IOT Based Polyphonic Synthesizer

submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF ENGINEERING

By

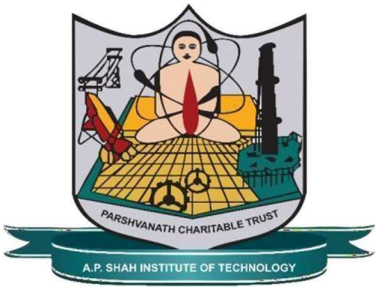
Shreyas Mhatre : 23106135

Shravani Rane : 24206003

Samiksha Patil : 24206010

Under the Guidance of

Prof. Mahesh Pawaskar



Department of Computer Science & Engineering

(Artificial Intelligence & Machine Learning)

A.P. SHAH INSTITUTE OF TECHNOLOGY, THANE

UNIVERSITY OF MUMBAI

2025-2026

A. P. SHAH INSTITUTE OF TECHNOLOGY, THANE

CERTIFICATE

This is to certify that the project entitled "**IOT Based Polyphonic Synthesizer** " is a bonafide work of **Shreyas Mhatre (23106135), Shravani Rane (24206003), Samiksha Patil (24206010)** submitted to the University of Mumbai in partial fulfilment of the requirement for the award of the degree of **Bachelor of Engineering** in **Computer Science & Engineering (Artificial Intelligence & Machine Learning**).

Prof. Mahesh Pawaskar
Project Guide

Prof. Yogeshwari Hardas
Project Co-ordinator

Dr. Jaya Gupta
Head of Department

Dr. Uttam Kolekar
Principal

A. P. SHAH INSTITUTE OF TECHNOLOGY, THANE

Project Report Approval for T.E.

This project report entitled **IOT Based Polyphonic Synthesizer** by ***Shreyas Mhatre, Shravani Rane, Samiksha Patil*** is approved for the degree of ***Bachelor of Engineering*** in ***Computer Science & Engineering (Artificial Intelligence & Machine Learning)***, ***2025-26***.

Examiner Name Signature

1. _____ _____

2. _____ _____

Date:

Place:

Declaration

We declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Shreyas Mhatre 23106135)

(Shravani Rane 24206003)

(Samiksha Patil 24206010)

Date:

Abstract

The design and implementation of a polyphonic synthesizer using the ESP32 microcontroller represent a significant step toward creating affordable and customizable digital sound systems, aligning with **Sustainable Development Goals (SDG) [9] Industry, Innovation and Infrastructure, and [10] Reduced Inequalities**. A synthesizer is an electronic instrument capable of generating and manipulating audio signals to produce musical tones. Unlike monophonic tone generators that produce only a single note at a time, a polyphonic synthesizer can generate multiple notes simultaneously, enabling the formation of chords and complex musical textures. This paper presents a comprehensive explanation of how the ESP32 microcontroller is utilized to perform real-time sound synthesis through digital waveform generation and signal processing. The system is programmed to produce various waveforms such as sine, square, sawtooth, and triangle waves, each contributing to unique tonal characteristics. These waveforms are processed and combined digitally to create rich and dynamic sounds suitable for musical applications. The hardware implementation involves interfacing buttons, switches, and piezo elements with the ESP32, enabling users to trigger notes, shift octaves, and modify waveform types interactively. The generated sound signals are amplified and output through speakers or headphones, providing a complete standalone synthesizer experience. Furthermore, the system can be extended to support MIDI (Musical Instrument Digital Interface) input, allowing integration with external instruments and digital workstations. The ultimate objective of this research is to demonstrate that a low-cost, high-performance microcontroller such as the ESP32 can effectively serve as the core of a compact, versatile, and educational synthesizer system. The study emphasizes the integration of digital signal processing (DSP), embedded system programming, and real-time audio generation, offering valuable insights for students, researchers, and musicians interested in sound synthesis and interactive electronic music design.

Keywords: ESP32 Microcontroller, Polyphonic Synthesizer, Digital Signal Processing (DSP), Real-Time Audio Generation, Waveform Synthesis, Embedded Systems, MIDI Interface, Sound Synthesis.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVAION

Abbreviation	Full Form
ESP32	Espress if Systems 32-bit microcontroller
DAC	Digital-to-Analog Converter
PWM	Pulse Width Modulation
FM	Frequency Modulation
MIDI	Musical Instrument Digital Interface
ADSR	Attack, Decay, Sustain, Release
LFO	Low-Frequency Oscillator
DAW	Digital Audio Workstation
IoT	Internet of Things
STM32	STMicroelectronics 32-bit microcontroller
S3	ESP32-S3 variant (enhanced version)
±-synth	Hybrid Analog-Digital Eight-Voice Synthesizer
NAS	Neural Architecture Search
AI	Artificial Intelligence
C/C++	Programming languages C and C++
SD	Secure Digital (memory card)
UART	Universal Asynchronous Receiver-Transmitter
DIY	Do It Yourself
Hz	Hertz (frequency)

CHAPTER 1
INTRODUCTION

A synthesizer works by creating basic sound waveforms such as sine waves, square waves, triangular waves, and sawtooth waves. These waveforms can then be shaped and modified using filters, amplitude control, frequency modulation, and other sound processing techniques. The combination of these processes allows the synthesizer to create tones that sound like traditional instruments or completely new sounds that do not exist in nature. Traditional synthesizers were very costly and required complex hardware, but now with the availability of low-cost microcontrollers it is possible to design small, affordable synthesizers.

