Audio Playback and Recording System using MSPM0 and I^2C

Kairong Xu, Gerald Lu, Oscar Zhang

May 6, 2025

1 Introduction

In this project, we designed and implemented an audio player and recorder using the Texas Instruments MSPM0G3507 microcontroller. Our design was inspired by musical synthesizers and keyboards, which can blend multiple tones together in real-time through physical input. We aimed to replicate a similar experience at a basic level by integrating digital signal generation with hardware-based playback.

The system uses GPIO-controlled push buttons to generate different audio pitches, an I^2C-based DAC to convert digital signals into analog voltages, and an external amplifier to drive a speaker. An LED provides visual feedback to indicate when the system is in recording or playback mode. Additionally, we implemented a simple recording and playback function using the SRAM of the microcontroller, allowing users to capture and replay sequences of tones. This project not only strengthened our understanding of embedded systems and signal processing but also highlighted practical challenges in real-time audio systems.

2 Design Overview

2.1 System Concept

The audio system is structured around the MSPM0 microcontroller, which serves as the central controller for signal generation and user interaction. The MSPM0 communicates with an MCP4725 DAC using the I^2C protocol to transmit waveform data. This data is continuously updated based on user input from the push buttons.

The DAC output is fed into a PAM8302 amplifier, which boosts the signal strength to an audible level suitable for an 8Ω speaker. The tone generation is controlled by seven pitch-specific push buttons corresponding to musical notes C, D, E, F, G, A, and B. Two additional buttons are used for control functions: one to toggle recording and stopping, and another to control playback and pause.

Each button is tied to a specific GPIO pin configured with an internal pull-down resistor, enabling the system to detect rising edge transitions. By associating different waveform patterns with different buttons, we enabled the system to produce distinct tones upon user interaction. The modular design also supports a clear separation of hardware components, I^2C communication, and signal logic, improving system maintainability.

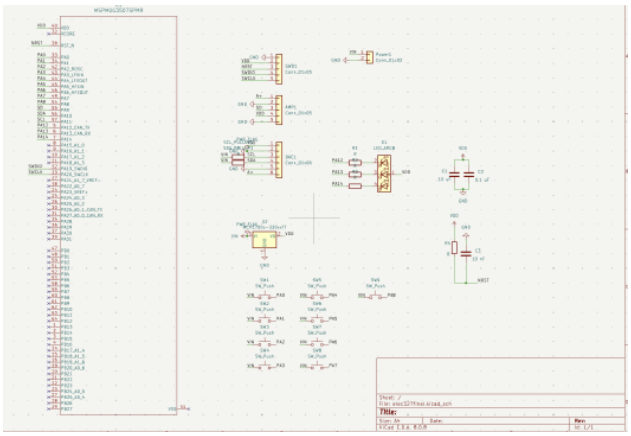

Figure 1: System-level schematic of audio playback device

3 System Implementation

3.1 Hardware Components

- **MSPM0 LaunchPad**: Serves as the main processing unit.
- **MCP4725 DAC (12-bit)**: Converts I^2C digital signals into analog voltages.
- **PAM8302 Amplifier**: Boosts the DAC output to audible levels.
- **Speaker (8Ω)**: Outputs the final sound.
- **Push Buttons (x9)**: GPIO-controlled switches for tone and control.
- **Voltage Regulator**: Provides stable power to all components.

We also used additional passive components. Capacitors help filter power supply noise. We implemented pull-up resistors on the SDA and SCL lines at the hardware level, since the internal resistors in the MSP were not at optimal values.

3.2 Software and Code Structure

The code was written in C. TI's SysConfig example was used to initialize GPIOs, I^2C communication, and timers. On each button press, the system:

1. Detects GPIO input (non-inverting logic).
2. Sends corresponding waveform data over I^2C to the DAC.
3. Enables LED blink through GPIO to confirm button press.
4. Amplifies the DAC signal and plays it.

First, the system initializes all peripherals with their respective I/O configurations. A boolean variable is used to track which button is pressed. The corresponding tone is played by modulating the delay between each DAC write, effectively controlling the sampling rate.
The recording functionality saves tone sequences in the SRAM of the MSPM0G3507 and allows for playback.

4 Team Contributions

The images used for Figure 1 and Figure 2 are available on the GitHub repository for more detailed reference.

- **Kairong Xu**: Concept design, PCB layout, SysConfig setup, low-level logic, button tone generation, LED testing, code architecture, and report writing.
- **Gerald Lu**: Code architecture, SysConfig setup, tone recording/playback logic, and slide/report drafting.
- **Oscar Zhang**: PCB soldering, report writing, Github Website setup and code architecture.

Special thanks to Professor Kemere for his guidance throughout the project!

5 Conclusion

By the end of the project, we successfully built a working audio playback and recording system that responded to button inputs and played corresponding tones through a speaker. While the system was functionally complete, one major issue we encountered was the low audio output volume. We believe this was due to the amplifier's input requirements not being well matched to the DAC output voltage levels. This mismatch limited the effectiveness of the sound playback despite correct logic implementation.

To verify system functionality, we probed the I^2C data line and observed signal activity when buttons were pressed, confirming data transmission between the microcontroller and DAC. In the process, we also realized the limitations of using SRAM as a storage medium for recorded tones, since it is volatile and requires continuous power. For future iterations, integrating non-volatile memory or adding power management (such as sleep mode) would make the recording feature more practical for longer-term use. This project provided valuable hands-on experience in combining digital logic, embedded C programming, hardware communication, and audio signal handling.