# IOT BASED PLOYPHONIC SYNTHESIZER

Submitted in partial fulfilment of the requirements 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 CSE (AI&ML)**

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 “Polyphonic Synthesizer Using ESP32” is a bonafide work of **“Shreyas Mhatre(23106135),Samiksha Patil(24206010), Shravani Rane(24206003)”** 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 (AI&ML**)

(Prof.Mahesh Pawaskar)
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

This project report entitled ***"Iot based polyphonic synthesizer"*** by ***"Shravani Rane, Shreyas Mhatre, Samiksha Patil"*** is approved for the degree of ***Bachelor of Engineering*** in **Computer Science & Engineering (AI&ML)**, ***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

This project presents the design and implementation of a polyphonic synthesizer using the ESP32 microcontroller. A synthesizer is an electronic device that generates and manipulates sounds. Unlike simple tone generators that can only play one note at a time, a polyphonic synthesizer can produce multiple notes simultaneously, allowing the creation of chords and rich musical patterns.

The ESP32 is chosen because it is low-cost, powerful, and versatile, with built-in Wi-Fi, Bluetooth, and dual-core processing**.** It also has sufficient speed and memory to handle real-time audio processing. In this project, the ESP32 is programmed to generate digital sound waves of different types such as sine, square, sawtooth, and triangle waves. These waveforms are combined to produce different tones and musical effects.

For user interaction, buttons, switches, and piezo elements are connected to the ESP32. These components allow the player to trigger notes, change octaves, and select waveform types. The audio output is then amplified and connected to a speaker or headphones for listening. The synthesizer can also be extended to support MIDI input, making it compatible with external musical instruments and keyboards.

The main aim of this project is to demonstrate that **a** low-cost microcontroller like ESP32 can be used to build a compact, affordable, and customizable synthesizer. This system is useful for students, hobbyists, and musicians who want to explore sound synthesis, experiment with music, or build DIY instruments. The project highlights the importance of digital signal processing, embedded systems, and real-time sound generation in creating practical applications.

The Polyphonic Synthesizer using ESP32 combines music and technology, offering a simple yet powerful tool for learning and creativity.

## CONTENTS

## LIST OF FIGURES

# Chapter 1
# Introduction

# 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. In this project, buttons, switches, or piezo elements are used as input devices. Each button corresponds to a musical note, and when pressed, the ESP32 generates the required frequency.

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. At the same time, it provides a good learning experience in the fields of embedded systems, digital signal processing, and real-time system design.

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.

# Chapter 2

# Literature Survey/ Existing system

# 2. LITERATURE SURVEY

## Polyphonic FM Synthesizer with STM32F031 (Kehribar, 2015)

In 2015, Kehribar demonstrated how even a very low-cost ARM Cortex-M0 microcontroller, the STM32F031, could be used to implement a polyphonic FM synthesizer. This chip has very limited memory and processing power compared to modern controllers, yet the author successfully created an instrument capable of producing real-time music

The synthesizer supports MIDI input, which makes it compatible with existing keyboards and music production systems. The project achieves up to eight-note polyphony, which is impressive given the hardware limitations. By using frequency modulation synthesis, it produces complex tones rather than just simple sine waves.

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

Building on earlier work, hobbyists and researchers have turned to the ESP32 microcontroller because it offers dual-core processing, faster clock speeds, and more memory. The ESP32 FM synthesizer project by marcel-licence demonstrates how this chip can handle advanced synthesis tasks. Inspired by the YM2612 chip used in Sega Mega Drive consoles, this project shows how digital sound chips from the 1980s can be recreated with today's low-cost microcontrollers.

The synthesizer supports six-voice polyphony, ADSR envelopes, and real-time audio effects like reverb and delay. Velocity sensitivity allows for more expressive playing, where the strength of a key press affects the volume or tone of the note. Channel-specific tuning also enables more complex musical textures.

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

While FM synthesis focuses on generating sound mathematically, another approach is sample-based synthesis, where pre-recorded audio is played back at different pitches. The ESP32-S3 sampler project takes this approach and adds the ability to play multiple samples at the same time. By streaming directly from an SD card, it avoids filling up the microcontroller's RAM with large audio files. This makes it possible to use multi-gigabyte sample libraries, something that was previously only possible on computers or professional synthesizers.

The system supports 15–20 stereo voices, ADSR envelopes, reverb, and MIDI control. This allows musicians to use it with external controllers and play complex arrangements. The design shows how careful resource management on the ESP32-S3 can unlock features usually expected from expensive samplers.

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

The AcidBox project represents a step towards building a complete standalone instrument using ESP32. Inspired by the legendary Roland TB-303, it emulates acid-style basslines, but also extends the design with two emulation engines, a drum machine, and multiple effects such as filtering, distortion, reverb, and delay.