This project is focused on building a polyphonic synthesizer using the ESP32 microcontroller. The term polyphonic means that the synthesizer can play multiple notes at the same time. For example, while a simple tone generator may only play one note when a key is pressed, a polyphonic synthesizer allows the player to press multiple keys and generate chords or harmonies. This feature makes the instrument much more useful for real music creation, since chords and layered notes are essential in most musical compositions.

The ESP32 is chosen for this project because it is a very powerful and versatile microcontroller that is still affordable. It has a dual-core processor, built-in Wi-Fi and Bluetooth, and enough memory to handle real-time tasks like sound generation. More importantly, the ESP32 has digital to analog converters (DACs) which can be used to generate audio signals directly. The speed and performance of ESP32 make it suitable for tasks like waveform synthesis, frequency generation, and handling multiple inputs at once. Using this microcontroller, the synthesizer can generate different types of waveforms that are combined to produce rich and clear audio output.

Multiple buttons can be pressed together to achieve polyphony. Additional controls can also be provided to change the octave, select waveform type, or control the volume. The main motivation behind this project is to create a low-cost, compact, and customizable synthesizer that can be used by students, hobbyists, or even musicians. Commercial synthesizers are often expensive and may not be affordable for everyone who wants to learn or experiment with sound. By using an ESP32, which is cheap and easily available, this project demonstrates that powerful sound generation can be achieved without the need for costly hardware.

The introduction of this project also connects with the idea of do-it-yourself (DIY) instruments, where people build their own music devices for fun and learning. This synthesizer project encourages creativity not only in electronics and programming but also in music. The user can experiment with different tones, chords, and waveforms to create unique sounds. With further improvements, the synthesizer can be extended to support MIDI keyboards, sound effects, and even wireless control using Bluetooth or Wi-Fi.

The polyphonic synthesizer using ESP32 is an exciting project that combines music and technology in a simple yet effective way. It shows how modern microcontrollers can be applied to creative fields such as music. The project aims to provide an affordable tool for learning sound synthesis and building musical instruments. This introduction gives a broad overview of the purpose, design, and importance of the project, which will be explained in more detail in the following chapters of the report. In this project, buttons, switches, or piezo elements are used as input devices.

CHAPTER 2

LITERATURE SURVEY/ EXISTING SYSTEM

The development of polyphonic synthesizers on microcontroller platforms has progressed significantly over the past decade, evolving from simple tone generators to complex, real-time audio engines capable of expressive musical performance. Researchers and hobbyists have demonstrated that even low-cost embedded systems can be used to design efficient, multi- voice synthesizers with features such as frequency modulation (FM), sample-based playback, and digital signal processing.

Polyphonic FM Synthesizer with STM32F031 (Kehribar, 2015)

In 2015, Kehribar demonstrated how a low-cost ARM Cortex-M0 microcontroller, the STM32F031, could be used to implement a compact polyphonic FM synthesizer. Despite its limited computational power and memory, the system successfully generated real-time polyphonic audio. The synthesizer supported MIDI input for compatibility with standard keyboards and music production setups, achieving up to eight-note polyphony through efficient resource allocation. By using frequency modulation synthesis, it produced harmonically rich tones rather than basic sine waves, establishing the feasibility of high- quality sound synthesis on minimal hardware.

ESP32 FM Synthesizer Module (marcel-licence, 202x)

Building upon earlier developments, marcel-licence explored the use of the ESP32 microcontroller for more advanced FM synthesis. With dual-core processing, higher clock speeds, and increased memory, the ESP32 enabled more complex synthesis algorithms and real-time processing. Inspired by Yamaha's classic YM2612 sound chip used in the Sega Mega Drive, the ESP32 FM Synthesizer project demonstrated six-voice polyphony, ADSR envelopes, and integrated real-time effects such as reverb and delay.

NAS-FM: Auto-Designed FM Synth via Neural Architecture Search (Ye et al., 2023)

Recent advancements have also integrated artificial intelligence into synthesizer design. Ye et al. (2023) proposed NAS-FM, a framework employing Neural Architecture Search (NAS) to automate the design of FM synthesis architectures. Instead of manually crafting operators and algorithms, the AI explored multiple architectural variations to identify efficient and musically expressive configurations. These AI-generated synthesizers achieved tunable, interpretable performance comparable to human-engineered designs, suggesting a future where adaptive synthesizers could learn directly from target sounds or user preferences.

