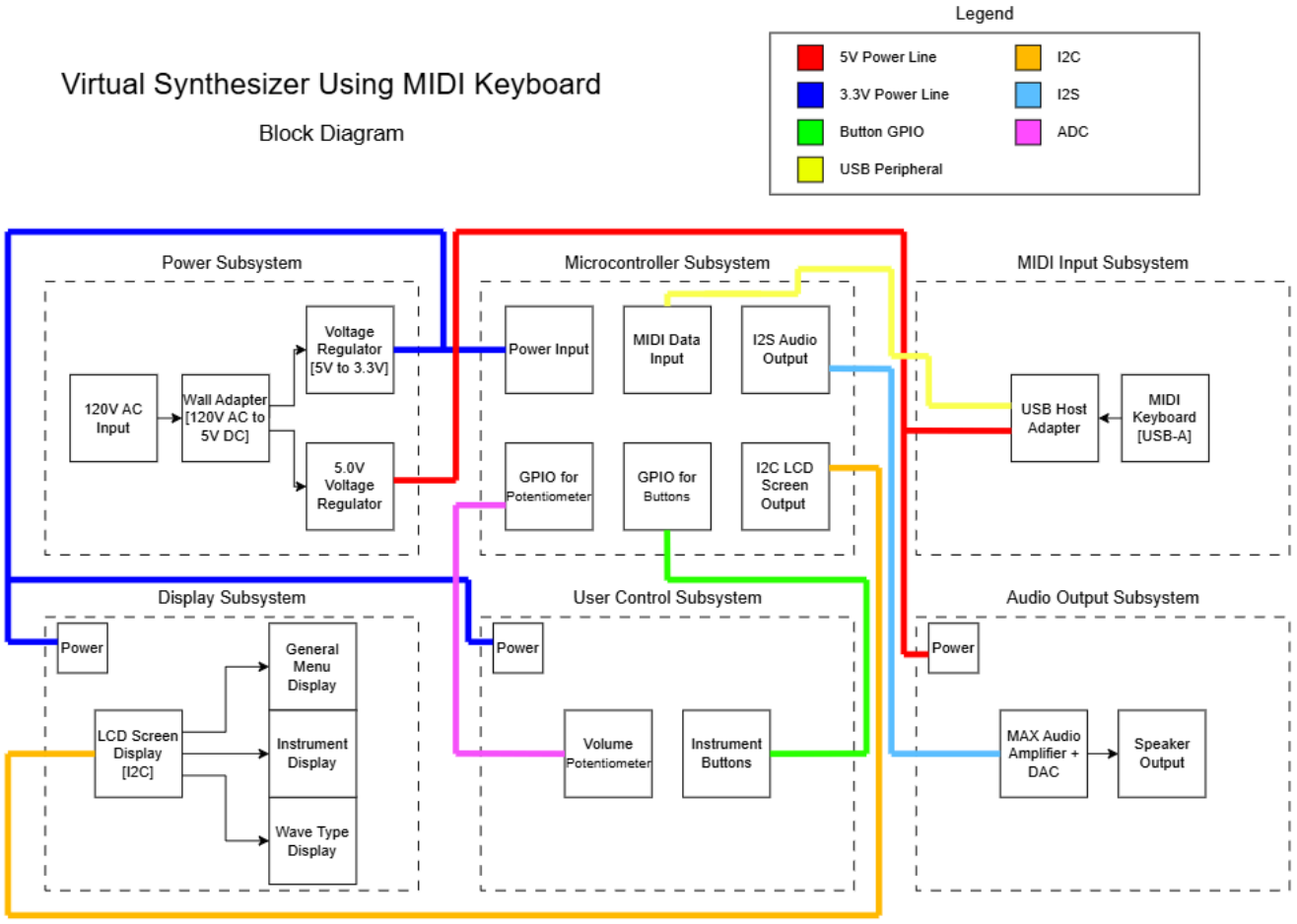
1.4 Potential Enhancements (Reach Goals)

If time allows, we will implement these features to make our virtual synthesizer more developed:

- Wavetable Synthesis – Blending of multiple waveforms to create richer, more complex sounds.
- Live Chord & Scale Feedback – Display real-time visual feedback of chords and scales being played on the screen.
- DAW-Like Features – Enable saving and layering of notes from different instruments for simultaneous playback.
- Note Velocity – Incorporate dynamic volume control based on key press intensity for more expressive playing.