This project makes full use of ESP32's dual cores, dedicating one to sound synthesis and the other to real-time effects. It delivers stereo, CD-quality audio and integrates MIDI input for external control. Unlike earlier projects that focused on demonstrating one synthesis method, AcidBox shows how ESP32 can host an entire production-ready music workstation.

For microcontroller-based synthesizers, this represents an advanced stage of development where the line between DIY instruments and professional tools begins to blur. It highlights the potential of ESP32 as a core platform for musicians.

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

The ±-synth project moves beyond microcontrollers into hybrid systems that combine digital and analog processing. It uses digital additive oscillators, called Big Fourier Oscillators, each capable of handling 1,024 partials. These are paired with analog low-pass filters to produce warm, natural sound while avoiding digital aliasing. The system achieves eight-voice polyphony and low latency, producing audio quality on par with professional synthesizers.

Although it does not use microcontrollers directly, this work is relevant because it shows the direction of high-fidelity digital synthesis. It also sets a benchmark that microcontroller- based systems like ESP32 can aspire to reach. Hybrid designs show that combining digital control with analog circuits can balance flexibility and sound quality.

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

The most recent research goes beyond traditional hardware design into artificial intelligence. The NAS-FM project proposes using Neural Architecture Search (NAS) to automatically design FM synthesis architectures. Instead of engineers manually defining operators and algorithms, the AI explores many possible designs and picks ones that balance accuracy, efficiency, and interpretability.

The synthesizers created through NAS are tunable and interpretable, meaning musicians can still understand and adjust parameters. At the same time, the models rival manually crafted FM synthesizers in sound quality. This research opens up possibilities for adaptive synthesizers that can learn from target sounds or even user preferences..

# Chapter 3
# Limitation of Existing system

# 3. Limitation of Existing system

Every research work begins by looking at what has already been done in the past. The existing systems for building polyphonic synthesizers mostly rely on either expensive hardware or very complex software frameworks. These solutions are good in terms of quality, but they are not always affordable or simple to implement for beginners, students, or small hobby projects.

One of the major limitations is cost. Traditional synthesizers are built using dedicated sound chips and advanced processors. This increases the price of the system, making it difficult for learners or small-scale users to try them. Low-cost alternatives exist, but they usually provide very limited sound features and cannot handle multiple notes (polyphony) effectively.

Another problem is complexity. Many existing systems use complicated coding platforms and require deep knowledge of sound engineering. This makes it hard for people who are new to electronics or programming to understand and build their own synthesizers. As a result, the learning curve becomes steep, and students often lose interest.

There is also a limitation in flexibility. Most existing synthesizers come with fixed features and cannot be easily modified. For research or experimental purposes, flexibility is very important, but the current systems do not always provide that..

All these points create a research gap. There is a clear need for a system that is affordable, simple to use, flexible for learning, and capable of producing polyphonic sounds. The ESP32 microcontroller provides an opportunity to fill this gap because it is low-cost, has good processing power, and is easy to program.

Thus, the limitation of the existing system is that it does not combine affordability, simplicity, flexibility, and portability together. This research tries to cover that gap by developing a polyphonic synthesizer using ESP32.

# Chapter 4

# Problem Statement and Objective

# 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

1. To design and develop a polyphonic synthesizer using the ESP32 microcontroller.
2. To generate different basic waveforms such as sine, square, triangle, and sawtooth.
3. To enable polyphony so that multiple notes can be played simultaneously, supporting chords and layered music.
4. To provide a simple user interface using buttons, switches, or piezo elements for note selection and control.
5. To integrate real-time audio output that can be connected to a speaker or headphones.
6. To explore the possibility of adding MIDI support for compatibility with external musical keyboards.
7. To keep the system low-cost, compact, and energy-efficient, so that it is suitable for students, hobbyists, and researchers.

# Chapter 5

# Proposed System

# 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.

**The algorithm for sound generation works as follows:**

Detect input from buttons or sensors.

Assign available "voices" using the voice allocation engine.

Generate waveforms (sine, saw, square) through oscillators.

Apply an envelope generator (attack, decay, sustain, release) to shape the sound.

Add filter and LFO effects for modulation.

Send the final signal through DAC/PWM to the audio output.