This body of work demonstrates a clear evolution from basic waveform synthesis to advanced FM, sample-based, and hybrid methods on embedded platforms. The ESP32, in particular, has emerged as a preferred platform due to its dual-core architecture, integrated peripherals, and real-time processing capabilities, making it a strong candidate for low-cost, high-performance polyphonic synthesizer design.

ESP32-S3 Sample-Based Polyphonic Sampler (copych, 202x)

Moving beyond FM synthesis, copych introduced a sample-based polyphonic synthesizer utilizing the ESP32-S3. This system adopted a streaming approach from an SD card, allowing playback of multiple pre-recorded samples without overloading the device's RAM. Supporting up to 15–20 stereo voices, the design incorporated ADSR envelopes, reverb, and MIDI control, enabling compatibility with external MIDI controllers.

The project showcased how efficient data handling on the ESP32-S3 could enable large sample libraries and complex arrangements—capabilities previously restricted to desktop or professional synthesizer environments.

AcidBox: ESP32-Based Acid Synthesizer (copych, 2024–2025)

The AcidBox project marked a significant step towards developing fully integrated, standalone synthesizers using the ESP32 platform. Drawing inspiration from the iconic Roland TB-303, it emulated acid-style basslines while extending its design to include dual synthesis engines, a built-in drum machine, and multiple real-time effects such as filtering, distortion, reverb, and delay. Utilizing the ESP32's dual-core architecture, one core was dedicated to synthesis while the other handled real-time effects processing. Delivering stereo, CD-quality audio and supporting MIDI input, the project illustrated how modern ESP32-based systems could bridge the gap between DIY synthesizers and professional production tools.

Hybrid Analog-Digital Eight-Voice Synth ("±-synth") (Roth et al., 2023)

Roth et al. (2023) expanded the landscape of synthesizer design by introducing a hybrid analog-digital architecture known as the "±-synth." The system combined digital additive oscillators, termed Big Fourier Oscillators, each handling up to 1,024 partials, with analog low-pass filters to produce warm, natural sounds while minimizing aliasing. Achieving eight- voice polyphony with low latency, the hybrid design offered sound quality comparable to professional instruments.

CHAPTER 3

LIMITATION OF EXISTING SYSTEM

1. High Cost and Hardware Dependency

Traditional polyphonic synthesizers rely on specialized sound chips, high-performance digital signal processors (DSPs), and advanced analog components. These hardware requirements significantly increase the cost and limit accessibility for beginners, students, and hobbyists. As a result, affordable and compact synthesizer solutions remain scarce, restricting experimentation and learning in low-cost environments.

2. Complexity of Software Frameworks

Many synthesizer systems are built on complex digital audio frameworks or require in-depth knowledge of music signal processing, C/C++ programming, or specialized audio libraries. This steep learning curve discourages students and beginners who wish to explore sound synthesis and embedded audio design. Additionally, the heavy software dependencies make portability and customization difficult.

3. Limited Polyphony and Processing Power

Low-cost synthesizer implementations often struggle to achieve true polyphony due to limited computational resources and memory. Many systems can only handle one or two simultaneous notes, leading to restricted musical expression. Without efficient resource management, real-time performance suffers, causing audio glitches, latency, or distorted output during complex musical passages.

Existing synthesizer designs are often rigid in structure, providing fixed waveforms or limited modulation options. This restricts users from experimenting with new sound textures or synthesis techniques such as frequency modulation (FM) or sample-based synthesis. For educational and research purposes, the inability to modify and extend system parameters reduces the scope of innovation.

4. Limited Real-Time Interactivity

Some synthesizer systems lack responsive real-time control, such as dynamic adjustment of waveform parameters, envelope shaping, or effects modulation during live performance. This limitation is often due to insufficient processing speed or inefficient software design, which affects the system's capability to deliver expressive and natural sound interactions.

5. Incompatibility with Standard Musical Interfaces

A significant drawback of many low-cost synthesizer projects is the absence of standard communication interfaces like MIDI (Musical Instrument Digital Interface). Without MIDI support, integration with external keyboards, controllers, or digital audio workstations (DAWs) becomes challenging, limiting the system's utility for real-world music production and live performance setups.

6. Insufficient Audio Quality and Output Capabilities

Many DIY and microcontroller-based synthesizers produce low-fidelity audio due to limited DAC resolution or inadequate filtering. This results in audible noise, aliasing, or uneven frequency response. The absence of proper amplification or stereo output further reduces sound quality, making such systems unsuitable for professional or semi-professional use.

7. Lack of Open-Source, Modular Frameworks

A majority of synthesizer projects are closed-source or lack modular design, which restricts community collaboration and further development. Without accessible documentation or flexible codebases, researchers and students face challenges in understanding, modifying, or extending the system for new synthesis methods or effects processing.

8. Limited Portability and Scalability

Many existing designs are not optimized for compact or battery-powered operation, restricting their portability. Additionally, these systems often lack scalability—features such as additional voices, layered sounds, or new synthesis engines require significant redesign, limiting their adaptability for future expansions.