2 Design & Requirements

2.1 Block Diagram



2.2 Subsystem Descriptions/Requirements

Microcontroller – This subsystem handles the signal processing from the MIDI keyboard and user controls and outputs digital I2S signals to the I2S DAC in the audio output subsystem. An ESP32-S3 receives MIDI audio through its USB Host Controller. The analog inputs 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 performs the signal processing and outputs an I2S audio signal through its I2S0 peripheral.

Microcontroller Requirements:

- ESP32-S3 correctly reads MIDI messages from USB peripheral
- ESP32-S3 assigns analog input between 0V and 1.1V to a 12-bit digital value
- ESP32-S3 outputs repeating I2S signal while a single keypress signal is held constant
- Synthesizable waveforms are correctly generated from defined equations
- Sample waveforms are accessible from memory
- Waveforms can be correctly modified by formulas dependent on user controls

Power – The power subsystem draws power from a 120V 60Hz AC wall outlet using a 5V adapter. The 5V supply will be regulated to 3.3V and 5.0V using the 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, and the I2S DAC in the audio output subsystem. The 3.3V line will power the ESP32-S3, the user controls that send signals to the ESP32-S3, and the LCD screen.

Power Requirements:

- 5.0V voltage regulator outputs 4.75-5.25V, consistent with USB 2.0 specifications
- 5.0V voltage regulator is capable of outputting at least 900mA
- 3.3V voltage regulator outputs 3.0-3.6V, consistent with ESP32-S3 maximum ratings

- 3.3V voltage regulator is capable of outputting at least 800mA
- Adapter interfaces with wall outlet at 120V 60Hz AC to provide 4.5-5.5V DC

Audio Output – The Audio Output subsystem is responsible for converting the I2S digital audio signal from the ESP32-S3 into audible sound. The ESP32-S3 microcontroller will output a PCM I2S signal through its I2S0 peripheral. The signal will then be sent to a MAX98357A I2S DAC Amplifier to produce the necessary analog audio output. The amplified signal will drive a 4-Ohm speaker rated between 3W and 5W, ensuring sufficient volume while maintaining a high level of sound clarity with no unnecessary noise. The power for the amplifier will be drawn directly from the 5V power subsystem, depending on the specific requirements of the amplifier.

Speaker Requirements:

- I2S DAC converts the digital signal from the microcontroller into a useable analog signal to be forwarded to the speakers
- Speakers are powered below the maximum output power of 5W to maintain safe operation

User Control – For user control, the synthesizer will use a potentiometer connected to the ESP32 that allows for a continuous volume control range. The ESP32 will read analog input and convert it in software with a 12-bit ADC to a percentage that will scale the output audio. Additionally, multiple buttons will be used to navigate through menus regarding instrument selection and post effects such as distortion. Each button will simply be connected to pins on the ESP-32 that are polling for their activation.

User Control Requirements:

- Volume Potentiometer measurable with a multimeter with tolerance
- Effect Potentiometer
- Left and Right scroll Buttons
- Select button
- Menu button
- All buttons have pull-up resistors and safe paths to ground

Display – The LCD screen will be used to show information about the selected instrument and volume level on the synthesizer. When the user switches instruments, the LCD screen will update to display the instrument name, helping beginners easily understand what sound they are working with. Additionally, when the volume is adjusted, the screen will show the current volume level, providing clear feedback to the user. Since the display uses I2C communication, it will have 2 data pins and 2 power pins connected to the ESP-32. We plan on using the *GeeekPi 1602* to accomplish this.

LCD Screen Requirements:

- Working display for sequence numbers and letters
- Connectors for the 4-pin output from the display

MIDI Input– To connect the MIDI keyboard to the ESP32 microcontroller, we will use a USB female connector that allows the keyboard's USB cable to interface with the microcontroller. The USB connector will split the MIDI signal into two data lines, D+ and D-. The USB VCC and GND pins will be connected to the 5.0V and GND from the power susbsystem. The two data

outputs will then be connected to the corresponding USB-capable pins on the ESP32, enabling it to read and process MIDI data. When in USB host mode, the ESP-32 can read the key presses from the keyboard and relay this data to be processed.