## 5.2 Design Details

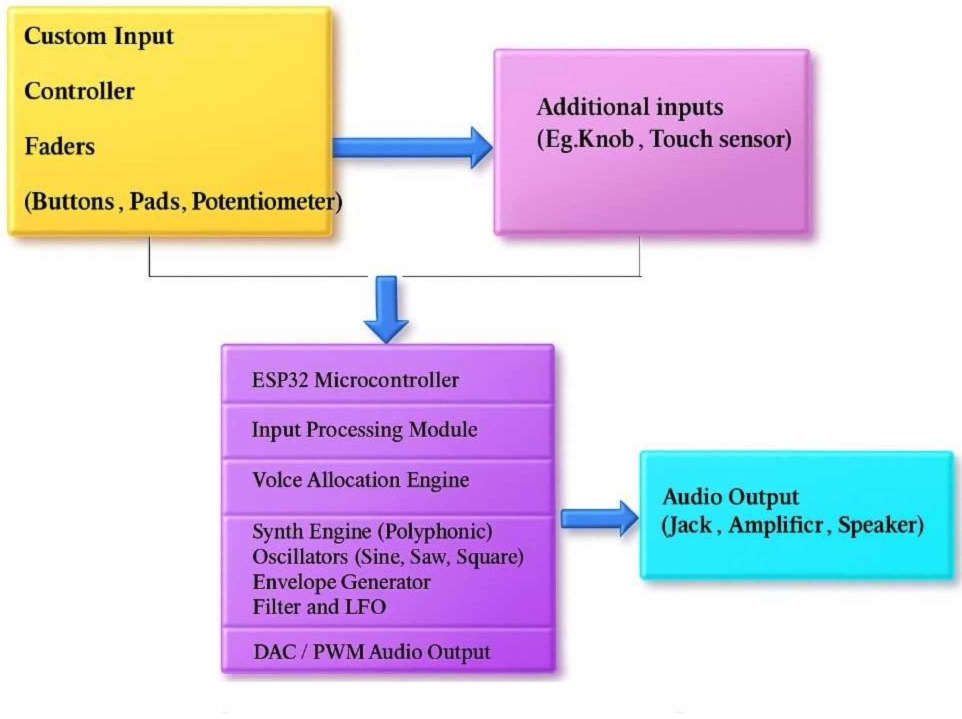

**Fig.5.2.1 System Design**

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).
- **Envelope Generator**: Shapes the sound (soft start, sustain, fade).

**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**
    - Study limitations of existing low-cost synthesizers.
    - Identify features required: polyphony, multiple waveforms, real-time audio.
2. **System Design**
    - Create block diagram and define modules (input, processing, output).
    - Select ESP32 as the main controller due to its performance and low cost.
3. **Hardware Setup**
    - Connect input devices (buttons, pads, potentiometers) to the ESP32.
    - Connect output to amplifier and speaker through DAC/PWM pins.

4. **Software Development**
    - Implement waveform generation algorithms (sine, saw, square).
    - Develop voice allocation logic for handling multiple notes.
    - Add envelope shaping and filtering.
    - Optimize code for real-time performance.
5. **Testing and Validation**
    - Test the synthesizer for single-note and multi-note sound generation.
    - Verify quality of audio output and response time.
    - Fine-tune envelope and filter parameters.
6. **Final Integration**
    - Combine hardware and software into a working prototype.
    - Evaluate the synthesizer for usability, cost, and performance.

# Chapter 6

# Experimental Setup

# 6. Experimental Setup

## 6.1 Hardware Setup

The hardware setup includes all physical components required for input, processing, and output.

- **ESP32 Microcontroller**
  - Acts as the main processing unit.
  - Handles input processing, voice allocation, sound synthesis, and output.
- **Input Devices**
  - **Buttons and Pads** – Trigger musical notes.
  - **Potentiometers/Knobs/Faders** – Control pitch, filter cutoff, volume, and modulation depth.
  - **Capacitive Touch Sensors** – Provide touch-based note triggering similar to a digital keyboard.
- **MIDI Interface**
  - Uses **UART or Bluetooth** to receive MIDI signals.
  - Allows connection with external devices such as MIDI keyboards or Digital Audio Workstations (DAWs).
- **Audio Output**
  - Utilizes the **ESP32's internal DAC** or an external I2S DAC.
  - Produces analog signals that are connected to **headphones, amplifiers, or speakers**.
- **Power Supply**
  - ESP32 powered via USB or external regulated supply for stable operation.

## 6.2 Software Setup

The software setup defines the programming environment, libraries, and optimization methods.

- **Development Environment**
    - Programming done in **Arduino IDE** using **C/C++**.
    - Libraries like **Mozzi** or custom DSP code used for waveform generation.
- **Input Processing**
    - Software modules read inputs from buttons, knobs, and touch sensors.
    - MIDI handling code interprets Note On/Off, Pitch Bend, and Control Change messages.
- **Voice Allocation Engine**
    - Distributes available voices across multiple notes to achieve polyphony.
    - Ensures smooth transitions when new notes are triggered.
- **Synth Engine**
    - Oscillators generate different waveforms (sine, sawtooth, square).
    - Envelope generators (ADSR) control amplitude shaping.
    - Filters (low-pass, high-pass) and LFOs add modulation effects.
- **Audio Output Handling**
    - Real-time scheduling ensures smooth audio playback without glitches.
    - DAC or PWM output module converts digital signals to analog audio.
- **Optimization**
    - Efficient memory use for multi-voice polyphony.
    - Task prioritization for input reading, synthesis, and output.
    - Latency kept below 20 ms for real-time performance.