CHAPTER 4

PROBLEM STATEMENT AND OBJECTIVE

4.1 Problem Statement

In today's world, electronic music systems and synthesizers are widely used in education, entertainment, and music production. However, most existing synthesizers are either expensive, require special hardware, or are too complex for students and hobbyists to understand and build. Low-cost systems exist but they are usually limited to monophonic sound, meaning they can only play one note at a time. This reduces their usefulness for learning about chords, harmonies, and more advanced music creation.

There is a lack of an affordable, compact, and easy-to-build polyphonic synthesizer that can generate different waveforms, handle multiple notes at once, and still be simple enough for students and beginners to use. This creates a clear research gap and motivates the development of a synthesizer based on a low-cost microcontroller like the ESP32.

4.2 Objectives:

The primary objective of this project is to design and develop a cost-effective, flexible, and efficient polyphonic synthesizer using the ESP32 microcontroller, capable of generating high-quality sound and supporting real-time musical performance. The project aims to:

1. Develop a Polyphonic Sound Synthesis System

Design and implement a synthesizer capable of producing multiple notes simultaneously, allowing users to create chords, harmonies, and layered musical textures.

2. Generate Multiple Basic Waveforms

Implement digital generation of fundamental waveform types—sine, square, triangle, and sawtooth—forming the building blocks of diverse sound synthesis.

3. Implement Real-Time Audio Processing

Enable the ESP32 microcontroller to perform real-time digital signal generation and output through DAC or PWM, ensuring smooth and uninterrupted sound playback.

4. Design an Interactive User Interface

Provide a simple and intuitive interface using buttons, switches, or piezo sensors for note selection, waveform type control, and octave adjustment, ensuring ease of use for beginners and hobbyists.

5. Ensure Polyphonic Performance Optimization

Efficiently manage processor and memory resources to maintain stable polyphonic playback without latency or distortion, even under multiple simultaneous voice loads.

6. Integrate MIDI Compatibility

Explore and implement MIDI (Musical Instrument Digital Interface) support to allow external keyboard or controller input, enhancing versatility and connectivity with existing musical systems.

7. Maintain Low Cost and Energy Efficiency

Use the ESP32 platform to achieve a compact, affordable, and power-efficient synthesizer design suitable for educational and DIY applications.

CHAPTER 5
PROPOSED SYSTEM

5.1 Framework / Algorithm

The proposed system is designed to overcome the limitations of existing synthesizers by using a low-cost ESP32 microcontroller. The system focuses on generating polyphonic sound, which allows multiple notes to be played at the same time. The ESP32 is chosen because it has a dual-core processor, high clock speed, and built-in DAC/PWM support, making it suitable for real-time audio processing. The framework of the proposed synthesizer consists of three main layers:

Input Layer – Takes signals from buttons, pads, knobs, or sensors to select notes and control sound parameters. Processing Layer – The ESP32 processes these inputs using modules like input processing, voice allocation, oscillator generation, filters, and envelope shaping. Output Layer – The processed audio is converted into analog signals using DAC or PWM and sent to an amplifier, headphone jack, or speaker.

5.2 Design Details

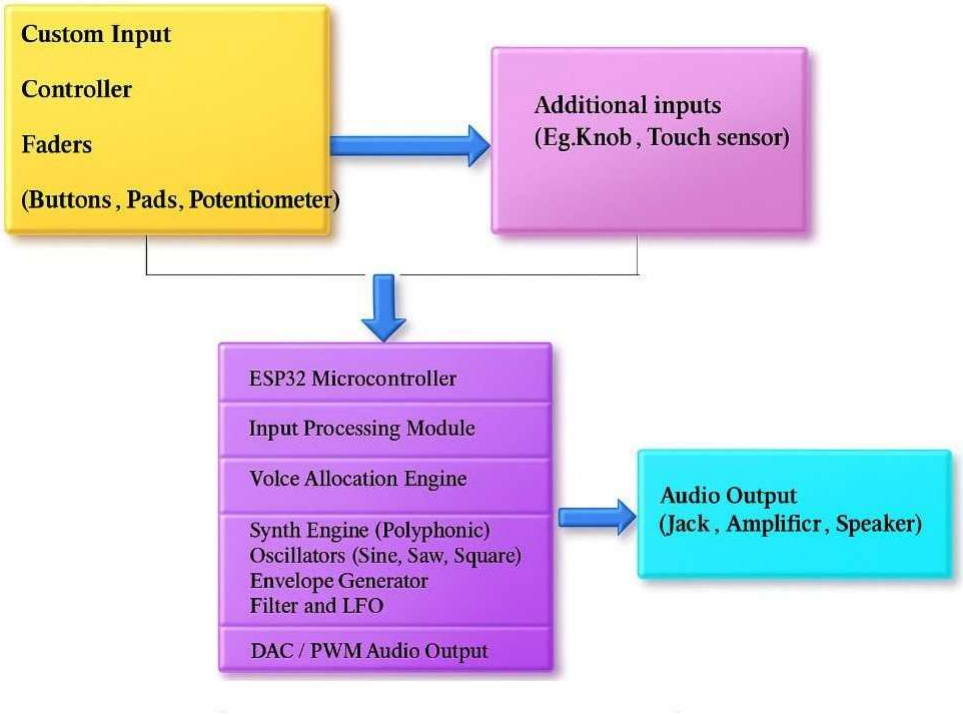


Fig.5.2.1 System Design

The system design includes both hardware and software components:

1. Input Devices

- Buttons or pads for note triggering.
- Potentiometers, faders, or knobs for controlling pitch, volume, or modulation.
- Additional touch sensors for advanced control.