MIDI Keyboard Requirements:

- A female Type-A usb connector splits the MIDI output into two data lines and two power lines
- MIDI keyboard supports at least 10-note polyphony
- MIDI keyboard pitch wheel sends pitch bend MIDI message

2.3 Risk Analysis

The component that poses the highest risk and greatest difficulty to implement into our virtual synthesizer is digital sound processing along with polyphony handling. This crucial piece of the synthesizer is responsible for instrument simulation, sound generation, and polyphony support, making it act as the center of our project. With such an important and highly-relying role, the implementation of this component comes with a high level of difficulty and risk.

2.3.1 Justification of Risk

Real-Time Processing Constraints – The ESP32-S3 must handle multiple real-time operations at once including MIDI keyboard signal processing, waveform synthesis, and polyphony management. Making all these tasks execute together in a smooth manner while ensuring a real-time output adds a high level of difficulty. Microcontroller processing power must also be taken into consideration, increasing this implementation's risk.

Polyphony Complexity – Supporting four or more simultaneous notes requires efficient mixing of waveforms, ultimately increasing the computational load of the component. This feature must be carefully handled to avoid glitches, note dropouts, or delays in sound generation.

Instrument Sound Quality – A key part of our synthesizer is ensuring that the sound quality of each instrument is of high quality. This requires precise synthesis methods that must be implemented properly to allow instruments to sound different from one another or unnatural, which is one of our high-level requirements for the project.

Data Transmission Bottlenecks – Finally, the ESP32-S3 microcontroller must be able to read input from the MIDI Keyboard over USB, process it, and generate the desired audio signal in real time. If the microcontroller fails to keep up with incoming inputs from the user through the keyboard, latency will begin to increase, decreasing the effectiveness of our virtual synthesizer.

2.3.2 Acceptable Tolerances

The wall adapter will provide approximately 5 volts, however, we will use voltage regulators at both 5V and 3.3V to maintain a narrow range of variation when providing power to the rest of our circuit. This will allow us to power the USB Female connector (used to power keyboard) as well as the I2S DAC as both require 5V ± 5%. Additionally, the microcontroller requires 3.3V ± 0.3V to operate so we will use the output of the 3.3V regulator to supply power to the ESP32. Furthermore, the LCD screen requires 3.3V ± 5% so we will also power it with the 3.3V regulator output.

2.3.3 Relation to High-Level Requirements

This risk is directly tied to the *Instrument Simulation and Sound Generation* requirement, as poor polyphony handling and/or digital signal processing failures would result in distorted, laggy, or inaccurate instrument sounds. It also significantly impacts another one of our high-level requirements- *Polyphony and Chord Support*- as failing to properly handle multiple simultaneous notes would break the synthesizer's ability to play chords cleanly and smoothly. Lastly, *Octave Range and Note Accuracy* are also affected by this risk, since improper processing could lead to frequency drift, poor note mapping, or unwanted distortions when switching between octaves.

2.3.4 Risk Reduction Strategies

To reduce the likelihood of these risks occurring and affecting performance, we will implement:

- Optimized DSP Algorithms – Using efficient synthesis methods will help us minimize processing demand.
- Priority-Based Processing – Ensuring time-sensitive MIDI input is processed first to avoid latency will help handle the flow of multiple inputs at once.
- Performance Benchmarking – Continuously testing latency and polyphony under different workloads to ensure real-time performance is maintained will be a crucial part of developing our virtual synthesizer.

3 Ethics & Safety

3.1 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.

3.2 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.

4 Citations and References

[1] IEEE - IEEE Code of Ethics, https://www.ieee.org/about/corporate/governance/p7-8.html (accessed Feb. 13, 2025).

[2] “MAX98357A/MAX98357B Datasheet.” Maxim Integrated

[3] ESP-IDF Programming Guide, https://docs.espressif.com/projects/esp-idf/en/v5.4/esp32s3/index.html (accessed Feb. 13, 2025).

[4] *ESP32-S3 Series Datasheet*. Espressif.