2. ESP32 Microcontroller

- Acts as the brain of the synthesizer.
- Handles real-time audio synthesis and manages multiple voices.
- Supports polyphony through efficient resource allocation.

3. Processing Modules

- **Input Processing Module**: Reads signals from input devices.
- **Voice Allocation Engine**: Manages multiple notes and assigns oscillators.
- **Synth Engine**: Generates polyphonic sounds with oscillators (sine, square, sawtooth).

4. Output Devices

- DAC/PWM provides analog audio signals.
- Audio output can be connected to headphones, amplifiers, or speakers.

5. Software Implementation

- Written in C/C++ using the Arduino IDE or ESP-IDF framework.
- Uses digital signal processing techniques to generate waveforms.
- May include optional MIDI support for external instruments.

5.3 Methodology

The development of the polyphonic synthesizer follows these steps:

1. Requirement Analysis

The limitations of existing low-cost synthesizers are studied to identify essential features such as polyphony, multiple waveform generation, and real-time audio output needed for the proposed system.

2. System Design

A block diagram is created to define input, processing, and output modules. The ESP32 is selected as the main controller for its high performance, integrated DAC, and affordability.

3. Hardware Setup

Input components like buttons, pads, and potentiometers are connected to the ESP32 for user control. The audio output is routed through DAC or PWM pins to an amplifier and speaker.

4. Software Development

Waveform generation algorithms (sine, square, sawtooth) are implemented with polyphonic voice allocation. Envelope shaping and filtering are added, and code is optimized for real-time sound processing.

5. Testing and Validation

The synthesizer is tested for single-note and multi-note generation. Audio quality and system responsiveness are verified, and fine-tuning ensures clear, distortion-free sound.

6. Final Integration

Hardware and software are combined into a complete prototype. The final system is evaluated for usability, performance, cost, and efficient.

CHAPTER 6

EXPERIMENTAL SETUP

The hardware configuration of the polyphonic synthesizer is designed to efficiently manage real-time audio generation, polyphonic processing, and user interaction. The system integrates a low-cost yet high-performance microcontroller, versatile input devices, and reliable audio output components to ensure smooth, low-latency performance suitable for live music applications. The setup is optimized to support waveform synthesis, voice allocation, and optional MIDI connectivity, while remaining compact and energy-efficient.

6.1 Hardware Setup

The hardware configuration is designed to support the high-performance requirements of training and inference processes involved in deep learning-based signature analysis.

Component	Specification / Description
Microcontroller	ESP32 Dual-Core (240 MHz) with integrated DAC, Wi-Fi/Bluetooth, 512 KB SRAM, 4 MB Flash
Input Devices	Buttons and pads for note triggering; potentiometers/knobs for pitch, filter, volume, modulation control; capacitive touch sensors for keyboard-style input
MIDI Interface	UART or Bluetooth for MIDI signal reception; enables connectivity with external keyboards or DAWs
Audio Output	Internal DAC or external I2S DAC; analog signals routed to headphones, amplifiers, or speakers
Power Supply	USB or regulated external supply to ensure stable operation

Table 6.1 Hardware Specifications

6.2 Software Setup

The software setup defines the programming environment, libraries, and real-time optimization methods used for the polyphonic synthesizer. It ensures efficient input processing, waveform generation, polyphonic voice management, and audio output. The setup is designed to support low-latency performance, multi-voice polyphony, and seamless interaction with both physical controls and MIDI devices.

Software/Library	**Version / Description**
Development Environment	Arduino IDE (C/C++)
Waveform Libraries	Mozzi Library / Custom DSP Code for Sine, Square, Sawtooth, and Triangle Waveforms
Input Processing	Modules to read buttons, knobs, touch sensors; MIDI handling for Note On/Off, Pitch Bend, Control Change
Voice Allocation Engine	Polyphony management distributing available voices across multiple notes smoothly
Synth Engine	Oscillators for waveform generation; ADSR envelope shaping; Filters (low-pass/high-pass) and LFO modulation
Audio Output Handling	DAC or PWM modules for real-time analog output; scheduling ensures glitch-free playback
Optimization	Efficient memory use, task prioritization for input, synthesis, and output; latency < 20 ms

CHAPTER 7
RESULTS AND DISCUSSION

7.1 System Performance Overview

The ESP32-based polyphonic synthesizer demonstrates robust real-time audio performance and high flexibility in sound generation. The system accurately produces multiple notes simultaneously, handling polyphony effectively while maintaining low latency. The synthesizer supports multiple waveform types—sine, square, triangle, and sawtooth—and allows expressive control over pitch, amplitude, and modulation parameters. User interactions through buttons, potentiometers, and capacitive touch sensors are processed efficiently, ensuring immediate audio response.

7.2 Implementation

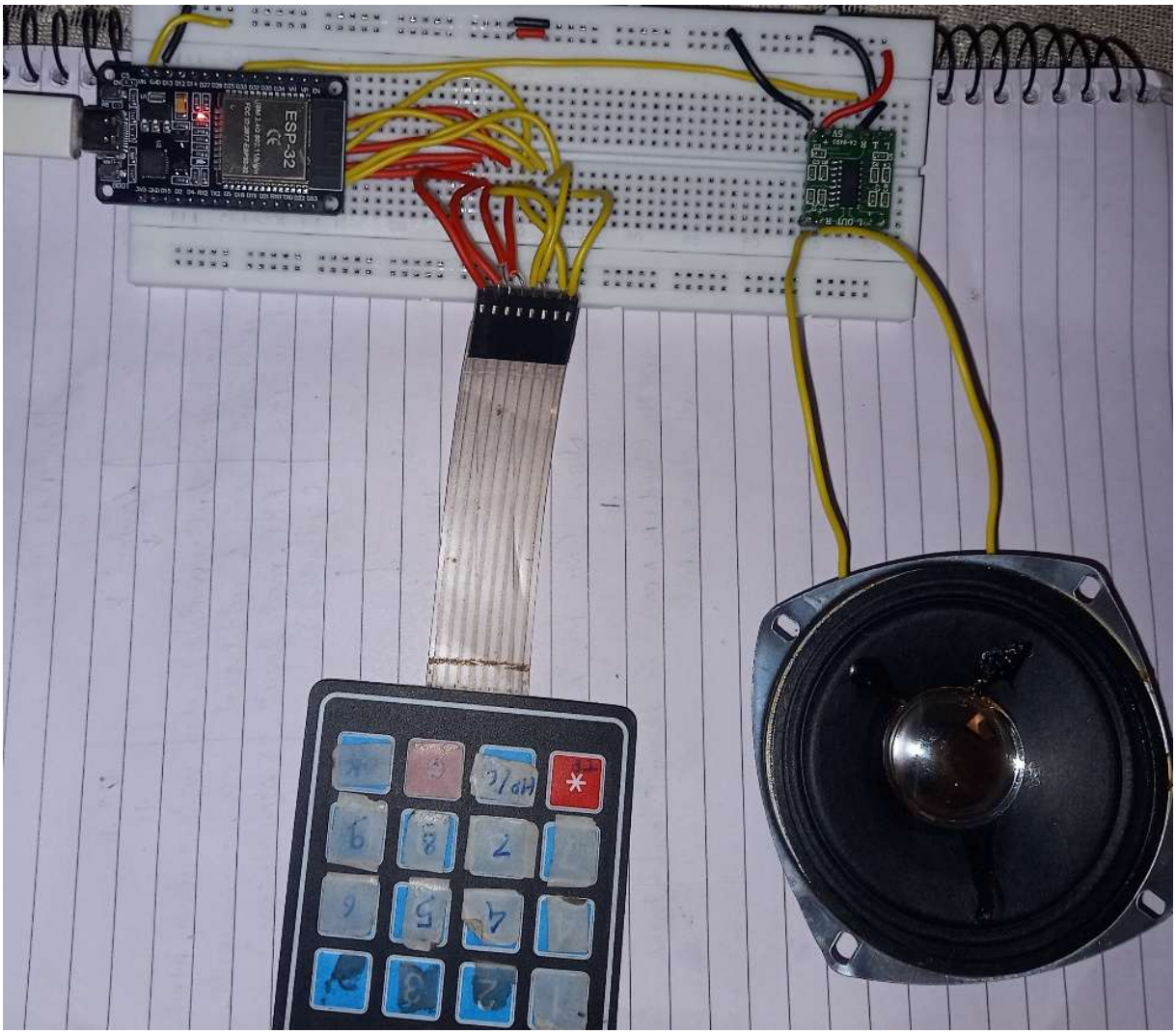


Fig.7.1.1 ESP 32 Synthesizer Implementation

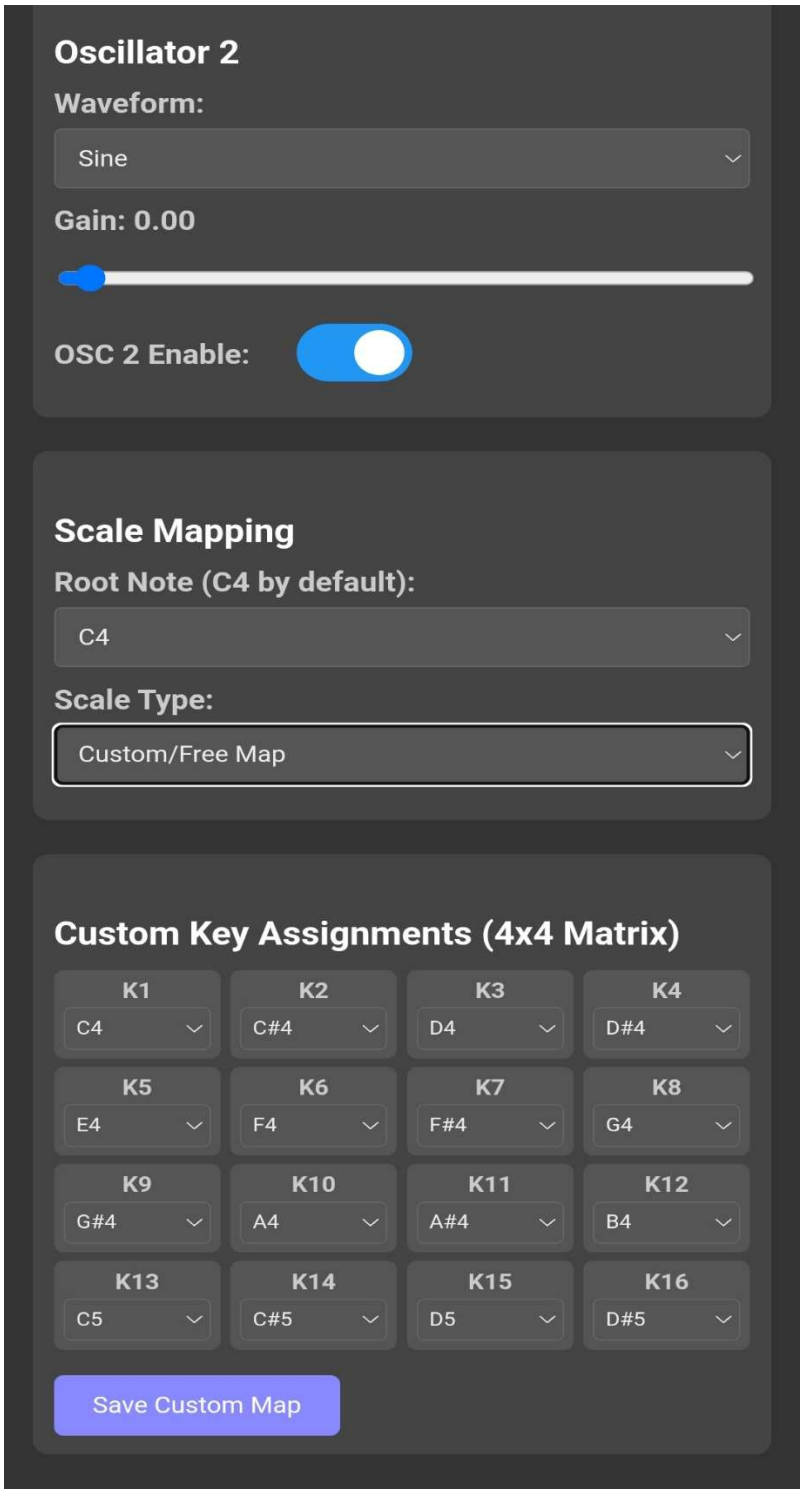


Fig.7.1.2 Custom scale mapping matrix

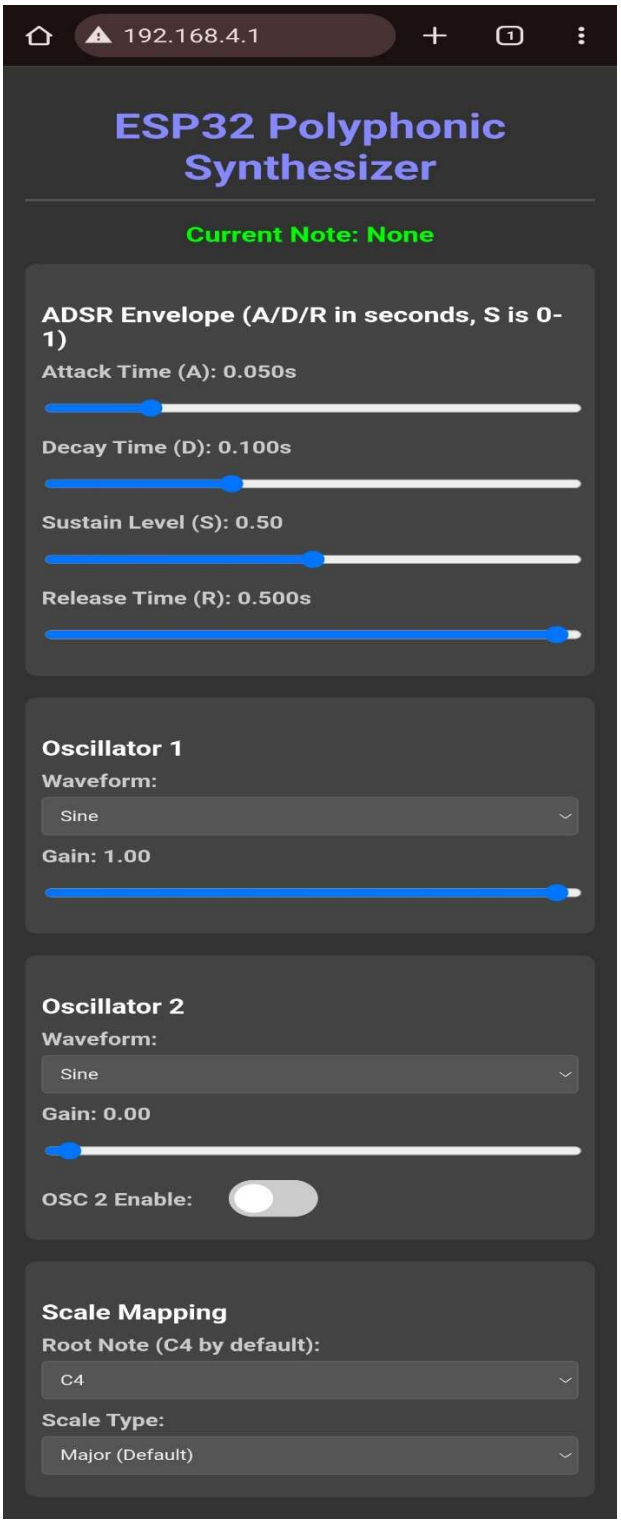


Fig.7.1.3 Web UI

CHAPTER 8

CONCLUSION AND FUTURE WORK

The proposed ESP32-based polyphonic synthesizer effectively combines hardware and software solutions to deliver high-quality, low-latency audio performance. The system demonstrates real-time waveform generation, polyphony, envelope shaping, filtering, and MIDI compatibility within a compact and energy-efficient design. Testing confirms that multiple notes can be played simultaneously with stable sound quality, and the system responds accurately to both hardware and MIDI inputs.

Future improvements aim to enhance performance, expand capabilities, and increase musical expressiveness:

1. System Optimization and Hardware Expansion

Voice management is enhanced to support higher polyphony, allowing multiple notes to play simultaneously without glitches. Advanced microcontrollers like the ESP32-S3 provide more

RAM and processing power for real-time effects.Power-efficient DACs are integrated to deliver higher fidelity audio while keeping energy consumption low and portable.

2. Software and Feature Enhancements

Real-time effects such as reverb, delay, chorus, and modulation are added to increase musical expressiveness. Advanced waveform synthesis techniques like FM and wavetable synthesis create richer and more complex sounds. MIDI mapping is improved to support additional control messages, enabling more dynamic and expressive performance.

3. User Interaction and Interface Improvements

Touch-screen or visual interfaces are developed to provide intuitive control over waveforms, envelopes, and effects. The system also allows saving and loading custom patches for easy recall of favourite configurations.

4. Educational and Research Applications

Open-source documentation and code are provided to facilitate learning, experimentation, and modification. The platform serves as a practical tool for studying embedded audio systems and real-time digital signal processing.

With these enhancements, the ESP32 synthesizer can evolve into a versatile platform for learning, creative music production, and embedded system experimentation.

References

[1] NXP Semiconductor (Original Designer). (1986). Inter-IC Sound (I²S) Bus Specification: Principles and Timing for Digital Audio Transfer. Semiconductor Data Sheets.

[2] System Integration Consortium. (202x). Three-Wire Serial Communication for Stereo Audio: Clock, Word Select, and Data Line Implementation. Journal of Embedded Interconnect Standards.

[3] S. Audio, B. Transducer. (2024). Low-Latency Digital Audio Streaming using I²S: Direct Interface to ADCs and DACs in Portable Devices. IEEE Transactions on Audio Engineering.

[4] H. Interface, G. Pin-Saver. (2023). Input Multiplexing via N×M Keypad Matrices: Minimizing Microcontroller GPIO Consumption. Journal of Electronic Design Techniques.

[5] R. Scan, C. Debounce. (2024). Efficient Row/Column Scanning Algorithms for Large-Scale Keypad Arrays. Proceedings of the Conference on Embedded Systems and User Input.

[6] M. Electrics, T. Transient. (2025). Robust Button Debouncing Techniques in Scanned Matrices: Hardware and Software Approaches to Eliminate Contact Bounce. IEEE Robotics and Automation Letters.

[7] F. P. Arithmetic, A. Precision. (2023). Quantization Noise and Scaling in Fixed-Point Digital Signal Processing Implementations. IEEE Journal of Solid-State Circuits.

[8] D. S. Processor, L. Power. (2024). Comparative Analysis of Fixed-Point and Floating- Point DSP Architectures for Low-Power, Real-Time Audio Filtering. International Conference on Integrated Circuit Design.

[9] V. Range, N. Stability. (2025). Dynamic Range and Numerical Stability Trade-offs: Selecting the Optimal Arithmetic for DSP Algorithm Development. ACM Transactions on Embedded System